MEF UNIVERSITY

CREDIT CARD CHURN PREDICTION WITH MACHINE LEARNING ALGORITHMS

Capstone Project

Serap Konuksal

İSTANBUL, 2018

MEF UNIVERSITY

CREDIT CARD CHURN PREDICTION WITH MACHINE LEARNING ALGORITHMS

Capstone Project

Serap Konuksal

Advisor: Prof. Semra Ağralı

İSTANBUL, 2018

MEF UNIVERSITY

Name of the project: Credit Card Churn Prediction With Machine Learning Algorithms

Name/Last Name of the Student: Serap Konuksal

Date of Thesis Defense: 16/08/2018

I hereby state that the graduation project prepared by Serap Konuksal has been completed under my supervision. I accept this work as a "Graduation Project".

16/08/2018

Prof. Semra Ağralı

I hereby state that I have examined this graduation project by Serap Konuksal which is accepted by her supervisor. This work is acceptable as a graduation project and the student is eligible to take the graduation project examination.

16/08/2018

Prof. Özgür Özlük
Director
of
Big Data Analytics Program

We hereby state that we have held the graduation examination of _____ and agree that the student has satisfied all requirements.

THE EXAMINATION COMMITTEE

Committee Member	Signature
1. Prof. Semra Ağralı	……..
2. ……..	……..

Academic Honesty Pledge

I promise not to collaborate with anyone, not to seek or accept any outside help, and not to give any help to others.

I understand that all resources in print or on the web must be explicitly cited.

In keeping with MEF University's ideals, I pledge that this work is my own and that I have neither given nor received inappropriate assistance in preparing it.

Name Date Signature

EXECUTIVE SUMMARY

CREDIT CARD CHURN PREDICTION WITH MACHINE LEARNING ALGORITHMS

Serap Konuksal

Advisor: Semra Ağralı

AUGUST, 2018, 40 pages

Credit card is one of the main products in banking sector and there is a big competition in credit card business. This competition makes retention of customers critical. To retain the customers, it is very important to interpret the customers that may churn. Targeting right customers with right offer is the main aim of Customer Relationship Management (CRM) in marketing. When the churn probability of customers is predicted, it is easier to retain the customers by proposing the retention offers directly to the ones with high churn probability. This will allow banks to manage their marketing budgets efficiently. In this project, a private bank's credit card customer data is used. Data includes many different types of features of customers, such as number and type of transactions, credit card limits, feature usage, credit bureau information and demographic information. We develop a set of churn prediction models by implementing different machine learning algorithms. We compare these algorithms to find the best model with highest accuracy to be offered to the bank. We also share the main indicators that affect churn so that the bank can use them in retention activities.

Key Words: Churn, credit card, CRM

ÖZET

MAKİNE ÖĞRENME ALGORİTMALARI İLE KREDİ KARTI MÜŞTERİ KAYBI TAHMİNİ

Serap Konuksal

Tez Danışmanı: Semra Ağralı

AĞUSTOS, 2018, 40 sayfa

Bankacılık sektöründe kredi kartı en temel ürünlerden biridir ve bankalar arasında rekabet yüksektir. Bu kadar rekabetin olduğu bir ortamda müşterilerin tutundurması kritik bir hal alıyor. Müşterileri tutundurabilmek için hangi müşterilerde kayıp yaşanacağının önden bilinmesi önemlidir. Pazarlamada müşteri ilişki yönetiminde en temel amaç doğru müşteriye doğru teklif yapılmasıdır. Müşteri kayıp ihtimali önden bilindiği taktirde, bu müşterilere ikna teklifleri sunularak gitmeleri engellenebilir. Hedef kitlenin doğru belirlenmesi pazarlama bütçelerinin etkili yönetilmesini sağlayacaktır, yanlış müşteriler hedeflenerek oluşacak ek maliyetler engellenmiş olur. Bu projede özel bir bankanın kredi kartı müşteri datası kullanılmıştır. Datada işlem bilgileri, ürün limit bilgileri, ürün kullanımları, Kredi Kayıt Bürosu bilgileri ve demografik bilgiler bulunmaktadır. Farklı makina öğrenme algoritmaları kullanılarak müşteri gitme olasılığı tahmin edilmeye çalışılmış ve modeller birbirleri ile karşılaştırılmıştır. En iyi tahmin eden algoritma seçilerek banka ile paylaşılacaktır. Ayrıca müşteri kaybını etkileyen önemli değişkenler tespit edilerek, tutundurma faaliyetlerinde kullanılması için banka ile paylaşılacaktır.

Anahtar Kelimeler: Kredi kartı, Müşteri İlişki Yönetimi, Müşteri Kaybı

TABLE OF CONTENTS

Academic Honesty Pledge vi
EXECUTIVE SUMMARY vii
ÖZET viii
TABLE OF CONTENTS ix
1. INTRODUCTION 1
2. LITERATURE REVIEW 3
3. ABOUT THE DATA 4
4. PROJECT DEFINITION 6
    4.1. Problem Statement 6
    4.2. Project Objective 6
    4.3. Project Scope 6
5. METHODOLOGY 7
    5.1. Decision Tree 7
    5.2. Logistic Regression 7
    5.3. Random Forest 8
    5.4. Exploratory Data Analysis 8
6. RESULTS 11
7. CONCLUSION 16
APPENDIX A 17
REFERENCES 39

1. INTRODUCTION

Customer Relationship Management (CRM) is the most important strategic marketing action for companies. If a company does not know the shopping habits of their customers, then their marketing activities will not serve the company. The main objective of the marketing is to understand customers and their needs, and then provide relevant solutions to customers. CRM helps companies to learn about customers' needs and behaviors, so that they can set a proper relationship with their customers. CRM is an analytical process. With data mining methods, it is easier to learn about customers. Farquad et al.(2014) state that CRM is a method that is used for understanding the customer behavior and building strong relationship with them.

Data gives us lots of information about customers, such as the customer's demographic information, things that the customers like, the locations that the customers live, etc. Collecting the data and extracting information from data are very important for CRM activities. Bolton et al. (2000) state that understanding the purchase behavior of customers and then targeting them with this insight, helps organizations to develop loyalty programs and strong relationship with customers.

Mainly banking and telecommunication sectors are the leading sectors where CRM is successfully applied. They have huge amount of data, which is called big data in recent years. By applying different data analytics methods on this big data, they can use the power of CRM. These companies can prepare strategies for each point of life cycle of the customers such as activation, usage increase, upsell, cross sell or churn.

In these customer life cycle points, churn is the most important one since acquisition of a new customer is much expensive than retaining the customer. Colgate and Danaher(2000) state that acquiring of a new customer is 5 more times expensive than retaining an existing customer. This makes retention activities more important. In banking sector, all banks have similar products and services and they all try to survive at a highly competitive environment. In order to survive in this environment and become a leading company, they need powerful tools to identify the customers that may churn and take effective retention actions. Marketing budgets are limited; and therefore, finding the right customer and making anti churn offers to right customers improve the efficiency and help banks to increase their profitability. Gordini and Veglio(2016) show that customer churn

prediction helps marketing decision, and this represents hundreds of thousands of euro in B2B industry. Since predicting churn is a difficult problem, companies should use prediction models to find these customers. There exist many methods for prediction in literature; however, it is important to find the best method that suits the company.

Remaining of this project report is organized as follows. Section 2 provides the literature review. Credit card data and churn data set are explained in Section 3. Credit card churn problem is defined in Section 4. In section 5, the methodology used in building churn prediction is explained. All results are provided in Section 6. Section 7 concludes the project.

2. LITERATURE REVIEW

There are many definitions of churn for different sectors. Some companies define churn as not logging into a web page or stop using a product or ending the contract between the customer and the company. Glady et al. (2009) define a churned customer as the one whose Customer Lifetime Value (CLV) is decreased over a period. CLV or income margin of customers show the value of each customer for companies. Some companies focus on high value customers and take action only for them. In this case, the churn probability of high value customers becomes more important. Bolton (1998) states that companies in service sector should be proactive to understand the customer needs before they churn.

Churn can be predicted from the customer's past behaviors and transactions. Coussement et al. (2016) state that the customer churn prediction assigns a probability to each customer by using the signs that shows churn from the customer's historical behaviors. Churned customers and loyal customers differ from each other since some of them decide to end the relationship with the company and others stay as being customers. There should be patterns that affect churn. Larivière and Poel (2004) identify churn periods in two ciritical parts; the first one covers the early years after acquisition, and the second one covers the period that includes 20 years and more after being customer. In addition, Hadden et al.(2008) show that the complaint types and the number of complaints are the most significant variables that affect churn. Bolton (1998) states that tenure that shows the time period of being a customer and customer experience affect churn probability. Moreover, Ballings and Poel (2012) show that in churn prediction, reducing the length of period of being customer can have a decreasing effect on predictive performance.

Defining all parameters that may affect churn and predicting the churn potential help companies to build right offers in CRM strategies; and hence, they can avoid attrition of customers and build strong relationship with their customers. Burez and Poel (2007) state that when customers are ranked according to the probability of ending the relationship with the company, companies can offer campaigns to customers who have high propensity of churn; and hence, double the profits of retention campaigns.

3. ABOUT THE DATA

Credit card is the most popular product of banks, and there is a high competition in this area. Credit cards are highly regulated products by the government in Turkey. All interest rates, most fees and the number of installments of each purchasing sectors are capped by the government . Moreover, credit card limits are capped with sector limits due to the income of customers. In this highly regulated environment, keeping the profitabilty of credit card is very difficult. Therefore, the credit card churn is one of the main problems of banks that needs to be considered urgently.

Lower churn means keeping more existing customers.In this project, a private bank's credit card data is used.The data includes the customers who have at least one active credit card in April 2018. The customers who churned voluntarily during previous2 months period are flagged. Churn is defined as customer that does not want to use the credit card and deactivates all credit cards him/herself. There is a decision made by the customer. Bank should know this decision and try to change it, and convince the customer for not closing the credit card.

In the data set, for all customers we have purchasing behaviors, payments, limits, tenure information, demographic information, credit bureau information, feature usages and churn flag as churned/not churned. We give the variable groups present in the data set in Table 1. The bank has millions of credit card customers, so churn ratio is high due to this portfolio. There are some actions that prevent churn, but these actions are not included in the data set. Although the retention actions are applied, there are still many churned customers.

Table 1. Variable groups in data

Variable Groups	Variable Definitions
Transactions	Monthly transaction amounts Sector Based Transactions in last 3 or 6 months Min, max, mean amount of transactions in last 12 months
Demographic Information	Gender Age Marital Status
Credit Bureau Information	Product based limits Product based balances

	Product based tenures
Credit Card Limit	Credit card limit Days passed since limit increase date Days passed since limit decrease date Limit Increase Channel
Feature Usage	Activeness Credit card feature usage
Annual Fee	Exemptions Next annual fee date Last annual fee charged/cancelled
Churn Flag	Churned/Live flag

The dataset comprises of 272 variables, with 276 predictor variables and 1 class variable. It includes182,934 customers of which 136,232 customers are loyal customers and 46,702 customers represent churned customers. Thus,there are 74% loyal customers and 26 % churned customers.

4. PROJECT DEFINITION

4.1. Problem Statement

In banking sector, credit card is the main product for retaining customers. So detecting credit card customers with high churn probability is important to offer appropriate solutions to the right customers. CRM helps us to take actions to right customer with right offer. Before customers churn, banks should know the churn probability and take action immediately. In this project, we aim to propose methods that will determine the credit card customers who will churn voluntarily.

4.2. Project Objectives

For the project we have a sample of credit card data that contains anonymously credit card usage, transaction counts and sums, payments, limits, demographic information for open credit card customers and flags that show which customers still have active credit card and which customers deactivated their credit card in the previous two months period. By using this data, we predict the customers who may churn during next 2 months. At the end of this project, for the high churn potential customers, there will be a time for taking anti-churn actions in these 2 months period.

4.3. Project Scope

In the project, since business customers have different payment and usage behaviors than individuals, business credit cards are excluded from the data set. Also involuntarily churn, which are the closures by bank, are also excluded from the project. Banks close the credit cards due to delinquency, risk issues, operational or delivery problems. These closures are out of scope of this project. In this project, we focus on the voluntary churn of individual customers, which are the closures made by customers. The customer behavior and information will be used to predict the voluntary churn.

5. METHODOLOGY

Banks have a huge amount of data, which can be called as big data. All kind of data is held in the data warehouse (DWH). From DWH environment, the data is prepared with using SQL. SQL is very useful for data preparation. After data is prepared, randomly 181,548 customers are selected as sample and exported to a csv file. For building machine learning algorithms, Python programming language is used. Python helps to process big sized data, and it is easier to build and develop prediction models using Python.

In churn prediction many methods can be used in the project. Decision Tree (DT), Logistic regression (LR) and Random Forest (RF) algorithms will be applied and then their results, accuracy and other parameters will be compared. All algorithms will use the same data set. With cross validation, over fit is avoided.

In the data set, churned customers are flagged. This flag is the target variable, so supervised learning methods will be used. For tenure and customer age, grouping can help to classify the customers. The target variable has two values; 0 shows live customers, 1 shows churned customers. From 0 to 1, any predicted value shows the probability of churn. Data has numeric and categorical variables. For categorical variables, they are turned into binary variables. Dates are turned into days between the date variable and current date. So that DT and LR models can have more prediction power.

5.1. Decision Tree

Decision Tree (DT) algorithms are tree shaped diagrams that show statistically classification of the data. These algorithms are more understandable than other methods. In each leaf of a DT, there are some simple rules so that the model can be easily explained to the management. This model helps us to draw the conclusion diagram, which is complex. In the model, the number of leaves can be limited or the observation number can be fixed to a specific number. With these arrangements, the accuracy of the model can change in a good/bad way.

5.2. Logistic Regression

Logistic regression (LR) is a regression method when the target variable is binary. The target variable has only two values, if churned, then 1; else, 0. The model aims to find

out the probability of being 1 or 0. The model predicts the maximum likelihood between independent and dependent (target) variables. It uses linear regression like coefficients but predictions are transformed into a logistic function. Outliers have big effects in LR models; so all outliers should be cleaned from the data set. Moreover, if there are highly correlated variables, LR may over fit.ROC curves and accuracy of thresholds are the main parameters to evaluate the model.

5.3. Random Forest

Random Forest (RF) is a supervised machine learning algorithm that can be used for classifications and regressions. In churn prediction, there are only two groups churned and not churned customers. RF algorithm builds random multiple DTs, and combines them into one model. Model can be tuned with parameters like; number of estimators, maximum tree depth, and maximum number of features used in tree, min sample leaf for internal nodes, etc

5.4. Exploratory Data Analysis

In data, there are 181,548 customers with 145 variables. We applied an EDA to the main variables such as voluntary churned, customer age, tenure, credit card limit, and the last month's transaction amount. We found that the churn ratio is 49%; and hence,51% of customers are loyal in the data set. The data is balanced for the target variable.

While importing data, there is no missing value in the data. All cleaning data process is completed in SQL, where the data is prepared. We checked for the outliers in the data. In last month expenditure variable, there were some negative values that show the customers have cash back transactions. We excluded these customers. Also, there were some outliers in the credit card limit, total sector credit card limit and customer age. We excluded these outlier customers from data.

When we look at the age of customers in Figure 1, the median age of churned customers are lower than not churned customers. This graph shows that churned customers are younger.

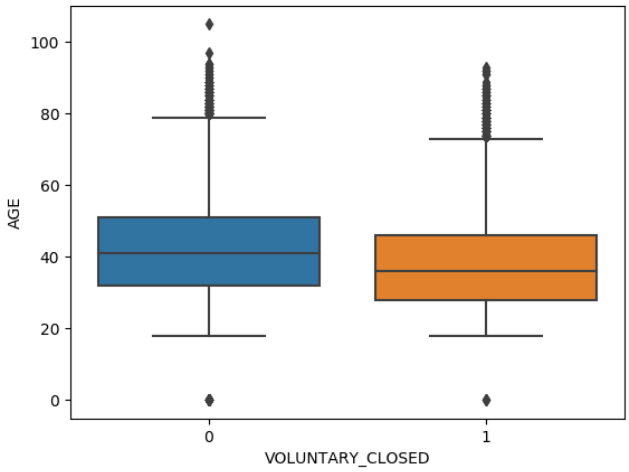

Figure 1. Churn based age distribution of customers

In tenure, churned customer are mostly new customers, first quartile is approximately 25 months and third quartile is 100 months, while live customers' first quartile is 40 months and third quartile is 160 months. Also the median of churned customer tenure is lower than 50 months, while live customer's tenure is 90 months (see Figure 2).

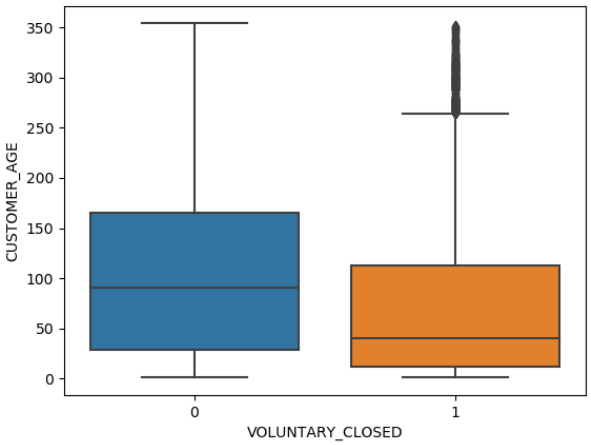

Figure 2. Churn based customer age (tenure) distribution

The customer limit has a big range in data, so we restrained the customer limit between 1 and 35,000 TL. It is seen that churned customer have lower customer limits than live customers (see Figure 3).

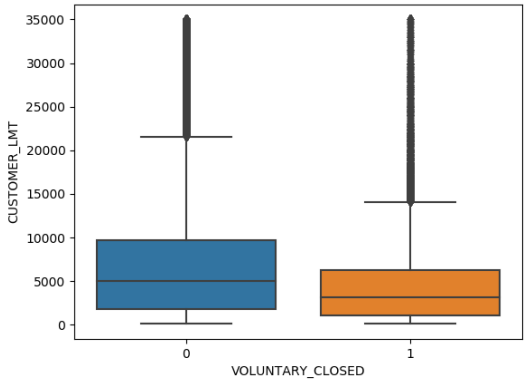

Figure 3. Churn based customer limit distribution

When we look at the activation distribution of customers, churn ratio is lower than active customers (see Figure 4). Inactive customers' churn ratio is two times bigger than active customers' churn ratio.

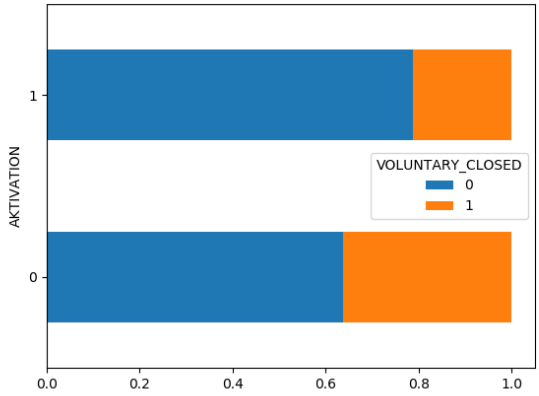

Figure 4. Activation vs. churned customers

6. RESULTS

In the Project, we aim to predict customer's churn probability. When we performa Exploratory Data Analysis, we realize that there are some outliers in the data set, especially in "Total Other Banks Limit", "Credit card limit" and "Age" variables. Also, the last month's purchase volume column has negative values, which shows that some customers have only charge back and, do not have any purchase transaction. We extracted these outlier customers. Data set has 145 variables and 181.247 customers. The target label is the voluntary churn in 3 months period. This label shows if the customer is churned or stays live. Therefore, we flagged 89.437 customers as churned (49% of the data), 91.810 customers as live (51% of the data). The data is separated into 144 Features and 1 label.

We analyze data in 3 different ways. First, we use original data and build the models using all features. Second, we perform Component Analysis (PCA) and try to select minimum number of features with high information value. After PCA, we build models on this data. Third, we first scale all features and then apply PCA and select again minimum number of features with high information value. The models are built in this data again. Before building models, we split randomly 70% of data as training and 30% of data as testing.

When we try different number of components, they have different explained variance ratio. 100 components out of 144 variables have 98% explained variance ratio. This means with 100 components approximately have the all information of the data to us. In data, to compare different models, we build models by using 3 different numbers of components, 70, 85 and 100.

Table 2. Number of components and explained variance ratio in PCA

PCA	Explained Variance Ratio
25	63%
45	77%
70	90%
75	92%
80	93%
85	95%
100	98%

First, we build Logistic regression to predict the churn, which is a binary label. There are no difference in using different C values for penalty. Also, train and test scores are almost the same. However, the accuracy score is increased with increasing the number of components in PCA. The maximum score is 63,3% with scaled and PCA data with 100 components.

Table 3. Accuracy scores of Logistic Regression in train and test data

Logistic regression	Original Data		Pca data (n=45)		Scaled and Pca Data (n=70)		Scaled and Pca Data (n=85)		Scaled and Pca Data (n=100)	
	train	test	train	test	train	test	train	test	train	test
default	57,8%	58,0%	56,9%	57,0%	60,6%	60,6%	62,8%	62,8%	63,2%	63,3%
c=0.01	57,8%	58,0%	56,9%	57,0%	60,6%	60,6%	62,8%	62,8%	63,2%	63,3%
c=100	57,7%	57,9%	56,9%	57,0%	60,6%	60,6%	62,8%	62,8%	63,2%	63,3%

Then, we build Decision Tree models. To find the best max depth, we draw ROC chart for different max depth values. While the tree depth increases, the difference of AUC score between train and test data increases. As it is seen in Figure 5, 6 is the maximum scored max depth both train and test data set.

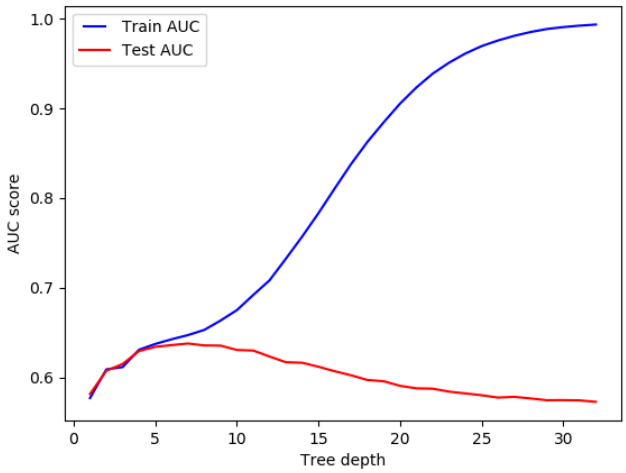

Figure 5. Max depth ROC curve.

We build decision trees with 6 max depth value. We compare models by changing decision criteria with entropy and gini. Although in building models, PCA helps to decrease the number of features, in original data, decision tree has the highest scores with 63%. In Table 4, it is seen that there is a small difference (0.3%) between gini and entropy models.

Table 4. Accuracy scores of Decision Tree in train and test data

Decision Tree	Original Data		Pca data (n=45)		Scaled and Pca		Scaled and Pca		Scaled and Pca	
	train	test	train	test	train	test	train	test	train	test
criterion="gini", max_depth=6	64,0%	63,5%	58,1%	57,4%	59,7%	58,9%	61,3%	60,7%	61,1%	60,6%
criterion="entropy", max_depth=6	63,7%	63,2%	58,3%	57,6%	59,6%	58,9%	61,1%	60,4%	60,8%	60,5%

As the third algorithm, we build Random Forest Algorithms. First, we try different models with changing the parameters and compare them between original data, PCA data and scaled & PCA data with different number of components. Table 5 shows that accuracy scores are higher in original data and scaled & PCA data with 100 components. Also, the accuracy score difference betweeen traind and test data decreases when max depth value become smaller.

Table 5. Accuracy scores of Random Forest in train and test data

Random Forest	Original Data		Pca data (n=45)		Scaled and Pca		Scaled and Pca		Scaled and Pca	
	train	test	train	test	train	test	train	test	train	test
n_estimators=100	99,8%	64,6%	99,8%	58,6%	99,8%	61,0%	99,8%	62,5%	99,8%	62,9%
max_depth=20, n_estimators=100	95,8%	65,2%	95,5%	59,5%	98,0%	61,5%	97,9%	63,1%	97,7%	63,4%
max_depth=10, n_estimators=100	69,3%	64,9%	69,3%	63,4%	67,6%	61,6%	69,4%	63,2%	69,3%	63,6%
max_depth=6, n_estimators=100	63,6%	63,0%	62,9%	62,2%	61,1%	60,5%	62,9%	62,2%	63,0%	62,3%

For the purpose to tune the parameters and find the best ones in Random Forest, we use GridSearch. In GridSearch, we define many different values for the parameters such as max tree depth, number of estimator, min samples leaf, max features. The best parameters found in GridSearch are as in Table 6.**Table 6.** Best random forest parameters in GridSearch.

Parameter	Value
Max_features	'sqrt'
n_estimators	145
Min_samples_leaf	5
Max_depth	20

In Random Forest, we apply the parameters shown in Table 6. Although the accuracy of train data is 94,2%, which is the highest accuracy, the accuracy score of test data is 65,1%. There is a huge difference in accuracy score between test and train, which means the model is over fitted. So, we try to tune the parameters again to avoid over fitting. In Table 7, it is seen that when we decrease the max depth again, over fitting problem is not totally solved. The gap between test and train accuracy score becomes smaller but still there is a small gap. As we increase the value of minimum samples leaf parameter, the accuracy score difference between train and test data decreases. While changing maximum depth parameter, we should tune minimum sampled leaf parameter too. Then, we use cross validation with 5 random parts and run this model again. Mean scores of all the scores after cross validation is 64%, which is very close to the model.

Table 7. Accuracy scores of Random Forest in train and test data

Random Forest	Original Data	
	Train	test
max_depth=30, n_estimators=145, min_samples_leaf=5, max_features='sqrt'	94,2%	65,1%
max_depth=25, n_estimators=145, min_samples_leaf=5, max_features='sqrt'	93,6%	65,4%
max_depth=20, n_estimators=145, min_samples_leaf=5, max_features='sqrt'	90,8%	65,2%
max_depth=15, n_estimators=145, min_samples_leaf=5, max_features='sqrt'	81,5%	65,2%
max_depth=12, n_estimators=145, min_samples_leaf=5, max_features='sqrt'	73,3%	65,0%
max_depth=10, n_estimators=145, min_samples_leaf=50, max_features='sqrt'	67,2%	64,6%
max_depth=10, n_estimators=145, min_samples_leaf=40, max_features='sqrt'	67,4%	64,6%
max_depth=8, n_estimators=145, min_samples_leaf=40, max_features='sqrt'	65,5%	64,1%
max_depth=7, n_estimators=145, min_samples_leaf=40, max_features='sqrt'	64,5%	63,6%
max_depth=7, n_estimators=145, min_samples_leaf=50, max_features='sqrt'	64,5%	63,5%
max_depth=12, n_estimators=145, min_samples_leaf=50, max_features='sqrt'	68,8%	64,8%
max_depth=12, n_estimators=145, min_samples_leaf=40, max_features='sqrt'	69,3%	64,9%

max_depth=15, n_estimators=145, min_samples_leaf=40, max_features='sqrt'	71,4%	65,1%
max_depth=15, n_estimators=145, min_samples_leaf=50, max_features='sqrt'	70,3%	65,1%
max_depth=15, n_estimators=145, min_samples_leaf=60, max_features='sqrt'	69,8%	65,1%
max_depth=15, n_estimators=145, min_samples_leaf=70, max_features='sqrt'	69,2%	65,0%
max_depth=20, n_estimators=145, min_samples_leaf=70, max_features='sqrt'	69,1%	65,1%
max_depth=25, n_estimators=145, min_samples_leaf=70, max_features='sqrt'	69,1%	65,0%
max_depth=15, n_estimators=145, min_samples_leaf=100, max_features='sqrt'	68,0%	64,8%

In conclusion, for churn prediction problem, we clean data, make feature selection and apply three different algorithms; Logistic regression, Decision tree and Random forest. We tune many parameters to find the best model. At the end, the best algorithm is Random forest in original data, with these parameters; max depth value is 20, number of estimator value is 145, min sampled leaf value is 70 and max feature selection is ‘sqrt’.

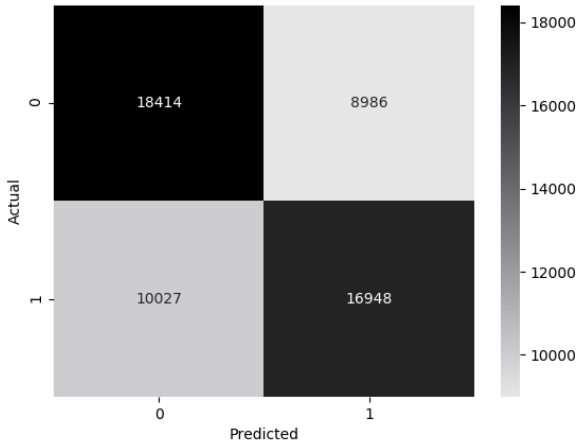

Figure 5. Confusion matrix of final model

7. CONCLUSION

In literature, it is seen that Principal Component Analysis and scaling are the main steps before building a machine learning algorithm. However, in this project, the scores are the highest in original data. In random forest algorithm, there are many trees for the target and also the number of features is defined in the model, so there is no need for scaling and feature selection again. In the project, we aim to predict the customers whom will churn. In data 41% of customers churned, with the Random Forest models, we predict 65% of customers who will churn. This is good prediction score. Bank can use this model for retention activities. In project, Support Vector Machine and Gradient boosting methods are tried but the results are not better than the RF models, so they are excluded. For the aim of increasing score, the data can be enriched more and we can add some other variables. After this point, again same models or other algorithms can be tried and more accuracy scores may be found.

APPENDIX A

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import codecs
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn import cross_validation
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_score, StratifiedKFold, LeaveOneOut
from sklearn import tree
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier


tag_file = "ss_proje_analiz_data_v4_180k.txt"
with codecs.open(tag_file, encoding='utf-8', errors='replace') as fh:
```

```
analiz = pd.read_csv(fh, header="infer", sep=";")


### EDA

analiz.describe()

# Churn based age distribution of customers

import seaborn as sns

sns.boxplot(x=analiz["VOLUNTARY_CLOSED"],y=analiz ["AGE"],data=analiz )


# Churn based tenure distribution of customers

sns.boxplot(x=analiz["VOLUNTARY_CLOSED"],y=analiz["CUSTOMER_AGE"],data=a
naliz)


# Churn based customer limit distribution

sns.boxplot(x=analiz["VOLUNTARY_CLOSED"],y=analiz["CUSTOMER_LMT"],data=
analiz)


#Activation vs. churned customers

churn1 = pd.crosstab(analiz['AKTIVATION'],data['VOLUNTARY_CLOSED'])

print(churn1)

churn_rate1 = churn1.div(churn1.sum(1).astype(float),

axis=0) # normalize the value

churn_rate1.plot(kind='barh', stacked=True)


data_new = analiz[(analiz.PREV_01_MONTH_SUM>=0) &
(analiz.TOPLAM_KKB_LIMIT<200000) &
(analiz.NUMBER_OF_OPEN_ACCOUNTS>0) & (analiz.AGE > 17)& (analiz.AGE<81)
&( analiz.TOPLAM_YKB_LIMIT<75000) ]
```

```
data_new.describe()

data_new.dtypes

#### Original Data Codes

X=data_new.drop('VOLUNTARY_CLOSED_3M',axis=1)

y=data_new['VOLUNTARY_CLOSED_3M']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=20)

# Logistic Regression

logreg = LogisticRegression()

logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)

print('Logistic regression score =',round(metrics.accuracy_score(y_test, y_pred),2))

#0,6 accuracy score

score_train = logreg.score(X_train, y_train)

score_test = logreg.score(X_test, y_test)

print(score_train)

print(score_test)


logreg2 = LogisticRegression(C = 100,random_state = 0)

logreg2.fit(X_train, y_train)

y_pred = logreg2.predict(X_test)

print('Logistic regression score =',round(metrics.accuracy_score(y_test, y_pred),2))

#0,6 accuracy score

score_train = logreg2.score(X_train, y_train)

score_test = logreg2.score(X_test, y_test)
```

```
print(score_train)

print(score_test)


conf = (metrics.confusion_matrix(y_test, y_pred))

cmap = sns.cubehelix_palette(50, hue=0.05, rot=0, light=0.9, dark=0, as_cmap=True)

sns.heatmap(conf,cmap = cmap,xticklabels=['0','1'],yticklabels=['0','1'],annot=True, fmt="d",)

plt.xlabel('Predicted')

plt.ylabel('Actual')

# Cross Validation

scores = cross_val_score(logreg, X, y, cv=5, scoring='accuracy')

print('Logistic regression of each partition\n',scores)

print('Mean score of all the scores after cross validation =',round(scores.mean(),2))

conf = (metrics.confusion_matrix(y_test, y_pred))

FP = conf[1][0]

FN = conf[0][1]

TP = conf[0][0]

TN = conf[1][1]

print('False Positive ',FP)

print('False Negative ',FN)

print('True Positive ',TP)

print('True Negative ',TN)

# Sensitivity, hit rate, recall, or true positive rate

TPR = TP/(TP+FN)

print('\nTrue Positive Rate :',round(TPR,2))
```

```
# Specificity or true negative rate
TNR = TN/(TN+FP)
print('\nTrue Negative Rate :',round(TNR,2))
# Precision or positive predictive value
PPV = TP/(TP+FP)
print('\nPositive Predictive Value :',round(PPV,2))
# Negative predictive value
NPV = TN/(TN+FN)
print('\nNegative Predictive Value :',round(NPV,2))
# Fall out or false positive rate
FPR = FP/(FP+TN)
print('\nFalse Positive Rate :',round(FPR,2))
# False negative rate
FNR = FN/(TP+FN)
print('\nFalse Negative Rate :',round(FNR,2))
# False discovery rate
FDR = FP/(TP+FP)
print('\nFalse Discovery Rate :',round(FDR,2))
# Overall accuracy
ACC = (TP+TN)/(TP+FP+FN+TN)
print('\nOverall accuracy :',round(ACC,2))

# Decision Tree
clf_tree=tree.DecisionTreeClassifier(criterion="gini", max_depth=6)
```

```
clf_tree.fit(X_train, y_train)

predicted=clf_tree.predict(X_test)

print(confusion_matrix(y_test, predicted))

print(accuracy_score(y_test,predicted))

score_train = clf_tree.score(X_train, y_train)

score_test = clf_tree.score(X_test, y_test)

print(score_train)

print(score_test)


clf_tree=tree.DecisionTreeClassifier(criterion="gini", max_depth=12)

clf_tree.fit(X_train, y_train)

predicted=clf_tree.predict(X_test)

print(confusion_matrix(y_test, predicted))

print(accuracy_score(y_test,predicted))

score_train = clf_tree.score(X_train, y_train)

score_test = clf_tree.score(X_test, y_test)

print(score_train)

print(score_test)


clf_tree1=tree.DecisionTreeClassifier(criterion="entropy",random_state=100,
max_depth=12)

clf_tree1=clf_tree1.fit(X_train, y_train)

tree.export_graphviz(clf_tree1, out_file='tree.dot')


predicted1=clf_tree1.predict(X_test)
```

```
print(confusion_matrix(y_test, predicted1))
print(accuracy_score(y_test,predicted1))
score_train = clf_tree1.score(X_train, y_train)
score_test = clf_tree1.score(X_test, y_test)
print(score_train)
print(score_test)

from sklearn.ensemble import RandomForestClassifier
# Random Forest
forest=RandomForestClassifier(n_estimators=100, random_state=0)
forest.fit(X_train, y_train)
print("Accuracy on training set: {: .3f}".format(forest.score(X_train, y_train)))
print("Accuracy on test set: {: .3f}".format(forest.score(X_test, y_test)))

forest2=RandomForestClassifier(max_depth=20, n_estimators=100, random_state=0)
forest2.fit(X_train, y_train)
print("Accuracy on training set: {: .3f}".format(forest2.score(X_train, y_train)))
print("Accuracy on test set: {: .3f}".format(forest2.score(X_test, y_test)))


forest2=RandomForestClassifier(max_depth=20, n_estimators=110, random_state=0)
forest2.fit(X_train, y_train)
print("Accuracy on training set: {: .3f}".format(forest2.score(X_train, y_train)))
print("Accuracy on test set: {: .3f}".format(forest2.score(X_test, y_test)))
```

```
forest2=RandomForestClassifier(max_depth=6, n_estimators=100, random_state=0)

forest2.fit(X_train, y_train)

print("Accuracy on training set: {: .3f}".format(forest2.score(X_train, y_train)))

print("Accuracy on test set: {: .3f}".format(forest2.score(X_test, y_test)))


from sklearn.model_selection import GridSearchCV

rfc = RandomForestClassifier(n_jobs=-1,  oob_score=True)

# Use a grid over parameters of interest

param_grid = {

"n_estimators": [45, 90, 120, 145],

'max_features': ['auto', 'sqrt', 'log2']}


CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv=2)

CV_rfc.fit(X_train, y_train)

print(CV_rfc.best_params_)


forest2 = RandomForestClassifier(max_depth=12, n_estimators=145,
min_samples_leaf=5, max_features='sqrt', random_state=0)

forest2.fit(X_train, y_train)

print("Accuracy on training set: {: .3f}".format(forest2.score(X_train, y_train)))

print("Accuracy on test set: {: .3f}".format(forest2.score(X_test, y_test)))


forest3 = RandomForestClassifier(max_depth=12, n_estimators=120,
min_samples_leaf=3, max_features='sqrt', random_state=0)
```

```
forest3.fit(X_train, y_train)

print("Accuracy on training set: {: .3f}".format(forest3.score(X_train, y_train)))

print("Accuracy on test set: {: .3f}".format(forest3.score(X_test, y_test)))


from sklearn.model_selection import GridSearchCV

# Create the parameter grid based on the results of random search

param_grid = {

'bootstrap': [True],

'max_depth': [80, 90, 100, 110],

'max_features': [2, 3],

'min_samples_leaf': [3, 4, 5],

'min_samples_split': [8, 10, 12],

'n_estimators': [100, 200, 300, 1000]

}


# Create a based model

rf = RandomForestRegressor()

# Instantiate the grid search model

grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,

cv = 3, n_jobs = -1, verbose = 2)

forest2 = RandomForestClassifier(max_depth=12, n_estimators=145, min_samples_leaf=5, max_features='sqrt', random_state=0)

forest2.fit(X_train, y_train)

print("Accuracy on training set: {: .3f}".format(forest2.score(X_train, y_train)))

print("Accuracy on test set: {: .3f}".format(forest2.score(X_test, y_test)))
```

```
forest3 = RandomForestClassifier(max_depth=20, n_estimators=145, min_samples_leaf=70, max_features='sqrt', random_state=0)

forest3.fit(X_train, y_train)

print("Accuracy on training set: {: .3f}".format(forest3.score(X_train, y_train)))

print("Accuracy on test set: {: .3f}".format(forest3.score(X_test, y_test)))

y_pred = forest3.predict(X_test)

conf = (metrics.confusion_matrix(y_test, y_pred))

cmap = sns.cubehelix_palette(50, hue=0.05, rot=0, light=0.9, dark=0, as_cmap=True)

sns.heatmap(conf,cmap = cmap,xticklabels=['0','1'],yticklabels=['0','1'],annot=True, fmt="d",)

plt.xlabel('Predicted')

plt.ylabel('Actual')



# Cross Validation

scores = cross_val_score(forest3, X, y, cv=5, scoring='accuracy')

print('Random Forest of each partition\n',scores)

print('Mean score of all the scores after cross validation =',round(scores.mean(),2))

conf = (metrics.confusion_matrix(y_test, y_pred))

FP = conf[1][0]

FN = conf[0][1]

TP = conf[0][0]

TN = conf[1][1]

print('False Positive ',FP)
```

```
print('False Negative ',FN)

print('True Positive ',TP)

print('True Negative ',TN)

# Sensitivity, hit rate, recall, or true positive rate

TPR = TP/(TP+FN)

print('\nTrue Positive Rate :',round(TPR,2))

# Specificity or true negative rate

TNR = TN/(TN+FP)

print('\nTrue Negative Rate :',round(TNR,2))

# Precision or positive predictive value

PPV = TP/(TP+FP)

print('\nPositive Predictive Value :',round(PPV,2))

# Negative predictive value

NPV = TN/(TN+FN)

print('\nNegative Predictive Value :',round(NPV,2))

# Fall out or false positive rate

FPR = FP/(FP+TN)

print('\nFalse Positive Rate :',round(FPR,2))

# False negative rate

FNR = FN/(TP+FN)

#### Pca and Scaled Data

pca = PCA(n_components=100)

# fit PCA model to data

pca.fit(X)
```

```
# transform data onto the first two principal components
X_pca = pca.transform(X)
print("Original shape: {}".format(str(X.shape)))
print("Reduced shape: {}".format(str(X_pca.shape)))
print(pca.explained_variance_ratio_)



pca2=PCA(n_components=85)
pca2.fit(X)
X_pca2 = pca2.transform(X)
print(pca2.explained_variance_ratio_)

pca3=PCA(n_components=70)
pca3.fit(X)
X_pca3 = pca3.transform(X)
print(pca3.explained_variance_ratio_)

pca4=PCA(n_components=45)
pca4.fit(X)
X_pca4 = pca4.transform(X)
print(pca4.explained_variance_ratio_)
pca5=PCA(n_components=25)
pca5.fit(X)
X_pca5 = pca5.transform(X)
```

```
print(pca5.explained_variance_ratio_)


scaler = preprocessing.StandardScaler()

X_Scaled = scaler.fit_transform(X)

# keep the first two principal components of the data

pca = PCA(n_components=100)

# fit PCA model to data

pca.fit(X_Scaled)

# transform data onto the first two principal components

X_pca = pca.transform(X_Scaled)

print("Original shape: {}".format(str(X_Scaled.shape)))

print("Reduced shape: {}".format(str(X_pca.shape)))


print(pca.explained_variance_ratio_)



X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.30, random_state=20)

# Logistic Regression

logreg = LogisticRegression()

logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)

##print('Logistic regression score train=',round(metrics.accuracy_score(y_train, y_pred),2))
```

```
print('Logistic regression score test=',round(metrics.accuracy_score(y_test, y_pred),2))
score_train = logreg.score(X_train, y_train)
score_test = logreg.score(X_test, y_test)
print(score_train)
print(score_test)

logreg2 = LogisticRegression(C = 100,random_state = 0)
logreg2.fit(X_train, y_train)
y_pred = logreg2.predict(X_test)
print('Logistic regression score =',round(metrics.accuracy_score(y_test, y_pred),2))
#0,6 accuracy score
score_train = logreg2.score(X_train, y_train)
score_test = logreg2.score(X_test, y_test)
print(score_train)
print(score_test)

logreg3 = LogisticRegression(C = 0.001,random_state = 0)
logreg3.fit(X_train, y_train)
y_pred3 = logreg3.predict(X_test)
print('Logistic regression score =',round(metrics.accuracy_score(y_test, y_pred3),2))
#0,57 accuracy score
score_train = logreg3.score(X_train, y_train)
score_test = logreg3.score(X_test, y_test)
print(score_train)
```

```
print(score_test)


conf = (metrics.confusion_matrix(y_test, y_pred))

cmap = sns.cubehelix_palette(50, hue=0.05, rot=0, light=0.9, dark=0, as_cmap=True)

sns.heatmap(conf,cmap = cmap,xticklabels=['0','1'],yticklabels=['0','1'],annot=True, fmt="d",)

plt.xlabel('Predicted')

plt.ylabel('Actual')



# Cross Validation

scores = cross_val_score(logreg, X, y, cv=5, scoring='accuracy')

print('Logistic regression of each partition\n',scores)

print('Mean score of all the scores after cross validation =',round(scores.mean(),2))


# SVM Classifier

clf = SVC(kernel='linear',C=10)

clf.fit(X_train,y_train)

scoretrain = clf.score(X_train,y_train)

scoretest  = clf.score(X_test,y_test)

print("Linear SVM value of C:{}, training score :{:2f} , Test Score: {:2f} \n".format(scoretrain,scoretest))



# Cross Validation
```

```
clf1 = SVC(kernel='linear', C=20).fit(X_train, y_train)

scores = cross_val_score(clf1, X_train, Y_train, cv=5)

print("The Cross Validation Score :" + str(scores))

print("The Average Cross Validation Score :" + str(scores.mean()))


# Decision Tree

clf_tree=tree.DecisionTreeClassifier(criterion="gini", max_depth=6)

clf_tree.fit(X_train, y_train)

predicted=clf_tree.predict(X_test)

print(confusion_matrix(y_test, predicted))

print(accuracy_score(y_test,predicted))

score_train = clf_tree.score(X_train, y_train)

score_test = clf_tree.score(X_test, y_test)

print(score_train)

print(score_test)


clf_tree1=tree.DecisionTreeClassifier(criterion="entropy",random_state=100,
max_depth=6)

clf_tree1=clf_tree1.fit(X_train, y_train)

tree.export_graphviz(clf_tree1, out_file='tree.dot')


predicted1=clf_tree1.predict(X_test)

print(confusion_matrix(y_test, predicted1))

print(accuracy_score(y_test,predicted1))

score_train = clf_tree1.score(X_train, y_train)
```

```
score_test = clf_tree1.score(X_test, y_test)
print(score_train)
print(score_test)

from sklearn.metrics import roc_curve, auc
max_depths = np.linspace(1, 32, 32, endpoint=True)
train_results = []
test_results = []
for max_depth in max_depths:
dt = DecisionTreeClassifier(max_depth=max_depth)
dt.fit(X_train, y_train)
train_pred = dt.predict(X_train)
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_train, train_pred)
roc_auc = auc(false_positive_rate, true_positive_rate)
# Add auc score to previous train results
train_results.append(roc_auc)
y_pred = dt.predict(X_test)
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(false_positive_rate, true_positive_rate)
# Add auc score to previous test results
test_results.append(roc_auc)
from matplotlib.legend_handler import HandlerLine2D
line1, = plt.plot(max_depths, train_results, 'b', label="Train AUC")
line2, = plt.plot(max_depths, test_results, 'r', label="Test AUC")
```

```
plt.legend(handler_map={line1: HandlerLine2D(numpoints=2)})
plt.ylabel('AUC score')
plt.xlabel('Tree depth')
plt.show()

from sklearn.ensemble import RandomForestClassifier
# Random Forest
forest=RandomForestClassifier(n_estimators=100, random_state=0)
forest.fit(X_train, y_train)
print("Accuracy on training set: {: .3f}".format(forest.score(X_train, y_train)))
print("Accuracy on test set: {: .3f}".format(forest.score(X_test, y_test)))

forest2=RandomForestClassifier(max_depth=20, n_estimators=100, random_state=0)
forest2.fit(X_train, y_train)
print("Accuracy on training set: {: .3f}".format(forest2.score(X_train, y_train)))
print("Accuracy on test set: {: .3f}".format(forest2.score(X_test, y_test)))

forest3=RandomForestClassifier(max_depth=10, n_estimators=100, random_state=0)
forest3.fit(X_train, y_train)
print("Accuracy on training set: {: .3f}".format(forest3.score(X_train, y_train)))
print("Accuracy on test set: {: .3f}".format(forest3.score(X_test, y_test)))
```

```
forest4=RandomForestClassifier(max_depth=15, n_estimators=100, min_samples_leaf=5, max_features='sqrt', random_state=0)

forest4.fit(X_train, y_train)

print("Accuracy on training set: {: .3f}".format(forest4.score(X_train, y_train)))

print("Accuracy on test set: {: .3f}".format(forest4.score(X_test, y_test)))



#### PCA Data

data_new = analiz[(analiz.PREV_01_MONTH_SUM>=0) & (analiz.TOPLAM_KKB_LIMIT<200000) & (analiz.AGE > 17)& (analiz.AGE<81) &( analiz.TOPLAM_YKB_LIMIT<75000)]



X=data_new.drop('VOLUNTARY_CLOSED_3M',axis=1)

y=data_new['VOLUNTARY_CLOSED_3M']



pca = PCA(n_components=45)

# fit PCA model to data

pca.fit(X)

# transform data onto the first two principal components

X_pca = pca.transform(X)

print("Original shape: {}".format(str(X.shape)))

print("Reduced shape: {}".format(str(X_pca.shape)))

print(pca.explained_variance_ratio_)

X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.30, random_state=20)



# Logistic Regression
```

```
logreg = LogisticRegression()

logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)

print('Logistic regression score =',round(metrics.accuracy_score(y_test, y_pred),2))


logreg2 = LogisticRegression(C = 100,random_state = 0)

logreg2.fit(X_train, y_train)

y_pred = logreg2.predict(X_test)

print('Logistic regression score =',round(metrics.accuracy_score(y_test, y_pred),2))

logreg3 = LogisticRegression(C = 0.001,random_state = 0)

logreg3.fit(X_train, y_train)

y_pred = logreg3.predict(X_test)

print('Logistic regression score =',round(metrics.accuracy_score(y_test, y_pred),2))

conf = (metrics.confusion_matrix(y_test, y_pred))

cmap = sns.cubehelix_palette(50, hue=0.05, rot=0, light=0.9, dark=0, as_cmap=True)

sns.heatmap(conf,cmap = cmap,xticklabels=['0','1'],yticklabels=['0','1'],annot=True,
fmt="d",)

plt.xlabel('Predicted')

plt.ylabel('Actual')


# Cross Validation

scores = cross_val_score(logreg, X, y, cv=5, scoring='accuracy')

print('Logistic regression of each partition\n',scores)

print('Mean score of all the scores after cross validation =',round(scores.mean(),2))

# SVM Classifier
```

```
from sklearn import svm

model = svm.svc(kernel='linear', c=1, gamma=1)

model.fit(X_train, y_train)

model.score(X_test, y_test)


clf = SVC(kernel='linear',C=10)

clf.fit(X_train,y_train)

scoretrain = clf.score(X_train,y_train)

scoretest  = clf.score(X_test,y_test)

print("Linear SVM value of C:{}, training score :{:2f} , Test Score: {:2f} \n".format(scoretrain,scoretest))


# Cross Validation

clf1 = SVC(kernel='linear', C=20).fit(X_train, Y_train)

scores = cross_val_score(clf1, X_train, Y_train, cv=5)

print("The Cross Validation Score :" + str(scores))

print("The Average Cross Validation Score :" + str(scores.mean()))


# Decision Tree

clf_tree=tree.DecisionTreeClassifier(criterion="gini", max_depth=6)

clf_tree.fit(X_train, y_train)

predicted=clf_tree.predict(X_test)

print(confusion_matrix(y_test, predicted))

print(accuracy_score(y_test,predicted))
```

```
clf_tree1=tree.DecisionTreeClassifier(criterion="entropy",random_state=100,
max_depth=6)

clf_tree1=clf_tree1.fit(X_train, y_train)

tree.export_graphviz(clf_tree1, out_file='tree.dot')

predicted1=clf_tree1.predict(X_test)

print(confusion_matrix(y_test, predicted1))

print(accuracy_score(y_test,predicted1))

# Random Forest

forest=RandomForestClassifier(n_estimators=100, random_state=0)

forest.fit(X_train, y_train)

print("Accuracy on training set: {: .3f}".format(forest.score(X_train, y_train)))

print("Accuracy on test set: {: .3f}".format(forest.score(X_test, y_test)))


forest2=RandomForestClassifier(max_depth=20, n_estimators=100, random_state=0)

forest2.fit(X_train, y_train)

print("Accuracy on training set: {: .3f}".format(forest2.score(X_train, y_train)))

print("Accuracy on test set: {: .3f}".format(forest2.score(X_test, y_test)))

forest3=RandomForestClassifier(max_depth=6, n_estimators=100, random_state=0)

forest3.fit(X_train, y_train)

print("Accuracy on training set: {: .3f}".format(forest3.score(X_train, y_train)))

print("Accuracy on test set: {: .3f}".format(forest3.score(X_test, y_test)))
```

REFERENCES

Ballings, M., Poel, D. (2012), Customer event history for churn prediction: How long is long enough? Expert Systems with Applications 39, 13517–13522

Bolton, R. N. (1998), A dynamic model of the duration of the customer's relationshipwith a continuous service provider: the role of satisfaction, Marketing Science17 (1), 45–65

Bolton, R. N., Kannan, P. K., Bramlett, M. D. (2000), Implications of loyalty program mem-bership and service experiences for customer retention and value, Journal ofthe Academy of Marketing Science 28 (1), 95–108.

Burez, J., Poel, D. (2007), CRM at a pay-TV company: using analytical models to reduce customer attrition by targetedmarketing for subscription services. Expert Syst. Appl. 32 (2), 277–288.

Colgate, M., Danaher, P.(2000) , Implementing a customer relationship strategy: the asymmetric impact of poor versus excellent execution. J. Acad. Market. Sci., 28 (3) 375–387 .

Coussement, K., Lessmann, S., Verstraeten, G. (2017), A comparative analysis of data preparation algorithms for customer churn prediction: A case study in the telecommunication industry. Decision Support Systems, 95 , 27–36

Farquad M A. H, Ravia, V., Raju, S. B(2014), Churn prediction using comprehensible support vector machine:An analytical CRM application. Applied Soft Computing 19, 31–40

Glady, N., Baesens, B., Croux, C. (2009), Modeling churn using customer lifetime value. a European Journal of Operational Research 197, 402–411

Gordini, N., Veglio, V. (2017), Customers churn prediction and marketing retention strategies. An application of support vector machines based on the AUC parameter-selection technique in B2B e-commerce industry. Industrial Marketing Management 62, 100–107

Hadden, J, Tiwari, A., Roy, R., Ruta, D. (2008), Churn Prediction: Does Technology Matter? World Academy of Science, Engineering and Technology International Journal of Industrial and Manufacturing Engineering Vol:2, No:4

Larivière, B., Poel, D. V.(2004), Investigating the role of product features inpreventing customer churn, by using survival analysis and choice modelling:the case of financial services, Expert Systems with Applications 27 (2),277–285.