COMPARISON OF MACHINE LEARNING ALGORITHMS ON CONSUMER CREDIT CLASSIFICATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

OĞUZ KOÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
FINANCIAL MATHEMATICS

JULY 2019

Approval of the thesis:

COMPARISON OF MACHINE LEARNING ALGORITHMS ON CONSUMER CREDIT CLASSIFICATION

submitted by **OĞUZ KOÇ** in partial fulfillment of the requirements for the degree of **Master of Science in Financial Mathematics Department, Middle East Technical University** by,

Prof. Dr. Ömür Uğur
Director, Graduate School of **Applied Mathematics**

Prof. Dr. A. Sevtap Kestel
Head of Department, **Financial Mathematics**

Prof. Dr. A. Sevtap Kestel
Supervisor, **Actuarial Science, METU**

Prof. Dr. Ömür Uğur
Co-supervisor, **Scientific Computing, METU**

Examining Committee Members:

Assoc. Prof. Dr. Ceylan Talu Yozgatlıgil
Department of Statistics, METU

Prof. Dr. A. Sevtap Kestel
Actuarial Science, METU

Assoc. Prof. Dr. Özge Sezgin Alp
Accounting and Financial Management, Başkent University

Date:

Name, Last Name: OĞUZ KOÇ

Signature :

ABSTRACT

COMPARISON OF MACHINE LEARNING ALGORITHMS ON CONSUMER CREDIT CLASSIFICATION

Koç, Oğuz
M.S., Department of Financial Mathematics
Supervisor : Prof. Dr. A. Sevtap Kestel
Co-Supervisor : Prof. Dr. Ömür Uğur

July 2019, 110 pages

Like other prediction models, credit scoring is a tool used to evaluate the amount of risk associated with applicants or customers. Scoring models identify clients individually as good or bad applicants. They offer statistical odds or probabilities for prediction either the applicant will be default or not in the future. It is beneficial for banks and credit analysts to measure customers' non-payment risk by statistically tested algorithms in many aspects such as reduction in workload and evaluation time. Also, only demanding features that have the most significant impact on credit assessment process in terms of obtaining more explanatory outcomes, emphasizes the benefits mentioned formerly. Today, Machine Learning (ML) algorithms are commonly applied for data analysis in various areas. The algorithms learn how to determine complicated patterns and create smart choices by generating a mathematical model depending on sample dataset without direct programming.

In this thesis, a comparative study is performed using Logistic Regression (LR), Support Vector Machine (SVM), Gaussian Naïve Bayes (GNB), Decision Trees (DT), Random Forest (DT), XGBoost (XGB), K-Nearest Neighbors (KNN) and Multilayer Perceptron Neural Network (MLP) algorithms. In addition to these, we strive to achieve more explanatory outcomes in terms of dimentionality with Wrapper Feature Selection (WFS), and investigate its performance in a way of important attributes detection capacity. We also analyze the impact of Grid Search (GS) hyper-parameters optimizing method, and effect of four data transformation techniques Natural Logarithm (LN), Standard, Box-Cox and Min-Max to these algorithms and methods. We compare these cases to determine the most appropriate way for credit classification by considering accuracy, AUC, type I and type II error rates. All measurements are conducted on German and Australian real world consumer credit datasets commonly used in literature.

Keywords: Machine Learning, Credit Classification, German Credit Dataset, Australian Credit Dataset, Data Transformation, Wrapper Feature Selection, Data Mining

ÖZ

TÜKETİCİ KREDİLERİNİN SINIFLANDIRMASI ÜZERİNDE MAKİNE ÖĞRENİMİ ALGORİTMALARININ KARŞILAŞTIRMASI

Koç, Oğuz
Yüksek Lisans, Finansal Matematik Bölümü
Tez Yöneticisi : Prof. Dr. A. Sevtap Kestel
Ortak Tez Yöneticisi : Prof. Dr. Ömür Uğur

Temmuz 2019, 110 sayfa

Diğer tahmin modelleri gibi, kredi değerlendirme de başvuru sahipleriyle veya müşterilerle ilişkili olan risk miktarını değerlendirmek için kullanılan bir araçtır. Değerlendirme modelleri müşterileri bireysel olarak iyi ya da kötü başvuranlar olarak tanımlar. Başvuru yapanların gelecekte temerrüde düşüp düşmeyecekleri için istatistiksel olasılıklar veya tahmin olanakları sunarlar. İstatistiksel olarak test edilmiş bir algoritma kullanarak kredinin geri ödenmeme riskinin ölçülmesi bankaların ve kredi analistlerinin iş yükünün ve değerlendirme sürecinin azaltılması gibi birçok açıdan faydalıdır. Ayrıca, sadece kredi geri ödemesinde anlamlı etkisi olan değişkenlerin kredi talebinde bulunan kişilerden istenmesi daha açıklayıcı sonuçlar alınması açısından bahsedilen yararların etkinliğini artırmaktadır. Günümüzde Makine Öğrenimi (ML) algoritmalarıyla yaygın olarak çeşitli alanlarda veri analizi yapılmaktadır. Bu lagoritmalar doğrudan programlama yapmadan örnek veri setine bağlı olarak oluşturulan matematiksel bir model ile karmaşık ilişkilerin nasıl belirleneceğini ve akıllı seçimler yaratmayı öğrenirler.

Bu tezinde, Lojistik Regresyon (LR), Destek Vektör Makinesi (SVM), Gaussian Naïve Bayes (GNB), Karar Ağaçları (DT), Rasgele Karar Ormanları (RF), XGBoost (XGB), K-En Yakın Komşu (KNN) ve Çok Katmanlı Algılayıcı Sinir Ağları (MLP) algoritmaları kullanılarak kapsamlı bir çalışma yapılmaktadır. Bunlara ek olarak, Wrapper Özellik Seçilimi (WFS) ile boyutluluk açısından daha açıklayıcı sonuçlara ulaşmayı amaçlıyoruz ve bunun önemli özellikleri belirleme yönünden kabiliyetini araştırıyoruz. Biz ayrıca hiper-parametere optimizasyon yöntemi olan Kare Arama'nın etkinliğini ve dört farklı veri dönüşümü tekniği olan Doğal Logaritma (LN), Standard, Box-Cox ve Min-Max'un bu algoritma ve metotlara olan etkilerini analiz ediyoruz. Biz kredi sınıflandırması için en uygun yolu belirlemek için doğruluk, AUC, tip I ve tip II hata oranlarını göz önünde bulundurarak bu durumları karşılaştırıyoruz. Tüm ölçümler literatürde yaygın olarak kullanılan Alman ve Australya gerçek dünya tüketici kredisi verileri üzerinde gerçekleştirilmektedir.

Anahtar Kelimeler: Makine Öğrenimi, Kredi Sınıflandırması, Alman Kredi Verisi, Avustralya Kredi Verisi, Veri Dönüşümü, Wrapper Özellik Seçilimi, Veri Madenciliği

To My Family

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to thesis advisor, Prof. Dr. A. Sevtap Kestel for her patient guidance, enthusiastic encouragement and motivation during the development and preparation of this thesis from beginning to end. Her willingness to give her time and to share his experiences has brightened my path.

I would also like to express my sincere gratitude to my co-advisor Prof. Dr. Ömür Ugur for his useful suggestions and contributions.

I am thankful to my close friends Ali Amirfaridi, Savaş Kaptan and Rıdvan Memiş for their motivation and helps.

I would like to thank Mervan Aksu for his suggestion and guidance at the very beginning of Thesis.

Special thanks to thesis committee members, Assoc. Prof. Dr Ceylan Yozgatlıgil. and Assoc. Prof. Dr. Özge Sezgin Alp for their contribution.

Finally, I am grateful to my dear family, who have provided me through moral and emotional support in my life.

TABLE OF CONTENTS

CHAPTERS

APPENDICES

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

LR	Logistic Regression
SVM	Support Vector Machine
XGB	XGBoost
RF	Random Forest
DT	Decision Tree
MLP	Multilayer Perceptron
KNN	K-Nearest Neigbor
GNB	Gaussian Naïve Bayes
ML	Machine Learning
WFS	Wrapper Feature Selection
GS	Grid Search
BDDK	Banking Regulation And Supervison Agency
AI	Artificial Intelligence
UCI	University of California, Irwin's
LDA	Linear Discriminant Analysis
PD	Probability of Default
NN	Neural Network
CART	Classification and Regression Trees
RBF	Radial Basis Function
GA	Genetic Algorithms
PSO	Particle Swarm Optimization
AUC	Area Under Curve
LN	Natural Logarithm
KFCV	K-Fold Cross Validation
SFS	Sequential Forward Selection
KKT	Karush-Kuhn-Tucker
RKHS	Reproducing Kernel Hilbert Space
LK	Linear Kernel

SK	Sigmoid Kernel
PK	Polynomial Kernel
MI	Mutual Information
OOB	Out of Bag
BT	Ball Tree
LBFGS	The limited memory Broyden-Fletcher-Goldfarb-Shanno
RR	Ridge Regression
CG	Conjugate Gradient
SAG	Stochastic Average Gradient
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
TPR	True Positive Rate
FPR	False Positive Rate
BCL	Best Case List
CAL	Common Attribute List

CHAPTER 1

INTRODUCTION

As a result of the unrestrained growth of banking sector and financial globalization, in 2008, the financial crisis arise from housing market fluctuations in USA, and spread worldwide. In a consequence of this crisis, it is appeared that the banking sector conduct their management tasks insufficiently. During the crisis, the total amount of loses of banking sector is 2 trillion dollars. After these incidents, regulatory criterias such as Basel III are developed[1], and risk management become one of the main factors determining contestability of banks. Besides, the current and future risk level of the bank's loan portfolio is crucial in terms of its competitiveness, resilience to financial crises and profitability. Additionally, the ratio of consumer loans to total number of all loans of all banks operating in Turkey is 71% in 1988, this ratio increased to 93% in 2016[2]. According to Turkish Banking Regulatory Authority (BDDK) report[3], the total amount of retail loans was 173 billion TL in 2010 while this amount reached 518 billion TL in September 2018. Additionally, the total amount of non-performing loans has increased notably since 2010 [28].

Banks can make credit classification in two different ways. The first is a classification method based solely on the individual judgment of the credit analyst. This assessment depends on the personal judgment and expertise of individual customer representatives [19]. Another method used by banks is the credit scoring method. This approach is the classification of the new application according to a statistics based on classification algorithm, such as ML, that takes into account current and

[1] BDDK Sorularla Basel III December 2010 report
[2] www.tbb.com.tr
[3] Banking Regulation and Supervison Agency Sector Main Indicator 2018 September report

past payment habits of customers. With this method, the payment habits of the new customer can be estimated based on the dataset [31]. In order to measure the risk of non-performing credit using a statistical testing algorithm can significantly reduce the workload of banks and credit experts. Additionally, contradictory and objective credit decisions can be decreased, speed of the process and system adjustments against to changing environment would be increased. Statistical methods can make decisions that based on huge datasets that credit analysts can not reach that level of experience during their entire lives and deduce by using that much information. This methods make the credit process accessible for costumers since there is no need for face to face contact and reduce the operating costs in case of high volume of workload [1]. ML offers the ability to automatically learn to detect complicated patterns and create smart data-based choices without direct programming. These algorithms generate a mathematical model depending on sample instances, also known as training data. Recently, ML methods are widely used in many data analysis fields [28].

Credit dataset cannot be provided from Turkish banks due to the confidentiality customer information protection. For this reason, we conduct our research on two real-world datasets commonly used in the literature which also enables us to make sound comparisons with respect to the other studies in the literature. The datasets are taken form the University of California, Irwin's (UCI) [23] ML data repository. German dataset consists of 1000 observation labeled as 1 or 0, and 30% of the data is defaulted borrowers. Australian data set includes 690 examples and 45% of them is belong to class 0. All machine learning analysis are conducted using sckit-learn libraries on Python programming language. We employ eight different ML classification algorithm.

1.1 The Aim of the Study

Credit assessment has a great importance for banks in terms of competing against the other banks, its resilience to financial crises and its profitability. Therefore, the risk management system used for credit decision should be modernized respect to the changes and be robust. In this study, we investigate the most suitable and current classification algorithm fulfilling these conditions. We aim to obtain more explanatory results in terms of dimentionality, and investigate capability of detection significant features in case WFS method is used with ML algorithms. We also analyse the contribution of hyper-parameter optimization, these parameters are included in all ML algorithms, and can be manually adjusted, with GS. Additionally, effect of four types of data transformation to these algorithms and methods is also in the scope of this study.

The thesis is structured as follows: Chapter 2 contains a literature study of evaluation of credit scoring and recent papers that related to our study. Chapter 3 demonstrates credit scoring aspects, and its importance. Also, the dissimilarities between judgmental and credit scoring systems are revealed. In Chapter 4, we give detailed information about two credit data set with attribute distribution and bar-plot graphs. Then, we proceed with the statistical background explanations of scaling/normalization techniques. After that, we state the mechanism behind the wrapper attribute selections, and GS a method for ML to determine the optimum hyper-parameters. In Chapter 5, we introduce statistical and mathematical background of eight different ML algorithms, and also techniques that used within these algorithms such as different optimization methods to estimate optimum coefficients in LR. Chapter 6 demonstrates the results in three aspects consisting of large variety of different approaches. First of them is the impact of data normalization techniques to efficiency of the ML algorithm. The effect of wrapper methods on ML is the other aspect. After that we examine how GS parameter optimization method affects these algorithms.

CHAPTER 2

LITERATURE

2.1 Evaluation of Credit Assessment

The history of credit may dates back to appearance of commerce. The credit assessment history is beginning just about sixty years ago, but credit history goes back to about 2000 BC conversely. Credit assessment history is very short comparison to emergence of credit. The concept of discriminative statistical assessment between groups in samples is established by Fisher [25] in 1936. In this study, he attempts to distinguish among two iris types by measuring the plant's physical size and by using their physical measurements to distinguish the root of the skulls by using Linear Discriminant Analysis (LDA). The first usage of the same techniques for differentiation among the default and non-default credits is implemented by Durand [24] in 1941. This method is a research project in the US National Bureau of Economic Research. However, in this project LDA does not have an aim for forecasting purpose. In additionally, some of the financial institutions have problems with their loan operations at the same moment. Credit assessments for deciding whether or not giving the a loans to a customer is made by credit analysts judgmentally for a long time. These loan analysts are drafted into military service, therefore, there is a serious need for the people that has this kind of specialty. Hence, the companies wanted the analysts to write down practical rules needed for credit assessments [34].

Inexperienced employees then used these guidelines to create loan choices. In this way, one of assessment system is established for the first time. In the late 1960s, with the appearance of the credit cards, the banks and other institutions that use the credit cards noticed the benefit of credit assessment. Every day, the number of individuals is making application for credit cards, and the assessment of all these applications with manpower is unfeasible except with automatized credit decision systems. These institutions discovered that automatized credit decision is a much stronger predictor than any other judgmental system when they apply this approach. Also, they noticed that their default rates could decrease by 50% or more [45].

The achievement of credit assessment in credit cards in the 1980s led to financial institutions started to apply for scoring the other financial services such as consumer credits. In the United States, the full quantity of outstanding personal loans is more than $700 billion. Eastern European nations and China also begin to notice the benefit of consumer credit since a central point in a personal loan is providing credit widely accessible and also profitable. Developments in computing enables other methods to be attempted to construct scorecards. Logistic Regression and Linear programming which today's commonly used card builders are introduced in the 1980's. The Basel standards for determining capital requirements are introduced in 1998, and also enhance and advance so far. Capital is needed for every kind of financial major risks. Credit risk typically has been the biggest risk factor facing banks, and generally the one needs the highest capital [30].

With advancements in computer technologies, artificial intelligence methods such as expert systems and neural networks are tested in credit scoring more recently. Probability of Default (PD) estimations are made for classification and survival analysis by new models constructed with strong methods. This models are used to calculate the customer's default likelihood and when the borrower's default would take place [37]. A variety of credit scoring methods are compared, including discriminating analysis, logistic regression, Bayes classifier, nearest neighbor, neural networks, and classification trees. It has been found that artificial neural networks are more accurate in classification than the other five techniques [69]. In addition, it is noticed that advanced techniques such as Neural Network (NN) are working outstandingly better than the extreme learning machine on loan assessment [37].

2.2 Literature on Machine Learning

In Brown and Mues study [13], many machine learning methods are analyzed on imbalanced data distribution sets at which five different credit data exist. The imbalance rates of these data are increased by 1%, 2.5%, 5%, 10%, 15% and 30% respectively and the performance of the algorithms are compared. Performance criterias AUC, Friedman's statistics and Nemenyi post hoc test are used to validate these methods under this ratios. According to result, RF and Gradient Boosting yielded the best results, while C4.5 and KNN are the worst techniques in cases of large class imbalance cases. Following study of Lessmann et al. [37] has tested 41 different predictive methods on eight different credit data. Each algorithm used in these analyzes is tested 50 times, and averages of the test results are taken. Six different criteria PCC, AUC, Partial Gini index, H-measure, Brier Score, and the Kolmogorov Smirnov statistics are applied in the evaluations. They conclude that many machine learning methods have much better results than Logistic Regression which is accepted as an industrial standard. Especially the ensemble algorithms preferable to the other methods. In another study of Ince and Aktan [31], performances of discriminant analysis, logistic regression, artificial neural networks and Classification and Regression Trees (CART) based Decision Tree are compared. Analyzes are made on data consisting of 1260 observations, which included nine characteristics from a Turkish bank. As the result, it is found that Decision tree has the best performance. In paper of Basens et. al. [5], mainly used classification methods in ML literature are compared on 8 different real-world data. Evaluation of these methods is based on AUC and PCC criterias, which conclude that NN and SVM algorithms are prominent methods. LR, C4.5, SVM and NN methods are examined in work of Yu et. al. [70], analyzing them with 8 different training data ratios from 10 percent to 80 percent, which result in LR and SVM have the best results comparison to the other methods in general, with SVM being more robust against different data split ratios. 1-NN, NB, LR, MLP, Radial Basis Function (RBF) neural networks, SVM and C4.5 based DT algorithms are used in study of Marqués et. al. [43]. The comparison of the algorithms, and their compatibleness with 5 different ensemble techniques is examined. It is found that DT based on C4.5 and SVM gives better results than other classifiers.

In feature selection area on ML algorithms, there are a wide variety of methods such as wrapper, filter and hybrid algorithms. The paper of Van et. al. [63] approaches this subject by using German and Australian data sets. In this research, the features are 20 times trained on RF and median values and standard deviation of outcome values are evaluated. As a result, the features with the highest median and the lowest standard deviation are determined. Obtained subset is validated by NN algorithm, and these scores also are used in specifying the final form of attribute subset. Efficiency of this subset is assessed with the five different ML methods, and as a result they see that usage of the subset increase performance of these ML algorithms generally comparing to normal data set. On another research by Liang et. al. [40], same data sets as the research above and two additional credit data sets are used and 5 different attribute selection approach are applied. This methods include two wrapper methods with Genetic Algorithms (GA) and Particle Swarm Optimization(PSO), and the three widely used filter methods t-Test, LDA, and LR. It is summarized that there is no best feature selection method for the four data sets. Efficiency of the base line algorithms can be enhanced if the attribute selection method is chosen carefully. Another study by Chen and Li [15] on attribute selection practices with the same two data sets of German and Australian credit data has taken place, analyzing four selection methods on SVM algorithm. The methods examined are LDA, DT, RST and F-score. It is concluded that LDA+SVM combination is slightly better than the other combinations, but DT+SVM gives accuracy result that has the minimum standard deviation. In the following study of Suto et. al. [59], evaluations on human activity recognition data is conducted. One NB wrapper and six filter methods are compared on both NN and NB. They find that attribute subset created with NB wrapper gives the best results in comparison to other selection algorithms. Another research about attribute selection in our literature review is the paper of Nnomoko et al. [47]. In this study, they examine a filter and two wrapper methods on diabetes data, and they see that wrapper approach have the highest performance against filter method in terms of ROC and accuracy. However, they also add that filter methods are more robust and less time consuming approach than wrapper methods.

In literature, the effects of data transformation are not commonly examined. One of the paper by Szymanski et al. [60] analyzes are performed on masquerade data consists of 5000 observations. Efficiency of SVM is examined on Standard and Minimum Maximum data transformation methods. They find that combining SVM with Max-Min transformation gives more accurate results. In the other research of Jayalakshmi and Santhakumaran [33], performance of NN with five different scaling methods is measured. These methods are Min-Max, Median Norm, Sigmoid Norm, Z-score and Statistical Column Normalization. Studies performed on the PIMA Indian Diabetes dataset consist of 768 samples. They conclude that effectiveness of NN depends on scaling methods, and NN with Statistical Column Normalization provides better results than with the other methods, while worst result is obtained by median normalization.

CHAPTER 3

CREDIT SCORING

Credit scoring, like other prediction models, is a instrument used to assess the amount of risk related to candidates or clients. Scoring models identifies customers as good or bad applicant on an individual basis. They offer statistical odds or likelihood that the applicant with any specified score will be good or bad in the future. These odds or scores are then used as a basis for decision making process along with other factors such as predicted approval rates, income, and losses [56].

The simplest meaning of the loan is to provide monetary service before payment. It could be for short-term expenditure, consumer durables and other benefits providing valuable services to customers or business enterprise. The meaning of the word credit is rely on or trust. When you give someone something, you have to trust him or her to honor the responsibility. On the other hand, scoring relates to using a numerical tool to rank order instances based on some actual or perceived quality to discriminate between them and to guarantee objective and stable choices. The available information is integrated into a single value that means a certain quality, generally linked to desirability or suitability. Measurements are commonly described as numbers representing a single quality, while these scores may be presented with classifications (high, medium, low, etc.) representing one or more qualities. Scoring has become a common method in processes where predictions are required that can only be stated statistically, and do not express certainty. If there were methods that predicted the future in a perfect way, it would make a bank or financial institution's credit portfolio very robust, and this method give an big advantage in competition. Extrapolate is an unreachable ideal, and creditors must do their utmost with the available data. The total losses and the elements of profitability are ignored sometimes when application, behavior and credit history ratings are mistakenly considered to be creditworthiness measures. Indeed, a individual with greater than average risk can be creditworthy at the correct price or increasing the interest rate, lowering the quantity, shortening the duration of the debt. Predictive credit models are used to evaluate the relative probability of a future event based on past information. If there is no sufficient data for scoring models, then judgmental models could be used. Combining scores and strategies by automatically usage of the computers offers a significant reduction in decision making process costs and time [1].

A robust credit risk management can be stated as providing a strong portfolio of performing assets that can compete against to others. The interest rate and margin of the loans has to reflect the risk. A good quality assessment approach is a requirement to prevent high-level losses. Credit assessment is a method for creditworthiness that analyzes the risk of the borrower. High probabilities or scores represent low risk and small probabilities demonstrate high risk level in a good credit evaluation model, so it must be very discriminatory. To have better ranking assessment, the scoring system has to be highly discriminatory. Risk measurements is classified to each score or score categories during the calibration stage. The robustness, risk ranking and calibration of credit scores can be verified by testing observed old credit losses per score. Credit validation scores are often categorized into segments. Categorized scores are stratified risk estimates that are also known as risk classes and ratings [62].

3.1 Credit Scoring Versus Judgmental Systems

The general concept of loan assessment is to compare a clients features with attributes that belongs to other previous customers whose loans they have already paid back. If the features of a customer are sufficiently comparable to those granted credit and have therefore failed, the request will usually be dismissed. The credit request will usually be given if the customer’s characteristics are satisfactorily comparable to those that have not defaulted [19].

Lenders make judgmental decisions for potential borrowers in traditional lending based on the applicant's character by measuring the borrower's reputation and honesty, the borrower's ability to pay, the capital that the borrower's assets, collateral provided in the event of payment problems occurrence, and condition that the borrower's circumstances. These evaluations are based on the knowledge of the lenders themselves and what they have learned from their mentors, taking into account not only historical experience, but also a forward-looking perspective of the prospects of the borrowers. Obtaining information through customer relationships makes it very hard for clients to grow their company. The efficiency of a judgmental method relies on the loan analyst's knowledge and common sense [1].

Consequently, judgmental methods are linked to subjectivity, inconsistency and individual preferences that motivate choices; and also there are some strengths in judgmental techniques, such as taking qualitative features into consideration and dealing with inexact information, and missing information [2].

On the other hand, analysts use their historical knowledge with debtors in a credit scoring model to derive a quantitative model to segregate acceptable and unacceptable loan requests. A credit application is mainly a self-operating method using a credit scoring model and is continuously applied to all credit choices. The scoring process can create choices based on a statistically extracted amount of points related to the applicant's score provided by the predictor factors, such as age or educational background. As a consequence, it can be said that credit scoring allows progress to quickly evaluate creditworthiness [58]. In addition, credit assessment provides advancers an opportunity to enhance customer services and retain steady clients. An analyst can, of course, distinguish the acceptable from the unacceptable loan candidates by using a statistically extracted cut-off credit score. On the other side, due to statistical issues with the information used to develop the model, credit assessment has been criticized as well as assumptions of the specific statistical method used to obtain point scores. These models can be considered one of the most effective models used in business and finance despite the criticism of credit scoring systems [6].

Credit assessment needs less data to make a choice, since credit scoring models have been estimated to include only those variables that are statistically and substantially correlated with repayment results, while judgmental choices have no statistical importance and therefore no variable reduction techniques are accessible. Credit scoring systems try to correct the error that would occur with only considering accepted applicant and not all applications repayment history. The process is done by assuming, if rejected requests approved, how they would have been fulfilled. Features of those who have been accepted and who have subsequently failed is usually a base for judgmental methods [19]. Credit scoring models consider both good and bad borrowers' characteristics, whereas judgmental techniques are commonly biased towards bad borrowers. Credit scoring models are based on much bigger samples than can be remembered by a credit analyst. Credit assessment models reveal the correlation between the attributes which belong to customer and repayment behavior. While in the event of judgmental methods this correlation can not be reviled since many of the features that a loan analyst may use are not measured impartially [14].

A credit assessment model can handle a large number of the characteristics of a customer at the same time, including their relationships, however The mind of a loan analyst can not possibly do the same, because the task is too difficult and complicated. Another vital advantage of credit scoring is that different loan experts or data analysts can analyze the same data effectively and accurately and offer the same weights. In the event of judgmental techniques, this is extremely unlikely [19].

Some other credit scoring benefits are more effective processing time, reduced costs and effort of loan process, modeling with real data, occurrence of fewer error rates, providing estimates to be compared in previous assessments,consideration of mutual relations between variables, objective analysis can be used to support the credit risk assessment, with the feature selection techniques fewer customer information can be required for loan assessment, and the threshold value can be altered depending on environmental variables influencing the banking sector [14].

CHAPTER 4

DATA PROCESSING, METHODS AND DATASETS

4.1 Data Transformation Techniques

In preprocessing phase, feature scaling is a vital part that could be easily ignored by the analyst. DT, RF and XGB are some of tree based algorithms that do not require feature scaling. These algorithms are insensible against to scaling methods, whereas most of ML and optimization methods work more efficiently when attributes are on the same range [51]. For instance, different attributes are typically measured on various ranges, and if there is a usage of Euclidean distance or Manhattan distance metric, it can cause some features completely outweigh the others that had smaller range of measurement [68]. Normalization and standardization are two of the most common methods for bringing distinct attributes to same scale [51].

Natural Logarithm

The natural logarithmic scaling is a particular case of a group of transformations known as power transformation. The logarithmic function (LN) is formed by taking the inverse of the exponential function. The log function shrink the range of large numbers and expands the range of small numbers. The transformation can simply be defined as the greater x, the slower the increases in the function. The function is expressed as

$$x_i' = \ln x_i, \quad x > 0 \tag{4.1}$$

where x_i is the i-th sample in the data, and $x_i^{'}$ is the transformed sample. It compresses the long tail into a shorter tail, and expands the lower tale into longer. The transform is a strong instrument to deal with a heavy tailed distribution of positive numbers [71].

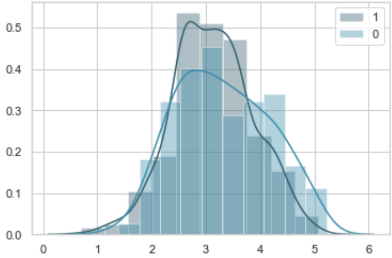


Figure 4.1: Attribute Amount under LN transformation.

In Figure 4.1, it is seen that the amount attribute in German dataset is normalized, and scaled values falls within the range between 0 and 6.

Box-Cox

Box-Cox transformation also known as power transformation. Statistically, these transformations are aimed at stabilizing the variance. We can define this transformation as

$$x_i^{'} = \begin{cases} \frac{x_i^{\lambda}-1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln x_i & \text{if } \lambda = 0, \end{cases} \tag{4.2}$$

where λ is transformation parameter, and x is the i-th sample. To avoid the variance dependence on the mean, Box-Cox transformation change the distribution of the variable. Setting λ to be less than 1 compresses the higher values, and setting it higher than 1 has the opposite effect. While applying the transformation, we have to determine a value for the λ parameter. Determination for specific value of λ could be done via maximum likelihood estimation or Bayesian methods. Additionally, to apply a Box-Cox transformation, observations have to be positive [71].

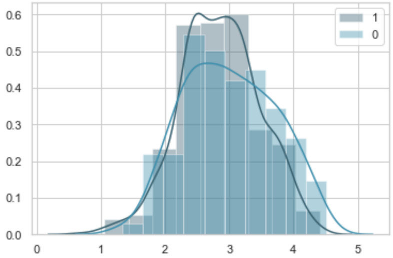


Figure 4.2: Attribute Amount under Box-Cox transformation.

It is seen that the amount attribute values are centered, and its values are scale into the range [0,5] in Figure 4.2.

Min-Max

Min-max scaling function provides a linear transformation of the initial data. It maintains the connection between the values of the initial data [28].

Let x be an individual feature value (i.e., a value of the feature in some data point), and $min(x)$ and $max(x)$, respectively, be the minimum and maximum values of this feature over the entire dataset. Min-max scaling squeezes (or stretches) all feature values to be within the range between 0 and 1. Consider that x is an attribute value in the data, and $max(x)$ is the maximum value of an attribute column and $min(x)$ is the minimum value of the same column. With the usage of these values this function scale by compressing or expanding the values into the range between 0 and 1 [71].

The formula for min-max scaling is:

$$x_i^{'} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.3}$$

It is seen in Figure 4.3 that the amount attribute values are only scaled into the range [0,1], but this attribute keeps its long tail distribution.

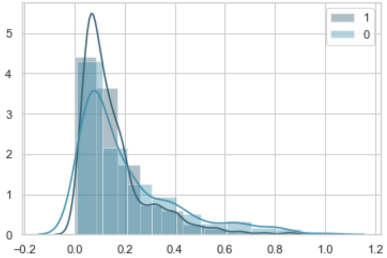


Figure 4.3: Attribute Amount under Min-Max transformation.

Standard

Standard scale is also known as the z-score standardization can be more practical for many ML algorithms, as well as for optimization methods such as stochastic gradient descent. Some linear models like SVM start the weights small random values which is close to zero. In this transformation, feature columns are set in the midst at standard deviation one and mean zero. It makes the process easier. In min-max scaling, the feature variables are squeezed into the range between 0 and 1 which makes algorithms sensitive to the extreme values, but the standard scaling sustain useful outlier information which makes the algorithms more robust. The standardization procedure can be demonstrated by the following equation:

$$x_i^{'} = \frac{x - \mu}{\sigma} \tag{4.4}$$

Here, μ is the sample mean of a particular feature column and σ is the corresponding standard deviation [51].

It is understood in Figure 4.3 that the amount attribute values are only scaled into the range [0,1], but this attribute is kept its long tail distribution.

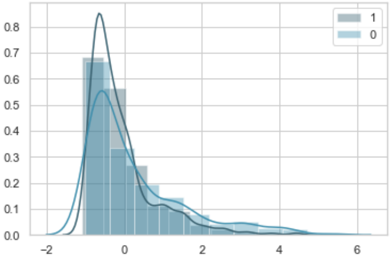


Figure 4.4: Attribute Amount under Standard transformation.

Similar to the Max-Min, with the Standard scale in Figure 4.4 amount attribute maintains its distribution, but its range is between [0,5].

4.2 Feature Selection

Feature Selection is one of the most influential analysis for performance of the model since irrelevant or partially relevant features can cause negative effect. Besides, this analysis provides various benefits such as reduction in run time and obtaining more accurate results.

Wrapper Method

The learning algorithm is wrapped into variable subset selection procedure for this reason it is called wrapper method. It would be simple to make an independent evaluation of an attribute subset if there was a silver bullet way to determine when an attribute was related to the class. Yet, there is no widely accepted measure of significance, even though several variety of measurements have been proposed [71].

The general concept of the wrapper method is to pick a subset of features using a learning algorithm as part of the evaluation function. In this method a black box function is used to search relevant attributes, instead of usage of explicitly described feature assessment method, such as entropy. In the assessment process, each candidate attribute subset is validated by the learning algorithm itself, and it returns a score value corresponding to each subset, visualization of WFS is in Figure 4.5. This can be rather time consuming, since for each candidate feature subset evaluated during the search, the target learning algorithm is usually applied several times (e.g., in the case of ten-fold cross-validation used to estimate model quality). This process could be taken time because each of the attribute in the data is evaluated generally several times, especially in case of K-Fold Cross Validation (KFCV) usage. The wrapper method can be used with any machine learning algorithm. KFCV is common option for wrapper methods since usage of algorithm itself may cause overfitting problem. Besides cross validation, the greedy search methods such as forward and backward selection are some typical implementation algorithms [8].

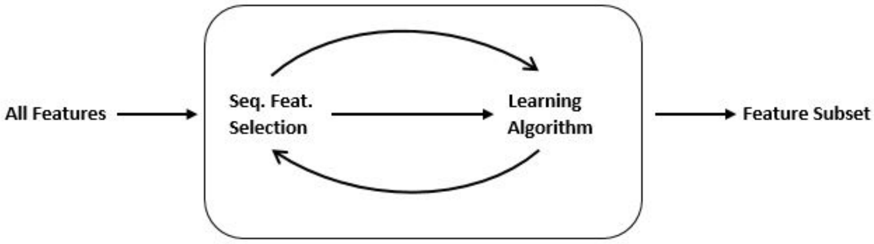


Figure 4.5: Wrapper Mechanism.

Sequential Forward Selection

In a nutshell, Sequential Forward Selection (SFS) processes begins with the induction of machine learning algorithms for *n* single-attribute subsets from which the best performer is selected, which finished the first selection phase. Among the $n - 1$ attributes most fitted one with the already selected feature is spotted, and this attribute is added to the subset. This process goes on until the number of subset reaches to n. In that way, whole attribute combinations are validated and the subset that has the highest score, such as accuracy or minimum weighted error,is detected. Algorithm 0 is given for simplicity of the SFS mechanism. We are dealing with more and more variables in this methodology with each iterative phase of the method, a question is could be examined in each step that whether it is enough, whether we have the set of attributes that fulfill our criteria. The answer is not quite clear since when accuracy criteria is considered as the most efficient factor for base decision measure, reaching to the maximum point is not an easy process. The best point that we achieve could be an local maximum instead of a global optimum. Even if it is greedily continued, the next optimum point could also be another local maximum point. Only when the brute force approach is applied, the global maximum point is guaranteed. However it is an extremely exhaustive and time-consuming process. When the dataset include *n* attributes, the combination number is 2^n for all the possible assessment. Without setting an initial stopping criteria, the brute force strategy the process could not be over even in a lifetime. Since the execution time increases exponentially as the number of attributes go up. In such a scenario, the process can be limited by initially stating the number of features that contained in subset, but it does not make certain the optimum subset. The time consumption is the most important problem in wrapper methods. In the SFS approach, execution time is considerably low in comparison to brute force because the number of combination is $\frac{n(n+1)}{2}$. SFS is a more appropriate way to apply wrapper methods, as we will use SFS in our study [57]. The mechanism of SFS is demonstrated in Table 4.1 for better understanding.

Table 4.1: Forward Feature Selection Algorithm

```
$k = 0; i = 0; best = 0; score = 0; OFS = [\,]$
$FC = [f_0, f_1, f_2, ..., f_t]$
$n = len(FC)$
while $i < n$ : do
    $k = len(FC)$
    $max = 0$
    $feature = 0$
    for $j\ in\ range(k)$ : do
        $score = eval(FC[j])$
        if $score > max$ : then
        $max = score$
        $feature = FC[j]$ ;
        if $max > best$ : then
        $best = max$ ;
    end
    $OFC = OFC + feature$
    $FC = FC - feature$
    $i = i + 1$
end
```

4.3 Parameter Optimization

In machine learning, there are two kind of parameters. The first one is learned from the training data, as an example the weights in LR, and the other one are hyper-parameters of a learning algorithm that are optimized separately. For example, the iteration number parameter in LR or the number of trees in a random forest [51].

Search of the possible combinations of hyper-parameters, and where to look for optimum point is a hard task. An hyper-parameter space may contain value combinations that perform better or worse. Even after detecting a good combination, there may always be a better combination of the parameters since this point could be an optimum point in local minimum area. A practical way to overcome this problem, verifying hyper-parameters for an algorithm applied to specific data is to test them in large variety by KFCV, and to pick the best combination based on the highest accuracy rate. GS makes this process easier by allowing user by sampling the range of possible values to input into the algorithm and to spot when the general error rate is in minimum [44].

It is a brute-force approach that a list of prespecified values belongs to different hyperparameters is used, and the model performance is assessed to find the highest accuracy among the combination of those parameters. To obtain more robust and general results, GS is commonly applied with 10-fold cross validation [51]. The structure of GS is given in Table 4.2.

Table 4.2: Grid Search Algorithm

```
$max = 0; best = 0; score = 0$
$A = [a_0, a_1, a_2, ..., a_n]$
$B = [b_0, b_1, b_2, ..., b_n]$
for $i\ in\ A$ do
    for $k\ in\ B$ do
        $score = eval(\phi_{ik}(x_0, x_1, ..., x_t))$
        if $score > max$ : then
        $max = score$
        $feature = FC[j]$ ;
    end
end
```

where x_t's are sample tuples in dataset, A is the set of one type hyperparameter, and B is another type of hyperparameter set. a_n's and b_m's are different parameter values aimed to tune.

4.4 Datasets in Credit Classification

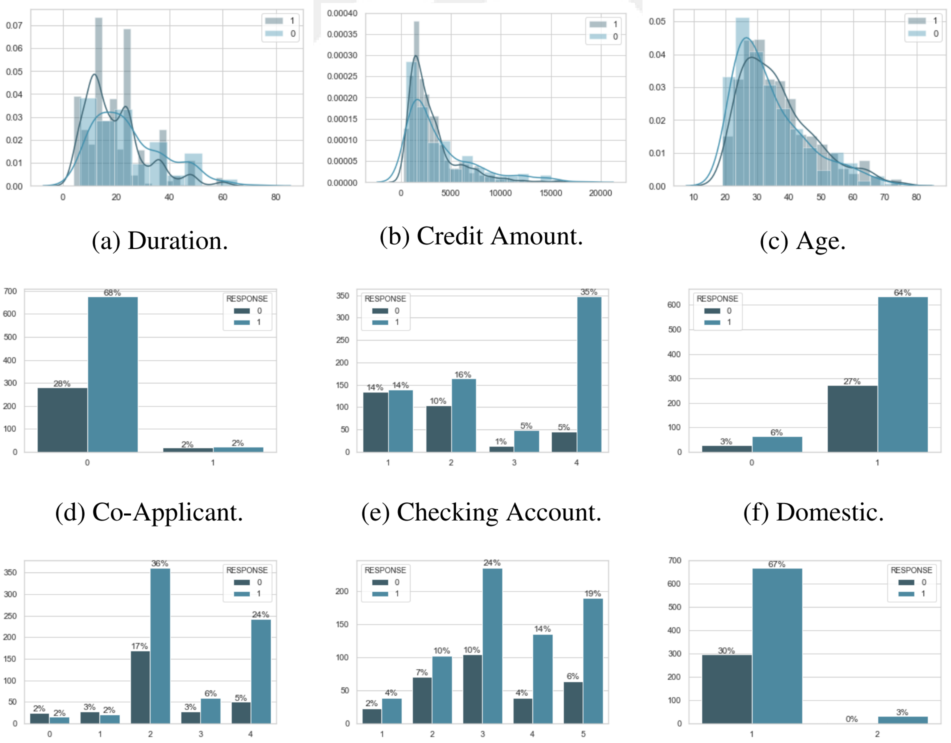
The performance of the ML algorithms and different approaches that we work on requires real-world data. However, due to Law On The Protection Of Personal Data, published in the official gazette and entered into force on 07.04.2016, customer data cannot be provided from Turkish banks [1]. For this reason, two real-world data sets, German and Australian, are used to conduct our research. Since these data sets are

[1] http://www.resmigazete.gov.tr/eskiler/2016/04/20160407-8.pdf

frequently used in the literature, we will be able to compare our own results with the literature.

4.4.1 German Dataset

German dataset is taken form the UCI machine learning data repository [23], and there are two available variety of the data . The first one of these is the original one consists of symbolic, categorical and continuous attributes. The second and the numeric form of the data produced by Strathclyde University for algorithms that work only with numerical attributes contains 24 attributes, which are 20 attributes in the original version. There are 1000 observations in the data set, 700 of which belong to class 1 (Not default) and 300 of them belong to class 0 (Default). The plots of these attributes are visualized in Figure 4.4.6.



(a) Duration.

(b) Credit Amount.

(c) Age.

(d) Co-Applicant.

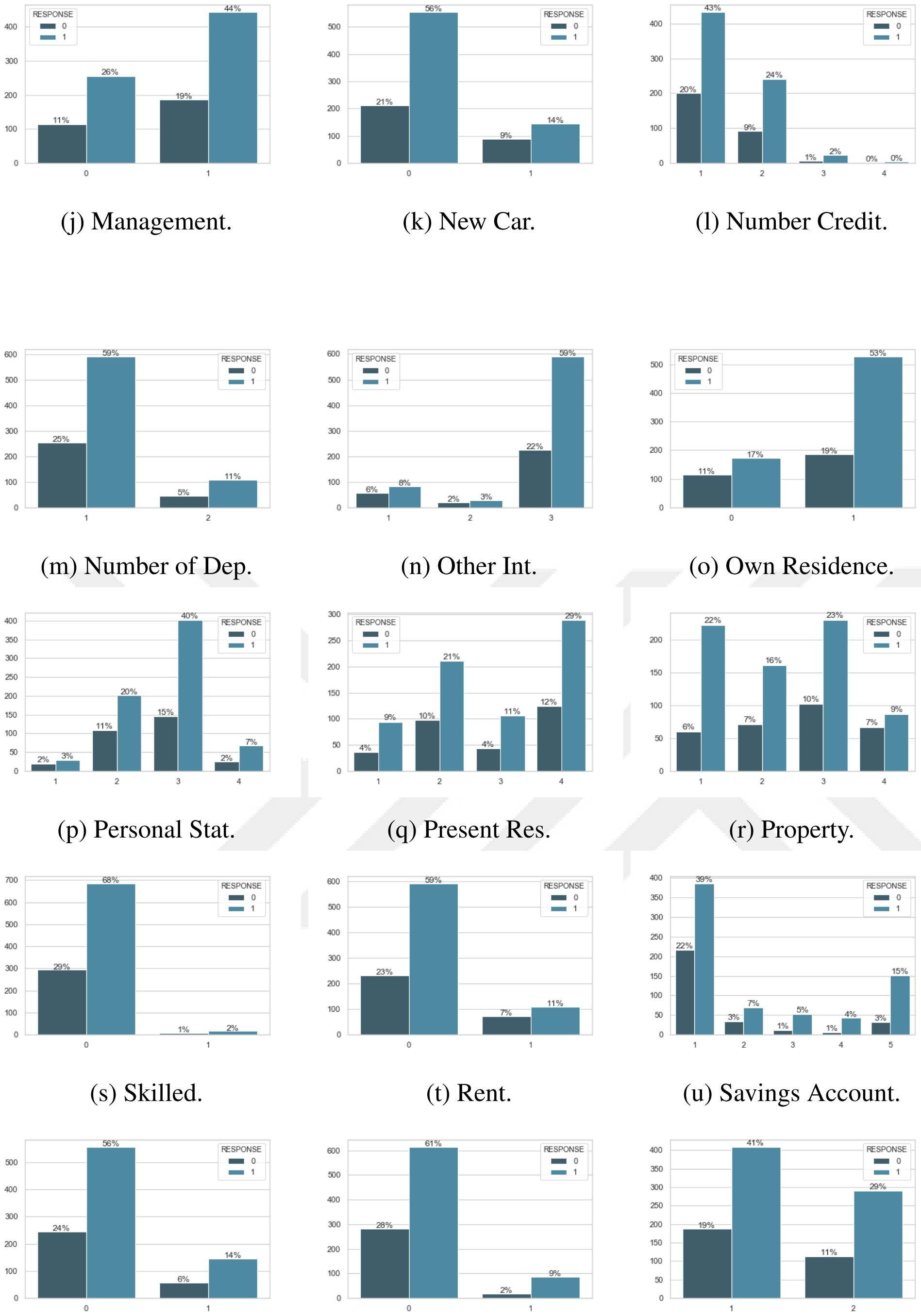
(e) Checking Account.

(f) Domestic.

(g) Credit History.

(h) Employment.

(i) Foreign.



(j) Management.

(k) New Car.

(l) Number Credit.

(m) Number of Dep.

(n) Other Int.

(o) Own Residence.

(p) Personal Stat.

(q) Present Res.

(r) Property.

(s) Skilled.

(t) Rent.

(u) Savings Account.

(v) Unskilled.

(w) Used Car.

(x) Telephone.

Figure 4.4.6: German Data Attributes.

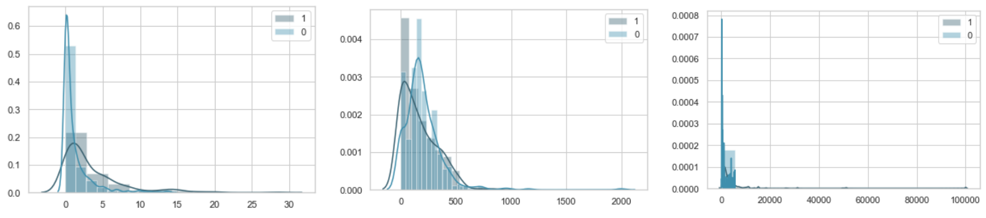
Figure 4.4.6 includes the continuous and categorical attributes in German dataset. Three of the features 4.4.6b, 4.4.6c and 4.4.6a follows long tail distribution, and it is seen that distribution of each class values are very similar. The remaining 21 attributes are categorical. Their class percentages are also added to top of the category bars.

Table 4.3: German Credit Data Features

Attribute	Description	Type
Attribute 1	Age in years	Numerical
Attribute 2	Duration in month	Numerical
Attribute 3	Credit history	Categoric
Attribute 4	Purpose	Categoric
Attribute 5	Credit amount	Numerical
Attribute 6	Savings account/bonds	Categoric
Attribute 7	Present employment since	Categoric
Attribute 8	Instalment rate	Numerical
Attribute 9	Personal status and sex	Categoric
Attribute 10	Other debtors/guarantors	Categoric
Attribute 11	Present residence since	Numerical
Attribute 12	Property	Categoric
Attribute13	Status of existing checking account	Categoric
Attribute 14	Telephone	Categoric
Attribute 15	Other instalment plans	Categoric
Attribute 16	Housing	Numerical
Attribute 17	Number of existing credits at this bank	Categoric
Attribute 18	Foreign worker	Numerical
Attribute 19	Job	Categoric
Attribute 20	Number of dependants	Categoric

4.4.2 Australian Dataset

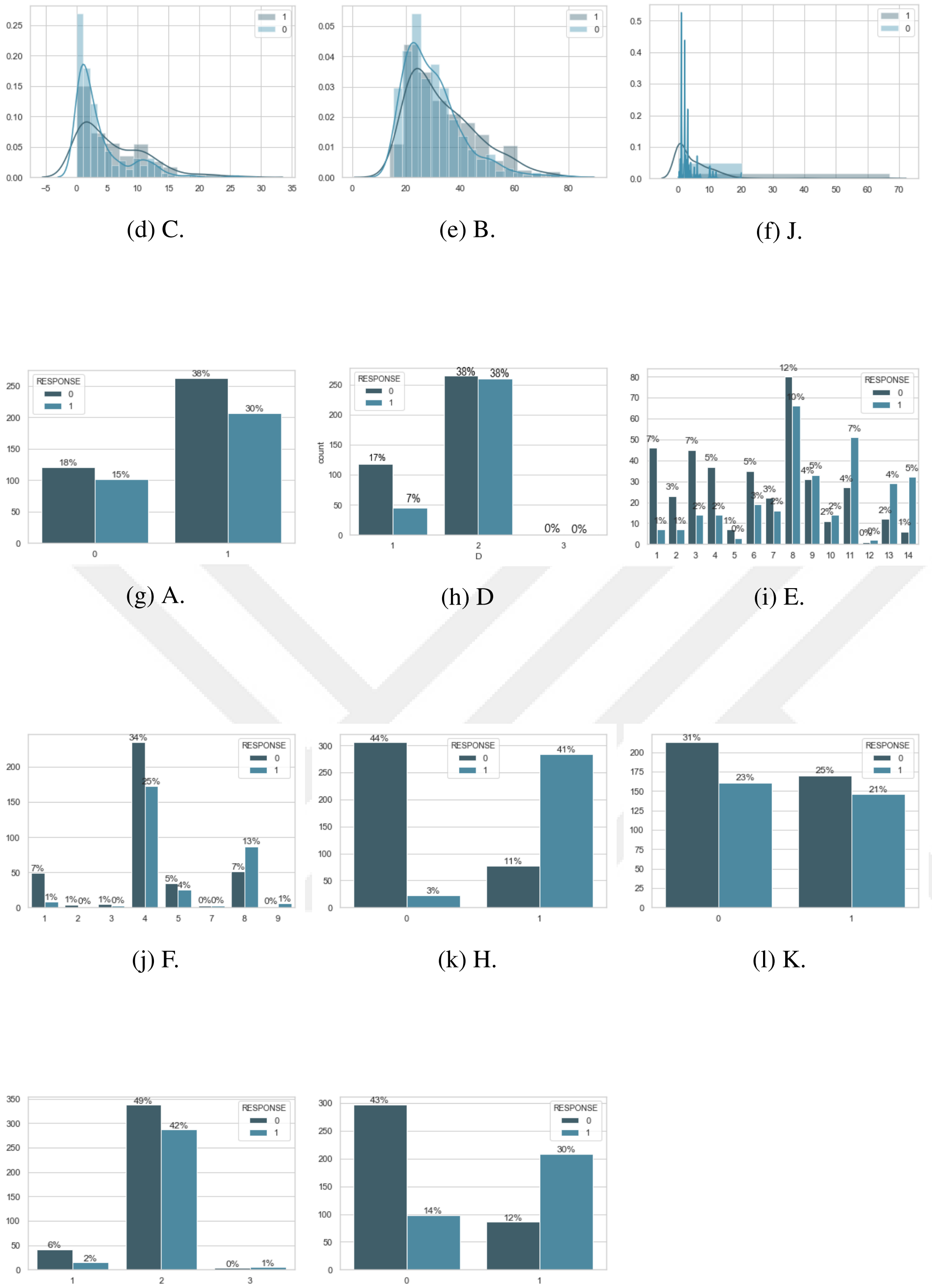
Same as the German dataset, this data is also taken from the UCI machine learning repository [23]. The data set consists of credit card applications. The data source and the attributes contained within it are represented by letter codes for confidentiality reasons. The data consists of six numeric and 8 categorical attributes. Percentage of the missing values in the data is 5%, and these values are changed by the attribute's mode if it is a categorical data, and the values are replaced by the attribute's mean if it is continuous. The data includes 690 observations, 307 of which belong to class 0 and 383 of them belong to class 1. The graphs of each features can be found in Figure 4.4.7.



(a) G.

(b) M.

(c) N.



(d) C.

(e) B.

(f) J.

(g) A.

(h) D

(i) E.

(j) F.

(k) H.

(l) K.

(m) L.

(n) I.

Figure 4.4.7: Australian Data Attributes.

In Figure 4.4.7, it is seen that the Australian dataset includes both categorical and continuous features. The subfigures 4.4.7a, 4.4.7b, 4.4.7c, 4.4.7d, 4.4.7e and 4.4.7f demonstrate the continuous attributes which follow a long tailed distribution. Rest of the subfigures show the class percentages of each category.

Table 4.4.4: Australian Credit Data Features

Attribute	**Type and/or value**
Attribute 1 (A)	1, 0 categorical
Attribute 2 (B)	Continuous
Attribute 3 (C)	Continuous
Attribute 4 (D)	1, 2, 3 categorical
Attribute 5 (E)	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 categorical
Attribute 6 (F)	1, 2, 3, 4, 5, 6, 7, 8, 9 categorical
Attribute 7 (G)	Continuous
Attribute 8 (H)	1, 0 continuous
Attribute 9 (I)	1, 0 continuous
Attribute 10 (J)	Continuous
Attribute 11 (K)	1, 0 continuous
Attribute 12 (L)	1, 2, 3 continuous
Attribute 13 (M)	Continuous
Attribute 14 (N)	Continuous
Attribute 15	1, 0 class attribute

CHAPTER 5

MACHINE LEARNING ALGORITHMS

In this part, mathematical and statistical aspects of eight commonly used ML algorithms in literature, are comprehensively provided. Besides, the hyper-parameters requiring mathematical explanation, and included in the algorithms, are detailed.

5.1 Gaussian Naïve Bayes

Naïve Bayes classifier is a statistical classifier that has a white box nature which means the ML process is transparent and clear understanding of how they behave [39]. Bayesian classification method named after Thomas Bayes is based on Bayesian Theory [28]. In this approach, the posterior probability of every samples is calculated by considering their particular classes [26]. This classifier assign the most probable class to a specified instance defined by its attribute vector. In this algorithm, assuming that features are independent for corresponding class can greatly simplify the learning process. For this reason, the algorithm is called Naïve [53]. Some machine learning classifier comparison studies have found that a simple Bayesian learning algorithm known as Naïve Bayes Classifier has a competitive performance with and neural network classification algorithms. Also, when applied to big databases, Bayesian classifiers showed good accuracy and execution time efficiency [28].

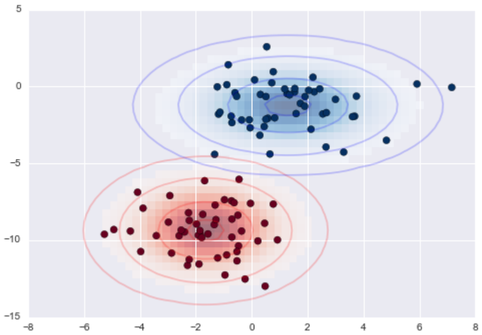


Figure 5.1.1: Schematic for GNB. [64]

In the following, we give general intuition for GNB.

The conditional probability of an example $E = (x_1, x_2, ..., x_n)$ belonging a class c is

$$P(c|E) = \frac{P(E|c)P(c)}{P(E)} \tag{5.1}$$

Due to the samples in each feature are assumed to be normally distributed and to deal with the continuous variables, the Gaussian transformation is required, so calculation of P(xlc) is

$$P(E|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(x-\mu_c)^2}{2\sigma_c^2}} \tag{5.2}$$

In the above formula 5.2, μ_c is the mean of class c and σ_c is the standard deviation of class c. With this transformation the conditional probability of numeric attributes can be calculated. E is classified as the class $C = +$ if and only if the estimation

$$f_b(E) = \frac{P(C=+|E)}{P(C=-|E)} \geqslant 1, \tag{5.3}$$

is satisfied, where $f_b(E)$ is called a Bayesian classifier. With the assumption of all attributes are independent given the value of the class variable; that is,

$$P(E|c) = P(x_1, x_2, ..., x_n|c) = \prod_{i=1}^{n} P(x_i|c), \tag{5.4}$$

the final form of the classifier is then:

$$f_{nb}(E) = \frac{P(C=+)}{P(C=-)} \prod_{i=1}^{n} \frac{P(x_i|C=+)}{P(x_i|C=-)}. \tag{5.5}$$

The corresponding classifier $\hat{c}$ is:

$$\hat{c} = argmaxP(c|x) \tag{5.6}$$

Because $P(x)$ is the same for each, $\hat{c}$ is:

$$\hat{c} = argmaxP(c|x) = argmaxP(c)P(x|c) \tag{5.7}$$

The function $f_{nb}(E)$ is called a GNB classifier [7].

5.2 Support Vector Machines

The Support Vector Machine (SVM) converts the initial training data into a higher dimension, it employs a nonlinear mapping. Within this new dimension, it searches for the linear optimal separating hyperplane (i.e., a decision boundary separating the tuples of one class from another). In this high dimensional space, the algorithm search decision boundary which separates linearly the variables belong to different classes. This optimum separating boundary is called hyperplane. The SVM decides the hyperplane location and margins by using the support vectors which are training points. For simplicity, consider a two class example where the classes are linearly separable.

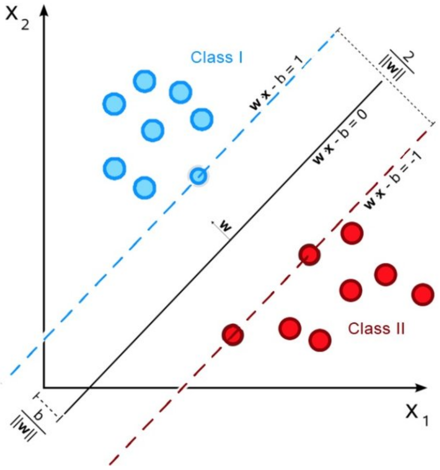


Figure 5.2.1: Schematic for SVM [67]

Let assume that the data set D be given as $(X_1, y_1), (X_2, y_2), ..., (X_{|D|}, y_{|D|})$,

where X_i is the set of training variables with associated class labels y_i.

Corresponding to the classes each y_i can take one of two values either $+1$ or -1. There is an infinite number of dividing hyperplanes that could be drawn for a linearly separation of two-class dataset. SVM deals with discovering the optimal hyperplane, that is one on earlier invisible variables that will have lesser classification error is found by seeking for the largest marginal hyperplane. We expect the hyperplane with the larger margin to be more accurate at classifying future data tuples than the hyperplane with the smaller margin. In this approach, the hyperplane that has bigger margin to be more precise than the hyperplane that the lower margin when classifying future data inputs is aimed. Definition of margin, we can say that the shortest distance from a hyperplane to one side of its margin is equal to the shortest distance from the hyperplane to the other side of its margin, where the "sides" of the margin are parallel to the hyperplane. The shortest range from a hyperplane to one side of its distance is equal to the shortest range from the hyperplane to the other side of its distance, where the distance borders are parallel to the hyperplane is the definition of margin.

A dividing hyperplane can be identified as

$$W.X + b = 0, \tag{5.8}$$

where W is a weight vector, in that, $W = (w_1, w_2, \ldots, w_n)$; n is the number of features; and b is a scalar, often referred to as a bias. With considering two input attributes, A_1 and A_2, and x_1 and x_2 are the values of this features.

Redesigning Eq: 5.8 we get,

$$w_0 + w_1x_1 + w_2x_2 = 0. \tag{5.9}$$

If any point lies above the dividing hyperplane, this points satisfy the following condition

$$w_0 + w_1x_1 + w_2x_2 > 0. \tag{5.10}$$

Alike in the above inequality 5.10, if any point lies below the dividing hyperplane satisfies the below condition

$$w_0 + w_1x_1 + w_2x_2 < 0. \tag{5.11}$$

The weights can be adapted to allow the hyperplanes defining the margin's borders to be written as

$$H_1 : w_0 + w_1x_1 + w_2x_2 \geq 1 \; for \; y_i = +1, \tag{5.12}$$

$$H_2 : w_0 + w_1x_1 + w_2x_2 \leq -1 \; for \; y_i = -1. \tag{5.13}$$

Combining these two inequalities 5.12 and 5.13, we get

$$y_i(w_0 + w_1x_1 + w_2x_2 \geq 1), \; \forall i. \tag{5.14}$$

The training features which is on hyperplanes H_1 or H_2 are defined as support vectors. With this, a formula can be obtained for the size of the maximal distance. The distance between the dividing hyperplane to any point on H_1 is $\frac{1}{||W||}$,where $||W||$ is the Euclidean norm of W. This distance range is same as the distance from any point on H_2 to the dividing hyperplane. Hence, the maximal margin is $\frac{2}{||W||}$. It is a convex quadratic optimization problem needed for maximal marginal hyperplane in this situation. Reorganizing inequality in Eq:5.14 with using a Lagrangian formulation and then solving for the solution using Karush-Kuhn-Tucker (KKT) condition, we get

$$d(X^T) = \sum_{i=1}^{l} y_i\alpha_iX_iX^T + b_0, \tag{5.15}$$

where y_i is the class label of support vector X_i; X^T is a test tuple; α_i and b_0 are numeric parameters that are calculated by the optimization process of SVM algorithm, l is the support vector quantity and the α_i are Lagrangian multipliers. In nonlinear cases, it is needed to find a way to separate classes. Mapping variables into higher dimensional space enables to deal with separation of this cases.

To overcome with this problem, we can apply a kernel function, such as $K(X_i, X_j)$, to the original input data. That is,

$$K(X_i, X_j) = \phi(X_i).\phi(X_j). \tag{5.16}$$

After applying this trick to Eq: 5.15, we can proceed to find a maximal separating hyperplane. With this our equation become

$$d(X) = \sum_{i=1}^{l} y_i \alpha_i \phi(X_i).\phi(X) \tag{5.17}$$

form [28].

5.2.1 SVM with Sigmoid Function

SVM generates an uncalibrated value that is not a likelihood. Assume an SVM's unthresholded output be

$$f(x) = d(x) + b, \tag{5.18}$$

where $d(x)$ given in Eq: 5.15 lies in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{F}$ induced by a kernel k. Training an SVM minimizes an error function that penalizes an approximation to the training misclassification rate plus a term that penalizes the norm of d in the RKHS:

$$C \sum_i (1 - y_i f_i) + \frac{1}{2} ||d|| \mathcal{F}, \tag{5.19}$$

where $f_i = f(x_i)$. In order to decrease a bound on the test data classification error ratio, minimization of error function is required. An extra benefit of this error function is that minimizing it will result in a sparse machine where the final machine uses only a subset of possible kernels. By sorting the attribute space $\mathcal{F}$ to an orthogonal direction to the dividing hyperplane, and rest of all the $N - 1$ dimensions of the attribute space, SVMs can map the outputs into the probabilities.

t which is a transformed version of $f(x)$ parameterizes the orthogonal direction, as a vector u parameterizes all the other directions. The posterior probabilities is generally dependent on t and u : $P(y|t, u)$. By summing cosine values, probabilities could be calculated

$$P(y|t, u) = a_0(u) + \sum_{n=1}^{N} a_n(u) cos(nt). \tag{5.20}$$

Regularization function is minimized with usage of cosine expansion coefficients, which can be define as a linear equation for the a_n. To fit posterior probability $P(y|f)$, a parametric model is employed. The conditional densities of margins with respect to classes are exponential,so by use of sigmoid function we have

$$P(y|f) = \frac{1}{1 + exp(Af + B)}. \tag{5.21}$$

By using maximum likelihood calculation, A and B parameters could be fit form the set (f_i, y_i).

Assume that identify a training set as (f_i, t_i). In this tuple the t_i are target probabilities identified as follow

$$t_i = \frac{y_i + 1}{2}. \tag{5.22}$$

By minimizing the negative log likelihood of the training data, A and B are calculated [50]. It is a cross-entropy error function

$$min \left\{ -\sum_i t_i log(P_i) + (1 - t_i) log(1 - P_i) \right\}, \tag{5.23}$$

where

$$P_i = \frac{1}{1 + exp(Af_i + B)}. \tag{5.24}$$

5.2.2 Kernel Functions

One of the main reason why SVMs gains popularity is that it can simply increase dimensionality to solve nonlinear classification tasks. The fundamental concept behind kernel methods for dealing with such linearly inseparable samples is to generate nonlinear combinations of the initial attributes to project them through a mapping function K into a higher-dimensional space where they turn into linearly separable [52].

a) Linear Kernel

The Linear Kernel (LK) uses Pearson standard correlation to assess the relation in a pair of instances [32].

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}. \tag{5.25}$$

b) Radial Basis Function

The Radial Basis Function (RBF) or simply called the Gaussian kernel is one of the most commonly used function. The function is defined as follows

$$K(x_i, x_{i'}) = exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2). \tag{5.26}$$

where $\gamma = \frac{1}{2\sigma^2}$ is a regulatory parameter can be optimized. When the gamma parameter values is increased, it brings a tighter and bumpier decision boundary [52].

c) Sigmoid Kernel

Sigmoid kernel (SK) is also knows as The hyperbolic tangent kernel. It is a nonlinear transformation of separating hyperplane.

$$K(X_i, X_j) = tanh(\kappa X_i.X_j - \delta) \tag{5.27}$$

Here, κ and δ are shape values for separation [44].

d) Polynomial Kernel

A polynomial kernel (PK) of degree d,

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d. \tag{5.28}$$

where d is a positive integer. Using such a kernel with $d > 1$ leads to a much more flexible decision boundary. It causes SVM to work in a higher-dimensional space by including polynomials of degree d [32].

5.3 Decision Tree

A decision tree (DT) consists of internal nodes, branches and leaf which seems like a tree structure. In each internal node indicates a test on a feature, all leaves are symbolize a class label, and each branch defines an outcome. In top of the tree structure, there is a root node which the splitting is started at this point [28].

DT's are structured general to specific information by usage of training data. In a situation that whole the examples in a node belong to same class, there is no need for a splitting because when a random dividing is occurred, it does not decrease the impurity ratio which will be explained later. Hence, the node is turn to a leaf node and the branching is completed. In another scenario, we consider the examples belong to different two classes half and half. In such a situation, it means that the node is not pure, and with the test splitting will be occurred. This is continuously repeated until there is no possible decrease for the impurity ratio. This process will be done for all internal nodes till the tree structure turn out to fully discriminating form. If a DT is fully discriminating all the information in the training set, it is highly possible that the tree is overfitted to the data. Another words, the tree could be specified to the outliers, and biased. To overcome this problem, pruning approach is taken into account. This approach eliminate the subtrees that when they split from parent node, but there is not a decrease in impurity rate. This elimination decisions are made by error estimation techniques [3].

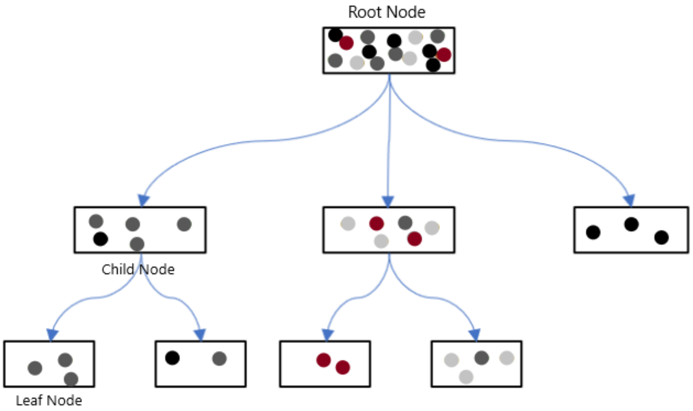


Figure 5.3.1: Schematic for DT

5.3.1 Splitting Criterions

Splitting criterions are used to measure impurity ratio of a node. In other words, these two criterions evaluate how much information the instances that are included in a node carry.

a) Gini Criterion

Let us define a specific node as t, and specify a class by j in a DT. Estimated class probabilities are $P(j|t)$ with respect to a node t, and impurity measurement on a node is calculations

$$i(t) = \phi(P(1|t), ..., P(j|t)),\ j = 1, \ldots, j. \tag{5.29}$$

When a splitting is occurred in a node t, it is done by searching the most reduction point in impurity rate. And this is applied by usage of above equation. The classical form of the Gini function is

$$\phi(P_1, ..., P_j) = -\sum_j P_j \log P_j. \tag{5.30}$$

To use this function in classification tusk, it needs to be redefined. Hence the appropriate design is

$$i(t) = \sum_{j \neq i} P(j|t)P(i|t). \tag{5.31}$$

Besides, the form in the below is more useful for calculation

$$i(t) = \left(\sum_{j} P(j|t) \right)^2 - \sum_{j} P^2(j|t) = 1 - \sum_{j} P^2(j|t). \tag{5.32}$$

Additionally, the Gini index classifies a variable selected randomly to j class with using likelihood $P(i|t)$, and it does not use the plurality rule for this purpose [11].

b) Entropy

The information theory uses data probabilities on minus the logarithm base 2 to measure the information in bits. When randomly selecting an example which is a member of a class C_j in a node belongs to a set D, it carries a probability of message such as

$$\frac{freq(C_i, D)}{|D|} \tag{5.33}$$

and the information values that it has calculated by

$$-\log_2 \left(\frac{freq(C_i, D)}{|D|} \right) \tag{5.34}$$

in bits. Here, D is the set of examples, $freq(C_i, D)$ is the the number of examples that belong to C_i class, and $|D|$ symbolize the number of samples in D.

For instance, there are four equally likely instances in a node. In such a situation, the information that carried by any of the instance is equal to $-\log_2(1/4)$ or 2 bits.

To calculate the expected information of a message that carry belonging to a class membership, the classes in proportion to its frequencies in D is summed up with the follows

$$\text{info}(D) = -\sum_{j=1}^{k} \frac{freq(C_j, D)}{|D|} \times \log_2 \left(\frac{freq(C_j, D)}{|D|} \right) \tag{5.35}$$

$info(t)$ is required for identifying the examples in t node while training process is occurred. $info(t)$ measures the average quantity of information.

After a splitting in t is occurred, and the splitting is in accordance with the n outcomes of a test X. By using the weighted summing up the subsets, the expected information which is required could be calculated as follows

$$info_x(t) = \sum_{i=1}^{n} \frac{|t_i|}{|t|} \times info(t_i). \tag{5.36}$$

Calculation of information gained by splitting in t in accordance with the test X is measured with

$$gain(X) = info(t) - info_x(t) \tag{5.37}$$

value. After that, the gain criterion spots a test that is most suited to maximize the information gain. This approach is also known as Mutual Information (MI) that measures the relation between the class and the test [11].

5.3.2 Gain Ratio Criterion

While gain criterion produce highly effective results, this approach has a quite biased in favor of tests in lots of outputs, which is a major deficiency. The bias intrinsic in the gain criterion can be rectified through some sort of standardization in which the obvious gain attributable to tests with many outputs is adjusted. By taking into account containing information of a case-related message that does not indicate the class to which the case belongs, but the test result.

The description of $info(S)$ is as follow

$$split\ info(X) = -\sum_{i=1}^{n} \frac{|t_i|}{|t|} \times \log_2\left(\frac{|t_i|}{|t|}\right). \tag{5.38}$$

Above equation (5.38) reflects the possible information occurred by splitting t into n sub nodes. However, information gain calculates the classification related information that subject occurred in the same splitting. Identifying the proportion of information that is occurred in splitting the way

$$gain\ ratio(X) = gain(X)/split\ info(X) \tag{5.39}$$

is beneficial for classification. When division information is resulted in very small ratio, this indicates that the ratio is not stable. To overcome this problem, the gain ratio criteria choose a test to maximize the ratio, with respect to value that the information gain is need to be higher than at least as large as the mean of the gain that is over all tests analyzed [11].

5.3.3 Pruning

For each branched subtrees $T \leq T_{max'}$, let's define its complexity as $\tilde{T}$. It is the number of terminal branch points in T, and also assume that $a \geq 0$ be a real valued number, which is the complexity parameter. The cost of complexity is described by $R_\alpha(T)$ as

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}|. \tag{5.40}$$

Here, α value give the cost of complexity for each terminal node, while $R_\alpha(T)$ is stand for complexity measurement the misclassication cost of the additional tree.When α value is low, $T(\alpha)$ is large and a high number of terminal nodes will bring small penalty. In other word, while each terminal nodes includes only one example, every instance will be assigned without any error and $R(T_{max})$ will be equal to zero. The subtrees in a tree are pruned by the previous one to upwardly. The pruning is done in a sequential process from bottom to the top root node of the tree, $T_1, T_2, ...,$. The core mechanism of minimal cost-complex pruning is operated by cutting the weakest branch link form the subtree. The subbranch of T_t, a node $t \in T_1$ denote by $\{t\}$, include the single node $\{t\}$.

The definition for a single node the cost of complexity is

$$R_\alpha(t) = R(t) + \alpha. \tag{5.41}$$

We can organize this definition in more suitable way for any branch T_t

$$R_\alpha(T_t) = R(T_t) + \alpha|\tilde{T}_t|. \tag{5.42}$$

When the statement

$$R_\alpha(T_t) < R_\alpha((t)), \tag{5.43}$$

is valid, the subbranch T_t carry smaller cost of complexity than a single node $\{t\}$. While the cost of complexity values of the two are equal at some breakeven point of α. At such a points the branching is smaller for subbranch $\{t\}$ than T_t while they carry the same amount of cost complexity. Hence, $\{t\}$ is preferable to T_t. With solving the inequality

$$R_\alpha(T_t) < R_\alpha((t)), \tag{5.44}$$

to search this breakeven point of α, we can reorganize this inequality as

$$\alpha < \frac{R(t) - R(T_t)}{|\tilde{T}_t| - 1}. \tag{5.45}$$

By defining the weakest link as $\bar{t_1}$ in T_1 the node such that

$$g_1(\bar{t_1}) = \underset{t \in T_1}{min}\, g_1(t). \tag{5.46}$$

While the value α increases, the node $\bar{t_1}$ will be the weakest link, it is the first node such that $R_\alpha(t)$ is equal to $R_\alpha(T_t)$. Then it is could be said that t_1 is preferable to $T_{\bar{t_1}}$, and α_2 is the value of α at which equality occurs. By pruning the branch $T_{\bar{t_1}}$, we can define a new tree $T_2 < T_1$ as

$$T_2 = T_1 - T_{\bar{t_1}}. \tag{5.47}$$

If we carry on this way, we will have a reduction sequence in subtrees

$$T_1 > T_2 > T_3 > ... > (t_1). \tag{5.48}$$

With choosing a specified number of $N(2)$ of instances randomly from L to determine the test example L_2. L_1 is used for the new learning examples. By usage of L_1, the tree T_{max} is grown to reach to highest possible impurity and it is upwardly pruned ot have the subtree sequence $T_1 > T_2 > T_3 > ... > (t_1)$. That is, the T_k sequence of trees is constructed and the terminal nodes assigned a classification without ever seeing any of the cases in L_2. By not using the cases in L_2, the T_k tree sequence is built and the terminal nodes are assigned to classes. Then, with taking the cases in L_2 and dropping them though T_r. Every tree T_k appoints a estimated classification to every cases in L_2. Misassigned cost of T_k operating on L_2 can be determined by usage of cases in L_2 known the real class. This generate the estimation of $R^{ts}(\bar{T}_k)$.

$$Q^{ts}(i|j) = \frac{N_{ij}^{(2)}}{N_j^{(2)}} \tag{5.49}$$

is calculated by the proportion of test instances class j cases that the tree T classifies as i.

$$R^{ts}(j) = \sum_{i} c(i|j) Q^{ts}(i|j) \tag{5.50}$$

Here, $c(i|j)$ is the misclassification cost of a class k subject as a class i. For the priors this gives the calculation

$$R^{ts}(T) = \sum_{k=1}^{K} R^{ts}(k)\pi(k). \tag{5.51}$$

If the priors are calculated from the data, L_2 can be used to calculate them as $\pi(k) = \frac{N_k^{(2)}}{N^{(2)}}$. The calculation can be simplified as the below form

$$R^{ts}(T) = \frac{1}{N^{(2)}} \sum_{i,k} c(i|k) N_{ik}^{(2)}. \tag{5.52}$$

By taking the average the cost of misclassification calculated in L_2 for each instance dropped by T [11]. The estimates of the test sample can be used to select the correct size tree T_{k0} by the rule

$$R^{ts}(T_{k0}) = \underset{k}{min} R^{ts}(T_k). \tag{5.53}$$

5.3.4 Cross-Validation Estimates

In validation part [11], the cross validation calculations are described in detail. However, this method is used in the growing process of tree also. Therefore the process is briefly explained in this part. The learning instances dataset L is splitted by randomly selecting to k subsets. The subsets L_k, $k = 1, ..., k$ each containing the same amount of instances as nearly as possible in k-fold cross-validation. The kth learning set is defined by

$$L^{(k)} = L - L_k, k = 1, ..., k. \tag{5.54}$$

In k-fold cross-validation, k trees are built together with the main tree grown on L. By using the learning set $L^{(k)}$, the k-th assistant tree is constructed. Start by constructing

k overly large trees $T_{max}^{(k)}, k = 1, ..., k$, as well as T_{max}, using the criterion that the splitting continues until nodes are pure or have fewer cases than N_{min}. With usage of dividing until to nodes are fully pure or get lesser cases than N_{min} criteria, begin by building k overly large trees $T_{max}^{(k)}, k = 1, ..., k$, such as closer to T_{max}.

Assume that $T(\alpha), T^{(k)}(\alpha), k = 1, ..., k$, be related to minimal cost-complexity subtrees of $T_{max}, T_{max}^{(k)}$ for each calculation of the cost complexity α. The trees $T_{max}^{(k)}$,for every k, $T^{(k)}(\alpha)$ is built the instances in L_k unseeingly. Hence, the instances in L_k can perform as an independent test set for the tree $T^{(k)}(\alpha)$. While fixing the value of the complexity α, process L_k top down the tree $T_{max}^{(k)}, k = 1, ..., k$. For each value of k, i, j, define $N_{ij}^{(k)}$ = the number of class j cases in L_k classified as and set

$$N_{ij} = \sum_{K} N_{ij}^{(k)}, \tag{5.55}$$

so N_{ij} is the total number of class j test cases classified as i.

By defining $N_{ij}^{(k)}$ = the number of class j cases in L_k for each values of k, i, j, is classified as and constituted

$$N_{ij} = \sum_{K} N_{ij}^{(k)}, \tag{5.56}$$

so N_{ij} is the total number of class j test instances assigned as i. Every instances in L is used only one times in test set L_k. The total number of cases in L which belongs to class j represented with N_j. Than, implement the earlier steps for each trees. The application is simplified by the fact that although α may vary continuously, the least cost complexity tree built on L are equal to T_h for $\alpha_h \leq \alpha < \alpha_{h+1}$. In order to $\alpha\prime_h$ is the geometric midpoint of the interval where $T(\alpha) = T_h$, put

$$\alpha\prime_h = \sqrt{\alpha_h \alpha_{h+1}} \tag{5.57}$$

After that, put

$$R^{cv}(T_h) = R^{cv}(T(\alpha\prime_h)), \tag{5.58}$$

where $R_{cv}(T(\alpha\prime_h))$. By using the $R^{cv}(T_{h0})$ as an calculation of the misclassification cost, choose the tree T_{h0} such that

$$R^{cv}(T_{h0}) = \underset{h}{min} R^{cv}(T_h). \tag{5.59}$$

5.4 Random Forest

A large number of classification trees are growing in Random Forests (RF). These trees are based on Breiman's CART tree approach which requires placing the input variable down each of the trees in the forest to classify newly added test object from an input variable. Every tree in the forest give a classification vote to classify an instance to appropriate category. The forest select the class that the instance belong by considering the most votes over all the trees. In training process, the amount of instances in the training set N is sampled randomly by replacement from the initial dataset. In each step, the training set is sampled to build a new tree. The number of input attributes is M, and a number m is specified with respect to $m < M$ such that at each node, m attributes are chosen randomly out of the M and the optimum split on these m is used to split the node. During the forest contraction, the m value is fixed. In the forest, every tree is constructed in the maximum possible extent without pruning. The correlation and strength between any two trees are reduced by decreasing the amount of inputs m, and inversely increasing the number of input increase correlation and error rate. The sampling in training instances for a new tree is nearly one third of the examples are left of the set. Out of Bag (OOB) technique is applied as trees are added to the forest to obtain a running unbiased calculation of the misclassification. This approach is also applied to obtain variable importance calculation. Whole data is run down the tree and for each pair of instances affinities are estimated after every trees are built. In a case of two instances are in the same terminal node, their affinities are increased by one. It provides calculations of which variables are essential in assignation process. The affinities are normalized by splitting with the amount of trees at the end of execution. Proximities are used in replacing missing data, locating outliers, and producing illuminating low-dimensional views of the data. In the cases of replacement of missing data, locating anomalies in data, and illustrative low dimensional view of the data, the affinities are applied [10] and [12].

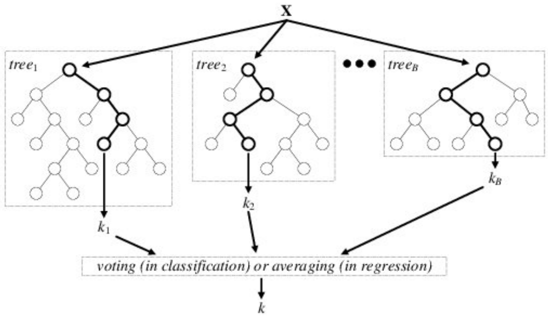


Figure 5.4.1: Schematic for RF. [65]

A random vector Θ_k is generated for the k-th tree independent form the earlier random vectors $\Theta_1, ..., \Theta_{k-1}$. k-th tree and the earlier trees have the same distribution. A tree is constructed with the Θ_k and training samples. The built classifier is defined as $h(x, \Theta_k)$ where x is an input vector. The dimensional structure of Θ relies on usage of it in the tree built. Θ parameter is formed by a number of independent random integers in the range between 1 and K in random division selection. The margin function is defined as

$$mg(X, Y) = av_k I(h_k(X) = Y) - \max_{j \neq Y} av_k(h_k(X) = j) \tag{5.60}$$

with the given training set selected randomly from the distribution of the random vector Y, X , and ensemble of classifiers $h_1(x), h_2(x), ..., h_k(x)$.

Here, $I(\cdot)$ is the indicator function. The margin estimates the scope that the average number of votes exceeds mean of the votes for any other class at X, Y for the right class. Greater margin leads to greater confidence in the rating. The generalization error of is defined as follows

$$PE^* = P_{\mathrm{X},Y}(mg(\mathrm{X}, Y) < 0). \tag{5.61}$$

Here, $P_{\mathrm{X},Y}$ is the probability over the X, Y space. With the notation $h_k(\mathrm{X}) = h(\mathrm{X}, \Theta_k)$, and a large number of trees follows the rule Strong Law of Large Numbers. It indicates that as the number of trees increases, for almost surely all sequences

$\Theta_1, ...PE^*$ converges to

$$P_{\mathrm{X},Y}(P_\Theta(h(\mathrm{X},\Theta)=Y)-\max_{j\neq Y}P_\Theta(h(\mathrm{X},\Theta)=j)<0). \tag{5.62}$$

5.4.1 Bootstrap

The bootstrap [10] is an extremely strong and commonly feasible statistical tool that can be used to measure the uncertainty related to given estimator or method of statistical learning. The bootstrap's strength lies in the fact that it can be applied simply to a broad spectrum of techniques of statistical learning. We can add the complete sum that the Gini index is reduced by splits over a specified predictor in the context of bagging classification trees, averaged over all B trees. Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the bagging variance of a statistical learning method. General aim of this process is redaction in statistical learning method's variance in bootstrap aggregation, or bagging. The variance of the mean Z of the instances a set of n independent $Z_1, ..., Z_n$ each have the variance σ^2 is given by σ^2/n. That is to say, it decreases σ^2 by averaging a set of instances. With repeated examples from a training dataset, bootstrap can be applied. B various bootstrapped training data sets are produced in this strategy. After that this technique is trained on the bootstrapped training set ot get $\hat{f}^b(x)$, and lastly by averaging these predictors in order to get

$$\hat{f}_{bag}(x)=\frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x). \tag{5.63}$$

5.4.2 Out of Bag Error Estimation

In bagging, the crucial point is that trees are recursively fit to bootstrapped subsets of instances. In this approach, approximately two third of the instances are used to generate bagged trees. OOB instances rest of the examples one third of the set are not used to fit a specified bagged tree. Using each of the trees in which that instances are OOB, the assessment of the i-th instance can be analyzed.

This approach produce for the i-th instance around $B/3$ predictions. By taking majority vote to achieve a single forecast for the i-th instance. This brings about a single OOB forecast for the assessment of i-th instance. By this way, misclassification can be estimated for each of the n instances. The outcome from the OBB classification error is an effectual calculation of the test set error for the bagged algorithm as the reaction for every instance is predicted by only usage of the trees that are not fit using those instances [12].

5.4.3 Variable Importance Measures

Explanatory interpretation is a much harder task for bagged trees approach than a decision tree. In this algorithm, a general overview of the significance of each predictor can be obtained using the Gini index. Total amount of Gini index that reduced by division over a specific predictor that is taken the average over B trees. In bagging the central aim is lowering the variance of a statistical learning algorithm [10].

5.5 XGBoost

XGBoost is an improved version of Gradient Boosting algorithm which based on CART decision tree approach. In this method, regularized objective and sparsity-awareness are two of the novel approach in gradient boosting. This algorithm can handle missing inputs, frequently zero values and binary data with default direction approach. The other powerful sides of this algorithm are faster than other most popular machine learning methods and preventing overfitting by shrinkage and column subsampling approaches.

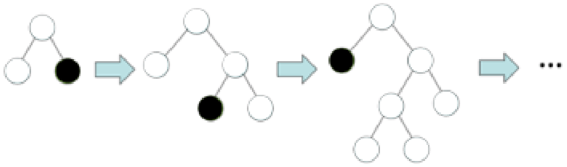


Figure 5.5.1: Schematic for Structure.

[1] Source of Figure 5.5.1: https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2017/06/11194227/depth.png

5.5.1 Regularized Objective

Let's $\mathcal{D} = \{(x_i, y_i)\}$ is a set of inputs with *n* examples and *m* features. $\{x_i \in \mathbb{R}^m\}$'s are the vector of independent variables and $\{y_i \in \mathbb{R}\}$'s are corresponding values.

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}, \tag{5.64}$$

where $\mathcal{F} = \{f(x) = w_{q(x)}\}(q : \mathbb{R}^m \rightarrow T, w \in \mathbb{R}^T)$ is the space of CART tree functions, and K is the number of additive functions of a tree ensemble model which makes prediction. In the above mapping, q represents the structure of each tree according to leaf index, and T is number of leaves corresponding to a specific tree. Every f_k have an independent tree with structure *q* and leaf weights w. In every tree, each of the leaves contains a continuous score w_i, it is the score of i-th leaf, which contrary to decision trees. q will be used as decision rules in trees to classify a specific example, and final forecasting is calculated by summing up w the corresponding scores in a leaf. With minimization of regularized objective (Eq: 5.65), function set used in the model is learned by

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{5.65}$$

where l measures the error between the predicted $\hat{y}_i$ and the actual value of y_i, and it is a differentiable convex function.

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2 \tag{5.66}$$

Above term in Eq: 5.66 Ω penalizes the complexity of the CART decision tree functions. To prevent over-fitting conclusive learned weights, the penalization term helps for smoothing. Intuitively, a model using simple and predictive functions will tend to be selected by the regularized objective. Regularization objective and learning process employed in this algorithm are less complex than regularized greedy forest algorithm which is a similar regularization technique. In the case of setting the parameter of regularization to zero, the objective function becomes the traditional gradient tree boosting [17].

5.5.2 Gradient Boosting Mechanism

Conventional optimization techniques for Euclidean space do not capable of optimization because the tree ensemble model consist of functions as parameters. Alternatively, the model is trained in an additive process. Let $\hat{y}_i^{(t)}$ be the predicted value of the i-th sample at the t-th iteration.

$$\begin{aligned}
&\hat{y_i}^0 = 0 \\
&\hat{y_i}^1 = f_1(x_1) = \hat{y_i}^0 + f_1(x_i) \\
&\hat{y_i}^2 = f_1(x_i) + f_2(x_i) = \hat{y_i}^1 + f_2(x_i) \\
&\quad\vdots \\
&\hat{y_i}^{\mathrm{t}} = \sum_{k=1}^{\mathrm{K}} f_k(x_i) = \hat{y_i}^{(\mathrm{t}-1)} + f_{\mathrm{t}}(x_i)
\end{aligned} \tag{5.67}$$

by adding f_t the following objective will be minimized.

$$\mathcal{L}^{(t)} = \sum_{t=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{5.68}$$

This implies that we are adding the f_t step by step that most improves our ensemble model. By using Taylor expansion of the loss function to the second order, optimization of the function becomes easier in general manner. Hence, other kind of loss functions could be used, and they could be optimized quicker.

$$\mathcal{L}^{(t)} \simeq \sum_{t=1^n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{5.69}$$

Here, g_i and h_i are the first and second order gradient statistics on the loss function and they are defined respectively $\partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $\partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$. The specific objective function at iteration *t* after removing all the constant terms is in the below.

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{5.70}$$

In Eq: 5.70, the objective function becomes only dependent on g_i and h_i, and flexible for different loss functions. By defining the sample set of leaf j as $I_j = \{i|q(x_i) = j\}$, and with the implementation of regularization term, Ω, Eq: 5.70 can be expanded as follows

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2 \tag{5.71}$$

$$= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T \tag{5.72}$$

With a given structure $q(x)$, optimal weight w_j^* of a leaf j can be computed by

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda}, \tag{5.73}$$

and the regarding optimal value is calculated as

$$\tilde{\mathcal{L}}^{(t)} = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T. \tag{5.74}$$

For assessment of a tree structure q quality, Eq: 5.74 can be applied as a measurement function. This measurement could be thought as a impurity ratio for assessing a decision tree, but it has a wider spectrum for objective functions.

By assuming I_L and I_R are the observation sets of right and left leaves resulting from a splitting $I = I_L \cup I_R$. Evaluation of the split candidates and reduction in the loss when a splitting is accrued are calculated by

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma \tag{5.75}$$

In this estimation (Eq: 5.75), if the score is smaller than γ value, than the branching does not take place [16].

5.5.3 Shrinkage Approach and Attribute Subsampling

Two additional techniques are used to further prevent over-fitting. One of them is shrinkage technique. After each iteration of gradient tree boosting, subsequently added new weights are shrunk by the factor η. Shrinkage approach decreases the impact of each individual tree and leaves room for future trees to enhance the model like a learning rate in stochastic optimization. The other approach is attribute subsampling. In each tree, attribute subsets are generated randomly with respect to the subsampling ratio. This method also decreases the execution time of the parallel algorithm [17].

5.5.4 Split Finding Algorithm

Approximate Algorithm

Although, the basic exact greedy algorithm is very powerful since it enumerates over all possible splitting points greedily. However, it is impossible to efficiently do so when the data does not fit entirely into memory. Although the fundamental exact greedy algorithm is very effective as it iteratively enumerates all potential splitting points, efficient implementation of it is almost unfeasible since there is a chance that entire data may not be fitted into memory. An approximate algorithm is required to support appropriate gradient tree boosting in these two conditions.

First, the method recommends candidate dividing points according to feature distribution percentiles. After that, the method then maps the continuous attributes into buckets divided by these candidate points, analyzes the statistics and finds the best alternative among suggestions based on aggregated statistics. Based on when the suggestion is given, there are two variations of the algorithm. During the original stage of tree building, the global version recommends all candidate splits and utilizes the same suggestions for split finding at all stages. After each divide, the local variant proposes again. The global technique needs less propose than the local method. Nevertheless, usually more candidate points are needed for the global proposal because candidates are not refined after each split.

However, for the global proposal, usually more suggestion points are required since after each split, candidates are not refined. After every division, the local suggestion refines the candidate points, and it could be more suitable for deeper trees [17].

Weighted Quantile Sketch

Proposing candidate split points is an important step in the approximate algorithm. Usually a feature's percentiles are used to enable candidates to distribute data evenly.

Let assume that the below set $\mathcal{D}_k = \{(x_{1k}, h_1), (x_{2k}, h_2)...(x_{nk}, h_n)\}$ symbolize the k-th attribute inputs and second order gradient values of each training example. Defining the mapping of the rank functions $r_k : \mathbb{R} \to [0, +\infty)$ as

$$r_k(z) = \frac{1}{\sum_{(x,h)\in\mathcal{D}_k} h} \sum_{(x,h)\in\mathcal{D}_k, x<z} h, \tag{5.76}$$

which identify the ratio of instances whose feature value k is lower than z.

The objective is to locate the candidate for a split point $\{s_{k1}, s_{k2}, ..., s_{kl}\}$, such that

$$|r_k(s_{k,j}) - r_k(s_{k,j+1})| < \epsilon,\ s_{k1} = \text{min} x_{ik}, s_{kl} = \text{max} x_{ik}. \tag{5.77}$$

where ϵ is an approximation factor. Empirically, this implies that there is approximately $\frac{1}{\epsilon}$ candidate points. In redesigned simplified objective function (Eq:5.78), every data points are weighted by h_i

$$\sum_{i=1}^{n} \frac{1}{2} h_i (f_t(x_i) - \frac{g_i}{h_i})^2 + \Omega(f_t) + \text{constant}, \tag{5.78}$$

which is exactly weighted squared loss with labels $\frac{g_i}{h_i}$ and weights h_i. For big datasets, finding candidate splits that meet the requirements is not an easy task. The problem is solved by an existing algorithm called quantile sketch as each instance has the same weights. However, there is no existing quantile sketch for the weighted datasets, but the weighted datasets do not have an current quantile sketch. One way to handle this kind of approximate algorithms problem is heuristics that do not have theoretical guarantee or resorted to sort on a random subset of data which have a chance of failure in most of the current approach [17].

Sparsity-aware Split Finding

In many real-world problem with the data, sparsity of input x is quite common. Presence of high frequent zero inputs, existence of missing values in data and formed binarization of categorical variables artifactuality are some of the reasons of sparsity in dataset. It is important to make the algorithm aware of the sparsity pattern in the data. Making the algorithm conscious against to sparse variables in the data has a importance. To overcame this problem, a default direction in every tree branch point for those attributes is added to the algorithm. An example is classified into a default direction, as the example is a missing value. In each branch point, two direction is exist for an sparse value, and the best direction is learned from data. When there are attribute columns that contain missing values, by taking into consideration just existed samples and treating these variables as a nonexistent the algorithm creates a direction.

Column Block for Paralel Learning

In tree learning the most time consuming process is sorting order the data. By storing the data in memory units, which it is called blocks, cost of storing is decreased. Data is stored in the compressed columns in each block, with each column being sorted by the appropriate feature value. This design of input data only requires to be calculated once before training and can be reused in subsequent iterations. Whole dataset is stored in a single block in the exact greedy process. Than, the split scanning algorithm linearly search over previously ordered inputs. A split search across the block collects the statistics of candidates in whole leaf branches. This search is done by jointly. Also, when using approximate algorithms, the block structure provides benefit. In this circumstances, multi-blocks can be used, with each block corresponding to row subset in the data. The quantile search phase turn out a linear scan across the sorted columns using the ordered framework. This phase has significance particularly for local proposal method which candidates are created at every branch continually. Additionally, with the column block structure column subsampling is promotes because a sub-set of columns in a block can be easily selected [17].

5.6 K-Nearest Neighbor

The K-Nearest Neighbor (KNN) algorithm originated in the early 1950s. The technique is time consuming considering big training sets and did not achieve popularity until the 1960s when enhanced computing power became accessible [28]. This algorithm is memory based method and requires no model to be fit. In other words, KNN does not make any assumption on data to produce a model.

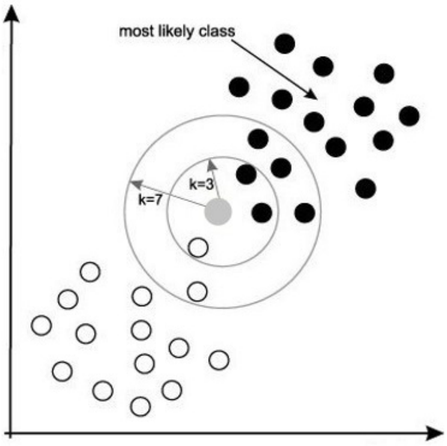


Figure 5.6.1: Explanatory Visual of KNN

In a nutshell, given a testing point x_0 that is in the sample space we can find the closest k point $x_{(r)}, r = 1, ..., k$ with distance assumptions to x_0. After that we can assign this instance to suitable class by usage of majority vote over all the k neighbors [61]. The efficiency of this method is highly dependent on two hyper-parameter. These are a proper similarity function and a appropriate value for parameter k the number of neighbors. When the k is too large number, the majority class members will overwhelm the small class members. And if the k value is too small, than the assessment will be biased [38]. The algorithm can become more or less flexible for different k values. KNN has an advantage that over other algorithms is that the neighbors can give an explanatory classification estimations, especially in where black box methods are insufficient [22]. However, one of the problems in K-NN algorithm is its effectiveness, since it require to compare a test sample set with all samples in the training dataset. That is, it may needs a huge memory while assessment is done. Closeness between two tuples or points is described in terms of a distance metric, such as Euclidean distance. Eq:5.79 can be given for an example to distance metric.

$X_1 = (x_{11}, x_{12}, ..., x_{1n})$ and $X_2 = (x_{21}, x_{22}, ..., x_{2n})$, is

$$dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}. \tag{5.79}$$

Typically, we normalize the values of each attribute before applying to algorithm. Commonly, normalizing or scaling each of data variables before applying to the algorithm avoids the problem of small ranged features are dwarfed by initially larger ranged (such as amount of money) features [28].

5.6.1 Distance Metrics

Distance metrics are used for calculating the distance between initial training samples and newly added test instances. These measurement metrics are generally applied to estimate ot discover the dissimilarity of numerical features of observations.

a) Euclidean Distance

The Euclidean distance is a assessment criterion to calculate distance between two points. It is identified as follows

$$\sqrt{\sum((x - y)^2)}. \tag{5.80}$$

The Euclidean distance is a specific form of the Minkowski metric with degree $p = 2$ [18].

b) Manhattan Distance

Another commonly used criterion is the Manhattan distance. This metric is also known as city block since it is used to calculate the distance between two points in a city [28].

$$\sum(|x-y|) \tag{5.81}$$

c) Minkowski Distance

The Minkowski distance is a way of finding distance based on Euclidean space, defined as

$$(\sum(x-y)^p)^{\frac{1}{p}}. \tag{5.82}$$

d) Chebyshev Distance

Chebyshev Distance is a specific form of the Minkowski distance for $p = \infty$. When degree parameter p is equal to 1, it turns into the Manhattan distance. To compute it, we find the attribute that gives the maximum difference in values between the two objects. By spotting the feature that provide the maximum margin in values between the two example [28]. This metric can be identified as

$$max(|x-y|). \tag{5.83}$$

e) Weighted Minkowski Distance

It is a variant version of the Minkowski distance. The weights are introduced to select feature importance [66]. The formulation is defined as

$$(\sum|w(x-y)|^p)^{\frac{1}{p}}. \tag{5.84}$$

f) Standardized Euclidean Distance

The Standardized Euclidean Distance is used to optimize the problem of finding the distance, and it is defined [18] by

$$\sqrt{\sum(\frac{(x-y)^2}{v})}. \tag{5.85}$$

g) Mahalanobis Distance

The Mahalanobis distance is used to measure the margin between a point and a distribution of samples. The defination is

$$\sqrt{(x-y)'v^{-1}(x-y)}. \tag{5.86}$$

where v is the covariance matrix [18].

5.6.2 Multi-Dimensional Search Algorithms

We consider three different multi-dimensional search algorithms commonly used in KNN analysis. These algorithms are used to find k nearest instances to new test observation.

a) Brute Force

The brute force method calculates the distances between each query point and all reference points using a specified metric,such as Manhattan, Chebyshev, Euclidean etc., for comparison. After this calculations, the nearest neighbors are determined by ordering the distances. Even though the method is very easy and clear, there is a strong computational complexity behind this obvious simplicity. To give an example, if we have a sample set in a *d*-dimensional space with n searching points and m reference points. In this case, $O(mnd)$ time is required for distance calculations and $O(nm \log m)$ for ordering, so the summation work $O(mnd) + O(mn \log m)$ is required for the total execution. In a case that every sample point is considered as both a query point and a reference point, the KNN search require to be implemented on $n = m$ points and subsequently, the execution time consumption becomes $O(n_2 d) + n_2 \log n$. The technique can simply become prohibitive on regular computers for dataset that consist of a big amount of points. Luckily, for every search point, such distance estimations and seeking can be applied separately. Hence, parallelizing the job is one of the practical option for decreasing the execution time consumption [4].

b) kD-Tree

kD-Tree is a binary tree that splits the sample space by using a hyperplane. This approach divides every partition iteratively. Divisions in kD-Tree are produced in the two dimensional space parallel to one of the axes, either vertically or horizontally. The data framework stores a set of samples in k-dimensional space, with the number of features being k, so the method is called k dimensional tree. By estimating the variance of the data samples along every axis separately, an optimum direction for the division is spotted, picking the axis that has the highest variance, and generating a perpendicular separation hyperplane. In order to spot an optimum hyperplane direction, the median value along that axis is located and with this value selection is made for an appropriate point. This assessment allow the separation perpendicular to the maximum spread direction, with half of the samples in the space either hand, and the process is resulted in a well balanced tree [68].

c) Ball Tree

To address the inefficiencies of kD Trees in higher dimensions, the ball tree data structure was developed. Where kD trees partition data along Cartesian axes, ball trees partition data in a series of nesting hyper-spheres. This makes tree construction more costly than that of the kD Tree, but results in a data structure which can be very efficient on highly structured data, even in very high dimensions [48].

In Ball Tree (BT) approach, a ball is defined as the area bounded by a hyper spare that is in the *k* dimensional Euclidean space $\mathcal{R}^n$.

The balls are constructed as floating sample point values of $n + 1$ that indicate the center coordinates and the radius size. A BT is an absolute binary tree in which every node is related with a ball in such a way that the ball of an inner node is the smallest that includes its balls of the children nodes. The inner nodes are only used to direct effective searching through the constructions of the leaf. The leaves only carry the appropriate information for the implementation. Except for node areas in kD-T or oct trees, balltrees sibling areas are permitted to intersect and do not need to partition the whole space. For their implementations and representational strength, these characteristics have a great importance.

A BT iteratively splits the data into nodes described by a $\mathcal{C}$ centroid and r radius, so that every node point is within the r and $\mathcal{C}$ hyper-sphere. By using the below triangle inequality, the amount of target points for a neighbor search is decreased.

$$|x + y| \leq |x| + |y| \tag{5.87}$$

A single margin estimation between a test point and the centroid is appropriate to evaluate a lower and upper limit on the margin to all sample points within the node in this structure. Due to the ball tree nodes' spherical geometry, it can predominate over kD-tree in a high dimensional cases, although the real efficiency is extremely dependent to the training dataset framework [48].

5.7 Multilayer Perceptron Neural Network

To approximate brain structure with neural feed-forward network consisting of discrete neuron layers, each linked to the next one is a prevalent approach since the brain structure is extremely complex. This framework ordinarily involves an input layer, one or more hidden layers, and the output layer. Initiation of the data samples are took place in the input layer. This layer feeds the samples to hidden layer without any change. After that in hidden layer, every one of the node in this layer receive the outputs from the previous layer, make some assumptions, and deliver the results to the output layer. Finally, output layer generates the last outputs that end up with a classification or regression [27].

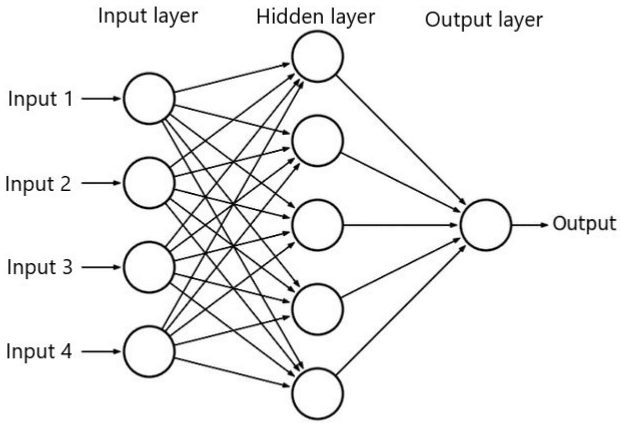


Figure 5.7.1: Schematic for MLP

5.7.1 Feed-Forward Neural Network

Neurons that is not a input neuron has a weight and a bias for associated to every one of its inputs . In order to give more straightforward definition of the structure, the bias vectors are added to vector weights and a bias that is always equal to one is used for every neuron. The outcomes of the neurons' inputs and weights will be summed up. However, it will be produced a gentle approximation of the step function instead of generating the step function given to that item. The weight of the link from the k-th neuron in the $(l-1)$-th layer to the j-th neuron in the j-th layer is defined by w^l_{jk}.

b^l_j is the bias and a^l_j is the activation of the j-th neuron in the l-th layer. This notations are used to make the structure more comprehensible with similar symbols. By using the notations, the a^l_j activations of the j-th neuron in the l-th layer are linked to the $(l-1)$-th layer activations. It is defined as the equation

$$a^l_l = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right), \tag{5.88}$$

Here, this gives the summation of all neurons k in the $(l-1)$-th layer. The weight matrix inputs w^l are the weights that connected to the l-th layer of neurons, and a bias vector will be identified as, b^l, likewise for every layer l.

Lastly, an activation vector and the activation function will be identified as a^l_j and σ

respectively [46]. The weighted input to the neurons in layer l is defined as

$$z^l = w^l a^{l-1} + b^l. \tag{5.89}$$

5.7.2 Backpropagation

The first appearance of the backpropagation algorithm was originally in the 1970s. However, value of this algorithm was not completely realized till a renowned article by David Rumelhart, Geoffrey Hinton, and Ronald Williams. A statement for the partial derivative $\partial C/\partial w$ of the cost function C with respect to any weight w or bias b underlies the center of backpropagation in the network. The expression shows how fast the cost varies when the weights and biases are changed. The objective of backpropagation is to calculate the partial derivatives $\partial C/\partial w$ and $\partial C/\partial b$ of the cost function C with regarding to any weight w or bias b in the network. The quadratic cost function is identified as

$$C = \frac{1}{2n} \sum_x \left| |y(x) - a^L(x)| \right|^2. \tag{5.90}$$

Here, n is the total number of training instances; the summation of over all individual training instances, and $x; y = y(x)$ is the appropriate outcome accordingly. Also, the number of layers in the network is defined as L, and $a^L = a^L(x)$ is the vector of activations outcome from the network for x inputs. The first assumption we need is that the cost function can be written as an average $C_x = \frac{1}{n} \sum_x C_x$ over cost functions C_x for individual training examples, x. For individual training instances x, the cost function can be define as an average $C_x = \frac{1}{n} \sum_x C_x$ over cost functions C_x. The reason why this assumption is needed is because backpropagation effectively make possible the calculations of the partial derivatives $\partial C_x/\partial w$ and $\partial C_x/\partial b$ for a single training instance.

Then, $\partial C/\partial w$ and $\partial C/\partial b$ are can be restored by averaging over training instances.

Let's identify the error on j-th neuron in l-th layer with the notation δ^l_j. Varying a network's weights and bias varies the cost function. A minor variation in Δz^l_j to the neuron's weighted input, in this way rather than outputting $\sigma(z^l_j)$, the neuron instead outputs $\sigma(z^l_j + \Delta z^l_j)$. This variation effects through next layers in the network, at last it leads to a change in the overall cost by an amount $\frac{\partial C}{\partial z^l_j}\Delta z^l_j$.

With weight variation, the optimum value of δz^l_j that reduces the cost value. By selecting Δz_j to have the opposite sign to $\partial C/\partial z^l_j$, cost value can be reduce in case of the cost is a high value. As previously stated, the error δ^l_j of neuron j in layer l is identified by

$$\delta^l_j = \frac{\partial C}{\partial z^l_j}. \tag{5.91}$$

The error in a output layer δ^L is stated as follows:

Corresponding factors of δ^L are

$$\delta^l_j = \frac{\partial C}{\partial a^L_j}\sigma\prime(z^L_j). \tag{5.92}$$

The first expression, $\frac{\partial C}{\partial a^L_j}$, estimates how quickly the cost changes as activation function in the j-th outcome. The other expression on the right, $\sigma\prime(z^L_j)$., estimates how quickly the activation function is varies at z^L_j.

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{5.93}$$

is the error form that is the matrix-based where $\nabla_a C$ is described as a vector with the partial derivatives $\partial C/\partial a^L_j$ components.

Let's define the error value δ^l in the sense of the error in the later layer δ^{l+1} as follows

$$\delta^l = ((w^{l+1})^T\delta^{l+1}) \odot \sigma'(z^l). \tag{5.94}$$

Here, the transpose of the weight matrix w^{j+1} for corresponding $(l+1)$-th layer is $(w^{j+1})^T$. To estimate error backward through the activation function in layer l, the error value on the $(l+1)$-th layer which already known is needed. With the transpose of the (w^{l+1}) and the Hadamard product $\odot\sigma'(z^l)$, this error value can be calculated. The following equation for the variation ratio of the cost value regarding to any bias in the network;

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l. \tag{5.95}$$

In this equation, the error δ_j^l is just equal to the ratio of variation $\frac{\partial C}{\partial b_j^l}$. And finally the following equation for the ratio of variation in the cost value regarding to any weight in the network;

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l. \tag{5.96}$$

With this equation the partial derivatives $\frac{\partial C}{\partial w_{jk}^l}$ can be estimated in the way of the quantities δ_l and a_{l-1}. With beginning from the final layer, the error vectors can be calculated backwardly. For this reason, the approach is called backpropagation [46].

5.7.3 Activation Functions

One of the most essential element in the neural network architecture is their net inputs using a scalar function known as activation function or threshold function, it provide an output a result value. The magnitude of a neuron's output can be restrained by applying an activation function. It limits the magnitude range of the output signal to bounded interval [41].

a) Identity

In this method any calculation or any transformation is not made. This method feed the next layer with just the summed up weighted inputs values. The identity function is defined as

$$f(x) = x. \tag{5.97}$$

b) Logistic

Logistic function is an attractive activation since in this approach computation capacity for training can be reduced. Resulted outcomes form this transformation has a range between 0 and 1. This function is used in many methods. It is also known as sigmoid because of its S-shape [35]. It is defined as

$$g(x) = \frac{1}{1 + e^{-x}}. \tag{5.98}$$

c) Hyperbolic Tangent Function

The structure of Hyperbolic tangent function can be simply defined as the ratio between the hyperpolic cosine and sine functions, but it also be identified as the ratio of the half difference and half sum of two exponential functions in the points x and $-x$. This function can be described as

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{5.99}$$

This function is similar to sigmoid function, but range of its outputs are between -1 and 1 [35].

d) Rectified Linear Unit

Rectified Linear Unit (ReLU) is commonly used activation function in NN algorithms. In the hearth of this function, it feed the next layer with nonnegative value. In other words, this method gives zero value instead of negative values [46]. The output of a rectified linear unit with input x is

$$max(0, x). \tag{5.100}$$

5.7.4 Weight Optimization Methods

Weight optimization is a process that weights of a model are fitted for a given training dataset by minimizing error values of a loss function.

a) The limited memory Broyden-Fletcher-Goldfarb-Shanno

In the limited memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) weight optimization method, the iterates are denoted by x_k, and with $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$ equations are defined to construct a base for iterations. The inverse formula of BFGS is used in this algorithm. That is identified as

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T, \tag{5.101}$$

where $\rho_k = 1/y_k^T s_k$, H_k is inverse Hessian matrix

$$V_k = I - \rho_k y_k s_k^T. \tag{5.102}$$

The LBFGS method works in the following order

At first,by choosing $x_0, m, 0 < \beta^{'} < \frac{1}{2}, \beta^{'} < \beta < 1$, a symmetric and positive definite beginning matrix H_0, by setting k to zero.

Here, H_0 is a notation for a positive definite and sparse symmetric matrix, which approximates to inverse Hessian of objective function.

From this point the iterations takes place as follows

$$d_k = -H_k g_k, \tag{5.103}$$

$$x_{k+1} = x_k + \alpha_k d_k. \tag{5.104}$$

Here, α_k meets the Wolfe conditions;

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \beta^{'} \alpha_k g_k^T d_k, \tag{5.105}$$

$$g(x_k + \alpha_k d_k)^T d_k \geq \beta g_k^T d_k. \tag{5.106}$$

Than, let $\hat{m} = min\{k, m-1\}.$, and adjust $H_0, \hat{m}+1$ times by usage of the tuples $\{y_j, s_j\}_{j=k-\hat{m}}^{k}$.

More specifically,

$$\begin{aligned} H_{k+1} = {} & (V_k^T ... V_{k-\hat{m}}) H_0 (V_{k-\hat{m}} ... V_k) \\ & + \rho_{k-\hat{m}} (V_k^T ... V_{k-\hat{m}+1}^T) s_{k-\hat{m}} s_{k-\hat{m}}^T (V_{k-\hat{m}+1} ... V_k) \\ & + \rho_{k-\hat{m}+1} (V_k^T ... V_{k-\hat{m}+2}^T) s_{k-\hat{m}+1} s_{k-\hat{m}+1}^T (V_{k-\hat{m}+2} ... V_k) \\ & \vdots \\ & + \rho_k s_k s_k^T. \end{aligned} \tag{5.107}$$

And than, setting $k = k+1$ and by repeating this process from beginning of the iterations [42].

b) Stochastic Gradient Descent

For simplicity, let's consider a clear supervised learning framework. Every instance z is a tuple that contains (x, y) comprised of an arbitrary input x and a scalar outcome y. $\ell(\hat{y}, y)$ is a cost function that estimates the loss of predicting $\hat{y}$ with respect to y the actual answer. Also, $f_w(x)$ parameterized functions by a weight vector w, and $\mathcal{F}$ is a family of these functions chosen. The aim is searching a function $f \in \mathcal{F}$ that reduces the loss $Q(z, w) = \ell(f_w(x), y)$ averaged on the instances.

In every iteration, the gradient of $E_n(f_w)$ is calculated on the basis of an only chosen instance at random z_t instead of estimating the gradient exactly. It process can be defined as

$$w_{t+1} = w_t - \gamma_t \nabla_w Q(z_t, w_t). \tag{5.108}$$

The stochastic procedure $\{w_t, t = 1, 2, ...\}$ subject to the instances that chosen at random in every iteration [9].

c) Adam

A method for optimizing stochastic objective functions based on first order gradient, and adaptive calculation of moments of lower order is an underlying base [36]. The technique is simple to execute, computationally effective, and required few memory consumption. In addition, this method is not affected by diagonal rescaling of the gradients, and it is appropriate to handle the difficulties that are large in terms of data or parameters.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{5.109}$$

with this first moment estimation, bias can be corrected. After that, by using

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{5.110}$$

second raw moment bias is estimated. The parameters can be updated by below equation

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+ \in}\hat{m}_t. \tag{5.111}$$

Let's consider a stochastic scalar function that is differentiable with respect to parameter θ as a noisy objective function $f(\theta)$. The aim is minimizing the expected outcome of this function $E[f(\theta)]$ with respect to its parameters θ. The stochastic function is described with $f_1(\theta), ..., f_T(\theta)$ at following timesteps $1, ..., T$. The stochasticity may result from the assessment of random minibatches of instances, or may result from inherent function noise. The gradient that is a vector of partial derivatives of f_t regarding to θ utilized at time step t is defined by

$$g_t = \nabla_\theta f_t(\theta) \tag{5.112}$$

The method adjusts the gradient (m_t) and the squared gradient (v_t). Here, the β_1, β_2 hyper-parameters in range [0,1) regulate the exponential decay ratios of these moving averages.

5.8 Logistic Regression

In LR analysis, we need to justify certain statistical conditions for the validity of the model. However, based on the studies done in LR as a ML method these diagnostic tests are mostly neglected. For the sake of simplicity, in the framework of this thesis the diagnostic checks are not implemented.

While the logistic distribution is used, the value $P(D|X_1, X_2, ..., X_n)$ will define the conditional probability of D given X. In order to get the logic model from the logistic function, z value will be identified as the linear summation $\beta_0 + \beta_1 X_1 + ... + \beta_n X_n$. Here, the β_i are constant parameters that symbolize the unknown parameters, and the X_s are independent attributes of interests. Indeed, to combine the X_s values, z is used as an index.

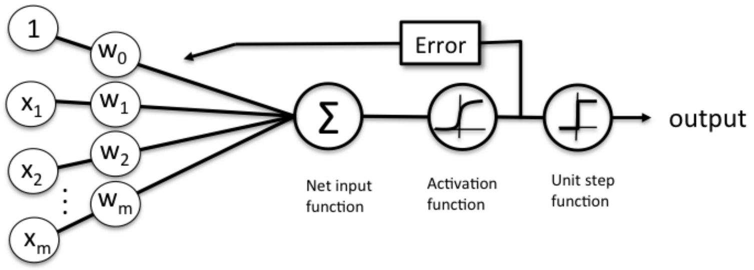


Figure 5.8.1: Schematic for LR.[1]

The specific form of the logistic regression model can be represented as the following

$$\begin{aligned}P(D|X_1, X_2, ..., X_n) &= \frac{e^{\beta_0+\beta_1 X_1+...+\beta_n X_n}}{1+e^{\beta_0+\beta_1 X_1+...+\beta_n X_n}}\\ &= \frac{e^{(\beta_0+\sum_i \beta_i X_i)}}{1+e^{(\beta_0+\sum_i \beta_i X_i)}}\\ &+ \rho_{k-\hat{m}+1}(V_k^T ... V_{k-\hat{m}+2}^T) s_{k-\hat{m}+1} s_{k-\hat{m}+1}^T (V_{k-\hat{m}+2} ... V_k)\\ &= \frac{1}{1+e^{-(\beta_0+\sum_i \beta_i X_i)}}.\end{aligned} \tag{5.113}$$

Central point in this transformation is it is designed to identify a probability that is always a number in the range between 0 and 1 due to the S-shape of the logistic

[1] Figure 5.8.1 source:http://rasbt.github.io/mlxtend/user_guide/classifier/LogisticRegression/

function. Hence, with this design the function can generate a probability value. The logistic model, therefore, is set up to ensure that whatever calculation of risk we obtain, it is a number between 0 and 1.

5.8.1 The Logistic Model

The logit form in the below is required to estimate an odds ratio from logistic model

$$logitP(D|X_1, X_2, ..., X_n) = \ln\left[\frac{P(D|X_1, X_2, ..., X_n)}{1 - P(D|X_1, X_2, ..., X_n)}\right]. \tag{5.114}$$

Simplest meaning of an odds is the ratio of the likelihood that an event will take place over the likelihood that the same event will not take place.

To construct the model, the vector $\beta' = (\alpha, \beta_1, \beta_2, ..., \beta_n)$ estimation is needed. These variables are computed by using the method maximum likelihood estimation.

The likelihood equation results can be identified as

$$\sum_{i=1}^{n}[y_i - P(D|X_i)] = 0 \tag{5.115}$$

and

$$\sum_{i=1}^{n} x_{ij}[y_i - P(D|X_i)] = 0 \tag{5.116}$$

for $j = 1, 2, 3, ..., p$. Let's consider that the solution to these equation will be defined as $\hat{\beta}$. Also, the estimated values for the multiple logistic regression model are denoted as $P(\hat{D}|X)$, the value is estimated with using $\hat{\beta}$ and X_i.

The calculation of the covariance and variance of the computed coefficients appropriately follows a theory of maximum likelihood estimation. The theory indicates that by using the the matrix of second partial derivatives of the logistic likelihood function, the calculations are procured. The general form of the partial derivatives in the above have the form

$$\frac{\partial^2 L(\beta)}{\partial \beta_j^2} = -\sum_{i=1}^{n} x_{ij}^2 P_i(1 - P_i) \tag{5.117}$$

and

$$\frac{\partial^2 L(\beta)}{\partial\beta_j \partial\beta_l} = -\sum_{i=1}^{n} x_{ij} x_{il} P_i (1 - P_i) \tag{5.118}$$

for $j, l = 0, 1, 2, ..., p$ where P_i denotes $P(x_i)$.

Considering the $(p+1) \times (p+1)$ matrix including the nonpositive of the terms specified in equation above be identified as $I(\beta)$. The matrix maintained in the above is known as the observed information matrix. By using the inverse of this matrix which defined as $Var(\beta) =^{-1} (\beta)$, the variance and covariance values of the calculated coefficients can be produced. It is not likely to define an explicit expression for the values in this matrix except in very particular situations. Therefore, the notation $Var(\beta_j)$ will be used to identify the j th diagonal element of this matrix, which is the variance of $\hat{\beta}_j$.And also, $Cov(\beta_j, \beta_l)$ identifies an arbitrary off-diagonal element, which is the covariance of $\hat{\beta}_j$ and $\hat{\beta}_l$.

The calculations of the variances and covariances, which will be identified as $\widehat{Var}(\hat{\beta})$, are computed by assessing $Var(\beta)$ at β. We use the calculated standard errors of the computed coefficients, which will be defined as

$$\widehat{SE}(\hat{\beta}_j) = [\widehat{Var}(\hat{\beta}_j)]^{1/2} \tag{5.119}$$

for $j = 0, 1, 2, ..., p$. A formulation structure of the information matrix helpful as examining the model adjustment and evaluation of adjusting is $\hat{I}(\hat{\beta}) = X'\hat{V}X$.

Here, X is an $n \times p+1$ matrix includes the data for every subject and V is an $n \times n$ diagonal matrix with general element $\hat{P}_i(1 - \hat{P}_i)$ [29].

5.8.2 Regularization

a) Ridge Regression

Except that the coefficients are computed by reducing a minor different amount while comparing the Ridge Regression (RR) and least squares. Apart form this, these two calculation method is very similar. Specifically, the ridge regression coefficient computes $\hat{\beta}$ are the values that reduce

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}\beta_j^2. \tag{5.120}$$

Here, $\lambda \geq 0$ is used to regulate the relative effect of RSS and $\lambda\sum_{j=1}^{p}\beta_j^2$, shrinkage penalty, terms on the regression coefficient computations [32].

b) Lasso Regression

Comparing the Lasso Regression with the Ridge regression, RR have an apparent drawback. Instead of the best subset, RR will contains all p attributes within the final form of model. Without setting the λ parameter to zero, the penalty expression shrink each one of the coefficients through to zero, but this will not lead any of them to reach exactly to zero. In model interpretation, this can generate a difficulty where the amount of variable is quite wide. The lasso is an alternative to ridge regression that obviates this disadvantage. The lasso coefficients, $\hat{\beta}_\lambda^L$, reduce the value

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = RSS + \lambda\sum_{j=1}^{p}|\beta_j|. \tag{5.121}$$

The only distinction is that the penalty expression β_j^2 in the ridge regression is replaced by $|\beta_j|$ in the lasso penalty. Like with ridge regression, the lasso shrinks the calculations of the coefficient through to zero, but in the case of lasso, if λ is large enough, the l_1 penalty will force some of the coefficient calculations to be precisely equivalent to zero.

Therefore, the lasso works feature selection method, similar to the optimum subset selection of attributes. For this reason, the models constructed form the lasso are commonly quite simpler compered to those by ridge regression in terms of interpretation [32]. (An introduction to statistical learning)

5.8.3 Optimization Techniques

a) Limited Memory BFGS Method

In this part, a summary will be given to explanation of LBFGS method since this technique is already identified under the MLP section.

The LBFGS method is redesigned version of BFGS for handling large scale optimization problems. In such a problem, $O(n_2)$ is the storing cost and updating cost B_t would be prohibitively expensive. In LBFGS, first m steps are identical to the BFGS approach. Inverse of the Hessian is calculated only by using the m steps that in parameter and gradient space.

The form of the inverse BFGS formula in this approach is

$$H_{k+1} = V_k^T H_k V_k + p_k s_k s_k^T, \tag{5.122}$$

where $p_k = \frac{1}{y_k^T s_k}$, and $V_k = 1 - p_k y_k s_k^T$.

The direction of the quasi newton is produced from by usage of a matrix free method [55].

The iterations in optimization are identified as $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$, with respect to x_k. By adjusting $H_0, \hat{m} + 1$ times with usage of the tuples $\{y_j, s_j\}_{j=k-\hat{m}}^k$, the optimization is performed [42].

b) Newton-Conjugate Gradient

The fundamental concept in Conjugate Gradient (CG) is to choose a seeking direction that is perpendicular to the seeking direction from the prior iteration. Let's assume that u is an arbitrary direction. w can be adjusted as

$$w^{'} \leftarrow w + \frac{g^T u}{\lambda u^T u + \sum_n \sigma(w^T x_n)\sigma(-w^T x_n)(u^T x_n)^2} u \tag{5.123}$$

Here, σ is the logistic function, $\sigma(a) = (1 + exp - 1)^{-1}$, g is the gradient, w is an arbitrary weight, and x_n's are independent data instances. The gradient can be estimated as

$$g = -\lambda w + \sum_{n} \sigma(-y_n w^T x_n) y_n x_n. \tag{5.124}$$

By using the Hestenes-Stiefel formula

$$\beta = \frac{g^{'T}(g^{'} - g)}{u^T(g^{'} - g)} \tag{5.125}$$

the direction u can be determined according to $u^{'} \leftarrow g - \beta u$, where an optimum value of β [20].

c) LIBLINEAR

By considering a set of sample and class label tuple is $(x_i, y_i), i = 1, ..., l, x_i \in R_n, y_i \in \{-1, +1\}$. The Liblinear method deal with the following unconstrained optimization case with variety of loss functions $\xi(w; x_i, y_i)$:

$$\min_{w} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi(w; x_i, y_i), \tag{5.126}$$

where $C > 0$ is a penalty parameter. $log(1 + e^{-y_i w^T x_i})$ is the loss function that is derived from a probabilistic model. This optimization method solve bias term, b, by augmenting the vector w and each sample x_i with an additional dimension $w^T \leftarrow [w^T, b], x_i^T \leftarrow [x_i^T, B]$, where B is a constant.

d) Stochastic Average Gradient

The Stochastic Average Gradient (SAG) technique is a randomized setting of the incremental aggregated gradient approach. SAG has the lesser iteration cost comparing to SG techniques, but reaches the converge rates for the Full Gradient algorithm.

The iteration of SAG takes the following form

$$x^{k+1} = x^k - \frac{\alpha_k}{n} \sum_{i=1}^{n} y_i^k. \tag{5.127}$$

Here, a random index i_k is chosen for every iteration. Also, we set

$$y_i^k = \begin{cases} f_i^{'}(x^k) & \text{if } i = i_k, \\ y_i^{k-1} & \text{if } otherwise. \end{cases}$$

This is the step includes a gradient with respect to every function. However, each iteration only calculates the gradient for a single instance, and iteration cost is independent of n. By specifying a constant value for step-size the SAG iterations have a linear convergence ratio for strongly-convex objectives, and a $O(1 = k)$ convergence ratio for convex objectives [54].

e) SAGA

SAGA method is based on theory behind SAG and Stochastic Variance Reduced Gradient algorithms. The algorithm reduces strongly convex finite sums, and it is faster in expectation than is likely without the construction of the finite sum. In the general situation that a quadratic regularization is used, the requirement of strong convexity is also met in machine learning issues. Specifically,in this approach, minimizing functions as the following form are aimed.

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x), \tag{5.128}$$

Here, $x \in \mathbb{R}^d$, every f_i has Lipschitz continuous derivatives with scalar L and, it is convex. The optimization is started with some vector $x^0 \in \mathbb{R}^d$ that is initially known, and $f_i^{'}(\phi_i^0) \in \mathbb{R}^d$ known derivatives with $\phi_i^0 = x^0$ for every i. Above derivatives are kept in a table data-framework of length n. By using a step size of γ and executing below adjustments, and starting with $k = 0$, the SAGA algorithm begins with given x^k value and each $f_i^{'}(\phi_i^k)$ quantity at the end of iteration k, the adjustments for iteration $k + 1$ is as follows

First, select a j value uniformly at random.

After that, obtain $\phi_j^{k+1} = x^k$, and keep $f_j^{'}(\phi_j^{k+1})$ in the table. All other inputs in the table remain untouched. The value ϕ_j^{k+1} is not explicitly kept.

Than, adjust x by using $f_i^{'}(\phi_i^{k+1})$, $f_i^{'}(\phi_i^k)$ and the table average

$$w^{k+1} = x^k - \gamma \left[f_j^{'}(\phi_j^{k+1}) - f_j^{'}(\phi_j^k) + \frac{1}{n} \sum_{i=1}^{n} f_i^{'}(\phi_i^k) \right], \tag{5.129}$$

$$x^{k+1} = prox_\gamma^h(w^{k+1}). \tag{5.130}$$

The description of the proximal operator applied above is

$$prox_\gamma^h(y) := \underset{x \in \mathbb{R}^d}{argmin} \left\{ h(x) + \frac{1}{2\gamma} ||x-y||^2 \right\} \tag{5.131}$$

where $h(x)$ is the regularization function [21].

CHAPTER 6

VALIDATION AND EMPIRICAL RESULTS

Validation criterias measure for assessing how good or how accurate a classifier is at predicting the class label of instances.

Using training data to derive a classifier and then estimating the accuracy of the resulting learned model can result in misleading overoptimistic estimates due to overspecialization of the learning algorithm to the data. Instead, it is better to measure the classifier's accuracy on a test set consisting of class-labeled tuples that were not used to train the model.

a) Accuracy Ratio

Accuracy ratio can be described in detail with four additional terms that is needed to know to create confusion matrix used in evaluation of classification. There are two dimensions in determining the accuracy. These are true positive (TP) which refers to the positive instances that are correctly assigned by the classifier, true negative (TN) which measures the negative observations that are correctly classified by the algorithm, false positive (FP) which shows the negative samples that are incorrectly assigned as positive, and false negative (FN) gives the ratio of the positive instances that are misclassified as negative.

These ratios are summarized in the confusion matrix (Table 6.0.1).

Table 6.0.1: Confusion Matrix.

		Predicted Class		
		1	0	Total
Actual Class	1	True Positive	False Positive	P
	0	False Negative	True Negative	N
	Total	P’	N’	P+N

The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier. That is,

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.1}$$

In the pattern recognition literature, this is also referred to as the overall recognition rate of the classifier, that is, it reflects how well the classifier recognizes tuples of the various classes [28].

b) Type I and Type II Error

Type I and Type II error rates measure the error that occur for each class. It calculates for class I by total number of falsely classified class I members number divided by the total number of class I members. These ratios are needed when it is an imbalance problem, where the main class of interest is rare. That is, the data set distribution reflects a significant majority of the positive class and a minority negative class, in our study credit assessment default instances are always lesser than the non-default ones.

$$\text{Type I Error} = \frac{FN}{FN + TN} \tag{6.2}$$

$$\text{Type II Error} = \frac{FP}{FP + TP} \tag{6.3}$$

c) Area Under the Receiver Operating Characteristics

Receiver operating characteristic (ROC) curve is a useful visual tool for comparing two classification models. An area under the receiver operating characteristics (AUC) curves comes from signal detection theory, and this curves are used for the analysis of radar images. ROC curve for a given model shows the trade-off between the true positive rate (TPR) and the false positive rate (FPR). Given a test set and a model, TPR is the proportion of positive (or "yes") tuples that are correctly labeled by the model; FPR is the proportion of negative (or "no") tuples that are mislabeled as positive. Given that TP, FP, P, and N are the number of true positive, false positive, positive, and negative tuples, respectively. A visual example of ROC curve for different threshold values is given in Figure 6.0.1. For a two-class problem, an ROC curve allows us to visualize the trade-off between the rate at which the model can accurately recognize positive cases versus the rate at which it mistakenly identifies negative cases as positive for different portions of the test set. Any increase in TPR occurs at the cost of an increase in FPR. The area under the ROC curve is a measure of the accuracy of the model [28].

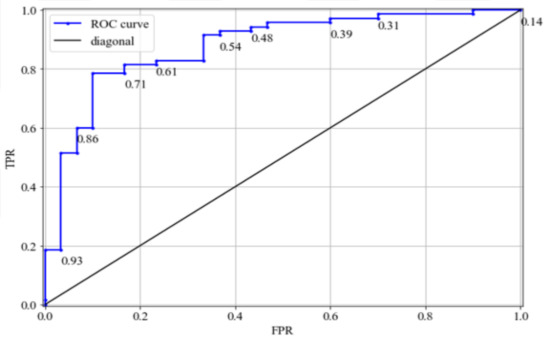


Figure 6.0.1: An example to ROC Curve.

6.1 K-Fold Cross Validation

In K-fold cross-validation, the present samples are split at random into k dissociated parts or subsets $D_1, D_2, ..., D_k$, every one of them have roughly equal number of sample. In this validation process, testing and training is executed k times. In step i, splitted part D_i is kept as a test set, and the rest of the data samples are used to train the ML model.

More specifically, the sample subsets $D_2, D_3, ..., D_k$ collectively perform as the training set to produce the first model in the first step, which is validated on D_1; the second step is trained on sample sets $D_1, D_3, D_4, ..., D_k$ and validated on D_2. Every instance is used just one time for validation, $k-1$ times in the training process. This validation technique is performed for classification as depending on the criteria estimation is total number of correctly classified instances from the k iterations, split by the total number of pairs in the recent sample set [28]. In the case of leave on out approach, just one instance is kept out at a time for the validation set. The structure is simply identified in the Figure 6.1.1.

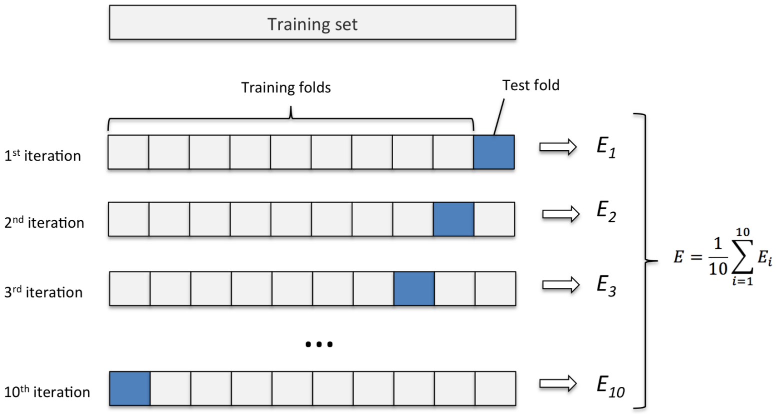


Figure 6.1.1: K-Fold Cross Validation.

6.2 Experimental Design

Experiments are conducted on two credit datasets taken from UCI machine learning repository. The details of the datasets can be seen in Chapter 2. In the experiments, the two datasets are randomly divided into two sets: 90% as the training set and 10% as the test set. However, the sample distribution in German data is 30% and 70%, and when the data is divided randomly, insufficiency in terms of default or non-default sample may be occurred on train or test datasets. To prevent this, the instances are selected so that the proportion of features from each classes remains the same in both the training set and the test set. To do this, the data is first divided into two data subsets, one of them contains the default instances and the other includes non-defaulted examples only. Then, these two data subsets are split separately at random, and than, the train and test datasets are obtained by combining corresponding subsets.

In our experiments, it is aimed to show the classification ability of eight ML algo-

[1] Source of Figure 6.1.1: http://karlrosaen.com/ml/learning-log/2016-06-20/

rithms, and the impact of four data transformation techniques. Beside, we employ three approaches in our analysis. The first one is application of GS for parameter optimization, the second one is implementation of Wrapper method for feature selection, and the last one is jointly usage of WFS and GS. 10-fold cross is applied to ensure avoidance of overfitting in these approaches. Each of the experiments are validated on three randomly divided train and test subsets. Average of AUC, accuracy, type I and type II error values of these three cases is used to compare effectiveness of algorithms and methods.

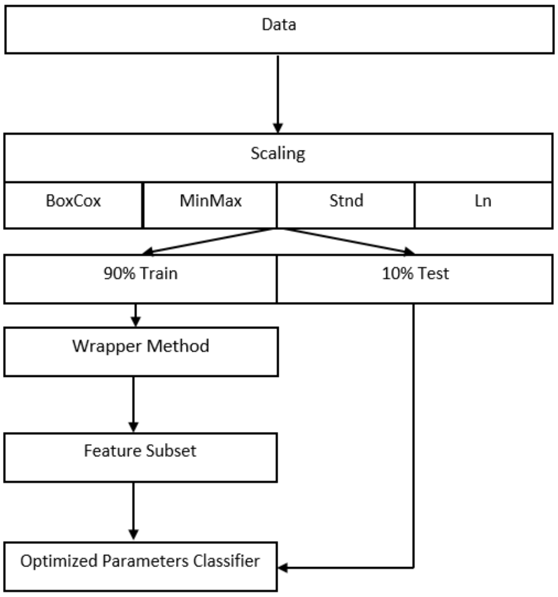


Figure 6.2.1: Experimental Desing with Wrapper Method.

Second design which is only application of grid search to the datasets is similarly executed just on training set. It reveals that which parameters of the machine learning algorithms are the optimum for the data. Same as the wrapper method part, to avoid overfitting 10 fold cross validation is used in grid search process. After determination of the parameters, this values are used to evaluate models performance on test set. This process is conducted for three times and their average is taken. One can find visual description on Figure 6.2.2.

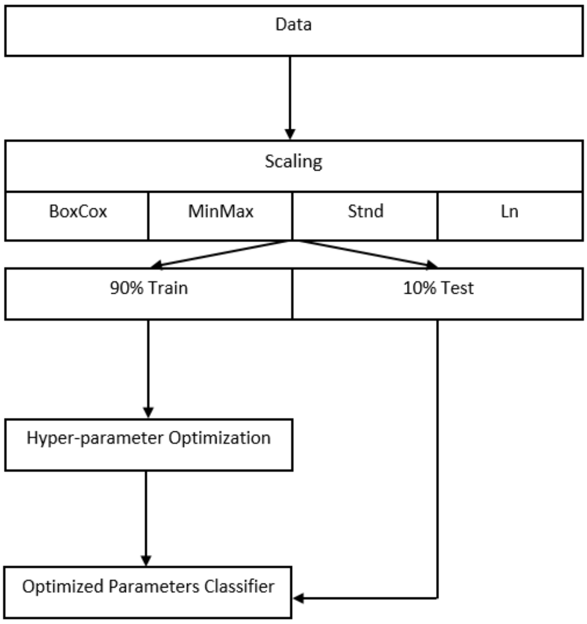


Figure 6.2.2: Experimental Desing with Grid Search.

In the last experiment, after the most successful attributes are selected by applying WFS, GS is used for parameter tuning on training dataset consists of the selected features. In Figure 6.2.3, visualization of this approach is given. This design is also applied three times and the results averages are taken.

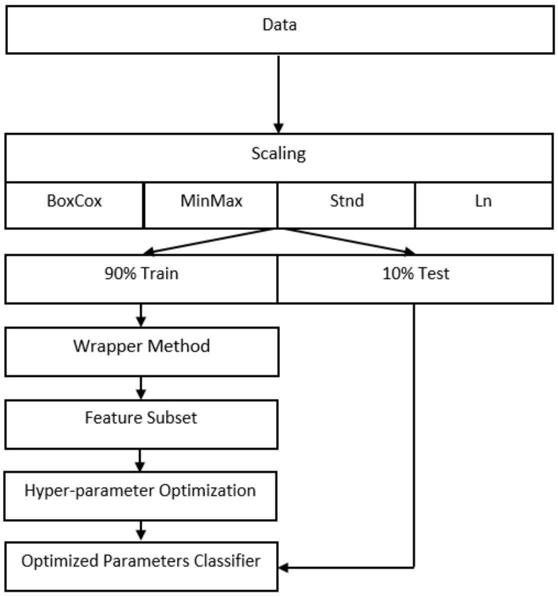


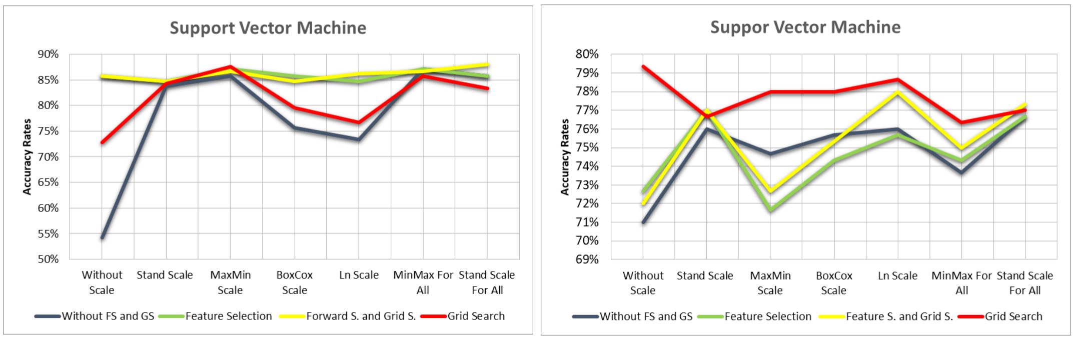
Figure 6.2.3: Experimental Desing with Wrapper Method and Grid Search.

Sklearn-learn libraries [49] are used to conduct the experiments for eight machine learning algorithms and the validation algorithms.

6.3 Application to German and Australian Datasets

In this section, we will compare the performance of eight different machine learning algorithms with three approaches on their default states and dataset having no feature selection. In addition, the impact of four different transformation methods to the results are analyzed. Six different datasets are created. These are four datasets created by using four transformation techniques to continuous attributes only and two datasets that Min-Max and Standard scaling are also applied to the categorical features. For the outcomes for feature selection, we give attribute numbers to ensure simple expression. Each of the numbers correspond to an attribute, and there are two tables in the below indicating which numbers the attributes are associated with. In the experiments on each data set 3 different attribute sets are obtained. To get general view of results, the attributes that are occurred at least two time in these three subsets are in the final feature subset. We share two different lists that first (common attribute list) aim to give an overview of the feature selection results on 7 different data formed by transformation techniques, and the second (best case list) is for the case that reaches the highest accuracy ratio. Features included in the best-case list are those contained in at least 4 of the final lists generated in seven cases. We share the selected features on Australian data in Appendix since the attributes are changed symbols.

a) SVM



(a) On Australian Data

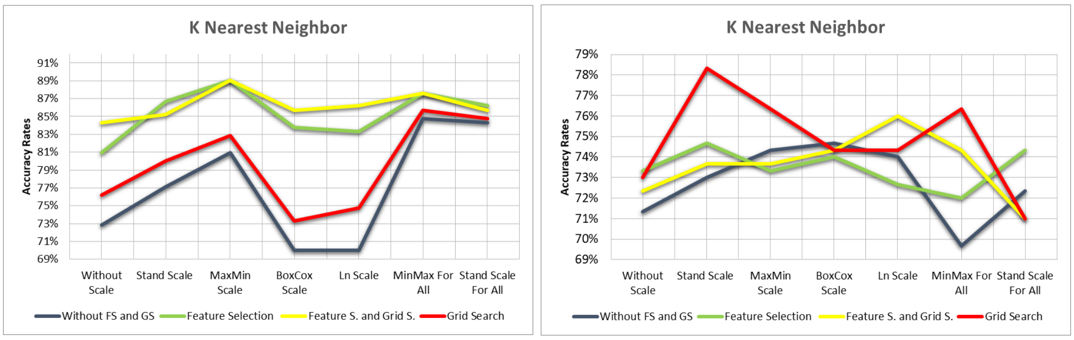
(b) On German Data

Figure 6.3.1: SVM Results

In the analysis made with SVM, it is seen in Figure 6.3.1a that WFS and GS approaches on Australian data together and only using WFS generally give more stable results than other approaches. Additionally, it is seen that with GS, and dataset generated by the Standard scaling which is applied to all attributes has the highest accuracy score 88.09% in the analysis. Also, outcomes obtained with data that Min-Max scale is applied to only continuous attributes have the highest average ratio than with the other techniques. The worst outcome is obtained with the data that no transformation is applied (54.29%), but on average, the lowest case equal to 80.24% is occurred on the data that generated with LN transformation. [3,5,7,8] is the Common Attribute List (CAL) and [0,1,2,3,5,6,7,8,9,12] is the Best Case List (BCL) with 87.14% accuracy.

In the outcomes on German data in Figure 6.3.1b, the performance of GS and SVM combination reaches the highest degree. It is understood that the algorithm does not give stable results when WFS and together WFS and GS are applied. Among the data transformation techniques, the data set which transformation is not implemented gives the highest score 79.33% with parameter optimization. However, it's execution time takes 05:40:49 hour although the others have average computation time is 00:08:34 minutes. Also, the lowest score 73.75% is obtained in this data set. On average, the most successful scale technique for this data is LN (77.08%) while the worst score is attained by the data with no transformation. Hence, it is seen that the transformation methods have a significant effect on the algorithm efficiency. On the other hand, [0,1,2,3,4,5,9,13,14,15,16,17,18] is CAL and [0,1,2,3,4,5,7,8,9,10,11,13,14,15,16,17,18,21,23] is BCL with 77% accuracy ratio. It can be said that each of the lists share similar attributes in general.

b) KNN



(a) On Australian Data

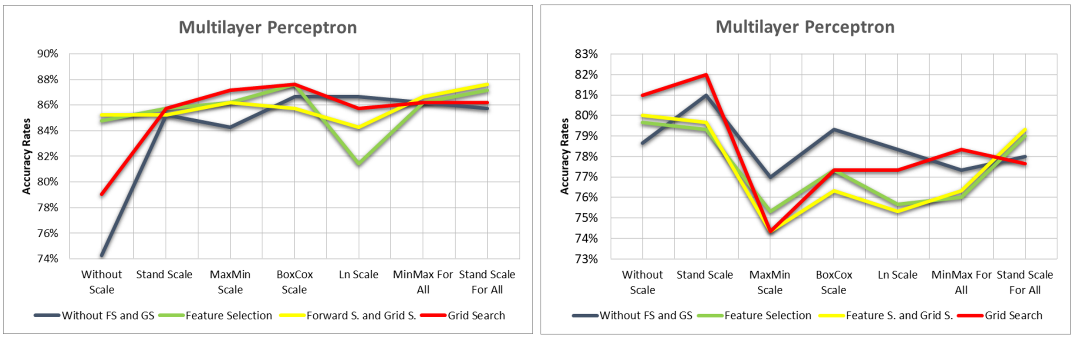
(b) On German Data

Figure 6.3.2: KNN Results

In the analysis with KNN, it is seen in Figure 6.3.2a that with WFS approach application to Australian data generally give stable and better results, and GS has slightly better outcomes than the algorithm with default parameters. Hence, KNN is more effective when WFS method is applied. The data set that generated by using Min-Max scale for only continuous features gives the highest score 89.05% while the ML algorithm has the lowest accuracy 70.00% with Box-Cox and LN transformation. In general, the case obtained by applying Min-Max transformation to all attributes has the better result on average (86.43%), but with Box-Cox it has the worst outcome 78.21%. On the feature selection side, [3,7,8,9,11,13] are the features that are in the CAL and [2,7,8,9,11,12,13] is the BCL with the accuracy rate 89.05%. This rate also is the highest point among the other ML algorithms. Most of the features are common to the two lists.

The studies with German credit dataset in Figure 6.3.2b show that KNN achieved the highest performance 78.33% with GS. The other two approaches have a positive effect on the accuracy ratio of KNN. The ML algorithm reaches the highest score with the dataset that Standard scale is implied to only continuous feature, and the lowest score 69.67% is obtained with data set Min-Max scale is applied for all features. On average, the most suitable transformation is Standard scale (74.92%) whereas the worst technique is Standard scale that is implemented to all features (72.22%). In the general sense, it is clear that the effectiveness of KNN on this data depends on transformation techniques. For this dataset, [0,1,2,3,4,5,14,16,17,18,19,21] is the CAL and [0,1,2,4,5,13,14,16,17,18,19,20,21] is the BCL with 74.67% accuracy rate.

c) MLP



(a) On Australian Data

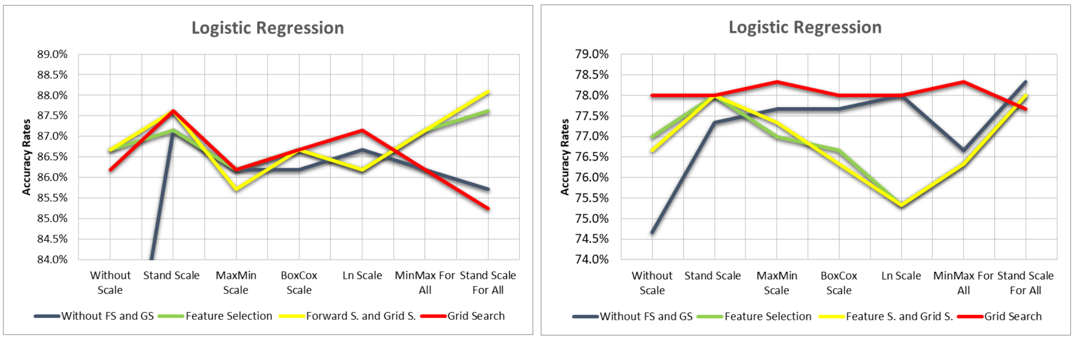
(b) On German Data

Figure 6.3.3: MLP Results

In the Figure 6.3.3a, it is seen that all approaches give results that are close to each other. WFS with GS is the most consistent approach than the others, but GS is slightly better in general. This algorithm reaches the highest accuracy ratio 87.62% with Box-Cox data transformation and the case that Standard scale is applied to all features, but the data that no transformation is implemented gives the worst outcome 74.29% with MLP. On average, Box-Cox is the most suitable techniques for this data whereas LN is the transformation that gives the lowest outcome 84.52%. [0,1,2,3,4,5,7,8,10,11,12,13] attributes are selected in the CAL, and [0,1,2,3,4,5,7,8, 10,11] are determined in the BCL with accuracy ratio 87.62%.

The results obtained on the German data in Figure 6.3.3b give unstable results compared to Australian data case. It is observed that the performance of MLP reaches the highest level in the case that GS is applied for parameter optimization. In terms of data transformation, the highest result 82.00% is achieved with the data set that Standard scale is applied to only continuous attributes. However, the worst result 74.33% is obtained with Min-Max scale is used for only continuous features. On the other hand, considering the average result MLP have better performance with Standard scale (80.50%) while this algorithm has the lowest average result 75.25% with Min-Max scale is applied data set. As a result, it can be said that performance of MLP is influenced significantly as the transformation techniques change. In the FS side, [0,1,2,3,4,5,7,8,10,14,15,16,17,19,20] are selected as a most efficient features in CAL, and [0,1,2,3,4,5,7,8,9,10,14,15,16,17,18,19,20,21] attributes are for the BCL with 79.33 % accuracy.

d) LR



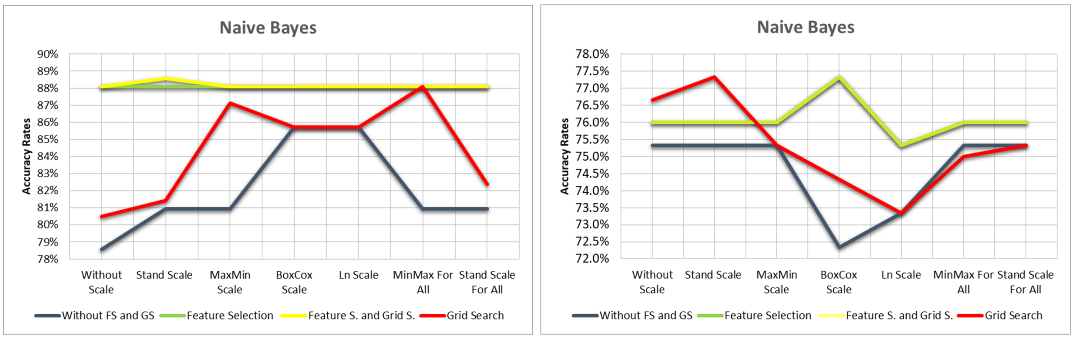
(a) On Australian Data (b) On German Data

Figure 6.3.4: LR Results

Studies conducted with Australian data in Figure 6.3.4a generally show robust results. The performances of the three different methods give close results, but GS has a better impact on LR in general. The most successful result 88.10% is obtained by the approach that WFS and GS is used together. The worst result 78.10% attained in the case where no method and data scaling is applied. In terms of data transformation techniques, the algorithm is reached the highest result 88.10% with the data that Standard scale is applied to all attributes while the worst result is obtained with data that no scale technique is applied. Also, the best average accuracy ratio is 87.38% obtained with dataset that Standard scale is implemented for only continuous features, but without transformation LR gives the lowest average results 84.40%. In this experiment chosen features for CAL is [0,1,2,3,4,5,7,8,9,11], and for BCL is [3,4,5,7,9,13] with accuracy rate 87.62 %.

In Figure 6.3.4b, it is seen that the variation of the results is high. It appears that jointly usage of GS and LR has more robust results than the other methods, and in general its average efficiency is pretty high. Although, it is seen that LR is less compatible with LN scaling among the transformation techniques, with this technique LR gives better results comparing to case that no scaling is applied. This algorithm has the best score 78.33% with Min-Max and Standard scale while it gives the lowest result in the case that no transformation is applied. Additionally, the highest average result 78.00% is obtained with the case that Standard scaling to all attributes, and again the worst result 76.58% is occurred in the dataset that no transformation is applied. Hence, it is clear that the efficiency of the algorithm depends on the transformation techniques. For the FS side, [0,1,2,3,4,6,7,8,10,11,14,15,16,17,18,19,21] is CAL, and [0,1,2,4,6,7,8,10,11,14,15,16,18,22] is BCL with 78 %.

e) GNB



(a) On Australian Data

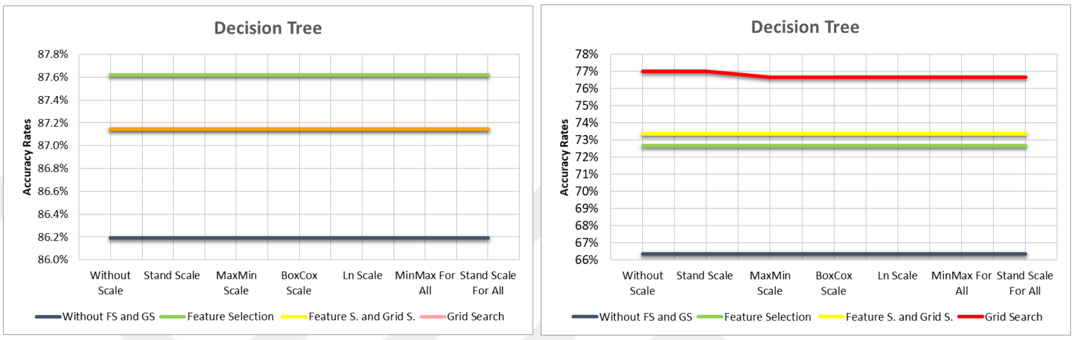
(b) On German Data

Figure 6.3.5: GNB Results

In the analysis with GNB, it is seen in Figure 6.3.5a that the approaches using WFS and WFS with GS together on Australian data give better and more consistent results than other situations, with GS its performance increases but the variation of accuracy ratios is high. It can be said that GNB is more compatible with the feature selection that is using GNB itself. In the case where GS is used only, performance of the algorithm is better than the situation where no method is applied. And, it is also seen that after implementing FS methods to data, transformation techniques have no effect on performance of GNB. Apart from these, the best accuracy 88.57% is obtained with data that Standard scale is applied to only continuous variables, and the lowest result 78.57% is occurred in the data which no transformation is implemented. Beside, average (86.91%) of results occurred on the cases Box-Cox and LN transformation is used is the highest among the other techniques while the worst average accuracy 83.81% is obtained in the dataset which no data scaling is applied. Determined attributes on this data set for CAL and BCL is [0,1,3,4,5,7,8,10,11,12] with 88.57% accuracy.

The results in Figure 6.3.5b show that WFS approach is more efficient compared to the situation where only GS is applied. All methods have a positive impact on GNB except the case where only GS is applied on the data Min-Max transformation is used for all features. When the data scaling techniques are considered, the highest result 77.33% achieved with Standard and Box-Cox is used to scale continuous attributes, and the lowest outcome 73.33% is occurred where LN is applied. Among the average results, the worst case (74.33%) is obtained where LN is used, and the best result 76.17% is gotten in the dataset that Standard scale is implemented to only continuous features. The CAL on this data is [0,1,2,4,5,6,7,8,11,12,13,15,20,22,23], and the BCL is [0,1,2,4,5,6,8,15] with 77.33 %.

f) DT



(a) On Australian Data

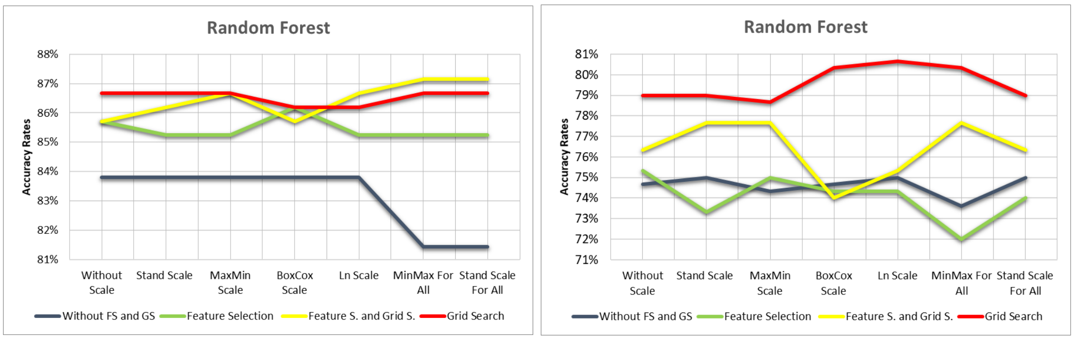
(b) On German Data

Figure 6.3.6: DT Results

The Figure in 6.3.6a shows that the performance of the decision tree reaches the highest point with the WFS method. Also, it is understood that all methods increase the efficiency of this algorithm. The accuracy results of GS and WFS with GS overlap. Therefore, they are seen as orange. Apart from this, it is understood that the performance of the algorithm is not affected by data transformation techniques [71] [52] [61], as stated in the literature. The approach that only GS is applied is less effective than WFS. The selected features in CAL and BCL is [0, 3, 7, 8, 11] with 87.62 % accuracy ratio.

On the other hand, in the studies conducted on German data in Figure 6.3.6b, it is clear that GS gives better results than other approaches. The efficiency of DT on the original data with default parameters appears to be very low in this data set as in Australian data. It is once again seen that DT is not affected by the transformation techniques. With this data features [0,2] are included in the CAL, and [0, 2, 14, 18, 23] are contained in the BCL with the accuracy rate 72.67 %.

g) RF



(a) On Australian Data

(b) On German Data

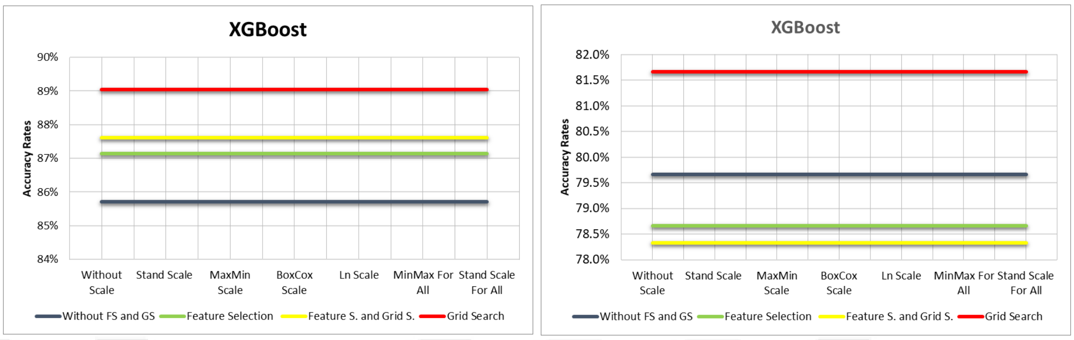
Figure 6.3.7: RF Results

In Figure 6.3.7a, it can be said that the efficiency of all methods are close to each other, but the case that WFS is applied gives slightly lower accuracy rate than the other methods in general. The best result 87.14% is occurred where WFS is used with GS, but the most consistent method is GS. In terms of data transformation, RF gives the highest result 87.14% with datasets that Standard and Min-Max scale are implemented to all attributes. However, the worst cases (81.43%) are occurred with the same transformation techniques where no method is applied. The best average accuracy rate 85.60% is achieved on dataset that Min-Max is applied only to continuous columns while the worst results 85.12% are obtained on the cases where the Min-Max and Standard scale is used for all attributes. In the FS side, [0,1,2,3,4,5,7,8,9,10,11,12,13] are the elements for both the CAL and BCL with accuracy rate 87.14 %.

The results seen in Figure 6.3.7b shows that the GS approach has more positive effect than the others. Outcomes of the case that only feature selection method is applied with this algorithm is poor since it even gives worse results than the one that no method is applied. RF algorithm has the highest result 80.67% with LN data transformation, but the worst outcome 72.00% is occurred in case that Min-Max scale is applied to all features. In general, the highest average accuracy value 76.42% is obtained with Min-Max scale whereas the lowest value 75.83% is obtained in case that Box-Cox transformation is used. Hence,the performance of the algorithm is dependent to transformation techniques, but it is not consistent. In other word, there is not an ideal transformation for this algorithm. The CAL is

[0,1,2,3,4,5,7,12,14,15,16,17,18,19,20,21,22,23] for this experiment and the BCL is [0,1,2,3,4,5,6,10,14,15,16,17,18,19,21,22,23] with 77.67 % accuracy.

h) XGB



(a) On Australian Data (b) On German Data

Figure 6.3.8: XGB Results

Finally, it can be said that the results in Figure 6.3.8a of the studies conducted with XGB on the Australian data with GS are more compatible than the other cases. In addition, WFS has a positive effect on performance when compared to the original data. Jointly usage of GS and WFS contributes a little to previous results. As mentioned in the literature, data transformation techniques do not have any effect on this algorithm. The elements in both the CAL and the BCL are same, [0,3,4,6,7,8,9,10,11,12] with accuracy rate 87.62%, since data transformation techniques do not have any effect on this ML algorithm.

In parallel with the results on other data in Figure 6.3.8b, compatibility of GS with this algorithm is much higher than other approaches on German data. However, the results of the analyzes performed with feature selection show that the performance of the algorithm is less efficient than the default case. In this analysis, when XGB algorithm is applied with wrapper method which uses XGB itself, it can be seen that this approach is inefficient on this dataset. Likewise, the results show that this method is not affected by data transformation once again. The common elements for the CAL and BCL are [0,1,2,3,4,5,6,7,12,13,14,15,16,17,18,21,22,23] with the accuracy 78.67%.

6.3.1 Additional Experiment

In this part, we give the outcomes of different experimental approach. The transformation techniques are applied in normal approaches, but we also want to show if the transformations are applied after the splitting, how the performances of ML algorithm are affected. To investigate this affect the best results MLP with GS on German dataset form that Standard scale is used for only continuous features, and KNN with WFS on Australian dataset version that Min-Max transformation is applied only to continuous attributes are taken into account. Although The experiment with XGB gives the same accuracy with the KNN, we use this experiment KNN since transformation techniques does not affect XGB. Same as the previous designs the test set ratio is 10% , and 90% left for train set. Transformation process is applied after the split. For Min-Max scaling approach, the highest and lowest values are used to scale the attribute columns, and Standard scale technique uses the sample mean and standard deviation. For that reason, the values in a specific column affect the transformation process. We also assume that all the test values are not seen in training process, so it also applies to scaling.

In the experiment with MLP, it is found that the accuracy ratio decrease form 82% to 81.33%. Rise on the type II error (it is increased form 8.09% to 9.04%) is the cause of performance decrease, but the type I error rates remain the same. However, accuracy rate drops approximately 3.80% to 85.24% from 89.05%. Compering to MLP case it has a dramatic influence on the performance of KNN. And, the selected features are also different from ones in the previous design. In this analysis, the instances E, H, J and L are prominent, while in the previous case, H, I, J, L, M and N features are selected.

6.3.2 Comparison of Machine Learning Algorithms Selected Methods

Table 6.3.1: The best performing three algorithms under with WFS

Performance indicators of top 3 algorithms (Australian Dataset)

Model	Accuracy (%)	AUC (%)	Type I Error (%)	Type II Error (%)	No of Feat.	Without FS (%)
KNN	89.05	89.18	11.96	9.67	6.66	80.95
GNB	88.10	88.76	17.09	5.37	9.33	80.95
LR	87.62	88.00	15.38	8.60	9.66	86.67

Performance indicators of top 3 algorithms (German Dataset)

Model	Accuracy (%)	AUC (%)	Type I Error (%)	Type II Error (%)	No of Feat.	Without FS (%)
MLP	79.67	71.51	48.88	8.09	18	78.67
XGB	78.67	70.16	51.11	8.57	15.66	79.67
LR	78.00	69.68	51.11	9.52	15.33	77.33

Table 6.3.1 presents only the three highest results of the performance of WFS method on all ML algorithms. These results are compared with accuracy, AUC, type I error and type II error rates. When the table is examined, it is seen that KNN has achieved a much better result than other algorithms only with an average of 6.66 attributes. The low dimensionality is a major advantage in terms of both the interpretation of the data and the execution time. GNB's efficiency is significantly higher than the default situation. Although WFS has a relatively small effect on MLP, it ranks third with a slightly higher result than others.

When we look at the top three performances on German data, MLP is at the top. It is seen feature selection contribute 1 % to the efficiency of this algorithm comparing to original data case. This performance is achieved with an average of 18 features, while the original data has 24 characteristics. It is seen that WFS has a negative effect on the efficiency of the XGB which comes after this ML algorithm. LR has the third highest accuracy with this result, although feature selection contributes slightly to its performance.

Table 6.3.2: The best performing three algorithms under with WFS+GS

Performance indicators of top 3 algorithms (Australian Dataset)

Model	Accuracy (%)	AUC (%)	Type I Error (%)	Type II Error (%)	No of Feat.	Without GS and FS (%)
KNN	89.05	89.18	11.96	9.67	6.66	80.95
GNB	88.55	89.19	16.23	5.37	9.33	80.95
LR	88.10	88.54	15.38	7.53	8.66	85.71

Performance indicators of top 3 algorithms (German Dataset)

Model	Accuracy (%)	AUC (%)	Type I Error (%)	Type II Error (%)	No of Feat.	Without GS and FS (%)
MLP	80.00	71.50	47.78	8.09	18	78.67
XGB	78.33	69.29	53.33	8.09	15.67	79.67
LR	78.00	69.68	51.11	9.52	15.33	77.33

When the highest three scores obtained from the case that WFS and GS are applied together, it shows that KNN has the highest accuracy ratio on Australian data similar to WFS is summarized in Table 6.3.2. With this method, the accuracy of KNN has increased from 80.95% to 89.05% at a significant level. Compared to the previous table, the enhancement of this approach to performance of GNB is seen in the second row. The efficiency of LR in the third row is increased significantly with approach, and it gives stronger results than MLP.

As a result of the analysis conducted with the other data set, it is seen that the performance of MLP is partially increased, efficiency of XGB is decreased, and this approach does not make any contribution to LR's accuracy ratio. Nevertheless, the XGB algorithm ranks second.

Table 6.3.3: The best performing three algorithms under with GS

Performance indicators of top 3 algorithms (Australian Dataset)

Model	Accuracy (%)	AUC (%)	Type I Error (%)	Type II Error (%)	Without GS (%)
XGB	89.05	89.18	11.96	9.67	85.71
GNB	88.10	88.76	17.09	5.37	80.95
MLP	87.62	87.89	14.52	9.67	86.67

Performance indicators of top 3 algorithms (German Dataset)

Model	Accuracy (%)	AUC (%)	Type I Error (%)	Type II Error (%)	Without GS (%)
MLP	82.00	75.71	40.00	8.57	81.00
XGB	81.67	73.89	45.56	6.67	79.67
RF	80.67	71.90	50.00	6.19	75.00

In the approach where only GS is applied, it is understood in Table 6.3.3 that XGB has reached the highest efficiency and its accuracy ratio is increased significantly comparing to the case that default parameters are applied. The next highest accuracy rate is achieved with NB. With this approach, the efficiency of this algorithm increases significantly compared to XGB. The highest results of MLP are occurred with this method.

On the other hand, MLP reaches the highest performance with this method considering the other approaches on German credit data. The next highest performance is belong to XGB, and it can be said that the efficiency of this algorithm also reach its highest level with this approach. Although LR significantly improves its performance with GS, it gives the third best result.

Table 6.3.4: Comparison of the results to previous studies

Accuracy, Type I and Type II Error Rates

	German Dataset				Australian Dataset			
Article name	**Method**	**Accuracy (%)**	**Type I (%)**	**Type II (%)**	**Method**	**Accuracy (%)**	**Type I (%)**	**Type II (%)**
- Our Study	MLP&G.S	82.00	40.00	8.57	XGB & G.S.	89.05	11.96	9.67
- Xia Y. et al. (2017)	Bagging NN	76.01	49.67	12.98	XGB	87.81	13.92	10.80
- D. Liang er al. (2015)	SVM	76.30	49.80	-	NB+PSO***	85.86	12.41	-
- A. I. Marques er al. (2012)	MLP+Adaboost	71.50	45.00	-	Adaboost+NB	79.57	3.00	-
- C. F. Tsai (2009)	MLP+FA*	78.76	48.69	10.66	MLP+PCA**	89.93	11.53	7.9

*FA=Factor Analysis **PCA=Principal Component Analysis ***PSO=Partial Swarm Optimization

Table 6.3.4 represents the results of the studies in literature at which type I and type II ratios are available from over twenty recently published studies. Results that the highest accuracy and least error rates in these studies are situated in the table. When we look at these studies, it can be said that the case which MLP ML algorithm is applied together with GS to German data has better results than the other studies in the table in terms of accuracy rate, type I and type II error ratios.

It is seen that the results obtained on XGB with GS on Australian credit data are more efficient than the studies in literature except that the case MLP is used with PCA. The type I error in the study using Adaboost with GNB appears to be very low, but it can be said that the type II error is very high because the accuracy rate is much lower than the other studies.

6.3.3 Conclusion

In this thesis, we investigate which of the eight classification methods is superior to others in terms of consumer credit classification. We also examined the effects of GS and WFS methods on the performance of these classification algorithms, and observed how four different data conversion techniques influence the effectiveness of these algorithms. WFS is applied to each algorithm with usage of SFS method. Analysis is carried out with German and Australian credit dataset taken from UCI ML repository. To make these measurements 90 percent of the data is kept for training and 10 percent for testing. Data transformations are implemented before splitting. In order to compare the analysis results, datasets are divided according to certain random states. For each case, three different train and test sets are generated according to random states, and average of these three results are considered for comparison. AUC, accuracy, type I and type II error rates are used for validation.

In our analysis, we find that performance of SVM is significantly improved with GS on both datasets. Although the WFS method leads to consistent and good results on Australian data, the same does not apply to German data. While KNN is more compatible with WFS on Australian data, usage of GS leads to better results in other data. MLP gives higher accuracy rates with GS on these datasets. WFS has a positive impact to this algorithm on German data, but the same is not observed in other data. Studies with LR shows that this algorithm gives better results with GS on both data whereas WFS is more suitable for Australian data than the other one. GNB and GS combination gives good classification performance, but jointly usage of GNB and WFS provides higher and more stable outcomes. Application of WFS to Australian data, and GS to German data is more appropriate in terms of DT's classification ability. Although there is a feature selection mechanism in DT called embedded method, WFS has a favorable impact. Three different approaches have a contribute significantly to the effectiveness of RF algorithm, but it is seen that while only GS is applied, better results are achieved compared to the others. While analyzing with XGB, WFS has a positive impact on Australian data and a negative effect on the other data. On the other hand, it is found that GS is the best method on both datasets.

In terms of data transformations, it is seen that impact of these techniques vary according to data set and ML algorithms. For SVM, the most efficient technique is Standard scale on Australian data. On German data, the case that no transformation technique is applied gives the highest accuracy rate, but although average of the computational time of the other situations is 8 minutes and 34 seconds, this case takes 5 hours, 40 minutes and 49 seconds. Therefore, LN transformation is more appropriate for SVM on this dataset. Min-Max is more effective for Australian data, and Standard scale for German data where KNN is employed. It is found that Standard scale is the best option on both dataset for MLP. However, the worst results are obtained with LN on Australian data, and Min-Max on the other data. Two datasets generated by implementing Min-Max and Standard scale give the highest scores with LR, while Standard scale is more suitable, but LN is incompatible technique for Australian data. For GNB, the results indicate that Min-Max leads to more accurate score on Australian data, while Standard scale is not as efficient as the other techniques. On German data, the highest outcomes are achieved with Box-Cox and Standard techniques, and the worst result is obtained with LN transformation. Although data transformation techniques do not affect RF, DT and XGB tree based algorithms [71] [52] [61], it is clearly seen in the result section that these techniques have an impact on RF algorithm.

Regarding WFS, it is observed that some of the attributes are commonly selected by all ML algorithms, which are contained in CAL and BCL sets at least five times. The features that are contained at least five of the CAL sets are Checking Account, Duration in month, Credit History, Credit Amount, Savings Account, Employment Since, Present Resident, Foreign Worker, Purpose-New Car, Purpose-Used Car, Purpose-Domestic Appliances, Co-Applicant. On the other hand, Checking Account, Duration in month, Credit History, Savings Account, Employment Since, Foreign Worker, Purpose-New Car, Purpose-Used Car, Purpose-Domestic Appliances, Co-Applicant, Skilled Employee-Official are included in more than four of the BCL sets. The selected features that occur on Australian data can be found within Table A.3.1 in Appendix.

As a result of this research, we find that there is no perfect algorithm, method or transformation for all datasets. We also see that the performance of ML algorithms depends on the transformation techniques, data itself and the methods. In general, GS is the most suitable method for all algorithms, while WFS is more appropriate for KNN and GNB because of the increase in their accuracy ratio comparing to their default cases that no method is applied. When all the outcomes are considered, it can be said that WFS gives consistent results as similar features are selected in all of the FS cases.

REFERENCES

[1] R. Anderson, *The credit scoring toolkit: theory and practice for retail credit risk management and decision automation*, Oxford University Press, 2007.

[2] I. Antonov, Crafting a market landscape: Quantitative vs. judgmental credit risk-rating systems, Journal of Lending & Credit Risk Management, 82(5), pp. 34–34, 2000.

[3] C. Apté and S. Weiss, Data mining with decision trees and decision rules, Future generation computer systems, 13(2-3), pp. 197–210, 1997.

[4] A. S. Arefin, C. Riveros, R. Berretta, and P. Moscato, Gpu-fs-knn: A software tool for fast and scalable knn computation using gpus, PloS one, 7(8), p. e44000, 2012.

[5] B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens, and J. Vanthienen, Benchmarking state-of-the-art classification algorithms for credit scoring, Journal of the operational research society, 54(6), pp. 627–635, 2003.

[6] M. Bailey, *Consumer credit quality: underwriting, scoring, fraud prevention and collections*, White Box Publishing, 2004.

[7] E. Blasch, S. Ravela, and A. Aved, *Handbook of dynamic data driven applications systems*, Springer, 2018.

[8] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, Feature selection for high-dimensional data, Progress in Artificial Intelligence, 5(2), pp. 65–75, 2016.

[9] L. Bottou, Large-scale machine learning with stochastic gradient descent, in *Proceedings of COMPSTAT'2010*, pp. 177–186, Springer, 2010.

[10] L. Breiman, Random forests, Machine Learning, 45(1), pp. 5–32, 2001.

[11] L. Breiman, *Classification and regression trees*, Routledge, 2017.

[12] L. Breiman and A. Cutler, *Random Forests Classification Description*, 2015 (Accessed 3 July 2019), https://www.stat.berkeley.edu/~breiman/RandomForests/.

[13] I. Brown and C. Mues, An experimental comparison of classification algorithms for imbalanced credit scoring data sets, Expert Systems with Applications, 39(3), pp. 3446–3453, 2012.

[14] G. G. Chandler and J. Y. Coffman, A comparative analysis of empirical vs. judgmental credit evaluation, Financial Review, 14(4), pp. 23–23, 1979.

[15] F.-L. Chen and F.-C. Li, Combination of feature selection approaches with svm in credit scoring, Expert systems with applications, 37(7), pp. 4902–4909, 2010.

[16] T. Chen, Introduction to boosted trees, University of Washington Computer Science, 22, p. 115, 2014.

[17] T. Chen and C. Guestrin, Xgboost: A scalable tree boosting system, in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, ACM, 2016.

[18] K. Chomboon, P. Chujai, P. Teerarassamee, K. Kerdprasop, and N. Kerdprasop, An empirical study of distance metrics for k-nearest neighbor algorithm, in *Proceedings of the 3rd International Conference on Industrial Application Engineering*, pp. 1–6, 2015.

[19] J. N. Crook, *Credit scoring: an overview; paper prepared for the British Association, Festival of Sciences, University of Birmingham, 11-13 September 1996*, University of Edinburgh, Department of Business Studies, 1996.

[20] H. Daumé III, Notes on cg and lm-bfgs optimization of logistic regression, Paper available at http://pub. hal3. name# daume04cg-bfgs, implementation available at http://hal3. name/megam, 198, p. 282, 2004.

[21] A. Defazio, F. Bach, and S. Lacoste-Julien, Saga: A fast incremental gradient method with support for non-strongly convex composite objectives, in *Advances in neural information processing systems*, pp. 1646–1654, 2014.

[22] S. Dreiseitl and L. Ohno-Machado, Logistic regression and artificial neural network classification models: a methodology review, Journal of biomedical informatics, 35(5-6), pp. 352–359, 2002.

[23] D. Dua and C. Graff, UCI machine learning repository, 2017 (Accessed 21 June 2019), http://archive.ics.uci.edu/ml.

[24] D. Durand, *Risk elements in consumer installment financing*, National Bureau of Economic Research, New York, 1941.

[25] R. A. Fisher, The use of multiple measurements in taxonomic problems, Annals of eugenics, 7(2), pp. 179–188, 1936.

[26] N. Friedman, D. Geiger, and M. Goldszmidt, Bayesian network classifiers, Machine learning, 29(2-3), pp. 131–163, 1997.

[27] J. Grus, *Data science from scratch: first principles with python*, O'Reilly Media, 2019.

[28] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*, Elsevier, 2011.

[29] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*, volume 398, John Wiley & Sons, 2013.

[30] J. Hull, *Risk management and financial institutions,+ Web Site*, volume 733, John Wiley & Sons, 2012.

[31] H. Ince and B. Aktan, A comparison of data mining techniques for credit scoring in banking: A managerial perspective, Journal of Business Economics and Management, 10(3), pp. 233–240, 2009.

[32] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, volume 112, Springer, 2013.

[33] T. Jayalakshmi, Dr. a. santhakumaran (2011), Statistical Normalization and Back Propagation for Classification.

[34] R. W. Johnson, Legal, social and economic issues in implementing scoring in the us, Credit Scoring and Credit Control, 19, p. 32, 1992.

[35] B. Karlik and A. V. Olgac, Performance analysis of various activation functions in generalized mlp architectures of neural networks, International Journal of Artificial Intelligence and Expert Systems, 1(4), pp. 111–122, 2011.

[36] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980, 2014.

[37] S. Lessmann, B. Baesens, H.-V. Seow, and L. C. Thomas, Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research, European Journal of Operational Research, 247(1), pp. 124–136, 2015.

[38] B. Li, S. Yu, and Q. Lu, An improved k-nearest neighbor algorithm for text categorization, arXiv preprint cs/0306099, 2003.

[39] X.-L. Li and Y. Zhong, An overview of personal credit scoring: techniques and future work, International Journal of Intelligence Science, 2(04), p. 181, 2012.

[40] D. Liang, C.-F. Tsai, and H.-T. Wu, The effect of feature selection on financial distress prediction, Knowledge-Based Systems, 73, pp. 289–297, 2015.

[41] R. P. Lippmann, Anintroduction to computing with neural nets,", IEEE Assp magazine, 4(2), pp. 4–22, 1987.

[42] D. C. Liu and J. Nocedal, On the limited memory bfgs method for large scale optimization, Mathematical programming, 45(1-3), pp. 503–528, 1989.

[43] A. Marqués, V. García, and J. S. Sánchez, Exploring the behaviour of base classifiers in credit scoring ensembles, Expert Systems with Applications, 39(11), pp. 10244–10250, 2012.

[44] J. P. Mueller and L. Massaron, *Machine learning for dummies*, John Wiley & Sons, 2016.

[45] J. H. Myers and E. W. Forgy, The development of numerical credit evaluation systems, Journal of the American Statistical association, 58(303), pp. 799–806, 1963.

[46] M. A. Nielsen, *Neural networks and deep learning*, volume 25, Determination press San Francisco, CA, USA:, 2015.

[47] N. Nnamoko, F. Arshad, D. England, J. Vora, and J. Norman, Evaluation of filter and wrapper methods for feature selection in supervised machine learning, Age, 21(81), pp. 33–2, 2014.

[48] S. M. Omohundro, *Five balltree construction algorithms*, International Computer Science Institute Berkeley, 1989.

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

[50] J. Platt et al., Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, Advances in large margin classifiers, 10(3), pp. 61–74, 1999.

[51] S. Raschka, *Python machine learning*, Packt Publishing Ltd, 2015.

[52] S. Raschka and V. Mirjalili, *Python machine learning*, Packt Publishing Ltd, 2017.

[53] I. Rish et al., An empirical study of the naive bayes classifier, in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pp. 41–46, 2001.

[54] M. Schmidt, N. Le Roux, and F. Bach, Minimizing finite sums with the stochastic average gradient, Mathematical Programming, 162(1-2), pp. 83–112, 2017.

[55] N. N. Schraudolph, J. Yu, and S. Günter, A stochastic quasi-newton method for online convex optimization, in *Artificial intelligence and statistics*, pp. 436–443, 2007.

[56] N. Siddiqi, *Credit risk scorecards: developing and implementing intelligent credit scoring*, volume 3, John Wiley & Sons, 2012.

[57] U. Stańczyk and L. C. Jain, *Feature selection for data and pattern recognition*, Springer, 2015.

[58] A. Sullivan, Consumer finance, EI Altman, Financial Handbook (9.3-9.27), New York: John Wiley & Sons, 1981.

[59] J. Suto, S. Oniga, and P. P. Sitar, Comparison of wrapper and filter feature selection algorithms on human activity recognition, in *2016 6th International Conference on Computers Communications and Control (ICCCC)*, pp. 124–129, IEEE, 2016.

[60] B. K. Szymanski and Y. Zhang, Recursive data mining for masquerade detection and author identification, in *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop*, pp. 424–431, IEEE, 2004.

[61] H. Trevor, T. Robert, and F. JH, *The elements of statistical learning: data mining, inference, and prediction*, Springer, 2009.

[62] T. Van Gestel and B. Baesens, *Credit Risk Management: Basic concepts: Financial risk components, Rating analysis, models, economic and regulatory capital*, OUP Oxford, 2008.

[63] H. Van Sang, N. H. Nam, and N. D. Nhan, A novel credit scoring prediction model based on feature selection approach and parallel random forest, Indian Journal of Science and Technology, 9(20), pp. 1–6, 2016.

[64] J. VanderPlas, *Python data science handbook: essential tools for working with data*, " O'Reilly Media, Inc.", 2016.

[65] A. Verikas, E. Vaiciukynas, A. Gelzinis, J. Parker, and M. Olsson, Electromyographic patterns during golf swing: Activation sequence profiling and prediction of shot effectiveness, Sensors, 16(4), p. 592, 2016.

[66] E. Vezzetti and F. Marcolin, *Similarity measures for face recognition*, Bentham Science Publishers, 2015.

[67] R. Vidhya, D. Vijayasekaran, and S. Ramakrishnan, Mapping invasive plant prosopis juliflora in arid land using high resolution remote sensing data and biophysical parameters, 2017.

[68] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufmann, 2016.

[69] I.-C. Yeh and C.-h. Lien, The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients, Expert Systems with Applications, 36(2), pp. 2473–2480, 2009.

[70] H. Yu, X. Huang, X. Hu, and H. Cai, A comparative study on data mining algorithms for individual credit risk evaluation, in *2010 International Conference on Management of e-Commerce and e-Government*, pp. 35–38, IEEE, 2010.

[71] A. Zheng and A. Casari, *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*, " O'Reilly Media, Inc.", 2018.

APPENDIX A

DETAILED PERFORMANCE OUTCOMES OF ML ALGORITHMS

In the following sections, detailed outcomes of ML algorithms and methods are presented. Additionally, FS outputs belonging to Australian dataset, their corresponding codes and tables of execution time are provided.

A.1 Detailed Performance Outcomes

SVM

Table A.1.1: Detailed Results of SVM on German Data

Support Vector Machine	Without FS and GS	Feature Selection	Feature S. and Grid S.	Grid Search	No of Feat.
Without Scale	71.00%	72.67%	72.00%	79.33%	9.7
Standard	76.00%	77.00%	77.00%	76.67%	16.7
Min-Max	74.67%	71.67%	72.67%	78.00%	11.7
Box-Cox	75.67%	74.33%	75.33%	78.00%	21.3
LN	76.00%	75.67%	78.00%	78.67%	16.7
Min-Max (All At.)	73.67%	74.33%	75.00%	76.33%	12.0
Standard (All At.)	76.67%	76.67%	77.33%	77.00%	13.7

Table A.1.2: Detailed Results of SVM on Australian Data

Support Vector Machine	Without FS and GS	Feature Selection	Forward S. and Grid S.	Grid Search	No of Feat.
Without Scale	54.29%	85.71%	85.71%	72.86%	5.7
Standard	83.81%	84.76%	84.76%	84.29%	7.7
Min-Max	85.71%	87.14%	86.67%	87.62%	8.0
Box-Cox	75.71%	85.71%	84.76%	79.52%	6.3
LN	73.33%	84.76%	86.19%	76.67%	7.0
Min-Max (All At.)	87.14%	87.14%	86.67%	85.71%	1.0
Standard (All At.)	85.71%	85.71%	88.10%	83.33%	6.0

KNN

Table A.1.3: Detailed Results of KNN on German Data

K-Nearest Neighbors	Without FS and GS	Feature Selection	Feature S. and Grid S.	Grid Search	No of Feat.
Without Scale	71.33%	73.33%	72.33%	73.00%	16.7
Standard	73.00%	74.67%	73.67%	78.33%	11.3
Min-Max	74.33%	73.33%	73.67%	76.33%	15.0
Box-Cox	74.67%	74.00%	74.33%	74.33%	17.7
LN	74.00%	72.67%	76.00%	74.33%	13.3
Min-Max (All At.)	69.67%	72.00%	74.33%	76.33%	11.0
Standard (All At.)	72.33%	74.33%	71.00%	71.00%	14.7

Table A.1.4: Detailed Results of KNN on Australian Data

K-Nearest Neighbors	Without FS and GS	Feature Selection	Feature S. and Grid S.	Grid Search	No of Feat.
Without Scale	72.86%	80.95%	84.29%	76.19%	4.0
Standard	77.14%	86.67%	85.24%	80.00%	7.0
Min-Max	80.95%	89.05%	89.05%	82.86%	6.7
Box-Cox	70.00%	83.81%	85.71%	73.33%	6.3
LN	70.00%	83.33%	86.19%	74.76%	6.7
Min-Max (All At.)	84.76%	87.62%	87.62%	85.71%	6.7
Standard (All At.)	84.29%	86.19%	85.71%	84.76%	7.7

MLP

Table A.1.5: Detailed Results of MLP on German Data

Multilayer NN Perceptron	Without FS and GS	Feature Selection	Feature S. and Grid S.	Grid Search	No of Feat.
Without Scale	78.67%	79.67%	80.00%	81.00%	18.0
Standard	81.00%	79.33%	79.67%	82.00%	17.3
Min-Max	77.00%	75.33%	74.33%	74.33%	11.7
Box-Cox	79.33%	77.33%	76.33%	77.33%	19.3
LN	78.33%	75.67%	75.33%	77.33%	12.3
Min-Max (All At.)	77.33%	76.00%	76.33%	78.33%	14.0
Standard (All At.)	78.00%	79.00%	79.33%	77.67%	13.7

Table A.1.6: Detailed Results of MLP on Australian Data

Multilayer NN Perceptron	Without FS and GS	Feature Selection	Feature S. and Grid S.	Grid Search	No of Feat.
Without Scale	78.67%	79.67%	80.00%	81.00%	18.0
Standard	81.00%	79.33%	79.67%	82.00%	17.3
Min-Max	77.00%	75.33%	74.33%	74.33%	11.7
Box-Cox	79.33%	77.33%	76.33%	77.33%	19.3
LN	78.33%	75.67%	75.33%	77.33%	12.3
Min-Max (All At.)	77.33%	76.00%	76.33%	78.33%	14.0
Standard (All At.)	78.00%	79.00%	79.33%	77.67%	13.7

LR

Table A.1.7: Detailed Results of LR on German Data

Logistic Regression	Without FS and GS	Feature Selection	Feature S. and Grid S.	Grid Search	No of Feat.
Without Scale	74.67%	77.00%	76.67%	78.00%	15.7
Standard	77.33%	78.00%	78.00%	78.00%	15.3
Min-Max	77.67%	77.00%	77.33%	78.33%	17.3
Box-Cox	77.67%	76.67%	76.33%	78.00%	17.0
LN	78.00%	75.33%	75.33%	78.00%	13.7
Min-Max (All At.)	76.67%	76.33%	76.33%	78.33%	17.3
Standard (All At.)	78.33%	78.00%	78.00%	77.67%	15.3

Table A.1.8: Detailed Results of LR on Australian Data

Logistic Regression	Without FS and GS	Feature Selection	Forward S. and Grid S.	Grid Search	No of Feat.
Without Scale	78.10%	86.67%	86.67%	86.19%	9.0
Standard	87.14%	87.14%	87.62%	87.62%	10.7
Min-Max	86.19%	86.19%	85.71%	86.19%	12.0
Box-Cox	86.19%	86.67%	86.67%	86.67%	9.0
LN	86.67%	86.19%	86.19%	87.14%	11.0
Min-Max (All At.)	86.19%	87.14%	87.14%	86.19%	8.3
Standard (All At.)	85.71%	87.62%	88.10%	85.24%	8.7

GNB

Table A.1.9: Detailed Results of GNB on German Data

Naïve Bayes	Without FS and GS	Feature Selection	Feature S. and Grid S.	Grid Search	No of Feat.
Without Scale	75.33%	76.00%	76.00%	76.67%	9.7
Standard	75.33%	76.00%	76.00%	77.33%	9.7
Min-Max	75.33%	76.00%	76.00%	75.33%	9.7
Box-Cox	72.33%	77.33%	77.33%	74.33%	8.7
LN	73.33%	75.33%	75.33%	73.33%	8.3
Min-Max (All At.)	75.33%	76.00%	76.00%	75.00%	9.7
Standard (All At.)	75.33%	76.00%	76.00%	75.33%	9.7

Table A.1.10: Detailed Results of GNB on Australian Data

Naïve Bayes	Without FS and GS	Feature Selection	Feature S. and Grid S.	Grid Search	No of Feat.
Without Scale	78.57%	88.10%	88.10%	80.48%	9.3
Standard	80.95%	88.10%	88.57%	81.43%	9.3
Min-Max	80.95%	88.10%	88.10%	87.14%	9.3
Box-Cox	85.71%	88.10%	88.10%	85.71%	9.3
LN	85.71%	88.10%	88.10%	85.71%	10.7
Min-Max (All At.)	80.95%	88.10%	88.10%	88.10%	9.3
Standard (All At.)	80.95%	88.10%	88.10%	82.38%	9.3

DT

Table A.1.11: Detailed Results of DT on German Data

Decision Tree	Without FS and GS	Feature Selection	Feature S. and Grid S.	Grid Search	No of Feat.
Without Scale	66.33%	72.67%	73.33%	77.00%	4.3
Standard	66.33%	72.67%	73.33%	77.00%	4.3
Min-Max	66.33%	72.67%	73.33%	76.67%	4.3
Box-Cox	66.33%	72.67%	73.33%	76.67%	4.3
LN	66.33%	72.67%	73.33%	76.67%	4.3
Min-Max (All At.)	66.33%	72.67%	73.33%	76.67%	4.3
Standard (All At.)	66.33%	72.67%	73.33%	76.67%	4.3

Table A.1.12: Detailed Results of DT on Australian Data

Decision Tree	Without FS and GS	Feature Selection	Feature S. and Grid S.	Grid Search	No of Feat.
Without Scale	86.19%	87.62%	87.14%	87.14%	4.7
Standard	86.19%	87.62%	87.14%	87.14%	4.7
Min-Max	86.19%	87.62%	87.14%	87.14%	4.7
Box-Cox	86.19%	87.62%	87.14%	87.14%	4.7
LN	86.19%	87.62%	87.14%	87.14%	4.7
Min-Max (All At.)	86.19%	87.62%	87.14%	87.14%	4.7
Standard (All At.)	86.19%	87.62%	87.14%	87.14%	4.7

RF

Table A.1.13: Detailed Results of RF on German Data

Random Forest	Without FS and GS	Feature Selection	Feature S. and Grid S.	Grid Search	No of Feat.
Without Scale	74.67%	75.33%	76.33%	79.00%	16.7
Standard	75.00%	73.33%	77.67%	79.00%	14.3
Min-Max	74.33%	75.00%	77.67%	78.67%	19.0
Box-Cox	74.67%	74.33%	74.00%	80.33%	17.7
LN	75.00%	74.33%	75.33%	80.67%	17.7
Min-Max (All At.)	73.60%	72.00%	77.67%	80.33%	15.7
Standard (All At.)	75.00%	74.00%	76.33%	79.00%	16.3

Table A.1.14: Detailed Results of RF on Australian Data

Random Forest	Without FS and GS	Feature Selection	Feature S. and Grid S.	Grid Search	No of Feat.
Without Scale	83.81%	85.71%	85.71%	86.67%	11.0
Standard	83.81%	85.24%	86.19%	86.67%	11.0
Min-Max	83.81%	85.24%	86.67%	86.67%	12.0
Box-Cox	83.81%	86.19%	85.71%	86.19%	11.3
LN	83.81%	85.24%	86.67%	86.19%	12.0
Min-Max (All At.)	81.43%	85.24%	87.14%	86.67%	12.0
Standard (All At.)	81.43%	85.24%	87.14%	86.67%	12.0

XGB

Table A.1.15: Detailed Results of XGB on German Data

XGBoost	Without FS and GS	Feature Selection	Feature S. and Grid S.	Grid Search	No of Feat.
Without Scale	79.67%	78.67%	78.33%	81.67%	15.7
Standard	79.67%	78.67%	78.33%	81.67%	15.7
Min-Max	79.67%	78.67%	78.33%	81.67%	15.7
Box-Cox	79.67%	78.67%	78.33%	81.67%	15.7
LN	79.67%	78.67%	78.33%	81.67%	15.7
Min-Max (All At.)	79.67%	78.67%	78.33%	81.67%	15.7
Standard (All At.)	79.67%	78.67%	78.33%	81.67%	15.7

Table A.1.16: Detailed Results of XGB on Australian Data

XGBoost	Without FS and GS	Feature Selection	Feature S. and Grid S.	Grid Search	No of Feat.
Without Scale	85.71%	87.14%	87.62%	89.05%	9.0
Standard	85.71%	87.14%	87.62%	89.05%	9.0
Min-Max	85.71%	87.14%	87.62%	89.05%	9.0
Box-Cox	85.71%	87.14%	87.62%	89.05%	9.0
LN	85.71%	87.14%	87.62%	89.05%	9.0
Min-Max (All At.)	85.71%	87.14%	87.62%	89.05%	9.0
Standard (All At.)	85.71%	87.14%	87.62%	89.05%	9.0

A.2 Execution Time Tables

Table A.2.1: Runtime with GS on German Dataset

Datasets	XGBoost	Random Forest	Decision Tree	G. Naïve Bayes	Support V.M.	Multilayer N.N.P.	K-Nearest Neighbor	Logistic Regression
Without Scale	00:10:07	00:02:36	00:00:36	00:00:28	00:08:37	00:25:33	00:00:42	00:03:04
Standard	00:10:08	00:02:39	00:00:36	00:00:27	00:06:15	00:41:48	00:00:43	00:02:18
Min-Max	00:10:06	00:02:29	00:00:36	00:00:27	00:03:50	00:37:21	00:00:43	00:02:45
Box-Cox	00:09:58	00:02:42	00:00:36	00:00:29	00:06:42	00:34:06	00:00:43	00:03:02
LN	00:09:58	00:02:30	00:00:36	00:00:27	00:04:23	00:30:28	00:00:43	00:02:33
Min-Max (All At.)	00:10:07	00:02:56	00:00:36	00:00:27	00:01:31	00:40:19	00:00:42	00:01:23
Standard (All At.)	00:09:58	00:02:33	00:00:36	00:00:27	00:01:41	00:42:16	00:00:44	00:00:55

Table A.2.2: Runtime with WFS on German Dataset

Datasets	XGBoost	Random Forest	Decision Tree	G. Naïve Bayes	Support V.M.	Multilayer N.N.P.	K-Nearest Neighbor	Logistic Regression
Without Scale	00:00:30	00:01:12	00:00:01	00:00:08	05:40:49	00:01:07	0:00:09	00:00:35
Standard	00:00:30	00:01:14	00:00:01	00:00:01	00:09:38	00:01:11	00:00:02	00:00:23
Min-Max	00:00:30	00:01:23	00:00:01	00:00:01	00:07:09	00:01:11	00:00:02	00:00:23
Box-Cox	00:00:30	00:01:13	00:00:01	00:00:01	00:08:24	00:01:10	00:00:02	00:00:38
LN	00:00:30	00:01:12	00:00:01	00:00:01	00:06:45	00:01:11	00:00:02	00:00:27
Min-Max (All At.)	00:00:30	00:01:13	00:00:01	00:00:01	00:08:04	00:01:17	00:00:02	00:00:11
Standard (All At.)	00:00:30	00:01:13	00:00:01	00:00:01	00:11:23	00:01:03	00:00:02	00:00:07

Table A.2.3: Runtime with GS on Australian Dataset

Datasets	XGBoost	Random Forest	Decision Tree	G. Naïve Bayes	Support V.M.	Multilayer N.N.P.	K-Nearest Neighbor	Logistic Regression
Without Scale	00:04:07	00:01:06	00:00:15	00:00:16	00:01:47	00:11:39	00:00:25	00:00:34
Standard	00:04:07	00:01:09	00:00:15	00:00:13	00:01:37	00:10:53	00:00:30	00:00:22
Min-Max	00:04:07	00:01:10	00:00:15	00:00:15	00:01:49	00:16:59	00:00:29	00:00:29
Box-Cox	00:04:07	00:01:09	00:00:15	00:00:17	00:00:45	00:15:31	00:00:29	00:00:33
LN	00:04:07	00:01:01	00:00:15	00:00:18	00:01:38	00:12:08	00:00:25	00:00:35
Min-Max (All At.)	00:04:07	00:01:12	00:00:15	00:00:15	00:00:58	00:13:37	00:00:29	00:00:21
Standard (All At.)	00:04:07	00:01:07	00:00:15	00:00:16	00:01:41	00:13:35	00:00:27	00:00:19

Table A.2.4: Runtime with WFS on Australian Dataset

Datasets	XGBoost	Random Forest	Decision Tree	G. Naïve Bayes	Support V.M.	Multilayer N.N.P.	K-Nearest Neighbor	Logistic Regression
Without Scale	00:01:07	00:00:38	00:00:01	00:00:01	00:01:26	00:00:11	00:00:43	00:00:06
Standard	00:01:07	00:00:44	00:00:01	00:00:01	00:01:26	00:00:22	00:01:03	00:00:08
Min-Max	00:01:07	00:00:36	00:00:01	00:00:01	00:01:25	00:00:20	00:00:56	00:00:01
Box-Cox	00:01:07	00:00:36	00:00:01	00:00:01	00:01:22	00:00:19	00:00:49	00:00:03
LN	00:01:07	00:00:38	00:00:01	00:00:01	00:00:13	00:00:19	00:00:40	00:00:03
Min-Max (All At.)	00:01:07	00:00:37	00:00:01	00:00:01	00:01:37	00:00:21	00:01:04	00:00:01
Standard (All At.)	00:01:07	00:00:36	00:00:01	00:00:01	00:01:20	00:00:22	00:01:07	00:00:01

A.3 Additional Feature Related Tables

Table A.3.1: Selected Features

Algorithms	BCL	CAL
Decision Tree	[0,3,7,8,11]	[0,3,7,8,11]
K-Nearest Neighbor	[2,7,8,9,11,12,13]	[3,7,8,9,11,13]
Logistic Regression	[3,4,5,7,9,13]	[0,1,2,3,4,5,7,8,9,11]
Multilayer Perceptron	[0,1,2,3,4,5,7,8,10,11]	[0,1,2,3,4,5,7,8,10,11,12,13]
Naïve Bayes	[0,1,3,4,5,7,8,10,11,12]	[0,1,3,4,5,7,8,10,11,12]
Support Vector Machine	[0,1,2,3,5,6,7,8,9,12]	[3,5,7,8]
Random Forest	[0,1,2,3,4,5,7,8,9,10,11,12,13]	[0,1,2,3,4,5,7,8,9,10,11,12,13]
XGBoost	[0,3,4,6,7,8,9,10,11,12]	[0,3,4,6,7,8,9,10,11,12]

Table A.3.2: Attribute Codes for German Data

Name of the Attribute	Corresponding code	Name of the Attribute	Corresponding code
Checking Account	0	Number of dependence	12
Duration in month	1	Telephone	13
Credit history	2	Foreign	14
Credit amount	3	Purpose-New car	15
Savings account	4	Purpose-Used car	16
Employment since	5	Purpose-Domestic	17
Personal status	6	Co-Applicant	18
Present residence	7	Housing-Rent	19
Property	8	Housing-Own	20
Age	9	Job-Skilled employee	21
Other installment	10	Job-Unskilled employee	22
Number of credits	11	Job-Management	23

Table A.3.3: Attribute Codes for Australian Data

Name of the Attribute	Corresponding code	Name of the Attribute	Corresponding code
A	0	H	7
B	1	I	8
C	2	J	9
D	3	K	10
E	4	L	11
F	5	M	12
G	6	N	13