DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

EVALUATING THE SUCCESSFULNESS OF STARTUP BUSINESSES BY DATA MINING TECHNIQUES

by
Farid BAGHERI

October, 2022
İZMİR

EVALUATING THE SUCCESSFULNESS OF STARTUP BUSINESSES BY DATA MINING TECHNIQUES

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Industrial Engineering

by
Farid BAGHERI

October, 2022
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"EVALUATING THE SUCCESSFULNESS OF STARTUP BUSINESSES BY DATA MINING TECHNIQUES"** completed by **FARID BAGHERI** under supervision of **ASSOC. PROF. DR. DERYA EREN AKYOL** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

……..

Assoc. Prof. Dr. Derya EREN AKYOL

Supervisor

……..……..	……..……..……..
Assoc. Prof. Dr. Derya BİRANT	Prof. Dr. Pınar MIZRAK ÖZFIRAT
(Jury Member)	(Jury Member)

Prof. Dr. Okan FISTIKOĞLU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Assoc. Prof. Derya EREN AKYOL for guiding me in this research and encouraging me to strive to be my best, personally and professionally, by always telling what I needed to hear, no matter how bad it was. I would like to thank Prof. Adil BAYKASOĞLU for the practical advice and for teaching me most of what I know about industrial engineering. Also, I would like to thank to Asst. Prof. Seren ÖZMEHMET TAŞAN, for all the advice and kindness during my very stressful first year in Turkey. Most importantly, I am more than grateful to have the unconditional and unfailing support of my family and my wife Merve BAGHERI, who always trusted in my potential and taught me to have faith in God and that hard work pays off.

Farid BAGHERI

EVALUATING THE SUCCESSFULNESS OF STARTUP BUSINESSES BY DATA MINING TECHNIQUES

ABSTRACT

Estimating the success of startups has attracted a lot of interest in the literature. In recent years, a lot of research has been carried out by researchers about the factors which influence startup businesses' successfulness and most corporations focused on developing various types of prediction models to accurately anticipate the fate of new businesses. In this work, we try to develop a reliable model for predicting startup businesses' success. Previous studies have concentrated on the accuracy of various algorithms without exploring the true influence of risk and success variables, or on understanding the effects of factors such as entrepreneur education, funding techniques, and timing on startup success or failure. In this thesis, we employ ensemble learning which is a method of machine learning to predict the success of startup businesses by using selected subsets of Crunchbase big data set. The goal is to use other variables from the dataset to distinguish between successes and failures. We include several factors that influence the startup success, and compare various prediction approaches, such as Extreme Gradient Boosting, Gradient Boosting, AdaBoost and Random Forest. The final result might assist Angel Investors and Venture Capitalists in developing more consistent and quantifiable startup portfolios.

We randomly partition the data into two datasets throughout the estimate phase: training data and testing data. The objective is to use the benchmark approach (XGBoosting) to fit the model in the training data and then test the model's performance and the importance of the variables in the testing data.

Keywords: Data mining; machine learning; ensemble learning; startup business, extreme gradient boosting.

VERİ MADENCİLİĞİ TEKNİKLERİ İLE GİRİŞİM ŞİRKETLERİNİN BAŞARISININ DEĞERLENDİRİLMESİ

ÖZET

Başlangıç işletmelerin başarını tahmin etmek literatürde çok ilgi çeken bir konudur. Son yıllarda, araştırmacılar tarafından Başlangıç işletmelerin başarını etkileyen faktörler ile ilgili bir çok çalışma gerçekleştirildi ve bir çok şirket, yeni iş modellerini geleceğini görmek ve isabetli şekilde tahmin etmek için çeşitli tahminleme modelleri geliştirmeye odaklandı.Bu çalışmada bir Startup şirketinin başarısını güvenilir şekilde tahmin edebilen bir model geliştirilmiştir. Önceki çalışmalar risk ve başarı faktörlerine yada girişimcinin eğitim seviyesi, finansman metodları, zamanlamanın uygunluğu gibi kriterlere bakmadan, yalnızca kullanılan çeşitli algoritmaların doğruluğuna odaklanmıştır.

Bu tezde Cruchbase büyük veri kümesinden seçilen bir Başlangıç işletme şirketinin başarısını tahmin edebilmek için bir makine öğrenmesi metodu olan topluluk öğrenmesini kullanıldı. Amaç veri setindeki diğer değişkenleri kullanarak, başarı ve başarısızlık faktörlerini ayırt etmektir. Başlangıç işletme şirketlerinin başarısnı etkileyen birçok faktör dahil edilmiş ve Aşırı Gradyan Arama, Gradyan Arama, Adaptif Arama ve Rastgele Orman gibi tahminleme yaklaşımları ile karşılaştırılmıştır. Ortaya çıkan sonuçlar melek yatırımcılara ve risk sermayedarlarına daha nitelikli ve ölçülebilir Başlangıç işletme portfolyoları oluşturmakta yardımcı olabilecek niteliktedir.

Tahminleme aşamasında veri, rassal olarak eğitim ve test verisi olmak üzere ikiye bölünmüştür. Amaç, kıyaslana tekniğinin (XGBoosting) kullanarak tahminleme modelini eğitim verisine uydurmak ve daha sonra model performansını ve test verisindeki değişkenlerin önemini ölçmektir.

Anahtar kelimeler: Aşırı grandyan aratma, makine öğrenmesi, başlangıç işletme, topluluk öğrenmesi, veri madenciliği.

CONTENTS

M.Sc THESIS EXAMINATION RESULT FORM ii

ACKNOWLEDGEMENTS iii

ABSTRACT iv

ÖZET v

LIST OF FIGURES ix

LIST OF TABLES x

CHAPTER 1 - INTRODUCTION 1

1.1. Literature Review 2

1.2. Novel Contribution of This Thesis 2

1.2. Organization of The Thesis 3

CHAPTER 2 – MODEL FRAMEWORK AND THEORY 4

2.1. Model Framework 4

2.2 Ensemble Learning 7

2.2.1 Decision Trees 12

2.2.2 Random Forest 14

2.2.3 Adaptive Boosting 19

2.2.4 Gradient Boosting 21

2.2.5 Extreme Gradient Boosting 23

CHAPTER 3 – MODEL DEVELOPMENT DATA 32

3.1 Preparation of Data 33
3.2 Target Variable 34
3.3 Independent Variables 35
3.3.1 Candidate Pool 35
3.3.2 Transformations 36
3.4 Rejection Inference 39
3.5 Training and Testing Data 40

CHAPTER 4 – MODEL ESTIMATION AND RESULTS 41

4.1 Metrics for Model Performance 42
4.1.1 Somers' D 42
4.1.2 Accuracy 42
4.1.3 Confusion Matrix 43
4.1.4 ROC Curve - AUC Score 43
4.1.5 Cross Validation 44
4.2 Model Selection 45
4.2.1 Removing Non-predictive Variables 45
4.2.2 Data From Training And Testing 45
4.3 Rejection Inference Application 46
4.4 Check Logit Assumptions 47
4.4.1 Multicollinearity 47
4.4.2 Heteroskedasticity 48
4.4.3 Independent Observations 49

4.4.4 Linearity 49
4.4.5 Sample Size 49
4.5 Interpreting The Coefficients 50
4.5.1 Function 1: (Education + Time From Graduation)^2 50
4.5.2 Function 2: (Milestones + Time To First Milestone)^2 52
4.5.3 Percent of Industry's Older Enterprises 54
4.5.4 Located At The Silicon Valley 55
4.5.5 Reached Venture Capital Rounds 55
4.5.6 Number of Angel Investors x Angel Investment (In Millions) 56
4.5.7 Time to Raise Investment x Total Funding Rounds 56
4.5.8 Industries 57
4.6 Comparing Alternative Algorithms 57
4.6.1 Sensitivity Analysis 59
4.6.2 Feature Importance 60
4.7 Advantages Of Extreme Gradient Boosting 62

CHAPTER 5 - CONCLUSION 66
5.1 Summary of The Thesis 66
5.2 Future Research Directions 67

REFERENCES 68

APPENDICES 68

LIST OF FIGURES

Figure 2.1 Pseudocode of the research framework 5
Figure 2.2 Schematic representation of the research framework 6
Figure 2.3 Ensemble learning 8
Figure 2.4 An illustration of decision trees 12
Figure 2.5 The components of a decision tree 13
Figure 2.6 Random forest structure. 15
Figure 2.7 Gradient boosting procedure. 21
Figure 2.8 Inputs values of ensemble model. 26
Figure 2.9 Tree ensemble model. 27
Figure 2.10 The tree structure 31
Figure 3.1 Counts for 'successes' and 'failures' of the target variable 34
Figure 3.2 Percentage of successful companies per category 35
Figure 3.3 Counts for 'successes' and 'failures' of the target variable after the rejection inference method 40
Figure 4.1 ROC - source: sklearn – python 44
Figure 4.2 Effect of function 1 on odds of success 51
Figure 4.3 Effect of function 2 on odds of success 53
Figure 4.4 Confusion matrix 59
Figure 4.5 Sensitivity analysis; XGBoost: Extreme Gradient Boosting, RF: Random Forest, GB: Gradient boosting, AdaB: AdaBoost 60
Figure 4.6 Feature importance to the odds of startup success; Extreme Gradient Boosting and Gradient boosting 62

LIST OF TABLES

Table 3.1 Literature review of variables 38
Table 4.1 Confusion matrix - source: glass box artificial intelligence…..……43
Table 4.2 Variance Inflation Factor - all variables from the final model…..……48
Table 4.3 Output of Extreme Gradient Boosting…….…..……50
Table 4.4 Algorithms' performance.....58
Table 4.5 Predictability benchmark with other researches 64

CHAPTER 1
INTRODUCTION

Startups are businesses with great potential for growth and, in most cases, they have contemporary business plans and technologies that result in entirely changing the global business environment, human lifestyle, and the way people spend their time in recent decades. Technology allows businesses to offer clients products and services that are less expensive, faster, and more convenient. This advantage, which is primarily the result of economies of scale, lowers labour costs, improves accessibility to products and produces rapid momentum and large returns for investors. The possibility of high rewards, on the other hand, comes with a far higher risk. According to Startup Genome (Marmer et al., 2011), over 90% of startups fail, making investment decisions for angel investors and venture capitalists more unclear. Startup failure is caused by a variety of reasons, such as money running out, being in the wrong market, and lack of research. There are a variety of causes for this pattern of a large number of failures and a small number of enormous successes. The majority of them are mentioned in this thesis. One of the most persuasive explanations for this phenomenon is what is known as "the winner-takes-all economy".

Economists have traditionally characterized winner-take-all markets as situations where little variations in performance result in enormous inequalities in rewards. Star performers and sportsmen are examples of such marketplaces. An average member of the Screen Actors Guild or the Professional Golfers Association makes a lot less money than, say, Julia Roberts or Tiger Woods. However, historically, slight variations in performance have resulted in similarly modest variations in financial rewards across industries. But there are a lot of shifts happening right now. Over the recent decade, individual product lines such as Microsoft's Windows operating systems, Intel's microprocessors, and Nokia's mobile phones have all exhibited extraordinary performance. The spread of this phenomenon to new fields is well underway at present.

The margin between winners and underperformers is likely to grow in the future. Organizations that are not the best in their category, even if they have the insight or good fortune to operate in attractive business areas, are likely to face competitive difficulty in this new economic landscape (Hulme & Campbell, 2001).

1.1 Literature Review

Several studies have been conducted on the elements that influence startup success. Wong (2002) investigates the characteristics and differences in terms of support and influence between angel investors and venture capitalists on startup management and networking. Solomon et al. (2008) look at how education affects the success of entrepreneurs. Kenney and Von Burg (1999), using Silicon Valley as an example, look at how the characteristics of a region affect the companies in it.

There are other models that use different approaches in order to predict startup success. For example, Krishna et al. (2016) use financial aspects of startups as predictors, as well as an artificially created "severity score" to proxy entrepreneur characteristics as a proxy, and Logistic Regression and some machine learning algorithms as methods. Shah (2019) considers a variety of risk and success criteria, including the entrepreneur's education, the quantity and source of funding, the number of milestones reached, regional characteristics, and the business category. Lussier and Corman (1996) uses factors such as financial control, time, staffing, and others that aren't commonly used in other studies, as well as the Linear Discrimination analysis as the benchmark algorithm.

1.2 Novel Contribution of This Thesis

Our study differs from other studies in the literature. Firstly, it solely examines startups that are located in the United States. Second, our definition of a successful startup business is different from previous studies, by considering the success of a startup business not only through operating or Initial public offerings (IPOs) and acquisitions but also startups

with newsworthy milestones after a few years of operation. Lastly, it uses transformations and a selection bias corrector to make sure that the results are not only accurate and statistically significant but also in line with what economists know and what other research has shown.

1.3 Organization of The Thesis

This thesis outlines a step-by-step model development process for predicting company success using several types of algorithms. Chapter 2 discusses the model's desired outcome, common statistical and machine learning techniques, and ways of comparing their accuracies. The data and their sources are detailed in Chapter 3, including the modifications required with the reasons; the definition of success and failure and the logic behind it; the success and risk factors to be tested; and the method to define the training and testing data. To ensure improved accuracy and interpretation, Chapter 4 compares the predicted and estimated results; removes and modifies variables using the results of the benchmark method (Extreme Gradient Boosting); implements the rejection inference; and compares the accuracy of all the algorithms. The results are summarized in Chapter 5.

CHAPTER 2
MODEL FRAMEWORK AND THEORY

As stated by Huang (2016), a startup business is successful if it has accrued, IPOs, or has a net worth of more than $1B (unicorn). There are two primary exit paths for a profitable startup. The business can go through an Initial Public Offering (IPO) or it can be sold to an existing corporation via an acquisition. Under an IPO, the enterprise receives a stock market listing, which enables the business to access more finance for its projects and permits executives to eventually sell their shares to the public. If the startup is purchased, the insiders get instant cash in return for their shares (Guo et al., 2015).

According to previous studies, a successful business is either the one that is acquired by others or one that is listed as an IPO business. Researchers who have studied this area believe in the fact that a business that has the operating conditions in Crunchbase's dataset is successful (Shah, 2019; Krishna et al., 2016). In this thesis, a successful company is defined as a company that earns back invested money to investors consequently. Also, this study looks from an investor's perspective to consider what their motivation is to start a business.

In this thesis, we focus on 3 main issues to define a company as a successful one: 1- an Initial Public Offering (IPO), 2-merged or acquired, and 3-still operating and having successful results after four years.

2.1 Model Framework

The data used in this study gives information on the status of startups (operating, closed, IPO, or Mergers and acquisitions (M&A)). However, performing an analysis using the state of startups is not enough on its own. In addition, there should be some additional features in the dataset. Additional features such as milestones, financing rounds, funding

amounts, and entrepreneur education can reveal whether the business is a successful one or not. The projected values are divided into two categories: failure (0) and success (1).

During the estimation phase, we randomly separate the data into two data sets: training data (representing 70% of the observations) and testing data (representing 30% of the observations). The objective is to use the benchmark approach (Extreme Gradient Boosting) to train the model with training data and then assess the model's performance and the importance of the variables with the testing data. This study compares the performance of approaches such as Random Forest, Extreme Gradient Boosting, Gradient Boosting, and AdaBoost. Figure 2.1 and 2.2 respectively present Pseudocode and Schematic representation of the research framework, these figures summarize all steps of our model for predicting the successfulness of startup businesses. To perform the methods and compare the results we use Python coding language and its library Kernel.

fit the initial variables into the bench mark model (Extreme gradient boosting) and test its performance
choose the best and suitable variables according the analysis of results
assign selected variables as features of startups(input) to run Ensemble Learning Algorithms
check each Ensemble Learning Algorithm's performance metrics
if the Method has better performance than others opt it,
else
while finding the optimum Algorithm
select Best Algorithm which has higher performance metrics
end

Figure 2.1 Pseudocode of the research framework

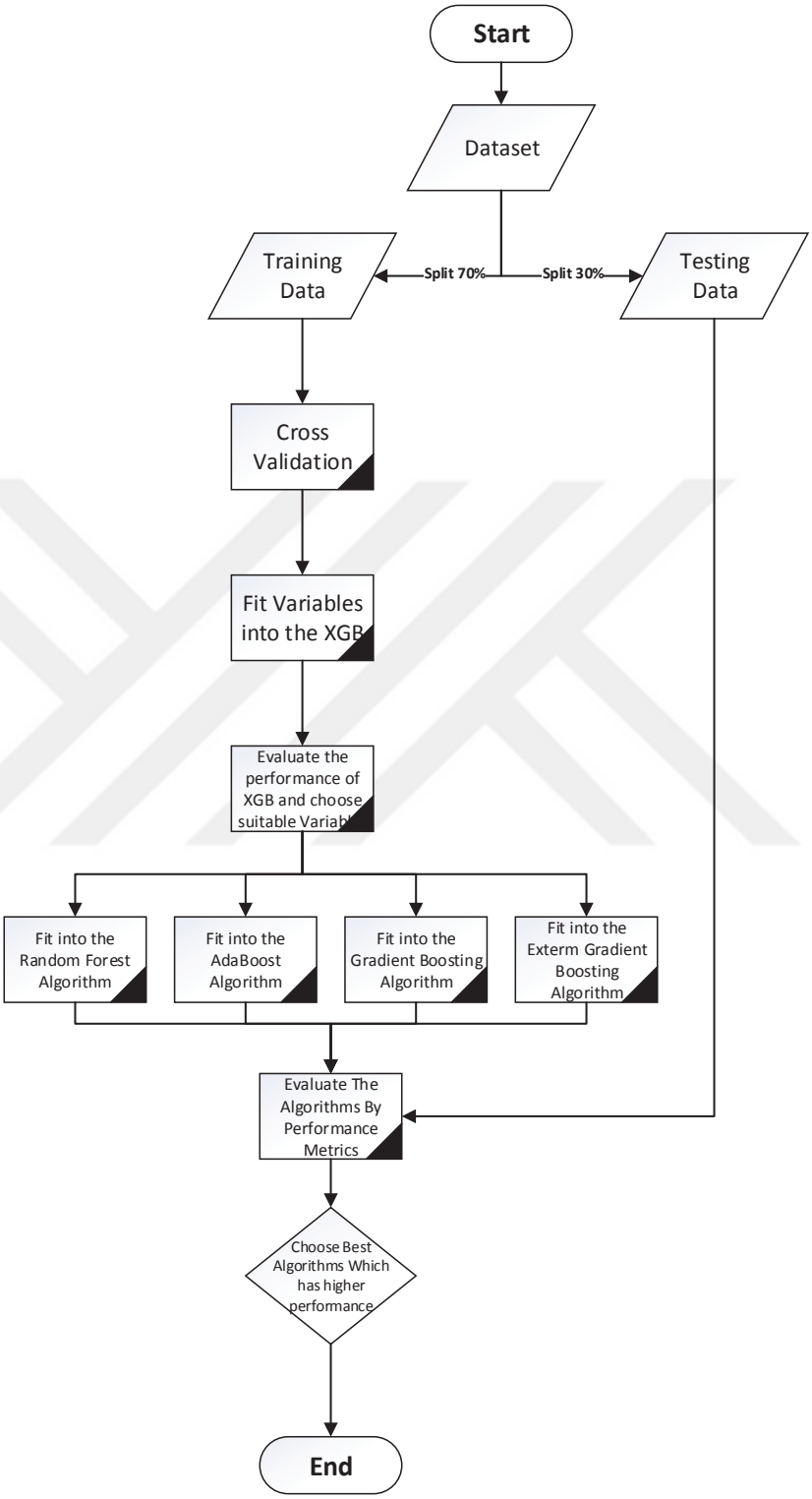

Figure 2.2 Schematic representation of the research framework

2.2 Ensemble Learning

Ensemble learning plays an important role in statistics and machine learning. It is an undeniable fact that ensemble learning methods employ multiple methods and their logic is to obtain better performance than when used individually (Wolpert, 1992). Ensemble learning has lots of advantages, which have been shown by previous studies. According to Hansen and Salamon (1990), the combination of some weak algorithms is much better than the best algorithm (Hansen & Salamon, 1990). A simplified illustration of Ensemble Learning and its architects is shown in Figure 2.3 in which a graph of noise level and error of average, best single and combination models are represented. According to this graph error level of combination model is lower than other models which means that Ensemble is often better than the best single.

It has been demonstrated that ensembles of predictors provide more accurate classification and regression results than a single predictor. In recent years, numerous alternative algorithms have been introduced and evaluated. However, our current understanding is primarily empirical, with little theory to explain why and how the generalisation error is so much lower for the majority of data sets.

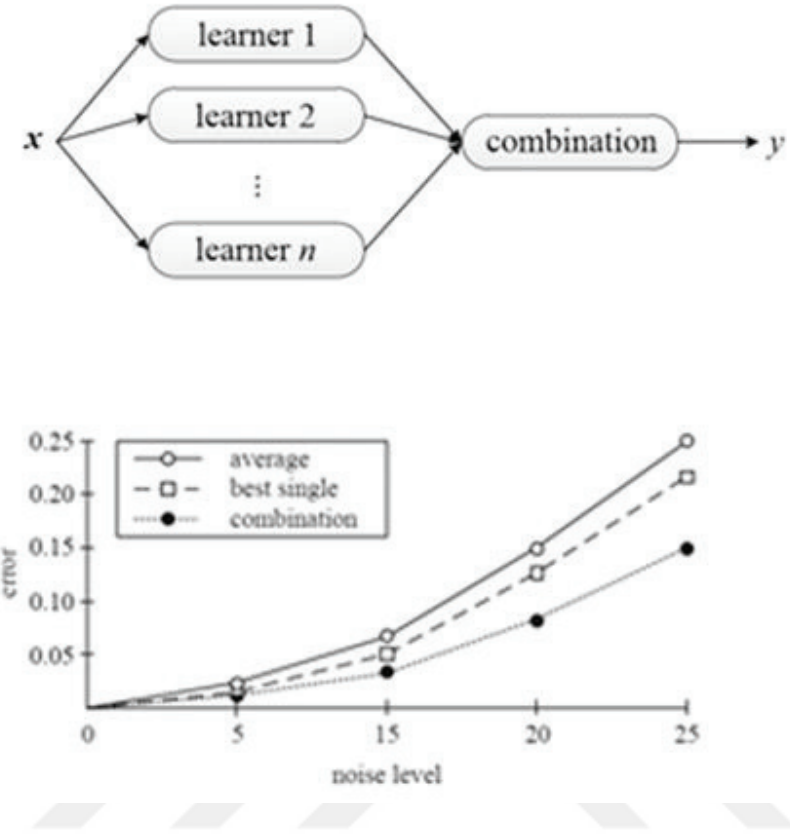

Figure 2.3 Ensemble learning (Zhou, 2012)

Three strategies must be selected in order to construct an effective ensemble system. Previously, we have mentioned to the following as the three pillars of ensemble systems: "(1) data sampling and selection, (2) training member classifiers, and (3) combining classifiers".

Data Sampling and Selection: In ensemble-based systems, making distinct errors on any given sample is of supreme importance. Ultimately, if all ensemble members produce identical results, there is no advantage to combining them. Therefore, ensemble members' decisions must be diverse, particularly when they commit an error. It is well-established that ensemble systems benefit from diversity. Classifier outputs must ideally be independent or, even better, negatively corresponded.

Numerous techniques can be used to generate diverse ensembles, with the most prevalent being the use of distinct subsets of the training data. Different sampling techniques result in distinct ensemble algorithms. The essence of boosting algorithms is sampling from a distribution that favours misclassified data from the past. Various subsets of the given features may be utilized each classifier to be trained, leading to random subspace approaches. Other less used methods include utilizing alternative splitter of the basis classifier (e.g., Creating a collection of MLPs (Multilayer perceptrons)), where each MLP has a different-sized hidden layer) or employing different base classifiers as ensemble members. Diverse definitions exist for diversity measurements. Although it is obvious that diversity is crucial and that a lack of diversity leads to poor group performance, a direct association between diversity and group accuracy has not yet been proven.

Training Member Classifiers: The individual member training strategy is the core of every ensemble-based system. The most common ensemble classifier training algorithms are bagging (and related algorithms such as arc-x4 and random forests), boosting (and its many variants), stack generalisation, and hierarchical Mixture of experts (hierarchical MoE).

Combining Ensemble Members: The final step in any ensemble-based system is the mechanism that combines the individual classifiers. The strategy for this step is partially determined by the classification algorithms employed as ensemble members. Support vector machines are one type of classifier that solely produces discrete-valued labels (Simple or weighted) majority voting is the most common combination rule for such classifiers, followed by the Borda count in distant second place. Other classifiers, such as multilayer perceptrons and Bayes classifiers, produce class-specific outputs with continuous values that are interpreted as the classifier's support for each class. These classifiers provide a larger range of options to voting-based techniques, including arithmetic (sum, product, mean, etc.) combiners and more intricate decision templates.

Before they can be utilized, complex combination algorithms (such as those utilized in stacked generalisation or hierarchical MoE) may require an additional training step.

There are three critical reasons (statistical, computational, and representational) to use ensemble learning methods (Hastie et al., 2009). In addition, bias-variance decomposition (Geman et al., 1992), margin theory (Breiman, 1999), and strength-correlation (Schapire, 1990) also explain why ensemble learning methods are so popular. It is a well-known method among researchers in many fields, such as artificial intelligence and statistics, because it works well in environments with distributed or parallel computing and makes predictions more accurate (Raudys & Roli, 2003).

From a statistical point of view, we try to find the best hypotheses from a hypothesis space (H) for data fitting. However, if we have limited training data in comparison to the size of the hypotheses space, it is possible to have statistical problems. The lack of training data causes many different hypotheses to fail, which might end up with the wrong classifier. However, one of the advantages of ensemble learning is that we can average the votes of classifiers and reduce the risk of wrong classification. If we look at the ensemble learning methods from a computational lens, the main problem with many learning algorithms is that they suffer from local optima. But rather, in ensemble learning methods, the local searchers are leveraged at different points to identify a stronger approximation, and it is possible to overcome the local optima.

Another reason to use ensemble learning methods is representational. There is a likelihood that true function might not be well represented by any of the hypotheses in space *H* and it results in expanding the space of representational space. On the other hand, ensemble learning methods have representation ability and strength by developing weighted sums of hypotheses which are rooted in *H*. As a result, lots of researchers in recent years have used ensemble methods to improve single models' weaknesses in different areas. There will be a review of recent studies in ensemble learning in 2.2.2, 2.2.3 and 2.2.5 sections.

“Ensemble learning (Zhi-Hua, 2012)” or “Multiple classifier systems (Xu et al., 1992)” has two steps as Perturbation and Combine. In the perturbation step, the learning algorithm (homogeneous or heterogeneous) is given a perturbed training dataset, and then in the combine step, the output and the result of classifiers are merged.

One of the main reasons that ensemble learning methods are so famous and popular is because of boosting and bagging (Zhi-Hua, 2012). In the bagging and booting aggregation methods, we try to train independent classifiers by using a set of instances that are taken from a training dataset with replacement. The tagging method is useful for models that are weak and unstable, where a small perturbation in the training dataset might cause a critical variation in their performance. Using the bagging method decreases the variation in those methods’ performance and improves their generalisation ability. In training of the classifiers (Zhi-Hua, 2012) boosting must be treated in a sequential manner with the current classifier by focusing on those samples that cannot be classified correctly by previous ensemble learning methods. In each iteration, the distribution of the training set is updated and those difficult samples that were not classified correctly in the previous ensemble learning method are changed to easy ones.

According to Beriman (2001), a random forest’s main structure is merging bagging and random subspace. On the other hand, Most aspects of random forest and AdaBoost are similar in most parts. The only difference between AdaBoost and Random Forest is that AdaBoost is sequential and Random Forest is parallel. This means that, in a random forest, the individual models or decision trees are constructed independently and concurrently from the primary data. In the random forest, multiple trees are constructed in parallel from the same data, and none of the trees are dependent on other trees. Consequently, this process is known as a "parallel process." In the sequential process, however, one tree is dependent on the previous tree.

Lin & Jeon (2006) propose that Random Forest is another form of the nearest neighbour. Random forest has also been used in high-dimensional and ill-posed problems (Couronné et al., 2018). Fernández-Delgado et al. (2014) evaluated the performance of the random forest with 179 classifiers by using data from the University of California Irvine's datasets. This method has caught a lot of researchers' interest, and they are trying to solve real-life problems by this method.

2.2.1 Decision Trees

Decision tree was first founded by Quinlan (1986) and is a prediction method that links data to classification labels. Diagrammatic depiction of a decision tree is shown in Figure 2.4 in this figure the leaves of decision trees contain category labels, whereas the branches reflect feature component pairings that result in a forecast. (Kingsford & Salzberg, 2008).

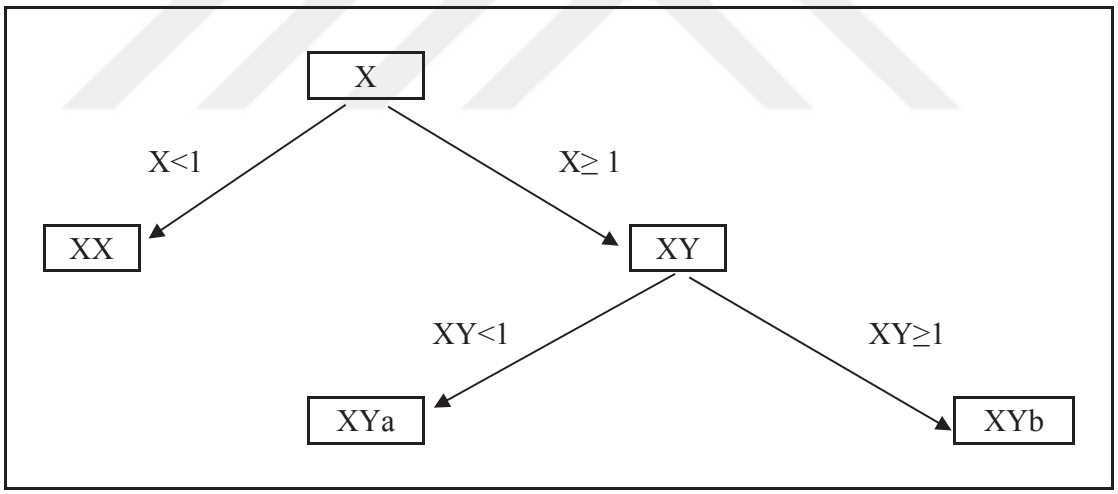

Figure 2.4 An Illustration of decision trees

In this instance, branch nodes contain decision statements, and the tree can make predictions based on X predictors. Starting at the root node, the prediction method examines the decision statement. The right branch is followed if the decision statement evaluates to true; otherwise, the left branch is used. Up until a leaf node is reached, the

procedure keeps evaluating the nodes' decision statements. The leaf node presents the choice. (De Ville, 2013).

The technique for constructing a decision tree relies on the splitting rule. At each split of the decision tree, training samples are divided into smaller partitions, and the dividing rule controls this procedure. The parameter used to determine whether separation on the data should proceed is the greatest homogeneity of a tree node. (Rokach & Maimon, 2005).

Components of a single DT are shown in figure 2.5 that constitutes of an attribute tree, in which each node represents a potential attribute linked with an event, with the root node representing the most crucial piece of knowledge gained, the branches representing attribute values, and the leaves representing classes. In the process of making a tree, choices are made about which variables and split values to use within a node, whether to continue splitting or not, and which classes to assign to the resulting leaf nodes. (Sunar et al., 2017; Sa et al., 2011; Horning, 2013)

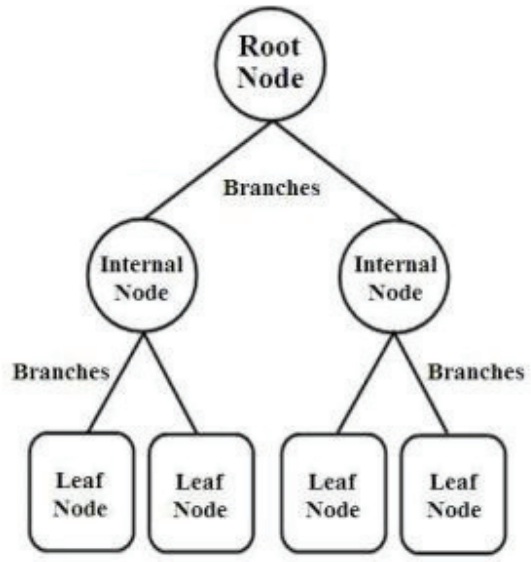

Figure 2.5 The components of a decision tree (Sa et al., 2011)

2.2.2 Random Forest

A Random Forest (RF) is a collection of Decision Trees (DT). Contrary to Linear Regression or Support Vector Machines (albeit in a highly multidimensional space), Random Forest does not require linear characteristics. In its simplest form, it may be conceived as utilizing bagging on a multiple tree classifier (Breiman, 2001).

In machine learning, Random Forest is frequently used for both classification and regression. Decision trees are the building blocks of a Random Forest. Several decision trees are built during the training phase. Each possible tree solution in a classification issue represents a unique class. The top-voted category across all trees is what the Random Forest algorithm returns as a result. The accuracy of random forests is lower than that of gradient boosted trees, but they generally exceed decision trees. However, their efficiency may be compromised by poor data quality (Belgiu & Dragut, 2016).

By combining the predictions of numerous trees, RF is a supervised machine learning classification and regression approach that yields better results than a single tree. Supervised classification requires the input data classes to be defined and labelled before being passed to the algorithm as part of the training data set. For this reason, the classification algorithm can employ pre-defined classes to learn the characteristics of undefined classes from training data. (Sunar et al., 2017; Jia, 2017; Belgiu & Dragut, 2016).

Despite the fact that RF is composed of ensemble DTs, it is distinct from DT due to the randomly occurring processes of locating the root node and splitting internal nodes. In addition, RF can overcome the disadvantages of a single DT while retaining its benefits. The values of the input variables are used to construct each tree in the RF algorithm via supervised learning. This is done to estimate the output variable's value. (Horning, 2010; Louppe, 2014; Pal, 2005).

RF has numerous benefits, including non-parametricity and the ability to handle outliers in training data. Additionally, RF prevents overfitting if a sufficient number of trees are discovered. RF automatically identifies the best predictors of a large number of data points, eliminating the need for variable selection or data reduction prior to algorithm application. Additionally, no pre-processing is necessary for the input dataset.By allowing RF to automatically deal with missing data, randomness in ensemble trees increases its reliability. The design of RF allows for the estimation of both classification error and the relative relevance of individual variables. RF is less sensitive to the parameters used than other machine learning algorithms like Support Vector Machine, but it is advantageously easier to establish which parameters to use. An RF model with many DTs is shown structurally in Figure 2.6, according to this RF model, training data is divided into *n* data samples and each of these data samples is used to construct separate decision trees, at the end predictions of these decision trees are averaged to build a RF. (Horning, 2010; Breiman, 2001).

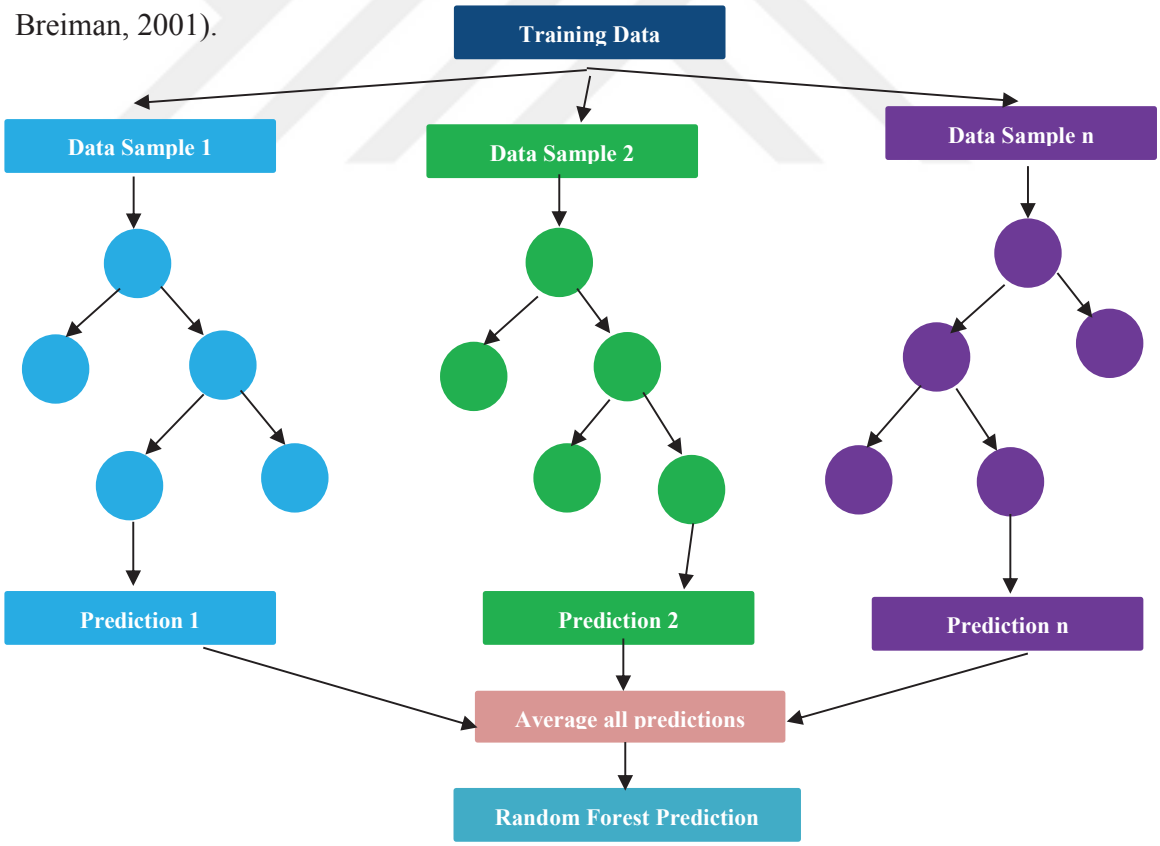

Figure 2.6 Random forest structure (Sa et al., 2011)

Since RF is an ensemble approach that considers all trees' outputs before making any judgments, increasing the number of trees tends to improve estimation accuracy. In order to identify which category a given pixel or object belongs to, RF classification typically takes into account the results from hundreds of separate DTs. Classes predicted by each tree are compared, and the most common one is used to label an image's pixels or objects. For instance, in a 500-tree pixel-based RF classification, if 350 trees think a given pixel is a tomato, 80 trees think it's a pepper, and 20 trees think it's a watermelon, then the pixel is categorised as a tomato (Horning, 2010; Breiman, 2001).

There are two phases involved in the production of trees. First, RF chooses training data at random to build the models of each DT. That is, variables from the input dataset are picked at random for each tree. The algorithm uses out-of-bag samples, which make up one-third of the total training data, to evaluate the quality of the model. The second phase involves randomly selecting the condition to exit each node of the trees in order to construct the binary rule. The user has the option of setting the random number of picks or letting the algorithm choose. The error rate and the correlation between trees are reduced. Most crucial to this process is settling on a sufficient number of variables to provide both low correlation and strong predictive ability (Horning, 2010; Breiman, 2001).

The RF algorithm contains the following parameters (Horning, 2010; Breiman, 2001).

1. Training information for prediction variables (such as image bands) and response variables (like land cover type).
2. The total number of trees
3. The number of prediction variables for each binary rule of each split/decision,
4. Parameters for error calculation and variable significance

Classification outcomes are strongly influenced by certain variables while being somewhat unaffected by others. As a result, RF is effective at identifying these factors, or

the relevance of the variables. As a result, it is possible to rerun the model without the variables that don't add much to the analysis the first time around. The classification of multi-band satellite images relies heavily on this detection because training sets can potentially include spurious information. As a bonus, RF may also assess data outliers, or instances that deviate significantly from the norm in the training set. This allows for the revision of the training set by locating classes that include outliers (Horning, 2010).

The "Strong Law of Large Numbers" states that as the number of trees in a random forest rises, the generalization error tends to converge to a limit (Breiman, 2001) Due to the fact that random forest's generalization accuracy does not drop on average when additional classifiers are added to the ensemble, this shows that the ensemble size is not a hyper-parameter that needs adjusting. It is more likely that the ensemble has converged to its asymptotic generalization error if there are more trees in the forest. One of random forest's main benefits is that it requires relatively few hyper-parameters, or at least that the default hyper-parameter setting provides excellent average performance. (Fernández-Delgado et al., 2014).

In any case, additional RF-adjustable hyper-parameters control the decision trees' depth. Typically, in a random forest, decision trees are grown until every leaf is pure. However, this may produce very tall trees as a result. In these circumstances, a maximum depth or a minimum number of instances per node before or after the split can be established to control the growth of the tree.

However, since it is not possible to construct multiple trees on the same dataset (because of getting the same results), two types of unpredictability are introduced:

1- Each tree is constructed with slightly distinct rows, sampled from the parent tree using iterations (bagging), and
2- Each tree (or in certain situations, each branch decision) is constructed using a subset of columns chosen at random.

The objective of RF is to prevent overfitting, and it achieves this by producing random subsets of features and building smaller (shallow) trees using the subsets.

According to Liaw and Wiener (2002), the following stages outline the method:

(1) From the original data, generate tree bootstrap samples.

(2) With the following adjustment made for each of the bootstrap samples, create an unpruned classification or regression tree: Instead of finding the optimal split among all predictors at each node, randomly select various predictors and determine the optimal split among those variables. The selection of predictors reduces the connection between the ensemble's classifiers.

(3) Predict new data by combining the forecasts of n trees (i.e., plurality votes for classification, average for regression).

The algorithm controls bias by exchanging variances. Both decision trees have the same inherent bias. The likelihood of overfitting decreases as the number of trees grows, as its variance decreases. In addition, it performs well on large datasets, can process thousands of input features without removing features, can determine which variables are crucial for the classification, can deal with missing data, and can keep its accuracy even when this percentage is high (Breiman, 2001).

In comparison to a simple decision tree, the main drawback of random forests is that it is more challenging to determine the nature of the connection between a dependent variable and the resulting rule set. A RF has the dual roles of describing and forecasting. It's easy to see why its features are important, but that might not be enough if you're trying to tease apart causes and effects. Because of its versatility, RFs has become one of the most widely used and popular machine learning algorithms nowadays. To better understand land cover issues, Gislason et al. (2006) applied the classifier to geographical data. Using data from several ecological scenarios, including the presence of invasive plant species in California and the location of enclosures for cavity-nesting birds in Utah,

USA, Cutler et al. (2007) showed that RFs were more accurate than other commonly used classifiers. Lariviere and Vandenpoel (2005) gleaned insights into customer behaviour and ways to increase revenue growth from the records of a large European financial services firm. They concluded that the single most important predictor of whether or not a customer will return and make additional purchases is the customer's past activity.

2.2.3 Adaptive Boosting

Adaboost was proposed by Freund and Schapire (1997) and is an ensemble method for machine learning. Valiant's (1984) PAC (Probably Approximately Correct) learning model is the inspiration for adaptive boosting. Under the PAC learning model, a powerful classifier is one that produces a test error that is arbitrarily small for randomly sampled examples from any high-probability distribution. The only difference is that weak classifiers have an error rate of $\varepsilon t > 0.5 - \gamma$, where γ is a scalar of any size. In fact, weak learners are the classifiers with the lowest error rate compared to random classifiers.

As shown by Freund and Schapire (1997), a combination of weak learning algorithms that are just marginally more efficient than random guessing can provide a highly accurate classifier. By continuously include a weak classifier in the ensemble, Adaboost works to create a robust classifier. Adaboost may be used to enhance the performance of a wide range of subpar classifiers. By relying on posterior error rates, Adaboost constructs a very precise weighted majority hypothesis without requiring any prior knowledge of the error rates of the weak classifiers. The model weights each weak classifier according to how accurate it is. The model uses the accuracy of each weak classifier to determine its weight. Previous boosting algorithms' (Schapire, 1990) performance bounds were determined by the accuracy of the least accurate weak hypothesis; However, the performance of the Adaboost method may be increased by any improvement in the combined weak hypothesis.

The term "boosting" refers to a voting system that uses a weighted majority for the distribution of resources among choices in a dynamic manner (Littlestone & Warmuth, 1994). The goal of the boosting algorithms is to provide a very precise prediction rule by combining rough and correspondingly inaccurate rules of thumb. The booster iteratively generates a distribution D_t across the training instances and requests from an unspecified oracle a weak hypothesis (or rule-of-thumb) h_t with low error t with respect to D_t. In this manner, the learning mechanism focuses on examples that were incorrectly predicted in each round. Boosting's final challenge is to merge weak hypotheses into a single strong prediction rule. As a result, the weighted error in the AdaBoost algorithm is defined as:

$$\varepsilon_t = Pr_{i-D_t}[h_t(x_t) \neq y_i] = \sum_{i:h_t(x_i)\neq y_i} D_t(i) \tag{2.1}$$

The weighted error ε_t represents the probability that hypothesis h_t misclassifies a random example if selected according to D_t. The goal in each iteration t in the AdaBoost algorithm is to select hypothesis h_t to minimize the weighted error. In Hypothesis h_t, which is also called base or weak classifier, t_{th} round is like a "rule" that is used to classify the category of all the examples in this certain round.

If we have hypotheses h_1 , h_2 … … h_t in each round of iteration, then we can represent a combined or final hypothesis(classifier) H with the formula:

$$\mathrm{H(x)} = \mathrm{sing}(\sum_{t=1}^{T} a_t\, h_t(x)) \tag{2.2}$$

In the formula, a_t , which is called the learning rate, is the composite coefficient of h_t, and it also shows the level of significance assigned to hypothesis h_t in the final hypothesis. AdaBoost runs in rounds, and the distribution is updated after each round to give more weight to the incorrectly classified examples. The goal is that the weak classifier added to ensemble in round t will correctly classify some of the instances misclassified by the ensemble created through rounds *t-1*.

2.2.4 Gradient Boosting

Like RFs, gradient boosting is an ensemble approach with the goal to enhance the prediction capabilities of a model by training numerous and merging the results (Friedman, 2001). In the context of gradient boosting, the model being boosted is termed as a weak learner. This weak learner should be any model, not simply a decision tree. However, in practice, these functions are exceptionally well and thus typically employed.

In gradient boosting, bootstrapped subsets are utilized. First, the results of the training-prepared data are averaged. From this average, the outcome data provided for training is retrieved, and the errors are computed. Based on the errors, a decision tree is constructed. Using this decision tree, a second set of outcomes is then calculated. Using these estimated outcome data, error values are recalculated and a new decision tree is constructed as shown in Figure 2.7 in this figure each weak learner uses the subset of training data for prediction and each time after prediction give more weight to features that performs weakly. With successive decision trees that learn from the mistakes of the previous tree, better outcomes can be achieved. In bagging, decision trees are parallel, whereas in boosting, they are cascaded and trained sequentially. Similar to bagging, a stronger model is produced from poorly learned trees in this manner (Freund & Schapire, 1994).

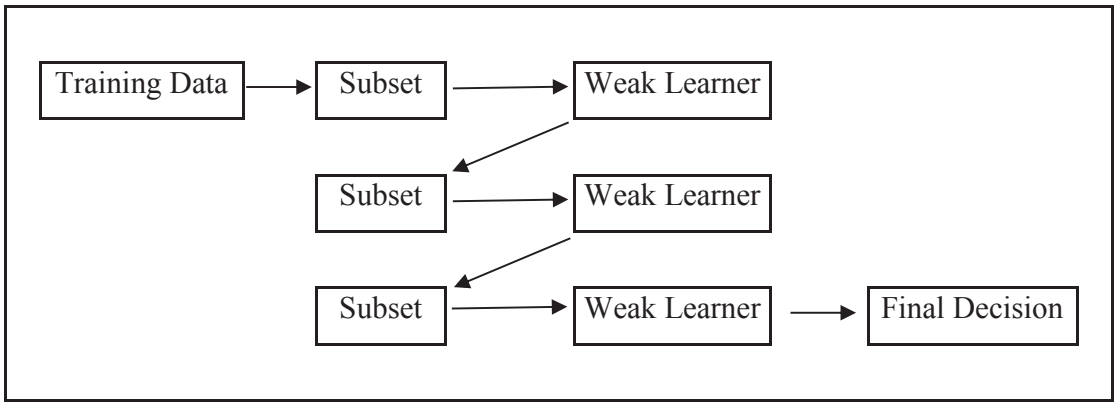

Figure 2.7 Gradient boosting procedure

Gradient boosting is a regression algorithm similar to boosting (Friedman, 2001). The objective of gradient boosting, given a training dataset $D = \{X_i, Y_i\}_1^N$, is to discover an approximation $\hat{F}(x)$ of the function $F^*(x)$, which delineates instances x to their output values y, by minimising the expected value of a given loss function $L(y, F(x))$. Gradient boosting utilizes approximates $F^*(x)$ a weighted sum of functions as an additive sum (Bentéjac et al., 2021):

$$F_m(x) = F_{m-1}(X) + \rho_m h_m(x), \tag{2.3}$$

where ρ_m is the weight of the m_{th} function, $h_m(x)$. These functions are the models of the ensemble (e.g. decision trees). The approximation is constructed iteratively. First, a constant approximation of $F^*(x)$ is obtained as:

$$F_0(x) = arg \min_{\propto} \sum_{i=1}^{N} L(y_i, \propto) \tag{2.4}$$

Subsequent models are expected to minimize

$$(\rho_m h_m(x)) = arg \min_{\rho,h} \sum_{i=1}^{N} L(y_i, F_{m-1}(X_i + \rho h(X_i)) \tag{2.5}$$

However, instead of solving the optimization problem directly, each h_m can be seen as a greedy step in a gradient descent optimization for F^*. To do this, each model, h_m, is trained on a new dataset $D = \{X_i, r_{\ddot{o}i}\}_{i=1}^N$, where the pseudo-residuals, r_{mi}, are computed by,

$$r_{mi} = \left[\frac{\partial L(y_i, F(x))}{\partial F(X)}\right]_{F(x) = F_{m-1}(x)} \tag{2.6}$$

The value of ρ_m is eventually determined by solving a line search optimization problem.

This algorithm is susceptible to overfitting if the iterative process is not regularized properly (Friedman, 2001). If the model *hm* fits the pseudo-residuals perfectly for some loss functions, then the pseudo-residuals become zero in the next iteration and the process terminates prematurely. Multiple regularization hyper-parameters are considered to regulate the additive process of gradient boosting. The natural way to regularize gradient boosting is to reduce each gradient decent step $Fm(X) = Fm\text{-}1\ (X) + \upsilon\rho_m h_m(x)$ with υ=(0, 1.0) with shrinkage.

Typically, the value of υ is set to 0.1. Additionally, improved by reducing the complexity of learned models. We can set limits on decision tree depth or the bare minimum of occurrences needed to divide a node. In contrast to random forest, gradient boosting's default settings for these hyper-parameters drastically limit the expressive capacity of the trees (for instance, the depth is typically restricted to less than ≈ 3−5). Last but not least, another category of hyper-parameters that might improve the generalization of the ensemble are those that randomize the base learners and are incorporated in the various gradient boosting implementations. Examples of these include random subsampling without replacement. (Friedman, 2002).

2.2.5 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is an additional ensemble algorithm based on decision trees. It is a gradient descent-based end-to-end tree boosting system that is highly scalable. The method continuously divides features to construct a tree. Each decision tree computes the feature and threshold that produces the optimal branch effect and completes the split construction. XGBoost employs a greedy search strategy to determine the optimal tree structure. XGBoost is an effective classifier algorithm due to its ability to effectively handle missing values, avoid overfitting, and support parallel structure, among other advantages (Chen & Guestrin, 2016).

In order to scale well, the decision tree ensemble known as XGBoost uses a technique called gradient boosting. While gradient boosting uses a loss function to generate an additive extension of the goal function, XGBoost uses a gain function. Since decision trees are used exclusively as the foundation for classification in XGBoost, a modified loss function is used to regulate the tree complexity.

$$L_{xgb} = \sum_{i=1}^{N} L(y, F(X_i)) + \sum_{m=1}^{M} \Omega(y, h_m) \tag{2.7}$$

$$\Omega(\mathrm{h}) = \gamma \mathrm{T} + \frac{1}{2}\lambda |w|^2 \tag{2.8}$$

where *T* is the tree's leaf count, and *w* are the leaves' output scores.

By incorporating this loss function into the split criterion of decision trees, pre-pruning techniques may be created. When γ is increased, the tree structure improves. γ calculates the needed minimum gain in loss reduction to partition an internal node. In XGBoost, the expansion steps may be made smaller by adjusting the Shrinkage regularization hyper-parameter. Tree complexity can be restricted in other ways as well, such by adjusting tree depth. Decreases in tree complexity also make model training faster and less space intensive. XGBoost uses randomization methods to speed up training and cut down on overfitting. XGBoost features two randomization methods: column subsampling at the tree and node levels, and random subsamples for training individual trees. Moreover, the "goal" hyper-parameter of XGBoost allows it to be used with any user-defined loss function by designing a function that outputs the gradient and the hessian (second order gradient) and then sending it through the optimizer. (Bentéjac et al., 2021).

Additionally, XGBoost provides a sparsity-aware method for determining the ideal split. Multiple missing values might be the source of attribute sparsity or entries having the value of zero. As part of calculating the gain for split candidates, XGBoost eliminates these rows automatically. Additionally, XGBoost trees reveal the default child node that branches instances with missing or null data. Further, XGBoost has requirements for

monotonic and feature interaction. The knowledge of domain-specific information can greatly enhance the utility of these traits. For each given set of input characteristics, XGBoost will always provide a monotonic regression result because to the limitations imposed by monotonicity (increasing or decreasing). The input characteristics that can be combined in pathways from the root to any leaf node are constrained by interaction restrictions between features. To meet these requirements, we restrict the number of splits that each node is allowed to evaluate (Bentéjac et al., 2021).

In addition, XGBoost uses other methods that have nothing to do with ensemble accuracy to speed up the training of decision trees (The tree ensemble model consists of a set of classification and regression trees which is also called CART). The most time-consuming part of decision tree construction algorithms is identifying the appropriate split, hence this is XGBoost's primary focus. Algorithms designed to discover splits often consider all of them and pick the one that yields the most benefit. The best way to divide each node needs a linear search across all of the attributes that have been ordered. XGBoost employs a compressed column-based structure with pre-sorted data to avoid sorting the data again in each node. Because of this, just a single sorting of each characteristic is required. The columnar data structure we've implemented here lets us calculate the best possible partitioning of all relevant properties concurrently. Furthermore, XGBoost employs an approach based on data percentiles in which just a sample of candidate splits is tried and their gain is determined utilizing aggregated statistics. Existing subsampling of data in CART trees is analogous to this idea (Breiman et al., 1984).

Boosting is based on the premise that if you tweak a learning technique, you end up with something even more effective. A "weak learner" or "weak hypothesis" is one whose results are only marginally better than those obtained from random chance. Hypothesis boosting works by first filtering out the data, leaving just what a weak learner can handle, and then training other weak learners to handle the remaining challenging data. These poor learners are recycled several times until the best one is created (Valiant, 2013).

XGBoost stands for “Extreme Gradient Boosting” (Friedman, 2001). XGBoost is a supervised machine learning algorithm that uses the input variables $X_i = \{ X_1, X_2, \ldots, X_n \}$ and tries to predict output variable $Y_i = \{ Y_1, Y_2, \ldots, Y_n \}$

Its high processing speed makes it a useful machine learning method, and its high power is another plus. Since it excels at solving classification and regression issues, it has become one of the most used machine learning algorithms. It has shown its resolve to eliminate the overfitting that typically occurs in decision trees by achieving impressive speeds and levels of performance (Hauskrecht, 2019). To deal with missing data, regularization is employed in XGBoost. Boosting is an error-correcting, data-adding ensemble learning technique. It keeps adding information until there is no more room for improvement. Each of the qualities is essential to the overall success of the product. Parameters are the unappreciated piece that requires data-driven learning. The model choice of the XGBoost is decision tree ensembles. A simple example can be seen in figure 2.6. In this example it is tried to find out if a person will like game X or not, it takes into account the age of the person if her or his age is less than 20, it is more likely to like the game and it results to higher score.

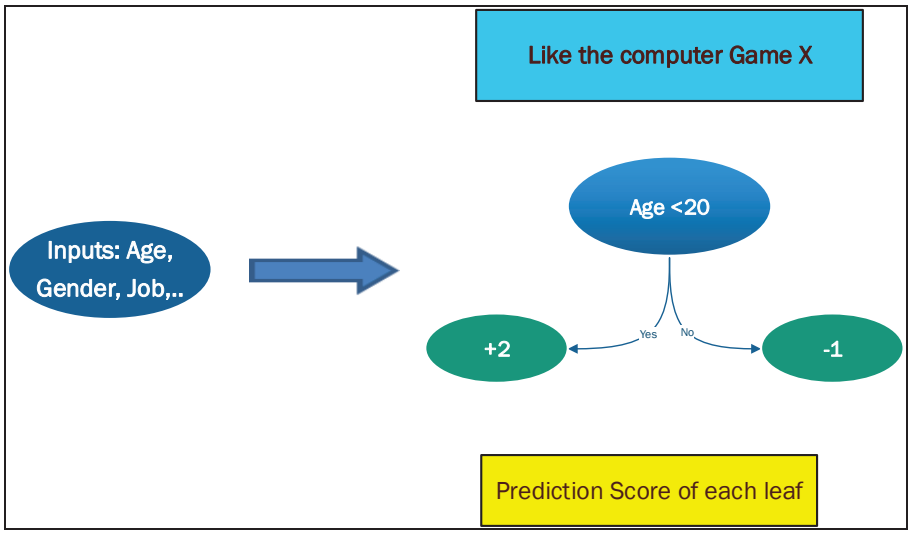

Figure 2.8 Inputs values of ensemble model (Chen & Guestrin, 2016)

It divides the family members into different leaves and assigns them the weight on the corresponding leaf. In a decision tree, each leaf just holds decision values, but in CART it is a bit different from that. The actual score in the leaves offers an accurate and rich interpretation to utilize in classification. In practice, we usually do not use a single tree. In fact, it is hoped to have an ensemble module that utilizes the aggregate prediction of several trees together (Hauskrecht, 2019).

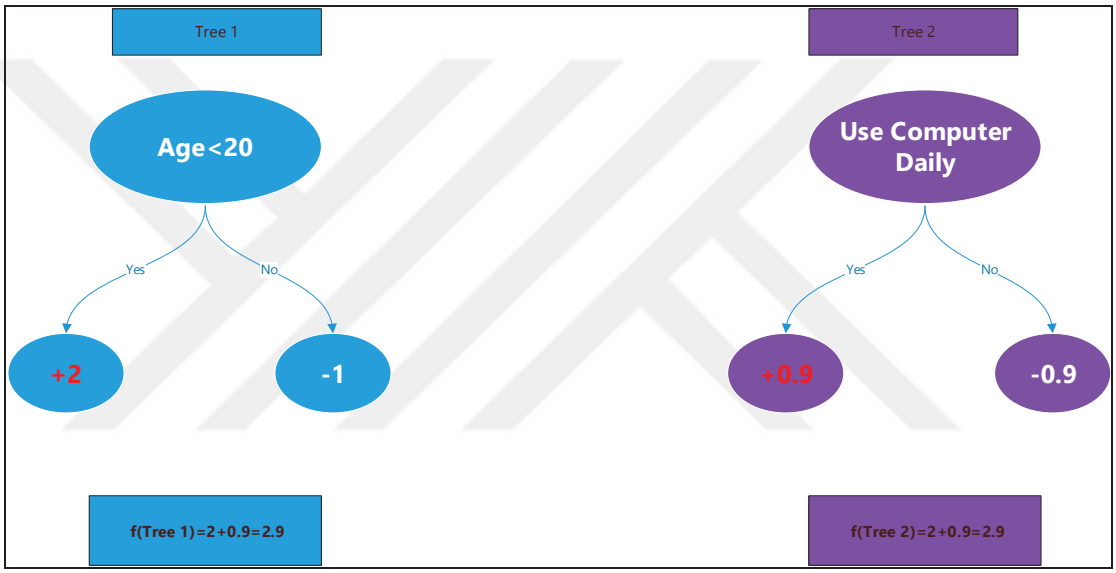

Figure 2.9 Tree ensemble model (Chen & Guestrin, 2016)

Figure 2.9 represents a tree ensemble model, in this example, we have a tree ensemble of two trees. The prediction score of each of them is used to get the final score. These two trees try to complement each other. In mathematical terms, we can say:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)\,, f_k \epsilon\, F \tag{2.9}$$

In this formula, *"k"* is the number of trees, *"f"* is the function in *"F"* function space, and *"F"* is the set of all possible CARTs. The objective function is:

$$obj(\theta) = \sum_i^n l(y_i, \hat{y}_{i)} + \sum_{k=1}^K w(f_k) \quad (2.10)$$

where $w(f_k)$ is the complexity of the tree f_k.

Tree Boosting

By defining an objective function and optimising it, we can learn trees. It must always include regularization and training loss in objective function. For the objective function, we have the following formula:

$$obj = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^t w(f_i) \quad (2.11)$$

Additive training

In this formula, each of the *"f_i"* parameters contains leaf scores and the structure of the tree. We need to learn its leaf scores. Learning from all trees is intractable. First of all, we fix what we have learned and add one new tree at a time. So we have:

$$\hat{y}_i^{(0)}=0$$

$$\hat{y}_i^{(1)}=f_{1(x_i)}=\hat{y}_i^{(0)}+f_{1(x_i)}$$

$$\hat{y}_i^{(2)}=f_{1(x_i)} + f_{2(x_i)}=\hat{y}_i^{(1)}+f_{2(x_i)}$$

…

$$\hat{y}_i^{(t)} = \sum_{k=1}^t f_{k(x_i)} = \hat{y}_i^{(t-1)} + f_{t(x_i)} \quad (2.12)$$

At each step, we want a tree that optimizes our objective.

$$obj^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^t w(f_i) = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + w(f_t) + constant \quad (2.13)$$

Our loss function will be mean squared error (MSE):

$$obj^{(t)} = \sum_{i=1}^{n}(y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)))^2 + \sum_{i=1}^{t} w(f_i) = \sum_{i=1}^{n}\left[2\left(\hat{y}_i^{(t-1)} - y_i\right)f_t(x_i\right) + f_t(x_i)^2] + w(f_i) + constant \quad (2.14)$$

For the first order, which we call residual, the structure of MSE is friendly, but for the rest of the losses of interest, getting such a good form is a little hard. Consequently, we take Taylor expansion of the loss function up to the second order:

$$obj^{(t)} = \sum_{i=1}^{n}[l(\hat{y}_i, \hat{y}_i^{(t-1)}) + g_i f_{t(x_i)} + \frac{1}{2}h_i f_t^2(x_i)] + \Omega(f_t) + \text{constant} \quad (2.15)$$

In which:

$$g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)}) \quad (2.16)$$

$$h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)}) \quad (2.17)$$

After eliminating all the constants, the objective at step *t* becomes:

$$\sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \Omega(f_t) \quad (2.18)$$

As a result, it activates our new tree's optimization objective. Because custom loss functions are supported by XGBoost and they are necessary for the goal function's value..

Model Complexity

After introducing the training step, now we will explain the regularization term. At first, we define the complexity of tree Ω(f) so we have *f(x)* as:

$$f_t(x) = w_{q(x)}, \omega \in R^T, q: R^d \rightarrow \{1,2,\ldots,T\} \quad (2.19)$$

The vector of scores on leaves is "*w*" and for the function assigning each data point to a corresponding leaf, we have *"q"*. On the other hand, *"T"* shows the number of leaves. The complexity in XGBoost's definition is:

$$\omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=!}^{T} \omega_j^2 \tag{2.20}$$

The Structure Score

Re-fomulation of the tree model gives a chance to write the objective value with the t-th tree as:

$$obj^t \approx \sum_{i=1}^{n}[g_i \omega_{q(x_i)} + \frac{1}{2} h_i \omega_{q(x_i)}^2] + \Upsilon \mathrm{T} + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 = \sum_{j=1}^{t}[(\sum_{i \in I_i} g_i)\omega_i + \frac{1}{2}\sum_{i \in I_i}(h_i + \lambda)\, \omega_j^2] + \gamma \mathrm{T} \tag{2.21}$$

Where $I_i = \{i | q(x_i) = j\}$ is a set of indices of data points assigned to the *j-th* leaf, expressions can be summarized by defining $G_j = \sum_{i \epsilon I_j} g_j$ and $H_j = \sum_{i \epsilon I_j} h_j$ so we have:

$$obj^t = \sum_{i=1}^{n}[G_i w_j + \frac{1}{2}(H_i + \lambda) w_j^2] + \gamma T \tag{2.22}$$

The best reduction we can get is:

$$w_j^* = -\frac{G_j}{H_j + \lambda} \tag{2.23}$$

$$obj^* = -\frac{1}{2}\sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T \tag{2.24}$$

Figure 2.10 shows an example of tree structure for probability of liking the game by a person according to gradient statistics. According to this tree if a person's age is less than 15 and its gender is male, it is more likely to like the game.

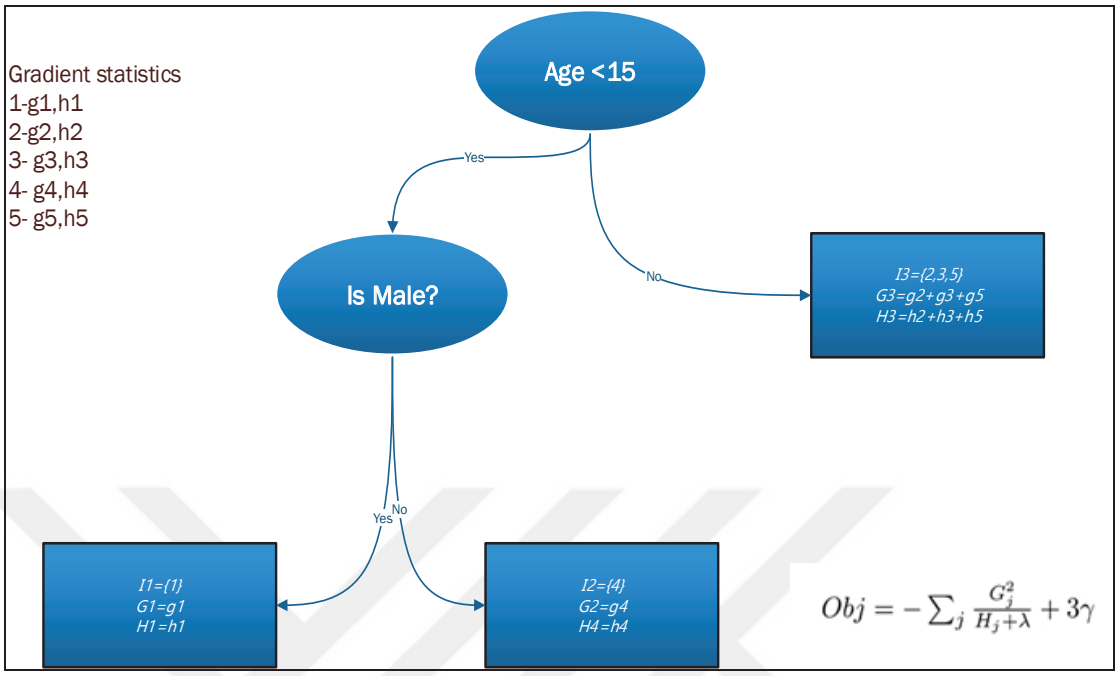

Figure 2.10 The tree structure (Chen & Guestrin , 2016)

We count all possible trees and pick the best one according to the previous section's explanation. In practice, this is intractable, so we try to optimise one level of tree at a time. In fact, we try to divide a leaf into two leaves and the score it gains is:

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \Upsilon \tag{2.25}$$

With this formula, we can calculate the score of new leaves, the score of the original leaf, and also regularisation on additional leaves. If the gain is smaller than Y, we won't add new branches. We call it "pruning techniques in tree base models."

A left-to-right scan is required to compute the structural score of all possible split solutions so we can learn the optimum split rapidly. XGBoost is a tool that is driven by the formal stated idea; it is built with both thorough consideration in terms of systems optimization and concepts in machine learning. The main goal of XGBoost is to push away the boundaries of machine learning methods and create a dependable and efficient machine learning approach.

CHAPTER 3
MODEL DEVELOPMENT DATA

Most of the data for this study is taken from Crunchbase, an online network that brings together entrepreneurs and investors. Crunchbase is a comprehensive database of companies on the market. As a result, it is used in the majority of studies on the issue. Crunchbase's database contains information from over 100,000 companies in a variety of sectors and places throughout the world. The database only contains businesses formed before 2012. Because the time period covers the financial crisis of 2008, the results may be conservative, which is a good thing in a good model. The database has eleven separate feature tables ("degrees," "IPO," "funds," "objects," "people," "acquisitions," "funding rounds," "investments," "milestones," "offices," and "relationships") linked by each company's ID.

Crunchbase dataset tracks an organization's status whether it is active, closed, acquired, or IPO (Initial public offering). Additionally, each organization is also identified in terms of its core function (business or investor) and categories and subcategories that describe the industry in which the organization operates.

The relevant tables like funds, funding rounds and investments contain more details on funding occasions, acquisitions, IPOs, and investors. They contain information on the dates of these events, the amount of money raised, and the sort of investment (seed, angel funding, series A, B, C, etc.). The People table represents individuals who are founders, investors, or employees of an organization. The table includes name, gender, address, links to social media accounts, organization, and position within the organization.

Degree table provides the information about entrepreneurs'' educations and their graduation date. There are five type of educations which are undergraduate, master, MBA and PhD.

Milestones table has information about number of the milestones and time of companies' last Milestone. These milestones are related to companies' achievements during their activity period of time. However, location of the companies and their geographical information is described in Office table, according to this table we have information about companies' Headquarters, region their address and zip code that come alongside the state and country information.

This raw data format allows for the testing of several hypotheses and various combinations of variables in order to develop a prediction model. However, because the database is "user-created," certain changes has been made to ensure that the data used accurately reflected the real startup environment, including risk and success elements that influence company outcomes. The following steps were performed for data selection.

3.1 Preparation of Data

The following were the stages involved in data preparation:

1. Only firms founded between 2000 and 2012 were included.
2. Only startups based in the United States were eligible.
3. Only startups with angel or venture capital investments greater than zero were eligible.
4. After four years of operation, startups to the category of "operating" without any milestones were dropped.
5. Python programme is used to put all of the important data from different feature tables and sources into a single data frame.

3.2 Target Variable

"Successful startup - likely to make a return to investors," as stated in the Model Framework, is the target or dependent variable. The steps to make a binary (0 and 1) target variable are:

-Failure "0" is assigned to startups with the status "Closed."

-Successful "1" startups are those that have had a successful exit (IPO or acquisition).

-After four years of operation, startups with the classification "Operating" and a new milestone (advancement with media attention) are categorised as successful "1."

Figure 3.1 shows our final observations which includes 927 samples that were classified as 262 samples for failure companies and 665 samples for successful companies. On the other hand, figure 3.2 illustrate percentage of successful companies per category according to this graph manufacturing, healthcare and hardware companies are three top successful companies in Crunchbase dataset.

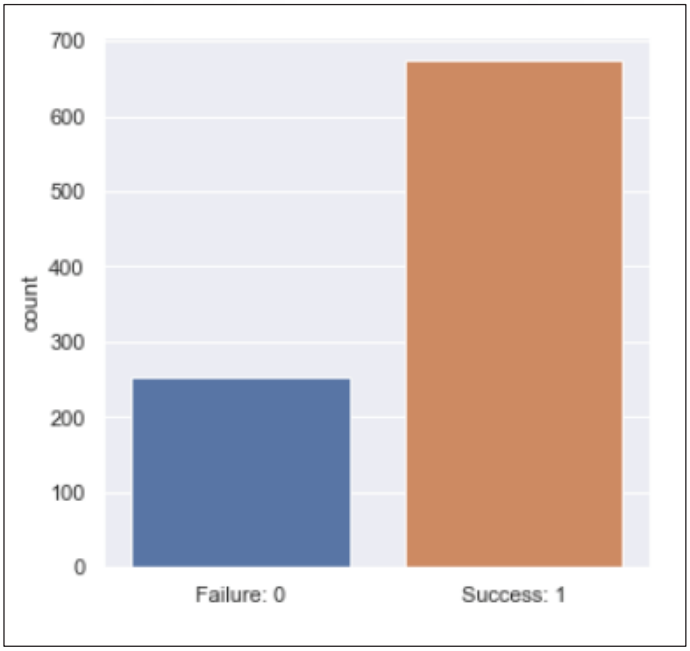

Figure 3.1 Counts for 'successes' and 'failures' of the target variable

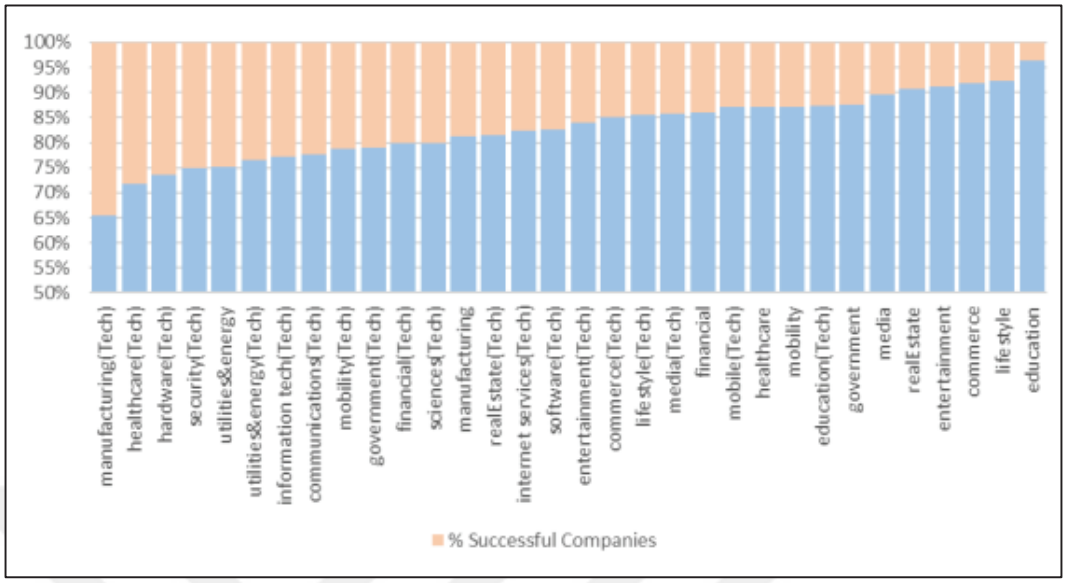

Figure 3.2 Percentage of successful companies per category

3.3 Independent Variables

3.3.1 Candidate Pool

The following are all of the factors that were taken into account when creating this model:

- Number of Angel Funding Rounds: The theory is that the more funding rounds a firm has, the more likely it is to have a fantastic product or market traction.
- Number of Milestones: This variable shows how many times the company has been talked about in the news in a good way. One of the most intuitive indicators of startup success.
- Months between the start date and the first milestone: This variable helps us to figure out what the entrepreneur is like as a person and how the business is growing.
- Percentage of businesses founded prior to the year of foundation: The percent of running companies founded before the year of each company's founding indicates the

stage of competition. A greater percentage indicates that the competition is more entrenched.

- Educational background and years since graduation: These variables reflect the entrepreneurs' educational backgrounds (undergraduate, master's, MBA, and Ph.D.) as well as their graduation dates. The impacts of education and experience on entrepreneurial success are captured by these factors.
- Team size: The number of individuals on the team is an effective way to capture the size of the firm, which may provide predictability to the model when other variables, such as industry and investment, are taken into account.
- Participants in the Angel investment rounds: The total number of angel investors demonstrates how fragmented the equity is, which has an impact on management and, as a result, the company's outcome.
- Amount of venture capital investment: Getting venture capital shows that a company has potential, but the amount of investment shows how much potential there is.
- Angel investment amount: The amount of angel investment can show how good the idea is and how passionate and driven the entrepreneur is.
- Months to get angel investment: The time is taken to get angel investment shows how quickly a business breaks even or moves up to a venture capital investment. It's important to figure out if the business is growing or wasting money.
- The current state of the industry: The increase in the number of enterprises and the average income in the sector may explain trends that impact the future capacity to locate talented human capital as well as demand in each industry.

3.3.2 Transformations

Some transformations are required to address variable collinearity and fix any non-linear behaviour of a single variable or a set of variables. The followings are the tried transformations, as well as the intuition that led to them:

• Rounds of Angel Funding and Time to Raise Angel Investment (years): Regarding these variables, there are two possible challenges to handle. One is their relationship.

Intuitively, the more rounds of funding a firm receives, the longer it takes to raise money. It has been attempted to use an interaction variable time (years) to raise the angel investment, multiplied by the number of angels, to tackle this probable issue and handle any possible collinearity or coefficient bias. The second issue is that those factors may not have a linear influence on startup success. Therefore, it has been attempted to add a second squared term for each, Number of Angel Funding Rounds squared and Time (years) to raise angel investment squared, because they appear to have a declining return per scale.

• Milestones and Time Required to Achieve Them (months): The behaviour of these two variables is like that of the previous item; the more milestones raised, the longer it is projected to take to raise those milestones. It has been attempted to introduce an interaction term between those two variables, like it has been done in the previous item. The number of milestones is multiplied by the amount of time it took to attain those milestones (in months). Another factor to consider is how those factors interact with one another. For example, early media attention represented by a negative value of the time to reach the first milestone (milestone prior to the company's foundation) could only mean something solid if it was followed by other milestones, implying that a potentially disruptive technology/idea was validated through subsequent milestones. A late first milestone, followed by a slew of others, might indicate that the firm took its time to learn about and adapt to the industry before expanding. The replacement of variables by a function is a realistic technique to capture this behaviour; it has been tested for square and cubic functions here: $(a+b)^2 = a^2+2ab+b^2$ and $(a+b)^3 = a^3+2a^2b+2ab^2+b^3$.

• Entrepreneurial Education and Years from Graduation: It appears that the type of education and years from graduation to business formation (a proxy for experience) have a non-linear relationship with successful entrepreneurship. To get the best mix of knowledge and experience, various levels of education necessitate different levels of experience. The influence of experience on startup results is not linear, but rather squared, with an upward slope for startup success in the first years and a decreasing slope thereafter.

Those variables which are mentioned above are turned into a squared function to represent this connection, as they have been done in the previous item, $(a + b)^2 = a^2 + 2ab + b^2$.

Literature review of variables in previous studies are summarized as follows in Table 3.1. In this table each variable and its effect in benchmark research are outlined.

Table 3.1 Literature review of variables

Variables	Exp. effect	Benchmark research
Number of funding rounds	Positive	Krishna et al. (2016)
Number of milestones	Positive	Shah (2019)
Time in months from foundation to the first milestone	Positive	Graham (2012)
Percent of firms in the industry before the year of foundation	Negative	Finkelstein (2002)
Education times months from graduation	Positive	Van Gelderen et al. (2005) Gimeno et al. (1997)
Small team -Team composed by three or fewer people (Dummy)	Negative	Roach& Sauermann(2015)
Participants in the angel investment rounds	Positive	Van Gelderen et al. (2005)
Startup reached Venture capital rounds of investment (Dummy)	Positive	Hellmann &Thiele (2015)
Time in months to raise Angel investment	Not clear	Feinleib (2012)
Dummy if the industry is expanding	Positive	Porter & Strategy (1980)
Type of industry (Dummy)	Both	-
State - Region (Dummy)	Both	-
Startup is location in the Silicon Valley and other micro regions (Dummy)	Positive	Krajcik & Formanek (2015)

3.4 Rejection inference

Businesses with no angel or venture capital investments have been removed from the sample, as indicated in the Model Selection section and Chapter 2 (Data Framework). The reason for this omission is a lack of faith in those observations; a business that does not receive some early funding, such as a seed, grant, or angel investment, is unlikely to function and succeed; therefore, the chances of missing information are significant. As a result of this exclusion, there is no information about the firms that had been received no funding and their chances of success based on the other parameters. It resulted in a censorship issue. According to the literature (described in Chapter 4), the quantity of angel investment has a convex influence on the chance of success; this implies that angel investment has a linear positive effect until it reaches a point, after which it declines. The assumption is that following a round of funding, the firm will need to gain enough traction to go on to more advanced kinds of funding (e.g., venture capital), where they will receive a greater degree of assistance. Too much angel investment might indicate that the firm did not properly utilize resources, was wasting too many funds, or was selling away too much ownership too soon. (Feinleib,2012). Because of the filtering, the quantity of angel investment displayed a strictly negative linear impact when it was fitted to an ensemble learning model, so it was required to include firms that did not get any funding in the data. According to Mok (2009), the approach which is used is Rejection Inference and it is a method of determining the likelihood of default for potential loan applicants who have been rejected by the acceptance policy. The problem is solved by the Rejection Inference to prevent data bias problem. When the data are missing completely at random, then there is no reject inference problem at all. If the data are missing at random, then the selection mechanism is ignorable. But when the data are missing not at random, then the selection mechanism is unavoidable. (Mok, 2009).

Figure 3.3 shows the balance between successful and failed startups and how the whole sample did in terms of successes and failures after the data from the rejection inference was added to the original sample.

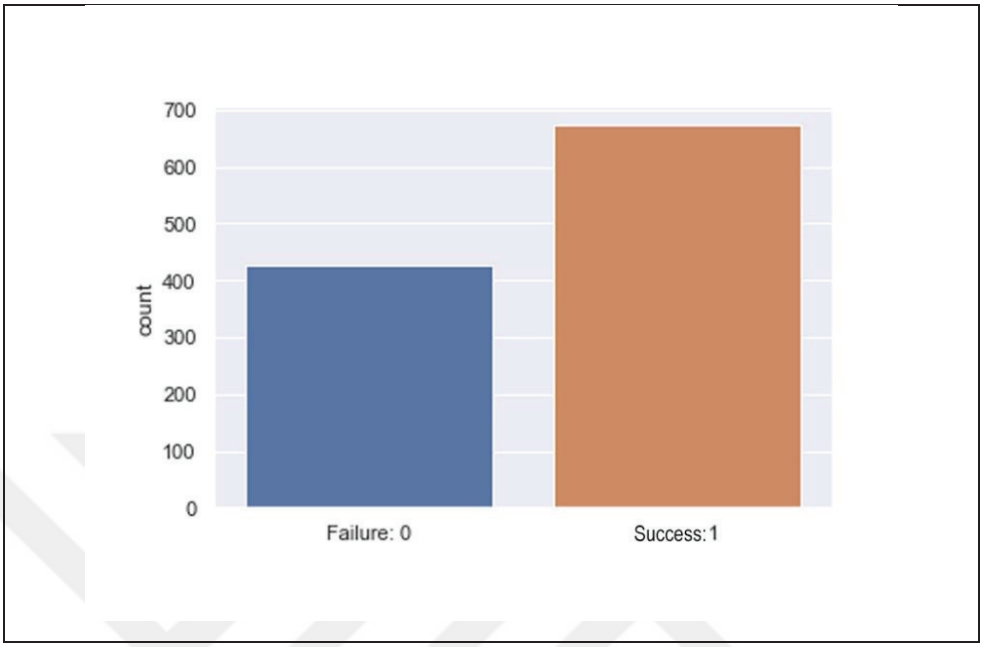

Figure 3.3 Counts for 'successes' and 'failures' of the target variable after the rejection inference method

3.5 Training and Testing Data

To fit and evaluate the correctness of models, the sample has been divided into two sub-datasets at random: training (in sample), which accounts for 70% of the observations, and testing (out of sample), which accounts for 30% of the data. The target variable is taken out of the testing sample to make sure the results are accurate.

CHAPTER 4
MODEL ESTIMATION AND RESULTS

In this work, Extreme Gradient Boosting is utilized as benchmark technique for variable selection, as mentioned in the previous chapters, and because of its simple interpretation and statistical approach, it is feasible to compare the estimated effect and statistical significance of each risk factor. To meet the research goal, it should be ensured that the final model is predictive and that the predicted coefficients accurately reflect the true influence on startup performance. As a result, it's critical to double-check that all of the variables are well-specified, that the results reflect economic intuition, and that past research backs up the conclusions. This study compares the accuracy of the model's prediction to that of other well-known approaches, such as Gradient Boosting, Random Forest, and Adaboost, after evaluating the variables and the model's dependability. The end results are compared to the results of other studies.

Following the stages of this method in detail:

1. Extreme Gradient Boosting is applied to all hyper-parameters and changes that have a conceptual impact on startup results.
2. Recognize that each variable has a random effect, make the necessary changes, and get rid of the risk variables whose p-values show that they are statistically insignificant.
3. Put all of the classification algorithms onto the final model, using the training data.
4. The result of each categorization technique were used to score the test data.
5. Somer's D, Accuracy, Confusion Matrix, and Area Under The Curve (AUC) were utilized to calculate the performance of each classification technique (area under the curve).

The analysis in this thesis was carried out by using the Python programming language and the following libraries:

Data manipulation with Pandas, DateTime, NumPy, and Scipy.

For performing the statistical and machine learning prediction algorithms, Sklearn, Statsmodels, and XGBoost were employied.

Graphviz, Matplotlib, and Seaborn were used to plot the visualizations.

4.1 Metrics for Model Performance

Somers' D, Accuracy, Confusion Matrix, ROC curve, and AUC score were the performance measurements.

4.1.1 Somers' D

The value of Somers' D (Somers, 1962) is defined as

$$D_{XY} = \frac{TXY}{TXX} \tag{4.1}$$

where *TXY* is the difference between the number of concordant (the predicted is equal to the observed) and discordant (the predicted is different than the observed) pairs, and *TXX* is the total number of pairs. The higher the value of Somers' D, the better the performance of the model.

4.1.2 Accuracy

The accuracy represents the percent of the total estimations that were correct (the predicted is equal to the observed). Following the accuracy equation

$$\text{Accuracy} = \frac{True\ Positives + True\ Negatives}{True\ Positives + True\ Negatives + False\ Positives + False\ Negatives} \tag{4.2}$$

Such as the Somers' D, the higher the accuracy, the more predictive the model is.

4.1.3 Confusion Matrix

The confusion matrix details the number of true positives, true negatives, false positives, and false negatives and measures performance of machine learning algorithms. As it presented in Table 4.1, Confusion matrix is a table with four combination of predicted and actual values and it is used to describe how well a categorization method performs. The output of a machine learning algorithm is shown and summarized in a confusion matrix.:

Table 4.1 Confusion Matrix - Source: Glass box artificial intelligence

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives(TNs)

4.1.4 ROC Curve - AUC Score

The ROC curve (Receiver Operating Characteristic Curve) is a graph with the true positive rate on the Y-axis and the false positive rate on the X-axis; the AUC (blue line in figure 4.1) is the Area Under This Curve; the red line in figure 4.1 represents the 0.5; every model with an AUC above it indicates that the model could be more predictive than tossing a coin randomly. As Kraus and Kuechenhoff (2014) explain, if there are two samples of failures and two samples of successes, the possible scores for each sample are:

$$\begin{cases} 1 \; if \; x_f > x_s \\ 0.05 \; if \; x_f = x_s \\ 0 \; if \; x_f < x_s \end{cases} \tag{4.3}$$

Where $x_f \; and \; x_s$ are the scores from the failures and successes, respectively. By taking the average over the comparisons, the AUC can be written as following:

$$AUC = \frac{1}{n_f.n_s}\sum_1^{n_f}\sum_1^{n_s} S(x_f.x_s) \quad (4.4)$$

As for all the metrics above, the result of the AUC score is expected to be higher for more predictive models.

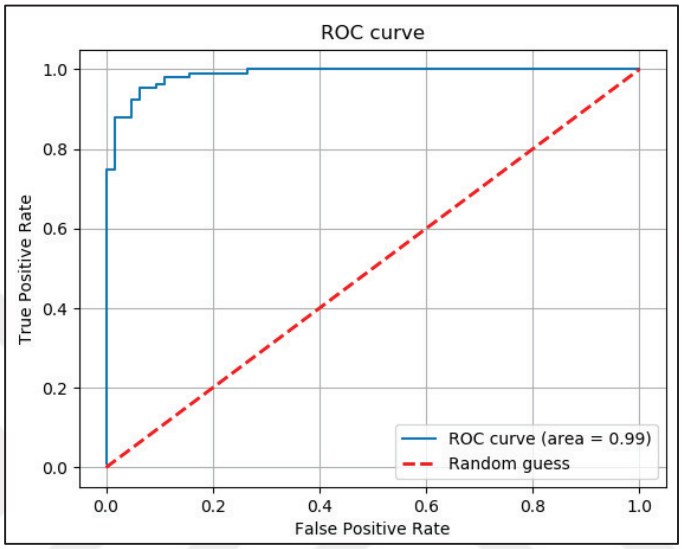

Figure 4.1 ROC - Source: Sklearn – Python

4.1.5 Cross Validation

Cross-validation is a method of determining the model's sensitivity. In this study, it used *k*-fold cross-validation, which divides a data set into *k* disjoint folds of roughly comparable size and uses each fold to test the model inferred by the other *k-1* folds using a classification technique. The average of the *k* levels of accuracy obtained via *k*-fold cross-validation is used to evaluate the classification method's efficiency, and so the level of averaging is deemed to be at fold (Rodriguez et al., 2009). The ROC-AUC (Area Under the Curve) is the performance parameter utilized as the output of each fold in this study. The difference in performance between the folds with lower and higher performance shows how good the model is. A big difference shows that the model is not consistent and may not be well described.

4.2 Model Selection

Model selection is the process of choosing a set of variables from a pool of candidates that gives a result that makes sense from an economic point of view and works well outside of the sample.

4.2.1 Removing Non-predictive Variables

Certain variables in the initial suggested model proved to be poor predictors of startup results based on their p-values after running the entire model using Extreme Gradient Boosting (not significant at a 5% significance level). To avoid incorporating non-significant factors in the final model, Cramer's approach calculated heteroscedasticity consistent (HC) standard errors (Cramer, 2007). The variables are: Gross Domestic Product (GDP) growth in the year of foundation; the state of the industry (expansion, stagnation, and recession) in the years leading up to the foundation; the state in which the company is based; the micro-region in the United States where the company is based (with the Silicon Valley area serving as an exception); and the majority of the specific industries as individual factors are among the variables. In the testing data, the model showed substantially improved prediction after deleting such factors. That isn't to say that such characteristics have no bearing on startup success; it just means that they aren't very effective predictors in our model. Future studies might benefit from testing those elements separately and in a different way to better understand their impact on startup results.

4.2.2 Data from Training and Testing

The data is split into two categories: training and testing, which account for 70% and 30% of the observations, respectively. The training set was used to fit the model, and the test sample, also known as the holdout sample, was used to calculate the results. In this study, *k*-fold cross-validation is also employed in this work, which entails fitting the model

with n observations and testing it with the rest, then fitting with $n + 1$ and testing it with the rest, and so on until it has had enough results to assess the model's consistency.

4.3 Rejection Inference Application

The majority of the random effects were as predicted after running the model with the optimal variables. The variable "Amount in $ of Angel Investment," on the other hand, showed a negative linear effect, which contradicts economic intuition and literature. According to the literature (details in the following section), angel investment should have a convex influence on company success. It means that while any investment is better than none, as it has increased the value of the company, at some point (depending on the industry, target market, and revenue stream), the company should stop receiving angel investment and instead pursue venture capital rounds, which provide better support to help the company grow to the next level. If the firm continues to attract additional angel investment, the return per share will begin to decline. The removal of data with zero total investment is the cause of a linear negative influence on this variable (explained in Chapter 3). Because the dataset only had startups with some amount of investment, the startups that didn't get angel capital only got venture capital rounds of investments, which, as explained in the next chapter, have a higher chance of succeeding. This means that the effect of angel investment was skewed because there were not enough observations with a total of zero investments.

The answer to this problem is to use Rejection Inference, which is a sample selection bias correction. Using a changed threshold, as explained in Chapter 3, it collects observations without spending any money.

4.4 Check Logit Assumptions

4.4.1 Multicollinearity

Before analyzing the results of Extreme Gradient Boosting, it is important to check whether the model is free of perfect multicollinearity, which means that two or more variables are not perfectly correlated. Otherwise, all the results are invalid. Variance Inflation Factor (VIF) was used for this step, which quantifies how much the variance inflates with each variable. When there is multicollinearity in the model, the standard errors and, consequently, the variance inflate. The "rule of thumb" for understanding whether a model presents multicollinearity differs from author to author. According to Hair et al. (2006), the tolerance level is 10. After that, collinearity is a problem. Rogerson (2001) says that the tolerance level is 5, while Pan and Jackson (2008) say that it is only 4.

Prior to the addition of functions F1 = (Education + Experience)^2 and F2 = (Time to First Milestone + Number of Milestones)^3, all VIF values in our model were less than 4, which is below the tolerance level. After the addition of the variables that compose the functions (F1 has 3 variables and F2 has 4 variables), VIFs of all independent variables are presented in Table 4.2, according to this table the VIF factors of some variables composing the functions surpass 5, which the highest is of "2 x Education x Time (Years) from Graduation-2ab-", with a value of 6.77. The cause of those VIF values higher than 5 is the format utilized to capture the effect of the variables; it does not imply multicollinearity. Besides, they are still lower than 10.

Table 4.2 Variance Inflation Factor - all variables from the final model

	VIF Factor	Variables
0	4.31355	Percent of companies before foundation (ind)
1	1.55350	Located at the Silicon Valley (Dummy)
2	1.43166	Industry: Business Services (Dummy)
3	1.06032	Industry: Social and Non-profit (Dummy)
4	1.42706	Reached Venture Capital round* (Dummy)
5	1.64307	Industry: Business use Technology (Dummy)
6	1.16206	Amount of Angel investment (in Millions) x N Participants
7	4.83431	Fl: Tine (Years) from Graduation *squared*
8	2.02884	Fl: Education (0-4) *squared*
9	6.77073	Fl: 2 x Education x Time (Years) from Graduation
10	2.97629	F2: Milestones cubic
11	1.84714	F2: Time (Months) to first milestone cubic
12	5.49895	F2: 3x Milestones squared x Time (Months) to first milestone
13	4.19376	F2: 3x Milestones x Time (Months) to first milestone squared
14	2.66551	Industry: Popular use Technology (Dunny)
15	1.45915	Total Funding rounds x Time (Years) to raise Investment

4.4.2 Heteroskedasticity

The inclusion of heteroskedasticity in the disturbances of an otherwise correctly specified model leads to coherent but ineffective estimated parameters and inconsistent covariance matrix estimates. It could make concessions in the model selection process as we utilize the t-values to determine the variables to be removed. To overcome this difficulty, the standard errors have been computed using an HC1-Heteroskedasticity-consistent covariance matrix estimator (Cribari-Neto & Galvão, 2003).

4.4.3 Independent Observations

Each observation in the database has a unique identification number; each observation refers to a different startup, which is unique and not related to others in the dataset. In the data cleaning process, all the observations were verified and checked for possible repeated cases. The observations that went into the final model are separate and show how a unique company grew.

4.4.4 Linearity

Most of the variables are linearly related to the log odds of success of a startup. The variables with non-linear effects were changed to fit an Extreme Gradient Boosting method. It worked because the individual variables and functions used are either statistically significant on their own or statistically significant when used together, and their coefficients make sense both in theory and in practice.

4.4.5 Sample Size

The final dataset has 1215 observations, of which 850 are for training and 365 are for testing. As Bujang et al. (2018) suggest, it has been calculated the minimum number of observations in a training dataset to deliver the statistics that represent the parameters using the following formula: $n = 100+50i$, where *"i"* represents the number of variables. The final model has 15 variables, so 850 observations would be a good number, which is exactly how many observations are in our training dataset.

4.5 Interpreting the Coefficients

The final output of Extreme Gradient Boosting is presented in table 4.3 which contains coefficient, standard error (SE), Z-score and P value of each variable. According to this table Total Funding rounds x Time (Years) to raise Investment Business Services and Reached Venture Capital rounds have higher coefficient.

Table 4.3 Output of extreme gradient boosting

	Coef.	S.E.	z	P>|z|
Percent of companies before foundation (ind)	-0,0265	0,0826	-10,3552	0,0000
Located at the Silicon Valley (Dummy)	0,6909	0,1988	3,4895	0,0005
Industry: Business Services (Dummy)	8,5692	0,2764	2,0591	0,0395
Industry: Social and Non-profit (Dummy)	-1,7373	0,8512	-2,0418	0,0412
Reached Venture Capital rounds (Dummy)	3,4313	2,4964	1,3745	0,1693
Industry: Business use Technology (Dummy)	0,4448	0,2927	1,5196	0,1286
Amount of Angel investment (in Millions) x U Participants	0,1452	0,0564	2,5751	0,0100
F1: Time (Years) from Graduation 'squared*	-0,5967	0,2899	-2,0582	0,0396
F1: Education (0-4) 'squared*	-0,0499	0,0587	-0,8499	0,3954
F1: 2 x Education x Time (Years) from Graduation	0,2702	0,1473	1,8347	0,0666
F2: Milestones cubic	0,0580	0,0190	3,0575	0,0022
F2: Time (Months) to first milestone cubic	-0,2908	0,0679	-2,9570	0,0031
F2: 3x Milestones squared x Time (Months) to first milestone	0,0990	0,0423	2,3423	0,0192
F2: 3x Milestones x Time (Months) to first milestone squared	0,2930	0,1043	2,8099	0,0050
Industry: Popular use Technology (Dummy)	-0,3702	0,2137	-1,7326	0,0632
Total Funding rounds x Time (Years) to raise Investment	8,9604	2,6865	3,4378	0,0006

The signs of the regression coefficients are consistent with our expectations and previous empirical research on the effect of each variable on startup success or failure.

4.5.1 Function 1: (Education + Time from graduation)^2

The projected effect of education on startup success is favorable; high education might reflect technical skills, general knowledge, a strong network, and a high IQ, all of which

could have a beneficial impact on the entrepreneurial process. The variable education, categorised as either low or high in Van Gelderen et al.'s (2005) model, has a favourable influence on startup performance. According to Gimeno et al. (1997), the number of years of schooling an entrepreneur has had a negative impact on the risk of the firm failing during the first five years of its founding.

Experience, like education, is believed to have a favourable impact on the success of entrepreneurship. It might suggest management experience, industry knowledge, a network, personal finances to support the company's early stages without giving up too much stock to early investors, and credibility, among other things. The total experience (work, management, and competence in beginning a firm) has a good influence on the success of a startup, according to Van Gelderen et al. (2005). Figure 4.2 shows effect of function 1 on odds of success, in this graph the impact of experience on startup success varies depending on the type of education. Transformations. After completing an undergraduate degree, *"x"* years of experience is not the same as *"x"* years of experience after completing a master's, MBA, or Ph.D. To capture this pattern, a quadratic function has been used to modify those two variables.

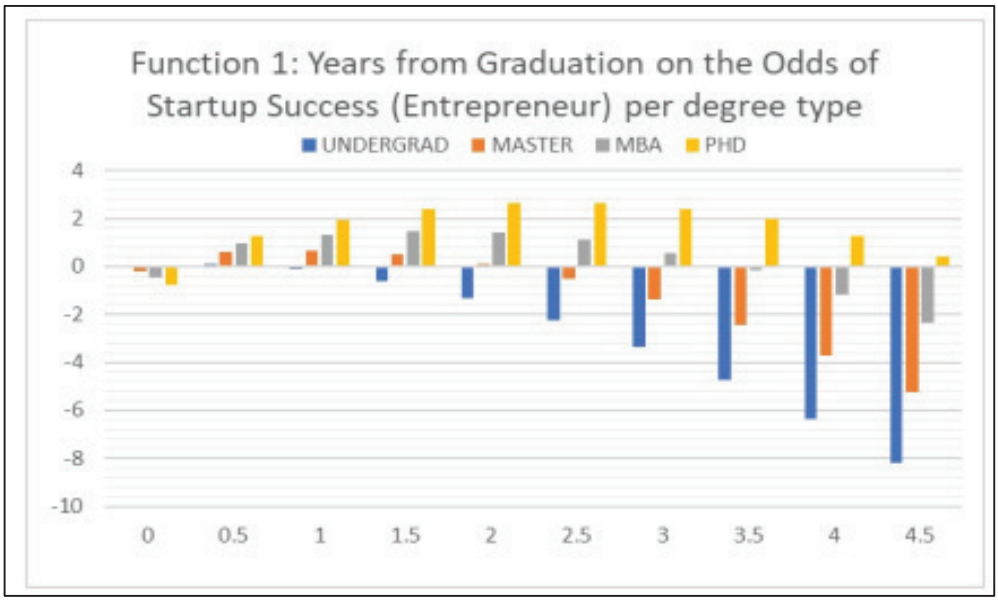

Figure 4.2 Effect of function 1 on odds of success

According to graph in figure 4.2, it can be seen that some post-graduation experience is preferable than none for all sorts of degrees, and each type has a distinct "optimal" quantity of experience prior to starting a business. Higher degrees necessitate more experience in order to improve chances of becoming a successful business. There are not enough observations of each degree type or time since graduation to make an exact statement about the magnitude of impacts, but the structure of them appears to be aggressive.

4.5.2 Function 2: (Milestones + Time to first Milestone)^2

According to the Cambridge dictionary, a milestone is a major event in the growth or development of something or someone's life. Milestones in our database indicate excellent news about the firm that has received media attention. A company's success is demonstrated by a milestone, which is a real occurrence. It usually occurs when a corporation introduces a new technology, opens new offices, achieves exceptional sales, or forms new collaborations. The impact of a new milestone on a startup's chances of success is favourable. In the model for predicting success, one of the most important things to look at is the number of milestones.

The coefficient's signal, however, is not stated by the author in the publication. The time (in months) between the founding and the first milestone shows how quickly the company accomplished its first goal. Our first assumption was that this variable had a negative effect, meaning that the faster a business received media attention, the more likely it was to succeed. Many academics, on the other hand, believe that the longer it takes to attain the first milestone, the more likely the firm will prosper.

This effect is explained by Graham (2012):

A solid startup usually goes through three stages of development:

1. There is an initial period of slow or no growth as the firm figures out what it's doing.

2. There is a period of rapid growth as the company learns how to manufacture something that many people want and how to connect those individuals.

3. In the end, a successful startup is likely to grow into a big company, but this growth is likely to stop, mostly because of the company's own limits and partly because of how close it is to the supply chain it serves. It implies that successful companies may take longer to mature and understand their businesses before they begin to expand. This fact might also be linked to the entrepreneur's psychological elements, notably resilience. Masten (2001) says that "resilience is the part of psychological capital that involves being able to deal with and adjust to risks." Intuitively, a tenacious entrepreneur would not quit after a difficult start, but rather adapt, learn the market, and successfully navigate the initial phase of a successful firm, as described above. Baluku et al. (2016) found that resiliency is positively associated with startup success.

Since the above studies were done correctly, combining these factors is likely to lead to accurate results, as described in Section 3.3.2.-Transformations. It's seen in the graph below:

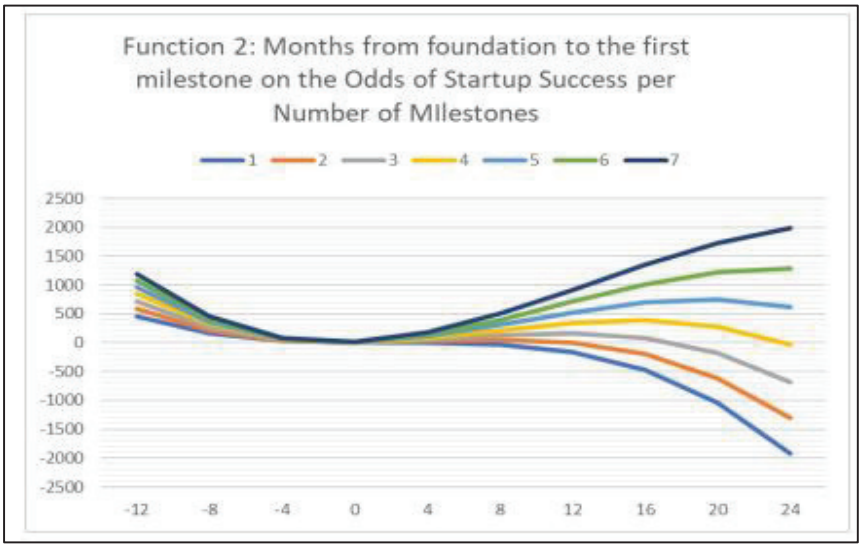

Figure 4.3 Effect of function 2 on odds of success

In figure 4.3 which is a graph of function 2's effects on odds of success, it can be seen that firms that receive media coverage before the foundation are more likely to succeed than those that receive coverage soon after the foundation, regardless of how many milestones such startups receive. For those startups that get the first milestone after the foundation, the number of milestones obtained later determines whether the early obtained media coverage reflected pure speculation/paid advertising or real disruptive prospective. Startups that waited longer to achieve their initial milestones and received less than four are likely to suffer. When a company takes longer to reach its first milestone and has more than four milestones, the slope of the function goes up with each new milestone after the fourth.

4.5.3 Percent of industry's Older Enterprises

This variable is used to determine how competitive each organisation is. It indicates the proportion of startups in the startup's industry that were started in years prior to the startup's founding year. The goal is to approximate the rivals' size, market share, and expertise. According to Finkelstein (2002), the first businesses to join a market have a competitive advantage over subsequent entrants, as he defines first mover advantage as "a competitive advantage that accrues to enterprises by virtue of their being the first entry into a market."

As a result, the sign of this variable should be negative, indicating that the larger the percentage of enterprises that existed before a startup's founding, the riskier the environment it would encounter and hence the less likely it would succeed. When the model was run, this variable had the negative coefficient that was expected, and it was statistically significant at the 1% level.

4.5.4 Located at the Silicon Valley

After testing the effects of U.S. states, macro-regions (e.g., mountains, mid-atlantic, New England, and others), and micro-regions (N.Y., Texas, Silicon Valley, Raleigh, and other "startup hubs"), the only one statistically significant was Silicon Valley. Supposing that with more observations, New York City and Dallas would also be significant, as they are the other two most important startup regions in the U.S. As expected, the "Silicon Valley effect" is positive, and companies located there are 99% more likely to succeed. Armour et al. (2004) say that a culture that encourages creative disruption and a concentration of highly skilled and eager employees, investors, and creative entrepreneurs from all over the world create an environment that attracts the most promising early-stage companies and improves their chances of success.

4.5.5 Reached Venture Capital Rounds

This dummy variable represents the firm's successful venture capital fundraising round. A company that made it to the venture capital stage of investment was 30 times more likely to succeed, according to this model. This variable enhances predictability, and its coefficient, while not statistically significant, satisfies economic expectations. It also has been played about with the amount and log of venture capital investment variables. They lacked the statistical significance and predictability of the dummy, though. However, the quantity of money spent varies depending on the market and the structure of the business, the amount of money invested is essentially the capital required to carry out the plan. As a consequence, the amount of money spent does not always indicate a greater chance of a startup's success. The basis for seeing venture capital investments as a success factor is the improved maturity of the businesses generally allowed in such rounds of financing.

4.5.6 Number of Angel Investors x Angel Investment (in Millions)

The quantity of angel investment (in millions) and the number of angel investors have a favourable impact on the success of a firm. At the 1% significance level, this variable is likewise statistically significant. The relationship between those factors is the basis for organising them as a product rather than as distinct bases; more investors tend to result in a greater quantity of angel investment. According to Wong (2002), since many angel investors have previously worked as entrepreneurs or industry specialists, they may be able to reap some personal benefits by assisting in the formation of a new form. Help in forming a management team and obtaining more funds are two of the most well-known ways of providing assistance. It shows that the more angel investors there are, the more angel money is raised. Another point to consider is that, because angel investors assume a bigger risk, they will only extend their investment or aggressively use their network to raise cash if the firms and their entrepreneurs have promising prospects. A company with bad management and a short-term focus won't be able to get new investors or more money from the ones it already has.

4.5.7 Time to Raise Investment x Total Funding Rounds

The product of the total number of financing rounds and the amount of time it took to raise the money in years is positive and statistically significant. The link between various factors is the basis for grouping them as a product. More financing rounds tend to lengthen the time it takes to raise capital. This variable encompasses two critical factors of a company's performance. The first is the notion that raising too much cash too quickly is a risk factor since a firm requires time to adjust its products and services to the market and develop a more efficient strategy. As Feinleib (2012) illustrates, too much early financing causes businesses to burn through money too quickly and lose equity, restricting future rounds. Second, additional investment rounds only take place if investors remain optimistic about the company's prospects. If it looks like a company will fail, investors are likely to stop putting money into it.

4.5.8 Industries

Business use of technology (positive impact), personal use of technology (negative effect), social/nonprofit (negative effect), and business services are the only four industries in the sample that are statistically significant and jointly significant (positive effect). These consequences are a reflection of market behaviour throughout the time period studied (2000–2012).

Intuitively, a concept known as "winner-takes-all" appears to be the cause of the difference in signal between business and personal use technologies. As explained by Marmer et al. (2011), in some marketplaces, particularly with big-scale standard products or services that broadly define personal use technology, the business with the most traction becomes a monopoly, and the others quit the market (e.g., Google, YouTube, Facebook, Uber/Lyft). Since there are so many gaps in this industry, businesses that use technology to solve specific problems tend to focus on niches and specific business issues.

Since they rely on government assistance and contributions in most cases, the social and nonprofit sectors are more subject to political and economic changes. They are also more easily affected by government changes and financial crises. Because it is made up of many diverse areas of goods and services, the business services industry is heterogeneous and difficult to monopolise. It can target firms on a local, regional, national, or global scale. Many startups have benefited from the increased demand for IT and internet services by businesses. Another odd feature of this market is the potential for acquisitions; startups that provide business solutions to large corporations are more likely to be bought out.

4.6 Comparing Alternative Algorithms

When tested with out-of-sample data (30% of total observations), Extreme Gradient Boosting, Gradient Boosting, and AdaBoost all performed very well, significantly better

than Random Forest. Table 4.4 summarized algorithms' performance which indicates that Extreme Gradient Boosting, Gradient Boosting, and AdaBoost have an accuracy of roughly 90%, which is nearly constant in the area under the curve, implying that the proportion of true positives and true negatives is consistent with accuracy. Somers' D is more than 0.8, which means that the model can tell the difference between successful and unsuccessful businesses, even when false positives and true negatives are taken into account. However, after running the model into different ensemble learning algorithms by a computer with 8 Gigabyte installed memory (RAM) and Intel(R) Celeron (R) N 2940 and 1.83 GHz characteristics, XGBoost and Gradient Boosting are faster than the Adaboost and Random Forest. In other words, XGBoost, Gradient Boosting , Adaboost and Random Forest respectively perform in 67.46 , 75.62 , 87.03 and 93.72 seconds which is a proof of the high speed of XGBoost algorithm.

Figure 4.4 illustrates the outcome of the confusion matrix and when analysing the results of the confusion matrix, the Extreme Gradient Boosting outperforms the Gradient Boosting and the AdaBoost in identifying true negatives (Predicted label = 0 and Actual label = 0), but falls short in predicting real positives (Predicted label = 1, Actual label = 1). Because AdaBoost classifies fewer startups as failures and so predicts fewer unsuccessful observations, it is less conservative than Extreme Gradient Boosting and Gradient Boosting for this model.

Table 4.4 Algorithms' performance

	Performance			
Metrics	**Accuracy**	**AUC**	**Somers' D**	**Speed**
XGBoosting	0.9168	0.942	0.8247	67.46
Random Forest	0.8164	0.8277	0.6438	63.72
Gradient Boosting	0.9057	0.9055	0.8037	45.62
AdaBoost	0.9011	0.8877	0.8017	57.03

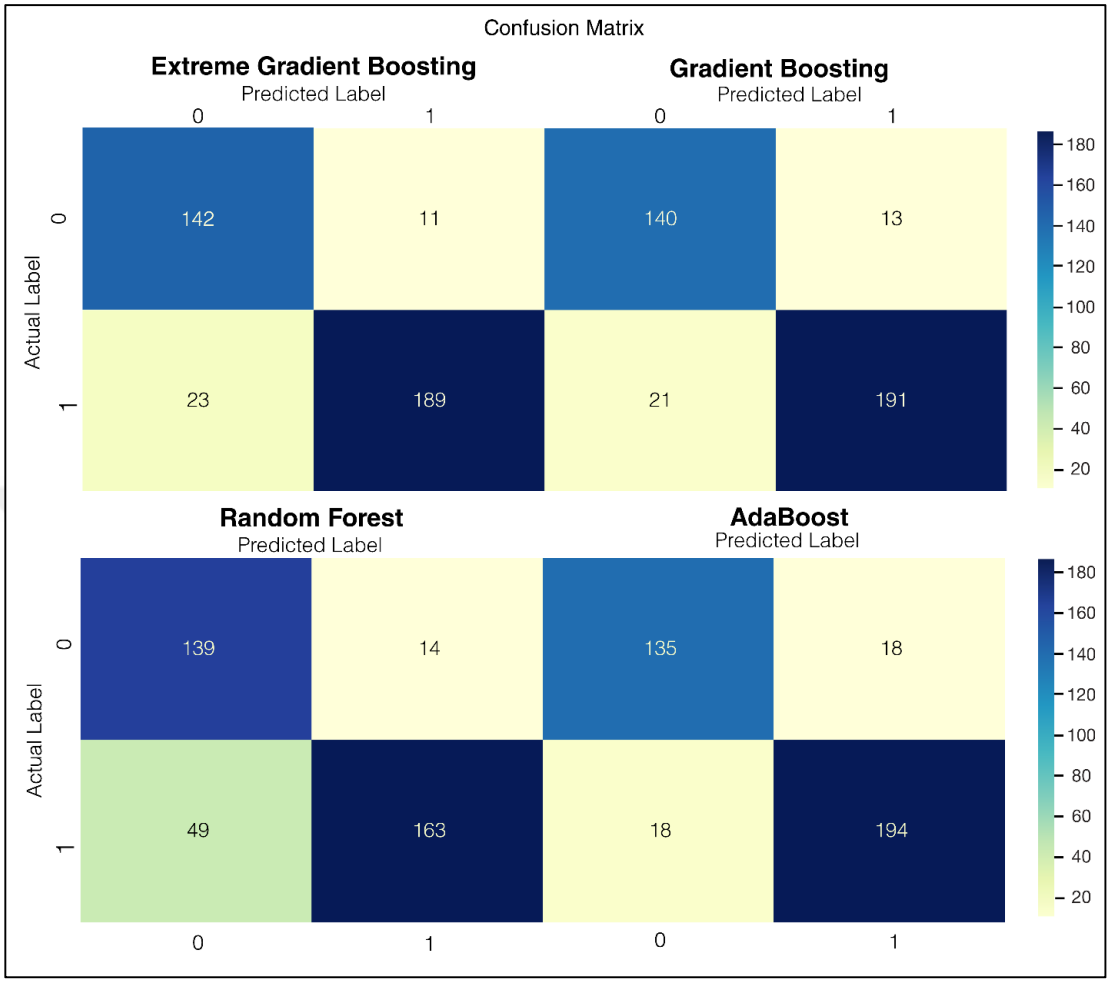

Figure 4.4 Confusion matrix

4.6.1 Sensitivity Analysis

Figure 4.5 displays box plot of AUC of each cross-validation fold in Extreme Gradient Boosting, Gradient Boosting, AdaBoost and Random Forest algorithms if fact it summarizes values between 25% and 75% of the observations. The lines inside the boxes reflect the average AUC of each cross-validation fold (described in section 4.1), and the lines above and below those boxes show the folds' greatest and lowest performances. The

goal of this investigation is to see if the models produce consistent results across the dataset; the smaller the range of those boxes, the more rational and consistent the models are.

Despite the fact that the performance of Extreme Gradient Boosting, Gradient Boosting, and AdaBoost is fairly comparable, Gradient Boosting deviates less from the mean, whereas AdaBoost has a little better average performance. These three models show acceptable sensitivity in general; the lowest AUC value of the three is over 0.86.

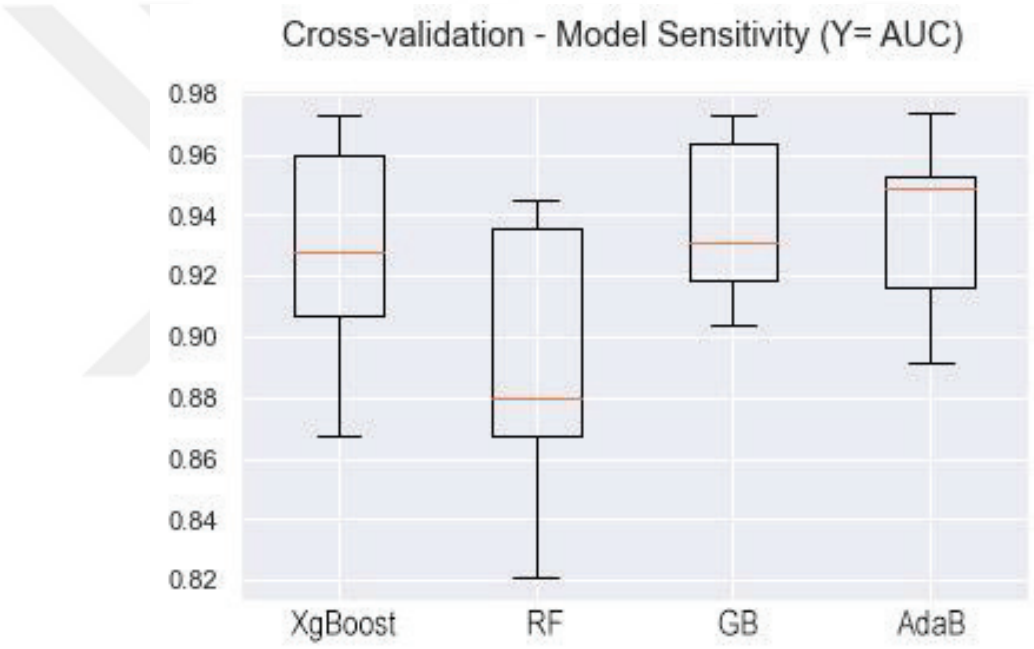

Figure 4.5 Sensitivity analysis; XGBoost: Extreme Gradient Boosting, RF: Random Forest, GB: Gradient boosting, AdaB: AdaBoost

4.6.2 Feature Importance

Random Forest doesn't work well, and AdaBoost's coefficients aren't properly specified. As a result, we focus solely on the feature value of Extreme Gradient Boosting and Gradient Boosting to increase startup success odds. Extreme Gradient Boosting gives Log Odds, so the exponent of the coefficients must be taken to see how the Odds are affected.

The coefficients were multiplied by the average non-zero value of each variable in order to create the bar chart above (in the Extreme Gradient Boosting, it has been used the exponent of the coefficient to convert Log Odds into Odds.)

Figure 4.6 represents feature importance to the odds of startup success in extreme gradient boosting and gradient boosting algorithms, according to this graph Function 2 is the most important variable in both algorithms by magnitude and Reached Venture Capital Rounds (Dummy) is the second most important variable. Gradient boosting, like all the other positive effects, is heavily reliant on Function 2. Another notable difference is that in gradient boosting, the effect of Function 1: (Education + Years from graduation)² is negative, which does not fit our sources or the economic intuition discussed in Section 4.5.1.

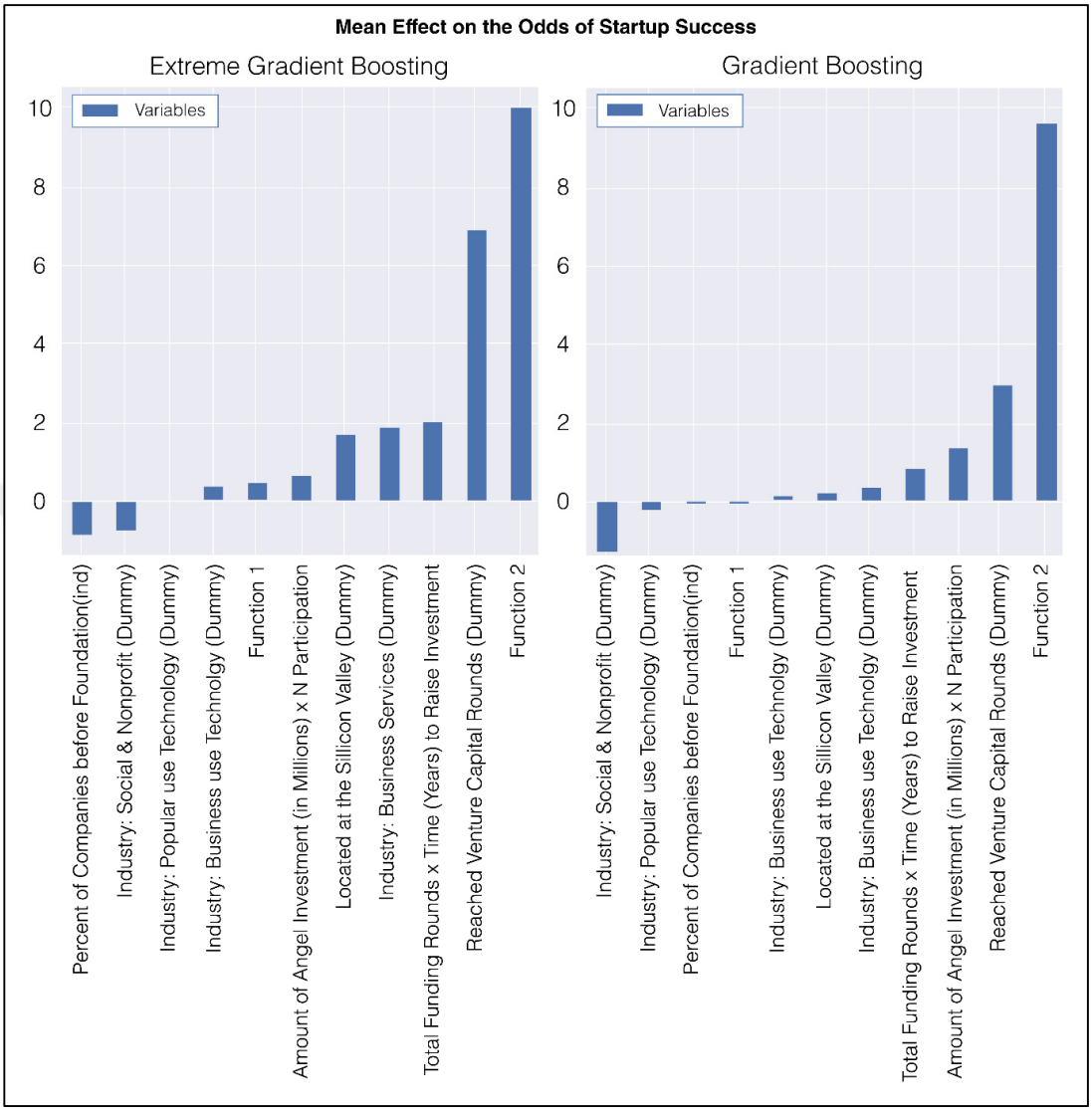

Figure 4.6 Feature importance to the odds of startup success; extreme gradient boosting and gradient boosting

4.7 Advantages of Extreme Gradient Boosting

Predictive, interpretive, cautious, and trustworthy are all qualities of a good scorecard. Because Random Forest is not particularly predictive and outperforms the other

algorithms by a substantial margin, it is not worth including in the final model. In the cross-validation investigation, AdaBoost outperformed the others by a small margin.

It is not conservative, however, because it ranks more companies as successful than the other models, despite the fact that the performance indicators are identical. It is difficult to examine the influence of each variable since it is a sort of decision tree with no well-defined coefficients. Because this method is considered a "black box" due to its lack of transparency, only exceptional performance would justify its inclusion in the final model. However, this is not the case in our study, as Extreme Gradient Boosting and Gradient Boosting have similar performance and more conservative outcomes. Gradient boosting is more transparent and predictive than AdaBoost. It is feasible to acquire a coefficient for each variable since it employs a linear method. Gradient boosting, on the other hand, does not produce statistically significant results, is strongly reliant on the variable "Function 2," and estimates a deceptive influence of education on startup success. For these reasons, it is not the thesis's winning algorithm.

As discussed in Chapter 2, Extreme Gradient Boosting provides a robust study of the coefficients and their statistical significance, as well as prediction findings. As a result, it is the most widely used approach for generating scorecards to anticipate credit defaults, employing procedures that are nearly identical to those used in this thesis. Extreme Gradient Boosting performed well in this study, accurately predicting successes and outperforming other algorithms in forecasting failures, implying that it produces more conservative outcomes. All of the Extreme Gradient Boosting coefficients corresponded to economic intuition and literature. They were determined if all of the factors were statistically significant individually or collectively. Extreme Gradient Boosting is still the safest and most reliable option for the final model, even though it is a little less stable (the range was larger in the Cross-Validation study).

Table 4.5 Predictability benchmark with other researches

Performance of different research		
Research	Best algorithm	Best AUC
Bagheri (2022)	Extreme Gradient Boosting	0.942 (Average CV)
Krishna et al. (2016)	AD Trees	0.972 (Highest CV)
Bento (2018)	Random Forest	0.932
Ünal (2019)	Extreme Gradient Boosting	0.929
Shah (2019)	Logistic Regression	0.810

Table 4.5 shows predictability benchmark with other researches all ,researches in table 4.5 used the same database (Crunchbase) for their models, but they have distinct time spans, success categories, data selection, methods, and analysis objectives. According to Krishna et al. (2016), successful startups are those that are still functioning, whereas failures are those that have closed or been purchased. The goal of this study is to use data mining to evaluate six alternative algorithms, add variables, and report performance (AUC) without examining the impacts of each variable. As the final model prediction, Krishna et al. (2016) use the highest AUC obtained throughout all cross-validation rounds. The greatest AUC obtained in cross-validation in this study is likewise about 0.97, but the average AUC is selected to compare with other models because it more precisely represents predicted predictability. According to Bento (2018), a firm is successful if it has an IPO and an M&A (Merger and Acquisition). The concentration is on data mining and testing various algorithms, with the emphasis on performance rather than varied interpretation. Bento (2018) examines the variables based on their impact on the model's performance without elaborating on the sign and magnitude of their coefficients. Successful companies are those that are either operational, acquired, or have an IPO, according to Ünal's research (2019). It exclusively evaluates failed startups that have closed their doors. Without any considerable data translation, it examines more than six distinct algorithms and variables from the financial, marketing, and industrial perspectives. Ünal (2019) uses Extreme Gradient Boosting coefficients to choose which

data to use. After choosing the final model, he then looks at how predictable different methods are.

Shah (2019) views successful startups as those that are still operating, whereas failures are those that have closed or been purchased. This study focuses on the procedures involved in creating the model using a variety of statistical and machine learning techniques. It provides some insight into the effects of each industry without looking at the other factors' coefficients.

CHAPTER 5
CONCLUSION

5.1 Summary of the Thesis

This study makes a model that uses Extreme Gradient Boosting to predict which businesses will be successful. The model is then tested with statistical and machine learning methods to see how good it is and how well it works. This thesis examines the impact of risk and success variables on startup outcomes, incorporating variable transformations where needed. Each variable, transformation, and iteration are discussed in terms of theory and economic intuition. Statistical and logical methods are used to choose the data to make sure that the model correctly identifies the dependent and independent variables. This makes it possible to understand the results and fits with economic intuition about how the independent factors affect startup success. All of the variables in the final model are statistically and economically significant. A method called "rejection inference" has been used to correct selection bias by choosing training and testing data sets that don't have any successful financing rounds.

The four models were constructed: Extreme Gradient Boosting, Random Forest, AdaBoost (decision tree enhancement), and Gradient Boosting. To analyse and compare the performance of ensemble learning methods AUC, accuracy, Sommers' D, and confusion matrix are used. Cross-validation has been used to test the sensitivity of the models, separating the data into folders and calculating the AUC obtained by fitting each iteration. The goal was to see if the model was consistent throughout the data set. In the 30% testing data set, we see that Extreme Gradient Boosting, Gradient Boosting, and AdaBoost have fairly similar performance, with AUCs around 0.942. Extreme Gradient Boosting, on the other hand, gave more stable coefficients and conservative results, as well as being less complicated and more widely used to develop predictive models.

5.2 Future Research Directions

The proposed model is consistent and easy to understand. However, there are some directions for future research. It might be interesting to create data to investigate the influence of different locations on startup success using data from other databases like "angellist and CB Insights " websites. Also, it may be possible to examine and survey about the universities and their effects on finding of new startup businesses by using relevant variables.

REFERENCES

Armour, J., & Cumming, D. (2004). *The legal road to replicating Silicon Valley*. ESRC Centre for Business Research, University of Cambridge.

Bagheri, F. & Eren Akyol, D. (2021) S-11 Using Ensemble Learning Methods for Predicting Successfulness of Startup Business. *In 1st International Congress on Artificial Intelligence and Data Science Proceeding Book (p. 223).*

Baluku, M. M., Kikooma, J. F., & Kibanja, G. M. (2016). Psychological capital and the startup capital–entrepreneurial success relationship. *Journal of Small Business & Entrepreneurship, 28*(1), 27-54.

Belgiu, M. & Drăguţ, L. (2016). Random forest in remote sensing: A review of applications and future directions. *ISPRS Journal of Photogrammetry and Remote Sensing*, *114*. 24-31.

Bentéjac, C., Csörgő, A., & Martínez-Muñoz, G. (2021). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review, 54(3),* 1937-1967.

Bento, F. R. D. S. R. (2018). *Predicting start-up success with machine learning* [Unpublished doctoral dissertation]. Universidade Nova de Lisboa.

Breiman L., Friedman J. H., Olshen R. A. & Stone C. J. (1984). *Classification and regression trees*. Chapman & Hall.

Breiman, L. (1999). Prediction games and arcing algorithms. *Neural computation, 11*(7), 1493-1517.

Breiman, L. (2001). Random forests. *Machine learning, 45*(1), 5-32.

Breiman, L., Culter, A., Liaw, A., & Wiener, M. (2002). Classification and regression by random forest. *R. News, 2*, 18-22.

Bujang, M. A., Sa'at, N., Bakar, T. M. I. T. A., & Joo, L. C. (2018). Sample size guidelines for logistic regression from observational studies with large population: emphasis on the accuracy between statistics and parameters based on real life clinical data. *The Malaysian journal of medical sciences: MJMS, 25*(4), 122.

Campbell, D., & Hulme, R. (2001). The winner-takes-all economy. *The McKinsey Quarterly,* (1), 82.

Chen T & Guestrin C. (2016). Xgboost: a scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining,* KDD'16. New York, pp 785–794.

Couronné, R., Probst, P., & Boulesteix, A. L. (2018). Random forest versus logistic regression: a large-scale benchmark experiment. *BMC bioinformatics*, *19*(1), 1-14.

Cramer, J. S. (2007). Robustness of logit analysis: Unobserved heterogeneity and mis-specified disturbances. *Oxford Bulletin of Economics and Statistics*, 69(4), 545-555.

Cribari-Neto, F., & Galvão, N. M. (2003). A class of improved heteroskedasticity-consistent covariance matrix estimators. *Communications in Statistics-Theory and Methods*, *32*(10), 1951-1980.

Cutler, D.R., Edwards, T.C., Jr., Beard, K.H., Cutler, A., Hess, K.T., Gibson, J. & Lawler, J.J. (2007), Random Forests for Classification in Ecology. *Ecology, 88*, 2783-2792.

De Ville, B. (2013). Decision trees. *Wiley Interdisciplinary Reviews: Computational Statistics*, *5*(6), 448-455.

Dickson, P. H., Solomon, G. T., & Weaver, K. M. (2008). Entrepreneurial selection and success: does education matter?. *Journal of small business and enterprise development.*

Feinleib, D. (2012). Invisible Startups. *In Why Startups Fail* (pp. 175). Apress.

Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems?. *The journal of machine learning research, 15*(1), 3133-3181.

Finkelstein, S. (2002). First-mover advantage for internet startups: Myth or reality. *Handbook of business strategy*, 2002, 39-46.

Freund, Y., & Schapire , R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 119 – 139.

Friedman J. H. (2002). Stochastic gradient boosting. *Comput. Stat. Data Anal., 38*(4), 367–378.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, *29*(5), 1189-1232.

Gelderen, M. V., Thurik, R., & Bosma, N. (2005). Success and risk factors in the pre-startup phase. *Small business economics*, *24*(4), 365-380.

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, *4*(1), 1-58.

Gimeno, J., Folta, T. B., Cooper, A. C., & Woo, C. Y. (1997). Survival of the fittest? Entrepreneurial human capital and the persistence of underperforming firms. *Administrative science quarterly*, 750-783.

Gislason, P. O., Benediktsson, J. A., & Sveinsson, J. R. (2006). Random forests for land cover classification. *Pattern recognition letters*, *27*(4), 294-300.

Graham, P. (2012). *Startup= growth. http://www. paulgraham. com/growth. html, 1.*

Guo, B., Lou, Y., & Pérez-Castrillo, D. (2015). Investment, duration, and exit strategies for corporate and independent venture capital-backed start-ups. *Journal of Economics & Management Strategy*, *24*(2), 415-455.

Hair, J. F., Black, W. C., Babin, B. J., Anderson, R. E., & Tatham, R. L. (2006). *Multivariate data analysis* (Vol. 6): Pearson Prentice Hall Upper Saddle River.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, *12*(10), 993-1001.

Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2). Springer.

Hauskrecht, M. (2019). *Decision trees.* https://people.cs.pitt.edu/milos/courses/cs2750-Spring03/lectures/class19. pdf.

Hellmann, T., & Thiele, V. (2015). Friends or foes? The interrelationship between angel and venture capital markets. *Journal of Financial Economics, 115*(3), 639-653.

Horning, N. (2010). Random Forests: An algorithm for image classification and generation of continuous fields data sets. *International Conference on Geoinformatics for Spatial Infrastructure Development in Earth and Allied Sciences 2010*, 1609-3631.

Horning, N. (2013). Introduction to decision trees and random forests. *Am. Mus. Nat. Hist*, *2*(1), 27.

Huang, B. G. (2016). *Predict startup success using network analysis and machine learning techniques.* CS224 Stanford University.

Richardson, D. (2017). *International encyclopedia of geography, 15 volume set: people, the earth, environment and technology (Vol. 1).* John Wiley & Sons.

Kenney, M., & Von Burg, U. (1999). Technology, entrepreneurship and path dependence: industrial clustering in Silicon Valley and Route 128. *Industrial and corporate change*, *8*(1), 67-103.

Kingsford, C., & Salzberg, S. L. (2008). What are decision trees?. *Nature biotechnology*, *26*(9), 1011-1013.

Kraus, A., & Kuechenhoff, H. (2014). Credit scoring optimization using the area under the curve. *The Journal of Risk Model Validation*, *8*(1), 31.

Krishna, A., Agrawal, A., & Choudhary, A. (2016). Predicting the outcome of startups: less failure, more success. *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)* (pp. 798-805).

Larivière, B., & Van den Poel, D. (2005). Predicting customer retention and profitability by using random forests and regression forests techniques. *Expert Systems with Applications*, *29*(2), 472-484.

Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R news*, *2*(3), 18-22.

Lin, Y., & Jeon, Y. (2006). Random forests and adaptive nearest neighbours. *Journal of the American Statistical Association*, *101*(474), 578-590.

Littlestone, N., & Warmuth , M. K. (1994). The weighted majority algorithm. *Information and Computation, 108*(2), 212 - 261.

Louppe, G. (2014). Understanding random forests: From theory to practice. *arXiv preprint* arXiv:1407.7502.

Lussier, R. N., & Corman, J. (1996). A business success versus failure prediction model for entrepreneurs with 0-10 employees. *Journal of Small Business Strategy*, *7*(1), 21-36.

Marmer, M., Herrmann, B. L., Dogrultan, E., Berman, R., Eesley, C., & Blank, S. (2011). Startup genome report extra: Premature scaling. *Startup genome*, 10, 1-56.

Masten, A. S. (2001). Ordinary magic: Resilience processes in development. *American psychologist, 56*(3), 227.

Mok, J. M. (2009). Reject inference in credit scoring. *Amsterdam: BMI paper.*

Pal, M. (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing, 26*(1), 217-222.

Pan, Y., & Jackson, R. T. (2008). Ethnic difference in the relationship between acute inflammation and serum ferritin in US adult males. *Epidemiology & Infection*, *136*(3), 421-431.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81-106.

Raudys, Š., & Roli, F. (2003, June). The behaviour knowledge space fusion method: Analysis of generalization error and strategies for performance improvement. n: Windeatt, T., Roli, F. (eds), *International Workshop on Multiple Classifier Systems* (pp. 55-64). Springer.

Ries, E. (2011). *The Lean Startup*. Crown Business. Rodriguez, J. D., Perez, A., & Lozano, J. A. (2009). Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE transactions on pattern analysis and machine intelligence, 32*(3), 569-575.

Roach, M., & Sauermann, H. (2015). Founder or joiner? The role of preferences and context in shaping different entrepreneurial interests. *Management Science, 61(9),* 2160-2184.

Rogerson, P. A. (2001). Data reduction: factor analysis and cluster analysis. *Statistical methods for geography*, 2001, 192-97.

Rokach, L., & Maimon, O. (2005). Decision trees. In Maimon, O., Rokach, L. (eds), *Data Mining and Knowledge Discovery Handbook*. Springer.

Sa, J. A. S., Almeida, A. C., Rocha, B. R. P., Mola, M. A. S., Souza, J. R. S. & Dentel, L. M. (2011). Lightning forecast using data mining techniques on hourly evolution of the convective available potential energy, *10th Brazilian Congress on Computational Intelligence (CB1C20II)*, Forlaleza, Brazil, pp.1-5.

Schapire, R. E. (1990). The strength of weak learnability. *Machine learning*, *5*(2), 197-227.

Shah, V. M. (2019). *Predicting the success of a startup company* [Unpublished Doctoral Dissertation]. *Oklahoma State University.*

Steigertahl, L., Mauer, R., & Say, J. B. (2018). EU Startup Monitor, 2018 Report. *ESCP Europe Jean-Baptiste Say Institute for Entrepreneurship*: Berlin, Germany, 1-34.

Sunar, F., Özkan, C., OK, A., Osmanoğlu, B., Uça Avcı, D., & Berberoğlu, S. (2017). *Dijital görüntü işleme. Eskişehir*, Turkey: TC Anadolu Üniversitesi Yayını.

Ünal, C. (2019). *Searching for a unicorn: A machine learning approach towards startup success prediction* [Unpublished Master's thesis], Humboldt-Universität zu Berlin.

Valiant, L. (1984). A theory of the learnable. *Magazine Communications of the ACM*, 1134-1142.

Valiant, L. (2013). Probably approximately correct: natureõs algorithms for learning and prospering in a complex world. *Basic Books (AZ).*

Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, *5*(2), 241-259.

Wong, A. Y. (2002). *Angel finance: the other venture capital.* Available at SSRN 941228.

Xu, L., Krzyzak, A., & Suen, C. Y. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE transactions on systems, man, and cybernetics*, *22*(3), 418-435.

Zhou, Z. H. (2012). *Ensemble methods: foundations and algorithms*. CRC press.

APPENDICES

Python Codes

```
import os

os.chdir("H:\Thesis_2\Database")
print("Current Working Directory", os.getcwd())
files = os.listdir(os.curdir)
print(files)

import pandas as pd

##Import Data##
objects  = pd.read_csv("cb_objects.csv")
acquisitions  = pd.read_csv("cb_acquisitions.csv")
funding_rounds  = pd.read_csv("cb_funding_rounds.csv")
degrees  = pd.read_csv("cb_degrees.csv")
funds  = pd.read_csv("cb_funds.csv")
investments = pd.read_csv("cb_investments.csv")
ipos  = pd.read_csv("cb_ipos.csv")
milestones  = pd.read_csv("cb_milestones.csv")
people  = pd.read_csv("cb_people.csv")
relationships  = pd.read_csv("cb_relationships.csv")
offices  = pd.read_csv("cb_offices.csv")

##Convert to data and Drops ##
##Only USA##
objects = objects[objects.country_code == 'USA']
objects = objects[objects.entity_type == 'Company']
objects['founded_at']= pd.to_datetime(objects['founded_at'])
objects['closed_at'] = pd.to_datetime(objects['closed_at'])
objects['first_funding_at'] =
pd.to_datetime(objects['first_funding_at'])
objects['last_funding_at'] = pd.to_datetime(objects['last_funding_at'])
objects['first_milestone_at'] =
pd.to_datetime(objects['first_milestone_at'])
objects['last_milestone_at'] =
pd.to_datetime(objects['last_milestone_at'])
## Founded After 2000##
objects = objects.fillna('')
objects = objects[objects.founded_at != '']
objects = objects[objects.founded_at.dt.year >= 2000]
## Closed within 5 years ##
milestones['object_id'] = milestones['object_id'].str[2:]
objects['id'] = objects['id'].str[2:]
objects.rename(columns={'id':'object_id'},inplace=True)
milestones['milestone_at']= pd.to_datetime(milestones['milestone_at'])
milestones_object = pd.merge(milestones,objects,on="object_id")

from datetime import date
```

```
from datetime import datetime, timedelta

for x in range (1, 14417):
    milestones_object.at[x,'milestone_founded'] =
(milestones_object.at[x,'milestone_at'] -
milestones_object.at[x,'founded_at']).days



milestones_object =
milestones_object[milestones_object.milestone_founded >= 1440 ]

## Create Working Environment ##
regress_df= pd.DataFrame()
regress_df= objects
regress_df=
regress_df.drop(columns=['entity_type','parent_id','normalized_name','p
ermalink','homepage_url', 'twitter_username',
'logo_url','logo_width','logo_height','short_description','description'
,'overview','overview','tag_list','created_at','updated_at'])
relationships.rename(columns={'person_object_id':'object_id'},inplace=T
rue)

## People ##
relationships['object_id'] = relationships['object_id'].str[2:]
relationships.rename(columns={'object_id':'object_id_2'},inplace=True)
relationships.rename(columns={'id':'object_id'},inplace=True)
people['object_id'] = people['object_id'].str[2:]
regress_df = pd.merge(regress_df, people,on='object_id')
regress_df= regress_df.drop(columns=['affiliation_name'])
regress_df.rename(columns={'id':'people_id'},inplace=True)

import numpy as np
degrees['object_id'] = degrees['object_id'].str[2:]

degrees_bs = degrees[degrees.degree_type == 'BS']
degrees_ba = degrees[degrees.degree_type == 'BA']
degrees_be = degrees[degrees.degree_type == 'BE']
degrees_ma = degrees[degrees.degree_type == 'MA']
degrees_ms = degrees[degrees.degree_type == 'MS']
degrees_jd = degrees[degrees.degree_type == 'JD']
degrees_mba = degrees[degrees.degree_type == 'MBA']
degrees_pdh = degrees[degrees.degree_type == 'PhD']

regress_bs = pd.merge(regress_df, degrees_bs,on='object_id')
regress_ba = pd.merge(regress_df, degrees_ba,on='object_id')
regress_be = pd.merge(regress_df, degrees_be,on='object_id')
regress_ma = pd.merge(regress_df, degrees_ma,on='object_id')
regress_ms = pd.merge(regress_df, degrees_ms,on='object_id')
regress_jd = pd.merge(regress_df, degrees_jd,on='object_id')
```

```
regress_mba = pd.merge(regress_df, degrees_mba,on='object_id')
regress_phd = pd.merge(regress_df, degrees_pdh,on='object_id')

## Diff in time ##

regress_bs['founded_at']= pd.to_datetime(regress_bs['founded_at'])
regress_bs['graduated_at']= pd.to_datetime(regress_bs['graduated_at'])
regress_bs['graduation_t'] = (regress_bs['founded_at'] -
regress_bs['graduated_at'])

regress_bs['graduat_t'] = ''

for x in range(1,2636):
      regress_bs.at[x,'graduat_t'] =
(regress_bs.at[x,'graduation_t']).days

regress_ba['founded_at']= pd.to_datetime(regress_ba['founded_at'])
regress_ba['graduated_at']= pd.to_datetime(regress_ba['graduated_at'])
regress_ba['graduation_t'] = (regress_ba['founded_at'] -
regress_ba['graduated_at'])

regress_ba['graduat_t'] = ''

for x in range(1,1762):
      regress_ba.at[x,'graduat_t'] =
(regress_ba.at[x,'graduation_t']).days

regress_be['founded_at']= pd.to_datetime(regress_be['founded_at'])
regress_be['graduated_at']= pd.to_datetime(regress_be['graduated_at'])
regress_be['graduation_t'] = (regress_be['founded_at'] -
regress_be['graduated_at'])

regress_be['graduat_t'] = ''

for x in range(1, 145):
      regress_be.at[x,'graduat_t'] =
(regress_be.at[x,'graduation_t']).days


## Diff in Time MS MA JD ##

regress_ms['founded_at']= pd.to_datetime(regress_ms['founded_at'])
regress_ms['graduated_at']= pd.to_datetime(regress_ms['graduated_at'])
regress_ms['graduation_t'] = (regress_ms['founded_at'] -
regress_ms['graduated_at'])

regress_ms['graduat_t'] = ''

for x in range(1,1325):
```

```
      regress_ms.at[x,'graduat_t'] =
(regress_ms.at[x,'graduation_t']).days

regress_ma['founded_at']= pd.to_datetime(regress_ma['founded_at'])
regress_ma['graduated_at']= pd.to_datetime(regress_ma['graduated_at'])
regress_ma['graduation_t'] = (regress_ma['founded_at'] -
regress_ma['graduated_at'])

regress_ma['graduat_t'] = ''

for x in range(1,184):
      regress_ma.at[x,'graduat_t'] =
(regress_ma.at[x,'graduation_t']).days

regress_jd['founded_at']= pd.to_datetime(regress_jd['founded_at'])
regress_jd['graduated_at']= pd.to_datetime(regress_jd['graduated_at'])
regress_jd['graduation_t'] = (regress_jd['founded_at'] -
regress_jd['graduated_at'])

regress_jd['graduat_t'] = ''

for x in range(1, 239):
      regress_jd.at[x,'graduat_t'] =
(regress_jd.at[x,'graduation_t']).days


regress_mba['founded_at']= pd.to_datetime(regress_mba['founded_at'])
regress_mba['graduated_at']=
pd.to_datetime(regress_mba['graduated_at'])
regress_mba['graduation_t'] = (regress_mba['founded_at'] -
regress_mba['graduated_at'])

regress_mba['graduat_t'] = ''

for x in range(1, 1789):
      regress_mba.at[x,'graduat_t'] =
(regress_mba.at[x,'graduation_t']).days

regress_phd['founded_at']= pd.to_datetime(regress_phd['founded_at'])
regress_phd['graduated_at']=
pd.to_datetime(regress_phd['graduated_at'])
regress_phd['graduation_t'] = (regress_phd['founded_at'] -
regress_phd['graduated_at'])

regress_phd['graduat_t'] = ''

for x in range(1, 446):
      regress_phd.at[x,'graduat_t'] =
(regress_phd.at[x,'graduation_t']).days
```

```
regress_bsba = pd.concat([regress_ba, regress_bs,regress_be])
regress_bsba = regress_bsba.drop_duplicates(['object_id'],keep='first')
regress_msma = pd.concat([regress_ma, regress_ms,regress_jd])
regress_msma = regress_msma.drop_duplicates(['object_id'],keep='first')

regress_bsba.rename(columns={'graduat_t':'timefrom_graduation'},inplace
=True)
regress_msma.rename(columns={'graduat_t':'timefrom_master'},inplace=Tru
e)
regress_mba.rename(columns={'graduat_t':'timefrom_mba'},inplace=True)
regress_phd.rename(columns={'graduat_t':'timefrom_phd'},inplace=True)

regress_bsba_msma = pd.merge(regress_bsba,
regress_msma[['object_id','degree_type','institution','subject','gradua
ted_at','timefrom_master']],on='object_id')
regress_bsba['degree_type_y'] = ''
regress_bsba['institution_y'] = ''
regress_bsba['subject_y']     = ''
regress_bsba['graduated_at_y']= ''
regress_bsba.rename(columns={'degree_type':'degree_type_x'},inplace=Tru
e)
regress_bsba.rename(columns={'institution':'institution_x'},inplace=Tru
e)
regress_bsba.rename(columns={'subject':'subject_x'},inplace=True)
regress_bsba.rename(columns={'graduated_at':'graduated_at_x'},inplace=T
rue)

regress_bsba_msma = pd.concat([regress_bsba, regress_bsba_msma])
regress_bsba_msma =
regress_bsba_msma.drop_duplicates(['object_id'],keep='last')

regress_bmm = pd.merge(regress_bsba_msma,
regress_mba[['object_id','degree_type','institution','subject','graduat
ed_at','timefrom_mba']],on='object_id')
regress_bsba_msma['degree_type'] = ''
regress_bsba_msma['institution'] = ''
regress_bsba_msma['subject']     = ''
regress_bsba_msma['graduated_at']= ''
regress_bmmba = pd.concat([regress_bsba_msma, regress_bmm])
regress_bmmba =
regress_bmmba.drop_duplicates(['object_id'],keep='last')
regress_bmmba.rename(columns={'degree_type':'degree_type_mba'},inplace=
True)
regress_bmmba.rename(columns={'institution':'institution_mba'},inplace=
True)
regress_bmmba.rename(columns={'subject':'subject_mba'},inplace=True)
regress_bmmba.rename(columns={'graduated_at':'graduated_at_mba'},inplac
e=True)
regress_bmmba.rename(columns={'degree_type_y':'degree_type_master'},inp
lace=True)
```

```
regress_bmmba.rename(columns={'institution_y':'institution_master'},inp
lace=True)
regress_bmmba.rename(columns={'subject_y':'subject_master'},inplace=Tru
e)
regress_bmmba.rename(columns={'graduated_at_y':'graduated_at_master'},i
nplace=True)

regress_phd.rename(columns={'degree_type':'degree_type_phd'},inplace=Tr
ue)
regress_phd.rename(columns={'institution':'institution_phd'},inplace=Tr
ue)
regress_phd.rename(columns={'subject':'subject_phd'},inplace=True)
regress_phd.rename(columns={'graduated_at':'graduated_at_phd'},inplace=
True)
regress_bmm_phd = pd.merge(regress_bmmba,
regress_phd[['object_id','degree_type_phd','institution_phd','subject_p
hd','graduated_at_phd','timefrom_phd']],on='object_id')

regress_bmmba['degree_type_phd'] = ''
regress_bmmba['institution_phd'] = ''
regress_bmmba['subject_phd']     = ''
regress_bmmba['graduated_at_phd']= ''
regress_finale = pd.concat([regress_bmmba, regress_bmm_phd])
regress_finale =
regress_finale.drop_duplicates(['object_id'],keep='last')

## Deleting Data Frames ##
del regress_ba
del regress_be
del regress_bmm
del regress_bmm_phd
del regress_bmmba
del regress_bs
del regress_bsba
del regress_bsba_msma
del regress_jd
del regress_ma
del regress_mba
del regress_ms
del regress_msma
del regress_phd
del degrees_ba
del degrees_be
del degrees_bs
del degrees_jd
del degrees_ma
del degrees_mba
del degrees_ms
del degrees_pdh
```

```
regress_nondegrees = pd.merge(regress_df,
degrees[['object_id','degree_type','institution','subject','graduated_a
t']],on='object_id')
regress_nondegrees =
regress_nondegrees.drop_duplicates(['object_id'],keep='last')
regress_nondegrees = pd.concat([regress_df, regress_nondegrees])
regress_nondegrees =
regress_nondegrees.drop_duplicates(['object_id'],keep=False)

regress_nondegrees.rename(columns={'degree_type':'degree_type_x'},inpla
ce=True)
regress_nondegrees.rename(columns={'institution':'institution_x'},inpla
ce=True)
regress_nondegrees.rename(columns={'subject':'subject_x'},inplace=True)
regress_nondegrees.rename(columns={'graduated_at':'graduated_at_x'},inp
lace=True)

regress_nondegrees['degree_type_master'] = ''
regress_nondegrees['institution_master'] = ''
regress_nondegrees['subject_master']     = ''
regress_nondegrees['graduated_at_master']= ''

regress_nondegrees['degree_type_mba'] = ''
regress_nondegrees['institution_mba'] = ''
regress_nondegrees['subject_mba']     = ''
regress_nondegrees['graduated_at_mba']= ''

regress_nondegrees['degree_type_phd'] = ''
regress_nondegrees['institution_phd'] = ''
regress_nondegrees['subject_phd']     = ''
regress_nondegrees['graduated_at_phd']= ''

regress_nondegrees =
regress_nondegrees[regress_nondegrees.funding_rounds != '']
regress_nondegrees =
regress_nondegrees[regress_nondegrees.funding_total_usd != '']
regress_degree_nondegree = pd.concat([regress_nondegrees,
regress_finale])
regress_degree_nondegree =
regress_degree_nondegree.drop_duplicates(['object_id'],keep='first')
regress_degree_nondegree =
regress_degree_nondegree[regress_degree_nondegree.funding_total_usd !=
'']

funding_round_angel = funding_rounds[funding_rounds.funding_round_type
== 'angel']
funding_round_angel =
funding_round_angel.drop_duplicates(['object_id'],keep='first')
funding_round_angel.rename(columns={'funded_at':'date_angel'},inplace=T
rue)
```

```
funding_round_angel.rename(columns={'raised_amount_usd':'usd_angel'},in
place=True)
funding_round_angel.rename(columns={'participants':'participants_angel'
},inplace=True)
funding_round_angel['object_id'] =
funding_round_angel['object_id'].str[2:]


funding_round_a = funding_rounds[funding_rounds.funding_round_code ==
'a']
funding_round_a =
funding_round_a.drop_duplicates(['object_id'],keep='first')
funding_round_a.rename(columns={'funded_at':'date_a'},inplace=True)
funding_round_a.rename(columns={'raised_amount_usd':'usd_a'},inplace=Tr
ue)
funding_round_a.rename(columns={'participants':'participants_a'},inplac
e=True)
funding_round_a['object_id'] = funding_round_a['object_id'].str[2:]

funding_round_b = funding_rounds[funding_rounds.funding_round_code ==
'b']
funding_round_b =
funding_round_b.drop_duplicates(['object_id'],keep='first')
funding_round_b.rename(columns={'funded_at':'date_b'},inplace=True)
funding_round_b.rename(columns={'raised_amount_usd':'usd_b'},inplace=Tr
ue)
funding_round_b.rename(columns={'participants':'participants_b'},inplac
e=True)
funding_round_b['object_id'] = funding_round_b['object_id'].str[2:]

funding_round_c = funding_rounds[funding_rounds.funding_round_code ==
'c']
funding_round_c =
funding_round_c.drop_duplicates(['object_id'],keep='first')
funding_round_c.rename(columns={'funded_at':'date_c'},inplace=True)
funding_round_c.rename(columns={'raised_amount_usd':'usd_c'},inplace=Tr
ue)
funding_round_c.rename(columns={'participants':'participants_c'},inplac
e=True)
funding_round_c['object_id'] = funding_round_c['object_id'].str[2:]

regress_degree_nondegree['date_angel'] = ''
regress_degree_nondegree['usd_angel'] = ''
regress_degree_nondegree['participants_angel']     = ''
angel = pd.merge(regress_degree_nondegree,
funding_round_angel[['object_id','date_angel','usd_angel','participants
_angel']],on='object_id')


regress_degree_nondegree['date_a'] = ''
```

```
regress_degree_nondegree['usd_a'] = ''
regress_degree_nondegree['participants_a']     = ''
angel = pd.merge(regress_degree_nondegree,
funding_round_a[['object_id','date_a','usd_a','participants_a']],on='ob
ject_id')

regress_degree_nondegree['date_b'] = ''
regress_degree_nondegree['usd_b'] = ''
regress_degree_nondegree['participants_b']     = ''
angel = pd.merge(regress_degree_nondegree,
funding_round_b[['object_id','date_b','usd_b','participants_b']],on='ob
ject_id')

regress_degree_nondegree['date_c'] = ''
regress_degree_nondegree['usd_c'] = ''
regress_degree_nondegree['participants_c']     = ''
angel = pd.merge(regress_degree_nondegree,
funding_round_c[['object_id','date_c','usd_c','participants_c']],on='ob
ject_id')

regress_degree_nondegree['date_angel'] = ''
regress_degree_nondegree['usd_angel'] = ''
regress_degree_nondegree['participants_angel']     = ''
angel = pd.merge(regress_degree_nondegree,
funding_round_angel[['object_id','date_angel','usd_angel','participants
_angel']],on='object_id')

regress_degree_nondegree['date_a'] = ''
regress_degree_nondegree['usd_a'] = ''
regress_degree_nondegree['participants_a']     = ''
a = pd.merge(regress_degree_nondegree,
funding_round_a[['object_id','date_a','usd_a','participants_a']],on='ob
ject_id')

regress_degree_nondegree['date_b'] = ''
regress_degree_nondegree['usd_b'] = ''
regress_degree_nondegree['participants_b']     = ''
b = pd.merge(regress_degree_nondegree,
funding_round_b[['object_id','date_b','usd_b','participants_b']],on='ob
ject_id')

regress_degree_nondegree['date_c'] = ''
regress_degree_nondegree['usd_c'] = ''
regress_degree_nondegree['participants_c']     = ''
c = pd.merge(regress_degree_nondegree,
funding_round_c[['object_id','date_c','usd_c','participants_c']],on='ob
ject_id')


a.rename(columns={'date_a_y':'date_a'},inplace=True)
```

```
a.rename(columns={'usd_a_y':'usd_a'},inplace=True)
a.rename(columns={'participants_a_y':'participants_a'},inplace=True)
a = a.drop(["date_a_x", "usd_a_x", "participants_a_x"], axis=1)

b.rename(columns={'date_b_y':'date_b'},inplace=True)
b.rename(columns={'usd_b_y':'usd_b'},inplace=True)
b.rename(columns={'participants_b_y':'participants_b'},inplace=True)
b = b.drop(["date_b_x", "usd_b_x", "participants_b_x"], axis=1)

c.rename(columns={'date_c_y':'date_c'},inplace=True)
c.rename(columns={'usd_c_y':'usd_c'},inplace=True)
c.rename(columns={'participants_c_y':'participants_c'},inplace=True)
c = c.drop(["date_c_x", "usd_c_x", "participants_c_x"], axis=1)

angel.rename(columns={'date_angel_y':'date_angel'},inplace=True)
angel.rename(columns={'usd_angel_y':'usd_angel'},inplace=True)
angel.rename(columns={'participants_angel_y':'participants_angel'},inpl
ace=True)
angel = angel.drop(["date_angel_x", "usd_angel_x",
"participants_angel_x"], axis=1)


angel['date_angel'] = pd.to_datetime(angel['date_angel'])
angel['first_funding_at'] = pd.to_datetime(angel['date_angel'])
angel['timetoraise_angel']= ''
for x in range(1, 3194):
      angel.at[x,'timetoraise_angel'] = (angel.at[x,'first_funding_at']
- angel.at[x,'founded_at']).days

regress_degree_nondegree['timetoraise_angel'] = ''

angel_a = pd.merge(angel,
a[['object_id','date_a','usd_a','participants_a']],on='object_id')
angel_ab =
pd.merge(angel_a,b[['object_id','date_b','usd_b','participants_b']],on=
'object_id')
angel_abc =
pd.merge(angel_ab,c[['object_id','date_c','usd_c','participants_c']],on
='object_id')

regress_degree_nondegree = pd.concat([regress_degree_nondegree , angel,
angel_ab, angel_abc])
regress_degree_nondegree =
regress_degree_nondegree.drop_duplicates(['object_id'],keep='last')

regress_degree_nondegree =  regress_degree_nondegree.drop(['usd_a_x',
'usd_a','usd_b_x','usd_b','usd_c_x','usd_c','updated_at','participants_
b_x','participants_b','participants_a_x','participants_a','participants
_c_x','participants_c','date_a_x','date_a','date_b','date_b_y','date_c_
x','date_c'], axis = 1)
```

```
regress_degree_nondegree['dummy'] = ''

regress_degree_nondegree =
regress_degree_nondegree.reset_index(drop=True)

regress_degree_nondegree.closed_at.fillna(0, inplace=True)

regress_degree_nondegree['closed_at']=
pd.to_datetime(regress_degree_nondegree['closed_at'])

regress_degree_nondegree['founded_at']=
pd.to_datetime(regress_degree_nondegree['founded_at'])

regress_degree_nondegree['timefrom_closed']  =
regress_degree_nondegree['closed_at'] -
regress_degree_nondegree['founded_at']
regress_degree_nondegree.timefrom_closed.fillna(0, inplace=True)

print(regress_degree_nondegree.at[1,'timefrom_closed'].days)

regress_degree_nondegree['closed_founded'] = ''

for x in range (0,9445):
    regress_degree_nondegree.at[x,'closed_founded'] =
(regress_degree_nondegree.at[x,'timefrom_closed']).days


regress_degree_nondegree.dummy[regress_degree_nondegree['status'] ==
'operating'] = 1
regress_degree_nondegree.dummy[regress_degree_nondegree['status'] ==
'ipo'] = 1
regress_degree_nondegree.dummy[regress_degree_nondegree['status'] ==
'acquired'] = 1
regress_degree_nondegree.dummy[regress_degree_nondegree['status'] ==
'closed' ] = 0

print(regress_degree_nondegree.at[1,'category_code'])

regress_degree_nondegree = regress_degree_nondegree.reset_index()

regress_degree_nondegree['founded_year'] =
pd.to_datetime(regress_degree_nondegree['founded_at']).dt.to_period('Y'
)
regress_df['founded_year'] =
pd.to_datetime(regress_df['founded_at']).dt.to_period('Y')

for x in range (0,9445):
 a = regress_df[regress_df['category_code'] ==
regress_degree_nondegree.at[x,'category_code']]
```

```
 regress_degree_nondegree.at[x,'competition_before'] =
a.city[a['founded_year']<regress_degree_nondegree.at[x,'founded_year']]
.count()
 del a

for x in range (0,9445):
 a = regress_df[regress_df['category_code'] ==
regress_degree_nondegree.at[x,'category_code']]
 regress_degree_nondegree.at[x,'competition_foundation'] =
a.city[a['founded_year']==regress_degree_nondegree.at[x,'founded_year']
].count()
 del a


for x in range (0,9445):
 regress_degree_nondegree.at[x,'percent_comp_foundation'] =
(regress_degree_nondegree.at[x,'competition_foundation']/
(regress_degree_nondegree.at[x,'competition_before']+regress_degree_non
degree.at[x,'competition_foundation']))*100
for x in range (0,9445):
 regress_degree_nondegree.at[x,'percent_comp_before'] =
(regress_degree_nondegree.at[x,'competition_before']/
(regress_degree_nondegree.at[x,'competition_before']+regress_degree_non
degree.at[x,'competition_foundation']))*100


regress_degree_nondegree =
regress_degree_nondegree.drop(['competition_before',
'competition_foundation'], axis=1)

regress_degree_nondegree['ba_top'] = 0
regress_degree_nondegree.ba_top[regress_degree_nondegree['institution_x
'] == 'Yale University'] = 1
regress_degree_nondegree.ba_top[regress_degree_nondegree['institution_x
'] == 'Harvard University'] = 1
regress_degree_nondegree.ba_top[regress_degree_nondegree['institution_x
'] == 'Columbia University'] = 1
regress_degree_nondegree.ba_top[regress_degree_nondegree['institution_x
'] == 'Massachusetts Institute of Technology (MIT)'] = 1
regress_degree_nondegree.ba_top[regress_degree_nondegree['institution_x
'] == 'MIT'] = 1
regress_degree_nondegree.ba_top[regress_degree_nondegree['institution_x
'] == 'Massachusetts Institute of Technology'] = 1
regress_degree_nondegree.ba_top[regress_degree_nondegree['institution_x
'] == 'Stanford University'] = 1
regress_degree_nondegree.ba_top[regress_degree_nondegree['institution_x
'] == 'University of Chicago'] = 1
regress_degree_nondegree.ba_top[regress_degree_nondegree['institution_x
'] == 'University of Pennsylvania'] = 1
```

```
regress_degree_nondegree.ba_top[regress_degree_nondegree['institution_x
'] == 'Northwestern University'] = 1
regress_degree_nondegree.ba_top[regress_degree_nondegree['institution_x
'] == 'Duke University'] = 1
regress_degree_nondegree.ba_top[regress_degree_nondegree['institution_x
'] == 'Johns Hopkins University'] = 1


regress_degree_nondegree['ba_dummy'] = ''
regress_degree_nondegree.subject_x.fillna(0, inplace=True)
regress_degree_nondegree.institution_x.fillna(0, inplace=True)

for x in range(0,9445):
 regress_degree_nondegree.at[x,'ba_dummy'] =
isinstance(regress_degree_nondegree.at[x,'institution_x'], str)

regress_degree_nondegree.ba_dummy[regress_degree_nondegree['ba_dummy']
== False] = 0
regress_degree_nondegree.ba_dummy[regress_degree_nondegree['ba_dummy']
!= False] = 1


regress_degree_nondegree['ma_dummy'] = ''
regress_degree_nondegree.institution_master[regress_degree_nondegree['i
nstitution_master'] == ''] = 0

for x in range(0,9445):
 regress_degree_nondegree.at[x,'ma_dummy'] =
isinstance(regress_degree_nondegree.at[x,'institution_master'], str)

regress_degree_nondegree.ma_dummy[regress_degree_nondegree['ma_dummy']
!= False] = 1
regress_degree_nondegree.ma_dummy[regress_degree_nondegree['ma_dummy']
== False] = 0


regress_degree_nondegree['mba_dummy'] = ''
regress_degree_nondegree.institution_mba[regress_degree_nondegree['inst
itution_mba'] == ''] = 0

for x in range(0,9445):
 regress_degree_nondegree.at[x,'mba_dummy'] =
isinstance(regress_degree_nondegree.at[x,'institution_mba'], str)

regress_degree_nondegree.mba_dummy[regress_degree_nondegree['mba_dummy'
] != False] = 1
regress_degree_nondegree.mba_dummy[regress_degree_nondegree['mba_dummy'
] == False] = 0

regress_degree_nondegree['phd_dummy'] = ''
```

```
regress_degree_nondegree.institution_phd[regress_degree_nondegree['inst
itution_phd'] == ''] = 0

for x in range(0,9445):
 regress_degree_nondegree.at[x,'phd_dummy'] =
isinstance(regress_degree_nondegree.at[x,'institution_phd'], str)

regress_degree_nondegree.phd_dummy[regress_degree_nondegree['phd_dummy'
] != False] = 1
regress_degree_nondegree.phd_dummy[regress_degree_nondegree['phd_dummy'
] == False] = 0

regress_degree_nondegree
=regress_degree_nondegree.drop(columns=['birthplace','country_code','cr
eated_by','date_a_y','date_angel','date_b_x','date_c_y','domain','entit
y_id','first_funding_at','first_investment_at','first_name','graduated_
at_master','graduated_at_phd','graduated_at_x','graduation_t','id','ins
titution_master','institution_mba','institution_phd','institution_x','i
nvested_companies','last_name','name','participants_a_y','participants_
b_y','participants_b_y','participants_c_y','people_id','timefrom_master
','timefrom_phd','timefrom_mba','timefrom_phd'])

regress_degree_nondegree.timefrom_graduation.fillna(0, inplace=True)

regress_degree_nondegree.timefrom_graduation[regress_degree_nondegree['
timefrom_graduation'] == ''] = 0
for x in range (0,9445):
 regress_degree_nondegree.at[x,'timefrom_graduation'] =
int(regress_degree_nondegree.at[x,'timefrom_graduation'])/365

regress_degree_nondegree =
regress_degree_nondegree[regress_degree_nondegree['timetoraise_angel']
!= '']

regress_degree_nondegree = regress_degree_nondegree.reset_index()

regress_degree_nondegree.timetoraise_angel.fillna(0, inplace=True)

for x in range (0,3193):
 regress_degree_nondegree.at[x,'timetoraise_angel'] =
int(regress_degree_nondegree.at[x,'timetoraise_angel'])/30


regress_degree_nondegree.usd_angel.fillna(0, inplace=True)

regress_degree_nondegree =
regress_degree_nondegree.reset_index(drop=True)

for x in range(0,3193):
```

```
 regress_degree_nondegree.at[x,'usd_angel'] =
int(regress_degree_nondegree.at[x,'usd_angel'])

regress_degree_nondegree.usd_a_y.fillna(0, inplace=True)
regress_degree_nondegree.usd_b_y.fillna(0, inplace=True)
regress_degree_nondegree.usd_c_y.fillna(0, inplace=True)

for x in range(21,3193):
 regress_degree_nondegree.at[x,'usd_a_y'] =
int(regress_degree_nondegree.at[x,'usd_a_y'])
for x in range(21,3193):
 regress_degree_nondegree.at[x,'usd_b_y'] =
int(regress_degree_nondegree.at[x,'usd_b_y'])
for x in range(21,3193):
 regress_degree_nondegree.at[x,'usd_c_y'] =
int(regress_degree_nondegree.at[x,'usd_c_y'])
for x in range(21,3193):
 regress_degree_nondegree.at[x,'funding_rounds'] =
int(regress_degree_nondegree.at[x,'funding_rounds'])


regress_degree_nondegree =
regress_degree_nondegree.reset_index(drop=True)

regress_degree_nondegree['first_milestone_at']=
pd.to_datetime(regress_degree_nondegree['first_milestone_at'])
regress_degree_nondegree['last_milestone_at']=
pd.to_datetime(regress_degree_nondegree['last_milestone_at'])


for x in range(0,3193):
 regress_degree_nondegree.at[x,'timefromfirst_milestone'] =
(regress_degree_nondegree.at[x,'first_milestone_at'] -
regress_degree_nondegree.at[x,'founded_at']).days

regress_degree_nondegree.timefromfirst_milestone.fillna(0,
inplace=True)

for x in range(0,3193):
 regress_degree_nondegree.at[x,'timefromlast_milestone'] =
(regress_degree_nondegree.at[x,'last_milestone_at'] -
regress_degree_nondegree.at[x,'founded_at']).days

regress_degree_nondegree.timefromlast_milestone.fillna(0, inplace=True)


regress_degree_nondegree['zumbi'] = 0
regress_degree_nondegree['zumbi'][regress_degree_nondegree['status'] ==
'operating'] = 1
regress_degree_nondegree['zumbi_2'] = 0
```

```
regress_degree_nondegree['zumbi_2'][regress_degree_nondegree['timefroml
ast_milestone'] < 1180] = 1
regress_degree_nondegree['zumbi_3'] = 0
regress_degree_nondegree['zumbi_3'] = regress_degree_nondegree['zumbi']
+ regress_degree_nondegree['zumbi_2']
regress_degree_nondegree =
regress_degree_nondegree[regress_degree_nondegree['zumbi_3'] < 2]

regress_degree_nondegree['timefromfirst_milestone'] =
regress_degree_nondegree['timefromfirst_milestone']/30

regress_degree_nondegree.timefromfirst_milestone[regress_degree_nondegr
ee['milestones']==0] = 60
regress_degree_nondegree['Venture_capital']=0
regress_degree_nondegree.Venture_capital[regress_degree_nondegree.usd_a
_y != 0] = 1
regress_degree_nondegree.Venture_capital[regress_degree_nondegree.usd_b
_y != 0] = 1
regress_degree_nondegree.Venture_capital[regress_degree_nondegree.usd_c
_y != 0] = 1

regress_degree_nondegree['usd_vc'] = 0
regress_degree_nondegree.usd_vc = regress_degree_nondegree.usd_a_y +
regress_degree_nondegree.usd_b_y + regress_degree_nondegree.usd_c_y
regress_degree_nondegree['log_usd_vc'] = 0
regress_degree_nondegree['log_usd_vc'] =
np.log(regress_degree_nondegree['usd_vc'])

regress_degree_nondegree
=regress_degree_nondegree.drop(columns=['zumbi','zumbi_2','zumbi_3','us
d_a_y','usd_b_y','usd_c_y','degree_type_x','degree_type_master','degree
_type_phd','subject_master','subject_mba','subject_phd','subject_x'])


regress_degree_nondegree['zumbi1'] = 0
regress_degree_nondegree['zumbi2'] = 0
regress_degree_nondegree['zumbi3'] = 0
regress_degree_nondegree.zumbi1[regress_degree_nondegree['usd_angel']
>= 0]=1
regress_degree_nondegree.zumbi2[regress_degree_nondegree['usd_vc'] >=
0]=1
regress_degree_nondegree.zumbi3 = regress_degree_nondegree.zumbi1 +
regress_degree_nondegree.zumbi2
regress_degree_nondegree =
regress_degree_nondegree[regress_degree_nondegree.zumbi3 >= 1]

regress_degree_nondegree
=regress_degree_nondegree.drop(columns=['usd_vc','zumbi1','zumbi2','zum
bi3'])
```

```
gdp  = pd.read_csv("gdp.csv")

del acquisitions
del angel
del angel_a
del angel_ab
del angel_abc
del b
del c
del degrees
del files
del funding_round_a
del funding_round_b
del funding_round_c
del funding_rounds
del funding_round_angel
del funds
del investments
del ipos
del milestones
del milestones_object
del objects
del people
del regress_finale
del regress_nondegrees
del relationships
df = regress_degree_nondegree
del regress_degree_nondegree

df["gdp_foundation"] = 0
df.gdp_foundation[df['founded_year']==2000] = 1
df.gdp_foundation[df['founded_year']==2001] = 1.7
df.gdp_foundation[df['founded_year']==2002] = 2.9
df.gdp_foundation[df['founded_year']==2003] = 3.8
df.gdp_foundation[df['founded_year']==2004] = 3.5
df.gdp_foundation[df['founded_year']==2005] = 2.9
df.gdp_foundation[df['founded_year']==2006] = 1.9
df.gdp_foundation[df['founded_year']==2007] = -0.1
df.gdp_foundation[df['founded_year']==2008] = -2.5
df.gdp_foundation[df['founded_year']==2009] = 2.6
df.gdp_foundation[df['founded_year']==2010] = 1.6
df.gdp_foundation[df['founded_year']==2011] = 2.2
df.gdp_foundation[df['founded_year']==2012] = 1.8

vix  = pd.read_csv("vix.csv")

df["vix_foundation"] = 0
df.vix_foundation[df['founded_year']==2000] = 26.85
df.vix_foundation[df['founded_year']==2001] = 23.8
df.vix_foundation[df['founded_year']==2002] = 28.62
```

```
df.vix_foundation[df['founded_year']==2003] = 18.31
df.vix_foundation[df['founded_year']==2004] = 13.29
df.vix_foundation[df['founded_year']==2005] = 12.07
df.vix_foundation[df['founded_year']==2006] = 11.56
df.vix_foundation[df['founded_year']==2007] = 22.5
df.vix_foundation[df['founded_year']==2008] = 40
df.vix_foundation[df['founded_year']==2009] = 21.68
df.vix_foundation[df['founded_year']==2010] = 17.75
df.vix_foundation[df['founded_year']==2011] = 23.4
df.vix_foundation[df['founded_year']==2012] = 18.02

cbp_total = pd.read_csv('cbp_90_2012.txt',sep=',')

df.category_code[df['category_code']=='biotech']='bio_clean_tech'
df.category_code[df['category_code']=='cleantech']='bio_clean_tech'
df.category_code[df['category_code']=='bio_tech']='bio_clean_tech'
df.category_code[df['category_code']=='clean_tech']='bio_clean_tech'
df.category_code[df['category_code']=='consulting']='Business_Services'
df.category_code[df['category_code']=='analytics']='Business_Services'
df.category_code[df['category_code']=='games_video']='game_video'

## Counting Business Patterns ##
df['cbp_ind_10'] = 0
df['cbp_ind_2'] = 0

cbp_total.diff_est.fillna(0, inplace=True)
cbp_total.diff_ap.fillna(0, inplace=True)

df = df.reset_index(drop=True)

list = cbp_total['industry']
list= list.drop_duplicates(keep='last')

for i in list:
 for x in range (2000,2013):
  df.cbp_ind_10[(df['founded_year']==x)&(df['category_code']==i)] =
(float(cbp_total.diff_ap[(cbp_total['year']==x-
1)&(cbp_total['industry']==i)])+float(cbp_total.diff_ap[(cbp_total['yea
r']==x-
2)&(cbp_total['industry']==i)])+float(cbp_total.diff_ap[(cbp_total['yea
r']==x-
3)&(cbp_total['industry']==i)])+float(cbp_total.diff_ap[(cbp_total['yea
r']==x-
4)&(cbp_total['industry']==i)])+float(cbp_total.diff_ap[(cbp_total['yea
r']==x-
5)&(cbp_total['industry']==i)])+float(cbp_total.diff_ap[(cbp_total['yea
r']==x-
6)&(cbp_total['industry']==i)])+float(cbp_total.diff_ap[(cbp_total['yea
r']==x-
7)&(cbp_total['industry']==i)])+float(cbp_total.diff_ap[(cbp_total['yea
```

```
r']==x-
8)&(cbp_total['industry']==i)])+float(cbp_total.diff_ap[(cbp_total['yea
r']==x-
9)&(cbp_total['industry']==i)])+float(cbp_total.diff_ap[(cbp_total['yea
r']==x-10)&(cbp_total['industry']==i)]))/10

for i in list:
 for x in range (2000,2013):
  df.cbp_ind_2[(df['founded_year']==x)&(df['category_code']==i)] =
(float(cbp_total.diff_ap[(cbp_total['year']==x-
1)&(cbp_total['industry']==i)]))

df.cbp_ind_10= pd.to_numeric(df.cbp_ind_10)
df.cbp_ind_2= pd.to_numeric(df.cbp_ind_2)

df['cbp_yield_2_10'] =  df.cbp_ind_2 - df.cbp_ind_10

df['dummy_ind'] = 0

df.dummy_ind[(df['cbp_yield_2_10'] >= 0)] = 1

df['ba_dummy'] = pd.to_numeric(df['ba_dummy'] )
df['ma_dummy'] = pd.to_numeric(df['ma_dummy'] )
df['mba_dummy'] = pd.to_numeric(df['mba_dummy'] )
df['phd_dummy'] = pd.to_numeric(df['phd_dummy'] )

df['high_education'] = 0
df.high_education[(df['ba_dummy']==1)]  = 1
df.high_education[(df['mba_dummy']==1)] = 2
df.high_education[(df['ma_dummy']==1)]  = 3
df.high_education[df['phd_dummy']==1]   = 4

df['educ_exp'] = df['high_education'] * df.timefrom_graduation

df['milestones'] = pd.to_numeric(df['milestones'] )
df['milestones_decrease'] = 0
df.milestones_decrease[df['milestones']>3] = 1
df.milestones_decrease[df['milestones']>4] = 2
df.milestones_decrease[df['milestones']>5] = 3
df.milestones_decrease[df['milestones']>6] = 4




## Regions - Divisions ##
#Northeast#
## New England ##
df.state_code[df['state_code']=='ME']='New_England'
df.state_code[df['state_code']=='NH']='New_England'
df.state_code[df['state_code']=='VT']='New_England'
```

```
df.state_code[df['state_code']=='MA']='New_England'
df.state_code[df['state_code']=='RI']='New_England'
df.state_code[df['state_code']=='CT']='New_England'
df.state_code[df['state_code']=='NY']='Middle_Atlantic'
df.state_code[df['state_code']=='PA']='Middle_Atlantic'
df.state_code[df['state_code']=='NJ']='Middle_Atlantic'
#Midwest#
df.state_code[df['state_code']=='WI']='East_North_Central'
df.state_code[df['state_code']=='MI']='East_North_Central'
df.state_code[df['state_code']=='IL']='East_North_Central'
df.state_code[df['state_code']=='IN']='East_North_Central'
df.state_code[df['state_code']=='OH']='East_North_Central'
df.state_code[df['state_code']=='ND']='West_North_Central'
df.state_code[df['state_code']=='SD']='West_North_Central'
df.state_code[df['state_code']=='KS']='West_North_Central'
df.state_code[df['state_code']=='MN']='West_North_Central'
df.state_code[df['state_code']=='IA']='West_North_Central'
df.state_code[df['state_code']=='MO']='West_North_Central'
df.state_code[df['state_code']=='NE']='West_North_Central'
#South#
df.state_code[df['state_code']=='DE']='South_Atlantic'
df.state_code[df['state_code']=='MD']='South_Atlantic'
df.state_code[df['state_code']=='VA']='South_Atlantic'
df.state_code[df['state_code']=='WV']='South_Atlantic'
df.state_code[df['state_code']=='KY']='East_South_Central'
df.state_code[df['state_code']=='NC']='South_Atlantic'
df.state_code[df['state_code']=='SC']='South_Atlantic'
df.state_code[df['state_code']=='DC']='South_Atlantic'
df.state_code[df['state_code']=='GA']='South_Atlantic'
df.state_code[df['state_code']=='FL']='South_Atlantic'
df.state_code[df['state_code']=='TN']='East_South_Central'
df.state_code[df['state_code']=='MS']='East_South_Central'
df.state_code[df['state_code']=='AL']='East_South_Central'
df.state_code[df['state_code']=='OK']='West_South_Central'
df.state_code[df['state_code']=='TX']='West_South_Central'
df.state_code[df['state_code']=='AR']='West_South_Central'
df.state_code[df['state_code']=='LA']='West_South_Central'
#West#
df.state_code[df['state_code']=='ID']='Mountain'
df.state_code[df['state_code']=='MT']='Mountain'
df.state_code[df['state_code']=='WY']='Mountain'
df.state_code[df['state_code']=='NV']='Mountain'
df.state_code[df['state_code']=='UT']='Mountain'
df.state_code[df['state_code']=='CO']='Mountain'
df.state_code[df['state_code']=='AR']='Mountain'
df.state_code[df['state_code']=='NM']='Mountain'
df.state_code[df['state_code']=='AK']='Pacific'
df.state_code[df['state_code']=='WA']='Pacific'
df.state_code[df['state_code']=='OR']='Pacific'
df.state_code[df['state_code']=='CA']='Pacific'
```

```
df.state_code[df['state_code']=='HI']='Pacific'
df['ca'] = 0
df.ca[df['state_code'] == 'CA'] = 1
df['ny'] = 0
df.ny[df['state_code'] == 'NY'] = 1
df['texas'] = 0
df.texas[df['state_code'] == 'TX'] = 1
df['fl'] = 0
df.fl[df['state_code'] == 'FL'] = 1

df['wa'] = 0
df.wa[df['state_code'] == 'WA'] = 1

##Categories##
df.category_code[df['category_code']=='messaging']='mobile'
df.category_code[df['category_code']=='search']='web'
df.category_code[df['category_code']=='public_relations']='other_servic
es'
df.category_code[df['category_code']=='travel']='other_services'
df.category_code[df['category_code']=='education']='other_services'
df.category_code[df['category_code']=='security']='other_services'
df.category_code[df['category_code']=='legal']='other_services'
df.category_code[df['category_code']=='hospitality']='other_services'
df.category_code[df['category_code']=='enterprise']='Business_Services'

df_1 = df
df_1['relationships'] = pd.to_numeric(df_1['relationships'])
df_1.relationships[df_1['relationships']==''] = 0

df_1['sf_bay'] = 0
df_1.sf_bay[df_1['region'] == 'SF Bay'] = 1

df_1 =
df_1.drop(columns=['high_education','dummy_ind','gdp_foundation','vix_f
oundation','ca','ny','texas','fl','wa','ba_top',
'cbp_ind_10','cbp_ind_2','cbp_yield_2_10','percent_comp_foundation','le
vel_0','index','city','closed_at','created_at','degree_type_mba','first
_milestone_at','founded_at','graduated_at_mba','investment_rounds','las
t_funding_at','last_investment_at','last_milestone_at','object_id','reg
ion','status'])
df1 = df_1

df1.fillna(0, inplace=True)

from sklearn.datasets import make_classification
from matplotlib import pyplot as plt
import xgboost as xgb
import seaborn as sns
sns.set()
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import confusion_matrix
import pandas as pd
from sklearn.feature_selection import RFE
import statsmodels.api as sm


df1 = df1.drop(columns=['founded_year'])


df1.timefromfirst_milestone[df1['milestones']==0]=60
df1['milestones'] = pd.to_numeric(df1['milestones'] )
df1['timefrom_graduation'] = pd.to_numeric(df1['timefrom_graduation'] )
df1.milestones.fillna(0, inplace=True)
df1['dummy'] = pd.to_numeric(df1['dummy'] )
df1['funding_rounds'] = pd.to_numeric(df1['funding_rounds'] )
dummies = pd.get_dummies(df['state_code'])
dummies_1 = pd.get_dummies(df['category_code'])
df1 = pd.concat([df1,dummies_1,dummies],axis=1)
df1 = df1.drop(columns=[''])
df1 = df1.drop(columns=['state_code'])
df1 =
df1.drop(columns=['East_North_Central','AZ','category_code','other','fu
nding_total_usd','nonprofit'])
df1 = df1.dropna()
y = df1['dummy']
df1 = df1.drop(columns=['dummy','timefromlast_milestone'])
df1 =
df1.drop(columns=['transportation','sports','semiconductor','real_estat
e','photo_video','news','health','fashion','automotive','music','hardwa
re','finance','network_hosting','software'])
df1 = df1.dropna()
df1['log_usd_angel'] = np.log(df1['usd_angel'])
df1 = df1.drop(columns=['closed_founded','timefrom_closed'])
df1 =
df1.drop(columns=['funding_rounds','usd_angel','ba_dummy','ma_dummy','p
hd_dummy','mba_dummy','timefrom_graduation','other_services','advertisi
ng','local','manufacturing','medical','design','East_South_Central','Mi
ddle_Atlantic','Mountain','New_England','Pacific','West_North_Central',
'West_South_Central'])
df1['small_team'] = 0
df1.small_team[df1['relationships'] < 3] = 1
df1 = df1.drop(columns=['relationships'])
df1 = df1.drop(columns=['Venture_capital'])

from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from sklearn import metrics

df1.log_usd_angel[df1['log_usd_angel']<0] = 0
df1.log_usd_vc[df1['log_usd_vc']<0] = 0
```

```
xTrain, xTest, yTrain, yTest = train_test_split(df1, y, test_size =
0.3, random_state = 0)



a = xgb.fit(xTrain,yTrain)

y_pred = a.predict(xTest)




cnf_matrix= confusion_matrix(yTest, y_pred)

# Vizualizing Confusion Matrix #

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error
class_names=[0,1]
# name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu"
,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

#Confusion Matrix Evaluation Metrics#
print("Accuracy:",metrics.accuracy_score(yTest, y_pred))
print("Precision:",metrics.precision_score(yTest, y_pred))
print("Recall:",metrics.recall_score(yTest, y_pred))

#ROC Curve#
yTrain = pd.to_numeric(yTrain )
y_pred_proba =xgb.predict_proba(xTest)[::,1]
fpr, tpr, _ = metrics.roc_curve(yTest,  y_pred_proba)
auc_logit = metrics.roc_auc_score(yTest, y_pred_proba)
rmse_logit = np.sqrt(mean_squared_error(yTest, y_pred_proba))
plt.plot(fpr,tpr,label="data 1, auc="+str(auc_logit))
plt.legend(loc=4)
plt.show()
```

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve

logit_roc_auc = roc_auc_score(yTest, xgb.predict(xTest))
fpr, tpr, thresholds = roc_curve(yTest, xgb.predict_proba(xTest)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' %
logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
import statsmodels.api as sm
logit_model=sm.Logit(y.astype(float),df1.astype(float)).fit()
#logit_model=sm.OLS(yTrain,xTrain.astype(float)).fit()
#result = logit_model.fit()
#result=logit_model.fit(method='cg')
print(logit_model.summary2())
#import numpy as np
#import statsmodels.api as sm
#import statsmodels.formula.api as smf
#results = smf.ols('A ~ funding_rounds + milestones',
data=xTrain).fit()
#print(results.summary())

import matplotlib.pyplot as plt

from string import ascii_letters
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

corr = xTrain.corr()
 ##Correlation Matrix##
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

f, ax = plt.subplots(figsize=(11, 9))

cmap = sns.diverging_palette(220, 10, as_cmap=True)


sns.heatmap(corr, mask=mask, cmap="Blues", vmax=.3, center=0,
```

```
            square=True, linewidths=.5, cbar_kws={"shrink": .5})

## XGBoosting ##


import xgboost as xgb

import pandas as pd
import numpy as np



data_dmatrix = xgb.DMatrix(data=xTrain,label=yTrain)
#Running XGBoosting#
xg_reg = xgb.XGBRegressor(objective ='reg:linear', colsample_bytree =
0.3, learning_rate = 0.1,
                max_depth = 5, alpha = 10, n_estimators = 10)



xg_reg.fit(xTrain,yTrain)
preds = xg_reg.predict(xTest)
rmse_xgboosting = np.sqrt(mean_squared_error(yTest, preds))
print("RMSE: %f" % (rmse_xgboosting))

params = {"objective":"reg:linear",'colsample_bytree':
0.3,'learning_rate': 0.1,
                'max_depth': 5, 'alpha': 10}

cv_results = xgb.cv(dtrain=data_dmatrix, params=params, nfold=3,

num_boost_round=50,early_stopping_rounds=10,metrics="rmse",
as_pandas=True, seed=123)

cv_results.head()
print((cv_results["test-rmse-mean"]).tail(1))

xg_reg = xgb.train(params=params, dtrain=data_dmatrix,
num_boost_round=10)


import matplotlib.pyplot as plt

from graphviz import Digraph
dot = Digraph(comment='The Round Table')
dot  #doctest: +ELLIPSIS


xgb.plot_importance(xg_reg)
plt.rcParams['figure.figsize'] = [5, 5]
```

```
plt.show()

auc2 = metrics.roc_auc_score(yTest,preds)

plt.plot(fpr,tpr,label="data 1, auc="+str(auc2))
plt.legend(loc=4)
plt.show()


## Running others than SKlearning#

import pandas
import matplotlib.pyplot as plt
from sklearn import model_selection
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier

seed = 7
models = []
models.append(('XGB', XGBClassifier()))
models.append(('RF', RandomForestClassifier()))
models.append(('NB', GaussianNB()))
models.append(('GB', GradientBoostingClassifier()))
models.append(('Adab', AdaBoostClassifier()))


# evaluate each model in turn
results = []
names = []
scoring = 'roc_auc'
for name, model in models:
        kfold = model_selection.KFold(n_splits=10, random_state=seed)
        cv_results = model_selection.cross_val_score(model, xTrain,
yTrain, cv=kfold, scoring=scoring)
        results.append(cv_results)
        names.append(name)
        msg = "%s: %f (%f)" % (name, cv_results.mean(),
cv_results.std())
        print(msg)
# boxplot algorithm comparison
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

```
## Individual variable##

import statsmodels.api as sm

a = sm.Logit(y, df1, family=sm.families.Gamma())
b= a.fit(cov_type='HC1')
print(b.summary())

from sklearn.datasets import make_hastie_10_2
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.inspection import plot_partial_dependence



cfl = XGBClassifier(random_state=0).fit(xTrain,yTrain)

plot_partial_dependence(cfl, xTrain,
features=[0,1,2],feature_names=xTrain.columns, grid_resolution=10)
plot_partial_dependence(cfl, xTrain,
features=[3,4,5],feature_names=xTrain.columns, grid_resolution=10)
plot_partial_dependence(cfl, xTrain,
features=[6,7,8],feature_names=xTrain.columns, grid_resolution=10)
plot_partial_dependence(cfl, xTrain,
features=[9,10,11],feature_names=xTrain.columns, grid_resolution=10)
plot_partial_dependence(cfl, xTrain,
features=[12,13,14],feature_names=xTrain.columns, grid_resolution=10)
plot_partial_dependence(cfl, xTrain,
features=[15,16,17],feature_names=xTrain.columns, grid_resolution=10)
plot_partial_dependence(cfl, xTrain,
features=[18,19,20],feature_names=xTrain.columns, grid_resolution=10)
plot_partial_dependence(cfl, xTrain,
features=[21,22,23],feature_names=xTrain.columns, grid_resolution=10)
plot_partial_dependence(cfl, xTrain,
features=[24,25,26],feature_names=xTrain.columns, grid_resolution=10)
plot_partial_dependence(cfl, xTrain,
features=[26,27,28],feature_names=xTrain.columns, grid_resolution=10)
plot_partial_dependence(cfl, xTrain,
features=[1],feature_names=xTrain.columns, grid_resolution=10)
plot_partial_dependence(cfl, xTrain,
features=[3],feature_names=xTrain.columns, grid_resolution=10)
plot_partial_dependence(cfl, xTrain,
features=[0],feature_names=xTrain.columns, grid_resolution=10)



#Causal graphe
#python setup.py install
log_y = df1['log_usd_angel']
log_x = df1.drop(columns=['log_usd_angel'])
```

```
logit_model=sm.OLS(log_y.astype(float),log_x.astype(float)).fit()
#logit_model=sm.OLS(yTrain,xTrain.astype(float)).fit()
#result = logit_model.fit()
#result=logit_model.fit(method='cg')
print(logit_model.summary2())
```