T.R.
YILDIZ TECHNICAL UNIVERSITY
FACULTY OF MECHANICAL ENGINEERING
INDUSTRIAL ENGINEERING

ASSET ESTIMATION OF BANK CUSTOMERS VIA MACHINE LEARNING APPROACH

BELCE KAYA
15061049

BACHELOR THESIS

ADVISOR

ASSOC. PROF. NEZIR AYDIN

ISTANBUL, 2020

PREFACE

The basis of this undergraduate thesis is to estimate the asset information of customers owned but unknown whether they are wealthy or not with the model established based on loyal customers in the bank through the instrumentality of data science principles. Thus, it will be possible to deliver the right campaigns to the right customers and to make the bank more profitable thanks to the customer asset estimation model.

As the world moves towards the digital age, producing large amounts of data and born-digital content, it is becoming more inclined to use this data inefficiently. For this reason, millions of data stored in the bank's data warehouse have been implemented with this project and used correctly and effectively. Big data is arranged, analyzed and visualized to reach loyal customers of the company, which is a crucial level for the training process with machine learning algorithms.

In truth, I could not have achieved my current level of success without a strong support group in my company. First of all, my parents, who supported me with love and understanding. Secondly, my committee members, each of whom has provided patient advice and guidance throughout the research process. Last but not least, I would like to express my sincere wishes to my valuable mentor Yahya Yavuz in the workplace and my precious university advisor Assoc. Prof. Nezir Aydin for their support in the project and patiently sharing their knowledge with me. Endless thanks to all of you!

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

CC	Credit Card	TOT	Total
WPS	Worst Payment Status	PROD	Product
CC_O	Open Credit Card	CRE	Credit
HL	Home Loan (Mortgage)	FLG	Flag
AMT	Amount	STAT	Status
BAL	Balance	CUST	Customer
CNT	Count	NO	Number of
L12M	Last 12 Months	INC	Income
L6M	Last 6 Months	EMP	Employed
L3M	Last 3 Months	LKP	Look up
LM	Last Month	GLM	Generalized Linear Model
L3Y	Last 3 Years	GBM	Gradient Boosting Model
LY	Last Year	RF	Random Forest
OD	Overdraft	XGB	XGBoost (Extreme
DDEP	Demand Deposit		Gradient Boosting)
TDEP	Time Deposit		
FINANCE_AMT	The opening balance	GS	Grid Search
REV	Revolving	CD	Coordinate Descent
DLQ	Delinquency	TP	Type
MAH	Neighborhood	PL	Personal Loan
ILC	District	C	Close
Y	Age	OCC_CLUSTER	Occupational Cluster
INST	Installment	BEH_CLUSTER	Behavioral Cluster
AVG	Average	TRX	Transaction
2ND	Second	AL	Auto Loan
NREV	Non-Revolving	EOD	End of Day
PMT	Payment	MTD	Month to Date

LIST OF FIGURES

LIST OF TABLES

ÖZET

Makine Öğrenimi Yaklaşımı ile Banka Müşterilerinin Varlık Tahmini

Belce KAYA

Endüstri Mühendisliği Bölümü

Lisans Tezi

Danışman: Doç. Dr. Nezir AYDIN

Günümüzde teknoloji dünyası hızla ilerleme kaydetmektedir. Hemen hemen tüm sektörler, büyük veriden konuşur duruma gelmiştir. Büyük veri, günümüzde son derece popüler bir kavramdır ve yeni bir devrin başlangıcı olarak yorumlanmaktadır. Büyük veri ile birlikte, dünyada büyük bir dönüşüm gerçekleşirken, kurum ve kuruluşların veriye olan bakışı ve veriden sağladığı fayda farklı bir noktaya gelmiştir. Sektör farketmeksizin, çoğu firmanın milyonlarca, milyarlarca hatta trilyonlarca verisi mevcuttur. Özellikle bankacılık sektöründe, inanılmaz derecede, işlendiğinde çok önemli fayda sağlayacak veri bulunmaktadır. Ancak bu büyük verinin düzenlenmesi, iyi projelerin çıkarılması oldukça karmaşık ve zordur. Neyse ki birçok büyük veriden kazanım elde etme, onu kullanma ve gelecekle alakalı çıkarımda bulunma yöntemi popüler hale gelmiştir.

Bu projenin amacı, bankanın sahip olduğu veri tabanındaki müşterinin verisi ve KKB verisini kullanarak toplam varlığını tahmin etmektir. Bunu yaparken veri düzenleme, analiz ve temizleme işlemleri için Oracle SQL programı kullanılmıştır. Burada model değişkenleri, gerekli kısıtlar verilerek elde edilmiştir. Bu kısıtlar genel hatlarıyla, müşterinin bankadaki kredi ürünleri, kredi ürünlerinin son 6 aylık KKB sorgusu, müşterinin yaptığı banka içersindeki düzenli para transfer tutarı, kendi adına başka bankalara yolladığı düzenli EFT tutarı, yollanan paranın açıklama kısmında (BIRIKIM) veya (MAAS) yazması durumu, vadesiz, vadeli ve yatırım hesapları para miktarı, kredi ürünlerinin KKB sorgusundaki toplam, maksimum, ortalama periyot bazlı değeri,

müşterinin oturduğu mahallenin ortalama geliri, üniversite ve lise mezun oranı gibi kısıtlar oluşturulmuştur. Oracle SQL deki asıl amaç varlığını bizim bankamızda tutan müşteri popülasyonunu bulmaktır. Bunun sonucunda modelin öğrenme aşaması için hedef kitle belirlenmiştir. Birden fazla makine öğrenmesi algoritması kullanılarak, en az hata oranlı, en iyi sonuç veren model seçilmiştir. Seçilen model XGBoost modeli olmuştur. XGBoost modeli, hedef olmayan kitle üzerinden skorlanmıştır. Sonuç olarak, müşterinin profili, yaptığı banka hareketleri sayesinde %17 hata oranıyla tahmin edilebilmiştir. %17 hata oranı, PFA miktarı 50.000 den fazla tahmin edilen müşteri için hesaplanmıştır. Çünkü projenin genel amacı varlığı yüksek banka müşterisine, banka ürünlerini pazarlamaktır.

Anahtar Kelimeler: Bankacılık, finans, KKB, veri bilimi, makine öğrenmesi, yapay zeka, tahminsel modelleme, optimizasyon, XGBoost, GLM, GBM, RF, SQL, R

ABSTRACT

ASSET ESTIMATION OF BANK CUSTOMERS VIA MACHINE LEARNING APPROACH

Belce KAYA

Department of Industrial Engineering

Undergraduate Thesis

Adviser: Assos. Prof. NEZIR AYDIN

Nowadays, the world of technology is progressing rapidly. Almost all sectors are talking about big data. Big data is a very popular concept today and is interpreted as the beginning of a new era. Along with the big data, while a big transformation took place in the world, the point of view of the institutions and organizations to the data and its benefit from the data came to a different point. Regardless of the sector, most companies have millions, billions or even trillions of data. In the banking sector, in particular, there is an enormous amount of data that will provide significant benefits when processed. However, it is quite complicated and difficult to organize this big data and to create good projects. Fortunately, the method of gaining, using and making inferences about big data has become popular.

The aim of this project is to estimate the total assets by using the customer data and KKB data in the database owned by the bank. While doing this, Oracle SQL program is used for data editing, analysis and cleaning. Here, model variables are obtained by giving the necessary constraints. Generally, these constraints include the customer's credit products in the bank, the KKB query of the credit products in the last 6 months, the amount of regular money transfer within the bank made by the customer, the amount of the EFT sent to other banks on his/her behalf, (BIRIKIM) or (MAAS) description in money transfer, demand deposit, time deposit and investment accounts money amount, total, maximum, average value of loan products period-based in the KKB query,

average income of customer's neighborhood, university and high school graduation rate. The main goal in Oracle SQL is to find the customer population that keeps its wealth in our bank. As a result, the target group was determined for the learning stage of the model. Using multiple machine learning algorithms, the model with the lowest error rate and the best result was chosen. The selected model was the XGBoost model. The XGBoost model was scored on a non-target audience. As a result, the customer's asset can be estimated with an error rate of 17% thanks to the model. 17% error rate was obtained with customers that model predicted their PFA amount more than 50,000. Because the overall objective of the project is to market the Bank's products to a high-level bank customer.

Keywords: Banking, finance, KKB, data science, machine learning, marketing strategy with data science, artificial intelligence, predictive modeling, optimization, XGBoost, GLM, GBM, RF, SQL, R

1 INTRODUCTION

1.1 Literature Review

1.1.1 Predictive Modeling

Predictive modeling is a process that uses data and statistics to predict outcomes with data models. Predictive modeling is also often referred to as: Predictive analytics, predictive analysis and machine learning. Predictive modeling is useful because it gives accurate insight into any question and allows users to create forecasts. To maintain a competitive advantage, it is critical to have insight into future events and outcomes that challenge key assumptions. [1] Analytics professionals often use data from the following sources to feed predictive models:

- Transaction data
- CRM data
- Customer service data
- Survey or polling data
- Digital marketing and advertising data
- Economic data
- Demographic data
- Machine-generated data (for example, telemetric data or data from sensors)
- Geographical data
- Web traffic data

Also, BI tools provide insights in the form of dashboards, visualizations, and reports. A process should be put in place to ensure continued improvement. Important things to consider when integrating predictive models into business practices include:

- Benchmark analysis
- Data-gathering
- Data-cleansing
- Analysis
- Evaluating goals and KPIs
- Creating action plans based on analysis
- Executing on plans
- Streamlining processes

1.1.1.1 Predictive Modeling in Finance and Banking

Predictive modeling is one such AI application that could help banks to optimize their processes while simultaneously reducing cost and resources deployed. Nowadays,

customers interact with banks and financial institutions across several different channels which has led to an explosion in customer data being collected by these organizations. This data can be effectively leveraged using AI to gain insights on current and future customer behavior. There have been many instances of financial institutions instating innovation centers focused on artificial intelligence and machine learning to take advantage of their data 'plumes.' This history hints that banks and financial institutions might need to acquire the technological skills to create better products and customized experiences which can potentially increase revenues and decrease costs. [2]

1.1.2 Types of Predictive Models

Broadly speaking, predictive models fall into two categories: parametric and non-parametric. The essential difference is that parametric models make more assumptions and more specific assumptions about the characteristics of the population used in creating the model. Specifically, some of the different types of predictive models are:

- Ordinary Least Squares
- Generalized Linear Models (GLM)
- Logistic Regression
- Random Forests
- Decision Trees
- Neural Networks
- Multivariate Adaptive Regression Splines (MARS)
- Gradient Boosting Models (GBM)
- Extended Gradient Boosting Models (XGBoost)

Each of these types has a particular use and answers a specific question or uses a certain type of dataset. Despite the methodological and mathematical differences among the model types, the overall goal of each is similar: to predict future or unknown outcomes based on data about past outcomes. [1][4]

1.1.2.1 Generalized linear model

In statistics, the generalized linear model (GLM) is a flexible generalization of ordinary linear regression that allows for response variables that have error distribution models other than a normal distribution. The GLM generalizes linear regression by allowing the linear model to be related to the response variable via a link function and by allowing the magnitude of the variance of each measurement to be a function of its predicted value. [2]

1.1.3 Ensemble Learning and Random Forests

An Ensemble method is a technique that combines the predictions from multiple machine learning algorithms together to make more accurate predictions than any individual model. Ensemble learning, in general, is a model that makes predictions based on a number of different models. By combining individual models, the ensemble model tends to be more flexible (less bias) and less data-sensitive (less variance). Two most popular ensemble methods are bagging and boosting.

1.1.4 Bootstrap Aggregation (Bagging) and Pasting

Training a bunch of individual models in a parallel way. The approach is to use the same training algorithms for every predictor and train them on different random

subsets of training set. When sampling is performed with replacement, this method is called “bagging”. (https://homl.info/20). [21] When sampling is performed without replacement, it is called pasting. (https://homl.info/21). [22]

Bootstrap Aggregation (In statistics, resampling with replacement is called bootstrapping) or Bagging has two distinct features which define its training and prediction. For training it leverages a Bootstrap procedure to separate the training data into different random subsamples, which different iterations of the model use to train on. For prediction, a bagging classifier will use the prediction with the most votes from each model to produce its output and a bagging regression will take an average of all models to produce an output. Bagging is typically applied to high variance models such as Decision Trees and the Random Forest algorithm is a very close variation on bagging. [4]

1.1.5 Boosting: Training a bunch of individual models in a sequential way. Each individual model learns from mistakes made by the previous model.

Boosting (originally called hypothesis boosting) refers to any Ensemble method that can combine several weak learns into a strong learner. It is typically applied to trees. The general idea of most boosting methods is to train predictors sequentially, each trying to correct its predecessor. [23] More explicitly, a boosting algorithm adds iterations of the model sequentially, adjusting the weights of the weak learners along the way. This reduces bias from the model and typically improves accuracy. Popular boosting algos are AdaBoost, Gradient Tree Boosting, and XGBoost. [3]

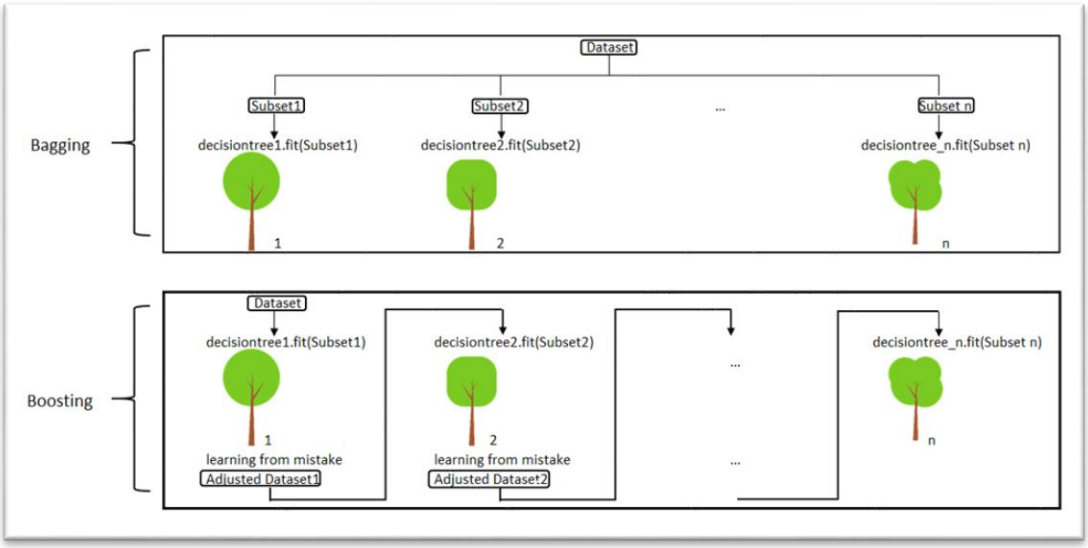


Figure 1-1 Bagging and Boosting Approaches

1.1.6 Basic Ensemble Learning Algorithms

1.1.6.1 Random Forest

Random forest is an ensemble model of Decision Trees, generally trained via the bagging method (or sometimes pasting).

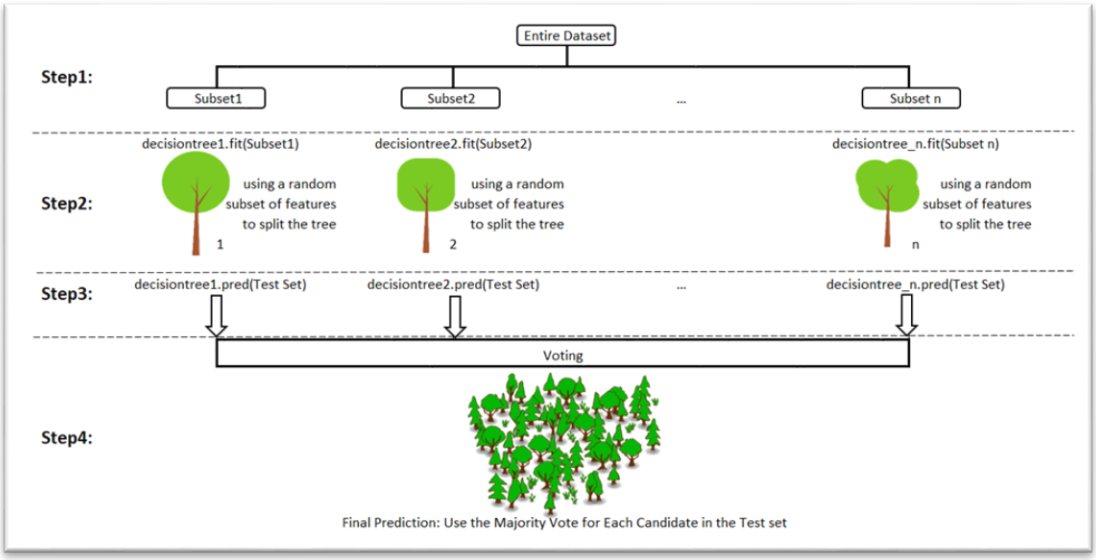


Figure 1-2 Random Forest Approach

- ✓ Step 1: Select n (e.g. 1000) random subsets from the training set
- ✓ Step 2: Train n (e.g. 1000) decision trees one random subset is used to train one decision tree. The optimal splits for each decision tree are based on a random subset of features (e.g. 10 features in total, randomly select 5 out of 10 features to split)
- ✓ Step 3: Each individual tree predicts the records/candidates in the test set, independently.
- ✓ Step 4: Make the final prediction

For each candidate in the test set, Random Forest uses the class (e.g. cat or dog) with the majority vote as this candidate's final prediction. [4]

1.1.6.2 Adaptive Boosting (AdaBoost)

AdaBoost is a boosting ensemble model and works especially well with the decision tree. Boosting model's key is learning from the previous mistakes, e.g. misclassification data points. AdaBoost learns from the mistakes by increasing the weight of misclassified data points.

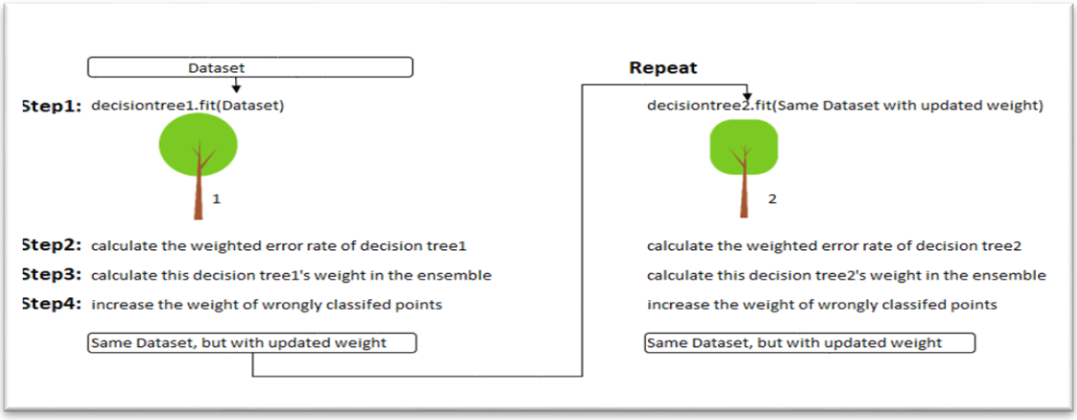


Figure 1-3 AdaBoost Approach

- ✓ Step 0: Initialize the weights of data points. if the training set has 100 data points, then each point's initial weight should be 1/100 = 0.01.
- ✓ Step 1: Train a decision tree
- ✓ Step 2: Calculate the weighted error rate (e) of the decision tree. The weighted error rate (e) is just how many wrong predictions out of total and you treat the wrong predictions differently based on its data point's weight. The higher the weight, the more the corresponding error will be weighted during the calculation of the (e).
- ✓ Step 3: Calculate this decision tree's weight in the ensemble.

The weight of this tree = learning rate * log ((1 — e) / e)

- the higher weighted error rate of a tree, the less decision power the tree will be given during the later voting

- the lower weighted error rate of a tree, the higher decision power the tree will be given during the later voting

- ✓ Step 4: Update weights of wrongly classified points

The weight of each data point:

- if the model got this data point correct, the weight stays the same

- if the model got this data point wrong, the new weight of this point = old weight * np.exp(weight of this tree)

The higher the weight of the tree (more accurate this tree performs), the more boost (importance) the misclassified data point by this tree will get. The weights of the data points are normalized after all the misclassified points are updated.

- ✓ Step 5: Repeat Step 1(until the number of trees we set to train is reached)
- ✓ Step 6: Make the final prediction

The AdaBoost makes a new prediction by adding up the weight (of each tree) multiply the prediction (of each tree). Obviously, the tree with higher weight will have more power of influence the final decision. [4]

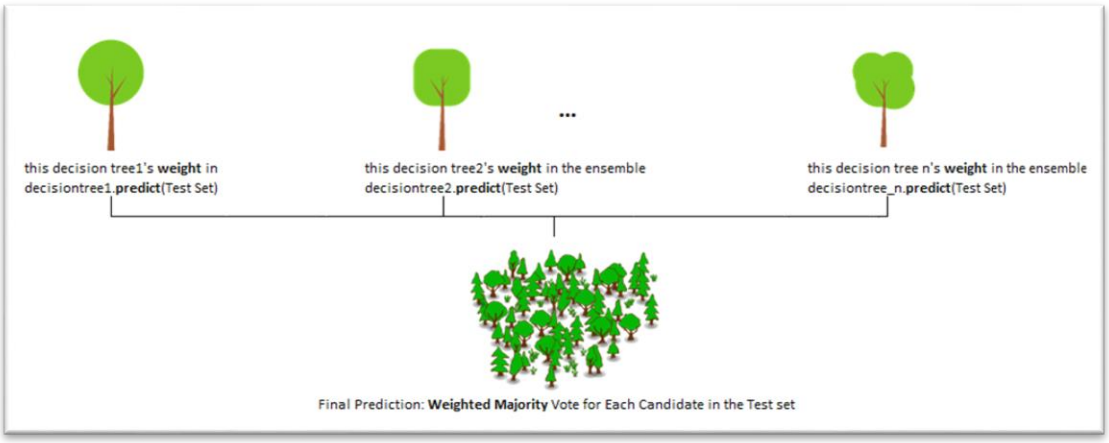


Figure 1-4 AdaBoost Adaptation Process

1.1.6.3 Gradient Boosting

Gradient boosting is another boosting model. As one may remember, boosting model's key is learning from the previous mistakes. Gradient Boosting learns from the residual error directly, rather than update the weights of data points.

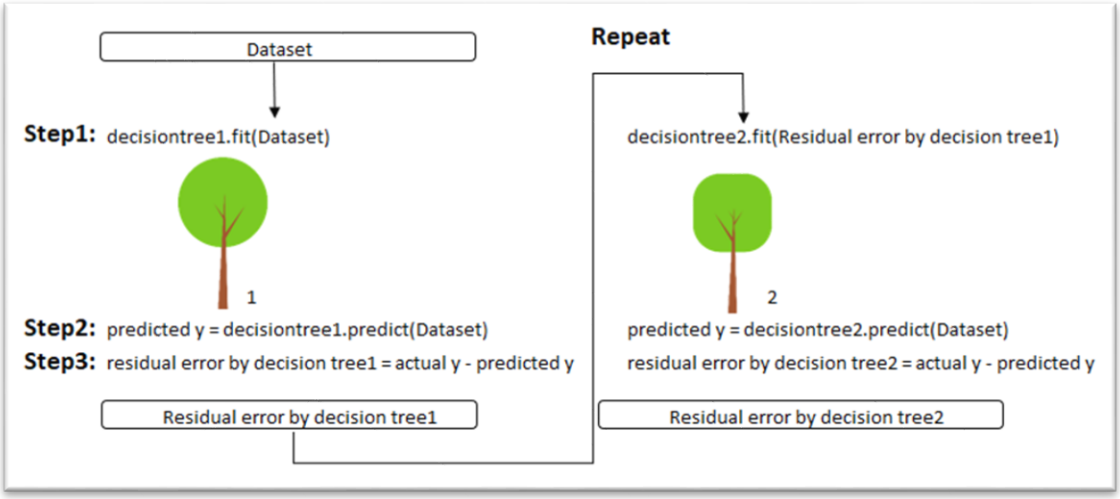


Figure 1-5 Gradient Boosting Approach

- ✓ Step 1: Train a decision tree
- ✓ Step 2: Apply the decision tree just trained to predict
- ✓ Step 3: Calculate the residual of this decision tree, Save residual errors as the new y
- ✓ Step 4: Repeat Step 1 (until the number of trees we set to train is reached)
- ✓ Step 5: Make the final prediction

The Gradient Boosting makes a new prediction by simply adding up the predictions (of all trees). [4]

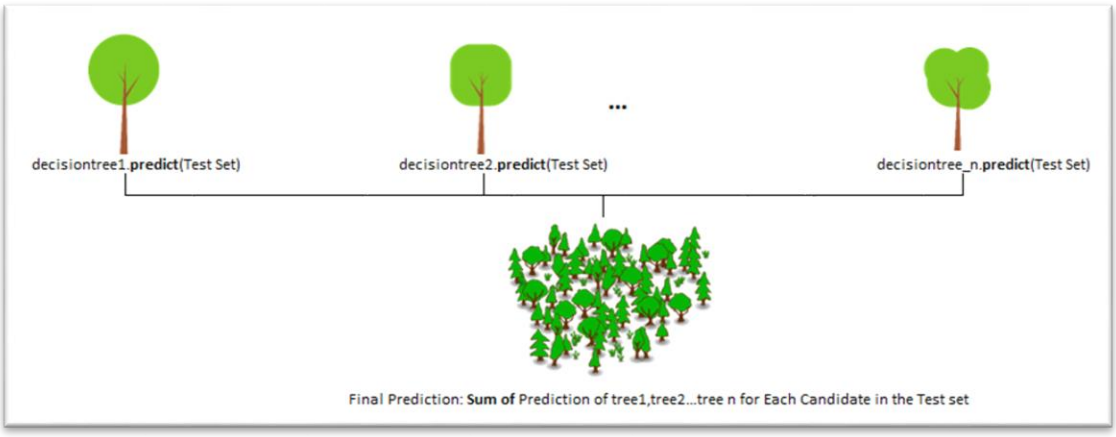


Figure 1-6 Gradient Boosting Adaptation Process

1.1.6.4 Extreme Gradient Boosting (XGBoost)

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree-based algorithms are considered best-in-class right now. [5]

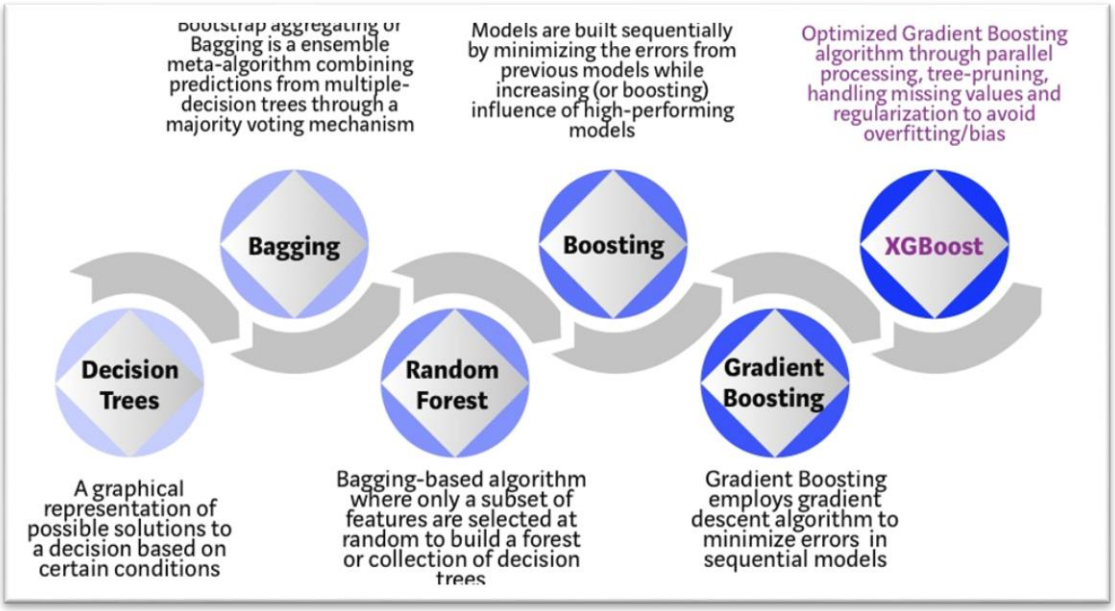


Figure 1-7 Evolution of Tree-based Algos

1.1.6.4.1 XGBoost Advantages

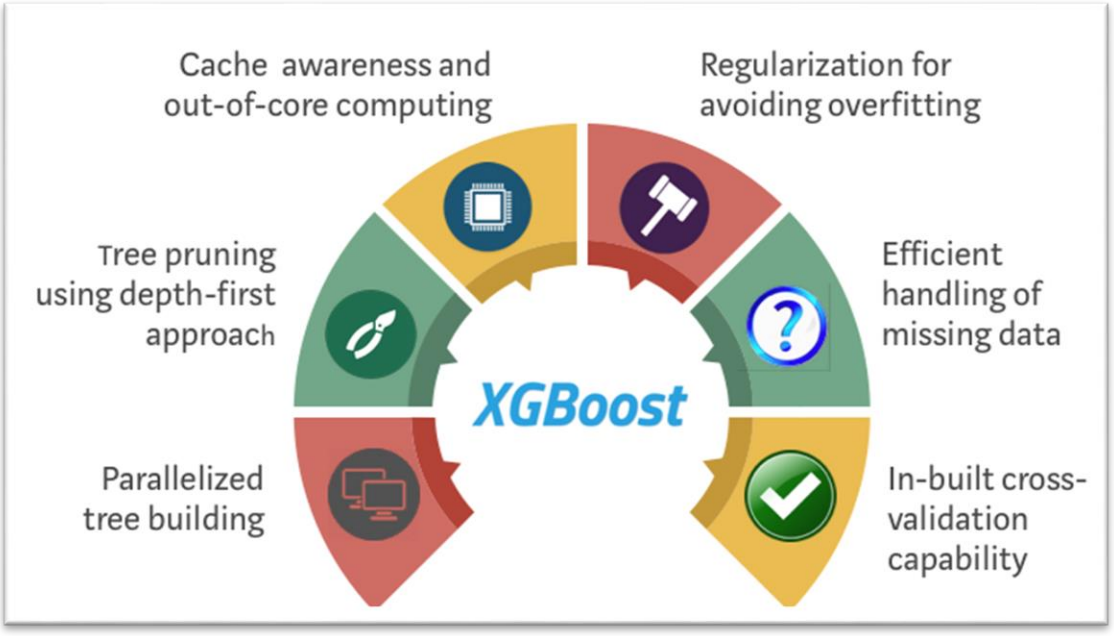


Figure 1-8 XGBoost Advantages Schema

Regularization for Avoiding Overfitting:

- ✓ Standard GBM implementation has no regularization like XGBoost, therefore XGBoost algorithm also helps to reduce overfitting. In fact, XGBoost is also known as a 'regularized boosting' technique.

Parallelized Tree Building:

- ✓ XGBoost approaches the process of sequential tree building using parallelized implementation. This is possible due to the interchangeable nature of loops used for building base learners; the outer loop that enumerates the leaf nodes of a tree, and the second inner loop that calculates the features. This nesting of loops limits parallelization because without completing the inner loop (more computationally demanding of the two), the outer loop cannot be started. Therefore, to improve run time, the order of loops is interchanged using initialization through a global scan of all instances and sorting using parallel threads. This switch improves algorithmic performance by offsetting any parallelization overheads in computation. [5]

High Flexibility:

- ✓ XGBoost allows users to define custom optimization objectives and evaluation criteria.

Efficient Handling of Missing Data:

- ✓ XGBoost has an in-built routine to handle missing values. The user is required to supply a different value than other observations and pass that as a parameter. XGBoost tries different things as it encounters a missing value on each node and learns which path to take for missing values in future.

Tree Pruning Using Depth-First Approach:

- ✓ A GBM would stop splitting a node when it encounters a negative loss in the split. Thus, it is more of a greedy algorithm. XGBoost on the other hand make splits up to the max_depth specified and then start pruning the tree backwards and remove splits beyond which there is no positive gain.
- ✓ Another advantage is that sometimes a split of negative loss says -2 may be followed by a split of positive loss +10. GBM would stop as it encounters -2. But XGBoost will go deeper and it will see a combined effect of +8 of the splits and keep both.

In-built Cross Validation Capability

- ✓ XGBoost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run. This is unlike GBM where we have to run a grid-search and only a limited value can be tested.

Continue on Existing Model:

- ✓ User can start training an XGBoost model from its last iteration of previous run. This can be of significant advantage in certain specific applications. [6]

1.1.7 Doing XGBoost Hyper-parameter Tuning

A hyperparameter is a parameter whose value is set before the learning process begins. Hyperparameter tuning is choosing a set of optimal hyperparameters for a learning algorithm. Four general purpose discrete optimization algorithms aimed at search for the optimal hyper-parameter combination: grid-search, gradient descent, coordinate descent and genetic algorithms.

The most powerful ML algorithms are famous for picking up patterns and regularities in the data by automatically tuning thousands (or even millions) of so-called “learnable” parameters. For instance, in tree-based models (decision trees, random forests, XGBoost), the learnable parameters are the choice of decision variables at each node and the numeric thresholds used to decide whether to take the left or right branch when generating predictions. In neural networks, the learnable parameters are the weights used on each connection to amplify or negate the activation from a previous layer into the next one.

The more flexible and powerful an algorithm is, the more design decisions and adjustable hyper-parameters it will have. These are parameters specified by “hand” to the algorithm and fixed throughout a training pass. Further, the algorithm typically does not include any logic to optimize them. In tree-based models, hyper-parameters include things like the maximum depth of the tree, the number of trees to grow, the number of variables to consider when building each tree, the minimum number of samples on a leaf, the fraction of observations used to build a tree, and a few others. For neural networks, the list includes the number of hidden layers, the size (and shape) of each layer, the choice of activation function, the drop-out rate and the L1/L2 regularization constants.

1.1.7.1 L1/L2 Regularization

A regression model that uses L1 regularization technique is called “Lasso Regression” and model which uses L2 is called “Ridge Regression”. The key

difference between these techniques is penalty term that Lasso shrinks the less important feature's coefficient to zero thus, removing some feature altogether. So, this works well for feature selection in case we have a huge number of features. [24]

1.1.7.1.1 L1 Norm or Lasso Regression

Least Absolute Shrinkage and Selection Operator) (LASSO) adds "absolute value of magnitude" of coefficient as penalty term to the loss function as seen square symbol in the below figure.

$$\hat{\beta}^{\text{lasso}} = \underset{\beta}{\operatorname{argmin}} \left\{ \frac{1}{2} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

Figure 1-9 L1 Norm Formula

The aim is to avoid from over or underfitting situation with L1 penalty term. If this parameter is not put to the formula, it leads to overfitting or underfitting. Model should predict surrounding of the real point. Gradient Descent should try to travel to the global minimum indicated by green dot.

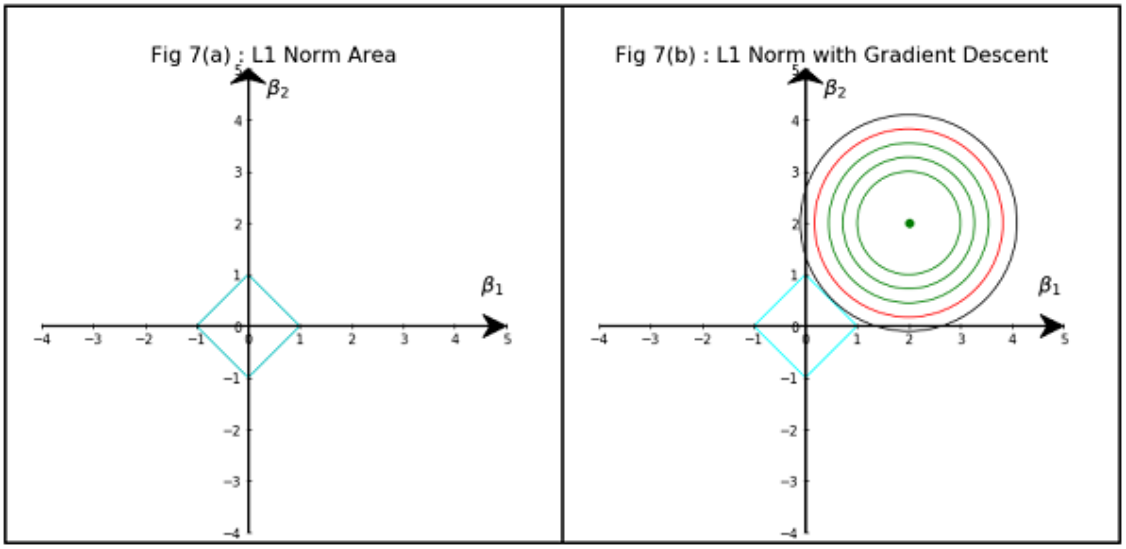


Figure 1-10 L1 Norm Area and with GD

1.1.7.1.2 L2 Norm or Ridge Regression

L2 Norm is Euclidean distance norm of the form $|\beta 1|^2 + |\beta 2|^2$. It means that L2 regularization method uses square magnitude of coefficient as penalty term to the loss function. [24]

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

Figure 1-11 L2 Norm Formula

As seen in the above figure, junction point represents the overfit point. If lambda value is set to zero, the model variable will be evaporated and there will no error rate to find the global optimal. It means model is overfitted. However, if lambda is very large then it will add too much weight and it will lead to underfitting. [25]

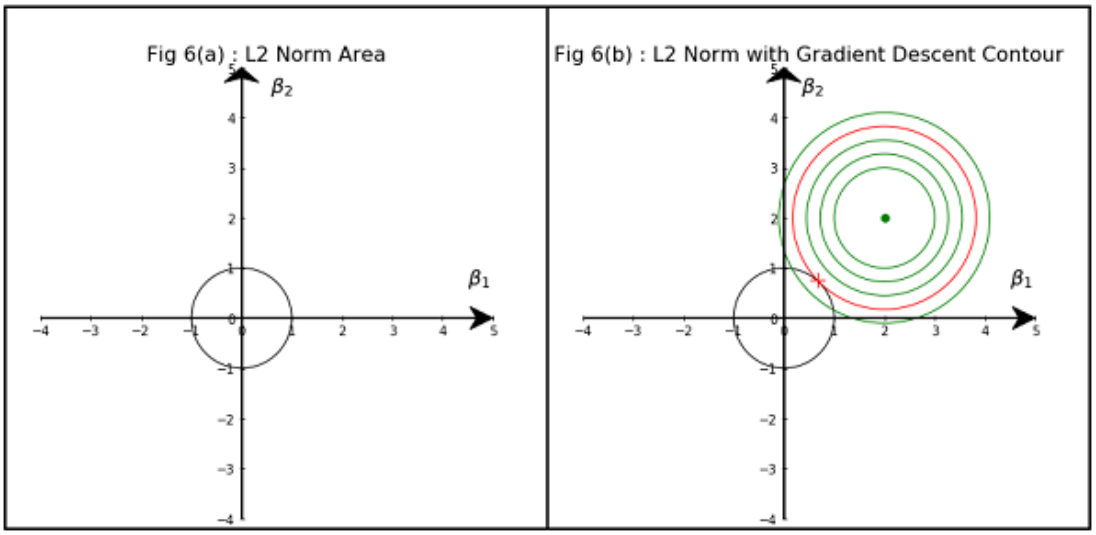


Figure 1-12 L2 Norm Area and with GD

As a result, regularization is a penalty scoring method. It protects the model from learning too much or too little. We expect that the data predicted by the model is true around the actual data in the model. It should not be over full or far away.

1.1.8 Introducing the Hyper-Parameter Grid

One important thing to note about hyper-parameters is that, often, they take on discrete values, with notable exceptions being things like drop-out rates or regularization constants. Thus, for practical reasons and to avoid the complexities involved in doing hybrid continuous-discrete optimization, most approaches to hyper-parameter tuning start off by discretizing the ranges of all hyper-parameters in question. For example, for our XGBoost experiments below we will fine-tune five hyperparameters. The ranges of possible values that we will consider for each are as follows:

```
{"learning_rate"    : [0.05,0.10,  0.15,  0.20,  0.25,  0.30 ] ,
 "max_depth"        : [3,   4,     5,     6,     8,     10,    12,   15],
 "min_child_weight" : [1,   3,     5,     7 ],
 "gamma"            : [ 0.0, 0.1,  0.2,   0.3,   0.4 ],
 "colsample_bytree" : [ 0.3, 0.4,  0.5,   0.7 ] }
```

For an explanation of what each of these means in the context of XGBoost.

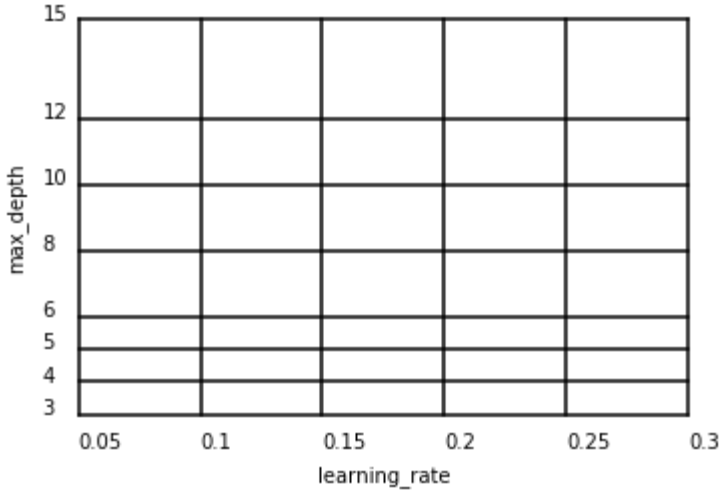


Figure 1-13 Section of Hyper-param Grid

A section of the hyper-param grid, showing only the first two variables (coordinate directions). Despite having limited the range for the (continuous) learning_rate hyper-parameter to only six values, that of max_depth to 8, and so forth, there are 6 x 8 x 4 x 5 x 4 = 3840 possible combinations of hyper parameters. This discrete subspace of all possible hyper-parameters is called the hyper-parameter grid. In what follows, the vector notation symbol h = [h0, h1, …, hp] to denote any such combination any point will be used in the grid.

1.1.9 Hyper-param Optimization Methods

1.1.9.1 Exhaustive Grid Search (GS)

Exhaustive grid search (GS) is an approach that scans the whole grid of hyper-param combinations h in some order, computes the cross-validation loss for each one and finds the optimal h in this manner. Generally, the use of lexicographic ordering, that is, the dictionary order imposed on hyper-param vectors, is discouraged and a different order should be considered. The reason is that with lexicographic ordering there is a high chance that the search will focus on a rather uninteresting part of the search space for a rather long time. An interesting alternative is scanning the whole grid in a fully randomized way that is, according to a random permutation of the whole grid. With this type of search, it is likely that one encounters close to optimal regions of the hyper-param space early on.

1.1.9.2 Gradient Descent

Gradient Descent is a very generic optimization algorithm capable of finding optimal solutions to a wide range of problems. The general idea of Gradient Descent is to tweak parameters iteratively in order to minimize a cost function. It measures the local gradient of the error function with regards to the parameter vector θ, and it goes in the direction of descending gradient. Once the gradient is zero, minimum point has been reached.

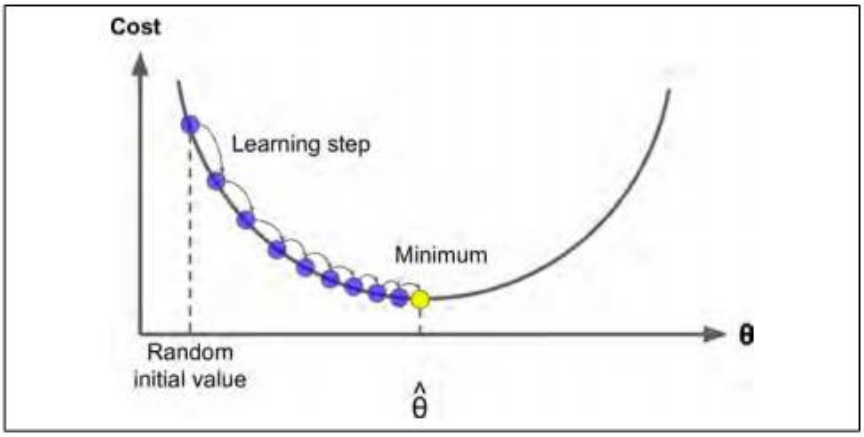


Figure 1-14 Gradient Descent

An important parameter in Gradient Descent is the size of the steps, determined by the learning rate hyperparameter. If the learning rate is too small, then the algorithm will have to go through many iterations to converge, which will take a long time.

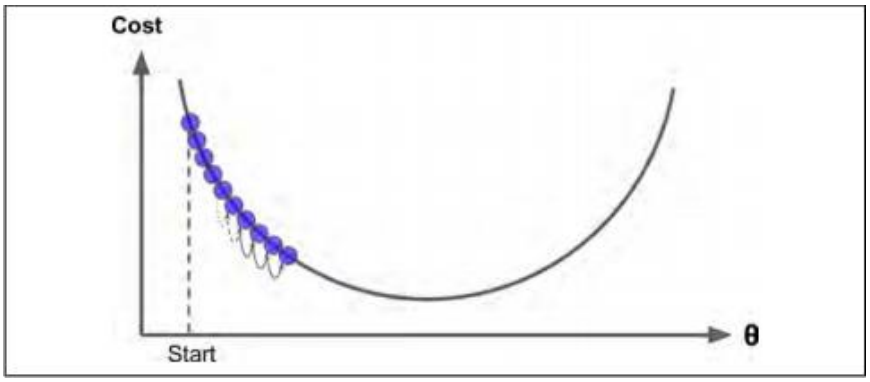


Figure 1-15 Too Small Learning Rate

On the other hand, if the learning rate is too high. This might make the algorithm diverge, with larger and larger values, failing to find a good solution.

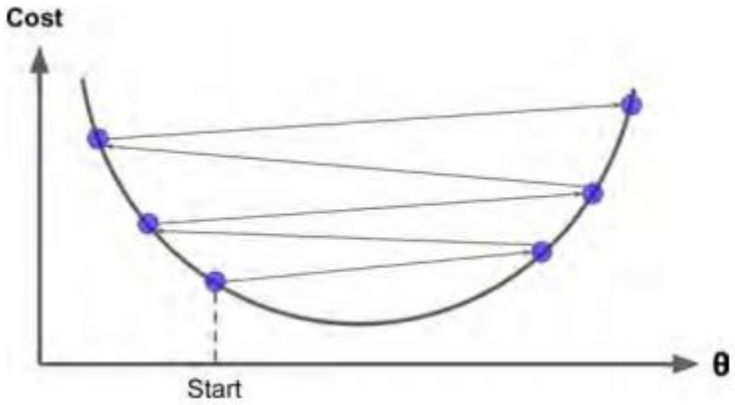


Figure 1-16 Too Large Learning Rate

As seen in the below figure, if the random initialization starts the algorithm on the left, then it will converge to a local mini- mum, which is not as good as the global minimum.

If it starts on the right, then it will take a very long time to cross the plateau, and if learning is stopped too early, the global minimum will never be reached. [23]

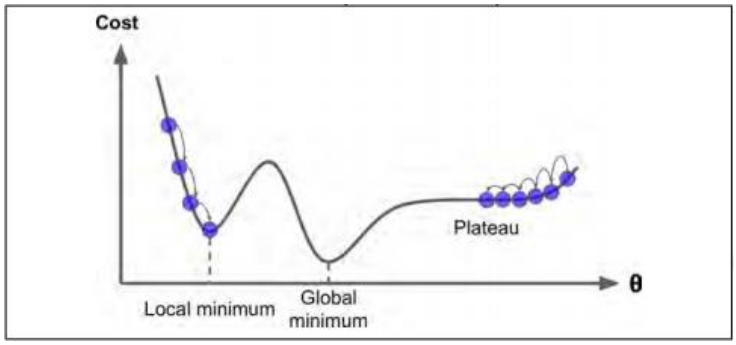


Figure 1-17 Gradient Descent Pitfalls

In the below figure, on the left, the learning rate is too low: the algorithm will eventually reach the solution, but it will take a long time. In the middle, the learning rate looks pretty good: in just a few iterations, it has already converged to the solution. On the right, the learning rate is too high: the algorithm diverges, jumping all over the place and actually getting further and further away from the solution at every step. To find a good learning rate, grid search can be used. However, you may want to limit the number of iterations so that grid search can eliminate models that take too long to converge.

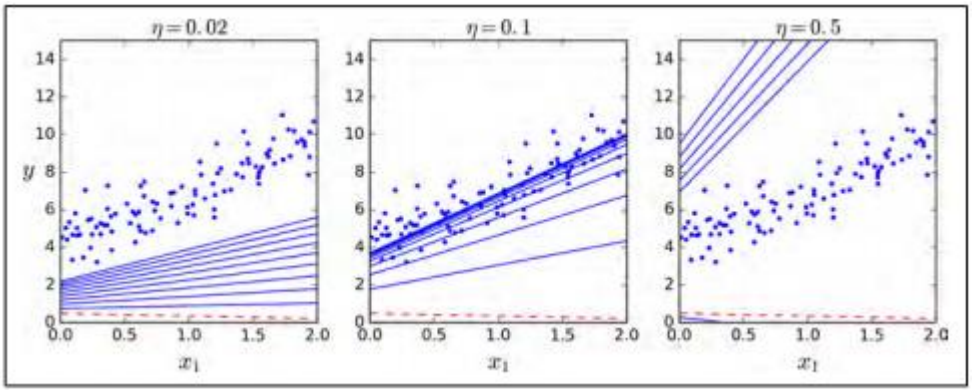


Figure 1-18 Gradient Descent with Various Learning Rate

1.1.9.3 Coordinate Descent (CD)

Coordinate descent (CD) is one of the simplest optimization algorithms. It's an iterative algorithm, similar to gradient descent, but even simpler. The basic idea is that, at each iteration, only one of the coordinate directions of the search vector h is altered. To pick which one examined each coordinate direction turn and minimize the objective function by varying that coordinate and leaving all the other constant. Then we pick the direction that yields the most improvement. The algorithm stops when none of the directions yields any improvement. The main advantage of this simple approach is that it is easily adaptable to the fully discrete case for which the arbitrary directions given by gradient descent cannot be easily used.

1.1.9.4 Genetic algorithm

Figure 1-19 Genetic Algorithm

Genetic algorithms (GAs) are a whole class of optimization algorithms of rather general applicability and are particularly well adapted for high-dimensional discrete search spaces. A genetic algorithm tries to mimic nature by simulating a population of feasible solutions to an optimization problem as they evolve through several generations and survival of the fittest is enforced. There are two basic mechanisms for generating a new generation from the previous one. One is crossbreeding in which two individuals (feasible solutions) are combined to produce two offspring. The other one is mutation. With a given mutation probability, any individual can change any of their params to another valid value. Survival of the fittest is enforced by letting fitter individuals crossbreed with higher probability than less fit individuals. The fitness of an individual is of course the negative of the loss function.

1.1.10 Measuring Model Performance Methods

1.1.11 Error

Basically, error is the forecast minus the demand. If the forecast overshoots the demand, the error will be positive and if the forecast undershoots the demand, then the error will be negative.

$$e_t = f_t - d_t$$

Figure 1-20 Error Formula

1.1.12 Bias

The bias is defined as the average error.

$$bias = \frac{1}{n}\sum_{n} e_t$$

Figure 1-21 Bias Formula

Where ‘n’ is the number of historical periods where you have both a forecast and a demand. As a positive error on one item can offset a negative error on another item, a forecast model can achieve a very low bias and not be precise at the same time. Obviously, the bias alone won’t be enough to evaluate the forecast precision. But a highly biased forecast is already an indication that something is wrong in the model. [7]

1.1.13 MAPE

The Mean Absolute Percentage Error (MAPE) is one of the most commonly used KPIs to measure forecast accuracy. MAPE is the sum of the individual absolute errors divided by the demand (each period separately). Actually, it is the average of the percentage errors.

$$MAPE = \frac{1}{n}\sum \frac{|e_t|}{d_t}$$

Figure 1-22 MAPE Formula

MAPE is a really strange forecast KPI. As seen in the formula, MAPE divides each error individually by the demand. High errors during low-demand periods will have a major impact on MAPE. Due to this, optimizing MAPE will result in a strange forecast that will most likely undershoot the demand. [7]

1.1.14 MAE

The Mean Absolute Error (MAE) is a very good KPI to measure forecast accuracy. As the name implies, it is the mean of the absolute error. [7]

$$MAE = \frac{1}{n}\sum |e_t|$$

Figure 1-23 MAE Formula

One of the first issues of this KPI is that it is not scaled to the average demand. It is common to divide MAE by the average demand to get a %:

$$MAE\% = \frac{\frac{1}{n}\sum |e_t|}{\frac{1}{n}\sum d_t} = \frac{\sum |e_t|}{\sum d_t}$$

Figure 1-24 MAE% Formula

1.1.15 RMSE

The Root Mean Squared Error (RMSE) is a strange KPI. It is defined as the square root of the average squared error. [7]

$$RMSE = \sqrt{\frac{1}{n}\sum e_t^2}$$

Figure 1-25 RMSE Formula

Just as for MAE, RMSE is not scaled to the demand. RMSE% is defined such as,

$$RMSE\% = \frac{\sqrt{\frac{1}{n}\sum e_t^2}}{\frac{\sum d}{n}}$$

Figure 1-26 RMSE% Formula

Actually, many algorithms (especially for machine learning) are based on the Mean Squared Error (MSE), which is directly related to RMSE.

$$MSE = \frac{1}{n}\sum e_t^2$$

Figure 1-27 MSE Formula

MSE is used by many algorithms as it is faster to compute and easier to manipulate than RMSE. But it is not scaled to the original error (as the error is squared), resulting in a KPI that it cannot really relate to the original demand scale. Therefore, MSE will not be used to evaluate forecast models. [7]

1.2 Purpose of the Thesis

The aim of the project is to predict the asset amount of the bank customer with the available data. In that way, it is aimed to make the right campaign for the right customer. In order to establish this predictive model, it is aimed to use machine learning algorithms and to make the model in a software program that supports these algorithms. As mentioned in the literature research, there are various algorithms suitable for the predictive model. It is aimed to choose the most popular algorithm which gives the most optimal result. As a result, the main objective is to identify the customers that has high predicted asset amount with low error rates in order to sell the products of the bank and increase the profit share of the bank.

1.3 Hypothesis

After literature review, many machine learning algorithms that can be used for predictive modeling were found and searched in detail, their approach, pros and cons. Then, it can be seen that XGBoost algorithm is the best one with tabular (structured) data. The other tree-based algorithms are not stronger and faster than XGBoost. Also, for comparison, neural network and gradient boosting algorithms are better. In prediction problems involving unstructured data (images, text, etc.) artificial neural

networks tend to outperform all other algorithms or frameworks. However, project data type is tabular data as mentioned before. Therefore, tree-based algorithms will be better for this case. To reach the optimal solution XGBoost hyper parameters will be important. The parameters can be changed wisely according to the model data. For the project, grid search algorithm will be useful, meaningful and at the same time faster. Thanks to grid search approach, many combinations can be tried at the same time and compared as well to reach the optimal model parameters. As a result, tree-based algorithms should be used and additionally, linear regression can be tested. On the

2 METHODOLOGY

2.1 Methodological Approach

First of all, main problem was defined in detail. Business problem was tried to understand. Problem was occurring because of wrong marketing strategies. There was a mass marketing strategy that goal have been sending or calling all customers. In a meeting with data science department, marketing talked about knowing high PFA amount customers could increase the bank's profit and efficiency. With regard to this, data that may have a relationship with the project tried to analyze and generate features of the project. Creation variables and data acquisition for getting target population was held in Oracle SQL data warehouse. In SQL part, number of customers was checked whether there was a duplication or not when all tables merged. Data was analyzed with estimated income when it was necessary to check number of customers' distribution makes sense or not in Excel. After that, prepared data were imported to R software, feature engineering, model training and evaluation processes were studied for all algorithms. At the end, performance of the model was scored with non-target audience. After that, project will be estimated the customer and marketing, or sales departments will try to reach those customers and to sell the bank's products to gain more profit.

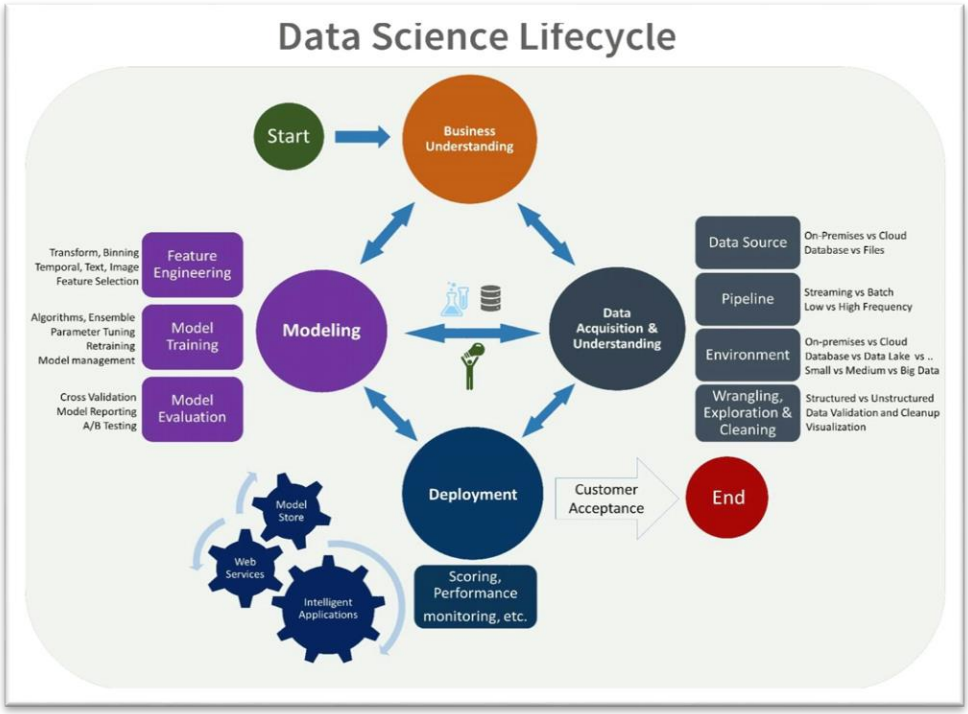


Figure 2-1 Data Science Methodology

2.2 Methods of Data Collection and Analysis

Data was stored in a database. Therefore, data collection process was managed in Oracle Database. To generate all variables that may be useful to build the model rightly, all of tables were created inside of Oracle Database environment. At the same time, data was checked that whether it makes sense or not by Excel environment. In excel, data that was created with SQL codes, was examined with Pivot tables, statistical tools, administrative reviews etc. especially for data visualization. Other interpretative information that may be a benefit for the project discussed in daily, weekly or monthly meetings with related departments.

2.3 Method of Modelling

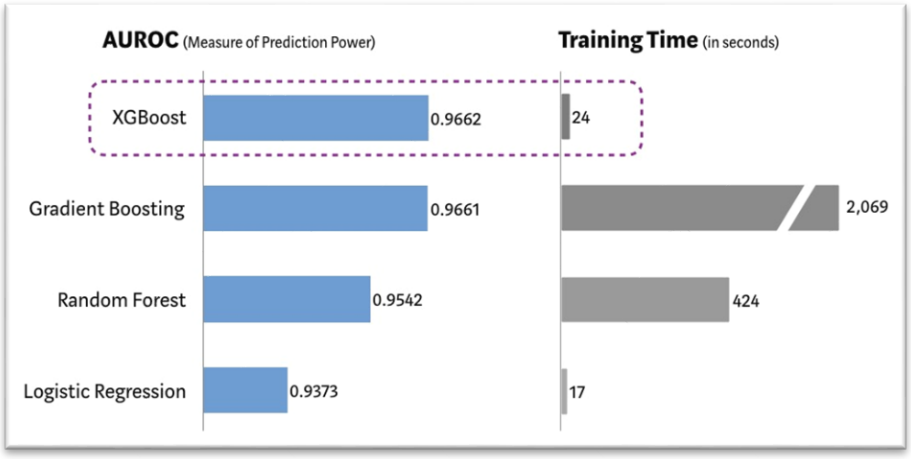
All machine learning, and evaluation of predictive models were performed within the R statistical environment. Our model needs to be robust to predict well in real world i.e. out of sample data or the data it has not seen during training (test data). The overfitting during training is what prevents it from being robust. Shrink the coefficients or weights of features in the model and getting rid of high degree polynomial features from the model. Dimension reduction was performed using feature selection (FS). Data was pre-processed by removing highly correlated features. (Pearson's $|r| > 0.80$; Corrplot library)

2.4 Evaluation and Justification of Methodological Choices

2.4.1 Comparison Between Boosting Algorithms and Other ML Algorithms

Neural networks and Genetic algorithms are naive approaches to imitate nature. They work well for a class of problem, but they do have various hurdles such as overfitting, local minima, vanishing gradient and much more. There is another set of algorithms that do not get much recognition compared to others and they are boosting algorithms.

XGBoost belongs to a family of boosting algorithms that convert weak learners into strong learners. A weak learner is one which is slightly better than random guessing. Boosting is a sequential process; i.e., trees are grown using the information from a previously grown tree one after the other. This process slowly learns from data and tries to improve its prediction in subsequent iterations. [26]



3 DATA ACQUISITION & UNDERSTANDING

3.1 Using Oracle SQL Developer to Get Data

3.1.1 Oracle SQL Developer

Oracle SQL Developer is a free, integrated development environment that simplifies the development and management of Oracle Database in both traditional and cloud deployments. SQL Developer offers complete end-to-end development of your PL/SQL applications, a worksheet for running queries and scripts, a DBA(Data Base Administrative) console for managing the database, a reports interface, a complete data modeling solution, and a migration platform for moving 3rd party databases to Oracle.

3.1.2 Creating Tables step For Generating Variables

Figure 3-1 Project Variables Schema

For this process, tables that are needed by the project are first defined and called. Then, the project variables are generated with SQL codes.

First of all, this process aim is to define target customers that they are known, loyal customers. We should clearly understand that they are holding their money and also using more our bank's products. (at least one product is necessary to obtain whether

the customer is active or not for our bank.) In addition to this, there are many significant terminologies as follows:

Asset: Anything of monetary value owned by a person or business. In this case, Turkish lira, demand deposit, foreign currency (FX), time deposit, Investment, Bond, repo etc. In codes all asset terms mentioned like that: DDEP, TDEP, FON, FON_PIONEER, BOND_TL, BOND_YP, REPO_TL, HISSE_YKY, HISSE_BANKA

Debt: A sum of money that is owed or due. In this case, Credit card and Credits.

Period that used for the project: 1-year period between 2017 and 2018 is used for this project. Because there were many data to train the machine. The period of study is the period from 2017 to 2018 at 6-month intervals as 2017-12, 2018-06 and 2018-11.

In this project two targets were defined as Regular Employees & Retired and Self-Employed segments. In the following header, Regular Employees & Retired target segment target will be defined. All SQL codes can be seen in Appendix-SQL Codes.

3.1.2.1 PFA Determination Tables

USED TABLES:

-EDW.CON_BALANCE: This table, which is given all information about EOD, MTD balance based on customer ID.

-CMN_SOURCE_SYSTEM_LKP: This table, which is given all information about source_system_id and source_system_code. It means that every process that we can do in a bank has a source system id and code. For example; Time Deposit, Demand Deposit, EFT, credit card etc. The aim of using this table is to connect based on source system id with EDW.CON_BALANCE table.

-DM.CNF_CUSTOMER_HST: This table includes lots of information like customer SBU code, customer Branch ID etc. In this case as customer SBU, retail customers SBU codes are used.

*Used Customer SBU's: 410, 430, 431, 411, 311

-EDW.PTY_CUSTOMER_TP_LKP: This table shows the customer type id and codes. The aim of using the table is to remove bank's employees. Because analyzing bank's employees is forbidden. This project aim is to do marketing and other gaining profit ways. Therefore, except employee's customer IDs, other customers are examined in detail.

-EDW.CHN_BRANCH: This table is used to remove unnecessary branch type which is ('1111').

First, the sum of the investment balances is specified in the asset definition of the customers as follows in codes. The aim of this tables is to find the sum of balance of all investment types like both "end of day (EOD)" and "month to date (MTD)" balances. On the other hand, the most fundamental issue is to sum all balance types in TRY (Turkish liras currency). When tables are analyzed, the version of TRY can be seen. So, this statement must be identified inside of the case when statement like "CURRENCY_CODE='TRY'. Also, analyzing bank's employees is forbidden. When this study was examined, employees' data were not used in this project.

ASSET_TL_FX_DDEP_PFA_201811/201806/201712

While preparing all of the above tables, 12 months before the date of the period, a 12-monthly review was made. The created tables were then merged into a single table.YKB_ASSET_PFA_1

These merged tables are, for example 6 months prior to 201712, examined and merged. YKB_ASSET_PFA_2. The reason for creating this table is the comparison of whether the 12-month or 6-month examination provides more useful information.

Furthermore, asset PFA additional tables were created as ASSET_PFA_ADDITIONAL_TABLE_L12M and ASSET_PFA_ADDITIONAL_TABLE_L6M. The aim of this step to calculate the minimum, maximum, average, median, standard deviation and the number of customer's PFA month which was used to find out how many months a customer has been used PFA based on both last 12- and 6-months examination period.

In addition to these statistics tables, ASSET_DDEP_ADDITIONAL_TABLE_L12M and ASSET_DDEP_ADDITIONAL_TABLE_L6M tables are created to calculate the minimum, maximum, average, median, standard deviation and the number of customer's PFA month which was used to find out how many months a customer has been used demand deposit based on both last 12- and 6-months examination period. The aim of this step to remove some customers that they have only demand deposit source in our bank. Also, to obtain exactly the maximum level of money that customer has. Because in the following target variables step, the calculation of demand deposit will be used in "If a customer has more than or equal to 1 million TRY, the customer must be taken to the target customers committee." condition.

After all these constraints and getting all the summation of every investment type of balance, all these PFA amounts will be merged into one table to use in the main data table which will be used for modelling process.

In conclusion, all of the above table were merged into YKB_ASSET_PFA_CALC based on party_id and year month. As you can see in creating merged table, all statistics calculations, customer IDs and periods were selected. Therefore, customer all PFA and demand deposit amount and number of month that customer has been used collected on this table. The table will be used later to create the all base population and also another generating variable.

3.1.2.2 Salary Tables

In this section, salary customers were examined under the employed & retired segment.

3.1.2.2.1 EMPLOYED SALARY CUSTOMERS:

Thanks to salary information, the customer ID and the average salary amount on a monthly and customer ID basis are obtained. In other words, the average amount of customer's salary, maximum payment month according to paid salaries and number of months that paid to customer were taken from used tables. The reason of this calculation is to use in this rule "customers with 6 distinct month payment in last 12 months" and "customers with at least 1-month payment within last 2 months.

Another important rule about salary transaction status is only paid and returned cases must be received. In this case ('1','5') codes removed from the transaction statuses.

Because thanks to these statuses, we can understand that customers got their salary regularly. It means that there is a transaction in customer's salary account.

Used tables:

1. EDW.PCO_SALARY_TRANSACTION: In this table, amount of transaction in TRY (transaction_amount_tl), customer ID (party_id), payment date (transaction_date) columns were used. The most important variable that created is AY(Month). This variable created with this following code. "TRUNC (PCO.TRANSACTION_DATE,'MM')". Thanks to this syntax, the data date can be changed from the transaction date whenever it was to the first day of data date's month. For example, if a customer's transaction date is 19/03/2018 then with TRUNC function the new date becomes 01/03/2018. In other words, all the dates in the calculation of salary section were assigned to the beginning of the month and analyzed in that way.
2. EDW.PCO_TRANSACTION_STAT_LKP: This table includes information about the status of the payment collection process. The reason to connect to this table is to access transaction status id information. The most important thing that should be considered is some corporations which are related with bank's subsidiary corporations are removed after discussing with all work-units.
3. EDW.PCO_CORPORATION: This table includes corporation ID and codes which are the name of the corporation.
4. EDW.PCO_PAYMENT_TP_LKP: This table includes ID, code and explanation of salary and social security institution (SSI) payment types. The aim of using this table is to select the payment type codes. During selection process, selected payment types that seen in the SQL code were taken as work units' last decision.

After connecting to the above tables, YKB_YKB_ASSET_SALARY_ALL table was created.

3.1.2.2.2 PENSION:

For gathering average retired salary amount and number of months that retired customers received their salary regularly or not can be analyzed and those customer IDs can be pulled from all of the above tables.

In this step, as a payment type code, only c-pension is chosen and also connected to another table (EDW.PCO_SGK_SALARY_TRANSACTION). The reason why connected that table is that some retired customers receive their pension from social security institutions. After connecting to the above tables, YKB_ASSET_PENSION table was created.

On the other hand, 6-months period was taken into considered as another way to analyze whether there is a big difference or not.

3.1.2.3 KKB (Credit Risk Bureau)

3.1.2.3.1 KKB Definition and Goal

Kredi Kayıt Bürosu (KKB) (credit risk bureau) was founded by nine leading banks in Turkey on April 11, 1995. As a highly regarded financial institution, KKB has a total of 156 members, including 44 banks, 61 factoring companies, 23 leasing companies, 4 insurance companies, 14 consumer financing companies, 7 asset management companies and 3 other types of companies. Pursuant to the Banking Law no. 5411 (art.

73/4), KKB was founded by at least five banks to facilitate the exchange of information and documents between the financial institutions. As per the same article of the relevant Law, the members of KKB have been sharing credit information of their customers since April 1999.

With Law no. 6111 issued on February 25, 2011, additional Article 1 and Provisional Article 28 were added to the Banking Law no. 5411. And as per the Additional Article 1, a Risk Center (RM) has been established within the organization of the Banks Association of Turkey (TBB) to collect the risk data of customers of credit institutions and other financial institutions, deemed fit by the Banking Regulation and Supervision Agency, and to share such data with these institutions, with natural or legal persons themselves or subject to prior consent thereof, private legal persons and third party natural persons. Upon transfer of the Risk Centralization Center within the organization of the Central Bank of Turkey (CBRT), the Risk Center of the Banks Association of Turkey started up operations on June 28, 2013. KKB conducts all operational and technical activities through its own organization as an agency of the Risk Center of TBB and provides data collection and sharing services to 180 financial institutions which are members of the Risk Center.

KKB offers its services not only to financial institutions, but also to individuals and the real sector through “Cheque Report”, “Risk Report” and “Electronic Report” systems launched in January 2013. As of September 2014, KKB gathered its services aimed at individuals and the real sector under the umbrella of Findeks, the consumer service platform of KKB. KKB continued to create added value primarily for the banking-finance industry, as well as the real sector, through sectoral collaborations alongside its new products and services launched in 2015. Ultimately, through the QR Code Cheque System – launched by KKB in 2015, which later became mandatory by law in 2016 and entered into force as of January 1, 2017 – an important step was taken for more transparent and secure commercial transactions. [9]

3.1.2.3.2 Data preparation for KKB Variables

KKB contains information related to the credit information received from all banks (credit card and credit information). It collects this information from all banks and collects it in a pool. Systematically processes maintained as follows: credit card and credit types are listed based on query IDs. The bank only sees its name. The names of other competing banks are invisible. The logic is as follows:

Table 3-1 KKB Definition and Understanding the Approach

QUERY_ID	CREDI TYPE	BANK
2	02	YELLOW BANK
2	23(CC)	YELLOW BANK
2	26	YELLOW BANK
2	23(CC)	X(INVISIBLE)

Here, a case of the same ID is given based on query ID. For example, this person has a credit card from both our bank and another bank. In this case, we do not see the name of the other bank in the query we run.

Besides, even if we cannot see the bank name of this customer in the X bank, we can obtain information such as how much the credit card limit is and when she/he pays her/his credit card expenses and debts. So, whether the customer pays his/her debt late or regularly information can get thanks to KKB.

- **USED TABLES AND JOINED TABLE:**

-ETL_CAD.INC_DATAPREP_KKB_OLD: This table was created for the project as a data preparation table. The aim was to take these parameters: CIF_ID as PARTY_ID, YEAR_MONTH, KKB_QUERY_DATE, KKB_WPS_L12M, KKB_CC_O_TOT_LIM, KKB_C_O_TOT_RISK, KKB_O_C_LAST_PROD_DATE, KKB_O_HL_CNT, KKB_O_HL_FINANCE_AMT and KKB_O_HL_BAL columns are selected based on project's period. KKB_O_C_LAST_PROD_DATE: the last opened product date.

-INC_DATAPREP_KKB_YKB_ONLY: The aim of creating this table is to find out the credit types from only our bank. This table is a flag table. The results of this table are 1 or 0. So, the table has a binary variable. In detail, if a customer has only one query related to our bank (in a word, the customer does not have any query from the other banks) in this table customer's value becomes 1. If the customer has a query also from another bank, the score becomes 0. Therefore, our goal is to get all of the customers who have 1. As a result of this, we can be sure that the customer is only using our bank and we can determine his/her income so accurately. Thanks to this variable, these customers can be our target customers.

-YY_KKB_YKB_ONLY_201811: This table was created to get only for 201811 period. Because of the database data it had to be created like that.

-YKB_ASSET_KKB_DATAPREP: Main merged table. After joined all of the above tables, PARTY_ID, YEAR_MONTH, KKB_QUERY_DATE, KKB_WPS_L12M, KKB_CC_O_TOT_LIM, KKB_C_O_TOT_RISK, KKB_O_C_LAST_PROD_DATE, KK_YKB_ONLY_FLAG, LOAN_YKB_ONLY_FLAG and YKB_ONLY_FLAG columns were selected for target table.

3.1.2.4 CREDIT CARD

The most significant role to analyze credit card data is that we can determine our customer's inflow with credit cards. In this sentence inflow means

To generate this credit card variable, ETL_CAD.INC_DATAPREP_CREDIT_CARD_HST and ETL_CAD.INC_DATAPREP_CREDIT_CARD_OLD tables were joined. CC_LIMIT_LM, CC_TOT_STAT_BAL_LM, PARTY_ID and YEAR_MONTH were pulled from joined table and YKB_ASSET_CC_DATAPREP table created.

- CC_LIMIT_LM: Customer Credit Card Limit Last Month. This is customer's limit of his/her credit card.

- CC_TOT_STAT_BAL_LM: Customer Credit Card Total Status Balance Last Month. This is customer's monthly bill (debt).

This variable will be used to generate primary card flag variable.

3.1.2.5 EFT (ELEKTRONIC FUNDS TRANSFER) / MONEY TRANSFER

In this section, customers that doing regular going and coming to EFT and HAVALE (money transfer) situations were examined in detail. As an output, another variable table created to be used later for creating target customers.

The idea of looking for money transfer is to determine customer's certain inflow and thanks to this guess customer's asset amount. We can be sure with these customers because they are almost just using our bank. This thing is so important for the determination of the target.

Another idea is to take EFT money transfers which is between two different banks and at the same time remove HAVALE money transfer type which is between the same bank's money flow.

- **USED TABLES:**

-EDW.TRX_ MONEY_TRANSFER_TRANSACTION (A): For this query, this table is the main one. The other tables are look-up tables that can be used for binary variables.

-EDW.TRX_MONEY_TRANSFER_TRX_TP_LKP(P), EDW.PCO_TRANSACTION_STAT_LKP(B), EDW.CMN_SOURCE_SYSTEM_LKP (S): The aim of using these tables is connecting with the above table on transaction type, source system, and transaction status IDs and also on transaction status code.

Using all of the above tables, many variables were created at the first inside table query. Here, the most important thing was to obtain between SENDER_PARTY_ID and RECEIVER_PARTY_ID relationship. Because we cannot take a customer who churns his/her money to another bank or another user of our bank. In EFT situation, if sender and receiver IDs match and in addition to this, source system must be outgoing EFT and old outgoing EFT ('EFT_GIDEN','ESKI_EFT_GIDEN'). In the meanwhile, the reason why used both old and current was that the system of the EFTs was different in the past. Database was arranged later. Therefore, both old and current money transfers needed to be considered in the Money Transfer Variable's code.

Table-A was analyzed and BIRIKIM_FLG and MAAS_FLG variables were created. As a result, YKB_ASSET_EFT_HAVALE merged.

3.1.2.6 Estimated Income

For estimated income variable,

This table already created for data preparation process, only joined process completed. The estimated income will be used as a predicted income later inside of the target table.

3.1.2.7 Customer Information

DATAPREP_CUSTOMER_INFO table was created to collect all necessary data for customer information as customer tenure, customer age etc. Afterwards, YKB_ASSET_CC_OWNERSHIP table was created based on party id and year month columns.

3.1.2.8 MEMZUC

3.1.2.8.1 MEMZUC Definition

In banking, Memzuc means a table in which the total limit-risks of companies with credit limits and risks are compiled and collected in detail.

The Central Bank receives monthly reports on credit limits and disbursements from financial institutions (deposit banks, participation banks, leasing and factoring companies). Each financial institution prepares this report on company basis and reports it to the Central Bank. The Central Bank compiles these reports, which are received by the Bank, and combines them to the use and information of financial institutions. In this way, the credit summary used by any company from all financial institutions for the month, new credit limits allocated, and default elements are determined. This report, which is defined as Memzuc records, can be followed only through a common portal - information systems accessible to financial institutions. Financial institutions monitor memzuc records and see the credibility of the related company not only in its own presence but also in all institutions. In the final analysis, memzuc can be described as sharing intelligence information through the Central Bank, an official institution. Memzuc records reflect approximately 1.5-2 months before the current period. This table, which can be obtained by giving the date range, also provides the possibility of comparing credibility with the previous periods of the companies. [10]

3.1.2.8.2 MEMZUC Variable

This variable was created for the self-employed segment and collected Memzuc data into YKB_ASSET_MEMZUC table. Memzuc variable is similar with KKB variable. Actually, both of them focus on risks. The only difference is for KKB taken into consideration of customer risks and for MEMZUC company(self-employed) risks. In memzuc table all columns were generated to define non-risk, having risk, risk in other banks, total risk and its ratio related with division between our bank risk and total risk value.

3.1.2.9 Segmentation Status

Segmentation was arranged with “employed and retired field” and “self-employed field”.

3.1.2.10 Having Product or not Flag

The goal of this step is to eliminate inactive customers from our population. The active customer definition is customers who have at least one product in our bank in the project. Therefore, these inactive customers were removed from the target table. 201811, 201806 and 201712 periods were examined based on PARTY_ID and CONTRACT_CLOSE_DATE with using EDW.CON_CONTRACT table which has information about mainly contract open and close date. Customer contract close date must be more than examination period date. So, it can be understood that customers’ contract is active during the period.

To determine also no product situation another ‘having’ table was created again based on PARTY_ID and CONTRACT_CLOSE_DATE. According to this limitation, customer contract close date must be less than examination period initially. Then also it must be less during the 12-month period.

Number of customers that have product in 201811 period: 19.264.828

Number of customers that have no product in 201811 period: 5.505.363

3.1.2.11 Regular Coming EFT

In this step, the goal was to determine loyal customer that bank has. If a customer has a regular "coming" EFT, that means customer storages his/her money in our bank. In other words, customer can be our target customer for the training model.

3.1.2.12 Transaction Count

In review period, total amount of transaction for each customer. This is really important to determine valuable customers.

3.1.2.13 Work Type

Work type of customer was determined according to following constraints.

'CALISMIYOR','EMEKLI','OGRENCI','HOME OFFICE', 'ISTEGE BAGLI', 'KAMU','OZEL SEKTOR','SERBEST MESLEK', 'EV HANIMI', 'EMEKLI+CALISIYOR'

In that way, customers that have different work type from above statements were removed.

3.1.3 Target Determination Tables

Finally, all variables were created, and customers defined. In this step, all customers were merged with Left Join function.

3.1.3.1 Full Population Table

In full population table (YKB_ASSET_FULL_POPULATION), all variables that were generated above merged into this table on distinct party_id. If select * from statement is written, all variable names can be seen.

3.1.3.2 Base Table

In base table (YKB_ASSET_BASE), "from" statement is full population table. In addition to this table,

- Firstly, predicted income is divided into groups as called ESTIMATED_INC_GROUP.
- CC_TRX_CNT >= 12, more than or equal to 12 credit card transaction count.
- Primary card flag variable was generated. It indicates customer that uses our bank more. With codes, more than 0.3 or 0.5 ratio of credit card limit/KKB total risk amount, this equivalent says that customer's credit card limit amount / total limit of all credit cards in KKB (includes our bank).
- There is a constraint called REGULAR_EFT_FLG about EFT count (more than 6) and coming EFT average amount (more than 2000 condition).
- Pension is as the same conditions as salary. (review period month - maximum pension as month < 3, number of pensions in last 12 months > 3).
- Customer Tenure condition is more than or equal to 3. (CUST_TENURE >= 3)
- KKB query in 6 months condition (KKB_QUERY_DATE > ADD_MONTHS (TO_DATE (year_month, 'YYYYMM'), -5) -1). It means that all customers that will be in target audience have a status in KKB query in 6 months.

- Number of worst payments of a customer should be less than 2 in KKB query. (KKB_WPS_L12M < 2).
- A customer should have a product. (NVL (NO_PRODUCT_FLG, 0) = 0)

3.1.3.3 Base Table for Employed Retired Segment

In this table, all data taken from Base table. Only in where statement, selected 'EMPLOYED RETIRED' and added PREDICTED_INCOME > 6000.

3.1.3.4 Base Table for Self Employed Segment

In this table, all data taken from Base table. Only in where statement, selected 'SELF EMPLOYED' and added PREDICTED_INCOME > 6000.

3.1.3.5 Employed Retired Target Table

This table was created with all base for employed retired segment and some conditions to obtain target. In the above figure, all constraints as seen in the below figure were written, all codes are in appendix – A.

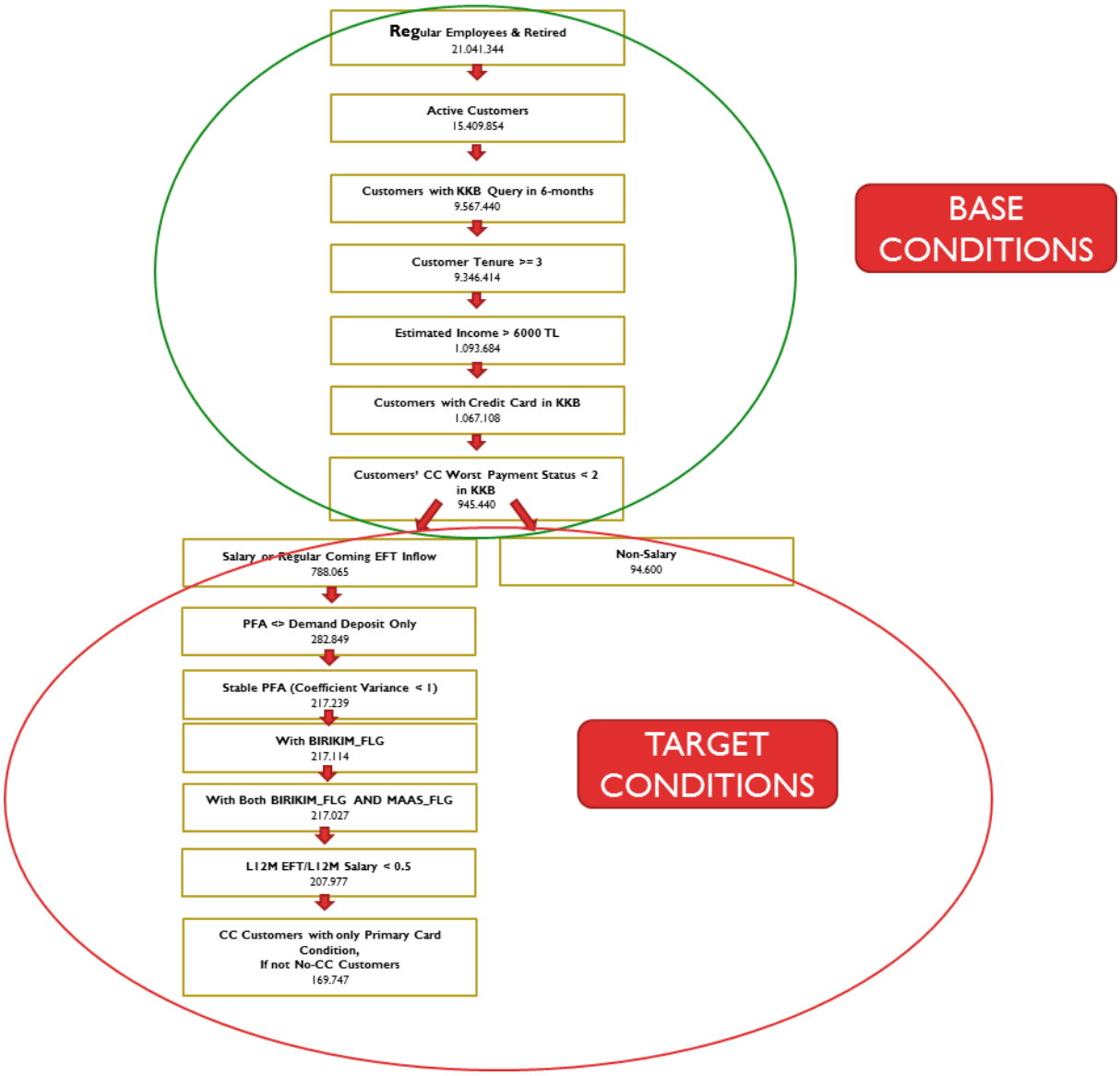


Figure 3-2 Employed Retired Base-Target Number of Customer Analysis

3.1.3.6 Self Employed Target Table

This table was created with all base for employed retired segment and some conditions to obtain target. In the above figure, all constraints as seen in the below figure were written, all codes are in appendix – A.

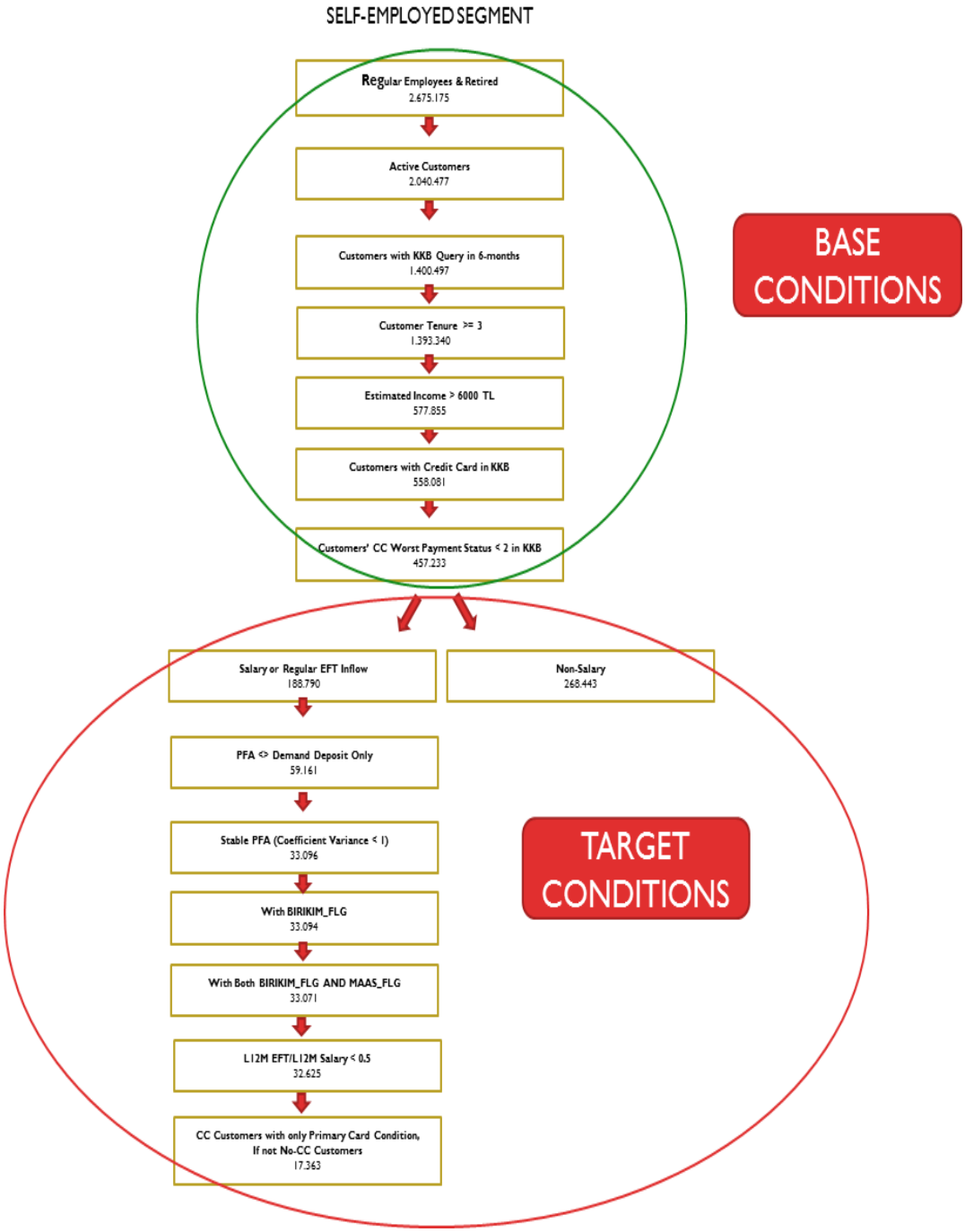


Figure 3-3 Self Employed Base-Target Number of Customer Analysis

3.1.3.7 Analysis in Excel

3.1.3.7.1 Analysis Between Customer Age vs Estimated Income Based on Ten Salary Amount in Last 12 Months with and without 'MAAS' Statement in the Description of EFT

Table 3-2 At Least 10 Months' Salary Amount with MAAS_FLG in Last 12 Months according to Review Period

MAAS_AY_CNT>10 with maas_flg										
Sum of TARGET_FLAG	**Estimated Income**									
Customer Age	**6K-7K**	**7K-8K**	**8K-9K**	**9K-10K**	**10K-12.5K**	**12.5K-15K**	**15K-17.5K**	**17.5K-20K**	**20K+**	**Grand Total**
18_25	472	258	126	57	66	18	13	7	6	1023
26_35	6844	4912	3611	2854	4863	2015	768	410	1354	27631
36_50	7723	5896	4829	3921	7398	5289	3340	1951	4428	44775
50P	1634	1422	1144	971	1943	1318	883	641	2144	12100
Grand Total	**16673**	**12488**	**9710**	**7803**	**14270**	**8640**	**5004**	**3009**	**7932**	**85529**

When two pivot table analysis are compared, with MAAS flag, 77 customers were removed from the target population. It means, it could be reached to more loyal customers with MAAS_AY_CNT > 10 situation.

Table 3-3 At Least 10 Months' Salary Amount without MAAS_FLG in Last 12 Months according to Review Period

MAAS_AY_CNT>10 without maas_flg										
Sum of TARGET_FLAG	**Estimated Income**									
Customer Age	**6K-7K**	**7K-8K**	**8K-9K**	**9K-10K**	**10K-12.5K**	**12.5K-15K**	**15K-17.5K**	**17.5K-20K**	**20K+**	**Grand Total**
18_25	472	258	126	57	66	18	13	7	6	1023
26_35	6851	4917	3612	2856	4863	2017	768	413	1354	27651
36_50	7731	5902	4833	3926	7410	5293	3342	1954	4429	44820
50P	1638	1422	1145	972	1945	1319	883	641	2147	12112
Grand Total	**16692**	**12499**	**9716**	**7811**	**14284**	**8647**	**5006**	**3015**	**7936**	**85606**

3.1.3.7.2 Analysis Between Customer Age vs Estimated Income Based on Six Salary Amount in Last 12 Months with and without 'MAAS' Statement in the Description of EFT

Table 3-4 At Least 6 Months' Salary Amount with MAAS_FLG in Last 12 Months according to Review Period

MAAS_AY_CNT>6 with maas_flg										
Sum of TARGET_FLAG	**Column Labels**									
Row Labels	**4.6K-7K**	**5.7K-8K**	**6.8K-9K**	**7.9K-10K**	**8.10K-12.5K**	**90.12.5K-15K**	**91.15K-17.5K**	**92.17.5K-20K**	**93.20K+**	**Grand Total**
0.18_25	550	284	138	63	72	18	13	8	10	1156
1.26_35	7766	5526	3977	3131	5207	2155	827	441	1411	30441
2.36_50	8636	6576	5310	4318	8141	5793	3629	2101	4742	49246
50P	1817	1544	1262	1065	2115	1451	974	697	2302	13227
Grand Total	**18769**	**13930**	**10687**	**8577**	**15535**	**9417**	**5443**	**3247**	**8465**	**94070**

When two pivot table analysis are compared, with MAAS flag, 103 customers were removed from the target population. It means, it could be reached to more loyal customers with MAAS_AY_CNT > 6 situation.

Table 3-5 At Least 10 Months' Salary Amount without MAAS_FLG in Last 12 Months according to Review Period

MAAS_AY_CNT>6 without maas_flg										
Sum of TARGET_FLAG	**Column Labels**									
Row Labels	**6K-7K**	**7K-8K**	**8K-9K**	**9K-10K**	**10K-12.5K**	**12.5K-15K**	**15K-17.5K**	**17.5K-20K**	**20K+**	**Grand Total**
18_25	550	284	138	63	72	18	13	8	10	1156
26_35	7776	5533	3978	3133	5209	2158	827	444	1411	30469
36_50	8646	6584	5314	4324	8155	5802	3633	2104	4743	49305
50P	1821	1545	1264	1066	2117	1452	974	697	2307	13243
Grand Total	**18793**	**13946**	**10694**	**8586**	**15553**	**9430**	**5447**	**3253**	**8471**	**94173**

After this analysis, as seen, MAAS_FLG should be used as a constraint.

3.1.3.7.3 Analysis of Customer Count Distribution Based on Customer Age and Estimated Income Groups

Table 3-6 Analysis Between Customer Age and Estimated Income

Sum of TARGET_FLAG	ESTIMATED INCOME								
CUST_AGE	**6K-7K**	**7K-8K**	**8K-9K**	**9K-10K**	**10K-12.5K**	**12.5K-15K**	**15K-17.5K**	**17.5K-20K**	**20K+**
18_25	47.58%	24.57%	11.94%	5.45%	6.23%	1.56%	1.12%	0.69%	0.87%
26_35	25.51%	18.15%	13.06%	10.29%	17.11%	7.08%	2.72%	1.45%	4.64%
36_50	17.54%	13.35%	10.78%	8.77%	16.53%	11.76%	7.37%	4.27%	9.63%
50P	13.74%	11.67%	9.54%	8.05%	15.99%	10.97%	7.36%	5.27%	17.40%
Grand Total	**19.95%**	**14.81%**	**11.36%**	**9.12%**	**16.51%**	**10.01%**	**5.79%**	**3.45%**	**9.00%**

In the above pivot table that shows the relationship between customer age and estimated income groups, as seen in the grand total, many of the customers are in 6K – 7K estimated income group.

- 20K + estimated income group is the highest (17.40%) with the 50+ age group, it means when age increases, the number of 50+ years old customers increase.
- 17.5K – 20K group is the highest (5.27%) with 50+ again, this group is the minimum of the all customers.
- As the estimated income group decreases, the number of groups with low customer age increases.
- The highest percentage in the table is 47.58 between 6K - 7K income group and between 18-25 years.

Analysis between Estimated Income and PFA Bands

Table 3-7Income-PFA Based on Asset Base Table

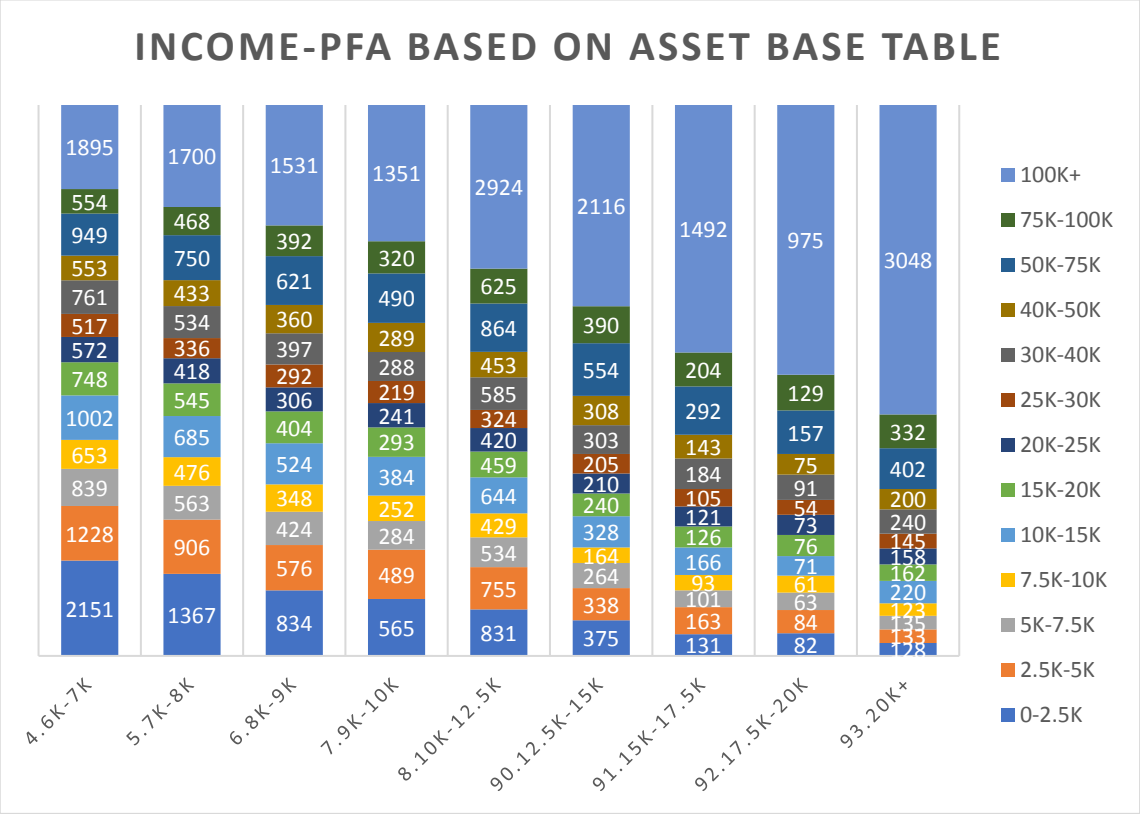


Table 3-8Anaylsis of Estimated Income and PFA Bands Relationship

TARGET_FLAG1	PFA Bands												
Estimated Income	0-2.5K (%)	2.5K-5K (%)	5K-7.5K (%)	7.5K-10K (%)	10K-15K (%)	15K-20K (%)	20K-25K (%)	25K-30K (%)	30K-40K (%)	40K-50K (%)	50K-75K (%)	75K-100K (%)	100K+ (%)
6K-7K	17.32	9.89	6.75	5.26	8.07	6.02	4.60	4.16	6.13	4.45	7.64	4.46	15.26
7K-8K	14.89	9.87	6.13	5.18	7.46	5.94	4.55	3.66	5.82	4.72	8.17	5.10	18.52
8K-9K	11.90	8.22	6.05	4.97	7.48	5.76	4.37	4.17	5.66	5.14	8.86	5.59	21.84
9K-10K	10.34	8.95	5.20	4.61	7.03	5.36	4.41	4.01	5.27	5.29	8.97	5.86	24.72
10K-12.5K	8.44	7.67	5.42	4.36	6.54	4.66	4.27	3.29	5.94	4.60	8.77	6.35	29.69
12.5K-15K	6.47	5.83	4.56	2.83	5.66	4.14	3.62	3.54	5.23	5.31	9.56	6.73	36.51
15K-17.5K	3.94	4.91	3.04	2.80	5.00	3.79	3.64	3.16	5.54	4.31	8.79	6.14	44.93
17.5K-20K	4.12	4.22	3.16	3.06	3.57	3.82	3.67	2.71	4.57	3.77	7.89	6.48	48.97
20K+	2.36	2.45	2.49	2.27	4.05	2.99	2.91	2.67	4.42	3.69	7.41	6.12	56.17
Grand Total	**10.69**	**7.73**	**5.30**	**4.30**	**6.66**	**5.05**	**4.17**	**3.63**	**5.60**	**4.65**	**8.40**	**5.65**	**28.17**

In the above analysis, as seen, population of target customers' distribution, customer asset ratio increases as estimated revenue rate increases.

3.1.3.8 Final Model Variables

The queries have been made so far to find the target customer. Model variables will be used for all customers. The machine learns with the model variables over the target population and in the scoring section, the model variables queries for each customer are thrown and asset estimation is performed. KKB model variables are the most significant ones. Because customer estimated asset amount is tried to find. Asset means all money that customer has. Therefore, our bank and other banks data are very important. That is why a lot of KKB model variables were used as seen in the below table.

Table 3-9Model Variables Definition

PARTY_ID	Customer ID
TARGET_PFA	PFA amount of customer
YEAR_MONTH	Review period (201712, 201806, 201811)
KKB_CC_LAST_PROD_TENURE	Month difference between the expiry date of the customer's open / close CC credit products and the review period in the KKB query
CUST_AGE	Customer's age at the end of the relevant year month
CUST_TENURE	Month difference between the customer's reporting date (201811,201806,201712) and being the first customer in the bank
KKB_TIME_SINCE_FIRST_CC	The age of the card, which is the oldest opening date of the open / closed cards in the period examined as a result of the KKB query of the customer, in months
KKB_LAST_PROD_TENURE	Month difference between customer's end product date and review date in KKB query
KKB_LOAN_LAST_PROD_TENURE	Month difference between customer's end loan product date and review date in KKB query
KKB_0PEN_CC_AVG_PMT_AMT_L3M	Customer average payment amount in last 3 months of all open CC accounts in KKB query

KKB_0PEN_CC_T0T_PMT_AMT_L3M	Customer total payment amount in last 3 months of all open CC accounts in KKB query
KKB_0PEN_CC_MAX_PMT_AMT_L3M	Customer maximum payment amount in last 3 months of all open CC accounts in KKB query
GENDER_E_FLAG	If customer is male, value is 1; else 0
GENDER_K_FLAG	If customer is female, value is 1; else 0
WIDOW_DIVORCED_FLAG	If customer is widow, value is 1; else 0
SINGLE_FLAG	If customer is single, value is 1; else 0
MARRIED_FLAG	If customer is married, value is 1; else 0
KKB_INST_C_MAX_INST_AMT_L3Y	Monthly installment amount of the credit which has the maximum installment credit closed in 3 years before the report date in KKB query
KKB_INST_C_MAX_CRE_AMT_L3Y	Monthly installment amount of the credit which has the maximum credit amount closed in 3 years before the report date in KKB query
KKB_OD_C_TOT_LIM_L3Y	Customer's total limit of overdraft credit products in last 3 years from KKB query (customer may have more than one OD credit product)
KKB_OD_C_MAX_LIM_L3Y	Customer's maximum limit of overdraft credit products between all OD credit products in last 3 years from KKB query
KKB_CC_C_MAX_LIM_L3Y	Customer's maximum limit of CC credit products between all CC credit products in last 3 years from KKB query
KKB_REV_C_MAX_LIM_L3Y	Customer's maximum limit of revolving credit products between all CC credit products in last 3 years from KKB query
KKB_CC_O_TOT_RISK	Customer total risk of all open CC products in KKB query (total risk means customer' CC usage amount, extract)

KKB_CC_O_MAX_RISK	Customer maximum risk with all open CC products in KKB query (maximum risk means the highest usage)
ESTIMATED_INCOME	Customer estimated income value
KKB_CC_O_MAX_LIM	Customer maximum limit of all open CC products in KKB query
KKB_CC_MAX_O_C_LIM_L3Y	Maximum limit of the CC credit product which has the maximum credit amount (open/closed in 3 years before the report date) in KKB query
KKB_CC_O_DLQ0_LIM	Customer's open CC product limit without delinquency in KKB query
KKB_CC_O_TOT_LIM	Customer's open CC products total limit in KKB query
KKB_WPS_L12M	Worst payment status information in KKB query
KKB_INST_O_TOT_INST_AMT	Customer total monthly installment amount of customer's individual installment credits in KKB query
KKB_INST_O_MAX_INST_AMT	Customer maximum monthly installment amount of customer's individual installment credits in KKB query
MAX24_GPL_INST	In the last 24 months, the customer's monthly maximum GPL installment amounts. All credit products are included.
MAX6_GPL_INST	In the last 6 months, the customer's monthly maximum GPL installment amounts. All credit products are included.
MAX24_RL_INST	In the last 24 months, the customer's monthly maximum installment amounts. All credit products are included.
MAX6_RL_INST	In the last 6 months, the customer's monthly maximum installment amounts. All credit products are included.

KKB_O_HL_CNT	Number of customer open home loan in KKB query
KKB_O_HL_FINANCE_AMT	Sum of the customer's open home loan opening balance in KKB query
KKB_O_HL_BAL	Customer's open home loan total balance
KKB_INST_O_TOT_RISK	Total risk amount in all open credits of customer in KKB query
KKB_CC_2ND_MAX_O_C_LIM	The second highest open / close (closed within 3 years before the query date) CC credit product in KKB query
KKB_CC_O_CRE_CNT	Number of open CC credit products of the customer in KKB query
KKB_LOAN_O_CRE_CNT	Number of open loan products of the customer in KKB query
KKB_APP_CNT_L6M	Number of applications made by customer within last 6 months in KKB query
KKB_OPND_CRE_CNT_L6M	Number of customer's credit products (CC, OD, PL, HL, AL) within last 6 months in KKB query
KKB_O_PL_CNT	Total number of customer's all personal loan in KKB query
KKB_PL_AL_O_RISK	Customer open auto loan risk inside of the personal loan in KKB query
KKB_OD_O_TOT_RISK	Total monthly risk of customer's OD credits in KKB query (risk: customer that uses overdraft amount)
KKB_O_HL_LAST_INS_AMT	Customer last payment installment home loan amount in open mortgage credits from KKB query
MAH_ILC_YETISKIN_LISE_ORAN	High school graduation ratio in neighborhood data that customer belongs. If the neighborhood information of the customer is empty, the district average will be used.

MAH_ILC_YETISKIN_UNI_ORAN	University graduation ratio in neighborhood data that customer belongs. If the neighborhood information of the customer is empty, the district average will be used.
MAH_ILC_Y_0_4	0-4 years of age in the customer's neighborhood information. If the neighborhood information of the customer is empty, the district average will be used.
MAH_ILC_Y_5_9	5-9 years of age in the customer's neighborhood information. If the neighborhood information of the customer is empty, the district average will be used.
MAH_ILC_Y_10_14	10-14 years of age in the customer's neighborhood information. If the neighborhood information of the customer is empty, the district average will be used.
MAH_ILC_Y_15_19	15-19 years of age in the customer's neighborhood information. If the neighborhood information of the customer is empty, the district average will be used.
MAH_ILC_Y_20_24	20-24 years of age in the customer's neighborhood information. (If the neighborhood information of the customer is empty, the district average will be used.)
MAH_ILC_Y_25_29	25-29 years of age in the customer's neighborhood information. If the neighborhood information of the customer is empty, the district average will be used.
MAH_ILC_Y_30_34	30-34 years of age in the customer's neighborhood information. If the neighborhood information of the customer is empty, the district average will be used.

MAH_ILC_Y_35_39	35-39 years of age in the customer's neighborhood information. If the neighborhood information of the customer is empty, the district average will be used.
MAH_ILC_Y_40_44	40-44 years of age in the customer's neighborhood information. If the neighborhood information of the customer is empty, the district average will be used.
MAH_ILC_Y_45_49	45-49 years of age in the customer's neighborhood information. If the neighborhood information of the customer is empty, the district average will be used.
MAH_ILC_Y_50_54	50-54 years of age in the customer's neighborhood information. If the neighborhood information of the customer is empty, the district average will be used.
MAH_ILC_Y_55_59	55-59 years of age in the customer's neighborhood information. If the neighborhood information of the customer is empty, the district average will be used.
MAH_ILC_Y_65_	65+ years of age in the customer's neighborhood information. If the neighborhood information of the customer is empty, the district average will be used.

4 APPLICATION IN R

4.1 Installation of Libraries & Definition of Parameters

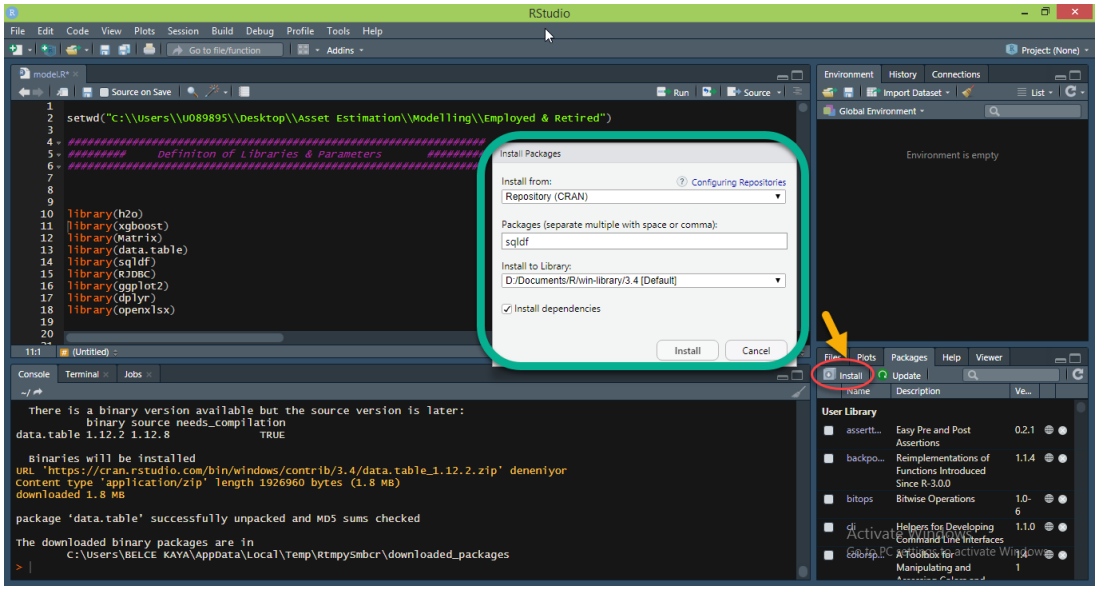


Figure 4-1 Installation of Libraries Step

4.1.1 Package 'data.table'

'data.table' is a package is used for working with tabular data in R. It provides the efficient data. table object which is a much improved version of the default data.frame. It is super-fast and has intuitive and terse syntax. [11]

In addition to this, it offers fast and memory efficient file reader and writer, aggregations, updates, equi, non-equi, rolling, range and interval joins, in a short and flexible syntax, for faster development. [12]

The data.table is an alternative to R's default data.frame to handle tabular data. The reason it's so popular is because of the speed of execution on larger data and the terse syntax. So, effectively you type less code and get much faster speed. It is one of the most downloaded packages in R and is preferred by Data Scientists. It is probably one of the best things that have happened to R programming language as far as speed is concerned.

4.1.1.1 Convertion from data.frame to data.table

With as.data.table(df) function creates a copy of data frame to a data table.

Though data.table provides a slightly different syntax from the regular R data.frame, it is quite intuitive. So once you get it, it feels obvious and natural that you wouldn't want to go back the base R data.frame syntax.

4.1.1.2 Dropping Columns

Place them in a vector and use the '!' in front to drop them. This effectively returns all columns except those present in the vector. [12]

In the codes: modelTestSample <- DT [! modelDevSeq]

4.1.2 Package 'XGBoost'

Word expansion is Extreme Gradient Boosting, which is an efficient implementation of the gradient boosting frame. The package includes efficient linear model solver and tree learning algorithms. The package can automatically do parallel computation on a single machine which could be more than 10 times faster than existing gradient boosting packages. It supports various objective functions, including regression, classification and ranking. The package is made to be extensible, so that users are also allowed to define their own objectives easily. [13]

4.1.3 Package 'RJDBC'

The RJDBC package is an implementation of R's DBI interface using JDBC as a backend. This allows R to connect to any DBMS that has a JDBC driver. [14]

Usage; JDBC (driverClass = "", classPath = "", identifier.quote = NA)

4.1.4 Package 'sqldf'

'sqldf' provides an easy way to perform SQL selects on R data frames.[15]

4.1.5 Package 'dplyr'

'dplyr' provides a flexible grammar of data manipulation that transforms and summarizes data tabular data with rows and columns. The package contains a set of functions (or "verbs") that perform common data manipulation operations such as filtering for rows, selecting specific columns, re-ordering rows, adding new columns, data cleaning, visualization and summarizing data. Additionally, uses the same interface to work with data no matter where it's stored, whether in a data frame, a data table or database. Provide blazing fast performance for in-memory data by writing key pieces in C++. [16] There are many useful functions contained within the dplyr package. Such as: select (), filter (), mutate (), group_by (), summarise (), arrange (), join () etc. [27]

4.1.6 Package 'openxlsx'

Simplifies the creation of Excel .xlsx files by providing a high-level interface to writing, styling and editing worksheets. [17] In the project, for example write.xlsx function was used to transfer data from R to Excel platform.

4.1.7 Package ‘ggplot2’

A system for 'declaratively' creating graphics, based on ``The Grammar of Graphics''. 'ggplot2' provides how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details. [18] In the project, many functions of ggplot2 package were used under the model performance graphs header.

4.1.8 Package ‘h2o’

R interface for 'H2O', the scalable open source machine learning platform that offers parallelized implementations of many supervised and unsupervised machine learning algorithms such as Generalized Linear Models, Gradient Boosting Machines (including XGBoost), Random Forests, Deep Neural Networks (Deep Learning), Stacked Ensembles, Naive Bayes, Cox Proportional Hazards, K-Means, PCA, Word2Vec, as well as a fully automatic machine learning algorithm (AutoML). [37]

4.1.9 Package ‘matrix’

A rich hierarchy of matrix classes, including triangular, symmetric, and diagonal matrices, both dense and sparse and with pattern, logical and numeric entries. [38]

4.1.10 Definition of Parameters

```
20
21  options(java.parameters = "-Xmx8g")
22  memory.limit(size=10000000000024)    #max memory usage
23  Sys.setenv("R_MAX_VSIZE"=64000000000)
24
```

Figure 4.1.8 Beginning Parameters

As a database, we used Oracle Database which has Java platform. Therefore, before we connect with database, we should define java parameter as well.

The default heap size for rJava: 512MB. Therefore, our Java virtual machine (JVM) may run out of memory during this process, resulting in an OutOfMemoryError. To fix this, the first row in code was used. [java.parameters] are evaluated exactly once per R session when the JVM is initialized. This is usually once loaded the first package that uses Java support, java parameter should be decided. In that case the model java parameter was written “-Xmx8g”. [19]

4.2 Oracle Connection & Data Extraction

4.2.1 Connection to Oracle Database in R with RJDBC

```
25
26  ######################################################
27  ########     Oracle Connection & Data Extraction      ########
28  ######################################################
29
30  jdbcDriver = JDBC(driverClass="oracle.jdbc.OracleDriver", classPath="C:\\Oracle\\instantclient_12_1\\ojdbc6.jar")
31  jdbcConnection = dbConnect(jdbcDriver, "jdbc:oracle:thin:[redacted]")
32
33
34  segment_name="EMPLOYED_RETIRED" #segment selection
35
36  SQL_Code = paste("select * from BK_YKB_ASSET_MODEL_TABLE3 where SEGMENT='",segment_name,"' and TARGET_FLAG2=1",sep="")
```

Figure 4-2 Oracle Connection and Data Extraction Codes

To connect Oracle SQL that we generated our model data, RJDBC (R Java Database Connectivity) package was used. (RJDBC is able to connect to an Oracle database and execute SQL commands directly in R.) [20]

4.2.2 Creation a Driver Object in R

To create a driver object, following code is used.

- ✓ jdbcDriver=JDBC("oracle.jdbc.OracleDriver",classPath="C:\\Oracle\\instantclient_12_1\\ojdbc6.jar")

4.2.3 Connection to the Oracle Database

- ✓ jdbcConnection =dbConnect(jdbcDriver,"jdbc:oracle:thin:@//database.hostname.com:port/service_name_or_sid", "username", "password") In the above figure host name, username and password information was blurred because of data privacy policy.

4.2.4 Running Oracle SQL Query

In this step, model data (BK_YKB_ASSET_MODEL_TABLE3) that have been generated in Oracle platform runs as SQL query. Thanks to 'sqldf' package, SQL language can be written into code as seen in the figure. Here, only the information of customers whose target_flag2 value is equal to 1 is taken.

```
35
36  SQL_Code = paste("select * from BK_YKB_ASSET_MODEL_TABLE3 where SEGMENT='",segment_name,"' and TARGET_FLAG2=1",sep="")
37  #paste("BELCE","KAYA",":))))))", sep="___")
38
39  MAIN_TABLE<- dbGetQuery(jdbcConnection, SQL_Code)
40  MAIN_TABLE<- as.data.table(MAIN_TABLE)
41
42  colnames(MAIN_TABLE)
43
44  DT<-dbGetQuery(jdbcConnection, SQL_Code)
45  DT<- as.data.table(DT)
```

Figure 4.2.4 Running SQL Codes

- ✓ dbGetQuery: read the result from a SQL statement to a data frame
- ✓ table2= dbGetQuery(con,'select * from table1 where name=\'string\'')
- ✓ table2: new table name that will be used in R, table1: table from SQLhe

Additionally, twice a time the same table were run from SQL. MAIN_TABLE was generated for only analyzing correlation statement. DT is the table that have been used through every step of the modelling part.

4.3 Data cleaning & Feature Engineering & Missing Values

4.3.1 Replacing (-9999) and NAs with 0 and Explicit level "Missing"

Cleaning data from meaningless values is one of the most important steps. In this step, for and if functions were used to replace with 0 and missing values.

```
52 ############################################################
53 ############    Replace (-9999) and NAs with 0      ############
54 ###########   Replace with explicit level "Missing"   ###########
55 ############################################################
56
57
58 #make 0 for value= -9999 and NAs!!
59 for(colName in names(DT)) {
60   if (is.numeric(DT[[colName]]) == TRUE) {
61     DT[is.na(get(colName)),   (colName) := 0]
62     DT[get(colName) == -9999, (colName) := 0]
63   }
64 }
65
66
67 for(colName in names(DT)) {
68   if(is.factor(DT[[colName]])) {
69     DT[i = is.na(get(colName)), j = (colName) := "Missing"]
70   }
71 }  #If the factor variable NA, make colname "missing".
72    #BUT OUR DATA HAS ALREADY ONLY 2 FACTOR VARIABLES, so we do not actually change anything.
73
```

Figure 4-3 Cleaning NAs, -9999 and Missing Values Codes

4.4 Feature Selection (Correlation & P-value)

All of the features we find in the dataset might not be useful in building a machine learning model to make the necessary prediction. Using some of the features might even make the predictions worse. So, feature selection plays a huge role in building a machine learning model.

4.4.1 Correlation Definition

Correlation is a statistical term which in common usage refers to how close two variables are to having a linear relationship with each other. Features with high correlation are more linearly dependent and hence have almost the same effect on the dependent variable. So, when two features have high correlation, we can drop one of the two features.

4.4.1.1 Positive Correlation

The two variables move in the same direction (i.e., one variable increases as the other increases or one decreases as the other decreases).

4.4.1.2 Negative correlation

The two variables move in opposite directions (i.e., one variable increases as the other decreases, and vice versa)

4.4.1.3 Neutral correlation

The two variables show no relationship to one another.

Figure 4-4 Correlation Types

Concerning the form of a correlation, it could be linear, non-linear, or monotonic:

4.4.1.4 Linear correlation

A correlation is linear when two variables change at constant rate and satisfy the equation Y = aX + b (i.e., the relationship must graph as a straight line).

Non-Linear correlation: A correlation is non-linear when two variables don't change at a constant rate. In this case the relationship between the variables does not graph as a straight line, but as a curved pattern (parabola, hyperbola … etc.).

Monotonic correlation: In a monotonic relationship, the variables tend to move in the same relative direction, but not necessarily at a constant rate. So, all linear correlations are monotonic, but the opposite is not always true, because we can have also monotonic non-linear relationships.

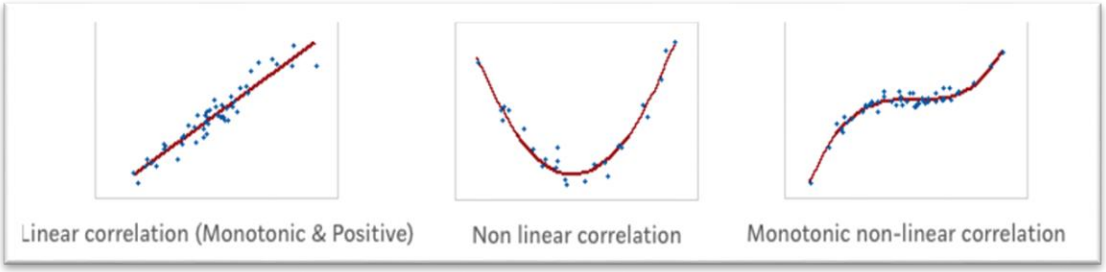


Figure 4.4.1.4 Linear Correlation

4.4.2 Correlation Coefficient

The main result of a correlation is called the correlation coefficient (or "r"). It ranges from -1.0 to +1.0. The closer r is to +1 or -1, the more closely the two variables are related. If r is close to 0, it means there is no relationship between the variables. If r is positive, it means that as one variable gets larger the other gets larger. If r is negative it means that as one gets larger, the other gets smaller (often called an "inverse" correlation). [28]

The Formula for Correlation Is

$$r = \frac{\sum(X - \overline{X})(Y - \overline{Y})}{\sqrt{\sum(X - \overline{X})^2}\sqrt{(Y - \overline{Y})^2}}$$

where:

r = the correlation coefficient

$\overline{X}$ = the average of observations of variable X

$\overline{Y}$ = the average of observations of variable Y

Figure 4-5 General Formula of Correlation

There are several types of correlation coefficients but the one that is most common is the Pearson correlation r. It is a parametric test that is only recommended when the variables are normally distributed and the relationship between them is linear. Otherwise, non-parametric Kendall and Spearman correlation tests should be used.

4.4.2.1 Pearson's correlation coefficient

Pearson correlation (r) is used to measure strength and direction of a linear relationship between two variables. Mathematically this can be done by dividing the covariance of the two variables by the product of their standard deviations.

$$r = r_{xy} = \frac{cov(x, y)}{S_x * S_y}$$

Figure 4.4.2.1 Pearson's Coefficient

The value of r ranges between -1 and 1. A correlation of -1 shows a perfect negative correlation, while a correlation of 1 shows a perfect positive correlation. A correlation of 0 shows no relationship between the movement of the two variables. The table below demonstrates how to interpret the size (strength) of a correlation coefficient.

Size of Correlation	Interpretation
.90 to 1.00 (−.90 to −1.00)	Very high positive (negative) correlation
.70 to .90 (−.70 to −.90)	High positive (negative) correlation
.50 to .70 (−.50 to −.70)	Moderate positive (negative) correlation
.30 to .50 (−.30 to −.50)	Low positive (negative) correlation
.00 to .30 (.00 to −.30)	negligible correlation

In the model, values that more than or equal to ±0.80 were examined to determine column names that will be removed.

4.4.3 Correlogram

A correlogram (also called Auto Correlation Function ACF Plot or Autocorrelation plot) is a visual way to show serial correlation in data that changes over time (i.e. time series data). Serial correlation (also called autocorrelation) is where an error at one point in time travels to a subsequent point in time. For example, you might overestimate the value of your stock market investments for the first quarter, leading to an overestimate of values for following quarters. [29]

Actually, correlogram was used to examine feature correlations. Only visualization was examined.

4.4.4 Implementation of Correlogram

For this process, MAIN_TABLE was created to do correlation analysis. First of all, NA and factor values was replaced with “0” and “missing” statements.

```
74  ######### main table #########
75
76
77  for(colName in names(MAIN_TABLE)) {
78    if (is.numeric(MAIN_TABLE[[colName]]) == TRUE) {
79      MAIN_TABLE[is.na(get(colName)),    (colName) := 0]
80      MAIN_TABLE[get(colName) == -9999, (colName) := 0]
81    }
82  }  #değeri -9999 ve NA olanları 0 yap!!!!!!!!!
83
84
85  for(colName in names(MAIN_TABLE)) {
86    if(is.factor(MAIN_TABLE[[colName]])) {
87      MAIN_TABLE[i = is.na(get(colName)), j = (colName) := "Missing"]
88    }
89  }
```

Figure 4-6 Replacement of Missing and NAs for Main Table

Pearson correlation formula was applied in correlation analysis. There is a standard deviation in the denominator of the Pearson formula. If the minimum and maximum values of a variable are equal, the standard deviation is zero. In case of a standard deviation is zero, the system cannot calculate according to the Pearson correlation formula. The denominator cannot be zero. Therefore, to determine variables whose minimum and maximum values are equal, “summary” function was used.

```
########################################################
#######   Correlation test to detrmine model variables ########
########################################################

str(MAIN_TABLE)

#check if there are variables that have the same min and max values or not.
summary(MAIN_TABLE)

#variables that lead to zero standard deviation
vars_to_remove <- c("TARGET_FLAG",
                    "TARGET_FLAG2" ,
                    "KKB_OPEN_NREV_MAX_PMT_AMT_LM" ,
                    "KKB_OPEN_NREV_TOT_PMT_AMT_LM" ,
                    "GENDER_NULL_FLAG" ,
                    "KKB_C_CNT_DEF_DL36" ,
                    "KKB_OPEN_NREV_AVG_PMT_AMT_LM" ,
                    "KKB_O_PL_LAST_INS_AMT",
                    "BEH_CLUSTER0",
                    "MAH_CLUSTER0",
                    "MAH_CLUSTER2")

#remove variables that have the same minimum and maximum values to avoid from "standard deviation is equal to zero" error.
MAIN_TABLE[,  c( vars_to_remove) := NULL]

#remove variables that are not numeric.
MAIN_TABLE$YEAR_MONTH <- NULL
MAIN_TABLE$SEGMENT <- NULL

str(MAIN_TABLE)
str(MAIN_TABLE.cor)
```

Figure 4-7 Removal of Variables that lead to zero standard deviation

Removal of variables that have the same minimum and maximum value and factor variables

4.4.4.1 First Correlation Analysis (ASSET_CORR_ANL.PNG SCHEME)

4.4.4.1.1 Correlation Parameters Definition

```
########################## ASSET_CORR_ANL.png ######################

#+++++++++++++++++++++++++
# Computing of correlation matrix
#+++++++++++++++++++++++++
# Required package : corrplot
# x : matrix
# type: possible values are "lower" (default), "upper", "full" or "flatten";
#display lower or upper triangular of the matrix, full  or flatten matrix.
# graph : if TRUE, a correlogram or heatmap is plotted
# graphType : possible values are "correlogram" or "heatmap"
# col: colors to use for the correlogram
# ... : Further arguments to be passed to cor or cor.test function
# Result is a list including the following components :
# r : correlation matrix, p :  p-values
# sym : Symbolic number coding of the correlation matrix
```

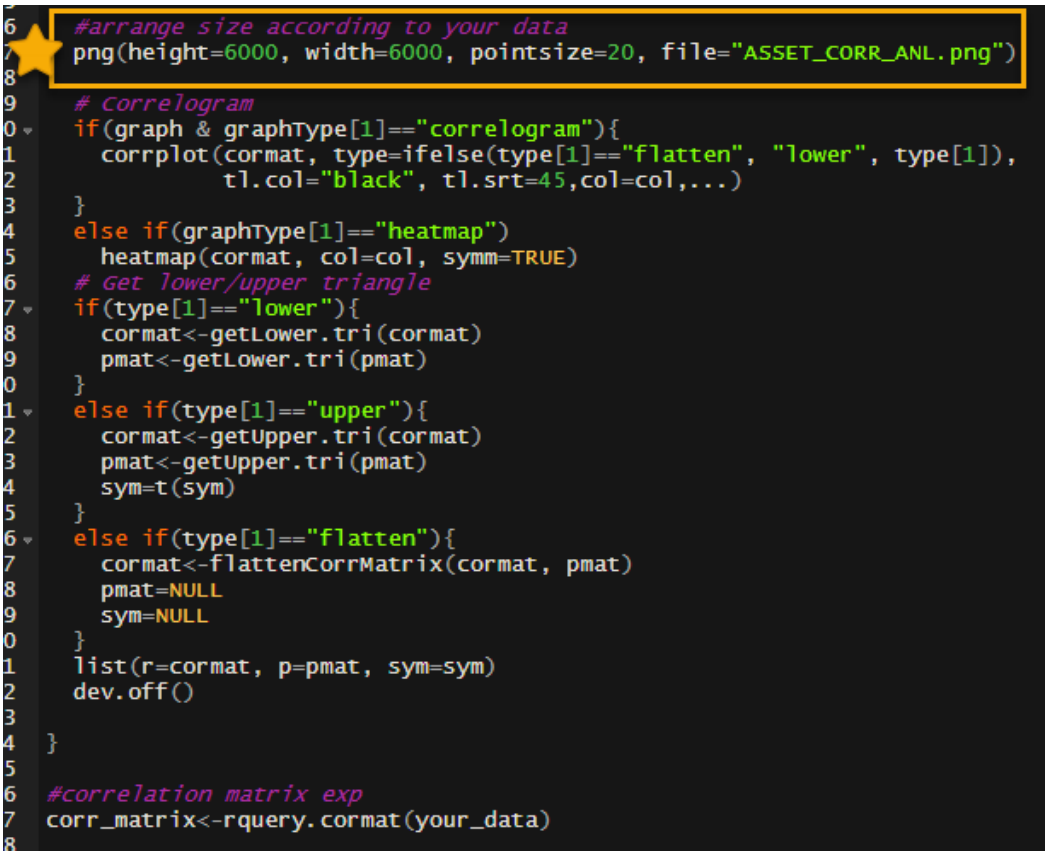
Figure 4-8 Definition of Parameters that will be used

```r
your_data <- MAIN_TABLE
# name
work="ASSET_COR_"

rquery.cormat<-function(x,
                        type=c('lower', 'upper', 'full', 'flatten'),
                        graph=TRUE,
                        graphType=c("correlogram", "heatmap"),
                        col=NULL, ...)
{
  # Result Location
  setwd("C:\\Users\\U089895\\Desktop\\Asset Estimation\\Modelling\\Employed & Retired")

  library(corrplot)
  # Helper functions
  #+++++++++++++++++
  # Compute the matrix of correlation p-values
  cor.pmat <- function(x, ...) {
    mat <- as.matrix(x)
    n <- ncol(mat)
    p.mat<- matrix(NA, n, n)
    diag(p.mat) <- 0
    for (i in 1:(n - 1)) {
      for (j in (i + 1):n) {
        tmp <- cor.test(mat[, i], mat[, j], ...)
        p.mat[i, j] <- p.mat[j, i] <- tmp$p.value
      }
    }
    colnames(p.mat) <- rownames(p.mat) <- colnames(mat)
    p.mat
  }
```

Figure 4-9 Calling 'corrplot' package and computing the matrix of correlation p-values

```r
  # Get upper triangle of the matrix
  getUpper.tri<-function(mat){
    lt<-mat
    lt[lower.tri(mat)]<-""
    mat<-as.data.frame(lt)
    mat
  }
  # Get flatten matrix
  flattenCorrMatrix <- function(cormat, pmat) {
    ut <- upper.tri(cormat)
    data.frame(
      row = rownames(cormat)[row(cormat)[ut]],
      column = rownames(cormat)[col(cormat)[ut]],
      cor  =(cormat)[ut],
      p = pmat[ut]
    )
  }
  # Define color
  if (is.null(col)) {
    col <- colorRampPalette(
      c("#67001F", "#B2182B", "#D6604D", "#F4A582",
        "#FDDBC7", "#FFFFFF", "#D1E5F0", "#92C5DE",
        "#4393C3", "#2166AC", "#053061"))(200)
    col<-rev(col)
  }

  # Correlation matrix
  cormat<-signif(cor(x, use = "complete.obs", ...),2)
  pmat<-signif(cor.pmat(x, ...),2)
  # Reorder correlation matrix
  ord<-corrMatOrder(cormat, order="hclust")
  cormat<-cormat[ord, ord]
  pmat<-pmat[ord, ord]
  # Replace correlation coeff by symbols
  sym<-symnum(cormat, abbr.colnames=FALSE)
```

Figure 4-10 Creating Correlation Matrix and Defining Color



```
  #arrange size according to your data
  png(height=6000, width=6000, pointsize=20, file="ASSET_CORR_ANL.png")

  # Correlogram
  if(graph & graphType[1]=="correlogram"){
    corrplot(cormat, type=ifelse(type[1]=="flatten", "lower", type[1]),
             tl.col="black", tl.srt=45,col=col,...)
  }
  else if(graphType[1]=="heatmap")
    heatmap(cormat, col=col, symm=TRUE)
  # Get lower/upper triangle
  if(type[1]=="lower"){
    cormat<-getLower.tri(cormat)
    pmat<-getLower.tri(pmat)
  }
  else if(type[1]=="upper"){
    cormat<-getUpper.tri(cormat)
    pmat<-getUpper.tri(pmat)
    sym=t(sym)
  }
  else if(type[1]=="flatten"){
    cormat<-flattenCorrMatrix(cormat, pmat)
    pmat=NULL
    sym=NULL
  }
  list(r=cormat, p=pmat, sym=sym)
  dev.off()

}

#correlation matrix exp
corr_matrix<-rquery.cormat(your_data)
```

Figure 4-11 Getting ASSET_CORR_ANL.png file

4.4.4.1.2 Output of Correlogram:

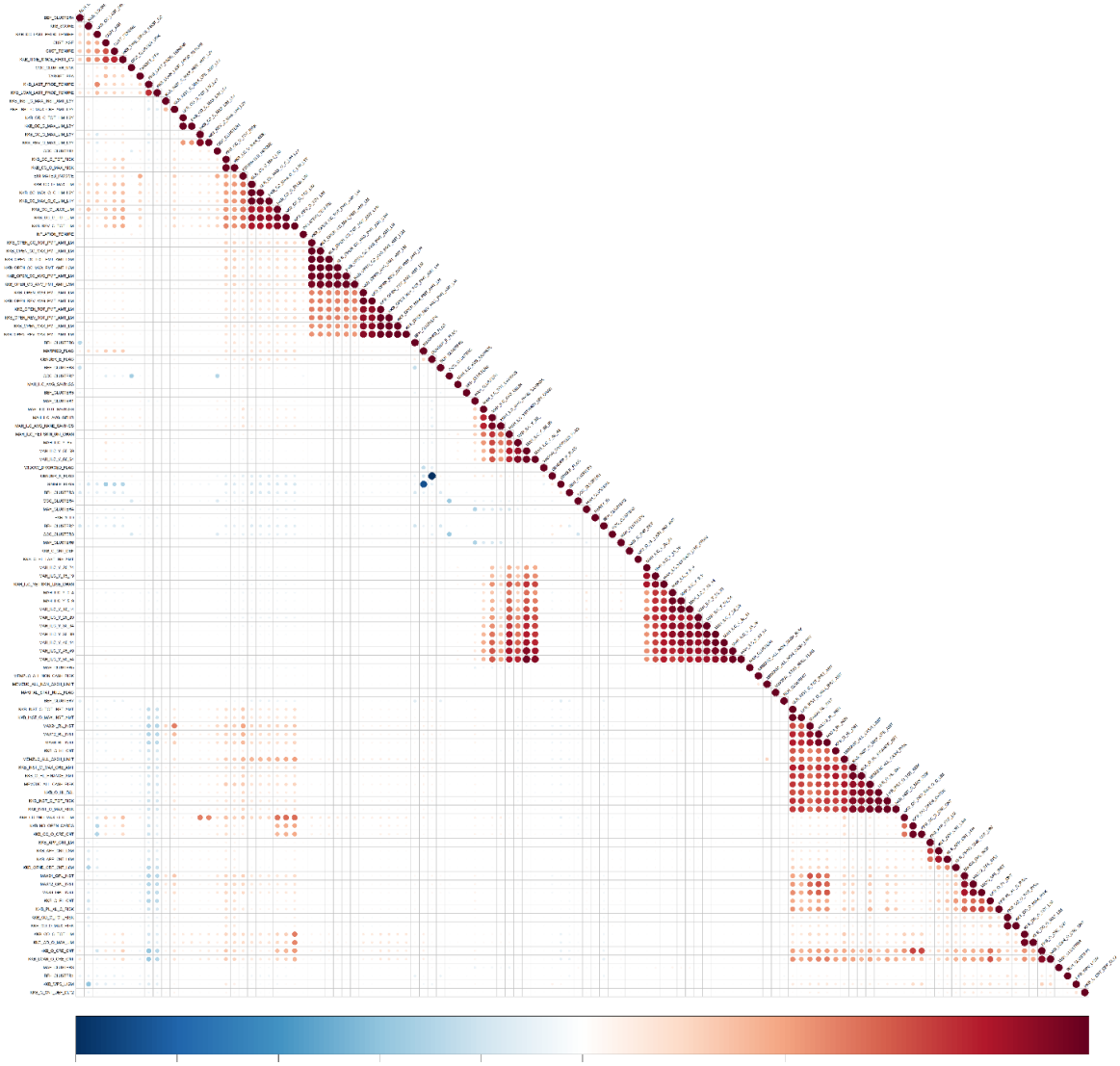


Figure 4-12 ASSET_CORR_ANL Schema

4.4.4.2 Second Correlation Analysis (ASSET_CORR_ANL3.PNG SCHEME)

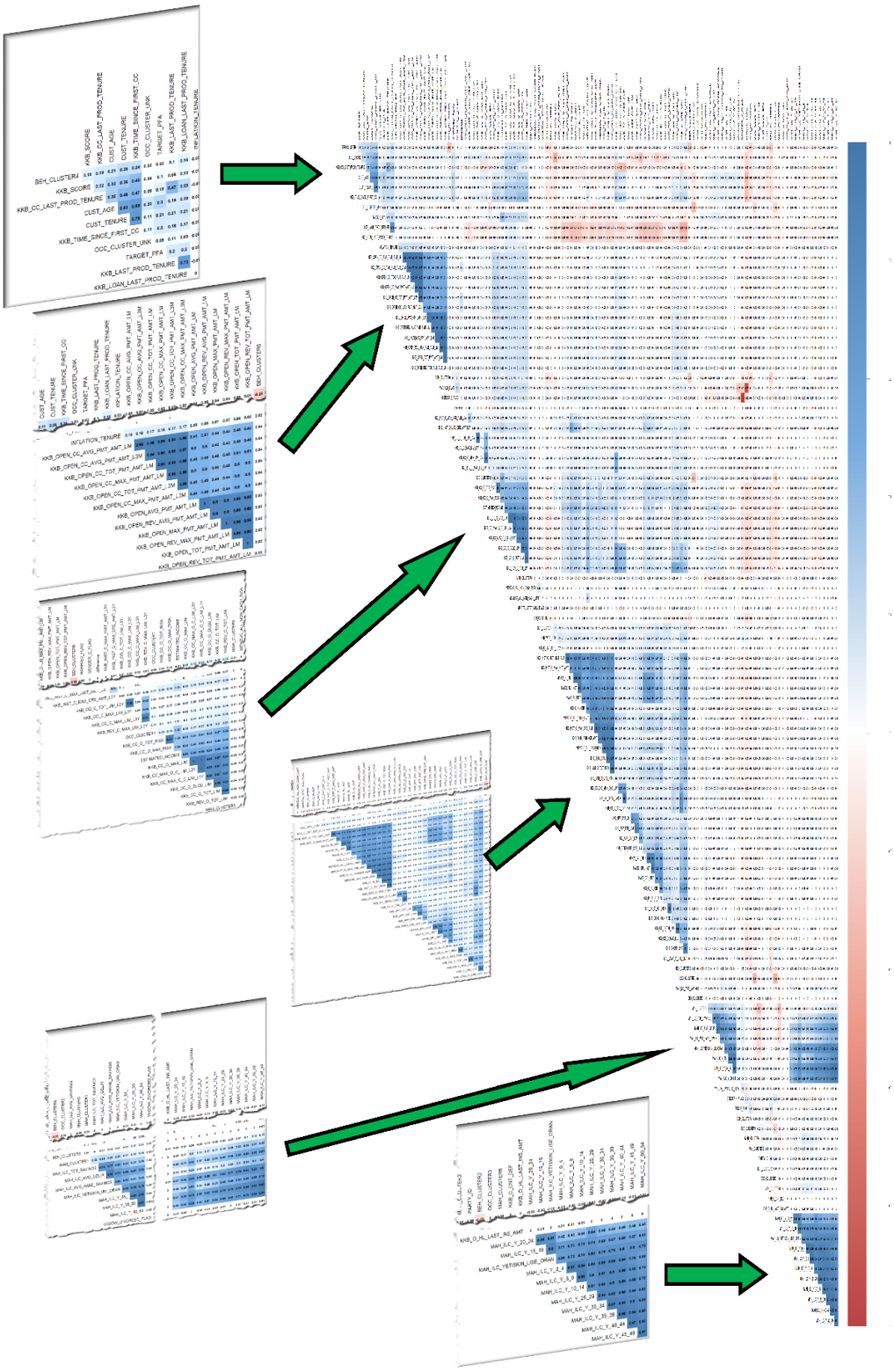
4.4.4.2.1 Codes for Second Correlation Analysis

With coefficients correglam

```
###ASSET_CORR_ANL3.png

library(corrplot)

M <- cor(MAIN_TABLE)
png(height=6000, width=6000, pointsize=20, file="ASSET_CORR_ANL3.png")
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot(M, method = "color", col = col(200),
         type = "upper", order = "hclust", number.cex = .7,
         addcoef.col = "black", # Add coefficient of correlation
         tl.col = "black", tl.srt = 90, # Text label color and rotation
         # Combine with significance
         sig.level = 0.01, insig = "blank",
         # hide correlation coefficient on the principal diagonal
         diag = FALSE)
```

Figure 4-13 Calling 'corrplot' package and Creating Cor Matrix

4.4.4.2.2 Output:



As shown in the figure above, the parts distributed in blue are emphasized. Dark blue boxes symbolize highly positive correlated variables.

Table 4-1 List of ±80 and Over 80 Percent Variables

Correlation Value	Left	to	Right
1.00	KKB_OPEN_AVG_PMT_AMT_LM	<=>	KKB_OPEN_REV_AVG_PMT_AMT_LM
1.00	KKB_OPEN_MAX_PMT_AMT_LM	<=>	KKB_OPEN_REV_MAX_PMT_AMT_LM
1.00	KKB_OPEN_TOT_PMT_AMT_LM	<=>	KKB_OPEN_REV_TOT_PMT_AMT_LM
1.00	KKB_CC_O_MAX_LIM	<=>	KKB_CC_MAX_O_C_LIM_L3Y
1.00	KKB_CC_O_MAX_LIM	<=>	KKB_CC_MAX_O_C_LIM_L3Y
1.00	KKB_CC_MAX_O_C_LIM_L3Y	<=>	KKB_CC_MAX_O_C_LIM_L1Y
0.99	KKB_OPEN_CC_TOT_PMT_AMT_LM	<=>	KKB_OPEN_CC_MAX_PMT_AMT_LM
0.99	KKB_OPEN_CC_TOT_PMT_AMT_LM	<=>	KKB_OPEN_CC_MAX_PMT_AMT_LM
0.99	KKB_OPEN_CC_MAX_PMT_AMT_LM	<=>	KKB_0PEN_CC_T0T_PMT_AMT_L3M
0.99	KKB_0PEN_CC_T0T_PMT_AMT_L3M	<=>	KKB_0PEN_CC_MAX_PMT_AMT_L3M
0.98	KKB_OPEN_CC_AVG_PMT_AMT_LM	<=>	KKB_0PEN_CC_AVG_PMT_AMT_L3M
0.98	KKB_OPEN_CC_AVG_PMT_AMT_LM	<=>	KKB_0PEN_CC_AVG_PMT_AMT_L3M
0.98	KKB_OPEN_CC_TOT_PMT_AMT_LM	<=>	KKB_OPEN_CC_MAX_PMT_AMT_LM
0.98	KKB_OPEN_CC_MAX_PMT_AMT_LM	<=>	KKB_0PEN_CC_T0T_PMT_AMT_L3M
0.98	KKB_OPEN_MAX_PMT_AMT_LM	<=>	KKB_OPEN_REV_MAX_PMT_AMT_LM
0.98	KKB_OPEN_MAX_PMT_AMT_LM	<=>	KKB_OPEN_REV_MAX_PMT_AMT_LM
0.98	KKB_OPEN_REV_MAX_PMT_AMT_LM	<=>	KKB_OPEN_TOT_PMT_AMT_LM
0.98	KKB_OPEN_REV_MAX_PMT_AMT_LM	<=>	KKB_OPEN_TOT_PMT_AMT_LM
0.98	KKB_OD_C_TOT_LIM_L3Y	<=>	KKB_OD_C_MAX_LIM_L3Y
0.98	KKB_INST_O_TOT_RISK	<=>	KKB_INST_O_MAX_RISK
0.98	MAH_ILC_Y_5_9	<=>	MAHJLC_Y_10_14
0.98	MAH_ILC_Y_35_39	<=>	MAH_ILC_Y_40_44
0.97	KKB_0PEN_CC_AVG_PMT_AMT_L3M	<=>	KKB_OPEN_CC_TOT_PMT_AMT_LM
0.97	KKB_CC_O_TOT_RISK	<=>	KKB_CC_O_MAX_RISK
0.97	KKB_INST_O_TOT_INST_AMT	<=>	KKB_INST_O_MAX_INST_AMT
0.97	KKB_O_HL_BAL	<=>	KKB_INST_O_TOT_RISK
0.97	KKB_NO_OPEN_CARDS	<=>	KKB_CC_O_CRE_CNT
0.97	KKB_OD_O_TOT_RISK	<=>	KKB_OD_O_MAX_RISK
0.97	MAH_ILC_Y_55_59	<=>	MAH_ILC_Y_60_64
0.97	MAH_ILC_Y_55_59	<=>	MAH_ILC_Y_60_64
0.97	MAH_ILC_Y_0_4	<=>	MAH_ILC_Y_5_9
0.97	MAH_ILC_Y_30_34	<=>	MAH_ILC_Y_35_39
0.97	MAH_ILC_Y_40_44	<=>	MAH_ILC_Y_45_49
0.97	MAH_ILC_Y_45_49	<=>	MAH_ILC_Y_50_54
0.96	KKB_OPEN_CC_AVG_PMT_AMT_LM	<=>	KKB_0PEN_CC_AVG_PMT_AMT_L3M
0.96	KKB_OPEN_CC_AVG_PMT_AMT_LM	<=>	KKB_0PEN_CC_AVG_PMT_AMT_L3M
0.96	KKB_0PEN_CC_AVG_PMT_AMT_L3M	<=>	KKB_OPEN_CC_TOT_PMT_AMT_LM
0.96	KKB_CC_O_TOT_LIM	<=>	KKB_REV_O_TOT_LIM
0.96	KKB_INST_O_MAX_CRE_AMT	<=>	KKB_O_HL_FINANCE_AMT
0.96	KKB_O_HL_BAL	<=>	KKB_INST_O_TOT_RISK

0.96	KKB_OD_O_TOT_LIM	<=>	KKB_OD_O_MAX_LIM
0.95	KKB_OPEN_CC_AVG_PMT_AMT_LM	<=>	KKB_0PEN_CC_AVG_PMT_AMT_L3M
0.95	KKB_0PEN_CC_AVG_PMT_AMT_L3M	<=>	KKB_OPEN_CC_TOT_PMT_AMT_LM
0.95	KKB_CC_O_DLQ0_LIM	<=>	KKB_CC_O_TOT_LIM
0.95	MAX12_RLJNST	<=>	MAX6_RL_INST
0.95	MEMZUC_ALL_CASH_RISK	<=>	KKB_O_HL_BAL
0.95	MAH_ILC_AVG_GELIR	<=>	MAH_ILC_AVG_HANE_SAVINGS
0.94	KKB_0PEN_CC_AVG_PMT_AMT_L3M	<=>	KKB_OPEN_CC_TOT_PMT_AMT_LM
0.94	MEMZUC_ALL_CASH_RISK	<=>	KKB_O_HL_BAL
0.94	MAH_ILC_Y_0_4	<=>	MAH_ILC_Y_5_9
0.94	MAH_ILC_Y_25_29	<=>	MAH_ILC_Y_30_34
0.94	MAH_ILC_Y_35_39	<=>	MAH_ILC_Y_40_44
0.93	MAX12_GPL_JNST	<=>	MAX6_GPL_INST
0.93	MAH_ILC_Y_30_34	<=>	MAH_ILC_Y_35_39
0.92	KKB_INST_O_MAX_CRE_AMT	<=>	KKB_O_HL_FINANCE_AMT
0.92	KKB_O_HL_FINANCE_AMT	<=>	MEMZUC_ALL_CASH_RISK
0.92	MAH_ILC_Y_65_	<=>	MAH_ILC_Y_55_59
0.92	MAH_ILC_Y_40_44	<=>	MAH_ILC_Y_45_49
0.91	KKB_CC_C_MAX_LIM_L3Y	<=>	KKB_REV_C_MAX_LIM_L3Y
0.91	KKB_CC_O_DLQ0_LIM	<=>	KKB_CC_O_TOT_LIM
0.91	KKB_INST_O_MAX_CRE_AMT	<=>	KKB_O_HL_FINANCE_AMT
0.91	MEMZUC_ALL_CASH_RISK	<=>	KKB_O_HL_BAL
0.91	MAH_ILC_Y_55_59	<=>	MAH_ILC_Y_60_64
0.91	MAH_ILC_Y_60_64	<=>	WIDOW_DIVORCED_FLAG
0.90	KKB_OPEN_AVG_PMT_AMT_LM	<=>	KKB_OPEN_REV_AVG_PMT_AMT_LM
0.90	KKB_OPEN_AVG_PMT_AMT_LM	<=>	KKB_OPEN_REV_AVG_PMT_AMT_LM
0.90	KKB_OPEN_REV_AVG_PMT_AMT_LM	<=>	KKB_OPEN_MAX_PMT_AMT_LM
0.90	KKB_OPEN_REV_AVG_PMT_AMT_LM	<=>	KKB_OPEN_MAX_PMT_AMT_LM
0.90	KKB_O_CRE_CNT	<=>	KKB_LOAN_O_CRE_CNT
0.90	MAHJLC_Y_15_19	<=>	MAH_ILC_YETISKIN_LISE_ORAN
0.90	MAH_ILC_Y_5_9	<=>	MAH_ILC_Y_10_14
0.90	MAH_ILC_Y_10_14	<=>	MAH_ILC_Y_25_29
0.90	MAH_ILC_Y_30_34	<=>	MAH_ILC_Y_35_39
0.89	KKB_INST_O_MAX_CRE_AMT	<=>	KKB_O_HL_FINANCE_AMT
0.89	KKB_O_HL_FINANCE_AMT	<=>	MEMZUC_ALL_CASH_RISK
0.89	KKB_O_HL_FINANCE_AMT	<=>	MEMZUC_ALL_CASH_RISK
0.89	MAX24_GPL_INST	<=>	MAX12_GPL_JNST
0.89	MAH_ILC_Y_0_4	<=>	MAH_ILC_Y_5_9
0.89	MAH_ILC_Y_10_14	<=>	MAH_ILC_Y_25_29
0.89	MAH_ILC_Y_25_29	<=>	MAH_ILC_Y_30_34
0.88	KKB_OPEN_AVG_PMT_AMT_LM	<=>	KKB_OPEN_REV_AVG_PMT_AMT_LM
0.88	KKB_OPEN_AVG_PMT_AMT_LM	<=>	KKB_OPEN_REV_AVG_PMT_AMT_LM
0.88	KKB_OPEN_REV_AVG_PMT_AMT_LM	<=>	KKB_OPEN_MAX_PMT_AMT_LM
0.88	KKB_OPEN_REV_AVG_PMT_AMT_LM	<=>	KKB_OPEN_MAX_PMT_AMT_LM
0.88	MAX24_RL_INST	<=>	MAX12_RL_INST

0.88	MAH_ILC_Y_0_4	<=>	MAH_ILC_Y_5_9
0.88	MAH_ILC_Y_5_9	<=>	MAH_ILC_Y_10_14
0.88	MAH_ILC_Y_35_39	<=>	MAH_ILC_Y_40_44
0.87	KKB_CC_O_MAX_LIM	<=>	KKB_CC_MAX_O_C_LIM_L3Y
0.87	KKB_CC_MAX_O_C_LIM_L3Y	<=>	KKB_CC_MAX_O_C_LIM_L1Y
0.87	KKB_CC_MAX_O_C_LIM_L1Y	<=>	KKB_CC_O_DLQ0_LIM
0.87	KKB_INST_O_MAX_CRE_AMT	<=>	KKB_O_HL_FINANCE_AMT
0.87	MAH_ILC_Y_5_9	<=>	MAH_ILC_Y_10_14
0.86	MAH_ILC_Y_65_	<=>	MAH_ILC_Y_55_59
0.86	MAH_ILC_Y_10_14	<=>	MAH_ILC_Y_25_29
0.85	KKB_O_HL_FINANCE_AMT	<=>	MEMZUC_ALL_CASH_RISK
0.85	MAH_ILC_Y_10_14	<=>	MAH_ILC_Y_25_29
0.85	MAH_ILC_Y_25_29	<=>	MAH_ILC_Y_30_34
0.84	MAH_ILC_Y_55_59	<=>	MAH_ILC_Y_60_64
0.84	MAH_ILC_Y_60_64	<=>	WIDOW_DIVORCED_FLAG
0.84	MAH_ILC_Y_20_24	<=>	MAH_ILC_Y_15_19
0.84	MAH_ILC_Y_0_4	<=>	MAH_ILC_Y_5_9
0.84	MAH_ILC_Y_25_29	<=>	MAH_ILC_Y_30_34
0.84	MAH_ILC_Y_30_34	<=>	MAH_ILC_Y_35_39
0.83	MAX24_RL_INST	<=>	MAX12_RL_INST
0.83	KKB_APP_CNT_L3M	<=>	KKB_APP_CNT_L6M
0.83	MAH_ILC_Y_0_4	<=>	MAH_ILC_Y_5_9
0.82	KKB_CC_O_MAX_LIM	<=>	KKB_CC_MAX_O_C_LIM_L3Y
0.82	KKB_CC_MAX_O_C_LIM_L3Y	<=>	KKB_CC_MAX_O_C_LIM_L1Y
0.82	KKB_CC_MAX_O_C_LIM_L1Y	<=>	KKB_CC_O_DLQ0_LIM
0.82	MAX24_GPL_INST	<=>	MAX12_GPL_JNST
0.82	MAH_ILC_YETISKIN_UNI_ORAN	<=>	MAH_ILC_Y_65_
0.82	MAH_ILC_YETISKIN_LISE_ORAN	<=>	MAH_ILC_Y_0_4
0.82	MAH_ILC_Y_5_9	<=>	MAH_ILC_Y_10_14
0.81	KKB_CC_O_MAX_LIM	<=>	KKB_CC_M.AX_O_C_LIM_L3Y
0.81	KKB_CC_MAX_O_C_LIM_L3Y	<=>	KKB_CC_MAX_O_C_LIM_L1Y
0.81	KKB_CC_MAX_O_C_LIM_L1Y	<=>	KKB_CC_O_DLQ0_LIM
0.81	KKB_O_HL_CNT	<=>	MEMZUC_ALL_CASH_LIMIT
0.81	MEMZUC_ALL_CASH_LIMIT	<=>	KKB_INST_O_MAX_CRE_AMT
0.81	MAH_ILC_YETISKIN_UNI_ORAN	<=>	MAH_ILC_Y_65_
0.81	MAH_ILC_Y_20_24	<=>	MAH_ILC_Y_15_19
0.81	MAH_ILC_Y_10_14	<=>	MAH_ILC_Y_25_29
0.81	MAH_ILC_Y_25_29	<=>	MAH_ILC_Y_30_34
0.80	MAH_ILC_YETISKIN_UNI_ORAN	<=>	MAH_ILC_Y_65_
0.80	MAH_ILC_YETISKIN_LISE_ORAN	<=>	MAH_ILC_Y_0_4
0.80	MAH_ILC_YETISKIN_LISE_ORAN	<=>	MAH_ILC_Y_0_4
0.80	MAH_ILC_Y_5_9	<=>	MAH_ILC_Y_10_14
-0.88	MARRIED_FLAG	<=>	GENDER_E_FLAG
-1.00	GENDER_E_FLAG	<=>	KKB_INST_C_M.AX_INST_AMT_L3Y

4.4.4.3 Determination of Variables to Exclude

4.4.4.4 According to the above list, all LM (last month) variables has a high correlation with L3M (Last 3 Month) or L6M (Last 6 Month). Therefore, removing all LM variables makes sense to avoid correlated variables.

As first step, all LM variables were removed from DT. In addition to this, before training process, base variables (that are PARTY_ID and YEAR_MONTH) and target variable (which is TARGET_PFA) were removed. Also type of base variables were factor. In XGBoost algoritm, factor type data cannot be allowed. That is why they were eliminated.

As second step, variables that lead to zero standard deviation were removed. TARGET_FLAG and TARGET_FLAG2 variables indicate the mass of the model to be scored. Therefore, we cannot put these variables into the training stage.

Variables that lead to zero standard deviation
"TARGET_FLAG" ,
"TARGET_FLAG2" ,
"KKB_OPEN_NREV_MAX_PMT_AMT_LM",
"KKB_OPEN_NREV_TOT_PMT_AMT_LM",
"GENDER_NULL_FLAG" ,
"KKB_C_CNT_DEF_DL36" ,
"KKB_OPEN_NREV_AVG_PMT_AMT_LM",
"KKB_O_PL_LAST_INS_AMT",
"BEH_CLUSTER0",
"MAH_CLUSTER0",
"MAH_CLUSTER2"

Figure 4-14 Variables that Lead to Zero Standard Deviation

As third step, all high correlated variables were examined in detail to find which variables should enter the model.

Table 4-2Selection Variables from High-correlated Pairs

Variables	High Corrrelated Matches		Variables to be removed	Why?
WIDOW_DIVORCED_FLAG		MAH_ILC_Y_60_64	MAH_ILC_Y_60_64	The widowed and divorced segments correlated with the 60-64 age group in the neighborhood data. So, MAH_ILC_Y_60_64 is removed.
MARRIED_FLAG	GENDER_E_FLAG		**X**	Okay, they are correlated variables. On the other hand, they are so different variables. So, keeping them makes sense.
GENDER_E_FLAG	KKB_INST_C_MAX_INST_AMT_L3Y	MARRIED_FLAG	KKB_INST_C_MAX_INST_AMT_L3Y	In this correlation, male flag as gender is correlated with both KKB_INST_C_MAX_INST_AMT_L3Y (-100%) and MARRIED_FLAG(-88%). So, it can be understood that it is so high correlated with KKB_INST_C_MAX_INST_AMT_L3Y.

KKB_APP_CNT_L6M		KKB_APP_CNT_L3M	KKB_APP_CNT_L3M	In this situation, the number of applications made within the last 6 months and last 3 months as a result of the KKB query are high correlated. In that order, the reason of last 3 months inside of last 6 months, KKB_APP_CNT_L3M variable is removed.
KKB_O_HL_CNT	MEMZUC_ALL_CASH_LIMIT		MEMZUC_ALL_CASH_LIMIT	In employed & retired segment, MEMZUC variables will not be used. Therefore, MEMZUC_ALL_CASH_LIMIT is removed.
KKB_NO_OPEN_CARDS	KKB_CC_O_CRE_CNT		KKB_NO_OPEN_CARDS	More or less two variables are so close each other. So, this high correlation makes sense. KKB_CC_O_CRE_CNT variable contains KKB_NO_OPEN_CARDS. So KKB_NO_OPEN_CARDS is removed.
KKB_CC_MAX_O_C_LIM_L3Y	KKB_CC_MAX_O_C_LIM_L1Y	KKB_CC_O_MAX_LIM	KKB_CC_MAX_O_C_LIM_L1Y	In this situation, last 1 month and last 3 months are correlated. Removing KKB_CC_MAX_O_C_LIM_L1Y makes sense.
KKB_CC_O_TOT_LIM	KKB_REV_O_TOT_LIM	KKB_CC_O_DLQ0_LIM	KKB_REV_O_TOT_LIM	As seen, KKB_CC_O_TOT_LIM variable has unaccepted correlation value that is 91% with both KKB_CC_O_DLQ0_LIM and KKB_REV_O_TOT_LIM. Therefore, the most useful variable should be chosen. In our case, total credit card delinquency limit is more important than customer revolving credit total limit.
KKB_CC_O_TOT_RISK	KKB_CC_O_MAX_RISK		KKB_CC_O_MAX_RISK	In that order, if a decision had to be made between maximum and total risk, the total one would have been chosen. Because total value is more obvious which has also maximum value inside of it. So, KKB_CC_O_MAX_RISK is removed.
KKB_INST_O_MAX_CRE_AMT		MEMZUC_ALL_CASH_LIMIT	KKB_INST_O_MAX_CRE_AMT	MEMZUC_ALL_CASH_LIMIT variable was already removed before, also determined that KKB_INST_O_MAX_CRE_AMT variable is not important for the project. Because it contains maximum value. In the model, KKB_INST_O_TOT_CRE_AMT is already existing.
KKB_INST_O_MAX_RISK		KKB_INST_O_TOT_RISK	KKB_INST_O_MAX_RISK	In that order, if a decision had to be made between maximum and total installment, the total would have been chosen. Because total value is more obvious, which has also maximum value. So, KKB_INST_O_MAX_RISK is removed.
KKB_OD_O_MAX_RISK		KKB_OD_O_TOT_RISK	KKB_OD_O_MAX_RISK	In this order, if a decision had to be made between the maximum and the total installment, the total would have been chosen. Because the total value is more obvious, which has the maximum value. This removes KKB_INST_O_MAX_RISK.
KKB_O_HL_FINANCE_AMT	MEMZUC_ALL_CASH_RISK	KKB_INST_O_MAX_CRE_AMT	MEMZUC_ALL_CASH_RISK KKB_INST_O_MAX_CRE_AMT	Firstly, in employed & retired segment, MEMZUC variables will not be used. Therefore, MEMZUC_ALL_CASH_RISK is removed. Secondly, the other variables are correlated. It is dangerous. So KKB_INST_O_MAX_CRE_AMT is removed due to lack of importance.
MAX24_RL_INST	MAX12_RL_INST		MAX12_RL_INST	In this comparison, MAX24_RL_INST and MAX6_RL_INST variables are chosen. Because they are not correlated with each other except MAX12_RL_INST. That is why it is removed.
MAX6_RL_INST		MAX12_RL_INST		

MAX24_GPL_INST	MAX12_GPLJNST		MAX12_GPL_INST	In this comparison, MAX24_GPL_INST and MAX6_GPL_INST variables are chosen. As they are not correlated with each other, except MAX12_GPL_INST. That is why it is removed.
MAX6_GPL_INST		MAX12_GPL_INST		
KKB_LOAN_O_CRE_CNT		KKB_O_CRE_CNT	KKB_LOAN_O_CRE_CNT	KKB_O_CRE_CNT variable contains CC, OD and Loan. So if these two variables are correlated, KKB_LOAN_O_CRE_CNT should be removed.
MAH_ILC_Y_50_54		MAH_ILC_Y_45_49	**X**	No variables can be extracted, even if they are correlated with each other. Because both variables show different information.
MAH_ILC_AVG_HANE_SAVINGS		MAH_ILC_AVG_GELIR	MAH_ILC_AVG_HANE_SAVINGS	In this project, the aim is to obtain true customer income information that is so significant for estimation. Therefore, MAH_ILC_AVG_HANE_SAVINGS is removed.

4.5 Training, Validation and Test Sets: Splitting Data

```
#############################################################
#############    Create train/test/oot samples  #################
#############################################################

#split 75-25

modelDevSeq <- sample(1:nrow(DT), round(nrow(DT)*0.75) , replace = FALSE)
modelDevSample  <- DT[modelDevSeq]
modelTestSample <- DT[!modelDevSeq] #dropping modelDevSeq table from DT and then getting modelTestSample
```

Figure 4-15 Splitting Data with 75-25 ratio

In machine learning, a common task is the study and construction of algorithms that can learn from and make predictions on data. Such algorithms work by making data-driven predictions or decisions, through building a mathematical model from input data.

The data used to build the final model usually comes from multiple datasets. In particular, three data sets are commonly used in different stages of the creation of the model.

4.5.1 Training Dataset

The model is initially fit on a training dataset that is a set of examples used to fit the parameters (e.g. weights of connections between neurons in artificial neural networks) of the model. The model (e.g. a neural net or a naive Bayes classifier) is trained on the training dataset using a supervised learning method (e.g. gradient descent or stochastic gradient descent). In practice, the training dataset often consist of pairs of an input vector (or scalar) and the corresponding output vector (or scalar), which is commonly denoted as the target (or label). The current model is run with the training dataset and produces a result, which is then compared with the target, for each input vector in the training dataset. Based on the result of the comparison and the specific learning

algorithm being used, the parameters of the model are adjusted. The model fitting can include both variable selection and parameter estimation.

4.5.2 Validation Dataset

The successfully fitted model is used to predict the responses for the observations in a second dataset called the validation dataset. The validation dataset provides an unbiased evaluation of a model fit on the training dataset while tuning the model's hyperparameters. Validation datasets can be used for regularization by early stopping that stops training when the error on the validation dataset increases, as this is a sign of overfitting to the training dataset. [30]

Usage of validation dataset normally is not common. In the model, validation set was not created.

4.5.3 Test Dataset

A test dataset is a dataset that is independent of the training dataset, but that follows the same probability distribution as the training dataset. If a model fit to the training dataset also fits the test dataset well, minimal overfitting has taken place as it can be seen in the below figure. A better fitting of the training dataset as opposed to the test dataset usually points to overfitting.

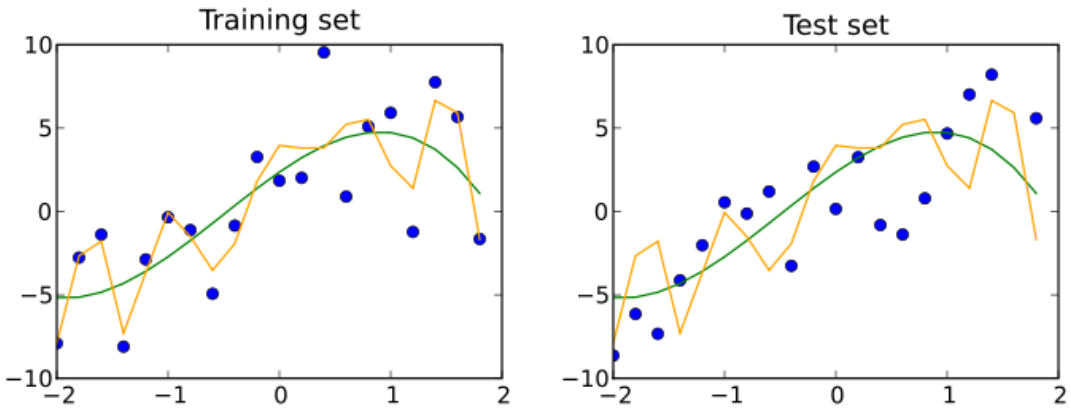


Figure 4-16 Training and Test set Schemas

A training set (left) and a test set (right) from the same statistical population are shown as blue points. Two predictive models are fit to the training data. Both fitted models are plotted with both the training and test sets. In the training set, the MSE of the fit shown in orange is 4 whereas the MSE for the fit shown in green is 9. In the test set, the MSE for the fit shown in orange is 15 and the MSE for the fit shown in green is 13. The orange curve severely overfits the training data, since its MSE increases by almost a factor of four when comparing the test set to the training set. The green curve overfits the training data much less, as its MSE increases by less than a factor of 2. [30]

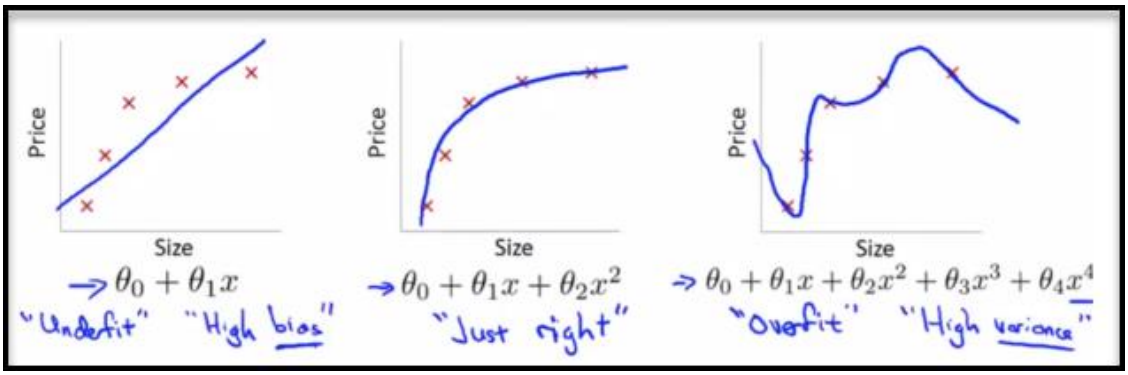


Figure 4-17 Graphics of Underfit, Overfit, High Bias and Variance, Best Model

4.5.4 Cross-validation

A dataset can be repeatedly split into a training dataset and a validation dataset: this is known as cross-validation. These repeated partitions can be done in various ways, such as dividing into 2 equal datasets and using them as training/validation, and then validation/training, or repeatedly selecting a random subset as a validation dataset [citation needed]. To validate the model performance, sometimes an additional test dataset that was held out from cross-validation is used. Cross-validation doesn't work in situations where you can't shuffle your data, most notably in time-series. [30]

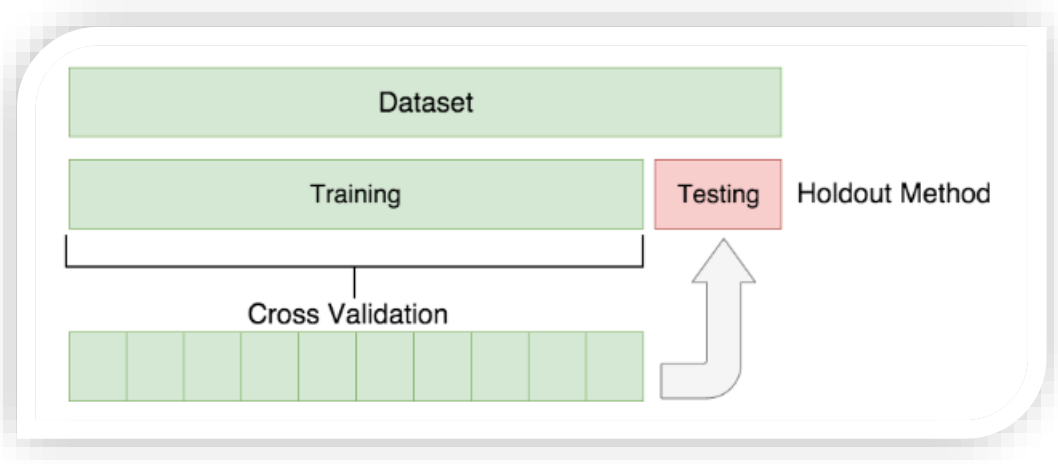


Figure 4-18 Cross Validation Schema

4.6 XGBoost Algorithm in R

4.6.1 Preparation of Data for Using XGBoost

Model data has categorical data which are variables that contain label values rather than numeric values. On the other hand, XGBOOST only works with numeric vectors. Therefore, converting all other forms of data into numeric vectors is needed. A simple method to convert categorical variable into numeric vector is “One Hot Encoding”. This term emanates from digital circuit language, where it means an array of binary signals and only legal values are 0s and 1s. [31]

4.6.1.1 Conversion from Categorical Data to Numeric Data

4.6.1.1.1 Label Encoding (Integer Encoding)

Each unique category value is assigned an integer value. The integer values have a natural ordered relationship between each other, and machine learning algorithms may be able to understand and harness this relationship. [32]

Let's assume we're working with categorical data, like cats and dogs. Looking at the name of label encoding, you might be able to guess that it encodes labels, where label is just a category (i.e. cat or dog), and encode just means giving them a number to represent that category (1 for cat and 2 for dog). By giving each category a number, the computer now knows how to represent them, since the computer knows how to work with numbers. [33]

4.6.1.1.2 One Hot Encoding

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. The categorical values start from 0 goes all the way up to N-1 categories. [34]

4.6.1.1.3 Comparison between Label and One Hot Encoding

In label encoding, the problem is that, the categories have natural ordered relationships. The computer does this because it's programmed to treat higher numbers as higher numbers; it will naturally give the higher numbers higher weights. We can see the problem with this in an example:

Imagine if you had 3 categories of foods: apples, chicken, and broccoli. Using label encoding, you would assign each of these a number to categorize them: apples = 1, chicken = 2, and broccoli = 3. But now, if your model internally needs to calculate the average across categories, it might do 1+3 = 4/2 = 2. This means that according to your model, the average of apples and chicken together is broccoli. Obviously that line of thinking by your model is going to lead to it getting correlations completely wrong, so we need to introduce one-hot encoding.

Rather than labeling things as a number starting from 1 and then increasing for each category, we'll go for more of a binary style of categorizing. You might have been thinking that if you knew what a one-hot is. Difference between label and one-hot encoding works like that:

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

→

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

Figure 4-19 Label and One Hot Encoding

In label encoding, categories were formerly rows, but in one hot encoding table as it can be seen they're columns. In other words, In the table there is no categorical column, apart from it there is an additional binary column for each category. [35]

4.6.1.2 Conversion from Categorical Data to Numeric Data (One Hot Encoding in R)

This step will essentially make a sparse matrix using flags on every possible value of that variable. Sparse Matrix is a matrix where most of the values of zeros and '0' are not stored in memory. Therefore, in a dataset mainly made of '0', memory size is reduced. It is very usual to have such dataset. Conversely, a dense matrix is a matrix where most of the values are non-zeros.

4.6.2 Creation of the Sparse Model Matrices; Label and Dense Matrix for All Samples

In R, sparse model matrices for modelDevSample and modelTestSample data were created. The methodology of creating sparse matrix is given under the below.

```
sparse_matrix <- sparse.model.matrix(TARGET_PFA ~ .-1, data = modelDevSample)
```

When the code breaks down, "sparse.model.matrix" is the command and all other inputs inside parentheses are parameters. The parameter "TARGET_PFA ~. - 1" says that this statement should ignore "TARGET_PFA" variable. Because the aim is to estimate TARGET_PFA. "-1" removes an extra column which this command creates as the first column and finally, dataset name can be specified.

```
Label_train <- modelDevSample$TARGET_PFA
```

"modelDevSample$TARGET_PFA" defines the `label` which is the outcome of our dataset.

The data are stored in a 'dgCMatrix' which is a sparse matrix and 'label' vector is a 'numeric' vector.

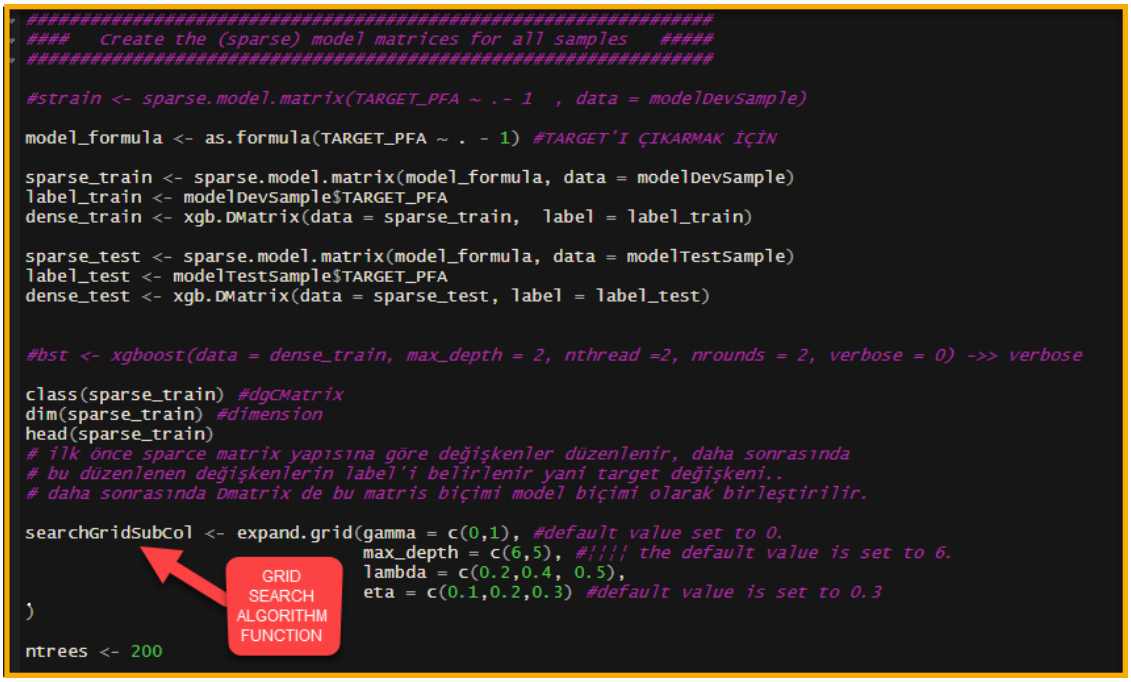
XGBoost offers a way to group them in a 'xgb.DMatrix'. You can even add other meta data in it. It will be useful for the most advanced features we will discover later.

```
dense_train <- xgb.DMatrix(data = train$data, label = train$label)
```

The train data is used. As explained above, both 'data' and 'label' are stored in a 'list'.

4.6.3 Grid Searching Algorithm

The grid search algorithm tries to find the best combination with the lowest error rate over the given values. In this case, given parameters are gamma, maximum depth, lambda and eta. Also, there are other parameters. But parameter usage may vary depending on the user.




```
#################################################################
####   Create the (sparse) model matrices for all samples   #####
#################################################################

#strain <- sparse.model.matrix(TARGET_PFA ~ .- 1  , data = modelDevSample)

model_formula <- as.formula(TARGET_PFA ~ . - 1) #TARGET'I ÇIKARMAK İÇİN

sparse_train <- sparse.model.matrix(model_formula, data = modelDevSample)
label_train <- modelDevSample$TARGET_PFA
dense_train <- xgb.DMatrix(data = sparse_train,  label = label_train)

sparse_test <- sparse.model.matrix(model_formula, data = modelTestSample)
label_test <- modelTestSample$TARGET_PFA
dense_test <- xgb.DMatrix(data = sparse_test, label = label_test)


#bst <- xgboost(data = dense_train, max_depth = 2, nthread =2, nrounds = 2, verbose = 0) ->> verbose

class(sparse_train) #dgCMatrix
dim(sparse_train) #dimension
head(sparse_train)
# ilk önce sparce matrix yapısına göre değişkenler düzenlenir, daha sonrasında
# bu düzenlenen değişkenlerin label'i belirlenir yani target değişkeni..
# daha sonrasında Dmatrix de bu matris biçimi model biçimi olarak birleştirilir.

searchGridSubCol <- expand.grid(gamma = c(0,1), #default value set to 0.
                                max_depth = c(6,5), #!!!! the default value is set to 6.
                                lambda = c(0.2,0.4, 0.5),
                                eta = c(0.1,0.2,0.3) #default value is set to 0.3
)

ntrees <- 200
```

Figure 4-20 Creating the sparse model matrices for all samples and Grid Search Parameters

Gamma, max_depth, lambda and eta parameters are so common to use. Therefore, above parameter default values were obtained to find out the best boosting model. Default values were determined according to speed of the train model and closeness of default value of parameters. In that case, model could be faster and at the same time more optimal.

Table 4-3 Grid Search 36 Combination and Their RMSE Values as Train and Test

	gamma	max_depth	lambda	eta
1	0	6	0.2	0.1
2	1	6	0.2	0.1
3	0	5	0.2	0.1
4	1	5	0.2	0.1
5	0	6	0.4	0.1
6	1	6	0.4	0.1
7	0	5	0.4	0.1
8	1	5	0.4	0.1
9	0	6	0.5	0.1
10	1	6	0.5	0.1
11	0	5	0.5	0.1
12	1	5	0.5	0.1
13	0	6	0.2	0.2
14	1	6	0.2	0.2
15	0	5	0.2	0.2
16	1	5	0.2	0.2
17	0	6	0.4	0.2
18	1	6	0.4	0.2
19	0	5	0.4	0.2
20	1	5	0.4	0.2
21	0	6	0.5	0.2
22	1	6	0.5	0.2
23	0	5	0.5	0.2
24	1	5	0.5	0.2
25	0	6	0.2	0.3
26	1	6	0.2	0.3
27	0	5	0.2	0.3
28	1	5	0.2	0.3
29	0	6	0.4	0.3
30	1	6	0.4	0.3
31	0	5	0.4	0.3
32	1	5	0.4	0.3
33	0	6	0.5	0.3
34	1	6	0.5	0.3
35	0	5	0.5	0.3
36	1	5	0.5	0.3

Test RMSE	Train RMSE	currentGamma	currentLambda	currentDepth	currentEta
191862.7469	134563.6375	1	0.4	6	0.1
191909.0281	135150.0938	1	0.5	6	0.1
192954.6	111235.4812	1	0.5	6	0.2
192996.6188	134941.3094	1	0.2	6	0.1
193331.7219	134308.2469	0	0.4	6	0.1
193371.8406	110986.325	0	0.5	6	0.2
193418.7406	151562.5781	1	0.2	5	0.1
193508.275	152480.4031	0	0.4	5	0.1
193531.3906	133688.1344	0	0.2	6	0.1
193748.5906	135554.175	0	0.5	6	0.1
193754.1719	110486.9344	1	0.4	6	0.2
193795.0563	152847.125	0	0.5	5	0.1
194207.3688	152523.1156	1	0.4	5	0.1
194522.9313	110061.1375	1	0.2	6	0.2
194690.8063	133694.9531	0	0.5	5	0.2
194732.5031	151728.7188	0	0.2	5	0.1
194776.6531	133481.1781	1	0.4	5	0.2
194864.975	110233.4688	0	0.2	6	0.2
194952.025	133268.2063	0	0.4	5	0.2
195029.8719	153308.2813	1	0.5	5	0.1
195163.4438	110599.0719	0	0.4	6	0.2
196246.9813	133425.5438	1	0.5	5	0.2
196247.0938	132284.5281	0	0.2	5	0.2
196275.7281	131924.4031	1	0.2	5	0.2
196612.6594	118691.8219	1	0.5	5	0.3
196803.7344	120124.6094	0	0.5	5	0.3
197662.7375	118977.6984	0	0.4	5	0.3
197771.65	92786.7125	1	0.2	6	0.3
197774.5281	94370.97031	0	0.4	6	0.3
198032.4	93143.16875	0	0.2	6	0.3
198100.2063	93332.60156	1	0.4	6	0.3
198249.2719	119066.3359	1	0.4	5	0.3
198543.8094	94194.02344	1	0.5	6	0.3
198643.6594	95039.075	0	0.5	6	0.3
199103.8469	117429.2828	1	0.2	5	0.3
199232.3594	117822.2281	0	0.2	5	0.3

T

4.6.4 Parameters used in XGBoost Model

Every parameter has a significant role to play in the model's performance. There are three types of parameters: General Parameters, Booster Parameters and Task Parameters.

- ❖ General parameters control the booster type in the model which eventually drives overall functioning.
- ❖ Booster parameters controls the performance of the selected booster.
- ❖ Learning Task parameters that sets and evaluates the learning process of the booster from the given data

4.6.4.1 General Parameters

- ♠ Silent [default = 0]: The default value is 0. It needs to specify 0 for printing running messages, 1 for silent mode. If it is set to 1, R console will get flooded with running messages. But better not to change it.

- ♠ Booster [default = gbtree]: The default value is gbtree. It sets the booster type (gbtree-tree based, gblinear-linear function or dart) to use. In the project “dart” booster was chosen. Dart can be used with tree-based models.
- ♠ Nthread [default=maximum cores available]: Nthread activates parallel computation. Generally, people do not change it as using maximum cores leads to the fastest computation.

4.6.4.2 Booster Parameters

Parameters for tree and linear boosters are different. Therefore, parameters for tree booster will be explained.

- ♠ Eta [default=0.3] [range: (0,1)]: The default value is set to 0.3. It controls the learning rate. (the rate at which our model learns patterns in data.) After each boosting step, the weights of new features can be gotten. eta shrinks the feature weights to make the boosting process more conservative and to reach the best optimum. The range is 0 to 1. Low eta value means model is more robust to overfitting. Lower eta leads to slower computation. It must be supported by increase in nrounds. c (0.1,0.2,0.3) was determined for the model.
- ♠ gamma[default=0] [range: (0, Inf)]: The default value is set to 0. You need to specify minimum loss reduction required to make a further partition on a leaf node of the tree. The larger, the more conservative the algorithm will be. The range is 0 to ∞. Larger the gamma more conservative the algorithm is.
- ♠ max_depth [default=6] [range: (0, Inf)]: The default value is set to 6. You need to specify the maximum depth of a tree. The range is 1 to infinity (∞). c (6,5): the trees will be a bit deep, because our case is complicate.
- ♠ lambda[default=0]: L2 regularization term on weights (analogous to Ridge regression) This used to handle the regularization part of XGBoost. Though many data scientists don’t use it often, it should be explored to reduce overfitting. Increasing this value will make model more conservative.

4.6.4.3 Learning Task Parameters

These parameters are used to define the optimization objective the metric to be calculated at each step:

```
rmseErrorsHyperparameters <- apply(searchGridSubCol, 1, function(parameterList){
  #Extract Parameters to test
  currentGamma <- parameterList[["gamma"]]
  currentDepth <- parameterList[["max_depth"]]
  currentLambda <- parameterList[["lambda"]]
  currentEta <- parameterList[["eta"]]
  xgboostModelCV <- xgb.cv(data =  dense_train,
                           nrounds = ntrees,
                           nfold = 5,
                           showsd = F,
                           print_every_n = 10, #her 10 tanede bir öğren, her 10 dallanmada öğren.
                           "eval_metric" = "rmse",
                           "objective"   = "reg:linear",
                           "booster" = "dart",
                           "gamma" = currentGamma,
                           "eta" = currentEta,
                           "lambda" = currentLambda,
                           "max.depth" = currentDepth,
                           "seed" = 123456,
                           maximize = TRUE)

  xvalidationScores <- as.data.table(xgboostModelCV$evaluation_log)
  rmse <- tail(xvalidationScores$test_rmse_mean, 1)
  trmse <- tail(xvalidationScores$train_rmse_mean,1)
  output <- c(rmse, trmse, currentGamma,currentLambda , currentDepth, currentEta)})
```

Figure 4-21 XGBoost Parameters

- ♠ objective [default=reg:linear]: we will train a continuous classification model This defines the loss function to be minimized.
- ♠ nrounds: there will be 200 passes on the data, the second one will enhance the model by further reducing the difference between ground truth and prediction.
- ♠ eval_metric [default according to objective]: The metric to be used for validation data. The default values are rmse for regression and error for classification. Typical values are:
  - ✓ rmse – root mean square error: used one
  - ✓ mae – mean absolute error
  - ✓ logloss – negative log-likelihood
  - ✓ error – Binary classification error rate (0.5 threshold a different than 0.5 binary classification threshold value could be specified by providing a numerical value through ‘t’.)
  - ✓ merror – Multiclass classification error rate
  - ✓ mlogloss – Multiclass logloss
  - ✓ auc: Area under the curve
- ♠ seed [default=0]: The random number seed. Can be used for generating reproducible results and also for parameter tuning.

At the end of grid search, output is written to excel file and the lowest test set rmse is chosen. Then in cross validation process, gamma, eta, max_depth and lambda parameters of lowest combination is changed as shown in the blow figure. [36]

Figure 4-22 Cross Validation with XGBoost

Firstly, XGBoost parameters are generated: objective(reg:linear), booster(dart) and eval_metric(rmse). Then apart from params, the combination that had the lowest rmse test ratio as a result of grid search algorithm is written. “gamma = 1, eta = 0.1, lambda = 0.4 and max_depth = 6.”

Using the xgb.cv function, the aim is calculation of the best nround for this model. In addition, this function also returns CV (cross validation) error, which is an estimate of test error. In this section the main idea is getting rid of overfitting. In that way early_stopping_rounds parameter (the number of rounds with no improvement in the evaluation metric in order to stop the training) is very significant one. The idea of usage is that model starts to learn and rmse is calculated for every n (because as it can be seen “print_every_n =1” was written that will be gotten 200 train and test rmse value.) By this time thanks to this parameter, rmse ratio in every n will be calculated in order to train and test error will be following each other. Machine will analyze and when it finds the lowest rmse value, it will look for other possible lowest error rate in 30 rounds. If machine cannot find the lowest value in 30 rounds, it will stop and eventually it will stop looking for another rmse value.

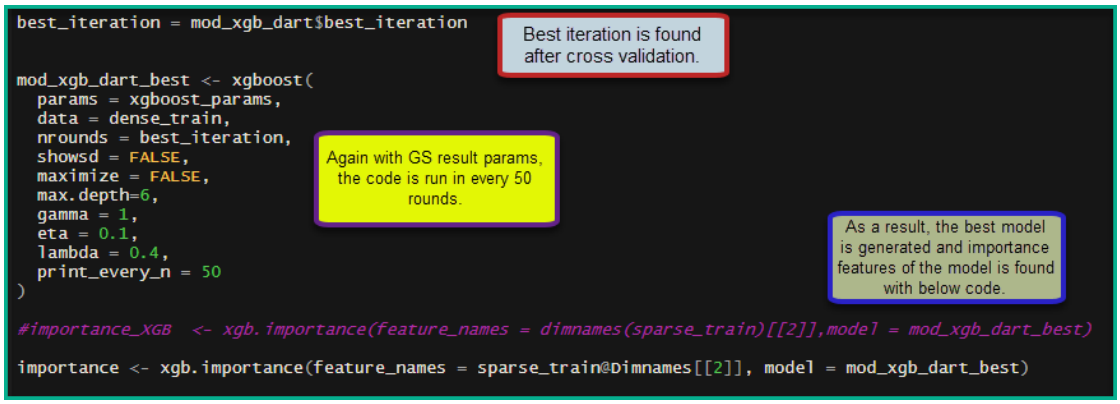


Figure 4-23 Running the Model with the Best Combination from GS

4.7 Prediction with Best Model

In this section, linear regression, random forest, gradient boosting algorithms will be used to compare results with XGBoost.

4.7.1 Prediciton with XGBoost Algorithm

With “predict” function, best training model was predicted with dense train and test data.

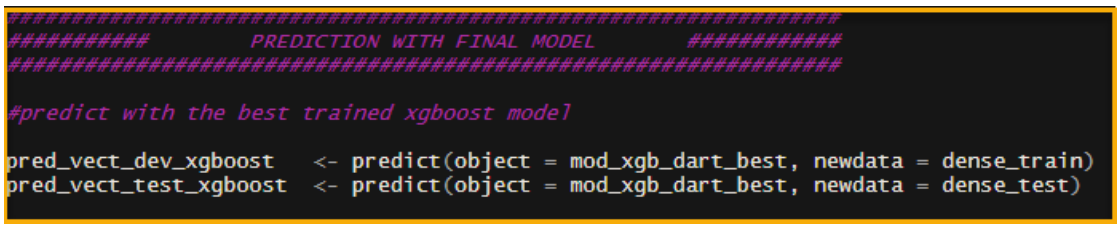


Figure 4-24 Prediction with h2o

4.7.2 Prediction with H2O package Algorithms

H2O includes many common machine learning algorithms, such as generalized linear modeling (linear regression, logistic regression, etc.), Naive Bayes, principal components analysis, k-means clustering, and word2vec. H2O implements best-in-class algorithms at scale, such as distributed random forest, gradient boosting, and deep learning. H2O also includes a Stacked Ensembles method, which finds the optimal combination of a collection of prediction algorithms using a process known as” stacking.” With H2O, customers can build thousands of models and compare the results to get the best predictions.

First of all, “as.h2o ()” transfers data from R to the H2O instance.

```
as.h2o (ModelDevSample, destination_frame = "ModelDevSample.hex")

as.h20 (ModelTestSample, destination_frame = "ModelDevSample.hex")
```

.hex: The dataset is assigned an identifier (the .hex file type in H2O) used as a reference in commands to the web server. [37]

4.7.2.1 Prediction with Linear Regression Algorithm

4.7.2.1.1 Fitting a Generalized Linear Model Function and Parameters

```
#converts R objects "modelDevSample" into h2o object "modelDevSample.hex"
#converts R objects "modelTestSample" into h2o object "modelTestSample.hex"
#import data-table to h2o cluster platform


modelDevSample.hex<- as.h2o(modelDevSample, destination_frame= "modelDevSample.hex")
modelTestSample.hex<- as.h2o(modelTestSample, destination_frame= "modelTestSample.hex")


myX =setdiff(colnames(modelDevSample.hex ), c("TARGET_PFA"))

regression.model <- h2o.glm( y = "TARGET_PFA",
                             x = myX,
                             training_frame = modelDevSample.hex,
                             family = "gaussian",
                             lambda = 0,
                             remove_collinear_columns = TRUE,
                             compute_p_values = TRUE)

h2o.performance(regression.model)

pred_vect_dev_glm <- as.data.table(h2o.predict(regression.model,modelDevSample.hex))
pred_vect_test_glm <- as.data.table(h2o.predict(regression.model,modelTestSample.hex))
```

Regression model is run with these parameters

Figure 4-25 GLM Prediction

- ❖ h2o.glm (): Fits a generalized linear model, specified by a response variable, a set of predictors, and a description of the error distribution.
- ✓ x: (myX) A vector containing the names or indices of the predictor variables to use in building the model.

```
myX = setdiff (colnames(modelDevSample.hex), c("TARGET_PFA"))
```

"setdiff(colnames(matrix), not_target_column)" structure duplicates columns with the opposite of the column name in R.[1]

- ✓ y: The name or column index of the response variable in the data. The response must be either a numeric or a categorical/factor variable. If the response is numeric, then a regression model will be trained.
- ✓ training_frame: ID of the training data frame which is modelDevSample.hex in the code.
- ✓ family: Uses binomial for classification with logistic regression, others are for regression problems. Must be one of: "gaussian", "binomial", "quasibinomial", "ordinal", "multinomial", "poisson", "gamma", "tweedie", "negativebinomial". Defaults to gaussian, used default value in the model.
- ✓ lambda: Regularization strength. Defaults to zero (0).
- ✓ remove_collinear_columns: In case of linearly dependent columns, removes some of the dependent columns. Defaults to FALSE. But in the model, used TRUE.
- ✓ compute_p_values: Request p-values computation, p-values work only with IRLSM solver and no regularization. Defaults to FALSE. But in the model, used TRUE.
- ❖ h2o.performance (): a model performance metrics in h2o that computes the given trained h2o model performance.
- ❖ h2o.predict (): predicts on h2o model. "h2o.predict(object, newdata, ...)"

[1] https://stackoverflow.com/questions/42897160/how-can-i-select-columns-rows-with-the-opposite-of-the-column-row-names-in-r

- ✓ object: a fitted model object for which prediction is desired. In the model, regression.model is written as fitted model object.
- ✓ newdata: An H2O Frame object in which to look for variables with which to predict. In the model, modelDevSample.hex and modelTestSample.hex are used to predict.

4.7.2.1.2 Output of Prediction with Linear Regression

```
> modelDevSample.hex<- as.h2o(modelDevSample, destination_frame= "modelDevSample.hex")
  |=============================================================================================================| 100%
> modelTestSample.hex<- as.h2o(modelTestSample, destination_frame= "modelTestSample.hex")
  |=============================================================================================================| 100%
> myX =setdiff(colnames(modelDevSample.hex ), c("TARGET_PFA"))
> regression.model <- h2o.glm( y = "TARGET_PFA",
+                              x = myX,
+                              training_frame = modelDevSample.hex,
+                              family = "gaussian",
+                              lambda = 0,
+                              remove_collinear_columns = TRUE,
+                              compute_p_values = TRUE)
  |=============================================================================================================| 100%
> h2o.performance(regression.model)
H2ORegressionMetrics: glm
** Reported on training data. **

MSE:  41602098115
RMSE:  203965.9
MAE:  119361.1
RMSLE:  NaN
Mean Residual Deviance :  41602098115
R^2 :  0.2838148
Null Deviance :2.233501e+15
Null D.o.F. :38449
Residual Deviance :1.599601e+15
Residual D.o.F. :38400
AIC :1049375

> pred_vect_dev_glm <- as.data.table(h2o.predict(regression.model,modelDevSample.hex))
  |=============================================================================================================| 100%
> pred_vect_test_glm <- as.data.table(h2o.predict(regression.model,modelTestSample.hex))
  |=============================================================================================================| 100%
```

Figure 4-26 GLM Results

As seen in the figure, MSE, RMSE and MAE values can be discussed.

4.7.2.2 Prediction with Gradient Boosting Algorithm

4.7.2.2.1 Building Gradient Boosted Regression Trees

```
my_gbm <- h2o.gbm(x = myX,
                  y = "TARGET_PFA",
                  training_frame = modelDevSample.hex,
                  validation_frame = modelTestSample.hex,
                  distribution = "gaussian",
                  ntrees = 200,
                  max_depth = 6,
                  keep_cross_validation_predictions = TRUE,
                  nfolds = 5,
                  seed = 1)
h2o.performance(my_gbm,valid = TRUE)

pred_vect_dev_gbm <- as.data.table(h2o.predict(my_gbm,modelDevSample.hex))
pred_vect_test_gbm <- as.data.table(h2o.predict(my_gbm,modelTestSample.hex))
```

h2o.gbm function parameters

h2o performance measurement

Predicton with gradient boosting

Figure 4-27 GBM Prediction

- ❖ h2o.gbm (): Builds gradient boosted regression trees on a parsed data set. The default distribution function guesses the model type based on the response column type. In order to run properly, the response column must be a numeric for "gaussian" or an enum for "bernoulli" or "multinomial".
- ✓ x: (myX) A vector containing the names or indices of the predictor variables to use in building the model.
- ✓ y: The name or column index of the response variable in the data. The response must be either a numeric or a categorical/factor variable. If the response is numeric, then a regression model will be trained.

- ✓ training_frame: ID of the training data frame which is modelDevSample.hex in the code.
- ✓ validation_frame: ID of the validation data frame which is modelTestSample.hex in the code.
- ✓ distribution = Distribution function must be one of: "AUTO", "bernoulli", "quasibinomial", "multinomial", "gaussian", "poisson", "gamma", "tweedie", "laplace", "quantile", "huber", "custom". Defaults to AUTO. In the model "gaussian" is used as a distribution.
- ✓ ntrees: Number of trees. Defaults to 50. In the model 200 is defined as ntrees parameter.
- ✓ max_depth: Maximum tree depth. Defaults to 5. In the model 6 is defined as max_depth parameter.
- ✓ keep_cross_validation_predictions: Whether to keep the predictions of the cross-validation models. Defaults to FALSE. In the model this parameter is used with TRUE.
- ✓ nfolds: Number of folds for K-fold cross-validation (0 to disable or >= 2). Defaults to 0. In the model, obtained 5.
- ✓ seed: Seed for random numbers (affects certain parts of the algo that are stochastic and those might or might not be enabled by default) Defaults to -1 (time-based random number). In the model, obtained 1.
- ❖ h2o.performance (): a model performance metrics in h2o that computes the given trained h2o model performance.
- ❖ h2o.predict (): predicts on h2o model. "h2o.predict(object, newdata, ...)"
- ✓ object: a fitted model object for which prediction is desired. In the model, regression.model is written as fitted model object.
- ✓ newdata: An H2O Frame object in which to look for variables with which to predict. In the model, modelDevSample.hex and modelTestSample.hex are used to predict.

4.7.2.2.2 Output of Prediction with Gradient Boosting Algorithm

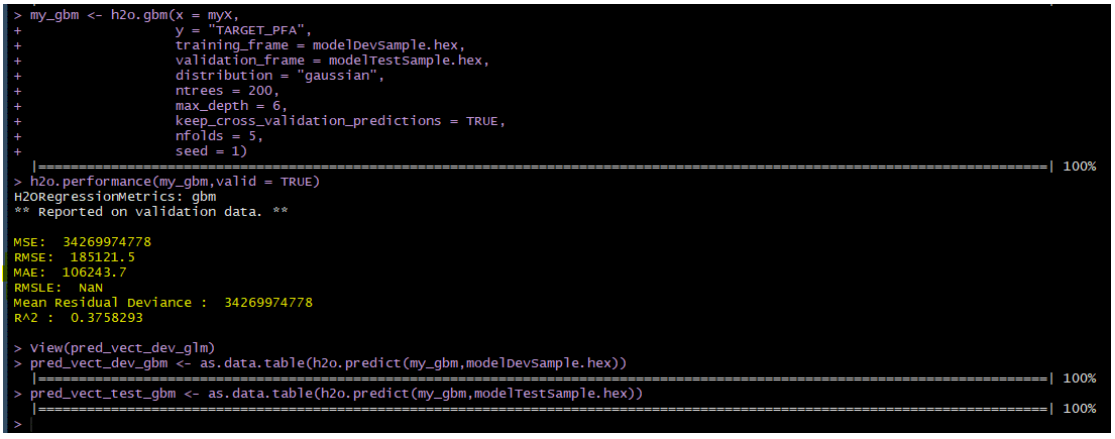


Figure 4-28 GBM Results

As seen in the figure, MSE, RMSE and MAE values can be discussed.

4.7.2.3 Prediction with Random Forest Algorithm

4.7.2.3.1 Building Random Forest Regression Trees

```
my_rf <- h2o.randomForest(x = myX,
                          y = "TARGET_PFA",
                          training_frame = modelDevSample.hex,
                          validation_frame = modelTestSample.hex,
                          ntrees = 10,
                          nfolds = 5,
                          max_depth=6,
                          keep_cross_validation_predictions = TRUE,
                          seed = 1)
h2o.performance(my_gbm,valid = TRUE)

pred_vect_dev_rf <- as.data.table(h2o.predict(my_rf,modelDevSample.hex))
pred_vect_test_rf <- as.data.table(h2o.predict(my_rf,modelTestSample.hex))
```

Parameters for h2o.randomforest

Performance measurement for GBM

Prediction with random forest algo

Figure 4-29 RF Prediction

- ❖ h2o.randomforest (): Builds a Random Forest model on an H2OFrame.
- ✓ x: (myX) A vector containing the names or indices of the predictor variables to use in building the model.
- ✓ y: The name or column index of the response variable in the data. The response must be either a numeric or a categorical/factor variable. If the response is numeric, then a regression model will be trained.
- ✓ training_frame: ID of the training data frame which is modelDevSample.hex in the code.
- ✓ validation_frame: ID of the validation data frame which is modelTestSample.hex in the code.
- ✓ distribution = Distribution function must be one of: "AUTO", "bernoulli", "quasibinomial", "multinomial", "gaussian", "poisson", "gamma", "tweedie", "laplace", "quantile", "huber", "custom". Defaults to AUTO. In the model "gaussian" is used as a distribution.
- ✓ ntrees: Number of trees. Defaults to 50. In the model 200 is defined as ntrees parameter.
- ✓ max_depth: Maximum tree depth. Defaults to 5. In the model 6 is defined as max_depth parameter.
- ✓ keep_cross_validation_predictions: Whether to keep the predictions of the cross-validation models. Defaults to FALSE. In the model this parameter is used with TRUE.
- ✓ nfolds: Number of folds for K-fold cross-validation (0 to disable or >= 2). Defaults to 0. In the model, obtained 5.
- ✓ seed: Seed for random numbers (affects certain parts of the algo that are stochastic and those might or might not be enabled by default) Defaults to -1 (time-based random number). In the model, obtained 1.
- ❖ h2o.performance (): a model performance metrics in h2o that computes the given trained h2o model performance.
- ❖ h2o.predict (): predicts on h2o model. "h2o.predict(object, newdata, ...)"
- ✓ object: a fitted model object for which prediction is desired. In the model, regression.model is written as fitted model object.
- ✓ newdata: An H2O Frame object in which to look for variables with which to predict. In the model, modelDevSample.hex and modelTestSample.hex are used to predict.

4.7.2.3.2 Output of Prediction with Random Forest

```
> my_rf <- h2o.randomForest(x = myX,
+                           y = "TARGET_PFA",
+                           training_frame = modelDevSample.hex,
+                           validation_frame = modelTestSample.hex,
+                           ntrees = 10,
+                           nfolds = 5,
+                           max_depth=6,
+                           keep_cross_validation_predictions = TRUE,
+                           seed = 1)
  |=============================================================================================================| 100%
> h2o.performance(my_gbm,valid = TRUE)
H2ORegressionMetrics: gbm
** Reported on validation data. **

MSE:  34269974778
RMSE:  185121.5
MAE:  106243.7
RMSLE:  NaN
Mean Residual Deviance :  34269974778
R^2 :  0.3758293

> pred_vect_dev_rf <- as.data.table(h2o.predict(my_rf,modelDevSample.hex))
  |=============================================================================================================| 100%
> pred_vect_test_rf <- as.data.table(h2o.predict(my_rf,modelTestSample.hex))
  |=============================================================================================================| 100%
>
```

Figure 4-30 RF Results

As seen in the figure, MSE, RMSE and MAE values can be discussed.

4.7.3 Measuring Feature Importance

Details can be seen with graphs perfectly. In that case, varimp_plot () function from h2o library was used to create performance schemes. In the model, there are notable amount of variable, therefore number of features were taken up to ten.

4.7.3.1 Variable Importance Plotting Function for XGBoost Trained Model

```
importance <- xgb.importance(feature_names = sparse_train@Dimnames[[2]], model = mod_xgb_dart_best)
```

Figure 4-31 Variable Importance with XGBoost Model

In the code below, sparse_matrix@Dimnames[[2]] represents the column names of the sparse matrix. These names are the original values of the features.

Table 4-4 XGBoost Variable Importance with Gain, Cover and Frequency

	Feature	Gain	Cover	Frequency
1	ESTIMATED_INCOME	0.2981838441	0.1444567627	0.108893324
2	CUST_AGE	0.1495833555	0.0786246345	0.142714571
3	MAX24_RL_INST	0.0908734082	0.0332113133	0.027722333
4	KKB_LOAN_LAST_PROD_TENURE	0.0541667736	0.0431839366	0.061321801
5	KKB_CC_O_TOT_RISK	0.0452076036	0.0378762571	0.053670437
6	KKB_TIME_SINCE_FIRST_CC	0.0347831106	0.0319959179	0.061876248
7	KKB_CC_MAX_O_C_LIM_L3Y	0.0256623676	0.0337318224	0.043579508
8	KKB_CC_LAST_PROD_TENURE	0.0256121967	0.0446182736	0.045353737
9	KKB_CC_2ND_MAX_O_C_LIM	0.0216026423	0.0441004305	0.041583500
10	KKB_CC_O_TOT_LIM	0.0178369596	0.0420910063	0.029496562
11	MAH_ILC_Y_45_49	0.0153427299	0.0325390993	0.020403637
12	MAH_ILC_Y_10_14	0.0137373656	0.0142332373	0.015524507
13	MAH_ILC_Y_0_4	0.0117582295	0.0183123429	0.016411621
14	MAH_ILC_Y_25_29	0.0115612889	0.0092739438	0.014193835
15	KKB_INST_C_MAX_CRE_AMT_L3Y	0.0109100820	0.0291653058	0.021290752
16	KKB_INST_O_TOT_RISK	0.0104518267	0.0067360124	0.009647372
17	MAH_ILC_YETISKIN_UNI_ORAN	0.0100534069	0.0077328130	0.012974052
18	MAH_ILC_Y_20_24	0.0094875018	0.0175849220	0.016189843
19	KKB_CC_C_MAX_LIM_L3Y	0.0094739026	0.0273921200	0.016078953
20	MAH_ILC_Y_15_19	0.0090254173	0.0198718760	0.015746285
21	MAX6_RL_INST	0.0080998201	0.0136833150	0.009869151
22	MAH_ILC_YETISKIN_LISE_ORAN	0.0076880647	0.0159047013	0.011199823
23	MAH_ILC_Y_5_9	0.0073566505	0.0130350954	0.012419605
24	MAH_ILC_Y_65_	0.0072964285	0.0097184812	0.010312708
25	MAH_ILC_Y_40_44	0.0072321583	0.0213810575	0.013750277
26	KKB_OPEN_CC_TOT_PMT_AMT_L3M	0.0070898956	0.0306495391	0.019183855
27	MAH_ILC_Y_50_54	0.0064481460	0.0068073673	0.011421601
28	GENDER_E_FLAG	0.0063882192	0.0096859033	0.008427589
29	MAH_ILC_Y_35_39	0.0062309341	0.0051130359	0.008760257
30	MAH_ILC_Y_30_34	0.0059645687	0.0093001709	0.011643380
31	MAH_ILC_Y_55_59	0.0054261176	0.0138341746	0.009758261
32	KKB_OPEN_CC_AVG_PMT_AMT_L3M	0.0050218621	0.0167294873	0.008871147
33	KKB_INST_O_TOT_INST_AMT	0.0047735274	0.0069412335	0.009647372
34	KKB_CC_O_MAX_LIM	0.0043074135	0.0231575813	0.006986028
35	MARRIED_FLAG	0.0040583383	0.0027149748	0.007096917
36	SINGLE_FLAG	0.0036965101	0.0085551064	0.005655356
37	KKB_OD_O_TOT_RISK	0.0036246526	0.0123819122	0.007096917
38	KKB_CC_O_CRE_CNT	0.0036100523	0.0066128320	0.004990020
39	KKB_APP_CNT_L6M	0.0034980209	0.0006251354	0.009647372
40	KKB_OPEN_CC_MAX_PMT_AMT_L3M	0.0026943932	0.0185334086	0.004435573
41	MAX24_GPL_INST	0.0026040059	0.0090194116	0.006542471
42	MAX6_GPL_INST	0.0021861947	0.0096278354	0.004990020
43	KKB_O_HL_FINANCE_AMT	0.0016133060	0.0102561354	0.004546463
44	KKB_O_PL_CNT	0.0015356254	0.0040639546	0.005988024
45	KKB_O_HL_BAL	0.0015277779	0.0016241036	0.001330672
46	WIDOW_DIVORCED_FLAG	0.0015238825	0.0001189754	0.004435573
47	KKB_OPND_CRE_CNT_L6M	0.0014154821	0.0002051994	0.002217787
48	KKB_PL_AL_O_RISK	0.0013303450	0.0013185912	0.002328676
49	KKB_O_HL_CNT	0.0004435939	0.0016692531	0.001774229

Features are classified by Gain. Gain is the improvement in accuracy brought by a feature to the branches it is on. The idea is that before adding a new split on a feature X to the branch there was some wrongly predicted elements, after adding the split on this feature, there are two new branches, and each of these branches is more accurate. Cover column measures the relative quantity of observations concerned by a feature. Lastly, frequency is a simpler way to measure the Gain. It just counts the number of times a feature is used in all generated trees. [39]

4.7.4 XGBoost Feature Importance Graphs

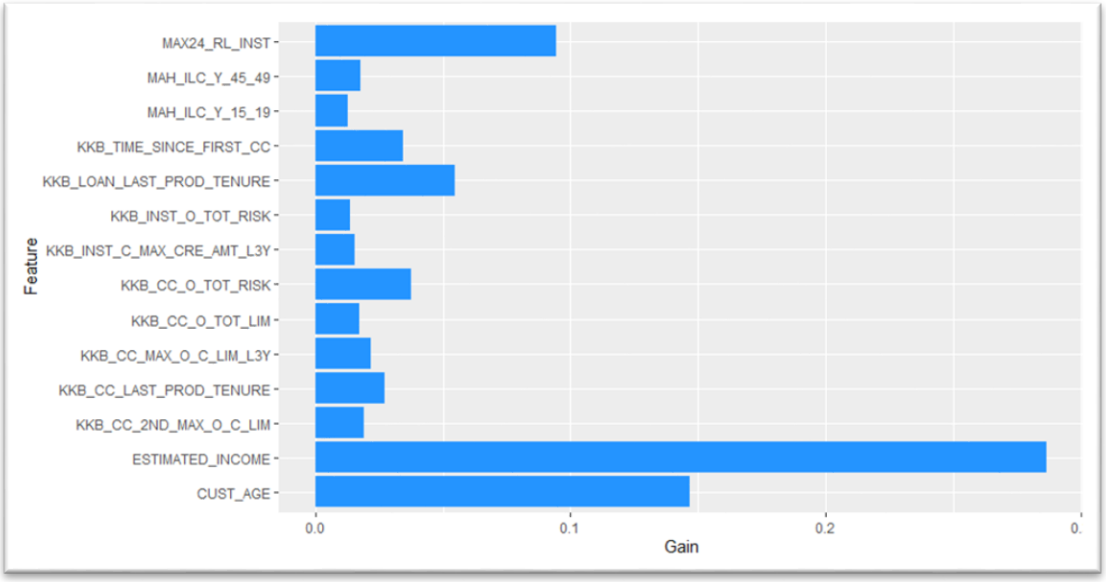


Figure 4-32 XGBoost Feature Gain Relationship Graph

As seen in the above figure, estimated income model variable is the most important one, customer age follows it and MAX24_RL_INST places as third one.

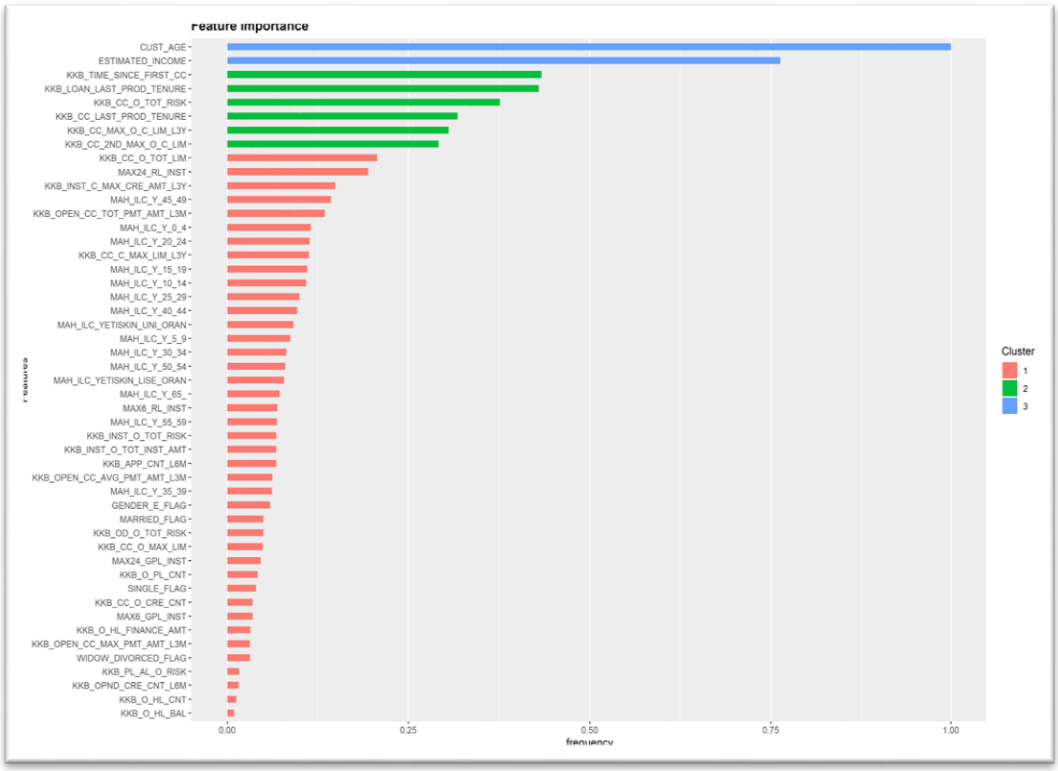


Figure 4-33 XGBoost Frequency Feature Relationship Graph

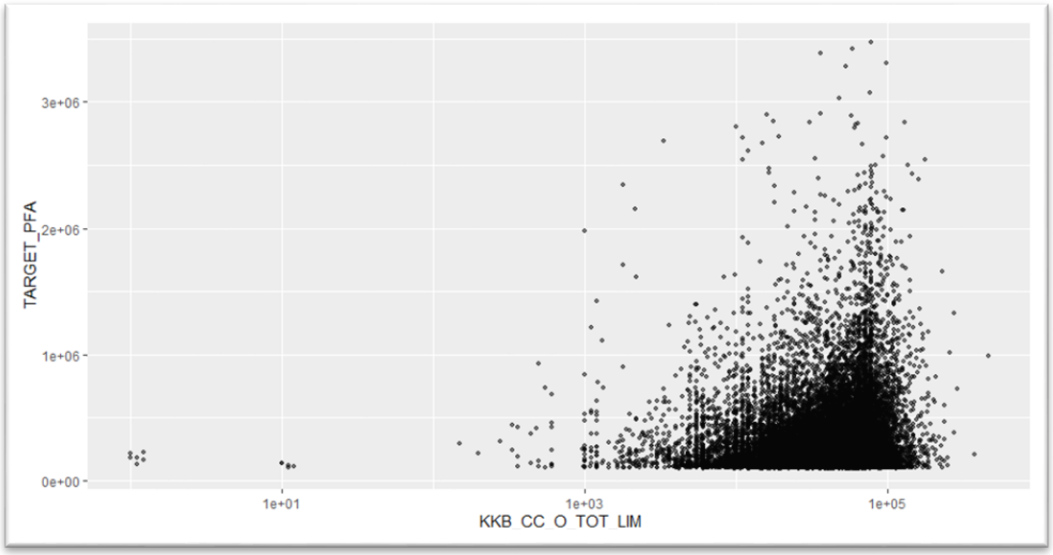


Figure 4-34 Distribution of Target PFA and Total CC Limit in KKB

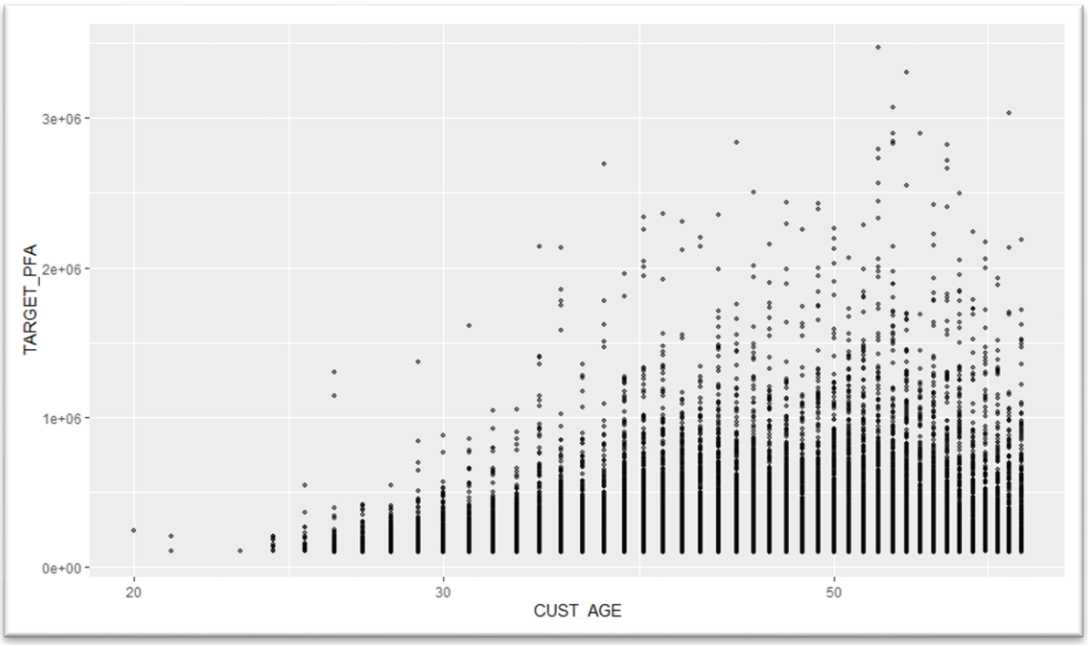


Figure 4-35 Distribution of Target PFA and Customer Age

4.7.4.1 Variable Importance Plotting Functions for GLM, GBM and RF Trained Models

```
h2o.std_coef_plot(regression.model,10)

h2o.varimp_plot(my_gbm,num_of_features = 10)

h2o.varimp_plot(my_rf,num_of_features = 10)
```

Figure 4-36 Codes for Finding Feature Importance in GLM, GBM and RF Models

- ❖ h2o.std_coef_plot (model, num_of_features = NULL): plots a GLM model's standardized coefficient magnitudes.
  - ✓ model: A trained generalized linear model.
  - ✓ num_of_features: The number of features to be shown in the plot.
- ❖ h2o.varimp_plot (model, num_of_features = NULL)
  - ✓ model: A trained model (accepts a trained random forest, GBM, or deep learning model.
  - ✓ num_of_features: The number of features shown in the plot (default is 10 or less than 10).

4.7.4.2 Feature Performance Graphs for All Trained Models

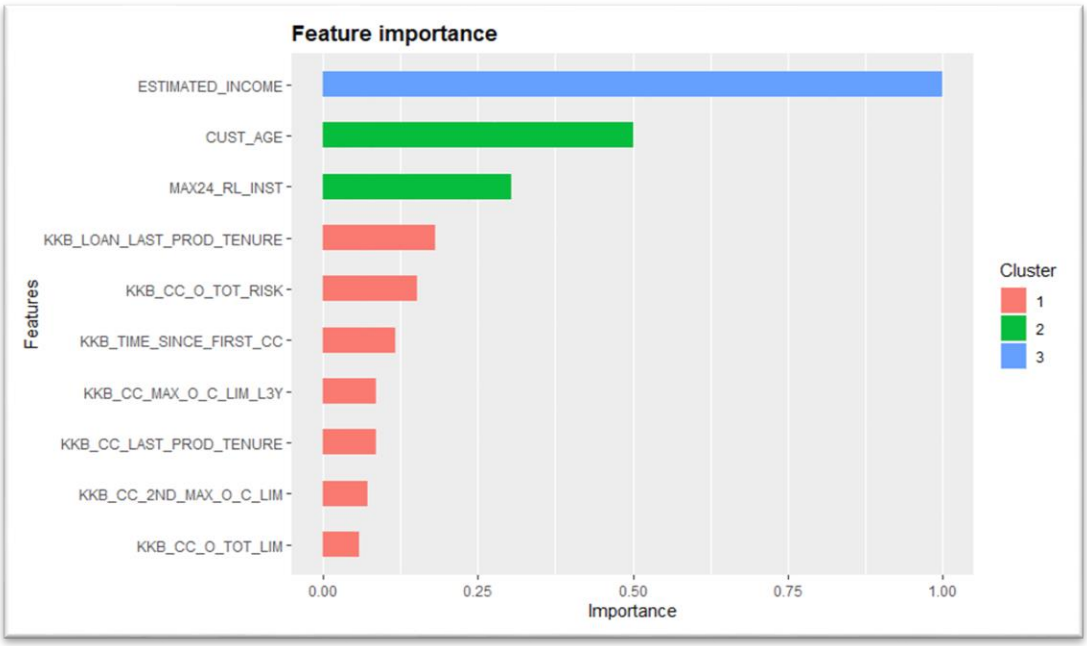


Figure 4-37 Variable Importance Graph of XGBoost

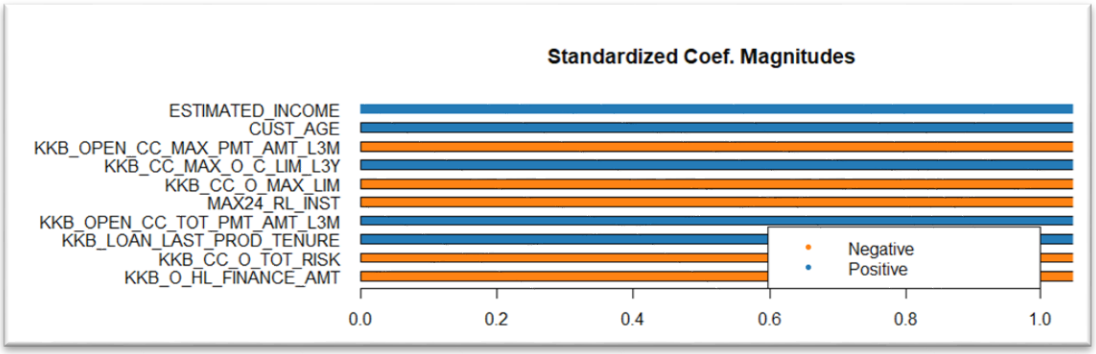


Figure 4-38 Standardized Coeficient Magnitudes Graph

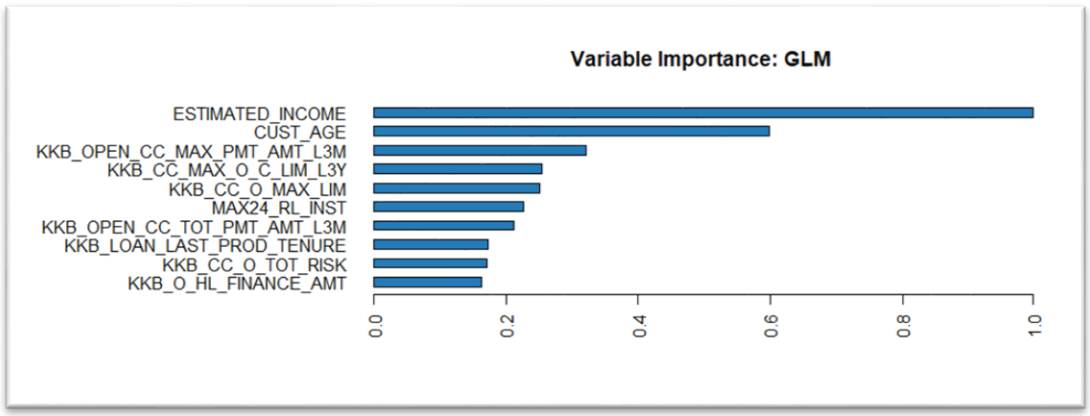


Figure 4-39 Variable Importance Graph of GLM

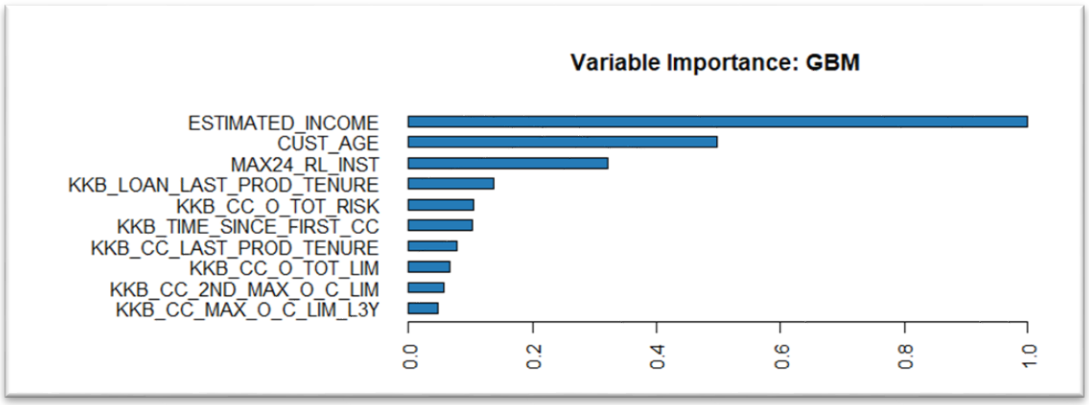


Figure 4-40 Variable Importance Graph of GBM

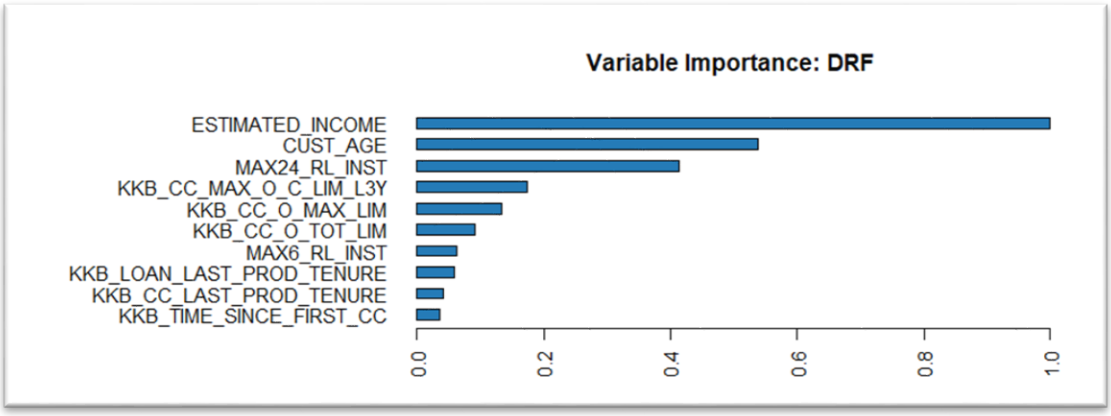


Figure 4-41 Variable Importance Graph of Random Forest

4.7.4.3 Model Performance Interpretation

As seen, trained regression model variable importance graph is more different than XGBoost, random forest and gradient boosting algorithms. In XGBoost, RF and GBM, the first three high importance variables are the same. It means gain and frequency of features are almost the same. After interpretation of all graphs, the most significant variables are estimated income, customer age and the maximum of the customer's monthly loan GPL installment amounts over the last 24 months.

4.7.5 Gathering All Predicted Data Frames

```
DT_PRED_S <- rbindlist(
  use.names = TRUE,
  fill = TRUE,
  l = list(
    data.table(sample = "dev", modelDevSample,PARTY_ID=model_dev_sample_party_id , YEAR_MONTH=model_dev_sample_year_month,
               predicted_ASSET_xgboost
               =pred_vect_dev_xgboost,predicted_ASSET_glm
               =pred_vect_dev_glm$predict,predicted_ASSET_gbm
               =pred_vect_dev_gbm$predict,predicted_ASSET_RF=pred_vect_dev_rf$predict),
    data.table(sample = "test",modelTestSample,PARTY_ID=model_test_sample_party_id ,YEAR_MONTH=model_test_sample_year_month,
               predicted_ASSET_xgboost
               =pred_vect_test_xgboost,predicted_ASSET_glm
               =pred_vect_test_glm$predict,predicted_ASSET_gbm
               =pred_vect_test_gbm$predict,predicted_ASSET_RF
               =pred_vect_test_rf$predict)  )
)

dbWriteTable(jdbcConnection,name=paste("BK_YKB_ASSET_",segment_name,"RESULTS",sep=""),DT_PRED_S)
```

Wiriting the DT_PRED_S table to Oracle Database as BK_YKB_ASSET_EMPLOYED_RETIRED_RESULTS with RJDBC library function.

Figure 4-42 Gathering All Predicted Data Frames

All best models from using machine learning algorithms were chosen after training and testing process and these models predicted the PFA amount as much as possible. In this section, all predicted data frames were merged into one data table to calculate each RMSE value. Afterwards, DT_PRED_S data table will have been used for measurement of model performance statistics. Furthermore, to analyze data table and storage it, was transformed to the Oracle database as BK_YKB_ASSET_EMPLOYED_RETIRED_RESULTS by using of RJDBC library function.

4.7.6 Model Performance Statistic

In this step of application, MAE and MAPE forecast KPIs were used that explained in detail before in Literature Review.

Mean absolute error	$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n}\lvert e_t\rvert$
Mean absolute percentage error	$\mathrm{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n}\left\lvert\frac{e_t}{y_t}\right\rvert$

Figure 4-43 Formulas of MAE and MAPE

4.7.6.1 XGBoost Model Statistics

```
#development data
absErrorDev_TargetvsEstimated_xgb <-  mean(abs(DT_PRED_S[DT_PRED_S$sample == "dev"
                                  &   TARGET_PFA>50000,TARGET_PFA] - DT_PRED_S[DT_PRED_S$sample == "dev"
                                  &   TARGET_PFA>50000,predicted_ASSET_xgboost]))
mapeErrorDev_TargetvsEstimated_xgb <- mean(abs(DT_PRED_S[DT_PRED_S$sample == "dev"
                                  &   TARGET_PFA>50000,TARGET_PFA] - DT_PRED_S[DT_PRED_S$sample == "dev"
                                  &   TARGET_PFA>50000,predicted_ASSET_xgboost])/abs(DT_PRED_S[DT_PRED_S$sample == "dev"
                                  &   TARGET_PFA>50000,TARGET_PFA]))


# test data
absErrorTest_TargetvsEstimated_xgb <- mean(abs(DT_PRED_S[DT_PRED_S$sample == "test"
                                   &  TARGET_PFA>50000,TARGET_PFA] - DT_PRED_S[DT_PRED_S$sample == "test"
                                   &  TARGET_PFA>50000,predicted_ASSET_xgboost]))
mapeErrorTest_TargetvsEstimated_xgb <-mean(abs(DT_PRED_S[DT_PRED_S$sample == "test"
                                   &  TARGET_PFA>50000,TARGET_PFA] - DT_PRED_S[DT_PRED_S$sample == "test"
                                   &  TARGET_PFA>50000,predicted_ASSET_xgboost])/abs(DT_PRED_S[DT_PRED_S$sample == "test"
                                   &  TARGET_PFA>50000,TARGET_PFA]))
```

Figure 4-44 XGBoost Model Statistics Code

TARGET_PFA amount was determined more than 50,000 quantities because the customers that has huge PFA amount is target group of the project. Quite a lot of products can be sold, and the right campaign is made to these target customers. Thanks to this, company wins. For this reason, the error rates (MAE and MAPE) between the real value and the estimated value of the customer groups with high PFA ratio were examined.

```
col0 = c(absErrorDev_TargetvsEstimated_xgb, absErrorTest_TargetvsEstimated_xgb)
col1 = c(mapeErrorDev_TargetvsEstimated_xgb, mapeErrorTest_TargetvsEstimated_xgb)
col2 = c("Train", "Test")
performance_fulllist_xgb = data.frame(col2, col0,col1)
names(performance_fulllist_xgb) = c("Sample", "Target_vs_Estimated_MAE", "Target_vs_Estimated_MAPE")
rm(col0,col1,col2)
```

Figure 4-45 Merging All Error Rate into XGBoost Performance Table Code

In the above codes, all gathered error rate values for both train and test samples were merged into one table.

Table 4-5 XGBoost Model MAE and MAPE Values

TARGET_PFA > 50,000			
performance_fulllist_xgb	Sample	Target_vs_Estimated_MAE	Target_vs_Estimated_MAPE
1	Train	69.895	12%
2	Test	98.642	17%

4.7.6.2 GLM Model Statistics

In the blow codes, target customer PFA amount limited with the same value. For project, the aim is reaching to high PFA customers. So, prediction error is not that much important for all customers. MAE and MAPE values are a bit high for GLM model.

```
#######glm#####
absErrorDev_TargetvsEstimated_glm  <-  mean(abs(DT_PRED_S[DT_PRED_S$sample
                                   &   TARGET_PFA>50000,TARGET_PFA] - DT_PRED_S[DT_PRED_S$sample == "dev"
                                   &   TARGET_PFA>50000,predicted_ASSET_glm]))
mapeErrorDev_TargetvsEstimated_glm <-  mean(abs(DT_PRED_S[DT_PRED_S$sample
                                   &   TARGET_PFA>50000,TARGET_PFA] - DT_PRED_S[DT_PRED_S$sample == "dev"
                                   &   TARGET_PFA>50000,predicted_ASSET_glm])/abs(DT_PRED_S[DT_PRED_S$sample == "dev"
                                   &   TARGET_PFA>50000,TARGET_PFA]))

# test data
absErrorTest_TargetvsEstimated_glm  <-  mean(abs(DT_PRED_S[DT_PRED_S$sample
                                    &  TARGET_PFA>50000,TARGET_PFA] - DT_PRED_S[DT_PRED_S$sample == "test"
                                    &  TARGET_PFA>50000,predicted_ASSET_glm]))
mapeErrorTest_TargetvsEstimated_glm <-  mean(abs(DT_PRED_S[DT_PRED_S$sample
                                    &  TARGET_PFA>50000,TARGET_PFA] - DT_PRED_S[DT_PRED_S$sample == "test"
                                    &  TARGET_PFA>50000,predicted_ASSET_glm])/abs(DT_PRED_S[DT_PRED_S$sample == "test"
                                    &  TARGET_PFA>50000,TARGET_PFA]))



col0 = c(absErrorDev_TargetvsEstimated_glm, absErrorTest_TargetvsEstimated_glm)
col1 = c(mapeErrorDev_TargetvsEstimated_glm, mapeErrorTest_TargetvsEstimated_glm)
col2 = c("Train", "Test")
performance_fulllist_glm = data.frame(col2, col0,col1)
names(performance_fulllist_glm) = c("Sample", "Target_vs_Estimated_MAE", "Target_vs_Estimated_MAPE")
rm(col0,col1,col2)
```

Figure 4-46 GLM Model Statistics Code

In the above codes, all gathered error rate values for both train and test samples were arranged in one table.

Table 4-6 GLM Model MAE and MAPE Values

TARGET_PFA > 50,000			
performance_fulllist_glm	Sample	Target_vs_Estimated_MAE	Target_vs_Estimated_MAPE
1	Train	146.974	56%
2	Test	153.162	58%

4.7.6.3 GBM Model Statistics

In the blow codes, target customer PFA amount limited with the same value. For project, the aim is reaching to high PFA customers. So, prediction error is not that much important for all customers. MAE and MAPE values are almost for GLM model when comparing with the XGBoost.

```
absErrorDev_TargetvsEstimated_gbm  <-  mean(abs(DT_PRED_S[DT_PRED_S$sample
                                   &   TARGET_PFA>50000,TARGET_PFA] - DT_PRED_S[DT_PRED_S$sample == "dev"
                                   &   TARGET_PFA>50000,predicted_ASSET_gbm]))
mapeErrorDev_TargetvsEstimated_gbm <-  mean(abs(DT_PRED_S[DT_PRED_S$sample
                                   &   TARGET_PFA>50000,TARGET_PFA] - DT_PRED_S[DT_PRED_S$sample == "dev"
                                   &   TARGET_PFA>50000,predicted_ASSET_gbm])/abs(DT_PRED_S[DT_PRED_S$sample == "dev"
                                   &   TARGET_PFA>50000,TARGET_PFA]))

# test data
absErrorTest_TargetvsEstimated_gbm  <-  mean(abs(DT_PRED_S[DT_PRED_S$sample
                                   &  TARGET_PFA>50000,TARGET_PFA] - DT_PRED_S[DT_PRED_S$sample == "test"
                                   &  TARGET_PFA>50000,predicted_ASSET_gbm]))
mapeErrorTest_TargetvsEstimated_gbm <-  mean(abs(DT_PRED_S[DT_PRED_S$sample
                                   &  TARGET_PFA>50000,TARGET_PFA] - DT_PRED_S[DT_PRED_S$sample == "test"
                                   &  TARGET_PFA>50000,predicted_ASSET_gbm])/abs(DT_PRED_S[DT_PRED_S$sample == "test"
                                   &  TARGET_PFA>50000,TARGET_PFA]))



col0 = c(absErrorDev_TargetvsEstimated_gbm, absErrorTest_TargetvsEstimated_gbm)
col1 = c(mapeErrorDev_TargetvsEstimated_gbm, mapeErrorTest_TargetvsEstimated_gbm)
col2 = c("Train", "Test")
performance_fulllist_gbm = data.frame(col2, col0,col1)
names(performance_fulllist_gbm) = c("Sample", "Target_vs_Estimated_MAE", "Target_vs_Estimated_MAPE")
rm(col0,col1,col2)
```

Figure 4-47 GBM Model Statistics Code

In the above codes, all gathered error rate values for both train and test samples were arranged in one table.

Table 4-7 GBM Model MAE and MAPE Values

TARGET_PFA > 50,000			
performance_fulllist_gbm	Sample	Target_vs_Estimated_MAE	Target_vs_Estimated_MAPE
1	Train	82.975	15%
2	Test	106.243	18%

4.7.6.4 RF Model Statistics

In the blow codes, target customer PFA amount limited with the same value. For project, the aim is reaching to high PFA customers. So, prediction error is not that much important for all customers. MAE and MAPE values are the highest one for RF model.

```
absErrorDev_TargetvsEstimated_rf  <-  mean(abs(DT_PRED_S[DT_PRED_S$sample
                                  &   TARGET_PFA>50000,TARGET_PFA] - DT_PRED_S[DT_PRED_S$sample == "dev"
                                  &   TARGET_PFA>50000,predicted_ASSET_RF]))
mapeErrorDev_TargetvsEstimated_rf <-  mean(abs(DT_PRED_S[DT_PRED_S$sample
                                  &   TARGET_PFA>50000,TARGET_PFA] - DT_PRED_S[DT_PRED_S$sample == "dev"
                                  &   TARGET_PFA>50000,predicted_ASSET_RF])/abs(DT_PRED_S[DT_PRED_S$sample == "dev"
                                  &   TARGET_PFA>50000,TARGET_PFA]))

# test data
absErrorTest_TargetvsEstimated_rf  <-  mean(abs(DT_PRED_S[DT_PRED_S$sample
                                  &  TARGET_PFA>50000,TARGET_PFA] - DT_PRED_S[DT_PRED_S$sample == "test"
                                  &  TARGET_PFA>50000,predicted_ASSET_RF]))
mapeErrorTest_TargetvsEstimated_rf <-  mean(abs(DT_PRED_S[DT_PRED_S$sample
                                  &  TARGET_PFA>50000,TARGET_PFA] - DT_PRED_S[DT_PRED_S$sample == "test"
                                  &  TARGET_PFA>50000,predicted_ASSET_RF])/abs(DT_PRED_S[DT_PRED_S$sample == "test"
                                  &  TARGET_PFA>50000,TARGET_PFA]))



col0 = c(absErrorDev_TargetvsEstimated_rf, absErrorTest_TargetvsEstimated_rf)
col1 = c(mapeErrorDev_TargetvsEstimated_rf, mapeErrorTest_TargetvsEstimated_rf)
col2 = c("Train", "Test")
performance_fulllist_rf = data.frame(col2, col0,col1)
names(performance_fulllist_rf) = c("Sample", "Target_vs_Estimated_MAE", "Target_vs_Estimated_MAPE")
rm(col0,col1,col2)
```

Figure 4-48 RF Model Statistics Code

In the above codes, all gathered error rate values for both train and test samples were arranged in one table.

Table 4-8 RF Model MAE and MAPE Values

TARGET_PFA > 50,000			
performance_fulllist_rf	Sample	Target_vs_Estimated_MAE	Target_vs_Estimated_MAPE
1	Train	137.478	66%
2	Test	141.874	69%

Creating Model Performance Graphs

All performance graphs were specified with more than 1000 PFA amount limit. The reason is that in the model also there were many customers which has low amount of PFA. The blow codes were run to show the train and test graph of the distribution of the actual value and the estimated value of all algorithms. Therefore, a graph was created to show their distribution.

4.7.6.5 Model Graph Codes

```
ggplot(data = filter(DT_PRED_S,sample == "test"& predicted_ASSET_xgboost>1000 & TARGET_PFA >1000),
       aes(x=TARGET_PFA, y=predicted_ASSET_xgboost ))+
       geom_point(alpha=0.5,size=1)+
       geom_smooth()+
       scale_x_log10()+scale_y_log10()+ #grafiği daraltıyo.
       geom_abline(aes(slope=1,intercept=0),colour='red')
ggsave(paste(segment_name,"Test_xgboost",".png",sep=""),width =5,height=5)

ggplot(data = filter(DT_PRED_S,sample == "dev"& predicted_ASSET_xgboost>1000
       & TARGET_PFA >1000),aes(x=TARGET_PFA, y=predicted_ASSET_xgboost ))+
       geom_point(alpha=0.5,size=1)+
       geom_smooth()+
       scale_x_log10()+scale_y_log10()+
       geom_abline(aes(slope=1,intercept=0),colour='red')
ggsave(paste(segment_name,"Dev_xgboost",".png",sep=""),width =5,height=5)

ggplot(data = filter(DT_PRED_S,sample == "test"& predicted_ASSET_glm>1000 & TARGET_PFA >1000),
       aes(x=TARGET_PFA, y=predicted_ASSET_glm ))+
       geom_point(alpha=0.5,size=1)+
       geom_smooth()+
       scale_x_log10()+scale_y_log10()+
       geom_abline(aes(slope=1,intercept=0),colour='red')
ggsave(paste(segment_name,"Test_glm",".png",sep=""),width =5,height=5)

ggplot(data = filter(DT_PRED_S,sample == "dev"& predicted_ASSET_glm>1000
       & TARGET_PFA >1000),aes(x=TARGET_PFA, y=predicted_ASSET_glm ))+
       geom_point(alpha=0.5,size=1)+
       geom_smooth()+
       scale_x_log10()+scale_y_log10()+
       geom_abline(aes(slope=1,intercept=0),colour='red')
ggsave(paste(segment_name,"Dev_glm",".png",sep=""),width =5,height=5)
```

XGBoost

GLM (Regression Model)

Figure 4-49 Model Graph Codes for XGBoost and GLM

```
ggplot(data = filter(DT_PRED_S,sample == "test"& predicted_ASSET_gbm>1000 & TARGET_PFA >1000),
       aes(x=TARGET_PFA, y=predicted_ASSET_gbm ))+
       geom_point(alpha=0.5,size=1)+
       geom_smooth()+
       scale_x_log10()+scale_y_log10()+
       geom_abline(aes(slope=1,intercept=0),colour='red')
ggsave(paste(segment_name,"Test_gbm",".png",sep=""),width =5,height=5)

ggplot(data = filter(DT_PRED_S,sample == "dev"& predicted_ASSET_gbm>1000 & TARGET_PFA >1000),
       aes(x=TARGET_PFA, y=predicted_ASSET_gbm ))+
       geom_point(alpha=0.5,size=1)+
       geom_smooth()+
       scale_x_log10()+scale_y_log10()+
       geom_abline(aes(slope=1,intercept=0),colour='red')
ggsave(paste(segment_name,"Dev_gbm",".png",sep=""),width =5,height=5)

ggplot(data = filter(DT_PRED_S,sample == "test"& predicted_ASSET_RF>1000 & TARGET_PFA >1000),
       aes(x=TARGET_PFA, y=predicted_ASSET_RF ))+
       geom_point(alpha=0.5,size=1)+
       geom_smooth()+
       scale_x_log10()+
       scale_y_log10()+
       geom_abline(aes(slope=1,intercept=0),colour='red')
ggsave(paste(segment_name,"Test_rf",".png",sep=""),width =5,height=5)

ggplot(data = filter(DT_PRED_S,sample == "dev"& predicted_ASSET_RF>1000 & TARGET_PFA >1000),
       aes(x=TARGET_PFA, y=predicted_ASSET_RF ))+
       geom_point(alpha=0.5,size=1)+
       geom_smooth()+
       scale_x_log10()+
       scale_y_log10()+
       geom_abline(aes(slope=1,intercept=0),colour='red')
ggsave(paste(segment_name,"Dev_rf",".png",sep=""),width =5,height=5)
```

GBM

RF

Figure 4-50 Model Graph Codes for GBM and RF

4.7.6.6 Model Performance Graphs

4.7.6.6.1 XGBoost Train and Test Graphs

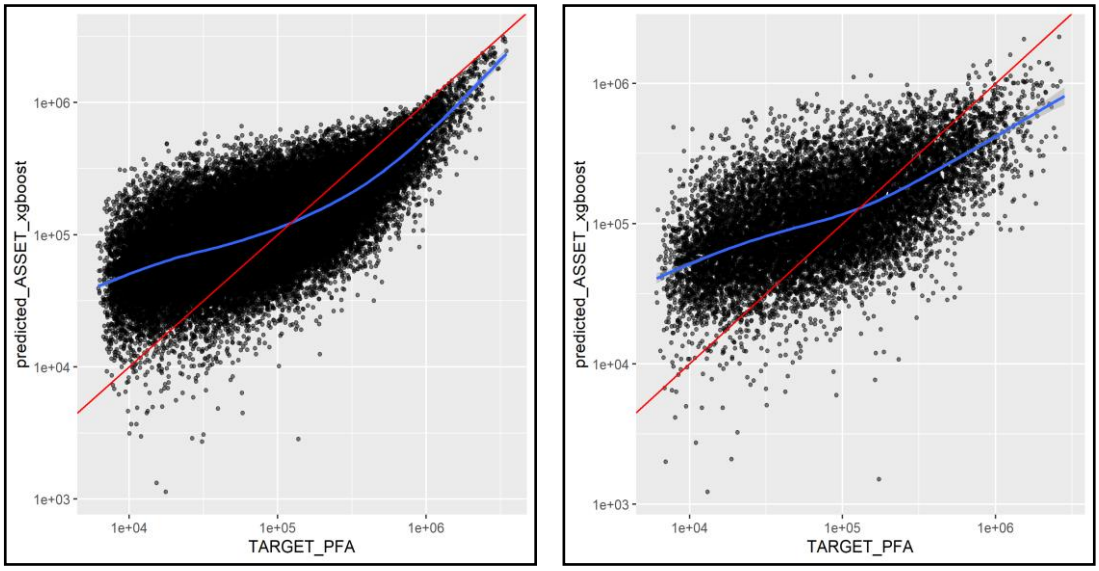


Figure 4-51 XGBoost Train and Test Sample Performance Graphs

In these graphs, the relationship between target customers' PFA and predicted XGBoost model will be encompassed. As one may remember, predicted_ASSET_xgboost was created inside of the DT_PRED_S after model prediction and during the creation, as all algos XGBoost was divided into dev and test samples. Therefore, the left graph shows the developed (it can be said trained also) model and the right one shows the test one. Also, the blue line expresses the model and red one points regression line which is the distribution of the real population values. As seen, there is an overfitting situation in dev sample. In the beginning, model real and estimated values are a bit far from each other. At one point that they start to go together which means overfitting. The model starts to memorize all parameters lead to that PFA amount. On the other hand, in test sample

there is no overfitting, of course it makes sense. Because in train sample, label values are shown to the machine. It leads to overfitting after learning a lot. Test sample performance is good enough. As a result, there is less deviation when the PFA amount increases as seen in the model and regression curve. This means that the model is sufficient to estimate the wealthy segment. The error resulted in a slightly greater value. But it is impossible to know how much money a person actually has.

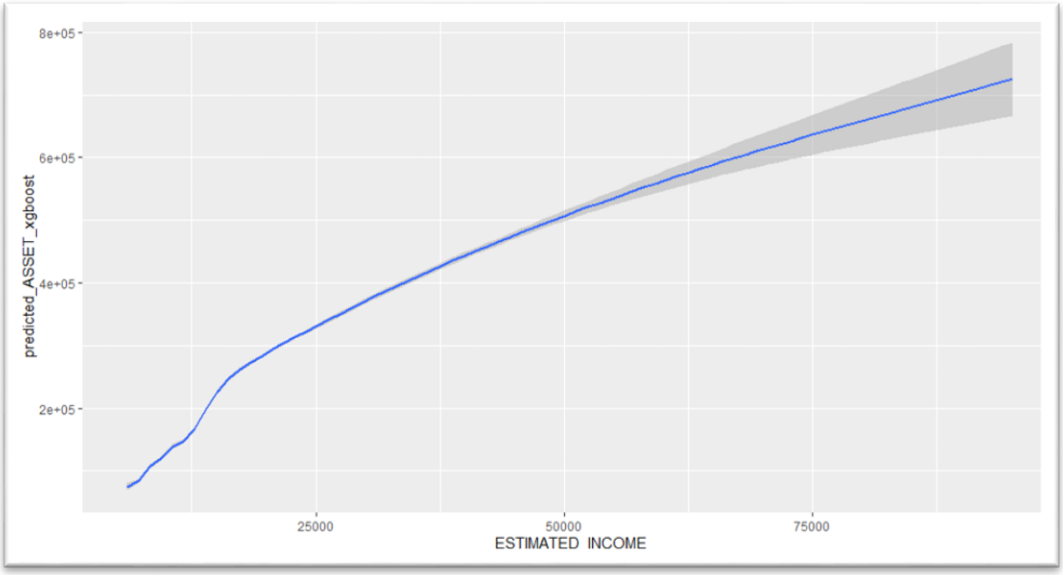


Figure 4-52 Estimated Income with Predicted XGB

4.7.6.6.2 Linear Regression Train and Test Graphs

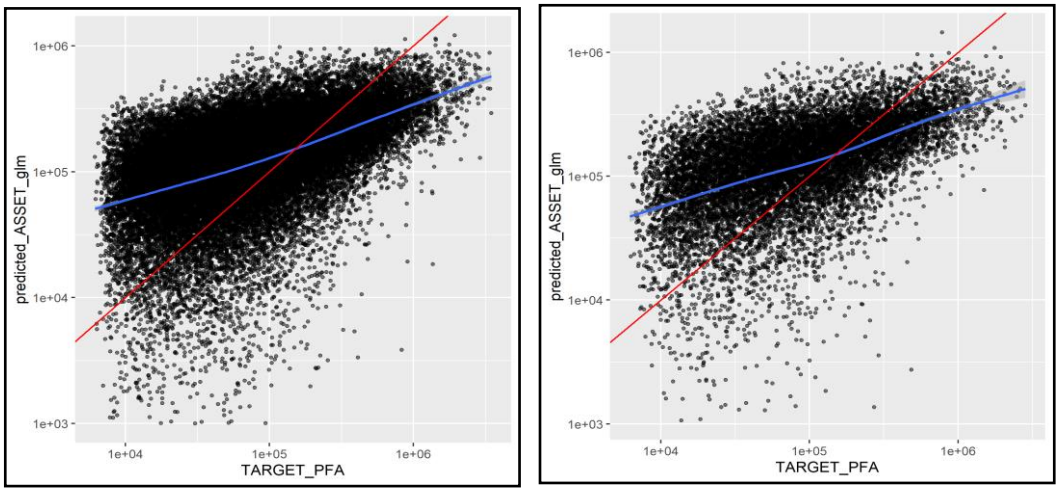


Figure 4-53 GLM Train and Test Sample Performance Graphs

In these graphs, both the red regression line is a template to show the real values and the blue one is the model line to show how the GLM model predicted. As seen from the deviations, the model is not sufficient. Model estimated that many customers with low PFA amount, as wealthy. For example, at the beginning normally customer's PFA amount is 1,000 TRY, however model estimates 15,000. It is so bad for aim of the project. Because the purpose of the project is to reach wealthy customers. Left graph represents

the dev and right one represents the test sample. First of all, dev model is underfitting. And also has high bias. Apart from variance, bias is way better. However, high bias and high variance should be avoided for good model performance. When performance of test sample is taken into consideration, it is better as prediction. Because model estimation at the beginning is better than dev sample. The population density is closer to the regression line. After fitting point for two graphs, predictions are almost the same. However, when GLM model performance graphs compare with the XGBoost model that explained before, GLM model is not an appropriate model that we look for. Estimation in high PFA band is worse than XGBoost model.

4.7.6.6.3 Gradient Boosting Train and Test Graphs

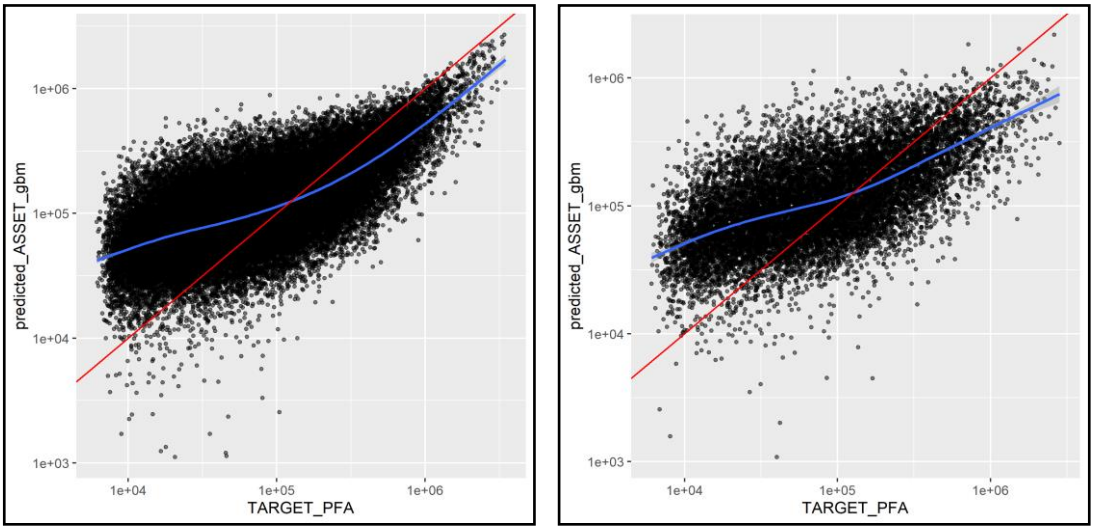


Figure 4-54 GBM Train and Test Sample Performance Graphs

XGBoost is an implementation of gradient boosted decision trees. Therefore, GBM is so similar to the XGBoost model. It can be understood the similarity of two model performance graphs. However, there is a bit different than XGBoost is better. When two model graphs are examined in detail, the prediction is closer to the regression line in some points. All interpretation that is written in the XGBoost model performance graphs header is valid for GBM too. Because they are very similar.

4.7.6.6.4 Random Forest Train and Test Graphs

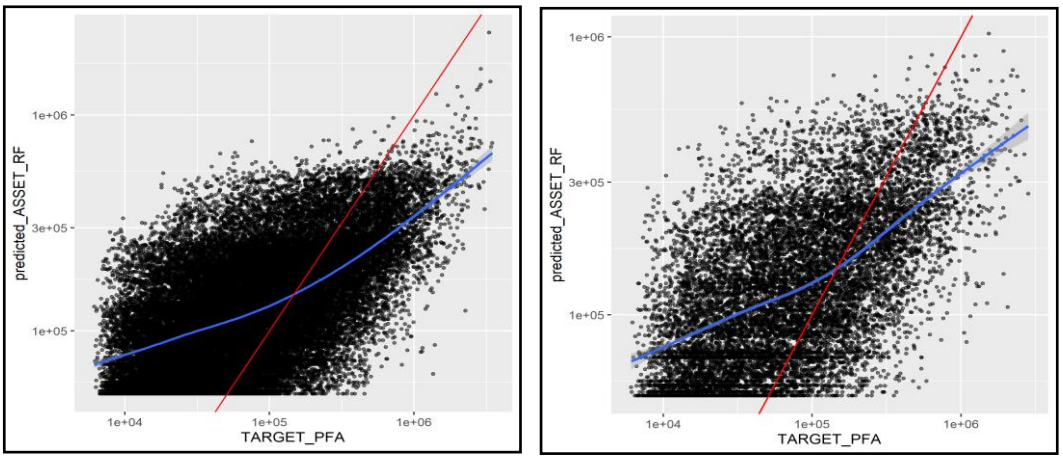


Figure 4-55 RF Train and Test Sample Performance Graphs

In both figures, the regression curve appears somewhat skewed. This is because the estimates are too much at the low PFA ratio. The training graph estimation performed poorly. The model could not estimate the asset of the customer with high PFA properly. It estimated with a very large deviation. The test sample is better in the high PFA band. Unfortunately, the amount of deviation in the test is quite high for wealthy customers. It was the worst-performing RF model among all models. It did not meet the aim of the project.

4.8 Scoring Performance

```
SQL_Code_BASE = paste("select * from BK_YKB_ASSET_MODEL_TABLE3 where SEGMENT='",segment_name,"' and TARGET_FLAG2=0 ",sep="")
DT_BASE<-dbGetQuery(jdbcConnection, SQL_Code_BASE)

DT_U<-as.data.table(DT_BASE)

for(colName in names(DT_U)) {
  if (is.numeric(DT_U[[colName]]) == TRUE) {
    DT_U[is.na(get(colName)),   (colName) := 0]
    DT_U[get(colName) == -9999, (colName) := 0]
  }
}
for(colName in names(DT_U)) {
  if(is.factor(DT_U[[colName]])) {
    DT_U[i = is.na(get(colName)), j = (colName) := "Missing"]
  }
}

DT_U_party_id <- DT_U$PARTY_ID
DT_U_year_month <- DT_U$YEAR_MONTH
DT_U[, c( vars_to_remove) := NULL]


model_formula <- as.formula(TARGET_PFA ~ . - 1)
sDT <- sparse.model.matrix(model_formula, data = DT_U)
lDT <- DT_U$TARGET_PFA
dDT <- xgb.DMatrix(data = sDT,  label = lDT)

pred_unknown <-predict(object = mod_xgb_dart_best, newdata = dDT)
```

Connection with Oracle Database

Data cleaning & Feature Engineering & Missing Values

Removing High Correlated Features

Sparse model matrix, label and dense matrix

Prediction with the best model

Figure 4-56 Scoring Performance Code

This part is the scoring part. Only this part will be used after the model goes live. The model that has learned with customers that TARGET_FLAG2 equals to 1, can be scored with customers that TARGET_FLAG2 equals to 0. These non-target segment' asset is wanted to learn. Thanks to the scoring, customers with unknown asset amount can be predicted. After model goes live, model should be tested whether variables are good for prediction or not.

5 CONCLUSION and SUGGESTIONS

In this thesis, four machine learning algorithms that were examined in the literature review header used to predict the asset of bank's customers according to many conditions created in Oracle SQL software. Model performance was measured with MAE and MAPE KPI forecast error types. After examination of MAE and MAPE values for test samples and also model performance graphs, the XGBoost algorithm was chosen as an optimal model for this case. Test MAE and MAPE values were found 98.642, 17% respectively. As a limited value for target PFA amount of customers was obtained 50,000 for each model. The reason for choosing a constraint was to examine the wealthy segment. On the other hand, all model performance was run via codes, however, the result of error types was higher. The significant reason was KKB data that supplies from another company for every bank was not clear. If KKB data were clearer, the optimal model would have better accuracy. Additionally, the bank data warehouse was very complex. To determine the right table and in that way to build variable tables took lots of time. Maybe for the future project, the data warehouse could rearrange. Another suggestion of the thesis, if there were enough time, other machine learning algorithms and hyper params, mentioned in the literature review, could be used. Because all algos and methodologies are a strong effect on finding the optimal one. Therefore, the best method could not be known without trying.

As a result, not all models but for wealthy customers, the XGBoost model predicted well. GBM model followed it with almost no difference. In other words, the aim of the project could be reached.

6 REFERENCES

[1] Predictive Modeling. microstrategy.com. Retrieved November 20, 2019, from https://www.microstrategy.com/us/resources/introductory-guides/predictive-modeling-the-only-guide-you-need

[2] Predictive Analytics in Finance. emerj.com. Retrieved October 15, 2019, from https://emerj.com/ai-sector-overviews/predictive-analytics-in-finance/

[3] Generalized linear model. wikipedia.com. Retrieved November 25, 2019, from https://www.wikizeroo.org/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvR2VuZXJhbGl6ZWRfbGluZWFyX21vZGVs

[4] Chen, L. (2019, January 2) Basic Ensemble Learning. towardsdatascience.com. Retrieved December 2, 2019, from https://link.medium.com/K3x7L5R8Z2

[5] Morde, V. (2019, April 8). XGBoost Algorithm. towardsdatascience.com. Retrieved December 2, 2019, from https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

[6] Parameter Tuning in XGBoost. analyticsvidhya.com. Retrieved November 8, 2019, from https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

[7] Vandeput, N. RMSE, MAE, MAPE & Bias, medium.com. Retrieved January 2, 2020, from https://medium.com/analytics-vidhya/forecast-kpi-rmse-mae-mape-bias-cdc5703d242d

[8] Restrepo, M. Doing XGBoost hyper-parameter tuning the smart way, towardsdatascience.com. Retrieved November 8, 2019, from https://towardsdatascience.com/doing-xgboost-hyper-parameter-tuning-the-smart-way-part-1-of-2-f6d255a45dde

[9] About us page. kkb.com. Retrieved September 26, 2019, from www.kkb.com.tr

[10] Yilmaz Velioglu, Memzuc, muhasebetr.com, Retrieved September 30, 2019, from http://www.muhasebetr.com/yazarlarimiz/yilmazvelioglu/011/

[11] Prabhakaran, S. (2019, November 13). data.table in R. machinelearningplus.com. Retrieved November 4, 2019, from https://www.machinelearningplus.com/data-manipulation/datatable-in-r-complete-guide/

[12] data.table package from CRAN

[13] xgboost package from CRAN

[14] RJDBC package from CRAN

[15] sqldf package from CRAN

[16] dplyr package from CRAN

[17] openxlsx package from CRAN

[18] ggplot2 package from CRAN

[19] Executing a code with R connecting Impala via JDBC (RJDBC). ibm.com. Retrieved November 9, 2019, from https://www.ibm.com/support/pages/executing-code-r-connecting-impala-jdbc-

rjdbc-results-error-jcallrp-i-fetch-stride-javalangoutofmemoryerror-java-heap-space-rjdbcdbgetquery-gc-overhead-limit-exceeded
[20] Zhang T. (2015, February 16). Connect to Oracle Database in R with RJDBC. linkedin.com. Retrieved November 6, 2019, from https://www.linkedin.com/pulse/connect-oracle-database-r-rjdbc-tianwei-zhang/
[21] Leo Breiman, "Bagging Predictors", Machine Learning 24, no. 2 (1996): 123-140.
[22] Leo Breiman, "Pasting Small Votes for Classification in Large Databases and On-line", Machine Learning 36, no. 1-2 (1999): 85-103.
[23] Aurelien Geron, "Hands-on Machine Learning with Scikit-learn, Keras & TensorFlow", (2019), Second Edition
[24] Nagpal, A. (2017, October 13). L1 and L2 Regularization Methods. towardsdatascience.com. Retrieved December 5, 2019, from https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c
[25] Chekka, V. (2018, August 30). Regularization in Machine Learning: Connect the dots. towardsdatascience.com. Retrieved December 5, 2019, from https://towardsdatascience.com/regularization-in-machine-learning-connecting-the-dots-c6e030bfaddd
[26] Gandhi, R. (2018, May 5). Boosting Algorithms: AdaBoost, Gradient Boosting and XGBoost. hackernoon.com. Retrieved November 4, 2019, from https://hackernoon.com/boosting-algorithms-adaboost-gradient-boosting-and-xgboost-f74991cad38c
[27] Walsh, S.R. (2017, July 10). Dplyr Functions. r-bloggers.com. Retrieved November 18, 2019, from https://www.r-bloggers.com/useful-dplyr-functions-wexamples/
[28] Correlation. surveysystem.com. Retrieved November 13, 2019, from https://www.surveysystem.com/correlation.htm
[29] Correlogram/Auto Correlation Function ACF Plot. statisticshoto.datasciencecentral.com. Retrieved November 13, 2019, from https://www.statisticshowto.datasciencecentral.com/correlogram/
[30] Training, validation, and test sets. wikipedia.org. Retrieved November 9, 2019, from https://www.wikizeroo.org/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvVHJhaW5pbmcsX3ZhbGlkYXRpb24sX2FuZF90ZXN0X3NldHMjY2l0ZV9ub3RlLWNhbm4tZmFxLTg
[31] Srivastava, T. (2016, January 22). How to use XGBoost algorithm in R in easy steps. analyticsvidhya.com Retrieved November 2, 2019, from https://www.analyticsvidhya.com/blog/2016/01/xgboost-algorithm-easy-steps/
[32] Brownlee, J. (2017, July 28). Why One-Hot Encode Data in Machine Learning. machinelearningmastery.com. Retrieved November 24, 2019, from https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/
[33] DelSole, M. (2018, April 25). What is One Hot Encoding and How to Do It. medium.com. Retrieved November 24, 2019, from https://medium.com/@michaeldelsole/what-is-one-hot-encoding-and-how-to-do-it-f0ae272f1179
[34] Vasudev. (2019, October 25) What is One Hot Encoding? Why and When do you have to use it? hackernoon.com Retrieved November 24, 2019, from https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f

[35] Vasudev. (2019, October 25) What is One Hot Encoding? Why and When do you have to use it? hackernoon.com Retrieved November 24, 2019, from https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f
[36] XGBoost Parameters. xgboost.readthedocs.io. Retrieved November 19, 2019, from https://xgboost.readthedocs.io/en/latest/parameter.html
[37] h2o package from CRAN
[38] matrix package from CRAN
[39] Understand your dataset with XGBoost. xgboost.readthedocs.io. Retrieved November 19, 2019, from https://xgboost.readthedocs.io/en/latest/R-package/discoverYourData.html