BANK CUSTOMER CHURN PREDICTION PROJECT REPORT

By: Reagan Odhiambo Otieno

Date: 10/22/2025

Table of Contents

BANK CUSTOMER CHURN PREDICTION PROJECT REPORT

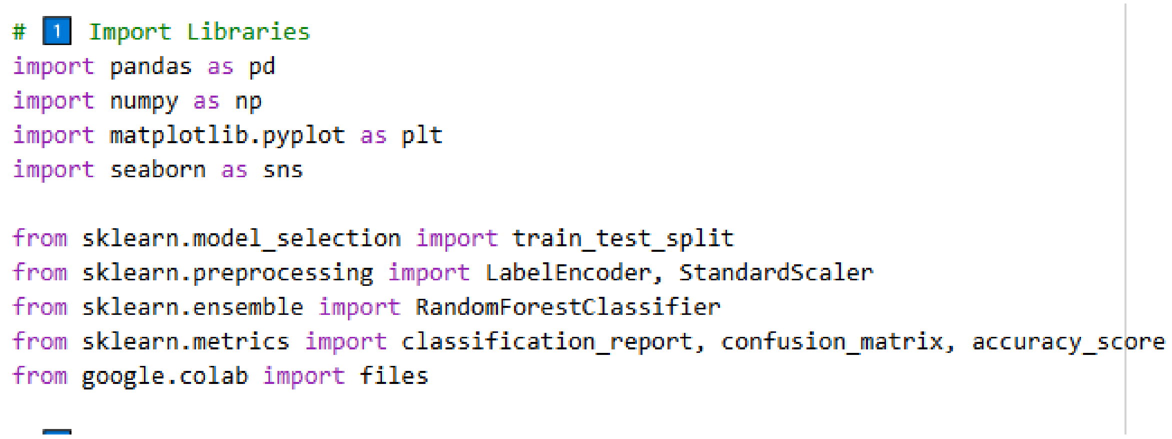
1. Introduction

Customer churn is a major concern for financial institutions. It refers to customers leaving or closing their accounts with the bank. Understanding the reasons behind churn and predicting which customers are likely to leave can help banks take preventive action to retain them.

This project aims to build a Machine Learning model that predicts customer churn using various customer attributes such as age, balance, activity status, and credit card ownership. The analysis is performed in Python using libraries such as pandas, scikit-learn, matplotlib, and seaborn.

2. Importing Libraries

The first step involves importing essential Python libraries required for data manipulation, visualization, and machine learning.



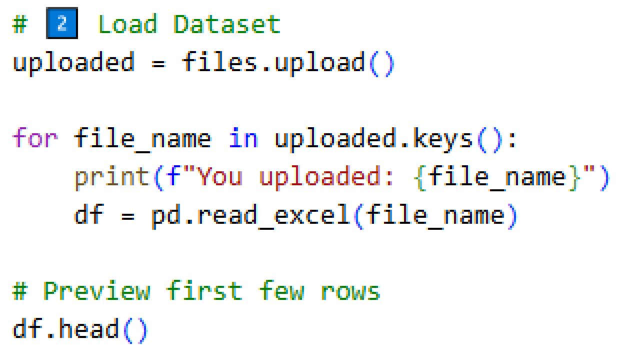
```
# 1️⃣ Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from google.colab import files
```

These libraries help in:

- **pandas**: Data loading and manipulation
- **numpy**: Numerical operations
- **matplotlib & seaborn**: Data visualization
- **scikit-learn**: Machine learning model creation and evaluation

3. Loading the Dataset

The dataset is uploaded and read into a pandas DataFrame using the following code:

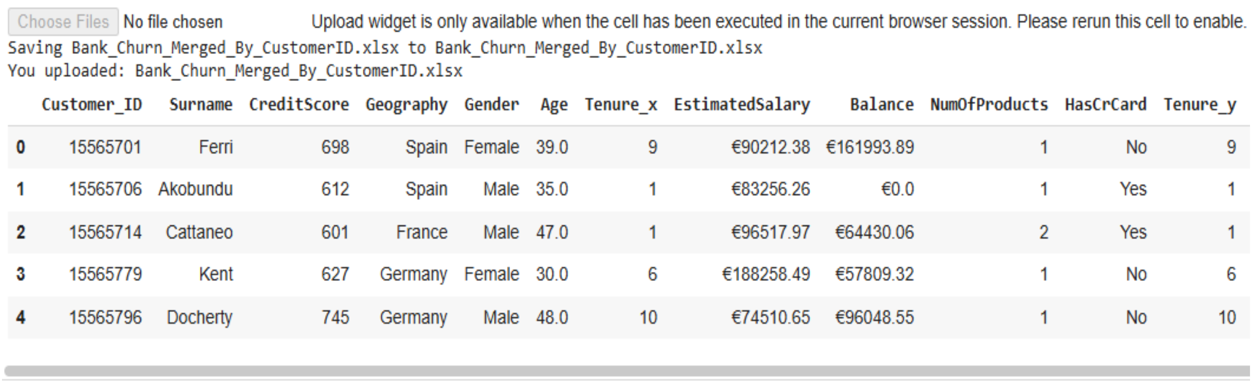


```
# 2️⃣ Load Dataset
uploaded = files.upload()

for file_name in uploaded.keys():
    print(f"You uploaded: {file_name}")
    df = pd.read_excel(file_name)

# Preview first few rows
df.head()
```

Explanation

- files.upload() opens a dialog to upload the dataset file.
- pd.read_excel() loads the Excel data into a DataFrame.

- df.head() displays the first five records for preview.



```
Choose Files  No file chosen          Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Bank_Churn_Merged_By_CustomerID.xlsx to Bank_Churn_Merged_By_CustomerID.xlsx
You uploaded: Bank_Churn_Merged_By_CustomerID.xlsx
```

	Customer_ID	Surname	CreditScore	Geography	Gender	Age	Tenure_x	EstimatedSalary	Balance	NumOfProducts	HasCrCard	Tenure_y
0	15565701	Ferri	698	Spain	Female	39.0	9	€90212.38	€161993.89	1	No	9
1	15565706	Akobundu	612	Spain	Male	35.0	1	€83256.26	€0.0	1	Yes	1
2	15565714	Cattaneo	601	France	Male	47.0	1	€96517.97	€64430.06	2	Yes	1
3	15565779	Kent	627	Germany	Female	30.0	6	€188258.49	€57809.32	1	No	6
4	15565796	Docherty	745	Germany	Male	48.0	10	€74510.65	€96048.55	1	No	10

4. Exploring the Data

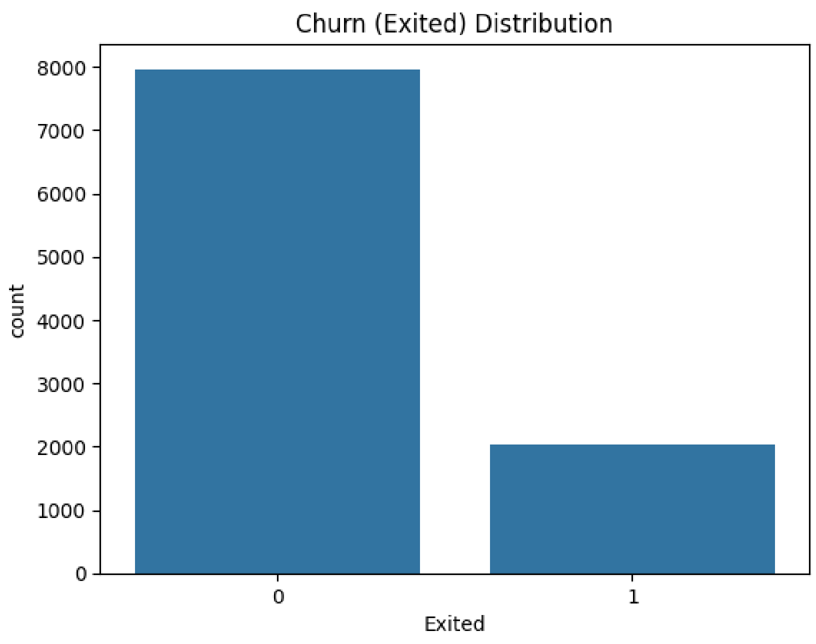
Before modeling, it’s essential to understand the data structure and check for missing values.

```
# Basic info
df.info()

# Check missing values
print("\nMissing values per column:")
print(df.isnull().sum())

# Descriptive stats
print("\nDescriptive statistics:")
print(df.describe())

# Churn distribution
sns.countplot(x='Exited', data=df)
plt.title('Churn (Exited) Distribution')
plt.show()
```

Explanation

- df.info() shows data types and non-null counts.
- df.isnull().sum() detects missing values in each column.
- df.describe() provides summary statistics (mean, std, min, max, etc.).
- The churn distribution plot helps visualize how many customers exited (1) vs. stayed (0).

```
Missing values per column:
Customer_ID        0
Surname            3
CreditScore        0
Geography          0
Gender             0
Age                3
Tenure_x           0
EstimatedSalary    0
Balance            0
NumOfProducts      0
HasCrCard          0
Tenure_y           0
IsActiveMember     0
Exited             0
dtype: int64
```



5. Data Cleaning and Encoding

To ensure the data is ready for modeling, unnecessary columns are dropped, and categorical variables are encoded into numerical form.

```
# Convert columns to lowercase for consistency
df.columns = df.columns.str.lower()
print("Columns:", df.columns.tolist())

# Drop irrelevant columns
cols_to_drop = ['customerid', 'surname','customer_id','tenure_y']
df = df.drop(columns=cols_to_drop, errors='ignore')

# Clean currency symbols and convert to float
cols_to_clean = ['balance', 'estimatedsalary']  # update based on your dataset

for col in cols_to_clean:
    # Remove € and commas, then convert to float
    df[col] = df[col].replace('[€$,]', '', regex=True).astype(float)


# Separate features and target
X = df.drop('exited', axis=1)
y = df['exited']


 # Identify categorical columns
 cat_cols = ['geography', 'gender', 'hascrcard', 'isactivemember']

 # Encode Yes/No columns if they exist
 if 'hascrcard' in X.columns:
     X['hascrcard'] = X['hascrcard'].map({'Yes': 1, 'No': 0})
 if 'isactivemember' in X.columns:
     X['isactivemember'] = X['isactivemember'].map({'Yes': 1, 'No': 0})

 # Label encode categorical columns
 le = LabelEncoder()
 for col in ['geography', 'gender']:
     if col in X.columns:
         X[col] = le.fit_transform(X[col])
```

Explanation

- Dropped irrelevant columns such as *Customer ID* and *Surname* (they don't influence churn).
- Cleaned numeric columns by removing symbols (e.g., € or $).
- Encoded categorical values into numbers using LabelEncoder().
  Example:

- ○ Gender: Male → 1, Female → 0
- ○ Geography: France → 0, Spain → 1, Germany → 2

```
Columns: ['customer_id', 'surname', 'creditscore', 'geography', 'gender', 'age', 'tenure_x',
```

6. Feature Scaling

Scaling ensures that large-valued features don't dominate the model.

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Explanation

- StandardScaler standardizes data (mean = 0, standard deviation = 1).
- It ensures fair comparison between variables such as *Age* and *Salary*.

7. Model Training (Random Forest Classifier)

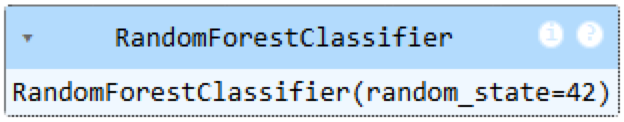
The dataset is split into training and testing sets before fitting the model.

```
# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y
)

# Train model
rf = RandomForestClassifier(
    n_estimators=100,
    random_state=42,
    max_depth=None,
    min_samples_split=2
)
rf.fit(X_train, y_train)
```

Explanation

- train_test_split() divides data into 80% training and 20% testing sets.
- The **Random Forest** algorithm is used because it is robust, handles both categorical and numeric data, and reduces overfitting.
- Parameters:

- n_estimators=100: number of trees.
- max_depth=None: allows trees to grow fully.
- min_samples_split=2: minimum samples required to split.



```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

8. Model Evaluation

After training the Random Forest Classifier, the model's performance was evaluated using the test dataset.

The following metrics were obtained:

🎯 Model Accuracy: 86.06%

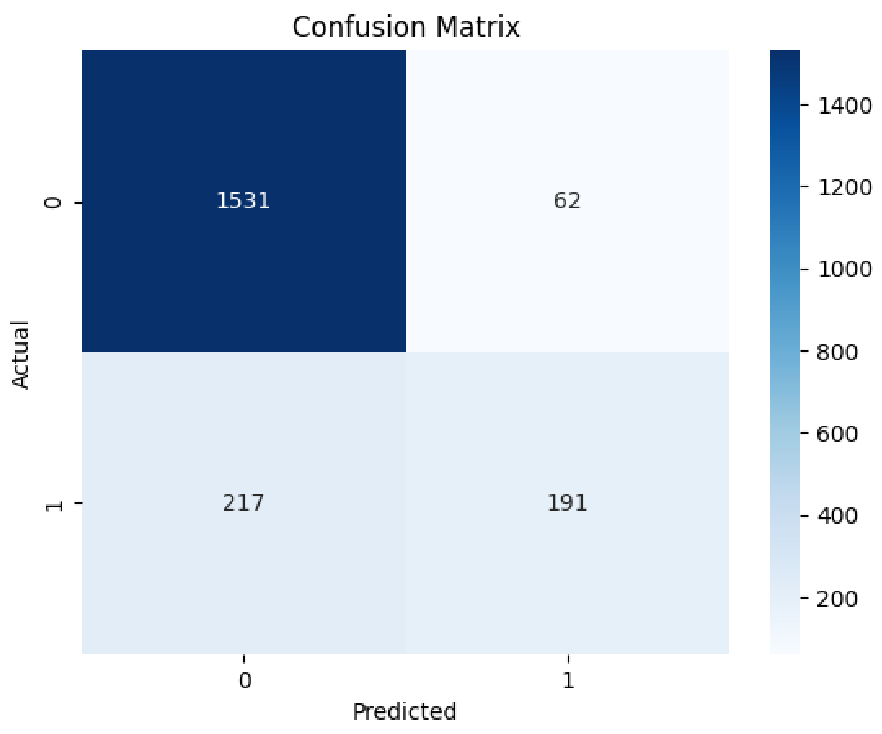
Classification Report

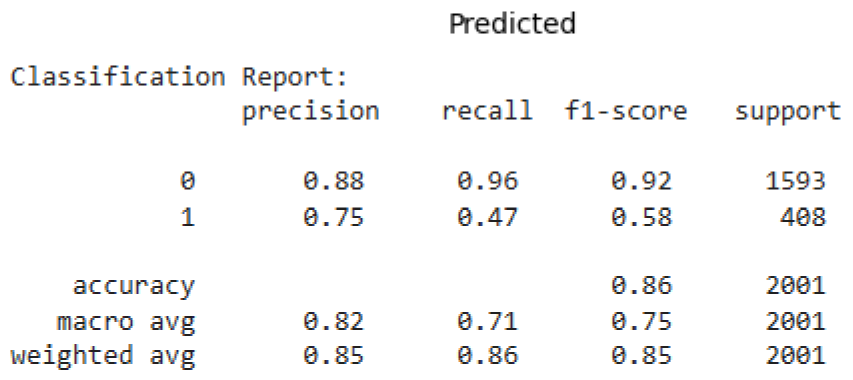
Metric	**Precision**	**Recall**	**F1-Score**	**Support**
Class 0 (Not Churned)	0.88	0.96	0.92	1593
Class 1 (Churned)	0.75	0.47	0.58	408
Overall Accuracy	**0.86**			2001

Interpretation

- The model achieved an overall accuracy of 86.06%, meaning it correctly predicted customer churn status for about 86 out of every 100 customers.
- Precision (0.75) for churned customers indicates that 75% of customers predicted as churned were actually churned.
- Recall (0.47) for churned customers means the model identified 47% of all true churners.
- F1-score (0.58) shows a moderate balance between precision and recall for churned customers.

This suggests that while the model performs very well in predicting non-churned customers (Class 0), it can be further improved to better detect churned customers (Class 1).

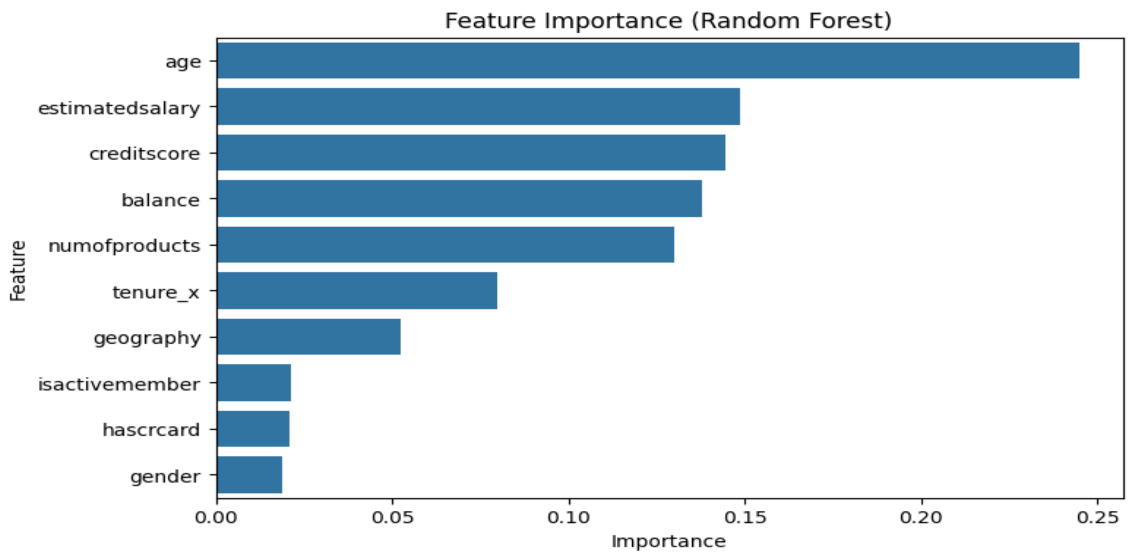




```
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.96      0.92      1593
           1       0.75      0.47      0.58       408

    accuracy                           0.86      2001
   macro avg       0.82      0.71      0.75      2001
weighted avg       0.85      0.86      0.85      2001
```

9. Feature Importance

The Random Forest algorithm also provides feature importance scores, indicating which features most influenced churn predictions.



Key influential features included:

- **Age** – Older customers were more likely to churn.
- **Balance** – Customers with specific balance ranges showed different churn behaviors.
- **IsActiveMember** – Active customers had lower churn rates.
- **EstimatedSalary** – Had moderate importance in predicting churn.

These insights can help the bank target high-risk groups with retention strategies such as loyalty programs, personalized offers, or improved engagement.

Link to colab code: https://colab.research.google.com/drive/1pfaXjD55C9NWht6HPSOx8Z50-I3lQnOg?usp=sharing

10. Conclusion

This project successfully implemented a Bank Customer Churn Prediction Model using a Random Forest Classifier. The model achieved an accuracy of 86.06%, which demonstrates strong predictive performance.

Key Findings

- The model is highly reliable at identifying customers who will stay.
- Further tuning or applying advanced models like XGBoost may improve detection of actual churners.
- Feature importance analysis revealed key behavioral and financial indicators influencing churn.