Bank Customers Churn Classification

Table of contents

01

Introduction

Introduction

In this work the purpose is to build a model that predicts if a customer will churn form the bank (or not) given various data points and information from historical data:

Goals:

- Predicting if a customer will leave or not.
- Banks will be able to predict the risk of the customer on whether they will leave or not.

02

Methodology

Methodology

Data Extraction

-Data from Kaggle
-10000 Row
-14 Columns

EDA

-Data Wrangling
-Data Cleaning
-Visualization

Feature Engineering

-Outliers
-Dummy Variables

Models

-Model Preprocessing
-Building Models

Tools

Jupyter Notebook

Pandas

Seaborn

Matplotlib

Sklearn

NumPy

Dabl

Data Split

20%

Test

20%

Validation

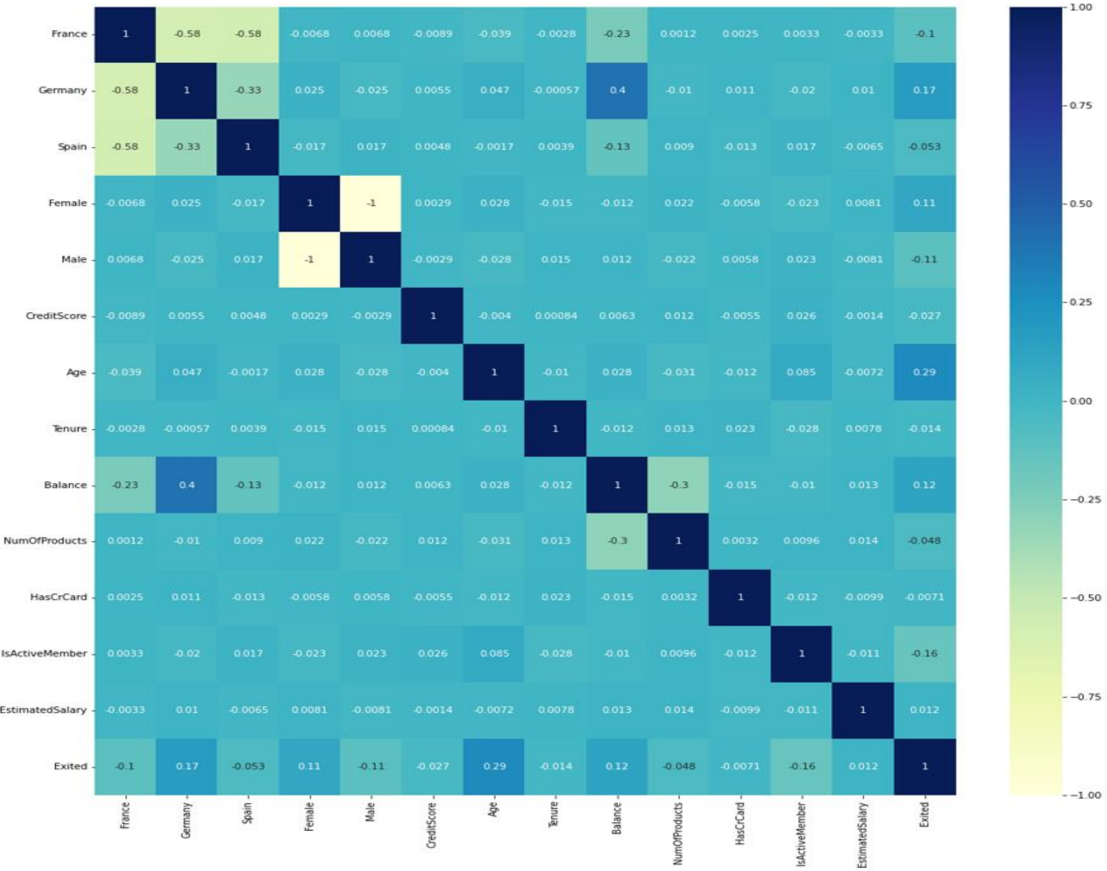
60%

Training

Exploratory Data Analysis

- Data Cleaning
- Same size after cleaning
- No extreme values
- Imbalanced data

Heatmap



Features that has big correlation with exiting the bank:

- Age (29%)
- Customers in Germany (17%)
- Female customers (11%)

03

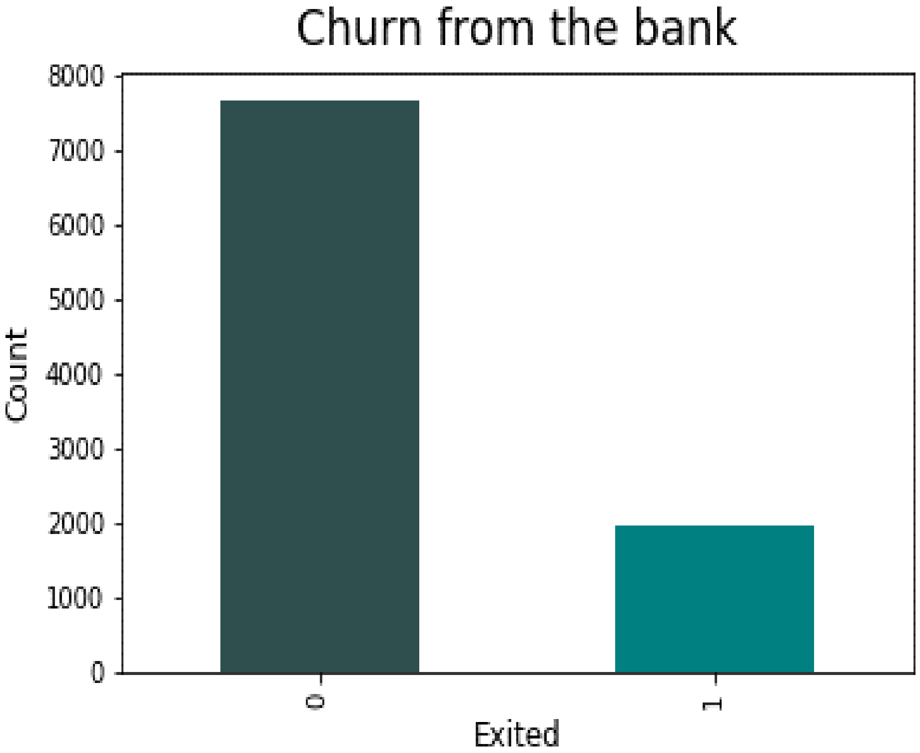
Data Preprocessing

Feature Engineering

Steps:

- Feature selection using LASSO and Recursive Feature Elimination
- Dummy variables
- Outliers

Data Imbalance

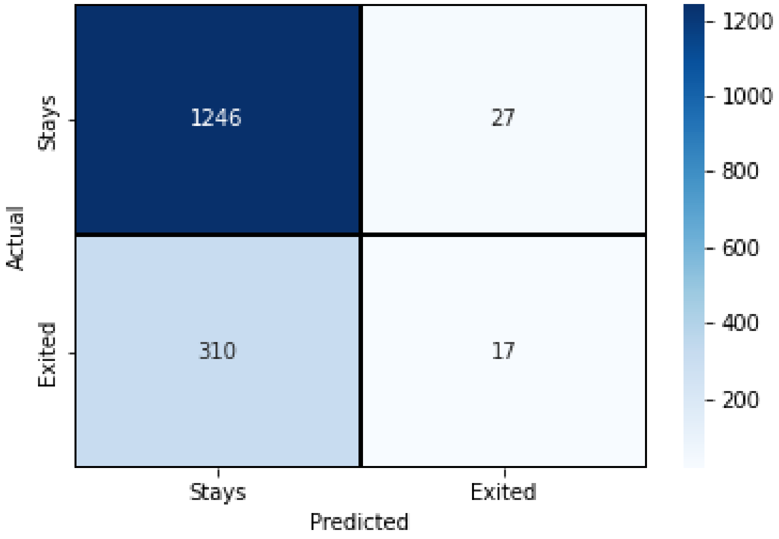


Solved using RandomOverSampler (ROS)

04

Models

Baseline Model: Logistic Regression



Score	Training	Validation
Accuracy	0.6659	0.6641
Precision	0.6636	0.6636
Recall	0.6800	0.6801
F1	0.6717	0.6718

DABL Modeling

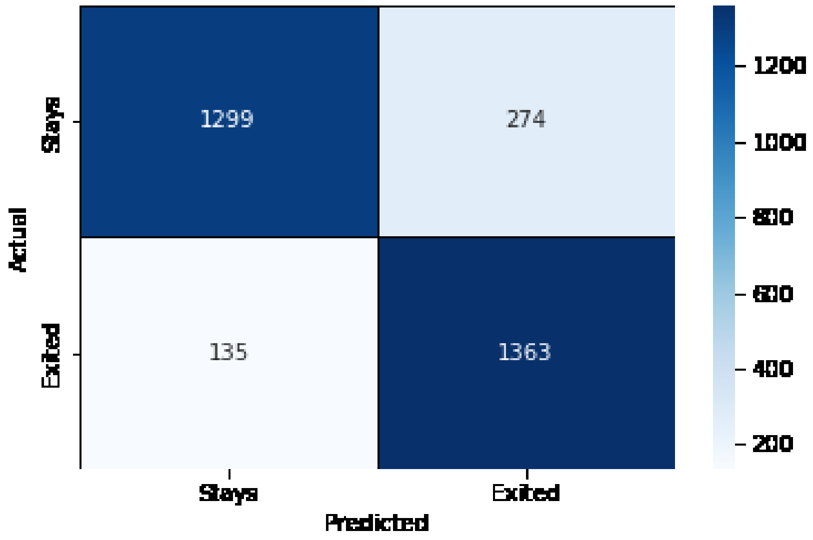
```
In [116]: survivor_classifier = dabl.SimpleClassifier(random_state=42).fit(X_train, y_train)
```

```
Running DummyClassifier()
accuracy: 0.502 average_precision: 0.498 roc_auc: 0.500 recall_macro: 0.500 f1_macro: 0.334
=== new best DummyClassifier() (using recall_macro):
accuracy: 0.502 average_precision: 0.498 roc_auc: 0.500 recall_macro: 0.500 f1_macro: 0.334

Running GaussianNB()
accuracy: 0.719 average_precision: 0.776 roc_auc: 0.792 recall_macro: 0.719 f1_macro: 0.719
=== new best GaussianNB() (using recall_macro):
accuracy: 0.719 average_precision: 0.776 roc_auc: 0.792 recall_macro: 0.719 f1_macro: 0.719

Running MultinomialNB()
accuracy: 0.637 average_precision: 0.687 roc_auc: 0.702 recall_macro: 0.637 f1_macro: 0.637
Running DecisionTreeClassifier(class_weight='balanced', max_depth=1)
accuracy: 0.700 average_precision: 0.632 roc_auc: 0.700 recall_macro: 0.700 f1_macro: 0.697
Running DecisionTreeClassifier(class_weight='balanced', max_depth=5)
accuracy: 0.766 average_precision: 0.819 roc_auc: 0.846 recall_macro: 0.766 f1_macro: 0.766
=== new best DecisionTreeClassifier(class_weight='balanced', max_depth=5) (using recall_macro):
accuracy: 0.766 average_precision: 0.819 roc_auc: 0.846 recall_macro: 0.766 f1_macro: 0.766

Running DecisionTreeClassifier(class_weight='balanced', min_impurity_decrease=0.01)
accuracy: 0.725 average_precision: 0.730 roc_auc: 0.779 recall_macro: 0.725 f1_macro: 0.723
Running LogisticRegression(C=0.1, class_weight='balanced', max_iter=1000)
accuracy: 0.725 average_precision: 0.765 roc_auc: 0.209 recall_macro: 0.725 f1_macro: 0.725
Running LogisticRegression(class_weight='balanced', max_iter=1000)
accuracy: 0.725 average_precision: 0.765 roc_auc: 0.209 recall_macro: 0.725 f1_macro: 0.725

Best model:
DecisionTreeClassifier(class_weight='balanced', max_depth=5)
Best Scores:
accuracy: 0.766 average_precision: 0.819 roc_auc: 0.846 recall_macro: 0.766 f1_macro: 0.766
```

Classification Models

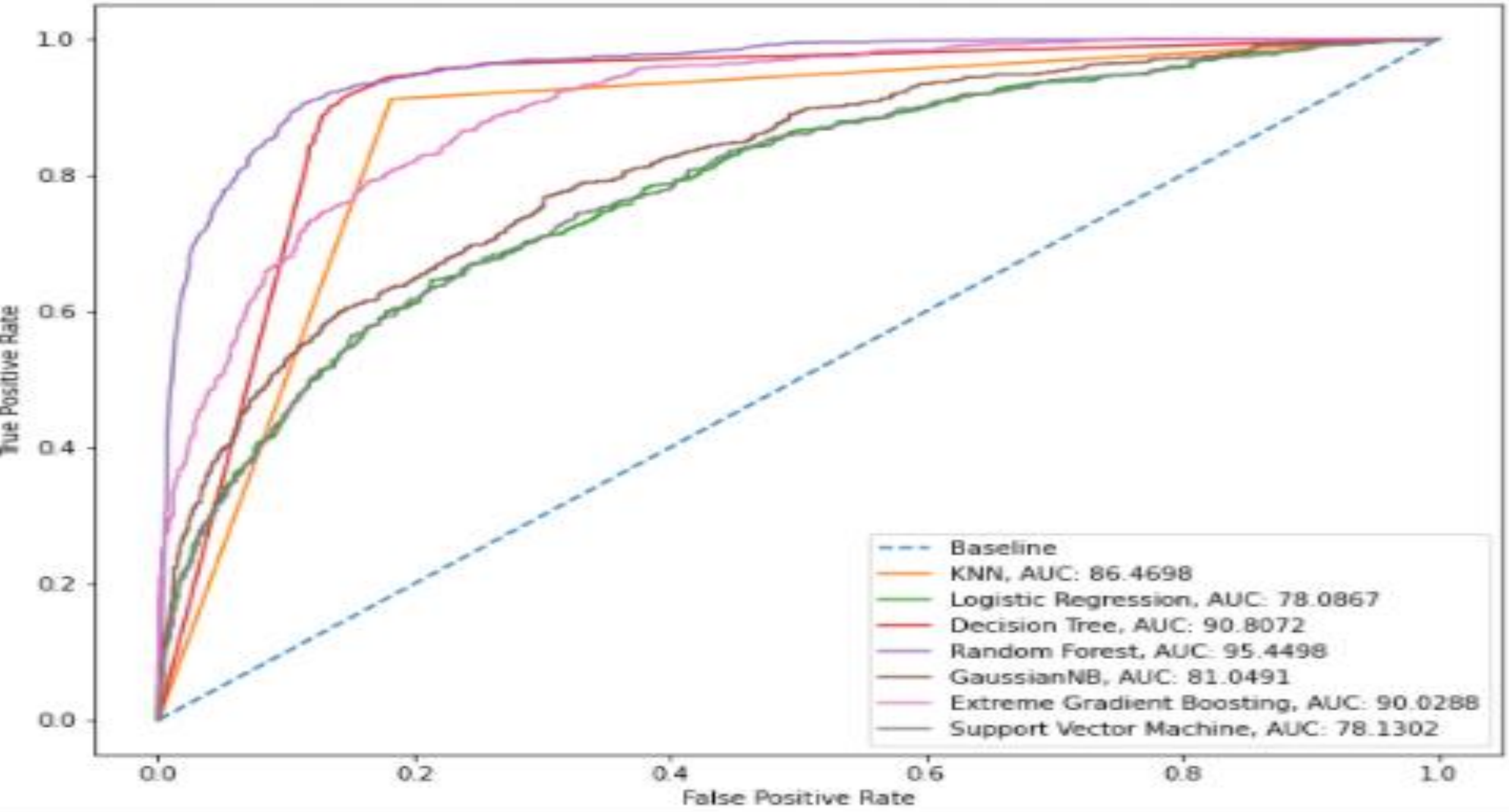
Model	Accuracy	F1-Score
Logistic Regression	0.7468	0.7461
KNN	0.8702	0.8781
Random Forest	0.8812	0.8881
Gaussian Naïve Bayes	0.7444	0.7411
SVM	0.7424	0.7372
Decision Tree	0.8751	0.8830
XGB	0.8221	0.8236

Ensemble: Max Voting Classifier

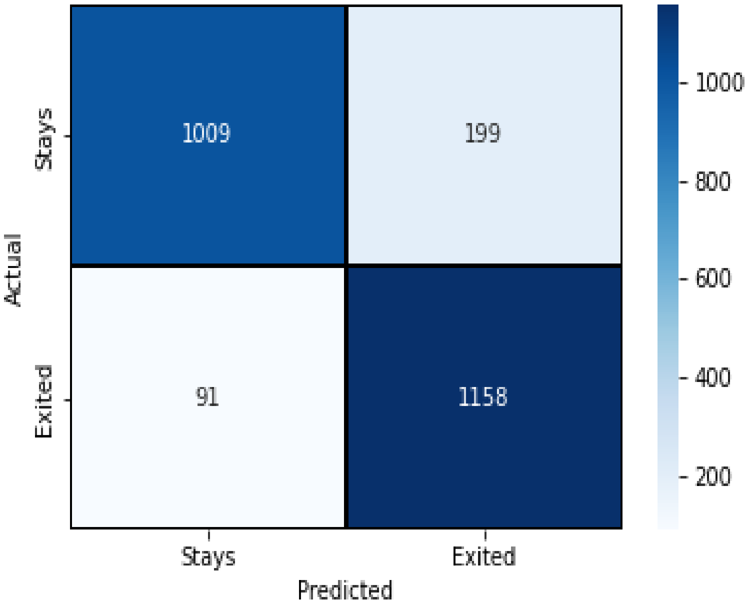


Training Accuracy: 0.9438
Testing Accuracy: 0.8668

ROC Curve Graph



Best Model: Random Forest



RF	Training	Valid	Testing
Accuracy	0.9619	0.8799	0.8828
Precision	0.9658	0.8497	0.8411
Recall	0.9580	0.9279	0.9366
F1	0.9619	0.8871	0.8863

Conclusions

- The best model to predict is Random Forest.
- Highest validation accuracy.

Thanks!

Do you have any questions?