ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

ANOMALY DETECTION USING MACHINE LEARNING TECHNIQUES: A COMPARATIVE STUDY ON FIRST PAYMENT DEFAULT PREDICTION IN RETAIL LOANS

M.Sc. THESIS

Ahmet Talha YİĞİT

Department of Industrial Engineering

Industrial Engineering Programme

JUNE 2022

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

ANOMALY DETECTION USING MACHINE LEARNING TECHNIQUES: A COMPARATIVE STUDY ON FIRST PAYMENT DEFAULT PREDICTION IN RETAIL LOANS

M.Sc. THESIS

Ahmet Talha YİĞİT
(507191101)

Department of Industrial Engineering

Industrial Engineering Programme

Thesis Advisor: Prof. Dr. Alp ÜSTÜNDAĞ

JUNE 2022

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YAPAY ÖĞRENME YÖNTEMLERİYLE ANOMALİ SAPTANMASI: BİREYSEL KREDİLERDE İLK ÖDEMEDE BATMA TAHMİNİ ÜZERİNE KARŞILAŞTIRMALI BİR ÇALIŞMA

YÜKSEK LİSANS TEZİ

Ahmet Talha Yiğit
(507191101)

Endüstri Mühendisliği Anabilim Dalı

Endüstri Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Alp ÜSTÜNDAĞ

HAZİRAN 2022

Ahmet Talha Yiğit, a M.Sc. student of ITU Graduate School student ID 507191101, successfully defended the thesis/dissertation entitled "ANOMALY DETECTION USING MACHINE LEARNING TECHNIQUES: A COMPARATIVE STUDY ON FIRST PAYMENT DEFAULT PREDICTION IN RETAIL LOANS", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Alp ÜSTÜNDAĞ** Istanbul Technical University	.....
Jury Members :	**Assis. Prof. Dr. Ömer Faruk BEYCA** Istanbul Technical University	.....
	Prof. Dr. Selim ZAİM Istanbul Sabahattin Zaim University	.....

Date of Submission : 5 June 2022
Date of Defense : 29 June 2022

To my loved ones,

FOREWORD

I owe my deepest gratitude to my thesis advisor Prof. Dr. Alp Üstündağ. He has always been supportive and result-oriented during the preparation of this study. I wish to appreciate the Department of Industrial Engineering and all the lecturers who have made a great effort for us in any circumstances.

Finally, I must show my rather profound appreciation to my family, and my friends for supporting me in every stage of this long and tiring period of my life. I would not be able to finish this study without their unconditional and everlasting support.

Thank you.

June 2022 Ahmet Talha YİĞİT

TABLE OF CONTENTS

Page

ABBREVIATIONS

ABOD : Angle Based Outlier Detection

AE : Autoencoder

DT : Decision Tree

DQN : Deep Q-Learning

FCNN : Fully Connected Neural Networks

FPD : First Payment Default

IF : Isolation Forest

KNN : K Nearest Neighbors

LightGBM : Light Gradient Boosting Machine

LR : Logistic Regression

MDP : Markov Decision Process

PCA : Principal Component Analysis

PR-AUC : Area Under Precision-Recall Curve

ROS : Random Over-sampling

RUS : Random Under-sampling

SMOTE : Synthetic Minority Over-sampling Technique

SMOTEENN : Synthetic Minority Over-sampling Technique with Edited Nearest Neighbours

SVM : Support Vector Machine

SYMBOLS

$\boldsymbol{S}$	**:** State Space
$\boldsymbol{A}$	**:** Action Space
$\boldsymbol{T}$	**:** Transition Model
$\boldsymbol{R}$	**:** Reward Model
γ	**:** Discount Factor
λ	**:** Imbalance Ratio

LIST OF TABLES

Page

LIST OF FIGURES

Page

ANOMALY DETECTION USING MACHINE LEARNING TECHNIQUES: A COMPARATIVE STUDY ON FIRST PAYMENT DEFAULT PREDICTION IN RETAIL LOANS

SUMMARY

The banking sector is one of the sectors that adapt quickly and effectively to the modern methods provided by technological developments and is extremely flexible. As the digitization of banking products and applications becomes more widespread, data sources related to user behavior and products are also increasing. As a result, the datasets generated by these contemporary data sources have high potential as they allow banks to make their business more efficient by implementing better customer segmentation and targeting, and by improving risk management-related practices such as fraud detection in various ways. Various machine learning algorithms are widely applied and used in many different areas of banking. Loan default prediction is a crucial problem for banks to acquire less risky customers and avoid customers who are likely to default and go bankrupt. Detection of first payment default is a matter of great importance in order to prevent possible fraudulent activities and attacks against banks and to prevent possible financial losses that they may cause. In this study a dataset with 75000 observations has been used to benchmark and find an optimal model for the first payment default prediction problem. 11 different algorithms from supervised, unsupervised, and reinforcement learning based approaches are applied to the same dataset and compared to each other using the metrics such as F1 and lift score. In the model building process, the best-performing models in each of the supervision types have been selected. The isolation forest algorithm has been selected as the best performing unsupervised model, LightGBM has been select as the best supervised model while the DQN with 0.1 λ value has been selected as the best reinforcement learning model after the performance comparison. As the final step, the ensembling by weighted average techniques has been applied to the best 3 models. However, it is seen that the LightGBM model performs better than the ensemble models in terms of both F1 and lift scores. As a result, the LightGBM model is selected as the proposed model of the study, and the model has been interpreted and discussed in terms of the business perspective of the first payment default problem.

YAPAY ÖĞRENME YÖNTEMLERİYLE ANOMALİ SAPTANMASI: BİREYSEL KREDİLERDE İLK ÖDEMEDE BATMA TAHMİNİ ÜZERİNE KARŞILAŞTIRMALI BİR ÇALIŞMA

ÖZET

Bankacılık sektörü, teknolojik gelişmelerin sağladığı çağdaş yöntemlere hızlı ve etkin bir şekilde uyum sağlayan ve son derece esnek olan sektörlerden biridir. Bankacılık sektörünün benimsediği teknolojik yöntem ve ürünler, atm güvenliğini artıran şifreleme teknolojilerinden, online ve mobil bankacılık ürünleri gibi fintech uygulamalarına kadar çok çeşitli alanlarda etkin rol oynamaktadır. Bankacılık ürün ve uygulamalarının dijitalleşmesi giderek yaygınlaşırken, kullanıcı davranışları ve ürünlerle ilgili veri kaynakları da artıyor. Sonuç olarak, bu çağdaş veri kaynakları tarafından oluşturulan veri kümeleri, bankaların işlerini daha iyi müşteri segmentasyonu ve hedefleme uygulamaları yaparak ve dolandırıcılık tespiti gibi risk yönetimi ile ilgili uygulamaları çeşitli şekillerde geliştirerek daha verimli hale getirmelerine olanak sağladığı için yüksek bir potansiyele sahiptir.

Çeşitli yapay öğrenme algoritmaları, bankacılığın birçok farklı alanında yaygın olarak uygulanmakta ve kullanılmaktadır. Algoritma türlerinin birbirlerinin yerine olarak kullanılabilmesine rağmen, denetimli yapay öğrenme algoritmaları ağırlıklı olarak kredi kartı dolandırıcılık tespiti ve kredi puanlama gibi problemlerin çözümü için uygulanmakta, denetimsiz yapay öğrenme algoritmaları genellikle müşteri segmentasyonu ve hedefleme problemleri için kullanılmıştır ve son olarak pekiştirmeli öğrenme algoritmaları ise genellikle borç tahsilat aksiyonu optimizasyonu gibi problemler için kullanılmıştır.

Anomali kelimesi Türkçe'de yaygın bir kullanıma sahip olsa da anomaliye karşılık gelen Türkçe kelime sapaklıktır. Sapaklık ise "doğal duruma, alışılagelen ölçüye, kurala uymama, aykırılık gösterme durumu" şeklinde tanımlanmaktadır. Veri bilimi alanındaki anomali tanımı ise, "nadiren veya tamamen beklenmedik bir şekilde gerçekleşen bir olgu tarafından oluşturulabilen örnek bir gözlem" olarak yapılabilir. Anormal gözlemler iyi veya kötü olarak genellenemez, çünkü normalden sapma olgusu, spesifik alana ve duruma bağlı olarak olumlu veya olumsuz bir anlama sahip olabilir. Aykırı değer tespiti olarak da adlandırılan anomali tespiti sorunu, azınlık sınıfına ait örneklerin nadiren gözlenmesi durumunda ortaya çıkan dengesiz bir sınıflandırma durumu olarak tanımlanabilir. Spam e-posta tespitinden hastalık teşhisine kadar çeşitli alana özgü sorunları çözmek için siber güvenlik, sağlık ve askeriye gibi çok çeşitli alanlarda ve sektörlerde uygulanır. Bankacılık sektörünün de anomali tespit teknikleri kullanılarak çözülen birçok sorunu bulunmaktadır. Kredi kartı dolandırıcılığı tespiti, finansal zaman serisilerinde anomali tespiti ve anormal işlem tespiti, bankacılık sektöründeki aykırılık tespit problemlerine örnek olarak verilebilir. Dolandırıcılık tespiti, bankacılık sektöründe en yaygın anomali tespit problemlerinden biridir. Ayrıca, batık kredi ve ilk ödeme batma tahmini, en önemli dolandırıcılık tespit problemlerinden bazıları olarak kabul edilmektedir. Dolandırıcılık

tespit problemlerini çözmek için birçok yapay öğrenme yaklaşımı ve algoritması bulunmaktadır. Veri setinde gözlemlerin sınıflarını belirten etiketler varsa, denetimli, denetimsiz ve pekiştirmeli öğrenme yaklaşımları uygulanarak ve verimli sonuçlar alınabilir.

Batık kredi tahmini, bankaların daha az riskli müşterileri elde etmesi ve temerrüde düşme ve batma olasılığı olan müşterilerden kaçınması için çok önemli bir problemdir. Yasal takipteki veya batık kredi, ödemeleri 90 gün geçmiş olan krediler olarak tanımlanır. Takibe alınması gereken kredinin indirimli satılması gibi maliyetli aksiyonlar olduğu için kredi veren kurumlar tarafından potansiyel batık kredilerden kaçınılmaya çalışılmaktadır. Bu sorunu önlemek için, bankalar olası problemli müşterileri önceden tahmin etmek için çeşitli karar destek sistemlerini kullanmaktadır. Sonuç olarak, bankalar başarılı yapay öğrenme modelleri kurarak ve bunları batık kredileri tahmin etmek için karar destek sistemleri olarak kullanarak daha verimli çalışabilir ve karlarını maksimize edebilirler. Batık kredilerin özel bir durumu olan ilk ödemede batma durumunun tahmini, bir müşterinin bir kredide ilk ödemeye geç kalıp kalmayacağını tahmin etmeye odaklanır. İlk ödemede batma olgusunun tam tanımı farklı ülke ve kurumlarda farklılık gösterebilmekle birlikte, bu çalışmada borçlunun bir kredinin ilk ödemesini vadesi 90 gün geçmesine rağmen yapmaması durumu olarak tanımlanmaktadır. Bir diğer deyişle müşteri, ilk ödeme tarihinden sonraki ilk 90 gün içinde tek bir ödeme bile yapmamalıdır. Batık kredilerin özel bir versiyonu olduğu için nadiren gerçekleşmektedir çünkü batık krediler bile nadiren gözlemlenmektedir. İlk ödemede batma tahmini, bankalara yönelik olası dolandırıcılık aktivitelerinin ve saldırıların önüne geçilebilmesi ve bunların yol açabilecekleri olası finansal zararların önlenmesi açısından büyük önem taşıyan bir konudur.

Bu çalışmanın amacı, Türkiye'de faaliyet gösteren bir bankadan alınan bir veri setini kullanılarak kredi başvurularının ilk ödeme batma olasılıklarını tahmin etmektir. Veri seti, her gözlemin bir müşterinin kredi başvurusunu temsil ettiği 75000 gözlemden oluşmaktadır. 75000 gözlemden sadece 750'si ilk ödemede batmış müşetirler ve bu gözlemlerin anomali olduklarını belirtmek için 1 olarak etiketlendiler, örneklerin geri kalanı çoğunluk sınıfı olarak kabul edildi. Azınlık sınıfın tüm datadaki toplam gözlemlerin içindeki oranı yüzde 1 kadar az. Azınlık sınıf oranının son derece düşük olması nedeniyle problem bir anomali tespit problemi olarak değerlendirilmektedir. Bu amaca ulaşmak için otokodlayıcılardan tam bağlantılı yapay sinir ağlarına, karar ağaçlarından pekiştirmeli öğrenme tekniklerine kadar çeşitli yapay öğrenme modelleri uygulanmış ve karşılaştırılmıştır. Ayrıca, daha etkili modeller elde etmek için farklı denetim türleri arasında algoritmalara çeşitli birleştirme teknikleri uygulanmıştır. Tüm modelleri karşılaştırabilmek için tüm anomali tespit modelleri için ortak birer metrik olarak kullanılabilen kesinlik, anma, F1, ve kaldıraç metrikleri kullanılmıştır. Yazarın bildiği kadarıyla, bu çalışma, ilk ödemede batma sorununu çözmek için denetimsiz, denetimli ve pekiştirmeli öğrenme tabanlı anomali tespit yöntemlerinden çeşitli algoritmaları uygulayan ve karşılaştıran ilk çalışmadır. Sonuç olarak, bu çalışma hem araştırmacılar hem de uygulayıcılar için anormallik tespiti ve ilk ödemede batma tahmini literatüründe neredeyse her tür anormallik tespit yöntemini tek bir veri setinde kıyaslayan kompakt bir kaynak bulma konusunda faydalı olacaktır. Ayrıca, denetimli yapay öğrenme modelleri oluşturulurken veri kümesine çeşitli örnekleme teknikleri uygulanmıştır ve performansa olan etkileri gözlemlenmiştir.

Uygulamanın en başında, model kurma bölümünde temiz ve kullanıma hazır bir veri setine sahip olmak için, veri seti bir takım ön işleme aşamalarından geçirilmiştir. Bazı algoritmalar, eksik değerler ve ölçeklenmemiş verilerle çalışma yeteneğine sahip olsa

da, bu çalışmada tüm modeller tarafından kullanılabilecek tek bir veri kümesine sahip olmak için veri kümesindeki boş değerler dolduruldu ve değişkenler ölçeklendirildi. Bunun yanında, öncelikle verileri daha iyi anlamak için tüm değişkenlerin hedef değişkenle olan ilişkileri kontrol edilerek tek tek incelendi. Daha sonra değişken seçimi ve boş değerlerin doldurulması ile ilgili gerekli uygulamalar yapılarak veri seti hazır hale getirildi.

Model kurma aşamasında çeşitli yapay öğrenme tabanlı anomali tespit algoritmaları uygulanmış ve karşılaştırılmıştır. Açı bazlı dışa düşen tespiti (ABOD), temel bileşen analizi (PCA), k en yakın komşular (KNN), izolasyon ormanı (IF) ve otokodlayıcılar (AE) bu çalışamada eğitilen gözetimsiz öğrenme bazlı anomali tespit algoritmalarıdır. lojistik regresyon (LR), karar ağaçları (DT), destek vektör makineleri (SVM), hafif gradyan artırma makineleri (LightGBM) ve tam bağlantılı yapay sinir ağları (FCNN) algoritmaları ise uygulanan gözetimli öğrenme yöntemleridir. Gözetimli öğrenme algoritmalarının yanında örnekleme metodları da kullanılarak verideki dengesizlik probleminin çözülmesi hedeflenmiştir. Rastgele örneklem artırma (ROS), rastgele örneklem azaltma (RUS), sentetik azınlık artırma (SMOTE), ve düzenlenmiş en yakın komuşlar ile sentetik azınlık artırma (SMOTEENN) yöntemleri gözetimli öğrenme algoritmaları eğitilirken uygunlamıştır. Veriyi yeniden örnekleme yöntemleri ve uygulanan düzeltme oranları bütün gözetimli modeller için model üst parametrelerinin yanında optimize edilmiştir. Son olarak, derin q öğrenme (DQN) modeli eğitilmiştir ve diğer üst parametrelerin yanı sıra, verideki hedef sınıf dengesizliğini gösteren λ üst parametresi optimize edilmiştir.

Modeller değerlendirme verisi üzrinde test ediliğinde, en başarılı olan gözetimsiz öğrenme metodunun IF, en başarılı olan gözetimli öğrenme metodunun LightGBM ve DQN modelinde en iyi performans gösteren λ değerinin 0.1 olduğu gözlemlenmiştir. Son bir aşama olarak bu 3 en iyi modelin tahminlerinin kombinasyonlarından oluşan çeşitli modeller kurulmuştur fakat performanslarda bir iyileşme gözlemlenmemiştir. Sonuç olarak en iyi performansı gösteren model 0.208 F1 skoru ve 18.7 kaldıraç skoru ile LightGBM modeli olmuştur. Bunun yanında nihai LightGBM modelinin kullandığı en önemli 3 değişken sırasıyla; müşteri aktifliği, müşterinin son 6 ay içinde yaptığı kredi başvuru sayısı, ve KKB skoru olarak bulunmuştur.

1. INTRODUCTION

The banking industry is one of the various other industries which are highly flexible by means of adapting the contemporary methods provided by technological improvements rapidly and effectively. The technological methods and products adopted by the banking industry are related to a wide variety of disciplines from encryption technologies, which are increasing the safety of automated teller machines, to fintech applications such as online and mobile banking products. As the digitalization of banking products and applications is becoming increasingly widespread, user behavior and products related data sources are increasing. As a result, the datasets generated by these contemporary data sources are having a high potential for banks to improve their business in several ways such as making better customer segmentation and targeting applications, and risk management-related applications such as fraud detection [1].

Machine learning is a subfield of artificial intelligence discipline besides the fields such as robotics, computer vision, etc. Machine learning can be defined as optimizing a criterion metric by utilizing the data from similar examples or earlier experiences to conclude with predictive and/or descriptive results concerning the aim of the application [2]. There is a vast variety of machine learning algorithms to achieve this optimization task which is basically grouped under three main classes as supervised, unsupervised, and reinforcement learning algorithms. Each class of algorithms is utilized in different kinds of situations and domains to solve different problems.

Several types of machine learning algorithms have been applied and utilized widely in many different areas of banking. Despite the interchangeable application possibilities of different algorithm types, supervised machine learning algorithms have been mainly applied for the solution of problems such as credit card fraud detection, and credit scoring [3,4], unsupervised machine learning algorithms have been generally utilized for problems related to customer segmentation and targeting [5], and finally reinforcement learning algorithms have been used in problems like collection action optimization [6].

1.1 Anomaly Detection and Banking Industry

An anomaly is defined as “something different, abnormal, peculiar, or not easily classified, something anomalous” or “deviation from the common rule” in the English dictionary [7]. In the data science domain, the definition can be translated as an abnormal sample in a dataset that can be created by a phenomenon that is expected to happen rarely or totally unexpected. The anomalous samples cannot be generalized as good or bad because the deviation from the expected pattern can have a positive or negative meaning depending on the specific domain and case.

Anomaly detection, which is also called outlier detection, the problem can be defined as an extreme case of imbalanced classification when the samples of minority class are observed rarely. It is applied in a wide range of domains and sectors such as cybersecurity, healthcare, and the military to solve various domain-specific problems from spam email detection to disease diagnosis. The banking industry also has many problems which are solved by using anomaly detection techniques. Credit card fraud detection, financial time series anomaly detection, and abnormal transaction detection are some examples of anomaly detection problems in the banking sector.

Fraud detection is one of the most common anomaly detection problems in the banking sector. Moreover, loan default and first payment default prediction are considered as one of the most important fraud detection problems. There are many machine learning approaches and algorithms to solve the fraud detection problems. If the dataset has the labels available, supervised [8], unsupervised [9] and reinforcement learning [10] approaches can be applied and give fruitful results.

1.2 First Payment Default

Non-performing loan, also called loan default, prediction is a crucial task for banks to acquire the customers who are less risky and avoid the ones who are likely to default. A non-performing loan is defined as the loan in which the payments are 90 days past due. Non-performing loans are tried to be avoided by the lender institutions because there are costly actions such as selling the loan at a discount which is needed to be taken. To avoid this problem, banks use various decision support systems to predict the possible delinquent borrowers in advance. Consequently, banks can operate more efficiently and maximize their profits if they build successful predictive machine

learning models and utilize them as decision support systems to predict non-performing loans.

First payment default (FPD) prediction, which is a specific case of loan default, focuses on predicting whether a customer will be late for the first payment on a loan [11]. Although the exact definition of first payment default can vary in different countries and banks, in this study it is defined as the case when a borrower does not make the first payment till 90 days past due on a loan. This means not even a single payment has done in the first 90 days after the first payment date. Because it is a specific version of non-performing loans, it happens rarely. It is a significant area of study because possible fraud attacks on banks can be prevented and the potential financial damage that they may cause is avoided. FPD is a topic of interest for banks along with the loan default problem because it is considered as an attack on the bank planned by an individual or a group of people. It is assumed that the intention of the applicants, that does not pay any single payment on a loan, is already shaped before the application. This is the main reason why the FPD problem is studied separately from the loan default problem.

1.3 Problem Statement

The aim of this study is to predict the first payment default probabilities of loan applications using a dataset from a bank which is operating in Turkey. The dataset contains 75000 observations where each observation represents a loan application of a customer. From 75000 observations, only 750 have defaulted in the first payment, so labeled as 1, the rest of the samples have been considered as majority class. Having a minority class ratio as small as 1 percent, the problem is considered an anomaly detection problem because of the extremely low minority class ratio. Several machine learning models such as the ones based on autoencoders, fully connected neural networks, gradient boosted trees, and reinforcement learning has been applied and compared to achieve this goal. Besides, several ensembling techniques are applied to the algorithms in or between the different supervision types to obtain more accurate models. To make the comparison for all models, precision, recall, F1 score and lift score metrics, which can be utilized as a common metric for all anomaly detection models, are utilized. After the model selection, the final model has been interpreted by the feature importances and results in terms of the business-related insights.

To the best of the author's knowledge, this study is the first which applies and compares various algorithms from unsupervised, supervised and reinforcement learning based anomaly detection methods to solve the first payment default problem. Consequently, this study can be beneficial for both researchers and practitioners to find a compact source in the anomaly detection and first payment default prediction literature which benchmarks almost all kinds of anomaly detection methods on one dataset. Moreover, several sampling techniques are applied to the dataset when building supervised machine learning models.

1.4 Report Structure

The rest of the study is organized as follows: Literature review of machine learning, anomaly detection, and first payment default, methodology used in the study, dataset explanation (exploration), model building and comparison, and finally conclusion and discussion. In the literature review section, the earlier studies about the topic in the literature have been examined to understand the key concepts of the study. Moreover, the algorithms that are applied in the study have been introduced briefly. In the methodology section, the roadmap of the study is introduced. In the dataset section, the dataset used in the study is investigated and pre-processed in various aspects. The model building and comparison section reports the model building processes and compares and interprets the final models. In the conclusion and discussion part, the results and findings are discussed and ideas for future studies are presented.

2. LITERATURE REVIEW

The literature review section introduces the main concepts related to the study and creates connections with the introduced concepts and the earlier related studies done in the literature. It is organized as an introduction of machine learning concepts and methods, the applications of them for anomaly detection problems in the literature, and finally, the anomaly detection methods applied for first payment default prediction and earlier studies in the literature.

2.1 Machine Learning

Artificial intelligence is a field which collaborates with many other fields and covers a large variety of subfields such as machine learning, natural language processing, and computer vision [12]. Machine learning is the subfield of artificial intelligence aiming to make computers to adapt new and unseen circumstances and extract useful patterns from data by utilizing different kinds of algorithms and approaches [3,12]. Machine learning methods can be classified into two categories as classical machine learning and deep learning by considering the algorithmic difference between having an architecture based on artificial neural networks or not. The main difference between the classical machine learning and deep learning methods is that the success of classical machine learning algorithms highly depends on the quality of the hand-crafted features, which requires domain knowledge, while deep learning algorithms parameterize and automatize the feature engineering process [13]. Consequently, the deep learning approaches are more suitable for the tasks which are based on the processing of unstructured data such as computer vision and natural language processing [14], and for the tasks with large datasets [15]. On the other hand, the gradient boosted tree algorithms such as XGBoost [16] and LightGBM [17], which are considered as classical machine learning approaches, generally outperform the deep learning based models in the classification and regression problems with tabular data datasets [18].

Machine learning algorithms are grouped under four categories in the literature which are supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning algorithms with respect to the supervision type and amount needed [19]. In supervised learning, for each observation, there is a label to which the fitted machine learning model learns a mapping from the predictor with the aim of either predicting labels for the future unseen observations or understanding the relationship between the predictors and the label [20]. On the other hand, there are only predictor variables for each observation in the unsupervised learning problems. Because having no associated labels creates a somehow blind environment, machine learning models are used to extract relationships either between the predictors or between the observations to provide beneficial insights about the problem [20]. Finally, semi-supervised learning has various application settings for both supervised and unsupervised machine learning problems. One of the most common applications of semi-supervised learning techniques is semi-supervised classification when there is a small amount of labeled data and a huge amount of unlabeled data available. Pseudo-labeling the unlabelled data as the first step and using them together with the labeled data as the second step is expected to show a better performance than using only the labeled data [21].

Reinforcement learning is based on learning from the interaction which makes the concept of it differ from other machine learning approaches. It is a goal-directed learning method that learns from interactions with a given environment. Reinforcement learning aims to learn mappings from situations to actions to maximize a reward signal in the long run. The learning from interactions process requires balancing a trade-off between exploration and exploitation which is a challenge exclusive to reinforcement learning [22].

2.2 Anomaly Detection

Anomaly detection term in data science refers to the identification of the rarely observed samples by their distinguishing characteristics compared to the majority of the samples in a dataset [23]. For anomalous observations in a dataset, “anomaly” or “outlier” terms are used interchangeably in the literature [24]. Depending on the aim of the study, anomaly detection techniques can be utilized as a data pre-processing and cleaning tool to detect and discard outliers from the dataset [25]. On the other hand, in

various domains such as healthcare, cybersecurity, and finance, the anomaly detection itself can be the main point of interest and the aim of the studies in order to detect potentially malicious events or explore previously unknown behaviors, which is also called novelty detection [26,27]. For the solution of the problem settings which have the aim of detecting anomalies, there are a wide variety of methods and approaches applied and discussed in the literature as Figure 2.1 illustrates.

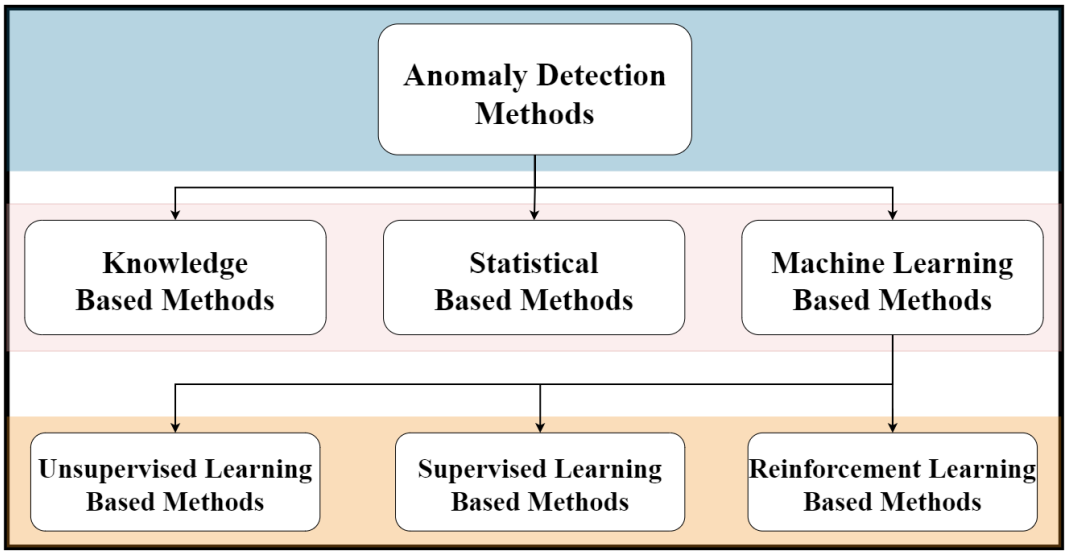

Figure 2.1 : Anomaly detection methods taxonomy.

Anomaly detection methods have been categorized by many distinctive characteristics in the literature. As shown in Figure 2.1, in the first layer, the anomaly detection methods are separated into three distinct categories as statistics-based, knowledge-based, and machine learning based approaches [28]. As discussed widely in the literature, machine learning based approaches have a higher potential to achieve better performances when enough data is provided in the high dimensional settings [28, 29]. For example, a well-known statistics-based anomaly detection method called Grubb's test can be applied only in a one-dimensional setting [30]. Moreover, knowledge or rule-based methods require domain expertise and are not flexible enough for constantly changing problem settings. On the other hand, machine learning based anomaly detection methods are more convenient to apply because of their characteristics of being suitable to work with high dimensional settings and being generalizable and adaptive to changes [28]. Machine learning based anomaly detection methods can be further categorized by considering either the supervision type of the algorithms such as supervised, unsupervised, and reinforcement learning based approaches, or the main characteristics of the algorithms such as density-based,

distance-based, and rank-based methods [31]. As seen in Figure 2.1, in this study, the further categorization of machine learning based anomaly detection methods is done considering the supervision type of the algorithms.

In the following subsections, the main characteristics and applications of unsupervised learning, supervised learning, and reinforcement learning based anomaly detection methods and the algorithms in the literature are introduced.

2.2.1 Unsupervised learning based approaches

Unsupervised machine learning algorithms are utilized widely in the anomaly detection literature. In the unsupervised setting, when there are no labels available, it can be assumed that the dataset contains only normal observations because of the extremely low percentage of existing anomalies. Under this assumption, an unsupervised anomaly detection algorithm basically creates a model of the normal data in order to measure the deviations of samples from the normal model and use the resulting metric as an outlier score [24]. The calculated outlier, or anomaly, the score shows how much a sample is close to being a normal sample; consequently, a high anomaly score means that the individual sample has a high potential to be an anomaly. To make classification, generally a threshold set which assumably separates the anomaly observations from the normal observations. Therefore, they are commonly utilized as classifiers that learn how to identify samples that do not belong to the normal class.

Although the unsupervised anomaly detection models are very commonly applied and studied in the literature because of the advantage of having no need for labels to be trained, they generally perform worse when the labels are available compared to the supervised learning based anomaly detection algorithms. Thus, generally supervised learning based algorithms are preferred over unsupervised ones when labels are available [24]. However, in some cases, unsupervised models can perform better than the supervised ones, or they can be used to improve the performances of supervised models in ways of using either ensemble techniques such as majority voting or using the anomaly scores generated by the unsupervised models as an input for the supervised models [32].

There are a wide variety of unsupervised anomaly detection algorithms used by both researchers and practitioners. The unsupervised anomaly detection algorithms have

been classified in the surveys and books from the literature differently because of the distinctive characteristics of them [24, 31, 33]. For example, the isolation forest algorithm is categorized as a classification-based algorithm in some studies while it is categorized as an ensemble-based algorithm in others [33, 34]. While both of the classifications are true with respect to the criteria that the studies choose for classification, it is important to decide on a consistent classification criterion as in Figure 2.2.

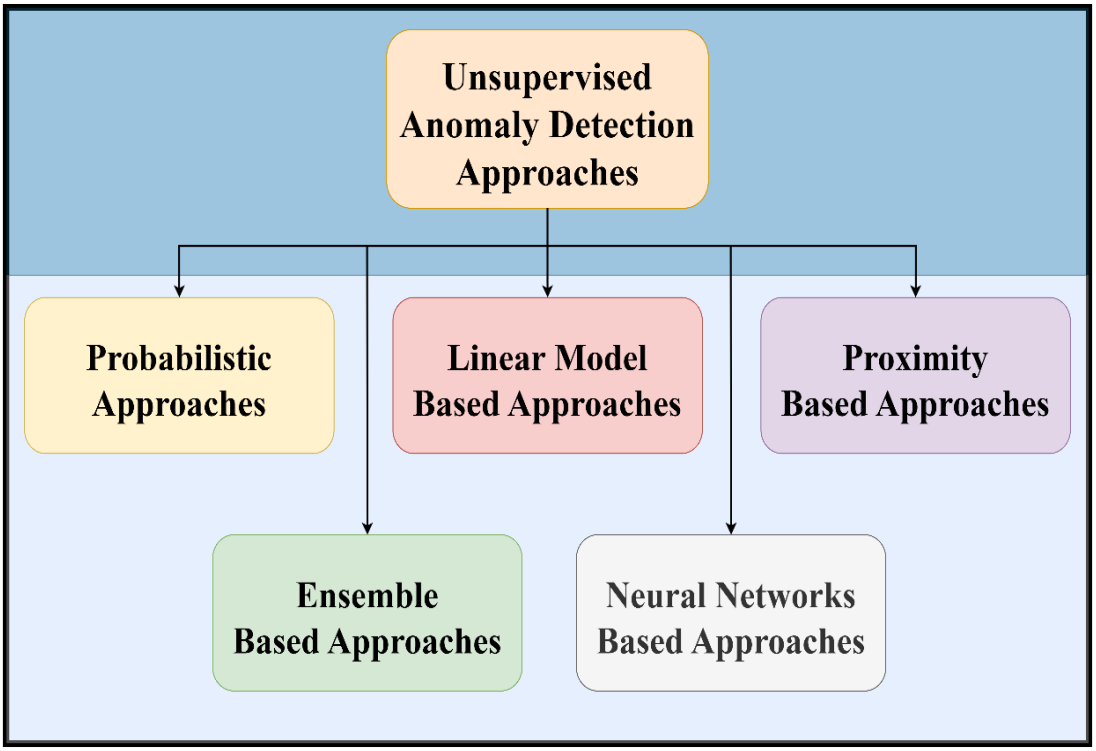

Figure 2.2 : Unsupervised anomaly detection methods taxonomy.

As shown in Figure 2.2, in this study, the taxonomy of the unsupervised anomaly detection algorithms is adopted from the PyOD library [34]. To cover all the unsupervised anomaly detection methods, an algorithm from each class is selected for further explanation and application in the model building section. The unsupervised machine learning based algorithms used in this study are listed as: angle-based outlier detection (ABOD), principal component analysis (PCA), k nearest neighbours (KNN), isolation forest (IF), and autoencoders (AE).

2.2.1.1 Angle-based outlier detection

Angle-based outlier detection (ABOD) is a probabilistic type of unsupervised learning based anomaly detection algorithms such as stochastic outlier detection and copula-

based outlier detection algorithms [35-37]. ABOD has been proposed as a solution for the "curse of dimensionality" problem widely encountered in proximity-based unsupervised anomaly detection approaches. It achieves this by using the variance of angles between the difference vectors of samples to other pairs of samples [35] as illustrated in Figure 2.3.

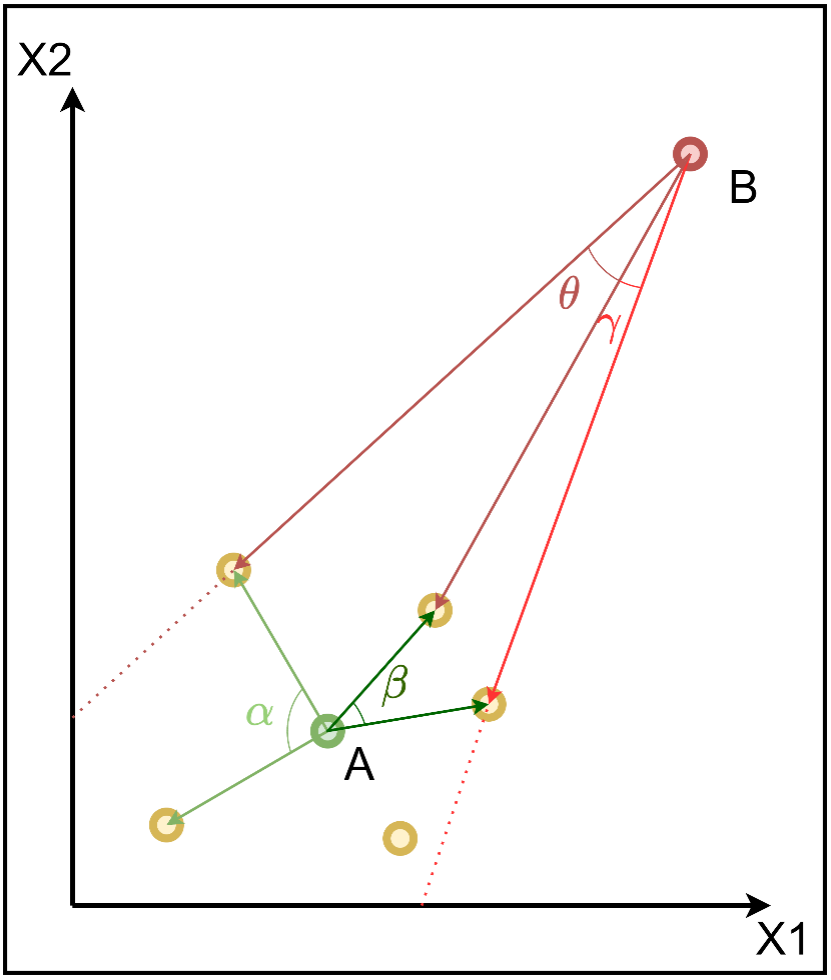

Figure 2.3 : Illustration of angles between pair samples of a normal sample and an abnormal sample.

ABOD algorithm depends on the fact that the variance of the angles between difference vectors from a sample to each possible sample pair is higher when the sample is a member of a cluster than the sample is an outlier. In other words, when the variety of observed angles for a sample gets narrower, the probability of that sample being an anomaly gets higher [24]. As shown in Figure 2.3, the sample A cannot have an angle to comprise all the observations, which means that the distance vector angles between A and other sample pairs have a high variance. For example, the α angle is much bigger than β. On the other hand, the sample B has quite a narrow angle which comprises all the observations, resulting in a lower variance of distance vector angles

between B and other sample pairs. For example, the θ angle is quite similar to γ. As a result, sample B can be labeled as an anomaly while A can be considered a normal sample.

ABOD algorithm calculates a metric, called angle-based outlier factor (ABOF), for each observation basically by computing scalar products of the difference vectors of all possible point triples and weighting them inversely concerning their distance from the query point while calculating the variance. As a result, the ABOD algorithm computes anomaly scores for each observation without the need for additional hyperparameter tuning [35]. ABOD algorithm has been widely applied in anomaly detection literature for benchmarking and performance comparison [38, 33].

2.2.1.2 Principal component analysis

Principal component analysis (PCA) is generally used for dimensionality reduction in machine learning literature [39]. However, it is used as a classifier in anomaly detection literature as other linear model based unsupervised anomaly detection methods such as one-class support vector machines, and minimum covariance determinant methods [40-42]. The main advantage of PCA based anomaly detection over statistical-based or other machine learning based anomaly detection techniques is that it is computationally cheap when deployed to action, and it does not need many restrictions or assumptions on the dataset [40].

PCA based anomaly detection algorithm firstly performs PCA on the correlation matrix of the normal samples. Then it computes two PCA scores by considering both the extreme feature values and the correlation structure. Outliers, or anomalies, are expected to have a tendency to have extreme feature values and to be discordant with the correlation structure. Finally, makes the decision to classify if the sample is an outlier or not by comparing the calculated PCA scores with pre-specified threshold values [40] in the literature, PCA based anomaly detection is applied in a wide range of domains from cybersecurity attack detection [43] to network intrusion detection [44].

2.2.1.3 K-nearest neighbours

K nearest neighbours (KNN) based outlier detection is a distance, or proximity, based outlier detection algorithm such as local outlier factor and connectivity-based outlier

factor algorithms [45,46,47]. Proximity-based approaches have a limitation on performance because of the "curse of dimensionality". It can be said that the higher the number of dimensions of the dataset gets, the less effective the proximity-based approaches are. As can be seen in Equation 2.1, the contrast between the minimum and maximum distanced points fades as the dimensionality *d* increases [35]. On the other hand, the dataset used in this study is not a remarkably high dimensional dataset; consequently, this approach may perform well in this study.

$$\lim_{d\to\infty} \frac{dist_{max} - dist_{min}}{dist_{min}} \to 0 \tag{2.1}$$

KNN based outlier detection algorithm basically calculates an outlier score for each sample in the dataset using the sum of distances from their k nearest neighbours. The higher the score, the more probably the sample is an anomalous one [48]. In the literature, KNN based outlier detection algorithm is highly adopted generally in the studies analyzing lower-dimensional datasets [49].

2.2.1.4 Isolation forest

The isolation forest (IF) algorithm is an ensemble-based unsupervised anomaly detection method such as feature bagging and lightweight online detector of anomalies algorithms [50-52]. Contrary to the general concept of constructing a profile for normal samples to detect the abnormal observations, the IF algorithm works with the principle of explicitly isolating anomalies [50]. When compared with the proximity-based approaches, IF is computationally more efficient because it does not require calculating any distance measure; moreover, IF is more immune to the "curse of dimensionality" and large datasets [50].

IF algorithm uses isolation term in the meaning of separating anomalies from the rest of the observations. To achieve this task of isolation, the IF algorithm uses random trees to partition the data recursively until all the samples are isolated. Because anomalous observations are expected to be different and minority compared to the normal observations, they are expected to be isolated in the earlier partitions. As shown in Figure 2.4, point A is probably an anomalous sample because it is isolated in the first partition while point B is isolated in the third partition. Consequently, when an ensemble of a sufficient number of random trees isolates a sample with a shorter path of consecutive partitions then it can be said that the observation is a potential anomaly

[50, 53]. In the literature, the IF algorithm has been utilized successfully in the context of anomaly detection for high-dimensional datasets [54, 55].

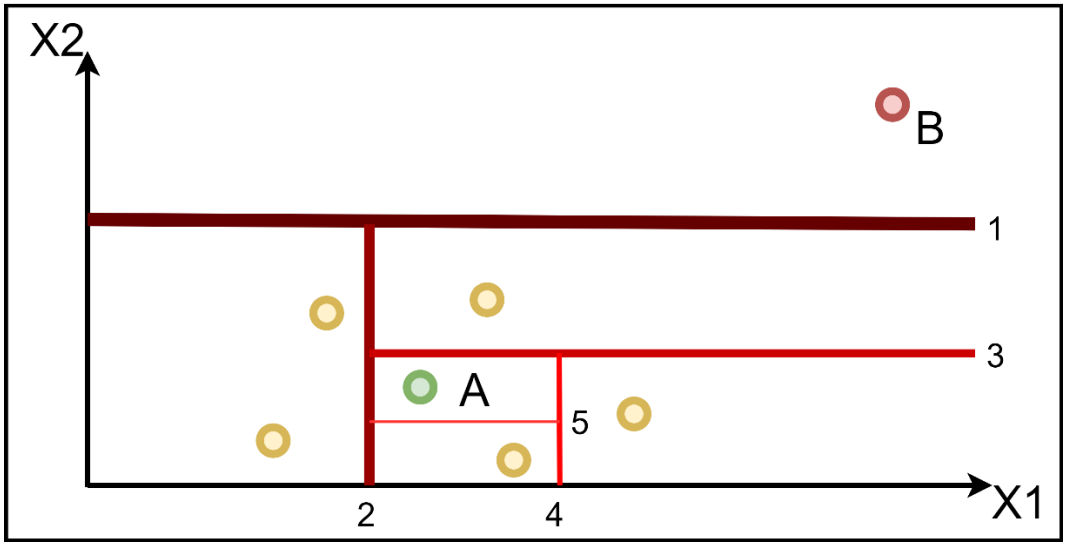

Figure 2.4 : Illustration of isolating a normal sample in 5 splits and an abnormal sample by 1 split.

2.2.1.5 Autoencoders

Representation learning, also referred to as feature learning, is a common concern for artificial intelligence and data science researchers. The goal of progressing towards and creating artificial intelligence can only be achieved by making it learn to detect and use the hidden beneficial information from the raw visual or any other form of data to understand the surrounding environment [56]. Deep learning algorithms are outperforming the other machine learning algorithms by learning the data representation themselves in the tasks such as speech recognition and signal processing, object recognition, and natural language processing. However, when the data is unlabelled, there is a natural need for various kinds of feature learning algorithms. One of the most widely used algorithms to learn representations of unlabelled data is autoencoders. Autoencoders are deep learning architectures that learn representations from raw data by using it directly as both input and output.

Basically, an autoencoder is formed by two parts of stacked hidden layers which are called encoder and decoder as Figure 2.5 shows. The encoder function maps the input (raw data) to a feature (latent) space, and the decoder function maps the feature space back to the input space [56]. The loss function is called reconstruction error which is calculated as the difference between the input and output data. The ultimate goal of

the algorithm is to encode the image to a lower-dimensional representation which can be decoded to the original image by minimizing the reconstruction loss.

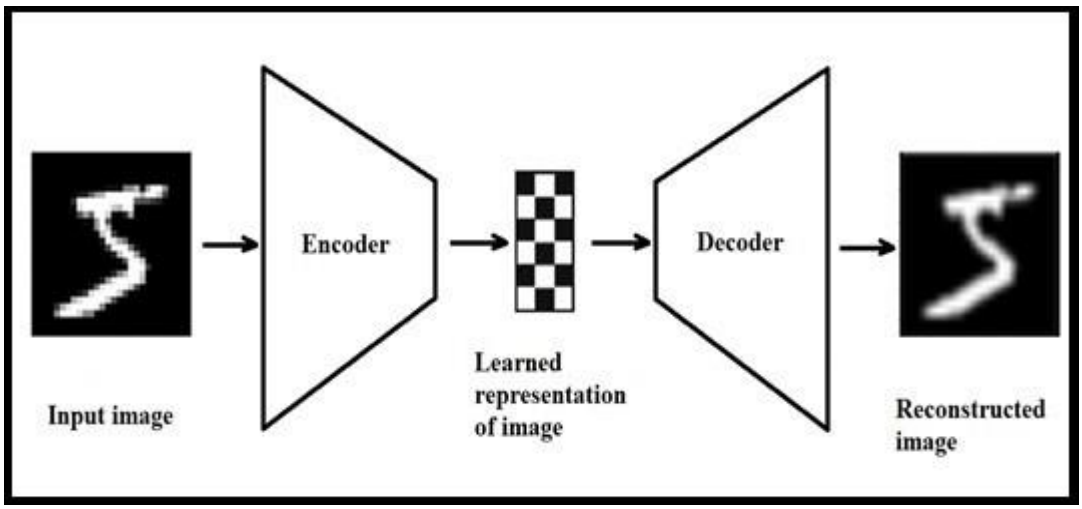

Figure 2.5 : An illustration of a basic auto encoder which learns the representation of a sample image from MNIST handwritten digit dataset.

Despite its wide usage as a feature extractor and/or a data compressor in the deep learning and machine learning literature, autoencoders (AE) have also been used as an unsupervised anomaly detection algorithm [57]. In AE based outlier detection concept, the anomaly score of a sample is the average reconstruction error over all the features. It basically trains the model with only normal samples to learn effectively encoding and decoding them. After training, the anomalous samples are expected to have higher reconstruction losses compared to the normal ones because they were not involved in the training of the model [57]. In the literature, there are numerous studies that apply and/or improve over AE based anomaly detection technique successfully [58, 59].

2.2.2 Supervised learning based approaches

Among various anomaly detection methods, supervised machine learning based approaches are considered the most efficient and effective. In an unsupervised anomaly detection scenario, the detected anomalies can contain noise and they can be from different kinds of anomalies related to the domain. The main reason for this problem of targeting the right anomalies is the lack of labels. On contrary, because the labels of the interested types of anomalies are present in the supervised learning based anomaly detection cases, the models can be trained to detect the specific type of anomaly. Supervision in anomaly detection models is highly advantageous to improve model performances so that even a small amount of supervision, which means semi-

supervised machine learning based approaches, may improve the performance a lot [24].

There is a vast literature on semi-supervised learning based anomaly detection with a variety of different methodologies and algorithms applied. However, the main purpose of using semi-supervised learning based techniques in anomaly detection is to utilize a greater amount of data by labeling them with the small amount of labeled data at hand [23, 60, 61]. In this study, the dataset has the ground truth labels for all instances of normal and abnormal samples; therefore, there is no possibility to use semi-supervised learning algorithms to leverage the number of usable samples. Because creating a model of the normal behavior and measuring the distances of test samples from it is a completely different setting, unsupervised models have the potential to perform well in this study. On the other hand, in the supervised learning concept, only fully supervised algorithms and methods are applied in this study because of the unnecessity of labeling the already labeled samples again.

Fully supervised learning based anomaly detection can actually be considered a special case of the classification problem. It can be basically thought of as a classification problem which learns to classify samples as normal and abnormal. The distinguishable aspect of fully supervised anomaly detection from the classification is that in fully supervised anomaly detection, there is at least one class that is represented in the data very sparsely. The exceedingly small amount of presence of the minority class results in a problem of high or extreme imbalance in the dataset [32]. Another issue which is encountered in the literature of anomaly detection using supervised machine learning is that the labels may be inaccurate and misleading for especially the abnormal samples. In this study, the data is historic, and the labels are assigned carefully according to the previous records of the customers' payments and default status; consequently, the possibility of having misleading labels is ignored.

There are various supervised classification algorithms in the literature that are used in the anomaly detection studies. Several types of algorithms have diverse types of advantages and disadvantages. For example, a logistic regression based approach provides interpretability of the model in cost of a possible performance drop compared to ensemble learning based approaches. To cover the supervised learning algorithms as much as possible, 5 algorithms have been decided to be applied in this study: First of all logistic regression (LR) and decision tree (DT) models have been selected as

they are the most interpretable members of parametric and nonparametric algorithms classes while they may achieve a significant performance themselves as baselines for other algorithms or support other algorithms with ensembling techniques. Secondly, the support vector machine (SVM) algorithm is decided to be applied because of its wide adoption in the anomaly detection literature. As the third step, light gradient boosting machine (LightGBM) is selected among the ensemble learning approaches because of its high potential for performance as a boosting method. Finally, a fully connected neural network (FCNN) is decided to be applied as a neural network based approach because it is a suitable architecture for tabular datasets and has a high potential to achieve superior performance.

2.2.2.1 Logistic regression

Logistic Regression (LR) is a supervised parametric classification method which is suitable for problems with binary target variables. It is considered the most basic parametric supervised learning based anomaly detection method. Although LR is a very simple algorithm in intuition it has a high potential to achieve efficient results on classification and anomaly detection tasks. Moreover, the most important and beneficial aspect of applying LR to solve a problem is that the resulting models are easily interpretable. The interpretability of the model can be more important than a high-performance model in some domains. LR can be explained as a linear regression model of which predictions are constrained between 0 and 1 by applying the logistic function, which is also called as sigmoid function, transformation shown in Equation 2.2 [20].

$$\sigma(\hat{y}) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}} \tag{2.2}$$

The main reason behind applying the logistic function to the results of linear regression for classification problems is that the predictions of a binary classification problem should be between 0 and 1 which shows the probability of being a member of a class. In linear regression cases, the probabilities can always be lower than 0 and higher than 1 for certain inputs. As shown in Figure 2.6, after applying the logistic function, the predictions have been arranged to be between 0 and 1 [62]. After this transformation, the predictions for classes are made using 0.5 as a decision boundary. A sample with

an output of 0.75 is classified as a member of class 1 while a sample with 0.25 is classified as 0.

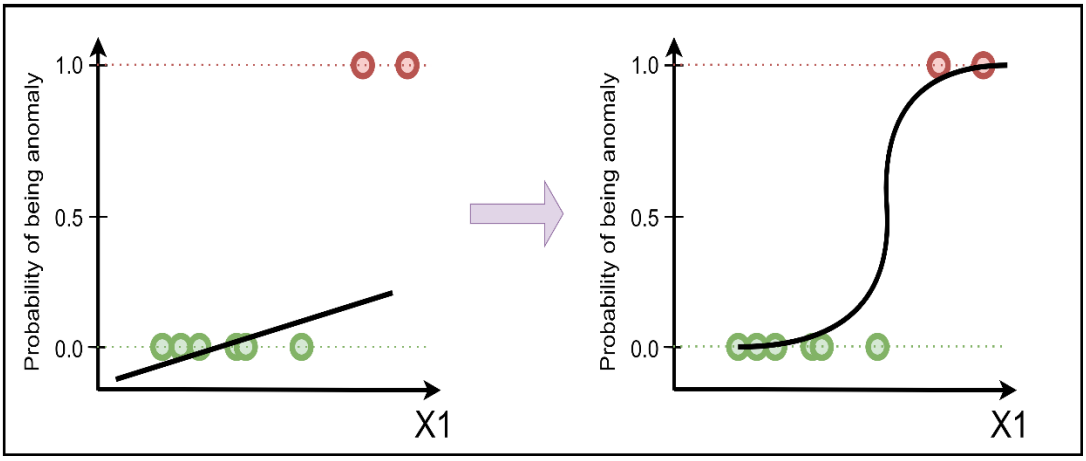

Figure 2.6 : Linear regression versus logistic regression on classification task.

As already shown in Equation 2.2, the LR model is a parametric machine learning model. The parameters which are aimed to optimize are weights (w) and bias terms (b). To optimize them, a loss function called cross-entropy loss, which is also called negative log-likelihood loss, is minimized using stochastic gradient descent, which will be explained further in the neural network section. The cross-entropy loss is shown in Equation 2.3 [20].

$$L_{CE}(\hat{y}, y) = -[y \, log \, \sigma(\hat{y}) + (1 - y) \, log(1 - \sigma(\hat{y}))] \tag{2.3}$$

LR based anomaly detection has been applied and utilized in the literature by many researchers because of its sufficient performance and computational effectiveness. In [63], the author has suggested an anomaly detection method based on logistic regression which has shown superior performance over the benchmarking algorithms such as naïve bayes, support vector machines, and deep neural networks on different datasets with respect to the metrics of AUC, accuracy, and F1 score. In another study [64], the authors assess the logistic regression based anomaly detection performance on a dataset created synthetically by using a cyber-physical system and showed that precision, recall, and specificity metrics are achieved higher than 99 percent. On the other hand, a study shows that LR based anomaly detection is time and memory effective compared to other supervised machine learning based anomaly detection methods such as support vector machines and random forests experimenting with a variety of datasets [65].

2.2.2.2 Decision tree

A decision tree (DT) is a supervised nonparametric machine learning algorithm that is used for both regression and classification problems which is also suitable for the problems with binary target variables. Decision tree based models use independent variables to divide the samples into regions one by one, so that similar observations will be in the same region [20]. As Figure 2.7 illustrates, a two-dimensional space is partitioned into for different regions. For example, the observations with a feature "X1" value lower than or equal to the threshold "t1" are assigned to the region "R1", otherwise they are partitioned further.

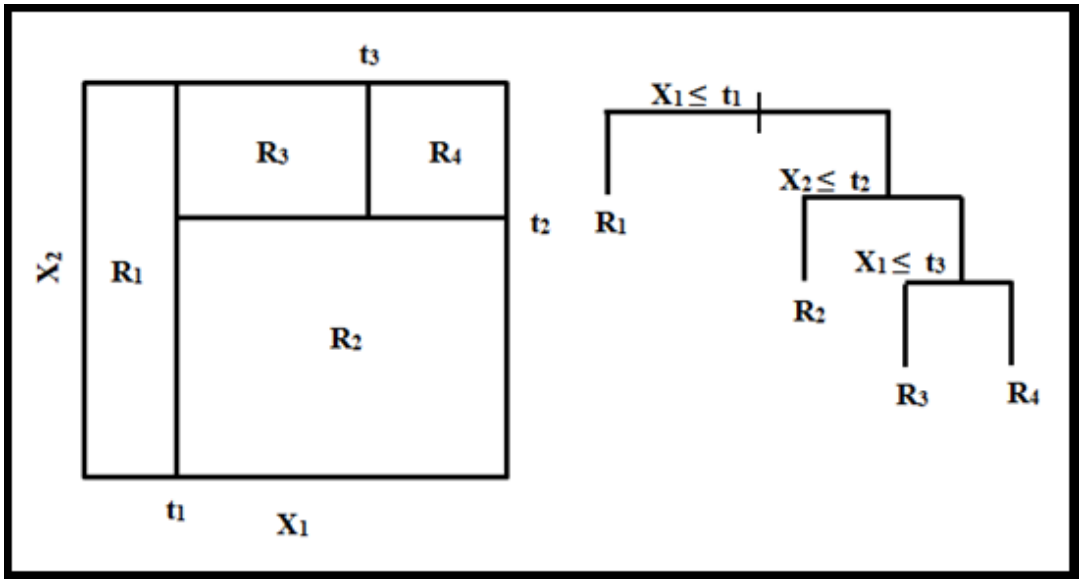

Figure 2.7 : Binary splitting and decision tree model visualization.

To create the best possible regions (R), decision tree based models use the below formula and try to minimize the Gini index value. And the method that is used to build this tree is called recursive binary splitting. Gini index is a measure of purity for the splits, because a Gini value close to 0 means that the specific node contains mostly samples from a single class [20]. Equation 2.4 shows the calculation of the Gini index. $\hat{P}_{mk}$ represents the proportion of the samples from class k in the region m.

$$G = \sum_{k=1}^{K} \hat{P}_{mk}(1 - \hat{P}_{mk}) \tag{2.4}$$

After building the tree model on a training dataset, the predictions of test samples are made according to the majority class in a region. If the region that the test sample has

assigned contains a majority of normal samples, the test sample is classified as a normal sample, otherwise it is classified as abnormal.

DT models are quite easy to interpret, even easier than LR models because their hierarchical structure of prediction is easily interpretable and suitable for visualization. On the other hand, DT models are much better at handling outliers and unscaled variables. Depending on the problem and the structure of the dataset, DT models may perform well [20].

DT based anomaly detection has been utilized in the anomaly detection literature by many researchers. In [66], the authors have proposed an anomaly detection method combining the decision tree and K-means clustering algorithms which achieves better performance than both individual algorithms and other algorithms such as SVM and KNN. Another study [67] shows that the DT classifier performed best on benchmark datasets compared to the much more complex and extreme gradient Boosting method. On the other hand, as discussed in the LR section, DT based anomaly detection is also time and memory effective compared to many other supervised approaches [65].

2.2.2.3 Support vector machine

Support vector machine (SVM) is a machine learning approach designed for classification problems. SVM is an improved version of the support vector classifier algorithm which is an extension of the simple base algorithm called maximal margin classifier. The basic tool that all three methods use is called a hyperplane which is an affine subspace with n-1 dimensions to separate the n-dimensional space. The equation of a hyperplane of an n-dimensional space is shown in Equation 2.5. For example, in a two-dimensional space, the hyperplane is a line which partitions the two-dimensional space into two separate regions. As a result, this hyperplane can be utilized as a classifier when it gets a value higher than 0, the sample is assigned to one of the binary classes, otherwise it is assigned to the other class [20].

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_n x_{in} = 0 \tag{2.5}$$

Maximal margin classifier algorithms are designed for problems where the classes are perfectly separable by an affine hyperplane. In this case, by fitting the hyperplane to the space in a way to maximize the distance, also called as margin, between the hyperplane and the samples of each class closest to the hyperplane, which are also

called as support vectors. By maximizing this margin, a simple hyperplane is selected by maximal margin classifier algorithm among infinite possible alternatives. On the other hand, because the algorithm only considers the support vectors when fitting the hyperplane, it is highly likely to overfit to training samples. An illustration of maximal margin classifier can be seen in Figure 2.8.

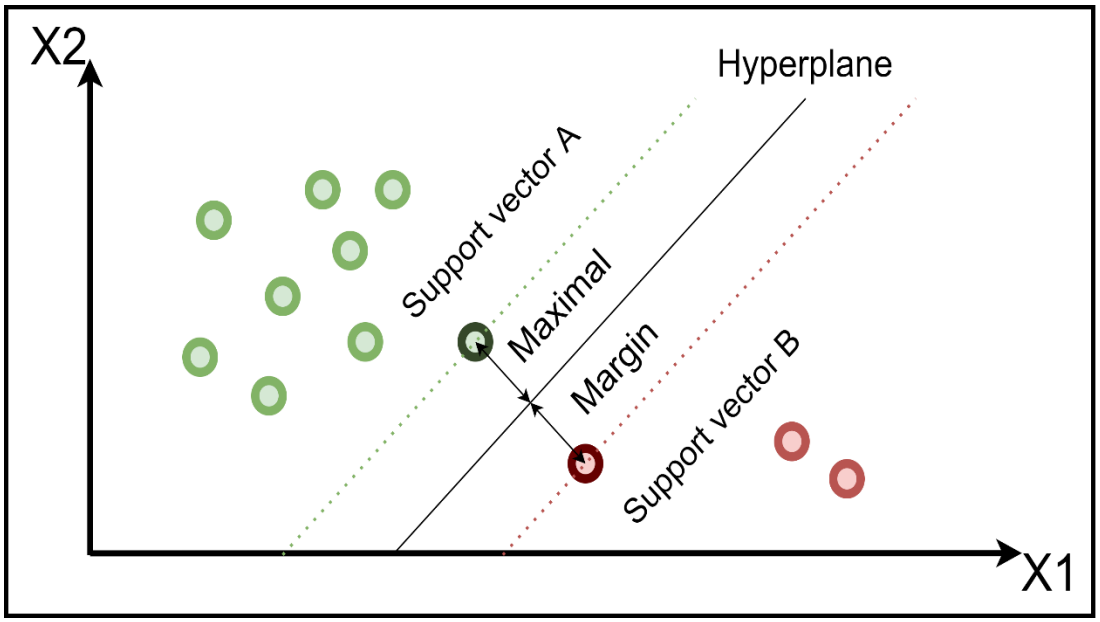

Figure 2.8 : Maximal margin classifier visualization.

To avoid the possible overfitting caused by the nature maximal margin classifier, support vector classifier algorithm has been used to enable a proportion of each class can be located on the wrong side of the hyperplane. This way, the algorithm became more robust to the slight changes in the dataset. Even though support vector classifiers are pretty well suited for the problems that a linear hyperplane meets the requirements, in cases where boundaries between classes are non-linear, support vector classifiers will not perform well enough. In this kind of cases, the SVM algorithm is used to enlarge the feature space by adding higher-order polynomial terms with the aim of finding a linear boundary between classes on the higher dimensional space and mapping it back to the original feature space by using polynomial or radial kernels. In other words, SVM is a satisfactory performance classification method utilizing the blessing of dimensionality [20].

In the anomaly detection literature, a linear unsupervised model based on SVM called one-class support vector machine algorithm is immensely popular [41]. However, in this study, SVM is used as a supervised binary classification based anomaly detection

method; thus, the literature review focuses on the studies which use it in this way. In a study, SVM is found as the best performing algorithm among many other machine learning based algorithms such as KNN and ANN in the task of electrocardiogram anomaly detection [68]. On the other hand, in many studies, it is used as a benchmark model, however, could not achieve to perform better than other models [65].

2.2.2.4 Light gradient boosting machine

DT algorithm has been a subject of ensemble learning approaches widely in the machine learning literature. There are various ensemble learning approaches to improve the performance of DT algorithms by increasing accuracy and avoiding overfitting such as bagging, which creates different versions of the dataset by bootstrapping to fit different DT models on each copy and aggregate all DT models as a single predictor, and random forests, which uses the same principle of bagging with just a difference of using only a proportion of all features by random selection in each bootstrapped dataset. Another popular ensemble learning approach applied with DT is the gradient boosting algorithm which is a combination of gradient descent and boosting methods. Basically, the gradient boosting algorithm fits an ensemble of DTs sequentially on a dataset by using the previous learners' residuals as the target value for the new predictor [20].

LightGBM is a novel approach to gradient boosting decision tree methods such as extreme gradient boosting in the aspects of efficiency, scalability, and performance on large datasets with large feature spaces. LightGBM improves over the existing gradient boosting methods by using gradient-based one-side sampling and effective feature bundling techniques. Gradient-based one-side sampling algorithm basically down-samples the data by selecting the samples with the higher residual in the estimation of gain. On the other hand, effective feature bundling method groups the sparse features together to avoid the unnecessary computational costs. Applying these methods, LightGBM decreased the computational complexity of the gradient boosting algorithm significantly with no significant loss in performance [17].

LightGBM algorithm has been utilized in the anomaly detection literature for both its sufficient performance compared to other machine learning based anomaly detection techniques and computational efficiency. In one study, a methodology based on adaptive synthetic over-sampling technique, to overcome the data imbalance problem,

and LightGBM was adopted to create a fast and high accuracy network intrusion detection system [69]. As a result, the LightGBM algorithm has beaten other supervised machine learning models in terms of time efficacy and accuracy on the benchmarking datasets used in the study [69]. In another study, the authors applied LightGBM on a benchmark dataset, and achieved better results in metrics such as accuracy and false alarm rate compared to the earlier studies which applied different supervised machine learning based methods on the same dataset [70].

2.2.2.5 Fully connected neural networks

Artificial neural network models have been applied in a wide range of domains and the problems in recent years. Although the history of artificial neural networks dates back to the 1940s, the computational resources, data sources, and the implementations of the algorithms were not ready to support the algorithm. Only after many years, with the increased availability of data and computational resources, and with the improvements on the implementations, the artificial neural networks have become a popular machine learning method [13].

One of the main reasons why artificial neural networks, also called as deep learning, algorithms have outperformed many traditional machine learning algorithms in many domains is that the deep learning algorithm does not need feature engineering. Achieving high performances with traditional learning algorithms heavily depends on feature engineering whereas deep learning algorithms allow end-to-end learning by automatizing the feature engineering phase, as stated in Goodfellow et al. [13]. For example, in a binary classification problem of images of cats and dogs, the features such as fur colour, body height and eye width etc. should be extracted to train a successful classifier using a traditional machine learning algorithm. The need for the hand-crafted features requires domain knowledge and time to create a well-prepared representation of the data for the algorithm. In contrast, deep learning algorithms can be trained using the raw image data and achieve even better performance. The main reason deep learning allows training with raw data is that it parameterizes the feature extraction (or engineering) process and optimizes them to learn the most efficient features to achieve the desired task. Consequently, the deep learning models extract useful features, which are not easily interpretable, for the problem and use them as the final inputs for the classifier, or output layer. An illustration of the process of

parameterization and automation of features using the raw data is done by a simple example of image processing with neural networks in Figure 2.9.

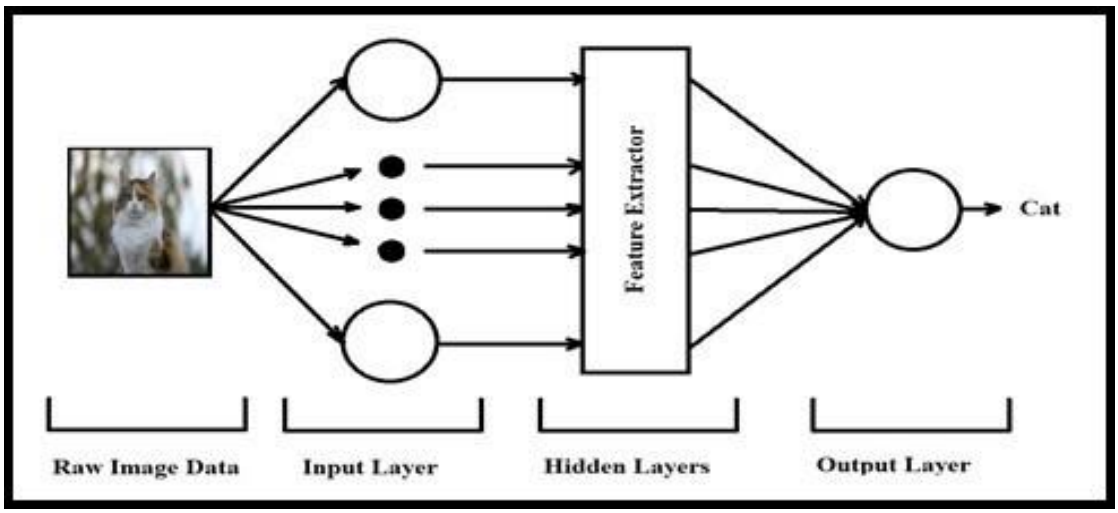

Figure 2.9 : Illustration of a simple fully connected neural networks architecture.

As shown in the Figure 2.9, hidden layers between the input and output layers of fully connected neural network architectures are playing the role of a parameterized feature extractor. The parameters of the neural network model are optimized by optimization algorithms such as stochastic gradient descent. Consequently, it can be said that the superiority of deep learning algorithms over other machine learning algorithms comes from their ability to learn and use a representation of raw data instead of using the raw data itself or handcrafted features to maximize its success in the given learning task. As illustrated in Figure 2.10, gradient descent algorithm basically starts with a random initial point on the parameters space and tries to find the local minimum point of the loss function by stepping in the negative direction of the derivative of the point [13].

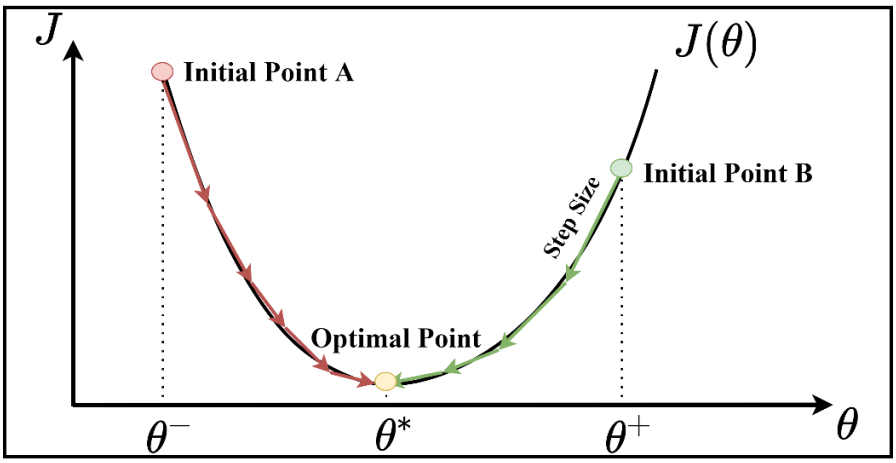

Figure 2.10 : Illustration of gradient descent algorithm.

The gradient descent algorithm basically finds the optimum parameters of the FCNN model which minimizes the loss function as shown in Figure 2.10. On point A, the initial weights θ^- are lower than the optimal weights θ^*, so the gradient descent follows the Equation 2.6 and increases the weight because of the negative direction of the slope. On the other hand, the other random initial point B has weights θ^+ higher than the optimal weights θ^*, so it decreases the weights till reaching the optimal point.

$$\theta := \theta - \alpha \frac{\partial J(\theta)}{\partial \theta} \tag{2.6}$$

While finding the optimal point, gradient descent uses a learning rate α which is a hyperparameter critical for learning. If it is set to a higher or a lower value than it should be, the model does not converge to the optimal parameters. To help the gradient descent algorithm, there are improved versions such as stochastic gradient descent with momentum and Adam algorithms [71]. On the other hand, the architecture of the model can also be defined as a hyperparameter. If the network is too complex for the model, it can overfit to the training set. To avoid overfitting, batch normalization and dropout techniques can be applied to have a network model which generalizes well.

In anomaly detection literature FCNN algorithm has been applied in numerous studies in recent years. For example, the authors have built a magnetic anomaly detection model based on FCNN to improve over a traditional method called orthonormal basis function detector. The results show that the FCNN has performed better than the existing method [72]. In another study, the authors have used two different public network anomaly detection datasets to benchmark several deep learning models such as long-short term memory, variational autoencoder, and FCNN, and traditional machine learning models such as Adaboost, random forest, and SVM. In both of the datasets, the authors observed that long-short memory has achieved the best performance; however, the FCNN method performed better than the traditional machine learning algorithms in both of the datasets [73].

2.2.3 Reinforcement learning based approach

Reinforcement learning is a method to solve sequential decision-making under uncertainty problems. Reinforcement learning algorithms basically aim to find an optimal policy for the sequential decision-making problems by making no assumptions about the dynamics of the environment and interacting with the environment in a blind

manner. By the trial-and-error interactions, the agent collects data related to the states and rewards of the model. Using the data collected, the policy is updated periodically till a satisfactory policy, or an optimal policy is reached [22]. As shown in Figure 2.11, the reinforcement learning problem can be illustrated as trying to train a robot mouse to learn to find an optimal solution for reaching a cheese reward by learning from its observations collected from the interactions with the maze.

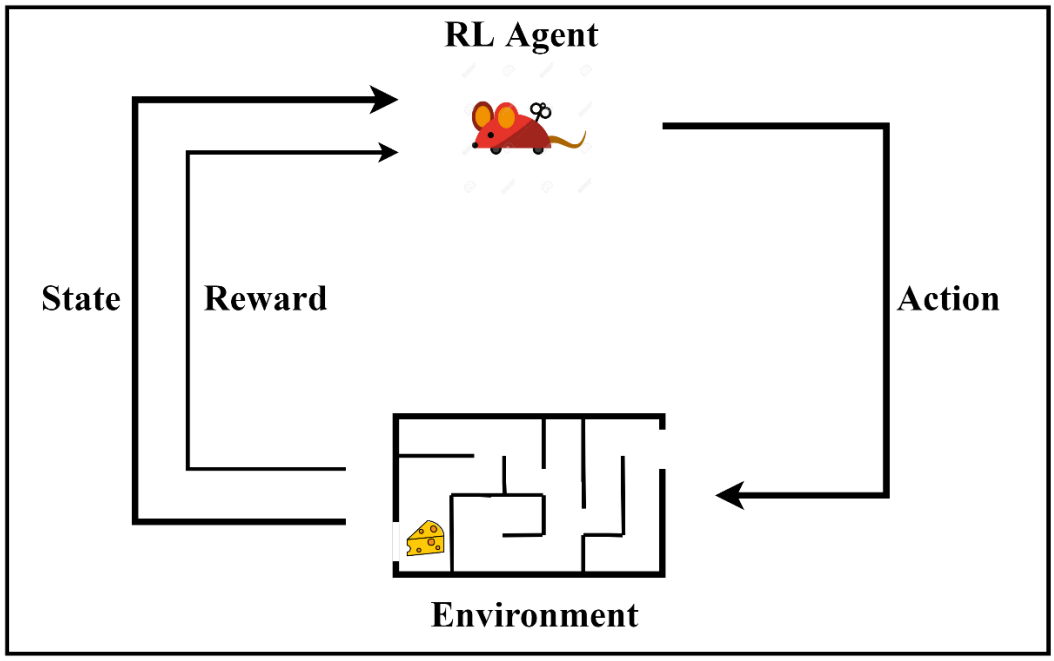

Figure 2.11 : Illustration of a simple reinforcement learning problem and interaction between agent and environment.

A reinforcement learning problem can be formalized as a Markov decision process (MDP). An MDP is constituted by a state space S, an action space A, a transition model T, a reward model R and a discount factor γ. In a deterministic MDP setting, the state space is the observation that the agent takes from the environment. The agent uses a policy π, which is mapping from states to actions, to take an action in the current state. After taking the action, the next state is determined by the transition model of the environment and the reward signal is determined by the reward model. The process of being in a current state, taking an action, and transitioning to the next state and having a reward signal is called a trajectory. Consecutive trajectories starting from the initial state and ending in the terminal state is called an episode [74]. As a result, any problem from any domain must be formulated as an MDP to be solved by a reinforcement learning algorithm.

There are various reinforcement learning algorithms to solve MDPs in the literature. The reinforcement learning algorithms are categorized as on-policy and off-policy with respect to the policy that is used to make decisions. If the value function and the data collection policies are the same, then the algorithm is called on-policy, otherwise it is called off-policy. Moreover, they can be categorized as online and offline reinforcement learning algorithms concerning the availability of the agent interacting with the environment. If the agent can interact with the environment during training and samples the trajectories in an online manner, the algorithm is called an online reinforcement learning algorithm, other it is classified as offline [74].

Q-learning algorithm is an online off-policy reinforcement learning algorithm. It is online because it interacts with the environment during the training process to sample trajectories. It is off-policy because it uses the epsilon greedy policy, which allows exploration during the training process, while collecting trajectories instead of the greedy policy itself. Q-learning algorithm basically samples an episode of trajectories from the environment and uses them to update the state-action value function $Q(s, a)$ accordingly using the temporal difference update shown in Equation 2.7.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{2.7}$$

After updating the state-action value function, the policy is updated as selecting the actions which maximize the value in each state as shown in Equation 2.8.

$$\pi(s) \leftarrow \operatorname*{argmax}_{a} Q(s,a) \tag{2.8}$$

In the problems with large state space, the Q table is not efficient to use because of the high memory requirement for keeping a parameter for each state and action. Instead of keeping all the values themselves, an approximator of the values is used. The value table can be replaced with a linear regression model which approximates the original values of the states with using much fewer parameters and memory. On the other hand, the value function approximator can be a deep neural network as well, as it is done on DQN [75]. The reinforcement algorithms that use deep neural net architectures as function approximators are called deep reinforcement algorithms.

Even though the application work of reinforcement learning for feature extraction of classification is widely used, using reinforcement learning for classification task is an

idea which is not a deeply studied topic [76]. In the study [77], the authors have proposed an MDP formulation for the classification task. In their proposed formulation, they have different agents for each class in the problem, and the agents simply decide to keep or ignore the features of a sample to create a state representation actively to make classification. This approach has achieved better scores than the multi-layer perceptron model while performing remarkably close to the SVM model [77]. On the other hand, the imbalanced classification problem has been studied by [76]. The authors have designed the classification problem as a sequential guessing game with aim of using the reward function as a leverage for the importance of the minority class. They have trained a DQN agent on the imbalanced classification MDPs created from some benchmark datasets and achieved the best or the second-best performance compared to the different modified versions of deep learning, such as trained in over-sampled data, trained with cost-sensitive approach etc., models in all of them [76]. Finally, the authors of the article [10] have proposed an approach very similar to the previous paper [76] to solve the anomaly detection case of credit card fraud detection. They have trained 3 different networks of autoencoder, mediator network and a DQN agent simultaneously to extract features. The proposed DQN approach has overperformed the traditional methods like DT and LR in their benchmark dataset [10].

2.3 Sampling Methods

In supervised anomaly detection applications, a major challenge for training a well-performing model is that the extremely imbalanced nature of the datasets. Most of the classification algorithms such as FCNN, LightGBM, SVM etc, have been designed for the scenarios where the target ratios of classes are close to each other. The performance of these algorithms on imbalanced datasets is generally not satisfactory. To increase the performance of classification algorithms on imbalanced class distribution, there are many studies in the literature to either modify the algorithm or the dataset. At the algorithmic level, for example, the contributions of the minority class samples on the loss are increased purposely to increase the importance of the minority class. Similarly, the techniques on the data level are aiming to increase the importance of the minority class by simply either increasing the amount of minority class samples or decreasing the amount of majority class samples in the dataset.

Although there are various sampling methods to balance the imbalanced datasets, in this study, random under-sampling (RUS), random over-sampling (ROS), synthetic minority over-sampling technique (SMOTE), and synthetic minority over-sampling technique with edited nearest neighbours (SMOTEENN) techniques applied to apply a candidate from all kinds of sampling methods. RUS technique randomly down-samples the majority class samples in the dataset to create a more balanced target distribution in the dataset. Similarly, ROS technique randomly duplicates the minority class samples to increase the share of minority targets in the dataset. SMOTE technique is an over-sampling technique which creates synthetic data for minority class instead of duplicating them. The synthetic data creation is done as the following steps: find the nearest minority class sample to the sample of interest, take the difference between them, multiply the difference with a number from 0 to 1, and finally add the result to the sample of interest. This way the algorithm creates synthetic data which helps the machine learning models to improve the generalization on the minority class [78]. At the same time, the authors have asserted that the combination of SMOTE and under-sampling increases over the performance of under-sampling methods [78]. The final sampling method which will be applied in this study is the SMOTEENN method. SMOTEENN method is a mixed sampling method which uses SMOTE as an over-sampling method while using Wilson's edited nearest neighbours rule as an under-sampling method [79]. The basic idea behind this method is to improve generalization of the model by increasing the representation of minority class using synthetic over-sampling and avoiding class overlapping via under-sampling at the same time. SMOTEENN method has shown a superior performance over ROS on the datasets with an exceedingly small number of minority class samples [79]. However, the correct method of sampling depends on the problem and the dataset; consequently, all sampling methods discussed are used in the model building stage for comparison.

2.4 Loan Default and FPD Prediction

Loan default is the case when the borrower fails to make any payments on time. FPD is a special case of loan default in which the loan defaults without any payments. Consequently, the FPD prediction related literature is much smaller than the loan default prediction literature. Because both concepts overlap on the basic structures

such as being considered as an imbalanced classification or an anomaly detection problem, this section reviews studies on both topics together.

There has been a lot of interest in the topics of credit scoring and loan default prediction in the literature because of the highly competitive and risky nature of the financial loan lending sector. Although there have been a variety of studies and applications for the risk assessment mainly based on opinions of professionals on the borrower and simple statistical analysis, the recent applications and studies about credit scoring and loan default prediction have shifted to the machine learning discipline [80].

In another literature survey study of credit risk assessment studies using statistical or machine learning based methods; the algorithms, the techniques used to manage the data imbalance, and the feature extraction methods used in the studies published in the period between 2010 and 2018 have been discussed. According to the study, the most widely utilized 3 algorithms are LR, SVM, and FCNN algorithms, while the most frequently applied 3 techniques to overcome data imbalance are clustering, RUS, and SMOTE. Moreover, the top 3 feature extraction methods that are applied most frequently in the surveyed studies are stepwise feature selection, principal component analysis and genetic algorithm. Finally, the first 3 of the most utilized performance evaluation metrics in the studies are percentage correctly classified, the area under receiver operating characteristics curve, and type II error [4].

There is a variety of studies which compare different machine learning based loan default detection techniques in the literature. In one of the many studies, the authors aimed to predict if the installments of a purchase will be done by the customer or not using a dataset obtained by a household appliances company which offers payments by installments. Although it is not exactly a loan default prediction problem, it has remarkably similar characteristics. In the study, the problem has been considered as a supervised, imbalanced classification problem. In order to solve the problem, the authors have applied random forest probability estimation trees, logistic regression, k-nearest neighbours and bagged k-nearest neighbours methods and found out that the random forest model has performed much better in terms of ROC-AUC score on the test dataset [81].

Another comparative study aims to compare the performances of DT and random forest algorithms using the public Lending Club dataset from Kaggle. The researchers who conducted the study have used accuracy as a comparison metric to choose the better performing model among DT and RF because the dataset is not a highly imbalanced one. RF has been found as a better model because it has achieved an 80 percent accuracy while the DT model has achieved 73 percent accuracy [82].

Ensemble learning approaches are used frequently in retail loan default detection literature as can be seen in the examples using the RF algorithm [84,85]. On the other hand, another ensemble learning approach based on boosting is also applied by researchers. A study proposes an extreme gradient boosting classifier as a loan default detector [83]. The dataset used in the study is a loan credit dataset which contains three different groups of features as demographic, loan performance and previous loan related features. Using the data with these features, the author has proposed a well-performing extreme gradient boosting classifier model with 79 percent accuracy [83].

Artificial neural networks based approaches have been widely adopted in many sectors and domains as the banking sector because of their high potential to solve many unsolved business problems and improve over the current or earlier solutions. Deep learning applications in the banking sector can be divided into three main categories as customer relationship management, marketing and finally risk management. Under the topic of risk management, the loan approval problem has been approached by many researchers using various variants of artificial neural network models such as restricted Boltzmann machine, convolutional neural networks, deep belief networks, and FCNN [1].

In the study [84], the authors have designed a comparative methodology for the assessment of credit default swap contracts. Firstly, the contracts have been graded as A, B, and C according to their ratings, which serve as labels for the machine learning models. Because of having 3 different classes, the problem has become a multinomial classification problem in the study. Multinomial logistic regression, multilayer perceptron, support vector machine, and deep belief networks with restricted Boltzmann machine algorithms have been applied to the dataset and compared. In terms of all accuracy, false negative, false positive, and ROC-AUC metrics, deep belief networks have performed best while the multinomial regression model has the worst performance [84].

Another study that compares machine learning algorithms and deep learning algorithms for credit risk assessment uses a highly imbalanced dataset [85]. The original dataset in the study had less than a 2% percent share for the minority class, which is default loans. To overcome the imbalanced data problem, the authors have applied SMOTE technique to the dataset and balanced the dataset by having a ratio of 47 to 53 percent. After balancing the data, logistic regression, random forest, gradient boosting and four different FCNN models have been applied for the comparison. As a result, the study proposes the 10 most important features for different models, and that the tree-based ensemble learning models such as random forest and gradient boosting achieves better and more stable performances in both all features and the different most important features settings compared to binomial LR and different versions of FCNN models [85].

Besides supervised learning methods, unsupervised learning based anomaly detection algorithms are also applied to the loan default prediction in the literature. Unsupervised anomaly detection algorithms have generally been applied in the studies which investigate the reject inference of loans. Reject inference refers to the problem of predicting the possible outcome of the rejected loans. As the loans are already rejected by a policy, there is no possibility to have labels for these cases; consequently, applications of unsupervised and semi-supervised learning approaches are suitable to apply [86].

On the other hand, there is a handful of studies that compare unsupervised and supervised learning approaches on the same problem of loan default prediction. For example, in one study, the authors have designed a comparative approach to investigate the performances of supervised and unsupervised learning based anomaly detection approaches on two different open-source loan default datasets from Kaggle [87]. The difference between the two datasets is that one of them is highly imbalanced with a target ratio of 1% while the other is slightly imbalanced with a target ratio of 20%. In both datasets, both supervised learning approaches such as DT, and RF and unsupervised learning approaches such as IF, and one-class SVM have been applied and compared using metrics of ROC-AUC, F1 score, and normalized diagonal scores metrics. The results show that the unsupervised learning based approaches have achieved better performances of the highly imbalanced dataset compared to the

supervised learning models. On the other hand, supervised learning based approaches have achieved better performances on the less imbalanced dataset [87].

Although deep reinforcement learning applications have been proposed in the literature for imbalanced data classification, it is not widely adopted in the literature. In the financial risk domain, there is one study which uses the DQN algorithm to solve a credit card fraud detection problem. In the study, the proposed DQN based algorithm has achieved better performance than the LR and DT algorithm applied on the balanced dataset [10].

Finally, there are very few studies conducted directly on FPD prediction using any kind of anomaly detection technique in the literature. In one of the few studies, the authors have designed a comparative methodology to compare supervised learning based anomaly detection techniques such as LR, DT, SVM and naïve bayes classifier on an FPD dataset obtained from a Turkish bank [11]. In the dataset, the class imbalance is quite high with the minority class of 0.5 %; consequently, the authors have applied the algorithms on the balanced versions of the dataset using RUS and SMOTE techniques [11]. Using the performance metrics of accuracy, precision, recall and F1 score, the authors have found that all the algorithms have successful and similar performances on the under-sampled dataset. On the other hand, the models trained on the over-sampled version of the dataset have achieved worse performance on the test set with the original distribution because of the possible misguide of the synthetically created samples [11].

3. METHODOLOGY

In this section, the roadmap of the study has been described in detail, and the algorithms, sampling methods and performance metrics have been introduced and discussed.

3.1 Solution Framework for Model Selection

In this study, the main goal is to compare and benchmark various algorithms on the same dataset, and propose a final model by considering the determined performance metrics. The diagram in the Figure 3.1 shows the roadmap of this study clearly. First of all, the original dataset has been split into training, validation and test sets with the percentages of 80 %, 10 % and 10 %. After splitting the dataset and deciding on the training split, the data preprocessing applications such as feature elimination, imputation and standardization have been applied on all 3 splits using the information from the training set to avoid data leakage. When the data preprocessing steps have been finalized, the model building stage has started. In this stage, there are 3 main groups of algorithms which are unsupervised, supervised and reinforcement learning algorithms. For the unsupervised and reinforcement learning based approaches, only the original training dataset has been used for training. On the other, for the supervised learning algorithm, 5 different sets of training datasets have been used as 1 original and 4 resampled versions of the training split. For the resampled datasets, the sampling ratios have been treated as a hyperparameter for each algorithm. In all model training processes, the validation set approach has been used to guarantee the performance of the model on unseen data, and to handle with issues such as overfitting. Finally, performances of all algorithms on the test set have been reported and compared firstly among the algorithms which are in the same family such as unsupervised, supervised and reinforcement learning. Finally, 3 best models from each family, and an ensemble

model created by them have been compared and the one with the best performance has been proposed.

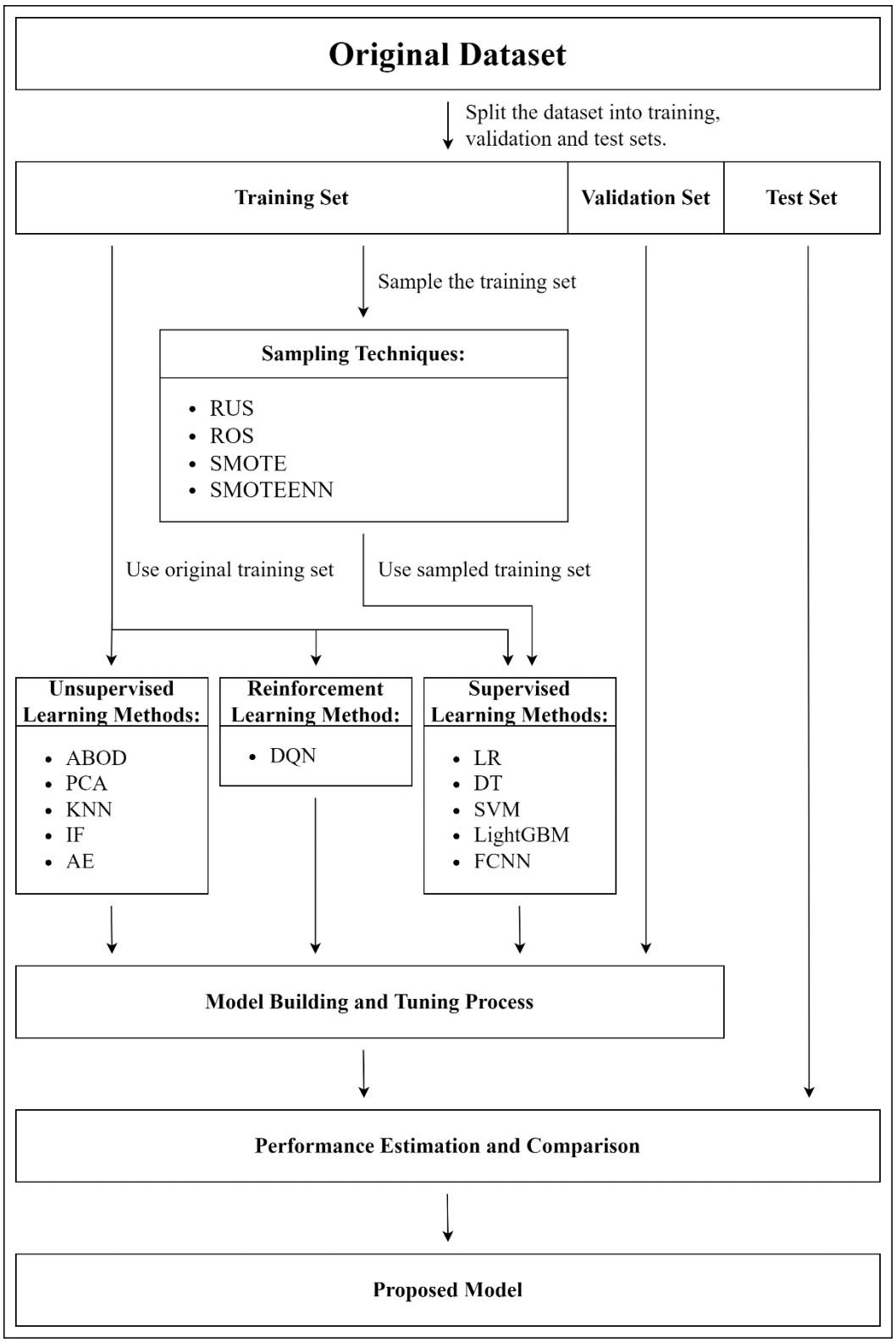

Figure 3.1 : Diagram of the roadmap of the study.

3.2 Algorithms

Several machine learning based anomaly detection algorithms have been applied and compared in this study. All of the algorithm implementations have been made in the Python programming language. For the applications of ABOD, PCA, KNN, IF, and AE algorithms, the implementation of the PyOD library has been used [34]. For the applications of the LR, DT, and SVM algorithms, the implementations in scikit-learn library have been utilized [88]. LightGBM has been used from the original python library [17]. For the implementation of the FCNN models Pytorch library has been preferred to have as much flexibility in the architectural designs of the model [89], while for the grid-search applications for FCNN, Scorch library has been used [90]. Finally, for the DQN model, the imbDRL library has been utilized to build the reinforcement learning based outlier detection model [91].

3.3 Sampling Methods

One of the most commonly and effectively applied data imbalance handling methods is resampling the data to create more balanced versions of the dataset. Sampling methods that have been applied are RUS, ROS, SMOTE, and SMOTEENN methods. They are applied for only supervised learning algorithms because the other methods exploit and handle the imbalanced nature of the original data. The hyperparameters of the sampling methods have also been determined by grid-search. All the sampling methods were applied using the implementations of them in the scikit-learn library [88].

3.4 Performance Metrics

Performance metric selection is a crucial issue for any kind of optimization problem. In machine learning applications, the aim is roughly to optimize a selected performance metric using the given dataset and algorithms. In the classification setting the most basic performance metric is accuracy while in the regression setting it is mean-squared-error. However, these general and basic metrics do not meet the performance requirements for some problems and domains. For example, in some

cases, the accuracy can be meaningless when evaluating the performance. For this kind of a situation, several different classification performance metrics have been created and used for evaluating models using the confusion table that is illustrated in Figure 3.2. For instance, in a medical diagnosis problem, there can be 4 different scenarios for the classification results. Firstly an ill person can be classified as ill (TP), an ill person can be classified as healthy (FN), a healthy person can be classified as healthy (TN) and finally a healthy person can be classified as ill (FP). Depending on the illness, any of these scenarios can be more important. For a model predicting the probability of patients getting a cold, the FP rate may not be so crucial because the treatment for getting a cold would not be very harmful for a healthy person. On the other hand, if the illness that is predicted is cancer, the FP rate should be minimized because of possible harm to a healthy person.

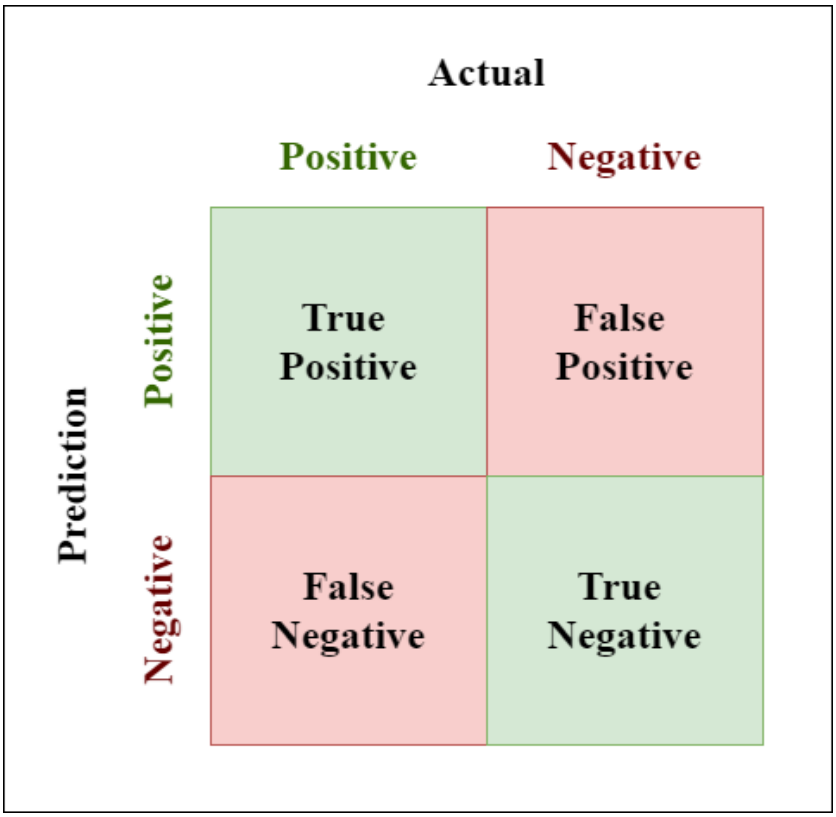

Figure 3.2 : Illustration of a confusion matrix.

In a fraud detection setting, the most important thing is to detect as many fraudulent applications as possible to avoid possible financial damage. To assure this, the recall metric, which means the proportion of true anomalies that are identified by the model,

should be maximized. On the other hand, if so many of the applications are labeled as fraud and declined by the bank, again there would be a financial loss. To be sure that not so many normal applications are declined by the bank, the precision metric, which means the proportion of true anomalies among the identified anomalies, should be maximized. Finally, the F1 score which combines both precision and recall metrics is also a good evaluation metric for anomaly detection problems. It is the harmonic mean of precision and recall metrics as shown in Equation 3.1

$$\frac{2 \times precision \times recall}{(precision + recall)} \tag{3.1}$$

Another common metric used for the comparison of model performances is the lift score. The lift score is basically interpreted as the enhancement over a random choice model. In this study, 1 % lift score has been considered as the most probable 1 % of the cases are very important. The metric is calculated as the percentage of anomalies in the most probable 1 % of the applications over the percentage of anomalies in the original data. For example, when the target ratio among the top 1 % of samples is 0.10, the lift score is calculated as 10 because the target ratio of the original data is 0.01.

As a result, for all models, precision, recall, F1 and lift scores have been calculated and the performance comparisons are made considering these 4 metrics. Apart from the performance metrics, the area under precision-recall curve metric (PR-AUC) has been used to control overfitting.

4. DATASET

The dataset used to train and test the models in this study is provided by a bank that operates in Turkey. The utilized dataset is an open-source data which is an anonymized and sampled version of the original dataset to be used in academic studies. The dataset consists of 75000 samples with a target variable and 40 predictor variables. The variables are selected according to domain knowledge and common sense; consequently, each variable has a possibility to be important and significant for the model performance. This section of the study aims to understand the dataset better and apply preprocessing before the modeling to have a clean and ready-to-use dataset for the modeling section. Although some methods such as LightGBM have the capability to handle with data with missing values and unscaled data, in this study the dataset is imputed and scaled in order to have a single dataset which can be used by all the models for benchmarking. First of all, to understand the data better, all the features are examined one by one by checking their relationships with the target variable. After that, the necessary decisions about data imputation and feature selection are made to make the data ready for the model building part.

4.1 Feature Analysis

In this part, the given dependent and independent features in the dataset will be analyzed with respect to their meanings, types, distributions, and missing values. The dataset was firstly split into 3 different sets as training, validation, and test sets as stratified using the target variable. All the analyses on features are made on the training set to avoid possible data leakage which can be caused by many preprocessing steps such as the missing value imputations, encodings of categorical variables, or normalization.

First of all, the dependent variable called “Target” is examined. It basically shows the credits which have at least one instalment with the label “0” and the ones without any instalments in the first six consecutive months are labelled as “1”, which are the anomalies aimed to be detected in this study. As it is shown in Figure 4.1, the

distribution of the dependent variable is highly imbalanced, with the 59.400 normal cases and 600 abnormal cases.

Figure 4.1 : The distribution of the target classes in the dataset. Red bar shows anomaly cases while the green bar shows the normal ones.

To further clarify the process of analysing and explaining the variables, they are grouped as binary, categorical and numerical variables. There are 14 binary variables, 8 categorical variables and 18 numerical variables in the dataset which are deeply investigated and explained in the remaining part of this section.

4.1.1 Binary variables

In the dataset, there are 14 binary variables. The explanations of the variables are shown in the Table 4.1.

Table 4.1 : Binary variables and their explanations.

Variable	Explanation
Var_1	Made credit restructuring request
Var_7	Employer's address exists
Var_8	Branch and workplace are in the same city
Var_9	Main and application branches are the same
Var_10	Customer is active at the period
Var_15	Previous loan disbursement exists

Table 4.1 (continued) : Binary variables and their explanations.

Variable	Explanation
Var_16	Branch manager opinion is positive
Var_17	Branch credit bureau instant check
Var_22	Home phone exists
Var_23	Branch sales expert deficiency opinion
Var_30	Customer has active futures products
Var_32	Application is in branch waiting pool
Var_35	Branch and home are in the same city
Var_39	Headquarters disbursement process

The binary variables have been explored deeply, and the needs for mappings, imputations and scaling have been detected for the variables that are encoded as categorical levels. All the process applied to the variables is reported one by one in the rest of this chapter.

Var_1 shows if the customer who made the application has made a credit restructuring request before ("Y") or not ("N"). As Figure 4.2 shows, the variable has mainly "N" classes, however, we can see that the minority class has a much higher proportion of "Y" classes in comparison with the majority classes. It can empirically be said that the restructured credit request can be a sign that the credit has no instalment (anomaly class). In this variable, there are 480 missing values. The one-hot-encoding and imputation processes should be applied to make use of this variable.

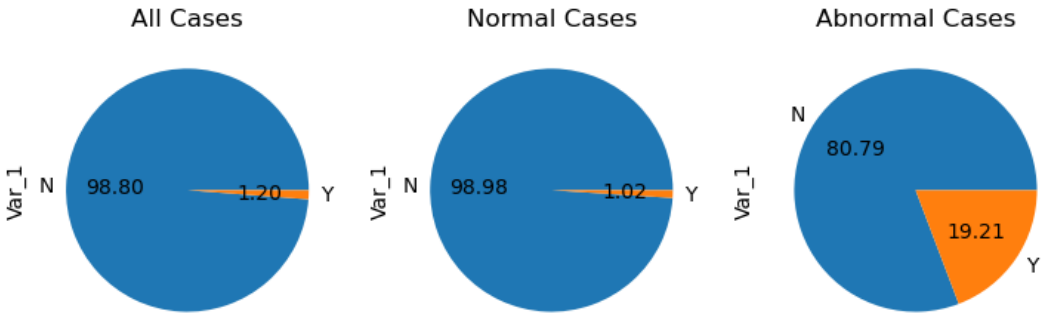

Figure 4.2 : The frequency distribution of *Var_1* variable with respect to target classes.

Var_7 is a binary variable showing if the employer's address of the customer is existing in the database ("Y") or not ("N"). When the category distributions of each class are inspected, it is found out that the abnormal samples show a slightly less tendency to have the employer's address available with a ratio of 81.1% compared to the normal samples with a ratio of 81.3%. This can be interpreted as a sign that this variable may not be so

useful for models. There are 480 missing values in this variable. . The one-hot-encoding and imputation processes should be applied to make use of this variable.

Var_8 is a binary variable showing if the application branch and workplace are in the same city. Figure 4.3 shows that there is a slight difference between the distributions of *Var_8* in normal and abnormal cases. Empirically, it can be said that if someone uses a branch outside of his/her city of work address, it is a higher possibility to have an abnormal case. There are no missing values in this variable, but still, one-hot-encoding is needed.

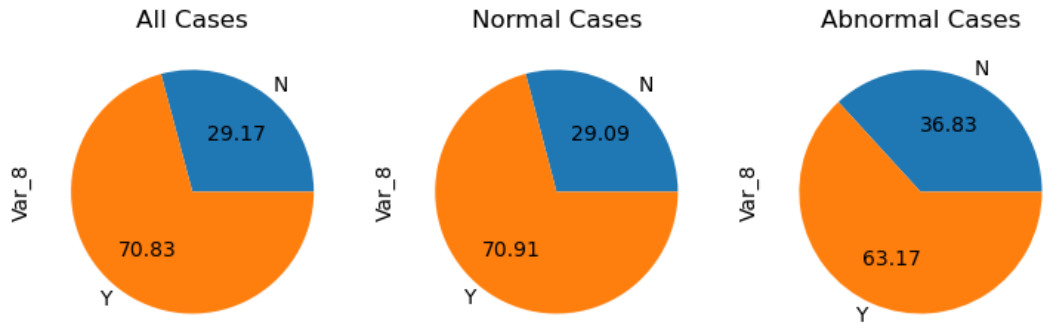

Figure 4.3 : The frequency distribution of *Var_8* variable with respect to target classes

Var_9 is a binary variable showing if the main branch and application branch codes are the same ("Y") or not ("N"). By intuition, it can be thought that when the application and main branches of a customer are the same, it may indicate a normal case. There are no missing values in this variable. Figure 4.4 shows that there is a slight difference between the distributions of *Var_9* in normal and abnormal cases. It can be said that if someone applies for a credit from a non-main branch, it is a higher possibility to have an abnormal case as expected. There are no missing values in this variable, but one-hot-encoding is needed.

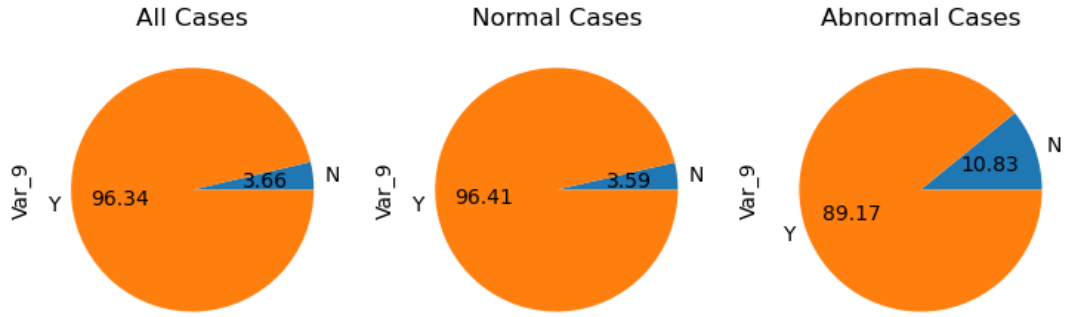

Figure 4.4 : The frequency distribution of *Var_9* variable with respect to target classes

Var_10 is a binary variable showing if the customer is active (“1”) or not (“0”). By intuition, it can be expected that “1” value may indicate a normal case. There are no missing values in this variable. Figure 4.5 shows that there is a significant difference between the distributions of *Var_10* in normal and abnormal cases. It can be said that if someone is an active customer, it is a higher possibility to have a normal case as we have expected. From another angle, it is seen that almost half of the abnormal cases are coming from customers who are not active. *Var_10* needs no preprocessing because it has neither missing values nor categorical levels.

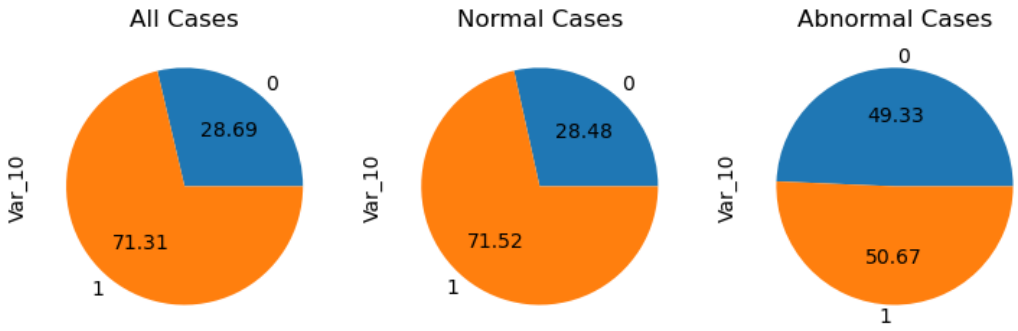

Figure 4.5 : The frequency distribution of *Var_10* variable with respect to target classes

Var_15 variable shows if there is a previous loan disbursement of the customer (“1”) or not (“0”). By intuition, it can be said that “1” value may indicate a normal case. When the frequency distributions of the variable are inspected, it is seen that the normal samples have customers with a previous loan disbursement with a 78.98% while the abnormal samples have 69.83%. There is no need to make missing value imputation or encoding for *Var_15*.

Var_16 feature has been labeled as “1” if the manager of the application branch has a positive opinion about the application, and “0” otherwise. None of the abnormal cases has a positive opinion from the branch manager and all of them are labeled as “0” in *Var_16*. It may be interpreted as a sign of possible overfitting for the model, consequently, the variable has been deleted from the dataset.

Var_17 feature indicates if an instant credit bureau check is made for the application. There is a very small variance for this variable by having only 0.03 % of “1” values and the rest as “0”. To decrease the dimensionality as much as possible and have a cleanear data, this variable has also been dropped.

Var_22 variable shows if the customer of the application has a home phone (“1”) or not (“0”). Having a home phone would probably have a positive impact on the

application. 36.66 % of the normal applications have home phone, while only 31.05 % of the abnormal samples have home phone. There are 496 missing values in this variable. *Var_22* needs both one-hot-encoding and missing value imputation before modeling process.

Var_23 variable shows the deficiency opinion of the branch sales expert for the application. If the sales expert thinks that the application might be deficient, this variable has been filled with "1", otherwise it is filled with "0". It can be said that the sales experts have told their opinions cautiously, because only a very small number of applications have been told to be deficient. 5 % of the fraud applications have been told to be deficient while only 1.31 % of normal ones. This variable does not need any preprocessing steps.

Var_30 variable indicates that the customer has at least one active futures product in the bank. It affects the applications positively in terms of being a member of the normal class. Only 0.05 % of the fraud applications have active futures products in the bank. This variable does not require preprocessing steps.

Var_32 means if the application is in the branch waiting pool or not. However, non of the applications in the dataset are in a branch waiting pool so this variable has been filled with "0" for all the samples. It does not contain any information, consequently, it has been deleted from the dataset.

Var_35 is a binary variable showing if the application branch and home are in the same city. Empirically, it can be said that if someone uses a branch in the same city with his/her city of residence, it is a higher possibility to have a normal case. There are no missing values in this variable, but still one-hot-encoding is needed.

Var_39 variable shows the headquarters disbursement process. If the customer has another disbursement process, it is filled with "0", otherwise with "1". When the frequency distribution of this variable in different classes, it can be seen that the fraudulent applications have a higher percentage of having another disbursement process with the ratios of 7.67 % to 5.14 %. This distribution can be thought of as counterintuitive because having another disbursement process may indicate a positive attitude of the customer by intuition. On the other hand, the simultaneous disbursement processes can be thought of as a preparation for the fraudelent application attacks on the bank. This variable does not require preprocessing steps for modeling.

4.1.2 Categorical variables

In the dataset, there are 8 categorical variables which have at least 3 different levels. The explanations of the categorical variables are shown in the Table 4.2.

Table 4.2 : Categorical variables and their explanations.

Variable	Explanation
Var_2	Customer good-bad flag
Var_21	Worst credit card status
Var_27	Residential status
Var_28	Max delinquency status in last 6 months
Var_31	Current payment status
Var_33	Home address district code
Var_34	Level of education
Var_40	Preapproval credit type

The categorical variables have been explored deeply, and the needs of mappings, imputations and scaling have been detected for the variables that are encoded as categorical levels. All the process applied to the categorical variables is reported one by one in the rest of this chapter.

Var_2 is a categorical variable that shows the state of the customer by his/her payment performance with the available levels of "G", "B", "I", and "X". As Figure 4.6 shows, while the class "G" (good) dominates the normal cases with 56.15 % frequency, there is a more balanced distribution in abnormal cases by the increased percentage of class "B" and the decreased percentage of class "G". Here it can be said that "B" (bad) has a potential to resemble a minority class case while "G" resembles a majority class case. There are 1526 missing values in this feature; consequently, there is a need for both missing value imputation and encoding for *Var_2*.

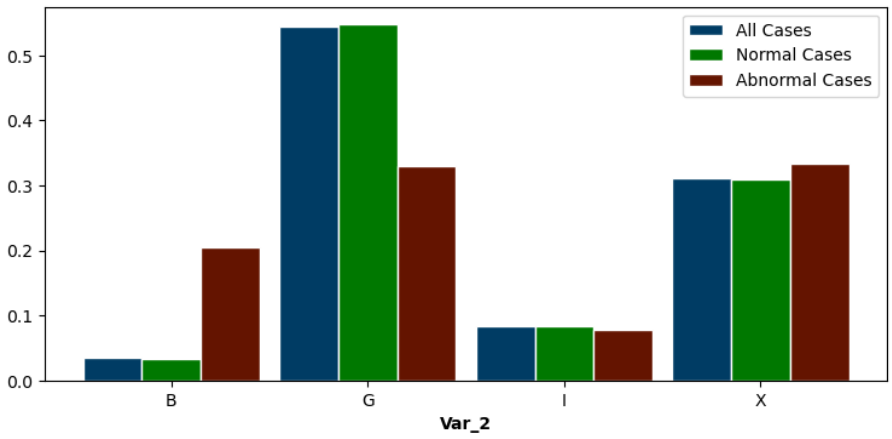

Figure 4.6 : The frequency distribution of *Var_2* variable with respect to target classes.

Var_21 is a categorical variable showing the worst credit card status with levels "G", "N", and "D". There are 20816 null values in this variable. Figure 4.7 shows that "D" class may indicate an abnormal case. The missing value imputation and encoding should be applied for *Var_21*. Here it can be said that "B" has a potential to resemble a minority class case while "G" resembles a majority class case. There are 1960 missing values in this feature; consequently, there is a need for both missing value imputation and encoding for *Var_2*.

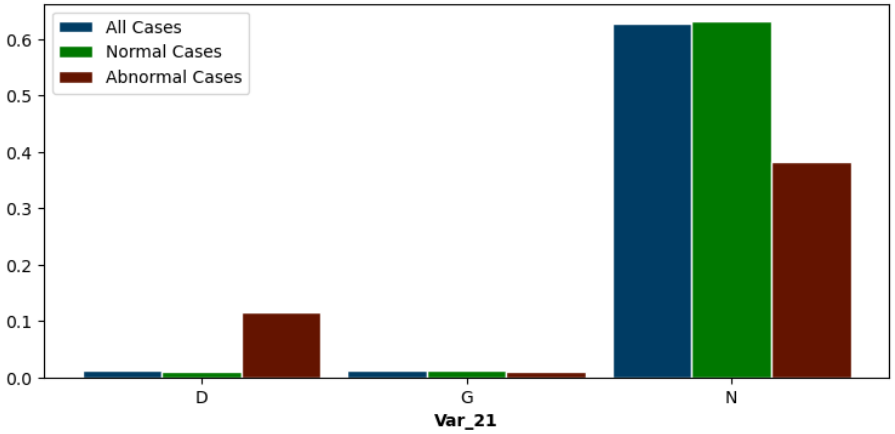

Figure 4.7 : The frequency distribution of *Var_21* variable with respect to target classes.

Var_27 is a categorical variable showing the residential status of the customers with levels "1", "2", "3", "4", and "5". When the frequency distribution of the residential status among normal and abnormal classes, there is not a significant difference that can be detected easily. It can be said that the residential status does not play a very important role in distinguishing between the fraud and normal applications. There are 4487 null values in this variable; consequently, there is a need for both missing value imputation and encoding for *Var_27*.

Var_28 is a categorical variable showing the max delinquency status in the last 6 months with levels "0", "2", "3", and "4". When the frequency distribution of *Var_28* among normal and abnormal classes, the level "0" has been decreased dramatically from 94.21 % to 76.29 % between normal and fraud cases. Thus, the level "0" indicates a higher probability of being a normal application. There are 12724 null values. Missing value imputation and encoding for *Var_28* should be done before modeling process.

Var_31 is a categorical variable showing the current payment status of the customer with levels "0", "1", "U", "2", "3", "4", and "6". Contrary to *Var_28*, in *Var_31*, the level "0" has a negative effect on the application to be a member of the normal

application class. While 87.71 % of the normal application have the “0” value in *Var_31*, the percentage decreases to 94.62 % in fraud applications. There are 28810 missing values in *Var_31*. Missing values and encodings should be handled before the model building process.

Var_33 is a categorical variable which shows the home address district code. As Figure 4.8 shows, there are some districts with a higher share of normal cases and some districts with a higher share of fraud cases. For example, around 7 % of the fraud applications are made from the district with code “16” while only around 4 % of the normal ones are made. On the other hand, almost 12 percent of normal applications have been made in the district code “0” compared to around 9 % share in fraud cases. Moreover, there are some district codes which are only seen in the normal applications such as “32” and “34”. There are 1093 missing values in this feature; consequently, both missing value imputation and encoding for *Var_33* are needed to be done.

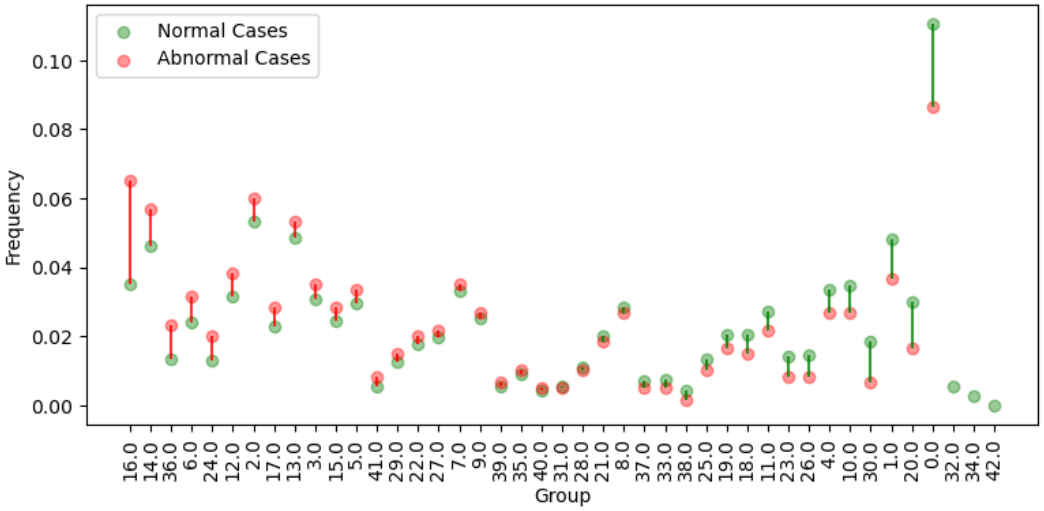

Figure 4.8 : The frequency distribution of *Var_33* variable with respect to target classes.

Var_34 is a categorical variable showing the education level with levels “N”, “I”, ”O”, ”L”, ”U”, and ”Y”. When the frequency distributions are inspected, it is seen that the higher education levels are decreasing the tendency to be a fraud. There are 3418 null values in this variable; so that, the missing value imputation and encoding should be done before modeling process.

Var_40 is a categorical variable showing the preapproval credit type. However, there are only 5 observations with a value for this variable; thus, it is eliminated from the dataset directly.

4.1.3 Numerical variables

In the dataset, there are 18 numerical variables which may be discrete or continuous. The explanations of the numerical variables are shown in the Table 4.3.

Table 4.3 : Numerical variables and their explanations.

Variable	Explanation
Var_3	Total number of credit applications in the last 6 months
Var_4	KKB bureau score
Var_5	Total payments for credit cards
Var_6	Total number of credit applications in the last 1 month
Var_11	Monthly family income
Var_12	Times of overdraft payment delay in the last 24 months
Var_13	Times of overdraft payment delay in the last 12 months
Var_14	Times of credit card payment delay in the last 24 months
Var_18	Credit card total limit
Var_19	Total monthly installments over total assets in the bank
Var_20	Total payments over balance of credit cards
Var_24	Number of unpaid cheques in the last 6 months
Var_25	Number of credit card limit increase rejects in last 3 months
Var_26	Total monthly installments over monthly net income
Var_29	Number of unpaid cheques in the last 7 to 12 months
Var_36	Total balance in litigation
Var_37	Number of unpaid cheques in the last 13 to 24 months
Var_38	Maximum mortgage limit over monthly net income

The numerical variables have been explored deeply, and the needs of imputations and scaling have been detected. All the process applied and comments on the variables are reported one by one in the rest of this chapter.

Var_3 is a discrete numerical variable that shows the total count of the credit applications made in the last 6 months. As illustrated in Figure 4.9, the mean of this variable has a higher value in abnormal cases compared to the normal cases. This situation can be interpreted as that the customers who have a tendency to make fraudulent applications make more applications in a recent period. There are 463 missing values in this variable. Thus, data imputation, and scaling are needed to be applied.

Var_4 is a discrete numerical variable which shows KKB score of the customer who makes the application. KKB, Credit Registry Bureau, is an organization founded by some leading banks in Turkey to serve banks with several products [92]. One of the products that have been utilized by many banks is a scoring system, called KKB score, to assess customers' reliability by considering several features. As illustrated in Figure

4.9, the mean KKB score of abnormal applications is much lower than the mean KKB score of normal applications. It is an expected situation because the higher the KKB score is, the more reliable the customers are. There are 503 missing values in this variable; consequently, data imputation and scaling are needed to be done before model building.

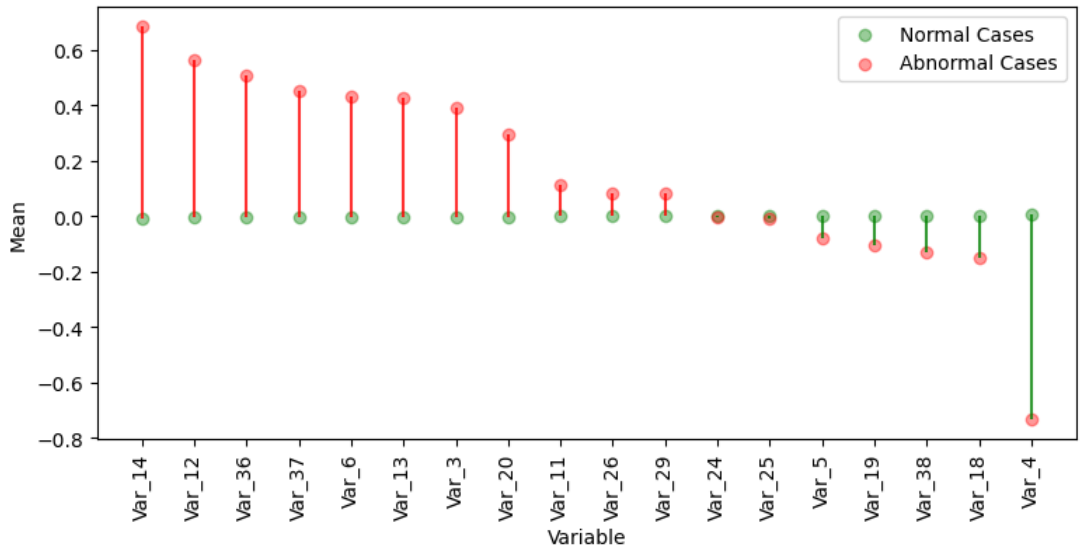

Figure 4.9 : The mean values of the standardized numerical variables with respect to target classes.

Var_5 is a continuous numerical variable showing the total amount of payments of the customer, who made the application, to the credit cards. It can be said that the higher amounts of payments for credit cards increase the chance of being a normal applicant. 503 of the rows have missing values for this variable; consequently, data imputation, and scaling are needed to be done before model building.

Var_6 is a discrete numerical variable that shows the total number of the credit applications such as loan, credit card etc. of the customer in the last month. As shown in Figure 4.9, the customers who have made the fraud applications have applied for any kind of credit more times compared to the normal applications. It is another version of *Var_3*, and there are 463 missing values in this variable. Data imputation and scaling are needed for this variable.

Var_11 is a continuous numerical variable showing the monthly net income of the applicant. It is a variable that is created by the declaration of the applicant, so that it may affect the target variable counterintuitively. There are 496 missing values for this variable which means it needs to be imputed before modeling.

Var_12, *Var_13,* and *Var_14* are similar discrete numerical variables that are related to the delays of payments of different products in different periods. *Var_12* means the total count of the customer is being late on credit overdraft payments in the last 24 months. *Var_13* is the same as *Var_12* except the period is the last 12 months. Finally, *Var_14* shows the count of credit card payment delays of the customer in the last 24 months. As expected by intuition, high values for all of the 3 variables indicate a higher chance of being fraud, as shown in Figure 4.9, because it means that the applicant has struggled with the earlier payment recently. There are many missing values for these variables; consequently, missing value imputation and scaling processes should be applied to them.

Var_18 is a continuous numerical variable showing the total limit of all credit cards of the customer. If the total limit is high, the customer is seen as more reliable. There are 503 missing values in this variable which gives rise to a need for data imputation and scaling.

Var_19 and *Var_26* are similar continuous variables that are created by the amount of total monthly installments. *Var_19* is created by dividing the total monthly installments of the customer by his/her total amount of assets in the bank while *Var_26* is created by dividing it by the total monthly net income of him/her. Both variables have missing values and they needed to be preprocessed.

Var_20 is a continuous variable which is created by the ratio of total payments over the balance of credit cards. There are 405 missing values for *Var_20*; therefore, missing value imputation and scaling processes must be applied to it before the modeling process begins.

Var_24, *Var_29* and *Var_37* are similar discrete numerical variables that are related to unpaid cheques of the customer in different periods. The only difference between the variables is their periods. *Var_24* shows the number of unpaid cheques in the last 6 months while *Var_29* shows it in the period between 7^{th} and 12 months and *Var_37* shows it in the period between 13^{th} and 24 months. As shown in Figure 4.9, higher values for all the 3 variables indicates a higher chance of being a fraud. On the other hand, because all of these 3 variables have very sparse values, a new variable called *Var_41* has been crafted by the sum of them and the original variables have been deleted from the dataset. The new variable's meaning is the number of unpaid cheques

in the last 24 months. There are missing values for *Var_41*; thus, missing value imputation and scaling processes are needed.

Var_25 is the count of rejected credit card limit increase requests of the customer in the last 3 months. There are only 7704 observations that have a value for this variable. Data imputation for is *Var_25* needed before modeling.

Var_36 is a continuous numerical variable that shows the total balance in the litigation. It is naturally expected to have a higher litigation balance in fraud applicants. Figure 4.9 confirms this expectation by showing the higher mean balance in litigation in fraud applications compared to the normal ones. 389 of samples have missing value for *Var_36*. Missing value imputation is needed for this variable.

Var_38 is a continuous numerical variable which shows the maximum mortgage limit amount over monthly net income. It is also a variable which shows the reliability of the customer. There are missing values for *Var_38*; in consequence, all the preprocessing steps should be applied to this variable.

4.2 Missing Value Imputation

After inspecting the variables one by one and detecting the preprocessing needs for each of them, the second step of data preparation is to handle with missing values. All the variables that need missing value imputation are already specified; however, the imputation strategy has been decided in this section. There are several types of missing values such as missing completely at random, missing at random and missing not at random. In this study, there are examples of missing at random and missing not at random types of missingness. For the case of missing at random, a null indicator has been used to conserve the knowledge of the missingness. On the other hand, for the cases of missing not at random, mode/median and "0" values have been imputed according to the reason for the missingness.

First of all, in the feature analysis section, 4 variables, which are *Var_16*, *Var_17*, *Var_32*, and *Var_40*, have been deleted from the dataset and 36 of the 40 variables remained for the further analysis. From the remaining 36 variables, *Var_8*, *Var_9*, *Var_10*, *Var_15*, *Var_23*, *Var_30*, *Var_35* and *Var_39* do not have any missing values; consequently, they are not the subject of this section.

For *Var_2*, *Var_21*, *Var_27*, and *Var_28* variables, the missingness type has been assumed as missing at random; therefore, the missing values have labeled as "Null" to preserve the information from the missingness. For example, when the residential status of a customer (*Var_27*) is missing, it may indicate that the customer has some deficient information in the system. This situation may give some insight about the reliability of the customer.

For the rest of the variables, the missingness type is missing not at random; however, the imputation method differs among 3 different groups. For the first group of variables, which are *Var_12*, *Var_13*, *Var_14*, *Var_25*, *Var_31*, *Var_36* and *Var_41*, the missing values have been replaced with "0" value. For example, when the number of credit card payments delays in the last 24 months (*Var_14*) has a missing value, it indicates that the customer does not have a credit card; thus, the customer does not have any delays. From the remaining variables that still have missing values, the categorical ones, which are *Var_1*, *Var_7*, *Var_22*, *Var_33* and *Var_34*, have been imputed with the mode value. On the other hand, the numerical ones, which are *Var_3*, *Var_4*, *Var_5*, *Var_6*, *Var_11*, *Var_18*, *Var_19*, *Var_20*, *Var_26* and *Var_38*, have been imputed with the median value.

4.3 Category Encoding

For the binary features *Var_1*, *Var_7*, *Var_8*, *Var_9*, *Var_22* and *Var_35*, which are detected in the feature analysis section, and for the other categorical variables, there is a need for encoding to use them in the model building. In binary features, the levels are "Yes" and "No" for all variables. They are one-hot encoded by converting "Yes" values to "1" and "No" values to "0" for all binary variables. On the other hand, the categorical variables are encoded by using a technique called target encoding to both keeps as much as the information from the original variables and keep the dimensionality of the dataset lower. Target encoding basically replaces each categorical level of the variable with its mean target value; thus, it conserves the information of the original variable efficiently. After the encoding process, the dataset consists of 36 variables that are all represented as numerical values.

4.4 Normalization and Scaling

Data normalization and scaling is a very crucial preprocessing step for some of the machine learning algorithms such as logistic regression and artificial neural networks while some algorithms such as tree-based learning algorithms can handle the unscaled data. In this study, the dataset has been normalized and scaled because it is used as a benchmark dataset for various machine learning algorithms that may or may not require standardization. First of all, log normalization for the variables *Var_3*, *Var_5*, *Var_6*, *Var_11*, *Var_12*, *Var_13*, *Var_14*, *Var_18*, *Var_19*, *Var_20*, *Var_26*, *Var_36*, and *Var_38* have been applied. Because these features are related to the monetary values and counts, they are likely to have right-skewed distributions naturally. For example, the original distribution of *Var_3* variable, which shows the total number of credit applications in the last 6 months, can be seen on the left-hand side of the Figure 4.10. As can be seen, the distribution is highly right-skewed originally and it is normalized very well after the logarithmic transformation.

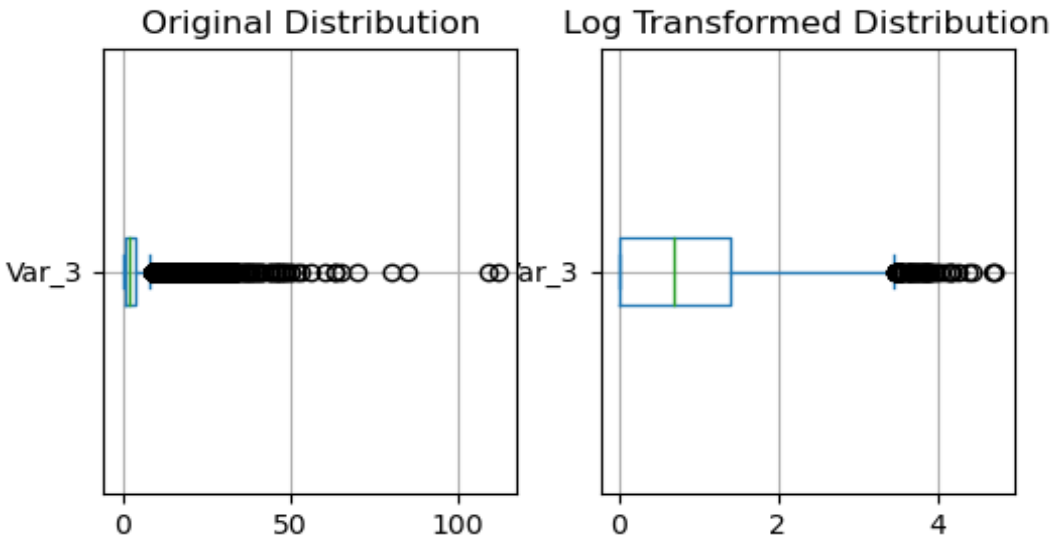

Figure 4.10 : Distribution of *Var_3* before and after the logarithmic transformation.

The existence of outliers in *Var_3* can also be seen from the Figure 4.10. There are many variables with outliers in the dataset, however, the outlier handling methods have not been applied because the outliers may contain crucial information about the anomalies in the dataset.

After normalization, standardization has been applied for all variables to have 0 mean and 1 standard deviation for all of them. Standardization, or scaling, is a very crucial step for the parametric machine learning approaches because it helps to use many

different measurement units, such as count and monetary value, together effectively. After the standardization, there are 34 variables ready for the correlation analysis.

4.5 Correlation Analysis

In this stage, it is checked if there are any highly correlated variables in the dataset. As Figure 4.11 illustrates, there is only 1 problematic relationship between variables in terms of correlation.

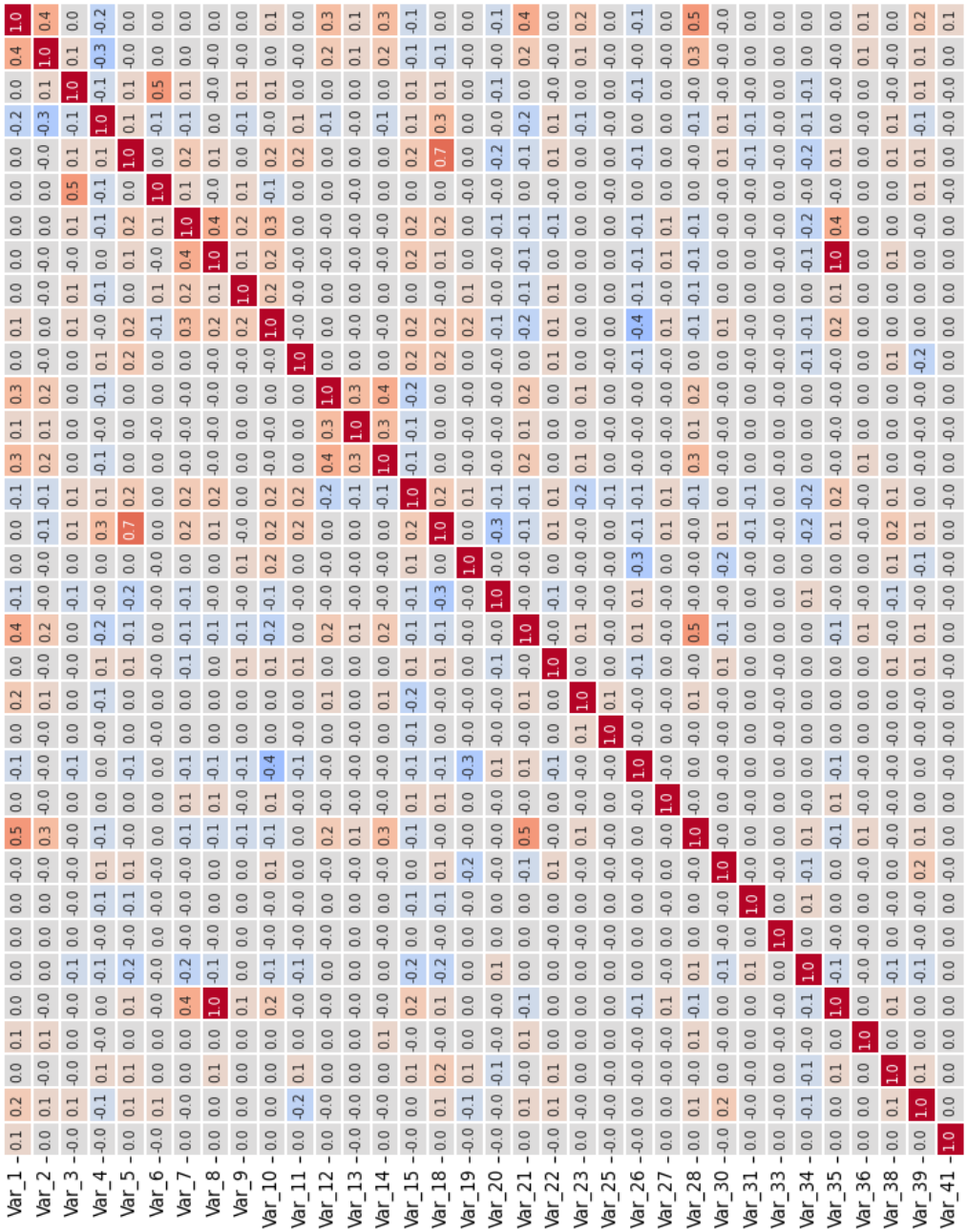

Figure 4.11 : Correlation matrix of the preprocessed variables

The correlation between the variable *Var_8* and *Var_35* equals to 1 which means that the variables are identical. Because the variables check if branch and home are in the same city (*Var_35*) and branch and workplace are in the same (*Var_8*), this situation means that for each customer the workplace and home addresses are in the same city. As a result, *Var_35* have been eliminated from the dataset and the data has become ready to use in the model building with 33 variables that are in a numerical format and scaled.

5. MODEL BUILDING AND COMPARISON

After preparing the dataset by processes of missing value imputation, encoding, normalization and standardization, the model building stage of the study has started. In this section, several supervised, unsupervised and reinforcement learning based anomaly detection models have been built, optimized, and compared to each other by several metrics. In the rest of this chapter, the model building processes, and results of the models have been reported and compared.

5.1 Unsupervised Learning Models

Among a big variety of the unsupervised learning models which might be utilized in the anomaly detection problems, ABOD, PCA, KNN, IF and AE methods have been selected to represent the probabilistic, linear, ensemble, proximity and neural networks based methods. For all of the models, the basic process of model building has been kept the same for benchmarking. First of all, all the models have been trained using the default settings on the training split of the data. Secondly, the hyperparameter search for all of the models has been applied using the validation set approach. In other words, the parameter setting which provides the best performance on the validation set has been selected for each model. For the hyperparameter optimization, F1 score has been used, while the other performance metrics are also considered for model interpretation.

As can be seen on Table 5.1, the model performance of ABOD method with the default parameters is not very promising compared to the other default models. However, in the ABOD model, the model should consider all the training samples to calculate the variation of each sample. It creates a really high time complexity which causes infeasibility for the model building and prediction processes. In the default setting, the model considers the angles between the nearest 30 samples (n_neighbors) for each data point while calculating the variance of the angles. For the hyperparameter search, 30, 40 and 50 values for “n_neighbors” parameter have been tried. The best F1 score performance of 0.069 has been achieved by the model which takes the “n_neighbors”

parameter as 40. Increasing this parameter would result in better performing models; however, the time complexity also increases, and the model starts to become more and more unfeasible.

Table 5.1 : The performance metrics of default and hyperparameter tuned unsupervised models on the validation set.

Model	Version	Precision	Recall	F1_Score	Lift
ABOD	Default	0.069	0.066	0.068	6.7
PCA	Default	0.114	0.107	0.110	10.7
KNN	Default	0.074	0.066	0.070	6.7
IF	Default	0.118	0.143	0.112	10.7
AE	Default	0.112	0.106	0.110	10.7
ABOD	HPTuned	0.073	0.066	0.069	6.7
PCA	HPTuned	0.114	0.107	0.110	10.7
KNN	HPTuned	0.101	0.093	0.097	9.3
IF	HPTuned	0.143	0.120	0.131	12.0
AE	HPTuned	0.123	0.120	0.127	12.0

As shown in Table 5.1, PCA model has achieved a good performance with the default parameters. In the hyperparameter search, the n_components parameter which resembles the number of principal components to keep has been searched. In the default setting all the principal components are kept, and the 0.3 and 0.5 of the total number of components have been searched to see if there is any improvement. As a result, the default parameters have shown the best performance and hyperparameter tuning did not increase the default model performance.

KNN model has achieved a promising baseline performance with the default parameters having an F1 score of 0.069 as can be seen on Table 5.1. In the hyperparameter search, a grid of 3 different hyperparameters has been searched. The first hyperparameter that is considered is “n_neighbors” parameter which means the number of neighbors that are considered by the model while calculating the distance metric. The default value of the parameter is 5 while the optimized model selected it as 50 among the alternative values of 10, 25 and 50. The increase in this parameter may increase the performance as well, however, the time complexity problem, which is observed in ABOD model, has occurred in KNN algorithm also. The second hyperparameter is called “method” which corresponds to the distance calculation method for the algorithm. In the default setting, it is set to “largest” which takes the largest distance among the k nearest neighbors as the distance metric. In the tuned model, the “largest” option is also preferred over the “mean” option, which takes the

mean of the distances with the k nearest neighbors as the distance metric. Finally, the "metric" hyperparameter has been searched among the alternatives of "minkowski", "cosine", and "Manhattan". The default value of "minkowski" has been also selected for the optimized model. The final tuned model has increased its performance to the level of 0.097 in terms of F1 score.

Table 5.1 shows that the IF algorithm has achieved a very good performance with a 0.112 F1 score with the default parameter setting. To increase the performance, the grid-search over the hyperparameters of "n_estimators", "max_features", and "bootstrap" has been applied. "n_estimators" parameter determines the number of trained models in the ensemble. In the default setting it is set to 100, and the tuned model has also selected it as 100 among the alternatives of 200 and 300. "max_features" parameter means the number of randomly selected features for each estimator in the ensemble. From the alternatives of 5, 10, 15, and 25, the optimum number of features to use in each estimator is found as 10 in the grid search. Finally, the parameter called "bootstrap" is used to select the sampled from the dataset for each estimator with or without replacement. In both default and tuned models, "bootstrap" parameter has been chosen as "False" to do sampling without replacement. After the parameter tuning, the final IF model has achieved a 0.131 F1 score as can be seen on the Table 5.1. As the Table 5.1 shows, the tuned IF model has achieved the best performances in terms of the precision and F1 score metrics while the lift and recall scores are achieved as equally good by the AE model.

The final unsupervised learning model trained is the AE model. It is basically an autoencoder model which learns the distribution of the normal samples and detects the abnormal samples by the high reconstruction errors. In the hyperparameter search for the AE model, the model architecture itself is considered as a hyperparameter along with the batch size, number of epochs to train, optimizer, and dropout rate. As Table 5.1 shows, the default version of the AE model has achieved a good performance compared to the other default models with respect to all the performance metrics. After the hyperparameter tuning, the model architecture has been decided as illustrated in the Figure 5.1. The batch size of the tuned model has decreased from 64 to 16, the number of epochs has decreased from 100 to 10 and the dropout layer has been deleted from the model. The optimizer has been selected as the Adam optimizer among the alternatives of stochastic gradient descent and RMSprop optimizers. The performance

of the tuned model is also one of the highest compared to the other tuned models; however, in terms of F1 score and precision, the IF model has achieved slightly better performances compared to the AE model as shown in Table 5.1.

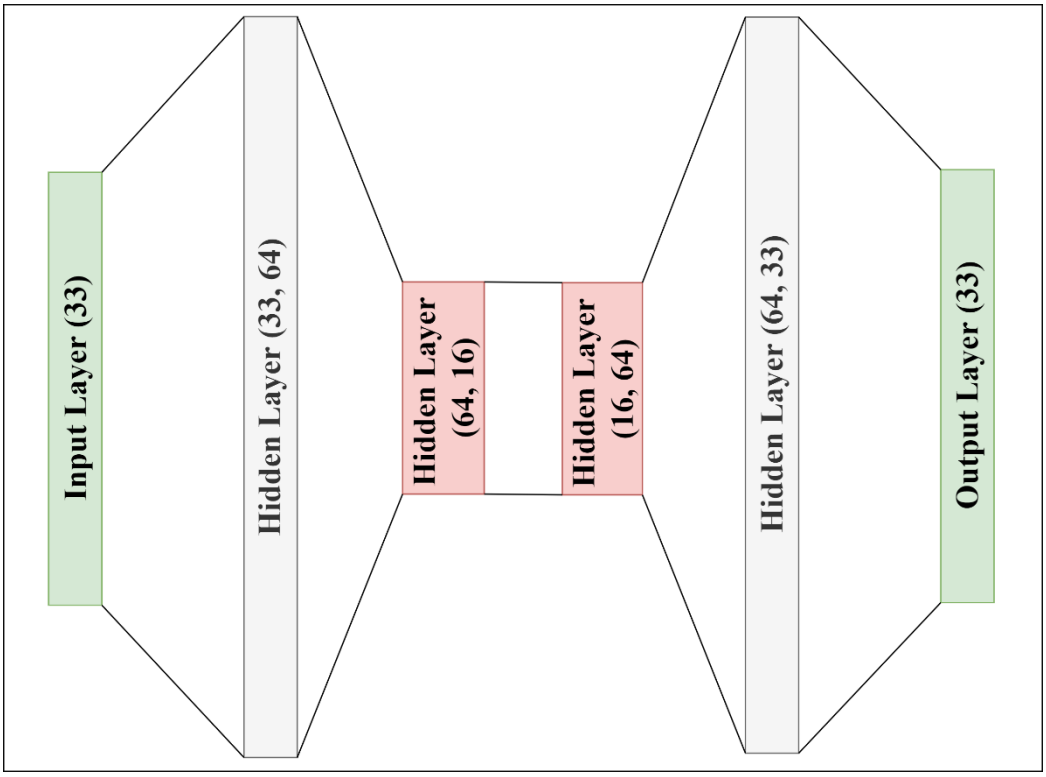

Figure 5.1 : The neural network architecture of the tuned AE model.

As a result of the unsupervised model building process, the best performance has been achieved by the tuned IF model. After selecting the IF model among the unsupervised alternatives, as a final step, the possible generalization failure has been controlled by comparing the PR-AUC metrics of training, validation, and test sets. As can be seen in Figure 5.2, training, validation, and test PR-AUC scores are close to each other, which is a sign of a good generalization.

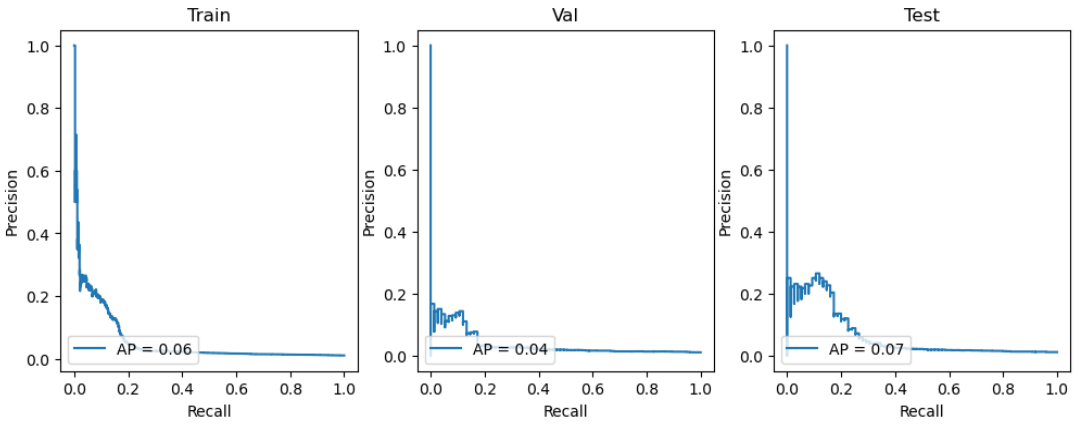

Figure 5.2 : Precision-recall curves of IF model on train, validation, and test sets.

5.2 Supervised Learning Models

As discussed widely in the literature review section, there are numerous algorithms and applications of supervised learning based anomaly detection. LR, DT, SVM, LightGBM, FCNN algorithm has been selected to cover a wide range of supervised learning types from tree-based methods to ensemble learning and neural networks based methods. For each of the models, the same model building process has been applied. Firstly, all of the models have been trained on the original training data using their default parameter settings to create a baseline performance. In the second step of supervised model building process, model parameters are tuned on the original training data by applying grid search with validation set approach. Finally, the data sampling methods, which are RUS, ROS, SMOTE, and SMOTEENN, have been applied to the dataset to find the best performance for each model. In the applications of the resampling methods, the resampling ratio has been treated as a hyperparameter and tuned alongside the model hyperparameters. After the applications of resampling techniques, the best model has been decided by considering the performance metrics listed in the Table 5.2.

Table 5.2 : The performance metrics of default, hyperparameter tuned, and both resampled and hyperparameter tuned versions of supervised models on the validation set.

Model	Version	Precision	Recall	F1_Score	Lift
LR	Default	0.333	0.013	0.026	9.3
DT	Default	0.047	0.053	0.050	0.0
SVM	Default	0.000	0.000	0.000	14.7
LightGBM	Default	0.143	0.027	0.045	13.3
FCNN	Default	0.250	0.013	0.025	16.0
LR	HPTuned	0.031	0.720	0.059	13.3
DT	HPTuned	0.035	0.587	0.066	16.0
SVM	HPTuned	0.029	0.707	0.056	13.3
LightGBM	HPTuned	0.174	0.186	0.186	17.3
FCNN	HPTuned	0.062	0.573	0.112	17.3
LR	Resampled&HPTuned	0.120	0.200	0.150	13.3
DT	Resampled&HPTuned	0.231	0.160	0.189	16.0
SVM	Resampled&HPTuned	0.164	0.160	0.162	16.0
LightGBM	Resampled&HPTuned	0.203	0.213	0.208	18.7
FCNN	Resampled&HPTuned	0.116	0.293	0.166	20.0

As the first supervised learning model, the LR algorithm has been trained with its default parameters on the original training dataset. As can be seen in Table 5.2, the resulting performance of the model is low in all performance metrics except precision.

Precision is high because the model is averse to predicting samples as “abnormal” because of the data imbalance. As the first improvement to the base LR model, the parameters of inverse regularization strength and class weights have been optimized on the original training data using grid search. Among the available levels of 0.01, 0.1, 1, 10, and 100, the tuned model has increased the regularization strength by choosing the 0.01 instead of the default value of 1. Moreover, the tuned model has used the class weights parameter to handle with the class imbalance in contrary to the default model. Table 5.2 shows that these changes in the parameters have increased the F1 score from 0.026 to 0.059; although, it decreased the precision significantly. As the last step of the model building process of the LR model, the RUS, ROS, SMOTE, and SMOTEENN techniques have been applied to the training data to under-sample or over-sample the dataset. All of the resampling methods have been applied with the target ratio levels of 0.01, 0.03, 0.05, 0.07, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, and 0.5; moreover, on each version of the training dataset, model hyperparameters tuned to find the final model setting. As Figure 5.3 shows, the optimal sampling strategy for the final model has been selected as SMOTEENN with a 0.1 target ratio value. On the other hand, the optimum model parameters have been found again with 0.01 inverse regularization strength; however, without weighting the samples. The final model has achieved a much higher F1 score compared to the other versions of LR and achieved a balanced pair of precision and recall metrics.

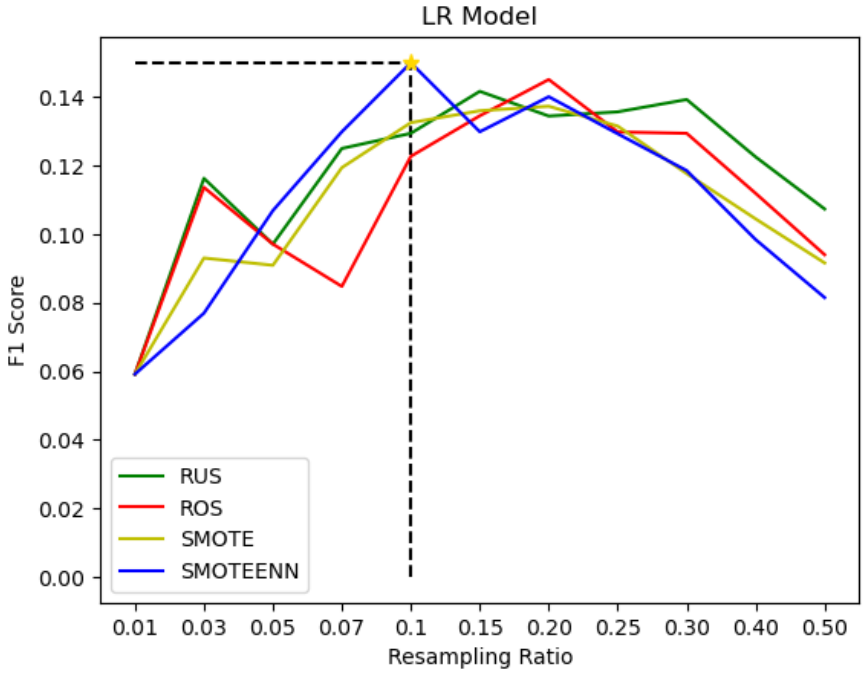

Figure 5.3 : F1 scores of tuned LR models trained on each resampled training set.

The second supervised model is a simple DT model. With the default parameter setting, the DT model is firstly trained on the original training dataset. As Table 5.2 shows, the precision score of the default DT model is very low compared to other alternatives. However, with a high recall, the model achieves a high F1 score. On the other hand, the lift score of the model is 0, which means that the model cannot push the probability of being an anomaly for anomalous samples to the levels close to 1. In other words, the model is not so certain about the predictions of anomalies. After tuning the hyperparameters of the model on the original training data, the model starts to use sample weighting to handle with the class imbalance, changes the split quality measure from Gini to entropy, and limits the maximum depth of the tree with 3. As shown in Table 5.2, the tuned model has increased the F1 score and lift score significantly. Finally, the final DT model has been trained on the over-sampled version of the data using ROS with the ratio of 0.1 as shown in Figure 5.4. In the final model, the sample weighting is not used. The F1 score and lift score metrics have increased significantly both compared to the other algorithms and the other versions of the DT algorithm as reported in Table 5.2.

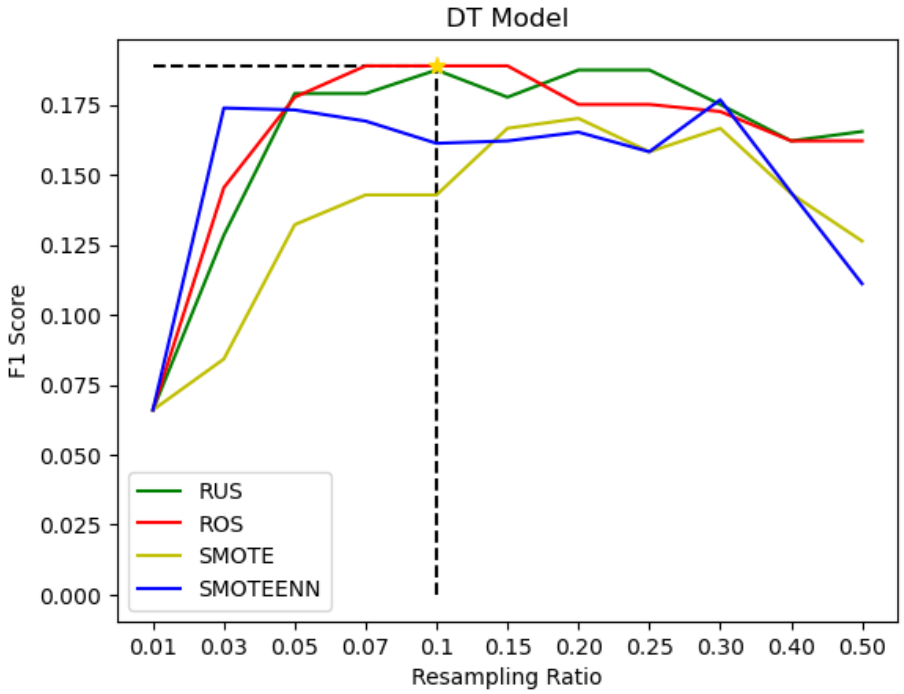

Figure 5.4 : F1 scores of tuned DT models trained on each resampled training set.

Another supervised model used for benchmarking is SVM. The default version of the SVM algorithm has not achieved a good performance in the metrics of precision, recall and F1 score because the imbalanced nature of the dataset has made the algorithm to

make predictions biased towards to the majority class. On the other hand, it has achieved one of the highest lift scores among the algorithms with default parameters. It is a sign that by changing the prediction threshold, a good model can be obtained. After the parameter tuning, the model kernel has been changed to linear, and the samples are weighted according to the target distribution. The tuned model on the original training data has increased precision, recall and F1 score metrics to an average level; however, the lift score has decreased slightly. It is seen on the Table 5.2 that the worst performance among the tuned versions of the model belongs to the SVM model. Finally, the model tuning with the resampling methods has been applied as seen on the Figure 5.5. RUS with a 0.15 sampling ratio has shown the best performance compared to the other methods. When compared to the performance of the other final models, it can be said that the final SVM model has an average performance.

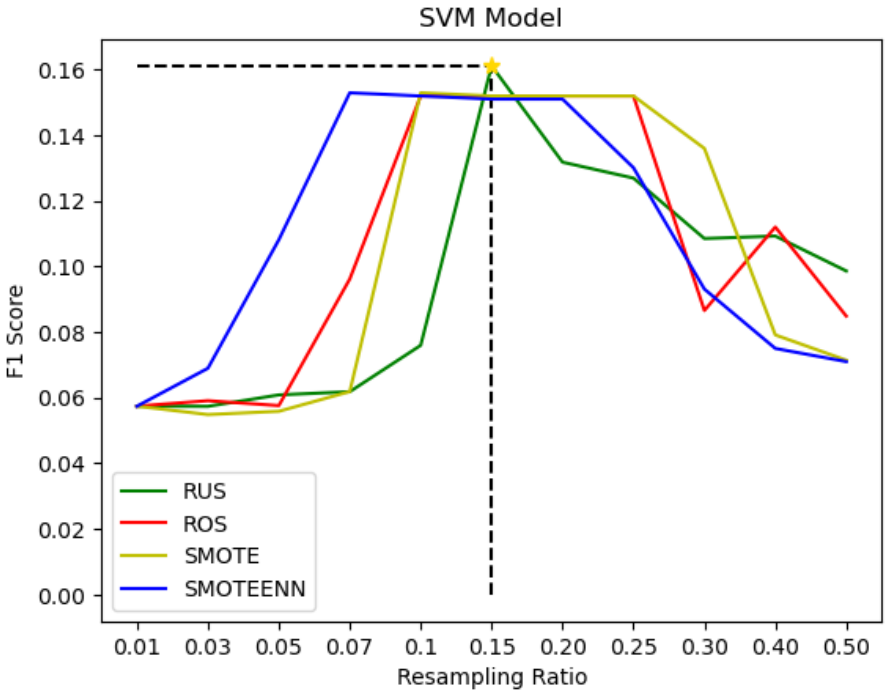

Figure 5.5 : F1 scores of tuned SVM models trained on each resampled training set.

The fourth supervised model built is a LightGBM model trained on the original training dataset with the default parameters. The default LightGBM model has an average performance compared to the other algorithms with the measures of F1 score and lift score. On the other hand, despite of the models such as DT, SVM, and FCNN, it performed well in both of the scores instead of performing well in just one of them. After the parameter tuning using the grids each on validation set method, the learning rate of the model has decreased from 0.1 to 0.01, the maximum depth of the model has

limited to 3, the number of minimum observations needed in one leaf has increased from 20 to 50, alpha and lambda regularizations have been applied and the data weighting has been activated to handle the imbalanced data. The tuned LightGBM model has shown the best performance in lift and F1 scores as can be seen on the Table 5.2. To further increase the performance of the model, resampling methods have been applied as Figure 5.6 shown. Applying the RUS with the ratio of 0.05, the LightGBM method has the final parameter configuration and the best performance achieved. As shown in the Table 5.2, the F1 score has increased over 0.2 and the lift score has increased to 18.7.

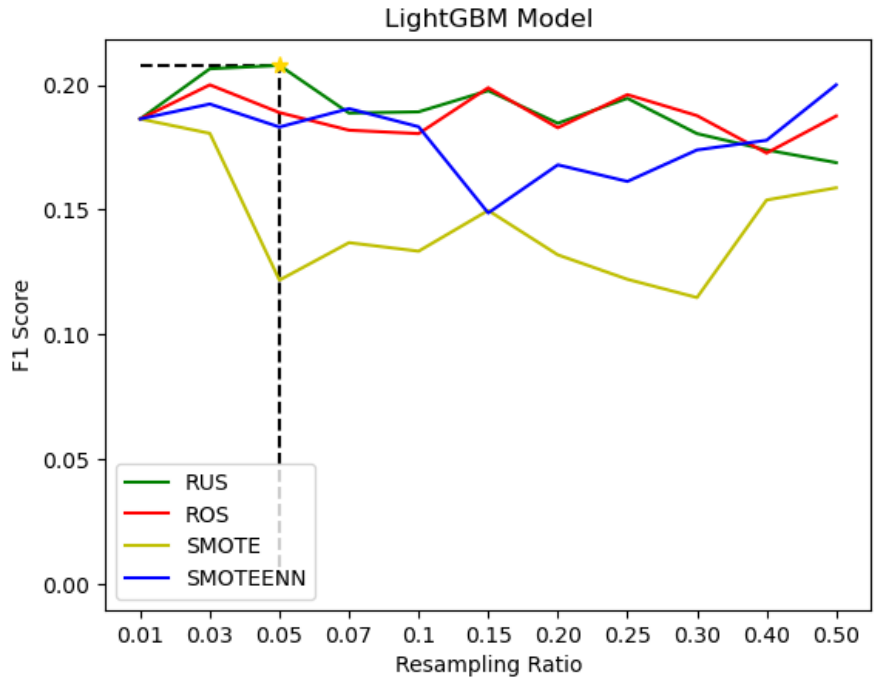

Figure 5.6 : F1 scores of tuned LightGBM models trained on each resampled training set.

The final supervised model trained for the problem FCNN. In the default settings, FCNN algorithm has shown a similar behavior to SVM because it did not make many predictions of anomaly class. On the other hand, the lift score achieved by the FCNN model is the highest among the default models. First of all, the model architecture has been decided as Figure 5.7 with applying grid search over the dropout rates, batch normalization usage, and layer sizes. After tuning the parameters of batch size and number of epochs to train in this architecture, another grid search over the learning rate and sample weighting parameters has been applied. The model has increased its F1 score to the level of 0.112 which is the second-best score after the LightGBM model. On the other hand, the lift score has been the same as the LightGBM model.

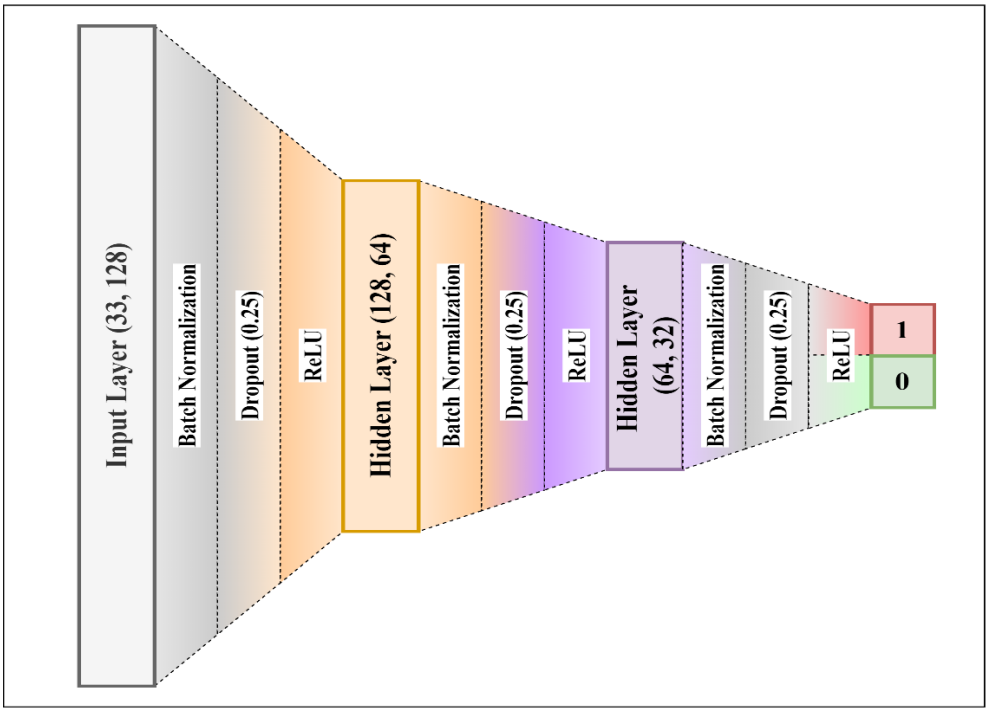

Figure 5.7 : The neural network architecture of the tuned FCNN model.

To obtain the final version of FCNN model, the grid-search over the resampling techniques has been applied as shown in Figure 5.8. The best performing model has resampled the training set with a 0.1 ratio using SMOTEENN method. The model has achieved the best lift score, while achieving an above-average F1 score.

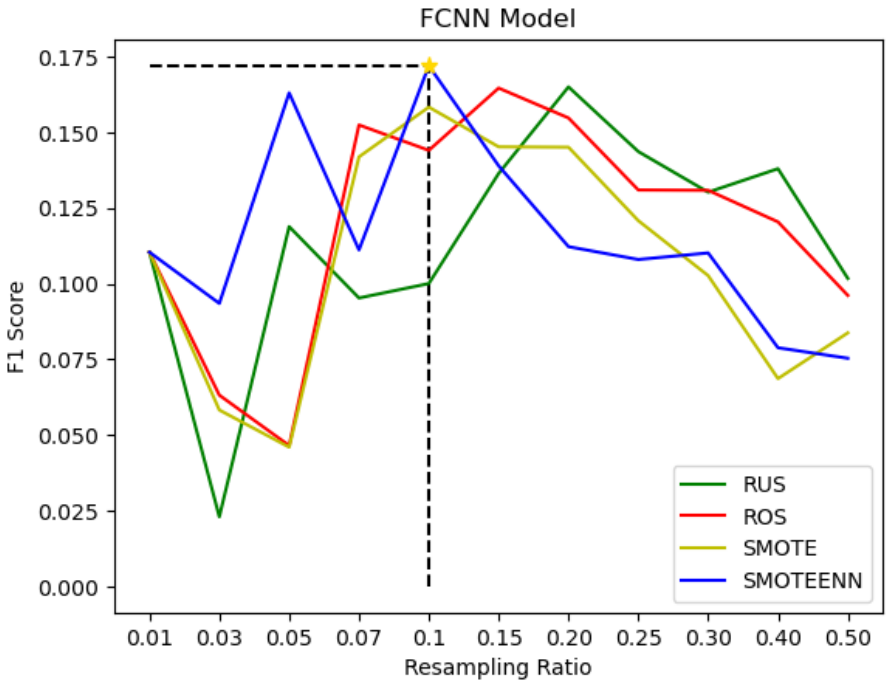

Figure 5.8 : F1 scores of tuned FCNN models trained on each resampled training set.

As one of the results of the supervised model building process, it is seen that the resampling methods increase the model performances effectively. Compared to the algorithm level techniques, such as sample weighting in loss calculation, of handling the target imbalance problem, data level techniques of under-sampling or over-sampling methods have shown a much better performance, especially in terms of the F1 score metric. The best performing resampling ratios have been observed in the levels between 0.05 and 0.15 while the best performing methods vary significantly according to the learning algorithm that is applied.

In general, supervised models are performing better in all of the performance metrics compared to unsupervised models in the resampled data. In the original version of the dataset, the performances of supervised learning models were close the unsupervised learning models.

As the result of the supervised model building process, the best performing model has been selected. Table 5.2 shows that there 2 strong candidates of being the best supervised model. One of the candidates is the final FCNN model with the highest lift score of 20.0, while the other is the final LightGBM model with the best F1 score of 0.208. The final LightGBM model is decided as the best supervised model because of the balance of precision and recall scores. It means that the model has a low amount of both false negative and false positive predictions which are both harmful to the performance of the bank. After deciding on the LightGBM model over the other supervised learning trials, the possibility of overfitting is controlled by comparing the PR-AUC metrics of training, validation, and test sets. As Figure 5.9 shows, both validation and test PR-AUC scores are close to training, which means that the model does not overfit to the training dataset.

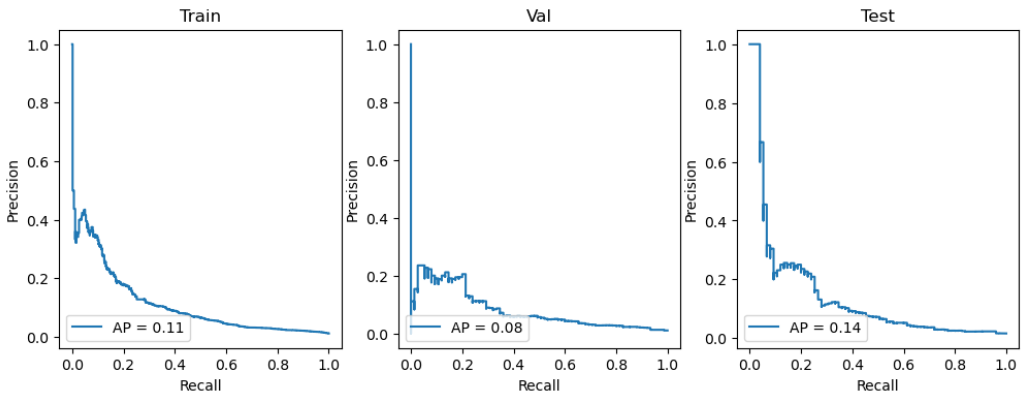

Figure 5.9 : Precision-recall curves of LightGBM model on train, validation, and test sets.

5.3 Reinforcement Learning Model

The final type of anomaly detection method applied in the model building stage is the reinforcement learning based anomaly detection. The basic idea of the approach is to use the reward function of the model to handle with the imbalanced nature of the anomaly detection dataset. A higher reward is earned by the agent if it predicts an anomaly class sample compared to the normal samples. The environment created for the classification task shuffles the training samples and simply shows random samples to the agent one by one. At each sample, there is a reward signal coming from the environment according to the class of the sample and the prediction made by the agent. For majority class samples, true predictions earn a positive reward that is equal to the imbalance ratio (λ) while the false predictions earn a negative reward – λ. On the other hand, the anomaly class samples take 1 as reward for true predictions while taking -1 as the reward for false predictions. Finally, the episodes are terminated if the agent makes a false prediction for an anomaly class sample or if the max number of steps is reached. Having this environment, the imbalanced classification problem can be solved by any reinforcement learning algorithm in the assumption of that the problem is a Markov decision process (MDP) [76].

In this study, the implementation of double DQN in imbDRL library has been applied to solve this MDP [91]. As shown in the Figure 5.10, there are several models trained with different values of λ and showed different performances and characteristics.

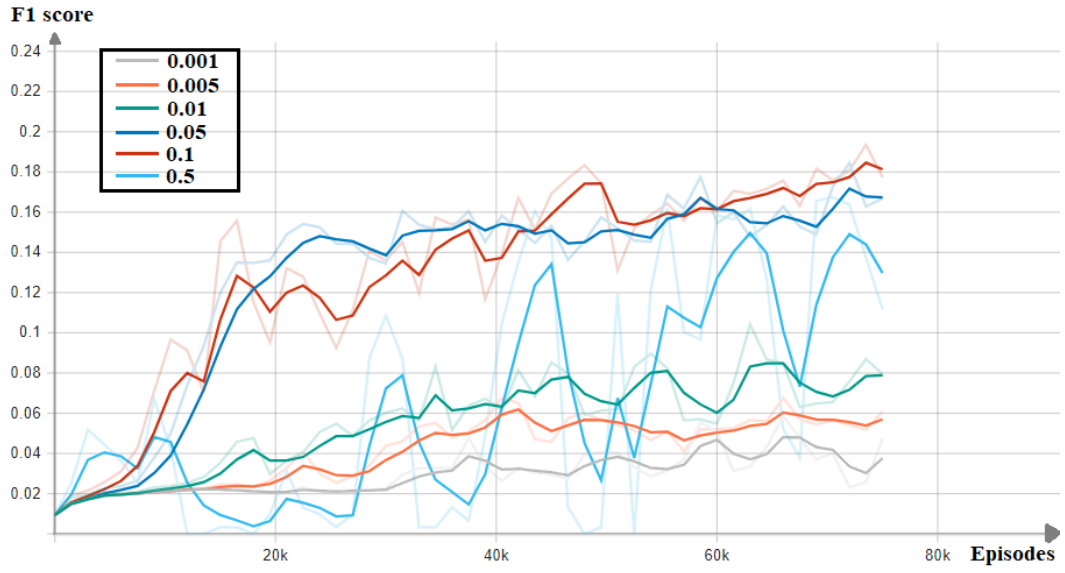

Figure 5.10 : F1 scores with respect to iterations of training the same DQN with different λ values

As Figure 5.10 illustrates, the λ optimum lambda value for this problem is 0.1. When the value of lambda decreases, the stability of learning is protected but the performance has stuck in the lower values. On the other hand, if the λ is chosen as 0.5, the learning curve starts to oscillate too much. After choosing the λ as 0.1, the precision, recall, F1, and lift scores of the model on the validation set is calculated as 0.104, 0.213, 0.140 and 16.0. The performance of the model is satisfying compared to the unsupervised learning models, and the supervised models with sample weighting. As the final step of this stage, the PR-AUC curves of the model have been inspected to see if the model is overfitting to the training dataset. As Figure 5.11 shows, the PR-AUC scores of each data split are very close to each other; thus, it can be said that the model performs a good generalization.

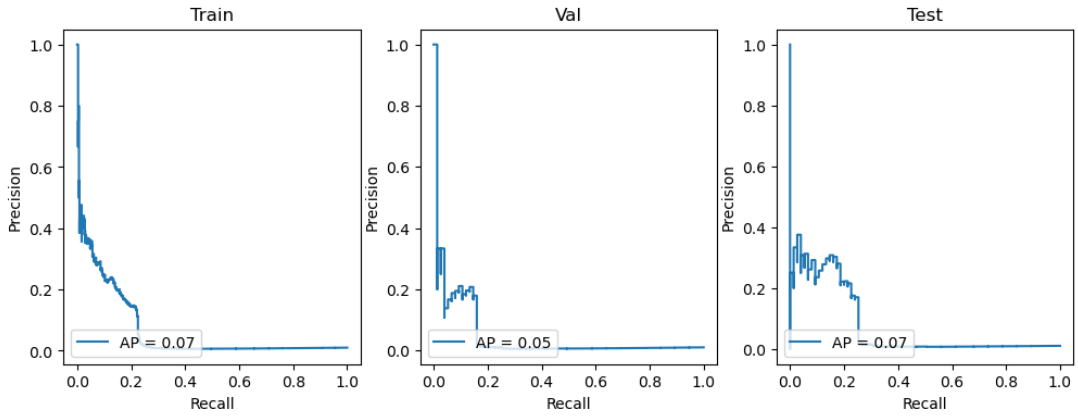

Figure 5.11 : Precision-recall curves of final DQN model on train, validation, and test sets.

5.4 Model Comparison and Final Model

Several machine learning based anomaly detection methods have been applied on the dataset. The best models from all unsupervised, supervised and reinforcement learning approaches are found in the previous sections. As shown in the Table 5.3, IF, LightGBM and DQN methods are the best candidates among the other algorithms from their class. Although the best performing model among the best candidates is the LightGBM model, ensembling approaches are tried to increase its performance using the other models. The first combination of models showed as LightGBMvIFvDQN in Table 5.3 is the weighted average of prediction probabilities of all models with a weight of 0.8 for LightGBM, 0.1 for IF and 0.1, LightGBMvIF is the weighted average of LightGBM and IF with the weights of 0.9 and 0.1, and LightGBMvDQN is the weighted average of LightGBM and DQN with the weights of 0.9 and 0.1. None of

the ensemble methods has increased the performance of the LightGBM model in the validation dataset.

Table 5.3 : The performance metrics best model candidates and ensemble models on the validation set

Model	Precision	Recall	F1_Score	Lift
LightGBM	0.203	0.213	0.208	18.7
DQN	0.104	0.213	0.140	16.0
IF	0.143	0.120	0.131	12.0
LightGBMvIFvDQN	0.182	0.160	0.170	17.3
LightGBMvIF	0.191	0.173	0.182	17.3
LightGBMvDQN	0.185	0.160	0.171	17.3

As a result of the model building process, the best model found is the LightGBM model trained on a randomly under-sampled version of the training data. The performance of the score has been checked after selecting the LightGBM model as the proposed model of the study. The precision, recall, F1 and lift scores are found as 0.202, 0.240, 0.220 and 22.7 which are slightly better than the validation set performance and closer to the training set performances.

5.5 Final Model Interpretation and Business Discussion

The proposed model of the study has been selected after a series of experimentations on the same dataset using several machine learning techniques. It is shown by the performance metrics that the LightGBM model has the best performance on the task of detecting FPD occurrences in the dataset. However, the model has been further interpreted by examining the feature importances and resulting performance to discover the possible insights about the business problem and to understand the possible usage methods. First of all the feature importances and the model interpretation have been shown in the Figure 5.12. The plot has been created using SHAP library to visualize the relationship of the most important features with the target [93].

For example, *Var_10,* which shows if the customer is active or not. As seen on the Figure 5.12, if the customer is active, the probability of being an anomalous application decreases. At the same time, *Var_10* is the most important variable in the model. The second most important variable is the total number of credit applications in the last 6 months, which is shown as *Var_3*. As Figure 5.12 illustrates, higher values for *Var_3* indicate fraud cases. It can be said that the customers who make so many applications

in a recent time period are more probable to cause FPD cases. The third most important feature has been found as the KKB score showed as Var_4. As expected higher KKB scores decrease the probability of being an anomaly case.

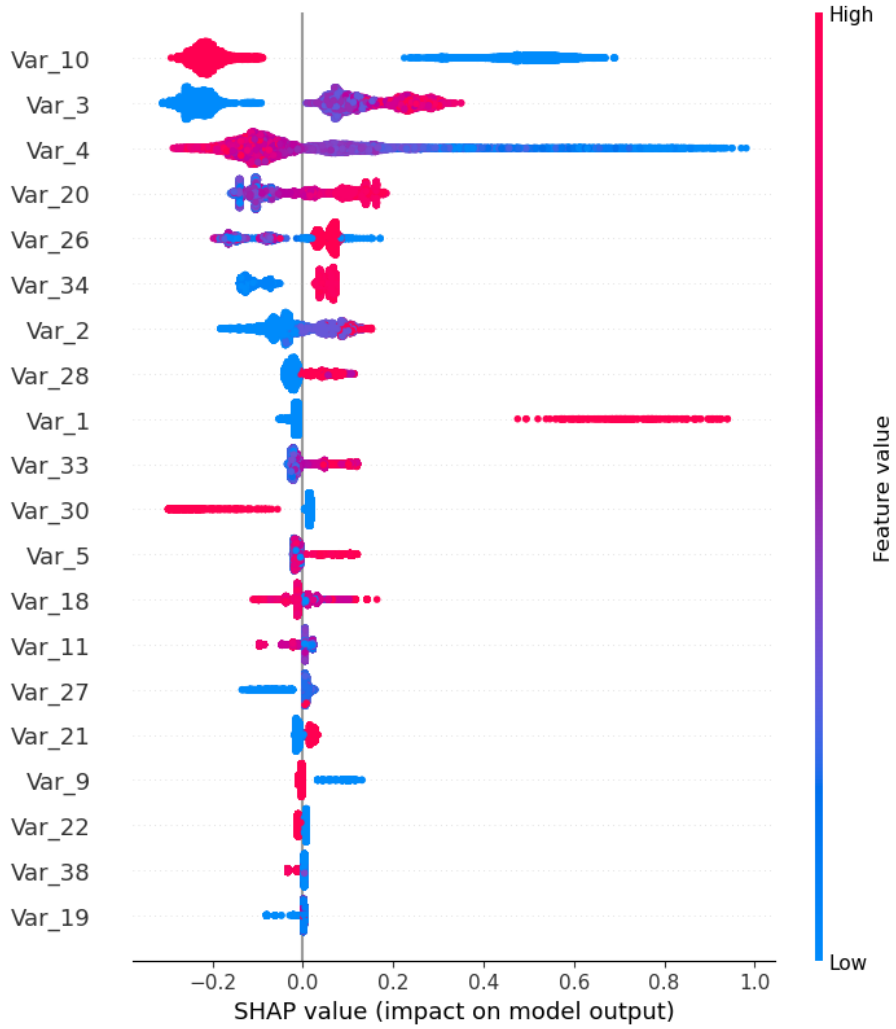

Figure 5.12 : The feature importance of LightGBM model and their relationship with the target variable.

Another example of important variables is *Var_1* which shows if the customer has made a restructured credit request before. It can be seen on the Figure 5.12 that the customers who have not made a restructured credit request have a lower probability of making anomalous applications. On the other hand, as Var_30 shows in Figure 5.12, the active futures products decrease the probability of being an anomaly.

Finally, the potential application and interpretation of the model have been explored by examining the confusion matrix of the model predictions on the test dataset, which is shown in the Figure 5.13. There are 7500 applications 75 of which are anomalies in the test set which have never been seen on any of the model building processes by the

LightGBM model. As figure 5.13 shows, the model has 89 of the 7500 applications as fraud and predicted the rest of the applications as normal ones. Among the 89 anomaly predictions, 71 samples are falsely classified. These 71 applications are rejected by the model which would result in a loss of a potential profit. Because there are no monetary values of the potential profits or losses, the number of false negatives is minimized.

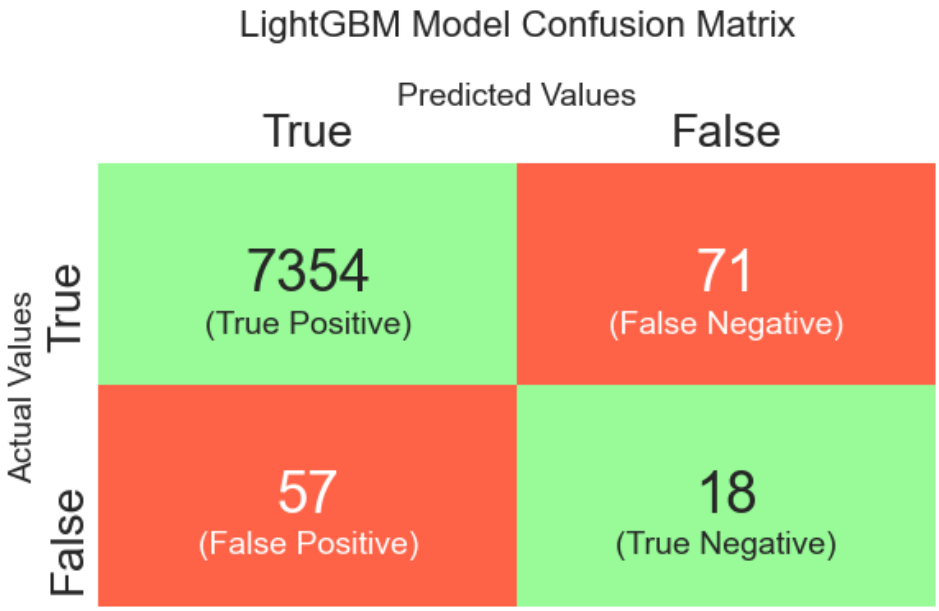

Figure 5.13 : The confusion matrix of LightGBM model predictions on test dataset.

On the other hand, as Figure 5.13 shows, 57 anomalous applications which are classified as normal applications by LightGBM model. These 57 applications are considered as the potential loss for the bank because they are loans that are defaulted in the first payment. The number of false positives is also minimized by optimizing the F1 score. On the other hand, the prediction threshold could have been changed to find a sweet spot if the monetary values of the false positive and false negative samples were given.

As a result, the LightGBM model has a successful performance on the test dataset to detect FPD applications with low error rates. The most important features that it uses while making the predictions are the activeness status, total number of credit usage applications and the KKB score of the customer.

6. CONCLUSION AND DISCUSSION

There are numerous comparative studies in the literature that compares the machine learning based anomaly detection techniques in different domains. Most of the studies have only compared different algorithms from just one group of the unsupervised, supervised or reinforcement learning based methods. However, in this study, a comprehensive literature review of all three types of anomaly detection techniques in different domains and anomaly detection applications in FPD detection studies has been done. After the literature review, 11 different algorithms, which are ABOD, PCA, KNN, IF and AE methods from unsupervised learning, LR, DT, SVM, LightGBM, and FCNN from supervised learning and DQN from reinforcement learning, are selected to apply on the dataset. Moreover, 4 different data resampling methods, which are RUS, ROS, SMOTE, and SMOTEENN, are selected to improve and compared the performances of supervised techniques. As far as the authors' knowledge, this is the most comprehensive FPD detection study which compares a vast range of algorithms and techniques to solve the problem.

In the anomaly detection literature, the best performing models are highly dependent on the domain, and the dataset. In some studies, the unsupervised models are performing better than other models while in some of them they cannot even perform close to the supervised methods. Consequently, the application of a wide variety of models in a novel dataset may result in different insights about the nature of the problem.

In the unsupervised learning based anomaly detection concept, the main idea is to identify the normal class of samples by using the machine learning model and classifying the samples that are not recognized by the model as anomalies. This approach has the potential to be effective in detecting the different kinds of anomalies that can occur in the future and do not exist in the dataset. However, they generally perform worse in the high dimensional datasets. In this study, among the 5 unsupervised learning applied to the dataset, IF model has the best performance with a 0.131 F1 score and 12.0 lift score. It is seen that the dataset used in this study is not

very compatible with the assumptions of the unsupervised learning based anomaly detection methods as they do perform not very well.

The main idea of supervised learning based anomaly detection concept is to identify the normal and abnormal samples in the dataset at hand with a good generalization to detect future anomalies. However, for example in the loan default prediction domain, the attackers may change their strategy to bypass the security systems. Thus, the dataset should contain a wide range of different kinds of possible anomaly examples to create an effective model for the future changes in the domain nature. Otherwise, the model may overfit to the anomalies in the dataset and perform worse in the future. In this study, the best performing supervised model is the LightGBM model trained on a randomly under-sampled version of the training data with a 0.208 F1 score and an 18.7 lift score. Showing a similar performance on the unseen test set with a 0.220 F1 score and a 22.7 lift score, the model has a promising generalization for deployment.

The resampling techniques applied in the process of the supervised model building have increased the performances of the models significantly. It can be said that the data level adjustments for handling data imbalance are found more effective compared to the algorithm level methods in this study. Moreover, it is found out that the selected algorithm highly affects the best method of resampling method among RUS, ROS, SMOTE, and SMOTEENN. Finally, although there is not a single point of resampling ratio for each experiment to find the best performance, generally the ratios between 0.05 and 0.15 are generally the best options.

Finally, the reinforcement learning based anomaly detection method has been applied in this study. The basic idea of the method is to handle the data imbalance using a reward function that rewards and penalizes the true or false predictions of anomaly samples. In this study, the method performed better than unsupervised methods and worse than supervised learning methods with the 0.1 λ value. The model achieved a 0.140 F1 score and a 16.0 lift score. The advantage of having a reinforcement learning model for this kind of a problem is that the model may learn and explore some previously unknown states and adapt to benefit from them in an online manner. On the other hand, the assumption that the classification task is an MDP can be discussed further. Because in the classification environment proposed for this method transitions from the current state to the next state in a completely random manner independently from the action taken.

As the final step, the ensembling by weighted average techniques has been applied to the best 3 models. It can be assumed that the different advantages of different models may increase the performance of the LightGBM. After the trials, it is seen that the LightGBM model performs better than the ensemble models in terms of both F1 and lift scores. As a result, the LightGBM model has been proposed by the study as the best-performing model.

6.1 Further Studies

In this study, the FPD detection topic has been studied by comparing various techniques in the same dataset. However, as can be seen in the literature, the dataset itself has a very important role in the performances of different models. A further comprehensive study can be conducted by applying all the methods in many different datasets from different domains. By this approach, a better understanding of the model performances in anomaly detection can be gained. Moreover, the dataset utilized in this study is a small dataset which includes samples from a short period of time. The period of the dataset can be extended to reach more samples and novel examples of fraud cases. Finally, the dataset might have other potentially important features such as the occupation of the applicant or the city of residence of the applicant to improve the model performances. Another open area for further research can be in the algorithm selection strategy. In this study, the algorithms have been chosen to involve as many types of algorithms as possible, however, a more diverse set of algorithms can be selected for benchmarking.

Many studies in the literature use unsupervised learning methods to extract features for the supervised models and improve the performance. For example, the latent space of an autoencoder, resulting clusters of a clustering method, or the anomaly scores of any unsupervised learning based algorithms can be used as features in supervised learning applications to experiment if they improve the performance or not. Another methodological improvement can be done by applying feature selection for each algorithm itself. It could improve the performances of especially the unsupervised learning models because of diminishing the dimensionality.

In the model interpretation and business discussion section, the analysis could have been done on the monetary values. The potential profit of the bank could be maximized

by changing the prediction threshold if the monetary values of the credit usage amounts and loan rates have existed.

Finally, in this study, some of the models have not been able to be used with their full performance because of the computational and time constraints. For example, in the ABOD method, if all the other samples in the dataset have been considered in the variance calculation, it would last many hours to train the model. With a parallelized training approach in a high-capacity server, the parameter grids searched in the study for each of the algorithms can be extended and better results might be obtained.

In conclusion, although this study presents an extensive experimentation of machine learning based anomaly detection techniques to assess their performances in the FPD detection domain, there are open fields of research related to the data quality, scope of the study, methodology, and computational availability.

REFERENCES

[1] **Hassani, H., Huang, X., Silva, E., & Ghodsi, M.** (2020). Deep learning and implementations in banking. *Annals of Data Science*, *7*(3), 433–446. doi:10.1007/s40745-020-00300-1.

[2] **Alpaydin, E.** (2004). Introduction to Machine Learning. The MIT Press.

[3] **Alhazmi, A. H., & Aljehane, N.** (2020). A survey of Credit Card Fraud Detection Use Machine Learning. 2020 *International Conference on Computing and Information Technology (ICCIT-1441)*, 1-6. doi:10.1109/iccit-144147971.2020.9213809.

[4] **Dastile, X., Celik, T., & Potsane, M.** (2020). Statistical and Machine Learning Models in credit scoring: A systematic literature survey. *Applied Soft Computing, 91*. doi:10.1016/j.asoc.2020.106263.

[5] **Raiter, O.** (2021). Segmentation of Bank Consumers for Artificial Intelligence Marketing. *International Journal of Contemporary Financial Issues*, *1*(1), 39–54, doi:10.17613/q0h8-m266.

[6] **Kuzmin, G., Panov, A. I., Razvorotnev, I., & Rezyapkin, V.** (2021). Application of deep reinforcement learning methods in debt collection. *Lecture Notes in Networks and Systems*, 66-77. doi:10.1007/978-3-030-87178-9_7.

[7] anomaly. 2022. In Merriam-Webster.com. Retrieved March 3, 2022, from https://www.merriam-webster.com/dictionary/anomaly

[8] **Khatri, S., Arora, A., & Agrawal, A. P.** (2020). Supervised Machine Learning Algorithms for Credit Card Fraud Detection: A Comparison. *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. doi:10.1109/confluence47617.2020.9057851.

[9] **Rai, A. K., & Dwivedi, R. K.** (2020). Fraud detection in credit card data using unsupervised machine learning based scheme. *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 421-426. doi:10.1109/icesc48915.2020.9155615.

[10] **Zhinin-Vera, L., Chang, O., Valencia-Ramos, R., Velastegui, R., Pilliza, G., & Socasi, F.** (2020). Q-credit card fraud detector for imbalanced classification using Reinforcement Learning. *Proceedings of the 12th International Conference on Agents and Artificial Intelligence*. doi:10.5220/0009156102790286.

[11] **Koç, U., & Sevgili, T.** (2020). Consumer loans' first payment default detection: a predictive model. *Turkish Journal of Electrical Engineering & Computer Sciences, 28* (1), 167-181. doi:10.3906/elk-1809-190.

[12] **Russell, S. J., & Norvig, P.** (1995). Artificial intelligence: A modern approach. Englewood Cliffs, N.J: Prentice Hall.

[13] **Goodfellow, I., Bengio, Y., & Courville, A.** (2017). Deep learning. MIT Press.

[14] **O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G. V., Krpalkova, L., Riordan, D., & Walsh, J.** (2019). Deep learning vs. Traditional Computer Vision. *Advances in Intelligent Systems and Computing*, 128–144. doi:10.1007/978-3-030-17795-9_10.

[15] **Andrew, N.G.** (2019). Machine Learning Yearning: Technical Strategy for AI Engineers, In the Era of Deep Learning.

[16] **Chen, T., & Guestrin, C.** (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. doi:10.1145/2939672.2939785.

[17] **Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.** (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *NIPS*.

[18] **Shwartz-Ziv, R., & Armon, A.** (2022). Tabular data: Deep Learning is not all you need. *Information Fusion*, *81*, 84–90. doi:10.1016/j.inffus.2021.11.011.

[19] **Kang, M., & Jameson, N. J.** (2018). Machine Learning: Fundamentals. *Prognostics and Health Management of Electronics*, 85–109. doi:10.1002/9781119515326.ch4.

[20] **James, G., Witten, D., Hastie, T., & Tibshirani, R.** (2017). An introduction to statistical learning. Springer.

[21] **Zhu, X., & Goldberg, A. B.** (2009). Introduction to Semi-Supervised Learning. Synthesis *Lectures on Artificial Intelligence and Machine Learning*, *3*(1), 1–130. doi:10.2200/s00196ed1v01y200906aim006.

[22] **Sutton, R. S., & Barto, A. G.** (2018). Reinforcement learning: An introduction. MIT press.

[23] **Chandola, V., Banerjee, A., & Kumar, V.** (2009). Anomaly Detection: A Survey. *ACM Computing Surveys*, *41*(3), 1–58. doi:10.1145/1541880.1541882.

[24] **Aggarwal C. C.** (2017). Outlier Detection. Springer.

[25] **Chu, X., Ilyas, I. F., Krishnan, S., & Wang, J.** (2016). Data Cleaning. *Proceedings of the 2016 International Conference on Management of Data*. doi:10.1145/2882903.2912574.

[26] **Leite, B., Abdalrahman, A., Castro, J., Frade, J., Moreira, J., & Soares, C.** (2021). Novelty detection in physical activity. *Proceedings of the 13th International Conference on Agents and Artificial Intelligence*. doi:10.5220/0010254908590865.

[27] **Parkar, P., & Bilimoria, A.** (2021). A survey on cyber security IDS using ML methods. *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*. doi:10.1109/iciccs51141.2021.9432210.

[28] **Makani, R., & Reddy, B. V. R.** (2018). Taxonomy of machine leaning based anomaly detection and its suitability. *Procedia Computer Science*, *132*, 1842–1849. doi:10.1016/j.procs.2018.05.133.

[29] **Muruti, G., Rahim, F. A., & bin Ibrahim, Z.-A.** (2018). A survey on anomalies detection techniques and measurement methods. *2018 IEEE Conference on Application, Information and Network Security (AINS)*. doi:10.1109/ains.2018.8631436.

[30] **Adikaram, K. K., Hussein, M. A., Effenberger, M., & Becker, T.** (2015). Data transformation technique to improve the outlier detection power of Grubbs' test for data expected to follow linear relation. *Journal of Applied Mathematics*, 1–9. doi:10.1155/2015/708948.

[31] **Mehrotra, K. G., Mohan, C. K., & Huang, H.** (2019). Anomaly Detection Principles and Algorithms. Springer.

[32] **Gogoi, P., Borah, B., & Bhattacharyya, D. K.** (2010). Anomaly detection analysis of Intrusion Data Using supervised & unsupervised approach. *Journal of Convergence Information Technology*, *5*(1), 95–110. doi:10.4156/jcit.vol5.issue1.11.

[33] **Falcão, F., Zoppi, T., Silva, C. B., Santos, A., Fonseca, B., Ceccarelli, A., & Bondavalli, A.** (2019). Quantitative comparison of unsupervised anomaly detection algorithms for intrusion detection. *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. doi:10.1145/3297280.3297314.

[34] **Zhao, Y., Nasrullah, Z. & Li, Z.** (2019). PyOD: A Python Toolbox for Scalable *Outlier Detection. Journal of machine learning research (JMLR), 20*(96), 1-7.

[35] **Kriegel, H.-P., Schubert, M., & Zimek, A**. (2008). Angle-based outlier detection in high-dimensional data. *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 08*, 444–452. doi:10.1145/1401890.1401946.

[36] **Li, Z., Zhao, Y., Botta, N., Ionescu, C., & Hu, X.** (2020). COPOD: Copula-based outlier detection. *2020 IEEE International Conference on Data Mining (ICDM)*. doi:10.1109/icdm50108.2020.00135.

[37] **Janssens, J. H. M.** (2013). Outlier selection and one-class classification. *TiCC PhD Dissertation Series No.27.*

[38] **Radovanovic, M., Nanopoulos, A., & Ivanovic, M.** (2015). Reverse nearest neighbors in unsupervised distance-based outlier detection. *IEEE Transactions on Knowledge and Data Engineering*, *27*(5), 1369–1382. doi:10.1109/tkde.2014.2365790.

[39] **Cunningham, J. P., & Ghahramani, Z.** (2015). Linear dimensionality reduction: Survey, insights, and generalizations. *Journal of Machine Learning Research*, *16*(89), 2859–2900.

[40] **Shyu, M.-L., Chen, S.-C., Sarinnapakorn, K., & Chang, L. W.** (2003). Principal component-based Anomaly Detection Scheme. *Foundations and Novel Approaches in Data Mining*, 311–329. doi:10.1007/11539827_18.

[41] **Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C.** (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, *13*(7), 1443–1471. doi:10.1162/089976601750264965.

[42] **Hardin, J., & Rocke, D. M.** (2004). Outlier detection in the multiple cluster setting using the minimum covariance determinant estimator. *Computational Statistics & Data Analysis*, *44*(4), 625–638. doi:10.1016/s0167-9473(02)00280-3.

[43] **Ullah, I., Mengersen, K., Hyndman, R. J., & McGree, J.** (2021). Detection of cybersecurity attacks through analysis of web browsing activities using principal component analysis. *arXiv preprint arXiv*:2107.12592.

[44] **Hadri, A., Chougdali, K., & Touahni, R.** (2018). A network intrusion detection based on improved nonlinear fuzzy robust PCA. *In 2018 IEEE 5th International Congress on Information Science and Technology (CiSt)*, 636-641. IEEE.

[45] **Ramaswamy, S., Rastogi, R., & Shim, K.** (2000). Efficient algorithms for mining outliers from large data sets. *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 427-438.

[46] **Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J.** (2000). LOF: identifying density-based local outliers. *Proceedings of the 2000 ACM SIGMOD international conference on Management of da*ta, 93-104. doi:10.1145/342009.335388.

[47] **Tang, J., Chen, Z., Fu, A. W., & Cheung, D. W.** (2002). Enhancing Effectiveness of Outlier Detections for Low Density Patterns. *Lecture Notes in Computer Science*, 535–548. doi:10.1007/3-540-47887-6_53.

[48] **Angiulli, F., & Pizzuti, C.** (2002). Fast Outlier Detection in High Dimensional Spaces. *Lecture Notes in Computer Science*, 15–27. doi:10.1007/3-540-45681-3_2.

[49] **Zhu, R., Ji, X., Yu, D., Tan, Z., Zhao, L., Li, J., & Xia, X.** (2020). KNN-Based Approximate Outlier Detection Algorithm Over IoT Streaming Data. *IEEE Access*, *8*, 42749–42759. doi:10.1109/access.2020.2977114.

[50] **Liu, F. T., Ting, K. M., & Zhou, Z.-H.** (2008). Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining*. doi:10.1109/icdm.2008.17.

[51] **Lazarevic, A., & Kumar, V.** (2005). Feature bagging for outlier detection. *Proceeding of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining - KDD '05*. doi:10.1145/1081870.1081891.

[52] **Pevný, T.** (2015). Loda: Lightweight on-line detector of anomalies. *Machine Learning*, *102*(2), 275–304. doi:10.1007/s10994-015-5521-0.

[53] **Liu, F. T., Ting, K. M., & Zhou, Z.-H.** (2012). Isolation-Based Anomaly Detection. *ACM Transactions on Knowledge Discovery from Data*, *6*(1), 1–39. doi:10.1145/2133360.2133363.

[54] **Susto, G. A., Beghi, A., & McLoone, S.** (2017). Anomaly detection through on-line isolation Forest: An application to plasma etching. *2017 28th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*. doi:10.1109/asmc.2017.7969205.

[55] **Zhong, S., Fu, S., Lin, L., Fu, X., Cui, Z., & Wang, R.** (2019). A novel unsupervised anomaly detection for gas turbine using Isolation Forest. *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*. doi:10.1109/icphm.2019.8819409.

[56] **Bengio, Y., Courville, A., & Vincent, P.** (2013). Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828. doi:10.1109/tpami.2013.50.

[57] **Hawkins, S., He, H., Williams, G., & Baxter, R.** (2002). Outlier Detection Using Replicator Neural Networks. *Lecture Notes in Computer Science*, 170–180. doi:10.1007/3-540-46145-0_17.

[58] **Chen, Z., Yeo, C. K., Lee, B. S., & Lau, C. T.** (2018). Autoencoder-based network anomaly detection. *2018 Wireless Telecommunications Symposium (WTS)*. doi:10.1109/wts.2018.8363930.

[59] **Zhai, S., Cheng, Y., Lu, W., & Zhang, Z.** (2016). Deep Structured Energy Based Models for Anomaly Detection. *ICML*. doi:10.48550/arXiv.1605.07717.

[60] **Ruff, L., Vandermeulen, R. A., Görnitz, N., Binder, A., Müller, E., Müller, K. R., & Kloft, M.** (2019). Deep semi-supervised anomaly detection. *arXiv preprint arXiv*:1906.02694.

[61] **Blanchard, G., Lee, G., & Scott, C.** (2010). Semi-supervised novelty detection. *The Journal of Machine Learning Research*, *11*, 2973-3009.

[62] **Jurafsky, D., & Martin, J. H.** (2009). Speech and Language Processing.

[63] **Palmieri, F.** (2019). Network anomaly detection based on logistic regression of nonlinear chaotic invariants. *Journal of Network and Computer Applications*, *148*. doi:10.1016/j.jnca.2019.102460.

[64] **Noureen, S. S., Bayne, S. B., Shaffer, E., Porschet, D., & Berman, M.** (2019). Anomaly Detection in Cyber-Physical System using Logistic Regression Analysis. *2019 IEEE Texas Power and Energy Conference (TPEC)*, 1-6, doi:10.1109/TPEC.2019.8662186.

[65] **Huč, A., Šalej, J., & Trebar, M.** (2021). Analysis of Machine Learning Algorithms for Anomaly Detection on Edge Devices. *Sensors*, *21*(14), 4946. doi:10.3390/s21144946.

[66] **Muniyandi, A. P., Rajeswari, R., & Rajaram, R.** (2012). Network Anomaly Detection by Cascading K-Means Clustering and C4.5 Decision Tree algorithm. *Procedia Engineering*, *30*, 174-182. doi:10.1016/j.proeng.2012.01.849.

[67] **Garg, R., & Mukherjee, S.** (2022). A comparative study using supervised learning for anomaly detection in network traffic. *Journal of Physics: Conference Series*. doi:10.1088/1742-6596/2161/1/012030.

[68] **Venkatesan, C., Karthigaikumar, P., Paul, A., Satheeskumaran S., & Kumar, R.** (2018). ECG Signal Preprocessing and SVM Classifier-Based Abnormality Detection in Remote Healthcare Applications. *IEEE Access, 6,* 9767-9773, doi:10.1109/ACCESS.2018.2794346.

[69] **Liu, J., Gao, Y., & Hu, F.** (2021). A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM. *Computers & Security, 106,* 102289. doi:10.1016/j.cose.2021.102289.

[70] **Islam, M. K., Hridi, P., Hossain, M. S., & Narman, H. S.** (2020). Network anomaly detection using lightgbm: A gradient boosting classifier. *2020 30th International Telecommunication Networks and Applications Conference (ITNAC),* 1-7. doi:10.1109/ITNAC50341.2020.9315049.

[71] **Kingma, D. P., & Ba, J.** (2014). Adam: A method for stochastic optimization. *arXiv preprint. arXiv*:1412.6980.

[72] **Liu, S., Chen, Z., Pan, M., Zhang, Q., Liu, Z., Wang, S., ... & Wan, C.** (2019). Magnetic anomaly detection based on full connected neural network. *IEEE Access, 7,* 182198-182206. doi:10.1109/ACCESS.2019.2943544.

[73] **Malaiya, R. K., Kwon, D., Kim, J., Suh, S. C., Kim, H., & Kim, I.** (2018). An empirical evaluation of deep learning for network anomaly detection. *2018 International Conference on Computing, Networking and Communications (ICNC),* 893-898. doi:10.1109/ICCNC.2018.8390278.

[74] **Busoniu, L., Babuska, R., De Schutter, B., & Ernst, D.** (2017). Reinforcement learning and dynamic programming using function approximators. CRC press.

[75] **Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M.** (2013). Playing atari with deep reinforcement learning. *arXiv preprint. arXiv*:1312.5602.

[76] **Lin, E., Chen, Q., & Qi, X.** (2020). Deep reinforcement learning for imbalanced classification. *Applied Intelligence*, *50*(8), 2488-2502. doi:10.48550/arXiv.1901.01379.

[77] **Wiering, M. A., Van Hasselt, H., Pietersma, A. D., & Schomaker, L.** (2011). Reinforcement learning algorithms for solving classification problems. *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning,* 91-96. doi:10.1109/ADPRL.2011.5967372.

[78] **Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P.** (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16,* 321-357. doi:10.1613/jair.953.

[79] **Batista, G. E. A. P. A., Prati, R. C., & Monard, M. C.** (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*. *6*(1), 20–29. doi:10.1145/1007730.1007735.

[80] **Aslam, U., Aziz, H. I. T., Sohail, A., & Batcha, N.** (2019). An Empirical Study on Loan Default Prediction Models. *Journal of Computational and Theoretical Nanoscience, 16,* 3483-3488. doi:10.1166/jctn.2019.8312.

[81] **Kruppa, J., Schwarz A., Arminger, & G., Ziegler, A.** (2013). Consumer credit risk: Individual probability estimates using machine learning. *Expert Systems with Applications*, *40*(13), 5125-5131. doi:10.1016/j.eswa.2013.03.019.

[82] **Madaan, M., Kumar, A., Keshri, C., Jain, R., & Nagrath, P.** (2021). Loan default prediction using decision trees and random forest: A comparative study. *IOP Conference Series: Materials Science and Engineering*, *1022*(1). doi:10.1088/1757-899X/1022/1/012042.

[83] **Odegua, R.** (2020). Predicting Bank Loan Default with Extreme Gradient Boosting. *arXiv preprint arXiv*:2002.02011.

[84] **Luo, C., Wu, D., & Wu, D.** (2017). A deep learning approach for credit scoring using credit default swaps. *Engineering Applications of Artificial Intelligence*, *65*, 465-470. doi:10.1016/j.engappai.2016.12.002.

[85] **Addo, P., Guegan, D., & Hassani, B.** (2018). Credit Risk Analysis Using Machine and Deep Learning Models. *Risks*, *6*(2), 38. doi:10.3390/risks6020038.

[86] **Shen, F., Zhao, X., & Kou, G.** (2020). Three-stage reject inference learning framework for credit scoring using unsupervised transfer learning and three-way decision theory. *Decision Support Systems*, *137*, doi:10.1016/j.dss.2020.113366.

[87] **Sun, F., Tang, Y., & Wu, Z.** (2021). Comparison of binary classifier and outlier detection in different equilibrium. *Journal of Electronics and Information Science*, *6*(2), 5-8. doi:10.23977/jeis.2021.060202.

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

[89] **Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., … Chintala, S. (2019).** PyTorch: An Imperative Style, High-Performance Deep Learning Library. *In Advances in Neural Information Processing Systems*, *32*, 8024–8035.

[90] **Tietz, M., Fan, J. T., Nouri, D., … Bossan, B. (2017).** skorch: A scikit-learn compatible neural network library that wraps PyTorch. Retrieved from https://skorch.readthedocs.io/en/stable/

[91] **Berg, T.** (2021). imbDRL, *GitHub repository*, https://github.com/Denbergvanthijs/imbDRL

[92] **Kredi Kayıt Bürosu** (2022). https://www.kkb.com.tr/en

[93] **Lundberg, S., & Lee, S.** (2017). A Unified Approach to Interpreting Model Predictions. *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, 4768–4777

CURRICULUM VITAE

Name Surname **: Ahmet Talha YİĞİT**

Place and Date of Birth **: Istanbul - 09.09.1995**

E-Mail **: ahmetyiit@gmail.com**

EDUCATION **:**

- **B.Sc.** **:** 2019, Istanbul Technical University, Management Faculty, Management Engineering Department

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2019-2021, Data Executive at Ipsos
- 2021-Present, Data Scientist at TEB Arf