DATA SCIENCE FINAL PROJECT

CUSTOMER MASS FORECAST
IN THE BANKING INDUSTRY

Our Process

STEP 01

Data Analysis

STEP 02

Modelling

STEP 03

Feature Engineering

STEP 04

Feature Selection

STEP 05

Best Model Selection for the Target Audience

CONTENTS

01

PROBLEM & SOLUTION

To identify customers who can enter this audience in order to present the new product to customers with annual net sales of more than 5 million TL.

02

OUR PROCESS

To determine which customer's net sales exceed 5 million TL by correlating data and establishing the best model.

03

AIM

Which customers can have net sales of more than 5 million TL? How can we predict this and introduce the new product with the best model by looking at the customer's data?

OUR PROJECT

The bank develops a new product for commercial customers. As a result of the studies, customers with annual net sales of more than 5 million TL are considered to be the most suitable audience to offer this new product. For this reason, we aimed to establish the best model and estimate the target audience by examining the statistical relationships between them by obtaining new variables with the information of customers with incomplete financial information and 10 variables consisting of 11 K records.

UNDERSTAND THE PROBLEM

MISSING VALUES

Not every company has current financial statements. Without financial statements, we would not be able to obtain appropriate net sales information for customers. We want to make sure we reach all the targeted customers. Therefore, we have to find a way to decide which of these non-financial customers has net sales of more than TL 5 million.

DATA ASSOCIATION

If the total loan balance of the customers in the banking sector is over 5 million TL, can their net sales also exceed 5 million TL?

OUR SOLUTION

DATA ANALYSIS

By examining the data, it is decided what should be done on the data.

MODELLING

The best parameters for each model are determined and the results are compared.

FEATURE ENGINEERING

After looking at the relational analyses, new variables that can affect the model are derived.

FEAUTURE SELECTION

Selection of the variable that most influences the target variable

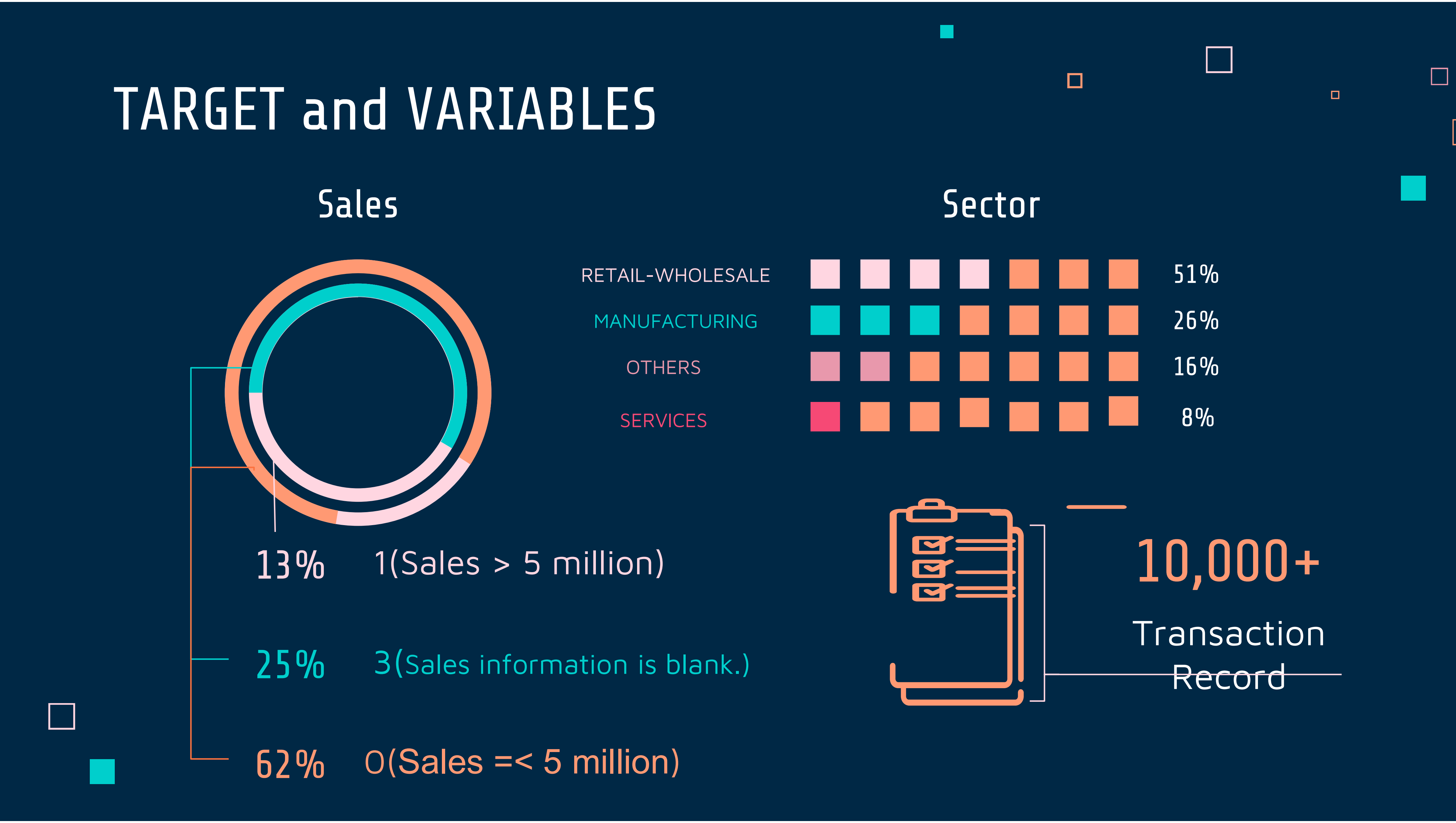
DECIDING THE BEST MODEL AND DETERMINING THE TARGET AUDIENCE

The model that gives the best result is selected and the final model is created and the target audience is estimated.

DATA ANALYSIS

Explanatory data analysis and examination of variables

01


TARGET and VARIABLES
Sales
13% 1(Sales > 5 million)
25% 3(Sales information is blank.)
62% 0(Sales =< 5 million)
Sector
RETAIL-WHOLESALE 51%
MANUFACTURING 26%
OTHERS 16%
SERVICES 8%
10,000+
Transaction Record

EDA(Exploratory Data Analysis)

→ Since the sales information is not known, we first removed the blank ones from the data.

```
df_3 = df[df['Sales'] == 3]

df = df[~(df['Sales'] == 3)]
```

Data information after removing the non-sales information

```
df.shape
```

```
(8526, 10)
```

After removing the records with empty sales information, when we look at the ratio of values with 0 sales information, we see that it is **83%**, and the ratio of those with 1 is **17%.** From this, we understand that your data has an unstable data structure.

EDA(Exploratory Data Analysis)

According to the analysis made, the results of our data

```
##################### Shape #####################
(8526, 10)
##################### Types #####################
YEAR                           int64
Customer_num                  object
Establishment_Date    datetime64[ns]
Number_of_Emp                float64
Profit                       float64
Sector                        object
Region                        object
Total_Risk                   float64
Total_Limit                  float64
Sales                          int64
dtype: object
```

```
##################### NA #####################
YEAR                    0
Customer_num            0
Establishment_Date      0
Number_of_Emp         391
Profit                  0
Sector                  0
Region                972
Total_Risk             93
Total_Limit            93
Sales                   0
dtype: int64
```

EDA(Exploratory Data Analysis)

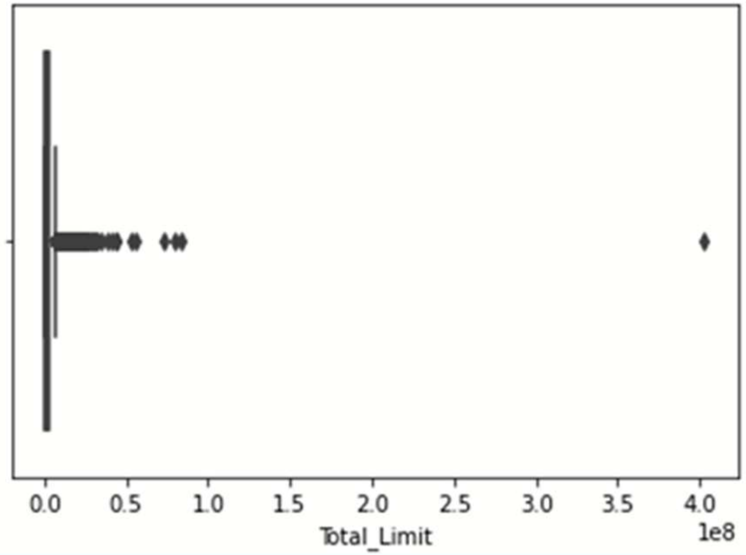
According to the analysis made, the results of our data

	YEAR	Customer_num	Establishment_Date	Number_of_Emp	Profit	Sector	Region	Total_Risk	Total_Limit	Sales
0	2017	RATI9590GZD	26.01.2001	8.000	NaN	RETAIL-WHOLESALE	Marmara Region	70917.000	7000007.000	3
1	2015	RATI2539VHR	24.02.1994	21.000	32615.000	MANUFACTURING	Central Anatolia Region	682602.000	2354029.000	0
2	2010	RATI4481GNN	25.01.1996	7.000	282834.000	RETAIL-WHOLESALE	Mediterranean Region	115581.000	592922.000	0
3	2012	RATI4948THA	7.04.2004	34.000	35597.000	MANUFACTURING	Southeastern Anatolia Region	39334.000	2471021.000	1
4	2013	RATI8841WYZ	24.04.2006	15.000	134259.000	SERVICES	Aegean Region	71295.000	506238.000	0
5	2011	RATI6581GZV	27.01.2007	25.000	NaN	OTHERS	Marmara Region	524053.000	1225401.000	3

EDA(Exploratory Data Analysis)

Box Plot Charts of Variables

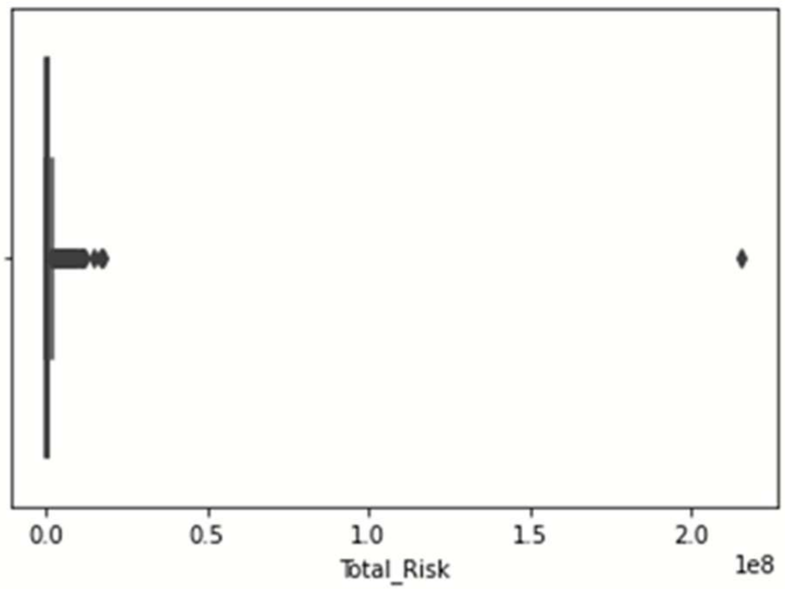
```
# #Total limit değişkenindeki son durum için box plot ile outlierlara baktık.
sns.boxplot(df['Total_Limit'])
plt.show()
```



EDA(Exploratory Data Analysis)

Box Plot Charts of Variables

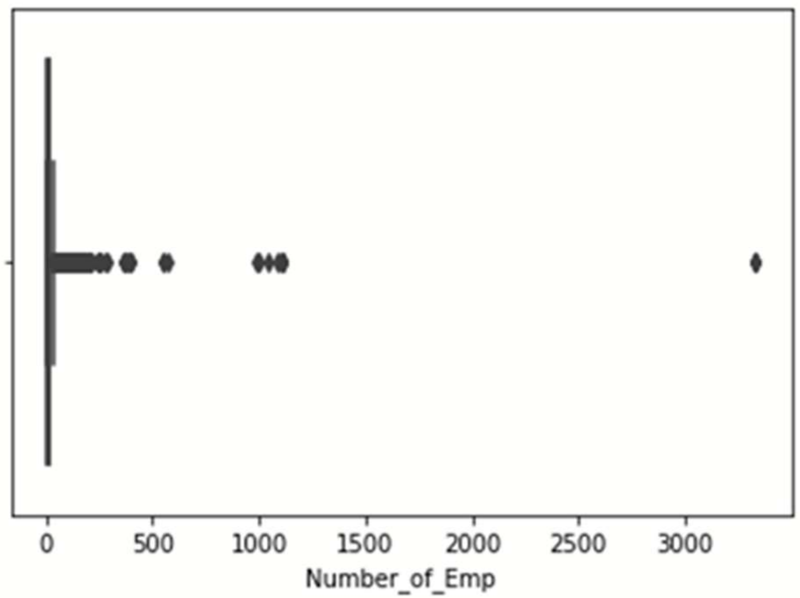
```
sns.boxplot(df['Total_Risk'])
plt.show()
```



EDA(Exploratory Data Analysis)

Box Plot Charts of Variables

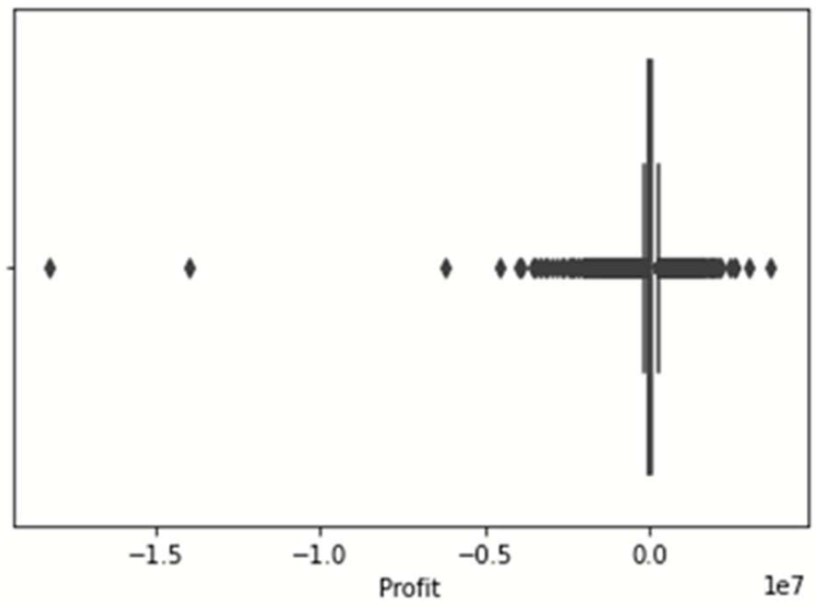
```
sns.boxplot(df['Number_of_Emp'])
plt.show()
```



EDA(Exploratory Data Analysis)

Box Plot Charts of Variables

```
sns.boxplot(df['Profit'])
plt.show()
```



EDA(Exploratory Data Analysis)

Outlier Analysis

```
for col in num_cols:
    print(col, check_outlier(df, col))
```

```
Establishment_Date True
Number_of_Emp True
Profit True
Total_Risk True
Total_Limit True
```

EDA(Exploratory Data Analysis)

Definition of Data

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
YEAR	8526.000	2013.981	2.208	2010.000	2012.000	2014.000	2016.000	2017.000
Number_of_Emp	8135.000	16.297	77.654	1.000	4.000	8.000	16.000	3333.000
Profit	8526.000	61383.427	408409.023	-18284524.000	13497.250	45052.500	116589.750	3724579.000
Total_Risk	8433.000	897268.432	2631801.492	0.000	156447.000	453724.000	1094305.000	215495035.000
Total_Limit	8433.000	2488763.672	5577535.906	2972.000	594056.000	1403481.000	3137171.000	402724973.000
Sales	8526.000	0.169	0.375	0.000	0.000	0.000	0.000	1.000

EDA(Exploratory Data Analysis)

Correlation Analysis Between Variables of Raw Data

```
# Korelasyonların İncelenmesi
df.corr()
```

	YEAR	Number_of_Emp	Profit	Total_Risk	Total_Limit	Sales
YEAR	1.000	-0.040	0.003	0.044	0.064	-0.017
Number_of_Emp	-0.040	1.000	0.005	0.012	0.020	0.055
Profit	0.003	0.005	1.000	-0.016	-0.010	0.089
Total_Risk	0.044	0.012	-0.016	1.000	0.890	0.094
Total_Limit	0.064	0.020	-0.010	0.890	1.000	0.146
Sales	-0.017	0.055	0.089	0.094	0.146	1.000

When we look at the correlation analysis, we see that the independent variable that most affects the target variable is the total limit, total risk, profit and number of employees.

When we looked at the correlation between independent variables, we saw that the variables with the highest correlation were total limit and total risk.

MODELLING

Establishing the base model as the first step, then obtaining the best model score

02

Creating the Base Model

→ Without creating all the new variables, we calculated how many years ago he contacted the bank by simply subtracting the date from the year information and added it to the data set.
→ For categorical variables, we encoded our data with one hot encoder.

→ We split our data as a train test and got our first results by inserting it into the models.

```
# roc_auc: 0.7732 (LightGBM)
# roc_auc: 0.7874 (CatBoost)
# roc_auc: 0.7528 (XGBoost)
```

FILLING IN MISSING VALUES

In order to run all models, we first fill in the missing values.
For the region variable, we filled in the missing values according to the mode of the sales variable.

```
def missing_value(dataframe):
    dataframe["Total_Risk"] = dataframe["Total_Risk"].fillna(dataframe.groupby("Sales")["Total_Risk"].transform("median"))
    dataframe["Total_Limit"] = dataframe["Total_Limit"].fillna(dataframe.groupby("Sales")["Total_Limit"].transform("median"))
    dataframe["Number_of_Emp"] = dataframe["Number_of_Emp"].fillna(dataframe.groupby("Sales")["Number_of_Emp"].transform("median"))
    dataframe = dataframe.apply(lambda x: x.fillna(x.mode()[0]) if (x.dtype == "O" and len(x.unique()) <= 10) else x, axis=0)
    check_df(dataframe)
    return dataframe
```

FEATURE ENGINEERING

Generating new variables that can affect the target variable

03

FEATURE ENGINEERING

Based on the correlation analysis we made, a total of 27 variables were derived after feature engineering.

```
df.shape
 (8526, 27)
```

```
(['CUSTOMER_NUM', 'NUMBER_OF_EMP', 'PROFIT', 'SECTOR', 'REGION', 'TOTAL_RISK', 'TOTAL_LIMIT', 'SALES', 'CUS_TENURE', 'LAST_CREDIT_TIME', 'NUMBER_OF_TRANSACTIONS',
  'SUM_OF_RISK', 'SUM_OF_LIMIT', 'SUM_OF_PROFIT', 'YEAR_TENURE', 'ESTABLISHMENT_TENURE', 'TETABLISEMENT+NEMPLOYEUR', 'TETABLISEMENT*NEMPLOYEUR', 'TOTAL_RISK_RATE',
  'TYEAR+TESTABLISMENT+NEMPL', 'TYEAR*TESTABLISMENT*NEMPL', 'NEW_ESTABLISHMENT_TENURE', 'CAT_NUMBER_OF_EMP', 'NEW_PROFIT', 'NEW_YEAR_TENURE', 'NEW_YEAR_PROFIT',
  'NEW_YEAR_SECTOR'],
 dtype='object')
```

→ We encode the data again with one hot encoder.

```
(8526, 82)
```

Robust Scaler

We used robust scale because it scales by quarters and there are many outliers in our data. When we run the model without scaling first and then scale it, when we look at the results, we saw that it did not affect the outliers, but it did affect the speed of the model.

Creating a Test-Train Set

→ Since our data is unbalanced, we used this method to divide the data homogeneously.

```
##################################################
# SPLIT INTO TEST AND TRAIN
##################################################
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold, cross_val_score, StratifiedShuffleSplit,cross_validate
def model_prep(dataframe,test_size=0.50):
    dataframe = dataframe.drop(['CUSTOMER_NUM'], axis=1)
    y = dataframe['SALES']
    X = dataframe.drop(['SALES'], axis=1)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=45,stratify=y)
    return ohe_df,X,y,X_train,y_train,X_test,y_test

ohe_df,X,y,X_train,y_train,X_test,y_test=model_prep(ohe_df)
```

Base Model Comparison

```
# skf = StratifiedKFold(n_splits=10) icin cikan sonuclar
# roc_auc: 0.761 (LR)
# roc_auc: 0.7221 (KNN)
# roc_auc: 0.7493 (SVC)
# roc_auc: 0.5985 (CART)
# roc_auc: 0.8123 (RF)
# roc_auc: 0.7752 (Adaboost)
# roc_auc: 0.7989 (GBM)
# roc_auc: 0.8006 (LightGBM)
# roc_auc: 0.8173 (CatBoost)
# roc_auc: 0.7995 (XGBoost)
```

Outliers

→ When we removed outliers from the data, we observed that it did not affect the success of the model, and for this reason, we found it appropriate to leave the outliers in the data.

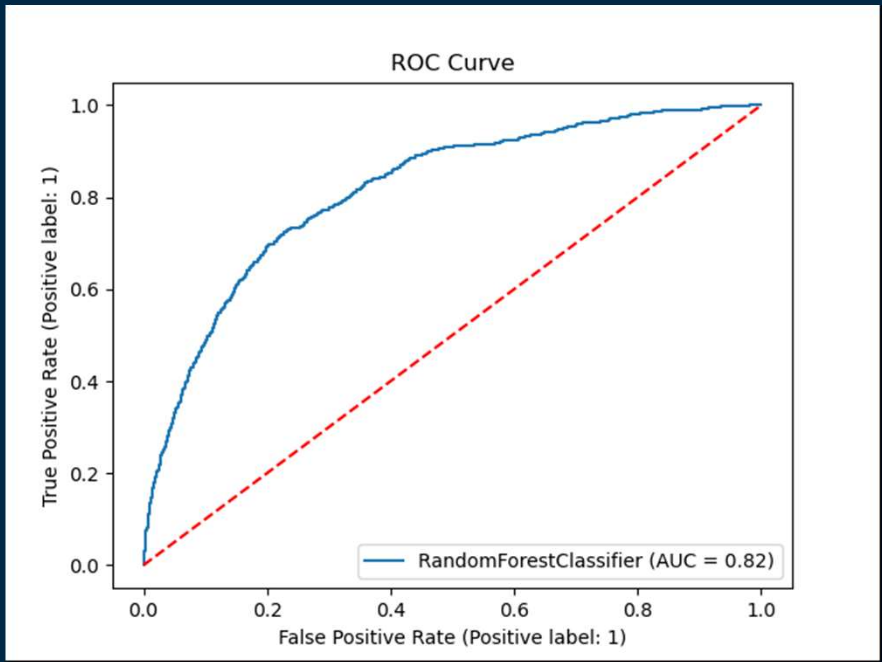
Hyperparameter Optimization

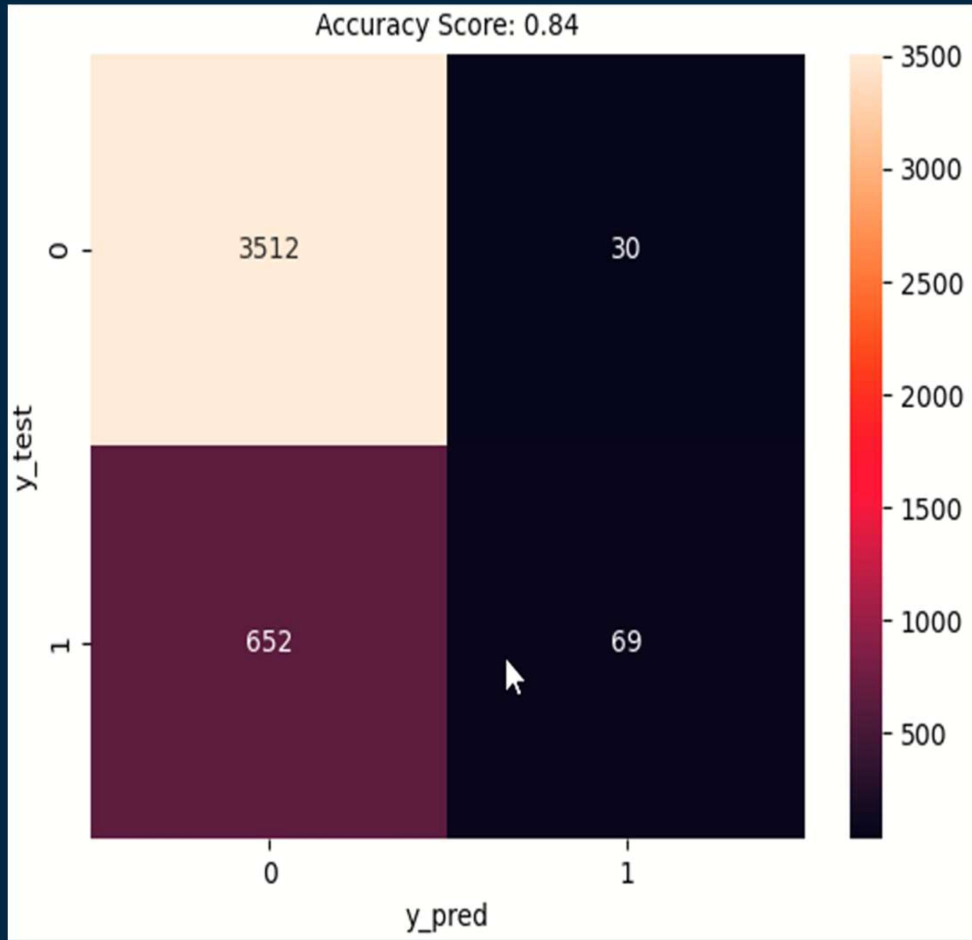
```
rf_val_params = [["max_depth", [5, 8, 15, 20, 30, None]],
                 ["max_features", [3, 5, 7, "auto"]],
                 ["min_samples_split", [2, 5, 8, 15, 20,40]],
                 ["n_estimators", [10, 50, 100, 200, 500,750]]]
gbm_val_params = [["max_depth", [5, 8, 15, 20, None]],
                 ['min_samples_leaf',[2,4,6,10]],
                 ["max_features", [3,7,15,24,30]],
                 ["min_samples_split", [2, 5, 8, 10]],
                 ["n_estimators", [100,500,1000,15000]],
                ['subsample',[0.5,0.7,1]]]
xgb_val_params=[["max_depth",[5,10,15,20]],
                ['gamma',[0.1,0.05,0.03,0.01]],
                ['min_child_weight',[0.1,0.2,0.5,1]],
                ['max_delta_step',[2,5,10]],
                ['subsample',[0.5,0.8,1]],
                ['colsample_bytree',[0.3,0.5,0.7,1]]]
lgb_val_params=[['num_iterations',[100,1000,10000]],
                ['bagging_fraction',[0.5,0.7,1]],
                ["max_depth",[5,10,13,20]],
                ['num_leaves',[2,5,10,15,20]],
                ['bagging_freq',[3,5,7]],
                ['min_data_in_leaf',[2,4,6]],
                ['max_bin',[20,30,34,40]],
                ['feature_fraction',[0.5,0.7,1]]]

catboost_val_params=[["iterations",[100,1000,2000,2500]],
                     ["depth",[2,5,7,10]]]
```

In the light of base models and newly created variables, the hyperparameters of the first 5 models that gave the best AUC value were extracted.

Random Forest Model

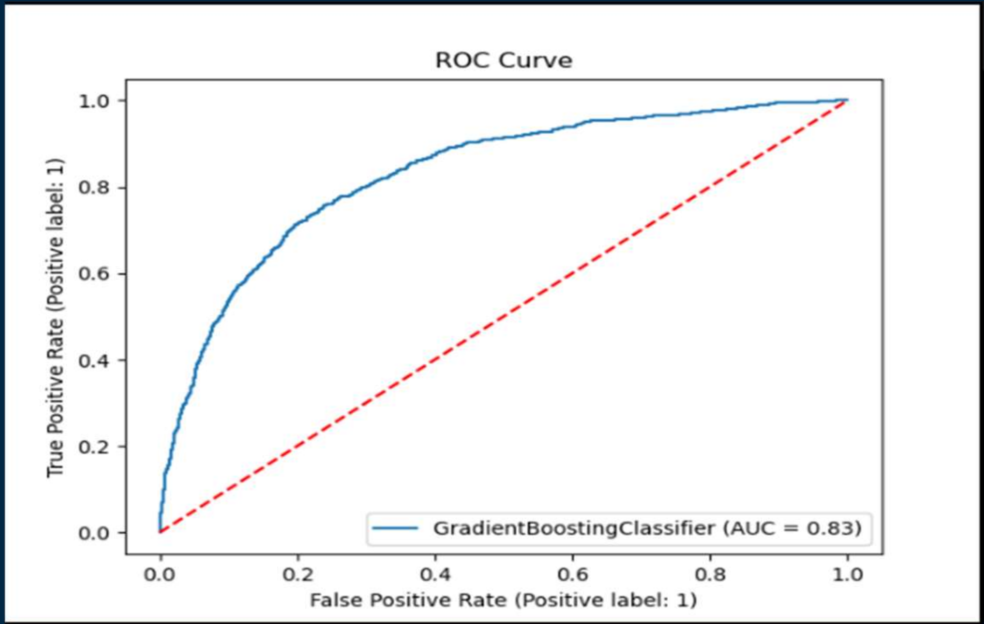
```
#########################  RF   #########################
# KFold cross validate
# roc_auc_mean :0.7867396179220114
# Stratifield Nfold cross val score
# roc_auc_mean:0.8110066161054071
```

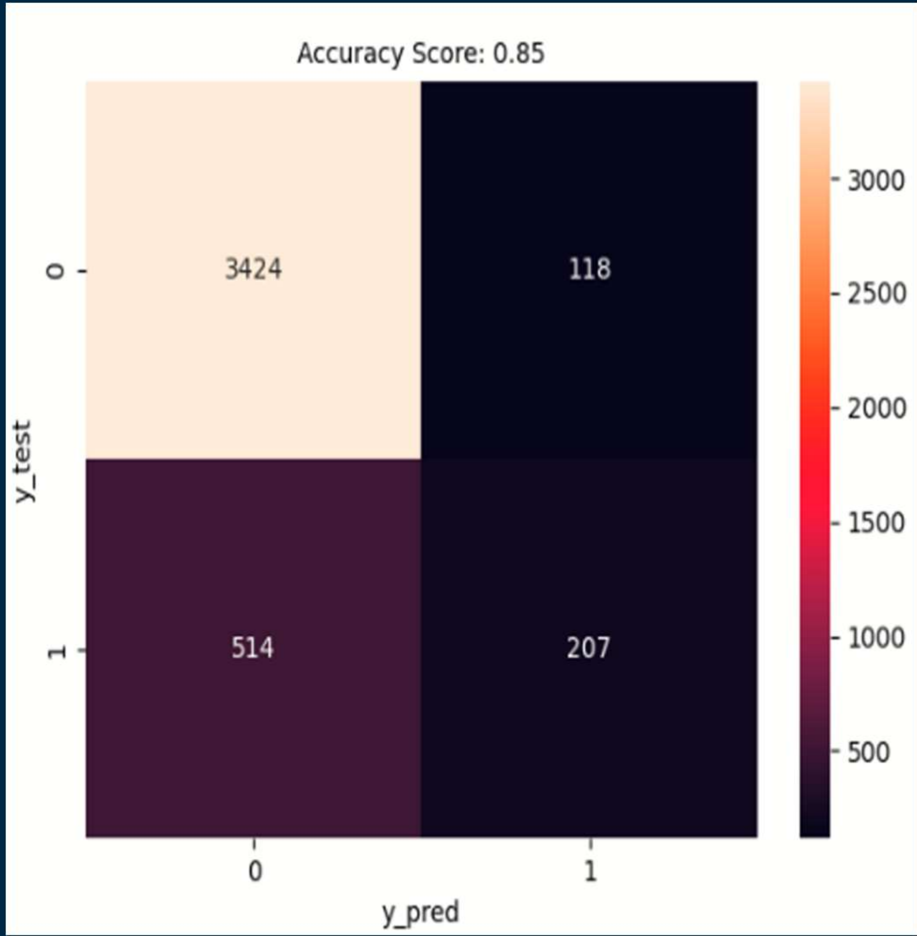




Gradient Boosting Model

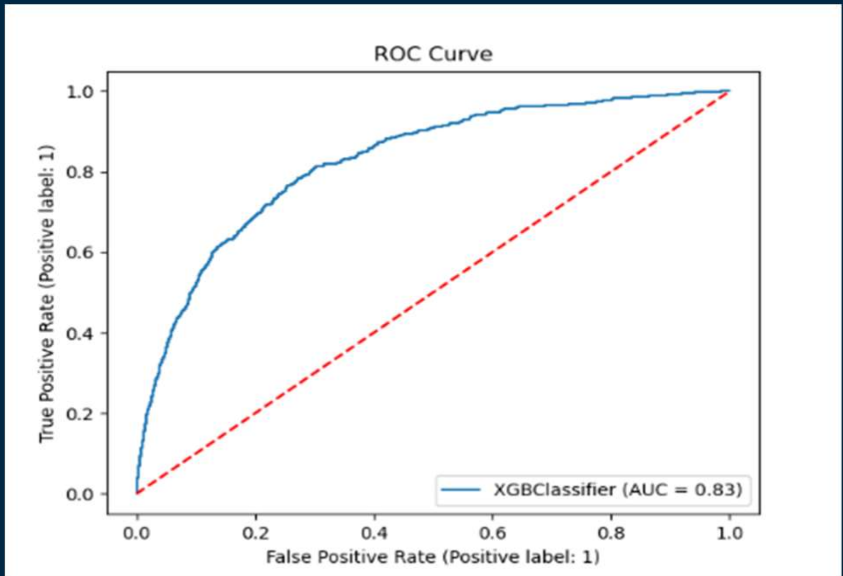
```
##########################   GBM   ################
# KFold cross validate
# roc_auc_mean :0.7854432498780366
# Stratifield Nfold cross val score
# roc_auc_mean:0.819024272820059
```

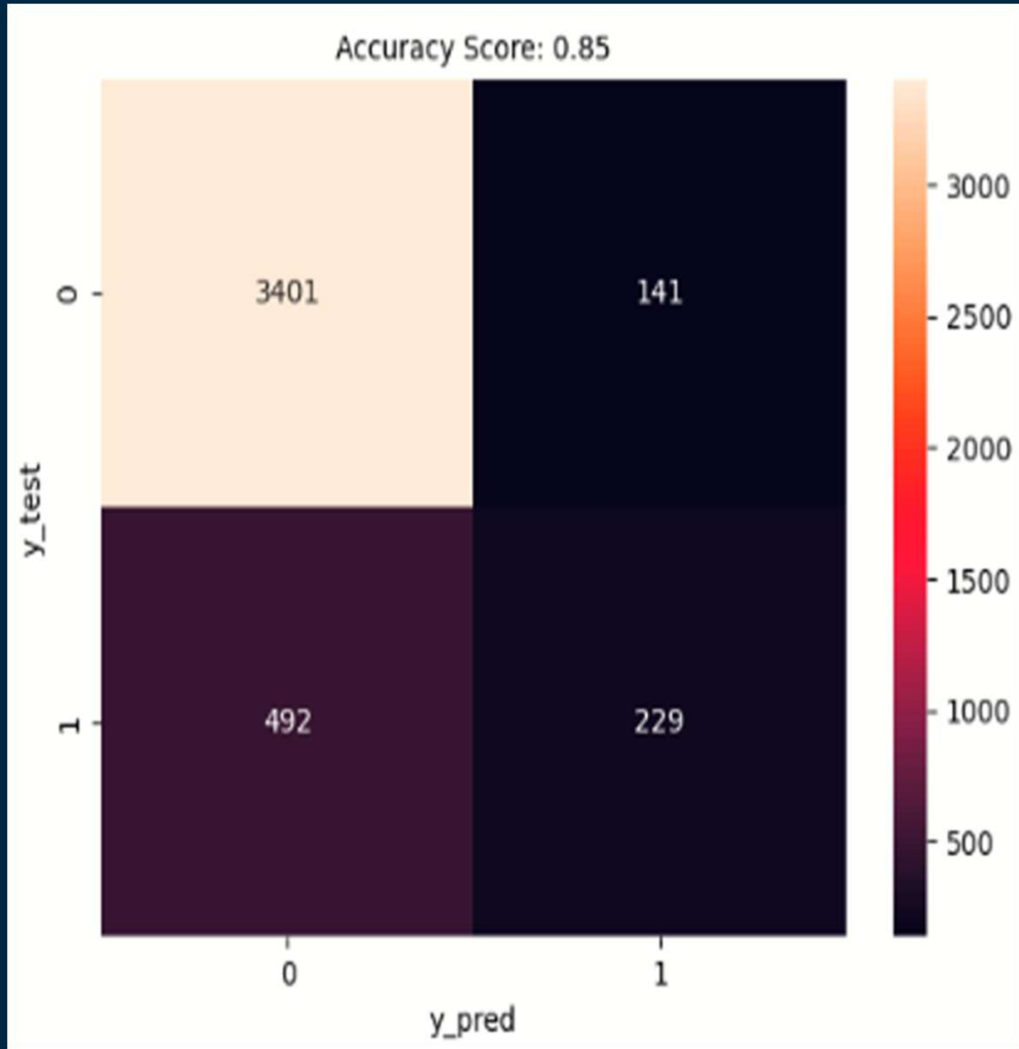




XGBoost Model

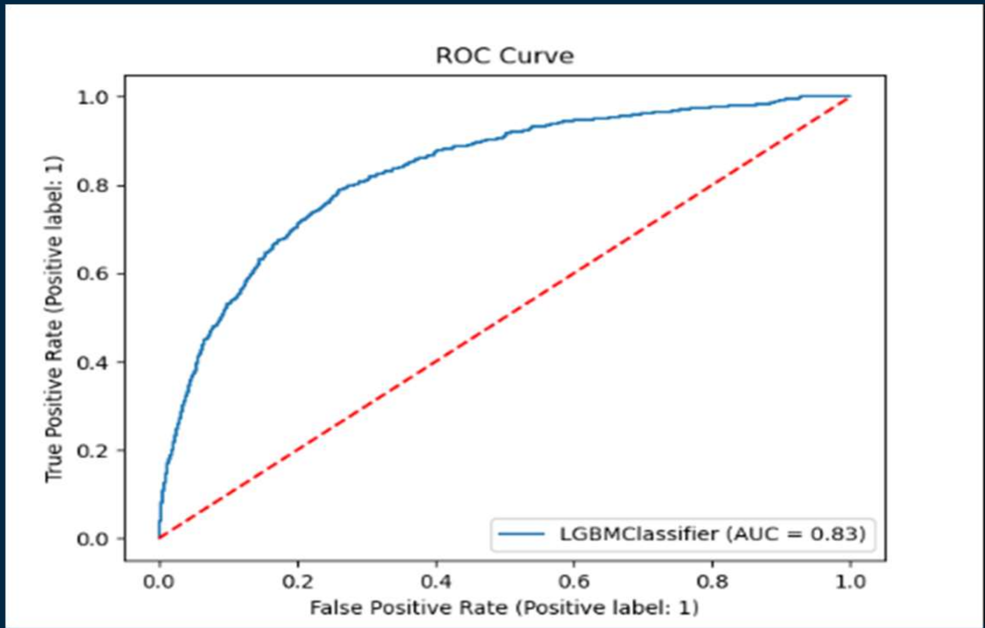
```
# [2496]    train-auc:0.99987   valid-auc:0.82485
# KFold cross validate
# roc_auc_mean :train-auc-mean   0.972
# train-auc-std    0.001
# test-auc-mean    0.845
# test-auc-std     0.012
# dtype: float64
# Stratifield Nfold cross val score
# roc_auc_mean:0.8100946031251185
```

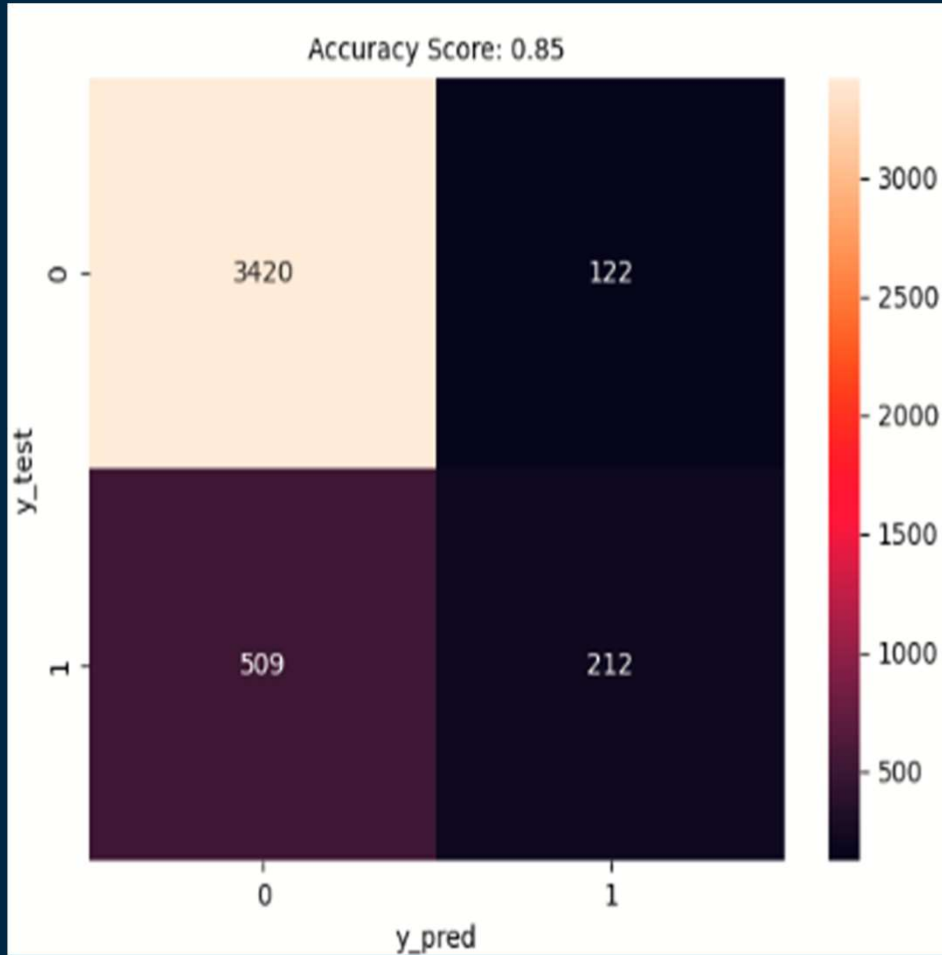




Light GBM Model

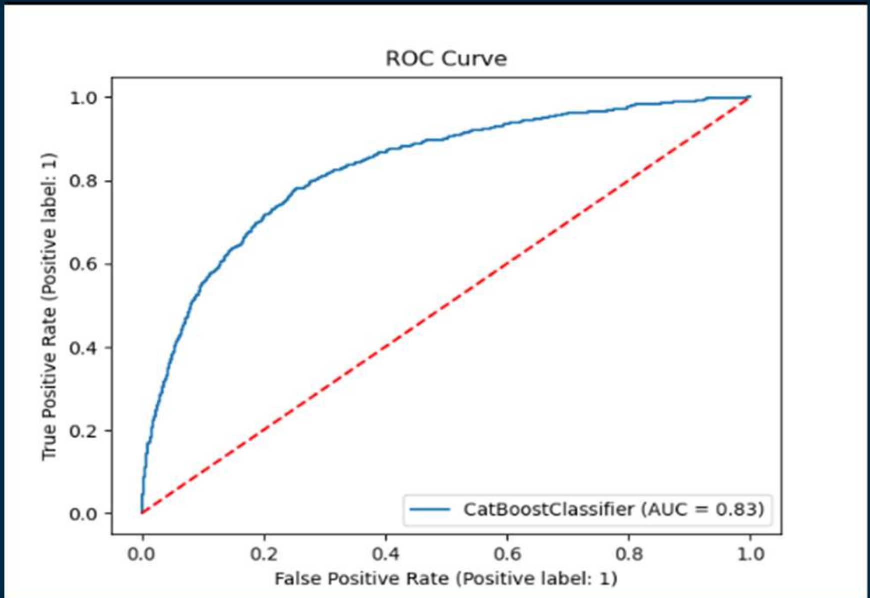
```
# '########################  LIGHTGBM   ########################
# [1200]    training's auc: 0.999991    valid_1's auc: 0.829049
# KFold cross validate
# roc_auc_mean :0.8376368086117013
# Stratifield Nfold cross val score
# roc_auc_mean:0.8219368198631892
```

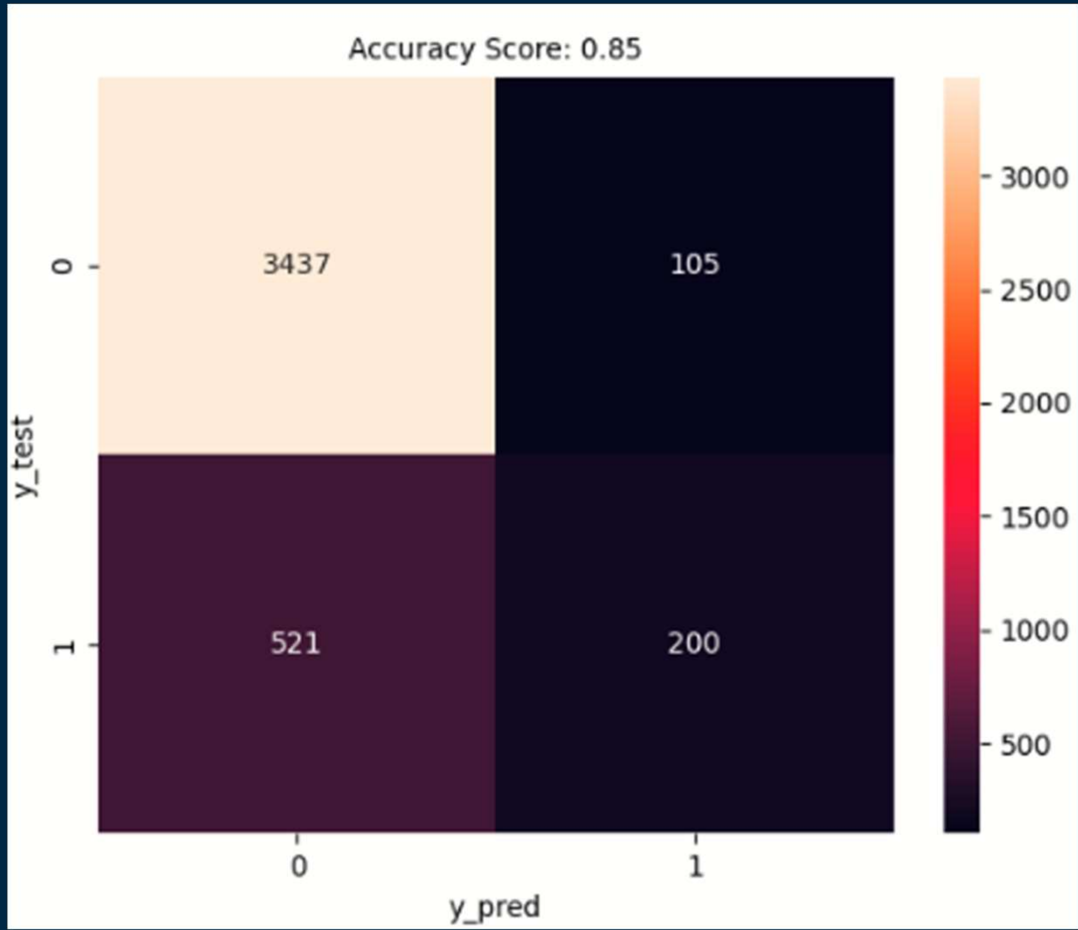




CatBoost Model

```
##########################  CATBOOST   ############
# KFold cross validate
# roc_auc_mean :0.7954677408240597
# Stratifield Nfold cross val score
# roc_auc_mean:0.8190237259794351
```





Stacking Ensemble Learning

As a result of hyper parameter optimization, a vote is made between the algorithms that give the best AUC value.

A vote is made according to each model, and as a result, the voting value is 82%. So we got an average result close to our models.

```
voting_auc_mean:0.8206176188240016
```

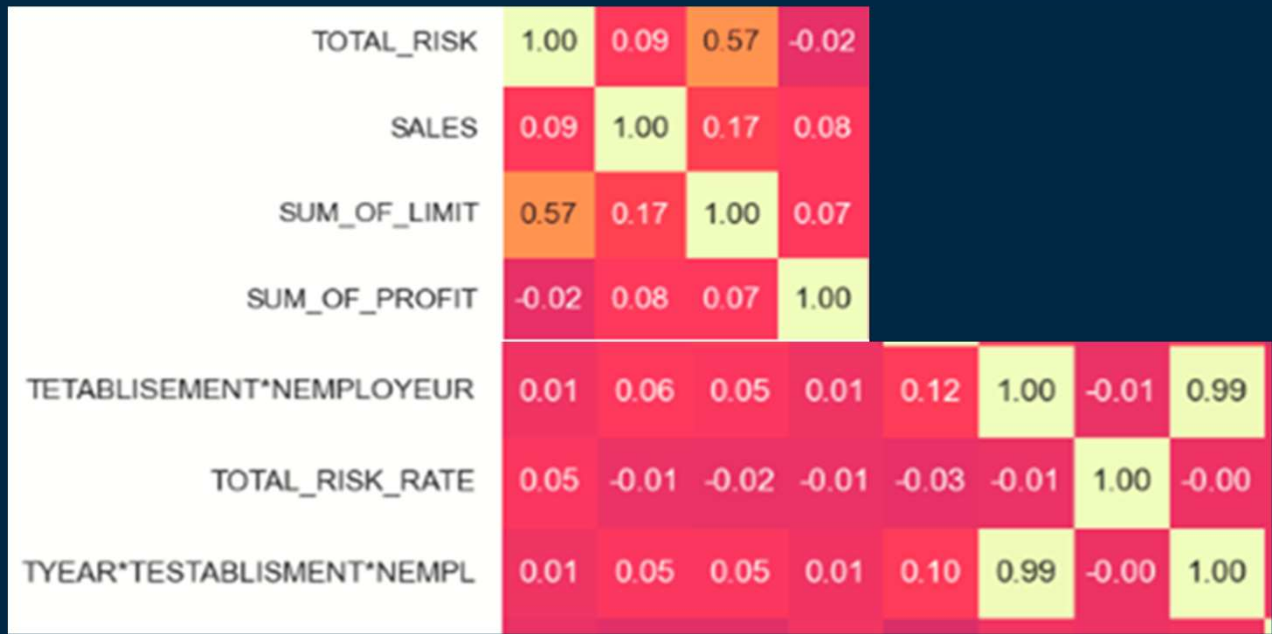
FEAUTURE SELECTION

Model complexity removal

04

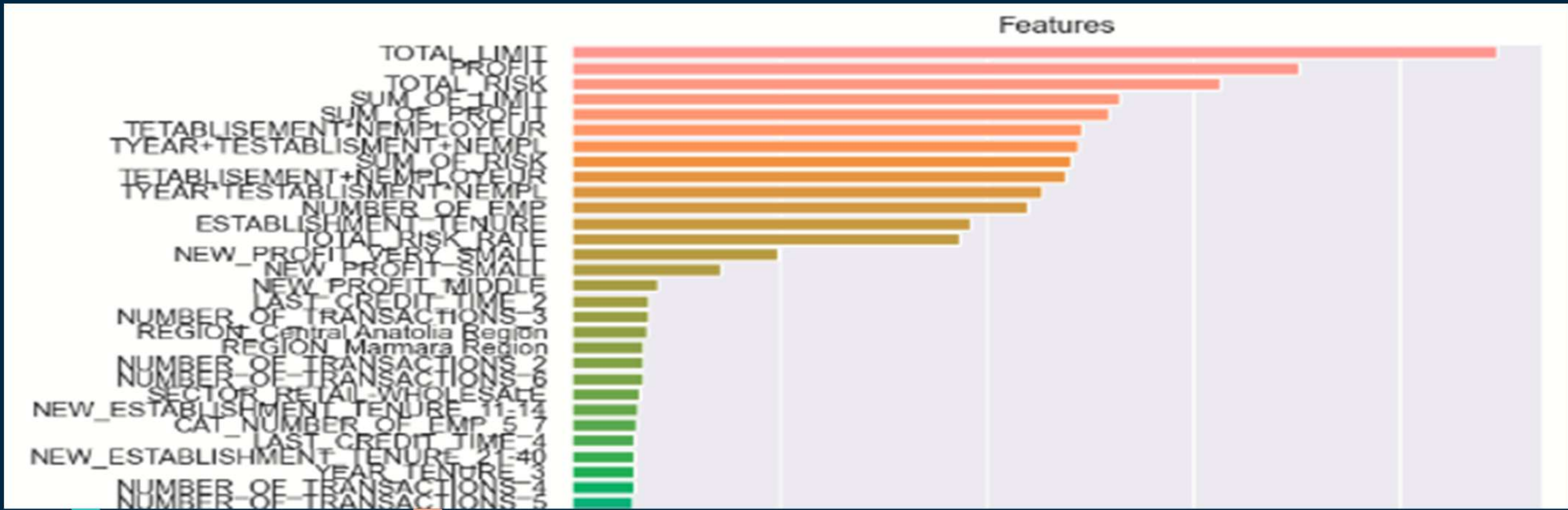
CORRELATION MATRIX

```
# TETABLISEMENT+NEMPLOYEUR    0.995
# TYEAR+TESTABLISMENT+NEMPL   0.993
# TETABLISEMENT*NEMPLOYEUR    0.938
# TYEAR*TESTABLISMENT*NEMPL   0.923
# LAST_CREDIT_TIME_8          0.081
# Name: NUMBER_OF_EMP, dtype: float64
# ################################################
# TOTAL_LIMIT                      0.890
# SUM_OF_RISK                      0.760
# SUM_OF_LIMIT                     0.573
# SALES                            0.094
# NEW_YEAR_SECTOR_YOUNG_SERVICES   0.064
# Name: TOTAL_RISK, dtype: float64

# ################################################
# SUM_OF_PROFIT                          0.647
# NEW_PROFIT_MIDDLE                      0.302
# NEW_YEAR_PROFIT_YOUNG_LOW_PROFIT       0.221
# NEW_YEAR_PROFIT_MIDDLE_NORMAL_PROFIT   0.205
# NEW_PROFIT_SMALL                       0.172
# Name: PROFIT, dtype: float64
```



FEAUTURE IMPORTANCE

RF FEAUTURE IMPORTANCE

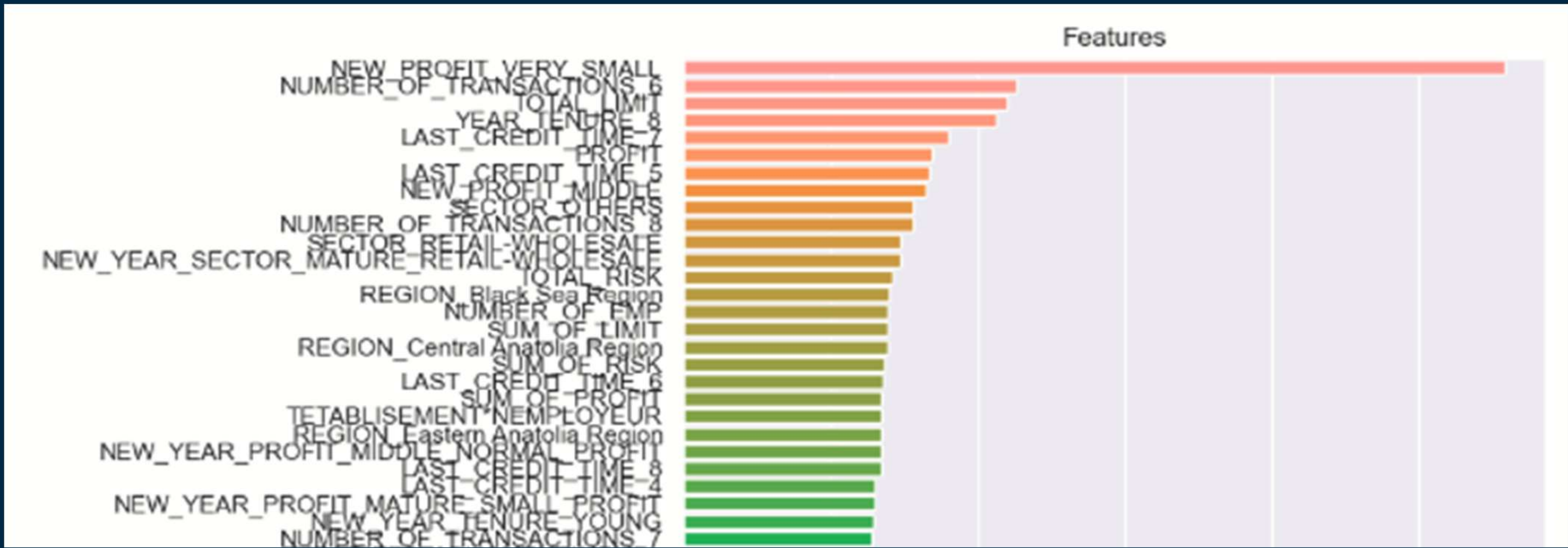


FEAUTURE IMPORTANCE

GBM FEAUTURE IMPORTANCE

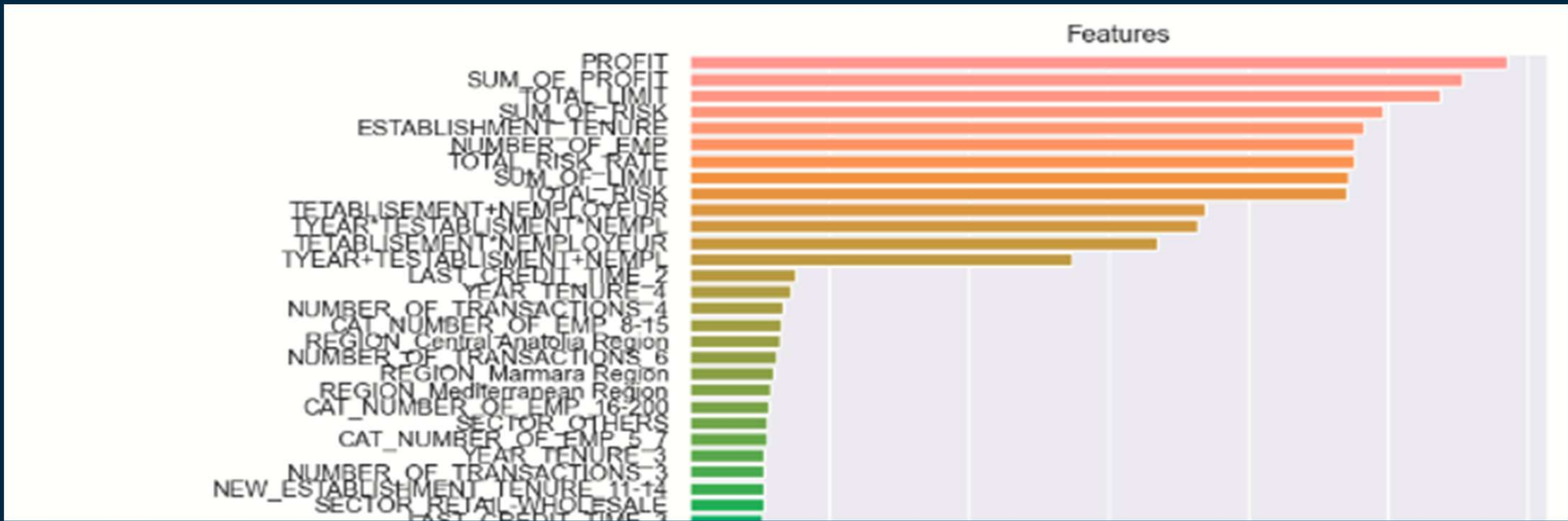
FEAUTURE IMPORTANCE

XGBOOST FEAUTURE IMPORTANCE



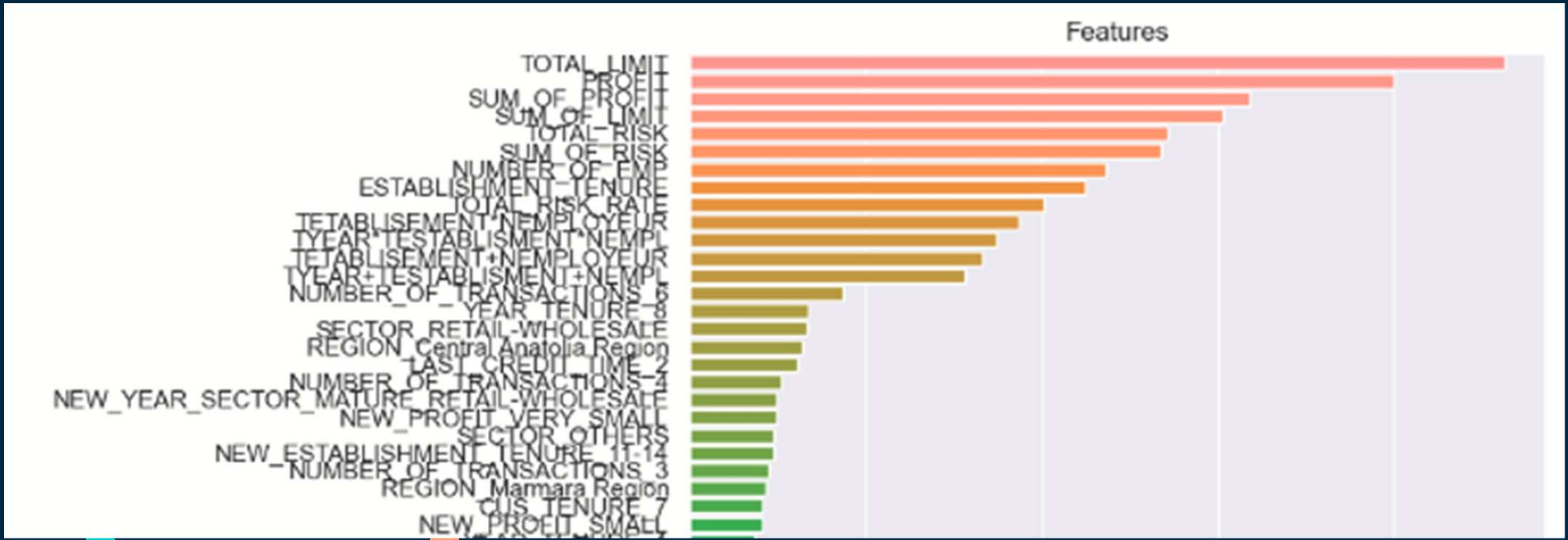
FEAUTURE IMPORTANCE

LİGHTGBM FEAUTURE IMPORTANCE



FEAUTURE IMPORTANCE

CATBOOST FEAUTURE IMPORTANCE



FEAUTURE IMPORTANCE

We will look at the average of feature importance of all models and make feature selection.

# NUMBER_OF_TRANSACTIONS_3	0.100
# CAT_NUMBER_OF_EMP_5_7	0.105
# CAT_NUMBER_OF_EMP_8-15	0.106
# NEW_YEAR_SECTOR_MATURE_RETAIL-WHOLESALE	0.107
# NEW_PROFIT_SMALL	0.110
# LAST_CREDIT_TIME_5	0.111
# NUMBER_OF_TRANSACTIONS_4	0.111
# SECTOR_OTHERS	0.112
# LAST_CREDIT_TIME_2	0.114
# SECTOR_RETAIL-WHOLESALE	0.121
# REGION_Central Anatolia Region	0.122
# YEAR_TENURE_8	0.151
# NUMBER_OF_TRANSACTIONS_6	0.158
# NEW_PROFIT_VERY_SMALL	0.294
# TYEAR+TESTABLISMENT+NEMPL	0.369
# TETABLISEMENT+NEMPLOYEUR	0.393
# TETABLISEMENT*NEMPLOYEUR	0.419
# NUMBER_OF_EMP	0.421
# TYEAR*TESTABLISMENT*NEMPL	0.421
# TOTAL_RISK_RATE	0.435
# ESTABLISHMENT_TENURE	0.464
# SUM_OF_RISK	0.528
# SUM_OF_LIMIT	0.561
# SUM_OF_PROFIT	0.564
# TOTAL_RISK	0.575
# PROFIT	0.749
# TOTAL_LIMIT	0.862

→ By looking at the correlations and feature importance, some of the features with high correlation and low importance were removed from the data.

Comparison of Models

A new base model was established with the remaining variables after the values extracted from the data.

```
base_models()
# roc_auc: 0.7579 (LR)
# roc_auc: 0.7087 (KNN)
# roc_auc: 0.7272 (SVC)
# roc_auc: 0.6126 (CART)
# roc_auc: 0.8022 (RF)
# roc_auc: 0.7664 (Adaboost)
# roc_auc: 0.7832 (GBM)
# roc_auc: 0.7923 (LightGBM)
# roc_auc: 0.8024 (CatBoost)
# roc_auc: 0.7902 (XGBoost)
```

Comparison of Models

Random Forest

```
# Random Forests
########################################
rf_tuned = RandomForest(roc_curve=True)
#########################  RF   ########
# KFold cross validate
# roc_auc_mean :0.7683454622688424
# Stratifield Nfold cross val score
# roc_auc_mean:0.7972859645814416
```

Gradient Boosting

```
#GBM
########################################
gbm_tuned=GradientBoosting(roc_curve=True)

# #########################  GBM   ########
# KFold cross validate
# roc_auc_mean :0.7657863404071507
# Stratifield Nfold cross val score
# roc_auc_mean:0.804952585042195
```

Comparison of Models

XGBoost

```
# XGBoost
###############################################
xgb_tuned=XGBoost(roc_curve=True)

#########################  XGBOOST   #############
# [0]   train-auc:0.70950   valid-auc:0.66591
# [200] train-auc:0.88132   valid-auc:0.77902
# [400] train-auc:0.91654   valid-auc:0.78748
# [600] train-auc:0.94372   valid-auc:0.79335
# [800] train-auc:0.96205   valid-auc:0.79788
# [1000]    train-auc:0.97495   valid-auc:0.80071
# [1200]    train-auc:0.98285   valid-auc:0.80273
# [1400]    train-auc:0.98844   valid-auc:0.80418
# [1600]    train-auc:0.99214   valid-auc:0.80485
# [1800]    train-auc:0.99482   valid-auc:0.80525
# [1844]    train-auc:0.99522   valid-auc:0.80531
# KFold cross validate
# roc_auc_mean :train-auc-mean   0.962
# train-auc-std    0.001
# test-auc-mean    0.829
# test-auc-std     0.016
# dtype: float64
# Stratifield Nfold cross val score
# roc_auc_mean:0.8004123898945791
```

Comparison of Models

LightGBM

```
# LightGBM
#################################################################
lgb_tuned=LightGBM(roc_curve=True)

# [1200]    training's auc: 0.999726    valid_1's auc: 0.804455
# KFold cross validate
# roc_auc_mean :0.823164547024853 8
# Stratifield Nfold cross val score
# roc_auc_mean:0.7962825931848518
```

Comparison of Models

CatBoost

```
# CatBoostClassifier
#########################################
catboost_tuned=CatBoost(roc_curve=True)

# #########################  CATBOOST
# KFold cross validate
# roc_auc_mean :0.7743598534305984
# Stratifield Nfold cross val score
# roc_auc_mean:0.803313898993026
```

Stacking Ensemble Learning

As a result of all models, we do the voting again and we get a result of 80%.

```
# Stacking & Ensemble Learning
#######################################################################################
def voting2():
    print("#####################    STACKING & ENSEMBLE LEARNING    #####################
    voting_clf = VotingClassifier(estimators=[('LGB', lgb_tuned),
                                              ('XGB', xgb_tuned)], voting='soft')
    voting_clf.fit(X_train, y_train)
    cv_results = cross_validate(voting_clf, X_test, y_test, cv=10, scoring="roc_auc")
    print(f"voting_auc_mean:{cv_results['test_score'].mean()}")
voting()

# voting_auc_mean:0.8084360613540961
```

FİNAL MODEL

Deciding on the best model

05

Selecting the Final Model

Among the models, LIGHTGBM and XGBOOST give the best results, so you can choose between the two.
We chose this model as the final model because XGBOOST was a little more successful than other models.

→ Finally, we pipelined all the operations and turned them into a pickle file.

RESULTS

	AUC Score	Explanation
Base Model	%78	The result we obtained after the first models were established after the EDA
Filling Missing Values & Feature Engineering	%84	Best result after filling in missing values and creating new variables
Feauture Selection	%82	The result obtained after the selection of the variables that have the best effect on the target variable
Best Model	%84	The result we got with XGBoost, which is the best model as a result of hyper parameter optimization

FİNAL AUC SCORE %84

As a result of our final model, XGBOOST, we can predict the target audience 84% accurately according to the AUC score.

FUTURE PLANS

In the next stage, we will try to increase the F1 score on unbalanced data.

F1 SCORE

%30

Our F1 score was low due to unstable data problem. For this reason, we aim to increase the F1 score.

AUC SCORE

%84

As a result of the best model, our AUC score is 84%.

THANKS

BAHA ULUĞ
linkedin.com/in/baha-ulug

ELİF ÖZDEMİR
linkedin.com/in/elif-özdemir-8892729b

ESRİN ERDEM
linkedin.com/in/esrin-erdem-3677161a4

İSMAİL HAKKI ÖZÇELİK
linkedin.com/in/ismail-ozcelik

ESRA BARBAROS
esrabarbaros@gmail.com