Bank Customers Analysis

Table of contents

Introduction

Introduction

Dataset:
https://www.kaggle.com/datasets/radheshyamkollipara/bank-customer-churn

Tool:
- Python
- PBI
- Google slide

Target:
- Illustrate customers characters and suggest some business ideas accordingly.
- Predict customer trend.

Introduction

Shape: 10.000 rows & 17 columns

Customer Anthropologies (5 columns)	Customer Characteristics (8 columns)	Customer Behaviers (4 columns)
1. CustomerId 2. Surname 3. Gender 4. Age 5. Geography	1. Balance 2. EstimatedSalary 3. NumOfProducts 4. HasCrCard 5. IsActiveMember 6. Card Type 7. Tenure	1. CreditScore 2. Complain 3. Satisfaction Score 4. Point Earned 5. Exited

Introduction

```
cb.duplicated().sum()
✓ 0.0s

0
```

```
cb.isnull().sum()
✓ 0.0s

RowNumber            0
CustomerId           0
Surname              0
CreditScore          0
Geography            0
Gender               0
Age                  0
Tenure               0
Balance              0
NumOfProducts        0
HasCrCard            0
IsActiveMember       0
EstimatedSalary      0
Exited               0
Complain             0
Satisfaction Score   0
Card Type            0
Point Earned         0
dtype: int64
```

Data **does not have** duplicated rows
or
null values.

Checking columns individually is **clean**.

Moving on **Illustration**.

```
cb.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 18 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   RowNumber           10000 non-null  int64  
 1   CustomerId          10000 non-null  int64  
 2   Surname             10000 non-null  object 
 3   CreditScore         10000 non-null  int64  
 4   Geography           10000 non-null  object 
 5   Gender              10000 non-null  object 
 6   Age                 10000 non-null  int64  
 7   Tenure              10000 non-null  int64  
 8   Balance             10000 non-null  float64
 9   NumOfProducts       10000 non-null  int64  
 10  HasCrCard           10000 non-null  int64  
 11  IsActiveMember      10000 non-null  int64  
 12  EstimatedSalary     10000 non-null  float64
 13  Exited              10000 non-null  int64  
 14  Complain            10000 non-null  int64  
 15  Satisfaction Score  10000 non-null  int64  
 16  Card Type           10000 non-null  object 
 17  Point Earned        10000 non-null  int64  
dtypes: float64(2), int64(12), object(4)
memory usage: 1.4+ MB
```

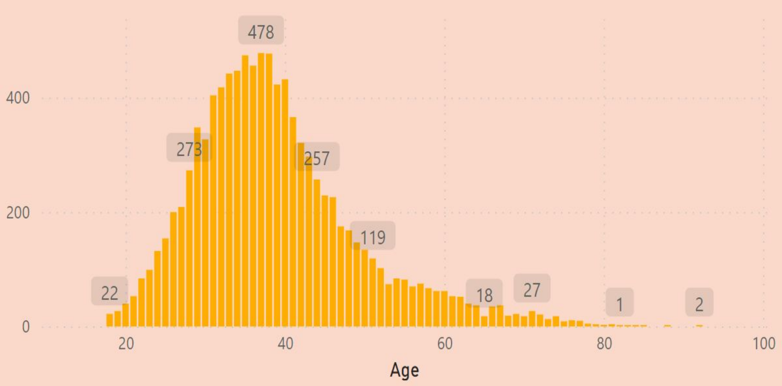
Analysis

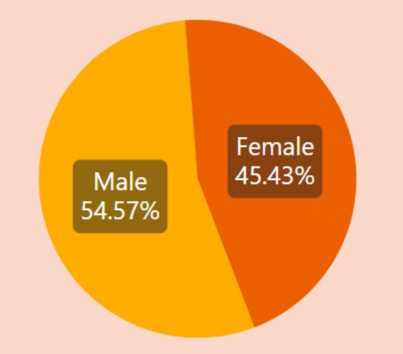
Overview

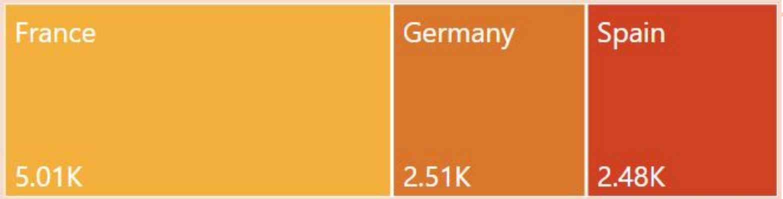
Dataset including
10.000 customers.

10.000
Customers

Customers are mainly
from **30 to 45 years old.**







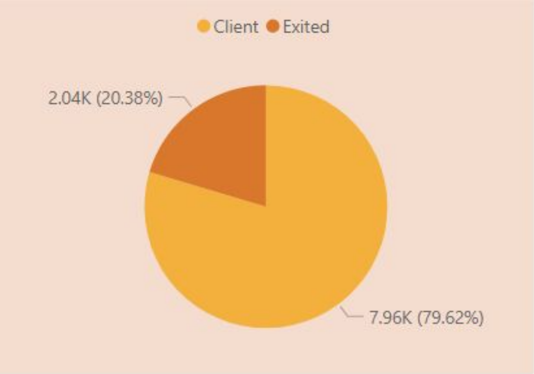
Half of customers from
France. Others come
from **Germany** and
Spain.

54% of them are
male.

Client comparision

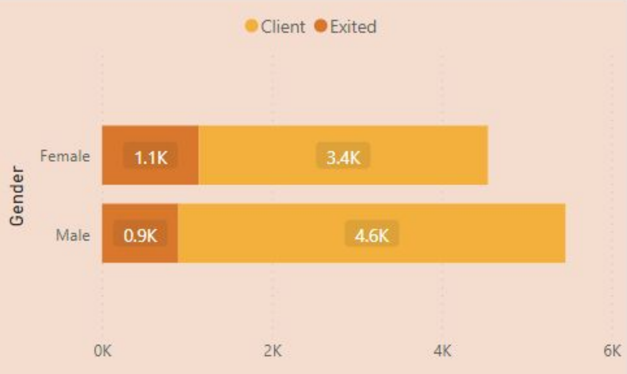
Between current and lost clients.

Over 2.000 customers had left, it contains of **20% percents** of total customers.



Most customers are **Adults.** Number of **Senior loyal customers to their loss** is quite the **same**.

Banks have **more Males** customer than Females. But, **Females lost rate is higher.**



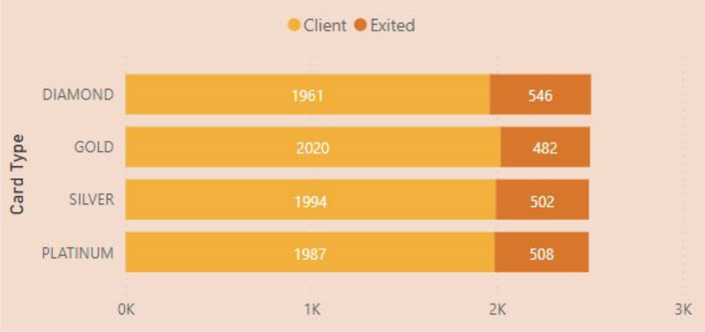
One thirds of German customer **left** the bank.

Client comparision

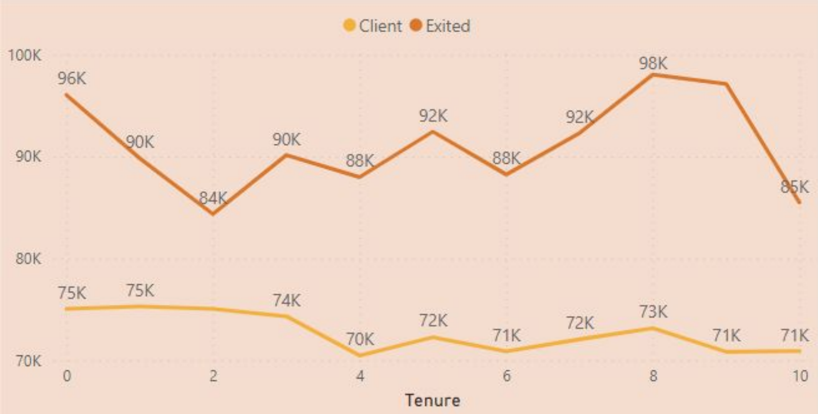
Between current and lost clients.

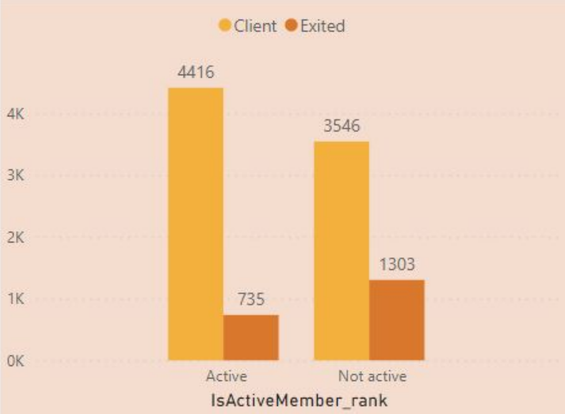
The fewer the products **higher the customers** and **higher churn customers.**

Number of customer remains uniform across all categories.



Average **balance** of **Loyal customer is lower** than Lost customer.



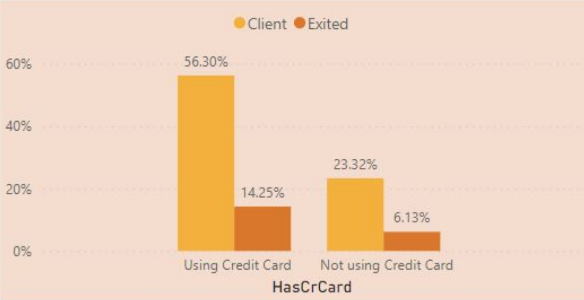


Not active customer are more likely to leave**.**

Client comparision

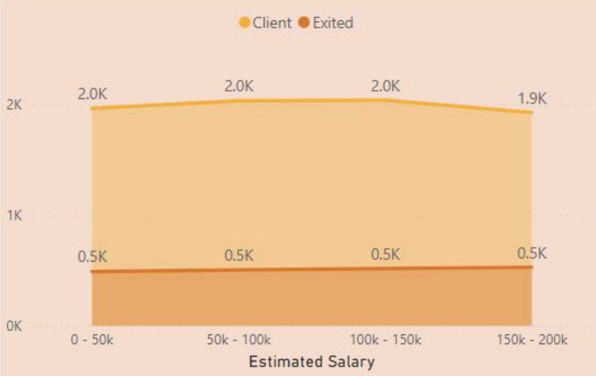
Between current and lost clients.

Numbers of customers in each satisfaction score are **similar**.



Customer using Credit Card has **higher leaving rate** than customer who not own credit card.

Lost clients has much **lower salary range** than current clients.



Credit score range of **600 - 700** has the **highest current customers** also has **highest lost customers** rate.

- Customers **mostly adult** from 30 to 45 years old, **churn rate** of this adult also **really high of 66%.**
- **Half** of the customer are **Frances**. German account of 25% however their **leaving rate nearly 40%.**
- There are **more male than female** but **leaving rate of female is higher**.
- Most customers use 1 to 2 products. **Churn rate of 1 product user is higher than the rest.**
- The amount of customers in each **Card Types pretty corresponding**, Gold customers slightly higher.
- Average **balance** of Loyal customers much **lower** than Lost customers, but their **salary** is **higher**.
- Customers **using Credit Card** have **higher leaving rate** than customers who not own credit card.

  Credit score from 600 to 700 account 30% of customers highest lost customers rate of 7%.
- **Satisfaction score does not effect** bank churn rate.

Prediction

Prepareation

Encoding

encoding

```python
for i in cb.columns:
    print(i)
    print(cb[i].unique())
    print('---------')

# there are 3 columns need to encode

# 'Geography': 3 unique
# 'Gender' : 2 unique
# 'Card Type' : 4 unique
```

✓ 0.0s

Outputs are collapsed ...

```python
# encode 3 columns have string unique:

dictionary_gender = {'Male' : 0, 'Female' : 1}
cb['Gender_encode'] = cb.Gender.map(dictionary_gender)

onehot_Geography = pd.get_dummies(cb['Geography'], prefix = 'Geography').astype(int)
onehot_Card_type = pd.get_dummies(cb['Card Type'], prefix = 'Card Type').astype(int)
```

✓ 0.0s

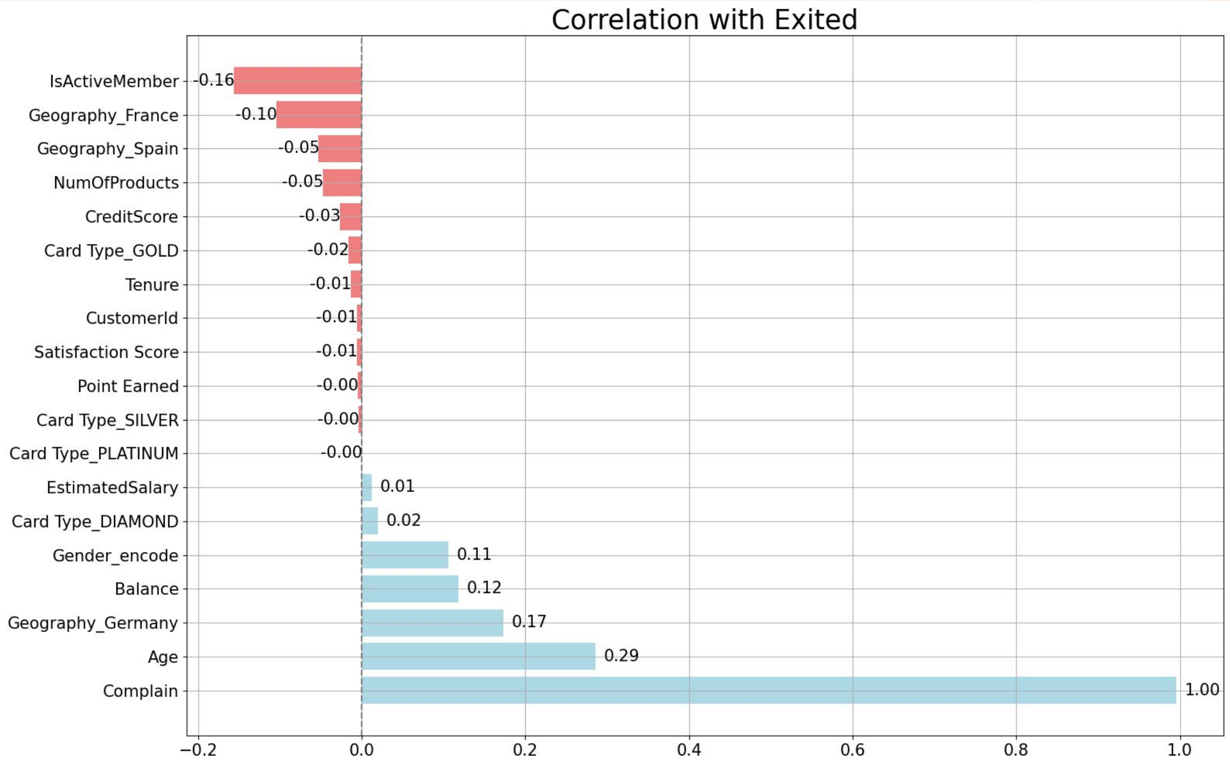
There are **3 columns** have string values and need to encode.

- Column **'Gender'** applied map dictionary method.
- Columns **'Geography'** and **'Card_type'** applied onehot method.

X, y defining

Complain has **absolute correlation. Customers who make complain more likely leave the bank.** Current Clients only make **10 complains/ 7962 clients.** In contrast, **2034 complains/ 2038 clients** who left the bank. -> **Not use.**

As showed, **correlation rate around -0.16 to 0.29.**

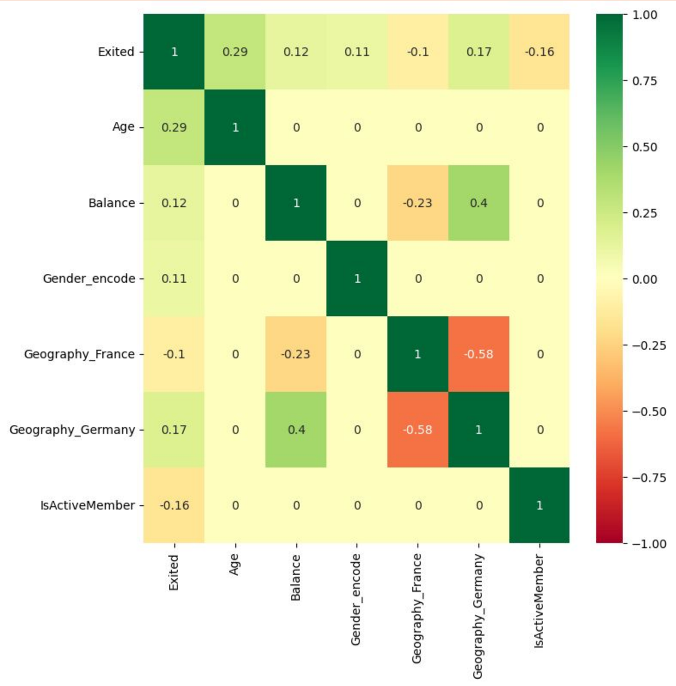


X, y defining

Checking correlation:

After **filter out** all of the columns have **high correlation columns** with columns **‘Exited’** which is **below -0.1** and **higher than 0.1,** then **check** their **correlation with each other.**

The result looks good. Continue.



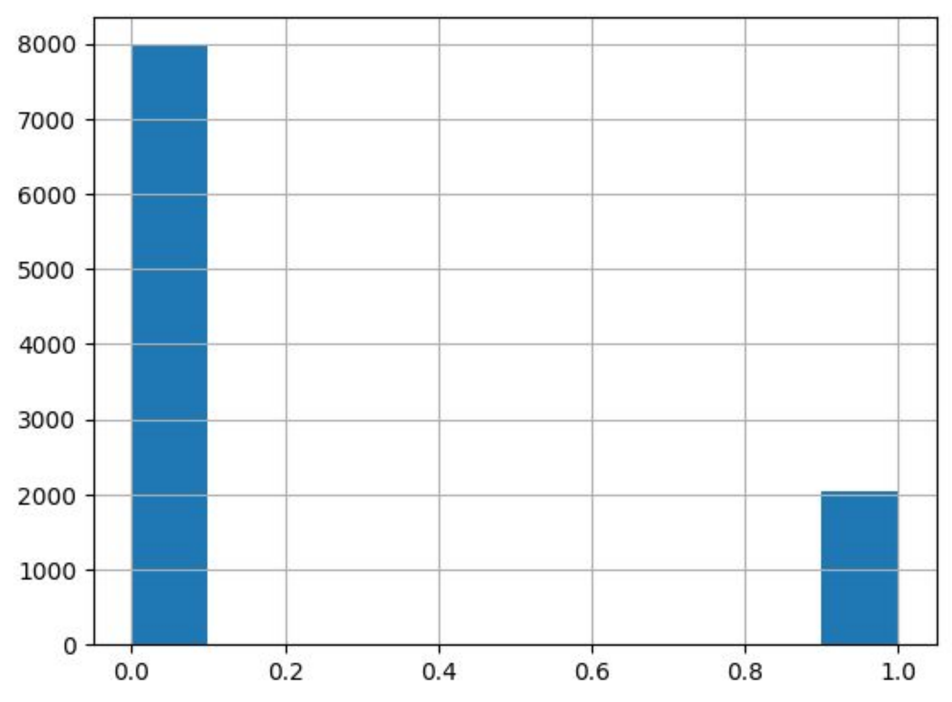
X, y defining

Select X, y for model:

```python
# chose x, y
y = cbc['Exited'].values
X = cbc[['Age', 'Balance', 'Gender_encode', 'Geography_France', 'IsActiveMember']].values
```

✓ 0.0s Python

Imbalance data



For the **data balance** of columns **‘Exited’,** the **0 values (current clients)** are **much higher** than **1 values (exited clients).**

Choose the **undersize method** to process.

Normalization

normalization: min-max scaler

```
# Scale X2
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(cb_balance.iloc[:, 1:11].values)

# create new dataframe
cbc = pd.DataFrame(data = X_scaled, columns = cb_balance.iloc[:, 1:11].columns)

# add y2 column
cbc['Exited'] = cb_balance['Exited']

cbc.head()
```

✓ 0.0s

	Age	Balance	Gender_encode	Geography_France	Geography_Germany	IsActiveMember	Exited
0	0.200	0.000	1.000	1.000	0.000	1.000	0
1	0.286	0.000	0.000	1.000	0.000	0.000	0
2	0.157	0.000	0.000	0.000	0.000	0.000	0
3	0.257	0.445	0.000	0.000	0.000	1.000	0
4	0.371	0.456	1.000	1.000	0.000	1.000	0

Split Train-Test

split train test dataset

```python
# chose x, y
y = cbc['Exited'].values
X = cbc[['Age', 'Balance', 'Gender_encode', 'Geography_France', 'Geography_Germany', 'IsActiveMember']].values
```

✓ 0.0s

```python
# create X_set for later use in model result
X_set = ['Age', 'Balance', 'Gender_encode', 'Geography_France', 'Geography_Germany', 'IsActiveMember']
```

✓ 0.0s

```python
# run split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

✓ 0.0s

```python
print(f'Data X_train: {X_train.sum()}')
print(f'Data y_train: {y_train.sum()}')
print(f'Data X_test: {X_test.sum()}')
print(f'Data y_test:{y_test.sum()}')
```

✓ 0.0s

```
Data X_train: 6768.618920505589
Data y_train: 1438
Data X_test: 2882.1677445508994
Data y_test:600
```

Prediction

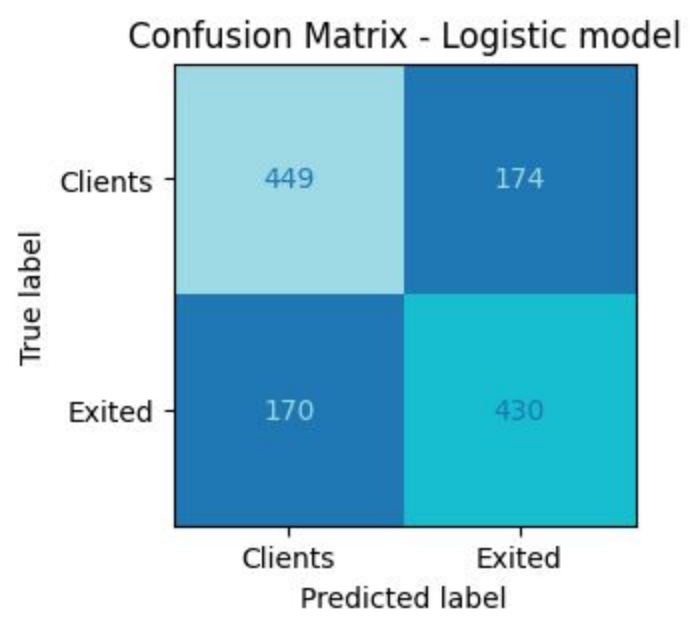
Machine learning

Models:

Models	Parameter
Logistic Regression	Default
Gaussian Naive Bayes	Default
Decision Tree	max_depth = 6
Random Forest	n_estimators = 71
K Nearst Neighbor	n_neighbors = 30

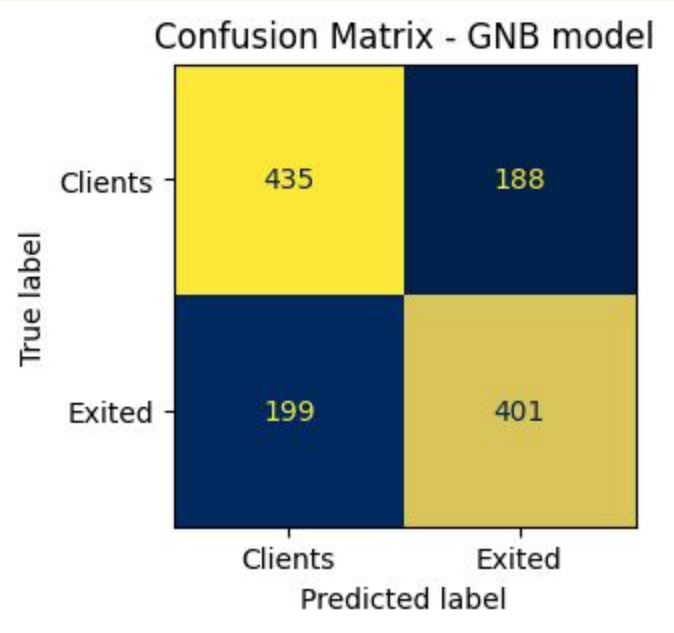
Logistic Regresion

	precision	recall	f1-score	support
0	0.73	0.72	0.72	623
1	0.71	0.72	0.71	600
accuracy			0.72	1223
macro avg	0.72	0.72	0.72	1223
weighted avg	0.72	0.72	0.72	1223

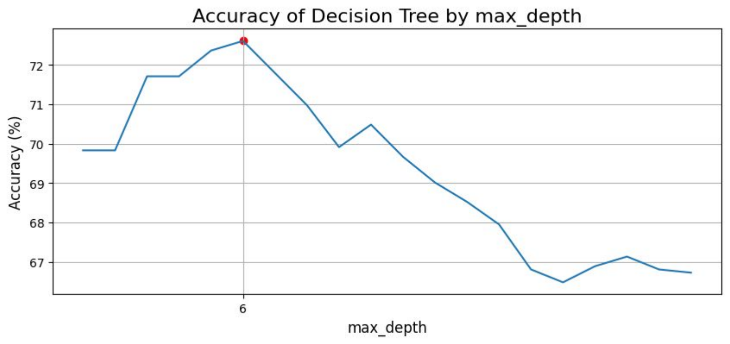


Gaussian Naive Bayes

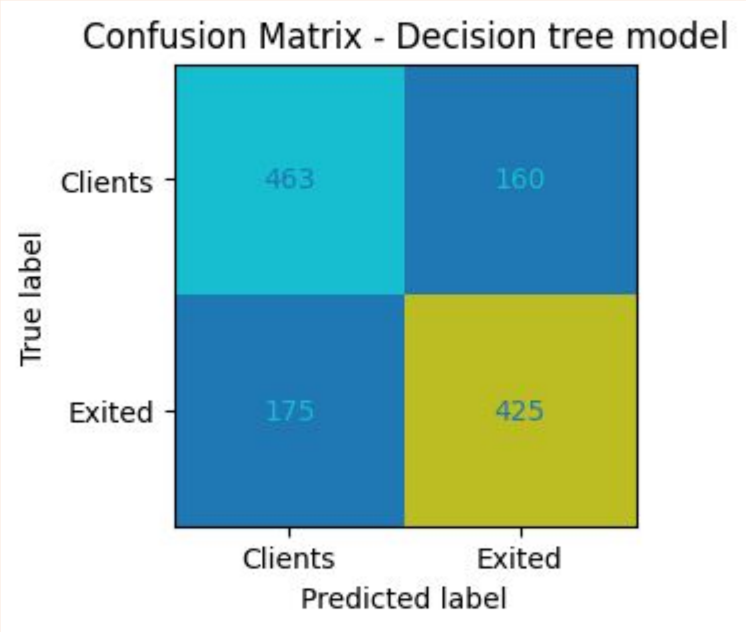
	precision	recall	f1-score	support
0	0.69	0.70	0.69	623
1	0.68	0.67	0.67	600
accuracy			0.68	1223
macro avg	0.68	0.68	0.68	1223
weighted avg	0.68	0.68	0.68	1223



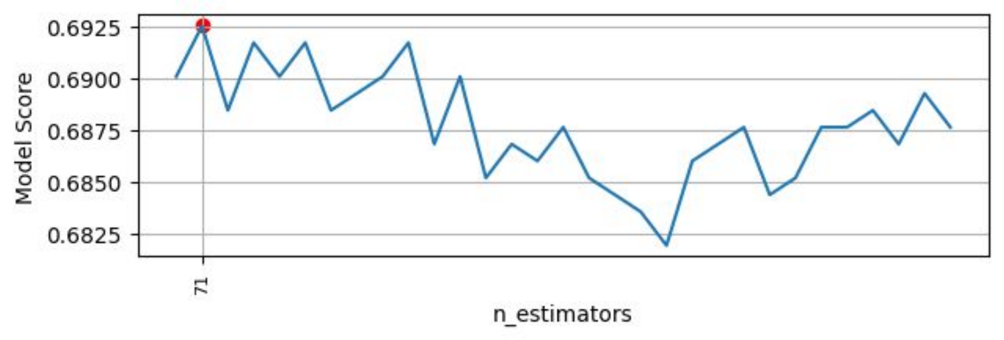
Decision Tree



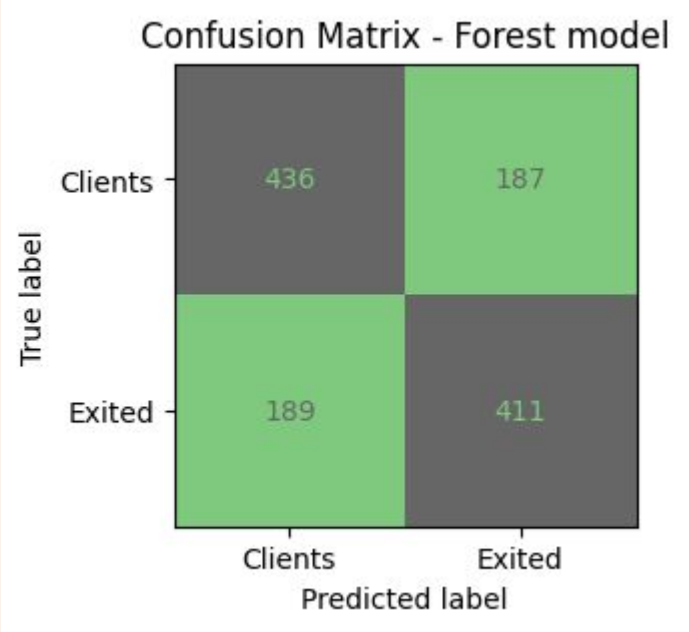
	precision	recall	f1-score	support
0	0.73	0.74	0.73	623
1	0.73	0.71	0.72	600
accuracy			0.73	1223
macro avg	0.73	0.73	0.73	1223
weighted avg	0.73	0.73	0.73	1223



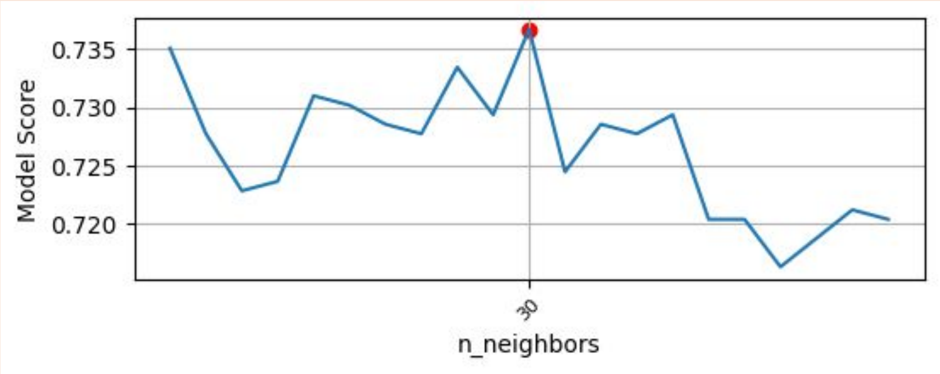
Random Forest



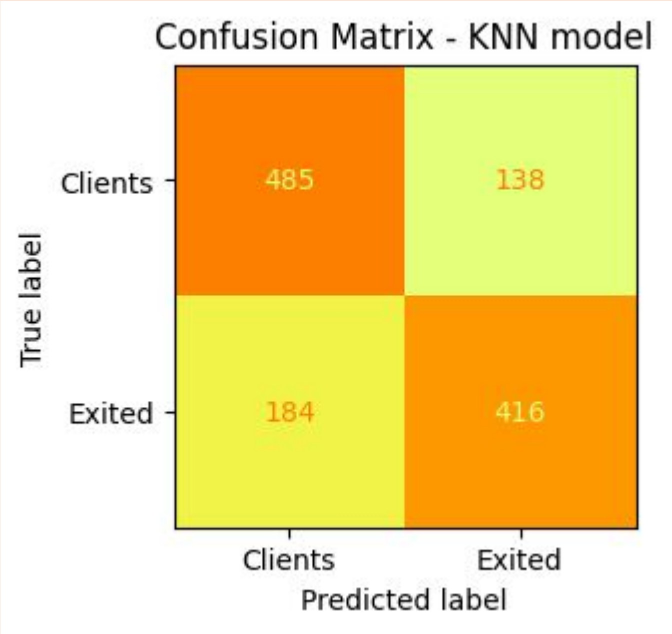
	precision	recall	f1-score	support
0	0.6976	0.6998	0.6987	623
1	0.6873	0.6850	0.6861	600
accuracy			0.6926	1223
macro avg	0.6924	0.6924	0.6924	1223
weighted avg	0.6925	0.6926	0.6925	1223



K Nearest Neighbors



	precision	recall	f1-score	support
0	0.7250	0.7785	0.7508	623
1	0.7509	0.6933	0.7210	600
accuracy			0.7367	1223
macro avg	0.7379	0.7359	0.7359	1223
weighted avg	0.7377	0.7367	0.7362	1223



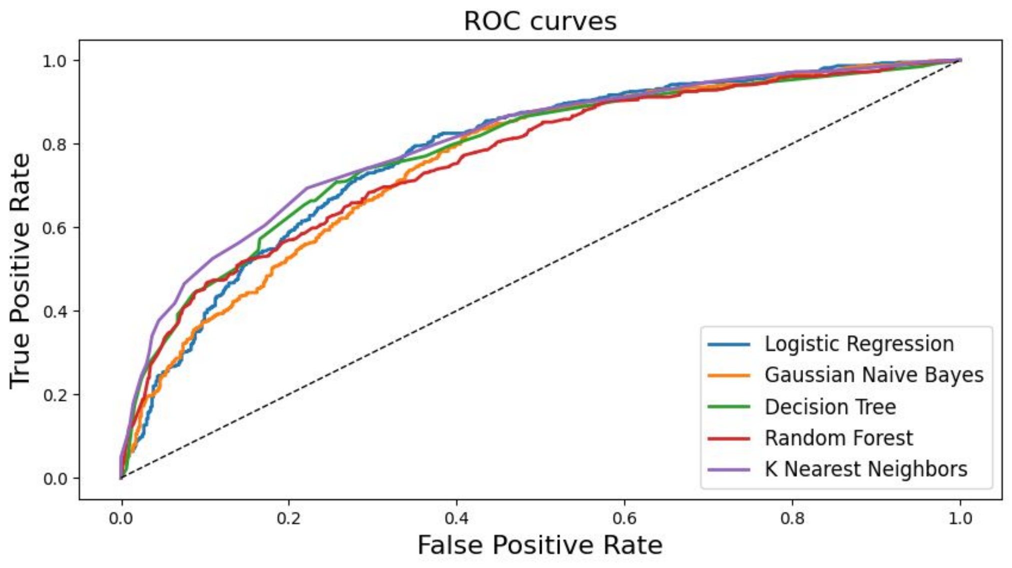
Result

Model	Accuracy Score	F1_Score	Precision	Recall	ROC AUC Score
Random forest	0.69	0.69	0.69	0.69	0.76
Logistic Regresson	0.72	0.71	0.71	0.72	0.78
KNN	0.74	0.72	0.75	0.69	0.80
GNB	0.68	0.67	0.68	0.67	0.76
Decesion Tree	0.73	0.72	0.73	0.71	0.78

KNN have the **highest result** in all score.

Decesion Tree is the **2nd best performance**.

Recomended this 2 models in this dataset.



Thank you for reading!

01 Introduction

Introduction of data and target

02 Data Analysis

Cleaning and Illustration

03 Prediction

Prediction by Machine Learning

04 Result

Summary