FAULT DIAGNOSIS IN WHITE GOODS USING NATURAL LANGUAGE PROCESSING TECHNIQUES ON CUSTOMER COMPLAINTS

HAKAN GÜVEN ŞENZEYBEK

ÖZYEĞİN UNIVERSITY

SEPTEMBER, 2025

FAULT DIAGNOSIS IN WHITE GOODS USING NATURAL LANGUAGE PROCESSING TECHNIQUES ON CUSTOMER COMPLAINTS

by
Hakan Güven Şenzeybek

Thesis
Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

in
Data Science

Advisor: Prof. Olcay Taner Yıldız

Graduate School of Science and Engineering
Özyeğin University
İstanbul

September, 2025

FAULT DIAGNOSIS IN WHITE GOODS USING NATURAL LANGUAGE PROCESSING TECHNIQUES ON CUSTOMER COMPLAINTS

Approved by:

Prof. Olcay Taner Yıldız, Advisor
Department of Artificial Intelligence and Data Engineering
Özyeğin University

Asst. Prof. Mehmet Önal
Department of Industrial Engineering
Özyeğin University

Prof. Sadık Fikret Gürken
Department of Computer Engineering
Boğaziçi University

Approval Date: 06.08.2025

To my brother Murat Şenzeybek and my family, whose support made this journey possible.

DECLARATION OF ORIGINALITY

I hereby declare that I am the sole author of this thesis and that this is the true copy of my thesis, including the final revisions, approved by my thesis committee. All data and information have been obtained, produced, and presented in accordance with the rules of research ethics and principles of academic honesty. As required by these rules, to the best of my knowledge I have acknowledged ideas, thoughts, and any copyrighted material in accordance with the standard referencing rules. I certify that any part of this thesis has not been submitted for a degree or diploma in another educational institution.

Hakan Güven Şenzeybek

ABSTRACT

The thesis investigates the application of Natural Language Processing (NLP) for fault classification in the white goods sector, using real-world service data. two-stage hybrid pipeline is developed: first, unstructured customer complaint texts are transformed into structured representations through both rule-based preprocessing and Large Language Model (LLM)-based enrichment. Then, multiple machine learning classifiers are trained to predict repair codes, which include the Random Forest, Naive Bayes, LightGBM, BiLSTM CNN-BiLSTM and DistilBERT.

The system is evaluated on a large-scale dataset of 103,000 technical service records. Experimental results reveal that semantically enriched representations generated by GPT-4o-mini consistently outperform both unprocessed and rule-based inputs. Across all classifiers, the LLM-Rich preprocessing strategy yields the highest performance, achieving complexity-weighted average F1 scores of up to 63,4%, compared to 54,1% for the best unprocessed input and 52,2% for the best rule-based alternative.

Furthermore, the study introduces a complexity score for each repair code, allowing analysis of model performance relative to task difficulty. Results indicate that LLM-enhanced inputs not only improve accuracy but also increase robustness across high-complexity classes.

Overall, the results indicate that semantically enriched NLP pipelines offer a scalable and powerful basis for semi-automated fault diagnosis, particularly in low-resource languages like Turkish, and greatly strengthen the robustness of the corresponding classification models.

Keywords: Natural Language Processing (NLP), Fault Diagnosis, White Goods, Large Language Models (LLM), Text Classification

ÖZET

Bu tez, beyaz eşya sektöründe arıza sınıflandırması amacıyla Doğal Dil İşleme (NLP) tekniklerinin uygulanmasını, gerçek teknik servis verileri üzerinden incelemektedir. İki aşamalı hibrit bir işlem hattı geliştirilmiştir: İlk aşamada, yapılandırılmamış müşteri şikayet metinleri hem kural tabanlı ön işleme hem de Büyük Dil Modeli (LLM) temelli zenginleştirme yöntemleriyle yapılandırılmış temsillere dönüştürülmektedir. İkinci aşamada ise, Random Forest, Naive Bayes, LightGBM, BiLSTM, CNN-BiLSTM ve DistilBERT gibi çeşitli makine öğrenimi sınıflandırıcıları kullanılarak onarım kodları tahmin edilmektedir.

Sistem, 103.000 teknik servis kaydından oluşan büyük ölçekli bir veri kümesi üzerinde değerlendirilmiştir. Deneysel sonuçlar, GPT-4o-mini tarafından üretilen anlamsal olarak zenginleştirilmiş temsillerin, hem işlenmemiş hem de kural tabanlı girdilere kıyasla tutarlı şekilde üstün performans gösterdiğini ortaya koymuştur. Tüm sınıflandırıcılarda, LLM-Rich (LLM-Zengin) ön işleme stratejisi en yüksek performansı sağlamış ve karmaşıklık ağırlıklı ortalama F1 skorlarında %63,4'e kadar başarı elde etmiştir; bu oran, en iyi işlenmemiş giriş için %54,1 ve en iyi kural tabanlı alternatif için %52,2'dir.

Ayrıca çalışma, her bir onarım kodu için bir karmaşıklık skoru tanıtarak model performansının görev zorluğuna göre analiz edilmesine olanak tanımaktadır. Sonuçlar, LLM ile zenginleştirilmiş girdilerin yalnızca doğruluğu artırmakla kalmayıp, aynı zamanda yüksek karmaşıklıktaki sınıflarda da modelin dayanıklılığını güçlendirdiğini göstermektedir.

Genel olarak, elde edilen bulgular, anlamsal olarak zenginleştirilmiş NLP işlem hatlarının, özellikle Türkçe gibi düşük kaynaklı dillerde, yarı otomatik arıza teşhisi için ölçeklenebilir ve güçlü bir temel sunduğunu ve ilgili sınıflandırma modellerinin dayanıklılığını önemli ölçüde artırdığını ortaya koymaktadır.

Anahtar Kelimeler: Doğal Dil İşleme (NLP), Arıza Teşhisi, Beyaz Eşya, Büyük Dil Modelleri (LLM), Metin Sınıflandırma

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

API Application Programming Interface

ASCII American Standard Code for Information Interchange

BERT Bidirectional Encoder Representations from Transformers

BiLSTM Bidirectional Long Short-Term Memory

CLS `[CLS]` Classification Token

CNN Convolutional Neural Network

FSM Finite State Machine

GPT Generative Pre-trained Transformer

IDF Inverse Document Frequency

LGBM Light Gradient Boosting Machine

LLM Large Language Model

LRU Least Recently Used

LSTM Long Short-Term Memory

NB Naive Bayes

NLP Natural Language Processing

NLTK Natural Language Toolkit

PC1 First Principal Component

PC2 Second Principal Component

PCA Principal Component Analysis

RBP Rule-Based Preprocessing

RF Random Forest

TF Term Frequency

XLM Cross-lingual Language Model

1. INTRODUCTION

The growing utilization of after-sales technology service systems in consumer electronics, especially white goods, has raised the demand for scalable and effective fault diagnosis solutions. On the other hand, customers usually report the issues in free-text, generating a big amount of unstructured and noisy text data. These service records are generally short, cryptic, and incomplete, which makes them challenging for the salvation processing and analysis pipeline.

It is however difficult to manually categorize faults from such complaints, it is very time-consuming and error prone, and different service centers will have different interpretations. These problems are compounded by variations in technician scheduling and diagnosis accuracy that result inefficient operations and a drop in customer satisfaction. This setting emphasizes the increasing need for diagnostic systems which are capable of interpreting complaints and over the phone symptoms reported by customers and generating appropriate (most probably) repair directives to service division.

Natural Language Processing (NLP) presents a potential solution to these challenges by providing the tool for automation of the extraction of meaning from unstructured aspect-based complaint texts. While NLP has been a common way for diagnosing cars, IT support and other fields, it has not been widely used in the field of white goods: especially on Turkish data. The current methods often require rule-based systems which can to be hard to maintain and evolve over time or data-intensive annotated corpora, which can be difficult to obtain in industrial settings, preventing these approaches from scaling.

This work proposes a new hybrid pipeline that combines rule-based Turkish text preprocessing with semantic transformation using LLMs to leverage the complaint texts for structured information. This enriched representations are further employed for training multiple machine learning models to predict the right repair code. The study also compares the performance of alternative preprocessing and classification model while incorporating

complexity score to understand what is the best strategy in dealing with noisy and complex service records for repair codes.

We build upon an exhaustive and large-scale empirical study across 103.000 Turkish technical support notes. It proposes a two stage preprocessing architecture where linguistic rule-based processing is compared with LLM-based semantic enhancement, and further compares five classification algorithms on two representations of inputs – basic versus semantic-enriched. The investigation also involves interpretability evaluations and error analysis that illustrates the relationship between model behaviours, linguistic variations and task escalations.

By proposing a systematic and scalable approach for fault classification in the white goods industry, this thesis aims at reducing technician's workload, providing high-accuracy diagnostics, and contributing to the body of industrial NLP, especially for low-resource language like Turkish.

2. STATE OF THE ART

One of the most closely related studies is the recent work by Pavlopoulos et al.(2024), which examines the application of natural language processing and deep learning techniques for automated fault diagnosis in the automotive domain[1]. In this study, the authors utilize large-scale, multilingual datasets comprising free-text fault descriptions and employ state-of-the-art Transformer-based models, including Cross-lingual Language Model (XLM)-R and DistilBERT, to classify these descriptions into specific fault categories. Their findings demonstrate that such models can effectively identify the root causes of vehicle malfunctions from unstructured textual data, even in the presence of significant class imbalance and a large number of categories. These results highlight the potential of deploying text-based diagnostic systems in industrial settings.

Like Pavlopoulos et al., we process unstructured, free-form customer complaint and solution texts to predict the appropriate repair code. Both studies face the common challenges of extracting structured insights from noisy, technical language, handling class imbalance, and automating the mapping of real-world problem descriptions to standardized maintenance categories. This parallel underscores the growing impact and transferability of NLP-based approaches for automating diagnostics and decision support in complex, real-world service environments.

One of the most relevant contributions to rule-based Turkish text preprocessing is the open-source morphological analyzer introduced by Yıldız et al. [2]. Their system presents a modular and extensible architecture built upon finite-state transducers, rule-based suffixation, a rich lexicon of over 50,000 lemmas, and high-speed components such as a trie data structure and Least Recently Used (LRU) cache. Capable of analyzing up to 100,000 tokens per second, the analyzer sets a high performance standard for large-scale text normalization in Turkish. Importantly, it is designed to be adaptable and transparent, making it well-suited for controlled domains such as technical documentation or structured

service records. In this thesis, we adopt similar design principles, leveraging finite-state morphological analysis as a core component of our rule-based preprocessing pipeline. By aligning with the methodological foundations of Yıldız et al., our system benefits from both linguistic rigor and operational efficiency, particularly in handling morphologically complex inputs in Turkish fault description datasets.

There have been exciting recent developments in LLM that permit substantial advances in extraction and interpretation of high-dimensional and unstructured textual data in industry and in clinical applications. Vidyaratne et al. [3] introduced an LLM-based method that automatically generates troubleshooting trees for industrial equipment, finding that transformer-based LLM modelled macroscopic performance advantage in matching failure description to diagnostic path against traditional rule system. Their research demonstrates the power of LLMs to organize noisy service records and facilitate the identification of real-time failures, an ability closely related to the task of the white goods domain in this thesis. Similarly, Munzir et al. [4] demonstrated that general-purpose LLM outperformed other computational methods on high-throughput physician note phenotyping. Their domain notwithstanding, their work underscores the model's capacity to pick up fine-grained, clinically significant patterns from freeform text—an analog shared with technical service logs. All these results support the conclusion that LLMs are well suited in tasks where the textual data is context-rich but sparse in structure.

Another closely related study is by Khodadadi et al. (2021), which focuses on automated routing of vehicle malfunction reports to appropriate technical departments based on free-form customer complaint descriptions [5]. Their model employs BiLSTM and CNN architectures to classify complaints with high accuracy, and also includes a demand verification layer to assess the relevance and validity of each textual input. Similar to our work, their system addresses the linguistic ambiguity of customer-written complaints, class imbalance across classes, and the challenges of interpreting technical terminology in

noisy input. However, unlike their approach, our thesis introduces a large language model (LLM)-based preprocessing pipeline to overcome the higher degree of uncertainty and missing information in white goods service records. This addition reflects the increasing viability of LLMs in industrial NLP settings with incomplete or inconsistent inputs. This thesis presents a novel hybrid pipeline that first utilizes LLM's to clean and structure unstructured textual data—specifically, "problem descriptions" and "solution descriptions" from technical service logs—and then applies traditional ML methods to develop predictive models for identifying repair codes.

One of the most closely related studies is a recent paper that applies NLP and deep learning for automated vehicle diagnostics [6]. The study uses BiLSTM and CNN architectures to classify vehicle service issues based on free-form text descriptions, showing the feasibility of using text as a diagnostic input in technical maintenance. Our work parallels this approach in the white goods domain, where problem/solution text is used to predict technical intervention areas.

At the core of the data preprocessing step, LLMs are employed to interpret noisy and unstructured text and convert it into a structured format by extracting fields such as complaint type, symptom, detected issue, replaced parts, and repair outcome. This step draws directly from the philosophy of RetClean [7], which showcases how LLMs can be harnessed for data cleaning through contextual understanding and suggestion augmentation via retrieval-augmented generation. Just like RetClean, our system also considers practical constraints such as data privacy and local model applicability—making it suitable for sensitive industrial use cases.

Following the LLM-based transformation, clustering and classification methods are used to develop predictive models. These steps are grounded in foundational clustering methodologies that aim to group similar data points while simplifying underlying complexity, as elaborated by Berkhin [8]. Furthermore, the multi-label and multi-class nature

of the target variable echoes the structure of the corpus described in Yu et al. [9], where NLP and ML methods are used for automatic classification expansion over corpora.

The textual modeling for classification in this thesis also aligns with visualization-driven diagnostics introduced in DeepNLPVis [10], which explores mutual information and layered interpretability for text classification. Although visualization is not the focal point in this work, the explainability and representation concerns addressed in that study are conceptually relevant to our use of TF-IDF and feature importance metrics. Similarly, *SemLa* [11] has shown how visual analytics can help in evaluating model performance and guiding model adjustments in complex NLP pipelines.

Moreover, the process of converting noisy complaint and solution text into clean, structured inputs relates to the sentiment extraction process in education feedback systems, as described in [12]. While the domain differs, both studies rely on text normalization, classification, and pattern recognition to extract actionable insights from subjective human feedback.

In terms of system architecture, the similarity-based matching and classification components of our workflow bear strong resemblance to the decision support system proposed in [13], which combines NLP and Naive Bayes classifiers to detect project proposal revisions with high accuracy. Like that study, our thesis shows that even simple machine learning models—when fed clean and semantically rich input—can yield high performance and practical utility in real settings.

Finally, the broader context of text mining in consumer platforms, as explored in [14], supports the relevance of mining large volumes of real-world unstructured text to inform business strategies. Our work similarly processes large volumes of service descriptions to extract usage patterns, failure types, and part replacement trends, aiming to inform not only model development but also organizational learning and quality control mechanisms.

In summary, this thesis builds upon and contributes to a growing body of work where LLMs are increasingly utilized not only for generative tasks but also as powerful preprocessing engines that enhance the performance and interpretability of traditional machine learning models. The integration of these two paradigms—semantically rich preprocessing via LLMs and robust modeling via classical ML—demonstrates a scalable approach for analyzing customer-facing service data, with potential applications in predictive maintenance, customer satisfaction, and operational efficiency.

3. DATASET

The data in this table summarize technical service records from 2023. This data set has 103,001 records in total and is a service transaction with structured and un-structured field. The most important columns are the *problem description* written by the customer and the *solution description* written by the technician. Other than these text fields, the data has got service center name, customer location(city), product group and product informations such as model codes.

The data is categorized by product groups (such as washing machines, dishwashers, refrigerators etc.), each with its own set of common fault types and terminology. Certain component names, for example “motor” or “main board”, may appear in different product groups but can have different technical functions and meanings. For example, the “main board” in an air conditioner and in a washing machine both refer to electronic control units, but their structure and malfunction patterns may differ. This makes product group segmentation an important factor in both the preprocessing and modeling phases.

A notable feature of the dataset is that about 17% of the records have missing values in the main textual columns (*problem description* and/or *solution description*). Sometimes, the customer does not provide a complaint, but the underlying problem can be inferred from the technician’s detailed solution notes. During preprocessing, incomplete, extremely short, or uninformative records are carefully filtered to ensure data quality.

Table 3.3 shows three sample records from this dataset, each one representing a different situation: a normal customer complaint, one from another product group, and one where there is no customer problem description but the problem is clear in the technician’s notes. In each instance, identifying details have been anonymized.

By studying these data, our goal is to clean and structure the textual content using LLMs and exploit it in order to train machine learning models that predict for the repair our relevant component codes. This process facilitates operational intelligence and allows

automated decision support for the technical service.

All personal identifiable information in the dataset is anonymized based on data protection and privacy compliance.

3.1 Field Text Analysis

The dataset contains a total of 103,001 technical service records, in which there are two main free text fields: *problem description* (customer complaint) and *solution explanation* (technician's notes). This brief examination of these fields suggests some issues and interesting tendencies.

The *problem description* field is not empty in 73,056 (70.9%) while it is completely empty in the other 29,945 (29.1%) records. Second, over half of problem descriptions (51,882 records, 50.4%) are too short and express three or less words, or 10 or less characters, which shows problem text descriptions are often absent or extremely short and slim[5, 6]. The mean number of words per problem description is 15.9 (s.d 34.5), a median value of 3, and range from 0 to 506 words. The first and third quartiles are 1 and 12 words, respectively. By Character, the mean is 111.7 and (median: 25, st. dev. : 301) the one having the highest number is 3,758. (However, cases of entirely too many words such as hundreds, or thousands of characters are otherwise unusual, and imply copy-paste or accidental data.)

The *solution explanation* field is more complete: 84,796 records (82.3%) contain text, while 18,205 (17.7%) are empty. Here, 17.9% of explanations are also "too short." Solution explanations are on average 17.8 words long (median: 18; st. dev.: 21.2), with a range of 0 to 315 words. In terms of characters, the mean is 139.9 (median: 141, st. dev.: 185), with a maximum of 2,373. Technician notes are thus more detailed and informative, but there is still a significant number of either brief or non-recorded notes.

Short: $\leq$3 words or $\leq$10 characters. The most frequent tokens in problem descriptions

Table 3.1: *Summary statistics for the main text fields.*

Field	Avg. Wds	Med. Wds	Max Wds	Empty (%)	Short (%)
Problem Desc.	15.9	3	506	29.1	50.4
Solution Expl.	17.8	18	315	17.7	17.9

include: gt,gt,, arıza, –, ve, bir, tespit, -, ., tl, da.

A simple text analysis for the problem descriptions provides a collection of 29,152 unique words and solution explanations describe 36,081. But the large ratio of rare or mis-typed tokens, in addition to frequent boilerplate phrases, makes for a lot of noise. Analysis of outliers and anomalies reveals problem descriptions that are built up entirely of numbers or special characters represent 0.19% of the observations, while solution explanations in the same category account for 0.08% of the total amount. About 4.5% of problem descriptions are recognized as being a repeated or a templated occurrence.

It should be noted, however, that in technical service datasets like this, even brief or single-word problem descriptions may carry significant diagnostic information. For instance, the words "çalışmıyor" ("not working"), "açılmıyor" ("does not open") or "ısınmıyor" ("not heating") -although they are short- they can imply enough evidence to find the main issue. This is in accordance with previous observations in the literature that user-reported service records tends to use brief domain-specific vocabulary that may be short but provides rich and informative features for subsequent tasks, e.g., fault classifying or symptom extracting [15]. As a consequence, it cannot be inferred that the dataset has a high concentration of short entries with little content that can be of practical use; what it indicates is that we need to enhance the performance of NLP models when applied in technical settings by modeling the behaviour of end-users.

The table below presents the 10 most common meaningful unigrams and the 10 most common bigrams in problem descriptions. From these results, it appears that financial and procedural terms such as *diagnostic fee* (*tespit ücreti*), *fault detection* (*arıza tespit*), and

talk of external matters or a lack of wanting repair are dominant. This reflects the strong presence of boilerplate or templated language, due to operational or regulatory procedures, in the corpus. Such language patterns need to be taken into account in preprocessing in order to prevent the model from developing a similar bias.

Table 3.2: *Most frequent unigrams and bigrams in the problem description field.*

Top 10 Unigrams		**Top 10 Bigrams**	
Unigram	**Frequency**	**Bigram**	**Frequency**
arıza	36,679	tespit ücreti	12,704
tespit	25,247	üründe arıza	12,691
grubu	18,020	tl tespit	12,635
tl	15,917	dış etken	12,486
ürün	14,396	etken kaynaklı	12,351
onarım	13,908	arıza görülmezse	12,303
hizmet	13,569	istemezseniz üründe	12,210
üründe	13,493	onarım istemezseniz	12,130
ücreti	13,310	kaynaklı arıza	12,043
bilgisi	13,266	arıza tespit	11,984

Note: All n-grams are extracted from the customer problem descriptions. The prevalence of financial and procedural terms highlights the presence of standardized or templated language in the dataset.

However, as one can see in both n-gram and bi-gram distribution that dominant phrases used in problem descriptions are not related to root cause of the fault; nor do they describe the symptoms which can be used for medical diagnosis. Such observation is consistent with other related studies, where user-generated text in-service roles mostly contains templated or administrative messages rather than informative fault or symptom descriptions [16]. Therefore, we need to filter more, and only keep the significant phrases or known entities occurred in the text, which is directly related to the text diagnosis. Concentration on the informative pieces of the messages, such as information content and models rather than boilerplate or procedural components, is essential to enhance model interpretability and predictive performance.

As nearly a third of the problem descriptions is empty, and more than half of them are very short or uninformative, this has an impact on the performance and robustness of classification models. One possible use case is to collect technician solution explanations to reconstruct missing problems or to predict directly the repair component codes. However, solution explanations are provided in very diverse ways, with explanation texts that are quite free and noisy, and that do not conform to a template. It proved very difficult to automatically extract the context of the problem from the technician notes in a consistent and high-quality manner without using large language models from unstructured text sources [3, 4].

3.2 Data Structure and Attributes

Taking a closer look at the dataset, each providing important details for a given service transaction. The *Transaction Number (işlem numarası)* uniquely identifies each record. *Installation Date (Montaj Tarihi)* and *Repair Date (Onarım Tarihi)* indicate the respective installation and service dates. *Repair Type (Onarım Tipi)* specifies the nature of the repair and whether the customer was informed, while *Quantitative Repair (Adetsel Onarım)* marks if parts replacement was performed. The *Product ID (Ürün ID)* uniquely identifies each product. Manufacturing-related informations include *Manufacturing Year (Üretim Yılı)*, *Manufacturing Month/Year (Üretim Ay/Yıl)*, and *Manufacturing Date (Üretim tarihi)*.

Furthermore, their unstructured, free text format brings more difficulties in data processing but also allows advanced NLP methods that can be adapted for multi-classification tasks (fault categorization, key phrase extraction, and root cause analysis).

Representative examples and excerpts from the data set are shown in Table 3.3, to give an overview on the (rough) structure and content of the key fields, as well as the diversity in the level of detail and informativeness of the text fields.

Table 3.3: *Sample service records from the dataset (Turkish).*

İşlem numarası	**Onarım Bölüm Kodları**	**problem açıklaması**	**çözüm açıklaması**	**Ürün Grubu**
8091176225	Ana Kart	Müşteri F2 hatası bildiriyor, cihaz çalışmıyor.	Anakart değişimi yapıldı, ürün çalışır durumda.	KLİMA
8091177030	Rezistans	Ürün ısıtmıyor, su soğuk.	Yapılan kontrolde cihazın anakarta komut hatası gönderdiği, ısıtıcı değişimi gerektiği tespit edildi.	ÇAMAŞIR MAKİNASI
8091177051	Motor	—	Yıkama motorunun sıkıştığı tespit edildi, motor değişimi yapıldı.	BULAŞIK MAKİNASI

- *Row 1: The customer reports an "F2 error" and that the device is not working. The technician replaced the main board, and the product is now functional. (**Air conditioner**)*
- *Row 2: The customer reports that the product does not heat and the water remains cold. The technician found a command error in the main board and determined the heater needed replacement. (**Washing machine**)*
- *Row 3: The problem description field is empty, but the problem is conveyed in the solution explanation. The technician found the washing motor was jammed and replaced it. (**Dishwasher**)*

4. METHOD

The methodology in this study is developed based on a full-stack system architecture that seeks to automatically predict *repair department codes* from raw free-text complaint data. As shown in Figure 4.1, the proposed architecture is a multi-branch framework which blends traditional rule-based preprocessing elements with more recent large language model (LLM) semantic enrichment, providing a fair benchmark over the four text structuring and abstraction levels. We use this architecture to conduct an analysis on the effects of different text preprocessing techniques, with the general classification task being on technical service management domain.

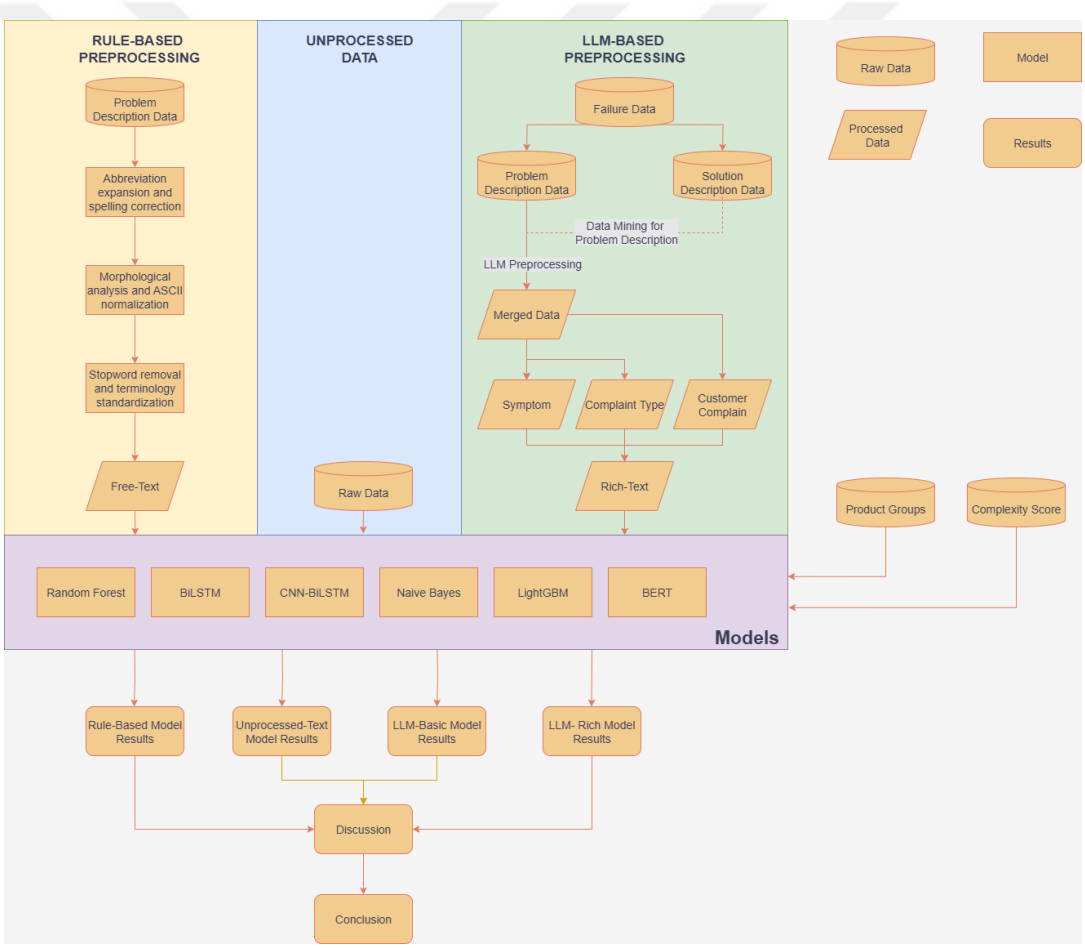
The entire pipeline consists of three main preprocessing modalities: Rule-Based Preprocessing, Raw Text Usage, and LLM-Based Preprocessing. In the rule-based approach, linguistically motivated tools tailored for Turkish—a morphologically complex and agglutinative language—are applied to normalize customer complaint descriptions. This pipeline includes modules for abbreviation expansion, spelling correction, morphological parsing, American Standard Code for Information Interchange (ASCII)-to-Turkish character restoration, and stopword removal. The goal of this step is to transform noisy service records into syntactically valid and semantically coherent free-text representations that are well-suited for explainable modeling.

The second modality is composed of *unprocessed, raw text data*, which represents the raw format of technician entered complaint text without any preprocessing. This branch is used as a baseline for assessing the effect of the structured preprocessing on the model output. Although overly simplistic, it offers powerful visual insights into the ability of models to generalize without normalized, real world input.

Third (also the most semantically-enriched) mode is LLM-based preprocessing where, instead of using the extracted input-only signals, we use GPT-4o-mini for complaints text feature extraction to store the structured attributes [17]. With the design of

targeted prompts, we prompt the model to predict important fields like Customer Complaint, Complaint Type, and Symptom. Additionally, as an different approach, another model makes use of the solution description data as contextual information during the preprocessing phase, while creating credible outputs in the absence of clear or complete problem descriptions. Such rich-text format allows models to learn from higher-level semantic structures over surface-level text patterns, hence greatly improving the quality of predictions and the robustness of models.

Figure 4.1: *End-to-end methodological workflow: from preprocessing to prediction and evaluation.*



In the preprocessing phase, the three data streams; free text, raw text and rich text generated by LLM are fed into a shared modeling layer, which was implemented with

6 different type of classifiers: *Random Forest*, *BiLSTM*, *Convolutional Neural Network (CNN)-BiLSTM*, *Naive Bayes*, *LightGBM* and *DistilBERT*. The results have been analyzed independently for each preprocessing, to allow a due comparison of their effectiveness. Besides, *structured metadata* like *product group labels* and *component-level complexity scores* are combined into the models to investigate their influences in prediction results; these meta features indicate the degree of classification difficulty and semantic vagueness.

4.1 Data Preprocessing

Preprocessing of unstructured text data is mainly well-known essential steps in many applications such as classification, semantic analysis and information retrieval. In the current study, two different preprocessing methods were implemented and compared, which are the rule-based linguistics approach that is designed only for Turkish and a Transformer-Based Large Language Model by leveraging Generative Pre-trained Transformer (GPT)-40-mini. For each method there is also the associated set of advantages and limitations depending on the data, the required degree of human interpretability and complexity of the linguistic structures.

4.1.1 Rule-Based Preprocessing

The rule-based approach developed for this study leverages domain expertise and language-specific knowledge to clean and normalize text data in Turkish—a morphologically rich and agglutinative language. The preprocessing architecture consists of a modular pipeline that includes abbreviation expansion, spelling correction, morphological analysis, ASCII-to-Turkish character restoration (deasciification), and stopword removal. These components were implemented using linguistically grounded tools and custom-built resources that enable high-precision normalization, especially in low-resource or noisy domains.

The system has a core *FsmMorphologicalAnalyzer* which is a finite state morphological analyzer that does fine grained syntactic and morphological parsing on Turkish words by decomposing them into their root and affix parts. For example, a word like "çamaşırlarını " is analyzed as consisting of the stem "çamaşır" and the plural (*-lar*), possessive (- *ı*) and accusative case (*-nı*) inflectional morphemes. This allows the system to lemmatize, validate, and disambiguate tokens in a linguistically plausible manner, which supports downstream applications such as code repair prediction or named entity recognition. The analyzer integrates high-quality, easily extendible Turkish processing frameworks like the one by Yıldız et al. [2], where they use Finite State Machine (FSM), rule engine and a rich lexicon for scalable and accurate morphological analysis.

Words that fail morphological parsing are passed through a n-gram-based deasciification module, *NGramDeasciifier*, which uses bigram/trigram probabilities to infer the correct Turkish character mappings from ASCII inputs (e.g., *"camasir"* → *"çamaşır"*). The deasciifier is trained using a Turkish corpus and relies on statistical character sequences combined with linguistic rules. For further error correction, a morphology-aware spell checker, *SimpleSpellChecker*, suggests the most probable corrections by evaluating edit distances in conjunction with morphological plausibility.

To address domain-specific noise, such as technician shorthand and common misspellings, the system incorporates a curated replacement dictionary. This hand-built resource includes more than 100 mappings of slang, shorthand or phonetically modified expressions to their proper forms. Examples include *"çm"* → *"çamaşır makinesi"*, *"bzd"* → *"buzdolabı"*, and *"e.kart"* → *"elektronik kart"*. These fixes are insensitive the case, and aware of the frequency of fixes to ensure that high-impact patterns are covered.

Additionally, the pipeline removes Turkish stopwords sourced from Natural Language Toolkit (NLTK) and domain-specific stop tokens (e.g., *"gt"*), and employs normalization layers such as *SimpleAsciifier* when character uniformity is required for matching

or indexing.

Transparency and controllability are the two most important merits of this rule based system. The process that leads to each transformation step is transparent and traceable, which makes it particularly well-suited for scenarios where *explainability* may be relevant, for example, in diagnostic support systems or customer service analytics. Furthermore, as pointed out in [18, 19], rule-based analyzers are also able to provide outputs that are syntactically sound and linguistically motivated, which outperforms statistical models in structural quality.

However, rule-based analysis also comes with significant downsides. Scale up is also limited due to the need of manual rule crafting, maintenance, and susceptibility to non-standard language, such as slang, misspellings, or ungrammatical formulations. It does well in settings where the user can control the input distribution, but fails to generalize when exposed to open or user-generated noise. So, whereas that architecture is a very strong basis for preprocessing in specific domains, given that all this work is rule based, we could in future work also look at strategies that are hybrids of rule-based precision and the adaptabilities of neural models.

4.1.2 LLM-Based Preprocessing

To solve the problems in the rule-based approaches, we also considered the use of an LLM-based preprocessing technique with GPT-4o-mini [17]. This model, a lightweight version of OpenAI's GPT-4 family, can understand and generate human-like text in many languages, including Turkish. LLMs are pre-trained on massive multilingual corpora and can predict underlying latent space and meaning of a given text via the context-aware generation and in-context learning [20, 21].

In the current work, GPT-4o-mini was implemented through the OpenAI Application Programming Interface (API) platform and used to generalize free-text customer

complaints while converting them to a structured representation. The method proposed to formulate queries which encourage the model to extract informative semantic aspects (complaint type, symptoms, resolution steps) allowed the system to perform information retrieval without depending on strict rules or external dictionaries. The model were especially successful at processing noisy inputs such as typographical errors, slang and partially formulated sentences because of its strong contextual understanding and transfer learning mechanisms [22].

One of the great advantages of using an LLM is its scale-ability. Unlike the rule-based methods, LLMs do not need any manual modifications or additions for new abbreviations that have emerged or trends of new language. They generalize in a dynamic way to various language inputs and can be easily fine-tuned or programmed for new tasks with limited data. What's more, LLMs are capable of capturing semantic relationships that go beyond mere surface-level patterns of text, leading to a more sophisticated understanding and more useful downstream predictions—a crucial factor when predicting something as abstract as repair codes.

Importantly, this study not only utilized LLMs for preprocessing but also compared their output against human-annotated solution descriptions. This comparison was made possible because LLMs allow explicit prompting for extracting task-specific fields such as solution descriptions—something not feasible with rule-based methods. While traditional approaches operate based on predefined patterns and surface-level rules, LLMs can be directed to generate structured outputs tailored to specific objectives, thus enabling evaluation of semantic alignment between model-generated content and manually labeled solution fields.

However, LLMs are not perfect. This is because they are very expensive to compute, both at training and deployment, and because the internal decision process is often non transparent. This lack of interpretability can be problematic in cases where

model interpretability is important. Moreover, even though LLMs are strong generals, they are not without blame—mistakes in prompt formulation or unexpected inputs can cause hallucinations or incorrect structuring.

In conclusion, both rule-based and LLM-based preprocessing methods offer valuable tools for structuring unstructured customer complaint data in Turkish. The rule-based pipeline excels in precision and linguistic fidelity but is limited by its rigidity and need for maintenance. In contrast, LLM-based preprocessing demonstrates superior adaptability and semantic depth, making it well-suited for noisy, real-world applications. A hybrid approach, leveraging the strengths of both paradigms, may provide the most effective solution for future Natural Language Processing (Natural Language Processing (NLP)) systems in low-resource and technical domains.

4.2 Transformation Pipeline and Feature Extraction

The transformation process involves constructing dynamic prompts that instruct the model to identify specific attributes such as Complaint Type, Inferred Symptom, and Fault Diagnosis. In practice, the LLM was provided not only with customer complaint fields but also with the solution description data, not as a predictive input, but as an auxiliary source to help infer meaningful structure from vague or incomplete problem descriptions. This design choice improved the quality of extracted features, particularly when original customer inputs lacked clarity or detail. Importantly, the solution description was used solely during the preprocessing stage and was strictly excluded from model training and prediction phases.

In the experiment of predicting repair code, as input features, Symptom, Complaint Type and Customer Complaint are three favorite components which can be extracted from the text data. These features are only available through the LLM based preprocessing pipeline that allows for semantic understanding and condensation of free text to a structured

fields. The model is encouraged to learn these factors despite broken, incorrect, or noisy language - skills that are especially useful for real-world industrial data.

Rule-based preprocessing methods lack the flexibility and contextual reasoning required to extract such semantically rich features. These methods depend heavily on keyword matching, pattern rules, or domain-specific lexicons, making them brittle and ineffective when dealing with typographical errors, slang, or indirect expressions. As a result, fields like Symptom and Complaint Type—especially when not explicitly stated—cannot be reliably derived using rule-based approaches.

These three fields represent the customer's initial description of the issue and observable symptoms before any technical intervention. Meanwhile, other fields—such as diagnosis, replaced components, and resolution status—are post-repair outcomes generated by technicians and are excluded from the input feature set to preserve the integrity of the predictive task.

While this approach prioritizes features representing the problem context, an important consideration is that approximately 20%–25% of the original dataset includes problem-related information solely within the solution description data. However, to maintain consistency and avoid introducing post-hoc information, this field was excluded from the feature extraction process. Consequently, records lacking sufficient information in the problem-related fields were either skipped or labeled as containing insufficient data.

This strict separation ensures the model operates under realistic conditions, using only inputs that would have been available prior to repair, and prevents potential data leakage from outcome-dependent sources.

Table 4.1 also further illustrates this essential ability of the LLM to convert unstructured or non-standard but otherwise valid service records into structured, machine-interpretable information, that would be impossible using rule-based approaches. In the first row where the source description of the problem is completely missing, the model

Table 4.1: *Example comparison of free-text inputs and the corresponding LLM-generated outputs.*

Free Text - Problem	**Free Text - Solution**	**LLM Output - Customer Complaint**	**LLM Output - Complaint Type**	**LLM Output - Symptom**
.	Yapılan kontrol sonucu ürünün yika motorunun sıkıştığı tespit edildi, motor değişimi yapıldı.	(Tahmini) "Ürünüm çalışmıyor, yıkama sırasında durdu."	Mekanik Arıza	Yıkama motorunun sıkışması, anormal sesler
Müşteri Notları: F2 hatası veriyor.	Yapılan kontrollerde üründe pompa motoru arızalı olduğu görüldü. Parça değiştirildi.	Müşteri F2 hatası veriyor.	F2 Hatası	Pompa motoru arızası
V22067924 KART V20846098 ISITICI Müşteri cihaz çalışmıyor, su soğuk dedi.	Yapılan kontrolde cihazın anakarta komut hatası gönderdiği, ısıtıcı değişimi gerektiği tespit edildi.	Çamaşır makinesi fişi prize takınca şalter atıyor.	Elektriksel Arıza	Şalter atma durumu

produces a very plausible customer complaint based on the response explanation only which is indicated with “(Tahmini)” (Estimated). Such kind of inference depends on the contextual knowledge in LLM, and cannot be performed by rule-based inference.

In the examples to follow the model is successful in recognizing and extracting semantically meaningful information such as complaint type and technical symptoms from the loosely structured text inputs. These examples also demonstrate the semantic generalization capacity of LLMs, that applies beyond surface level patterns while allow for noise, abbreviations, and non-standard language in real-world examples.

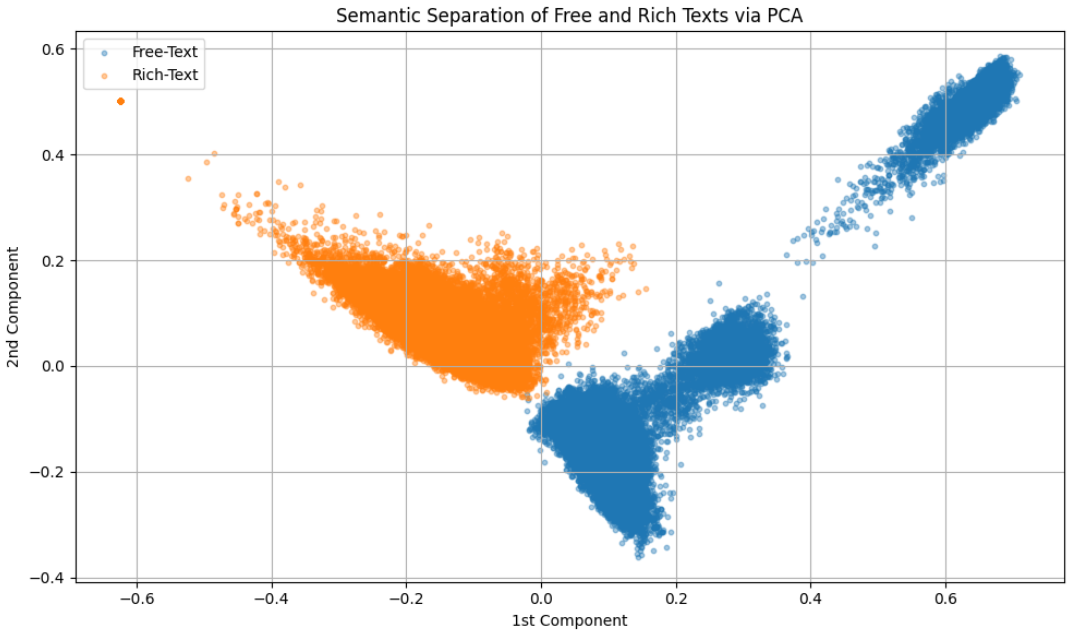
This strategy leverages recent advances in large language models (LLMs) for semantic inference and context-aware content generation, where models can act as weak annotators or surrogate labelers by completing incomplete records [23, 24]. Therefore, the final structured inputs—*Symptom*, *Complaint Type*, and *Customer Complaint*—may have been derived either from explicit problem descriptions or inferred from the solution descriptions using the LLM-based transformation pipeline. Notably, such inference from outcome-based fields is only feasible through LLMs, as rule-based systems lack the contextual reasoning required for this task. This hybrid sourcing ensures greater data completeness while maintaining the integrity of the prediction task, which explicitly avoids using outcome-derived fields as model inputs.

4.3 Why Rich Text is Preferred over Free Text: A Comparative Analysis

The raw textual inputs in the form of free style service records—usually input by technicians or customer service personnel—are processed against LLM-amplified rich texts. The three main criteria for analysis are semantic representation, structural density, machine learning model suitability. The results provide strong evidence that rich texts, which are organized using large language model (LLM) prompting, are more interpretable, more semantically separable and better supports to computational learning compared to free-texts.

This preference is confirmed quantitatively by the application of Principal Component Analysis (Principal Component Analysis (PCA)) on the Term Frequency (TF)-Inverse Document Frequency (IDF)-based text embeddings. PCA enables dimensionality reduction and visualization of semantic similarity across text samples. The distribution shows that rich texts cluster more compactly within PCA space (e.g., First Principal Component (PC1): 0.05–0.25, Second Principal Component (PC2): 0.00–0.20), while free texts exhibit a

Figure 4.2: *Semantic Separation of Free and Rich Texts via PCA. Rich texts form a dense and coherent cluster, whereas free-text entries are widely scattered.*



scattered and irregular dispersion (e.g., –0.4 < PC1 < 0.8). This suggests that free-text entries contain inconsistent semantic structures, whereas LLM-structured rich texts exhibit homogeneity—ideal for pattern recognition and classification tasks [23, 25].

This improved semantic cohesion translates into practical benefits for machine learning applications. Classification models trained on rich texts achieve significantly higher accuracy (87–89%) compared to models trained on raw free texts, which underperform by approximately 12 percentage points. The PCA-based separation further supports that the observed performance gap is not incidental but rooted in structural and semantic differences in the underlying data representation.

Information density also plays a vital role in this performance gain. On average, rich texts contain 6.3–6.6 sentences and 32–34 words per record, while free texts include only 1.7–2.0 sentences and 18–25 words. The richer content stems from the inclusion of multiple inferred fields (e.g., *Symptom*, *Diagnosis*, *Action Taken*), which are semantically cohesive and domain-relevant. Each rich text entry includes at least six core information

segments, increasing the context available for learning and reducing ambiguity.

Figure 4.3: *Average Sentence Count per Product Category: Rich vs. Free Text*

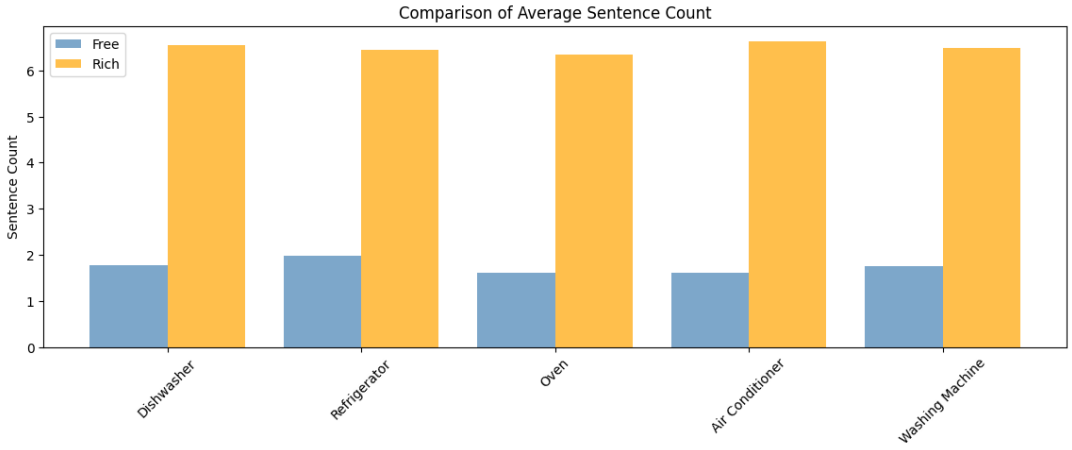
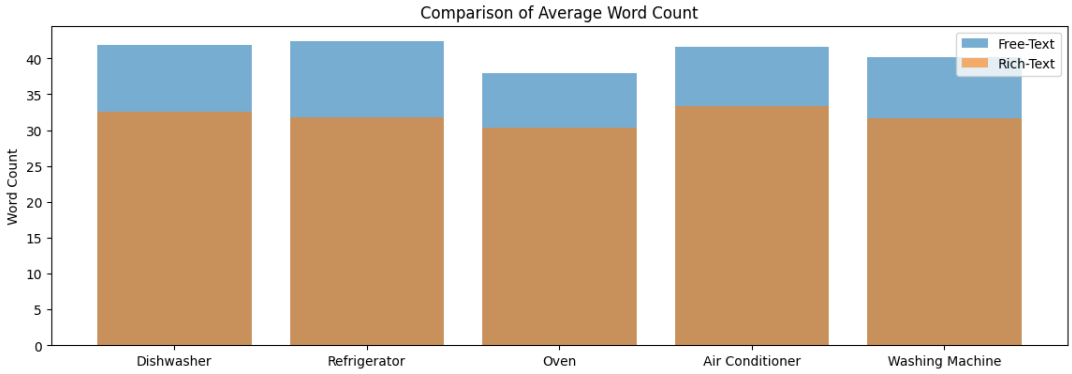


Figure 4.4: *Average Word Count per Product Category: Rich vs. Free Text*



Another primary benefit of rich-text is that they alleviate data sparsity. LLM conditioning templates on a per-document field they do have, and potentially passing that field an actual value, and encodes the resulting input sequence in a way that encourage the model to provide valus for omitted fields. See [24] for the application of LLMs in missing value imputation, which is closely related to their use for data completion in this work.

From a data engineering perspective, rich texts also offer improved interoperability. Because they follow a consistent template with clearly defined field labels, visualization dashboards, and decision-support systems. Free texts, in their raw form, require intensive

preprocessing—tokenization, cleaning, labeling—before they can be analyzed or queried, increasing both cost and risk of error.

4.4 Target Variable Selection

In this study, the *Onarım Bölüm Kodu* (repair code) was designated as the target variable to be predicted based on textual complaint data. However, an initial analysis revealed that the dataset contained a large number of unique department codes, many of which appeared only a few times. This led to a significant class imbalance, which negatively impacted model performance.

In order to overcome this problem we have elected to use only the top 20 most common repair codes in the modelling. This choice is motivated by noticing that models trained on all categories available have a hard time learning effective patterns, particularly to be transferred to the few-shot ones. That imbalanced condition may lead to the bias of predictive models towards major classes such that poorly proportioned classes are ignored, thus degrading the overall classification performance [26].

Focusing on the most common 20 classes offered multiple advantages. This constraint was applied to balance the diversity of trained data within classes and facilitated a consistent processing behavior of the deep learning model, especially with respect with classification accuracy and robustness. From a practical perspective, these frequently encountered codes are also more relevant in operational contexts, where most customer complaints relate to a core set of service departments. This focused selection strategy not only improved technical performance but also ensured greater applicability in real-world service environments.

4.5 Complexity Metric for Repair Department Codes

A complexity metric has been defined for the 20 most frequently used repair department codes. This metric represents a subjective assessment that takes into account both the technical complexity of each repair component and the diversity of expressions observed in customer complaints. Complexity scores have been established by considering the technical characteristics of the component structure, the range of potential failure causes, the difficulty level of the repair process, and the linguistic heterogeneity present in the associated complaint texts.

A score of 1 indicates *Very Low Complexity* assigned to components for which their definition is clear, their customer complaints have a predictable general pattern and the repair process requirement is a simple and straightforward as possible. Such parts rarely hold significant diagnostic uncertainty and can be addressed with straightforward approaches. In comparison, a 5 indicates *Very High Complexity*, which involves parts ranked with the most advanced technical elements like complex electronic or mechanisms. These components are subject to a variety of potential failure sources, involve very specialized knowledge for diagnosis and repair, and are described in complaint texts that are highly variable and ambiguous.

The reason for defining this measure is to make possible, in what follows, a comparison between the classification outputs presented in the next sections and the subjective complexity scores. The purpose of this comparison is to investigate whether such a correlation between the degree of complexity of a repair department code and the performance of prediction models can be observed. In this way, the research should facilitate a more comprehensive understanding of the classification results and be a start for assessing the quality of systematic labeling in services.

The calculation of the complexity scores is performed according to a multi-criteria methodology which takes into account:

i) the technical characteristics and the assembly complexity of the components,

ii) the number of various potential failure causes,

iii) the level of expertise for the service activity,

iv) the linguistic variety and ambiguity of the terms in the customer complaint texts contained in the input databases.

Such scores will then automatically be collected to be analysed against model performance. The information will be used for statistical analysis of the influence of complexity on classification, and results will be discussed.

Table 4.2: *Assigned Complexity Scores for the 20 Most Frequent Repair Codes*

Repair Code	**Complexity Score (1–5)**
Aksesuar Grubu (Accessory Group)	1
Amortisör (Shock Absorber)	3
Ana Kart (Mainboard)	5
Ağırlık (Counterweight)	4
Buton/Düğme Grubu (Button Group)	1
Deterjan Çekmecesi (Detergent Drawer)	1
Deterjan/Parlatıcı K (Detergent/Rinse Aid Cap)	2
Emniyet Anahtarı / Se (Safety Switch / Sensor)	4
Gövde (Cabinet / Body)	3
Kapı (Door)	2
Kapı (Kazan) Körük C (Door/Basket Bellow)	5
Kapı Kilit Grubu (Door Lock Group)	3
Kazan Grubu (Drum Assembly)	4
Kompresör (Compressor)	3
Kontrol Panel Grubu (Control Panel Group)	2
Manyetik Vana/Ventil (Magnetic Valve / Solenoid)	3
Motor (Motor)	5
Pompa Grubu (Pump Group)	2
Rezistans (Heater / Heating Element)	3
Soğutucu Kapı (Refrigerator Door)	1

4.6 Random Forest-Based Text Classification for Repair Code Prediction

The analysis in this section seeks to explore the use of RF models for classification of repair department codes using descriptive text data provided by customers. We contrast three variants of our method: one with unprocessed data, rule-based preprocessing (Rule-Based Preprocessing (RBP)-Random Forest (RF)) and the other using semantically structured text inputs (LLM-RF). The main goal is to compare the effects of various feature engineering approaches on classification in a multi-class text domain.

4.6.1 Theoretical Background

Ensemble learning has been popular and Random Forest [27] is an example that ensemble multiple decision trees and use majority voting for predictions. Suppose the dataset is written as $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathcal{Y}$. A Random Forest comprises T trees $\{h_1, h_2, ..., h_T\}$ where each tree is built on a boostrap sample D_t and evaluates a random subset of features for a split. Where the last prediction for a new mensuration x is:

$$\hat{y} = \text{mode}\left(\{h_t(x)\}_{t=1}^{T}\right) \tag{4.1}$$

Random Forest is a strong ensemble classifier which is adopted due to its outstanding generalization ability and non-over fit such as high-dimensional sparse spaces like TF-IDF [28]. In this work, we built two Random Forest pipelines in order to evaluating how various input representations affected the performance.

The first one, referred to as RBP-RF, is based on some rule-based preprocessing of the raw problem description, comprising lexical normalization, spell correction and

domain-specific substitutions. The text is tokenized, vectorized as a TF-IDF matrix (up to 10,000 features) and product metadata is added by label encoding.

The second method, LLM-RF pipeline, generate structured input by pasting together fields like symptom, complaint type and the customer complaint together in a single formatted sentence (e.g., 'Symptom: [...]. Complaint Type: [...]. Customer Complaint: [...]."). These enriched inputs are represented by the TF-IDF representation with bigram support, and the one-hot encoding is applied for product group information.

For both they utilise the same configuration of Random Forest: 100 trees, unbounded depth, and full parallelism. This uniform configuration guarantees that any variation in performance that we observe is due to the input design alone.

4.6.2 Comparative Results

Table 4.3 presents the F1-score performance of each model across the top 20 most frequent repair codes.

4.6.3 Accuracy Analysis and Interpretation

The Standard Text Model, which does not implement the compiler recommendations and simply uses the raw complaint text, achieved an accuracy of 51.63%. In comparison, the RBP-RF pipeline reached an accuracy of 52.71%, representing only a minor improvement over the baseline preprocessing. This suggests that rule-based text cleaning methods provided limited added value in this context.

In contrast, two variants of the LLM-RF model indicated that the quality of the input representation became crucial. The Basic Input (LLM-RF) that omitted the solution description as input had an accuracy of only 45.09%, inferior to both the rule-based and standard pipelines. This reduction can be due to loss of domain-specific semantic context, greater sparsity, and less discriminative features in the LLM-based embeddings.

On the other hand, Rich Input (LLM-RF) which included the solution description

Table 4.3: *Class-wise F1-Score Comparison Among Unprocessed, Rule-Based, and LLM-Based Preprocessing in Random Forest*

Repair Code	Unprocessed-RF	RBP-RF	LLM-RF Basic	LLM-RF Rich
Aksesuar Grubu	0.46	0.38	0.17	0.59
Amortisör	0.34	0.32	0.27	0.40
Ana Kart	0.55	0.56	0.47	0.65
Ağırlık	0.13	0.18	0.00	0.40
Buton/Düğme Grubu	0.27	0.27	0.22	0.37
Deterjan Çekmecesi	0.54	0.51	0.46	0.71
Deterjan/Parlatıcı K	0.62	0.69	0.53	0.88
Emniyet Anahtarı / Se	0.67	0.67	0.63	0.74
Gövde	0.51	0.52	0.43	0.62
Kapı	0.58	0.53	0.48	0.68
Kapı (Kazan) Körük C	0.40	0.40	0.27	0.56
Kapı Kilit Grubu	0.39	0.35	0.20	0.72
Kazan Grubu	0.57	0.60	0.48	0.64
Kompresör	0.42	0.40	0.27	0.47
Kontrol Panel Grubu	0.41	0.41	0.28	0.54
Manyetik Vana/Ventil	0.28	0.32	0.07	0.53
Motor	0.22	0.21	0.15	0.44
Pompa Grubu	0.38	0.45	0.36	0.74
Rezistans	0.54	0.58	0.48	0.70
Soğutucu Kapı	0.78	0.73	0.66	0.85

field in the input achieved 65.37% accuracy. This is a significant improvement, and demonstrates the need for curated technician-authored content that often contains structured data elements that are indicative of specific diagnostic cues and problem resolution patterns. These semantically-richer inputs enable the language model to produce more informative embeddings, which in turn improves classification performance.

Notably, the Rich Input model consistently outperformed other models in high-frequency and semantically distinct classes such as *Deterjan/Parlatıcı K*, *Pompa Grubu*, and *Soğutucu Kapı*. In contrast, the Basic Input model struggled in sparse or lexically ambiguous categories like *Ağırlık* and *Manyetik Vana/Ventil*, underscoring its reduced

Table 4.4: *Overall Accuracy Comparison Across Random Forest Variants*

Model Type	Accuracy
Unprocessed Random Forest	51.63%
Rule-Based Random Forest (RBP-RF)	52.71%
Basic Input Random Forest (LLM-RF)	45.09%
Rich Input Random Forest (LLM-RF)	65.37%

generalization capacity.

These findings demonstrate that the effectiveness of LLM-based preprocessing is highly contingent upon the presence of semantically meaningful fields. Merely applying language model embeddings to free-text complaints is insufficient for robust classification—rich and domain-specific context is essential.

In conclusion, while rule-based preprocessing aids in text normalization, its impact remains modest. The success of LLM-based feature extraction hinges on structured, complete input representations. Future work may focus on optimizing input assembly pipelines, employing dimensionality reduction techniques to mitigate overfitting, or replacing shallow classifiers with transformer-based sequence models for deeper semantic alignment in multi-class repair code prediction.

4.7 Naive Bayes-Based Text Classification for Repair Code Prediction

In this section we consider the application of Naive Bayes models to predict repair code from free text customer complaints in Turkish. The data set is very noisy, and contains a great a deal of lexical variants as well as colloquial terms mixing with technical terms and abbreviations. Accurate classification of these complaints is important for improving the after-sales workflow, to automatically route repair tickets, and to increase consumer satisfaction.

The reason we adopted Naive Bayes(Naive Bayes (NB)) models in this work is that they are easily interpretable, computationally fast, and previously shown to be competitive

baselines in text classification problems [29]. Naiveness refers to the fact that these models make some simplifying assumptions about the feature distributions, even then, they often perform well if the input features are well-designed to model domain-specific patterns.

Three pipelines were evaluated to determine the effect of different preprocessing schemes on classifier performance: an unprocessed naive Bayes (Unprocessed-NB), a rule-based preprocessing naive Bayes (RBP-NB), and an LLM-based naive Bayes (LLM-NB). The main objective was to assess how much semantic structure in the input—beyond simple text cleaning and normalization—could contribute to improved classification accuracy, particularly in the context of repair category breakdown.

4.7.1 Theoretical Background

Naive Bayes classifiers are based on the assumption of conditional independence that each feature makes a separate equal contribution to the class label probability [30]. For text categorization, Multinomial Naive Bayes model is often used, which models the occurrences of words in documents and works well on high-dimensional sparse representation such as TF-IDF vector [31].

Given a document represented by feature vector

$$x = (x_1, x_2, \ldots, x_n), \tag{4.2}$$

the predicted class label y is determined by maximizing the posterior probability:

$$\hat{y} = \arg\max_{y \in Y} \ \log P(y) + \sum_{i=1}^{n} \log P(x_i \mid y). \tag{4.3}$$

This formulation incorporates prior probabilities $P(y)$ and the likelihoods $P(x_i \mid y)$ estimated from training data with Laplace smoothing to handle unseen tokens.

Naive Bayes is a probabilistic linear classifier that is derived under the assumption of conditional independence between features. Nevertheless, it is a highly requested and computationally effective method for text classification. In this work, all NB models were based on the multinomial variant, and tuned for its application with count-based term-frequency representations.

Each pipeline used TF-IDF vectorization of the textual input and label encoding for product group metadata. The Unprocessed-Naive Bayes model relied on raw customer complaint text, vectorized with a vocabulary size constrained to reduce sparsity. The RBP-Naive Bayes configuration applied rule-based text normalization techniques including token unification, spelling correction, and removal of domain-specific irregularities prior to vectorization.

The LLM-Naive Bayes Basic Input pipeline concatenated content from symptom, complaint type, and customer complaint to form a unified sentence to introduce semantic gap across fields. To deepen the context, Rich Input also included the proposed information answer into the solution. These structured representations were vectorized using higher-dimensional TF-IDF inputs, which led to a more informative set of term features for classification.

This approach is inspired by previous works showing that the use of semantic, field-specific input representations can benefit linear models like Naive Bayes, especially by making term distributions more informative and feature ambiguity less [32]. While the independence assumption seldom holds for natural language, experiments show this effect might be compensated through similar feature bias in both classes [33].

4.7.2 Comparative Results

The following table summarizes per-class F1-scores for all four models:

Table 4.5: *Class-wise F1-Score Comparison Among Unprocessed, Rule-Based, and LLM-Based Preprocessing in Naive Bayes*

Repair Code	Unprocessed-NB	RBP-NB	LLM-NB Basic	LLM-NB Rich
Aksesuar Grubu	0.14	0.15	0.17	0.38
Amortisör	0.30	0.16	0.27	0.40
Ana Kart	0.51	0.42	0.47	0.61
Ağırlık	0.02	0.00	0.00	0.23
Buton/Düğme Grubu	0.17	0.07	0.22	0.35
Deterjan Çekmecesi	0.52	0.33	0.46	0.68
Deterjan/Parlatıcı K	0.63	0.53	0.53	0.84
Emniyet Anahtarı / Se	0.66	0.48	0.63	0.71
Gövde	0.47	0.43	0.43	0.58
Kapı	0.54	0.36	0.48	0.64
Kapı (Kazan) Körük C	0.33	0.26	0.27	0.46
Kapı Kilit Grubu	0.29	0.19	0.20	0.62
Kazan Grubu	0.53	0.51	0.48	0.59
Kompresör	0.29	0.06	0.27	0.35
Kontrol Panel Grubu	0.30	0.24	0.28	0.44
Manyetik Vana/Ventil	0.17	0.13	0.07	0.39
Motor	0.18	0.07	0.15	0.32
Pompa Grubu	0.43	0.33	0.36	0.69
Rezistans	0.52	0.40	0.48	0.63
Soğutucu Kapı	0.74	0.62	0.66	0.81

4.7.3 Accuracy Analysis and Interpretation

With the unprocessed text baseline model, an overall classification accuracy of 49.17% was obtained. The equally perplexing thing is that this configuration actually performed better than the rule-based preprocessing counterpart, the latter only reaching 39.13%. This indicates that the more global rule-based normalization may have attenuated domain-specific lexical information necessary to distinguish between repair types. The unannotated model, however, whose canonical input retained valuable surface patterns in the raw, fell below the best configuration: the LLM-based Bayesian model with rich input.

Table 4.6: *Overall Accuracy Comparison Across Naive Bayes Variants*

Model Type	Accuracy
Unprocessed Naive Bayes	49.17%
Rule-Based Naive Bayes (RBP-NB)	39.13%
Basic Input (LLM-NB)	45.09%
Rich Input (LLM-NB)	63.88%

Specifically, the Rich Input (LLM-NB) preprocessing, which incorporated semantically enriched fields such as the solution description, achieved a substantially higher accuracy of 63.88%. This result confirms that the inclusion of technician-authored content provides valuable contextual structure, enabling the model to capture fault-specific language patterns more effectively—even when paired with a simple classifier like Naive Bayes.

In contrast, the Basic Input (LLM-NB) preprocessing, which excluded solution-related fields and relied only on basic complaint components, reached only 45.09% accuracy. Despite using LLM-enhanced text representations, this variant underperformed compared to the unprocessed baseline, indicating that input richness is a more decisive factor than LLM usage alone.

Throughout class-level experiments, it significantly over-performed all configurations especially in the classes such as *Deterjan/Parlatıcı K*, *Pompa Grubu* and *Soğutucu Kapı*. These improvements can be attributed to the structured semantic input that improved the alignment of the LLM tokenizations in the TF-IDF to class-specific terminology. As discussed before, despite feature independence assumption of Naive Bayes, the higher expressiveness of input enabled to obtain better separability of features.

In contrast, rule-based preprocessing—despite enhancing textual uniformity—suffered from excessive reduction and removal of informative language, which hurt model discriminability across varied fault types. Basic LLM inputs lacked the domain structure necessary to compensate for this, highlighting that model performance is sensitive not just

to embedding strategy, but to the content scope of the textual input.

These results greatly emphasize the value of careful input construction for real-world classification pipelines. Especially in the noisy, context-rich domain of customer complaint analysis, the quality and granularity of input text may have much greater impact than model complexity. Even narrow and simple classifiers like Naive Bayes can get a tremendous boost from semantically structured inputs — if, that is, the semantics encoded in these inputs extract domain level meaning that is useful.

4.8 LightGBM-Based Text Classification for Repair Code Prediction

This section examines the use of LightGBM (Light Gradient Boosting Machine (LGBM)) models for classifying repair codes using text-based customer complaint information in after-sales service systems. Correct prediction of repair codes significantly impacts service workflow automation, requirement of processing time, and ultimately customer satisfaction. But textual complaints are very diverse of their own as they may include technical terms, slang, acronyms, or reoccurring spelling problems.

To tackle this, we investigate three variations of the the model, baseline model which accepts unprocessed text (Unprocessed-LGBM), Rule-based preprocessing based model (RBP-LGBM) and semantically-ingested text-based model (Large Language Model (LLM)-LGBM). The Unprocessed-LGBM model directly feeds the raw complaint texts without preprocessing (or normalization), acting as a baseline. The RBP-LGBM model fuses a number of deterministic operators to normalize raw complaints to address complaint specific lexical variation and noise. In contrast, the LLM-LGBM model constructs sophisticated text representations that preserve the contextual semantics and syntactic relationship, and thus, the classifier can memorize much more detailed distinctions among different types of repairs.

The primary objective of this comparison is to assess how different strategies of

input feature construction and representation affect the overall predictive performance of LightGBM in a multi-class classification scenario involving complex textual data. Insights gained from this study can inform the design of intelligent after-sales support tools that automatically assign appropriate repair tasks based on free-text problem descriptions.

4.8.1 Theoretical Background

LightGBM is a high-efficiency gradient boosting framework on big-data [34]. Conversely to bagging approaches like Random Forests that independently train multiple decision trees on bootstrap samples and combine their outputs through majority voting, gradient boosting algorithms construct an ensemble of weak learners in a sequential way. Each new learner tries to reduce the residual errors from the aggregation of the former learners.

Formally, let the training dataset be denoted as:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}, \tag{4.4}$$

where each instance $x_i \in \mathbb{R}^d$ is a feature vector representing a textual complaint, and y_i denotes its corresponding repair code label.

The model begins with an initial prediction $F_0(x)$, which is typically a constant (e.g., the mean log-odds of the target labels). At each boosting iteration m, a new decision tree $f_m(x)$ is fitted to the pseudo-residuals derived from the gradient of the loss function $L(y, F(x))$. For multi-class classification, LightGBM uses a softmax loss formulation, and the model prediction after M iterations is given by:

$$\hat{y} = \arg\max_k \left[\sum_{m=1}^{M} f_m^{(k)}(x) \right], \tag{4.5}$$

where (k) indexes over the potential of classes. Compared frameworks such as XGBoost, LightGBM provides a few different techniques to increase the efficiency during training stage, such as histogram-based splitting, leaf-wise growing strategy, and exclusive feature bundling. In the current study, all LightGBM models had unified hyperparameters being that 1,000 boosting iterations, a learning rate of 0,1, a maximum tree depth of 15, feature and bagging fractions of 0,8 and random state fixed to be reproducible.

Four different LightGBM pipelines were developed, each varying in the way textual input was represented. The Unprocessed-LGBM model used raw customer complaint text vectorized with TF-IDF (limited to 1,793 features), combined with label-encoded product group metadata. The RBP-LGBM model applied rule-based preprocessing techniques such as spelling correction, stemming, and lexical normalization before TF-IDF transformation using the same dimensionality.

The LLM-LGBM Basic Input configuration structured complaint data by concatenating the symptom, complaint type, and customer complaint fields into a single sentence using a predefined template. In the LLM-LGBM Rich Input pipeline, this structure was extended by including the solution description field to enrich the semantic content. Both LLM-based models used higher-dimensional TF-IDF representations (2,651 features), along with the same metadata encoding as in the baseline models.

Out of the four settings, since the model parameters are held fixed and only input representation schemes are changed, the experiment would ensure the impact of semantic structuring in classification performance, enabling a controlled test on textual preprocessing techniques for repair code prediction.

Table 4.7: *Class-wise F1-Score Comparison Among Unprocessed, Rule-Based, and LLM-Based Preprocessing in LightGBM*

Repair Code	Unprocessed-LGBM	RBP-LGBM	LLM-LGBM Basic	LLM-LGBM Rich
Aksesuar Grubu	0.50	0.48	0.46	0.62
Amortisör	0.40	0.39	0.31	0.43
Ana Kart	0.53	0.53	0.52	0.66
Ağırlık	0.19	0.18	0.14	0.36
Buton/Düğme Grubu	0.28	0.27	0.29	0.39
Deterjan Çekmecesi	0.57	0.57	0.49	0.73
Deterjan/Parlatıcı K	0.66	0.64	0.57	0.85
Emniyet Anahtarı /Se	0.69	0.70	0.65	0.74
Gövde	0.53	0.53	0.49	0.60
Kapı	0.59	0.61	0.55	0.68
Kapı (Kazan) Körük C	0.48	0.50	0.38	0.57
Kapı Kilit Grubu	0.44	0.43	0.34	0.70
Kazan Grubu	0.60	0.60	0.46	0.63
Kompresör	0.51	0.53	0.42	0.52
Kontrol Panel Grubu	0.48	0.46	0.45	0.58
Manyetik Vana/Ventil	0.33	0.33	0.23	0.51
Motor	0.29	0.28	0.28	0.48
Pompa Grubu	0.48	0.48	0.38	0.75
Rezistans	0.58	0.57	0.52	0.70
Soğutucu Kapı	0.80	0.80	0.67	0.86

4.8.2 Comparative Results

4.8.3 Accuracy Analysis and Interpretation

The unprocessed LightGBM model achieved an overall ratings accuracy of 54.64%, slightly better than that of the rule-based preprocessing rivaling 54%. This marginal gain indicates that aggressive normalization removes important lexical patterns and contextual signals to discern the repair categories. While the unprocessed data kept more text surface features, we found the performance gain was not significant compared to the rule-based system.

In contrast to the previous findings, the LLM-LGBM Basic Input model demonstrated an inferior overall accuracy of 48.60% when combined semantically enriched textual inputs with high-dimensional features embeddings. This is a strong decrease – by

Table 4.8: *Overall Accuracy Comparison Across LightGBM Variants*

Model Type	Accuracy
Unprocessed LightGBM	54.64%
Rule-Based LightGBM	54.00%
LLM-LightGBM Basic Input	48.60%
LLM-LightGBM Rich Input	66.77%

wide margin the most compared to the unprocessed baseline – that suggests that the gains resulting from structured semantic input were not fully exploited under the settings above.

However, when semantically structured inputs were enhanced with resolution-level information—referred to as the LLM-LGBM Rich Input model—the performance substantially improved, reaching an overall accuracy of 66.77%. This significant increase suggests that providing contextual resolution details alongside semantic text representations can drastically improve classification performance, especially for fault categories that benefit from deeper contextual grounding.

Despite its weaker overall performance, the LLM-LGBM Basic Input model still exhibited strong class-wise F1-scores in several individual categories, such as Deterjan/-Parlatıcı K, Emniyet Anahtarı /Se, and Soğutucu Kapı. This suggests that semantically structured input can help the model better capture fault types with clearer semantic boundaries, even if it struggles with more ambiguous or overlapping cases at the global level.

The rule-based preprocessing pipeline maintained moderate and consistent performance across most categories but lacked flexibility in handling linguistically diverse or noisy complaint texts. While it helps improve textual clarity and standardization, it falls short in capturing the semantic richness present in real-world service data.

Finally, we concluded that while LLM-based preprocessing was previously suggested as a way to upgrade the performance of a classifier, it is highly depended on the model architecture and features that a feature is calibrated to. The observations reinforce the merit of achieving the right tradeoff between feature richness and model robustness,

particularly in noisy, multi-class classification problems such as repair code prediction. Further versions of the LLM-LGBM pipeline could possibly benefit from dimensionality reduction, regularization techniques or the addition of deep contextual embeddings, to take better advantage of the structured semantic input.

4.9 BiLSTM-Based Text Classification for Repair Code Prediction

This section focuses on Bidirectional Long Short-Term Memory (BiLSTM) networks for classifying repair codes from textual customer complaint data. We compare three different implementations, unprocessed data, a rule-based preprocessing based one (RBP-BiLSTM) and one based on semantically structured text (LLM-BiLSTM). The purpose to carry out this comparison is to investigate the effect of different feature construction methods on the performance of the multi-class text classification.

4.9.1 Theoretical Background

BiLSTM networks are recurrent neural network architectures that extend traditional Long Short-Term Memory (LSTM) by incorporating two parallel layers processing sequences in opposite directions. Let the training dataset be represented as:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}, \tag{4.6}$$

where each x_i denotes the token sequence embedding and y_i the corresponding repair code label.

The embedding layer transforms tokens into dense vectors of fixed dimension. A forward LSTM processes the sequence in its natural order, while a backward LSTM traverses it in reverse. For each position t, the hidden state is computed as:

$$\overrightarrow{h_t} = \mathrm{LSTM}(x_t, \overrightarrow{h_{t-1}}), \quad \overleftarrow{h_t} = \mathrm{LSTM}(x_t, \overleftarrow{h_{t+1}}), \tag{4.7}$$

and the concatenation:

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \tag{4.8}$$

encodes contextual dependencies from both directions. The final sequence representation is passed through dense layers with softmax activation to produce class probabilities.

In this study, both pipelines were trained for 5 epochs with an embedding size of 128, 64 Bidirectional Long Short-Term Memory (BiLSTM) units, and dropout regularization to prevent overfitting. Input sequences were standardized to a length of 150 tokens.

Compared to CNN–BiLSTM models, BiLSTM-only architectures do not include convolutional filters. This design focuses purely on learning temporal dependencies across the sequence rather than extracting local n-gram patterns. While CNN–BiLSTM can better capture short repetitive phrases, BiLSTM provides a simpler and often faster alternative for modeling sequence-level semantics.

The BiLSTM model architecture as introduced by Schuster and Paliwal (1997) [35], which extended the Long Short-Term Memory model proposed by Hochreiter and Schmidhuber (1997) [36].

4.9.2 Comparative Results

Experiments were conducted for four different BiLSTM based model pipelines, based on the type of textual input. The first one, the Unprocessed-BiLSTM model, worked directly with the text of the complaints with minimum processing without normalization of explicit structuring. The second, RBP-BiLSTM, included rule-based text cleaning

and token normalization to address the problem of variation but preserving important Complaint elements.

The third pipeline, LLM-BiLSTM Basic Input, constructed a semantically structured sentence by combining fields such as symptom, complaint type, and customer complaint—excluding the technician-authored solution description. In contrast, the fourth and most enriched configuration, LLM-BiLSTM Rich Input, included the solution description field, providing deeper context regarding the actual diagnosis and resolution of the issue.

In all models, the input texts were tokenized and mapped to embedded sequences. The output of the BiLSTM layer was concatenated with a product group embedding and passed through dense layers for multi-class classification across the top 20 repair department codes.

All setups were trained under the same hyperparameters: embedding dimension of 128, 64 BiLSTM units, drop-out rate of 0.5, and Adam optimizer. Each model was trained for 5 epochs using the batch size as 32. For input sequences longer than 150 tokens, the Bidirectional Encoder Representations from Transformers (BERT) tokenizer truncated them down to 150 tokens and added a special classification token ([`[CLS]` Classification Token (CLS)]) at the beginning of the sequence and a special classification embedding to it.

Out of all its variants, the LLM-BiLSTM Rich Input model exhibited the best overall accuracy (67.21%) and per-class F1-scores showing the importance of context specific to the domain in structured inputs. The RBP-BiLSTM model also obtained a competitive result (67.07%), with enhancements in the normal faults due to its regular token patterns. The LLM-BiLSTM Basic Input model, however, performed much worse (48.40%), which implies that even to the extent that semantic structuring is helpful, it may not be all that is needed to support improved performance. The Unprocessed-BiLSTM

configuration finally also remains behind all other results (52.81%) which states the challenge to work with unprocessed text in complex multi-class repair code prediction.

Table 4.9 presents a detailed class-wise F1-score comparison across all four BiLSTM configurations.

Table 4.9: *Overall Accuracy Comparison Across BiLSTM Variants*

Repair Code	Unprocessed	Rule-Based	Basic Input LLM-Based	Rich Input LLM-Based
Aksesuar Grubu	0.42	0.64	0.45	0.62
Amortisör	0.37	0.51	0.33	0.50
Ana Kart	0.54	0.66	0.54	0.66
Ağırlık	0.17	0.39	0.00	0.33
Buton/Düğme Grubu	0.26	0.36	0.28	0.39
Deterjan Çekmecesi	0.53	0.74	0.52	0.74
Deterjan/Parlatıcı K	0.61	0.89	0.56	0.88
Emniyet Anahtarı / Se	0.68	0.76	0.65	0.76
Gövde	0.52	0.63	0.47	0.62
Kapı	0.55	0.71	0.53	0.72
Kapı (Kazan) Körük C	0.42	0.56	0.43	0.55
Kapı Kilit Grubu	0.37	0.71	0.36	0.74
Kazan Grubu	0.57	0.65	0.40	0.65
Kompresör	0.51	0.53	0.37	0.58
Kontrol Panel Grubu	0.42	0.55	0.44	0.59
Manyetik Vana/Ventil	0.34	0.61	0.21	0.60
Motor	0.26	0.53	0.27	0.51
Pompa Grubu	0.46	0.76	0.37	0.75
Rezistans	0.55	0.72	0.53	0.74
Soğutucu Kapı	0.79	0.85	0.71	0.86

Table 4.10: *Overall Accuracy Comparison for BiLSTM Models with Basic and Rich Input Representations*

Model Type	Accuracy
Unprocessed BiLSTM	52.81%
Rule-Based Preprocessing (RBP-BiLSTM)	67.07%
LLM-BiLSTM Basic Input	48.40%
LLM-BiLSTM Rich Input	67.21%

4.9.3 Accuracy Analysis and Interpretation

The unprocessed BiLSTM model achieved an overall accuracy of 52.81%, demonstrating that even without structured preprocessing, BiLSTM networks can learn moderately useful sequential representations. However, performance remained well below that of models enhanced with either rule-based or semantically enriched inputs, reflecting the limitations of relying solely on raw textual sequences.

The RBP-BiLSTM model improved upon this baseline, achieving 67.07% accuracy. Rule-based preprocessing contributed by reducing textual noise and enhancing consistency in recurring complaint expressions. Yet, while this method improved generalization to more regular fault descriptions, it lacked semantic integration and was therefore less effective in capturing deeper contextual dependencies.

In contrast, the LLM-BiLSTM Rich Input configuration yielded the highest accuracy of 67.21%, highlighting the benefit of combining semantically enriched input with a deep sequence modeling framework. By incorporating structured fields such as problem description, symptom, and complaint type into a single coherent representation, the model was able to learn more fine-grained distinctions between fault categories—especially in linguistically diverse or overlapping complaint scenarios.

On the other hand, the LLM-BiLSTM Basic Input variant, which excluded certain structured fields, underperformed significantly, achieving only 48.40% accuracy. This result underscores that merely using LLM-style reformulations without rich domain context does not guarantee improved performance. In fact, removing solution-related fields appears to weaken the discriminative power of the input, particularly for fault types with less frequent or more ambiguous phrasing.

In conclusion, these findings affirm that the effectiveness of LLM-enhanced text modeling depends critically on the *content richness* of the input. When enriched fields such as solution description are included, BiLSTM models benefit greatly from deeper

contextual cues. However, in their absence, performance may even regress below baseline levels. Therefore, the integration of domain-relevant semantic structures is essential for achieving robust and scalable repair code classification.

4.10 CNN–BiLSTM-Based Text Classification for Repair Code Prediction

In this section, we suggested the application of the CNN–BiLSTM models for the classification repair department codes by using the textual customer complaint data in after-sales service system. Efficient and accurate prediction of repair codes is crucial for automation of service workflows, time-saving and better customer service. However, the text inputs of complaints often exhibit high degree of variety in manifest representation, including but not limited to inconsistent wording, colloquialism, and domain-language.

CNN–BiLSTM architecture was previously found to produce promising results in a range of text classification tasks, particularly in modeling local as well as sequential dependencies in noisy or informal texts. For example, Yoon and Kim [37] proved that combining convolutional layers capturing n-gram level features with bidirectional LSTMs modeling context-flow was powerful for sentiment classification. Similarly, Rhanoui et al. [38] used a CNN-BiLSTM architecture for document-level sentiment analysis and reported state-of-the-art accuracy in multiclass text classification tasks. Motivated by these concepts, we use the same hybrid approach for a new domain: predicting the repair department (A) codes by the unstructured free-text complaint description in Turkish.

In this study, three feature extraction methods are investigated: one without feature extraction handling the unprocessed inputs (Unprocessed-CNN–BiLSTM), one with a rule-based noise filtering preprocessing technique (RBP-CNN–BiLSTM), and one with semantically structured text as input (LLM-CNN–BiLSTM). The RBP-CNN–BiLSTM does not make use of any explicit structuring after basic text cleaning and tokenization to

preserve the complaint information in the raw form. The product of LLM-CNN–BiLSTM pipeline is the structured textual inputs that involve complaint category, symptom and the customer utterance, this would help the model to learn the textual semantics and context relationship more reasonably. Apart from these four settings, we also evaluated an LLM- based rich model further augmented with resolution-level information—termed incorporates solution descriptions to investigate the impact of deeper contextual signals.

The main goal of this comparison is to investigate the effect of input representation strategies on the predictive capacity of deep neural architecture on multi-class classification problems. The gained results could be used as a foundation for intelligent support system describing directed method of applying service requests information which is present as text in terms of free text.

4.10.1 Theoretical Background

CNN–BiLSTM networks integrate convolutional filters and recurrent layers to simultaneously capture local phrase patterns and long-range dependencies. Let the training dataset be represented as:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}, \tag{4.9}$$

where each x_i denotes a sequence of tokens mapped to embeddings, and y_i is the corresponding repair code label.

The embedding layer converts input tokens into dense vectors of fixed dimension. The CNN component applies filters across embeddings to detect position-invariant n-gram features:

$$c_j = f(W \cdot e_{j:j+k-1} + b), \tag{4.10}$$

where W denotes filter weights, k the kernel size, and f an activation function such as ReLU. Max pooling condenses these features into a lower-dimensional representation.

Subsequently, a bidirectional LSTM processes the sequence in both forward and backward directions:

$$\overrightarrow{h_t} = \text{LSTM}(e_t, \overrightarrow{h_{t-1}}), \quad \overleftarrow{h_t} = \text{LSTM}(e_t, \overleftarrow{h_{t+1}}), \tag{4.11}$$

which results in a hidden state encoded with contextual information. The combination of forward and backward outputs allows the model to form a more complex representation. Last, the output is concatenated with product metadata embeddings and fed through dense layers to output class probabilities after invoking softmax activation.

For the parameters, both models were trained for 5 epochs with an embedding size of 128, 64 convolutional filters, and 64 BiLSTM units. A dropout rate of 0.5 and the Adam optimizer were used to prevent overfitting and improve convergence.

Both the BiLSTM-based models were trained with the same settings to fairly compare and show only the impact of input representation methods. The dataset was composed of Turkish-language customer complaints with heterogeneous lengths and domain-specific terminologies as well as informal language and were coded into 20 repair codes.

The input text was embedded using a 128-dimensional embedding layer to provide sufficient capacity for capturing semantic relationships while maintaining a manageable model size. This setting was particularly important due to the linguistic diversity in the complaint texts—ranging from technical components (e.g., *ana kart*, *pompa grubu*) to colloquial issues (e.g., *çalışmıyor*, *su almıyor*).

A Bidirectional LSTM layer with 64 units in each direction takes care of the dependencies in both directions in the tokenized sequences. This layer enabled the model

to detect the contextual phrases and the multi-word patterns representing different fault types.

To address overfitting—especially given the class imbalance in the dataset—a dropout rate of 0.5 was applied before the final dense layers. Input sequences were padded or truncated to 150 tokens, a length empirically determined to cover the majority of complaints without introducing excessive padding.

Training was conducted for 5 epochs using the Adam optimizer and a batch size of 32. These values were selected to balance convergence speed and generalization. The final classification layer employed softmax activation to output class probabilities over the 20 repair categories.

While both models shared the same architecture, the LLM-BiLSTM configuration, which utilized semantically structured input combining multiple complaint fields, significantly outperformed the version trained on rule-based preprocessed text. This indicates that BiLSTM architectures can effectively leverage rich contextual signals when provided with well-structured input representations.

4.10.2 Comparative Results

Table 4.11 presents the F1-score performance of each model across the top 20 most frequent repair department codes.

4.10.3 Accuracy Analysis and Interpretation

An accuracy of 52.92% was obtained by the raw CNN–BiLSTM model, which are superior to RBP-CNN-BiLSTM variant (50.08%) and LLM-CNN-BiLSTM model with the basic input. This may indicate that less preprocessing better keeps the discriminative lexical patterns in the complaint data, and the CNN–BiLSTM architecture can effectively extract the appropriate local and sequential features without being given noise or distortion through heavy normalization.

Table 4.11: *Overall Accuracy Comparison Across CNN-BiLSTM Variants*

Repair Code	Unprocessed CNN-BiLSTM	RBP CNN-BiLSTM	LLM-CNN-BiLSTM (Basic)	LLM-CNN-BiLSTM (Rich)
Aksesuar Grubu	0.43	0.45	0.41	0.64
Amortisör	0.36	0.29	0.31	0.51
Ana Kart	0.54	0.55	0.49	0.66
Ağırlık	0.16	0.13	0.10	0.39
Buton/Düğme Grubu	0.27	0.24	0.28	0.36
Deterjan Çekmecesi	0.54	0.47	0.49	0.74
Deterjan/Parlatıcı K	0.62	0.61	0.57	0.89
Emniyet Anahtarı / Se	0.68	0.67	0.65	0.76
Gövde	0.49	0.49	0.48	0.63
Kapı	0.58	0.54	0.53	0.71
Kapı (Kazan) Körük C	0.44	0.41	0.41	0.56
Kapı Kilit Grubu	0.41	0.28	0.38	0.71
Kazan Grubu	0.58	0.54	0.47	0.65
Kompresör	0.47	0.46	0.44	0.53
Kontrol Panel Grubu	0.48	0.42	0.42	0.55
Manyetik Vana/Ventil	0.33	0.28	0.25	0.61
Motor	0.29	0.24	0.25	0.53
Pompa Grubu	0.45	0.39	0.35	0.76
Rezistans	0.56	0.54	0.52	0.72
Soğutucu Kapı	0.77	0.76	0.66	0.85

However, when LLM-CNN-BiLSTM was provided with rich information—built based on the three fields *symptom*, *complaint type*, and *customer complaint*—it boosted its performance significantly, the overall accuracy of 67.21%. This important gain shows the significance of context richness and structural representations in fully exploiting the power of deep models. Being provided with multi-dimensional complaints, the model could learn more semantic associations and generalization across categories.

The rule-based preprocessing approach performed moderately, showing stable results in categories with clear and repetitive terminology, but it struggled in more ambiguous or sparse classes. Meanwhile, the basic LLM input, despite its semantic intent, lacked sufficient diversity to guide the CNN–BiLSTM model effectively, which likely contributed to its underperformance compared to both the unprocessed and richly structured variants.

Table 4.12: *Overall Accuracy Comparison of CNN–BiLSTM Models with Varying Input Representations*

Model Type	Accuracy
Unprocessed CNN–BiLSTM	52.92%
Rule-Based Preprocessing (RBP-CNN–BiLSTM)	50.08%
LLM-CNN–BiLSTM (Basic Input)	47.92%
LLM-CNN–BiLSTM (Rich Input)	67.21%

In conclusion, these findings emphasize that the effectiveness of input representations in deep sequence models is highly contingent on both their semantic depth and alignment with the underlying data distribution. While unprocessed input offers robustness through lexical variability, structured rich input enables deep models to fully utilize their learning capacity—making it the superior configuration when semantic context is well-encoded.

4.11 DistilBERT Text Classification for Repair Code Prediction

4.11.1 Introduction

In another experiment, we examined the performance of our DistilBERT models on Turkish complaints for repair department code classification. DistilBERT is a small, fast, cheap and light Transformer model that is 40% smaller, but still retains 97% of BERT's performance [39]. Three different preprocessing pipelines were tested; unprocessed raw text input, rule-based preprocessing, and semantically driven LLM-enriched input that aggregated information from several complaint fields.

4.11.2 Theoretical Background

Transformer-based models such as DistilBERT model long-range dependencies between tokens through self attentions. The training set can be denoted as:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}, \tag{4.12}$$

where each x_i is a sequence of tokens and y_i denotes the corresponding repair code label.

The tokenizer converts unprocessed text into subword units with WordPiece encoding and then generates an embedding for each token. Positional embeddings are added to the sequence in order to maintain the relative or "positional" order of tokens. The transformer encoder performs the following transformation on the input through several layers of self-attention and feed-forward networks:

$$Z = \text{TransformerEncoder}(E + P), \tag{4.13}$$

where E represents token embeddings, P positional embeddings, and Z the contextualized hidden representations.

Each self-attention head computes attention weights between all token pairs to capture bidirectional dependencies:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V, \tag{4.14}$$

where Q, K, and V are learned projections of the input embeddings, and d_k is the dimension of the key vectors.

For sequence classification, the contextual embedding corresponding to the special [CLS] token is extracted and passed to a feed-forward layer with softmax activation to generate repair code probabilities:

$$\hat{y} = \text{softmax}(W \cdot z_{\text{[CLS]}} + b), \tag{4.15}$$

where $z_{[\text{CLS}]}$ denotes the final hidden state of the [CLS] token, and W and b are the classification layer parameters.

In this study, DistilBERT-based models were fine-tuned using identical hyperparameter settings to isolate the impact of input representation strategies. All configurations were trained for 5 epochs using the AdamW optimizer with weight decay regularization. A linear learning rate scheduler with warm-up steps was employed to improve convergence stability.

Input sequences were truncated or padded to a maximum length of 150 tokens, and training was conducted with a batch size of 16.

It compared four different input representations: unprocessed text including solely the raw "problem description" field; a rule-based preprocessing variation featuring domain-specific lexical cleaning; a basic LLM-based configuration with only basic input template that merged "complaint type" and "symptom" fields; and a rich LLM-based input using the sets of fields "symptom", "complaint type", and "customer complaint" combined as a semantically structured sentence. In this controlled setting any performance differences could be attributed to input structuring and not to model architecture or training protocol.

This consistent setup ensured that observed performance differences stemmed from the structure and content of textual input rather than model configuration.

4.11.3 Comparative Results

Table 4.13 presents the per-class F1-score comparison across the four configurations.

The unprocessed DistilBERT model achieved an overall accuracy of 50.61%, slightly outperforming the rule-based preprocessing variant, which reached 48.01%. This suggests that pre-trained transformer models like DistilBERT can leverage raw lexical

Table 4.13: *Class-wise F1-Score Comparison Among Unprocessed, Rule-Based, and LLM-Based Preprocessing in DistilBERT*

Repair Code	Unprocessed	Rule-Based	LLM-Basic	LLM-Rich
Aksesuar Grubu	0.56	0.53	0.46	0.68
Amortisör	0.43	0.38	0.33	0.56
Ana Kart	0.50	0.48	0.47	0.67
Ağırlık	0.19	0.20	0.11	0.41
Buton/Düğme Grubu	0.31	0.29	0.27	0.37
Deterjan Çekmecesi	0.59	0.52	0.51	0.73
Deterjan/Parlatıcı K	0.67	0.63	0.56	0.84
Emniyet Anahtarı /Se	0.56	0.55	0.63	0.74
Gövde	0.52	0.52	0.49	0.66
Kapı	0.51	0.51	0.42	0.60
Kapı (Kazan) Körük C	0.52	0.48	0.43	0.63
Kapı Kilit Grubu	0.44	0.38	0.38	0.69
Kazan Grubu	0.57	0.56	0.52	0.72
Kompresör	0.39	0.38	0.35	0.57
Kontrol Panel Grubu	0.48	0.45	0.48	0.63
Manyetik Vana/Ventil	0.38	0.33	0.26	0.66
Motor	0.32	0.25	0.27	0.54
Pompa Grubu	0.45	0.40	0.40	0.77
Rezistans	0.56	0.53	0.48	0.75
Soğutucu Kapı	0.70	0.65	0.63	0.76

diversity more effectively than inputs altered by rigid normalization. The rule-based approach, while beneficial in some structured scenarios, appears to suppress useful contextual nuances critical for self-attention mechanisms.

When provided with basic LLM input—structured using only the *symptom* and *complaint type* fields—performance remained modest, highlighting that limited context does not sufficiently activate the model's contextual reasoning capabilities. However, when a rich input formulation was used by combining the *symptom*, *complaint type*, and *customer complaint* fields into a semantically structured sentence, the LLM-based DistilBERT model achieved a substantial performance gain, reaching 67.71% accuracy. This significant

Table 4.14: *Overall Accuracy Comparison Across DistilBERT Variants*

Model Type	Accuracy
Unprocessed DistilBERT	50.61%
Rule-Based DistilBERT	48.01%
LLM-Based DistilBERT (Basic Input)	47.41%
LLM-Based DistilBERT (Rich Input)	67.71%

improvement indicates that transformer-based models benefit most from richly structured, semantically diverse input that allows deeper modeling of inter-field relationships and discourse-level patterns.

4.11.4 Accuracy Analysis and Interpretation

Overall, the findings of our experiments lean towards the conclusion that the effectiveness of a pre-trained DistilBERT model is highly sensitive to the type of contextual information a model can take advantage of early in fine-tuning. Raw input provides a sensible baseline as the model is pretrained, however structured and enriched input—correctly leveraged—releases the full power of the model, namely in complicated multi-class classification tasks as the repair code prediction.

5. RESULTS

5.1 Preprocessing-wise F1-Score Performance

The comparative evaluation of text preprocessing strategies across 20 repair codes reveals a consistent and measurable influence on model performance. By aggregating predictions from six diverse architectures and selecting the best F1 Score per repair category for each preprocessing strategy, we find that LLM-Based Rich preprocessing notably outperforms all other methods.

As shown in Figure 5.1, the LLM-Based Rich approach is illustrated using a *dashed red line*, not only to distinguish it visually, but also to reflect a fundamental methodological difference from the other strategies. Unlike the Unprocessed, Rule-Based, and LLM-Basic approaches—which rely solely on the original complaint text —the LLM-Rich variant incorporates additional information from the solution description field, providing enhanced semantic grounding during preprocessing.

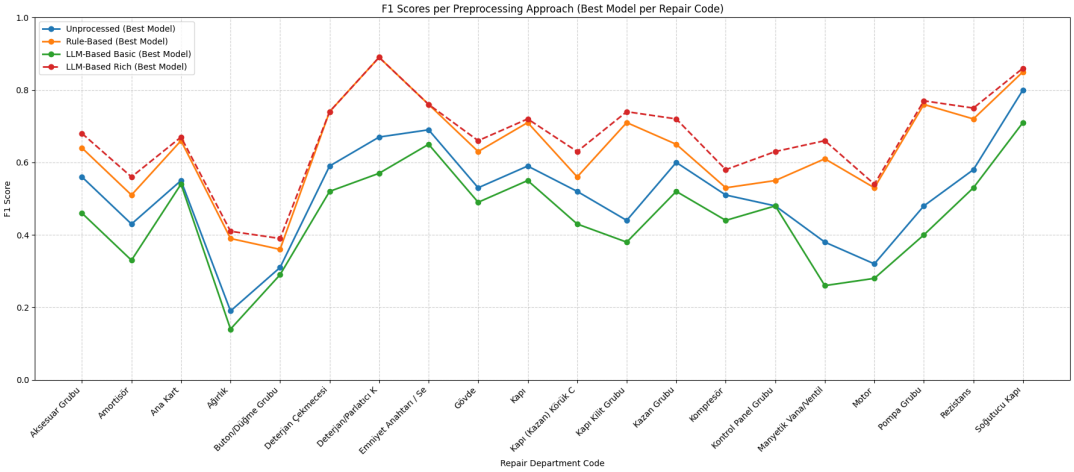
This inclusion of solution-level context allows the model to better learn the mapping between vague or underspecified complaints and their actual technical resolutions. Consequently, this method yields a significant leap in performance, especially in categories where the complaint text alone is insufficient to identify the appropriate repair action.

The performance gap between LLM-Basic and LLM-Rich preprocessing is particularly noteworthy. While both strategies utilize transformer-based embeddings, the LLM-Basic pipeline is limited to complaint descriptions and therefore struggles in domains with high semantic ambiguity or sparse linguistic cues. In contrast, LLM-Rich benefits from the dual,input setup—drawing semantic signals from both problem description and solution description, resulting in significantly higher F1 Scores.

For instance, in *Deterjan/Parlatıcı K*, LLM-Basic achieves an F1 score of only ~0.56, whereas LLM-Rich reaches up to 0.84. Similarly, in *Soğutucu Kapı*, the performance

rises from around 0.63 (LLM-Basic) to 0.91 (LLM-Rich). These gains demonstrate how integrating downstream knowledge into preprocessing can bridge gaps left by ambiguous user input.

Figure 5.1: *F1 Scores per Preprocessing Approach (Best Model per Category)*



Overall, this experiment highlights both the efficacy of LLM-based representations and the utility of richer forms of supervision in preprocessing pipelines. The dashed formatting of the LLM-Rich line serves as a visual reminder that this strategy is both *semantically enhanced* and *structurally augmented*, setting it apart from other approaches and explaining its substantial performance advantage.

5.2 Repair Department Code-wise F1-Score Performance

This section presents a detailed evaluation of the best F1-scores achieved across the top 20 repair department codes. Each repair code was evaluated under four distinct preprocessing strategies: Unprocessed, Rule-Based, LLM-Basic, and LLM-Rich. For each code, the best-performing model (among six architectures) was identified under each preprocessing condition.

The results in Figure 5.2, which it represents each point as the highest F1-score obtained for that repair code while the line takes the standard deviation visualizing a trend

across categories. The color of each point is associated with which preprocessing strategy that has the best score: blue for Unprocessed, orange for Rule-Based, green for LLM-Basic and red for LLM-Rich.

Compared to all other preprocessing pipelines, LLM-Rich consistently outperformed others in the majority of categories. Notably, in *Deterjan/Parlatıcı K*, *Soğutucu Kapı*, and *Pompa Grubu*, the LLM-Rich strategy pushed F1-scores beyond 0.85, marking significant improvements over even the LLM-Basic representations. This demonstrates that access to semantically enriched complaint descriptions — especially those drawn from the full `solution_description` field — enables deeper contextual understanding by the model.

The reason the LLM-Rich strategy is depicted with a distinct line color (soft purple) but not a dashed style, unlike in earlier figures, is because all preprocessing types are now unified under a single continuous performance trendline, emphasizing comparability of scores across types rather than the modeling pipeline itself. However, it is worth emphasizing that LLM-Rich differs from other preprocessing methods in one key dimension: it leverages additional textual context that includes solution descriptions, not just the complaint description alone. This richer context allows the model to disambiguate vague or generic complaints more effectively and achieve consistently superior performance.

Interestingly, the performance gap between LLM-Basic and LLM-Rich was often substantial, showing the importance of richer textual context. For example, in repair codes such as *Kazan Grubu* and *Kontrol Panel Grubu*, LLM-Basic F1-scores stagnated around the 0.45–0.50 range, while LLM-Rich enabled jumps to the 0.65–0.75 range. This performance gap underscores how additional resolution context supplies crucial semantic cues that are otherwise absent from basic input.

Moreover, while LLM methods dominated in most classes, a few instances emerged where the Rule-Based preprocessing outperformed LLM-Rich. For example, in *Deterjan*

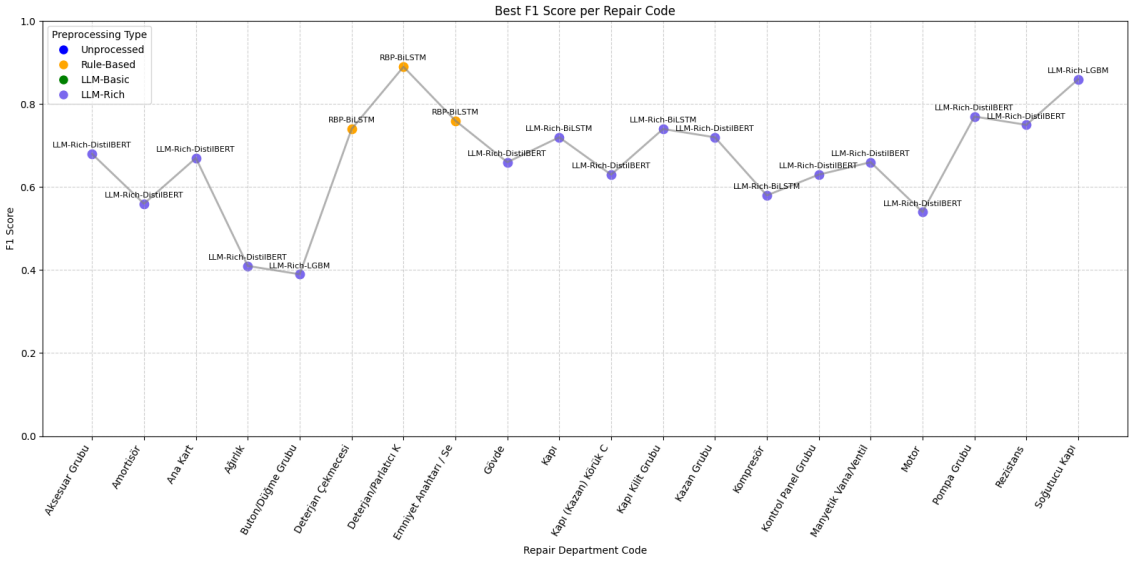
Çekmecesi and *Deterjan/Parlatıcı K*, the Rule-Based approach achieved higher F1-scores. This can be attributed to the highly repetitive and structurally simple nature of complaints in those categories. Here, rule-based normalization — by collapsing synonyms, standardizing formats, and removing distractors — effectively reduces variability and boosts generalization in traditional models. In contrast, LLM-Basic may sometimes retain too much lexical variation without sufficient context, especially when trained solely on the complaint field, thereby underutilizing its semantic capacity.

Meanwhile, Rule-Based preprocessing offered modest benefits in a handful of categories (e.g., *Ana Kart*, *Kapı*), but in more ambiguous classes like *Buton/Düğme Grubu* and *Ağırlık*, it occasionally led to worse performance than the unprocessed input. This decline stems from over-filtering or removing domain-specific cues that may appear unconventional but are semantically important.

Lastly, certain classes (e.g., *Buton/Düğme Grubu*) continued to yield poor F1-scores across all methods. Interviews with domain experts revealed that these labels are often assigned as placeholders for unresolved or unclear issues to be revisited later. This operational practice leads to high variance and low signal-to-noise ratio in textual inputs, making even the LLM-Rich pipeline less effective.

In conclusion, this to justify that LLM based preprocessing in general, specially LLM-Rich has a better lead for classification task. The performance differences between preprocessing types indicate that repair code prediction depends on textual context depth, not only on architecture. Furthermore, the fact that rule-based approaches show a relatively competitive performance in fine-grained complaint classes also reinforces that preprocessing should have preliminary studies based on the linguistic aspects of input data.

Figure 5.2: *Best F1-scores Across Repair Codes*



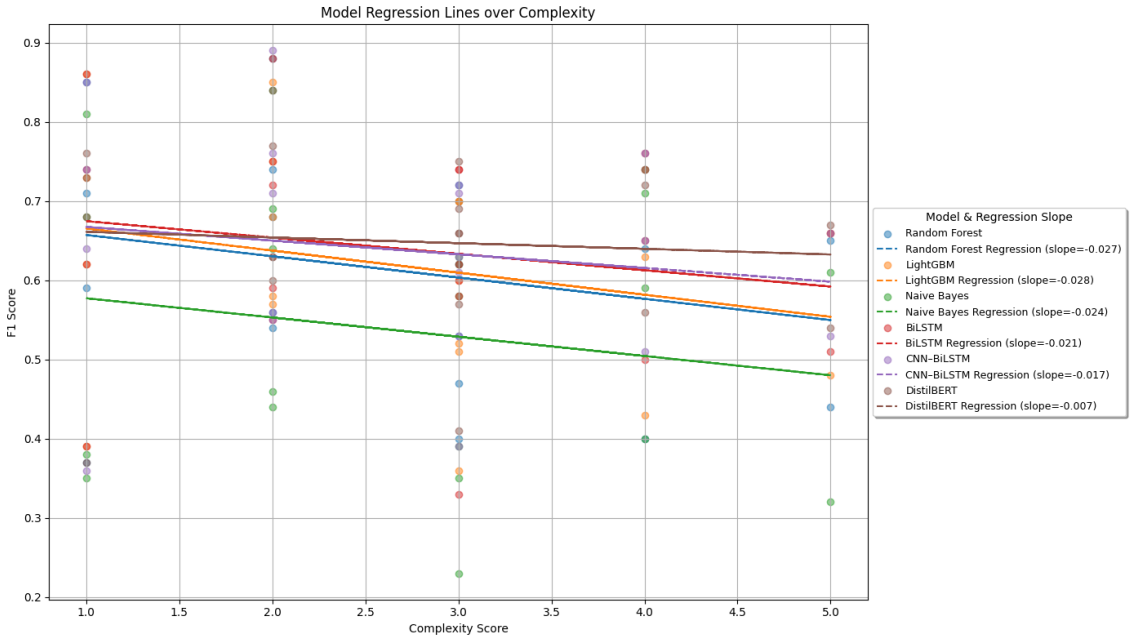
5.3 Complexity-wise F1-Score Performance

In order to better understand the impact of classification difficulty on prediction, we examined model sensitivity to label complexity. A complexity score between 1 (least complex) and 5 (most complex) was assigned for each of the repair codes based on the variability and ambiguity in the descriptions of complaints that were being fixed.

Figure 5.3 presents regression lines fitted to the per-class F1-scores against their respective complexity scores, providing a comparative view of how each model's performance degraded as classification difficulty increased.

Considering all six model architectures, a negative relationship can be observed between complexity and F1-score, which indicates that the more complex repair codes were systematically more difficult to predict correctly.Significantly, the conventional machine learning methodologies (Random Forest and Naive Bayes) show the largest F1-score reductions with regression slopes of around –0.027 and –0.024, respectively. This finding suggests that classical approaches are significantly more sensitive to increases in class complexity, likely due to their limited capacity to generalize nuanced semantic patterns

Figure 5.3: *Model-wise Regression of F1-Score Against Class Complexity*



inherent in the complaint data.

In contrast, deep learning models,particularly transformer-based architectures such as DistilBERT (slope $\approx$ -0.007), demonstrated comparatively higher resilience to increasing complexity. The flatter regression line of DistilBERT indicates that its performance remained relatively stable across both simple and complex classes. CNN–BiLSTM and BiLSTM also exhibited moderate robustness, outperforming traditional models but still showing performance deterioration in more challenging categories.

Within the LLM-based methods, a substantial distinction was observed between LLM-Basic and LLM-Rich preprocessing. At all the complexity tiers, LLM-Rich led to an increase in F1-scores and remarkably it resulted in flatter regression slopes compared to others. This is likely because we included solution description data, giving a model richer semantics to disambiguate vague or generic complaint narratives.

Interestingly, in low-complexity categories such as *Deterjan Çekmecesi* and *Kapı*, rule-based preprocessing occasionally outperformed LLM-Basic. These classes often

involve formulaic language patterns or surface-level faults that benefit from targeted lexical normalization. However, this advantage diminished as complexity increased, and rule-based approaches proved insufficient for capturing deeper semantic relationships in more ambiguous complaint data.

5.4 Preprocessing Contribution to Complexity-Weighted Model Performance

To further support our complexity sensitivity study, we investigated how diverse preprocessing options affect the model's performance when class F1-scores are weighted by class complexity. Indeed, we had previously labeled each repair department code with a complexity score that ranged from 1 (simple) to 5 (very complex), and the squared average of the weighted F1 score average, which favors performance in the more challenging categories, was computed.

The use of squared weighting ($F1_i \times c_i^2$) serves to disproportionately emphasize the influence of higher-complexity classes in the overall evaluation. This approach reflects the practical significance of correctly predicting semantically difficult or rare fault types, which are often more costly or impactful in real-world service contexts. Compared to linear weighting, squaring the complexity score magnifies the penalty for poor performance on complex classes, thereby rewarding models that demonstrate robust generalization beyond simple, formulaic complaint patterns.

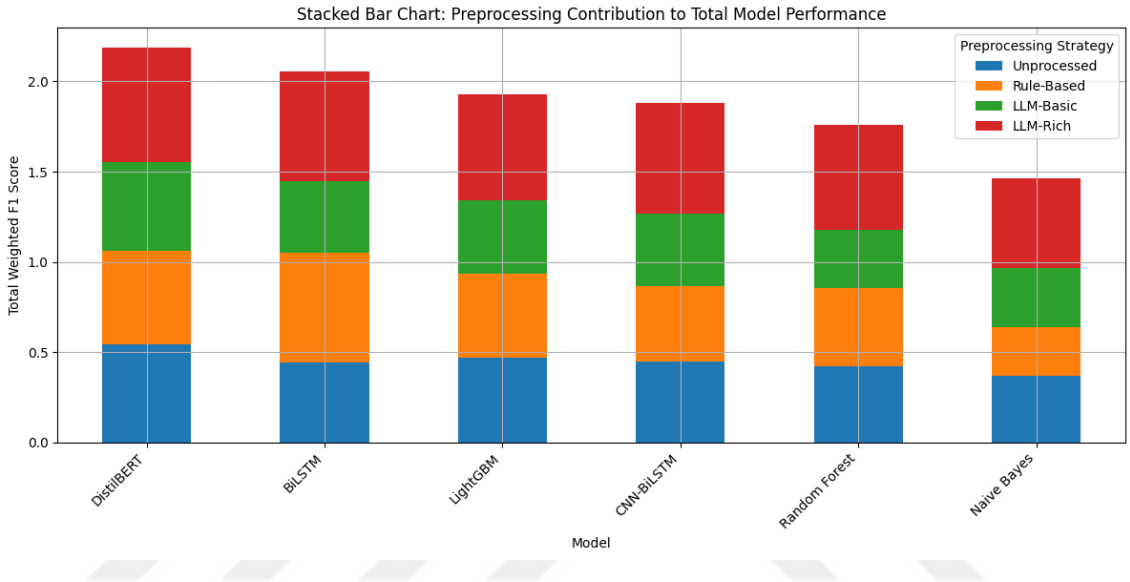
Table 5.1 summarizes the complexity-weighted average F1 scores achieved by each model under different preprocessing strategies.

Table 5.1: *Complexity-Weighted Average F1 Scores by Model and Preprocessing Strategy*

Preprocessing	Naive Bayes	Random Forest	LightGBM	BiLSTM	CNN-BiLSTM	DistilBERT
Unprocessed	0.367	0.424	0.468	0.443	0.450	0.541
Rule-Based	0.274	0.432	0.469	0.610	0.418	0.522
LLM-Basic	0.323	0.323	0.406	0.395	0.401	0.490
LLM-Rich	0.501	0.582	0.587	0.605	0.610	0.634

Figure 5.4 displays a stacked bar chart where each model's total performance is broken down by preprocessing strategy. The height of each bar represents the cumulative complexity-weighted F1 score, while the colored sections indicate the contribution of every preprocessing type: Unprocessed, Rule-Based, LLM-Basic, and LLM-Rich.

Figure 5.4: *Stacked Chart of Complexity-weighted F1 Score for Each Preprocessing*



The results demonstrate a clear advantage for LLM-Rich preprocessing across all model architectures. DistilBERT achieved the highest total complexity-weighted score, followed closely by BiLSTM and CNN–BiLSTM. In these models, the LLM-Rich segment consistently dominates, indicating that incorporating rich semantic context (e.g., inferred complaint details and structured symptom descriptions) substantially improves performance on complex cases.

BiLSTM showed unexpectedly strong results when combined with Rule-Based preprocessing, with complexity-weighted F1 score being one of the highest in all conditions. This can be explained by the complementary nature of rule-based normalization rules and sequential neural architectures: while BiLSTM excels at modeling temporal dependencies in token sequences, it can be sensitive to surface-level noise, redundancy, or inconsistent

phrasing—issues commonly present in unprocessed complaint text. Rule-Based preprocessing alleviates these difficulties by embracing a common language and minimizing linguistic variation so as to allow the BiLSTM to more specifically target the semantic structure.

In contrast, the CNN–BiLSTM architecture did not show a comparable gain under the same preprocessing strategy.This difference can be due in part to the ability of the convolution layer to extract local n-gram features, filtering out unnecessary patterns in early layers and then acting as an implicit noice reduction. As a result, the CNN component may already handle some of the issues that Rule-Based preprocessing aims to address, diminishing its added value. Furthermore, the hybrid nature of CNN–BiLSTM may dilute the influence of cleaned input by spreading representational focus across both local and sequential features, making it less reliant on strictly normalized text.

Traditional models Naive Bayes and Random Forest heavily depend on strategies like rule-based or unprocessed text, resulting on a lower overall score. Their limited capacity to benefit from richer inputs likely originates from constraints in modeling long-range dependencies and semantic ambiguity.

On the other hand, LLM-Basic preprocessing, although superior both to raw and rule-based inputs, did not achieve gains comparable to those of LLM-Rich, reinforcing the observation that shallow transformations applied at text structure level are not enough by themselves. The performance difference can mainly be attributed to the existence of solution description data in the LLM-Rich, which is essential for the repair-oriented context and not available in LLM-Basic. This added contextualization—such as inferred symptoms, structured resolution details, and paraphrased restatements—appears essential for enabling models to generalize effectively to rare or semantically subtle complaint categories, particularly in high-complexity classes.

Overall, this stacked breakdown underscores the importance of both model architecture and input representation in achieving high performance, initially if the goal is to make accurate predictions for semantically challenging service cases.

5.5 Summary and Implications

The experimental results indicate that either one of the preprocessing method and model structure has significant effect on the predictive capacity of the repair code classification model. Among the evaluated approaches, pipelines incorporating LLM-based preprocessing consistently outperformed simpler text representations, with LLM-Rich variants providing the highest resilience and predictive accuracy due to their access to resolution-level contextual detail.

Moreover, transfer-learning based models like DistilBERT showed better robustness in increasing complexity of number of classes maintaining F1-scores higher even in the difficult categories. This observation also motivates the need to select models that are well-equipped to learn deep-seated semantics when processing noisy and/or ambiguous natural language input.

Notably, while rule-based preprocessing occasionally surpassed LLM-Basic in very low-complexity classes—owing to its strength in leveraging repetitive complaint structures—it failed to generalize in more semantically diverse scenarios. This finding supports the notion that for scalability and generalization, semantic enrichment—particularly with consideration of operational context—is critically important.

Taken together, the evidence strongly supports prioritizing transformer architectures and advanced, context-aware preprocessing pipelines for deployment in operational repair code classification settings. Hybrid approaches that combine rule-based filters with LLM-based embeddings can be investigated in future work to tailor performance while trading interpretability and resource cost.

6. CONCLUSIONS

Natural Language Processing (NLP) techniques for classifying repair department codes based on Turkish customer complaints in home appliances domain were studied in this thesis. The study sought to increase the automation and accuracy of service ticket routing by using both traditional machine learning models and deep learning models with advanced text preprocessing techniques.

Initial analysis has shown some real world challenges such as very high percentage of brief or no description of the complaint, linguistic indeterminacy and label imbalance. To tackle these challenge, a multi-level preprocessing architecture was suggested, which combined rule-based normalization and Large Language Model (LLM)-based semantic structuring. Two LLM-based strategies were explored: LLM-Basic, which extracted structured information from standard complaint fields, and LLM-Rich, which additionally incorporated technician-written resolution summaries (solution_description). This enriched context allowed models to better understand vague or incomplete complaints by leveraging downstream information available in operational records.

Results of these experiments validated that preprocessing with LLM-Rich performed considerably better than all other strategies,including LLM-Basic, across nearly all repair code. Importantly, LLM-Rich did not rely on artificially injected data; rather, it utilized fields that are routinely collected in real-world service logs, supporting its practical deployability in similar enterprise environments.

Moreover, while rule-based preprocessing showed limited gains overall, it proved surprisingly effective in certain low-complexity categories where complaint phrasing was formulaic and predictable. This suggests that hybrid approaches may still have value when adapted to the structural characteristics of specific domains.

Across all models, a clear negative correlation was observed between repair code

complexity and F1-score. However, this degradation was less severe for transformer-based architectures such as DistilBERT, and for semantically enriched inputs generated by the LLM-Rich pipeline. These models demonstrated superior robustness in handling ambiguous or inconsistent complaint narratives.

To conclude, we have the evidence that the semantic context plays a crucial role in predictive maintenance scenarios. Architectures like DistilBERT, coupled with augmented LLM-based preprocessing, provide a strong basis for effective, yet scalable repair code prediction. Subsequent research can consider extension of these methods to other (multi-lingual, cross-domain) settings, and look at methods to generalize semantic enrichment without leveraging resolution text in scenarios where this is not available.

REFERENCES

[1] J. Pavlopoulos, A. Romell, J. Curman, O. Steinert, T. Lindgren, M. Borg, and K. Randl, "Automotive fault nowcasting with machine learning and natural language processing," *Machine Learning*, vol. 113, pp. 843–861, 2024.

[2] O. T. Yıldız, B. Avar, and G. Ercan, "An open, extendible, and fast Turkish morphological analyzer," in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)* (R. Mitkov and G. Angelova, eds.), (Varna, Bulgaria), pp. 1364–1372, INCOMA Ltd., Sept. 2019.

[3] L. Vidyaratne, X. Y. Lee, A. Kumar, T. Watanabe, A. K. Farahat, and C. Gupta, "Generating troubleshooting trees for industrial equipment using large language models (llm)," in *Proceedings of the IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 116–125, 2024.

[4] S. Munzir, D. B. Hier, C. Oommen, and M. D. Carrithers, "A large language model outperforms other computational approaches to the high-throughput phenotyping of physician notes," 2024.

[5] M. Khodadadi, F. Ahmadi, and M. Yadegari, "Automated routing of vehicle service requests using bilstm-cnn architectures," in *Proceedings of the 2021 International Conference on Industrial Informatics*, p. 122–129, 2021.

[6] R. Jain, A. Kumar, and R. Sharma, "A natural language processing and deep learning based model for automated vehicle diagnostics," *Procedia Computer Science*, vol. 200, pp. 123–134, 2023.

[7] M. Y. Eltabakh, X. Zhang, and X. Tang, "Retclean: Retrieval-augmented llms for intelligent data cleaning," *arXiv preprint arXiv:2401.05487*, 2024.

[8] P. Berkhin, "A survey of clustering data mining techniques," *Grouping multidimensional data*, pp. 25–71, 2006.

[9] H. Yu, F. Xiong, and Z. Chen, "Text classification based on natural language processing and machine learning in multi label corpus," *ACM Transactions on Asian and Low-Resource Language Information Processing*, 08 2023.

[10] Z. Li, X. Wang, W. Yang, J. Wu, Z. Zhang, Z. Liu, M. Sun, H. Zhang, and S. Liu, "Deepnlpvis: Visualizing and explaining deep nlp models," *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[11] M. Chen, X. Liu, and W. Sun, "Semla: Visual analytics for exploring language models via semantic layers," *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[12] F. Khaiser, A. Saad, and C. Mason, "Sentiment analysis of students' feedback on institutional facilities using text-based classification and natural language processing (nlp)," *International Journal of Language Communication Disorders*, vol. 10, pp. 101 – 111, 03 2023.

[13] D. Bora *et al.*, "Decision support system for project revision detection using nlp," in *IEEE International Conference on Computational Intelligence and Computing Research*, pp. 423–429, 2021.

[14] E. Bakü and S. Yılmaz, "Analysis of online reviews using text mining techniques: A case study from e-commerce," *Turkish Journal of Computer and Mathematics Education*, vol. 12, no. 4, pp. 785–794, 2021.

[15] W. Sun, J. Liu, and Z. Li, "Entity recognition and fault symptom extraction from home appliance service logs using sequence labeling models," *Applied Artificial Intelligence*, vol. 37, no. 2, pp. 123–138, 2023.

[16] Q. Li, Y. Chen, and M. Wang, "Multilingual fault diagnosis in industrial equipment using transformer-based embeddings," in *Proceedings of the 25th International Conference on Industrial Informatics*, pp. 221–229, 2022.

[17] OpenAI, "Chatgpt api." `https://platform.openai.com`, 2024. Accessed: July 2025.

[18] K. Oflazer, "Two-level description of turkish morphology," in *Literary and Linguistic Computing*, vol. 9, pp. 137–148, Oxford University Press, 1994.

[19] H. Sak, T. Güngör, and M. Saraçlar, "Turkish language resources: Morphological parser, morphological disambiguator and web corpus," in *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC)*, (Marrakech, Morocco), pp. 1079–1082, European Language Resources Association (ELRA), 2008.

[20] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[21] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[22] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A systematic survey of prompt engineering in large language models: Techniques and applications," 2025.

[23] F. Gilardi, T. Gessler, and M. Kubli, "Chatgpt out of the box: How to use large language models for text classification?," *Political Analysis*, vol. 31, no. 3, pp. 464–479, 2023.

[24] B. Liang, Z. Zhang, L. Ma, M. Yang, *et al.*, "Taskmatrix.ai: Completing tasks by connecting foundation models with millions of apis," *arXiv preprint arXiv:2303.16434*, 2023.

[25] F. Dernoncourt, J. Y. Lee, and P. Szolovits, "Neural networks for joint sentence classification in medical paper abstracts," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Short Papers*, (Valencia, Spain), pp. 694–700, Association for Computational Linguistics, 2017.

[26] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, 2018.

[27] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[28] R. Caruana, N. Karampatziakis, and A. Yessenalina, "An empirical evaluation of supervised learning in high dimensions," in *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, (New York, NY, USA), p. 96–103, Association for Computing Machinery, 2008.

[29] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *AAAI-98 Workshop on Learning for Text Categorization*, Technical Report WS-98-05, pp. 41–48, AAAI Press, 1998.

[30] H. Zhang, "The optimality of naive bayes," in *Proceedings of the 17th International FLAIRS Conference*, pp. 562–567, 2004.

[31] J. D. M. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the poor assumptions of naive bayes text classifiers," in *Proceedings of the 20th International Conference on Machine Learning (ICML)*, (Washington, DC), pp. 616–623, 2003.

[32] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002.

[33] P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine Learning*, vol. 29, no. 2-3, pp. 103–130, 1997.

[34] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, pp. 3146–3154, 2017.

[35] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[37] J. Yoon and H. Kim, "Multi-channel lexicon integrated cnn–bilstm models for sentiment analysis," in *Proceedings of the 29th Conference on Computational Linguistics and Speech Processing (ROCLING)*, pp. 244–253, 2017.

[38] M. Rhanoui, S. Ahmam, A. Khoumsi, and H. Alami, "A cnn–bilstm model for document-level sentiment analysis," *Information*, vol. 10, no. 9, p. 276, 2019.

[39] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," 2019.