CUKUROVA UNIVERSITY

INSTITUTE OF NATURAL AND APPLIED SCIENCES

MSc THESIS

Malware Detection Using Machine Learning and Feature Selection Methods

Vahdet Cemil ALTUN

Computer Engineering Department

March, 2024

CUKUROVA UNIVERSITY

INSTITUTE OF NATURAL AND APPLIED SCIENCES

MSc THESIS APPROVAL

Malware Detection Using Machine Learning and Feature Selection Methods

Vahdet Cemil ALTUN

Computer Engineering Department

This Master's Thesis was evaluated by the following Jury Members on 5/3/2024 and was approved by unanimity / ~~majority~~ of votes.

Jury : Assoc. Prof. Dr. Fatih ABUT (Advisor) ……

: Assoc. Prof. Dr. Serkan KARTAL ……

: Assoc. Prof. Dr. Yasin KAYA ……

This Thesis was written in the Department of Computer Engineering Institute of Natural and Applied Sciences.

Thesis Number:

Prof. Dr. Sadık DİNÇER
Director
Institute of Natural and Applied Sciences

CONTENTS

CUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

MSc THESIS

Malware Detection Using Machine Learning and Feature Selection Methods

Vahdet Cemil ALTUN

Advisor: Assoc. Prof. Dr. Fatih ABUT

Department of Computer Engineering

ABSTRACT

This thesis aims to develop effective prediction models for malware detection using various machine learning methods and feature selection methods. Specifically, the study utilizes Light Gradient Boosting Machine (LightGBM), Decision Tree (DT), K-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP), Support Vector Machines (SVM), and Naive Bayes (NB) classifiers, each combined with Relief-F and Minimum Redundancy Maximum Relevance (mRMR) feature selectors. The analysis is conducted on 5136 samples, comprising 2683 malicious and 2501 benign files, derived from PE file header information of the Windows operating system. Evaluation of model performance is based on accuracy, F1-Score, recall, and precision criteria. Furthermore, to enhance the generalization ability of each model, 10-fold cross-validation is employed. The complexity matrices of the best performing model for each method are also provided. The results indicate that the LightGBM algorithm demonstrates promising outcomes in malware detection, both with default and optimized parameters. Moreover, the Relief-F feature selector exhibits superior performance compared to mRMR. These findings underscore the potential of LightGBM as an effective approach in malware detection and suggest that techniques such as Relief-F has significant potential for feature selection in this particular area.

Keywords: Malware Detection, PE Headers, Machine Learning, Feature Selection, Classification

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

Makine Öğrenimi ve Özellik Seçim Yöntemlerini Kullanarak Kötü Amaçlı Yazılım Tespiti

Vahdet Cemil ALTUN

Danışman: Doç. Dr. Fatih ABUT

Bilgisayar Mühendisliği Anabilim Dalı

ÖZ

Bu tez, çeşitli makine öğrenmesi yöntemlerini ve özellik seçme yöntemlerini kullanarak kötü amaçlı yazılım tespiti için etkili bir tahmin modelleri geliştirmeyi amaçlamaktadır. Çalışmada, Light Gradient Boosting Machine (LightGBM), Karar Ağacı (DT), K-En Yakın Komşular (KNN), Çok Katmanlı Perceptron (MLP), Destek Vektör Makineleri (SVM) ve Naive Bayes (NB) sınıflandırıcıları kullanılmış ve özellikle, Windows işletim sistemine ait PE dosya başlık bilgilerinden oluşturulan 5136 örnek (2683 Kötü niyetli, 2501 İyi huylu dosyalar) analiz edilmiştir. Relief-F ve Minimum Redundancy Maximum Relevance (mRMR) özellik seçicileri kullanılarak geliştirilen modellerin performansının değerlendirilmesinde doğruluk (accuracy), F1 puanı (F1-Score), geri çağırma (recall), ve kesinlik (precision) kriterleri kullanılmıştır. Ayrıca, her modelin genelleme yeteneğini artırmak için 10 kat çapraz doğrulama kullanılmış ve her yöntem için en iyi performansa ait modelin karmaşıklık matrisleri de sunulmuştur. Sonuçlar, LightGBM algoritmasının varsayılan ve optimize edilmiş parametrelerle kötü amaçlı yazılım tespitinde umut verici sonuçlar sunduğunu göstermiştir. Bunun yanı sıra, Relief-F özellik seçicisinin mRMR'a göre daha başarılı olduğu gözlemlenmiştir. Bu bulgular, LightGBM'nin kötü amaçlı yazılım tespiti alanında etkili bir yöntem olarak öne çıkabileceğini ve Relief-F gibi tekniklerin özellik seçiminde faydalı olabileceğini göstermektedir.

Anahtar Kelimeler: Kötü Amaçlı Yazılım Tespiti, PE Başlıkları, Makine Öğrenimi, Öznitelik Seçimi, Sınıflandırma

GENİŞLETİLMİŞ ÖZET

Teknolojinin hızlı gelişimi, günlük yaşamın ayrılmaz bir parçası olmuş durumda ve bu durum siber tehditlerin artmasına neden olmaktadır. Hızla evrilen teknoloji, dijital altyapıların genişlemesiyle birlikte yeni güvenlik zayıflıklarını ve açıklarını gün yüzüne çıkarmaktadır. İnternetin ve diğer iletişim ağlarının daha geniş kullanımı, siber saldırganlar için yeni fırsatlar yaratmaktadır. Her yeni teknolojik ilerleme, siber tehditlerin çeşitlenmesine ve daha karmaşık hale gelmesine yol açmaktadır.

Kötü niyetli yazılımlar, dijital çağın önemli bir tehdidi haline gelmiştir. Dijital cihazlar, ağlar ve internet üzerinden hızla yayılan kötü amaçlı yazılımlar, siber suçlular tarafından çeşitli amaçlar için kullanılmaktadır. Bu yazılımlar, bilgi hırsızlığı, kişisel veri ihlalleri, finansal kayıplar ve altyapı saldırıları gibi zararlara neden olabilir. Bu durum, bireylerin, şirketlerin ve ülkelerin ekonomik istikrarını ve ulusal güvenliğini etkileyebilir. Kötü niyetli yazılımların hızlı yayılma yeteneği ve giderek karmaşık hale gelmesi, bilişim güvenliği alanında sürekli olarak yeni çözümler ve savunma stratejileri gerektirir. Bu nedenle, siber güvenlik önlemlerinin sürekli güncellenmesi ve güçlendirilmesi, malware kaynaklı tehditlere karşı etkili bir savunma sağlar.

Yapay zekâ, nesnelerin interneti ve bulut bilişim gibi çağdaş teknolojilerin benimsenmesi ve kullanılması, daha karmaşık saldırı vektörlerinin oluşmasına neden olurken aynı zamanda savunma stratejileri açısından da önemli avantajlar sağlamaktadır. Yapay zekâ destekli saldırılar, cihazlar arasında görülen güvenlik açıklarını tespit etme ve kötü niyetli yazılımların oluşturulmasında etkin rol oynamaktadır, bu da siber güvenlik uzmanlarının görevlerini karmaşıklaştırmaktadır. Bununla birlikte, kötü niyetli yazılım tespiti gibi siber güvenlik alanlarında yapay zekâ oldukça kullanışlıdır. Özellikle makine öğrenimi ve derin öğrenme algoritmaları, bilgisayar sistemlerindeki anormal davranışları algılamak ve potansiyel tehditleri tanımlamak için etkili bir biçimde kullanılmaktadır. Bu, geleneksel imza tabanlı yöntemlerin ötesine geçerek, bilinmeyen veya yeni geliştirilen kötü niyetli yazılımları tespit etme yeteneği sunar. Yapay zekâ, büyük veri kümelerini analiz edebilir ve belirli örüntüleri veya anormallikleri belirlemekte insanlardan çok daha hızlı ve etkili olabilir. Ayrıca, sürekli öğrenme yetenekleri sayesinde, zamanla gelişen tehditlere karşı daha adaptif ve güncel kalabilir. Bu da siber güvenlik uzmanlarının tehditlere karşı daha etkin ve dinamik bir savunma stratejisi geliştirmelerine yardımcı olur.

Bu tezin amacı modern makine öğrenimi algoritmaları ve öznitelik seçim araçlarını kullanarak etkili ve verimli bir kötü niyetli yazılım tespiti geliştirmeyi amaçlamaktadır. Windows işletim sisteminde kullanılan Taşınabilir Yürütülebilir (PE) dosya başlıkları üzerinde Minimum Artıklık Maksimum Uygunluk (mRMR) ve Relief-F gibi özellik seçim yöntemlerini kullanarak çeşitli modeller oluşturulmuştur. Tahmin modellerinin değerlendirilmesi için Hafif Gradyan Artırma (LightGBM), Karar Ağaçları (DT), K-En Yakın Komşu (KNN), Çok Katmanlı Algılayıcılar (MLP),

Destek Vektör Makineleri (SVM) ve Naive Bayes (NB) gibi makine öğrenimi algoritmaları hem varsayılan hem de hiper parametre ayarlamaları sonucunda elde edilen parametrelerle incelenmiştir.

Bu çalışmada, 31 Ocak 2019 tarihinde yayımlanan PE Başlıklarıyla Kötü Amaçlı Yazılımların Sınıflandırılması (CLAMP) veri kümesi kullanılmıştır. Kullanılan veri kümesi, Windows işletim sistemlerinde yaygın olarak kullanılan PE dosyalarının başlık bilgilerini içermektedir ve 56 özellik ve 5136 örnekten oluşmaktadır. Veri kümesinin oluşum süreci incelendiğinde, Python programlama dilinde bulunan Pefile modülü kullanılmış ve Windows PE başlık bilgilerinin çıkarılmasıyla oluşturulmuştur. Bununla birlikte, VirusShare çevrimiçi platformunda bulunan çeşitli veri türlerine sahip PE dosyaları da doğrudan analizde kullanılmıştır. Bu veri kaynaklarının birleşimi, analiz sürecine geniş bir veri yelpazesi sağlamıştır, bu da çalışmanın kapsamlı ve detaylı bir yapıya sahip olmasını sağlamıştır.

Veri ön işleme aşamasında, veri setindeki düzensizliklere sebep olan, gürültü oluşturan veya makine öğrenimi algoritmalarının performansını olumsuz etkileyebilecek sütunlar temizlendi. Bu süreç sonucunda, nihai veri seti 52 değişkenden oluşmaktadır. Verisetini makine öğrenimine hazır hale getirmek için standartlaştırma işlemi uygulandı. Bu adım, veri setinin analiz edilmesi ve modelleme sürecinde daha tutarlı ve güvenilir sonuçlar elde edilmesine olanak tanımıştır. Veri setinin gereksiz veya dengesiz olan özelliklerden arındırılması, modelin daha doğru ve etkili bir şekilde öğrenmesine ve sonuçlarının daha güvenilir olmasına yardımcı olur. Bu ön işleme adımları, analiz sürecinde veri setinin niteliklerini daha verimli bir şekilde kullanılabilir hale getirmektedir.

mRMR ve Relief-F gibi özellik seçicileri kullanılarak, her değişkenin önemi hesaplandı ve bu önemliliklerine göre azalan düzende sıralandı. Ardından, en düşük öneme sahip olan her değişken sırayla çıkarıldı. Bu süreç sonucunda, mRMR tabanlı 51 ve Relief-F tabanlı 51 olmak üzere toplamda 102 farklı tahmin modelleri oluşturuldu. Bu modeller, değişkenlerin farklı önem derecelerine göre nasıl etkileşime girdiğini ve hangi değişkenlerin model performansına daha fazla katkı sağladığını anlamak için ayrı ayrı incelenme imkânı sağladı. Bu sayede, her bir özellik seçici yönteminin modelin performansına olan etkisi daha detaylı bir şekilde değerlendirilebildi.

Her bir model, varsayılan ve optimize edilmiş değerler olmak üzere iki ayrı yaklaşımla değerlendirildi. Varsayılan değerlerler, her makine öğrenimi algoritması için sağlanan varsayılan parametre ayarları kullanılarak eğitim gerçekleştirildi ve performans değerlendirmesi yapıldı. Optimizasyon sürecinde ise her model, Optuna hiper parametre ayarlama çerçevesi modülüyle her bir yöntem için bağımsız bir şekilde eğitim yapıldı. Bu adımlar sonucunda, her bir model için en iyi makine öğrenimi parametreleri belirlendi ve bu parametrelerle modellerin performansları ölçüldü. Bu analiz, her algoritmanın farklı parametre ayarlarının performansa etkisini belirleme amacını taşımaktadır. Bu şekilde, her bir modelin en iyi performansı elde etmek için hangi parametrelerle yapılandırılması gerektiği üzerine daha ayrıntılı bakış açısı sağladı.

Genelleme yeteneğini daha güvenilir bir şekilde ölçebilmek için her bir modele 10 katlı çapraz doğrulama uygulandı. Bu süreç, her bir modelin performansını belirlerken aşırı uydurma

(overfitting) veya veriye özgü genelleme sorunlarını minimize etmeye yardımcı oldu. Doğruluk (accuracy), F1 puanı (F1-Score), geri çağırma (recall) ve kesinlik (precision) gibi metrikler, modelin ne kadar doğru sınıflandırma yaptığını ölçerken karmaşıklık matrisi, farklı sınıflar arasındaki karışıklığı ve hataları gözlemlemek için kullanıldı. Metriklerin kapsamlı bir şekilde değerlendirilmesi, her bir modelin performansının ayrıntılı bir analizini sağladı ve hangi sınıfların hassas veya daha az önemsiz tahmin edildiğini belirlemeye yardımcı oldu.

Sonuçlar, makine öğrenimi yöntemlerinin varsayılan veya optimize edilmiş parametrelerle eğitilen modeller içinde Relief-F tabanlı modellerin, mRMR tabanlı modellere kıyasla daha başarılı sonuçlar elde ettiğini göstermiştir. Bu durum, özellik seçimi için Relief-F'in, model performansını artırma konusunda daha etkili olduğunu göstermektedir.

Makine öğrenimi yöntemlerinin varsayılan ve optimize edilmiş parametrelerle eğitilmiş modellerinin genel ortalama çıktıları değerlendirildiğinde, Relief-F ve mRMR öznitelik seçicilerinin accuracy, F1-Score, recall ve precision metrikleri açısından oldukça benzer sonuçlar elde etmiştir. Ancak, makine öğrenimi algoritmalarının optimize edilmiş parametrelerle eğitilmesi, varsayılan parametrelere göre tüm metriklerde önemli ölçüde yüksek başarı sağlamıştır.

Elde edilen sonuçlar incelendiğinde kötü niyetli yazılım tehditlerine karşı mücadele etmek için makine öğrenimi tekniklerinin ve parametre optimizasyonunun ne kadar kritik olduğunu açıkça göstermektedir. Araştırma bulguları, optimize edilmiş parametrelerin siber güvenlik açısından büyük önem taşıyabileceğini ve bu alandaki başarıyı artırabileceğini ortaya koymaktadır. Ayrıca, siber güvenlik araştırmalarında daha öngörülü ve derinlemesine bir yaklaşımın ne kadar önemli olduğunu vurguluyor. Parametre optimizasyonunun etkisi, makine öğrenimi alanındaki başarı için hayati bir unsurdur. Modern tekniklerin, kötü niyetli yazılım tespiti gibi karmaşık alanlarda umut verici sonuçlar sunmaktadır.

ACKNOWLEDGEMENTS

I extend my sincere gratitude to my supervisor, Assoc. Prof. Dr. Fatih ABUT, whose unwavering guidance, patience, invaluable advice, constant encouragement, and insightful perspectives have been instrumental throughout this study.

I am deeply appreciative of Assoc. Prof. Dr. Serkan KARTAL and Assoc. Prof. Dr. Yasin KAYA for their invaluable suggestions, constructive comments, and for their dedicated service on my committee, which greatly enriched the quality of this research.

Furthermore, I would like to express my gratitude to Kumar, A. ve Kuppusamy, K.S., and Aghila, G. for generously sharing the CLaMP dataset used in this thesis with the wider research community.

Lastly, I owe a debt of gratitude to my family, whose unwavering presence and boundless support have been my pillars of strength throughout this journey. Their unwavering trust and belief in me have enabled me to devote myself fully to this endeavor. I am profoundly grateful for their unwavering support and encouragement.

LIST OF TABLES

LIST OF FIGURES

SYMBOLS AND ABBREVIATIONS

ADR	:	Attack Detection Rate
ANOVA	:	Analysis of Variance
API	:	Application Programming Interface
API-CTM	:	API-Call Transition Matrix
APT 1	:	Advanced Persistent Thread 1
ARM	:	Advanced RISC Machine
AUC	:	Area Under Curve
AdaBoost	:	Adaptive Boosting
BC	:	Bagging Classifier
CFS	:	Correlation-Based Feature Selection
CLAMP	:	Classification of Malware with PE Files
COFF	:	Common File Object
CPU	:	Central Process Unit
CUI	:	Conversation User Interface
DNS	:	Domain Name System
DNN	:	Deep Neural Network
DLL	:	Dynamic Link Library
DoS	:	Denial-of-Services
DT	:	Decision Tree
ELM	:	Extreme Learning Machine
EMBER	:	Endgame Malware Benchmark for Research
EM	:	Expectation Minimization
ETC	:	Extra Trees Classifier
EXE	:	Executable
FAR	:	False Alarm Rate
FLM	:	File Large Margin
FNR	:	False Negative Rate
FP	:	False Positive
FPR	:	False Positive Rate
GB	:	Gradient Boosting
GBT	:	Gradient Boosting Tree
GBDT	:	Gradient Boosting Decision Tree
GOSS	:	Gradient One-Sided Sampling
GUI	:	Graphical User Interface
HMM	:	Hidden Markov Model
HTML	:	Hypertext Markup Language
IBK	:	Instance-Based Learning with Parameter k
IDS	:	Instruction Detection System
JSON	:	JavaScript Object Notation
JPEG	:	Joint Photographic Experts Group
KNN	:	K-Nearest Neighbors
LMT	:	Logistic Model Tree
LR	:	Logistic Regression

LS	:	Simple Logistic
LSVM	:	Linear Support Vector Machines
LRC	:	Linear Ridge Classifier
LSTM	:	Long Short-Term Memory
MCC	:	Matthew Correlation Coefficient
ML	:	Machine Learning
MLP	:	Multi-Layer Perceptron
MSE	:	Mean Squared Error
MV	:	Majority Voting
MUI	:	Multilingual User Interface
NA	:	Naïve Bayesian
NB	:	Naïve Bayesian
NN	:	Neural Network
NT	:	New Technology
PCC	:	Pearson Correlation Coefficient
PCA	:	Principle Component Analysis
PE	:	Portable Executable
PRC	:	Precision-Recall Curves
RAM	:	Random-Access Memory
ReLU	:	Rectified Linear Unit
RBFN	:	Radial Basis Function Network
ROC	:	Receiver Operating Characteristic
RT	:	Random Tree
RVA	:	Relative Virtual Address
SHA	:	Secure Hash Algorithm
SG	:	Stack Generazation
SGD	:	Stochastic Gradient Descent
SNN	:	Shared Nearest Neighbors
SMOTE	:	Synthetic Minority Oversampling Method
SVM	:	Support Vector Machines
SYS	:	System
TDIDT	:	Top-Down Induction on Decision Trees
TF-IDF	:	Term Frequency-Inverse Document Frequency
TN	:	True Negative
TP	:	True Positive
TPR	:	True Positive Rate

1. INTRODUCTION

The combination of cybersecurity and machine learning (ML) has revolutionized the approach to combating evolving digital threats. This convergence involves developing and automating systems that can identify and neutralize malware through anomaly detection. Leveraging ML algorithms allows the analysis of large amounts of data, enabling the detection of previously unseen or complex threats. It increases the accuracy and efficiency of malware identification while reducing false positives, thus strengthening defense mechanisms in complex computing environments. The interaction between cybersecurity and ML not only supports existing protective measures but also paves the way for adaptive systems that can dynamically respond to emerging threats in real-time (Martínez Torres et al., 2019; J. Singh and Singh, 2021; Tahir, 2018). The development of such systems makes the job of malware developers difficult (Mohammed et al., 2020). For instance, ML methods allow for the neutralization of malicious activity hidden in Portable Executable (PE) files, and when combined with various analysis methods, even more powerful detection systems can be developed.

The PE file is a standard file format for executable files used in 32-bit or 64-bit versions of the Windows operating systems. It evolved from the Common Object File Format (COFF) norms in UNIX. The information required by the Windows operating system installer to handle compressed executable code is contained in the PE file format. In addition to system drivers and executables, this standard encompasses Dynamic Link Libraries (DLLs) and other components. Resource management, DLLs, Application Programming Interfaces (APIs), export and import tables, and other components are all included in PE files. File formats including Executable (EXE), DLL, System (SYS), and Multilingual User Interface (MUI) are stored in PE format in Windows NT-based operating systems. The Windows operating system can effectively handle many apps and components in this manner (Anderson and Roth, 2018; Pietrek, 2002).

Since the release of the Windows New Technology (Windows NT) 3.1 operating system, the PE file structure has changed to accommodate a variety of systems with distinct instruction set architectures, including x86, x64, Itanium Architecture 64 (IA-64), Advanced RISC Machine (ARM), and ARM64. This flexibility allows the Windows operating system to run smoothly on different devices and architectures. A single standard for drivers, DLLs, and executable applications is provided by the PE file format, which enables a wide range of users to benefit from a safe and compatible Windows environment (Pietrek, 2002).

The PE file headers contain important headers that make the PE file executable by the operating system. The headers in the executing PE file determine the method by which information is processed by the operating system and the behavior of the executed code. These include (a) the architecture of file, (b) operating system compatibility, (c) data partitions, (d) entry points, (e) dynamic links, (f) memory management, and (g) operating system dependencies. The PE file headers

ensure that the executable is installed and runs correctly, providing a compatible experience across various platforms (Microsoft, 2024).

The varieties of headers are included in the PE file to help organize the Windows operating systems the functionality of installer and ensure that the executable code loads and functions properly. The structure of the PE file is visually depicted in Figure 1.1. Each of these headers serves a specific purpose and defines the structure of the PE file:

- DOS Header: Contains information for Microsoft Disk Operating System (MS-DOS) from the period in which it was designed and is used for compatibility purposes only.
- File Header: Contains basic information. Defines file type, architecture, number of partitions, and other general properties.
- Optional Header: Determines operating system compatibility, file size, memory settings, entry points, and other details. It contains important information about how the PE file will be processed by the operating system. The optional header has become a major header for Windows NT 3.1 and later.
- Section Header: Defines the properties of each section in the PE file. Specifies the size, name, access rights, and other properties of partitions.
- Sections: Contains sections that physically represent the content of the PE file. Each section contains unique data and is organized according to properties defined by Section Headers.

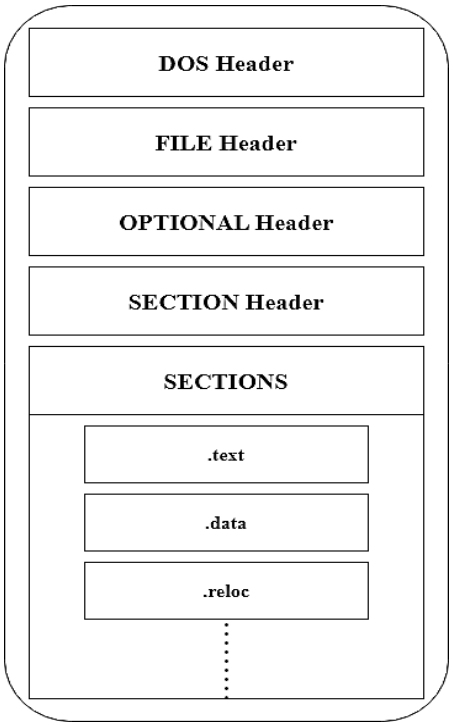


Figure 1.1. PE File Structure (Rkhouya and Chougdali, 2021)

Each header within the PE file serves a distinct purpose, defining the file's varied attributes and structure to ensure its proper loading and execution. With the exception of the DOS Header, the other headers notably influence how the file is processed, particularly in Windows NT 3.1 and later versions. The cohesive presence of these headers amalgamates to form the PE file, guaranteeing seamless functionality and compatibility within the Windows ecosystem. This intricate framework not only facilitates the proper handling and execution of the file but also explicitly outlines its diverse characteristics and structure.

1.1. Malware Variant and Analysis Methods

Malware continues to evolve in diversity, encompassing a variety of types designed to disrupt systems or compromise privacy. These malware variants pose different dangers and exhibit unique capabilities, thus underscoring the critical need to understand their specific characteristics and potential impacts in the field of cybersecurity. Information about the types of malwares and their potential effects is shown in Table 1.1.

Table 1.1. Types of Malwares (Tahir, 2018)

Type	Description
Viruses	It multiplies and spreads by being added to files.
Worms	It is malware that spreads independently and copies itself into other files.
Trojans	It is software that deceives the user by hiding harmful content.
Spyware	It is software that monitors computer user information and aims to steal information.
Adware	It is designed to show intrusive ads on the device.
Ransomware	It is malware that encrypts files and demands a ransom for decryption.
Botnets	It is software that connects and manages many infected computers to a control point on the network.
Keyloggers	It is software that records the victim's keystrokes and therefore leaks sensitive information.

The crucial process of malware analysis is to comprehend the characteristics of malware, assess its impact, and create defenses against it. It additionally serves as the foundation for developing an effective defensive strategy and analysis techniques to address cyberthreats. The fundamental analysis techniques comprise (a) static-based analysis, (b) dynamic-based analysis, (c) hybrid-based analysis, and (d) network-based analysis. These techniques are essential for determining whether a file contains malware or benign software.

Static-based analysis is a method where the code and structure of malware are thoroughly examined (Tahir, 2018). The features, potential, behavior, and ability of the malicious file are examined and the risks or possible hazards included in the code are discovered without running the malicious code of the file on any platform (J. Singh and Singh, 2021). This analysis approach has

benefits and drawbacks, despite containing the most fundamental analytical techniques. Since the file under examination is not run on any machine, the analysis process is safe. Analyzing the code of the file and the insight gained from the process build a framework that anticipates potential dangers. Nevertheless, the static-based analysis approach is less successful than other approaches in identifying distinct harmful files. The use of obfuscation techniques, which conceal harmful activity in the code, makes detection more challenging. However, the static analysis of big and complicated software files requires a lot of time and resources. Since it does away with the requirement for quick action, this would not be useful in the realm of cyber security.

Dynamic-based analysis examines in detail the behavior and effects of the malicious file on the system. A malicious file is executed in a controlled and isolated environment, mostly on virtual machines, and many effects such as potential of the file behavior, memory utilization, system calls, binding, web socket utilization are examined. During the dynamic analysis, the malicious behaviors of file and repercussions are observed in real time (J. Singh & Singh, 2021). Nevertheless, there is a risk of file propagation, resource consumption, and time management issues because the malicious file is run in real time throughout the dynamic-based analysis procedure.

Hybrid-based analysis examines the overall impact of malware on its characteristics, behavior, and potential effects by combining static and dynamic analysis techniques. This approach allows for the identification of malware's potential dangers and effects. By combining the benefits of static and dynamic analysis, an unconventional perspective is offered, providing comprehensive information for malware analysis (Hadiprakoso et al., 2020). The obfuscation technique, which is frequently used in malicious files, is overcome by the hybrid analysis method. However, compared to other approaches, keeping these tools integrated and up-to-date is more difficult and requires a significant investment of resources in this strategy that combines static and dynamic analysis.

Network-based analysis is an approach based on monitoring the communication patterns and network interactions, focusing on the communication of malware patterns, control infrastructure, as well as interactions with network resources by meticulously monitoring the network behavior of the malware. This method provides important insights into the detection and analysis of malicious activities by observing interactions of the malware over the network. Network-based analysis stands out as an effective tool for developing strategies to deal with cyber threats by identifying anomalies and distinct patterns in the network traffic of the malware.

1.2. Malware Analysis Prediction Model

Malware detection is often a binary classification problem. Determining whether a given file is malware is a process of assigning data to appropriate classes with two-labels (i.e., binary) or multi-class labels to assign the data. Therefore, in classification models, the output value is categorical.

Predicting the class to which a given input belongs is the foundation of classification models. The classification expression for a function determined by certain input variables x_1, x_2, x_n is

$g(x_1, x_2, x_n) = c$. Such classification predictions are called regular predictions. A classification model is used specifically in cases where the output classes are categorical, which generally attempts to estimate the probability distributions of particular classes, selecting the class with the highest probability. Considering the classification approach, a prediction output was obtained that applied the header information of the PE file to various classification methods for malware analysis.

1.3. Purpose and Contributions of the Thesis

The main goal of this thesis is to perform malware analysis on the PE header information, perform classification based on ML and feature selection methods, and achieve maximum efficiency with hyperparameter tuning. In this study, various ML and feature selection methods were used to create various new classification models. Light Gradient Boosting Machine (LightGBM), Decision Tree (DT), K-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP), Support Vector Machines (SVM), and Naive Bayesian (NB) models were used to create prediction models. Relief-F and mRMR algorithms were preferred for feature selection. The performance metrics such as accuracy, F1-Score, recall, precision, and confusion matrix are used along with 10-fold cross-validation to evaluate the performance of the proposed models.

The following concisely describes the contributions and significant distinctions this thesis makes from the other research in the relevant literature:

- This study introduces a novel perspective to existing literature by combining feature selection algorithms and machine learning techniques to identify relevant and redundant optimal variables for predicting malware analysis over PE files. Its objective is to develop models containing PE headers using filter-based feature selectors such as Relief-F and mRMR. Additionally, it aims to underscore the importance of optimization by comparing default and hyperparameter-adjusted parameters of machine learning algorithms.
- This represents the inaugural utilization of the Optuna, an advanced open-source platform for hyperparameter optimization.
- This study, using two feature selectors and six ML classifiers on the dataset used, is the most comprehensive study on the number of models developed for malware analysis of the PE file.
- This is the first study developed using the LightGBM, a new generation algorithm with low memory usage, fast training time and high performance, on the Classification of Malware (CLAMP) dataset. It stands out in the field of malware classification, especially thanks to its fast response time and high performance. It also attracts attention with its low False Positive Rate (FPR).

- This thesis allows ranking the evaluated methods applied to predict whether PE files are malware or not.

1.4. Roadmap of Thesis

Chapter 1 introduces the foundational aspects of malware analysis, prediction models, and outlines the objectives and contributions of this thesis. It begins with an exploration of malware variants and analysis methods, followed by an overview of malware analysis prediction models. Subsequently, it delineates the purpose and significance of the thesis, along with its contributions to the field. Finally, it provides a roadmap outlining the structure and organization of subsequent chapters.

Chapter 2 provides a comprehensive review of current research on malware activities, offering up-to-date information and a detailed examination. To facilitate a thorough understanding, these studies are systematically categorized into static, dynamic, hybrid, and network-based methods. Furthermore, this chapter delves into machine learning methods, evaluation criteria, dataset samples, and validation methods to provide a comprehensive overview of the subject matter.

Chapter 3 serves as a foundational segment, offering comprehensive theoretical insights into the operational principles of various ML algorithms and feature selection methodologies employed within this thesis. It elaborates on the theoretical underpinnings and operational mechanisms of the ML algorithms and delineates the rationale behind the selection and implementation of specific feature selectors.

Chapter 4 contains information on dataset preprocessing, tuning, and creation. Additionally, the construction strategies and evaluation criteria of default and optimized prediction models are discussed in this section. The default and hyperparameter ranges, which include the general methodology of the classifiers, are discussed in detail in this section.

Chapter 5 meticulously scrutinizes diverse evaluation criteria for default and optimized parameters within models generated by feature selectors, offering a nuanced comprehension of these performances of models across various configurations. By critically analyzing these findings, this section also elucidates their implications within the broader context of the research objectives. Additionally, details of earlier similar studies are also shown in this section.

Chapter 6 serves as a comprehensive conclusion to the present study, offering a synthesis of the primary findings and conclusions derived from extensive analysis and discussions. This section succinctly summarizes the results obtained and underscores the principal conclusions drawn from the analysis and discussions conducted. Moreover, it provides insights into potential directions for future research endeavors and suggests avenues for further exploration. By addressing identified gaps and limitations within the study, this chapter contributes to the continuous advancement of the field.

2. LITERATURE REVIEW

Various studies and approaches regarding malware analysis have been documented in the literature. This chapter provides a detailed discussion of these approaches, offering an overview of the landscape. Divided into four distinct sections, each section focuses on a specific aspect of malware analysis methodologies, outlined as follows: (a) Examination of studies centered on static-based malware analysis, (b) Examination into studies centered on dynamic-based malware analysis, (c) Examination of studies focused on hybrid-based malware analysis, and (d) Examination of studies concentrating on network-based malware analysis.

2.1. Studies Based on Static Malware Analysis

Balodi et al. (2023) proposed a straightforward yet effective malware detection system employing static malware analysis methods. The dataset comprises 5.000 benign and over 14.000 malware samples, with more than 70 features. Upon examination of the findings, it was observed that the Random Forests (RF) method exhibited an average performance of 99% in accuracy and precision criteria, achieving an average performance of 98% in recall and F1-Score criteria. The study showcased high performance in static malware analysis and the examination of PE files without employing dataset balancing, data preprocessing, or standardization with raw data.

Yousuf et al. (2023) introduced a novel perspective on static analysis within the context of malware analysis. Six distinct datasets were curated, encompassing DLL imports, API call functions, PE headers, PE sections, Integrated Features Set 1 (i.e., IFS 1, DLL imports/API calls), and IFS 2 (i.e., PE headers/PE sections). The dataset comprised a total of 27.920 malware samples and 1.878 benign samples. In addition to the regular feature selection method, Information Gain (IG), Principal Component Analysis (PCA), and SVM, Gradient Boosting (GB), KNN, Nearest Centroid (NC), DT, RF, and NB algorithms were employed. Ensemble methods such as Majority Voting (MV), Stack Generation (SG), and Adaptive Boosting (AdaBoost) were applied using the 10-fold cross-validation methodology. In the regular feature selection method, DT and AdaBoost exhibited high accuracy on the IFS 1 dataset, while RF and SG demonstrated high accuracy on the remaining datasets. When employing IG feature selection, DT and AdaBoost achieved notable accuracy on the DLL import dataset, SVM and MV for the API function dataset, and RF and SG for other datasets. With PCA feature selection, SVM and AdaBoost demonstrated high accuracy in the DLL import and IFS 1 datasets, DT and Majority Voting (MV) for the API function dataset, and RF and SG for other datasets.

Srastika et al. (2022) introduced an approach to malicious software detection using ML and feature selection methods. The dataset comprises 33.665 samples and was independently evaluated using Analysis of Variance (ANOVA) methodology, resulting in the selection of twenty active features. RF, KNN, and eXtreme Gradient Boosting (XGBoost) classification algorithms were

employed. The findings revealed that the RF achieved the highest accuracy at 99.2% when utilizing feature selection and demonstrated an overall accuracy of 99.1% across the entire dataset. When combined with ANOVA feature selection, the RF exhibited an accuracy of 96.9%. Other methods showed accuracies of 94.7% for KNN and 97.7% for XGBoost, respectively.

Moon et al. (2022) proposed a feature hashing approach to identify variants of ML-based malware. This approach utilized the feature hashing technique derived from the Endgame Malware Benchmark for Research (EMBER) dataset, which was then adapted to the dataset within the proposed system. The primary motivation behind this study was to conserve disk space. The dataset comprised 282.421 samples, utilizing headers extracted from the PE files. The ML technique employed the RF method, and the performance of model was evaluated based on accuracy, precision, recall, and F1-Score criteria. The findings revealed that the proposed approach achieved an accuracy of 96.19%.

Hussain et al. (2022) proposed a static malware analysis system designed to classify the PE files as benign or malignant, considering both 64-bit and 32-bit versions of these files. The dataset consisted of the 54 the PE Header features and 138.042 samples, encompassing both benign and malicious files. In the proposed system, the Extra Trees Classifier (ETC) algorithm was employed for feature engineering, selecting 15 out of the 54 features. Various classification algorithms, including RF, DT, SVM, Adaptive Boosting (AdaBoost), GB, and NB, were utilized in the system. The evaluation of model was based on F1-Score, accuracy, precision, recall, and support. The findings indicated that RF was the most effective algorithm, achieving an accuracy rate of 99.44%. The accuracy values for the other methods were DT 99.17%, GB 98.97%, AdaBoost 98.56%, SVM 97.68%, and NB 92.95%, respectively.

Rkhouya and Chougdali (2021) introduced a malware detection system utilizing ML techniques. The system employed a total of 135.387 (i.e., 124.987 malware and 10.400 benign) samples. The dataset consisted of unique the PE file headers from the Windows operating system, encompassing 54 features in total. Among these, the 9 features were selected through ETC feature selection. The selected features used for analysis using DT, GB, SVM, and RF ML techniques. Subsequently, the model was evaluated based on processing time, accuracy, FPR, False Negative Rate (FNR), and Area Under Curve (AUC) criteria. Experimental results demonstrated that the RF outperformed the other three algorithms, achieving an accuracy of 99.65% and AUC values of 99.33%.

Kumar et al. (2020) proposed a new system was aimed at detecting and preventing the PE files from being categorized as either malicious or benign. The dataset, known as Brazilian malware, consisted of 121.000 rows and 57 attributes. To prepare the dataset for classification, data preprocessing approaches like label encoding and one-hot encoding were applied, utilizing tools available in the Python Scikit-Learn module. Additionally, the dataset underwent feature selection using the ETC, which identified the 15 the most relevant features based on entropy. These selected

features were then utilized in various classification algorithms including DT, RF, GB, SVM, Logistic Regression (LR), XGBoost, and AdaBoost. The evaluation criteria included accuracy, precision, recall, training time, and the analysis of confusion matrices. The findings indicated that the RF emerged as the most effective algorithm, achieving a highest accuracy rate of 99.7%.

Mohammed et al. (2020) presented an approach where a website determines whether an uploaded the PE file to the system is malware or not. This approach encompasses supervised learning for the classification problem. The dataset comprises file parameters from both benign and malicious executable files, totaling 54 attributes and 138.048 records. The feature selection process for the proposed model favored a tree-based approach, resulting in the selection of the 12 features out of the initial 54. The evaluation criteria for the model encompassed accuracy and the confusion matrix. The findings revealed an accuracy rate of 99.43% for RF and 98.97% for DT.

Balram et al. (2019) proposed a malware detection system leveraging header information from the PE and sequences. They utilized the Advanced Persistent Threat 1 (APT1) dataset, released by China's cyber unit, comprising 427 malicious and 989 benign files. This dataset al.so includes detailed timelines for over 40 APT1 malware families. Their approach incorporated various methods, including classification algorithms, hybrid models, and 5-fold cross-validation. Supervised learning involved models such as SVM, LR, RF, XGBoost, along with hybrid models, namely hybrid model 1 (i.e., LR and XGB) and hybrid model 2 (i.e., LR, RF, and NB). The evaluation criteria encompassed accuracy, precision, recall, F1-Score, and model execution time. The study's results revealed that hybrid model 1 (i.e., LR and XGB) achieved an accuracy of 98% in sequence analysis and 91% in PE Header analysis.

Radwan (2019) proposed a system that focuses on detecting malicious files in the PE by emphasizing the use of headers within these files. The dataset consisted of 5.184 samples, including 2.683 malware and 2.501 benign files. The first class included only the PE file headers, while the other class covered the 46 additional features related to portable executables. Various classification algorithms were employed in the proposed model, including KNN, Gradient Boosting Tree (GBT), LR, DT, RF, File Large Margin (FLM), and NB. A 10-fold cross-validation method was utilized to evaluate performance, incorporating metrics such as accuracy, precision, recall, and F-measure. The results revealed notable performance differences between different verification methods. When the 70-30 split method was used, in the first class, the RF method emerged as the most successful algorithm with 97.56% accuracy, followed by GBT, DT, KNN, LR, FLM, and NB, respectively. For the other class, the GBT method was the most successful algorithm with 98.38% accuracy, while the others were DT, KNN, LR, FLM, and NB, respectively. However, when 10-fold cross-validation was used, the GBT method remained the most successful algorithm with 93.55% accuracy, followed by KNN, LR, RF, DT, FLM, and NB, respectively.

Zhang et al. (2019) presented a classification approach aiming to categorize the PE file malware types through machine learning techniques. The study utilized a dataset consisting of

350.000 benign and 400.000 malware samples, incorporating files from the EMBER dataset and validated by VirusTotal website. The findings revealed that the RF demonstrated remarkable performance across multiple benchmarks when evaluating both training and testing data, achieving an F1-Score of 1.0 for the micro-average, macro-average, and weighted-average. The F1-Score weighted-average values of other methods for the training data were as follows Linear Support Vector Machine (LSVM) 0.92, LR 0.93, and LightGBM 0.96, respectively. Similarly, in the evaluation of the test data, the weighted-average criteria were determined as LSVM 0.92, LR 0.93, and LightGBM 0.96. In terms of processing time, the RF emerged as the most efficient method, completing the analysis in 677 seconds (sec). In contrast, other methods required significantly more time, with LightGBM taking 1231 sec, LR taking 2068 sec, and LSVM taking 23024 sec, respectively.

Dada et al. (2019) highlighted the absence of performance analysis in machine learning approaches for malware detection. They utilized a dataset comprising 5.184 samples (i.e., 2.683 malicious and 2.501 benign) with 55 different features. Model validation involved both a 66% split and the 10-fold cross-validation methodologies. Various ML methods including J48, Logistic Model Tree (LMT), NB, RF, MLP Classifier, Random Tree (RT), REP Tree, Bagging Classification (BC), AdaBoost, KStar, Simple Logistics (SL), Instance Based Parameter Learning by k (IBK), Local Weighted Learning (LWL), SVM and Radial Basis Function Network (RBFN) were used. Performance evaluation included metrics such as accuracy, precision, recall, kappa statistics, F-measure, Matthew Correlation Coefficient (MCC), Receiver Operating Characteristic (ROC), and Mean Squared Error (MSE). When 10-fold cross-validation was applied, the highest accuracy was the RF method with 99.2%, while the others were LMT, IBK, BC, J48, MLP, REP Tree, SL, RT, SVM, AdaBoost, LWL, KStar, RBFN, and NB, respectively. In the 66% split approach, the highest accuracy value was the RF method with 98.8%, while the others were LMT, J48, IBK, MLP, BC, REP Tree, SL, SVM, RT, AdaBoost, LWL, KStar, RBFN and NB, respectively.

Anderson and Roth (2018) introduced a novel dataset constructed from the binary form of Windows PE files. This dataset incorporated all PE headers as input, with each file uniquely assigned a Secure Hash Algorithm 256 (SHA256) number. The header information was converted into JavaScript Object Notation (JSON) objects, while inputs with numerous attributes were represented as vector forms. The dataset encompasses 1.1 million (i.e., 900.000 training and 200.000 testing) samples. In addition to dataset creation, the Gradient Boosting Decision Tree (GBDT) model and the LightGBM classifier were employed to assess its effectiveness. The GBDT was optimized for maximum efficiency within minimal training time, while the LightGBM leveraged its advantages through histogram-based and discrete form conversion. Performance evaluation centered on metrics such as FPR, True Positive Rate (TPR), and ROC-AUC. The findings highlighted a TPR of 98.2% and an AUC threshold value of 87.7% for the generated dataset.

Raff et al. (2018) proposed a prediction model using raw byte sequences of the PE. In addition to using two separate datasets (i.e., groups A, and B) in the literature as a dataset, the raw data section of a PE containing 2 million rows in total was also used. It has created a shallow architecture (500-byte step) combined with a large filter width for the model built using Neural Networks (NN). Additionally, the n-gram approach has been used for byte strings in the PE Header. Accuracy and AUC metrics were the evaluation criteria for the separate models created. Group A provided 90.8% accuracy for the PE Header, while Group B provided 92.5% accuracy on n-grams. However, when trained with the approximately 2 million datasets created, group A provided 94% accuracy for the proposed NN model, while n-gram provided 91.6% accuracy.

Table 2.1. An overview of the Static-based Malware Analysis (Part 1)

Study	Year	Types	Methods	Metrics	Dataset Samples	Validations
Balodi et al.	2023	Regular	RF	Accuracy, Precision, Recall, F1-Score	19.000	n.a.
Yousuf et al.	2023	FS (IG, PCA)	SVM, GB, KNN, NC, DT, RF, NB, MV, SG, AdaBoost	Accuracy, Precision, Recall, F1-Score	29.797	10-fold
Srastika et al.	2022	FS (Boruta, ANOVA)	RF, KNN, XGBoost	F1-Score, Accuracy, MCC	33.665	n.a.
Moon et al.	2022	FS (FI)	RF	Accuracy, Precision, Recall, F1-Score	282.421	n.a.
Hussain et al.	2022	FS (ETC)	RF, DT, SVM, AdaBoost, GB, NB	Accuracy, Precision, Recall, F1-Score, Support	138.042	n.a.
Rkhouya and Chougdali	2021	FS (ETC)	DT, GB, SVM, RF	Processing Time, Accuracy, FPR, FNR, AUC	135.387	n.a.
Kumar et al.	2020	FS (ETC)	DT, RF, GB, SVM, LR, XGB, AdaBoost	Accuracy, Precision, Recall, F1-Score, Training Time	121.000	n.a.
Mohammed et al.	2020	FS (Tree-Based)	RF, DT	Accuracy, Confusion Matrix	138.048	n.a.

Abbreviations: RF, Random Forest; n.a., Not Available; FS, Feature Selection; IG, Information Gain; PCA, Principle Component Analysis; SVM, Support Vector Machine; GB, Gradient Boosting; KNN, K-Nearest Neighbors; NC, Nearest Centroid; DT, Decision Trees; NB, Naïve Bayes; MV, Majority Voting; SG, Stacking Generalization; AdaBoost, Adaptive Boosting; ANOVA, Analysis of Variance; XGBoost, eXtreme Gradient Boosting; MCC, Matthews Correlation Coefficient; FI, Feature Importance; ETC, Extra Trees Classifier; FNR, False Negative Rate; AUC, Area Under the Curve

Table 2.2. An overview of the Static-based Malware Analysis (Part 2)

Study	Year	Types	Methods	Metrics	Dataset Samples	Validations
Balram et al.	2019	Regular	SVM, LR, RF, XGBoost, NB, (LR, XGBoost), (LR, RF,NB)	Accuracy, Precision, Recall, F1-Score, Execution Time	1.416	5-fold
Radwan	2019	FS (Mean and Standard Deviation)	KNN, GB, DT, RF, FLM, LR, NB	Accuracy, F-Measure, Recall, Precision	5.184	10-fold / 70-30 Splitting
Zhang et al.	2019	Regular	RF, LSVM, LR, LGBM	F1-Score, Processing Time, Confusion Matrix	750.000	n.a.
Dada et al.	2019	Regular	J45, LMT, NB, RF, MLP, RT, REP Tree, AdaBoost, KStar, SL, IBK, LWL, SVM, RBFN	Accuracy, Precision, Recall, Kappa Statistics, F1-Score, ROC, RMSE	5.184	10-fold / %66 splitting
Anderson and Roth	2018	Regular	GBDT, LGB	FPR, TPR, AUC	1.1M	n.a.
Raff et al.	2018	Regular	NN	Accuracy, AUC	2M	n.a.

Abbreviations: SVM, Support Vector Machine; LR, Logistic Regression; RF, Random Forest; XGBoost, eXtreme Gradient Boosting; NB, Naive Bayes; FS, Feature Selection; KNN, K-Nearest Neighbors; GB, Gradient Boosting; DT, Decision Tree; FLM, File Large Margin; LGBM, Light Gradient Boosting Machine; n.a., Not Available; LMT, Logistic Model Trees; MLP, Multi-Layer Perceptron; RT, Regression Trees; REP Tree, Reduced Error Pruning Tree; AdaBoost, Adaptive Boosting; KStar, K* Algorithm; SL, Simple Logistic; IBK, Instance-Based Learning k-Nearest Neighbors; LWL, Locally Weighted Learning; RBFN, Radial Basis Function Network; GBDT, Gradient Boosted Decision Trees; FPR, False Positive Rate; TPR, True Positive Rate; AUC, Area Under Curve; NN, Neural Network

2.2. Studies Based on Dynamic Malware Analysis

Aboaoja et al. (2023) introduced a dynamic malware detection method focusing on API call sequences specific to the Windows platform. The dataset comprises 7.208 malware and 3.848 benign samples, with 60% reserved for training and 40% for testing. The evaluation utilized 10-fold cross-validation with KNN, CART, NB, SVM, ANN, RF, LR, and XGBoost ML methods. Model performance metrics, including accuracy, detection rate, sensitivity, and F1-Score, were assessed, and the whisker box plot method was applied. The K parameter was set to values of 0.5, 1.0, 1.5, and 2.0. Notably, when K was set to 0.5, both ANN and XGBoost methods exhibited an accuracy of 96.20%. As the K parameter varied, XGBoost consistently demonstrated high accuracy, achieving 96.70%, 95.5%, and 95.8% when K was set to 1.0, 1.5, and 2.0, respectively.

Akhtar and Feng (2023) proposed a high-performance system for dynamic malware analysis detection. They utilized a dataset available on the Kaggle online platform, comprising a total of 531 samples, with 373 being malware and 72 benign. The characteristic behavior of each sample was isolated in a controlled environment, specifically the Sandbox, and marked in a binary system for each file. Feature selection was performed using feature ranking, and the dataset was divided into training and testing sets. The model training involved KNN, DT, RF, AdaBoost, Stochastic Gradient Descent (SGD), ETC, and NB algorithms. The performance of model was evaluated using accuracy, F1-Score, recall, and precision criteria. Upon examination of the results, RF, SGD, ETC, and NB algorithms exhibited a 100% accuracy rate, while DT, KNN, and AdaBoost showed a 99% accuracy rate.

Mohanasruthi et al. (2020) initiated a new study focusing on malware detection in a complex network framework. The system demonstrated high detection rates using a complex network topology created from the API-Call Transition Matrix (API-CTM). This approach examined various metrics of the complex network to distinguish between malicious and benign applications. The dataset consisted of records detailing the behavior of PE files, specifically system calls recorded in a file covering 80 system calls. The Pearson Correlation Coefficient (PCC) and other feature selection methods were employed, including API calls for links that PE files point to other extensions. The findings revealed that MLP achieved the highest accuracy at 96.25%, followed by RF at 63.75%, NB at 53.75%, and SVM at 42.5%.

Li et al. (2020) introduced a dynamic malware classification and detection system using machine learning methods and opcode n-gram approaches. The dataset consisted of 7.927 malware and 4.070 benign files. Deep Neural Network (DNN), SVM and XGBoost algorithms were used in the proposed approach, and model performance was evaluated according to precision and recall criteria. In the first experimental results, XGBoost showed the highest sensitivity with 99%. XGBoost performed 99% of the recall criteria. Precision criteria for correct file classification were 96% for XGBoost, 96% for DNN, and 95% for SVM, while recall criteria were 99% for XGBoost,

97% for DNN, and 95% for SVM. In another set of experiments involving changes to the content of 10 randomly selected new malware samples, XGBoost outperformed DNN and SVM, respectively.

Liu et al. (2017) introduced an approach that leverages classification algorithms for static variants of malicious files and uses the opcode approach to identify new malware. A dataset containing 21.740 samples was used, the data were grouped into 8-bits using the grayscale approach and converted into binary form, where the numerical representation was assigned to data in the range of 0 to 255. Feature extraction was carried out using the N-gram method, and information gain feature selection was applied to reduce the features of the dataset and Shared Nearest Neighbors (SNN), RF, KNN, GB, NB, LR, SVM, and DT ML methods were included. Model evaluation was based on accuracy and the findings showed that GB achieved the highest accuracy value of 96.1% in the grayscale image approach. Regarding the opcode n-gram approach (i.e., n = 3), RF achieved the highest accuracy of 93.6%.

Manavi and Hamzeh (2017) proposed a new malware detection method using the opcode sequence visualization approach. The dataset instances published by VX Heavens were used. The dataset instances sourced from VX Heavens were employed, comprising a total of 3.400 samples, including 1.600 benign and 1.800 malicious instances. The dataset contained binary data, it underwent conversion into embedded system codes through various methods. A frequency matrix was generated from these embedded system codes. To transform this frequency matrix into grayscale picture form, representative color codes between 0 and 225 were assigned corresponding to the frequency numbers, resulting in the creation of a new picture matrix. Following this process, the KNN, SVM, and AdaBoost algorithms were applied. The evaluated metrics included TPR, FPR, accuracy, and F-measure. The findings indicated that SVM achieved an accuracy of 93.17%, followed closely by AdaBoost at 93.06%, and KNN at 91.7%.

Yewale and Singh (2016) proposed a malware detection system based on opcode frequency for the PE files. In the dataset, malicious software belonging to Cuckoo Sandbox and PE files in the system32 file in the Windows operating system were used. There are 100 samples, and analysis was completed in assembly language using reverse engineering tools. By using Principal Component Analysis (PCA) and feature rank feature selection algorithms, the twenty most repetitive opcode operations were used. Then, boosting, SVM, DT, and RF classification algorithms were applied and evaluated with True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN), TPR, FPR, and accuracy metrics. The results were as follows respectively RF 97%, SVM 93%, Boosting 87%, and DT 80% accuracies malware classification.

Galal et al. (2016) introduced a behavior-based dynamic feature model aimed at elucidating the malicious activity exhibited by malware samples. The primary focus was on capturing behavioral patterns, specifically system API calls categorized into five group such as process and subject management, files, registry, network, and internet. The dataset comprised a total of 9.993 samples. For behavioral analysis, each sample underwent opcode and n-gram (i.e., n = 2) approaches using

the ObjDump tool. The proposed model incorporated the utilization of DT, SVM and RF as classification algorithms. Additionally, 50 Hidden Markov Models (HMM) were employed to infer events resulting from internal factors that could not be observed directly. The evaluation metrics employed for assessment included sensitivity, specificity, accuracy, and AUC. The results obtained from the application of the proposed classification models revealed an accuracy of 97.19% for DT, 96.84% for RF, and 93.98% for SVM. Moreover, when classifying estimates were derived for the 50 HMMs, DT achieved 96.89%, RF yielded 96.14%, and SVM recorded 94.8%.

Fan et al. (2015) introduced a data mining technique for dynamic malware analysis using a dataset containing 7.000 malware samples and 251 benign samples. The focus of the method was on API function names in six key DLL files that represent API calls in PE files. J48, SVM, and NB were evaluated taking into account the algorithm and measurements such as TPR, FPR, sensitivity, recall, F1-Score, and accuracy. The findings unveiled two distinct approaches, both striving for maximum accuracy. The dataset was subdivided into subsets with varying features (i.e., 20, 40, 80, 120, 250, and 800) from a pool of 1.800 training data points, which were validated using a 10-fold cross-validation. In the first experiment, comprising 251 randomly selected benign and 263 malicious samples, J48 predicted an accuracy of 95.4%, NB showed 95.9%, and SVM reached 81.6%. In the second experiment with 251 benign and 773 malicious samples, J48 achieved 95.9% accuracy, NB 94.1%, and SVM 89.3% accuracy.

Singhal and Raul (2012) proposed a malware detection system that identifies malware activities and can even make predictions about a new file. A total of the 4.500 samples from a dataset consisting of user API calls on a network were investigated. These formats are archived in a database and processed with the aid of the PE file parser using ML techniques. The data dimensions were reduced using feature selection, and they were classified using RF. The results showed that the RF classification method has a 99.55% accuracy rate for correct classification.

Table 2.3. An overview of the Dynamic-based Malware Analysis (Part 1)

Study	Year	Types	Methods	Metrics	Dataset Samples	Validations
Aboaoja et al.	2023	FS (N-gram)	SVM, RF, CART, LR, KNN	Accuracy, F1-Score, Recall, Precision	11.056	10-fold / 60-40 Splitting
Akhtar and Feng	2023	FS (FR)	KNN, DT, RF, AdaBoost, SGD, ETC, NB	Accuracy, F1-Score, Recall, Precision	531	n.a.
Mohanasruthi et al.	2020	FS (PCM)	SVM, DT, NB, MLP	Accuracy	More than 1.000 samples	n.a.

Abbreviations: FS, Feature Selection; SVM, Support Vector Machine; RF, Random Forest; CART: Classification and Regression Trees; LR, Logistic Regression; KNN, K-Nearest Neighbors; DT, Decision Tree; AdaBoost, Adaptive Boosting; SGD, Stochastic Gradient Descent; ETC, Extra Trees Classifier; NB, Naive Bayes; n.a., Not Available; PCM, Principal Component Analysis; MLP, Multi-Layer Perceptron

Table 2.4. An overview of the Dynamic-based Malware Analysis (Part 2)

Study	Year	Types	Methods	Metrics	Dataset Samples	Validations
Li et al.	2020	Regular	XGB, DNN, SVM	Precision, Recall	11.997	n.a.
Liu et al.	2017	FS (IG)	RF, KNN, GB, NB, LR, SVM, DT, SNN	Accuracy, Precision, Recall, F1 Score, Model Training Time, Confusion Matrix	121.000	n.a.
Manavi and Hamzeh	2017	Regular	SVM, AdaBoost, KNN	TPR, FPR, Accuracy, F-Measure, Recall, Precision	3.400	n.a.
Yewale and Singh	2016	FS (PCA, FR)	Boosting, DT, SVM, RF	TP, TN, FP, FN, TPR, FPR, Accuracy	100	n.a.
Galal et al.	2016	Regular	DT, RF, SVM, HMM	Sensitivity, Specificity, Accuracy, AUC	9.993	n.a.
Fan et al.	2015	Regular	J48, NB, SVM	TPR, FPR, Precision, Recall, F1-Score, Accuracy	7.000	10-fold
Singhal and Raul	2012	FS (IG)	RF	Accuracy	4.500	n.a.

Abbreviations: XGB, eXtreme Gradient Boosting; DNN, Deep Neural Network; SVM, Support Vector Machine; n.a., Not Available; FS, Feature Selection; IG, Information Gain; RF, Random Forest; KNN, K-Nearest Neighbors; GB, Gradient Boosting; NB, Naive Bayes; LR, Logistic Regression; DT, Decision Tree; SNN, Shared Nearest Neighbors; AdaBoost, Adaptive Boosting; PCA, Principal Component Analysis; FR, Feature Ranking; HMM, Hidden Markov Model; TPR, True Positive Rate; FPR, False Positive Rate; TP, True Positive; TN, True Negative; FP, False Positive; FN, False Negative; AUC, Area Under Curve

2.3. Studies Based on Hybrid Malware Analysis

Ijaz et al. (2019) proposed an approach involving dynamic and static analysis using ML techniques for the PE files. The integrated dataset comprises the results of dynamic analysis consisting of more than 2.300 samples and static features extracted from 92 samples. A matrix was created detailing file operations such as reading, writing, and deleting. Additionally, the analysis took into account the processing of Domain Name System (DNS) queries in nine different combinations, as well as API calls that facilitate communication between software components and hardware. Conversely, static analysis was based on the PE Header files. Various classification models such as LR, DT, RF, BC, AdaBoost, Tree Classifier, and GB Classifier have been applied to the created models. Evaluation criteria were based on AUC. Experimental results showed that the highest AUC value in dynamic analysis was achieved by GB Classifier with 94.64%. The success of the other methods was AdaBoost, BC, LR, DT, RF, Tree Classifier, respectively. Regarding static analysis, GB Classifier exhibited the highest AUC value of 99.36%. The success of other methods was AdaBoost, Tree Classifier, RF, BC, DT, and Linear Ridge Classifier (LRC), respectively.

Dhammi and Singh (2015) proposed a hybrid approach system that can analyze behavior-based and signature-based malicious files. Datasets include the PE file format, Hypertext Markup Language (HTML) document format, compressed files, and Joint Photographic Experts Group (JPEG) file format. The evaluation criteria used include TPR, FPR, precision, recall, accuracy, and 10-fold cross-validation. LMT, NB, SVM, Ridor Classifier, and KNN were selected for classification. In the proposed system, two different approaches are presented, one with feature selection and another without feature selection. In the model created without feature selection, the LMT classifier was the most successful approach with 73.42% accuracy. The accuracy values of the other approaches were SVM 71.01%, NB 68.23%, KNN 61.36%, and Ridor Classifier 54.27%, respectively. In the model created using feature selection, the LMT classifier had the best approach with 98.28% accuracy. The accuracy values of the other approaches are SVM 97.71%, NB 96%, Ridor Classifier 94.28%, and KNN 81.63%, respectively.

Santos et al. (2013) proposed a machine-learning-based model that performs static and dynamic analysis of malicious files. In the static analysis, the opcode sequence was created at the operation code level of the Windows PE files, and many combinations were created with the n-gram model. The generated opcode sequences contain many machine language codes for the PE files. In dynamic analysis, Windows PE files were run in an isolated environment, and many movements such as registry keys, system calls, characteristic structures, and processes were monitored and recorded. Then, the data consisting of dynamic and static analysis were combined, and a hybrid model was obtained. The 13.189 malware samples were selected from online platforms, while 13.000 unique PE files were created. While the learning of the model was provided by RF, J48, KNN, NB, SVM, and TPR, FPR, accuracy, and AUC criteria were used to evaluate the model. Experimental results

showed that in the hybrid approach to malware analysis, SVM achieved 96.60%, KNN 96.22%, RF 95.19%, J48 93.59%, and NB 93.53% accuracies.

Table 2.5. An overview of the Hybrid-based Analysis

Study	Year	Types	Methods	Metrics	Dataset Samples	Validations
Ijaz et al.	2019	Regular	LR, DT, RF, BC, AdaBoost, TC, GB, LRC	AUC	Dynamic 2.300 samples, Static 92 samples	n.a.
Dhammi and Singh	2015	FS (Feature Importance)	LMT, NB, SVM, KNN, RC, K-Means	TPR, FPR, Precision, Recall, Accuracy	1.270	10-fold
Santos et al.	2013	Regular	RF, J48, KNN, NB, SVM	TPR, FPR, Accuracy, AUC	26.189	n.a.

Abbreviations: LR, Logistic Regression; DT, Decision Tree; RF, Random Forests; BC, Bagging Classifier; AdaBoost, Adaptive Boosting; GB, Gradient Boosting; LRC, Linear Ridge Classifier; n.a., Not Available; AUC, Area Under Curve; LMT, Logistic Model Tree; NB, Naive Bayes; SVM, Support Vector Machine; KNN, K-Nearest Neighbors; TPR, True Positive Rate; FPR, False Positive Rate; RC, Ridor Classifier; TC, Tree Classifier

2.4. Studies Based on Network Malware Analysis

Hidayat et al. (2023) proposed a ML-based instruction detection system using the TON_IoT dataset. Data preprocessing steps, including standardization, normalization, PCC, and RF feature selection, were applied to the dataset. In addition to DT, AdaBoost, and KNN machine learning methods, MLP and Long Short-Term Memory (LSTM) approaches were employed as deep learning methods. The model's performance was evaluated based on accuracy, recall, precision, and the ROC curve. Upon examination of the findings, AdaBoost emerged as the best machine learning method with 99.8% accuracy, while MLP stood out as the best deep learning method with 99.2% accuracy.

Thakkar and Lohiya (2023) introduced a novel approach for network instruction detection, employing deep learning techniques alongside feature selection. The study utilized a substantial dataset consisting of 631.935 samples drawn from three distinct datasets (i.e., NSL-KDD, UNSW_NB-15, and CIC-IDS-2017, respectively). Central to the research was the creation of a statistical feature selection process, which compared various methods including median, standard deviation, and mean difference. The methodology employed sequential deep learning networks. Evaluation of the models was conducted using multiple metrics including accuracy, precision, recall, F1-Score, FPR, and execution time. Analysis of the findings revealed that the implemented statistical feature selection method achieved notable success, demonstrating accuracy rates of 99.84%, 89.03%, and 99.80% across the three different datasets, respectively.

Ilyas and Alharbi (2022) utilized ML techniques to investigate modern internet traffic patterns. Their research involved compiling a comprehensive dataset comprising various cyber threats, such as bot attacks, Denial-of-Service (DoS) assaults, Brute-Force attempts, and infiltration maneuvers. The evaluation of the model included the use of the 10-fold cross-validation method, assessing its performance with algorithms like DT, RF, NB, LSVC, and MLP. Measurements were made based on accuracy, precision, recall, and F1-Score metrics. The results highlighted the effectiveness of DT and RF algorithms in detecting bot attacks based on accuracy metrics. However, for other types of attacks, the best-performing methods were as follows: LSVC in DoS-SlowHTTPTest/DoS-Hulk attacks; DT and RF in BruteForce-Web/SQL Injection/XSS attacks; DT, RF, and MLP in Infiltration attacks; DT and RF in DoS-GoldenEye/DoS-Slowloris attacks; and DT, RF, NB, LSVC, and MLP in BruteForce SSH/BruteForce-FTP attacks, with DT, RF, NB, and LSVC in DoS LOIC-UDP/DDOS-HOIC attacks.

Moualla et al. (2021) proposed a novel ML-based instruction detection system using the UNSW-NB15 dataset. To address irregularities in the dataset, the Synthetic Minority Oversampling Method (SMOTE), a resampling technique, was applied to generate new synthetic data. The dataset preparation included the classification of malware activity. Various metrics, including accuracy, sensitivity, recall, FAR, F1-Score, specificity, ROC, and Precision-Recall Curves (PRCs), were considered in the model evaluation. The results demonstrated 100% accuracy for shellcode, 98% accuracy for DoS and general activities, 97% accuracy for worms, and 94% accuracy for exploits. Additionally, the analysis predicted backdoors, fuzzers, and discoveries with 99% accuracy.

Jha and Sharma (2021) presented a method for cloud protection based on user activity, utilizing the UNSW-NB15 dataset, which includes over 82.000 samples and 49 features as well as nine different attack methods. The study comprehensively reviewed the utility of various ML techniques, including SVM, DT, NB, RF, KNN, and LR, applied to predict user behavior. The performance of the model with and without feature selection was compared in terms of accuracy, sensitivity, and specificity. Without feature selection, RF achieved an accuracy of 97.71%, sensitivity of 97.85%, and specificity of 97.60%. The DT showed an accuracy of 96.68%, and KNN exhibited an accuracy of 80.42%, while with feature selection, DT achieved an accuracy of 96.77%, and KNN reached 80.45%. The study aimed to create a framework for detecting malicious behavior in cloud environments to safeguard data from various types of attacks. The highest accuracy value was observed for RF at 97.71%.

Kocher and Kumar (2021) presented a ML-based model that detects anomalies in the mesh network. The proposed Instruction Detection System (IDS) can easily capture network traffic bypassed by the firewall, and when combined with ML techniques, it was suggested that a much more effective system could be achieved. Therefore, training and testing were performed on KNN, LR, NB, RF, and SGD ML algorithms using the UNSW-NB15 dataset. To select irrelevant and unrelated tags from the dataset, the feature was selected using the Chi-Square technique. In addition,

all labels and associated labels in the dataset were trained and evaluated separately. The model was evaluated with accuracy, precision, MSE, recall, F1-Score, TPR, and FPR. The highest accuracy in the model created without feature selection is RF with 99.57%, according to the experimental results. Others are LR 98.42%, KNN 98.28%, SGD 99.16%, and NB 76.59%, respectively. In the model built using feature selection, the highest accuracy is RF at 99.64%. Others are KNN 98.90%, LR 98.17%, SGD 97.99%, and NB 75.16%, respectively.

Hammad et al. (2020) presented a model for anomaly detection using clustering and classification approaches. The UNSW-NB15 dataset contained a total of 2.540.044 samples, 47 features, and binary labels. The Correlation-Based Feature Selection (CFS) was applied for size reduction, and eight features were selected. NB, RF, J48, and ZeroR classification methods are used in ML. In addition, K-means and Expectation Minimization (EM) were used in the clustering strategy. Accuracy, precision, recall, F-measure, FPR, specificity, and 10-fold cross-validation, were used as evaluation criteria for the model. The classification accuracy of the proposed model was NB 76.04%, J48 93.78%, RF 97.60%, and ZeroR 68.06%, respectively, while the K-Means misclustering rate for clustering was 4.6% and EM 32.03%. As a result, when feature selection is applied, the best classification model is RF with 97.6%.

Bagui et al. (2019) presented a hybrid model using the UNSW-NB15 dataset. From a total of 2 million samples in the dataset, 8.000 samples with the 9 rarely different attack types were randomly selected. In the proposed model, the size of the features in the dataset was reduced by using K-means clustering and CFS for feature selection. In feature selection, K-means clustering was selected separately based on the means of the features, and CFS evaluated the benefits of each candidate subset. By applying NB and J48 methods, which are classification algorithms for the selected model, accuracy was evaluated with Attack Detection Rate (ADR) and FAR metrics. The findings showed that feature selection is very important for probability-based NB algorithms, but not so important for J48 tree-based systems. Worm detection had the highest accuracy score of 99% for the NB Classifier, although the ADR and FAR rates were also high. The best accuracy of J48 Classifier for worm detection was 99.94%.

Nawir et al. (2019) introduced a model for anomaly detection using the UNSW-NB15 dataset, which includes various classification methods. The dataset consisted of 257.673 samples and 44 features focusing on binary data that distinguish normal from abnormal data. The model includes two different paths. The first is to use 10-fold cross-validation, while the other is the splitting method. In the splitting method, it was split as 20, 40, 60, 80, and 100 for training and testing. Accuracy and process time were taken into consideration to evaluate the models. Results showed that the J48 algorithm had the highest accuracy score of 98.71%, but the processing time was approximately 32 seconds. In contrast, the Average One-Dependence Estimators (AODE) algorithm processed in approximately 7 seconds with a 97.26% accuracy rate. Remarkably, NB showed the lowest accuracy at 76.12% but the fastest processing time at 1.22 seconds. MLP achieved 89.76% accuracy with an

extended processing time of approximately 8 hours, while RBFN achieved 84.41% accuracy and processed in 7.50 seconds. As a result, the AODE algorithm showed promising results, outperforming J48 in detecting network anomalies despite a slower processing time.

Table 2.6. An overview of the Network-based Analysis (Part 1)

Study	Year	Type	Methods	Metrics	Dataset Samples	Validations
Hidayat et al.	2023	FS (PCC, RF)	DT, AdaBoost, KNN, MLP, LSTM	Accuracy, Recall, Precision, ROC	461.043	n.a.
Thakkar and Lohiya	2023	FS (RFE, Chi-Square, CFS, GA, MI, Relief-F, RF, Median, SD, MD)	DNN	Accuracy, Precision, Recall, F1-Score, FPR, ET	631.935	n.a.
Ilyas and Alharbi	2022	FS (Gain Ratio)	DT, RF, NB, LSVC, MLP	Accuracy, Precision, Recall, F1-Score	7M.950K	10-fold
Moualla et al.	2021	FS (ETC)	ELM, LR	Accuracy, Precision, Recall, Specificity, F1-Score, FAR, ROC, PRC	257.673	n.a.
Jha and Sharma	2021	FS (Chi-Square)	SVM, DT, NB, RF, KNN, LR	Accuracy, Sensitivity, Specificity, Confusion Matrix	82.332	n.a.
Kocher and Kumar	2021	FS (Chi-Square)	KNN, LR, NB, RF, SGD	Accuracy, Precision, Recall, F1-Score, TPR, FPR, MSE	257.673	n.a.
Hammad et al.	2020	FS (CFS)	NB, RF, J48, ZeroR, K-Means, EM	Accuracy, Precision, F1-Score, FPR, Specificity	2.540.044	10-fold

Abbreviations: FS, Feature Selection; PCC, Pearson Correlation Coefficient; RF, Random Forest; DT, Decision Tree; AdaBoost, Adaptive Boosting; KNN, K-Nearest Neighbors; MLP, Multi-Layer Perceptron; K, Thousand; LSTM, Long Short-Term Memory; n.a., Not Available; RFE, Recursive Feature Elimination; CFS, Correlation-based Feature Selection; GA, Genetic Algorithm; MI, Mutual Information; SD, Standard Deviation; MD, Mean Difference; DNN, Deep Neural Network; FPR, False Positive Rate; ET, Execution Time; NB, Naive Bayes; LSVC, Linear Support Vector Classifier; ETC, Extra Trees Classifier; ELM, Extreme Learning Machine; LR, Logistic Regression; FAR, False Alarm Rate; ROC, Receiver Operating Characteristic; PRC, Precision-Recall Curve; SVM, Support Vector Machines; SGD, Stochastic Gradient Descent; MSE, Mean Squared Error; K-Means, K-Means Clustering; EM, Expectation-Maximization Algorithm

Table 2.7. An overview of the Network-based Analysis (Part 2)

Study	Year	Type	Methods	Metrics	Dataset Samples	Validations
Bagui et al.	2019	FS (K-Means, CFS)	NB, J48	Accuracy, ADR, FAR	8.000	n.a.
Nawir et al.	2019	Regular	NB, AODE, RBFN, MLP, J48	Accuracy, Processing Time	257.673	10-fold / 20,40,60,80 Splitting

Abbreviations: FS, Feature Selection; K-Means, K-Means Clustering; CFS, Correlation-based Feature Selection; NB, Naive Bayes; n.a., Not Available; ADR, Attack Detection Rate; FAR, False Alarm Rate; AODE, Average One-Dependence Estimators; RBFN, Radial Basis Function Network; MLP, Multi-Layer Perceptron

3. THEORETICAL FOUNDATIONS OF METHODS

This chapter is divided into two distinct parts. The initial section delves into the theoretical intricacies of ML classifiers, while the subsequent subsection elucidates the operational principles behind feature selection algorithms.

3.1. Machine Learning Classifiers

In this section, the theoretical information and working principles of six different ML classifiers, including LightGBM, DT, KNN, MLP, SVM and NB, and Relief-F and mRMR-based feature selectors are discussed in detail.

3.1.1. Light Gradient Boosting Machine

The LightGBM, a ML algorithm developed by Microsoft in 2017, efficiently handles large-scale datasets through Gradient Boosting (Ke et al., 2017). This algorithm distinguishes itself with its low memory consumption, Graphically Process Unit (GPU) support, and parallel processing capabilities, offering significant advantages in accuracy and efficiency, especially in multi-class classification problems (Ge et al., 2019). By employing Gradient Boosting, which aims to construct a robust learner by amalgamating weak learners like decision trees, the LightGBM swiftly processes extensive datasets. Its optimization of memory usage further enhances operational efficiency, surpassing traditional Gradient Boosting methods. Assume that input space X, towards the gradient space G. A training set comprising instances $\{x_1, \dots, x_n\}$, where each x_i is a vector dimension s in X. In each iteration of gradient boosting, all the negative gradients of a loss function concerning the outpu model are denoted as $\{g_1, \dots, g_n\}$. The decision tree divides each node based on the most informative feature, resulting in the largest gain in evidence. In this model, the improvement in data segregation can be quantified by the variance. It can be represented by following formula:

$$Y = BaseTree(X) - lr.Tree1(X) - lr.Tree2(X) - lr.Tree3(X) \tag{3.1.}$$

Let O be a training dataset on a fixed node of a decision tree and then the variance gain of dividing measure j at a point d for a node is defined as:

$$V_{i|O}(d) = \frac{1}{n_O}\left(\frac{\left(\sum_{\{x_i \in 0: x_{ij} \le d\}} g_i\right)^2}{n^j_{l|O}(d)} + \frac{\left(\sum_{\{x_i \in 0: x_{ij} > d\}} g_i\right)^2}{n^j_{r|O}(d)}\right) \tag{3.2.}$$

where, $n_O = \sum I[x_i \in O]$, $n^j_{l|O}(d) = \sum I\left[x_i \in O : x_{ij} \le d\right]$ and $n^j_{r|O}(d) = \sum I\left[x_i \in O : x_{ij} > d\right]$.

Gradient One-Sided Sampling (GOSS) uses every sample with large gradients and performs random sampling on samples with small gradients. The training data set for a particular node of the decision tree is denoted by the notation O. The variance gain or split measure of node j at point d is expressed as:

$$\tilde{V}_j(d) = \frac{1}{n}\left(\frac{\left(\sum_{\{x_i \in A_l\}} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i\right)^2}{n_l^j(d)} + \frac{\left(\sum_{\{x_i \in A_r\}} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i\right)^2}{n_r^j(d)}\right) \quad (3.3.)$$

where, $A_l = \{x_i \in A : x_{ij} \leq d\}, A_r = \{x_i \in A : x_{ij} > d\}, B_l = \{x_i \in B : x_{ij} \leq d\}, B_r = \{x_i \in B : x_{ij} > d\}$, and the coefficient $\frac{1-a}{b}$ is used to normalize the sum of the gradients over B back to the size of A^c (Ke et al., 2017).

3.1.2. Decision Tree Classifier

The DT, also known as classification and regression trees, emerges as a fundamental cornerstone within information extraction systems. Their inherent simplicity and comprehensible architecture render them an appealing and substantive choice for end-users. The fundamental aim behind their construction lies in the acquisition of a structured tree arrangement capable of facilitating predictions by unveiling intricate patterns inherent within the data. The recursive construction of these trees is facilitated through an induction process, notably the Top-Down Induction on Decision Trees (TDIDT), which initiates from higher nodes and progressively expands downwards. Classical algorithms like ID3, C4.5, and Classification and Regression Tree (CART) commonly serve as pivotal tools in the creation of these sophisticated tree structures (Martínez Torres et al., 2019).

The DT exhibit remarkable adaptability in handling both numerical and categorical datasets. Nevertheless, their susceptibility to minor fluctuations within the data renders them notably sensitive. Even minute alterations in the dataset possess the potential to yield entirely distinct tree structures, thereby generating intricate patterns that lack generalizability. Termed as the "instability" of decision trees, this phenomenon contributes to lack of stability of the model and can impede its reliability and robustness (Jain et al., 2022).

3.1.3. K-Nearest Neighbors

The KNN is a rudimentary yet potent non-parametric classification approach. Its core principle revolves around identifying the *k* nearest neighbors of a given data point and determining its classification based on the majority class among these neighbors. The pivotal aspect influencing its efficacy lies in the parameter *k* selection, significantly shaping the classification outcome. The

typical model optimization process entails iterative experimentation across varying k values to discern the optimal performance threshold.

However, a notable drawback of the KNN surfaces in its demanding computational cost during the classification of novel instances, owing to the real-time computation requirements. Operating as a case-based learning method, KNN retains all training data for classification, which, although effective, imposes constraints, notably in dynamic web mining for expansive datasets. To enhance efficiency of KNN, a proposed approach involves identifying representative subsets that encapsulate the entire dataset, thereby constructing a learning model. Utilizing this model for subsequent classifications becomes a viable strategy. The evaluation of different algorithms often incorporates performance measures as a benchmark standard (Guo et al., 2003).

The KNN classifier is an algorithm centered around the classification of data points. There isn't a specific formula for determining the k value that selects the nearest neighbor. This varies depending on the dataset size and the approach to problem-solving. However, there are numerous methods for measuring distance such as Euclidean, Manhattan, and Minkowski.

Euclidean distance is a common and fundamental metric used to measure the linear separation between individual data points. Mathematically, it involves obtaining the square root of the inequalities within the coordinates of two different points and thus determining the Euclidean distance as a measure indicating the linear distance between these points. The formula containing the Euclidean distance is as follows:

$$\sqrt{\sum_{i=1}^{n}(x_{i1} - x_{i2})^2} \tag{3.4.}$$

The Manhattan distance is an alternative distance measurement method to Euclidean distance. The formula for Manhattan distance is as follows:

$$\sum_{i=1}^{n}|x_i - y_i| \tag{3.5.}$$

Another distance measurement method of the KNN algorithm is Minkowski. This distance is a formula that can be used with the p parameter value. This parameter provides a generalized form of different distances. Minkowski distance measurement formula is as follows:

$$\left(\sum_{i=1}^{d}|x_i - y_i|^p\right)^{\frac{1}{p}} \tag{3.6.}$$

3.1.4. Multi-Layer Perceptron

The MLP is a type of artificial neural network that includes an input layer, an output layer, and at least one hidden layer. Each node in the layers is referred to as a neuron, and there are various

numbers of neurons in each layer. Each neuron is connected to every neuron in the preceding and succeeding layers, but not to neurons within the same layer. To optimize outputs towards desired results and minimize the difference between the network output and the desired output, every connection in the model possesses a value known as a weight (Ramchoun et al., 2017).

In artificial neural networks such as MLP, each neuron incorporates a crucial element known as an activation function, which significantly influences the network's behavior. Among the array of activation functions available, ReLU, Logistic (i.e., Sigmoid), and Hyperbolic Tangent (tanh) stand out as commonly utilized choices (Misra, 2019). This study includes the combined use of a number of parameters such as hidden layer, activation function, alpha, solver, batch, and momentum.

The Sigmoid activation function also called as Logistic presents an approach involving Logistic Regression classification algorithms. This function positions the inputs between 0 and 1, as shown in Figure 3.1. Particularly beneficial for classification problems, this characteristic stands out, making it suitable for usage in gradient-based algorithms. However, instances of excessively high or low gradient values result in training stagnation or increased computational costs. The formula of the logistic activation function, as shown in (3.7.):

$$f(x) = \frac{1}{1+e^{-x}} \tag{3.7.}$$

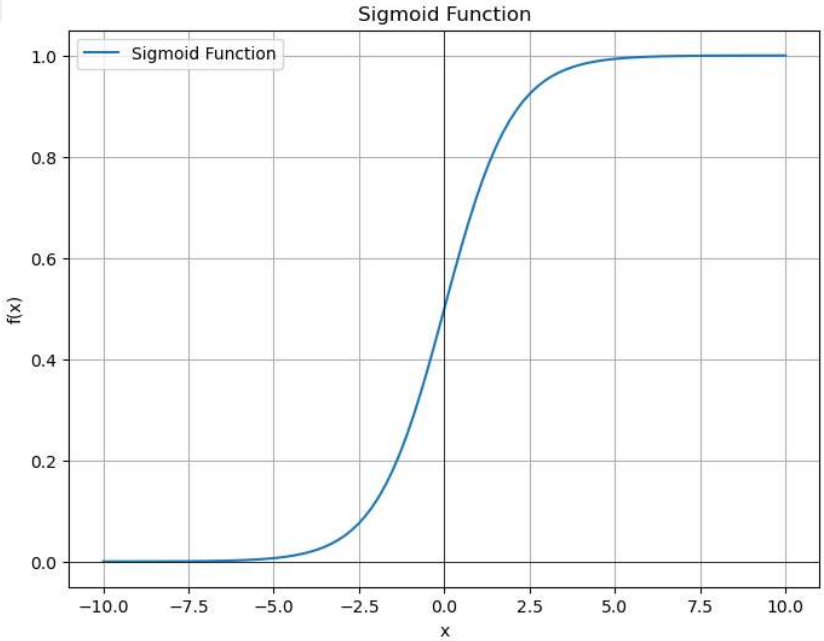


Figure 3.1. Graph of Sigmoid Function

The Hyperbolic Tangent function, also known as the tanh activation function similar to the sigmoid function, yields outputs between -1 and 1, as illustrated in Figure 3.2. This output range is advantageous as it centers around zero or close to zero on average. This characteristic can expedite the learning process. However, similar to the sigmoid function, excessively high or low gradient

values can escalate computational costs. The formula of the tanh activation function, as shown in (3.8.):

$$\tan(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.8.}$$

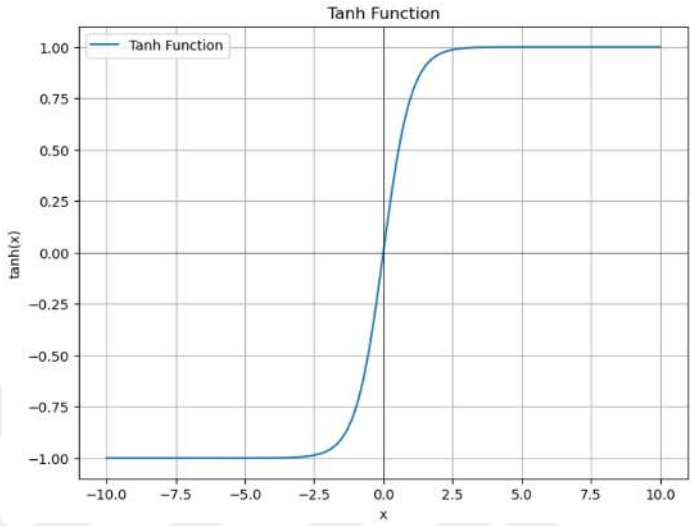


Figure 3.2. Graph of Hyperbolic Tangent Function

The Rectified Linear Unit (ReLU) stands out among activation functions with its simplicity and consistent performance (Misra, 2019). The primary aim here is to nullify negative values of the function while retaining positive ones. The conversion of negative values to zero serves to expedite the gradient, the vectorial measure of the function's slope at a specific point, throughout the learning process of the network. This concept is illustrated in Figure 3.3. This alteration in the gradient facilitates accelerated and effective learning, thereby enhancing the efficiency of the network's learning process. The formula of the ReLU activation function, as shown in (3.9.):

$$f(x) = \max(0, x) \tag{3.9.}$$

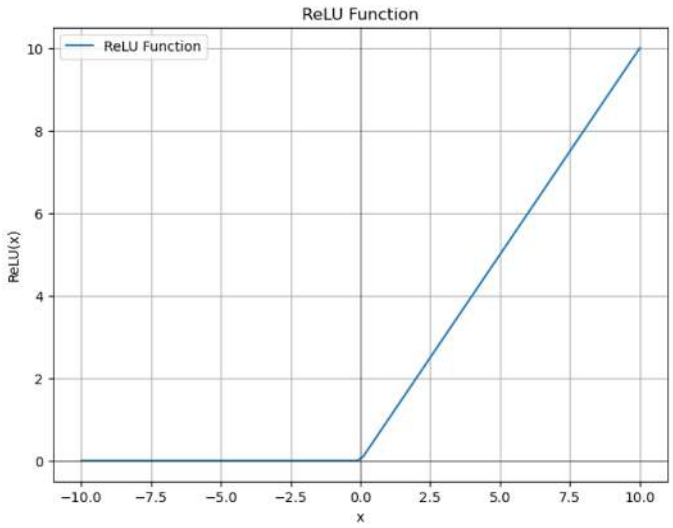


Figure 3.3. Graph of ReLU Function

Alpha aims to mitigate the risk of overfitting in an artificial neural network model by controlling the magnitude of weights. This is commonly known as L2 regularization or Ridge regularization. As the alpha value increases, the influence of weights decreases, resulting in a less complex model. The solver parameter is an optimization parameter in an artificial neural network model and updates weights using different parameters, facilitating the learning process. Batch aim to enhance the efficiency of the training process. The primary purpose is to partition the dataset into smaller segments instead of processing the entire dataset in one go. Although an optimal batch size may not directly enhance the network performance of the model, it indirectly contributes to its improvement.

3.1.5. Support Vector Machine

The SVM, an algorithm rooted in statistical learning theory for regression or classification learning, operates by partitioning data onto a hyperplane to minimize classification errors. This hyperplane serves as the division between two parallel hyperplanes, effectively segmenting the data. Enhancing the generalization of classifiers involves increasing the separation distance of this hyperplane.

Through the transformation of input vectors into a higher-dimensional space, SVM constructs the hyperplane, aiming for maximal separation to better categorize data. The objective lies in creating a hyperplane that maximizes the distance between two parallel planes, effectively partitioning the data points (Bhavsar & Ganatra, 2012). The graph separating margin, support vectors and hyperplane for SVM is shown in Figure 3.4.

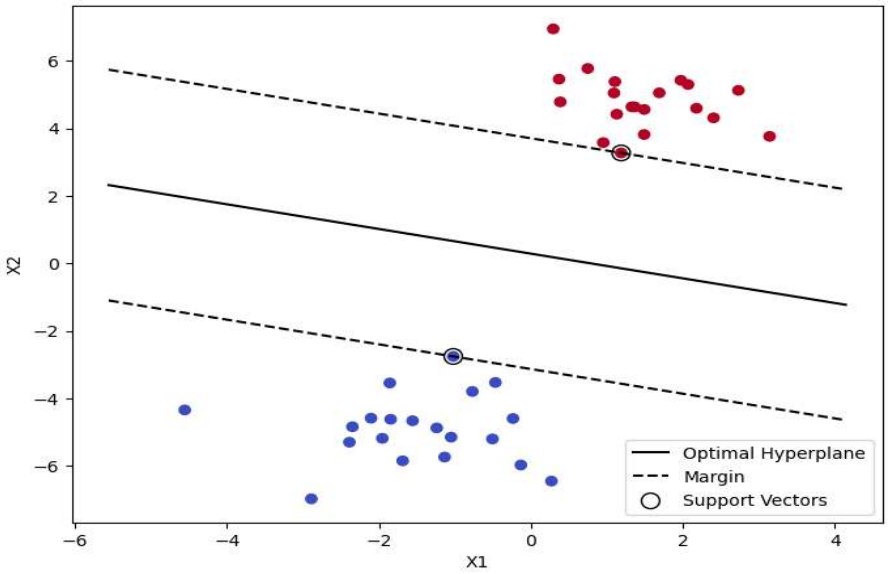


Figure 3.4. Graph of SVM Separating Hyperplane

3.1.6. Gaussian Naïve Bayes

The NB, rooted in Bayes' theorem, presents a straightforward and readily applicable probabilistic approach to classification. Its assumption of attribute variable independence contributes to its efficacy in supervised learning, enabling its utilization in complex real-world contexts. A notable advantage lies in its minimal dependency on extensive training data, a crucial aspect for characterization and classification tasks (Kamel et al., 2019).

The NB, particularly in handling continuous data, the classifier presupposes a Gaussian distribution for continuous attribute values associated with each class. This assumption guides the segregation of the training dataset by class, facilitating the computation of statistical parameters like mean and standard deviation for each class. Such an approach enables the characterization of attribute value distributions within respective classes based on Gaussian assumptions, thereby facilitating classification within continuous data domains (Kamel et al., 2019). The specific expression encapsulating the Gaussian NB formula is articulated as follows in (3.10.):

$$P(y|x_1, x_2, \dots, x_n) = \frac{P(y).P(x_1,x_2,\dots,x_n|y)}{P(x_1,x_2,\dots,x_n|y)} \quad (3.10.)$$

3.2. Feature Selection Methods

In this section, the definitions and working principles of important feature selectors such as Relief-F and mRMR are examined in detail. These selectors have been used to improve the classification performance of features, reduce their size, and improve models.

3.2.1. Relief-F

The Relief-F algorithm is a robust feature selection method that operates on instance-based principles. It functions by randomly selecting an instance and then identifying the nearest neighbors within the same class. Unlike the original Relief algorithm, Relief-F possesses the advantage of handling incomplete data and is not limited to binary classification problems. This versatility allows it to be applied across a broader range of scenarios, contributing to its effectiveness in feature selection across various datasets and problem types.

The Relief-F algorithm initializes weights for attributes to 0, similar to the original Relief algorithm (Line 1). Following this, like in the Relief algorithm, it randomly selects an instance (R_i) from the dataset (Line 3). However, unlike Relief, Relief-F identifies the k nearest neighbors of the same class as H_j (Line 4). Additionally, it seeks nearest misses, $M_j(C)$, by locating the k nearest neighbors from different classes (Lines 5 and 6). Subsequently, the algorithm updates the weights (W[A]) for all attributes (A) based on their values for R_i and H_j (Robnik-Sikonja & Kononenko, 2003).

Input: for each training instance, a vector of attribute values and the class value
Output: The vector W of estimations of the qualities of attributes

1. Set al.l weights W[A] := 0.0;
2. **for i** :=1 **to** m **do begin**
3. randomly select an instance R_i;
4. find k nearest hits H_j;
5. **for** each class C ≠ class(R_i) **do**
6. from class C find k nearest misses $M_j(C)$;
7. **for** A := 1 **to** a **do**
8. $W[A] := W[A] - \sum_{j=1}^{k} diff(A, R_i, H_j)/m.k +$
9. $\sum_{C \neq class(R_i)} [\frac{P(C)}{1-P(class(R_i))} \sum_{j=1}^{k} diff(A, R_i, M_j(C))]/(m.k.)$;

end;

Figure 3.5. Pseudo code for Relief-F Method (Robnik-Sikonja & Kononenko, 2003)

By incorporating both nearest neighbors from the same class and those from different classes, Relief-F devises a robust method for computing attribute weights. This approach enhances its adaptability across diverse datasets and class distributions in feature selection endeavors. It allows Relief-F to discern the relevance of attributes more effectively, catering to varied data structures and class complexities commonly encountered in feature selection tasks.

3.2.2. mRMR

The mRMR aims to identify features that are strongly associated with the class label while minimizing redundancy between selected features. It offers two main approaches. The first aim is to identify features that are highly correlated with the class label while simultaneously reducing

unnecessary overlap or redundancy between them (Acid et al., 2011). The formula for calculating redundancy between variables is as follows:

$$W_I(S) = \frac{1}{|S|^2} \sum_{X_i, X_j \in S} MI(X_i, X_j) \tag{3.11.}$$

where $MI(X_i, X_j)$ provides the measure of mutual information between variables X_i and X_j.

Another objective is to maximize relevance among features while minimizing redundancy. The overall relevance of variables within set S concerning C can be computed as:

$$V_I(S) = \frac{1}{|S|} \sum_{X_i \in S} MI(C, X_i) \tag{3.12.}$$

One straightforward approach to obtaining a quality subset of features by considering both redundancy and relevance can be formulated as follows:

$$S^* argmax_{S \subseteq U} (V_I(S) - W_I(S)) \tag{3.13.}$$

Since conducting an exhaustive search for the optimal subset is not feasible, the chosen subset is acquired incrementally. It begins with the feature exhibiting the highest value of $MI(C; X_i)$ ($S_0 = \{X_{i_0}\}$), then gradually expands by incorporating the feature that maximizes (Acid et al., 2011).

$$\left(\max_{X_j \in U \setminus S_{m-1}} \left(MI(C, X_j) - \frac{1}{m-1} \sum_{X_i \in S_{m-1}} MI(X_j, X_i) \right) \right) \tag{3.14.}$$

4. DATASET AND METHODOLOGY

This chapter is divided into three parts. The first part covers the strategies for creating the dataset and the details of the PE file headers. The second part examines the model-building processes, the default values of classifiers, and the methods developed for hyperparameter optimization. The final section encompasses the intervals defined and parameter specifications to attain the optimal results demonstrated by classifiers in the models.

4.1. Dataset Generation

This section provides a thorough examination of the structure of Windows PE files, encompassing detailed discussions on the DOS, File, and Optional headers, as well as elaborate descriptions of the partition segments inherent to PE files. Furthermore, it delves into the methodology behind dataset creation, elucidating the specific headings and statistical information to be incorporated into the dataset with meticulous attention to detail.

4.1.1. Portable Executable File Headers

The subheadings within the PE file headers outline the characteristic behavior of a PE file when loaded or anticipated to be loaded on the Windows operating system. These subheadings cover various details, such as the file's structure, loading mechanism, designated installation location, file type, executable status, compatibility with target operating systems, and version information. These details are crucial for determining how the file will be processed, executed, and integrated within the Windows environment.

In exploring the nuances of the PE file format, the DOS Header, File Header, and Optional Header emerge as pivotal constituents. Each of these elements encapsulates indispensable data crucial for the precise comprehension and execution of PE files within the Windows ecosystem. Their collective significance lies in furnishing fundamental information imperative for the accurate interpretation and seamless operation of executable files within the Windows environment.

The DOS Header is located at the beginning of the PE file for compatibility purposes. Although this header holds significant importance within the context of MS-DOS, serving as a default header standard for the PE file, it assumes a less active role in modern operating systems like Windows NT 3.1 and subsequent versions. Nonetheless, its relevance endures due to its inclusion of numerous features related to compatibility, security, and debugging. Notably, upon execution on a DOS system, an alert message is displayed, indicating the incompatibility of the PE file with the DOS environment, following which the program terminates. A comprehensive overview of the subheadings within the DOS section, along with their detailed names and definitions, is presented in Table 4.1.

Table 4.1. List of DOS Headers

Headers	Header Name	Description
e_magic	Magic Number	Includes MZ signature to run on DOS system.
e_cblp	Bytes on Last Page	Specifies the number of missing bytes in the last page of the file. It is usually filled with the value 0.
e_cp	Pages in File	Specifies the total number of pages (i.e., size) of the file.
e_crlc	Relocations	Contains the number of relocation records.
e_cparhdr	Size of Header in Pharagraphs	Expresses the total size of the DOS Header in paragraphs. It is usually set to 4 paragraphs (i.e., 64 bytes).
e_minalloc	Minimum Extra Paragraphs	It is the minimum number of additional paragraphs required by the program.
e_maxalloc	Maximum Extra Paragraphs	It is the maximum number of additional paragraphs required by the program.
e_ss	Initial Stack Segment	Specifies the stack segment that is initially loaded for the program.
e_sp	Initial Stack Pointer	Specifies the register of the pointer stack loaded when the program is started.
e_csum	Checksum	Includes checksum to ensure completeness.
e_ip	Initial Instruction Pointer	Specifies the execution pointer register of the command to be executed at started.
e_cs	Initial Code Segment	Indicates the code segment that is loaded at initially.
e_lfarlc	File Address of Relocation Table	Specifies the address of the relocation records in the file. It is not related to DOS Header. It is usually set to 0.
e_ovno	Overlay Number	Used for DOS compatibility. It is usually set to 0.
e_res	Reserved Words	It is a reserved field.
e_oemid	OEM Identifier	It is an OEM identifier used for DOS compatibility.
e_oeminfo	OEM Information	Contains additional information used for DOS compatibility.
e_lfanew	Extended Linear Address of New Exectable	It is a value at the end of the DOS Header that indicates where COFF headers begin. This value represents a 32-bits offset or Relative Virtual Address (RVA) value.
e_res2	Reserved Words	It is a reserved field.

Abbreviations: OEM, Original Equipment Manufacturer; COFF, Common Object File Format; RVA, Relative Virtual Address

The File Header represents the standard format of the file contained in the PE file and executable by the Windows operating system. It defines the general characteristics of the file and provides information about the intended use of the file by the operating system. In the realm of standardized characteristics inherent to file formats, the file header assumes a pivotal role in dictating the manner in which the file undergoes processing. The headings and definitions included in the file header are shown in Table 4.2.

Table 4.2. List of File Headers

Headers	Description
Machine	It is a value that indicates the target machine architecture of the file.
Number of Sections	Specifies the number of sections in the file.
Time Date Stamp	Contains a timestamp that represents the compilation of file time.
Pointer To Symbol Table	Specifies the address in the file of the symbol table associated with the file.
Number Of Symbols	Specifies the total number of symbols in the file.
Size Of Optional Header	Specifies the optional header size.
Characteristics	It is a series of bytes that specify the properties of the file.

The optional header is a data structure contained in the PE file and represents the standard file format of the executable file by the Windows operating system. The optional header contains special properties of the file, its alignment in memory, resources to be consumed at runtime, and other important information. The headers and definitions included in the optional header are shown in Table 4.3 and Table 4.4.

Table 4.3. List of Optional Headers (Part 1)

Headers	Description
Magic	Specifies the x64 or x86 file type.
Major Linker Version	Identifies the major version number of the linker program.
Minor Linker Version	Identifies the minor version number of the linker program.
Size Of Code	Specifies the total size of the code section.
Size Of Uninitialized Data	Specifies the total size of the uninitialized data partition.
Size Of Initialized Data	Specifies the total size of the initialized data partition.
Address of Entry Point	Specifies the starting address of the code that triggers the program to start.
Base Of Code	Specifies the starting address of the code section.
Base Of Data	Specifies the starting address of the data section.
Image Base	Specifies the base address in memory when the file starts loading.
Section Alignment	Specifies the alignment value of sections in memory.
File Alignment	Specifies the alignment value of the file on disk.
Major Operating System Version	Specifies the maximum version of the operating system on which the file is running.
Minor Operating System Version	Specifies the minimum version of the operating system on which the file is running.
Major Image Version	Specifies the maximum version number of the file.
Minor Image Version	Specifies the minimum version number of the file.
Major Subsystem Version	Specifies the major version number under which the file is running.

Table 4.4. List of Optional Headers (Part 2)

Headers	Description
Minor Subsystem Version	Specifies the minor version number under which the file is running.
Size Of Image	Specifies the total size of the loaded file in memory.
Size Of Headers	Specifies the total size of file headers.
Checksum	Contains a value used to check the integrity of the file.
Subsystem	Specifies GUI or CUI subsystems.
DLL Characteristics	It contains a series of bytes that specify DLL properties.
Size Of Stack Reverse	Specifies the total size of the stack reserved in memory.
Size Of Stack Commit	Specifies the total initially approved size of the stack.
Size Of Heap Commit	Specifies the initially approved total size of the heap.
Size of Heap Reserve	Specifies the total size of the heap reserved in memory.
Loader Flags	Contains a value that specifies flags for the relevant loader.
Number Of RVA and Sizes	Specifies the number of elements of the Data Directory array.

Abbreviations: GUI, Graphical User Interface; CUI, Conversational User Interface; DLL, Dynamic Library

The Sections define the sections contained within the PE file and represent the standard format of a file executable by the Windows operating system. Sections contain sections and data types of files that are reserved for different purposes. Commonly used Section Headers are shown in Table 4.5.

Table 4.5. List of Common Sections

Section Name	Description
Text	Contains executable code.
Data	Contains data used at runtime.
Resource	Contains resources of files such as icon, text, sound, etc.
Import	Contains other libraries and modules on which the program depends.
Export	Contains functions that the program shares with other programs or modules.
Relocation	Contains the changes that must be made when the file is moved to a different memory address.
Debug	Contains debugging information.

4.1.2. Dataset Arrangement

The CLAMP[1] dataset encompasses the PE file header information executed on Windows operating systems. Benign files were sourced from program files of Windows XP and Windows 7, as well as from a freely available online software archive. Malicious files, on the other hand, were obtained from VirusShare[2], an online software repository, and each malicious sample was verified

[1] https://www.kaggle.com/datasets/saurabhshahane/classification-of-malwares (Accessed: 16 Jan 2024)
[2] https://virusshare.com (Accessed: 16 Jan 2024)

using the VirusTotal[3] API. The Pefile module from the Python programming language was utilized to preprocess files extracted from the Windows ecosystem for ML suitability (Kumar et al., 2019).

The Pefile[4] is a Python-based framework that facilitates the extraction of Windows PE file header information. The utilized dataset incorporates three primary categories of information, namely DOS, File, and Optional Headers. To enable the utilization of features in ML, duplicate data and labeling steps were implemented. A hashing method Message-Digest 5 (MD5) was employed to prevent the inclusion of the same PE file in the dataset. In addition to examining the file name, a hashing check of the file content was conducted for each PE file. Furthermore, after verifying whether each file was benign, labeling was applied to the target column. Following the necessary preprocessing steps, a total of 55 features were extracted, comprising 19 features from the DOS Header, 7 from the File Header, and the remaining 29 from the Optional Header. This resulted in a dataset containing 5184 samples. All extracted features were meticulously configured according to the Windows PE file format documentation[5], ensuring the creation of a unique, comprehensive dataset. Consequently, this extensive dataset was publicly released online on January 31, 2019 (Kumar et al., 2019).

This study applies data preprocessing and standardization to the features in the CLAMP dataset rather than utilizing them directly. In the data preprocessing phase, reserved fields in the PE files (i.e., e_res and e_res2) were removed. These fields, initially reserved by Windows for potential future functions, are currently unused by contemporary operating systems. Additionally, features containing fixed values such as the e_magic header with the constant value of 0x5A4D (decimal 23117), representing the DOS signature, and the e_crlc header used for printing the DOS file header were excluded from the dataset. Despite being designed for the DOS header, these fields are obsolete in modern operating systems and are retained solely for compatibility purposes. Furthermore, the target column distinguishing malicious activity with binary labels was removed, resulting in a dataset consisting of 51 features and 5136 samples (i.e., 2683 Malicious, and 2501 Benign) for use in this study. This refined dataset is tailored to focus on the relevant features while eliminating unnecessary elements, contributing to a more effective analysis of the characteristics of the dataset.

Separate models for Relief-F and mRMR were constructed, utilizing all columns directly from the dataset. The entire dataset comprises int64 (i.e., numerical) data types, having undergone relevant preprocessing steps. Information containing statistical insights pertaining to the dataset is presented in Table 4.6 and Table 4.7. These tables encapsulate statistical details, encompassing the dataset's overall characteristics such as distribution of data, measures of central tendency, and indicators of variability. These statistics provide important information about the structure of the dataset before building a model.

[3] https://www.virustotal.com/gui/home/upload (Accessed: 16 Jan 2024)
[4] https://github.com/erocarrera/pefile (Accessed: 16 Jan 2024)
[5] https://learn.microsoft.com/en-us/windows/win32/debug/pe-format (Accessed: 16 Jan 2024)

Table 4.6. Descriptive statistics of the dataset (Part 1)

Variables	Minimum	Maximum	Mean ± Standard Deviation
e_cblp	0	37008	145.96 ± 512.42
e_cp	0	144	2.93 ± 1.99
e_cparhdr	0	4	3.97 ± 0.33
e_minalloc	0	4096	1.85 ± 57
e_maxalloc	0	65535	65153.37 ± 4937.43
e_ss	0	65520	12.63 ± 909.99
e_sp	0	65534	195.78 ± 907.82
e_csum	0	2	0 ± 0.03
e_ip	0	26621	10.52 ± 523.04
e_cs	0	65520	23.46 ± 1015.08
e_lfarlc	0	65	63.61 ± 4.93
e_ovno	0	26	1.75 ± 6.51
e_oemid	0	267	0.15 ± 6.42
e_oeminfo	0	8	0 ± 0.17
e_lfanew	12	648	223.26 ± 48.17
Machine	332	34404	404.32 ± 1567.98
NumberOfSections	1	34	4.66 ± 1.92
CreationYear	1970	2102	2008.88 ± 6.08
PointerToSymbolTable	0	2846720	1016.91 ± 43558.58
NumberOfSymbols	0	6417	6.31 ± 151.74
SizeOfOptionalHeader	224	240	224.03 ± 0.73
Characteristics	34	51358	4592.72 ± 8195.50
Magic	0	267	266.43 ± 12.28
MajorLinkerVersion	0	255	7.58 ± 7.30
MinorLinkerVersion	0	255	5.38 ± 10.60
SizeOfCode	0	3590647212	873221.29 ± 49871981.03
SizeOfInititializedData	0	81166336	276423.35 ± 2101901.83
SizeOfUninititializedData	0	522125312	264429.16 ± 10292031.10
AddressOfEntryPoint	0	42403872	157961.15 ± 971199.97
BaseOfCode	0	42225664	46731.80 ± 853410.73
BaseOfData	0	42405888	226986.69 ± 1081682.73
ImageBase	0	2133655552	146582616.49 ± 353351425.32
SectionAlignment	0	8192	4260.88 ± 1010.24
FileAlignment	0	4096	1191.06 ± 1409.07
MajorOperatingSystemVersion	0	10	4.60 ± 10.01
MinorOperatingSystemVersion	0	23	0.42 ± 0.87
MajorImageVersion	0	40386	107.66 ± 1556.16

Table 4.7. Descriptive statistics of the dataset (Part 2)

Variables	Minimum	Maximum	Mean ± Standard Deviation
MinorImageVersion	0	51126	98.31 ± 1545.57
MajorSubsystemVersion	0	6	4.58 ± 0.76
MinorSubsystemVersion	0	20	0.35 ± 0.84
SizeOfImage	0	81252352	558428.23 ± 2455638.23
SizeOfHeaders	0	2314240	4060.57 ± 48937.75
Checksum	0	4294967295	2321815.45 ± 84493184.37
Subsystem	0	16	2.13 ± 0.55
DllCharacteristics	0	37184	9459.24 ± 14828.17
SizeOfStackReverse	0	33554432	1989460.42 ± 4608161.61
SizeOfStackCommit	0	2097152	10389.19 ± 74829.86
SizeOfHeapReserve	0	33554432	2089821.15 ± 4424085.46
SizeOfHeapCommit	0	131072	5749.73 ± 6847.20
LoaderFlags	0	93827511	42573.84 ± 1747203.07
NumberOfRvaAndSizes	0	16	15.96 ± 0.74
class	0	1	0.51 ± 0.50

4.2. Methodology

The steps involving dataset creation and preparation, along with the models developed using feature selectors, are illustrated in this section. Additionally, explanations are provided regarding the default values of the classifiers, the frameworks employed for adjusting hyperparameters, and the chosen evaluation criteria to assess the models.

4.2.1. Model Creation

Feature importance levels were assessed using Relief-F and mRMR-based algorithms in this study. These methods ranked the dataset's features from 1 to 51 based on their respective importance. To construct various models, the initial dataset's features were gradually pruned based on their importance levels, as determined by the feature selection algorithms. Starting with the first regular model encompassing all features, iterations involved removing features with lower importance according to the algorithms' rankings. This iterative process led to the creation and sorting of 51 distinct models.

The rankings obtained from both the Relief-F and mRMR algorithms resulted in a total of 102 distinct models. Considering both default and optimized scenarios for these models, a total of 204 separate processes were executed. Each model underwent a 10-fold cross-validation technique to assess its generalizability. The performance of these models, constructed using default values and hyperparameter adjustments for ML algorithms such as LightGBM, KNN, DT, SVM, MLP, and NB, was evaluated using metrics like accuracy, F1-Score, recall, precision. Moreover, within each method, a detailed assessment of the best-performing results was conducted using the confusion

matrix. This enabled a comprehensive evaluation of the outcomes showcasing optimal performance for each approach.

The ranking of individual variables was computed and arranged in a descending order using the Relief-F and mRMR algorithms, demonstrating their respective positions in Tables 4.8 and 4.9. Furthermore, Tables containing a comprehensive representation of features in all models generated through feature selection algorithms are in the appendix (Table A.1.1 through Table A.1.8 for Relief-F and Table A.2.1 through Table A.2.9 for mRMR).

Table 4.8. List of ranks of predictor variables assigned by mRMR and Relief-F (Part 1)

Rank	Predictor Variables Ranked by Relief-F	Predictor Variables Ranked by mRMR
1	DllCharacteristics	ImageBase
2	CreationYear	e_lfanew
3	Characteristics	MinorSubsystemVersion
4	FileAlignment	Subsystem
5	MajorSubsystemVersion	SizeOfStackReserve
6	MajorOperatingSystemVersion	MajorSubsystemVersion
7	ImageBase	MinorOperatingSystemVersion
8	e_lfanew	SizeOfHeapCommit
9	SizeOfStackReserve	SectionAlignment
10	SizeOfHeapReserve	e_ovno
11	NumberOfSections	SizeOfHeapReserve
12	SectionAlignment	FileAlignment
13	Subsystem	MajorOperatingSystemVersion
14	MinorLinkerVersion	SizeOfInitializedData
15	SizeOfHeapCommit	e_cparhdr
16	MinorSubsystemVersion	NumberOfSections
17	MinorOperatingSystemVersion	CreationYear
18	e_cparhdr	DllCharacteristics
19	SizeOfImage	MinorLinkerVersion
20	e_ovno	BaseOfData
21	e_maxalloc	MajorLinkerVersion
22	e_lfarlc	Characteristics
23	SizeOfInitializedData	e_lfarlc
24	MajorLinkerVersion	SizeOfHeaders
25	BaseOfData	SizeOfImage
26	NumberOfRvaAndSizes	e_maxalloc
27	AddressOfEntryPoint	SizeOfUninitializedData
28	Magic	Machine
29	SizeOfOptionalHeader	e_minalloc
30	Machine	SizeOfStackCommit
31	SizeOfStackCommit	PointerToSymbolTable
32	SizeOfHeaders	SizeOfCode
33	MajorImageVersion	LoaderFlags

Table 4.9. List of ranks of predictor variables assigned by mRMR and Relief-F (Part 2)

Rank	Predictor Variables Ranked by Relief-F	Predictor Variables Ranked by mRMR
34	CheckSum	Magic
35	MinorImageVersion	BaseOfCode
36	BaseOfCode	SizeOfOptionalHeader
37	e_cp	e_oemid
38	SizeOfUninitializedData	NumberOfRvaAndSizes
39	e_oemid	NumberOfSymbols
40	e_cblp	AddressOfEntryPoint
41	e_oeminfo	CheckSum
42	NumberOfSymbols	e_cp
43	SizeOfCode	e_oeminfo
44	e_minalloc	e_cs
45	e_csum	e_csum
46	e_sp	MajorImageVersion
47	e_cs	e_ip
48	PointerToSymbolTable	e_ss
49	e_ip	e_sp
50	LoaderFlags	e_cblp
51	e_ss	MinorImageVersion

4.2.2. Default and Optimized Model Configuration

All models developed transform the standard scaling before being applied to the entire dataset and were analyzed using six distinct ML methods. This section introduces two approaches, (a) default models and (b) hyperparameter-tuned (i.e., optimized) models.

The default models section, default parameter values were extracted from the Python Sci-kit Learn module (Pedregosa et al., 2011) and employed for each of the five different ML classifiers, while the LightGBM algorithm utilized the API provided by Microsoft (jameslamb, 2024). This approach centered on assessing the fundamental inputs of each model by adhering to the standard API structure of the algorithms. The aim was to analyze the performances and fundamental capabilities inherent in the default configurations of these models.

In the hyperparameter tuning section, the essential parameters of the ML algorithms were adjusted to achieve maximum efficiency for each model. The Optuna was utilized to identify the optimal hyperparameters for these models. It is an open-source hyperparameter optimization framework, seamlessly integrates with Python, facilitating the process. It operates by executing a specified number of trials for each parameter setting within a designated trial space until reaching the specified trial count. Moreover, the Optuna leverages Bayes' Theorem to avoid making parameter adjustments that may probabilistically result in poor performance. This approach ensures enhanced performance while minimizing costs (Akiba et al., 2019).

In the appendix, Table A.3.1 through Table A.4.14 present the adjusted parameters and optimal parameter values for the ML algorithms utilized in the models developed using Relief-F and mRMR, respectively. Additionally, the best parameter values for each model were derived from 10-fold cross-validation. The optimal result for each model comprises 800 trials, while the total output for a method encompasses 40.800 trials.

4.2.3. Evaluation Metrics

The evaluation of prediction errors was carried out using accuracy, precision, recall, and F1-Score, as shown in Eqs. 4.1, 4.2, 4.3, and 4.4, respectively.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (4.1.)$$

$$Precision = \frac{TP}{TP+FP} \quad (4.2.)$$

$$Recall = \frac{TP}{TP+FN} \quad (4.3.)$$

$$F1 - \text{Score} = 2 \times \frac{PrecisionxRecall}{Precision+Recall} \quad (4.4.)$$

TP, TN, FP, and FN refer to true positive, true negative, false positive, and false negative, respectively. Furthermore, to assess generalization errors across all models, a 10-fold cross-validation was employed, and the errors were elaborately depicted using confusion matrices.

4.3. Details of Prediction Models

In this section, the default parameter values of six different ML classifiers, the ranges determined for hyperparameter tuning, and the parameter definitions of each classifier are presented in detail.

4.3.1. LightGBM-Based Models

The default API parameters and hyperparameter settings for the LightGBM algorithm were adjusted using different methods and specified ranges. The defined ranges and default parameter values for determining the optimum model performance are presented in Table 4.10.

Table 4.10. Defined parameter ranges and default values for achieving optimal model performance using LightGBM

Parameters	Values	Range
Learning Rate	0.1	[0.01, 1]
Num Leaves	31	[5, 500]
Max Depth	-1	[3, 50]
Minimum Child Samples	20	[1, 100]
Subsample	1	[0.1, 1]
Colsample Bytree	1	[0.1,1]
Alpha	0	[0,1]
Lambda	0	[0,1]
Class Weight	None	[False, True]

The learning rate influences training the model at each iteration. Lower values offer more precise learning but at a higher computational cost. Conversely, higher learning rates reduce costs and may enhance generalization but also elevate the risk of overfitting. The steps were calculated with increments of 0.1. The num leaves determines the number of leaves per tree, allowing for model flexibility. However, a high number of leaves escalates the overfitting risk. This study examined certain intervals in one-step increments. The max depth parameter sets the maximum depth of each tree. The subsample controls the data sampling rate in each iteration. Lower values heighten overfitting risks, while higher values provide more stable outputs. Ranges are calculated in increments of 0.5. The colsample bytree adjusts column sampling rates during tree construction, aiding in mitigating overfitting then, ranges were scanned in steps of 0.1. The alpha and lambda regulate L1 and L2 regularization (i.e., Lasso and Ridge, respectively) and mitigate overfitting risks. Ranges are calculated in increments of 0.1. The class weight is tailored to address imbalanced class distributions by weighting sampling on minority or majority class samples. This aspect was categorically set in this study.

The hyperparameter-tuning process was iteratively computed for each model. During each iteration, less important features were removed based on the prioritization determined by the feature selector, and optimal values were sought. Therefore, the parameters selected for each model were meticulously analyzed for their effectiveness.

4.3.2. DT-Based Models

Different parameter inputs and their specified ranges are outlined, encompassing both the default API parameters of the DT algorithm and hyperparameter tuning. Table 4.11 presents information regarding defined ranges and default parameter values aimed at determining optimal model performance.

Table 4.11. Defined parameter ranges and default values for achieving optimal model performance using DT

Parameters	Values	Range
Criterion	gini	[gini, entropy, log_loss]
Splitter	best	[best, random]
Max Depth	None	[None, (2, 100)]
Minimum Samples Split	2	[6, 150]
Minimum Samples Leaf	1	[1, 150]
Max Features	None	[None, sqrt, log2]
Class Weight	None	[None, balanced]

The criterion sets the approach used to split the nodes of a decision tree. The ranges specified in this study were determined and examined categorically. The splitter determines the strategy for splitting the nodes of a decision tree. The defined intervals were calculated categorically. The max depth specifies the maximum depth of the tree. As the tree depth increases, the risk of overfitting also increases. The ranges specified in this study were examined categorically. Minimum samples split denotes the minimum number of samples necessary to divide a node within a decision tree, whereas minimum sample leaf represents the minimum number of samples required to constitute a leaf within the tree. These parameters play a crucial role in mitigating overfitting, thereby facilitating enhanced generalization of the model. Furthermore, fine-tuning these parameters directly influences performance of the model. These predefined ranges are determined with incremental steps of one. The max feature determines the maximum number of features in each division of the decision tree. Thus, it limits the maximum number of features to be evaluated at each splitting step. The ranges specified in this study were scanned categorically. The class weight is adjusted to solve the imbalance problem in the dataset. The parameter gives more importance to rare classes, making these classes much more effective on the model. The ranges specified in this study were examined categorically.

4.3.3. KNN-Based Models

The KNN algorithm was initially trained on the models using default parameters, and each model was independently evaluated using assessment metrics. Hyperparameter optimization was employed to enhance performance of the models, determining the best parameter values. Subsequently, the models were retrained and evaluated based on these parameters.

The specified ranges and default parameter values for determining the models' optimal performance are presented in Table 4.12.

Table 4.12. Defined parameter ranges and default values for achieving optimal model performance using KNN

Parameters	Values	Range
n_neighbors	5	[1, 70]
Weights	uniform	[uniform, distance]
Algorithm	auto	[auto, ball_tree, kd_tree, brute]
P	2	[1, 10]
Leaf Size	30	if Algorithm is either ball_tree or kd_tree, then range set to [10,100], else [10,70]
Metric	Minkowski	if P is lower and equal to 2, then range set to [Euclidean, Manhattan], else [Minkowski]

The n_neighbors parameter denotes the quantity of neighbors considered for classification a crucial factor influencing the model in this context. Select for a higher neighbor count typically fosters a more generalized model, whereas a reduction in neighbors complicates the model. Throughout this study, the intervals were methodically explored in increments of one. The weights govern the influence of neighboring points on the classification process. It contains the formula applied to adjust the weights of neighbors. Categorical selections were made to determine the applicable ranges in this study. Parameters like Algorithm and Leaf Size exhibit interdependence. The algorithm dictates the fundamental approach to neighbor detection, profoundly impacting the speed of this process. Categorical selections were made to delineate the ranges pertinent to this parameter. Meanwhile, the leaf size governs the leaf dimensions within the tree structure. Notably, when the algorithm embraced a tree-based methodology (i.e., ball_tree or kd_tree), a wider leaf size range was employed to enhance the efficiency of these methods. Consequently, in this study, wider scanning (i.e., 10 to 100) was conducted for tree-based leaf size values, while narrower exploration (i.e., 10 to 70) occurred in other methods, with a step size of one. Parameters such as the P and the metric define the distance and proximity criteria, though the efficacy of the P materializes when metric assumes a defined state. Within this study, the P value was standardized as uniform, while the metric parameter was established as categorical. While searching for the best parameter value among the ranges specified in the P value, values with equal probabilities are randomly selected. Notably, when P exceeded two, the metric defaulted to Minkowski. Conversely, when P assumed a value of two or less, Euclidean and Manhattan metrics were categorically selected.

4.3.4. MLP-Based Models

The default API parameters and hyperparameter settings of the MLP algorithm underwent tuning using diverse methods and specific ranges. The defined ranges and default parameter values for determining the optimum model performance are presented in Table 4.13.

Table 4.13. Defined parameter ranges and default values for achieving optimal model performance using MLP

Parameters	Values	Range
Hidden Layer Sizes	(100,)	[1, 100]
Activation	relu	[logistic, tanh, relu]
Alpha	0.0001	[1e-4, 1e1]
Solver	adam	[adam, sgd, lbfgs]
Learning Rate	constant	[constant, invscaling, adaptive]
Batch Size	auto	[auto, (10, 256)]
Momentum	0.9	[0,1]

The hidden layer sizes parameter determines the quantity of hidden layers and neurons within each layer in the MLP algorithm. Steps of one scanned the specified ranges in this study incrementally. The activation functions determine the output of each neuron, directly influencing the generalization of model ability. The specified ranges in this study were scanned categorically. The alpha parameter contributes to L2 regularization, curbing overfitting tendencies. In this study, the specified ranges were considered on a logarithmic scale, allowing testing across a wider range of values to better illustrate the impact of hyperparameter. The solver parameter is instrumental in training artificial neural network models. It drives the minimization of loss functions and weight updates. Ranges specified in this study were scanned categorically.

The learning rate parameter determines the amount of updating of the models throughout the training of the model in the artificial neural network. If the learning rate parameter is selected as a high value, it will miss the learning details while providing savings in cost. However, a low learning rate parameter provides more reliable learning while losing cost. The ranges specified in this study were scanned categorically. Batch size specifies the number of samples processed during model training. It allows dividing the dataset into smaller groups, optimizing processing. In this study, solver parameters adam and sgd were scanned within specified ranges with a step of one, as they update model weights using gradient descent, processing one batch per step. Otherwise, default algorithm values were utilized. The momentum accelerates weight updates and stabilizes the process. When the solver parameter is sgd, specified ranges were scanned with a step of one, indicating combining the previous step with the gradient value during descent. Otherwise, default algorithm values were employed.

4.3.5. SVM-Based Models

The default API parameters and hyperparameter settings of the SVM algorithm were tuned using various methods and specific ranges. The defined ranges and default parameter values for determining the optimum model performance are presented in Table 4.14.

Table 4.14. Defined parameter ranges and default values for achieving optimal model performance using SVM

Parameters	Values	Range
C	1	[1, 100]
Kernel	rbf	[linear, poly, rbf, sigmoid]
Gamma	scale	if kernel is rbf, then range set to [1,100], else [scale]
Degree	3	if kernel is poly, then range set to [1, 10], else [3]
Class Weight	None	[None, balanced]

The C serves as both the regularization and cost parameter within the SVM algorithm. It requires more computational effort to precisely classify the dataset. Thus, it endeavors to accurately classify each data point within the models. However, this pursuit increases the likelihood of overfitting, despite its attempt to fit the data. The determined ranges in this study were scanned step one. The kernel function changes the scale of data points and serves as a mathematical function used to separate classes. Assume that x_i and y_i represent the data points and $K(x_i, y_i)$ represent the kernel function. Furthermore, if we assume that the φ function moves the data points x_i and y_i into another space, the kernel function is represented as:

$$K(x_i, y_i) = \varphi(x_i) \,.\, \varphi(y_i) \tag{4.6.}$$

The SVM establishes a decision boundary between the data, intending to distinguish between classes. However, this demarcation might not always be linear. In cases where classes are not linearly separable, the kernel function evaluates the data points in an alternate space, facilitating their segregation. The ranges established in this study were scanned categorically. The gamma effectively relocates data points into an infinite-dimensional space, intending to classify them in a circular manner. As the gamma value escalates, it enhances the classification granularity, but concurrently heightens the risk of overfitting. In this study, the kernel parameter rbf was systematically scanned in increments of one to identify the optimal gamma value. Alternatively, the default gamma value of the SVM algorithm was employed. The degree parameter is specifically employed for the polynomial kernel, enabling a polynomial-based classification of data in a distinct dimensional space. This parameter dictates the degree of the polynomial function, thereby providing the model with the capability for intricate separations. In this study, when the kernel was set as poly, it was incrementally scanned at intervals of one. Otherwise, the default values of the SVM algorithm were directly utilized. The class weight parameter serves as an alternative in handling unbalanced datasets within the SVM algorithm. Its purpose is to address the issue of imbalance by assigning greater significance

to the minority class. In this study, the specified ranges were systematically scanned to categorically address this imbalance.

4.3.6. NB-Based Models

The default API parameters and hyperparameter settings of the NB algorithm were adjusted using various methods and specific ranges. The defined ranges and default parameter values for determining the optimum model performance are presented in Table 4.15.

Table 4.15. Defined parameter ranges and default values for achieving optimal model performance using NB

Parameters	Values	Range
var_smoothing	1e-09	[1e-5, 1e5]

The Naive Bayes classifier, known for its simplicity and reliance on Bayes' Theorem, underwent hyperparameter optimization specifically targeting the var_smoothing parameter. This parameter acts as a regularization term when variances reach zero, effectively augmenting the variance of each feature within each class by a proportion of the maximum feature variance. The study employed logarithmic range scanning to ascertain the most effective parameter values.

5. RESULTS AND DISCUSSION

In this chapter, the default and optimized ML performances of models generated using Relief-F and mRMR-based feature selectors have been assessed in terms of evaluation criteria such as accuracy, F1-Score, recall, and precision. Particularly, the classification success of the methods has been demonstrated through confusion matrices and percentage increase rates in results.

5.1. Results of Models with Default Parameters

The accuracy, F1-Score, recall and precision results of algorithms created with their default configurations are presented. Confusion matrices are also provided to evaluate the classification of the best model for each method.

5.1.1. Results of Relief-F-Based Models

After applying the models created based on Relief-F with the default parameter values of the algorithms, this section presents the evaluation results, including accuracy, F1-Score, recall, precision, and confusion matrix criteria. The results obtained comprehensively present the detailed outputs of the algorithms and models and specifically highlight the best results among all models.

Table 5.1. Accuracy values for Relief-F-based predictor variables (Part 1)

Model	Accuracy (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 1	98.30	96.61	93.75	86.11	74.15	57.99
Model 2	98.30	96.53	93.75	88.72	74.15	57.99
Model 3	98.30	96.66	93.75	87.81	74.15	57.97
Model 4	98.30	96.62	93.75	**89.70**	74.15	57.97
Model 5	98.30	96.47	93.75	83.20	74.15	57.99
Model 6	98.30	96.53	93.75	84.72	74.15	57.99
Model 7	98.30	96.64	93.75	86.48	74.15	57.99
Model 8	98.30	96.59	93.75	79.92	74.15	57.99
Model 9	98.30	96.80	93.75	87.52	74.15	57.99
Model 10	98.36	96.68	93.71	88.68	74.17	58.08
Model 11	98.36	96.86	93.71	85.15	74.17	58.08
Model 12	98.36	96.68	93.71	81.40	74.17	58.08
Model 13	**98.48**	96.64	93.71	86.57	74.17	58.08
Model 14	**98.48**	96.66	93.71	85.47	74.17	58.08
Model 15	98.38	96.84	93.71	86.93	74.17	56.69
Model 16	98.46	**96.88**	93.71	87.02	74.17	56.69
Model 17	98.42	96.86	93.71	86.46	74.17	55.81
Model 18	98.38	96.60	93.71	88.16	74.17	55.81
Model 19	98.17	96.59	92.69	81.18	74.00	56.85

Table 5.2. Accuracy values for Relief-F-based predictor variables (Part 2)

Model	Accuracy (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 20	97.96	96.41	92.69	84.90	74.00	56.85
Model 21	98.03	96.33	92.69	82.23	74.00	67.58
Model 22	97.88	96.35	92.69	73.84	74.00	56.48
Model 23	97.88	96.30	92.69	77.20	74.00	56.48
Model 24	97.88	96.33	92.69	86.21	74.00	56.48
Model 25	97.88	96.20	92.69	81.82	74.00	56.48
Model 26	97.97	96.64	91.84	80.02	74.00	56.58
Model 27	97.97	96.43	91.84	83.29	74.00	56.58
Model 28	97.67	96.37	91.65	82.06	74.00	57.35
Model 29	97.43	96.30	91.65	83.39	74.00	57.35
Model 30	97.36	96.06	90.78	80.40	74.00	56.12
Model 31	97.36	96.12	90.78	77.82	74.00	56.12
Model 32	97.36	96.18	90.63	83.07	74.00	55.61
Model 33	97.43	96.08	90.63	79.11	74.00	55.61
Model 34	97.34	96.57	**94.95**	81.61	74.00	55.61
Model 35	97.36	96.51	94.93	73.55	74.00	55.61
Model 36	97.36	96.57	94.89	79.69	74.00	55.61
Model 37	97.40	96.59	93.60	79.74	74.00	55.61
Model 38	97.51	96.53	93.58	83.64	74.00	55.54
Model 39	97.55	96.61	93.36	81.28	74.00	55.54
Model 40	97.36	96.06	93.21	77.49	74.00	55.54
Model 41	97.22	96.16	93.17	81.50	74.00	55.54
Model 42	97.18	96.35	92.71	82.06	74.00	55.54
Model 43	97.16	96.37	93.04	83.70	74.00	**81.58**
Model 44	96.87	96.24	92.67	55.98	74.00	73.96
Model 45	95.91	95.74	85.26	50.95	74.00	73.96
Model 46	95.85	95.33	85.13	77.55	**84.82**	73.55
Model 47	95.37	94.77	83.93	81.29	**84.82**	72.47
Model 48	94.97	94.50	83.20	77.16	**84.82**	71.30
Model 49	94.58	94.62	72.70	81.00	**84.82**	59.05
Model 50	85.63	85.57	57.02	57.90	62.29	59.22
Model 51	81.42	81.37	48.88	67.80	62.29	59.47

Table 5.3. F1-Score values for Relief-F-based predictor variables (Part 1)

Model	F1-Score (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 1	98.37	96.90	93.99	86.61	79.38	32.97
Model 2	98.37	96.76	93.99	87.76	79.38	32.97
Model 3	98.37	96.74	93.99	87.70	79.38	33.32
Model 4	98.37	96.62	93.99	84.88	79.38	33.32
Model 5	98.37	96.87	93.99	85.08	79.38	33.21

Table 5.4. F1-Score values for Relief-F-based predictor variables (Part 2)

Model	F1-Score (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 6	98.37	96.66	93.99	86.06	79.38	33.21
Model 7	98.37	96.79	93.99	88.04	79.38	33.21
Model 8	98.37	96.85	93.99	**88.43**	79.38	33.21
Model 9	98.37	96.60	93.99	86.11	79.38	33.21
Model 10	98.42	96.88	93.96	83.36	79.40	33.59
Model 11	98.42	96.84	93.96	87.18	79.40	33.59
Model 12	98.42	96.87	93.96	86.74	79.40	33.59
Model 13	**98.53**	96.80	93.96	88.38	79.40	33.59
Model 14	**98.53**	96.79	93.96	87.26	79.40	33.59
Model 15	98.44	96.98	93.96	85.61	79.40	31.31
Model 16	98.51	97.02	93.96	85.89	79.40	31.31
Model 17	98.48	**97.04**	93.96	78.34	79.40	29.76
Model 18	98.44	96.96	93.96	84.38	79.40	29.76
Model 19	98.23	96.64	93.05	73.39	79.29	32.20
Model 20	98.03	96.48	93.05	74.72	79.29	32.20
Model 21	98.10	96.45	93.05	85.96	79.29	56.30
Model 22	97.96	96.50	93.05	86.29	79.29	31.27
Model 23	97.96	96.49	93.05	87.04	79.29	31.27
Model 24	97.96	96.54	93.05	73.30	79.29	31.27
Model 25	97.96	96.29	93.05	82.85	79.29	31.27
Model 26	98.05	96.63	92.24	88.16	79.29	31.12
Model 27	98.05	96.75	92.24	77.24	79.29	31.12
Model 28	97.75	96.47	92.09	80.16	79.29	32.96
Model 29	97.53	96.37	92.09	63.28	79.29	32.96
Model 30	97.46	96.35	91.34	77.79	79.29	29.55
Model 31	97.46	96.39	91.34	76.98	79.29	29.55
Model 32	97.46	96.29	91.20	82.60	79.29	28.17
Model 33	97.53	96.29	91.20	81.87	79.29	28.17
Model 34	97.44	96.72	**95.15**	77.19	79.29	28.26
Model 35	97.45	96.76	95.14	85.03	79.29	28.26
Model 36	97.45	96.83	95.10	84.48	79.29	28.26
Model 37	97.49	96.64	93.73	71.54	79.29	28.26
Model 38	97.60	96.63	93.71	77.81	79.29	28.05
Model 39	97.64	96.64	93.50	81.78	79.29	28.05
Model 40	97.45	96.26	93.34	73.07	79.29	28.05
Model 41	97.33	96.30	93.30	82.36	79.29	28.05
Model 42	97.29	96.42	92.79	82.61	79.29	28.05
Model 43	97.27	96.42	93.15	73.15	79.29	**81.24**
Model 44	97.00	96.36	92.75	30.91	79.29	79.29
Model 45	96.10	95.95	83.77	31.91	79.29	79.29
Model 46	96.05	95.57	83.68	83.86	**84.25**	72.28
Model 47	95.60	95.05	82.23	73.74	**84.25**	72.09

Table 5.5. F1-Score values for Relief-F-based predictor variables (Part 3)

Model	F1-Score (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 48	95.22	94.85	81.26	72.27	**84.25**	71.23
Model 49	94.87	94.90	65.12	77.77	**84.25**	42.52
Model 50	87.21	87.15	29.51	51.35	66.64	67.14
Model 51	84.33	84.26	2.55	71.71	66.64	67.24

Table 5.6. Recall values for Relief-F-based predictor variables (Part 1)

Model	Recall (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 1	98.66	96.65	94.37	83.16	96.01	19.98
Model 2	98.66	96.72	94.37	88.48	96.01	19.98
Model 3	98.66	96.79	94.37	87.60	96.01	20.31
Model 4	98.66	96.94	94.37	86.32	96.01	20.31
Model 5	98.66	96.65	94.37	78.97	96.01	20.20
Model 6	98.66	96.64	94.37	**94.15**	96.01	20.20
Model 7	98.66	96.76	94.37	86.43	96.01	20.20
Model 8	98.66	96.68	94.37	84.26	96.01	20.20
Model 9	98.62	96.64	94.37	86.66	96.01	20.20
Model 10	98.66	97.02	94.45	93.85	96.05	20.50
Model 11	98.66	97.09	94.45	84.27	96.05	20.50
Model 12	98.66	96.98	94.45	87.32	96.05	20.50
Model 13	**98.81**	96.79	94.45	84.66	96.05	20.50
Model 14	**98.81**	97.09	94.45	84.22	96.05	20.50
Model 15	98.70	**97.17**	94.48	83.93	96.05	19.09
Model 16	98.77	97.02	94.48	87.99	96.05	19.09
Model 17	98.73	96.94	94.48	80.16	96.05	18.12
Model 18	**98.81**	97.13	94.48	77.44	96.05	18.12
Model 19	98.47	96.87	94.60	77.75	**96.09**	19.83
Model 20	98.40	96.50	94.60	88.85	**96.09**	19.83
Model 21	98.40	96.46	94.56	82.78	**96.09**	47.08
Model 22	98.17	96.46	94.60	85.98	**96.09**	19.16
Model 23	98.17	96.38	94.60	85.27	**96.09**	19.16
Model 24	98.17	96.24	94.60	82.48	**96.09**	19.16
Model 25	98.17	96.42	94.60	91.85	**96.09**	19.16
Model 26	98.43	96.79	93.74	68.34	**96.09**	19.01
Model 27	98.43	96.94	93.74	79.98	**96.09**	19.01
Model 28	97.99	96.31	93.93	76.85	**96.09**	20.35
Model 29	97.91	96.42	93.93	78.87	**96.09**	20.35
Model 30	97.84	96.20	93.89	83.85	**96.09**	17.82
Model 31	97.84	96.38	93.89	80.19	**96.09**	17.82
Model 32	97.84	96.50	93.78	84.06	**96.09**	16.85
Model 33	97.95	96.57	93.78	78.30	**96.09**	16.85

Table 5.7. Recall values for Relief-F-based predictor variables (Part 2)

Model	Recall (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 34	97.76	96.68	95.82	68.33	**96.09**	16.92
Model 35	97.80	96.57	**95.86**	84.76	**96.09**	16.92
Model 36	97.80	96.61	95.79	91.59	**96.09**	16.92
Model 37	97.87	96.72	92.62	81.91	**96.09**	16.92
Model 38	97.91	96.87	92.58	92.77	**96.09**	16.77
Model 39	98.06	96.68	92.39	64.90	**96.09**	16.77
Model 40	97.65	96.20	91.99	83.23	**96.09**	16.77
Model 41	97.73	96.12	92.02	88.89	**96.09**	16.77
Model 42	97.73	96.20	90.72	86.23	**96.09**	16.77
Model 43	97.80	96.50	91.58	84.18	**96.09**	77.15
Model 44	97.50	96.38	90.68	15.17	**96.09**	**96.24**
Model 45	97.43	96.83	74.09	15.21	**96.09**	**96.24**
Model 46	97.35	96.23	74.28	88.44	78.57	66.83
Model 47	97.05	95.82	72.30	82.78	78.57	68.69
Model 48	96.94	95.64	70.89	85.16	78.57	68.47
Model 49	96.68	96.24	49.95	65.52	78.57	30.11
Model 50	94.63	94.52	17.48	61.09	72.75	80.47
Model 51	96.50	96.35	1.34	71.93	72.75	80.36

Table 5.8. Precision values for Relief-F-based predictor variables (Part 1)

Model	Precision (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 1	98.08	96.65	93.62	91.82	67.68	**94.56**
Model 2	98.08	96.69	93.62	85.21	67.68	**94.56**
Model 3	98.08	96.87	93.62	85.61	67.68	93.21
Model 4	98.08	96.80	93.62	88.53	67.68	93.21
Model 5	98.08	96.80	93.62	87.34	67.68	93.75
Model 6	98.08	96.73	93.62	**92.77**	67.68	93.75
Model 7	98.08	96.69	93.62	85.26	67.68	93.75
Model 8	98.08	96.98	93.62	89.01	67.68	93.75
Model 9	98.12	96.58	93.62	84.57	67.68	93.75
Model 10	98.19	96.71	93.48	91.65	67.69	93.30
Model 11	98.19	96.78	93.48	85.80	67.69	93.30
Model 12	98.19	96.84	93.48	85.35	67.69	93.30
Model 13	**98.26**	96.78	93.48	87.68	67.69	93.30
Model 14	**98.26**	96.73	93.48	89.29	67.69	93.30
Model 15	98.19	96.81	93.45	85.46	67.69	87.47
Model 16	**98.26**	96.88	93.45	80.19	67.69	87.47
Model 17	98.23	96.92	93.45	83.81	67.69	83.73
Model 18	98.08	**97.02**	93.45	88.08	67.69	83.73

Table 5.9. Precision values for Relief-F-based predictor variables (Part 2)

Model	Precision (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 19	98.00	96.59	91.57	85.39	67.52	85.93
Model 20	97.67	96.48	91.57	84.88	67.52	85.93
Model 21	97.82	96.50	91.60	85.61	67.52	84.77
Model 22	97.74	96.57	91.57	85.26	67.52	85.36
Model 23	97.74	96.59	91.57	78.12	67.52	85.36
Model 24	97.74	96.36	91.57	83.61	67.52	85.36
Model 25	97.74	96.55	91.57	85.39	67.52	85.36
Model 26	97.67	96.54	90.81	84.40	67.52	86.54
Model 27	97.67	96.27	90.81	84.29	67.52	86.54
Model 28	97.52	96.61	90.35	85.52	67.52	87.92
Model 29	97.17	96.39	90.35	89.06	67.52	87.92
Model 30	97.08	96.28	88.94	87.34	67.52	87.19
Model 31	97.08	96.03	88.94	82.53	67.52	87.19
Model 32	97.08	96.25	88.77	80.32	67.52	86.46
Model 33	97.12	95.96	88.77	84.89	67.52	86.46
Model 34	97.12	96.69	94.50	85.91	67.52	86.21
Model 35	97.12	96.82	94.43	79.03	67.52	86.21
Model 36	97.12	96.62	94.43	81.15	67.52	86.21
Model 37	97.12	96.80	94.89	84.61	67.52	86.21
Model 38	97.30	96.73	94.89	84.43	67.52	86.13
Model 39	97.23	96.97	94.66	85.22	67.52	86.13
Model 40	97.26	96.41	94.75	89.12	67.52	86.13
Model 41	96.94	96.63	94.64	82.38	67.52	86.13
Model 42	96.86	96.71	94.99	81.08	67.52	86.13
Model 43	96.76	96.43	94.82	83.46	67.52	85.84
Model 44	96.50	96.24	94.96	30.04	67.52	67.45
Model 45	94.83	95.18	96.69	35.56	67.52	67.45
Model 46	94.79	94.97	96.15	84.92	**90.88**	78.82
Model 47	94.19	94.28	95.62	77.13	**90.88**	75.91
Model 48	93.58	93.92	95.54	84.17	**90.88**	74.28
Model 49	93.14	93.57	94.90	85.04	**90.88**	76.21
Model 50	80.89	80.87	**97.34**	50.58	61.47	57.60
Model 51	74.91	74.90	77.13	66.48	61.47	57.81

(a) LightGBM (b) DT (c) KNN

(d) MLP (e) SVM (f) NB

Figure 5.1. Confusion matrices for Relief-F-Based models utilizing default parameters. (a) LightGBM, (b) DT, (c) KNN, (d) MLP, (e) SVM, and (f) NB

5.1.2. Results of mRMR-Based Models

Models constructed based on mRMR are executed using default parameter values of ML algorithms. Comprehensive evaluation results encompassing accuracy, F1-Score, recall, precision criteria, and including confusion matrices are presented. These outcomes provide a detailed overview of the outputs obtained from the algorithms and models.

Table 5.10. Accuracy values for mRMR-based predictor variables (Part 1)

Model	**Accuracy (%)**					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 52	**98.36**	96.70	**93.75**	86.63	**74.15**	57.99
Model 53	98.34	96.68	**93.75**	86.79	**74.15**	57.99
Model 54	98.34	96.55	**93.75**	87.04	**74.15**	57.99
Model 55	98.34	96.53	**93.75**	86.15	**74.15**	57.99
Model 56	98.34	96.55	**93.75**	**88.56**	**74.15**	57.99
Model 57	98.34	96.72	**93.75**	79.15	**74.15**	57.99
Model 58	98.30	96.49	**93.75**	81.28	**74.15**	57.99
Model 59	98.30	**96.93**	**93.75**	86.01	**74.15**	57.99

Table 5.11. Accuracy values for mRMR-based predictor variables (Part 2)

Model	Accuracy (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 60	98.30	96.64	**93.75**	87.25	**74.15**	57.99
Model 61	98.30	96.87	**93.75**	82.64	**74.15**	57.99
Model 62	98.34	96.84	**93.75**	83.79	**74.15**	57.99
Model 63	98.03	96.64	93.27	83.91	73.98	58.12
Model 64	97.96	96.26	92.55	85.24	73.98	58.20
Model 65	97.96	96.35	92.55	85.57	73.98	58.20
Model 66	97.96	96.33	92.55	83.20	73.98	58.20
Model 67	97.96	96.08	92.55	84.23	73.98	58.20
Model 68	97.96	96.22	92.55	81.04	73.98	58.20
Model 69	97.99	96.30	92.50	86.03	73.98	57.99
Model 70	97.99	96.43	92.50	80.27	73.98	57.99
Model 71	97.99	96.33	92.50	78.96	73.98	58.20
Model 72	97.88	96.14	92.24	85.46	74.00	58.28
Model 73	97.88	96.14	92.24	83.45	74.00	58.22
Model 74	97.84	96.33	92.09	83.74	74.00	58.31
Model 75	97.86	96.33	92.09	83.43	74.00	58.31
Model 76	97.86	96.32	92.09	87.04	74.00	58.31
Model 77	97.67	96.33	91.69	80.32	74.00	56.48
Model 78	97.67	96.22	91.67	80.90	74.00	55.96
Model 79	97.84	96.18	92.36	77.66	74.00	55.92
Model 80	97.94	96.60	92.25	77.64	74.00	55.67
Model 81	97.94	96.55	92.25	84.18	74.00	55.67
Model 82	97.51	96.24	92.25	86.09	74.00	55.61
Model 83	97.55	95.91	92.25	81.27	74.00	55.61
Model 84	96.99	95.47	91.99	81.37	74.00	55.67
Model 85	96.82	95.58	91.99	79.88	74.00	55.67
Model 86	96.55	95.22	91.49	79.77	74.00	55.69
Model 87	95.95	94.70	91.55	71.91	74.00	55.69
Model 88	95.68	94.52	91.59	80.09	74.00	55.69
Model 89	95.66	94.41	91.59	70.81	74.00	55.69
Model 90	95.10	94.52	80.77	86.75	74.00	55.63
Model 91	94.39	93.85	79.22	80.03	74.00	55.63
Model 92	94.33	94.00	76.10	82.68	74.00	55.63
Model 93	94.17	93.88	76.08	84.51	74.00	56.73
Model 94	94.10	93.98	75.89	76.67	74.00	56.73
Model 95	93.81	93.58	73.78	75.27	74.00	56.71
Model 96	93.77	93.54	73.11	80.02	74.00	73.82
Model 97	93.65	93.58	71.80	74.38	74.00	73.82
Model 98	93.69	93.54	68.40	77.06	74.00	73.82
Model 99	92.52	92.46	55.75	48.42	74.00	**73.96**
Model 100	88.70	88.56	55.69	48.13	74.00	**73.96**
Model 101	76.89	76.76	51.68	48.07	74.00	**73.96**
Model 102	75.98	76.04	49.98	48.24	74.00	**73.96**

Table 5.12. F1-Score values for mRMR-based predictor variables (Part 1)

Model	F1-Score (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 52	**98.42**	96.61	**93.99**	86.77	**79.38**	32.97
Model 53	98.40	96.72	**93.99**	87.53	**79.38**	32.97
Model 54	98.40	96.93	**93.99**	82.23	**79.38**	32.97
Model 55	98.40	96.70	**93.99**	85.94	**79.38**	32.97
Model 56	98.40	96.74	**93.99**	84.95	**79.38**	32.97
Model 57	98.40	96.68	**93.99**	**89.81**	**79.38**	32.97
Model 58	98.37	**96.99**	**93.99**	83.98	**79.38**	32.97
Model 59	98.37	96.93	**93.99**	86.98	**79.38**	32.97
Model 60	98.37	96.95	**93.99**	86.92	**79.38**	32.97
Model 61	98.37	96.87	**93.99**	87.26	**79.38**	32.97
Model 62	98.40	**96.99**	**93.99**	82.47	**79.38**	32.97
Model 63	98.10	96.66	93.57	83.64	79.27	33.08
Model 64	98.03	96.45	92.91	83.65	79.27	33.16
Model 65	98.03	96.45	92.91	80.98	79.27	33.16
Model 66	98.03	96.51	92.91	86.82	79.27	33.16
Model 67	98.03	96.34	92.91	80.93	79.27	33.16
Model 68	98.03	96.47	92.91	83.00	79.27	33.16
Model 69	98.07	96.51	92.85	79.43	79.27	32.87
Model 70	98.07	96.34	92.85	80.75	79.27	32.87
Model 71	98.07	96.46	92.85	84.27	79.27	34.49
Model 72	97.96	96.28	92.62	79.18	79.29	35.47
Model 73	97.96	96.35	92.62	85.80	79.29	34.78
Model 74	97.92	96.47	92.46	83.03	79.29	34.67
Model 75	97.94	96.40	92.46	79.78	79.29	34.67
Model 76	97.94	96.45	92.46	81.98	79.29	34.67
Model 77	97.75	96.67	92.09	72.34	79.29	30.69
Model 78	97.75	96.42	92.08	74.36	79.29	29.28
Model 79	97.92	96.55	92.75	82.48	79.29	29.30
Model 80	98.01	96.63	92.64	67.66	79.29	28.61
Model 81	98.01	96.59	92.64	74.75	79.29	28.61
Model 82	97.60	96.20	92.63	79.02	79.29	28.54
Model 83	97.64	96.20	92.63	82.25	79.29	28.54
Model 84	97.11	95.76	92.41	80.20	79.29	28.38
Model 85	96.94	95.68	92.41	75.83	79.29	28.38
Model 86	96.68	95.45	91.96	65.34	79.29	28.39
Model 87	96.13	95.17	92.03	79.36	79.29	28.39
Model 88	95.87	94.70	92.06	71.68	79.29	28.39
Model 89	95.85	94.52	92.06	81.85	79.29	28.39
Model 90	95.35	94.83	77.88	64.39	79.29	28.32
Model 91	94.69	94.22	75.63	80.80	79.29	28.32
Model 92	94.65	94.29	70.67	73.33	79.29	28.32
Model 93	94.51	94.26	70.67	68.91	79.29	32.22

Table 5.13. F1-Score values for mRMR-based predictor variables (Part 2)

Model	F1-Score (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 94	94.43	94.28	70.35	75.42	79.29	32.22
Model 95	94.24	94.02	66.87	69.95	79.29	32.17
Model 96	94.21	94.00	65.74	85.42	79.29	79.14
Model 97	94.10	93.98	63.44	87.06	79.29	79.14
Model 98	94.14	94.00	56.97	82.94	79.29	79.14
Model 99	93.07	93.02	26.69	0.00	79.29	**79.29**
Model 100	89.86	89.70	26.38	6.82	79.29	**79.29**
Model 101	80.93	80.88	13.49	13.65	79.29	**79.29**
Model 102	80.49	80.55	7.13	0.00	79.29	**79.29**

Table 5.14. Recall values for mRMR-based predictor variables (Part 1)

Model	Recall (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 52	98.66	96.50	94.37	74.26	96.01	19.98
Model 53	98.66	96.68	94.37	90.16	96.01	19.98
Model 54	**98.70**	96.65	94.37	85.15	96.01	19.98
Model 55	**98.70**	96.76	94.37	80.72	96.01	19.98
Model 56	**98.70**	96.72	94.37	85.42	96.01	19.98
Model 57	**98.70**	96.79	94.37	80.99	96.01	19.98
Model 58	98.58	96.83	94.37	89.97	96.01	19.98
Model 59	98.58	97.17	94.37	89.26	96.01	19.98
Model 60	98.58	96.87	94.37	80.34	96.01	19.98
Model 61	98.58	96.87	94.37	83.09	96.01	19.98
Model 62	98.66	96.76	94.37	90.76	96.01	19.98
Model 63	98.40	96.80	**94.67**	89.34	96.05	20.02
Model 64	98.40	96.24	94.26	82.57	96.05	20.05
Model 65	98.40	96.24	94.26	86.99	96.05	20.05
Model 66	98.40	96.20	94.26	81.29	96.05	20.05
Model 67	98.40	96.46	94.26	86.47	96.05	20.05
Model 68	98.40	96.27	94.26	77.69	96.05	20.05
Model 69	98.36	95.94	94.15	87.67	96.05	19.90
Model 70	98.36	96.42	94.15	81.74	96.05	19.90
Model 71	98.36	96.12	94.15	82.75	96.05	21.28
Model 72	98.29	96.42	94.04	82.65	**96.09**	22.18
Model 73	98.29	96.24	94.04	84.77	**96.09**	21.55
Model 74	98.21	96.61	93.74	83.59	**96.09**	21.40
Model 75	98.36	96.68	93.74	85.95	**96.09**	21.40
Model 76	98.36	96.57	93.74	81.58	**96.09**	21.40
Model 77	98.10	96.50	93.51	86.18	**96.09**	18.64
Model 78	98.10	96.61	93.59	81.87	**96.09**	17.63

Table 5.15. Recall values for mRMR-based predictor variables (Part 2)

Model	Recall (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 79	98.14	96.38	94.33	78.89	**96.09**	17.67
Model 80	98.28	96.79	94.22	78.86	**96.09**	17.18
Model 81	98.28	96.65	94.22	88.59	**96.09**	17.18
Model 82	97.76	96.31	94.15	80.26	**96.09**	17.15
Model 83	97.91	96.16	94.15	83.86	**96.09**	17.15
Model 84	97.47	95.45	94.15	86.06	**96.09**	17.00
Model 85	97.28	95.34	94.15	82.89	**96.09**	17.00
Model 86	97.13	95.23	94.00	79.14	**96.09**	17.00
Model 87	96.98	95.60	94.19	**91.84**	**96.09**	17.00
Model 88	96.87	95.01	94.19	70.78	**96.09**	17.00
Model 89	96.72	95.01	94.19	70.67	**96.09**	17.00
Model 90	96.83	96.27	65.82	72.31	**96.09**	16.96
Model 91	96.61	96.12	62.47	87.91	**96.09**	16.96
Model 92	96.98	96.65	55.80	72.68	**96.09**	16.96
Model 93	96.83	96.61	55.83	71.71	**96.09**	19.94
Model 94	96.68	96.72	55.46	78.51	**96.09**	19.94
Model 95	97.80	97.65	51.32	87.47	**96.09**	19.91
Model 96	97.84	**97.73**	50.02	79.51	**96.09**	95.83
Model 97	97.69	97.65	47.45	86.79	**96.09**	95.83
Model 98	97.76	97.69	40.44	87.21	**96.09**	95.83
Model 99	97.13	97.02	15.65	0.00	**96.09**	**96.24**
Model 100	96.68	96.50	15.43	10.11	**96.09**	**96.24**
Model 101	94.79	94.78	7.30	0.00	**96.09**	**96.24**
Model 102	95.60	95.75	3.73	30.00	**96.09**	**96.24**

Table 5.16. Precision values for mRMR-based predictor variables (Part 1)

Model	Precision (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 52	**98.19**	96.83	93.62	81.89	67.68	94.56
Model 53	98.16	96.73	93.62	84.04	67.68	94.56
Model 54	98.12	96.70	93.62	86.93	67.68	94.56
Model 55	98.12	96.44	93.62	88.76	67.68	94.56
Model 56	98.12	96.63	93.62	90.04	67.68	94.56
Model 57	98.12	96.62	93.62	87.87	67.68	94.56
Model 58	98.15	96.62	93.62	87.98	67.68	94.56
Model 59	98.15	96.85	93.62	86.42	67.68	94.56
Model 60	98.15	96.76	93.62	**90.99**	67.68	94.56
Model 61	98.15	**96.96**	93.62	89.68	67.68	94.56
Model 62	98.15	96.77	93.62	90.39	67.68	94.56
Model 63	97.82	96.47	92.50	83.17	67.51	95.53

Table 5.17. Precision values for mRMR-based predictor variables (Part 2)

Model	Precision (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 64	97.67	96.68	91.62	89.02	67.51	**96.07**
Model 65	97.67	96.64	91.62	80.78	67.51	**96.07**
Model 66	97.67	96.67	91.62	84.86	67.51	**96.07**
Model 67	97.67	96.54	91.62	89.96	67.51	**96.07**
Model 68	97.67	96.64	91.62	89.18	67.51	**96.07**
Model 69	97.78	96.46	91.61	85.04	67.51	95.06
Model 70	97.78	96.75	91.61	83.22	67.51	95.06
Model 71	97.78	96.87	91.61	86.71	67.51	91.41
Model 72	97.64	96.36	91.26	82.04	**67.52**	88.87
Model 73	97.64	96.62	91.26	82.38	**67.52**	90.56
Model 74	97.64	96.19	91.24	83.99	**67.52**	91.84
Model 75	97.53	96.44	91.24	85.15	**67.52**	91.84
Model 76	97.53	96.12	91.24	85.29	**67.52**	91.84
Model 77	97.41	96.55	90.72	86.77	**67.52**	87.24
Model 78	97.41	96.62	90.63	85.94	**67.52**	86.55
Model 79	97.70	96.47	91.22	86.09	**67.52**	86.13
Model 80	97.74	96.77	91.11	84.49	**67.52**	85.76
Model 81	97.74	96.84	91.11	84.94	**67.52**	85.76
Model 82	97.44	96.13	91.18	85.14	**67.52**	85.42
Model 83	97.37	96.24	91.18	83.36	**67.52**	85.42
Model 84	96.76	96.03	90.76	81.33	**67.52**	86.43
Model 85	96.61	95.78	90.76	87.01	**67.52**	86.43
Model 86	96.24	95.20	90.03	86.17	**67.52**	86.58
Model 87	95.30	94.90	89.98	82.83	**67.52**	86.58
Model 88	94.91	94.14	90.05	79.43	**67.52**	86.58
Model 89	95.01	94.24	90.05	84.31	**67.52**	86.58
Model 90	93.94	93.34	95.71	85.25	**67.52**	86.22
Model 91	92.88	92.36	96.01	85.05	**67.52**	86.22
Model 92	92.47	92.09	**96.59**	84.09	**67.52**	86.22
Model 93	92.33	92.10	96.46	81.88	**67.52**	89.41
Model 94	92.32	92.05	96.44	83.90	**67.52**	89.41
Model 95	90.96	90.63	96.30	79.72	**67.52**	89.40
Model 96	90.86	90.56	96.21	75.05	**67.52**	67.42
Model 97	90.79	90.59	96.09	77.71	**67.52**	67.42
Model 98	90.80	90.60	96.44	76.67	**67.52**	67.42
Model 99	89.36	89.35	93.37	0.00	**67.52**	67.45
Model 100	83.95	83.84	94.01	15.55	**67.52**	67.45
Model 101	70.73	70.63	91.52	5.18	**67.52**	67.45
Model 102	69.53	69.55	91.37	5.17	**67.52**	67.45

(a) LightGBM (b) DT (c) KNN

(d) MLP (e) SVM (f) NB

Figure 5.2. Confusion matrices for mRMR-Based models utilizing default parameters. (a) LightGBM, (b) DT, (c) KNN, (d) MLP, (e) SVM, and (f) NB

5.2. Discussion of Models with Default Parameters

This section provides a detailed analysis of the obtained results and emphasizes the overall significance of ML default configurations of algorithms. Particularly, findings derived from the Relief-F and mRMR feature selection methods comprehensively elucidate the potential impact on the effectiveness and applicability of these techniques.

5.2.1. Discussion of Relief-F-Based Models

The examination of models containing PE headers, created using Relief-F feature selection and evaluated with default parameters of ML classifiers, is as follows:

- Upon analyzing the average accuracy values of models constructed through feature selection, it becomes evident that the LightGBM algorithm demonstrates a notably higher accuracy compared to other methods. While the DT method exhibits an accuracy rate 1.31% lower than LightGBM, the KNN algorithm is 7.35% less accurate. Furthermore, MLP shows a 20.18% lower accuracy, SVM performs 30.41% less accurately, and NB showcases a significantly lower accuracy rate of 64.00% compared to the LightGBM. These findings underscore the effectiveness of using Relief-F feature

selection in conjunction with the LightGBM algorithm, particularly when dealing with datasets containing PE titles. This evaluation highlights relative of LightGBM remarkable among these algorithms, emphasizing the significant contribution of feature selection methods to the performance of ML models.

- When considering models classified with default values, the LightGBM exhibits the highest accuracy, followed by DT, KNN, MLP, SVM, and NB, respectively. Upon scrutiny of Relief-F-based models, distinct models show high accuracy for each method employed. Examining the methods and their corresponding models, within the LightGBM method, models 13 and 14 attain the highest accuracy. For DT, model 16 records the highest accuracy, while for KNN, it is model 34. Within the MLP method, model 4 demonstrates the highest accuracy. Notably, models from 46 to 49 yield the same result, securing the highest accuracy within the SVM method, and finally, model 43 exhibits the highest accuracy within the NB method. This comprehensive analysis underscores the variability in model performance across different methodologies and highlights specific models that excel within their respective classification techniques.
- Analyzing the average F1-Score values of models created through feature selection reveals that the LightGBM algorithm demonstrates higher F1-Score values compared to other methodologies. When examining F1-Score performance, other algorithms exhibit relatively lower performance percentages when compared to LightGBM. In comparison to LightGBM, the DT algorithm shows a 1.24% lower performance, KNN algorithm shows an 8.78% lower performance, MLP algorithm exhibits an 18.91% lower performance, SVM algorithm displays an 18.49% lower performance, and notably, NB method demonstrates a significantly lower performance of 61.11% in terms of F1-Score. These findings underscore the effectiveness of Relief-F feature selection in achieving superior classification performance, particularly when dealing with datasets containing PE headers, in conjunction with the LightGBM algorithm. This assessment clarifies the relative superiority of the LightGBM among these algorithms, emphasizing the significant contribution of feature selection methods in enhancing the performance of ML models.
- When considering models classified with default parameters, the LightGBM demonstrates the highest F1-Score value, followed by DT, KNN, MLP, SVM, and NB in sequence. Examining Relief-F-based models reveals distinct models showcasing high F1-Score values for each utilized method. Within the scope of the LightGBM method, upon scrutinizing the methods and their corresponding models, models 13 and 14 achieve the highest F1-Score. For DT, model 17 records the highest F1-Score, while for KNN, it is model 34. Particularly, models from 46 to 49 consistently yield the same result, securing the highest F1-Score within the SVM method, and within the MLP

method, model 8 showcases the highest F1-Score. Finally, model 43 exhibits the highest F1-Score within the NB method. This comprehensive analysis underscores the variability in model performance across different methodologies and highlights specific models that excel within their respective classification techniques.

- When analyzing the average recall values of models created through feature selection, it became evident that the LightGBM algorithm exhibits higher recall values compared to other methodologies. In terms of recall performance, algorithms closest to LightGBM show relatively lower percentages. Compared to LightGBM, the DT algorithm demonstrates a 1.55% lower recall, KNN 10.13%, MLP 18.46%, SVM 4.36%, and notably, the NB algorithm shows a 70.28% lower performance. These findings highlight the efficacy of Relief-F feature selection, particularly when dealing with datasets that include PE header attributes, in achieving superior classification performance alongside the LightGBM algorithm. This assessment not only clarifies the relative superiority of LightGBM among these algorithms but also emphasizes the significant contribution of feature selection methods in enhancing the performance of ML models.
- When considering models classified with default parameters, LightGBM demonstrates the highest recall value, followed by DT, NB, SVM, KNN, and MLP, in that order. Delving into Relief-F-based models reveals distinct models showcasing high recall values for each method employed. Among these methods, LightGBM attains the highest recall values for models 13, 14, and 18. For DT, model 15 records the highest recall value, while NB reaches its peak recall in models 44 and 45. Meanwhile, SVM encompasses similar recall values across models 19 through 45. Furthermore, KNN exhibits its optimal recall performance in model 35, whereas for MLP is model 6.
- When examining the average precision values of models created through feature selection, it becomes evident that the LightGBM algorithm demonstrates higher precision compared to other methodologies. Algorithms closest to LightGBM in terms of precision performance show relatively lower percentages. Compared to LightGBM, the DT algorithm shows a difference of 0.90%, the KNN algorithm 3.77%, the NB algorithm 11.35%, and the MLP algorithm 15.01%. Additionally, the SVM algorithm lags behind with a precision difference of 28.35%. These findings underscore the effectiveness of Relief-F feature selection, particularly when dealing with datasets that include PE attributes, in achieving superior classification performance alongside the LightGBM algorithm.
- When considering models classified with default parameters, LightGBM demonstrates the highest precision value, followed by KNN, DT, NB, MLP, and SVM, in that order. Examining Relief-F-based models reveals different models showcasing high precision values for each method employed. Among these methods, LightGBM attains the highest

precision values for models 13, 14, and 16, while model 50 stands out for KNN in terms of the highest precision value. Additionally, DT demonstrates the highest precision value for model 18, and NB attains the highest precision for models 1 and 2. Model 6 surfaces as the one yielding the best precision for MLP, whereas models ranging from 46 to 49 notably present the best precision values for SVM.

- When considering the aggregate mean metrics of Relief-F-based models, LightGBM outperformed all other ML methods evaluated with default parameters, emerging as the most effective classifier across all metrics.
- The study was conducted on a computer equipped with a 2.20 GHz Xenon CPU, 16 GB of RAM, and running a 64-bit Windows 10 Professional Operating System. The average execution times for the LightGBM, DT, KNN, MLP, SVM, and NB algorithms were calculated as follows: LightGBM took 315.9 seconds, DT 41.7 seconds, KNN 73.7 seconds, MLP 249.3 seconds, SVM 265.1 seconds, and NB 31.1 seconds, respectively.

5.2.2. Discussion of mRMR-Based Models

The examination of models containing PE headers, created using mRMR feature selection and evaluated with default parameters of ML classifiers, is as follows:

- When examining the average accuracy scores of models obtained through mRMR feature selection, it has been determined that the LightGBM algorithm exhibits higher accuracy scores compared to other methodologies. In terms of this performance metric, other algorithm, generally show lower results compared to the LightGBM. Specifically, while the DT method shows a 1.27% lower accuracy rate, KNN 9.93%, MLP 17.29%, SVM 22.87%, and NB 38.10% lower accuracy rates when compared to LightGBM. These findings underscore the effectiveness of combining mRMR feature selection with the LightGBM algorithm, especially when dealing with datasets containing PE attributes, in achieving superior classification performance.
- When considering default parameters for classification, LightGBM demonstrates the highest accuracy, followed by DT, KNN, MLP, SVM, and NB, respectively. Analysis of mRMR-based models reveals different models highlighting high accuracy for each method employed. For LightGBM, the model with the highest accuracy is found at model 52. The highest accuracy for DT is recorded at model 59, while KNN exhibits similar accuracy values between models 52 and 62. MLP's highest accuracy is found at model 56, and for SVM, models with similar accuracy values are between 52 and 62. The models displaying the best accuracy for NB fall between 99 and 102. This comprehensive analysis highlights the variability in model performance among

different methodologies, displaying specific models that stand out in terms of relevant classification techniques.

- When the average F1-Score values of models obtained through mRMR feature selection are examined, it is evident that the LightGBM algorithm showcases higher F1-Score ratings compared to other methodologies. In terms of this performance metric, other algorithms generally demonstrate lower results compared to the LightGBM. Specifically, while the DT method exhibits a 1.18% lower F1-Score rate compared to LightGBM, KNN shows a 13.96%, MLP 22.75%, SVM 17.70%, and the NB 60.37% lower F1-Score rates. These findings emphasize the effectiveness of combining mRMR feature selection with the LightGBM algorithm, particularly when dealing with datasets containing the PE attributes, in achieving superior classification performance.
- Consideration of default parameters for classification reveals that LightGBM demonstrates the highest F1-Score, followed by DT, KNN, MLP, SVM, and NB respectively. An analysis of mRMR-based models reveals distinct models displaying high F1-Score values for each method. The model with the highest F1-Score for LightGBM is found in Model 52. For DT, the highest F1-Score is recorded between Models 58 and 62, while KNN exhibits the same F1-Score values across Models 52 and 62. Model 57 holds the highest F1-Score for MLP, and for SVM, the models ranging from 52 to 62 demonstrate the best F1-Score values. Models 99 to 102 reveal the best F1-Score values for NB. This comprehensive analysis highlights specific models that stand out in relevant classification techniques, emphasizing the variability in model performance among different methodologies.
- When examining the average recall values of models obtained through mRMR feature selection, it is evident that the LightGBM algorithm demonstrates higher recall values compared to other methodologies. In terms of performance measurement, other algorithms exhibit lower results in comparison to LightGBM. Specifically, while the DT method shows a recall rate 1.47% lower than LightGBM, KNN shows 17.74%, MLP 21.27%, SVM 1.83%, and NB 69.61% lower recall rates compared to LightGBM. These findings underscore the effectiveness of combining mRMR feature selection with the LightGBM algorithm, particularly when dealing with datasets containing the PE attributes, in achieving superior classification performance.
- When considering the default parameters of ML methods for classification, the LightGBM demonstrates the highest recall value, followed by DT, NB, SVM, KNN, and MLP, respectively. The analysis of mRMR-based models reveals distinct models with high recall values for each method. For LightGBM, the highest recall values are achieved in models 54, 55, 56, and 57. Model 96 records the highest recall value for DT, while models between 99 and 102 yield similar results for NB. Moreover, SVM

exhibits a consistent highest recall value across models numbered from 72 to 102, whereas for KNN, this value is found in model 63. Additionally, the best recall value for MLP is observed in model 87. This comprehensive analysis highlights the variability in model performance among different methodologies by highlighting specific standout models in relevant classification.

- When examining the average precision values of models obtained through mRMR feature selection, it is observed that the LightGBM algorithm demonstrates higher values compared to other methodologies. In terms of performance measurement, other algorithms exhibit lower results compared to LightGBM. Specifically, while the DT method shows a rate 0.97% lower than LightGBM, KNN shows 2.46%, MLP 17.27%, SVM 28.91%, and NB 7.68% lower precision rates compared to LightGBM. These findings emphasize the impact of merging mRMR feature selection with the LightGBM algorithm, particularly when dealing with datasets containing PE attributes, in achieving superior classification performance.
- When considering the default parameters of ML methods for classification, LightGBM demonstrates the highest precision value, followed by DT, KNN, NB, MLP, and SVM, respectively. The analysis of mRMR-based models reveals various models with high precision values for each respective method. The highest precision values for LightGBM are observed in model 52. For DT, model 61 records the highest precision value, whereas for KNN, it is model 92. Additionally, NB consistently demonstrates a precision value within models numbered from 64 to 68, while for MLP, this value is found in model 60. Notably, SVM maintains consistent precision values across models numbered from 72 to 102. This comprehensive analysis emphasizes the variability in models and highlights specific standout models in the relevant classification, addressing model performance across different methodologies.
- When considering the collective average metrics of mRMR-based models, the LightGBM emerged as the most effective classifier among all metrics, outperforming all other ML methods evaluated with default parameters.
- The study was conducted on a computer equipped with a 2.20 GHz Xenon CPU, 16 GB of RAM, running on a 64-bit Windows 10 Professional Operating System. The average execution times for LightGBM, DT, KNN, MLP, SVM, and NB algorithms were calculated as follows: LightGBM took 207.8 seconds, DT took 38.3 seconds, KNN took 101.1 seconds, MLP took 228.4 seconds, SVM took 563.9 seconds, and NB took 29.4 seconds, respectively.

5.3. Results of Models with Optimized Parameters

The accuracy, F1-Score, recall and precision results of the algorithms created with their optimized configurations are presented in this section. Confusion matrices are also provided to evaluate classification of the best model for each method.

5.3.1. Results of Relief-F-Based Models

Models created based on Relief-F are applied with the best parameter values resulting from hyperparameter optimization of ML algorithms in this section. Evaluation results, encompassing accuracy, F1-Score, recall, precision, and the confusion matrix criteria, are comprehensively presented. These results provide a detailed overview of the outputs obtained from the algorithms and models.

Table 5.18. Accuracy values for Relief-F-based predictor variables (Part 1)

Model	**Accuracy (%)**					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 1	98.96	96.89	97.40	96.66	95.91	74.42
Model 2	98.90	96.78	97.40	97.05	95.93	74.98
Model 3	**98.98**	97.03	97.40	97.13	95.85	75.75
Model 4	98.90	97.13	97.40	96.74	95.85	75.83
Model 5	98.86	96.78	97.40	97.03	95.91	77.87
Model 6	98.92	97.03	97.40	96.49	95.95	77.89
Model 7	98.96	97.01	97.40	96.74	95.91	78.40
Model 8	98.90	97.11	97.40	96.51	95.93	76.97
Model 9	98.88	97.07	97.40	96.84	95.93	77.72
Model 10	98.82	97.03	97.38	97.05	95.06	77.91
Model 11	98.92	97.22	97.45	96.47	95.99	76.02
Model 12	98.90	**97.34**	97.45	96.84	95.91	76.21
Model 13	98.92	97.03	**97.47**	96.89	96.03	77.58
Model 14	98.92	96.93	**97.47**	96.87	95.93	76.89
Model 15	98.86	96.84	**97.47**	97.07	96.01	84.28
Model 16	98.94	97.18	**97.47**	97.05	95.18	85.17
Model 17	98.92	96.99	**97.47**	96.18	95.95	84.94
Model 18	98.88	96.97	**97.47**	97.01	96.18	84.90

Table 5.19. Accuracy values for Relief-F-based predictor variables (Part 2)

Model	Accuracy (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 19	98.46	96.51	97.40	96.95	95.25	86.32
Model 20	98.36	96.72	97.38	96.97	96.20	85.92
Model 21	98.32	96.06	97.40	**97.28**	95.27	83.83
Model 22	98.26	95.49	97.40	96.74	96.16	85.88
Model 23	98.36	96.61	97.40	96.82	96.20	85.21
Model 24	98.32	96.72	97.40	96.86	96.20	78.41
Model 25	98.26	96.22	97.40	96.89	96.18	77.47
Model 26	98.15	96.32	97.34	96.97	95.33	78.36
Model 27	98.11	96.60	97.34	96.49	96.24	77.91
Model 28	98.03	96.16	97.24	96.64	95.95	77.59
Model 29	97.90	96.45	97.03	96.57	95.62	77.86
Model 30	97.72	96.39	96.82	96.53	95.54	77.33
Model 31	97.82	96.41	96.82	96.51	95.66	77.35
Model 32	97.70	96.47	96.82	96.32	95.68	77.74
Model 33	97.76	96.45	96.78	96.62	95.79	81.21
Model 34	97.78	96.64	96.87	96.22	96.08	77.66
Model 35	97.72	96.60	96.87	96.32	96.22	81.48
Model 36	97.69	96.43	96.86	96.06	96.28	82.47
Model 37	97.61	96.51	96.82	95.97	**96.45**	81.52
Model 38	97.72	96.66	96.74	96.45	96.16	83.18
Model 39	97.61	96.68	96.80	96.33	96.22	82.70
Model 40	97.45	96.49	96.53	95.56	95.72	81.00
Model 41	97.43	95.43	96.43	95.08	95.74	77.97
Model 42	97.51	96.08	96.43	94.54	94.93	73.53
Model 43	97.34	96.26	96.41	94.25	94.79	76.43
Model 44	97.34	96.08	96.18	91.67	94.41	**86.57**
Model 45	96.14	95.54	95.68	91.22	93.54	86.54
Model 46	96.01	95.41	95.58	92.88	93.88	66.26
Model 47	95.51	94.87	94.77	91.40	93.73	66.42
Model 48	95.10	94.54	94.52	91.90	92.94	71.37
Model 49	94.85	94.60	94.46	91.94	92.86	77.59
Model 50	85.98	85.61	66.20	62.02	83.41	60.90
Model 51	81.42	81.37	54.76	59.47	77.74	51.77

Table 5.20. F1-Score values for Relief-F-based predictor variables (Part 1)

Model	F1-Score (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 1	98.99	96.99	97.50	97.10	94.48	79.40
Model 2	98.94	96.48	97.50	97.13	94.48	79.72
Model 3	**99.01**	97.15	97.50	96.97	94.45	79.97
Model 4	98.94	97.21	97.50	97.03	94.41	79.78

Table 5.21. F1-Score values for Relief-F-based predictor variables (Part 2)

Model	F1-Score (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 5	98.90	96.93	97.50	96.91	94.52	80.21
Model 6	98.96	97.14	97.50	96.63	94.52	80.36
Model 7	99.00	97.11	97.50	97.14	94.52	80.57
Model 8	98.94	97.09	97.50	96.89	94.52	80.31
Model 9	98.92	97.18	97.50	**97.24**	94.52	80.29
Model 10	98.87	97.29	97.48	97.03	96.27	80.04
Model 11	98.96	96.80	97.55	96.78	94.67	80.29
Model 12	98.94	97.14	97.55	97.15	94.56	80.08
Model 13	98.96	97.11	**97.57**	97.07	94.74	80.35
Model 14	98.96	97.07	**97.57**	**97.24**	94.56	80.24
Model 15	98.90	**97.51**	**97.57**	97.18	94.67	82.81
Model 16	98.98	97.14	**97.57**	97.21	96.64	83.95
Model 17	98.96	97.18	**97.57**	96.28	94.56	83.74
Model 18	98.92	97.10	**97.57**	97.02	94.97	83.78
Model 19	98.51	96.67	97.50	96.83	96.64	86.07
Model 20	98.42	96.56	97.48	97.22	95.08	85.72
Model 21	98.38	96.16	97.50	97.16	96.57	85.58
Model 22	98.33	95.52	97.50	96.77	95.01	85.91
Model 23	98.42	96.34	97.50	97.03	95.15	83.76
Model 24	98.38	96.56	97.50	96.94	95.15	81.22
Model 25	98.33	95.98	97.50	97.00	95.08	80.91
Model 26	98.22	96.49	97.44	97.04	96.57	81.24
Model 27	98.19	96.84	97.44	96.93	95.19	81.04
Model 28	98.11	96.33	97.35	96.81	95.97	81.05
Model 29	97.98	96.51	97.15	96.77	94.78	81.18
Model 30	97.81	96.55	96.95	96.48	94.86	81.18
Model 31	97.91	96.55	96.95	96.58	95.12	81.24
Model 32	97.80	96.58	96.95	96.76	95.12	81.51
Model 33	97.84	96.55	96.91	96.42	95.42	81.72
Model 34	97.86	96.75	97.01	96.14	96.91	81.17
Model 35	97.81	96.69	97.01	96.12	97.09	82.08
Model 36	97.77	96.30	96.99	96.48	**97.24**	82.88
Model 37	97.69	96.98	96.95	96.40	97.06	82.88
Model 38	97.80	96.72	96.88	96.59	96.20	83.65
Model 39	97.70	96.79	96.93	96.19	96.20	84.30
Model 40	97.54	96.67	96.67	95.69	96.27	81.87
Model 41	97.53	95.28	96.56	95.30	96.20	79.74
Model 42	97.60	96.04	96.56	94.70	96.09	79.44
Model 43	97.43	96.36	96.54	94.73	95.53	79.60
Model 44	97.44	96.32	96.32	92.42	94.26	87.94
Model 45	96.32	95.72	95.84	91.83	93.37	**87.95**
Model 46	96.19	95.57	95.78	93.30	95.49	75.28

Table 5.22. F1-Score values for Relief-F-based predictor variables (Part 3)

Model	F1-Score (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 47	95.72	95.11	94.98	91.93	95.23	75.36
Model 48	95.35	94.80	94.75	92.23	94.56	71.28
Model 49	95.11	94.90	94.73	92.52	95.23	78.54
Model 50	87.57	87.27	55.40	67.19	93.52	70.96
Model 51	84.33	84.24	25.46	67.24	93.92	68.22

Table 5.23. Recall values for Relief-F-based predictor variables (Part 1)

Model	Recall (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 1	99.03	96.98	97.91	96.72	94.48	94.30
Model 2	99.11	96.23	97.91	97.13	94.48	94.30
Model 3	99.18	**97.13**	97.91	97.13	94.45	92.85
Model 4	99.07	96.50	97.91	97.02	94.41	91.55
Model 5	99.03	96.94	97.91	97.20	94.52	86.93
Model 6	99.11	97.02	97.91	97.09	94.52	87.75
Model 7	99.14	96.98	97.91	96.68	94.52	87.01
Model 8	99.14	96.98	97.91	96.61	94.52	90.84
Model 9	**99.22**	97.02	97.91	97.24	94.52	87.79
Model 10	99.14	96.61	97.88	**97.32**	96.27	85.85
Model 11	99.07	96.46	98.02	97.24	94.67	93.93
Model 12	**99.22**	96.50	98.02	97.02	94.56	92.07
Model 13	99.11	96.79	98.06	96.98	94.74	88.87
Model 14	99.07	96.94	98.06	97.06	94.56	90.84
Model 15	99.14	96.94	98.06	97.13	94.67	78.94
Model 16	**99.22**	96.79	98.06	97.28	96.64	81.06
Model 17	99.18	96.98	**98.10**	96.94	94.56	80.80
Model 18	99.07	96.72	**98.10**	97.24	94.97	81.36
Model 19	98.66	95.79	97.99	96.61	96.64	82.74
Model 20	98.58	95.60	97.91	96.87	95.08	82.82
Model 21	98.55	96.05	97.95	97.24	96.57	92.40
Model 22	98.51	96.27	97.95	96.87	95.01	83.75
Model 23	98.58	95.97	97.95	96.87	95.15	79.57
Model 24	98.43	96.98	97.95	96.91	95.15	90.58
Model 25	98.55	95.34	97.95	97.09	95.08	92.29
Model 26	98.73	96.16	97.91	97.24	96.57	90.95
Model 27	98.81	95.98	97.91	97.28	95.19	91.51
Model 28	98.55	96.12	97.88	96.39	95.97	92.81
Model 29	98.40	96.35	97.65	97.02	94.78	92.48
Model 30	98.36	96.20	97.54	96.65	94.86	94.41
Model 31	98.51	96.53	97.54	97.06	95.12	94.75
Model 32	98.43	96.35	97.54	96.76	95.12	94.75

Table 5.24. Recall values for Relief-F-based predictor variables (Part 2)

Model	Recall (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 33	98.06	96.38	97.54	96.98	95.42	81.37
Model 34	97.99	96.68	97.80	96.16	96.91	93.07
Model 35	98.17	96.50	97.80	96.16	97.09	82.15
Model 36	98.14	95.75	97.80	96.24	**97.24**	82.30
Model 37	97.91	96.50	97.73	96.53	97.06	86.63
Model 38	97.95	96.68	97.65	96.61	96.20	83.34
Model 39	97.99	96.72	97.73	96.61	96.20	89.91
Model 40	97.65	95.56	97.39	96.27	96.27	83.01
Model 41	97.88	95.60	96.87	96.31	96.20	83.79
Model 42	97.91	96.20	96.68	96.38	96.09	98.70
Model 43	97.54	96.09	96.68	96.24	95.53	89.16
Model 44	97.91	96.05	96.65	95.71	94.26	94.56
Model 45	97.65	96.42	96.35	96.94	93.37	94.93
Model 46	97.43	96.61	96.83	95.34	95.49	99.14
Model 47	97.17	96.01	95.60	95.23	95.23	99.18
Model 48	96.98	95.86	95.60	94.11	94.56	68.47
Model 49	96.76	96.27	96.16	94.07	95.23	79.24
Model 50	95.38	94.63	40.66	80.47	93.52	92.48
Model 51	96.50	96.35	14.94	80.36	93.92	**100.00**

Table 5.25. Precision values for Relief-F-based predictor variables (Part 1)

Model	Precision (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 1	**98.96**	97.27	97.08	96.67	97.55	69.37
Model 2	98.78	97.54	97.08	96.54	97.59	69.78
Model 3	98.85	97.31	97.08	97.11	97.47	71.10
Model 4	98.81	**97.78**	97.08	97.17	97.51	71.74
Model 5	98.77	97.57	97.08	96.59	97.51	75.82
Model 6	98.81	97.35	97.08	96.05	97.59	75.27
Model 7	98.85	97.31	97.08	97.17	97.52	76.10
Model 8	98.74	97.24	97.08	96.44	97.55	72.97
Model 9	98.63	97.34	97.08	97.33	97.55	75.41
Model 10	98.59	97.14	97.08	96.86	94.31	76.60
Model 11	98.85	97.61	**97.09**	96.05	97.52	70.42
Model 12	98.67	97.10	**97.09**	97.03	97.48	71.49
Model 13	98.81	97.23	**97.09**	96.78	97.52	74.26
Model 14	98.85	97.24	**97.09**	97.36	97.52	72.80
Model 15	98.67	97.32	**97.09**	96.98	97.55	89.75
Model 16	98.74	97.42	**97.09**	96.97	94.21	89.64
Model 17	98.74	97.35	97.06	95.95	97.55	89.46
Model 18	98.78	97.30	97.06	97.11	**97.60**	88.95

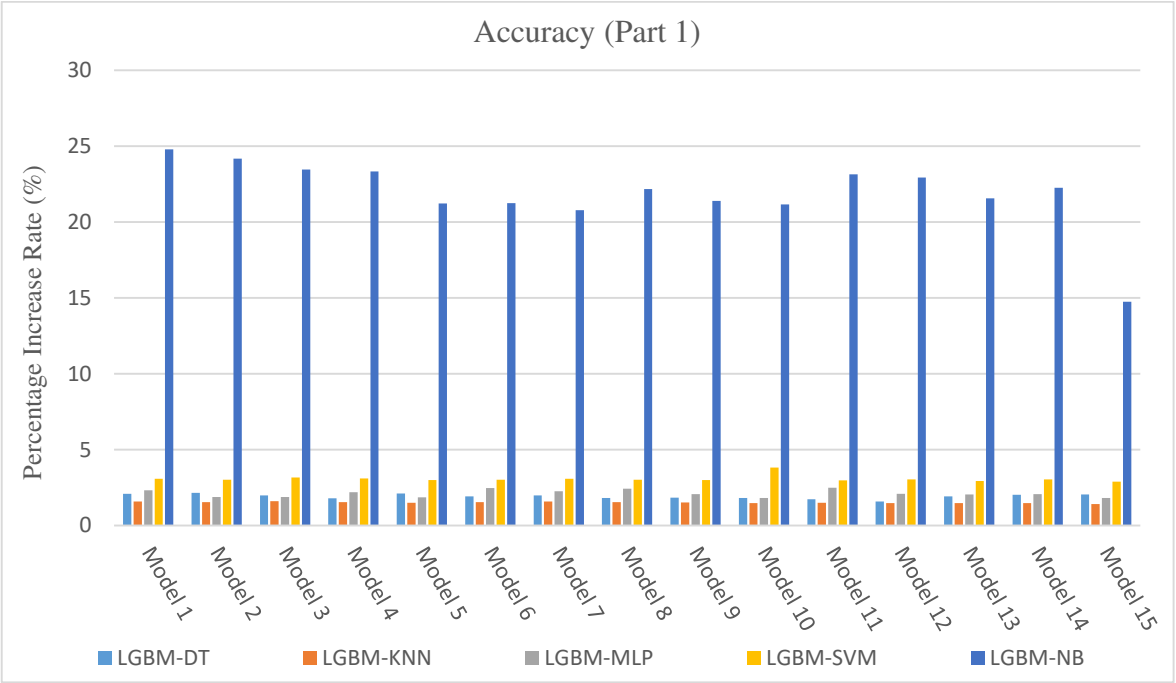
Table 5.26. Precision values for Relief-F-based predictor variables (Part 2)

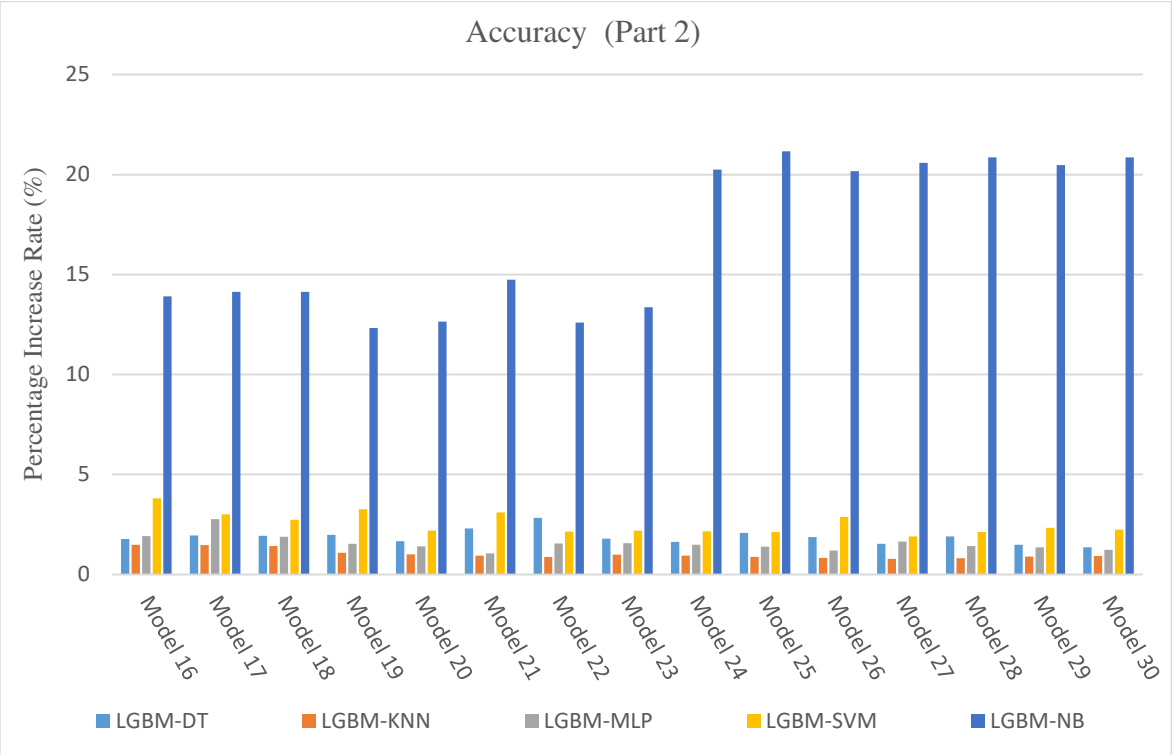
Model	Precision (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 19	98.37	96.55	97.02	96.72	94.34	90.18
Model 20	98.26	96.80	97.05	96.96	97.53	89.41
Model 21	98.22	96.32	97.05	97.11	94.44	79.99
Model 22	98.15	96.65	97.05	97.15	97.52	88.55
Model 23	98.26	96.57	97.05	96.97	97.46	**91.04**
Model 24	98.33	96.48	97.05	97.21	97.46	74.28
Model 25	98.11	96.68	97.05	97.31	97.49	72.47
Model 26	97.72	96.82	96.98	**97.57**	94.54	73.94
Model 27	97.57	96.61	96.98	97.21	97.50	73.14
Model 28	97.68	96.74	96.84	96.92	96.25	72.17
Model 29	97.57	96.75	96.65	96.63	96.72	72.60
Model 30	97.28	97.03	96.37	96.62	96.50	71.24
Model 31	97.31	96.72	96.37	96.21	96.47	71.14
Model 32	97.17	96.85	96.37	96.59	96.50	71.54
Model 33	97.63	96.99	96.30	96.07	96.44	82.35
Model 34	97.74	96.86	96.23	96.68	95.61	72.02
Model 35	97.45	96.80	96.23	96.23	95.68	82.24
Model 36	97.42	96.16	96.20	96.18	95.65	83.64
Model 37	97.48	96.82	96.19	96.37	96.14	79.74
Model 38	97.66	96.84	96.12	95.91	96.40	84.07
Model 39	97.41	96.91	96.16	96.36	96.50	79.55
Model 40	97.44	96.70	95.97	95.48	95.50	80.81
Model 41	97.20	95.49	96.27	95.21	95.61	76.11
Model 42	97.30	96.55	96.44	93.53	94.25	66.49
Model 43	97.33	96.45	96.40	93.12	94.49	72.03
Model 44	96.98	96.75	96.00	89.44	94.90	82.20
Model 45	95.05	95.05	95.36	87.63	94.14	81.94
Model 46	95.00	94.60	94.76	91.21	92.91	60.68
Model 47	94.34	94.15	94.39	88.98	92.87	60.78
Model 48	93.78	93.80	93.95	90.77	92.05	74.40
Model 49	93.54	93.48	93.37	90.48	91.38	77.92
Model 50	80.96	80.86	87.17	62.45	78.54	57.65
Model 51	74.91	74.92	86.62	57.81	71.81	51.77

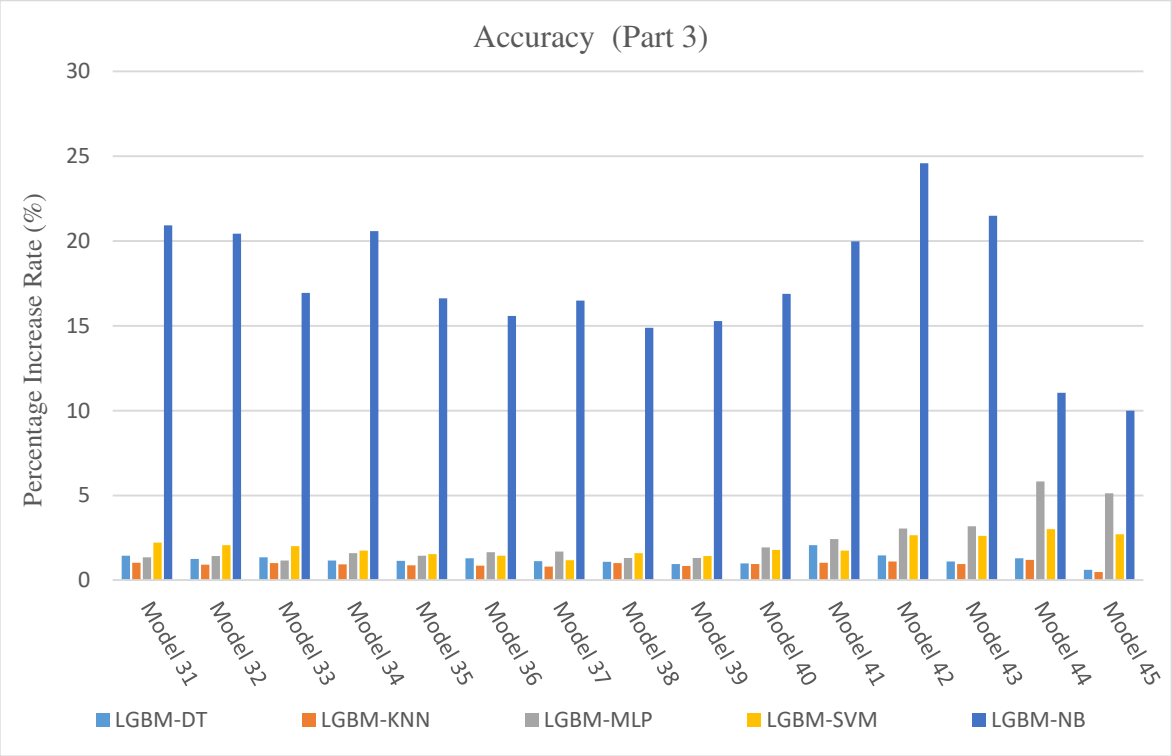
(a) LightGBM (b) DT (c) KNN

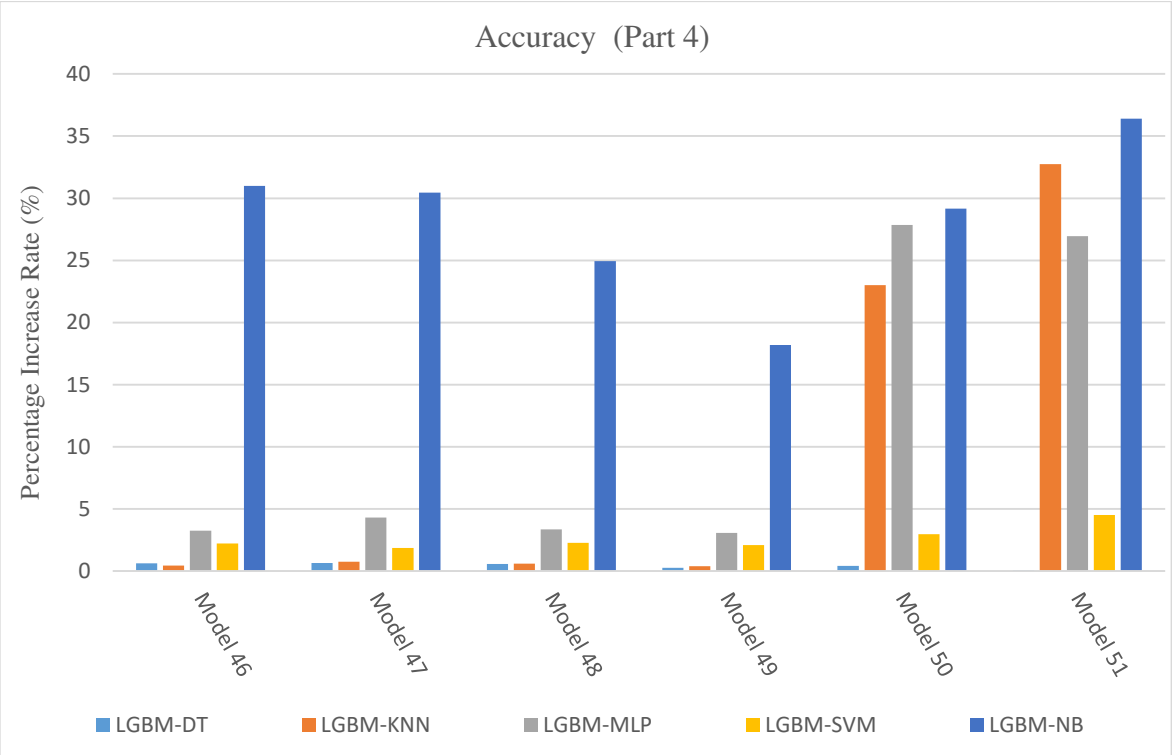
(d) MLP (e) SVM (f) NB

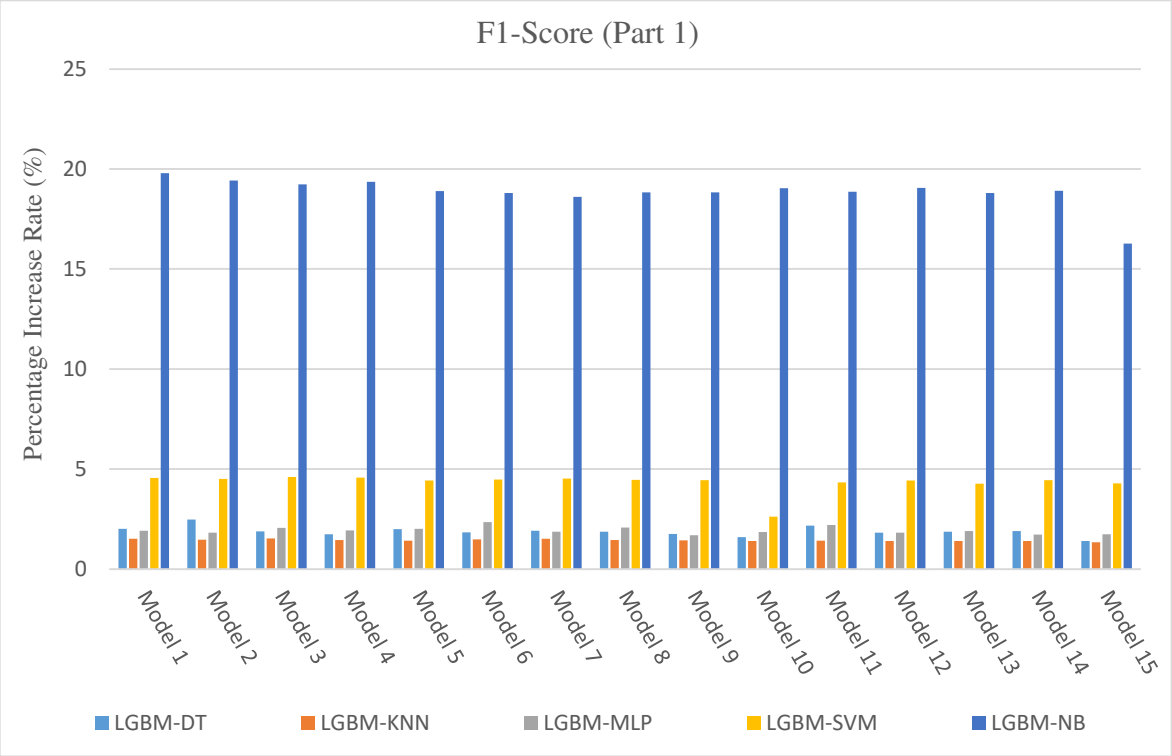
Figure 5.3. Confusion matrices for Relief-F-Based models utilizing optimized parameters. (a) LightGBM, (b) DT, (c) KNN, (d) MLP, (e) SVM, and (f) NB

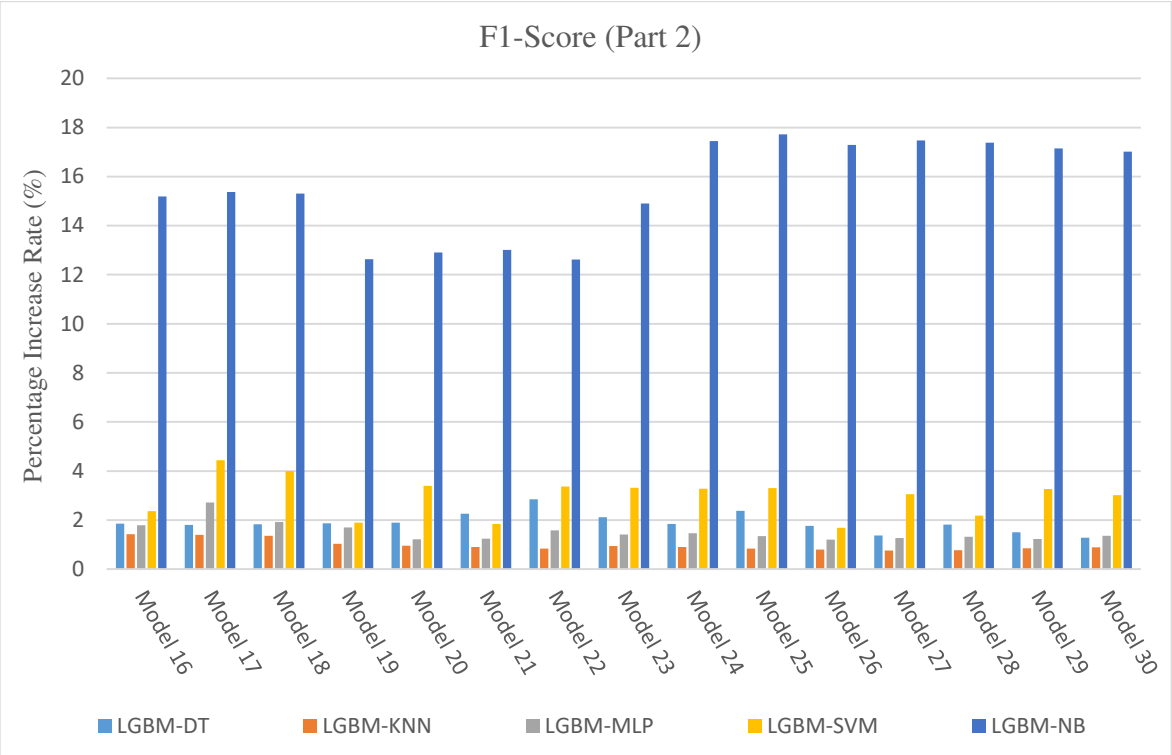


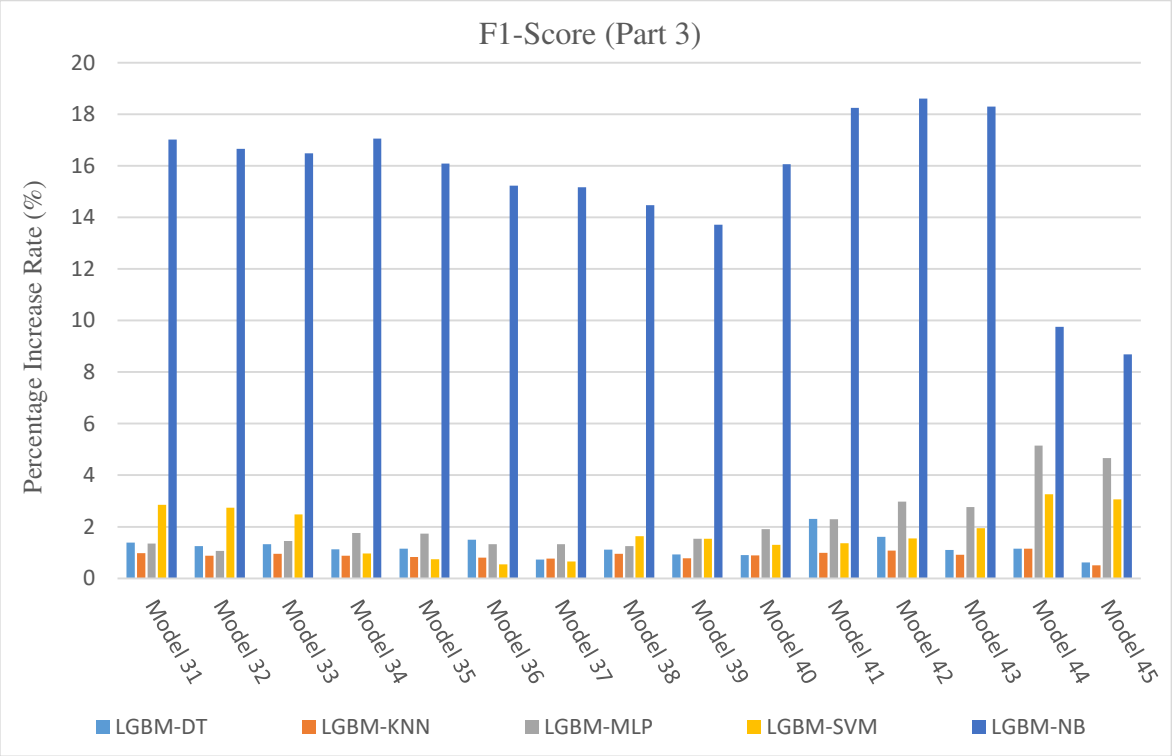

Accuracy (Part 2)
Percentage Increase Rate (%)
25
20
15
10
5
0
Model 16
Model 17
Model 18
Model 19
Model 20
Model 21
Model 22
Model 23
Model 24
Model 25
Model 26
Model 27
Model 28
Model 29
Model 30
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

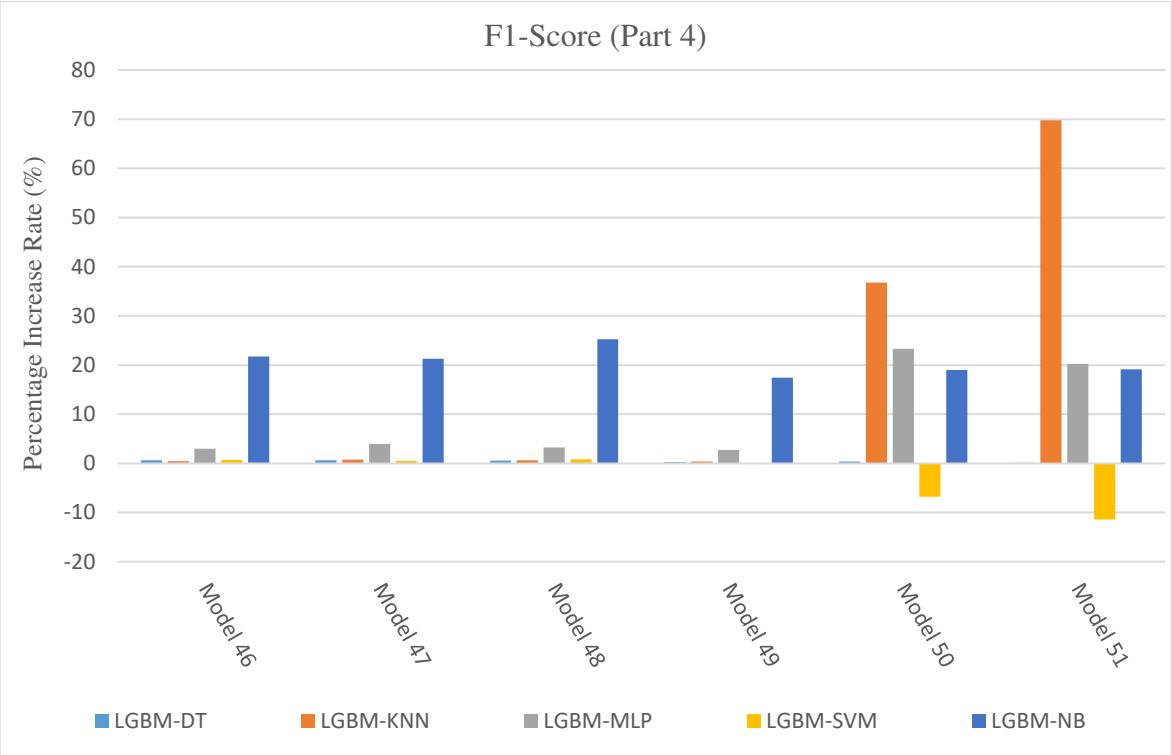

Accuracy (Part 3)
Percentage Increase Rate (%)
30
25
20
15
10
5
0
Model 31
Model 32
Model 33
Model 34
Model 35
Model 36
Model 37
Model 38
Model 39
Model 40
Model 41
Model 42
Model 43
Model 44
Model 45
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

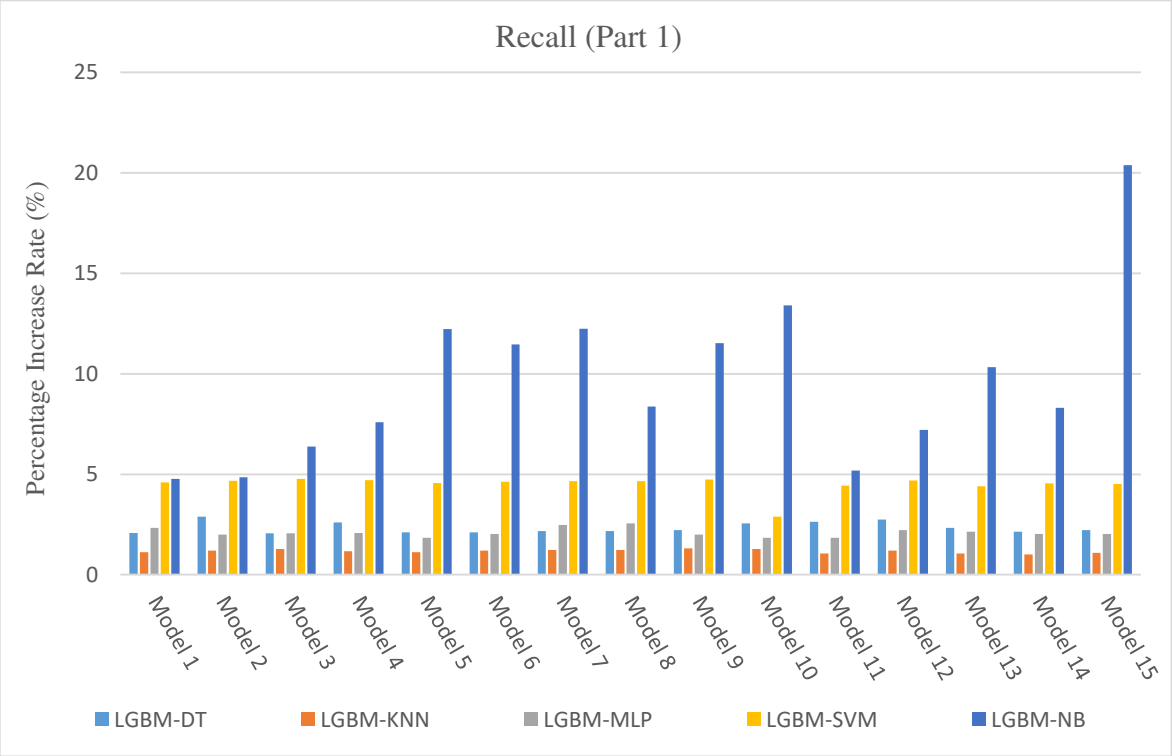

Accuracy (Part 4)
Percentage Increase Rate (%)
40
35
30
25
20
15
10
5
0
Model 46
Model 47
Model 48
Model 49
Model 50
Model 51
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

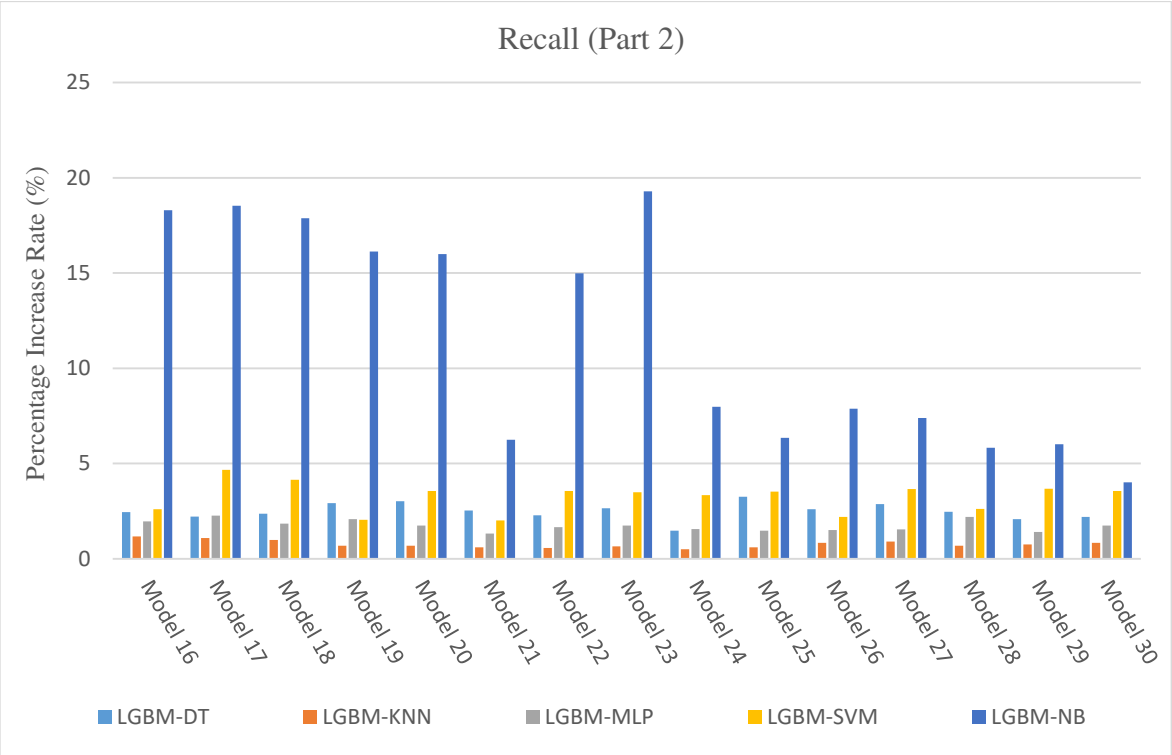

F1-Score (Part 1)
Percentage Increase Rate (%)
25
20
15
10
5
0
Model 1
Model 2
Model 3
Model 4
Model 5
Model 6
Model 7
Model 8
Model 9
Model 10
Model 11
Model 12
Model 13
Model 14
Model 15
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

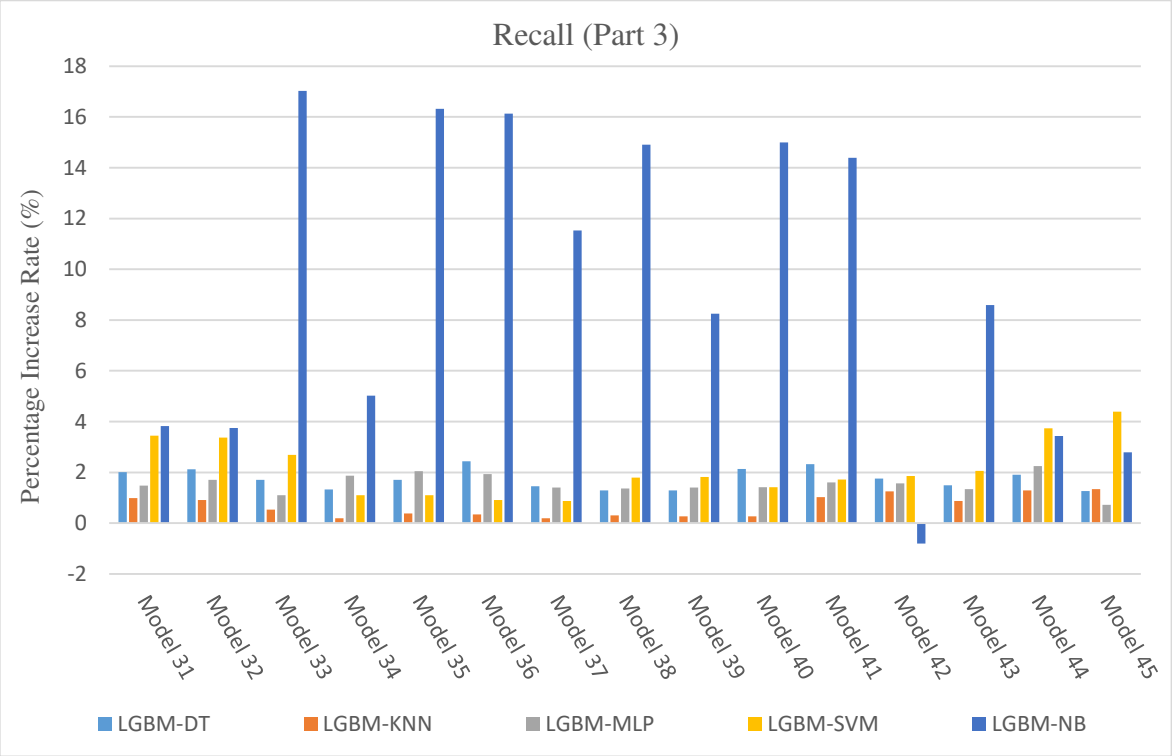

F1-Score (Part 2)
Percentage Increase Rate (%)
20
18
16
14
12
10
8
6
4
2
0
Model 16
Model 17
Model 18
Model 19
Model 20
Model 21
Model 22
Model 23
Model 24
Model 25
Model 26
Model 27
Model 28
Model 29
Model 30
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

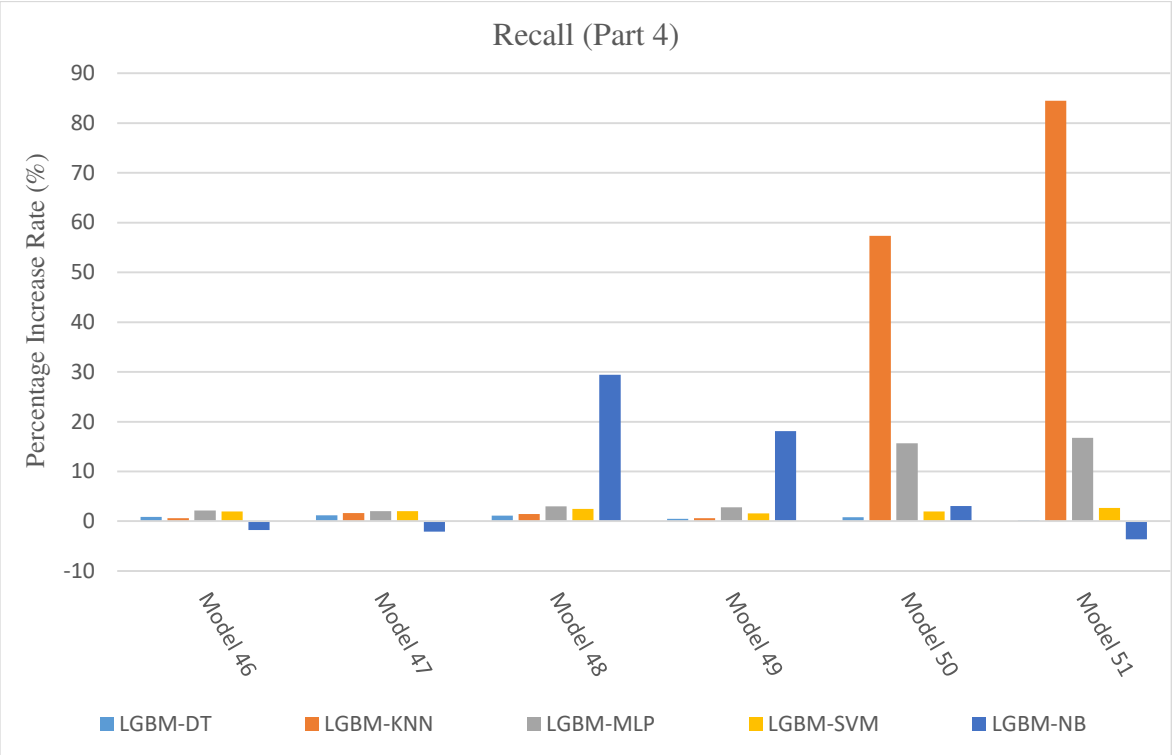

F1-Score (Part 3)
Percentage Increase Rate (%)
20
18
16
14
12
10
8
6
4
2
0
Model 31
Model 32
Model 33
Model 34
Model 35
Model 36
Model 37
Model 38
Model 39
Model 40
Model 41
Model 42
Model 43
Model 44
Model 45
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

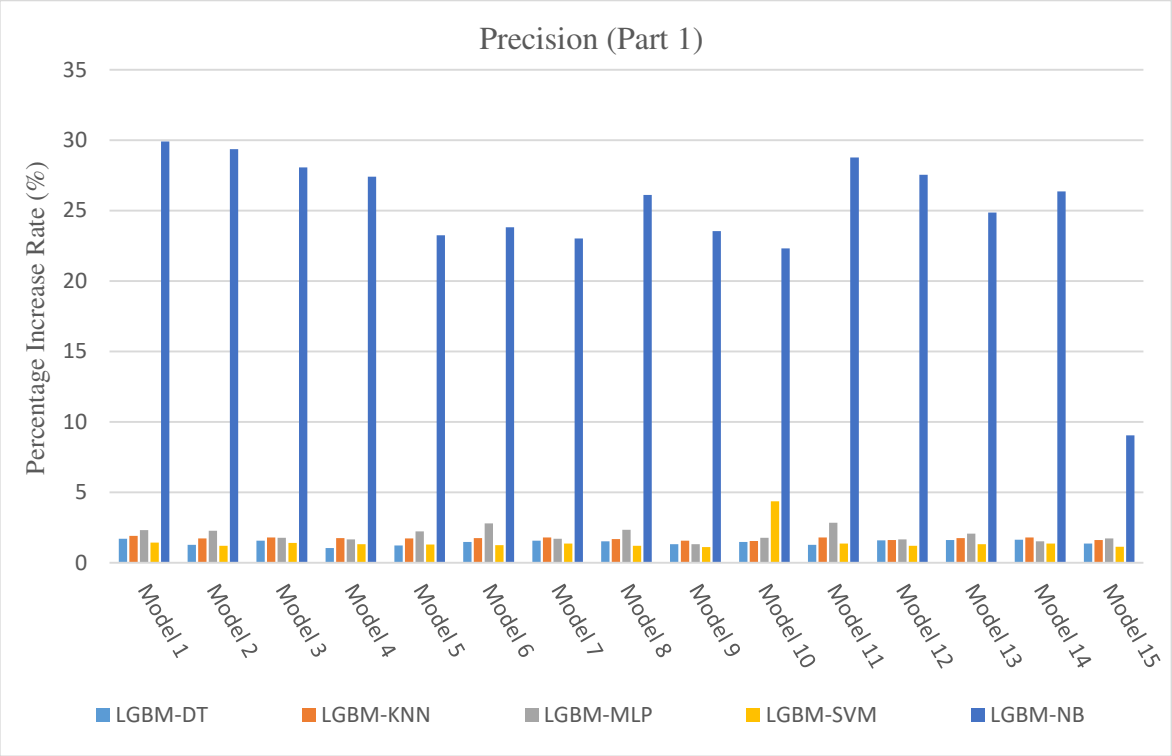

F1-Score (Part 4)
Percentage Increase Rate (%)
80
70
60
50
40
30
20
10
0
-10
-20
Model 46
Model 47
Model 48
Model 49
Model 50
Model 51
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

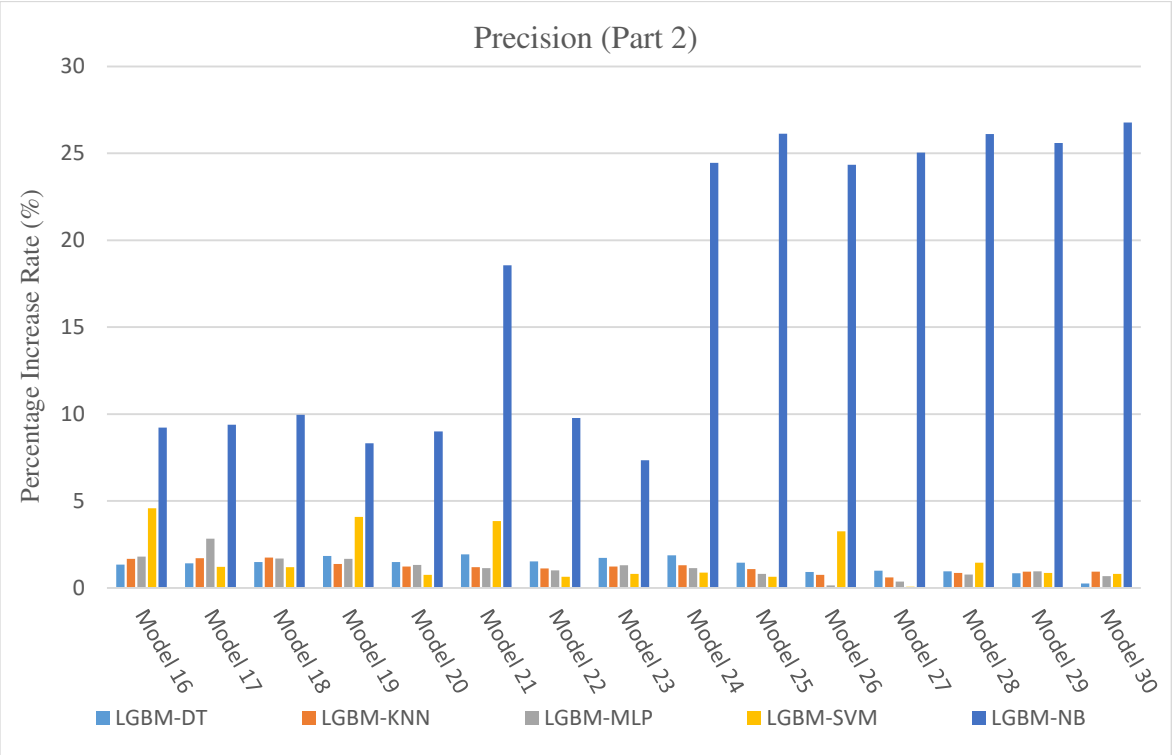

Recall (Part 1)
Percentage Increase Rate (%)
25
20
15
10
5
0
Model 1
Model 2
Model 3
Model 4
Model 5
Model 6
Model 7
Model 8
Model 9
Model 10
Model 11
Model 12
Model 13
Model 14
Model 15
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

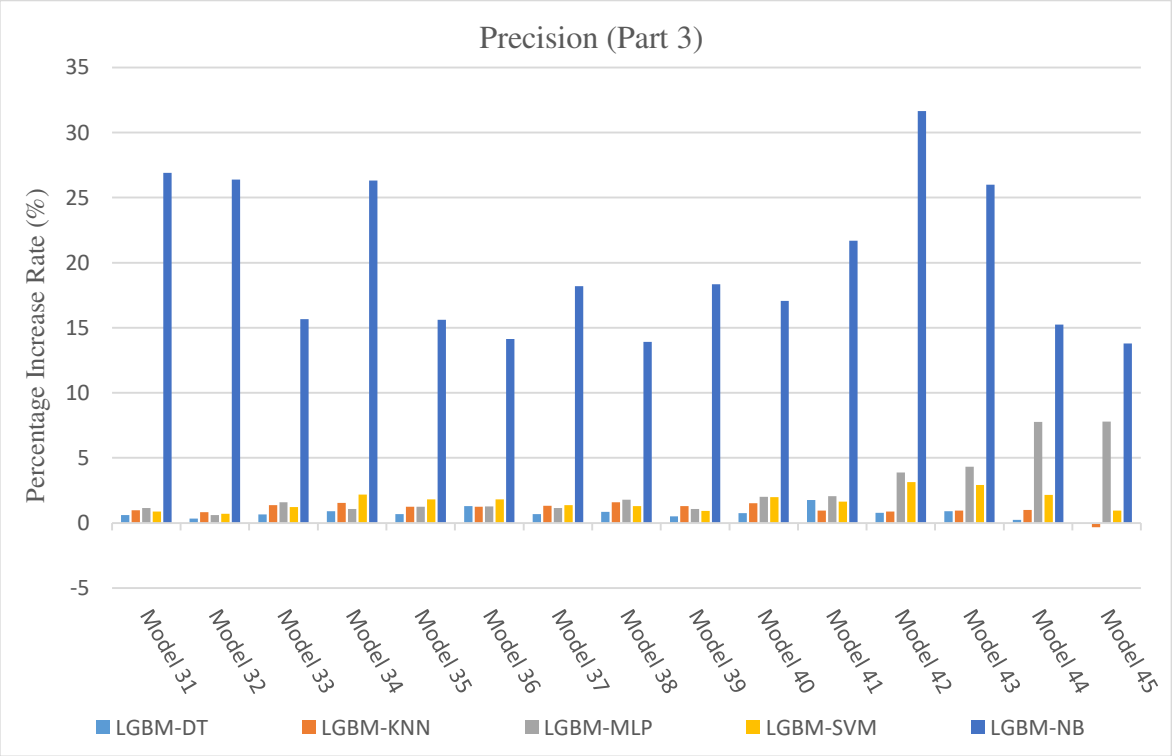

Recall (Part 2)
Percentage Increase Rate (%)
25
20
15
10
5
0
Model 16
Model 17
Model 18
Model 19
Model 20
Model 21
Model 22
Model 23
Model 24
Model 25
Model 26
Model 27
Model 28
Model 29
Model 30
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB


Recall (Part 3)
Percentage Increase Rate (%)
18
16
14
12
10
8
6
4
2
0
-2
Model 31
Model 32
Model 33
Model 34
Model 35
Model 36
Model 37
Model 38
Model 39
Model 40
Model 41
Model 42
Model 43
Model 44
Model 45
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB


Recall (Part 4)
Percentage Increase Rate (%)
90
80
70
60
50
40
30
20
10
0
-10
Model 46
Model 47
Model 48
Model 49
Model 50
Model 51
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB


Precision (Part 1)
Percentage Increase Rate (%)
35
30
25
20
15
10
5
0
Model 1
Model 2
Model 3
Model 4
Model 5
Model 6
Model 7
Model 8
Model 9
Model 10
Model 11
Model 12
Model 13
Model 14
Model 15
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB


Precision (Part 2)
Percentage Increase Rate (%)
30
25
20
15
10
5
0
Model 16
Model 17
Model 18
Model 19
Model 20
Model 21
Model 22
Model 23
Model 24
Model 25
Model 26
Model 27
Model 28
Model 29
Model 30
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB


Precision (Part 3)
Percentage Increase Rate (%)
35
30
25
20
15
10
5
0
-5
Model 31
Model 32
Model 33
Model 34
Model 35
Model 36
Model 37
Model 38
Model 39
Model 40
Model 41
Model 42
Model 43
Model 44
Model 45
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

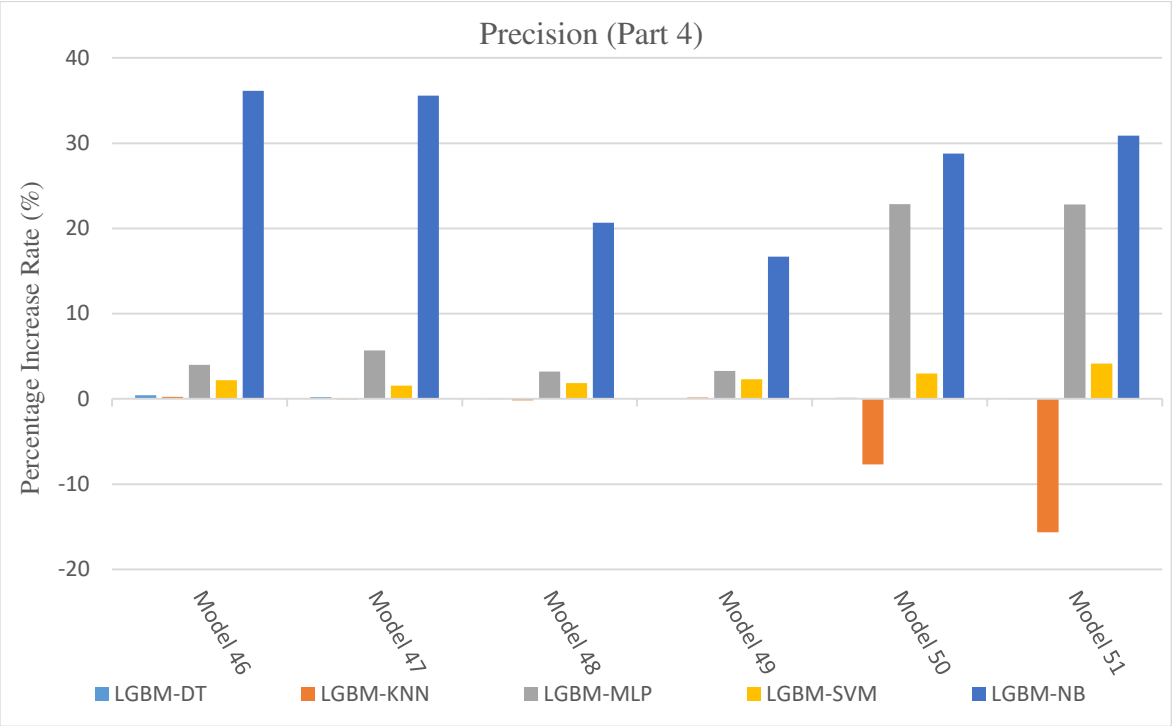


Figure 5.4. Percentage Increase Rates for Relief-F models including Accuracy, F1-Score, Recall, as well as Precision

5.3.2. Results of mRMR-Based Models

In this section, models created based on mRMR are applied with the best parameter values resulting from hyperparameter optimization of ML algorithms. Evaluation results, encompassing accuracy, F1-Score, recall, precision criteria, and confusion matrices, are comprehensively presented. These results provide a detailed overview of the outputs obtained from the algorithms and models.

Table 5.27. Accuracy values for mRMR-based predictor variables (Part 1)

Model	Accuracy (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 52	**98.94**	96.93	97.40	97.05	95.89	74.86
Model 53	98.88	96.68	97.41	96.93	95.95	74.92
Model 54	98.90	97.14	97.41	96.99	95.83	75.29
Model 55	98.88	97.05	97.41	96.64	95.95	76.10
Model 56	**98.94**	96.86	97.41	96.84	95.91	76.50
Model 57	**98.94**	**97.41**	97.41	96.76	95.91	75.83
Model 58	98.86	96.87	97.38	96.57	95.97	76.02
Model 59	98.86	96.95	97.38	96.53	95.89	76.02
Model 60	98.86	96.20	97.38	96.89	95.97	76.35
Model 61	98.88	97.03	97.38	96.82	95.95	76.87
Model 62	98.84	96.93	97.38	97.01	95.93	77.14
Model 63	98.32	96.55	97.28	97.05	95.08	76.25
Model 64	98.34	95.74	97.34	96.76	95.99	77.41

Table 5.28. Accuracy values for mRMR-based predictor variables (Part 2)

Model	Accuracy (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 65	98.28	95.89	97.38	96.78	96.01	78.09
Model 66	98.26	96.60	97.38	97.07	95.10	78.05
Model 67	98.28	96.53	97.38	96.89	96.14	77.89
Model 68	98.32	96.55	97.38	96.91	95.99	77.91
Model 69	98.34	95.56	97.36	97.09	96.14	78.28
Model 70	98.21	95.33	97.36	96.93	96.12	77.76
Model 71	98.28	96.41	97.36	96.72	95.35	78.12
Model 72	98.23	96.41	97.34	96.82	96.22	78.68
Model 73	98.15	96.32	97.36	96.97	96.16	77.99
Model 74	98.21	96.37	97.34	97.07	96.06	77.85
Model 75	98.23	96.20	97.34	96.68	96.18	78.38
Model 76	98.21	95.85	97.34	96.99	96.22	78.26
Model 77	98.17	96.26	97.34	96.87	96.24	78.20
Model 78	98.21	96.55	97.34	96.91	96.10	77.66
Model 79	98.28	96.22	97.47	96.91	**96.41**	77.57
Model 80	98.13	96.35	**97.49**	97.11	96.35	77.89
Model 81	98.26	96.82	97.47	**97.18**	95.56	77.91
Model 82	98.13	95.39	97.20	96.70	95.99	76.49
Model 83	97.90	95.99	96.93	96.55	95.79	76.91
Model 84	97.51	95.51	96.59	96.30	95.66	77.06
Model 85	97.36	95.27	96.60	96.39	95.83	78.07
Model 86	97.07	95.14	96.32	96.03	95.79	83.01
Model 87	96.41	94.89	95.43	95.49	95.04	83.45
Model 88	96.16	94.21	95.41	94.98	94.79	84.01
Model 89	96.03	94.54	95.47	95.08	94.83	86.34
Model 90	95.29	94.54	94.50	94.62	94.56	83.99
Model 91	94.66	94.10	93.73	94.08	93.92	82.62
Model 92	94.62	93.94	93.87	93.94	93.94	86.90
Model 93	94.54	94.17	93.88	92.79	93.81	87.71
Model 94	94.41	93.96	93.38	92.71	93.67	**87.92**
Model 95	94.04	93.63	91.18	91.98	93.25	87.56
Model 96	94.00	93.62	91.13	92.84	93.44	85.07
Model 97	94.02	93.73	91.07	91.26	93.50	86.81
Model 98	93.96	93.31	86.40	93.02	93.50	87.56
Model 99	92.82	92.38	82.93	90.12	92.52	80.28
Model 100	88.85	88.58	68.34	85.11	88.43	83.72
Model 101	77.12	76.49	60.92	74.34	76.99	74.34
Model 102	76.04	76.02	50.17	74.02	75.66	73.98

Table 5.29. F1-Score values for mRMR-based predictor variables (Part 1)

Model	F1-Score (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 52	**98.98**	97.11	97.50	96.97	94.48	79.46
Model 53	98.92	97.23	97.51	97.07	94.60	79.45
Model 54	98.94	97.12	97.51	97.24	94.41	79.80
Model 55	98.92	96.75	97.51	96.88	94.60	80.01
Model 56	**98.98**	96.72	97.51	97.06	94.56	80.38
Model 57	**98.98**	96.93	97.51	97.08	94.56	80.28
Model 58	98.90	97.09	97.48	96.64	94.63	80.41
Model 59	98.90	97.07	97.48	96.70	94.52	80.23
Model 60	98.90	**97.25**	97.48	97.08	94.63	80.45
Model 61	98.92	96.90	97.48	97.04	94.56	80.33
Model 62	98.88	97.13	97.48	96.81	94.56	80.40
Model 63	98.38	96.85	97.38	97.00	96.42	80.39
Model 64	98.40	95.76	97.44	96.94	94.60	80.57
Model 65	98.35	96.74	97.48	97.05	94.63	80.94
Model 66	98.33	96.42	97.48	96.99	96.83	80.92
Model 67	98.34	96.56	97.48	96.69	94.82	80.72
Model 68	98.38	96.43	97.48	97.11	94.60	80.68
Model 69	98.40	95.96	97.46	96.98	94.82	80.95
Model 70	98.27	96.09	97.46	97.25	94.78	80.79
Model 71	98.35	96.45	97.46	**97.34**	96.64	80.92
Model 72	98.29	96.43	97.44	97.15	94.97	81.22
Model 73	98.22	96.43	97.46	97.26	94.97	80.95
Model 74	98.27	96.56	97.44	96.96	94.89	80.83
Model 75	98.29	96.22	97.44	96.85	95.08	81.07
Model 76	98.28	95.87	97.44	97.03	95.15	80.97
Model 77	98.24	96.53	97.44	97.10	95.19	80.77
Model 78	98.27	96.61	97.44	97.01	95.01	80.85
Model 79	98.35	96.50	97.57	96.94	95.64	80.79
Model 80	98.20	96.01	**97.59**	97.17	95.53	81.09
Model 81	98.33	96.52	97.57	97.22	96.76	81.15
Model 82	98.20	95.95	97.31	96.91	95.27	80.44
Model 83	97.97	96.39	97.06	96.69	95.16	80.84
Model 84	97.60	95.20	96.72	96.21	97.13	81.06
Model 85	97.46	95.70	96.73	96.25	95.56	81.57
Model 86	97.18	95.48	96.45	96.43	95.68	84.09
Model 87	96.56	93.84	95.64	95.58	96.53	83.76
Model 88	96.31	94.91	95.61	95.33	96.50	84.22
Model 89	96.19	94.82	95.67	95.39	96.57	86.04
Model 90	95.52	94.76	94.79	95.09	96.79	84.27
Model 91	94.95	94.39	94.08	94.35	96.61	83.49
Model 92	94.93	94.36	94.23	94.16	97.24	86.77
Model 93	94.85	94.51	94.25	93.16	97.06	87.72

Table 5.30. F1-Score values for mRMR-based predictor variables (Part 2)

Model	F1-Score (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 94	94.71	94.17	93.73	93.02	96.87	87.89
Model 95	94.45	94.07	91.44	92.63	**97.39**	87.89
Model 96	94.41	94.00	91.37	93.21	97.02	86.08
Model 97	94.43	93.96	91.29	91.89	97.06	87.41
Model 98	94.39	93.83	85.86	93.53	97.24	**87.99**
Model 99	93.36	92.91	81.82	90.85	97.32	83.89
Model 100	90.00	89.74	59.36	86.46	96.53	84.84
Model 101	81.35	81.10	42.58	79.44	95.90	79.50
Model 102	80.53	80.51	7.48	79.30	95.57	79.32

Table 5.31. Recall values for mRMR-based predictor variables (Part 1)

Model	Recall (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 52	99.14	96.83	97.91	96.87	94.48	93.18
Model 53	**99.33**	96.72	97.95	96.87	94.60	93.03
Model 54	99.18	96.72	97.95	97.02	94.41	93.48
Model 55	99.25	96.61	97.95	97.06	94.60	92.18
Model 56	99.22	96.72	97.95	97.02	94.56	92.73
Model 57	99.14	96.27	97.95	97.09	94.56	94.37
Model 58	99.14	96.98	97.88	97.28	94.63	94.37
Model 59	99.11	96.98	97.88	97.24	94.52	93.41
Model 60	99.03	96.91	97.88	97.20	94.63	93.44
Model 61	99.14	96.64	97.88	96.83	94.56	90.91
Model 62	99.07	96.91	97.88	97.09	94.56	90.17
Model 63	98.58	96.20	97.73	96.68	96.42	93.56
Model 64	98.81	95.75	97.76	96.72	94.60	90.43
Model 65	98.70	96.20	97.88	96.98	94.63	90.09
Model 66	98.77	96.65	97.88	97.28	96.83	90.06
Model 67	98.62	95.90	97.88	96.94	94.82	89.69
Model 68	98.70	96.61	97.88	96.91	94.60	89.35
Model 69	98.62	95.30	97.88	96.79	94.82	89.57
Model 70	98.51	95.64	97.88	96.76	94.78	90.50
Model 71	98.66	96.20	97.88	96.94	96.64	89.72
Model 72	98.66	96.27	97.88	**97.50**	94.97	89.27
Model 73	98.51	96.24	97.91	96.87	94.97	90.73
Model 74	98.55	96.42	97.91	97.24	94.89	90.50
Model 75	98.62	95.82	97.91	97.13	95.08	89.83
Model 76	98.92	96.20	97.91	97.28	95.15	89.65
Model 77	98.55	96.38	97.91	97.09	95.19	89.05
Model 78	98.55	96.05	97.91	96.98	95.01	91.44
Model 79	98.55	96.42	**97.99**	96.27	95.64	91.40

Table 5.32. Recall values for mRMR-based predictor variables (Part 2)

Model	Recall (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 80	98.51	96.46	98.02	97.06	95.53	91.84
Model 81	98.84	96.57	**97.99**	96.79	96.76	92.07
Model 82	98.47	96.01	97.76	96.79	95.27	93.48
Model 83	98.17	94.78	97.69	96.87	95.16	94.11
Model 84	97.80	95.60	97.09	96.91	97.13	94.78
Model 85	97.73	95.42	97.17	96.01	95.56	93.70
Model 86	97.65	95.53	96.61	96.42	95.68	86.96
Model 87	97.32	94.30	96.94	96.79	96.53	82.63
Model 88	96.87	95.34	96.68	96.61	96.50	82.60
Model 89	96.83	95.27	96.72	96.72	96.57	81.44
Model 90	96.87	95.94	96.57	96.79	96.79	82.97
Model 91	96.98	96.27	96.16	96.65	96.61	84.98
Model 92	97.35	96.50	96.79	96.91	97.24	83.04
Model 93	96.98	96.87	96.72	94.63	97.06	84.87
Model 94	96.79	96.35	95.82	94.37	96.87	84.72
Model 95	97.95	**97.69**	91.31	95.30	**97.39**	87.29
Model 96	97.84	97.54	91.20	96.79	97.02	88.90
Model 97	97.88	97.43	90.83	95.34	97.06	88.45
Model 98	97.99	97.39	79.87	96.91	97.24	88.07
Model 99	97.43	96.80	75.36	94.86	97.32	98.66
Model 100	96.87	96.39	44.99	91.61	96.53	88.04
Model 101	96.24	94.19	28.10	95.75	95.90	95.98
Model 102	95.64	95.60	3.91	96.09	95.57	**96.31**

Table 5.33. Precision values for mRMR-based predictor variables (Part 1)

Model	Precision (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 52	**98.82**	97.27	97.08	97.13	97.51	70.18
Model 53	98.52	97.31	97.09	97.24	97.52	70.23
Model 54	98.71	**97.69**	97.09	97.11	97.47	70.40
Model 55	98.59	97.11	97.09	97.08	97.52	71.47
Model 56	98.74	96.95	97.09	96.95	97.48	71.63
Model 57	98.81	97.12	97.09	**97.43**	97.48	70.35
Model 58	98.67	97.13	97.08	96.23	97.52	70.59
Model 59	98.70	96.95	97.08	96.32	97.48	70.95
Model 60	98.78	97.20	97.08	96.90	97.52	71.20
Model 61	98.70	97.38	97.08	97.17	97.55	73.02
Model 62	98.70	97.17	97.08	97.14	97.52	73.61
Model 63	98.18	97.26	97.04	97.24	94.22	71.18
Model 64	98.01	96.09	97.12	97.03	97.59	73.73
Model 65	98.00	96.86	97.08	97.29	97.59	74.36

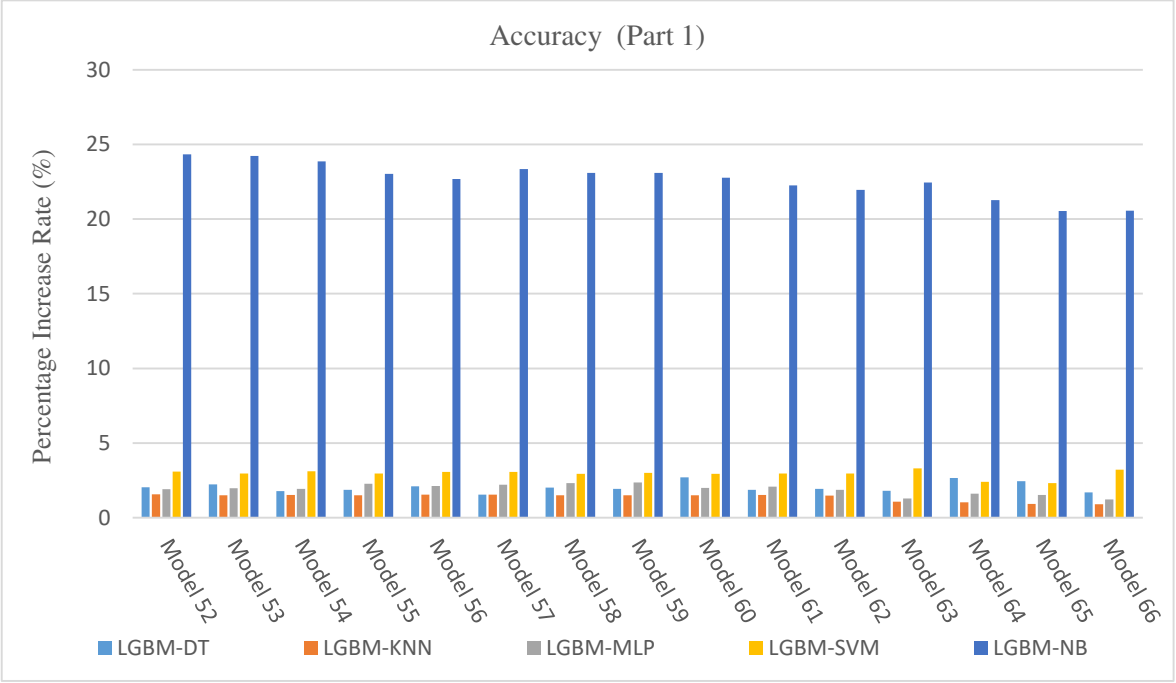
Table 5.34. Precision values for mRMR-based predictor variables (Part 2)

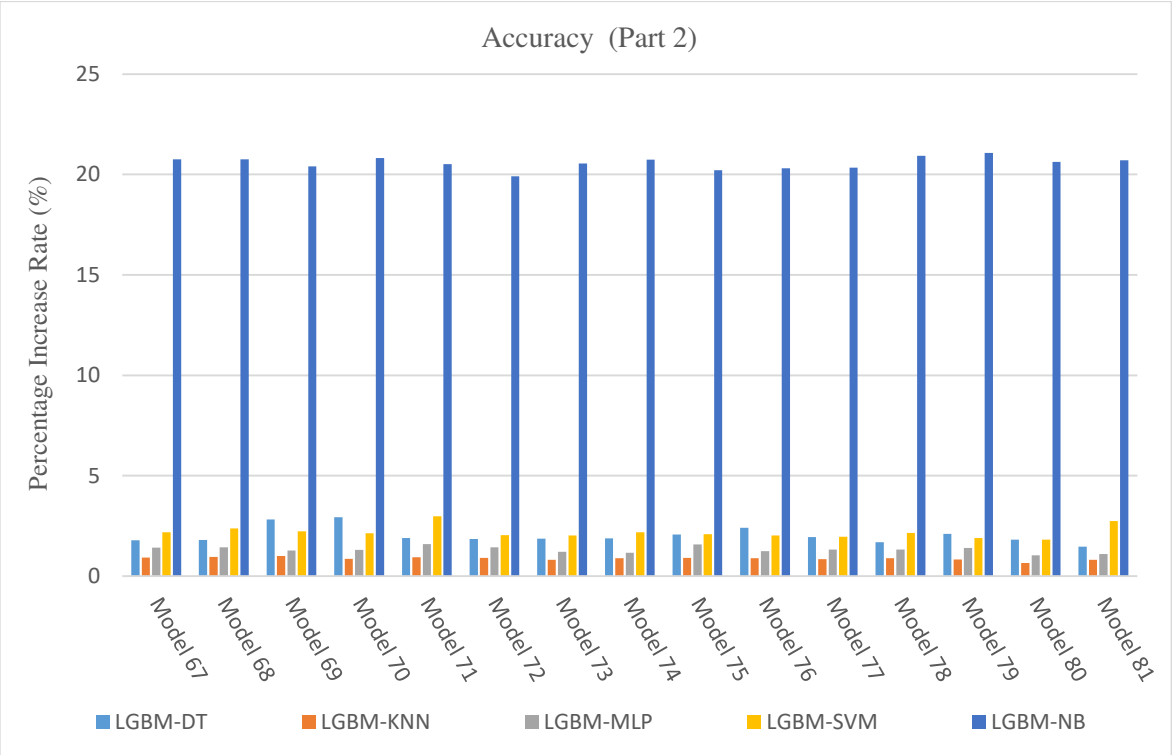
Model	Precision (%)					
	LightGBM	**DT**	**KNN**	**MLP**	**SVM**	**NB**
Model 66	97.90	96.86	97.08	97.01	93.91	74.36
Model 67	98.07	97.00	97.08	97.02	97.67	74.54
Model 68	98.08	96.51	97.08	97.23	97.59	74.71
Model 69	98.18	95.97	97.05	97.35	97.67	74.88
Model 70	98.03	95.46	97.05	96.80	97.67	73.86
Model 71	98.04	96.64	97.05	97.22	94.51	74.55
Model 72	97.93	96.55	97.01	96.79	**97.68**	75.30
Model 73	97.93	96.76	97.01	96.89	97.56	73.63
Model 74	98.00	96.70	96.98	97.07	97.45	73.52
Model 75	97.97	96.77	96.98	96.15	97.49	74.34
Model 76	97.65	96.56	96.98	97.29	97.49	74.28
Model 77	97.93	97.01	96.98	97.01	97.50	74.54
Model 78	98.00	96.72	96.98	97.25	97.42	72.90
Model 79	98.15	97.33	**97.16**	96.98	97.40	72.70
Model 80	97.89	96.98	**97.16**	97.18	97.40	72.89
Model 81	97.83	96.87	**97.16**	97.17	94.80	72.80
Model 82	97.93	96.67	96.87	96.92	96.90	70.68
Model 83	97.78	95.89	96.44	96.93	96.68	70.89
Model 84	97.41	95.78	96.35	95.66	94.64	70.83
Model 85	97.19	95.92	96.31	96.36	96.36	72.24
Model 86	96.72	96.12	96.29	95.94	96.19	81.54
Model 87	95.83	95.05	94.38	94.48	94.06	84.97
Model 88	95.77	94.23	94.59	94.16	93.66	85.95
Model 89	95.56	94.26	94.66	93.87	93.67	91.25
Model 90	94.23	93.31	93.10	93.58	93.02	85.65
Model 91	93.02	92.64	92.10	92.09	92.07	82.08
Model 92	92.65	92.35	91.83	91.71	91.59	90.91
Model 93	92.82	92.29	91.92	91.94	91.52	90.82
Model 94	92.75	92.14	91.79	91.74	91.44	**91.36**
Model 95	91.21	90.75	91.66	89.96	90.35	88.55
Model 96	91.24	90.80	91.65	90.37	90.94	83.59
Model 97	91.24	90.72	91.86	88.69	91.01	86.46
Model 98	91.06	90.54	92.88	90.04	90.87	87.94
Model 99	89.64	89.47	90.05	87.17	89.22	73.09
Model 100	84.06	83.97	88.12	81.74	83.66	81.90
Model 101	70.50	70.37	88.88	67.95	70.46	67.87
Model 102	69.58	69.57	95.21	67.54	69.22	67.45

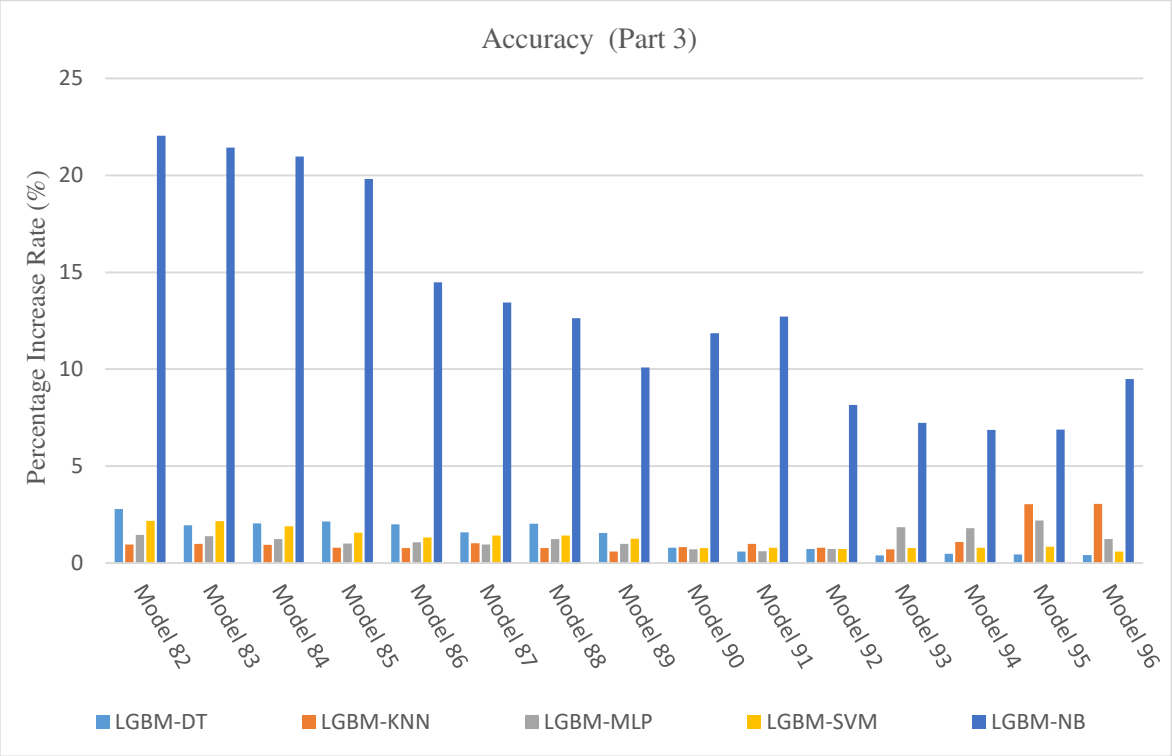
(a) LightGBM (b) DT (c) KNN

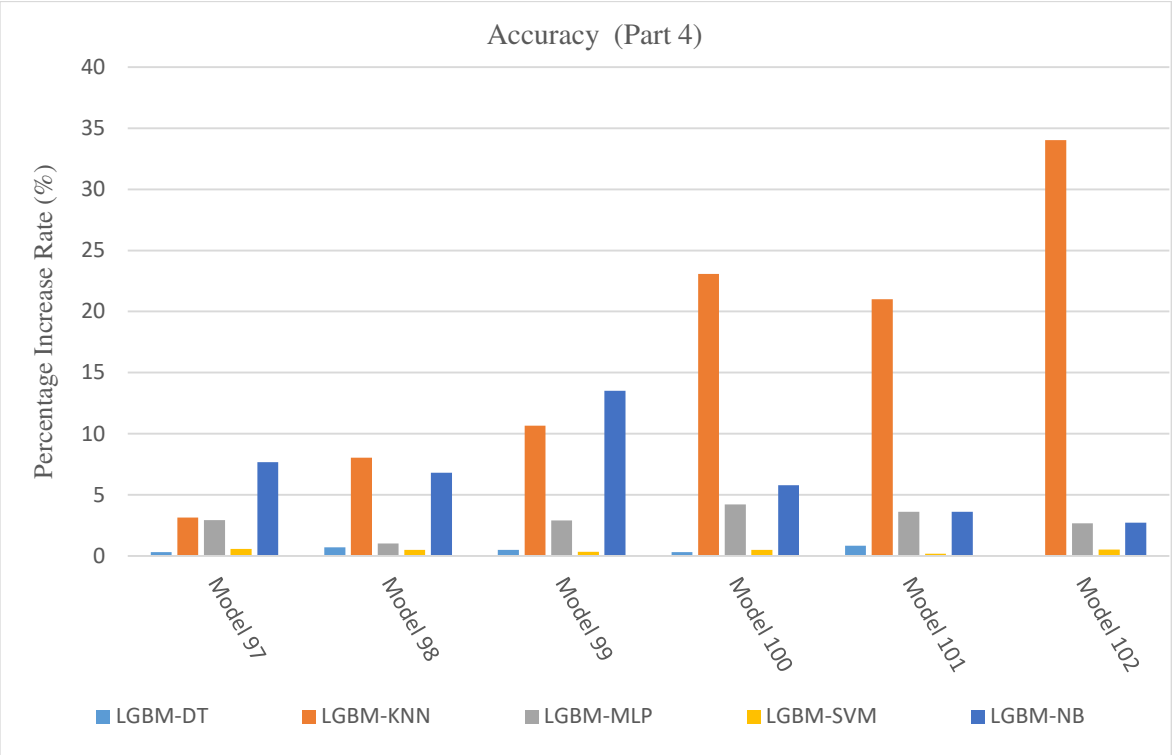
(d) MLP (e) SVM (f) NB

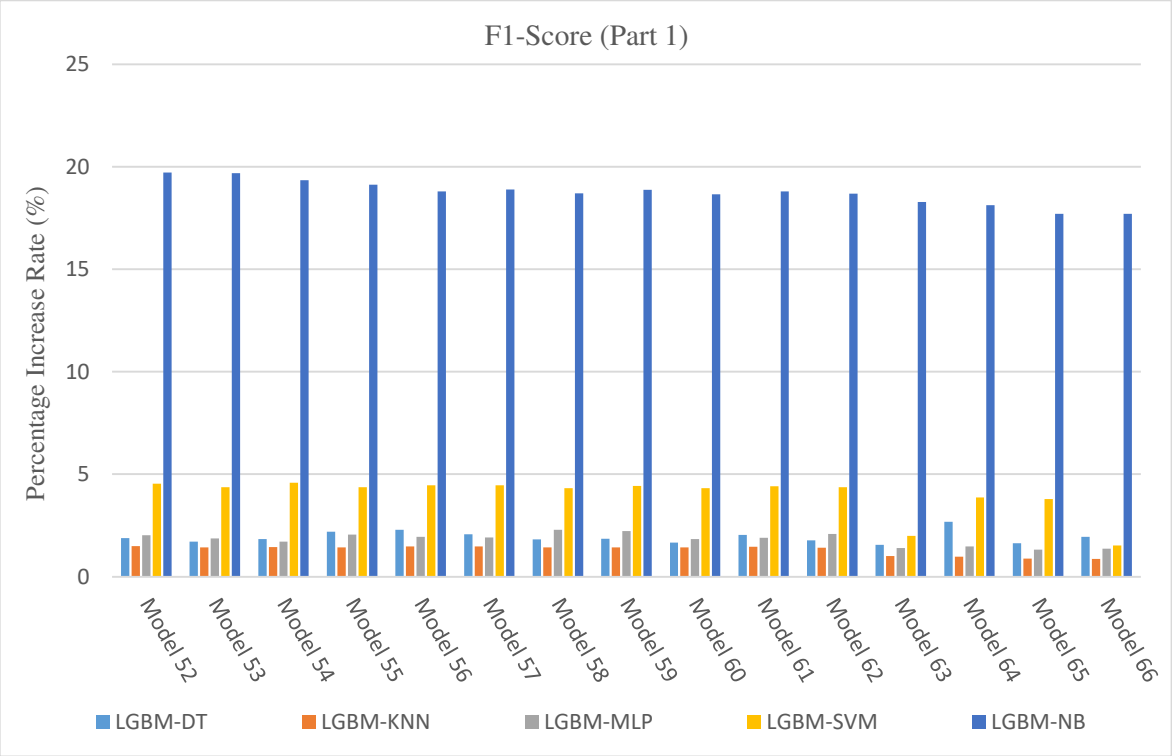
Figure 5.5. Confusion matrices of mRMR-Based models using optimized parameters. (a) LightGBM, (b) DT, (c) KNN, (d) MLP, (e) SVM, and (f) NB

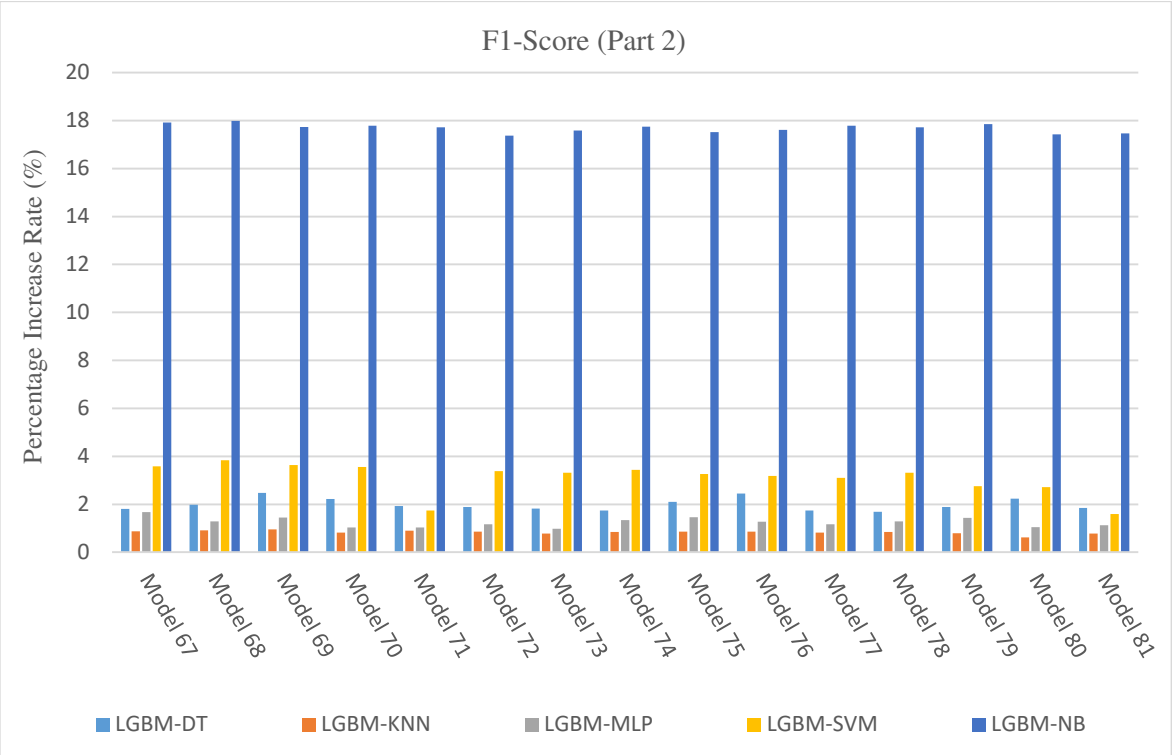


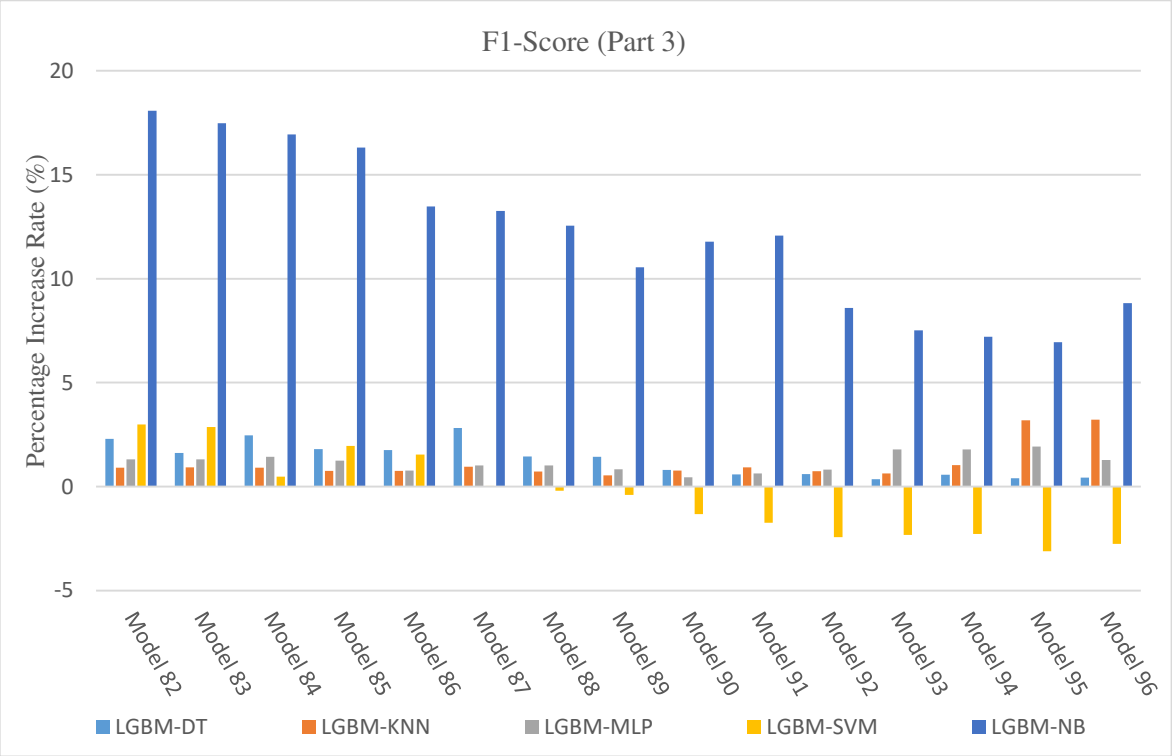

Accuracy (Part 2)
Percentage Increase Rate (%)
25
20
15
10
5
0
Model 67
Model 68
Model 69
Model 70
Model 71
Model 72
Model 73
Model 74
Model 75
Model 76
Model 77
Model 78
Model 79
Model 80
Model 81
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

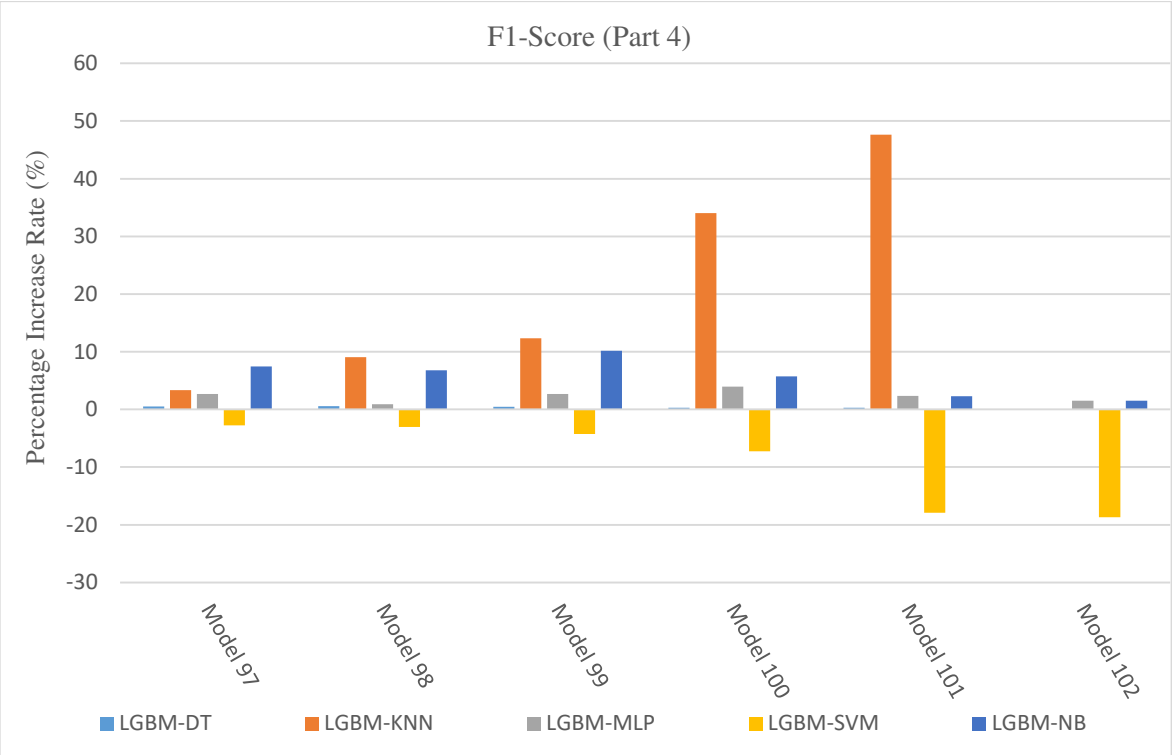

Accuracy (Part 3)
Percentage Increase Rate (%)
25
20
15
10
5
0
Model 82
Model 83
Model 84
Model 85
Model 86
Model 87
Model 88
Model 89
Model 90
Model 91
Model 92
Model 93
Model 94
Model 95
Model 96
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

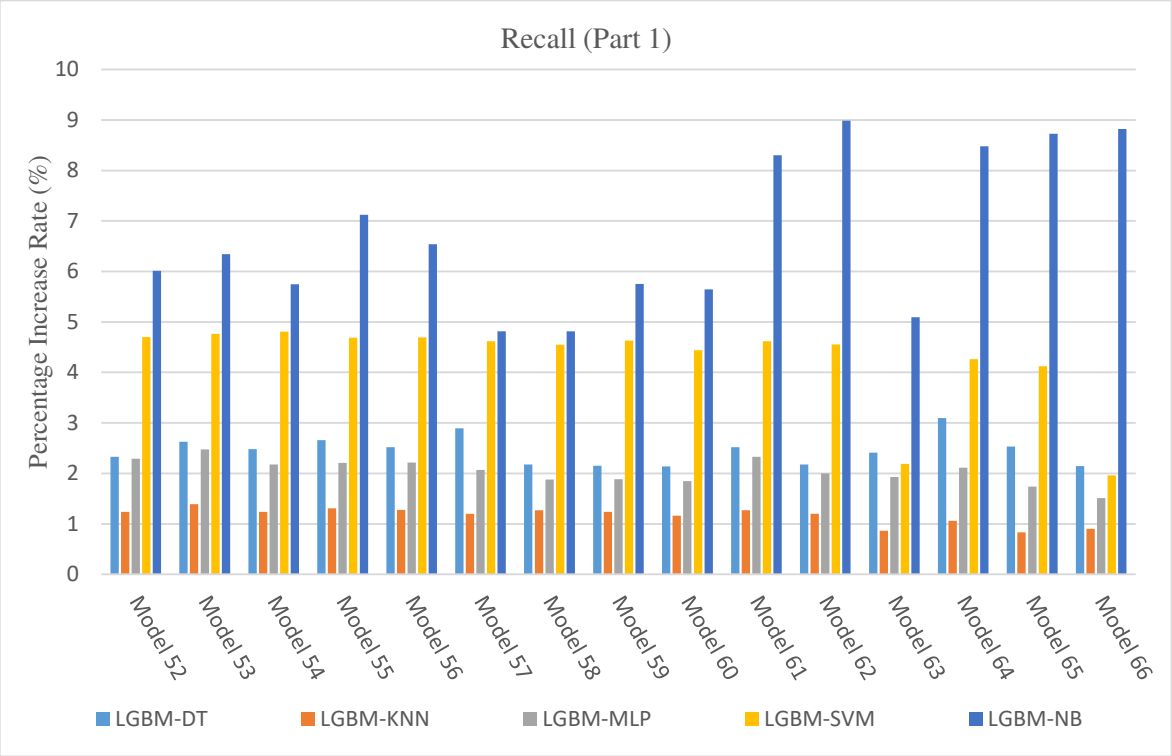

Accuracy (Part 4)
Percentage Increase Rate (%)
40
35
30
25
20
15
10
5
0
Model 97
Model 98
Model 99
Model 100
Model 101
Model 102
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

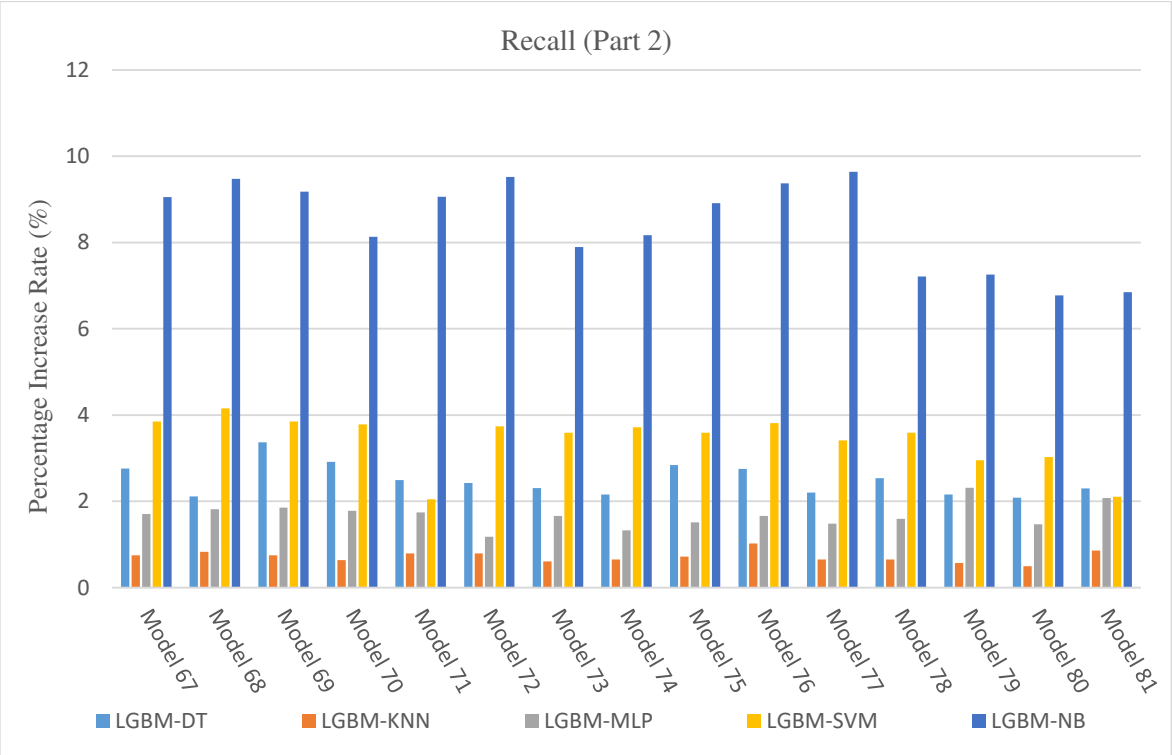

F1-Score (Part 1)
Percentage Increase Rate (%)
25
20
15
10
5
0
Model 52
Model 53
Model 54
Model 55
Model 56
Model 57
Model 58
Model 59
Model 60
Model 61
Model 62
Model 63
Model 64
Model 65
Model 66
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

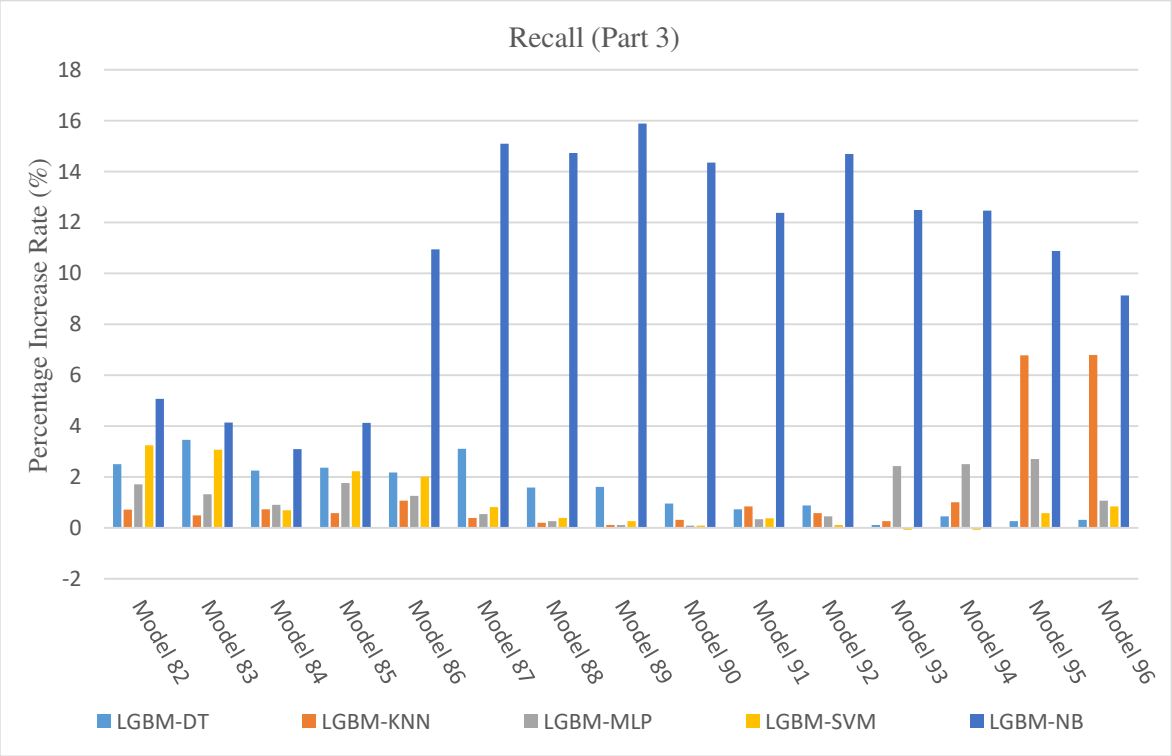

F1-Score (Part 2)
Percentage Increase Rate (%)
20
18
16
14
12
10
8
6
4
2
0
Model 67
Model 68
Model 69
Model 70
Model 71
Model 72
Model 73
Model 74
Model 75
Model 76
Model 77
Model 78
Model 79
Model 80
Model 81
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

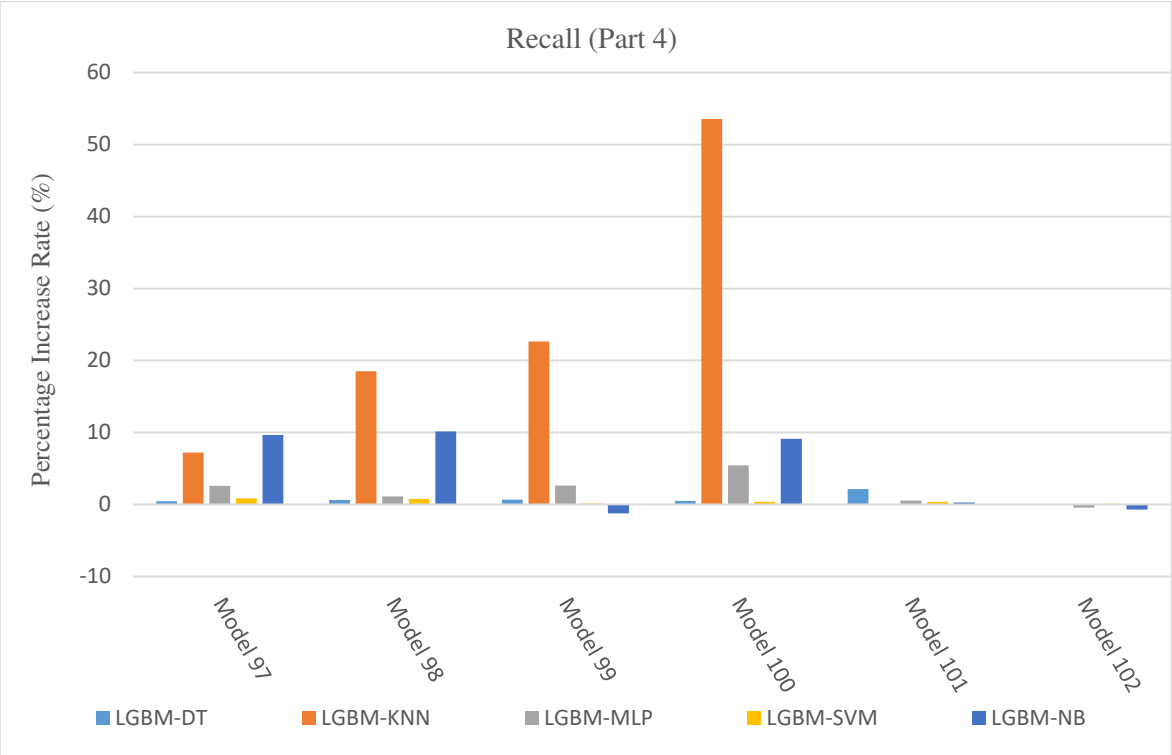

F1-Score (Part 3)
Percentage Increase Rate (%)
20
15
10
5
0
-5
Model 82
Model 83
Model 84
Model 85
Model 86
Model 87
Model 88
Model 89
Model 90
Model 91
Model 92
Model 93
Model 94
Model 95
Model 96
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

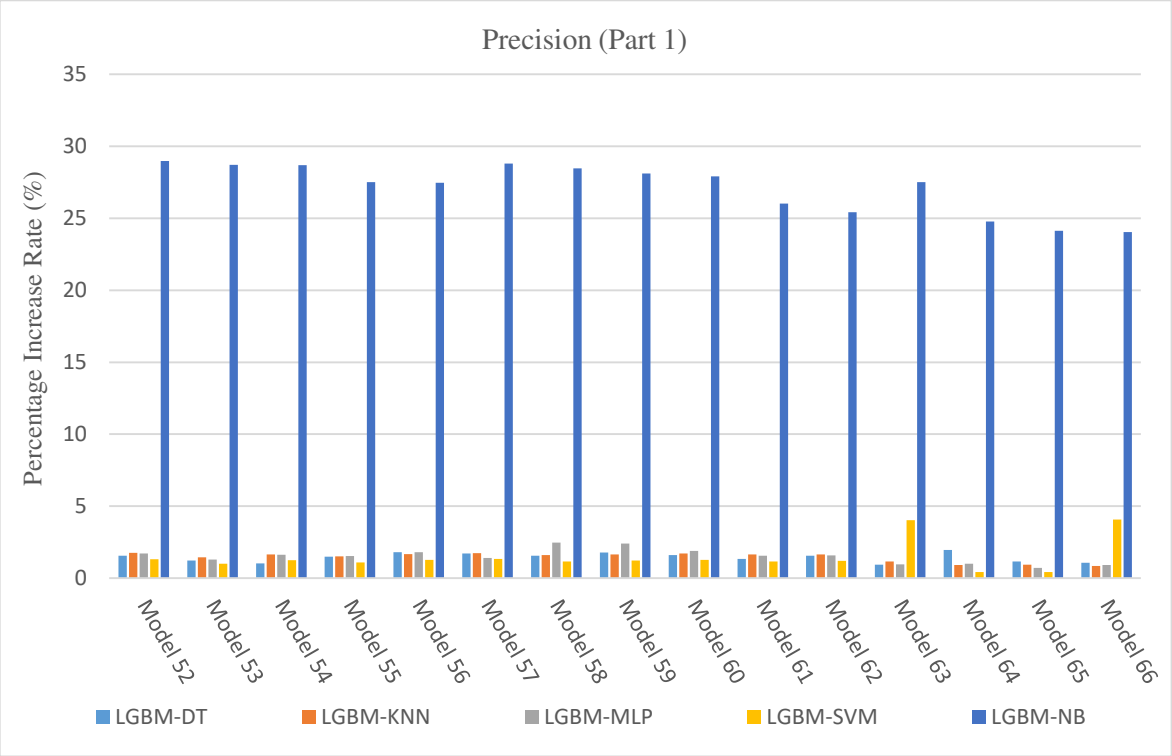

F1-Score (Part 4)
Percentage Increase Rate (%)
60
50
40
30
20
10
0
-10
-20
-30
Model 97
Model 98
Model 99
Model 100
Model 101
Model 102
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

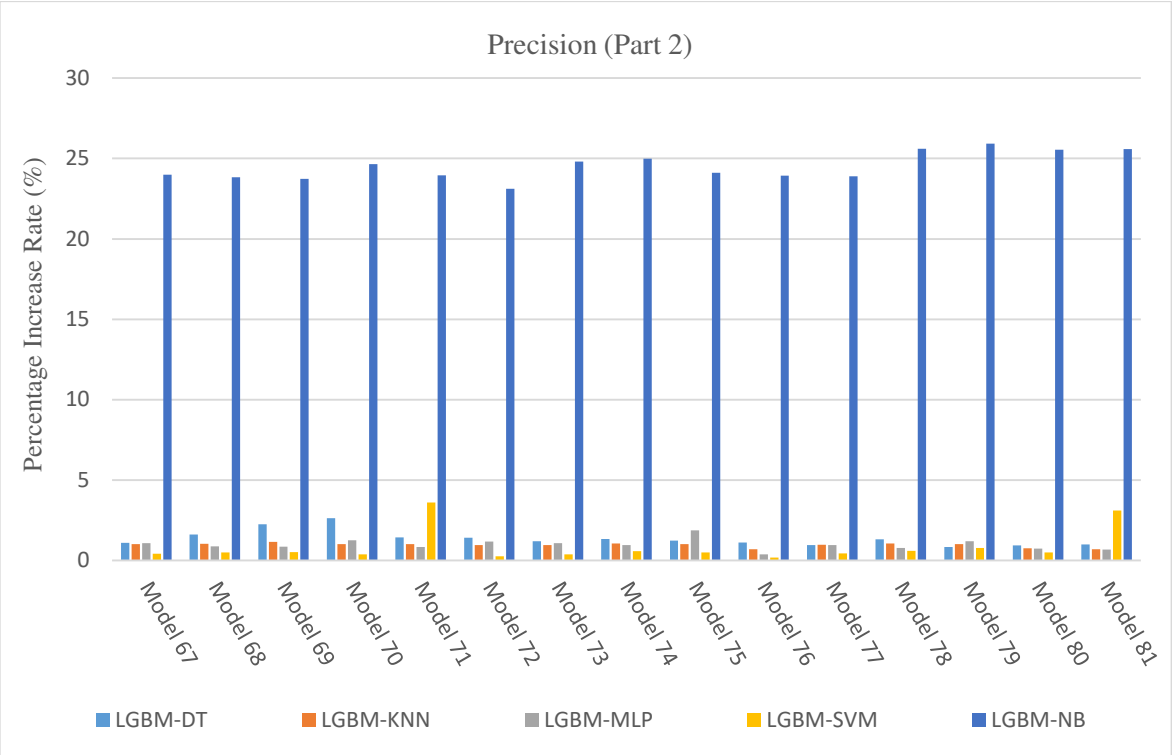

Recall (Part 1)
Percentage Increase Rate (%)
10
9
8
7
6
5
4
3
2
1
0
Model 52
Model 53
Model 54
Model 55
Model 56
Model 57
Model 58
Model 59
Model 60
Model 61
Model 62
Model 63
Model 64
Model 65
Model 66
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

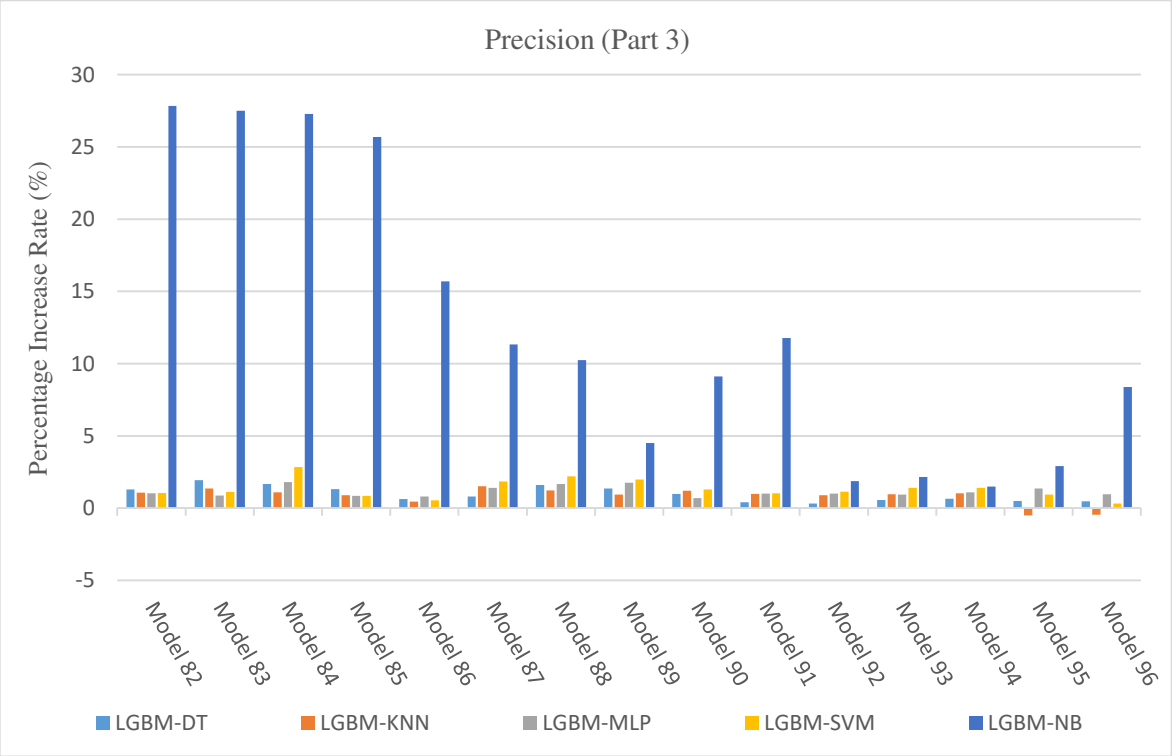

Recall (Part 2)
Percentage Increase Rate (%)
12
10
8
6
4
2
0
Model 67
Model 68
Model 69
Model 70
Model 71
Model 72
Model 73
Model 74
Model 75
Model 76
Model 77
Model 78
Model 79
Model 80
Model 81
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB


Recall (Part 3)
Percentage Increase Rate (%)
18
16
14
12
10
8
6
4
2
0
-2
Model 82
Model 83
Model 84
Model 85
Model 86
Model 87
Model 88
Model 89
Model 90
Model 91
Model 92
Model 93
Model 94
Model 95
Model 96
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB


Recall (Part 4)
Percentage Increase Rate (%)
60
50
40
30
20
10
0
-10
Model 97
Model 98
Model 99
Model 100
Model 101
Model 102
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB


Precision (Part 1)
Percentage Increase Rate (%)
35
30
25
20
15
10
5
0
Model 52
Model 53
Model 54
Model 55
Model 56
Model 57
Model 58
Model 59
Model 60
Model 61
Model 62
Model 63
Model 64
Model 65
Model 66
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB


Precision (Part 2)
Percentage Increase Rate (%)
30
25
20
15
10
5
0
Model 67
Model 68
Model 69
Model 70
Model 71
Model 72
Model 73
Model 74
Model 75
Model 76
Model 77
Model 78
Model 79
Model 80
Model 81
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB


Precision (Part 3)
Percentage Increase Rate (%)
30
25
20
15
10
5
0
-5
Model 82
Model 83
Model 84
Model 85
Model 86
Model 87
Model 88
Model 89
Model 90
Model 91
Model 92
Model 93
Model 94
Model 95
Model 96
LGBM-DT
LGBM-KNN
LGBM-MLP
LGBM-SVM
LGBM-NB

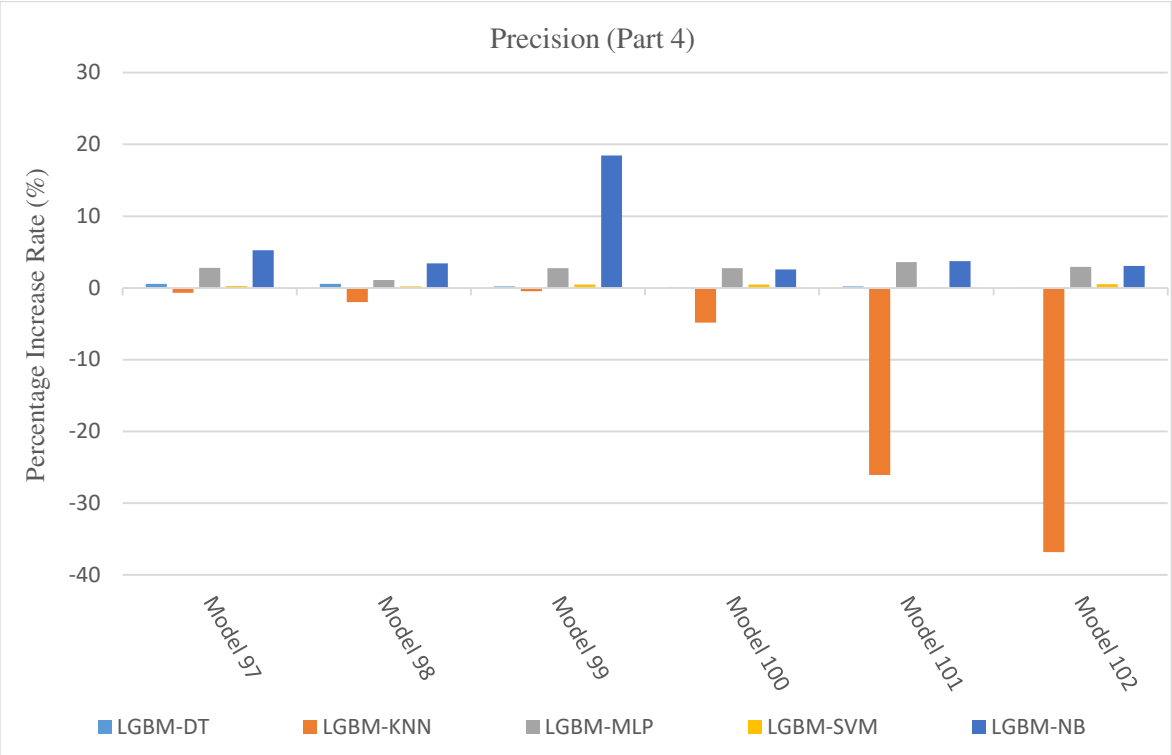


Figure 5.6. Percentage Increase Rates for mRMR models including Accuracy, F1-Score, Recall, as well as Precision

5.4. Discussion of Models with Optimized Parameters

This section offers an in-depth analysis of the results obtained and emphasizes the overall significance of algorithms of ML hyperparameter-tuned configurations. Specifically, the findings derived from Relief-F and mRMR feature selection methods comprehensively elucidate the potential impact on the effectiveness and applicability of these techniques.

5.4.1. Discussion of Relief-F-Based Models

Examining models containing the PE headers created using Relief-F feature selection and evaluated with hyperparameter tuning of ML classifiers is as follows:

- When the average accuracy values of the models created by feature selection are examined, it is understood that the LightGBM algorithm shows quite high accuracy compared to other methods. The DT method exhibits a 3.08% lower accuracy rate than LightGBM, while the KNN algorithm offers a 3.54% lower accuracy value. Additionally, the MLP shows 4.47% lower accuracy, while the SVM achieves 4.06% less accuracy, and the NB exhibits a significantly lower accuracy of 21.02% compared to the LightGBM. These findings highlight the effectiveness of using Relief-F feature selection in conjunction with the LightGBM algorithm, especially when working with

datasets containing the PE headers. In addition to highlighting the relative superiority of the LightGBM among these algorithms, this evaluation reveals that feature selection methods and hyperparameter tuning of ML algorithms directly contribute to the performance of models.

- When considering models classified with optimized values, the LightGBM demonstrates the highest accuracy, followed by KNN, DT, MLP, SVM, and NB respectively. Upon analyzing Relief-F-based models, it is observed that different models exhibit high accuracy for each applied method. Within the scope of the LightGBM method, upon examining methods and their corresponding models, model number 3 achieves the highest accuracy. For the KNN, models numbered from 13 to 18 with the same accuracy levels achieve the highest accuracy, while model 12 performs best for the DT. Model 21 exhibits the highest accuracy for MLP, model 37 for SVM, and model 44 for NB. This comprehensive analysis emphasizes variability in model performance among different methodologies and highlights specific models that excel in their respective classification techniques.
- When analyzing the average F1-Score values of models created through feature selection, it becomes evident that the LightGBM algorithm exhibits a higher F1-Score compared to other methods. While the DT method shows a 1.52% lower F1-Score rate than LightGBM, the KNN algorithm shows 2.84% less F1-Score. Additionally, MLP presents a 2.67% lower F1-Score, SVM shows 2.40% less F1-Score, and NB significantly records a 17.06% lower F1-Score compared to LightGBM.
- When considering models classified with optimized values, LightGBM shows the highest F1-Score, followed by KNN, DT, MLP, SVM, and NB respectively. Upon analyzing Relief-F-based models, it is observed that different models exhibit high F1-Score for each applied method. Within the scope of the LightGBM method, model number 3 achieves the highest F1-Score. For KNN, models numbered from 13 to 18 with the same F1-Score levels achieve the highest F1-Score, while for DT, model 15 performs the best. Models from 9 to 14 yield the highest F1-Score for MLP, all displaying same results. For SVM, model 36, and for NB, model 45 provide the highest F1-Score. This comprehensive analysis highlights variability in model performance among different methodologies and emphasizes specific models that excel in their respective classification techniques.
- When examining the average recall values of models created through feature selection, it becomes evident that the LightGBM algorithm exhibits higher recall compared to other methods. While the DT method shows a 2.03% lower recall rate than LightGBM, the KNN algorithm shows 3.58% less recall. Additionally, MLP presents a 2.41% lower recall, SVM shows 3.16% less recall, and NB significantly records a 9.83% lower recall

compared to LightGBM. These findings highlight the effectiveness of Relief-F feature selection, particularly when dealing with datasets containing PE headers, alongside the LightGBM algorithm. This evaluation emphasizes the relative superiority of LightGBM among these algorithms and underscores the significant contribution of feature selection and hyperparameter tuning in enhancing the performance of models.

- When considering models classified with optimized values, NB exhibits the highest recall, followed by LightGBM, KNN, MLP, SVM, and DT, respectively. Upon analyzing Relief-F-based models, it is observed that different models exhibit high recall for each applied method. Within the scope of the NB method, model number 51 achieves the highest recall value. For LightGBM, models 9, 12, and 16 with the same recall levels achieve the highest recall, while for KNN, models 17 and 18 show the best performance. Model 10 provides the highest recall for SVM, and model 3 for DT. This comprehensive analysis highlights variability in model performance among different methodologies and emphasizes specific models that excel in their respective classification techniques.
- When examining the average precision values of models created through feature selection, it becomes apparent that the LightGBM algorithm shows higher precision compared to other methods. While the DT method exhibits a 1.06% lower precision rate than LightGBM, the KNN algorithm shows 0.83% less precision. Additionally, MLP demonstrates a 2.73% lower precision, SVM shows 1.69% less precision, and NB significantly records a 21.56% lower precision compared to LightGBM. These findings underscore the effectiveness of Relief-F feature selection, especially when dealing with datasets containing PE headers, in conjunction with the LightGBM algorithm. This evaluation emphasizes LightGBM's relative superiority among these algorithms and highlights the significant contribution of feature selection methods in enhancing the performance of ML models.
- When considering models classified with optimized values, the LightGBM demonstrates the highest precision, followed by DT, SVM, MLP, KNN, and NB respectively. Upon analyzing Relief-F-based models, it is observed that different models exhibit high precision for each applied method. Within the scope of the LightGBM method, model 1 achieves the highest precision value. For DT, model 4 yields the highest precision, while for SVM, model 18 performs the best. The highest performance for MLP is presented in model 26, whereas models ranging from 11 to 16, with similar precision values, demonstrate the highest performance for KNN. Model 23 provides the highest precision value for NB. This comprehensive analysis emphasizes variability in model performance among different methodologies and highlights specific models that excel in their respective classification techniques.

- When considering the collective average metrics of Relief-F-based models, the LightGBM emerged as the most effective classifier among all other ML methods evaluated with optimized parameters.
- The investigation was conducted utilizing a computing system outfitted with a 2.20 GHz Xenon CPU and 16 GB of RAM, operating on a 64-bit Windows 10 Professional operating system. The computation times for various machine learning algorithms, namely LightGBM, DT, KNN, MLP, SVM, and NB, were meticulously evaluated. The obtained average execution times for each algorithm were as follows: LightGBM necessitated 50478.6 sec, DT required 8773.9 sec, KNN consumed 38818.8 sec, MLP utilized 214784.7 sec, SVM demanded 360445.1 sec, and NB utilized 3624.3 sec, respectively.

5.4.2. Discussion of mRMR-Based Models

Examining models containing PE headers created using mRMR feature selection and evaluated with hyperparameter tuning of ML classifiers is as follows:

- When examining the average accuracy values of models created through feature selection, it becomes apparent that the LightGBM algorithm exhibits considerably higher accuracy compared to other methods. The DT method shows a 1.66% lower accuracy rate than LightGBM, while the KNN algorithm presents a 2.70% lower accuracy. Additionally, MLP demonstrates a 1.63% lower accuracy, SVM shows 1.95% less accuracy, and NB significantly records a 17.61% lower accuracy compared to the LightGBM. These findings underscore the effectiveness of mRMR feature selection, especially when dealing with datasets containing PE headers, in conjunction with the LightGBM algorithm. This evaluation not only highlights relative superiority of the LightGBM among these algorithms but also demonstrates that feature selection methods and hyperparameter tuning in ML algorithms significantly contribute to the performance of models.
- Considering the models classified with optimized values, LightGBM highlights the highest accuracy, followed by KNN, DT, MLP, SVM and NB. Analysis of mRMR-based models reveals different models providing high accuracy for each method. According to the LightGBM method, models 52, 56 and 57 exhibit the highest accuracy. For KNN, model 80 achieves the best performance, while for DT, model 57 achieves the best performance. Model 81 for MLP, model 79 for SVM, and model 94 for NB provide the highest accuracy. This comprehensive analysis highlights the variability in model performance between different methodologies and highlights specific models that are superior across relevant classification techniques.

- When analyzing the average F1-Score values of models created through feature selection, it becomes evident that the LightGBM algorithm exhibits higher F1-Score compared to other methods. The DT method shows a 1.60% lower F1-Score rate than LightGBM, while the KNN algorithm presents a 4.34% lower F1-Score. Additionally, MLP demonstrates a 1.52% lower F1-Score, SVM shows 1.08% less F1-Score, and NB records a significantly lower F1-Score of 15.09% compared to LightGBM.
- Considering models classified with optimized values, LightGBM shows the highest F1-Score, followed by KNN, SVM, MLP, DT, and NB. Analyzing mRMR-based models reveals different models achieving high F1-Score for each applied method. Under the LightGBM method, models 52, 56, and 57 achieve the highest F1-Score. For KNN, model 60 attains the highest F1-Score levels, while model 95 performs best for SVM. Model 71 yields the highest F1-Score for MLP, while for DT, model 60 provides the highest F1-Score. Additionally, model 98 stands out as the most performant model for NB within this methodology. This comprehensive analysis underscores the variability in model performance among different methodologies and emphasizes specific models that excel in their respective classification techniques.
- When examining the average recall values of models created through feature selection, it is evident that the LightGBM algorithm exhibits higher recall performance compared to other methods. The DT method shows a 2.00% lower recall rate than LightGBM, while the KNN algorithm presents a 6.13% lower recall rate. Additionally, MLP demonstrates a 1.66% lower recall, SVM presents 2.59% less recall, and NB records a significantly lower recall of 8.11% compared to the LightGBM. These findings emphasize the effectiveness of mRMR feature selection, particularly when dealing with datasets containing PE headers, alongside the LightGBM algorithm. This evaluation not only highlights relative of the LightGBM superiority among these algorithms but also underscores the significant contribution of feature selection and hyperparameter tuning in enhancing the performance of ML models.
- Considering models classified with optimized values, the LightGBM presents the highest recall, followed by KNN, DT, MLP, SVM, and NB. Analyzing mRMR-based models reveals different models achieving high recall for each applied method. Under the LightGBM method, model 53 achieves the highest recall. For KNN, models 79 and 81 with similar recall levels perform the best, while for DT, model 95 delivers the optimal performance. Model 72 stands out for MLP, while model 95 performs best for SVM in terms of recall. Additionally, model 102 provides the highest recall for NB. This comprehensive analysis underscores the variability in model performance among different methodologies and emphasizes specific models that excel in their respective classification techniques.

- When examining the average precision values of models generated through feature selection, it becomes evident that the KNN algorithm demonstrates higher precision compared to other methodologies. While the LightGBM method shows a precision rate approximately 0.07% lower than KNN, the DT algorithm shows 1.23% less precision. Additionally, MLP exhibits 1.42% lower precision, SVM shows 1.19% less precision, and NB notably records 20.04% lower precision compared to KNN. These findings highlight the effectiveness of mRMR feature selection, especially when dealing with datasets containing PE headers. This evaluation underscores the relative superiority of KNN among these algorithms and emphasizes the significant contribution of feature selection in enhancing the performance of ML models.
- When considering models classified with optimized values, LightGBM highlights the highest precision, followed by DT, SVM, MLP, KNN, and NB. Analyzing mRMR-based models reveals that different models exhibit high precision for each applied method. Within the scope of the LightGBM method, model 52 attains the highest precision. For Decision Trees, model 54 provides the highest precision, whereas for SVM, model 72 stands out. Model 57 represents the peak performance for MLP, while models 79 through 81, showing similar precision values, excel for KNN. Finally, model 94 achieves the highest precision for NB. This comprehensive analysis highlights the variability in model performance among different methodologies and accentuates specific models that outperform others in classification techniques.
- Considering the collective average metrics of mRMR-based models, LightGBM emerged as the most effective classifier among all other ML methods evaluated with optimized parameters, except the precision criterion.
- KNN presented the best performance in the precision criterion.
- The investigation was conducted on a computing system operating with a 64-bit Windows 10 Professional operating system, featuring a 2.20 GHz Xenon CPU and 16 GB of RAM. The average execution times for a series of machine learning algorithms, namely LightGBM, DT, KNN, MLP, SVM, and NB, were meticulously determined. Specifically, the computational durations for each algorithm were recorded as follows: LightGBM required 38656.5 sec, DT necessitated 8992.6 sec, KNN consumed 43440.8 sec, MLP utilized 213424.4 sec, SVM demanded 539864.3 sec, and NB took 595.7 sec, respectively.

5.5. General Discussion

The examination of models created using mRMR and Relief-F feature selection, evaluated with both default and optimized parameters of ML classifiers, is as follows:

- When using default parameters in feature selection methods like Relief-F and mRMR, the LightGBM algorithm has been shown to be the most successful method in terms of evaluation criteria such as accuracy, F1-Score, recall, and precision.
- When models generated using the Relief-F algorithm utilize optimized parameters, it becomes apparent that the LightGBM algorithm emerges as the most successful method concerning evaluation criteria such as accuracy, F1-Score, recall, and precision. However, upon examining models created with mRMR feature selection, the LightGBM achieves the most successful results in accuracy, F1-Score, and recall, whereas the KNN outperforms in terms of precision.
- When comparing the performance of ML methods using Relief-F-based models with optimized parameters against default parameters in terms of the accuracy metric, notable differences were observed. In this analysis, the LightGBM algorithm exhibited a superior performance of 1.93% compared to another method. The DT method showed a 0.13% improvement, the KNN demonstrated a 5.3% enhancement, the MLP showed a 14.57% increase, the SVM displayed a 21.6% advancement, and the NB portrayed a 24.25% betterment in results. These results clearly demonstrate the impact of optimized parameters on the performance of different algorithms.
- The performance of ML methods using Relief-F-based models with optimized parameters, when compared to default parameters in terms of the F1-Score metric, showed notable differences. The LightGBM demonstrated a 0.09% superior performance over another method. The DT exhibited a 0.20% decrease, KNN presented a 6.20% improvement, MLP showed a 16.83% increase, SVM indicated a 16.57% progression, and the NB portrayed a substantial 53.14% improvement in results. While optimized parameters showed similar performance across some ML methods, MLP, SVM, and NB showed notably superior performance in this evaluation criterion. However, a slight decrease in performance was observed in the DT algorithm.
- The models using Relief-F-based techniques with optimized parameters demonstrated notable differences compared to default parameters in terms of the recall metric. The LightGBM showcased a 0.32% improvement, while DT exhibited a decrease of 0.18%. The KNN displayed an enhancement of 7.9%, MLP showed a substantial increase of 16.72%, SVM illustrated a modest improvement of 1.55%, and NB portrayed a significant 67.15% advancement in results. Although optimized parameters showed similar performance in certain ML methods, MLP and NB notably outperformed others in this evaluation criterion. However, a slight decrease in performance was observed in the DT algorithm.

- Models using Relief-F-based techniques with optimized parameters showed differences in the precision metric compared to default parameters. The LightGBM exhibited a 0.42% improvement, while DT demonstrated a 0.27% enhancement. The KNN showed an increase of 3.38%, MLP showed a noteworthy increase of 13.00%, SVM illustrated a big improvement of 27.43%, and NB displayed a significant decline in results by 12.53%. While optimized parameters exhibited similar performance across certain ML methods, MLP and SVM notably outperformed others on this evaluation criterion. However, a notable decrease in performance was observed in the NB algorithm.
- When comparing the performance of ML methods using mRMR-based models with optimized parameters to default parameters in terms of accuracy metric, notable differences were observed. In this analysis, the LightGBM algorithm demonstrated a 0.4% superior performance over another method. The DT exhibited similar performance, KNN displayed an improvement of approximately 7.4%, MLP showed an increase of 16.26%, SVM illustrated a progression of 21.65%, and NB portrayed a 25.14% improvement in results. Upon examining the findings, significant enhancements were observed in KNN, MLP, SVM, and NB, while DT and LightGBM showed nearly identical performance. Also, these outcomes clearly highlight the impact of optimized parameters on the performance of different algorithms.
- When the performance of ML methods using mRMR-based models with optimized parameters was compared with the default parameters in terms of the F1-Score metric, some differences were observed. In this observation, the LightGBM algorithm outperformed by 0.4%. DT exhibited a similar performance, while KNN displayed a 10.42% improvement. The MLP showed a substantial 21.88% enhancement, SVM illustrated a significant improvement of 17.08%, and NB portrayed a remarkable 53.51% improvement. Upon examination of these findings, notable enhancements were observed in KNN, MLP, SVM, and NB, whereas DT and the LightGBM demonstrated nearly identical performances.
- When comparing the performance of ML methods utilizing mRMR-based models with optimized parameters to default parameters in terms of the recall metric, notable differences were observed. In this observation, the LightGBM algorithm demonstrated a 0.35% superior performance. While DT exhibited a substantial performance decrease of 0.20%, KNN showcased an improvement of 12.67%. The MLP displayed a significant enhancement of 20.21%. The SVM experienced a slight decline of 0.42%, whereas NB portrayed a remarkable improvement of 67.04%. These findings highlight advancements in ML algorithms such as LightGBM, KNN, NB, and MLP, while DT and SVM demonstrated comparatively lower performances.

- When assessing the performance of ML methods using mRMR-based models with optimized parameters in terms of the precision metric, notable differences were observed compared to default parameters. In this observation, the LightGBM algorithm showed a superior performance by 0.41%. The DT demonstrated a performance increase of 0.22%, while KNN exhibited an improvement of 2.93%. The MLP showed a noteworthy increase of 16.48%. The SVM experienced a substantial enhancement of 28.40%, whereas NB recorded a significant decline of 14.91%. These findings underscore the enhancements in ML algorithms such as LightGBM, DT, KNN, SVM, and MLP, while highlighting notably poor performance in the NB method.
- When assessing Relief-F-based models using the accuracy criterion, the default parameter results demonstrated a performance of 82.97%, whereas the optimized parameter results yielded a higher performance of 93.03%.
- In terms of the F1-Score criterion, the default parameter outcomes for Relief-F-based models stood at 79.87%, while the optimized parameter results notably improved to 93.30%.
- Regarding the recall criterion, the default parameter performance for Relief-F-based models reached 80.93%, whereas the optimized parameter results exhibited a higher performance of 94.93%.
- Evaluating based on the precision criterion, Relief-F-based models with default parameters achieved a performance of 86.98%, while those with optimized parameters showed a slightly higher performance of 92.45%.
- When considering the comprehensive average across all evaluation criteria for Relief-F-based models, the default parameter settings yielded an average performance of 82.69%. Conversely, employing optimized parameters resulted in a notable enhancement, elevating the average performance significantly to 93.42%.
- When scrutinizing the accuracy criterion among mRMR-based models, the default parameter configuration exhibited a performance of 81.68%, whereas employing optimized parameters resulted in a heightened performance of 92.28%.
- Evaluating the F1-Score criterion across mRMR-based models, default parameters achieved a performance of 77.70%, whereas optimized parameters notably improved to 92.89%.
- In terms of the recall criterion, default parameters yielded a performance of 79.61%, while optimized parameters significantly enhanced performance to 94.85%.
- Assessing based on the precision criterion, default parameters achieved a performance of 85.94%, whereas optimized parameters improved to 91.66%.

- When considering the comprehensive average across all evaluation criteria for mRMR-based models, the default parameter settings yielded an average performance of 81.23%. Conversely, employing optimized parameters resulted in a notable enhancement, elevating the average performance significantly to 92.92%.
- The NB classifier operates based on assumptions, one of which is the assumption of independence among expressions or features. This concept of independence implies that the NB classifier might struggle to produce accurate results in situations involving correlations. In datasets with highly correlated features, this classifier typically exhibits low performance. This is because the model, unable to fulfill the assumption of independence, fails to accurately model true relationships, consequently leading to biased outcomes (Yang, 2018). Within the scope of this study, the PE file headers encapsulate the entire process from loading a file into memory to the termination of its execution in a spiral. Hence, many header information within it is interrelated. The dataset used in this study contains header information with high correlation. This constitutes the primary reason for the high percentage increase rate observed in both feature selectors for the NB classifier.
- Upon a detailed examination of the Relief-F's F1-Score criterion, it was observed that the *Characteristics* and *DllCharacteristics* parameters positively influenced the performance of models. However, it was observed that there was no negative or ineffective value on the performance of the model.
- Analyzing Relief-F's recall criterion in detail revealed the positive influence of the *DllCharacteristics* parameter on the performance of model. However, it was observed that there was no negative or ineffective value on the performance of the model.
- When the models created with both Relief-F and mRMR feature selectors are examined, the features considered important for all methods are as follows, respectively. For Relief-F, it is *SizeOfInitializedData, Subsystem, NumberOfSections, e_lfanew, Characteristics, CreationYear, DllCharacteristics*. For mRMR, it is *Magic, SizeOfInitializedData, SectionAlignment, SizeOfStackReserve, Subsystem, MinorSubsystemVersion, e_lfanew, ImageBase*.
- When examining mRMR's F1-Score criterion in detail, it became apparent that *Characteristics*, *MinorSubsystemVersion*, *e_lfanew*, and *ImageBase* parameters positively influenced the performance. However, no values were detected that had a negative or ineffective impact on model performance.
- In the detailed analysis of mRMR's recall criterion, it was observed that *Subsystem* and *ImageBase* parameters had a positive impact on performance of the model.

Nevertheless, no values could be identified that had a negative or ineffective effect on model performance.

- Upon examining the processing durations of Relief-F-based models, various algorithms exhibited differing processing times. LightGBM took 315.9 sec, DT completed in 41.7 sec, KNN required 73.7 sec, MLP processed for 249.3 sec, SVM endured 265.1 sec, and NB concluded within 31.1 sec. However, when the models were executed using optimized parameters, notable changes in processing durations were evident. Specifically, LightGBM required 50478.6 sec, DT was 8773.9 sec, KNN necessitated 38818.8 sec, MLP processed for 214784.7 sec, SVM endured 360445.1 sec, and NB concluded within 3624.3 sec. The utilization of optimized parameters resulted in a significant increase in processing durations models.
- When the processing times of mRMR-based models were examined, it was seen that various algorithms exhibited different processing times. LightGBM took 207.8 sec, DT completed in 38.3 sec, KNN completed in 101.1 sec, MLP processed in 228.4 sec, SVM completed in 563.9 sec, and NB completed in 29.4 sec. However, when the models were run using optimized parameters, notable changes in processing times were clearly evident. Specifically, LightGBM required 38656.5 sec, DT required 8992.6 sec, KNN required 43440.8 sec, MLP processed 213424.4 sec, SVM took 539864.3 sec, and NB concluded in 595.7 sec. Using optimized parameters resulted in a significant increase in processing times for certain models.
- When considering the overall average processing times, Relief-F-based classifiers with default parameters completed in an average time of 162.8 sec, whereas mRMR-based classifiers concluded in 194.82 sec. Conversely, the overall average processing time for all mRMR-based classifiers with default parameters was 112,820.90 sec, while Relief-F-based classifiers required 140,829.05 sec. Moreover, the difference between all Relief-F and mRMR-based classifiers with default parameters was 32.02 sec, whereas the difference between optimized classifiers stood at 28,01 sec. These findings indicate that Relief-F exhibited quicker processing times in both parameter methods.
- The study, incorporating both Relief-F and mRMR feature selectors along with various ML methods, was conducted on a computer system equipped with a 2.20 GHz Xenon CPU, 16 GB RAM, and running a 64-bit Windows 10 Professional Operating System.

5.6. Comparing Results with Other Studies in the Literature

Kumar et al. (2019) proposed a system based on malware detection using PE files. The overall performance of the dataset was evaluated using 10-fold cross-validation. When focusing on the accuracy metric, the DT method exhibited a performance of 96.00%, whereas the NB method showed a performance of 57.8%. When comparing the performance of the DT algorithm with other

models, it demonstrated a 0.92% improvement in model 1 and a 1.38% improvement in model 12 among Relief-F-based optimized models. In the mRMR-based optimized models, model 52 showed a 0.96% enhancement, and model 57 demonstrated a 1.45% improvement. The performance of the NB algorithm indicated a 22.33% enhancement in model 1 and a 33.23% improvement in model 44 among Relief-F-based optimized models, while among mRMR-based optimized models, model 52 showed a 22.79% enhancement, and model 94 exhibited a 34.26% improvement. These results illustrate comparable outcomes for the optimized DT algorithm using feature selection methods, while the NB algorithm shows notably successful outcomes. The performance outcomes are presented in Table 5.35.

Table 5.35. Comparison of Kumar et al. (2019) and our studies

Studies	**Accuracy (%)**	
	DT	**NB**
(Kumar et al., 2019)	96.00	57.80
Relief-F-based Model 1	96.89	74.42
Relief-F-based Model 12	**97.34**	76.21
Relief-F-based Model 44	97.34	**86.57**
mRMR-based Model 52	96.93	74.86
mRMR-based Model 57	**97.41**	75.83
mRMR-based Model 94	93.96	**87.92**

Moreles-Molina et al. (2018) presented a classification approach using CLAMP and CSDMC_API_Train datasets, and the dataset was divided into a 60 / 40 training-testing split. Various methods such as PCA, Term Frequency-Inverse Document Frequency (TF-IDF), SVM, and DT were employed for feature extraction and classification. Model performance was evaluated using precision, recall, and F1-Score metrics. Upon examining the findings, in training solely on the CLAMP dataset, the SVM algorithm demonstrated a precision of 61.54%, recall of 78.08%, and an F1-Score of 68.83%. Similarly, the DT algorithm exhibited a precision of 87.73%, recall of 88.21%, and an F1-Score of 87.97%. When comparing the methods and evaluation criteria used in this study with the proposed approach, significantly higher performance was observed for the SVM and DT algorithms. For Relief-F based models, model 1 showed a 36.91% higher performance value in the SVM algorithm precision criterion. Additionally, the best model among all models, model 18 exhibited a 36.96% higher performance. While model 1 provided a 17.36% higher performance for the recall criterion, the optimum model in this evaluation criterion, model 36, demonstrated a 19.70% higher performance. In terms of the F1-Score criterion, model 1 outperformed by 27.15%, while the optimum value for this criterion, model 36 showed a 29.22% higher performance. Considering the DT algorithm precision criterion, model 1 presented a 9.81% higher performance, while the optimum model for this criterion, model 4, showed a 10.28% higher performance. For the recall criterion, model 1 exhibited a 9.04% higher performance, whereas the optimum model, model 3 demonstrated

a 9.18% higher performance. Examining the F1-Score criterion, model 1 outperformed by 9.30%, while the optimum model for this criterion, model 15 provided a 9.78% higher performance. Turning to mRMR-based models, model 52 demonstrated a 36.89% higher performance in the SVM algorithm precision criterion. Additionally, model 72 showed a 37.00% higher performance, being the best model among all models. For the recall criterion, model 52 outperformed by 17.36%, while model 95, the optimum model, exhibited a 19.83% higher performance. Considering the F1-Score criterion, model 52 surpassed by 27.15%, while the optimum value for this criterion, model 95, outperformed with a 29.33% higher performance. Considering the DT algorithm precision criterion, model 52 showed a 9.81% higher performance, while the optimum model for this criterion, model 54, demonstrated a 10.20% higher performance. For the recall criterion, model 52 exhibited a 8.90% higher performance, whereas the optimum model, model 95, showcased a 9.70% higher performance. Examining the F1-Score criterion, model 52 outperformed by 9.41%, while the optimum model for this criterion, model 60 showcased a 9.54% higher performance. The evaluation of Moreles-Molina et al. (2018) and this study according to relevant metrics is presented in Table 5.36.

Table 5.36. Comparison of (Moreles-Molina et al., 2018) and our studies

Studies	**F1-Score (%)**		**Recall (%)**		**Precision (%)**	
	DT	**SVM**	**DT**	**SVM**	**DT**	**SVM**
(Moreles-Molina et al., 2018)	87.97	68.83	88.21	78.08	87.73	61.54
Relief-F-based Model 1	96.99	94.48	96.98	94.48	97.27	97.55
Relief-F-based Model 18	97.10	94.97	96.72	94.97	97.30	**97.60**
Relief-F-based Model 36	96.30	**97.24**	95.75	**97.24**	96.16	95.65
Relief-F-based Model 4	97.21	94.41	96.50	96.41	**97.78**	97.51
Relief-F-based Model 3	97.15	94.45	**97.13**	94.45	97.31	97.47
Relief-F-based Model 15	**97.51**	94.67	96.94	94.67	97.32	97.55
mRMR-based Model 52	97.11	94.48	96.83	94.48	97.27	97.51
mRMR-based Model 72	96.43	94.97	96.27	94.97	96.55	**97.68**
mRMR-based Model 95	94.07	**97.39**	**97.69**	**97.39**	90.75	90.35
mRMR-based Model 54	97.12	94.41	96.72	94.41	97.69	97.47
mRMR-based Model 60	**97.25**	94.63	96.91	94.63	97.20	97.52

(P. Singh et al., 2022) proposed that a malicious activity detection system utilizing PE header information was developed. The CLAMP dataset was directly employed, and various feature engineering techniques such as transformation and preprocessing of raw data containing PE header information were applied. SVM and KNN classifiers were utilized to construct models, and their generalization and training were completed through 10-fold cross-validation. Evaluation criteria comprised accuracy, F1-Score, recall, and precision. When compared with Relief-F-based models, Model 13 and Model 18 derived from the KNN method exhibited superior performance in accuracy and F1-Score criteria, with Model 18 showing higher recall and Model 13 demonstrating higher precision. Similarly, for the SVM method, Model 37 showed higher accuracy performance, while

Model 36 outperformed in F1-Score and recall criteria. Additionally, Model 18 showed high precision performance. Upon examining mRMR-based models, Model 53 derived from the KNN method appeared superior in accuracy criterion, whereas Model 79 showed higher performance in F1-Score, recall, and precision criteria. Likewise, when investigated for the SVM method, Model 79 demonstrated better performance in accuracy criterion, while Model 72 excelled in precision criterion. Model 95 exhibited relatively higher performance in F1-Score and recall criteria. Table 5.37 presents the evaluation of all models and optimal models included in our studies with (P. Singh et al., 2022).

Table 5.37 Comparison of (P. Singh et al., 2022) and our studies

Studies	**Accuracy (%)**		**F1-Score (%)**		**Recall (%)**		**Precision (%)**	
	KNN	**SVM**	**KNN**	**SVM**	**KNN**	**SVM**	**KNN**	**SVM**
(P. Singh et al., 2022)	94.61	94.19	97.27	96.96	97,66	97.59	96,89	96.40
Relief-F-based Model 13	**97.47**	96.03	**97.57**	94.74	98.06	94.74	**97.09**	97.52
Relief-F-based Model 18	**97.47**	96.18	**97.57**	94.97	**98.10**	97.24	97.06	**97.60**
Relief-F-based Model 36	96.86	96.28	96.99	**97.24**	97.80	**97.24**	96.20	95.65
Relief-F-based Model 37	96.82	**96.45**	96.95	97.06	97.73	97.06	96.19	96.14
mRMR-based Model 53	**97.41**	95.95	97.51	94.60	97.95	94.60	97.09	97.52
mRMR-based Model 72	97.34	96.22	97.44	94.97	97.88	94.97	97.01	**97.68**
mRMR-based Model 79	97.47	**96.41**	**97.57**	95.64	**97.99**	95.64	**97.16**	97.40
mRMR-based Model 95	91.18	93.25	91.29	**97.39**	91.31	**97.39**	91.66	90.35

6. CONCLUSION AND FUTURE WORK

In this chapter, the key findings and contributions of our study are summarized, highlighting the implications for the field of malware detection. Furthermore, avenues for future research aimed at enhancing the effectiveness and robustness of malware detection systems are outlined.

6.1. Conclusion

The advancement of technology and devices, the field of cybersecurity is increasingly gaining significance. Many organizations and governments are adversely affected by cyber threats, leading to the development of numerous defense strategies to cope with these challenges. Additionally, various domains such as artificial intelligence, big data, cloud computing, and the Internet of Things have become integral parts of our daily lives. Specifically, tools like artificial intelligence, ML, and deep learning, while simplifying our lives, also pose risks.

Malicious software developers exploit these tools to discover vulnerabilities within systems, create intricate malicious software, infiltrate personal devices, and engage in bullying, digital harassment, and ransomware attacks, among other detrimental actions that can significantly affect our lives. On the other hand, cybersecurity experts endeavor to prevent or safeguard against these adversities by developing ML, deep learning, or AI-supported malicious software detection systems. Utilizing such tools in cybersecurity is crucial due to their capacity for continuous self-improvement, automated threat identification, and mitigation, offering a less costly and more reliable means of combating cyber threats than human intervention.

In the field of cybersecurity, considering the constant development of increasingly sophisticated malicious software by malicious software developers, staying up-to-date is of utmost importance. Therefore, employing modern and effective algorithms is crucial. This study utilizes contemporary ML algorithms and feature selectors. Unique models incorporating the PE file headers were created using Relief-F-based and mRMR-based feature selectors, and these models were ranked in descending order based on the importance level determined by the feature selectors. The six distinct models, each employing different default parameter settings and hyperparameter optimization tuning for various ML classifiers, were trained. The findings underscore the significant role of hyperparameter tuning in ensuring accurate and effective detections by the ML classifiers. Additionally, the use of feature selectors proves advantageous in facilitating rapid decision-making by the ML classifiers when dealing with system infections or encountering new malicious activities. This swift decision-making is paramount for the integrity and security of the system.

Each model, generated through the use of feature selectors, underwent an evaluation of its generalization capacities using a 10-fold cross-validation method. To conduct a comprehensive assessment of the models, their performance was analyzed across various criteria, including accuracy, F1-Score, recall, precision, and the confusion matrix. This detailed evaluation sought to

offer a comprehensive insight into effectiveness of the models in managing various facets of classification and to guarantee a thorough assessment of their performance across diverse scenarios.

When assessed using Relief-F-based and mRMR-based models with default parameter configurations, the LightGBM emerged as the most proficient model across all metrics, while the NB exhibited the weakest performance, particularly in terms of accuracy, F1-Score, and recall. Additionally, the SVM demonstrated relatively poor performance, particularly in the precision metric. However, after evaluation with optimized parameters, the LightGBM represented promising results, whereas the NB yielded the least effective outcomes. For mRMR-based models with optimized parameters, the LightGBM demonstrated superior performance in accuracy, F1-Score, and recall metrics, while the KNN exhibited relatively higher precision. In this context, the NB emerged as the least performing model.

6.2. Future Work

This study holds promise for various future expansions in this field. Further research across multiple diverse datasets with different features is necessary to generalize the advantages of ML classifiers. Combining disparate datasets becomes crucial in enhancing classifier generalization capabilities and in the potential detection of novel malware types. Such investigations underscore the significance of amalgamating diverse data sources in augmenting classifier performance, thereby offering more effective protection against a broader spectrum of threats. Future research efforts should delve deeper into examining how studies conducted across diverse datasets impact the generalization abilities of classifiers and their role in detecting emerging threats.

The ML techniques play a crucial role in the detection of malware. Consequently, the preferred algorithms effectively contribute to malware detection. The rapid and efficient detection capabilities of modern algorithms significantly affect the malware analysis process. Therefore, this thesis proposes an approach committed to incorporating modern and other ML methods, aiming to enhance future research in malware detection.

The exploration of the majority-voting concept presents an alternative avenue for further scholarly inquiry. The majority-voting methodology entails the participation of multiple classifiers in rendering decisions, culminating in a consensus-based outcome. While this approach holds promise in augmenting the aggregate accuracy and dependability of malware detection, particularly in contexts where distinct the ML models demonstrate proficiency in discerning various facets of malicious activity, it may concurrently incur elevated computational costs. Nevertheless, leveraging the rapid processing capabilities and efficient methodologies inherent in modern algorithms, a comprehensive examination of the majority voting approach emerges as a viable subject for future academic investigations.

Feature selectors can enhance model performance by reducing the complexity within datasets. The utilization of various selectors, particularly in extensive datasets, may assist in

achieving a more generalized model with reduced overfitting. Moreover, opting for automated methods to determine features could be more suitable than manually selecting them. In domains like continual malware detection, where staying updated is crucial, employing automated feature selection methods ensures adaptability to new or evolving malware types and plays a significant role in cost reduction. This approach holds greater reliability in adjusting to the ever-changing landscape of threats. In automated systems like ML, it significantly contributes to optimizing feature selection and maintaining ongoing relevance. This study provides valuable information for future research efforts that focus on combining various feature selector approaches or using different feature selectors.

After the loading of the PE files into the operating system, a dynamic malware analysis process can be initiated, encompassing activity tracking, API calls, and string analysis. The dynamic analysis of malware activities involves monitoring and scrutinizing the behavior exhibited by the malware file following its execution within the system. Consequently, the identification of malicious software activities can be achieved with enhanced efficacy through the application of ML methods and heuristic techniques. In contrast to the limitations inherent in static analysis, particularly when dealing with novel malware variants, the dynamic analysis approach empowers the prevention of malicious activities before they propagate within the system. Furthermore, the static analysis approach, encompassing PE file headers as provided in this study, may illuminate a hybrid methodology. This hybrid approach, synthesizing the advantageous aspects of both dynamic and static analysis, holds promise for future research endeavors.

In malware analysis, besides accurate detection, the speed at which the system detects threats is also of great importance. Modern operating systems store most of the crucial information about the PE files to be executed in the Optional Header and File Header sections. Therefore, a thorough examination of these areas provides an effective starting point for quick scanning and assessing the key features of the file. This information is highly useful for understanding the general structure of the file and rapidly identifying potential threats. However, this analysis approach has some limitations. It may overlook more complex and advanced features of the file, neglect dynamic behavior examination, and might fail to detect certain malicious activities. Future studies may offer a more detailed insight into a fast malware activity detection system specific to modern operating systems.

Malicious software activities are no longer confined solely to Windows platforms, they are increasingly prevalent in mobile device operating systems and other platforms as well. This highlights the need for protection against malicious software across different platforms. Future-oriented research may involve the development of a comprehensive malicious software detection system encompassing not only Windows but also mobile operating systems like Android, iOS, and other operating systems such as macOS and Linux. Such an endeavor could mark a significant step toward understanding malicious software threats across diverse platforms and establishing effective

defenses against them. The distinctive features and security structures of various operating systems necessitate individual scrutiny to comprehend how malicious software behaves across different platforms. However, creating a detection system capable of operating across multiple platforms may encounter complexity due to the distinct security structures, application architectures, and behavioral traits of different platforms. Hence, prospective research in this domain may need to focus on effectively balancing these diversities to develop an efficient and comprehensive detection system.

The utilization of deep learning in malware analysis can significantly enhance the process of detecting complex and variable malicious software. Deep learning algorithms demonstrate a robust capability to learn from extensive datasets, enabling them to detect and comprehend intricate patterns specific to computer systems. In an environment where malicious software continually evolves, deep learning offers a more adaptive and flexible approach compared to traditional methods. If the dataset utilized in this study is expanded, it could establish a foundation for deep learning.

REFERENCES

Aboaoja, F. A., Zainal, A., Ali, A. M., Ghaleb, F. A., Alsolami, F. J., & Rassam, M. A. (2023). Dynamic Extraction of Initial Behavior for Evasive Malware Detection. *Mathematics*, *11*(2). https://doi.org/10.3390/math11020416

Acid, S., de Campos, L. M., & Fernandez, M. (2011). Minimum redundancy maximum relevancy versus score-based methods for learning Markov boundaries. *2011 11th International Conference on Intelligent Systems Design and Applications*, 619–623.

Akhtar, M. S., & Feng, T. (2023). Evaluation of Machine Learning Algorithms for Malware Detection. *Sensors*, *23*(2). https://doi.org/10.3390/s23020946

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). *Optuna: A Next-generation Hyperparameter Optimization Framework*. http://arxiv.org/abs/1907.10902

Anderson, H. S., & Roth, P. (2018). *EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models*. http://arxiv.org/abs/1804.04637

Bagui, S., Kalaimannan, E., Bagui, S., Nandi, D., & Pinto, A. (2019). Using machine learning techniques to identify rare cyber-attacks on the UNSW-NB15 dataset. *SECURITY AND PRIVACY*, *2*(6). https://doi.org/10.1002/spy2.91

Balodi, B., Sharma, S., Shukla, A. K., & Singh, B. (2023). Automated Static Malware Analysis Using Machine Learning. *Proceedings of the 10th International Conference on Signal Processing and Integrated Networks, SPIN 2023*, 255–258. https://doi.org/10.1109/SPIN57001.2023.10116580

Balram, N., Hsieh, G., & McFall, C. (2019). Static malware analysis using machine learning algorithms on apt1 dataset with string and PE header features. *Proceedings - 6th Annual Conference on Computational Science and Computational Intelligence, CSCI 2019*, 90–95. https://doi.org/10.1109/CSCI49370.2019.00022

Bhavsar, H., & Ganatra, A. (2012). A Comparative Study of Training Algorithms for Supervised Machine Learning. *International Journal of Soft Computing and Engineering (IJSCE)*, *2*(4), 74.

Dada, E. G., Stephen Bassi, J., Hurcha, Y. J., & Alkali, A. H. (2019). Performance Evaluation of Machine Learning Algorithms for Detection and Prevention of Malware Attacks. *IOSR Journal of Computer Engineering (IOSR-JCE)*, *21*(3), 18–27. https://doi.org/10.9790/0661-2103011827

Dhammi, A., & Singh, M. (2015). Behavior analysis of malware using machine learning. *International Conference on Contemporary Computing (IC3)*, 481–486.

Fan, C. I., Hsiao, H. W., Chou, C. H., & Tseng, Y. F. (2015). Malware detection systems based on API log data mining. *Proceedings - International Computer Software and Applications Conference*, *3*, 255–260. https://doi.org/10.1109/COMPSAC.2015.241

Galal, H. S., Mahdy, Y. B., & Atiea, M. A. (2016). Behavior-based features model for malware detection. *Journal of Computer Virology and Hacking Techniques*, *12*(2), 59–67. https://doi.org/10.1007/s11416-015-0244-0

Ge, X., Sun, J., Lu, B., Chen, Q., Xun, W., & Jin, Y. (2019). Classification of oolong tea varieties based on hyperspectral imaging technology and BOSS-LightGBM model. *Journal of Food Process Engineering*, *42*(8). https://doi.org/10.1111/jfpe.13289

Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). KNN model-based approach in classification. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *2888*, 986–996. https://doi.org/10.1007/978-3-540-39964-3_62

Hadiprakoso, R. B., Kabetta, H., & Buana, I. K. S. (2020). Hybrid-Based Malware Analysis for Effective and Efficiency Android Malware Detection. *Proceedings - 2nd International Conference on Informatics, Multimedia, Cyber, and Information System, ICIMCIS 2020*, 8–12. https://doi.org/10.1109/ICIMCIS51567.2020.9354315

Hammad, M., El-Medany, W., & Ismail, Y. (2020, December 20). Intrusion Detection System using Feature Selection with Clustering and Classification Machine Learning Algorithms on the UNSW-NB15 dataset. *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies, 3ICT 2020*. https://doi.org/10.1109/3ICT51146.2020.9312002

Hidayat, I., Ali, M. Z., & Arshad, A. (2023). Machine Learning-Based Intrusion Detection System: An Experimental Comparison. *Journal of Computational and Cognitive Engineering*. https://doi.org/10.47852/bonviewJCCE2202270

Hussain, A., Asif, M., Ahmad, M. Bin, Mahmood, T., & Raza, M. A. (2022). Malware Detection Using Machine Learning Algorithms for Windows Platform. *Lecture Notes in Networks and Systems*, *350*, 619–632. https://doi.org/10.1007/978-981-16-7618-5_53

Ijaz, M., Durad, M. H., & Ismail, M. (2019). Static and Dynamic Malware Analysis Using Machine Learning. *International Bhurban Conference on Applied Sciences & Technology, IBCAST*, 687–692.

Ilyas, M. U., & Alharbi, S. A. (2022). Machine learning approaches to network intrusion detection for contemporary internet traffic. *Computing*, *104*(5), 1061–1076. https://doi.org/10.1007/s00607-021-01050-5

Jain, P., Rajvaidya, I., Sah, K. K., & Kannan, J. (2022). Machine Learning Techniques for Malware Detection- a Research Review. *2022 IEEE International Students' Conference on Electrical, Electronics and Computer Science, SCEECS 2022*. https://doi.org/10.1109/SCEECS54111.2022.9740918

jameslamb. (2024). *The LightGBM*. Github. https://github.com/microsoft/LightGBM

Jha, P., & Sharma, A. (2021, January 27). Framework to Analyze Malicious Behaviour in Cloud Environment using Machine Learning Techniques. *2021 International Conference on Computer Communication and Informatics, ICCCI 2021*. https://doi.org/10.1109/ICCCI50826.2021.9402671

Kamel, H., Abdullah, D., & Al-Tuwaijari, J. M. (2019). Cancer Classification Using Gaussian Naive Bayes Algorithm. *Fifth International Engineering Conference on Developments in Civil & Computer Engineering Applications 2019 - (IEC2019) - Erbil - IRAQ*, 165–171.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems*, *30*.

Kocher, G., & Kumar, G. (2021). Analysis of Machine Learning Algorithms with Feature Selection for Intrusion Detection using UNSW-NB15 Dataset. *International Journal of Network Security & Its Applications*, *13*(1), 21–31. https://doi.org/10.5121/ijnsa.2021.13102

Kumar, A., Abhishek, K., Shah, K., Patel, D., Jain, Y., Chheda, H., & Nerurkar, P. (2020). Malware detection using machine learning. *Communications in Computer and Information Science*, *1232*, 61–71. https://doi.org/10.1007/978-3-030-65384-2_5

Kumar, A., Kuppusamy, K. S., & Aghila, G. (2019). A learning model to detect maliciousness of portable executable using integrated feature set. *Journal of King Saud University - Computer and Information Sciences*, *31*(2), 252–265. https://doi.org/10.1016/j.jksuci.2017.01.003

Li, X., Qiu, K., Qian, C., & Zhao, G. (2020). An adversarial machine learning method based on OpCode N-grams feature in malware detection. *Proceedings - 2020 IEEE 5th International Conference on Data Science in Cyberspace, DSC 2020*, 380–387. https://doi.org/10.1109/DSC50466.2020.00066

Liu, L., Wang, B. sheng, Yu, B., & Zhong, Q. xi. (2017). Automatic malware classification and new malware detection using machine learning. *Frontiers of Information Technology and Electronic Engineering*, *18*(9), 1336–1347. https://doi.org/10.1631/FITEE.1601325

Manavi, F., & Hamzeh, A. (2017). A New Method for Malware Detection Using Opcode Visualization. *2017 Artificial Intelligence and Signal Processing Conference (AISP)*.

Martínez Torres, J., Iglesias Comesaña, C., & García-Nieto, P. J. (2019). Review: machine learning techniques applied to cybersecurity. *International Journal of Machine Learning and Cybernetics*, *10*(10), 2823–2836. https://doi.org/10.1007/s13042-018-00906-1

Microsoft. (2024, January 31). *PE Format*. Microsoft. https://learn.microsoft.com/en-us/windows/win32/debug/pe-format

Misra, D. (2019). *Mish: A Self Regularized Non-Monotonic Activation Function*. http://arxiv.org/abs/1908.08681

Mohammed, A. R., Sai Viswanath, G., Sai babu, K., & Anuradha, T. (2020). Malware Detection in Executable Files Using Machine Learning. *Learning and Analytics in Intelligent Systems*, *3*, 277–284. https://doi.org/10.1007/978-3-030-24322-7_36

Mohanasruthi, V., Chakraborty, A., Thanudas, B., Sreelal, S., & Manoj, B. S. (2020). An Efficient Malware Detection Technique using Complex Network-based Approach. *2020 National Conference on Communications (NCC)*, 1–6.

Moon, D., Lee, J. K., & Yoon, M. K. (2022). Compact feature hashing for machine learning based malware detection. *ICT Express*, *8*(1), 124–129. https://doi.org/10.1016/j.icte.2021.08.005

Moreles-Molina, C. D., Santamaria-Guerrero, D., Sanchez-Perez, G., Toscano-Medina, K., Perez-Meana, H., & Hernandez-Suarez, A. (2018). Methodology for Malware Classification using a Random Forest Classifier. *2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC 2018). Ixtapa, Mexico*.

Moualla, S., Khorzom, K., & Jafar, A. (2021). Improving the Performance of Machine Learning-Based Network Intrusion Detection Systems on the UNSW-NB15 Dataset. *Computational Intelligence and Neuroscience*, *2021*. https://doi.org/10.1155/2021/5557577

Nawir, M., Amir, A., Yaakob, N., & Lynn, O. B. (2019). Effective and efficient network anomaly detection system using machine learning algorithm. *Bulletin of Electrical Engineering and Informatics*, *8*(1), 46–51. https://doi.org/10.11591/eei.v8i1.1387

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pietrek, M. (2002). Inside Windows: An In-Depth Look into the Win32 Portable Executable File Format, Part 2. *MSDN Magazine, March*, 1–10.

Radwan, A. M. (2019). Machine learning techniques to detect maliciousness of portable executable files. *Proceedings - 2019 International Conference on Promising Electronic Technologies, ICPET 2019*, 86–90. https://doi.org/10.1109/ICPET.2019.00023

Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., & Nicholas, C. (2018). Malware Detection by Eating a Whole EXE. *The Workshops of the Thirty-Second AAAI Conference on Artificial Intelligence*, 268–279. www.aaai.org

Ramchoun, H., Idrissi, M. A. J., Ghanou, Y., & Ettaouil, M. (2017). Multilayer perceptron: Architecture optimization and training with mixed activation functions. *ACM International Conference Proceeding Series*, *Part F129474*. https://doi.org/10.1145/3090354.3090427

Rkhouya, S., & Chougdali, K. (2021). Malware Detection Using a Machine-Learning Based Approach. *International Journal of Information Technology and Applied Sciences (IJITAS)*, *3*(4), 167–171. https://doi.org/10.52502/ijitas.v3i4.172

Robnik-Sikonja, M., & Kononenko, I. (2003). *Theoretical and Empirical Analysis of ReliefF and RReliefF* (Vol. 53).

Santos, I., Devesa, J., Brezo, F., Nieves, J., & Bringas, P. G. (2013). OPEM: A Static-Dynamic Approach for Machine-Learning-Based Malware Detection. *International Joint Conference CISIS'12-ICEUTE´12-SOCO´12 Special Sessions*, 271–280.

Singh, J., & Singh, J. (2021). A survey on machine learning-based malware detection in executable files. In *Journal of Systems Architecture* (Vol. 112). Elsevier B.V. https://doi.org/10.1016/j.sysarc.2020.101861

Singh, P., Borgohain, S. K., & Kumar, J. (2022). Investigation and pre-processing of CLaMP mlaware dataset for machine learning models. *6th International Conference on Electronics, Communication and Aerospace Technology, ICECA 2022 - Proceedings*, 891–895. https://doi.org/10.1109/ICECA55336.2022.10009153

Singhal, P., & Raul, N. (2012). *Malware Detection Module using Machine Learning Algorithms to Assist in Centralized Security in Enterprise Networks*.

Srastika, Bhandary, N., Shalakha, R. S., Honnavalli, P., & Sivaraman, E. (2022). An Enhanced Malware Detection Approach using Machine Learning and Feature Selection. *3rd International Conference on Electronics and Sustainable Communication Systems, ICESC 2022 - Proceedings*, 909–914. https://doi.org/10.1109/ICESC54411.2022.9885509

Tahir, R. (2018). A Study on Malware and Malware Detection Techniques. *International Journal of Education and Management Engineering*, *8*(2), 20–30. https://doi.org/10.5815/ijeme.2018.02.03

Thakkar, A., & Lohiya, R. (2023). Fusion of statistical importance for feature selection in Deep Neural Network-based Intrusion Detection System. *Information Fusion*, *90*, 353–363. https://doi.org/10.1016/j.inffus.2022.09.026

Yang, F. J. (2018). An implementation of naive bayes classifier. *Proceedings - 2018 International Conference on Computational Science and Computational Intelligence, CSCI 2018*, 301–306. https://doi.org/10.1109/CSCI46756.2018.00065

Yewale, A., & Singh, M. (2016). Malware Detection Based On Opcode Frequency. *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, 646–650.

Yousuf, M. I., Anwer, I., Riasat, A., Zia, K. T., & Kim, S. (2023). Windows malware detection based on static analysis with multiple features. *PeerJ Computer Science*, *9*. https://doi.org/10.7717/PEERJ-CS.1319

Zhang, S.-H., Kuo, C.-C., & Yang, C.-S. (2019). Static PE Malware Type Classification Using Machine Learning Techniques. *2019 International Conference on Intelligent Computing and Its Emerging Applications (ICEA)*, 81–87.

CURRICULUM VITAE

Vahdet Cemil ALTUN, graduated from Gülizar Şamil Aktaş Anatolian High School in 2015. Subsequently, he completed his studies at Hasan Kalyoncu University, earning a degree in Computer Engineering in 2020. He held a part-time position at EFA Bilişim from 2018 to 2020. In 2020, he transitioned to a full-time role at SimSoft Simulation and Computer Systems. He worked full-time at Kipaş Holding as a software engineering in 2021. Presently, he serves as a research assistant in the Department of Computer Engineering at Kahramanmaraş Sütçü Imam University.

APPENDICES

A.1. Overview of Relief-F-Based Models

Sequential models were created based on the importance levels of the selected PE file headers using the Relief-F algorithm. Additionally, model 1 represents the regular model, which contains all the features in the dataset. Table A.1.1 through Table A.1.8 offer an overview of the models generated utilizing the Relief-F algorithm.

Table A.1.1. Overview of prediction models of PE Header features selected by Relief-F (Part 1)

Models	Predictor Variables
Model 1	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion, BaseOfCode, e_cp, SizeOfUninitializedData, e_oemid, e_cblp, e_oeminfo, NumberOfSymbols, SizeOfCode, e_minalloc, e_csum, e_sp, e_cs, PointerToSymbolTable, e_ip, LoaderFlags, e_ss
Model 2	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion, BaseOfCode, e_cp, SizeOfUninitializedData, e_oemid, e_cblp, e_oeminfo, NumberOfSymbols, SizeOfCode, e_minalloc, e_csum, e_sp, e_cs, PointerToSymbolTable, e_ip, LoaderFlags
Model 3	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion, BaseOfCode, e_cp, SizeOfUninitializedData, e_oemid, e_cblp, e_oeminfo, NumberOfSymbols, SizeOfCode, e_minalloc, e_csum, e_sp, e_cs, PointerToSymbolTable, e_ip

Table A.1.2. Overview of prediction models of PE Header features selected by Relief-F (Part 2)

Models	Predictor Variables
Model 4	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion, BaseOfCode, e_cp, SizeOfUninitializedData, e_oemid, e_cblp, e_oeminfo, NumberOfSymbols, SizeOfCode, e_minalloc, e_csum, e_sp, e_cs, PointerToSymbolTable
Model 5	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion, BaseOfCode, e_cp, SizeOfUninitializedData, e_oemid, e_cblp, e_oeminfo, NumberOfSymbols, SizeOfCode, e_minalloc, e_csum, e_sp, e_cs
Model 6	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion, BaseOfCode, e_cp, SizeOfUninitializedData, e_oemid, e_cblp, e_oeminfo, NumberOfSymbols, SizeOfCode, e_minalloc, e_csum, e_sp
Model 7	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion, BaseOfCode, e_cp, SizeOfUninitializedData, e_oemid, e_cblp, e_oeminfo, NumberOfSymbols, SizeOfCode, e_minalloc, e_csum

Table A.1.3. Overview of prediction models of PE Header features selected by Relief-F (Part 3)

Models	Predictor Variables
Model 8	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion, BaseOfCode, e_cp, SizeOfUninitializedData, e_oemid, e_cblp, e_oeminfo, NumberOfSymbols, SizeOfCode, e_minalloc
Model 9	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion, BaseOfCode, e_cp, SizeOfUninitializedData, e_oemid, e_cblp, e_oeminfo, NumberOfSymbols, SizeOfCode
Model 10	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion, BaseOfCode, e_cp, SizeOfUninitializedData, e_oemid, e_cblp, e_oeminfo, NumberOfSymbols
Model 11	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion, BaseOfCode, e_cp, SizeOfUninitializedData, e_oemid, e_cblp, e_oeminfo

Table A.1.4. Overview of prediction models of PE Header features selected by Relief-F (Part 4)

Models	Predictor Variables
Model 12	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion, BaseOfCode, e_cp, SizeOfUninitializedData, e_oemid, e_cblp
Model 13	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion, BaseOfCode, e_cp, SizeOfUninitializedData, e_oemid
Model 14	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion, BaseOfCode, e_cp, SizeOfUninitializedData
Model 15	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion, BaseOfCode, e_cp
Model 16	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion, BaseOfCode

Table A.1.5. Overview of prediction models of PE Header features selected by Relief-F (Part 5)

Models	Predictor Variables
Model 17	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum, MinorImageVersion
Model 18	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion, CheckSum
Model 19	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders, MajorImageVersion
Model 20	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit, SizeOfHeaders
Model 21	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine, SizeOfStackCommit

Table A.1.6. Overview of prediction models of PE Header features selected by Relief-F (Part 6)

Models	Predictor Variables
Model 22	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader, Machine
Model 23	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic, SizeOfOptionalHeader
Model 24	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint, Magic
Model 25	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes, AddressOfEntryPoint
Model 26	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData, NumberOfRvaAndSizes
Model 27	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion, BaseOfData

Table A.1.7. Overview of prediction models of PE Header features selected by Relief-F (Part 7)

Models	Predictor Variables
Model 28	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData, MajorLinkerVersion
Model 29	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc, SizeOfInitializedData
Model 30	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc, e_lfarlc
Model 31	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno, e_maxalloc
Model 32	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage, e_ovno
Model 33	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr, SizeOfImage
Model 34	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion, e_cparhdr
Model 35	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion, MinorOperatingSystemVersion

Table A.1.8. Overview of prediction models of PE Header features selected by Relief-F (Part 8)

Models	Predictor Variables
Model 36	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit, MinorSubsystemVersion
Model 37	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion, SizeOfHeapCommit
Model 38	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem, MinorLinkerVersion
Model 39	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment, Subsystem
Model 40	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections, SectionAlignment
Model 41	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve, NumberOfSections
Model 42	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve, SizeOfHeapReserve
Model 43	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew, SizeOfStackReserve
Model 44	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase, e_lfanew
Model 45	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion, ImageBase
Model 46	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion, MajorOperatingSystemVersion
Model 47	DllCharacteristics, CreationYear, Characteristics, FileAlignment, MajorSubsystemVersion
Model 48	DllCharacteristics, CreationYear, Characteristics, FileAlignment
Model 49	DllCharacteristics, CreationYear, Characteristics
Model 50	DllCharacteristics, CreationYear
Model 51	DllCharacteristics

A.2. Overview of mRMR-Based Models

Sequential models were created based on the importance levels of the selected PE file headers using the mRMR algorithm. Additionally, model 52 represents the regular model, which contains all the features in the dataset. Table A.2.1 through Table A.2.9 offer an overview of the models generated utilizing the mRMR algorithm.

Table A.2.1. Overview of prediction models of PE Header features selected by mRMR (Part 1)

Models	Predictor Variables
Model 52	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode, SizeOfOptionalHeader, e_oemid, NumberOfRvaAndSizes, NumberOfSymbols, AddressOfEntryPoint, CheckSum, e_cp, e_oeminfo, e_cs, e_csum, MajorImageVersion, e_ip, e_ss, e_sp, e_cblp, MinorImageVersion
Model 53	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode, SizeOfOptionalHeader, e_oemid, NumberOfRvaAndSizes, NumberOfSymbols, AddressOfEntryPoint, CheckSum, e_cp, e_oeminfo, e_cs, e_csum, MajorImageVersion, e_ip, e_ss, e_sp, e_cblp
Model 54	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode, SizeOfOptionalHeader, e_oemid, NumberOfRvaAndSizes, NumberOfSymbols, AddressOfEntryPoint, CheckSum, e_cp, e_oeminfo, e_cs, e_csum, MajorImageVersion, e_ip, e_ss, e_sp

Table A.2.2. Overview of prediction models of PE Header features selected by mRMR (Part 2)

Models	Predictor Variables
Model 55	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode, SizeOfOptionalHeader, e_oemid, NumberOfRvaAndSizes, NumberOfSymbols, AddressOfEntryPoint, CheckSum, e_cp, e_oeminfo, e_cs, e_csum, MajorImageVersion, e_ip, e_ss
Model 56	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode, SizeOfOptionalHeader, e_oemid, NumberOfRvaAndSizes, NumberOfSymbols, AddressOfEntryPoint, CheckSum, e_cp, e_oeminfo, e_cs, e_csum, MajorImageVersion, e_ip
Model 57	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode, SizeOfOptionalHeader, e_oemid, NumberOfRvaAndSizes, NumberOfSymbols, AddressOfEntryPoint, CheckSum, e_cp, e_oeminfo, e_cs, e_csum, MajorImageVersion
Model 58	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode, SizeOfOptionalHeader, e_oemid, NumberOfRvaAndSizes, NumberOfSymbols, AddressOfEntryPoint, CheckSum, e_cp, e_oeminfo, e_cs, e_csum

Table A.2.3. Overview of prediction models of PE Header features selected by mRMR (Part 3)

Models	Predictor Variables
Model 59	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode, SizeOfOptionalHeader, e_oemid, NumberOfRvaAndSizes, NumberOfSymbols, AddressOfEntryPoint, CheckSum, e_cp, e_oeminfo, e_cs
Model 60	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode, SizeOfOptionalHeader, e_oemid, NumberOfRvaAndSizes, NumberOfSymbols, AddressOfEntryPoint, CheckSum, e_cp, e_oeminfo
Model 61	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode, SizeOfOptionalHeader, e_oemid, NumberOfRvaAndSizes, NumberOfSymbols, AddressOfEntryPoint, CheckSum, e_cp
Model 62	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode, SizeOfOptionalHeader, e_oemid, NumberOfRvaAndSizes, NumberOfSymbols, AddressOfEntryPoint, CheckSum

Table A.2.4. Overview of prediction models of PE Header features selected by mRMR (Part 4)

Models	Predictor Variables
Model 63	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode, SizeOfOptionalHeader, e_oemid, NumberOfRvaAndSizes, NumberOfSymbols, AddressOfEntryPoint
Model 64	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode, SizeOfOptionalHeader, e_oemid, NumberOfRvaAndSizes, NumberOfSymbols
Model 65	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode, SizeOfOptionalHeader, e_oemid, NumberOfRvaAndSizes
Model 66	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode, SizeOfOptionalHeader, e_oemid

Table A.2.5. Overview of prediction models of PE Header features selected by mRMR (Part 5)

Models	Predictor Variables
Model 67	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode, SizeOfOptionalHeader
Model 68	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic, BaseOfCode
Model 69	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags, Magic
Model 70	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode, LoaderFlags
Model 71	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable, SizeOfCode

Table A.2.6. Overview of prediction models of PE Header features selected by mRMR (Part 6)

Models	Predictor Variables
Model 72	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit, PointerToSymbolTable
Model 73	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc, SizeOfStackCommit
Model 74	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine, e_minalloc
Model 75	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData, Machine
Model 76	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc, SizeOfUninitializedData
Model 77	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage, e_maxalloc

Table A.2.7. Overview of prediction models of PE Header features selected by mRMR (Part 7)

Models	Predictor Variables
Model 78	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders, SizeOfImage
Model 79	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc, SizeOfHeaders
Model 80	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics, e_lfarlc
Model 81	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion, Characteristics
Model 82	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData, MajorLinkerVersion
Model 83	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion, BaseOfData
Model 84	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics, MinorLinkerVersion

Table A.2.8. Overview of prediction models of PE Header features selected by mRMR (Part 8)

Models	Predictor Variables
Model 85	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear, DllCharacteristics
Model 86	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections, CreationYear
Model 87	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr, NumberOfSections
Model 88	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData, e_cparhdr
Model 89	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion, SizeOfInitializedData
Model 90	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment, MajorOperatingSystemVersion
Model 91	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve, FileAlignment
Model 92	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno, SizeOfHeapReserve
Model 93	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment, e_ovno
Model 94	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit, SectionAlignment
Model 95	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion, SizeOfHeapCommit
Model 96	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion, MinorOperatingSystemVersion

Table A.2.9. Overview of prediction models of PE Header features selected by mRMR (Part 9)

Models	Predictor Variables
Model 97	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve, MajorSubsystemVersion
Model 98	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem, SizeOfStackReserve
Model 99	ImageBase, e_lfanew, MinorSubsystemVersion, Subsystem
Model 100	ImageBase, e_lfanew, MinorSubsystemVersion
Model 101	ImageBase, e_lfanew
Model 102	ImageBase

A.3. Selected Hyperparameters of Relief-F-Based Models

For Relief-F-based models, the best-selected parameters for LightGBM, DT, KNN, MLP, SVM, and NB machine learning methods using the Optuna hyperparameter optimization algorithm are presented Table A.3.1 through Table A.3.13.

Table A.3.1. The selected parameters of LightGBM for Relief-F-Based models (Part 1)

Model	Parameters								
	LR	NL	MD	MCS	S	CB	A	L	CW
Model 1	0.51	465	32	52	0.60	0.40	0.00	0.00	True
Model 2	0.51	373	14	14	0.10	0.40	0.00	0.20	False
Model 3	0.31	108	47	41	0.10	0.40	0.00	0.00	True
Model 4	0.21	463	48	37	0.10	0.40	0.00	0.50	True
Model 5	0.51	201	40	3	0.60	0.30	0.00	0.60	False
Model 6	0.21	393	33	45	0.60	0.40	0.00	0.20	False
Model 7	0.21	379	48	39	0.10	0.40	0.00	0.10	True
Model 8	0.21	113	40	18	0.60	0.30	0.00	0.30	False
Model 9	0.21	422	47	15	0.10	0.30	0.00	0.70	True
Model 10	0.41	385	41	26	0.10	0.20	0.00	0.40	True
Model 11	0.11	161	32	22	0.10	0.40	0.10	0.10	True
Model 12	0.21	222	48	30	0.60	0.30	0.00	0.00	True
Model 13	0.11	69	26	16	0.60	0.40	0.00	0.60	True
Model 14	0.21	298	21	20	0.60	0.40	0.00	0.10	True
Model 15	0.31	255	22	41	0.60	0.20	0.00	0.60	True
Model 16	0.21	62	41	3	0.10	0.40	0.10	0.30	True
Model 17	0.11	75	28	11	0.10	0.40	0.00	0.10	True
Model 18	0.21	224	38	31	0.10	0.40	0.10	0.60	True
Model 19	0.21	259	32	42	0.60	0.70	0.00	0.00	True
Model 20	0.31	244	40	42	0.60	1.00	0.00	0.00	True
Model 21	0.21	102	22	43	0.10	0.80	0.00	0.00	False

Abbreviations: LR, Learning Rate; NL, Num Leaves; MD, Max Depth; MCS, Min Child Samples; S, Subsample; CB, Colsample Bytree; A, Alpha; L, Lambda; CW, Class Weight

Table A.3.2. The selected parameters of LightGBM for Relief-F-Based models (Part 2)

Model	Parameters								
	LR	**NL**	**MD**	**MCS**	**S**	**CB**	**A**	**L**	**CW**
Model 22	0.61	461	44	3	0.10	0.80	0.00	0.50	False
Model 23	0.21	239	47	51	0.60	1.00	0.00	0.00	False
Model 24	0.21	331	11	22	0.10	1.00	0.00	0.00	False
Model 25	0.51	181	10	10	0.60	0.50	0.00	1.00	True
Model 26	0.31	63	12	2	0.60	0.20	0.30	0.30	True
Model 27	0.31	324	30	7	0.60	0.20	0.00	0.30	False
Model 28	0.41	234	16	1	0.10	0.30	0.30	0.40	True
Model 29	0.31	464	22	7	0.10	0.40	0.20	1.00	False
Model 30	0.11	476	49	4	0.10	0.40	0.00	0.80	True
Model 31	0.11	438	27	5	0.60	0.40	0.10	0.60	False
Model 32	0.11	286	16	2	0.60	0.40	0.20	0.80	True
Model 33	0.11	169	11	8	0.60	0.70	0.10	0.60	True
Model 34	0.11	499	18	1	0.60	0.40	0.10	0.20	True
Model 35	0.31	82	23	1	0.60	0.70	0.00	0.90	False
Model 36	0.21	497	37	8	0.60	1.00	0.00	0.00	False
Model 37	0.21	434	9	10	0.60	0.90	0.00	0.40	True
Model 38	0.21	447	20	3	0.60	0.80	0.00	0.50	True
Model 39	0.31	424	8	7	0.10	0.90	0.20	0.20	False
Model 40	0.41	168	48	24	0.60	0.90	0.30	0.30	False
Model 41	0.41	21	37	11	0.10	0.30	0.00	0.80	False
Model 42	0.21	382	37	1	0.60	0.50	0.40	0.00	True
Model 43	0.61	48	17	59	0.60	1.00	0.30	0.80	True
Model 44	0.31	252	14	8	0.10	0.90	0.00	0.30	False
Model 45	0.31	248	17	22	0.10	0.70	0.30	0.70	False
Model 46	0.31	226	16	36	0.10	0.80	0.20	0.70	False
Model 47	0.31	472	41	18	0.60	0.70	0.40	0.60	False
Model 48	0.51	20	33	45	0.10	0.70	0.400	0.70	False
Model 49	0.81	422	45	11	0.60	0.90	0.50	0.20	False
Model 50	0.41	91	3	6	0.60	1.00	0.00	0.80	False
Model 51	0.61	40	24	8	0.60	0.20	0.90	0.70	False

Abbreviations: LR, Learning Rate; NL, Num Leaves; MD, Max Depth; MCS, Min Child Samples; S, Subsample; CB, Colsample Bytree; A, Alpha; L, Lambda; CW, Class Weigh

Table A.3.3. The selected parameters of DT for Relief-F-Based models (Part 1)

Model	Parameters						
	C	S	MD	MSS	MSL	MF	CW
Model 1	entropy	best	83	6	1	None	None
Model 2	log_loss	random	92	11	1	None	None
Model 3	entropy	best	25	6	1	None	None
Model 4	log_loss	random	97	6	1	None	balanced
Model 5	entropy	random	42	6	1	None	None
Model 6	log_loss	best	45	6	1	None	None
Model 7	entropy	best	18	6	1	None	None
Model 8	log_loss	best	22	6	1	None	None
Model 9	entropy	best	63	6	1	None	None
Model 10	entropy	random	99	6	1	None	balanced
Model 11	log_loss	random	62	6	1	None	balanced
Model 12	entropy	random	60	6	1	None	None
Model 13	entropy	best	45	6	1	None	balanced
Model 14	log_loss	best	16	6	1	None	balanced
Model 15	entropy	random	88	6	1	None	None
Model 16	log_loss	best	75	6	1	None	None
Model 17	log_loss	best	27	6	1	None	None
Model 18	log_loss	best	62	6	1	None	balanced
Model 19	log_loss	random	97	9	1	None	None
Model 20	entropy	random	57	6	1	None	balanced
Model 21	log_loss	best	32	19	5	None	None
Model 22	entropy	best	31	6	1	sqrt	balanced
Model 23	log_loss	random	98	6	1	None	balanced
Model 24	entropy	random	20	6	1	None	None
Model 25	gini	random	None	9	1	None	None
Model 26	gini	best	33	8	1	None	balanced
Model 27	gini	random	75	6	1	None	balanced
Model 28	log_loss	random	60	8	1	None	balanced
Model 29	entropy	best	93	6	1	None	None
Model 30	entropy	best	80	9	1	None	None
Model 31	log_loss	best	69	6	1	None	balanced
Model 32	log_loss	best	91	6	1	None	None
Model 33	entropy	best	36	6	1	None	None
Model 34	entropy	best	83	6	1	None	None
Model 35	log_loss	best	49	6	1	None	None
Model 36	entropy	random	62	10	1	None	balanced
Model 37	entropy	random	89	6	1	None	balanced
Model 38	entropy	best	68	6	1	None	None

Abbreviations: C, Criterion; S, Splitter; MD, Maximum Depth; MSS, Minimum Samples Split; MSL, Minimum Samples Leaf; MF, Maximum Features; CW, Class Weight

Table A.3.4. The selected parameters of DT for Relief-F-Based models (Part 2)

Model	Parameters						
	C	S	MD	MSS	MSL	MF	CW
Model 39	log_loss	best	97	6	1	None	None
Model 40	entropy	random	72	6	1	None	None
Model 41	gini	best	15	6	1	log2	None
Model 42	gini	best	80	6	1	sqrt	balanced
Model 43	gini	best	65	6	1	None	balanced
Model 44	log_loss	random	71	6	1	None	balanced
Model 45	gini	best	82	10	1	None	balanced
Model 46	entropy	best	97	11	1	None	None
Model 47	entropy	best	92	6	1	None	balanced
Model 48	log_loss	best	59	16	1	None	None
Model 49	gini	best	51	6	1	None	None
Model 50	entropy	best	98	12	1	log2	None
Model 51	entropy	random	41	116	1	log2	balanced

Abbreviations: C, Criterion; S, Splitter; MD, Maximum Depth; MSS, Minimum Samples Split; MSL, Minimum Samples Leaf; MF, Maximum Features; CW, Class Weight

Table A.3.5. The selected parameters of KNN for Relief-F-Based models (Part 1)

Model	Parameters					
	NN	W	A	LS	P	M
Model 1	1	distance	kd_tree	38	1.45	manhattan
Model 2	1	distance	auto	42	1.25	manhattan
Model 3	1	distance	brute	26	1.21	manhattan
Model 4	2	distance	brute	55	1.06	manhattan
Model 5	1	distance	kd_tree	75	1.94	manhattan
Model 6	2	distance	auto	67	1.39	manhattan
Model 7	1	distance	kd_tree	12	1.04	manhattan
Model 8	1	distance	auto	30	1.41	manhattan
Model 9	1	uniform	kd_tree	82	1.90	manhattan
Model 10	1	distance	ball_tree	10	1.11	manhattan
Model 11	2	distance	ball_tree	12	1.17	manhattan
Model 12	2	distance	auto	64	1.70	manhattan
Model 13	1	distance	kd_tree	91	1.47	manhattan
Model 14	1	distance	ball_tree	12	1.05	manhattan
Model 15	2	distance	brute	42	1.83	manhattan
Model 16	1	distance	ball_tree	96	1.34	manhattan
Model 17	1	uniform	ball_tree	92	1.01	manhattan

Abbreviations: NN, Nearest Neighbors; W, Weights; A, Algorithm; LS, Leaf Size; P, P-Value; M, Metric

Table A.3.6. The selected parameters of KNN for Relief-F-Based models (Part 2)

Model	Parameters					
	NN	W	A	LS	P	M
Model 18	1	uniform	kd_tree	90	1.89	manhattan
Model 19	2	distance	auto	39	1.60	manhattan
Model 20	2	distance	kd_tree	84	1.26	manhattan
Model 21	1	distance	kd_tree	10	1.11	manhattan
Model 22	1	distance	ball_tree	95	1.02	manhattan
Model 23	2	distance	auto	20	1.01	manhattan
Model 24	2	distance	kd_tree	65	1.86	manhattan
Model 25	1	uniform	ball_tree	100	1.78	manhattan
Model 26	1	distance	brute	59	1.04	manhattan
Model 27	1	distance	ball_tree	98	1.03	manhattan
Model 28	1	distance	brute	32	2.13	minkowski
Model 29	1	distance	ball_tree	93	1.07	euclidean
Model 30	1	distance	kd_tree	71	2.00	minkowski
Model 31	1	distance	brute	55	2.01	minkowski
Model 32	1	distance	brute	35	2.02	minkowski
Model 33	1	distance	ball_tree	89	2.00	euclidean
Model 34	16	distance	ball_tree	20	1.89	manhattan
Model 35	16	distance	kd_tree	60	1.63	manhattan
Model 36	16	distance	kd_tree	13	1.36	manhattan
Model 37	16	distance	auto	62	1.09	manhattan
Model 38	16	distance	ball_tree	15	1.28	manhattan
Model 39	15	distance	ball_tree	99	1.69	manhattan
Model 40	21	distance	kd_tree	83	1.57	manhattan
Model 41	18	distance	auto	23	3.25	minkowski
Model 42	24	distance	brute	37	1.95	euclidean
Model 43	22	distance	brute	58	1.31	euclidean
Model 44	20	distance	brute	36	1.34	euclidean
Model 45	24	distance	auto	50	1.49	manhattan
Model 46	37	distance	brute	34	1.04	euclidean
Model 47	24	distance	brute	66	1.15	manhattan
Model 48	24	distance	ball_tree	75	1.66	euclidean
Model 49	70	distance	brute	69	9.92	minkowski
Model 50	66	distance	ball_tree	78	2.04	minkowski
Model 51	67	distance	auto	40	9.69	minkowski

Abbreviations: NN, Nearest Neighbors; W, Weights; A, Algorithm; LS, Leaf Size; P, P-Value; M, Metric

Table A.3.7. The selected parameters of MLP for Relief-F-Based models (Part 1)

Model	Parameters						
	HLS	**AC**	**A**	**S**	**LR**	**BS**	**M**
Model 1	94	tanh	0.00046	lbfgs	adaptive	Auto	0.90
Model 2	95	tanh	0.55000	lbfgs	constant	Auto	0.90
Model 3	97	tanh	0.36000	lbfgs	invscaling	Auto	0.90
Model 4	94	tanh	0.06000	lbfgs	adaptive	Auto	0.90
Model 5	93	tanh	0.99000	lbfgs	constant	Auto	0.90
Model 6	98	tanh	2.78000	lbfgs	invscaling	Auto	0.90
Model 7	96	tanh	0.02000	lbfgs	constant	Auto	0.90
Model 8	91	tanh	0.00030	lbfgs	adaptive	Auto	0.90
Model 9	98	tanh	0.20000	lbfgs	adaptive	Auto	0.90
Model 10	95	tanh	0.00200	lbfgs	constant	Auto	0.90
Model 11	92	tanh	2.41000	lbfgs	constant	Auto	0.90
Model 12	100	tanh	0.01000	lbfgs	adaptive	Auto	0.90
Model 13	93	tanh	0.00050	lbfgs	adaptive	Auto	0.90
Model 14	98	tanh	0.03200	lbfgs	constant	Auto	0.90
Model 15	95	tanh	0.03400	lbfgs	invscaling	Auto	0.90
Model 16	92	tanh	0.44000	lbfgs	invscaling	Auto	0.90
Model 17	92	tanh	4.58000	lbfgs	constant	Auto	0.90
Model 18	100	tanh	0.19000	lbfgs	adaptive	Auto	0.90
Model 19	98	tanh	0.00010	lbfgs	adaptive	Auto	0.90
Model 20	100	tanh	0.00080	lbfgs	adaptive	Auto	0.90
Model 21	91	tanh	0.17000	lbfgs	invscaling	Auto	0.90
Model 22	92	tanh	0.00010	lbfgs	adaptive	Auto	0.90
Model 23	98	tanh	0.00010	lbfgs	adaptive	Auto	0.90
Model 24	98	tanh	0.00200	lbfgs	adaptive	Auto	0.90
Model 25	99	tanh	0.01300	lbfgs	invscaling	Auto	0.90
Model 26	98	tanh	0.08000	lbfgs	constant	Auto	0.90
Model 27	100	tanh	0.00200	lbfgs	constant	Auto	0.90
Model 28	98	tanh	0.01100	lbfgs	adaptive	Auto	0.90
Model 29	99	tanh	0.02000	lbfgs	invscaling	Auto	0.90
Model 30	95	tanh	0.00070	lbfgs	adaptive	Auto	0.90
Model 31	94	tanh	0.62000	lbfgs	invscaling	Auto	0.90
Model 32	98	tanh	0.00100	lbfgs	adaptive	Auto	0.90
Model 33	95	logistic	0.10000	lbfgs	adaptive	Auto	0.90
Model 34	98	logistic	0.00020	lbfgs	adaptive	Auto	0.90
Model 35	99	logistic	0.00010	lbfgs	constant	Auto	0.90
Model 36	94	tanh	0.00500	lbfgs	adaptive	Auto	0.90
Model 37	100	tanh	0.00080	lbfgs	constant	Auto	0.90
Model 38	100	tanh	0.22000	lbfgs	adaptive	Auto	0.90
Model 39	92	tanh	0.00020	lbfgs	constant	Auto	0.90

Abbreviations: HLS, Hidden Layer Sizes; AC, Activation Function; A, Alpha; S, Solver; LR, Learning Rate; BS, Batch Size, M, Momentum

Table A.3.8. The selected parameters of MLP for Relief-F-Based models (Part 2)

Model	Parameters						
	HLS	**AC**	**A**	**S**	**LR**	**BS**	**M**
Model 40	100	tanh	0.00010	lbfgs	constant	Auto	0.90
Model 41	89	logistic	0.00900	lbfgs	invscaling	Auto	0.90
Model 42	100	tanh	0.47000	lbfgs	invscaling	Auto	0.90
Model 43	89	tanh	0.86000	lbfgs	constant	Auto	0.90
Model 44	75	logistic	4.68000	lbfgs	constant	Auto	0.90
Model 45	33	logistic	8.10000	lbfgs	invscaling	Auto	0.90
Model 46	86	tanh	0.00400	lbfgs	invscaling	Auto	0.90
Model 47	50	logistic	0.77000	lbfgs	adaptive	Auto	0.90
Model 48	63	tanh	0.07000	lbfgs	constant	Auto	0.90
Model 49	98	tanh	0.09000	lbfgs	adaptive	Auto	0.90
Model 50	95	logistic	0.12000	lbfgs	adaptive	Auto	0.90
Model 51	88	relu	0.83000	adam	invscaling	197	0.90

Abbreviations: HLS, Hidden Layer Sizes; AC, Activation Function; A, Alpha; S, Solver; LR, Learning Rate; BS, Batch Size, M, Momentum

Table A.3.9. The selected parameters of SVM for Relief-F-Based models (Part 1)

Model	Parameters				
	C	**Kernel**	**Class Weight**	**Gamma**	**Degree**
Model 1	44	rbf	balanced	1	3
Model 2	47	rbf	balanced	1	3
Model 3	54	rbf	None	1	3
Model 4	90	rbf	balanced	1	3
Model 5	46	rbf	None	1	3
Model 6	47	rbf	balanced	1	3
Model 7	39	rbf	balanced	1	3
Model 8	44	rbf	balanced	1	3
Model 9	43	rbf	balanced	1	3
Model 10	86	poly	balanced	scale	2
Model 11	39	rbf	balanced	1	3
Model 12	61	rbf	None	1	3
Model 13	38	rbf	balanced	1	3
Model 14	88	rbf	None	1	3
Model 15	43	rbf	balanced	1	3
Model 16	93	poly	balanced	scale	3
Model 17	52	rbf	balanced	1	3
Model 18	38	rbf	balanced	1	3
Model 19	94	poly	balanced	scale	3
Model 20	48	rbf	None	1	3
Model 21	86	poly	None	scale	2

Table A.3.10. The selected parameters of SVM for Relief-F-Based models (Part 2)

Model	Parameters				
	C	Kernel	Class Weight	Gamma	Degree
Model 22	77	rbf	None	1	3
Model 23	38	rbf	balanced	1	3
Model 24	36	rbf	None	1	3
Model 25	59	rbf	None	1	3
Model 26	92	poly	None	scale	2
Model 27	30	rbf	None	1	3
Model 28	51	rbf	None	1	3
Model 29	69	rbf	balanced	1	3
Model 30	52	rbf	balanced	1	3
Model 31	33	rbf	None	1	3
Model 32	31	rbf	balanced	1	3
Model 33	8	rbf	balanced	1	3
Model 34	87	rbf	None	1	3
Model 35	31	rbf	None	1	3
Model 36	7	rbf	None	1	3
Model 37	31	rbf	None	1	3
Model 38	23	rbf	None	1	3
Model 39	26	rbf	balanced	1	3
Model 40	83	rbf	None	2	3
Model 41	94	rbf	balanced	2	3
Model 42	83	rbf	balanced	2	3
Model 43	81	rbf	balanced	2	3
Model 44	100	rbf	None	4	3
Model 45	95	rbf	balanced	52	3
Model 46	84	rbf	None	92	3
Model 47	92	rbf	None	89	3
Model 48	96	rbf	None	83	3
Model 49	95	rbf	None	63	3
Model 50	46	rbf	None	89	3
Model 51	100	rbf	None	72	3

Table A.3.11. The selected parameters of NB for Relief-F-Based models (Part 1)

Model	Parameters
	var_smoothing
Model 1	184.56164
Model 2	170.84569
Model 3	155.91072
Model 4	142.28591
Model 5	148.10279
Model 6	136.02582

Table A.3.12. The selected parameters of NB for Relief-F-Based models (Part 2)

Model	Parameters
	var_smoothing
Model 7	121.76716
Model 8	115.18054
Model 9	99.07204
Model 10	85.25854
Model 11	97.96841
Model 12	83.87796
Model 13	66.71953
Model 14	57.15202
Model 15	0.00001
Model 16	0.00004
Model 17	0.00022
Model 18	0.00017
Model 19	0.01200
Model 20	0.01061
Model 21	0.00257
Model 22	0.01285
Model 23	0.00059
Model 24	26.91273
Model 25	42.40494
Model 26	32.57000
Model 27	45.59008
Model 28	44.34461
Model 29	31.98123
Model 30	46.96647
Model 31	35.71593
Model 32	23.58280
Model 33	3.20572
Model 34	20.34453
Model 35	1.91422
Model 36	3.44194
Model 37	18.10178
Model 38	3.43371
Model 39	1.54156
Model 40	8.58220
Model 41	16.77877
Model 42	20.27583
Model 43	3.95731
Model 44	0.00973
Model 45	0.01587
Model 46	12.47303
Model 47	7.82197
Model 48	0.00018

Table A.3.13. The selected parameters of NB for Relief-F-Based models (Part 3)

Model	Parameters
	var_smoothing
Model 49	3.67782
Model 50	11.35339
Model 51	7.27149

A.4. Selected Hyperparameters of mRMR-Based Models

The optimal parameters selected for LightGBM, DT, KNN, MLP, SVM, and NB machine learning methods using the Optuna hyperparameter optimization algorithm for mRMR-based models are presented in Tables A.4.1 through A.4.14.

Table A.4.1. The selected parameters of LightGBM for mRMR-Based models (Part 1)

Model	Parameters								
	LR	**NL**	**MD**	**MCS**	**S**	**CB**	**A**	**L**	**CW**
Model 52	0.31	253	50	56	0.60	0.40	0.00	0.00	True
Model 53	0.11	218	17	42	0.60	0.20	0.00	0.30	True
Model 54	0.21	318	27	19	0.60	0.40	0.00	0.30	False
Model 55	0.31	305	25	12	0.10	0.20	0.00	0.40	True
Model 56	0.31	500	18	19	0.10	0.30	0.00	0.00	False
Model 57	0.21	80	15	19	0.60	0.30	0.00	0.40	True
Model 58	0.71	333	10	1	0.10	0.50	0.00	1.00	True
Model 59	0.11	392	31	12	0.10	0.40	0.00	0.00	True
Model 60	0.11	342	28	35	0.10	0.40	0.00	0.00	True
Model 61	0.21	295	25	51	0.10	0.40	0.00	0.00	False
Model 62	0.61	288	32	26	0.10	0.40	0.00	0.10	True
Model 63	0.41	461	22	42	0.60	0.30	0.00	0.10	False
Model 64	0.21	469	11	23	0.10	0.40	0.00	0.00	True
Model 65	0.21	443	15	3	0.60	0.40	0.10	0.80	True
Model 66	0.21	416	49	15	0.10	0.40	0.00	0.20	False
Model 67	0.21	281	11	10	0.60	0.30	0.00	0.70	True
Model 68	0.21	371	24	15	0.10	0.40	0.00	0.10	False
Model 69	0.21	91	23	29	0.10	0.30	0.10	0.90	True
Model 70	0.41	56	44	52	0.60	0.40	0.00	0.00	False
Model 71	0.21	372	38	3	0.10	0.40	0.00	0.00	False
Model 72	0.61	105	42	14	0.60	0.30	0.00	0.40	True
Model 73	0.41	206	41	18	0.10	0.30	0.60	0.90	False
Model 74	0.81	446	26	3	0.10	0.60	0.00	0.80	True
Model 75	0.41	220	44	3	0.10	0.40	0.00	0.90	False

Abbreviations: LR, Learning Rate; NL, Num Leaves; MD, Max Depth; MCS, Min Child Samples; S, Subsample; CB, Colsample Bytree; A, Alpha; L, Lambda; CW, Class Weight

Table A.4.2. The selected parameters of LightGBM for mRMR-Based models (Part 2)

Model	Parameters								
	LR	NL	MD	MCS	S	CB	A	L	CW
Model 76	0.21	234	18	1	0.60	0.40	0.00	0.60	True
Model 77	0.61	289	38	50	0.60	0.80	0.00	0.00	True
Model 78	0.51	499	49	8	0.10	0.40	0.00	0.50	True
Model 79	0.41	54	43	15	0.60	0.40	0.00	0.10	False
Model 80	0.21	473	16	26	0.10	0.50	0.10	0.40	True
Model 81	0.11	129	28	5	0.60	0.40	0.00	0.00	False
Model 82	0.71	418	22	11	0.60	0.40	0.00	0.60	True
Model 83	0.61	409	34	11	0.10	0.50	0.00	0.20	False
Model 84	0.21	67	18	3	0.60	0.40	0.20	0.40	True
Model 85	0.21	359	34	6	0.60	0.80	0.50	1.00	False
Model 86	0.11	427	13	10	0.60	0.70	0.00	0.30	False
Model 87	0.11	106	17	11	0.10	0.90	0.50	0.00	True
Model 88	0.41	169	11	1	0.60	0.40	0.40	1.00	True
Model 89	0.31	241	27	3	0.60	0.50	0.80	0.50	False
Model 90	0.71	159	28	11	0.60	0.60	0.70	0.30	True
Model 91	0.81	96	44	9	0.60	0.60	0.70	0.20	False
Model 92	0.71	271	28	69	0.60	1.00	0.90	0.10	False
Model 93	0.21	110	28	3	0.10	0.40	0.40	0.80	True
Model 94	0.11	25	25	4	0.60	0.70	0.20	0.50	False
Model 95	0.51	493	19	9	0.10	0.50	0.80	0.00	False
Model 96	0.31	471	47	1	0.60	0.60	0.80	0.20	True
Model 97	0.61	370	34	1	0.10	0.70	0.80	0.10	True
Model 98	0.31	83	20	5	0.10	0.50	0.30	0.80	False
Model 99	0.91	435	11	3	0.10	0.60	0.10	1.00	False
Model 100	0.51	190	13	2	0.60	0.90	0.00	0.20	False
Model 101	0.01	347	24	1	0.60	0.90	0.00	0.00	False
Model 102	0.21	146	11	34	0.10	0.30	0.00	0.10	True

Abbreviations: LR, Learning Rate; NL, Num Leaves; MD, Max Depth; MCS, Min Child Samples; S, Subsample; CB, Colsample Bytree; A, Alpha; L, Lambda; CW, Class Weight

Table A.4.3. The selected parameters of DT for mRMR-Based models (Part 1)

Model	Parameters						
	C	S	MD	MSS	MSL	MF	CW
Model 52	entropy	best	None	6	1	None	None
Model 53	log_loss	random	29	6	1	None	None
Model 54	entropy	random	42	8	1	None	balanced
Model 55	entropy	random	62	6	1	None	None
Model 56	entropy	random	21	9	1	None	balanced

Abbreviations: C, Criterion; S, Splitter; MD, Max Depth; MSS, Min Samples Split; MSL, Min Samples Leaf; MF, Max Features; CW, Class Weight

Table A.4.4. The selected parameters of DT for mRMR-Based models (Part 2)

Model	Parameters						
	C	S	MD	MSS	MSL	MF	CW
Model 57	entropy	random	25	9	1	None	balanced
Model 58	log_loss	best	55	6	1	None	None
Model 59	entropy	best	34	6	1	None	None
Model 60	entropy	random	53	8	1	None	None
Model 61	entropy	random	42	9	1	None	None
Model 62	log_loss	best	24	6	1	None	balanced
Model 63	log_loss	random	79	6	1	None	balanced
Model 64	log_loss	best	41	6	1	sqrt	None
Model 65	log_loss	random	69	10	1	None	balanced
Model 66	entropy	random	83	6	1	None	None
Model 67	log_loss	random	73	8	1	None	balanced
Model 68	entropy	random	32	6	1	None	None
Model 69	gini	random	71	13	1	None	balanced
Model 70	entropy	best	38	6	1	log2	balanced
Model 71	log_loss	random	50	11	1	None	None
Model 72	entropy	random	78	8	1	None	balanced
Model 73	entropy	random	45	6	1	None	balanced
Model 74	entropy	random	77	6	1	None	balanced
Model 75	log_loss	random	64	9	1	None	balanced
Model 76	log_loss	random	54	13	1	None	balanced
Model 77	gini	random	41	6	1	None	None
Model 78	entropy	random	30	6	1	None	balanced
Model 79	entropy	random	78	6	1	None	balanced
Model 80	gini	random	92	10	1	None	None
Model 81	entropy	random	82	6	1	None	None
Model 82	log_loss	random	63	10	1	None	None
Model 83	gini	random	26	6	1	None	balanced
Model 84	entropy	random	22	12	1	None	None
Model 85	entropy	random	61	10	1	None	balanced
Model 86	log_loss	random	77	9	1	None	None
Model 87	entropy	best	66	10	1	sqrt	balanced
Model 88	entropy	random	74	14	1	None	None
Model 89	log_loss	random	34	12	1	None	balanced
Model 90	log_loss	best	26	8	1	sqrt	None
Model 91	entropy	best	14	9	1	None	balanced
Model 92	gini	best	57	19	1	sqrt	balanced
Model 93	entropy	best	14	9	1	None	balanced
Model 94	log_loss	best	55	7	1	sqrt	None
Model 95	entropy	best	84	11	1	None	None

Abbreviations: C, Criterion; S, Splitter; MD, Max Depth; MSS, Min Samples Split; MSL, Min Samples Leaf; MF, Max Features; CW, Class Weight

Table A.4.5. The selected parameters of DT for mRMR-Based models (Part 3)

Model	Parameters						
	C	S	MD	MSS	MSL	MF	CW
Model 96	gini	random	29	8	1	None	None
Model 97	log_loss	best	57	16	1	log2	None
Model 98	entropy	random	36	12	1	None	None
Model 99	log_loss	best	53	9	1	log2	None
Model 100	gini	best	86	15	4	None	None
Model 101	entropy	best	66	90	5	sqrt	None
Model 102	entropy	best	7	121	3	sqrt	None

Abbreviations: C, Criterion; S, Splitter; MD, Max Depth; MSS, Min Samples Split; MSL, Min Samples Leaf; MF, Max Features; CW, Class Weight

Table A.4.6. The selected parameters of KNN for mRMR-Based models (Part 1)

Model	Parameters					
	NN	W	A	LS	P	M
Model 52	1	distance	brute	41	1.66	manhattan
Model 53	2	distance	ball_tree	43	1.40	manhattan
Model 54	1	uniform	ball_tree	76	1.99	manhattan
Model 55	2	distance	auto	53	1.17	manhattan
Model 56	1	uniform	auto	14	1.13	manhattan
Model 57	1	distance	auto	35	1.11	manhattan
Model 58	2	distance	kd_tree	91	1.83	manhattan
Model 59	1	distance	ball_tree	94	1.89	manhattan
Model 60	2	distance	kd_tree	73	1.35	manhattan
Model 61	1	uniform	brute	51	1.20	manhattan
Model 62	1	distance	auto	51	1.94	manhattan
Model 63	1	distance	ball_tree	85	1.61	manhattan
Model 64	2	distance	kd_tree	15	1.24	manhattan
Model 65	1	distance	auto	68	1.90	manhattan
Model 66	1	distance	kd_tree	34	1.20	manhattan
Model 67	2	distance	ball_tree	48	1.01	manhattan
Model 68	2	distance	brute	67	1.14	manhattan
Model 69	1	distance	kd_tree	99	1.06	manhattan
Model 70	1	distance	ball_tree	85	1.85	manhattan
Model 71	2	distance	kd_tree	22	1.35	manhattan
Model 72	1	uniform	auto	19	1.54	manhattan
Model 73	1	distance	brute	18	1.99	manhattan
Model 74	1	distance	ball_tree	81	1.01	manhattan
Model 75	1	uniform	kd_tree	89	1.03	manhattan
Model 76	1	distance	auto	66	1.25	manhattan

Abbreviations: NN, Nearest Neighbors; W, Weights; A, Algorithm; LS, Leaf Size; P, P-Value; M, Metric

Table A.4.7. The selected parameters of KNN for mRMR-Based models (Part 2)

Model	Parameters					
	NN	W	A	LS	P	M
Model 77	1	distance	ball_tree	91	1.23	manhattan
Model 78	1	distance	ball_tree	85	1.31	manhattan
Model 79	2	distance	ball_tree	95	1.22	manhattan
Model 80	1	distance	kd_tree	85	1.82	manhattan
Model 81	1	distance	brute	11	1.96	manhattan
Model 82	1	uniform	ball_tree	76	1.37	manhattan
Model 83	1	uniform	auto	25	1.84	manhattan
Model 84	1	distance	brute	69	1.49	euclidean
Model 85	6	distance	ball_tree	89	1.22	manhattan
Model 86	1	distance	brute	27	4.51	minkowski
Model 87	15	distance	kd_tree	86	1.98	euclidean
Model 88	13	distance	brute	22	8.92	minkowski
Model 89	14	distance	kd_tree	26	2.89	minkowski
Model 90	33	distance	ball_tree	37	1.38	euclidean
Model 91	65	distance	ball_tree	55	1.36	euclidean
Model 92	55	distance	ball_tree	79	2.95	minkowski
Model 93	57	distance	ball_tree	95	3.49	minkowski
Model 94	67	distance	brute	70	7.10	minkowski
Model 95	70	distance	kd_tree	94	1.20	manhattan
Model 96	70	distance	kd_tree	22	1.95	manhattan
Model 97	69	distance	auto	30	1.78	manhattan
Model 98	61	distance	auto	60	1.34	manhattan
Model 99	69	distance	ball_tree	81	4.32	minkowski
Model 100	69	distance	kd_tree	90	1.61	manhattan
Model 101	48	distance	auto	21	3.23	minkowski
Model 102	16	distance	brute	36	9.70	minkowski

Abbreviations: NN, Nearest Neighbors; W, Weights; A, Algorithm; LS, Leaf Size; P, P-Value; M, Metric

Table A.4.8. The selected parameters of MLP for mRMR-Based models (Part 1)

Model	Parameters						
	HLS	AC	A	S	LR	BS	M
Model 52	98	tanh	0.06657	lbfgs	adaptive	auto	0.90
Model 53	98	tanh	0.00949	lbfgs	adaptive	auto	0.90
Model 54	86	tanh	0.25390	lbfgs	invscaling	auto	0.90
Model 55	98	tanh	0.00201	lbfgs	constant	auto	0.90
Model 56	100	tanh	0.00389	lbfgs	invscaling	auto	0.90

Abbreviations: HLS, Hidden Layer Sizes; AC, Activation Function; A, Alpha; S, Solver; LR, Learning Rate; BS, Batch Size, M, Momentum

Table A.4.9. The selected parameters of MLP for mRMR-Based models (Part 2)

Model	Parameters						
	HLS	**AC**	**A**	**S**	**LR**	**BS**	**M**
Model 57	95	tanh	0.01618	lbfgs	adaptive	auto	0.90
Model 58	100	tanh	2.20411	lbfgs	adaptive	auto	0.90
Model 59	97	tanh	1.96107	lbfgs	adaptive	auto	0.90
Model 60	100	tanh	0.56926	lbfgs	invscaling	auto	0.90
Model 61	98	logistic	0.00142	lbfgs	invscaling	auto	0.90
Model 62	96	tanh	0.00330	lbfgs	adaptive	auto	0.90
Model 63	94	tanh	0.00687	lbfgs	adaptive	auto	0.90
Model 64	95	tanh	0.00101	lbfgs	invscaling	auto	0.90
Model 65	100	tanh	0.00802	lbfgs	invscaling	auto	0.90
Model 66	86	tanh	0.85940	lbfgs	constant	auto	0.90
Model 67	96	tanh	0.00149	lbfgs	constant	auto	0.90
Model 68	97	tanh	0.26294	lbfgs	constant	auto	0.90
Model 69	100	tanh	0.14510	lbfgs	constant	auto	0.90
Model 70	100	tanh	0.00165	lbfgs	adaptive	auto	0.90
Model 71	100	tanh	0.00136	lbfgs	adaptive	auto	0.90
Model 72	94	tanh	0.58337	lbfgs	constant	auto	0.90
Model 73	100	tanh	0.02762	lbfgs	invscaling	auto	0.90
Model 74	91	tanh	0.00035	lbfgs	constant	auto	0.90
Model 75	91	tanh	2.02878	lbfgs	invscaling	auto	0.90
Model 76	99	tanh	0.42458	lbfgs	adaptive	auto	0.90
Model 77	92	tanh	0.46492	lbfgs	invscaling	auto	0.90
Model 78	98	tanh	0.00458	lbfgs	invscaling	auto	0.90
Model 79	100	tanh	0.00025	lbfgs	constant	auto	0.90
Model 80	100	tanh	0.39301	lbfgs	constant	auto	0.90
Model 81	100	tanh	0.15540	lbfgs	invscaling	auto	0.90
Model 82	96	tanh	0.40086	lbfgs	constant	auto	0.90
Model 83	94	tanh	0.01162	lbfgs	adaptive	auto	0.90
Model 84	100	tanh	0.68803	lbfgs	invscaling	auto	0.90
Model 85	92	tanh	0.08653	lbfgs	adaptive	auto	0.90
Model 86	100	tanh	0.15894	lbfgs	constant	auto	0.90
Model 87	83	tanh	0.25161	lbfgs	adaptive	auto	0.90
Model 88	98	logistic	0.06792	lbfgs	adaptive	auto	0.90
Model 89	100	tanh	0.03806	lbfgs	constant	auto	0.90
Model 90	97	logistic	0.00078	lbfgs	constant	auto	0.90
Model 91	92	tanh	0.07386	lbfgs	adaptive	auto	0.90
Model 92	94	logistic	0.00084	lbfgs	invscaling	auto	0.90
Model 93	28	tanh	5.93890	lbfgs	invscaling	auto	0.90
Model 94	68	logistic	1.62931	lbfgs	adaptive	auto	0.90
Model 95	55	tanh	8.12807	lbfgs	invscaling	auto	0.90

Abbreviations: HLS, Hidden Layer Sizes; AC, Activation Function; A, Alpha; S, Solver; LR, Learning Rate; BS, Batch Size, M, Momentum

Table A.4.10. The selected parameters of MLP for mRMR-Based models (Part 3)

Model	Parameters						
	HLS	**AC**	**A**	**S**	**LR**	**BS**	**M**
Model 96	68	tanh	1.64642	lbfgs	adaptive	auto	0.90
Model 97	30	logistic	2.22902	lbfgs	adaptive	auto	0.90
Model 98	99	tanh	0.22618	lbfgs	constant	auto	0.90
Model 99	76	tanh	5.59804	lbfgs	invscaling	auto	0.90
Model 100	64	logistic	5.34287	lbfgs	invscaling	auto	0.90
Model 101	68	logistic	6.39165	lbfgs	constant	auto	0.90
Model 102	91	tanh	7.08021	lbfgs	invscaling	auto	0.90

Abbreviations: HLS, Hidden Layer Sizes; AC, Activation Function; A, Alpha; S, Solver; LR, Learning Rate; BS, Batch Size, M, Momentum

Table A.4.11. The selected parameters of SVM for mRMR-Based models (Part 1)

Model	Parameters				
	C	**Kernel**	**Class Weight**	**Gamma**	**Degree**
Model 52	46	rbf	None	1	3
Model 53	39	rbf	balanced	1	3
Model 54	61	rbf	None	1	3
Model 55	37	rbf	balanced	1	3
Model 56	63	rbf	balanced	1	3
Model 57	45	rbf	None	1	3
Model 58	38	rbf	balanced	1	3
Model 59	57	rbf	None	1	3
Model 60	39	rbf	balanced	1	3
Model 61	48	rbf	balanced	1	3
Model 62	92	rbf	None	1	3
Model 63	82	poly	balanced	scale	2
Model 64	88	rbf	None	1	3
Model 65	83	rbf	balanced	1	3
Model 66	78	poly	balanced	scale	3
Model 67	72	rbf	None	1	3
Model 68	94	rbf	balanced	1	3
Model 69	74	rbf	None	1	3
Model 70	66	rbf	balanced	1	3
Model 71	90	poly	None	scale	2
Model 72	73	rbf	None	1	3
Model 73	98	rbf	None	1	3
Model 74	81	rbf	None	1	3
Model 75	66	rbf	balanced	1	3
Model 76	67	rbf	None	1	3
Model 77	30	rbf	None	1	3

Table A.4.12. The selected parameters of SVM for mRMR-Based models (Part 2)

Model	Parameters				
	C	Kernel	Class Weight	Gamma	Degree
Model 78	47	rbf	None	1	3
Model 79	41	rbf	None	1	3
Model 80	41	rbf	None	1	3
Model 81	97	poly	balanced	scale	3
Model 82	65	rbf	None	1	3
Model 83	44	rbf	None	1	3
Model 84	3	rbf	balanced	1	3
Model 85	17	rbf	balanced	1	3
Model 86	22	rbf	balanced	1	3
Model 87	2	rbf	None	1	3
Model 88	19	rbf	None	1	3
Model 89	13	rbf	None	1	3
Model 90	53	rbf	balanced	1	3
Model 91	99	rbf	balanced	1	3
Model 92	73	rbf	None	1	3
Model 93	74	rbf	None	1	3
Model 94	74	rbf	None	1	3
Model 95	83	rbf	None	2	3
Model 96	52	rbf	None	8	3
Model 97	94	rbf	None	7	3
Model 98	26	rbf	balanced	9	3
Model 99	11	rbf	None	13	3
Model 100	32	rbf	None	32	3
Model 101	100	rbf	None	9	3
Model 102	100	rbf	None	69	3

Table A.4.13. The selected parameters of NB for mRMR-Based models (Part 1)

Model	Parameters
	var_smoothing
Model 52	182.32071
Model 53	178.51892
Model 54	166.24599
Model 55	150.54308
Model 56	137.69940
Model 57	128.61934
Model 58	125.20570
Model 59	112.05137
Model 60	98.98805
Model 61	84.23417
Model 62	70.52404

Table A.4.14. The selected parameters of NB for mRMR-Based models (Part 2)

Model	Parameters
	var_smoothing
Model 63	63.21477
Model 64	50.55721
Model 65	52.68895
Model 66	65.04750
Model 67	53.48176
Model 68	66.40741
Model 69	54.02647
Model 70	68.73044
Model 71	54.95739
Model 72	41.03409
Model 73	55.85010
Model 74	68.29670
Model 75	54.14046
Model 76	67.15897
Model 77	54.94759
Model 78	46.41106
Model 79	54.73437
Model 80	42.24459
Model 81	31.07999
Model 82	29.02248
Model 83	17.84320
Model 84	15.67409
Model 85	6.40995
Model 86	1.60188
Model 87	0.57163
Model 88	0.35074
Model 89	0.03141
Model 90	0.34537
Model 91	0.65193
Model 92	0.15537
Model 93	0.00554
Model 94	0.00140
Model 95	0.02028
Model 96	0.00002
Model 97	0.00046
Model 98	0.00575
Model 99	0.00027
Model 100	0.01282
Model 101	0.00319
Model 102	0.01587