ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

BUSINESS CARD AS A BANK PRODUCT AND ESTABLISHMENT OF A NEW BUSINESS CARD TENDENCY MODEL

M.Sc. THESIS

Onur BOZKURT

Department of Industrial Engineering

Industrial Engineering Programme

APRIL 2023

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

BUSINESS CARD AS A BANK PRODUCT AND ESTABLISHMENT OF A NEW BUSINESS CARD TENDENCY MODEL

M.Sc. THESIS

Onur BOZKURT
(507191121)

Department of Industrial Engineering

Industrial Engineering Programme

Thesis Advisor: Assis. Prof. Ömer Faruk BEYCA

APRIL 2023

Onur Bozkurt, a M.Sc. student of İTU Graduate School student ID 507191121, successfully defended the thesis/dissertation entitled “BUSINESS CARD AS A BANK PRODUCT AND ESTABLISHMENT OF A NEW BUSINESS CARD TENDENCY MODEL”, which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assis. Prof. Ömer Faruk BEYCA** Istanbul Technical University
Jury Members :	**Assis. Prof. Mehmet Ali ERGÜN** Istanbul Technical University
Jury Members :	**Assis. Prof. Fuat KOSANOĞLU** Yalova University

Date of Submission : 30 December 2022
Date of Defense : 17 April 2023

FOREWORD

I would like to express my sincere thanks to my valuable colleagues, especially Celil TAŞKIN, for their support and guidance in the realization of this study, my advisor Assis. Prof. Ömer Faruk BEYCA for his valuable advice and guidance, and my family for their moral support throughout the process.

December 2022 Onur BOZKURT
(Industrial Engineer)

TABLE OF CONTENTS

Page

ABBREVIATIONS

ANN	: Artificial Neural Network
BKM	: Interbank City Center (Bankalararası Kart Merkezi in Turkish)
CCRI	: Credit Card Risk Identification
CM	: Confusion Matrix
DBS	: Direct Debiting System (Doğrudan Borçlandırma Sistemi in Turkish)
FFNN	: Feed-Forward Neural Network
GIB	: Turkish Revenue Administration (Gelir İdaresi Başkanlığı in Turkish)
GRU	: Gated Recurrent Unit
IV	: Information Value
KKB	: Credit Reference Agency (Kredi Kayıt Bürosu in Turkish)
LOC	: Local Outlier Factor
LR	: Logistic Regression
LSTM	: Long Short-Term Memory
MAE	: Mean Absolute Error
MRS	: Marginal Rate of Substitution
POS	: Point of Sale
RF	: Random Forest
RFC	: Random Forest Classifier
RMSE	: Root Mean Squared Error
RNN	: Recurrent Neural Network
ROC	: Receiver Operating Characteristic
SME	: Small and Medium-sized Enterprises
SSK	: Social Insurance Institution (Sosyal Sigortalar Kurumu in Turkish)
SVM	: Support Vector Machine
SWIFT	: Society for Worldwide Interbank Financial Telecommunication
WoE	: Weight of Evidence

LIST OF TABLES

Page

LIST OF FIGURES

Page

BUSINESS CARD AS A BANK PRODUCT AND ESTABLISHMENT OF A NEW BUSINESS CARD TENDENCY MODEL

SUMMARY

Specially issued for the commercial needs of SMEs, the business card is equipped with the features of different products that also allow personal use. Within the scope of this product; commercial credit cards, overdraft accounts, and credit products with equal installments or seasonal payments are combined into a single card. However, during the data analysis phase, it will be seen that this product is used by individual customers as well as SMEs. Individual customers referred to here are customers with legal entities called partnership customers.

The business card tendency model currently used for the related product, which has an important place for banks, does not work successfully. The success rate of this model is well below the other models used in the bank, and therefore the accuracy of the data and the model is doubted. In order to improve this situation, the existing model will be observed, deficiencies and errors will be examined, and then a new model will be established. In this process, there will be stages of data preparation, model building, analyzing the output, and evaluating the results.

In the literature, there are many studies prepared for credit cards and related issues by banks and various institutions. Some of these are artificial intelligence-supported credit card models, neural approaches to credit scoring, and calculation of the default rate of loans and fraud detection with the help of machine learning. This study aims to design an end-to-end business card modeling process in the light of other studies in a similar context, but with modern approaches that are not thought to be included in the literature.

In light of the studies in the above-mentioned literature, it can be said that there are quite advanced approaches to credit cards. The first problem here is that security-related models such as default rate and fraud are emphasized instead of sales-oriented models of credit cards. Another problem is that the existing sales-oriented models do not attach the necessary importance to sustainability and are established with a shorter-term profit and success focus.

In general, the studies conducted in the banks related to the subject were investigated, and various similar and different deficiencies were observed in these studies. A similar and common mistake is that the process ends when customers purchase a product that they do not already have. Another problem is the use of customers' information, which is likely to be erroneous and whose accuracy is doubtful, instead of market information with higher validity, such as BKM and GIB. Finally, most models put less emphasis on activity and continuity, and focus on customer balances.

During the case analysis phase, the business card product, which can be tracked through monthly sales at the bank, will be examined. In order to establish a model for this product, first the target definition will be determined, the necessary data will be collected from the relevant tables, and some of them will be selected by filtering these data. Afterward, this data set will be prepared for the model-building phase by making the necessary manipulations on it, and then the model-building phase will be started. Following the model setup, the results will be examined, the most appropriate option will be considered, and success will be measured.

In this study, the number of variables, which was 83 at the beginning, is reduced to 11 during the model-building phase, in accordance with the principle of parsimony. A meaningful result is tried to be obtained by entering these variables into the logistic regression and random forest models.

According to the results obtained, the logistic regression model works with 98% accuracy, while the random forest model works with 99% accuracy. In addition, the precision value obtained in the random forest model is higher than that in the logistic regression model. The precision metric shows how many of the values that are estimated as positive are actually positive.

For these reasons, it is decided that the model to be used should be random forest. In this way, the detection rate of customers with a target definition of 1 for the business card product will be higher, which will increase the bank's customer portfolio and profitability.

It is aimed in this thesis study to eliminate these deficiencies and errors in existing exercises to establish a more beneficial and efficient model for banks. In addition, it is recommended that banks expand their perspectives on which data they will use while establishing the relevant model. At the end of this process, financial institutions will be able to establish healthier models by integrating and using more accurate and consistent market data into their databases, if necessary. This research will provide a successful trend model with meaningful explanatory variables for business card-like products for future works.

BİR BANKA ÜRÜNÜ OLARAK İŞLETME KART VE YENİ BİR İŞLETME KART EĞİLİM MODELİNİN OLUŞTURULMASI

ÖZET

Yurtiçi ve yurtdışında milyonlarca müşterisi bulunan banka, bünyesinde oldukça fazla tablo ve veri bulundurmaktadır. Bu veriler farklı veritabanlarında ve sunucularda tutulduğu için, herhangi bir çalışma kapsamında bu verileri derleyip toparlamak ve bunları kullanarak anlamlı bir sonuca varmak zorlaşmaktadır.

Ayrıca, banka içerisinde oldukça büyük bir veri yığını olduğundan veri girişi sırasında yapılan hatalar ve başka sebeplerden doğan eksik ve sorunlu veri durumu söz konusudur. Bu verilerin doğru bir şekilde manipülasyonunun sağlanması, büyük verinin işlenmesinde en kritik aşamalardan biridir.

Banka ürünleri genel olarak mevduat ürünleri, krediler, ödemeler gibi birkaç gruba ayrılmıştır. Bu çalışmada nakit kart ürünleri kategorisinde yer alan işletme kart ürünü üzerinde durulacak ve bu ürünün özellikleri ile nasıl geliştirilebileceği ele alınacaktır.

KOBİ'lerin ticari ihtiyaçları için özel olarak düzenlenen işletme kart, kişisel kullanıma da izin veren farklı ürünlerin özellikleri ile donatılmıştır. Bu ürün kapsamında ticari kredi kartı, ticari kurtaran hesap, eşit taksitli veya sezon ödemeli kredi ürünleri tek bir kartta birleşir. Ancak, veri incelemesi aşamasında bu ürünün KOBİ'lerle birlikte bireysel müşteriler tarafından da kullandığı görülecektir. Burada bahsedilen bireysel müşteriler, tüzel varlığı bulunan ve şahıs şirketi olarak adlandırılan müşterilerdir.

Bankalar için önemli bir yer tutan işletme kart ürünü için halihazırda kullanılan eğilim modeli başarılı bir şekilde çalışmamaktadır. Bu modelin başarı yüzdesi bankada kullanılan diğer modellerin epey altında kalmakta ve bu nedenle veriler ile modelin doğruluğundan şüphe duyulmaktadır. Bu durumu iyileştirmek adına mevcut model incelenerek eksiklik ve hatalar incelenecek, sonrasında ise yeni model kurulacaktır. Bu süreçte veri hazırlama, model kurma, çıktıyı analiz etme ve sonuçları değerlendirme aşamaları yer alacaktır.

Banka içerisinde yer alan modeller, ilgilendirdiği büyük verinin durumu dikkate alınarak kurulmuştur. Modelin performansına göre karar ağaçlarından yapay sinir ağlarına kadar birçok algoritma denenip, en iyi performansı gösteren öğrenme algoritması kullanılmıştır. Çalışmaların bir kısmı model bazlı iken, diğer kısmı kural bazlıdır. Veri setine en uygun modelin seçilmesi aşamasında karar ağacı, regresyon analizi ve rastgele orman gibi farklı algoritma ve analitik yöntemler uygulanmaktadır.

Literatürde, banka ve çeşitli kurumlar tarafından kredi kartı ve ona yönelik hazırlanmış olan birçok çalışma bulunmaktadır. Bunlardan bazıları yapay zeka destekli kredi kartı modelleri, kredi skorlamaya yönelik nöral yaklaşımlar, makine öğrenmesi yardımıyla

kredilerin temerrüt oranının hesaplanması ve fraud tespiti gibi çalışmalardır. Bu çalışma kapsamında, benzer bağlamdaki diğer çalışmaların ışığında ancak literatürde pek yer almadığı düşünülen modern yaklaşımlarla baştan uca bir işletme kart modeli kurma süreci tasarlanması hedeflenmiştir.

Literatürde var olan çalışmalar ışığında, kredi kartıyla alakalı oldukça gelişmiş yaklaşımların mevcut olduğu söylenebilir. Burada problem, kredi kartına ait satış odaklı modeller yerine temerrüt oranı ve fraud gibi güvenlikle alakalı modellere ağırlık verilmiş olmasıdır. Halihazırda var olan satış odaklı modellerin sürdürülebilirlik konusuna gereken önemi vermeyip daha kısa vadeli kar ve başarı odaklı kurulmuş olmaları da bir diğer problemdir.

Konuyla ilgili bankalarda yapılan çalışmaların geneli incelenmiş olup, bu çalışmalarda çeşitli benzer ve farklı eksiklikler gözlemlenmiştir. Benzer ve yaygın olarak yapılan hatalardan biri müşterilerin halihazırda kendilerinde var olmayan ürünü satın almasıyla sürecin sona erdiği yanılgısıdır. Bir diğer sorun ise müşterilerin BKM ve GIB gibi geçerliliği daha yüksek piyasa bilgileri yerine banka içinde yer alan ve hatalı olma ihtimali olan ve doğruluğu şüpheli bilgilerinin kullanılmasıdır. Son olarak, çoğu model aktiflik ile sürekliliği daha az dikkate alıp müşteri bakiyelerine odaklanmaktadır.

Vaka analizi aşamasında, bankada her ay gerçekleşen satışlar üzerinden izlenebilecek olan işletme kart ürünü incelenecektir. Bu ürüne ait bir model kurabilmek adına önce target tanımı belirlenecek, ilgili tablolardan gerekli veriler toplanacak ve bu veriler filtrelenerek aralarından bir kısmı seçilecektir. Sonrasında bu veri seti, üzerinde gerekli manipülasyonlar yapılarak model kurma aşamasına hazırlanacak ve akabinde model kurma aşamasına geçilecektir. Model kurulumuyla birlikte sonuçlar incelenerek en uygun seçenek değerlendirilecek ve başarı ölçümü yapılacaktır. Bu çalışmayla işletme kart ürünü için başarılı ve anlamlı bir eğilim modeli kurulması hedeflenmektedir.

Bu çalışma sırasında başta 83 olan değişken sayısı, parsimoni ilkesi gereği, model kurma aşamasında 11'e düşürülmüştür. Bu değişkenlerin lojistik regresyon ve rastgele orman modellerine girmesiyle anlamlı bir sonuç elde edilmeye çalışılmıştır. Önce lojistik regresyon modeli için üç farklı yaklaşımla elde edilen sonuçlar kendi içinde değerlendirilmiş, sonrasında bunlar rastgele orman modeli için elde edilen sonuç ile kıyaslanmıştır.

Hesaplanan sonuçlara göre lojistik regresyon modeli %98 doğruluk ile çalışırken, rastgele orman modeli %99 doğruluk ile çalışmaktadır. Ayrıca rastgele orman modelinde elde edilen kesinlik değeri, lojistik regresyon modelindekine göre daha yüksektir. Kesinlik metriği, pozitif olarak tahmin edilen değerlerden kaçının gerçekte pozitif olduğunu gösterir.

Yukarıdaki sebeplerden ötürü, kullanılacak olan modelin rastgele orman olmasına karar verilmiştir. Bu sayede işletme kart ürünü için target tanımı 1 olan müşterilerin tespit edilme oranı daha yüksek olacak, bu da bankanın yüksek derecede önem verdiği müşteri portföyünü ve karlılığı artırma amacına daha uygun olacaktır.

Bu tez çalışması ile birlikte, mevcut uygulamalardaki eksiklik ve hatalar giderilerek bankalar adına daha faydalı ve verimli bir model kurulması hedeflenmektedir. Ayrıca bankaların ilgili modeli kurarken, hangi verileri kullanacakları adına perspektiflerini genişletmeleri önerilmektedir. Bu sürecin sonunda finansal enstitüler, daha doğru ve tutarlı piyasa verilerini, gerekirse kendi veritabanlarına entegre ederek ve kullanarak daha sağlıklı modeller kurabileceklerdir.

1. INTRODUCTION

In the study, the existing model will be observed, deficiencies and errors will be examined, and then a new model giving better and meaningful results will be established. In this process, there will be stages of data preparation, model building, analyzing the output, and evaluating the results.

It is aimed to eliminate these deficiencies and errors to establish a more beneficial and efficient model for banks. In addition, it is recommended that banks expand their perspectives on which data they will use while establishing the relevant model. At the end of this process, banks will be able to establish healthier models by integrating and using more accurate and consistent market data into their databases, if necessary.

1.1 Big Data Applications within the Bank

The bank, which has millions of customers at home and abroad, maintains a large number of tables and data. Since these data are kept in different databases and servers, it becomes difficult to compile and collect them within the scope of any study and reach a meaningful conclusion by using them.

In addition, since there is a very large data pile in the bank, there are incomplete and problematic data situations arising from errors made during data entry and other reasons. Ensuring the correct manipulation of this data is one of the most critical stages in processing big data.

The models within the bank have been established by taking the situation of the big data it concerns into account.

According to the performance of the model, many algorithms from decision trees to artificial neural networks were tried and the learning algorithm with the best performance is used. Some of the studies are model-based, while others are rule-based. Different algorithms and analytical methods such as decision trees, regression analysis, and random forest classifiers are applied during the selection of the most suitable model for the data set.

The emergence of data architecture, innovations, and data science methodologies along with the advent of the new digital world offer financial study an extraordinary opportunity to access data. Leveraging high-dimensional data gathered from BBVA customers' data records during credit or debit card transactions at a Spanish PoS terminal, Garcia et al. (2021) establish an alternative method of assessing retail activity in Spain in a published report. The findings of this study demonstrate that card transactions have a significant influence on retail banking.

A wide range of big data has been incorporated into predicting research as a result of the explosion in internet technologies and computer science, adding new information and strengthening forecasting models. The scientific work by Tang et al. (2022) is the first attempt to perform a comprehensive literature assessment on big data in forecasting research. Big data in forecasting research was categorized by source into user-generated digital information (from social media accounts in texts, images, etc.), device-monitored data (by weather trackers, smart devices, GPS, etc.), and activity log data (for web browsing/visiting, digital marketing, medical treatments, laboratory tests, etc.). Different data sets with unique information and features dominated various forecasting tasks, necessitated various technology for analysis, and enhanced various forecasting models. This study offers a comprehensive evaluation of big data-based forecasting research, describing what big data is (in terms of data types and sources), where it is (forecasting critical points), and how it is used (in terms of analysis and forecasting strategies) to enhance prediction and provide information about perspectives for the future.

2. LITERATURE REVIEW

In the literature, there are many studies prepared for credit cards and related issues by banks and various institutions. Some of these are artificial intelligence-supported credit card models, neural approaches to credit scoring, and calculation of the default rate of loans and fraud detection with the help of machine learning.

Credit card use has surpassed all other forms of payment, which has led to a sharp rise in a fraudulent activity involving credit card payment methods. Therefore, financial institutions are required to consider an automatic preventive system to stop these illegal conducts. Even though several studies have been conducted in this field using conventional statistical and machine learning techniques, the majority of them do not take into consideration the sequential nature of transactional data.

In this context, Forough and Momtazi (2021) suggested an ensemble model based on sequential data modeling utilizing deep recurrent neural networks and a novel voting mechanism based on an artificial neural network to spot fraudulent behavior. An FFNN serves as the voting mechanism in the recommended ensemble model, which combines the output of several recurrent networks used as its base classifier. Either some LSTM or GRU networks are employed for the categorization of recurrent networks.

An LSTM-RNN was suggested by Roseline et al. (2022) to identify instances of fraud. To further boost performance, a concentration mechanism has also been added. Models with this architecture have shown to be exceptionally effective in situations like fraud detection where the information sequence is composed of vectors with tightly linked attributes. In subsequent comparisons, LSTM-RNN is put up against classifiers like naive Bayes, support vector machines, and artificial neural networks.

Utilizing historical transaction data that includes both legitimate and invalid transactions in order to create behavioral features using machine learning algorithms is another approach to this problem. These features can then be used to determine whether a transaction is legitimate or invalid. Two distinct random forest types were created by Xuan et al. (2018) to train the behavior aspects of typical and unusual activities. The performance of the two random forests, which differ in their underlying classifiers, is compared, and their implications for the identification of credit fraud are examined. Data from a Chinese e-commerce company were used in the trials.

One other article on this topic attempts to forecast the possibility of fraud by utilizing a variety of machine learning techniques, including SVM, k-nearest neighbor, and ANN (Asha & Suresh Kumar, 2021). To further distinguish between counterfeit and genuine transactions, supervised machine learning and deep learning techniques are differentiated.

Additional transaction attributes are extracted from core transactional data in Nami and Shajari's (2018) method. These tools help to better understand how cardholders spend their money. These behaviors shift over time, making a cardholder's current conduct more similar to recent behavior. As a result, in the first phase, a new similarity metric is formed based on transaction time. Recent transactions are given more weight in this measure. In the second phase, initial detection is performed using the dynamic random forest method for the first time, and cost-sensitive detection is performed using the minimum risk model.

Another significant study by Shen et al. examines the suitability of decision trees, neural networks, and logistic regression as classification techniques for use in fraud detection (2007).

Analytical server performance in model development is improved by integrating the analytical framework with Hadoop, which can receive data fast and provide it to the analytical server for scam prediction. In their study, Patil et al. (2018) explain a big data analytical infrastructure for processing enormous volumes of data and put into practice different machine learning methods including logistic regression and decision trees for fraud detection. They evaluated the effectiveness of these algorithms using benchmark datasets for real-time fraud detection with minimal risk and maximum customer satisfaction.

Bellotti and Crook (2013) provide discrete-time survival models of defaulting for credit cards that take into account macroeconomic factors over the course of the credit card's lifetime as well as behavioral information about issuers. They discover that when applied to an out-of-sample data set, dynamic models including these behavioral and macroeconomic variables include statistically significant enhancements in model fit, which result in more accurate default predictions at both the account and portfolio levels. They demonstrate how these models may be utilized to stress test credit card portfolios by simulating excessive economic conditions.

Leow and Crook (2016) evaluate exposure at default at the level of the creditor by calculating an account's outstanding balance at any point during the loan duration, using a wide range of chronological findings on defaulted loans. They postulate that the balance due on a credit card account at any point throughout the loan is a result of the borrower's spending and is additionally influenced by the credit limit set by the card provider. The expected amount and limit are weighted and averaged to determine the forecast value, with the weights based on the likelihood that the borrower will have a balance above the limit. A discrete-time repetitive events survival model is used to estimate the weights so as to forecast the likelihood that an account will have a balance higher than its limit.

Machine learning models cannot be employed in delicate applications like identifying medical conditions and terrorism since an absence of explainability and interpretability reduces confidence in the predictions. This has produced several improvements in the understanding of machine learning. Different black-box models are employed by Srinath and Gururaja (2022) to categorize credit card defaulters. A model-agnostic explainer is used to provide interpretations of these models and to assess them using various performance indicators. Finally, the ideal model-explainer combination is suggested along with relevant research directions.

The selection of features is thought to be essential in determining credit card risk in large-scale, high-dimensional data in order to enhance classification performance and the fraud detection process. Random Forest Classifier (RFC), one of the frequently used feature selection techniques, is excellent for huge datasets. The accuracy of the classification of the credit card risk identification model may be significantly improved by RFC's good performance, which strives to specify the most predictive variables. To identify potential fraud, Rtayli and Enneya (2020) suggest an improved Credit Card Risk Identification (CCRI) methodology utilizing the feature selection algorithms RFC and SVM. The suggested technique beats the Local Outlier Factor, Isolation Forest, and Decision Tree in terms of classification performance on a bigger dataset, according to their test results.

Li et al. (2019) utilize Chinese credit card users as their research subjects in an experiment. Through research on the demographics of credit card applicants, consumer behavior data, macroeconomic environment data, and social capital data, they examine the effects of credit card customers' variability, individuality, and social indicators on credit card default. The data on consumer behavior is then divided into online and offline categories in order to determine how the growth of online transactions has affected credit card defaults. They discover that income stability is substantially more important in predicting credit card default than income itself.

The practice of banks discriminating against particular client groups depending on their creditworthiness is known as default discrimination of credit cards. A key component of default discrimination is feature selection, and the characteristics chosen have a direct impact on the outcomes. Hence, the primary issue addressed in this research is the discovery of a collection of ideal attributes to increase the capability of default identification. The uniqueness of the study of Zhou et al. (2022) is the feature combination selection, which is the extension of a single feature selection model based on the F-score and Fisher discriminant analysis F-score. In four credit data sets from Australia, Germany, Taiwan, and Japan as well as six comparison data sets, the combined model's better performance is validated.

In the past few years, among the banking and financial products with the quickest growth has been bank loans. Nevertheless, as more people acquire borrowed funds, banks must deal with an ever-rising percentage of loan decrease. Arora et al. (2022) argue that it is possible to assess the necessity of forecasting loan errors with the assistance of historical data. They believe that machine learning may provide possibilities for dealing with the existing problem and managing credit risk. The algorithm, which glanced at over 10 million Bank of Taiwan entries, has the purpose of predicting when a loan would not be repaid. The correlation between the class variable and a bunch of independent variables is discovered by logistic regression analysis. The initial analysis successfully generates interpretive viewpoints of the data. On the basis of transactional data, this article also employed ML algorithms to obtain accurate projections for identifying the default users.

Ala'raj et al. (2022) aim to assist financial institutions in scoring credit card clients using machine learning by modeling and estimating the behavior of consumers concerning three aspects: the likelihood of single and consecutive missed payments for credit card holders, consumer purchasing patterns, and clustering customers based on a statistical presumption of loss. Based on actual credit card transactional datasets, two models are created and trained. The analysis of customer behavior scores uses traditional performance evaluation metrics. According to experimental findings, consumer credit scoring via neural network techniques has greatly improved when compared to traditional approaches based on feature extraction.

One further research hints experimentally at the scope and characteristics of credit card usage and ownership in Saudi Arabia, as well as how they are influenced by customer demographics and attitudes about borrowing (Abdul-Muhmin & Umar, 2007). Based on the findings of a structured survey, the country has a relatively low credit card penetration rate, women are more likely than men to own the cards, card usage is typically picky, mindset against borrowing is a critical indicator of card ownership though not usage behavior, and cardholders' opinions of the card's features are generally favorable.

Credit has grown in importance for society and people with the growth of the Chinese economy, particularly with the maturation of the market economy. Currently, the credit system is mostly split into two components. An essential component of the social credit system is the enterprise credit system. However, the construction of the individual credit system, which serves as the cornerstone of the social credit system, is also crucial to lowering the cost of data collection and boosting the effectiveness of loan approvals. The data on credit cards is discretized in Wang and Yang's (2020) study at the bank level, and the parameters are chosen by computing the weight of evidence and information value, as well as information divergence, before using logistic regression to make predictions. Finally, a credit scoring model is developed using the findings of the Logistic Regression and displayed credit scores.

Yucelt and MacGregor's (2015) study of credit card usage tendencies among French Canadians and American customers states that two populations exhibit different attitudes toward having and using credit cards. Due to their more conventional attitudes regarding debt, American consumers generally believe that using credit cards allows them to make more impulsive decisions than French Canadians. Additionally, having a credit card in Quebec is said to depend heavily on one’s income.

In retail finance, the balance on a credit card is crucial. In their article, Hon and Bellotti (2016) take multivariate models of credit card balance into account and put the models' predicting abilities to the test using actual credit card data. Ordinary least squares, two-stage regression, and mixture regression are among the models taken into account in the context of cross-sectional regression. Then, they use a random effects panel model to simulate credit card balance using the time series structure of the data. Previous lagged balance is determined to be the most significant predictor variable, although other applications and behavioral variables are also significant. Finally, using each of the suggested models, they give an analysis of forecast accuracy for credit card balances over the next 12 months. The panel model outperforms others in terms of MAE, whereas the two-stage regression model works best in terms of RMSE.

One of the main indicators of a company's riskiness and competence to fulfill its financial commitments is its credit rating. It frequently takes a long time for rating organizations to adjust and offer fresh ratings. Artificial intelligence-based credit score evaluations have so attracted a lot of attention in past years. While revising outdated credit ratings on a regular basis, effective machine learning techniques can quickly analyze credit scores. Support vector machines and neural networks perform better than other methods by making predictions with more precision, according to relevant works.

To forecast credit rating, Golbayani et al. (2020) conduct a review and performance comparison of the literature's findings. Moreover, they utilize the same datasets to implement four machine learning algorithms (bagged decision trees, random forest, SVM, and multilayer perceptron) that are effective in prior research. They use a 10-fold cross-validation approach to assess the outcomes. Decision tree-based approaches perform better, as indicated by the results of the study for the selected datasets.

Based on statistical methods and Neuro Fuzzy, Akkoç's (2012) study suggests a three-step hybrid Adaptive Neuro Fuzzy Inference System credit rating model. The performance of the recommended model was compared to that of traditional and widely used models. A 10-fold cross-validation procedure is used to test the credit score models using credit card information from a worldwide bank with operations in Turkey. Results show that, in terms of average accurate classification rate and expected misclassification penalty, the proposed model consistently outperforms the linear discriminant analysis, logistic regression analysis, and ANN techniques. While the indicated model can learn like an ANN, it does not remain in a black box in contrast to ANN.

Credit card use is crucial for financing both formal and informal SMEs in China, which are generally financially struggling in developing nations, according to Xu et al. (2022). According to research, both categories of SMEs are more prone to use credit cards or to have more credit card debt when their financial situation is poor. Even while formal SMEs have access to banking services, they have large credit card debt. Formal SMEs with borrowed funds had higher credit card debt than those without money needs and bank loans.

Few studies have looked at the impact of credit cards on consumer debt in emerging economies, despite how widely they are used and owned. Based on a Malaysian questionnaire survey, the major goal of the experiment of Ahmed et al. (2010) is to comprehend customers' attitudes toward credit card-related spending habits. A model based on a thorough examination of the literature is created to determine the psychographic elements that affect customer behavior toward the use of credit cards.

The credit card portfolios of the biggest U.S. banks make up a significant portion of their balance sheets. Therefore, controlling the charge-off rates is essential for the credit card industry's profitability. The decision to pay off debt is influenced by several macroeconomic factors. Taghiyeh et al. (2021) in their article provide an expert method for predicting losses in the credit card business using macroeconomic variables. The experts' opinions and comprehensive analysis of the literature on all facets of the economy, customers, industry, and government entities are used to choose which statistics to use. The suggested expert system framework is created using cutting-edge machine-learning techniques.

Whilst competition rises, it is getting more crucial and difficult for numerous credit card merchants to forecast user adoption tendencies effectively. The customer utility function and the associated judgment attitudes need to be better understood in light of this. The frequently accepted logit model assuming a linear utility function and a fixed marginal rate of substitution (MRS) is discussed by Qi and Yang (2003) in their study, using a neural network model that can tolerate a nonlinear utility function and varying MRS amongst card qualities. They discover that the neural network model greatly surpasses the logit in determining customer card acceptance choices using data from a national study on credit card usage.

Digitalization and information-gathering technique developments over the past two decades have led to the rapid and massive production of data on a broad range of subjects. The influence of such data overload on product estimating is examined in the research by Boone et al. (2019), along with how it boosts this estimation. Although time series data are the main focus of this analysis, it also examines how certain data may be utilized to provide information on how customers behave and how that information influences the overall predictions.

In another article, the usage of credit cards is used by Ho et al. as a lens through which to evaluate the long-term correlations between consumer loyalty tendencies and business financial consequences (2021). This research introduces a classification of loyalty behavior by looking at credit card owners' interactions with a particular bank and its rivals. Then, it applies sequence analysis to divide the participants into groups according to their attitudinal characteristics over a period. Six separate customer segments are identified as a consequence of the analysis: Loyalist, defector, switching loyalist, switching defector, dormant loyalist, and dormant defector. Thereafter it examines variations along all segments and calculates the income, maintenance costs, and profitability related to each segment. In addition to discussing the effects of handling multiple credit card client categories, this article also covers how to allocate the bank's advertising resources most effectively.

The categorization of credit card customers at a Portuguese finance company is shown in a publication by Martins and Cardoso (2012). The cross-validation method of evaluating the result is the main emphasis. The suggested method handles a sizable data collection with hybrid (numerical and categorical) variables. In addition to assisting to distinguish the segments, it offers an assessment of the inner consistency of the segmentation methodology. For brand-new credit card customers, it also presents categorization principles. Finally, in order to facilitate current and upcoming strategic choices, this study demonstrates the segment descriptions of credit card users.

Particularly in the competitive and established credit card sector, customer turnover has become a serious problem. To keep consumers and distinguish decent credit customers from poor ones, it is crucial from a financial and risk management standpoint to recognize personal traits. Principles based on customer attributes and churn types of source data have still not been adequately described by researchers.

The study by Lin et al. (2011) first extracts rules linked to customer turnover using the rough set theory, a rule-based decision-making strategy. Next, it infers classification norms and parameters using a flow network graph, a path-dependent method, and then displays the connections between the norms and various types of churn. Additionally, an experimental scenario of credit card customer loss is shown. Throughout this investigation, 21,000 individual observations are gathered and evenly split into three groups: involuntary churn, voluntary churn, and survival. Demographic, psychographic, and transactional data are included in the records from all these observations for assessing and separating user profiles. The findings demonstrate that this combined model is capable of accurately forecasting customer churn and offering helpful data for decision-makers in developing a business plan.

A churn forecasting model is constructed utilizing credit card data that Nie et al. (2011) gathered from a genuine Chinese bank utilizing two data mining techniques, in a different work. Customer, card, risk, and transaction activity information are the four variable groups whose contributions are investigated. The technique for handling variables when data is gathered from a database rather than a questionnaire is examined in this study. The model chooses specific variables based on both correlation and economic logic, as opposed to simply taking all 135 variables into account. In addition to the precision of the analytical findings, the research develops a misclassification cost assessment that is more appropriate for assessing the credit card churn forecasting model by taking into consideration the two types of error and financial logic. Logistic regression and decision trees, two established capable and strong classification methods, are employed in this investigation. Logistic regression outperforms decision trees a little bit more, according to the test results.

In light of the studies in the above-mentioned literature, it can be said that there are quite advanced approaches to credit cards. The first problem here is that security-related models such as default rate and fraud are emphasized instead of sales-oriented models of credit cards. Another problem is that the existing sales-oriented models do not attach the necessary importance to sustainability and are established with a shorter-term profit and success focus.

This study aims to design an end-to-end business card modeling process in the light of other studies in a similar context, but with modern approaches that are not thought to be included in the literature.

3. BANK PRODUCTS

Bank products are divided into several groups such as deposit products, loans, and payments, which are mentioned below.

Current accounts consist of domestic and foreign currency accounts.

Deposit accounts & futures consist of domestic and foreign deposit accounts, stocks, equities and bonds, investment funds, foreign exchange transactions, forwards, options, and swaps.

Cards consist of retail cards, business cards, manufacturer cards, commercial cards, overdraft accounts, and cash advances in installments.

Loans (credits) consist of retail loans, letters of guarantee, letters of credit, agricultural credits, factoring, leasing, mortgage, and consumer loans such as vehicle and gold credits.

Channel-based services consist of call centers, mobile applications, and internet banking.

Payments consist of salary payments, school payments, bill payments, SSK payments, and tax payments.

Insurance consists of personal accident insurance, card protection, project insurance, agricultural insurance, and commercial insurance.

Other products include POS, DBS, import and export letters of credit, and SWIFT.

Since this study will focus on the business card product, mostly cash card products will be discussed.

3.1 Cash Products

Cash products consist of credit and debit cards. While retail and commercial cards can be converted directly into cash, commercial cards and manufacturer cards work by financing a product. Overdraft accounts kick in when the balance drops below zero. In addition to these, the business card product has both direct cash conversion and product financing features.

3.1.1 Business card

Specially issued for the commercial needs of SMEs, the business card is equipped with the features of different products that also allow personal use. Within the scope of this product; commercial credit cards, overdraft accounts, and credit products with equal installments or seasonal payments are combined into a single card. However, during the data analysis phase, it will be seen that this product is used by individual customers as well as SMEs. Individual customers referred to here are customers with legal entities called partnership customers.

4. CASE STUDY – CREATING A BUSINESS CARD MODEL

During the case analysis phase, the business card product, which can be tracked through monthly sales at the bank, will be examined. In order to establish a model for this product, first the target definition will be determined, the necessary data will be collected from the relevant tables, and some of them will be selected by filtering these data. Afterward, this data set will be prepared for the model-building phase by making the necessary manipulations on it, and then the model-building phase will be started. Following the model setup, the results will be examined, the most appropriate option will be considered, and success will be measured. This research aims to provide a successful and meaningful trend model for the business card product for future works..

4.1 Deciding on the Target

Earlier applications in the bank consisted of short-term goals and achievements. From now on, the bank decided on increasing its market share, while retaining its customers and avoiding potential churns. The target definition must be changed to a more long-term structure to succeed.

The general approach in the banking industry is that even if the customer does not use a product with its purchase, that product is flagged as owned. However, this is often abused by employees and customers. A verbal agreement can be made between the employee and the customer, and different purposes can be targeted with product sales causing fraud. As a result, the sales trees of these products may differ from what they should be and the models may deteriorate.

In order to avoid deterioration in this model, it is best to define a target, taking into account the pre-sales and post-sales. In addition, data such as usage data and the financial activity of customers should also be taken into account. Thus, both the accurate measurement of sales and long-term customer continuity will be targeted.

In the previous model, customers who did not have a business card in the observed month, but owned a business card in the following month were set as the target. In the new model to be established, customers who did not have a business card in the past three months, but owned a business card for the following three months, including this month, are determined as the target.

4.2 Collecting Data

SQL queries used while creating the population cannot be shared due to data security and credential reasons. However, these data will be studied in detail in the following sections.

4.3 Filtering and Choosing Data

All data are grouped according to periods and target-based numbers are prepared. Considering the pandemic and periodic effects, the period between 2020-2021 will not be taken into account. In addition, to keep the data more up-to-date and self-consistent, periods as close to the present and only SME customers should be taken into account as possible. In this way, the dataset is narrowed to obtain and apply better results.

After examining the data, which includes the target and non-target numbers of SME customers between October 2021 and September 2022, the periods of November 2021, February 2022, and April 2022 were selected (Figure 3.1 & Figure 3.2). The reason for this is that the numbers in those periods were closer to the average of the quarters they were in, that is, they represented the general population better.

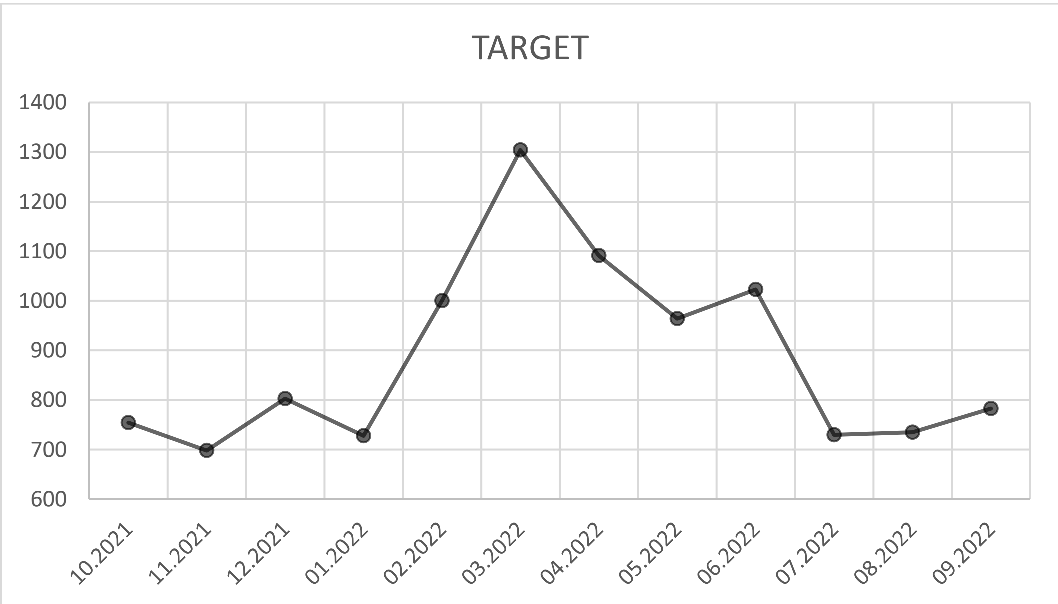

Figure 4.1 : Period-based target numbers for SME customers.

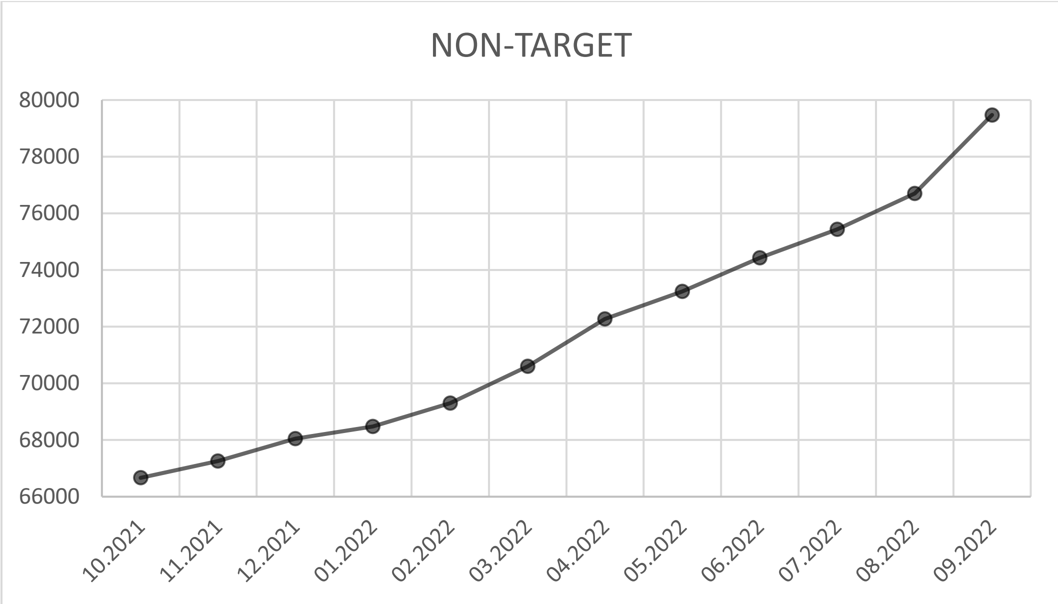

Figure 4.2 : Period-based non-target numbers for SME customers.

Since it is planned to establish a model for SME customers only, in order to ensure that the data set is neither too small nor too large, all SME customers in the three pre-decided periods will be included. The train and test data of these periods are randomly selected by the statistical methods and models to be used.

4.4 Preparing Dataset

First, all possible data about customers and their finances, demographics, and attitudes are selected from the bank's databases and brought together. Some rational variables can be derived using these data.

After all the variables are decided, data cleaning operations are performed using the functions in SQL. Here, for null values, the mean value for normally distributed numeric variables can be printed. For variables with different distributions, the median or mode value can also be used. Variables with a majority of null values can be directly removed from the data set so that they do not adversely affect the model.

After the data cleaning phase, the data is transferred to the python environment for insertion into the model. To measure the success of the model in the following stages, the data set is divided into test and control groups. After stages such as weight of evidence, information value, confusion matrix, and correlation graph, meaningful models are tried to be established and the results are evaluated by comparing them with each other.

4.5 Explaining Variables

The data set contains a huge amount of variables, as will be examined in later stages. While an important part of these variables consists of numerical values, a small part consists of categorical values. All variables will be discussed one by one in the following sections and different operations such as WoE and IV calculations will be performed on them.

4.6 Data Manipulation

First of all, there are 83 variables in the data set, including 82 explanatory and 1 target. Three of these variables were excluded from the data set due to data leakage, as they are thought to be directly related to the target variable.

Data leaking in statistics and machine learning refers to the usage of information during the model training phase that would not be anticipated to be readily accessible at the time of forecast, leading to an overestimation of the inferential ratings (scores) for the models when used in a real-world setting. It is challenging to find and stop leakage since it is frequently inward and oblique. A mathematician or designer may choose a leaky model, which may not function as well as a leakage-free version.

Leakage is simply the insertion of data mining target information that should not be lawfully provided for processing. Although leakage occurs often in both real-world data mining experiments and artificial contests, prior research has mostly ignored this concept. Kaufman et al. offer various techniques for identifying leakage when the designer has no responsibility on how the data were acquired and demonstrate that leaking can be avoided with a basic, specialized method for data handling accompanied with what researchers refer to a learn-predict distinction (2011).

4.6.1 Weight of evidence calculation

Not the volume or size of the evidence, but the credibility or assertiveness of the evidence in terms of standard of proof determines the weight of the evidence. The impact of the evidence on causing belief is what determines its weight rather than any mathematical formula. It has been discovered that the propensity of reliable evidence in a case to favor one side against the other is the key factor in identifying the weight of the evidence.

Since the missing values will be assigned to certain categories with the WoE, there is no need to eliminate them. In this way, missing values will also be overcome with this conversion.

For an incidental variable with WoE transformation, the change of event rate and WoE values according to bin ID of corresponding record are shown below (Figure 3.3 & Figure 3.4). As can be seen in the images, there is an inverse correlation between the calculated event rate and WoE values.

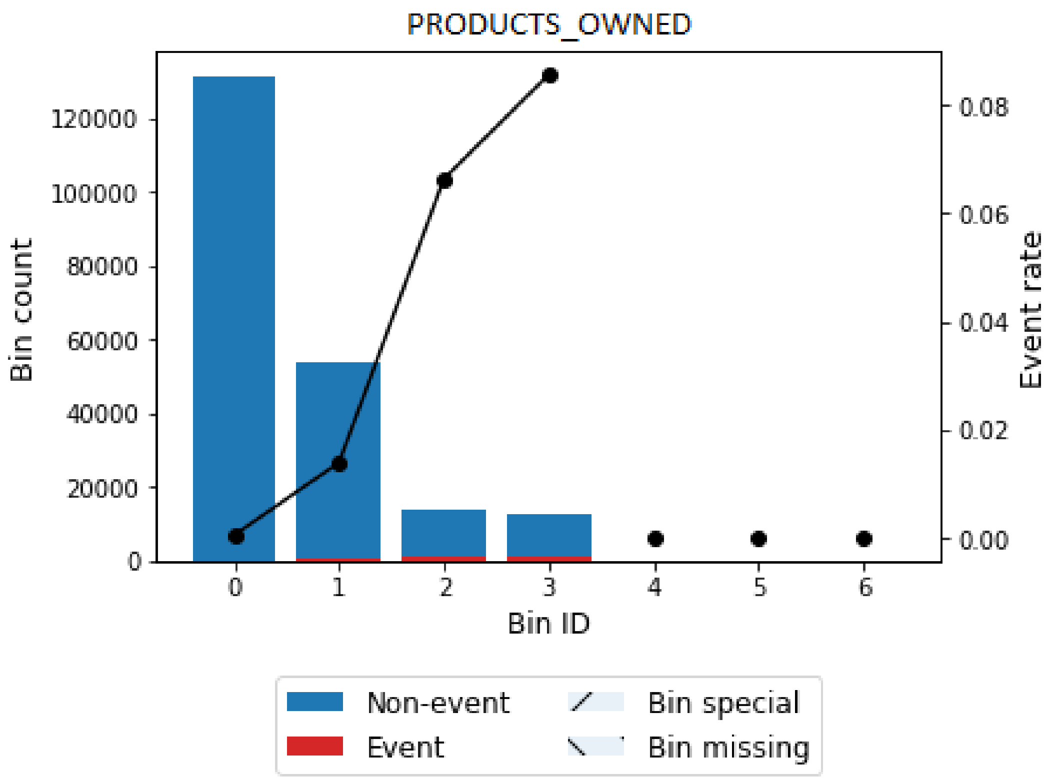

Figure 4.3 : Event rate values of the "PRODUCTS _OWNED" variable.

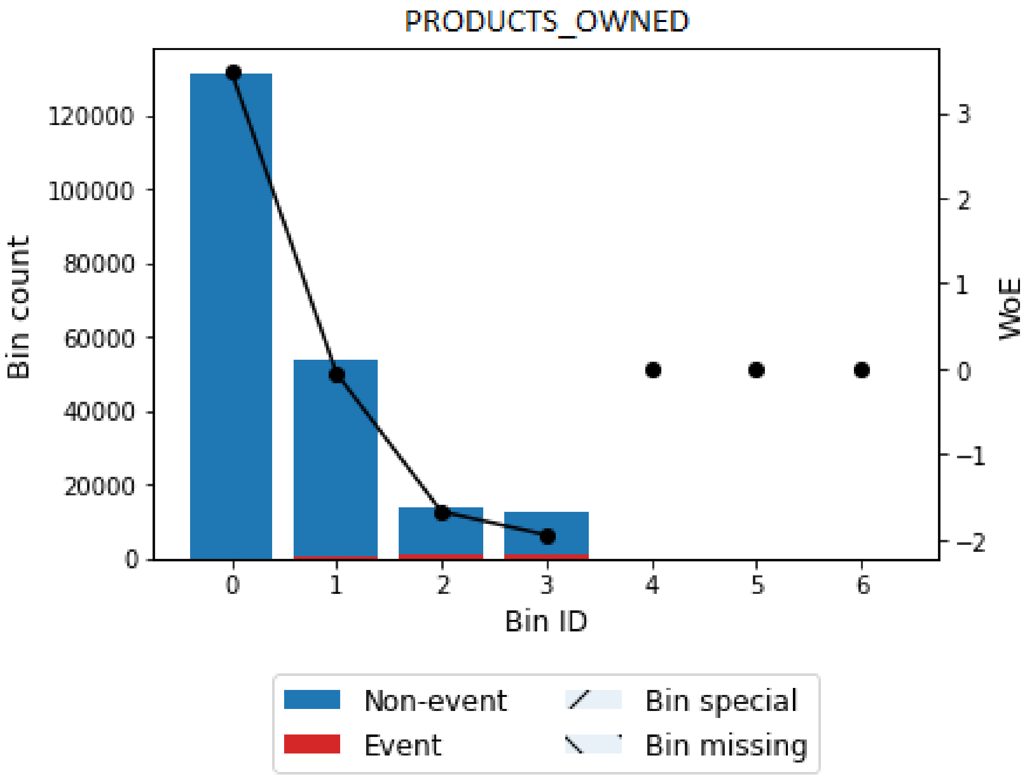

Figure 4.4 : WoE values of the "PRODUCTS _OWNED" variable.

Since there are quite a few records in the dataset with a target value of 1, the target can be said to be imbalanced as seen below (Figure 3.5). Imbalanced target data can cause several flaws while creating models and measuring them. There are some approaches to solve that problem will be examined in the following sections.

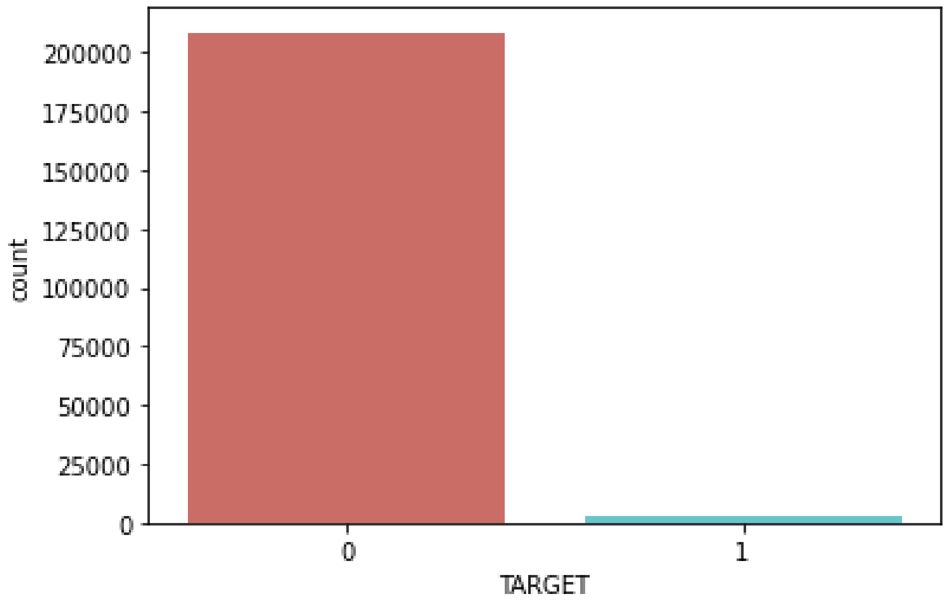

Figure 4.5 : Distribution of the target variable.

4.6.2 Information value calculation

Information value is a value to describe how much a variable affects the target definition. With this value, the representative power of a variable is calculated.

If the information value is below a certain level, these variables are less explanatory. Variables in this situation should not be taken into account when constructing the model. Within the scope of this study, 37 variables with an IV value below 0.8 will be eliminated and thus 43 variables will remain.

4.6.3 Correlation analysis

The occurrence of a perfect or precise linear connection between certain or all of the predictive variables within a regression model was the original meaning of the word multicollinearity. Recently, though, the word multicollinearity is often used extensively to refer to both situations in which there is perfect and imperfect multicollinearity of the variables.

According to Gujarati, multicollinearity can be caused by a number of factors, including the way the data was collected, restrictions on the model or on the sample that was selected, model configuration, and even a deterministic model (2002).

Within the scope of this study, in order to prevent multicollinearity, variables that are directly related to each other are tried to be integrated into the model by singularizing them by performing correlation analysis. Among the variables that are highly correlated with each other, the variable with a low IV value is eliminated, while the variable with a high IV value is included in the model. In this way, 24 more variables are eliminated to be excluded from the model, while 23 variables remain. In the correlation analysis phase, variables with a correlation value above 0.65 are considered as being highly correlated with each other.

4.6.4 Weight of evidence transformation

At this stage, individual WoE transformations are performed for the existing variables. The purpose of the WoE conversion is to directly consider the WoE value of that variable instead of the variable value for each record. In fact, with this process, both the effect of outliers is reduced (censoring) and the normalization process is performed.

During the WoE transformation, categorical variables are grouped and sorted together with numerical variables. Thus, categorical variables can be digitized and considered like numerical variables.

Only three of the 24 variables included in this stage are categorical. WoE conversion is performed for these three variables. However, in the new situation, different correlations may have occurred over these three previously categorical variables. For this reason, correlation elimination must be performed again.

With the correlation analysis, 13 different variables were eliminated. After this last stage of data manipulation, model-building studies can start with 11 variables that have a certain relationship with the target variable.

4.7 Building Models

Different models will be established by using the final variables obtained by the manipulation of the variables in the data set. These models are named logistic regression and random forest, respectively.

4.7.1 Logistic Regression

First, a logistic regression model is established using all variables in order to demonstrate why applying variable selection and other phases such as WoE, IV, and correlation elimination are required. Then, several logistic regression models are established with various approaches using the available variables. These logistic regression models, which are established by using three different methods as class weight, equal weight and p-value, will be examined in detail.

4.7.1.1 Full model

A p-value logistic regression model is established by considering all the variables in the data set. It is observed that the standard errors of the variable coefficients cannot be calculated and the perfect separation situation is observed. In addition, matrix operations cannot be performed correctly due to variables that are replicas of each other.

Due to perfect separation, all outputs can be accurately predicted with only one variable, thus the model is overfitted. In order to solve the problem, the variables that cause this situation can be manually removed from the data set, but this solution alone will not be enough since there are many variables that have a significant correlation between them.

As a result, it is not possible to obtain a successful model with this method. Therefore, as in the following process, meaningful models should be established by using selected variables and particular methods obtained after certain stages.

4.7.1.2 Class weight

In the class weight approach, a model is developed with 70% of the data and tested with the remaining 30%. Errors in target estimation are penalized. While calculating these loss penalties, a penalty with a weighting coefficient of 0.25 and 0.75, respectively, is applied for the incorrectly estimated target values of 0 and 1. Thus, during target estimation, any error made for values of 1 is penalized 3 times more than an error made for values of 0. This ratio is determined by the severity of estimating a value of 1 for the target. After these processes, the model is established. Confusion matrix graphs obtained for train, test, and the entire data set are shown below (Figure 4.6, Figure 4.7 & Figure 4.8).

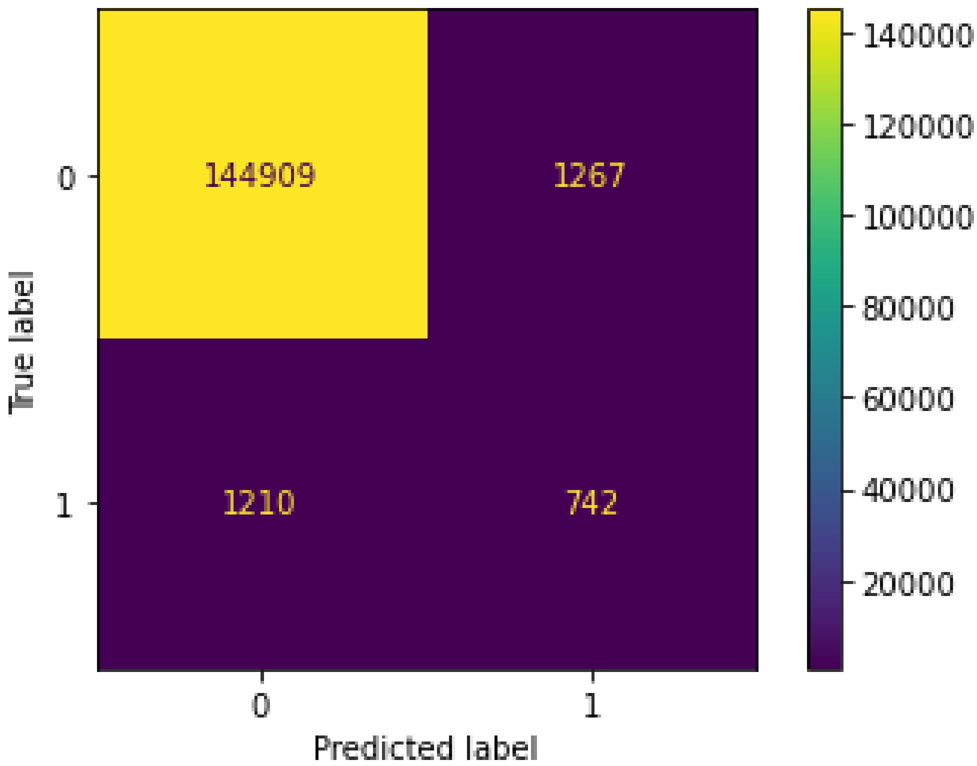

Figure 4.6 : Confusion matrix of the train data in class weight approach.

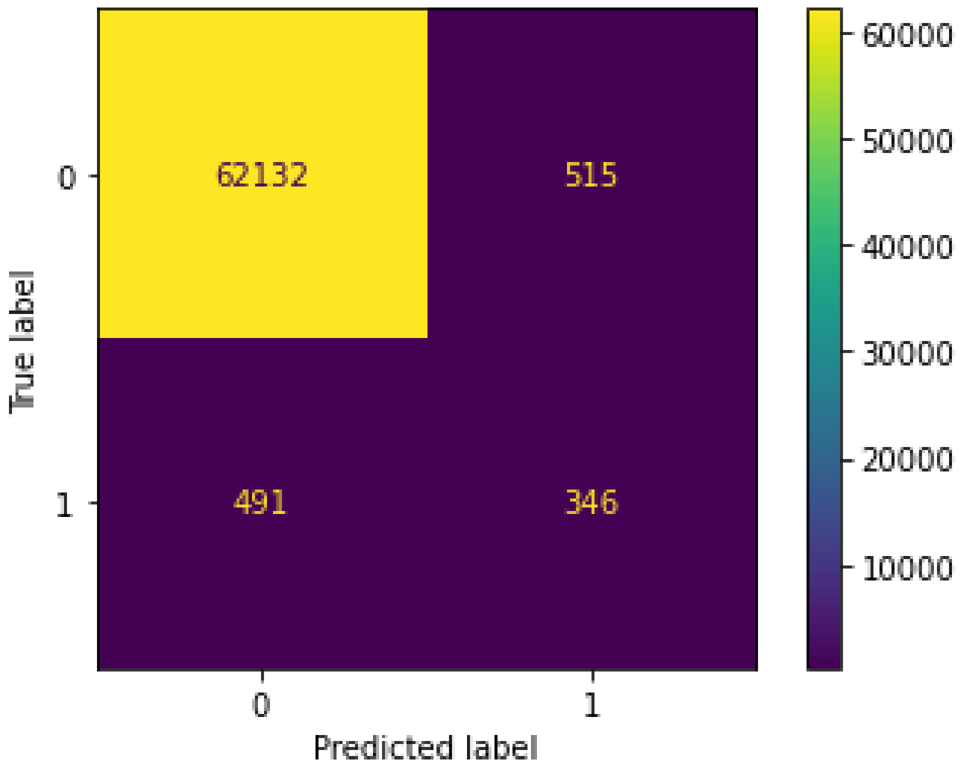

Figure 4.7 : Confusion matrix of the test data in class weight approach.

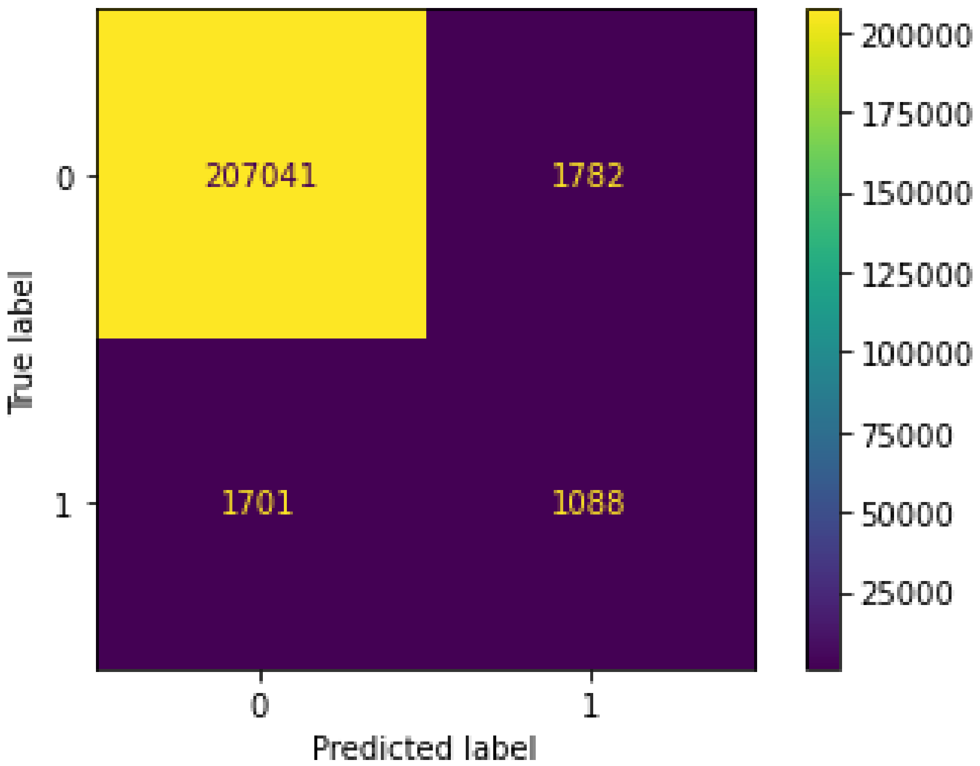

Figure 4.8 : Confusion matrix of all data in class weight approach.

When the table containing the model results is examined, the terms of model accuracy, F1-score, precision and recall are seen (Table 4.1). During the model selection phase, all these metrics will be taken into account by considering the objective.

Table 4.1 : Model metrics and results in class weight approach.

Metrics	Train Results	Test Results	Full Results
Accuracy	0.98	0.98	0.98
F1_score	0.37	0.41	0.38
Precision	0.37	0.4	0.38
Recall	0.38	0.41	0.39

The accuracy statistic counts the number of times a model correctly predicted over the full dataset. This measure can only be trusted if the dataset is class-balanced, meaning that each class contains an equal amount of observations.

A machine learning assessment statistic known as the F1-score integrates a model's precision and recall numbers to determine how accurate the model is. F1-score, as opposed to accuracy, is a more useful statistic in cases where classes are imbalanced.

The proportion of relevant occurrences among the retrieved occurrences is known as precision (positive predictive value), whereas the proportion of relevant instances that were retrieved is known as recall (sensitivity), as seen below (Figure 4.9). Thus, relevance serves as the foundation for both accuracy and memory.

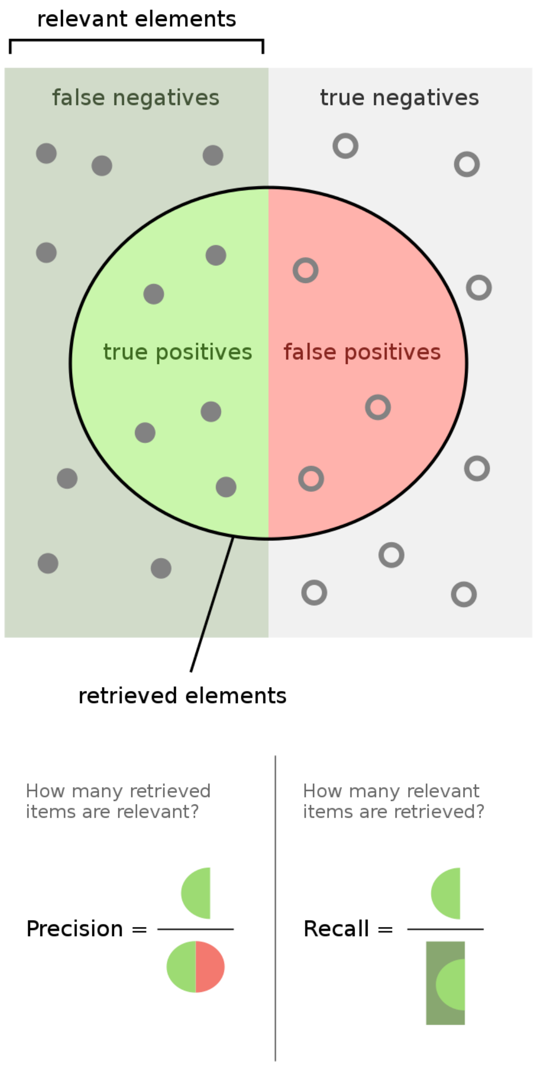

Figure 4.9 : Visualizations of the precision and recall terms.

4.7.1.3 Equal weight

In the equal weight approach, a model is developed with 70% of the data and tested with the remaining 30%. However, in this method, unlike the first method, the number of records with a target value of 1 and 0 is equalized by undersampling and progressed. Thus, the data set is narrowed and a model is established by preventing the class-imbalance. Confusion matrix graphs obtained for train, test, and the entire data set are shown below (Figure 4.10, Figure 4.11 & Figure 4.12).

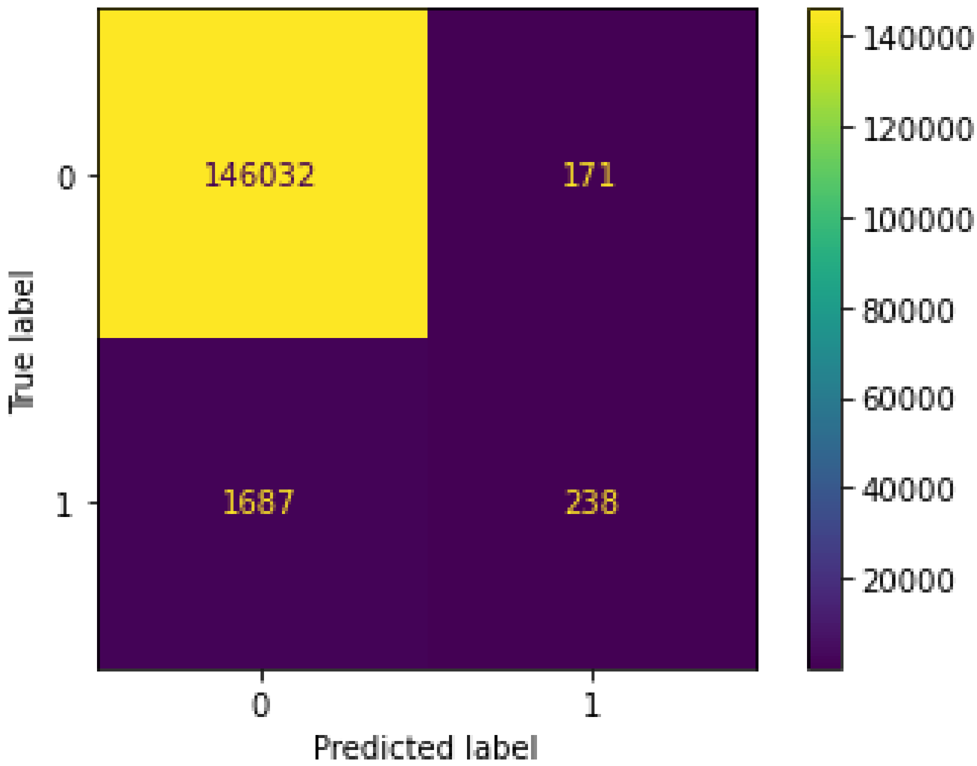

Figure 4.10 : Confusion matrix of the train data in equal weight approach.

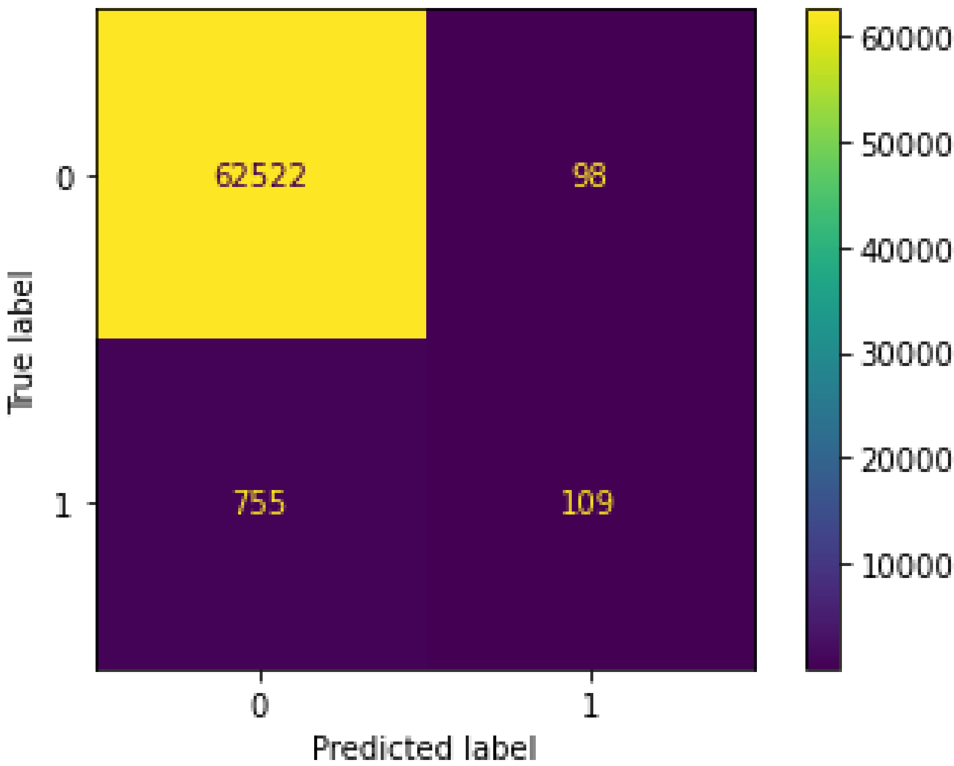

Figure 4.11 : Confusion matrix of the test data in equal weight approach.

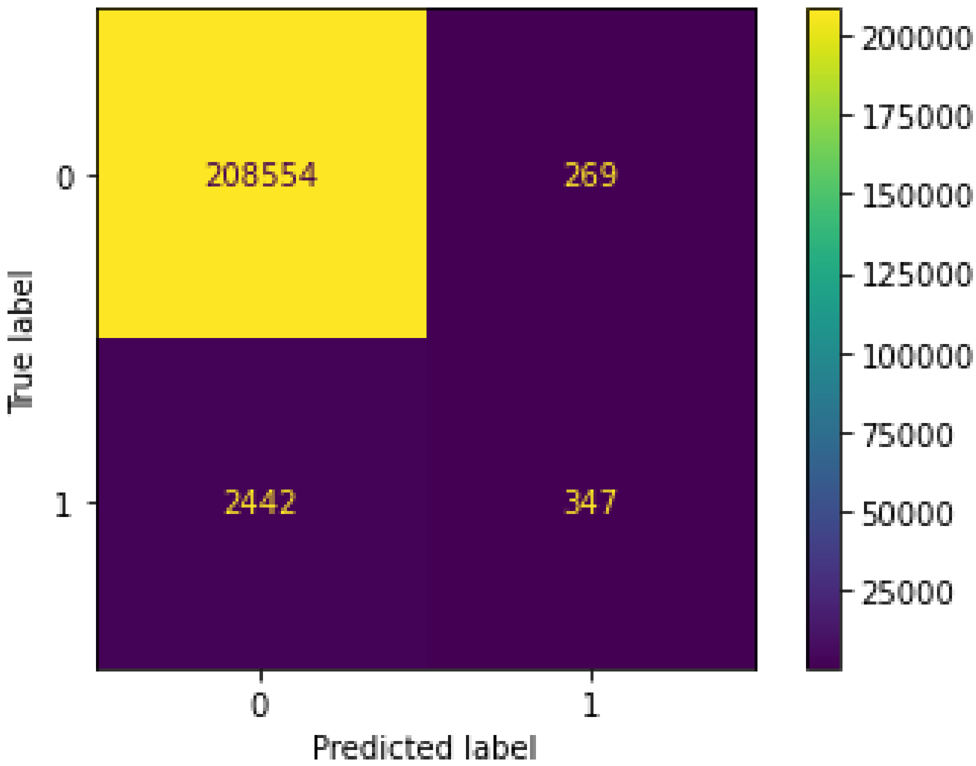

Figure 4.12 : Confusion matrix of all data in equal weight approach.

When the table containing the model results is examined, the terms of model accuracy, F1-score, precision and recall are seen (Table 4.2).

Table 4.2 : Model metrics and results in equal weight approach.

Metrics	Train Results	Test Results	Full Results
Accuracy	0.99	0.99	0.99
F1_score	0.2	0.2	0.2
Precision	0.58	0.53	0.56
Recall	0.12	0.13	0.12

4.7.1.4 P-value

In the p-value approach, a model is developed with 70% of the data and tested with the remaining 30%. However, unlike other methods, the p-values of the variables are included and the two variables with distorted values are removed and then model is built on the remaining variables. Confusion matrix graphs obtained for train, test, and the entire data set are shown below (Figure 4.13, Figure 4.14 & Figure 4.15).

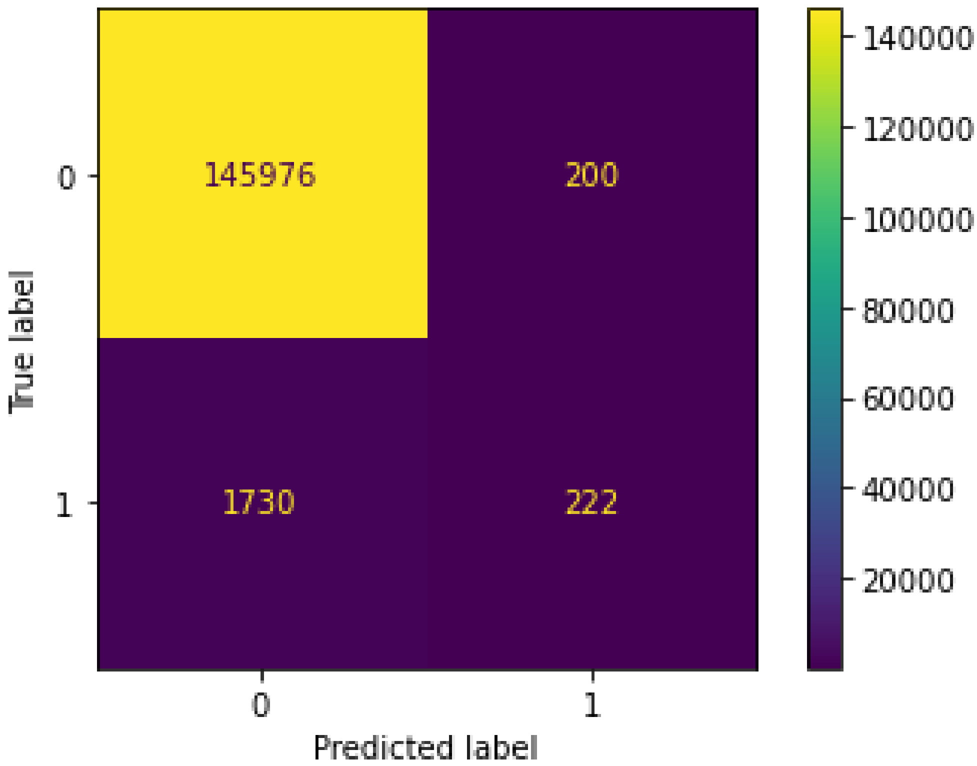

Figure 4.13 : Confusion matrix of the train data in p-value approach.

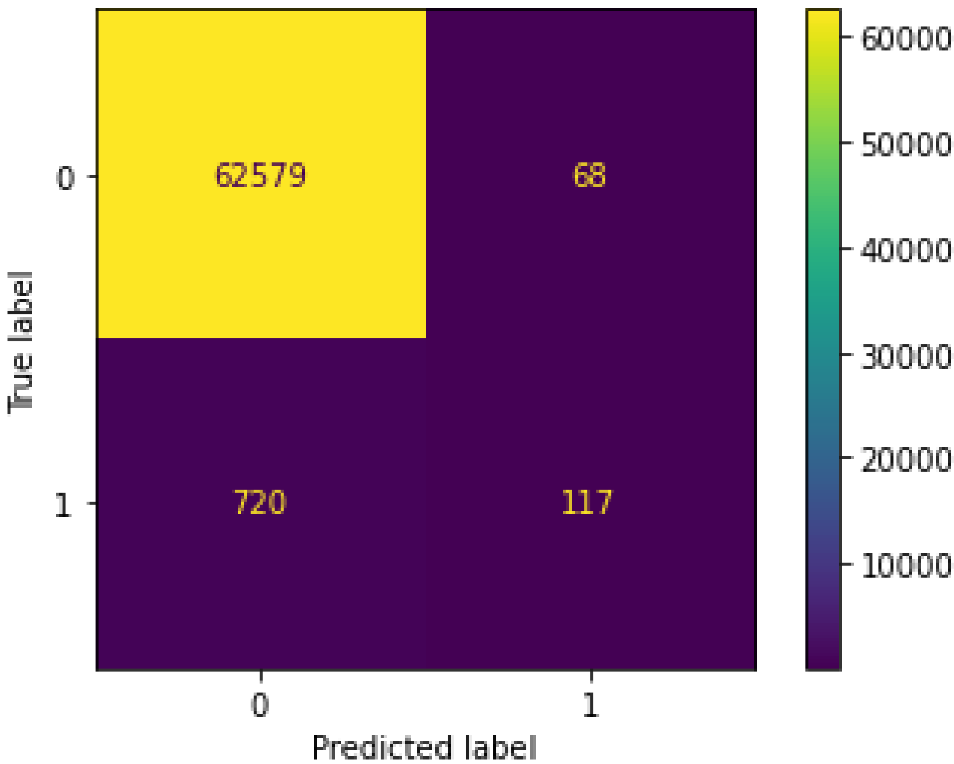

Figure 4.14 : Confusion matrix of the test data in p-value approach.

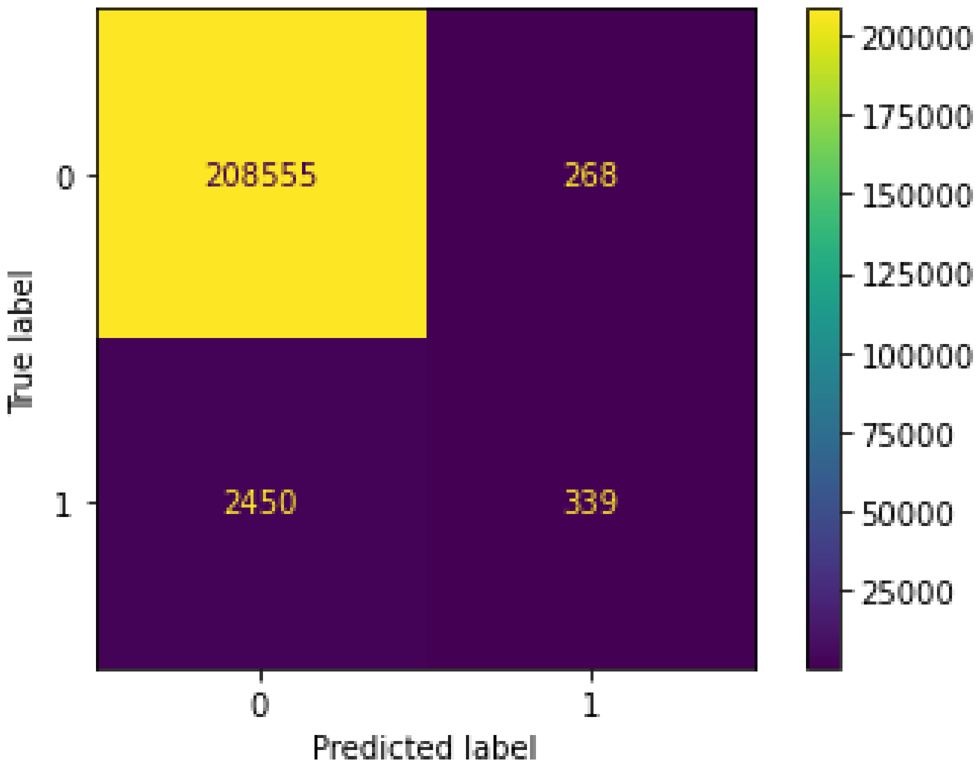

Figure 4.15 : Confusion matrix of all data in p-value approach.

When the table containing the model results is examined, the terms of model accuracy, F1-score, precision and recall are seen (Table 4.3).

Table 4.3 : Model metrics and results in equal weight approach.

Metrics	Train Results	Test Results	Full Results
Accuracy	0.99	0.99	0.99
F1_score	0.19	0.23	0.2
Precision	0.53	0.63	0.56
Recall	0.11	0.14	0.12

The results of the created logistic regression models were examined. Among the equal weight and p-value approaches with relatively high precision values, the p-value approach, which gives the highest precision for an unknown test population, was preferred. The reason why precision is taken into account in this selection is the idea that it is more important for the bank to predict customers with a target value of 1.

4.7.2 Random forest

Secondly, two random forest models are established using the available variables. One of them is the model established with default parameters, while the other is a model developed with the k-fold cross validation technique.

K-fold is validation technique in which the data is split into k-subsets and the holdout method is repeated k-times where each of the k subsets are used as test set and other k-1 subsets are used for the training purpose.

4.7.2.1 Default parameters

In the default approach, a model is developed with 70% of the data and tested with the remaining 30%. The random forest model is run with the default settings in its own document. In addition, the model is prepared with the help of scikit-learn, a free software machine learning library for programming languages. Confusion matrix graphs obtained for train, test, and the entire data set are shown below (Figure 4.16, Figure 4.17 & Figure 4.18).

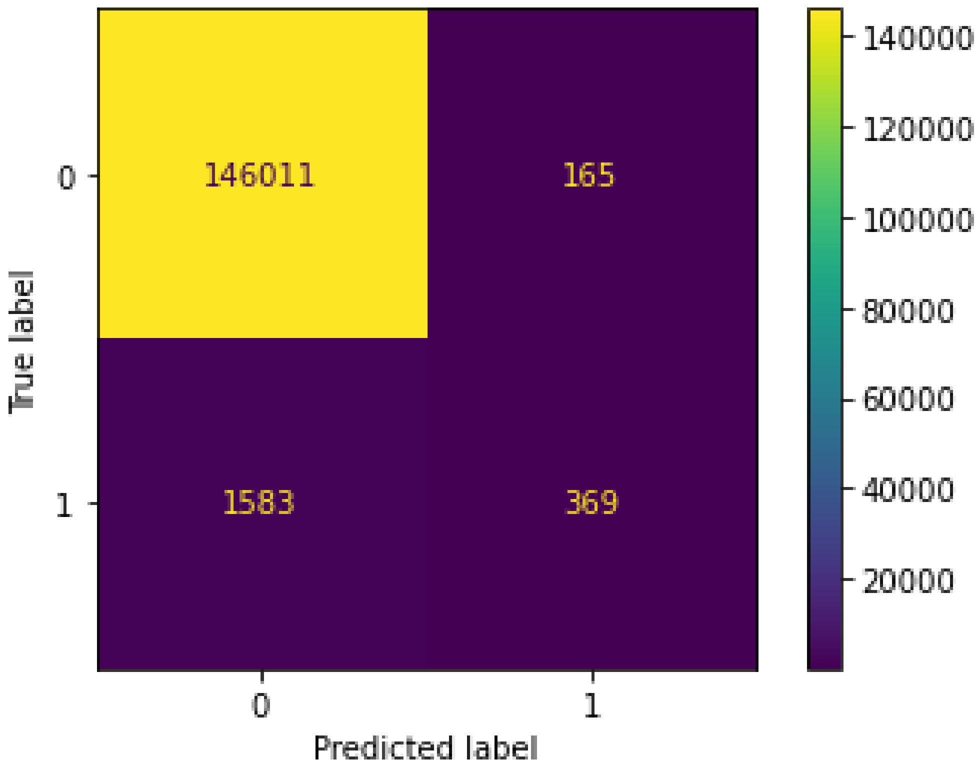

Figure 4.16 : Confusion matrix of the train data in default approach.

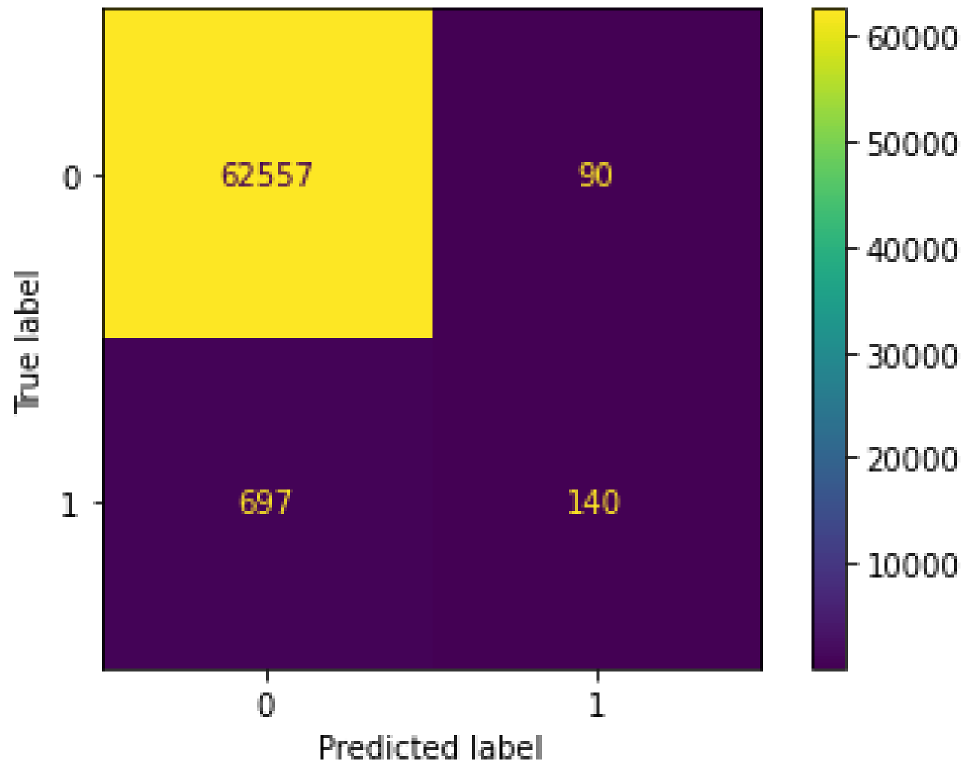

Figure 4.17 : Confusion matrix of the test data in default approach.

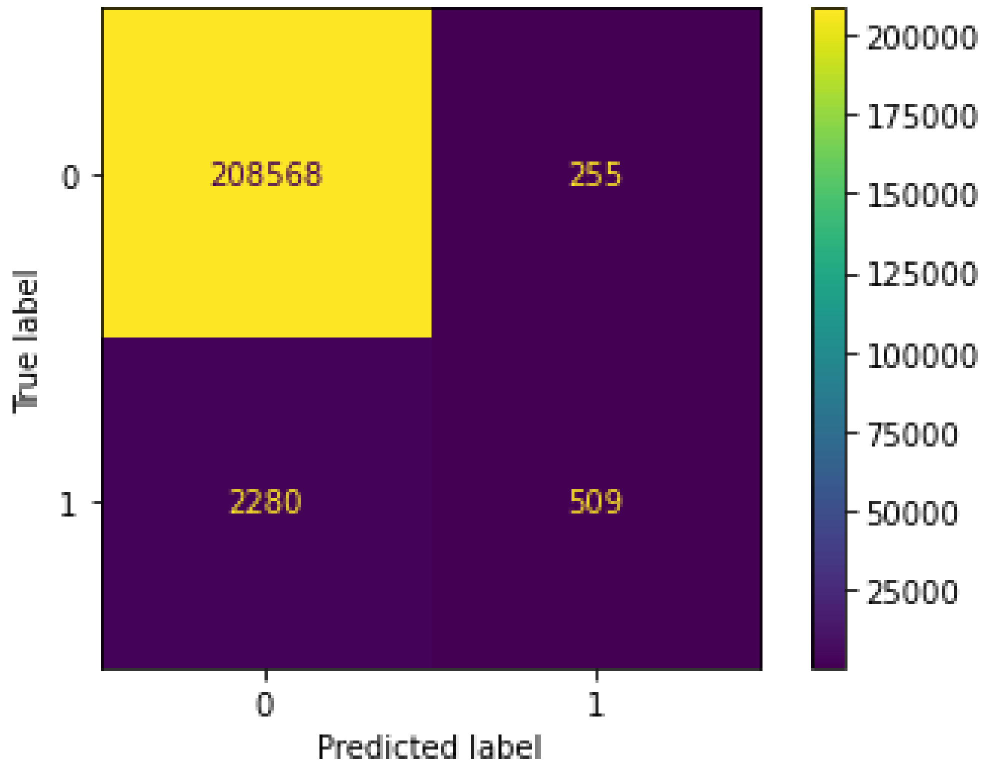

Figure 4.18 : Confusion matrix of all data in default approach.

According to values of metrics, it can be seen that random forest model with default parameters is successful in forecasting target values of 1 in the dataset (Table 4.4).

Table 4.4 : Model metrics and results in default approach.

Metrics	Train Results	Test Results	Full Results
Accuracy	0.99	0.99	0.99
F1_score	0.3	0.26	0.29
Precision	0.69	0.61	0.67
Recall	0.19	0.17	0.18

4.7.2.2 K-fold cross validation

In the k-fold cross validation approach, a model is developed with 70% of the data and tested with the remaining 30%. The random forest model is run with the advanced settings where k equals to 15. Principally, a k-fold cross validation is used to avoid overfitting.

According to the values of metrics, 0.987 as average and 0.002 as standard deviation of cross-validation score, it can be seen that random forest model with k-fold cross validation technique is also successful. These two metrics, close cross-validation scores and low standard deviation prove that the model is consistent.

While establishing random forest models, stratification function was used so as to maintain the target ratio between datasets. Stratification is used when the datasets contain unbalanced classes. Thus, if cross-validation is applied with a normal technique, it may produce subsamples that have a varying distribution of classes. Some unbalanced samples may produce exceptionally high scores leading to a high cross-validation score overall, which is undesirable. Therefore, stratified subsamples that preserve the class frequencies in the individual folds are created in order to ensure a clear picture of the model performance.

Since logistic regression and random forest models are accurate and precise, they can be compared head-to-head among the most powerful metric, which is precision. According to this indicator, random forest is the most successful model.

5. CONCLUSIONS AND RECOMMENDATIONS

Basic reasoning is preferred over complicated one, according to the concept of parsimony, which has long been employed as a justification for selecting one assumption over another. With the aim of limiting the number of events (changes) necessary to produce the dataset, the maximum parsimony analysis of the data is based on this idea. When changes are uncommon, this makes intuitive sense, but when changes are frequent, it might result in deliberate mistakes. Parsimony-based analyses are used in a variety of contexts notwithstanding this concern (Olsen, 2013).

In this study, the number of variables, which was 83 at the beginning, is reduced to 11 during the model-building phase, in accordance with the principle of parsimony. A meaningful result is tried to be obtained by entering these variables into the logistic regression and random forest models. According to the results obtained, the logistic regression model works with 98% accuracy, while the random forest model works with 99% accuracy. In addition, the precision value obtained in the random forest model is higher than that in the logistic regression model.

For these reasons, it is decided that the model to be used should be random forest. In this way, the detection rate of customers with a target definition of 1 for the business card product will be higher, which will increase the bank's customer portfolio and profitability.

5.1 Effects on Future Studies

It is aimed in this thesis study to eliminate these deficiencies and errors in existing exercises to establish a more beneficial and efficient model for banks. In addition, it is recommended that banks expand their perspectives on which data they will use while establishing the relevant model.

At the end of the project, financial institutions will be able to establish healthier models by integrating and using more accurate and consistent market data into their databases, if necessary. This research will provide a successful trend model with meaningful explanatory variables for business card-like products for future works.

REFERENCES

Abdul-Muhmin, A., & Umar, Y. (2007). Credit card ownership and usage behaviour in Saudi Arabia: The impact of demographics and attitudes toward debt. *Journal of Financial Services Marketing*, 219-234.

Ahmed, Z., Ismail, I., Sohail, M., Tabsh, I., & Alias, H. (2010). Malaysian consumers' credit card usage behavior. *Asia Pacific Journal of Marketing and Logistics*.

Akkoç, S. (2012). An empirical comparison of conventional techniques, neural networks and the three stage hybrid Adaptive Neuro Fuzzy Inference System (ANFIS) model for credit scoring analysis: The case of Turkish credit card data. *European Journal of Operational Research*, 168-178.

Ala'raj, M., Abbod, M., Majdalawieh, M., & Jum'a, L. (2022). A deep learning model for behavioural credit scoring in banks. *Neural Computing and Applications*, 5839-5866.

Arora, S., Bindra, S., Singh, S., & Kumar Nassa, V. (2022). Prediction of credit card defaults through data analysis and machine learning techniques. *Materials Today: Proceedings*, 110-117.

Asha, R., & Suresh Kumar, K. (2021). Credit card fraud detection using artificial neural network. *Global Transitions Proceedings*, 35-41.

Bellotti, T., & Crook, J. (2013). Forecasting and stress testing credit card default using dynamic models. *International Journal of Forecasting*, 563-574.

Boone, T., Ganeshan, R., Jain, A., & Sanders, N. (2019). Forecasting sales in the supply chain: Consumer analytics in the big data era. *International Journal of Forecasting*, 170-180.

Forough, J., & Momtazi, S. (2021). Ensemble of deep sequential models for credit card fraud detection. *Applied Soft Computing*.

García, J., Pacce, M., Rodrigo, T., Ruiz de Aguirre, P., & Ulloa, C. (2021). Measuring and forecasting retail trade in real time using card transactional data. *International Journal of Forecasting*, 1235-1246.

Golbayani, P., Florescu, I., & Chatterjee, R. (2020). A comparative study of forecasting corporate credit ratings using neural networks, support vector machines, and decision trees. *The North American Journal of Economics and Finance*.

Gujarati, D. (2002). Multicollinearity: What Happens If the Regressors Are Correlated? D. Gujarati içinde, *Basic Econometrics* (s. 341-386). McGraw-Hill.

Hon, P., & Bellotti, T. (2016). Models and forecasts of credit card balance. *European Journal of Operational Research*, 498-505.

Kaufman, S., Rosset, S., & Perlich, C. (2011). Leakage in Data Mining: Formulation, Detection, and Avoidance. (s. 556-563). New York, United States: ACM Transactions on Knowledge Discovery from Data.

Leow, M., & Crook, J. (2016). A new Mixture model for the estimation of credit card Exposure at Default. *European Journal of Operational Research*, 487-497.

Li, Y., Li, Y., & Li, Y. (2019). What factors are influencing credit card customer's default behavior in China? A study based on survival analysis. *Physica A: Statistical Mechanics and its Applications*.

Lin, C.-S., Tzeng, G.-H., & Chin, Y.-C. (2011). Combined rough set theory and flow network graph to predict customer churn in credit card accounts. *Expert Systems with Applications*, 8-15.

Martins, M., & Cardoso, M. (2012). Cross-validation of segments of credit card holders. *Journal of Retailing and Consumer Services*, 629-636.

Nami, S., & Shajari, M. (2018). Cost-sensitive payment card fraud detection based on dynamic random forest and k-nearest neighbors. *Expert Systems with Applications*, 381-392.

Nie, G., Rowe, W., Zhang, L., Tian, Y., & Shi, Y. (2011). Credit card churn forecasting by logistic regression and decision tree. *Expert Systems with Applications*, 15273-15285.

Olsen, G. (2013). Parsimony. *Brenner's Encyclopedia of Genetics (Second Edition)*, 229-232.

Patil, S., Nemade, V., & Soni, P. (2018). Predictive Modelling For Credit Card Fraud Detection Using Data Analytics. *Procedia Computer Science*, 385-395.

Qi, M., & Yang, S. (2003). Forecasting consumer credit card adoption: what can we learn about the utility function? *International Journal of Forecasting*, 71-85.

Roseline, J., Naidu, G., Samuthira Pandi, V., Alamelu Alias Rajasree, S., & Mageswari, N. (2022). Autonomous credit card fraud detection using machine learning approach. *Computers and Electrical Engineering*.

Rtayli, N., & Enneya, N. (2020). Selection Features and Support Vector Machine for Credit Card Risk Identification. *Procedia Manufacturing*, 941-948.

Shen, A., Tong, R., & Deng, Y. (2007). Application of Classification Models on Credit Card Fraud Detection. *International Conference on Service Systems and Service Management* (s. 1-4). Chengdu, China: IEEE.

Srinath, T., & Gururaja, H. (2022). Explainable machine learning in identifying credit card defaulters. *Global Transitions Proceedings*, 119-126.

Taghiyeh, S., Lengacher, D., & Handfield, R. (2021). Loss rate forecasting framework based on macroeconomic changes: Application to US credit card industry. *Expert Systems with Applications*.

Tang, L., Li, J., Du, H., Li, L., Wu, J., & Wang, S. (2022). Big Data in Forecasting Research: A Literature Review. *Big Data Research*.

Wang, M., & Yang, H. (2020). Research on Customer Credit Scoring Model Based on Bank Credit Card. *Intelligent Information Processing X*, 232-243.

Xu, N., Yuan, Y., & Rong, Z. (2022). Depressed access to formal finance and the use of credit card debt in Chinese SMEs. *China Economic Review*.

Xuan, S., Liu, G., Li, Z., Zheng, L., Wang, S., & Jiang, C. (2018). Random forest for credit card fraud detection. *15th International Conference on Networking, Sensing and Control (ICNSC)* (s. 1-6). Zhuhai, China: IEEE.

Yucelt, U., & MacGregor, R. (2015). Attitudinal and Behavioural Characteristics of American and French Canadian Credit Card Holders. N. Malhotra içinde, *Proceedings of the 1985 Academy of Marketing Science (AMS) Annual Conference* (s. 110-115). Springer International Publishing.

Zhou, Y., Chi, G., Liu, J., Xiong, J., & Wang, B. (2022). Default discrimination of credit card: Feature combination selection based on improved FDAF-score. *Expert Systems with Applications*.

APPENDICES

APPENDIX A: Step 1: Data preparation and WoE transformation queries
APPENDIX B: Step 2: Logistic Regression queries
APPENDIX C: Step 3: Random Forest queries

APPENDIX A: Step 1: Data preparation and WoE transformation queries

```
# -*- coding: utf-8 -*-
"""
@author: Onur Bozkurt
December 2022
"""

# Calling required libraries
import pandas as pd
import numpy as np
from optbinning import BinningProcess # for IV and GINI calculation
from optbinning import OptimalBinning # for monoton coarse classing
import matplotlib.pyplot as plt
import seaborn as sns

# Importing data
df_usage_habit = pd.read_csv('bc_dataset.csv',
encoding='unicode_escape')
df_usage_habit.set_index('ID', inplace=True) # indexing the ID column

# Exploratory Data Analysis (EDA)
df_usage_habit.info()
des = df_usage_habit.describe()
del des
target_dist =
df_usage_habit.groupby(["TARGET"]).size().reset_index(name='counts')
target_dist['percentage'] = (target_dist['counts'] /
target_dist['counts'].sum())
del target_dist

sns.countplot(x='TARGET',data=df_usage_habit, palette='hls')
plt.show()

# Categorical variable analysis - Target distribution in categories
categorical_variables =
list(df_usage_habit.select_dtypes(exclude=[np.number]).columns)
print(categorical_variables)
k1 = df_usage_habit.groupby(["KAC_AYDIR_FINANSAL_INAKTIF",
"TARGET"]).size().reset_index(name='counts')
k1['percentage'] =
k1.groupby('KAC_AYDIR_FINANSAL_INAKTIF')['counts'].apply(lambda x:
np.round(x*100/x.sum(), 2))
del k1

# Deleted due to its relation to target
df_usage_habit = df_usage_habit.drop(["DegerBazliSegment",
"YAKIN_TAKIP_KODU", "SATIS_AGACI_KANAL"], axis=1)
```

```
# Missing value control
missing = df_usage_habit.isnull().sum()
missing = missing[missing>0]
del missing

# WoE transformation will be applied to the data. WoE conversion is a
technique frequently used in credit scoring.
# Thus, the variables will be standardized and any outliers will be
overcome.
# In the WoE transformation, the continuous variables are first
categorized and the WoE value is calculated over these categories (in
other words, bins).
# Outliers or extreme values are assigned to a relevant category and
receive the WoE value of that category.
# A category is also created for the missing values and the WoE value
is calculated.

########## Calculation of IV and GINI for all variables ##########

# First of all, IV and GINI value will be calculated for all variables.
Variables that are less related to the target variable will be
eliminated.
# Only fine classing will be done at this stage. After the correlation
analysis and elimination, coarse classing will be made.
# Monotony was not taken into account at this stage. After coarse
classing, a monotonous structure will be tried to be established.

independent_variables = list(df_usage_habit.columns[1:]) # First, there
is the target variable that will be left out.

X = df_usage_habit[independent_variables]
y = df_usage_habit["TARGET"].values

selection_criteria = {
    "iv": {"min": 0.05, "max": 0.9, "strategy": "highest", "top": 100},
    "quality_score": {"min": 0.01}
}

binning_process = BinningProcess(independent_variables,
                                 categorical_variables=categorical_vari
ables,
                                 selection_criteria=selection_criteria)
binning_process.fit(X, y)

fine_classing_iv = binning_process.summary()
fine_classing_iv['iv'] = pd.to_numeric(fine_classing_iv['iv']) # The
object value is converted to numeric.
```

```
fine_classing_iv['gini'] = pd.to_numeric(fine_classing_iv['gini']) #
The object value is converted to numeric.

# fine_classing_iv.to_csv("fine_classing_iv.csv")

del independent_variables
del categorical_variables
del selection_criteria
del y

# Detailed calculations specific to a variable
optb = binning_process.get_binned_variable("URUN_SAHIP_ADET")
orn1 = optb.binning_table.build()
# Applying WoE transform to all variables
# X_transform = binning_process.transform(X, metric="woe")

iv_short = fine_classing_iv[["name", "iv"]]
iv_short1 = iv_short[(iv_short.iv < 0.08)]
iv_eliminated = list(pd.unique(iv_short1.name))

del iv_short
del iv_short1

# Variables with low IV values are eliminated.
df_usage_habit_el1 = df_usage_habit.drop(iv_eliminated, axis =1)

del df_usage_habit
del iv_eliminated

# Correlation Analysis and Elimination
# Categorical variables and the target variable are left out.
categorical_variables_cor =
df_usage_habit_el1.select_dtypes(exclude=[np.number]).columns
categorical_variables_cor = list(categorical_variables_cor)
categorical_variables_cor.append("TARGET")

# Pearson focuses on normally distributed continuous variables.
corr_matrix =
df_usage_habit_el1.drop(columns=categorical_variables_cor).copy().corr(
method='pearson')

del categorical_variables_cor

# Extract column names from unstacked correlation matrix.
corr_pairs =
corr_matrix.unstack().drop_duplicates().abs().sort_values(ascending =
False)
corr_pairs = corr_pairs[corr_pairs < 1]
```

```python
df_cor = pd.DataFrame(corr_pairs, columns =
['correlation']).rename_axis(['Var1', 'Var2']).reset_index()

del corr_pairs
del corr_matrix

# Merging two data and column name changes
cor_iv = pd.merge(df_cor, fine_classing_iv,
left_on='Var1',  right_on='name', how="left")
cor_iv = cor_iv.drop(['name', 'dtype', 'status', 'selected', 'n_bins',
'js', 'gini', 'quality_score'], axis=1).copy().rename({"iv":"Var1_iv"},
axis=1)
cor_iv1 = pd.merge(cor_iv, fine_classing_iv, left_on=
'Var2',  right_on= 'name', how='left')
df = cor_iv1.drop(['name', 'dtype', 'status', 'selected', 'n_bins',
'js', 'gini', 'quality_score'], axis=1).copy().rename({"iv":"Var2_iv"},
axis=1)
cor_el = df[df['correlation'] > 0.65]

del df
del df_cor
del cor_iv
del cor_iv1

# Detection of the variables with a high correlation between them with
a low IV.
# With the following loop, for example, because of a variable that was
eliminated in the 7th stage, another variable is prevented from being
eliminated in the 15th stage.

# Table to write the loop result
iv_eliminated_list = pd.DataFrame({'el': pd.Series(dtype='str')})
cor_el['Completed'] = 0 # Adding a new column to use in the loop

while len(cor_el[cor_el.Completed == 0]) > 0:

    qq = cor_el[cor_el.Completed == 0].head(1)

    v1= float(qq['Var1_iv'])
    v2= float(qq['Var2_iv'])

    if v1 > v2:
        var = qq["Var2"].iloc[0]
    else:
        var = qq["Var1"].iloc[0]

    s2 = pd.DataFrame({'el': [var]})
    iv_eliminated_list = pd.concat([iv_eliminated_list, s2],
ignore_index=True)
```

```
    # iv_eliminated_list = iv_eliminated_list.append({'eliminate':
var}, ignore_index=True)

    cond1 = cor_el['Var1'] == var
    cond2 = cor_el['Var2'] == var

    cor_el.loc[cond1,'Completed'] = 1
    cor_el.loc[cond2,'Completed'] = 1

cor_eliminated = pd.unique(iv_eliminated_list.el)
cor_eliminated = list(cor_eliminated)
df_usage_habit_el2 = df_usage_habit_el1.drop(cor_eliminated, axis=1)

del df_usage_habit_el1
del cond1
del cond2
del iv_eliminated_list
del cor_el
del cor_eliminated
del qq
del v1
del v2
del var

# Coarse classing - Continuous variables
# Defining the variable to implement WoE transformation on and the
target variable in binary structure

# Separation of variables by data type
numeric_variables_cc =
pd.DataFrame(df_usage_habit_el2.select_dtypes([np.number]).drop(['TARGE
T'], axis=1).columns) # Target variable is excluded.
numeric_variables_cc
categorical_variables_cc =
pd.DataFrame(df_usage_habit_el2.select_dtypes(exclude=[np.number]).colu
mns)
categorical_variables_cc

final_variables = pd.concat([numeric_variables_cc,
categorical_variables_cc])

final_variables.to_csv("final_variables.csv")

# A table with the target variable is created. As WoE is generated for
each variable, it is added here.
WoE_usage= df_usage_habit_el2[["TARGET"]]

# The final IV values produced are written in this table.
```

```
var_iv_list = pd.DataFrame({'Variables': [],
        'IV': []})

# The same step is repeated for each variable.
variable = "URUN_SAHIP_ADET"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .01,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage = WoE_usage.assign(URUN_SAHIP_ADET_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variable': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "KKB_SCORE"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .01,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
```

```
binning_table1 = optb.binning_table
binning_table1.build
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage = WoE_usage.assign(KKB_SCORE_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variable': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "UMUMI_LIMIT"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .01,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage = WoE_usage.assign(UMUMI_LIMIT_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "PUSULA_CIRO"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
```

```
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .005,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage = WoE_usage.assign(PUSULA_CIRO_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variable': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "GK_KMH_RISK"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .01,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")
```

```
# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage = WoE_usage.assign(GK_KMH_RISK_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "GK_TOPLAM_RISK"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .01,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage = WoE_usage.assign(GK_TOPLAM_RISK_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "GK_LIMIT_DOLULUK_ORANI"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .01,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
```

```
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage =
WoE_usage.assign(GK_LIMIT_DOLULUK_ORANI_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "BB_SAHIPLIK"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .01,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage = WoE_usage.assign(BB_SAHIPLIK_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)
```

```
variable = "BB_LIMIT"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .01,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage = WoE_usage.assign(BB_LIMIT_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "BB_LIMIT_DOLULUK_ORANI"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .01,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
```

```
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage =
WoE_usage.assign(BB_LIMIT_DOLULUK_ORANI_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "T0_BANKASAYI"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .005,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage = WoE_usage.assign(T0_BANKASAYI_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "T0_NAKDI_BIZ"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
```

```
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .005,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage = WoE_usage.assign(T0_NAKDI_BIZ_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "T0_TUM_KRLIMIT"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .005,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
```

```
WoE_usage = WoE_usage.assign(T0_TUM_KRLIMIT_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "T0_BIZ_KRLIMIT"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .005,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage = WoE_usage.assign(T0_BIZ_KRLIMIT_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "T0_F_300_BIZ"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .005,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
```

```
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage = WoE_usage.assign(T0_F_300_BIZ_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "T0_SORUNLU_F_TOT"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .005,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage = WoE_usage.assign(T0_SORUNLU_F_TOT_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "T0_SORUNLU_F_BIZ"
x = df_usage_habit_el2[variable].values
```

```
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp", monotonic_trend="auto_asc_desc", min_event_rate_diff= .005, special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is 'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the table after tried and decided.
WoE_usage = WoE_usage.assign(T0_SORUNLU_F_BIZ_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "T0_KRLIMIT_ORAN"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp", monotonic_trend="auto_asc_desc", min_event_rate_diff= .005, special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is 'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")
```

```
# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage = WoE_usage.assign(T0_KRLIMIT_ORAN_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "T0_BIZTOTRISK_BIZTOTLIMIT_ORAN"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .005,
special_codes=special_codes)
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage =
WoE_usage.assign(T0_BIZTOTRISK_BIZTOTLIMIT_ORAN_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "T0_TUMTOTRISK_TUMTOTLIMIT_ORAN"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
special_codes = {'special_1': -99999, "special_2": -11111} # So each
will come as a separate category.
optb = OptimalBinning(name=variable, dtype="numerical", solver="cp",
monotonic_trend="auto_asc_desc", min_event_rate_diff= .005,
special_codes=special_codes)
```

```
# min_event_rate_diff (float, optional (default=0)) The minimum event
rate difference between consecutive bins.
# This option currently only applies when monotonic_trend is
'ascending', 'descending', 'peak_heuristic' or 'valley_heuristic'.
optb.fit(x, y)
optb.status
optb.splits
binning_table1 = optb.binning_table
binning_table1.build()
binning_table1.plot(metric="woe")
binning_table1.plot(metric="event_rate")
var_woe = binning_table1.build()
x_transform_woe = optb.transform(x, metric="woe")

# The auto_asc_desc ascending descending options will be added to the
table after tried and decided.
WoE_usage =
WoE_usage.assign(T0_TUMTOTRISK_TUMTOTLIMIT_ORAN_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table1.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


# Categorical variables
variable = "PROFFESSION_NAME"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
optb = OptimalBinning(name=variable, dtype="categorical", solver="mip",
cat_cutoff=0.05)
optb.fit(x, y)
optb.status
optb.splits
binning_table = optb.binning_table
var_woe = binning_table.build()
binning_table.plot(metric="event_rate")
binning_table.plot(metric="woe")
x_transform_woe = optb.transform(x, metric="woe")

# Adding to the main table
WoE_usage = WoE_usage.assign(PROFFESSION_NAME_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "KAC_AYDIR_INAKTIF"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
```

```
optb = OptimalBinning(name=variable, dtype="categorical", solver="mip",
cat_cutoff=0.05)
optb.fit(x, y)
optb.status
optb.splits
binning_table = optb.binning_table
var_woe = binning_table.build()
binning_table.plot(metric="event_rate")
binning_table.plot(metric="woe")
x_transform_woe = optb.transform(x, metric="woe")

# Adding to the main table
WoE_usage = WoE_usage.assign(KAC_AYDIR_INAKTIF_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)


variable = "KAC_AYDIR_FINANSAL_INAKTIF"
x = df_usage_habit_el2[variable].values
y = df_usage_habit_el2.TARGET
optb = OptimalBinning(name=variable, dtype="categorical", solver="mip",
cat_cutoff=0.05)
optb.fit(x, y)
optb.status
optb.splits
binning_table = optb.binning_table
var_woe = binning_table.build()
binning_table.plot(metric="event_rate")
binning_table.plot(metric="woe")
x_transform_woe = optb.transform(x, metric="woe")

# Adding to the main table
WoE_usage =
WoE_usage.assign(KAC_AYDIR_FINANSAL_INAKTIF_WOE=x_transform_woe)
var_iv = pd.DataFrame({'Variables': [variable],
        'IV': [binning_table.iv]})
var_iv_list = pd.concat([var_iv_list, var_iv], axis=0)

# Correlation analysis and elimination after WoE transformation
# Target variable is excluded.

# Pearson focuses on normally distributed continuous variables.
corr_matrix = WoE_usage.drop("TARGET",
axis=1).copy().corr(method='pearson')

# Extract column names from unstacked correlation matrix.
```

```
corr_pairs =
corr_matrix.unstack().drop_duplicates().abs().sort_values(ascending =
False)
corr_pairs = corr_pairs[corr_pairs < 1]

df_cor = pd.DataFrame(corr_pairs, columns =
['correlation']).rename_axis(['Var1', 'Var2']).reset_index()
df_cor = df_cor[df_cor['correlation'] > 0.65]

df_cor.to_csv("df_cor.csv")
var_iv_list.to_csv("var_iv_list.csv")

# The correlated variables are eliminated. Since the number is small,
manual control was made over the excel output.
WoE_usage_final = WoE_usage.drop(columns=['BB_SAHIPLIK_WOE',
'BB_LIMIT_WOE', "GK_TOPLAM_RISK_WOE", "KAC_AYDIR_INAKTIF_WOE"], axis=1)

# Output of the result file
WoE_usage_final.to_csv("WoE_usage_final.csv")

# Additional Notes
# All or some of the split points can be defined manually by defining
user_splits and user_splits_fixed.
# This is possible for both numeric and categorical variables.
```

APPENDIX B: Step 2: Logistic Regression queries

```
# -*- coding: utf-8 -*-
"""
@author: Onur Bozkurt
December 2022
"""

# Calling required libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Importing data
bc =
pd.read_csv('C:/Users/Onur/Desktop/bc_dataset/WoE_usage_final.csv')
bc.set_index('ID', inplace=True) # indexing the ID column

cor_high = ['T0_NAKDI_BIZ_WOE','T0_TUM_KRLIMIT_WOE',
'T0_BIZ_KRLIMIT_WOE', 'T0_F_300_BIZ_WOE', 'T0_SORUNLU_F_TOT_WOE',
'T0_SORUNLU_F_BIZ_WOE',
                  'T0_KRLIMIT_ORAN_WOE',
'T0_BIZTOTRISK_BIZTOTLIMIT_ORAN_WOE',
'T0_TUMTOTRISK_TUMTOTLIMIT_ORAN_WOE']

bc = bc.drop(cor_high, axis=1)

# Split data into 70/30 while keeping the distribution of target in
test set same as that in the pre-split dataset.
from sklearn.model_selection import train_test_split

X = bc.drop('TARGET', axis = 1)
y = bc['TARGET']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 222, stratify = y)

########## Logistic Regression Sklearn v1: Class weight version
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(class_weight={0: 0.25, 1: 0.75})

model_glm_v1=logreg.fit(X_train, y_train)
pred_glm=model_glm_v1.predict(X_test)
pred_prob_glm=model_glm_v1.predict_proba(X_test)
pred_prob_glm[:,1]

model_glm_v1.get_params()
```

```
model_glm_v1.coef_ # Coefficients are the values that multiply the
predictor values.
model_glm_v1.intercept_ # The intercept (sometimes called the
'constant') in a regression model represents the mean value of the
response variable when all of the predictor variables in the model are
equal to zero.

# Evaluation of the model
from sklearn.metrics import accuracy_score, f1_score, precision_score,
recall_score, classification_report, confusion_matrix, roc_curve,
roc_auc_score, auc
from sklearn.metrics import ConfusionMatrixDisplay
# Confusion_matrix visualization

# Evaluation of the model over train data
pred_values = model_glm_v1.predict(X_train)

cm = confusion_matrix(y_train, pred_values)
cm_display = ConfusionMatrixDisplay(cm).plot()
tn, fp, fn, tp = cm.ravel()
tn, fp, fn, tp

print("Accuracy:", accuracy_score(y_train, pred_values))
print("F1_score:", f1_score(y_train, pred_values))
print("Precision:", precision_score(y_train, pred_values))
print("Recall:", recall_score(y_train, pred_values))

print(classification_report(y_train, pred_values))
print(confusion_matrix(y_train, pred_values))

Accuracy = round(accuracy_score(y_train, pred_values), 2)
F1_score = round(f1_score(y_train, pred_values), 2)
Precision = round(precision_score(y_train, pred_values), 2)
Recall = round(recall_score(y_train, pred_values), 2)

data = [['Accuracy', Accuracy], ['F1_score', F1_score], ['Precision',
Precision], ['Recall', Recall]]
log_reg_v1_result_train = pd.DataFrame(data, columns=['Metrics',
'Train_Results'])

# Evaluation of the model over test data
pred_values = model_glm_v1.predict(X_test)

cm = confusion_matrix(y_test, pred_values)
cm_display = ConfusionMatrixDisplay(cm).plot()
tn, fp, fn, tp = cm.ravel()
tn, fp, fn, tp
```

```
print("Accuracy:", accuracy_score(y_test, pred_values))
print("F1_score:", f1_score(y_test, pred_values))
print("Precision:", precision_score(y_test, pred_values))
print("Recall:", recall_score(y_test, pred_values))

print(classification_report(y_test, pred_values))
print(confusion_matrix(y_test, pred_values))

Accuracy = round(accuracy_score(y_test, pred_values), 2)
F1_score = round(f1_score(y_test, pred_values), 2)
Precision = round(precision_score(y_test, pred_values), 2)
Recall = round(recall_score(y_test, pred_values), 2)

data = [['Accuracy', Accuracy], ['F1_score', F1_score], ['Precision',
Precision], ['Recall', Recall]]
log_reg_v1_result_test = pd.DataFrame(data, columns=['Metrics',
'Test_Results'])


# Evaluating results for the entire population
df_full=bc
y3=df_full.TARGET
X3=df_full.drop(axis=0,columns=["TARGET"])

# Logistic Regression
pred_values = model_glm_v1.predict(X3)

cm = confusion_matrix(y3, pred_values)
cm_display = ConfusionMatrixDisplay(cm).plot()
tn, fp, fn, tp = cm.ravel()
tn, fp, fn, tp

print("Accuracy:", accuracy_score(y3, pred_values))
print("F1_score:", f1_score(y3, pred_values))
print("Precision:", precision_score(y3, pred_values))
print("Recall:", recall_score(y3, pred_values))

print(classification_report(y3, pred_values))
print(confusion_matrix(y3, pred_values))

Accuracy = round(accuracy_score(y3, pred_values), 2)
F1_score = round(f1_score(y3, pred_values), 2)
Precision = round(precision_score(y3, pred_values), 2)
Recall = round(recall_score(y3, pred_values), 2)

data = [['Accuracy', Accuracy], ['F1_score', F1_score], ['Precision',
Precision], ['Recall', Recall]]
log_reg_v1_result_full = pd.DataFrame(data, columns=['Metrics',
'Full_Results'])
```

```
# Combining version results.
log_reg_v1 = log_reg_v1_result_train.merge(log_reg_v1_result_test,
on='Metrics', how='left')
log_reg_v1 = log_reg_v1.merge(log_reg_v1_result_full, on='Metrics',
how='left')

# Output of the result file
log_reg_v1.to_csv("log_reg_v1_result.csv")

########## Balancing Phase v2: Equal weight version
sns.histplot(data=bc, x='TARGET')

majority_class_indices=bc[bc["TARGET"]==0].index
minority_class_indices=bc[bc["TARGET"]==1].index

random_majority_class_indices=np.random.choice(majority_class_indices,l
en(minority_class_indices),replace=False)

len(random_majority_class_indices)
len(minority_class_indices)

balanced_indices=np.concatenate([minority_class_indices,random_majority
_class_indices])
len(balanced_indices)

df_usage2=bc.loc[balanced_indices]

sns.histplot(data=df_usage2, x='TARGET')

from sklearn.utils import shuffle # Why do we need?
df_usage2 = shuffle(df_usage2)

X2 = df_usage2.drop('TARGET', axis = 1)
y2 = df_usage2['TARGET']
X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, test_size
= 0.3, random_state = 222)

# Logistic Regression
logreg = LogisticRegression(max_iter=500)
use_lg2=logreg.fit(X_train2, y_train2)

# Evaluation of the model over train data
pred_values = use_lg2.predict(X_train2)

cm = confusion_matrix(y_train2, pred_values)
cm_display = ConfusionMatrixDisplay(cm).plot()
tn, fp, fn, tp = cm.ravel()
tn, fp, fn, tp
```

```
print("Accuracy:", accuracy_score(y_train2, pred_values))
print("F1_score:", f1_score(y_train2, pred_values))
print("Precision:", precision_score(y_train2, pred_values))
print("Recall:", recall_score(y_train2, pred_values))

print(classification_report(y_train2, pred_values))
print(confusion_matrix(y_train2, pred_values))

Accuracy = round(accuracy_score(y_train2, pred_values), 2)
F1_score = round(f1_score(y_train2, pred_values), 2)
Precision = round(precision_score(y_train2, pred_values), 2)
Recall = round(recall_score(y_train2, pred_values), 2)

data = [['Accuracy', Accuracy], ['F1_score', F1_score], ['Precision',
Precision], ['Recall', Recall]]
log_reg_v2_result_train = pd.DataFrame(data, columns=['Metrics',
'Train_Results'])


# Evaluation of the model over test data
pred_values = use_lg2.predict(X_test2)

cm = confusion_matrix(y_test2, pred_values)
cm_display = ConfusionMatrixDisplay(cm).plot()
tn, fp, fn, tp = cm.ravel()
tn, fp, fn, tp

print("Accuracy:", accuracy_score(y_test2, pred_values))
print("F1_score:", f1_score(y_test2, pred_values))
print("Precision:", precision_score(y_test2, pred_values))
print("Recall:", recall_score(y_test2, pred_values))

print(classification_report(y_test2, pred_values))
print(confusion_matrix(y_test2, pred_values))

Accuracy = round(accuracy_score(y_test2, pred_values), 2)
F1_score = round(f1_score(y_test2, pred_values), 2)
Precision = round(precision_score(y_test2, pred_values), 2)
Recall = round(recall_score(y_test2, pred_values), 2)

data = [['Accuracy', Accuracy], ['F1_score', F1_score], ['Precision',
Precision], ['Recall', Recall]]
log_reg_v2_result_test = pd.DataFrame(data, columns=['Metrics',
'Test_Results'])


# Evaluating results for the entire population
df_use3=bc
```

```
y3=df_use3.TARGET
X3=df_use3.drop(axis=0,columns=["TARGET"])

# Logistic Regression
pred_values = use_lg2.predict(X3)

cm = confusion_matrix(y3, pred_values)
cm_display = ConfusionMatrixDisplay(cm).plot()
tn, fp, fn, tp = cm.ravel()
tn, fp, fn, tp

print("Accuracy:", accuracy_score(y3, pred_values))
print("F1_score:", f1_score(y3, pred_values))
print("Precision:", precision_score(y3, pred_values))
print("Recall:", recall_score(y3, pred_values))

print(classification_report(y3, pred_values))
print(confusion_matrix(y3, pred_values))

Accuracy = round(accuracy_score(y3, pred_values), 2)
F1_score = round(f1_score(y3, pred_values), 2)
Precision = round(precision_score(y3, pred_values), 2)
Recall = round(recall_score(y3, pred_values), 2)

data = [['Accuracy', Accuracy], ['F1_score', F1_score], ['Precision',
Precision], ['Recall', Recall]]
log_reg_v2_result_full = pd.DataFrame(data, columns=['Metrics',
'Full_Results'])

# Combining version results.
log_reg_v2 = log_reg_v2_result_train.merge(log_reg_v2_result_test,
on='Metrics', how='left')
log_reg_v2 = log_reg_v2.merge(log_reg_v2_result_full, on='Metrics',
how='left')

# Output of the result file
log_reg_v2.to_csv("log_reg_v2_result.csv")


########## Logistic Regression Statsmodels v3: P-values version
import statsmodels.api as sm
import statsmodels.formula.api as smf

X_train_constant = sm.add_constant(X_train)
X_test_constant = sm.add_constant(X_test)

logit_model=sm.Logit(y_train, X_train_constant)
result=logit_model.fit(method='bfgs')
```

```
print(result.summary2()) # We can see the p-values.
print(result.params)
corr = X_train.corr()
kot = corr[corr>=.9]
plt.figure(figsize=(18,10))
sns.heatmap(kot, cmap="Greens")

p_value_high = ['T0_NAKDI_BIZ_WOE','T0_TUM_KRLIMIT_WOE',
'T0_BIZ_KRLIMIT_WOE', 'T0_F_300_BIZ_WOE', 'T0_SORUNLU_F_TOT_WOE',
'T0_SORUNLU_F_BIZ_WOE', 'T0_KRLIMIT_ORAN_WOE',
                  'T0_BIZTOTRISK_BIZTOTLIMIT_ORAN_WOE',
'T0_TUMTOTRISK_TUMTOTLIMIT_ORAN_WOE', 'PROFFESSION_NAME_WOE',
'KAC_AYDIR_FINANSAL_INAKTIF_WOE']

y_train_p = y_train.copy()
X_train_p = X_train_constant.drop(p_value_high, axis=1)

y_test_p = y_test.copy()
X_test_p = X_test_constant.drop(p_value_high, axis=1)

logit_model1=sm.Logit(y_train_p, X_train_p)
result1=logit_model1.fit()
print(result1.summary2()) # We can see the p-values.


# Evaluation of the model over train data
pred_values1 = result1.predict(X_train_p)
pred_values = [ 0 if x < 0.5 else 1 for x in pred_values1]

cm = confusion_matrix(y_train_p, pred_values)
cm_display = ConfusionMatrixDisplay(cm).plot()
tn, fp, fn, tp = cm.ravel()
tn, fp, fn, tp

print("Accuracy:", accuracy_score(y_train_p, pred_values))
print("F1_score:", f1_score(y_train_p, pred_values))
print("Precision:", precision_score(y_train_p, pred_values))
print("Recall:", recall_score(y_train_p, pred_values))

print(classification_report(y_train_p, pred_values))
print(confusion_matrix(y_train_p, pred_values))

Accuracy = round(accuracy_score(y_train_p, pred_values), 2)
F1_score = round(f1_score(y_train_p, pred_values), 2)
Precision = round(precision_score(y_train_p, pred_values), 2)
Recall = round(recall_score(y_train_p, pred_values), 2)

data = [['Accuracy', Accuracy], ['F1_score', F1_score], ['Precision',
Precision], ['Recall', Recall]]
```

```
log_reg_v3_result_train = pd.DataFrame(data, columns=['Metrics',
'Train_Results'])


# Evaluation of the model over test data
pred_values1 = result1.predict(X_test_p)
pred_values = [ 0 if x < 0.5 else 1 for x in pred_values1]

cm = confusion_matrix(y_test_p, pred_values)
cm_display = ConfusionMatrixDisplay(cm).plot()
tn, fp, fn, tp = cm.ravel()
tn, fp, fn, tp

print("Accuracy:", accuracy_score(y_test_p, pred_values))
print("F1_score:", f1_score(y_test_p, pred_values))
print("Precision:", precision_score(y_test_p, pred_values))
print("Recall:", recall_score(y_test_p, pred_values))

print(classification_report(y_test_p, pred_values))
print(confusion_matrix(y_test_p, pred_values))

Accuracy = round(accuracy_score(y_test_p, pred_values), 2)
F1_score = round(f1_score(y_test_p, pred_values), 2)
Precision = round(precision_score(y_test_p, pred_values), 2)
Recall = round(recall_score(y_test_p, pred_values), 2)

data = [['Accuracy', Accuracy], ['F1_score', F1_score], ['Precision',
Precision], ['Recall', Recall]]
log_reg_v3_result_test = pd.DataFrame(data, columns=['Metrics',
'Test_Results'])


# Evaluation of the results for the entire population
df_use4=bc.drop(p_value_high, axis=1)

df_use4_constant = sm.add_constant(df_use4)

y3=df_use4_constant.TARGET
X3=df_use4_constant.drop(axis=0,columns=["TARGET"])

# Logistic Regression
pred_values1 = result1.predict(X3)
pred_values = [ 0 if x < 0.5 else 1 for x in pred_values1]

cm = confusion_matrix(y3, pred_values)
cm_display = ConfusionMatrixDisplay(cm).plot()
tn, fp, fn, tp = cm.ravel()
tn, fp, fn, tp
```

```
print("Accuracy:", accuracy_score(y3, pred_values))
print("F1_score:", f1_score(y3, pred_values))
print("Precision:", precision_score(y3, pred_values))
print("Recall:", recall_score(y3, pred_values))

print(classification_report(y3, pred_values))
print(confusion_matrix(y3, pred_values))

Accuracy = round(accuracy_score(y3, pred_values), 2)
F1_score = round(f1_score(y3, pred_values), 2)
Precision = round(precision_score(y3, pred_values), 2)
Recall = round(recall_score(y3, pred_values), 2)

data = [['Accuracy', Accuracy], ['F1_score', F1_score], ['Precision',
Precision], ['Recall', Recall]]
log_reg_v3_result_full = pd.DataFrame(data, columns=['Metrics',
'Full_Results'])

# Combining version results
log_reg_v3 = log_reg_v3_result_train.merge(log_reg_v3_result_test,
on='Metrics', how='left')
log_reg_v3 = log_reg_v3.merge(log_reg_v3_result_full, on='Metrics',
how='left')

# Output of the result file
log_reg_v3.to_csv("log_reg_v3_result.csv")


########################################### # Logistic Regression
Statsmodels v4: P-values version with all variables

import statsmodels.api as sm
import statsmodels.formula.api as smf

# Evaluation of the model
from sklearn.metrics import accuracy_score, f1_score, precision_score,
recall_score, classification_report, confusion_matrix, roc_curve,
roc_auc_score, auc
from sklearn.metrics import ConfusionMatrixDisplay # confusion_matrix
gorsellestirme

# Split data into 70/30 while keeping the distribution of target in
test set same as that in the pre-split dataset.
from sklearn.model_selection import train_test_split

# Importing data
data_ful = pd.read_csv('data/WoE_usage_final.csv')
data_ful.set_index('ID', inplace=True) # indexing the ID column
```

```
X = data_ful.drop('TARGET', axis = 1)
y = data_ful['TARGET']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 222, stratify = y)

X_train_constant = sm.add_constant(X_train)
X_test_constant = sm.add_constant(X_test)

logit_model=sm.Logit(y_train, X_train_constant)
result=logit_model.fit(method='bfgs')
print(result.summary2()) # We can see the p-values
```

APPENDIX C: Step 3: Random Forest queries

```
# -*- coding: utf-8 -*-
"""
@author: Onur Bozkurt
December 2022
"""

# Calling required libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Importing data
bc =
pd.read_csv('C:/Users/Onur/Desktop/bc_dataset/WoE_usage_final.csv')
bc.set_index('ID', inplace=True) # indexing the ID column

cor_high = ['T0_NAKDI_BIZ_WOE','T0_TUM_KRLIMIT_WOE',
'T0_BIZ_KRLIMIT_WOE', 'T0_F_300_BIZ_WOE', 'T0_SORUNLU_F_TOT_WOE',
'T0_SORUNLU_F_BIZ_WOE',
            'T0_KRLIMIT_ORAN_WOE',
'T0_BIZTOTRISK_BIZTOTLIMIT_ORAN_WOE',
'T0_TUMTOTRISK_TUMTOTLIMIT_ORAN_WOE']

bc = bc.drop(cor_high, axis=1)

# Split data into 70/30 while keeping the distribution of target in
test set same as that in the pre-split dataset.
from sklearn.model_selection import train_test_split

X = bc.drop('TARGET', axis = 1)
y = bc['TARGET']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 222, stratify = y)

from sklearn.metrics import accuracy_score, f1_score, precision_score,
recall_score, classification_report, confusion_matrix, roc_curve,
roc_auc_score, auc
from sklearn.metrics import ConfusionMatrixDisplay # confusion_matrix
visualization

# Random Forest
from sklearn.ensemble import RandomForestClassifier

rf_clf = RandomForestClassifier(random_state=42138)
rf_clf.fit(X_train, y_train)
```

```python
# Evaluation of the model over train data
pred_values = rf_clf.predict(X_train)

cm = confusion_matrix(y_train, pred_values)
cm_display = ConfusionMatrixDisplay(cm).plot()
tn, fp, fn, tp = cm.ravel()
tn, fp, fn, tp

print("Accuracy:", accuracy_score(y_train, pred_values))
print("F1_score:", f1_score(y_train, pred_values))
print("Precision:", precision_score(y_train, pred_values))
print("Recall:", recall_score(y_train, pred_values))

print(classification_report(y_train, pred_values))
print(confusion_matrix(y_train, pred_values))

Accuracy = round(accuracy_score(y_train, pred_values), 2)
F1_score = round(f1_score(y_train, pred_values), 2)
Precision = round(precision_score(y_train, pred_values), 2)
Recall = round(recall_score(y_train, pred_values), 2)

data = [['Accuracy', Accuracy], ['F1_score', F1_score], ['Precision',
Precision], ['Recall', Recall]]
rf_v1_result_train = pd.DataFrame(data, columns=['Metrics',
'Train_Results'])

# Evaluation of the model over test data
pred_values = rf_clf.predict(X_test)

cm = confusion_matrix(y_test, pred_values)
cm_display = ConfusionMatrixDisplay(cm).plot()
tn, fp, fn, tp = cm.ravel()
tn, fp, fn, tp

print("Accuracy:", accuracy_score(y_test, pred_values))
print("F1_score:", f1_score(y_test, pred_values))
print("Precision:", precision_score(y_test, pred_values))
print("Recall:", recall_score(y_test, pred_values))

print(classification_report(y_test, pred_values))
print(confusion_matrix(y_test, pred_values))

Accuracy = round(accuracy_score(y_test, pred_values), 2)
F1_score = round(f1_score(y_test, pred_values), 2)
Precision = round(precision_score(y_test, pred_values), 2)
Recall = round(recall_score(y_test, pred_values), 2)
```

```
data = [['Accuracy', Accuracy], ['F1_score', F1_score], ['Precision',
Precision], ['Recall', Recall]]
rf_v1_result_test = pd.DataFrame(data, columns=['Metrics',
'Test_Results'])

# Evaluation of the results for the entire population
df_rf3=bc
y3=df_rf3.TARGET
X3=df_rf3.drop(axis=0,columns=["TARGET"])

pred_values = rf_clf.predict(X3)

cm = confusion_matrix(y3, pred_values)
cm_display = ConfusionMatrixDisplay(cm).plot()
tn, fp, fn, tp = cm.ravel()
tn, fp, fn, tp

print("Accuracy:", accuracy_score(y3, pred_values))
print("F1_score:", f1_score(y3, pred_values))
print("Precision:", precision_score(y3, pred_values))
print("Recall:", recall_score(y3, pred_values))

print(classification_report(y3, pred_values))
print(confusion_matrix(y3, pred_values))

Accuracy = round(accuracy_score(y3, pred_values), 2)
F1_score = round(f1_score(y3, pred_values), 2)
Precision = round(precision_score(y3, pred_values), 2)
Recall = round(recall_score(y3, pred_values), 2)

data = [['Accuracy', Accuracy], ['F1_score', F1_score], ['Precision',
Precision], ['Recall', Recall]]
rf_v1_result_full = pd.DataFrame(data, columns=['Metrics',
'Full_Results'])

# Combining version results
rf_v1 = rf_v1_result_train.merge(rf_v1_result_test, on='Metrics',
how='left')
rf_v1 = rf_v1.merge(rf_v1_result_full, on='Metrics', how='left')

# Output of the result file
rf_v1.to_csv("rf_v1_result.csv")

########## Balancing Phase
sns.histplot(data=bc, x='TARGET')

majority_class_indices=bc[bc["TARGET"]==0].index
minority_class_indices=bc[bc["TARGET"]==1].index
```

```
random_majority_class_indices=np.random.choice(majority_class_indices,l
en(minority_class_indices),replace=False)

len(random_majority_class_indices)
len(minority_class_indices)

balanced_indices=np.concatenate([minority_class_indices,random_majority
_class_indices])
len(balanced_indices)

df_rf2=bc.loc[balanced_indices]

sns.histplot(data=df_rf2, x='TARGET')

from sklearn.utils import shuffle # Why do we need?
df_rf2 = shuffle(df_rf2)

X2 = df_rf2.drop('TARGET', axis = 1)
y2 = df_rf2['TARGET']
X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, test_size
= 0.3, random_state = 222)

# Random Forest
use_rf = RandomForestClassifier(random_state=42138)

use_rf.fit(X_train2, y_train2)

pred_values = use_rf.predict(X_train2)
print(classification_report(y_train2, pred_values))
print(confusion_matrix(y_train2, pred_values))

pred_values = use_rf.predict(X_test2)
print(classification_report(y_test2, pred_values))
print(confusion_matrix(y_test2, pred_values))

# Random Forest with K-Fold Cross Validation

from sklearn.model_selection import StratifiedKFold, cross_val_score

Stratified_cross_validate = StratifiedKFold(n_splits=15)

clf = RandomForestClassifier(random_state=20)

score = cross_val_score(clf, X, y, cv = Stratified_cross_validate)

print("Cross Validation scores: {}".format(score))
print("Average Cross Validation score: {}".format(score.mean()))
print("Std of Cross Validation score: {}".format(score.std()))
```

CURRICULUM VITAE

Name Surname **:** Onur BOZKURT

EDUCATION **:**

- **B.Sc.** **:** 2019, Istanbul Technical University, Faculty of Management, Industrial Engineering Department
- **M.Sc.** **:** 2023, Istanbul Technical University, Graduate School, Industrial Engineering Department

PROFESSIONAL EXPERIENCE:

- 2023-present, Pegasus Airlines, Senior Data Analytics Specialist
- 2022-2023, Denizbank A.Ş, CRM Analytics Senior Data Scientist
- 2021-2022, ÜNLÜ & CO, Senior Data Analyst
- 2018-2021, Kuveyt Turk Participation Bank, IFRS 9 and Impairments Specialist