A Project Report On

Bank Customer Churn Prediction

Submitted in partial fulfillment of the requirement for the award of the degree

MASTER OF COMPUTR APPLICATION
from
Marwadi University

Academic Year 2024 – 25

Rutik Parmar (92400584150)
Shyam Tanna (92400584172)

Internal Guide
Miss. Gehna Sachdeva

Rajkot-Morbi Road, At & PO : Gauridad, Rajkot 360 003. Gujarat. India.

Faculty of Computer Applications (FoCA)

Certificate

This is to certify that the project work entitled

Heart Failure Prediction Using Logistic Regression

submitted in partial fulfillment of the requirement for the award of the degree of

Master of Science (Data Science)

of the

Marwadi University

is a result of the bonafide work carried out by

Rutik Parmar (92400584150)
Shyam Tanna (92400584172)

during the academic year 2024 – 2025

Miss. Gehna Sachdeva	*Dr. Sunil Bajeja*	*Dr. R. Sridaran*
Faculty Guide	**HOD**	**Dean**

DECLARATION

We hereby declare that this project work entitled **Bank Customer Churn Prediction Using Classification** is a record done by us.

We also declare that the matter embodied in this project is genuine work done by us and has not been submitted whether to this University or to any other University / Institute for the fulfillment of the requirement of any course of study.

Place: Rajkot

Date:

Rutik Parmar (92400584150) Signature:_____

Shyam Tanna (92400584172) Signature:_____

ACKNOWLEDGEMENT

It is indeed a great pleasure to express our thanks and gratitude to all those who helped us. No serious and lasting achievement or success one can ever achieve without the help of friendly guidance and co-operation of so many people involved in the work.

We are very thankful to our guide **Miss. Gehna Sachdeva,** the person who makes us to follow the right steps during our project work. We express our deep sense of gratitude to for his guidance, suggestions and expertise at every stage. A part from that his valuable and expertise suggestion during documentation of our report indeed help us a lot.

Thanks to our friend and colleague who have been a source of inspiration and motivation that helped to us during our project work.

We are heartily thankful to the Dean of our department **Dr. R. Sridaran** for giving us an opportunity to work over this project and for their end-less and great support to all other people who directly or indirectly supported and help us to fulfil our task.

Rutik Parmar (92400584150) Signature:_____
Shyam Tanna (92400584172) Signature:_____

CONTENTS

1. Introduction:

1.1 Objective of the New System:

- To preprocess the dataset (ml.csv) by handling missing values, encoding categorical features, and scaling numerical features.
- To train and evaluate multiple **classification algorithms** (Logistic Regression, KNN, Decision Tree, Random Forest, SVM, Naïve Bayes) on the dataset.
- To compare the models based on **accuracy, confusion matrix, and classification report**.
- To visualize important insights using **heatmaps, histograms, scatter plots, and boxplots**.
- To build a robust model that can predict whether a customer will **exit (1) or stay (0)** with high accuracy

1.2 Problem Definition:

- In the competitive business environment, retaining customers is a major challenge. Companies face significant losses when customers discontinue their services (churn/exit). Predicting whether a customer will **exit or remain loyal** based on their historical and behavioral data can help businesses take preventive actions.
- This project focuses on developing a **machine learning classification model** to analyze customer data and predict the likelihood of customer exit (Exited column). By applying different classification algorithms and comparing their performance, the project aims to identify the most effective model for churn prediction

1.3 Core Components:

- **Dataset:** Bank Customer Churn dataset
- **Preprocessing:** Handling null values, encoding, scaling
- **Algorithms:** Logistic Regression, KNN, Decision Tree Classifier, Random Forest Classifier, SVM, NB
- **Evaluation:** Accuracy, Confusion Matrix, Graphs

1.4 Project Profile:

- **Project Title:** Bank Customer Churn Prediction Using Classification
- **Domain:** Bank
- **Technology:** Python (sklearn, pandas, seaborn, matplotlib)

1.5 Assumptions and Constraints:

- **Assumptions:**
  - The dataset is clean after preprocessing.
  - Models are evaluated using accuracy and classification reports.
- **Constraints:**
  - Dataset may not represent real-world noise.
  - Imbalanced data may affect results.

1.6 Advantages and Limitations of the Proposed System:

- **Advantages:**
  - Early Churn Prediction
  - Multiple Algorithms Used
- Data-Driven Decision Making

2. Requirement Determination & Analysis:

2.1 Requirement Determination:

To build a predictive system for heart failure, the following requirements were identified:

- Data preprocessing pipeline
- Machine learning algorithms
- Model evaluation techniques
- Visualization tools for understanding the data

2.2 Targeted Users:

Bank customers who are at risk of leaving

2.3 Tool details (Python / PowerBI/ Tableau):

This project is developed using Python due to its wide range of libraries and ease of implementation for data analysis and machine learning.

2.4 Library description (Details on various libraries / packages used):

- **pandas:** Data manipulation and analysis
- **numpy:** Numerical operations
- **sklearn:** Machine learning algorithms and preprocessing tools
- **seaborn:** Statistical data visualization
- **matplotlib:** Plotting graphs

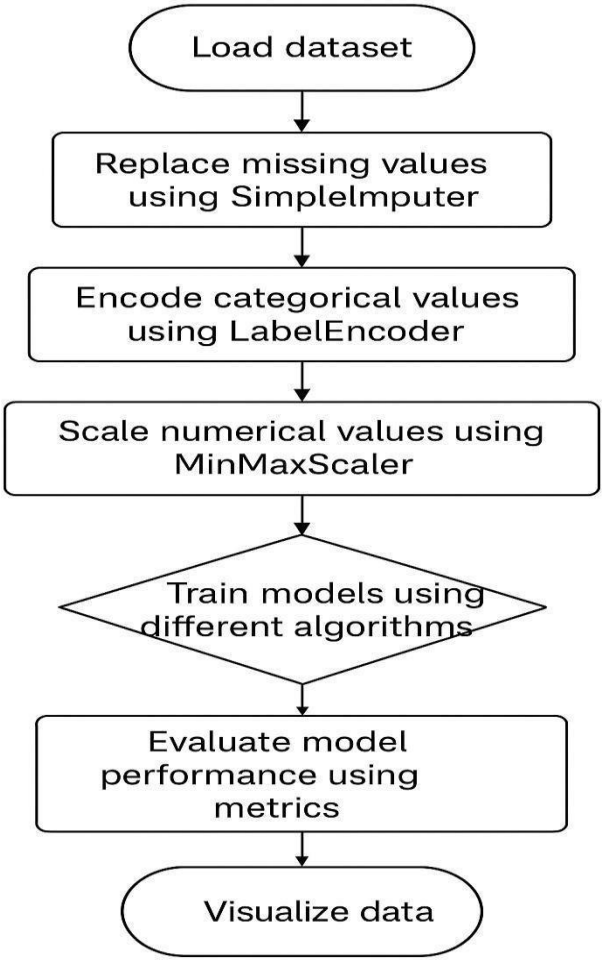
3. System Design:

3.1 Flowchart / Algorithm with steps:

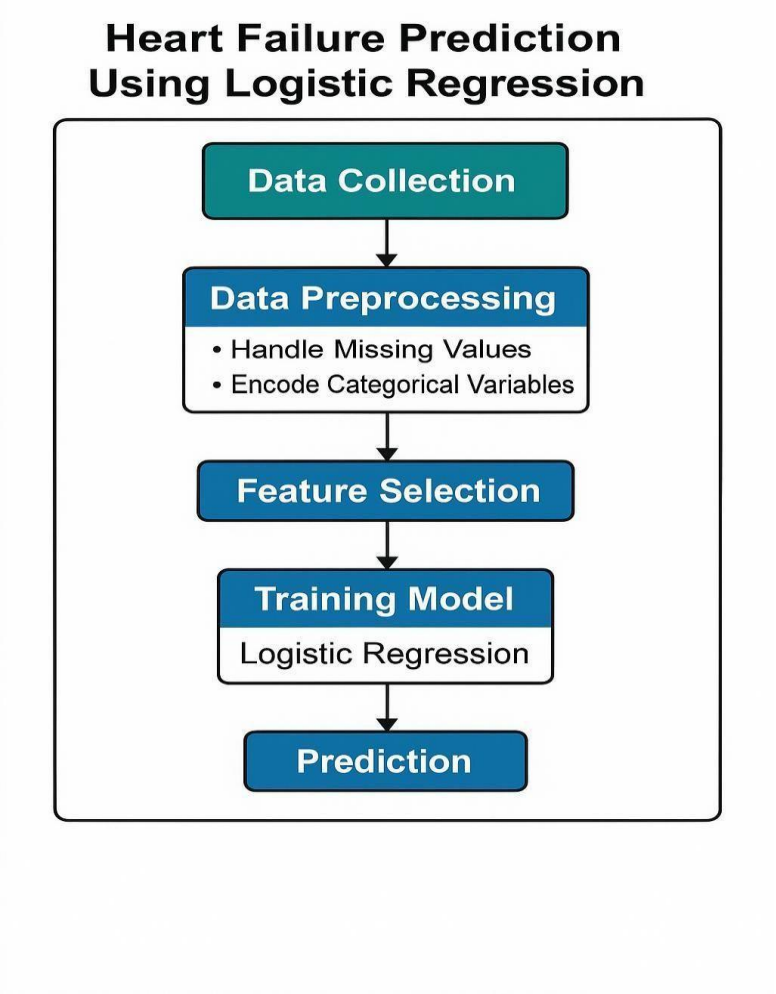
- **Algorithm:**

- Step 1: Load dataset
- Step 2: Replace missing values using SimpleImputer
- Step 3: Encode categorical values using LabelEncoder
- Step 4: Scale numerical values using MinMaxScaler
- Step 5: Split data into training and test sets
- Step 6: Train models using different algorithms
- Step 7: Evaluate model performance using metrics
- Step 8: Visualize data using heatmap, scatter, boxplot, and histogram

- **Flowchart:**



3.2 Dataset Design:



3.3 Details on preprocessing steps applied:

- **Missing Values:** Filled using SimpleImputer (mean for numerical, most frequent for categorical)
- **Categorical Encoding:** LabelEncoder applied to all object type columns
- **Feature Scaling:** MinMaxScaler used on numerical columns to normalize them between 0 and 1.

4. Development:

4.1 Script Details / Source Code:

The project is divided into two main Python scripts:

1. preprocessing.py: Responsible for cleaning the dataset using SimpleImputer, encoding with LabelEncoder, and scaling using MinMaxScaler.

➢ **Code:**

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.impute import SimpleImputer

# Models
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB

# Evaluation
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# 1. Load Dataset

df = pd.read_csv("D:/ML_Project/ml.csv")
df
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSal
0	1	18318661	Romeo	464	Austrelia	Female	34	4	202388.20	4	1	0	6433
1	2	75804843	Gerasimov	645	India	Female	30	2	10418.25	1	1	0	15836
2	3	18752486	Kay	676	China	Female	43	3	90185.31	2	0	1	1607
3	4	35153662	Jackson	650	USA	Male	23	1	228938.17	1	1	0	13816
4	5	32159796	Martinez	768	Austrelia	Female	25	7	170946.92	3	1	0	21043
...	...	...	...	...	...	...	...	...	...	...	...	...	...
29995	29996	57668068	Azikiwe	457	Spain	Male	60	10	117389.52	1	0	0	11116
29996	29997	99693067	Bearce	307	USA	Male	40	2	190170.01	1	0	0	11917
29997	29998	89022498	Anderson	791	China	Male	37	10	101442.38	2	0	0	8195
29998	29999	24911241	Odinakachukwu	619	America	Female	76	3	146576.48	1	0	0	15939
29999	30000	29983453	Lorenzo	781	Germany	Male	25	9	82276.28	3	0	1	2664

30000 rows × 14 columns

```
# 2. Handle Missing Values
print("Dataset preview: ")
print(df.head())
```

➢ **Output:**

```
Dataset preview:
   RowNumber  CustomerId    Surname  CreditScore  Geography  Gender  Age  \
0          1    18318661      Romeo          464  Austrelia  Female   34
1          2    75804843  Gerasimov          645      India  Female   30
2          3    18752486        Kay          676      China  Female   43
3          4    35153662    Jackson          650        USA    Male   23
4          5    32159796   Martinez          768  Austrelia  Female   25

   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       4  202388.20              4          1               0
1       2   10418.25              1          1               0
2       3   90185.31              2          0               1
3       1  228938.17              1          1               0
4       7  170946.92              3          1               0

   EstimatedSalary  Exited
0         64339.87       0
1        158364.88       1
2         16071.67       1
3        138163.15       1
4        210431.25       0
```

```
print(df.dtypes)
```

```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

```
num_cols = df.select_dtypes(include=['int64','float64']).columns.tolist()

cat_cols = df.select_dtypes(include=['object']).columns.tolist()

print("\n numeric columns: ")

print(num_cols)
```

```
 numeric columns:
['RowNumber', 'CustomerId', 'CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMembe
r', 'EstimatedSalary', 'Exited']
```

```
print("\n categorical columns: ")
print(cat_cols)
```

```
 categorical columns:
['Surname', 'Geography', 'Gender']
```

```
print("Missing values before cleaning:")
print(df.isnull().sum())
```

```
Missing values before cleaning:
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
dtype: int64
```

```
num_imputer = SimpleImputer(strategy="mean")
df[num_cols] = num_imputer.fit_transform(df[num_cols])

cat_imputer = SimpleImputer(strategy="most_frequent")
df[cat_cols] = cat_imputer.fit_transform(df[cat_cols])

# 3. Encode Categorical

label_encoders = {}
for col in cat_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

```
# 4. Choose Target Column

if "Exited" in df.columns:
    target = "Exited"
else:
    # fallback: last column ko target maan lo
    target = df.columns[-1]
```

🎯 Target Selected: Exited

```
X = df.drop(columns=[target])
y = df[target]

# 5. Feature Scaling
from sklearn.preprocessing import MinMaxScaler

# Identify numeric columns again
num_cols = df.select_dtypes(include=['int64','float64']).columns.tolist()

# Remove target column if present
if "Exited" in num_cols:
    num_cols.remove("Exited")

# Apply scaling
scaler = MinMaxScaler()
X[num_cols] = scaler.fit_transform(X[num_cols])

# 6. Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

#Train Models
models = {
    "Logistic Regression": LogisticRegression(max_iter=500),
    "KNN": KNeighborsClassifier(n_neighbors=5),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "SVM (RBF)": SVC(kernel="rbf", probability=True),
    "Naive Bayes": GaussianNB()
}

results = {}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    acc = round(accuracy_score(y_test, y_pred)*100, 2)
    results[name] = acc
```

```
print("\n================================")
print(f" 🔹 {name}")
print("Accuracy:", acc, "%")
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
==============================
🔹 Logistic Regression
Accuracy: 49.97 %
Confusion Matrix:
 [[2525 1993]
 [2510 1972]]
Classification Report:
              precision    recall  f1-score   support

         0.0       0.50      0.56      0.53      4518
         1.0       0.50      0.44      0.47      4482

    accuracy                           0.50      9000
   macro avg       0.50      0.50      0.50      9000
weighted avg       0.50      0.50      0.50      9000
```


```
==============================
🔹 Logistic Regression
Accuracy: 49.97 %
Confusion Matrix:
 [[2525 1993]
 [2510 1972]]
Classification Report:
              precision    recall  f1-score   support

         0.0       0.50      0.56      0.53      4518
         1.0       0.50      0.44      0.47      4482

    accuracy                           0.50      9000
   macro avg       0.50      0.50      0.50      9000
weighted avg       0.50      0.50      0.50      9000
```


```
==============================
🔹 KNN
Accuracy: 50.53 %
Confusion Matrix:
 [[2320 2198]
 [2254 2228]]
Classification Report:
              precision    recall  f1-score   support

         0.0       0.51      0.51      0.51      4518
         1.0       0.50      0.50      0.50      4482

    accuracy                           0.51      9000
   macro avg       0.51      0.51      0.51      9000
weighted avg       0.51      0.51      0.51      9000
```


```
==============================
🔹 KNN
Accuracy: 50.53 %
Confusion Matrix:
 [[2320 2198]
 [2254 2228]]
Classification Report:
              precision    recall  f1-score   support

         0.0       0.51      0.51      0.51      4518
         1.0       0.50      0.50      0.50      4482

    accuracy                           0.51      9000
   macro avg       0.51      0.51      0.51      9000
weighted avg       0.51      0.51      0.51      9000
```

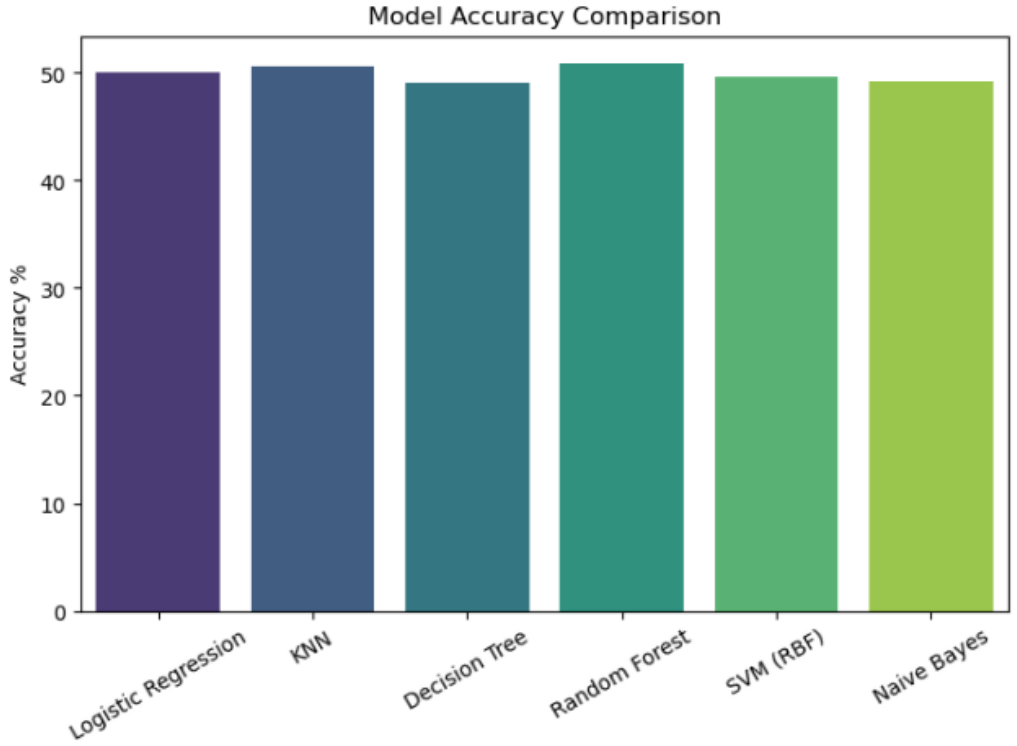

```
#Accuracy Comparison Plot
plt.figure(figsize=(8,5))
sns.barplot(x=list(results.keys()), y=list(results.values()), palette="viridis")
plt.title("Model Accuracy Comparison")
plt.ylabel("Accuracy %")
plt.xticks(rotation=30)
plt.show()

#Feature Corelation Heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show(block=False)
```

```
# Histogram for all numerical columns dynamically
fig, axes = plt.subplots(len(num_cols), 1, figsize=(10, 4 * len(num_cols)))

for i, col in enumerate(num_cols):
    sns.histplot(df[col], bins=30, kde=True, ax=axes[i])
    axes[i].set_title(f"Histogram of {col}")

plt.tight_layout()
plt.show()

#Box Plot
fig, axes = plt.subplots(1, len(num_cols), figsize=(15, 5))
for i, col in enumerate(num_cols):
    sns.boxplot(x=y, y=df[col], ax=axes[i])
    axes[i].set_title(f"{col} vs {target}")
plt.tight_layout()
plt.show()

#Scatter Graph
if "Age" in df.columns and "Balance" in df.columns:
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x=df["Age"], y=df["Balance"], hue=df["Exited"], palette="Set1",
alpha=0.7)
    plt.title("Scatter Plot: Age vs Balance (Colored by Exited)", fontsize=14)
    plt.show()

#Prediction
new_data = pd.DataFrame([{
    "CreditScore": 650,
    "Geography": "France",
    "Gender": "Male",
    "Age": 40,
    "Tenure": 5,
    "Balance": 60000.0,
    "NumOfProducts": 2,
    "HasCrCard": 1,
    "IsActiveMember": 1,
    "EstimatedSalary": 50000.0
}])

pred_class = model.predict(new_data)[0]
print("Here is prediction value from our sample data ")
print("Predicted Exited:", pred_class)
```

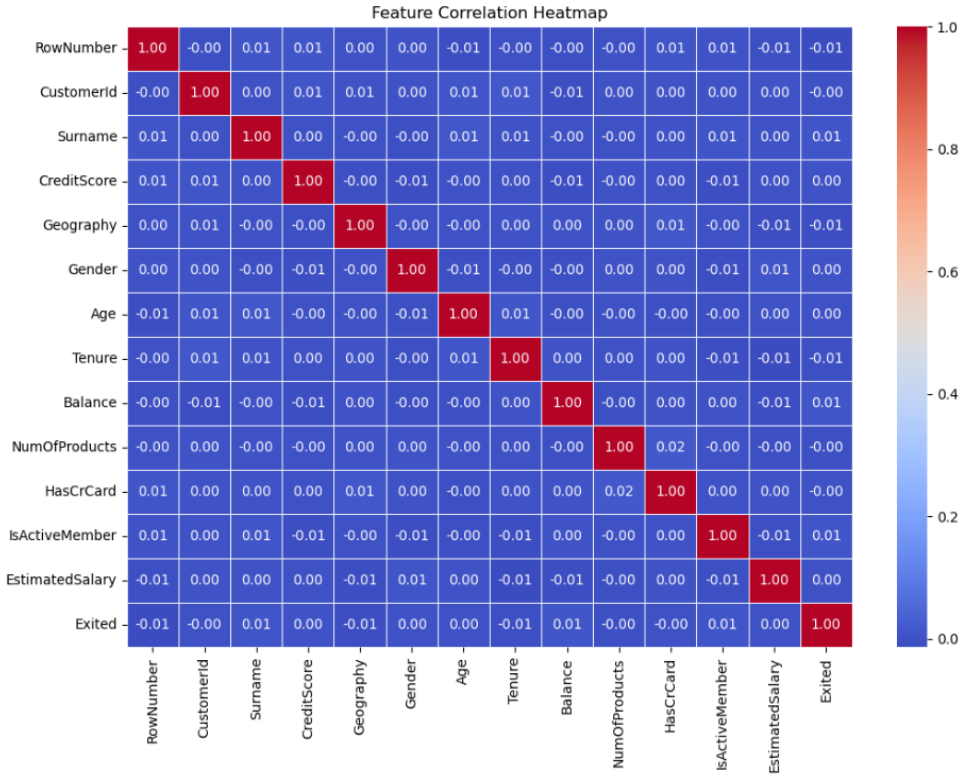
Graphical Visualization (heatmap, scatter plot, histogram, box plot):

The following visualizations were used:

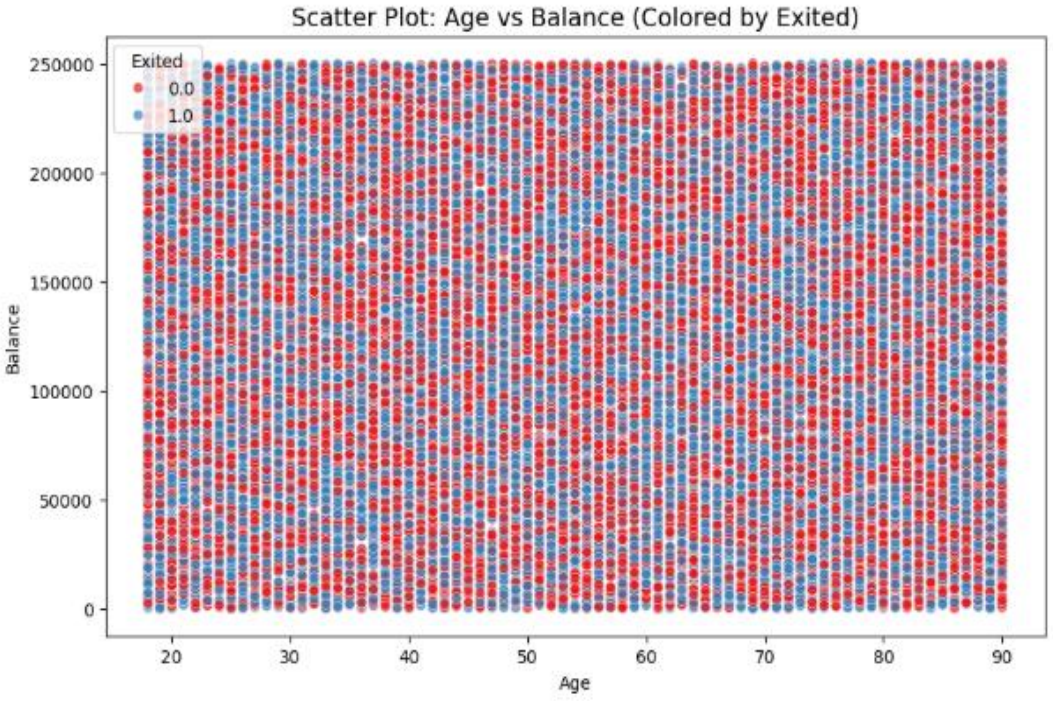
- **Accuracy Comparision Plot:**



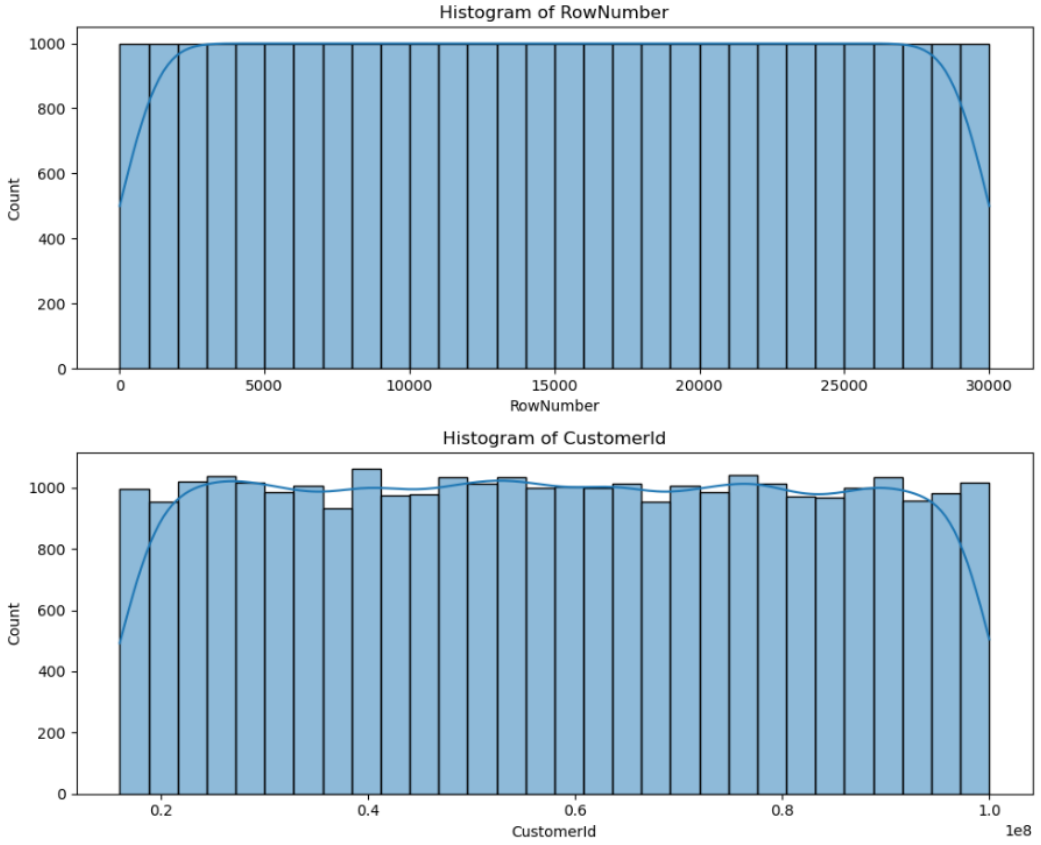
- **Heatmap:** Shows correlation between all features to identify the most influential variables.

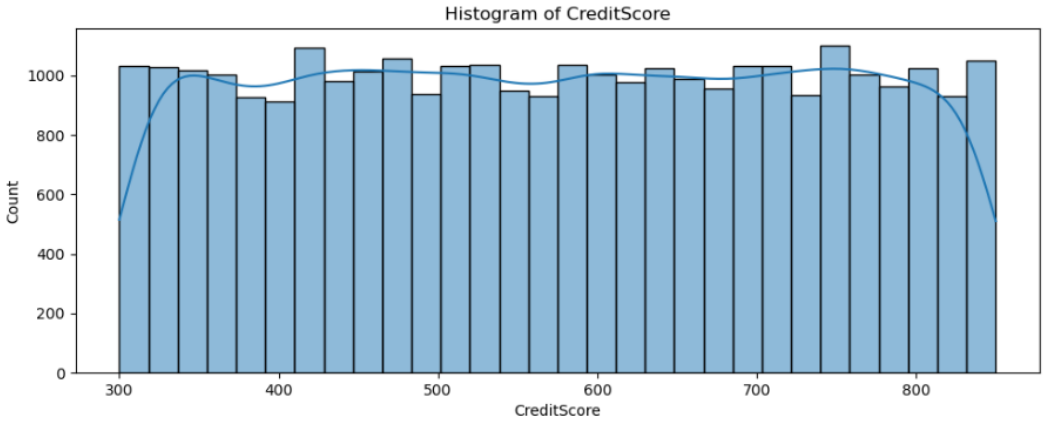


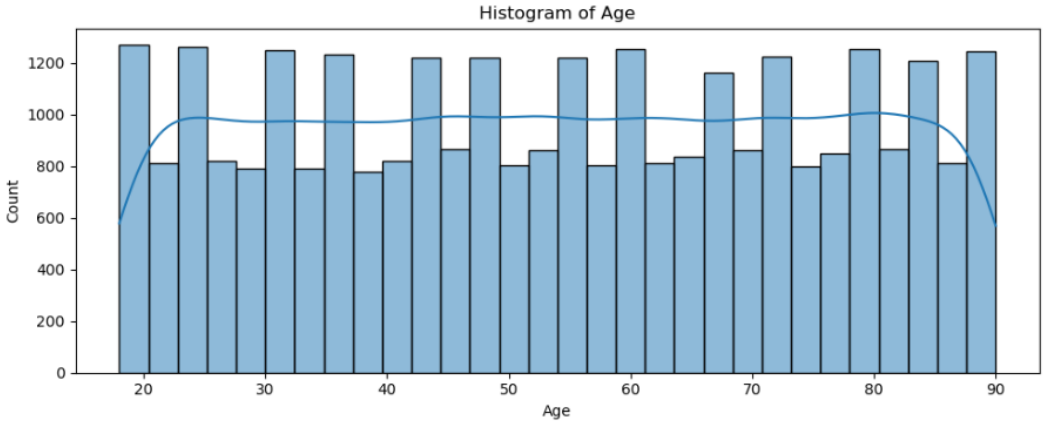
- **Scatter Plot:** Visualizes pairwise relationships between numerical features.

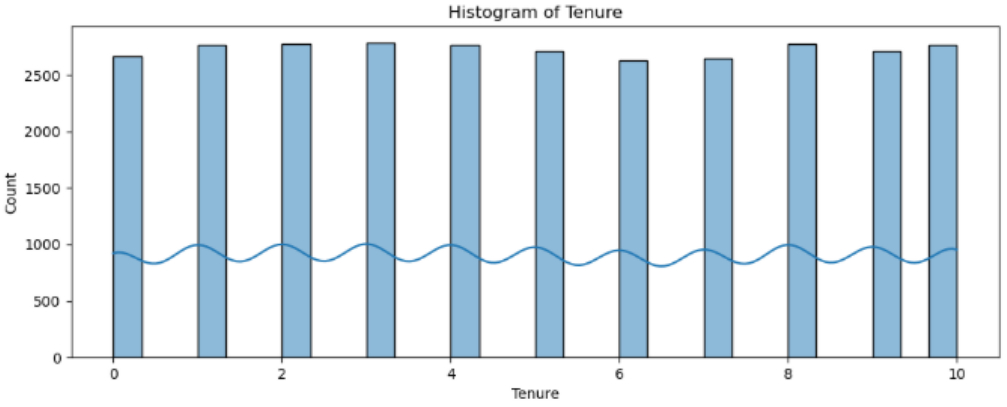


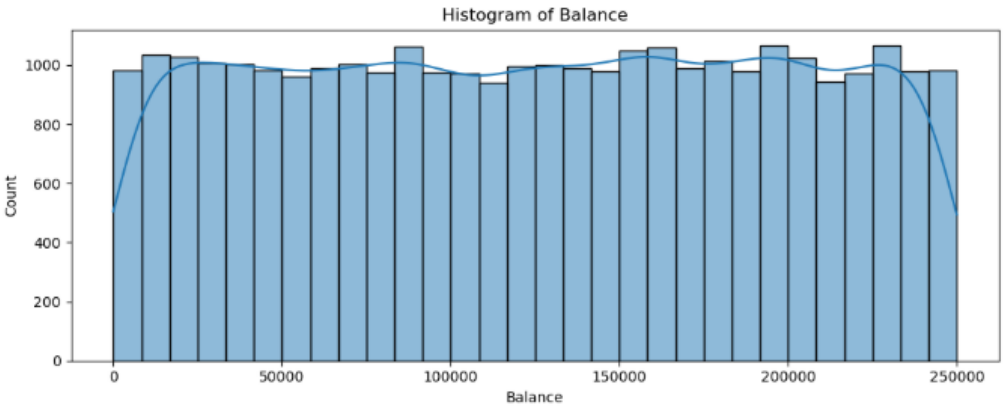
- **Histogram:** Displays the distribution of individual numerical features.

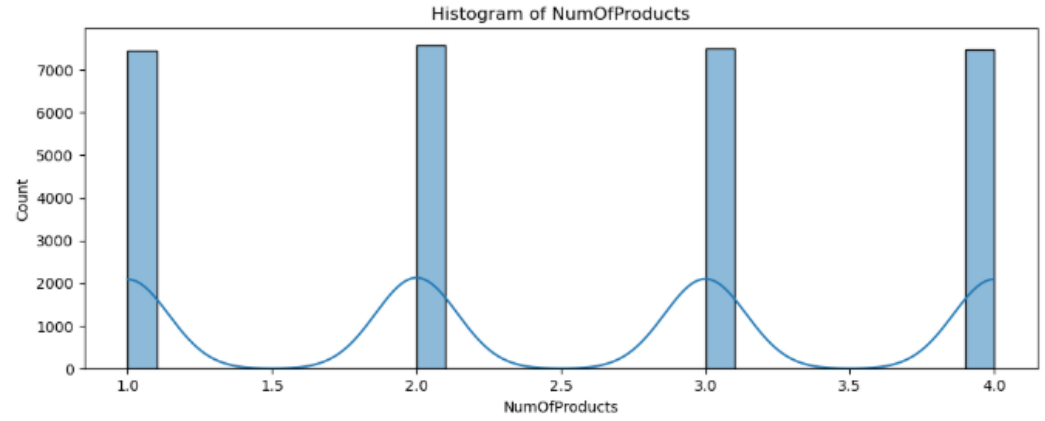


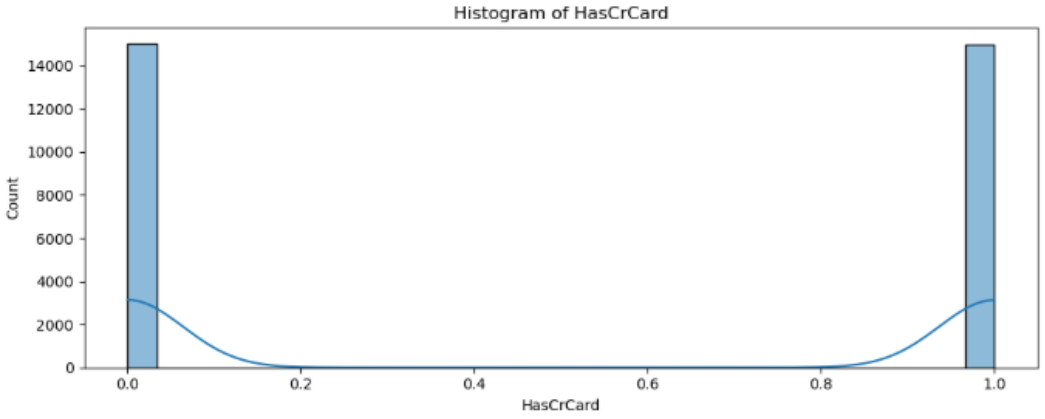

Histogram of CreditScore
Count
CreditScore
0
200
400
600
800
1000
300
400
500
600
700
800

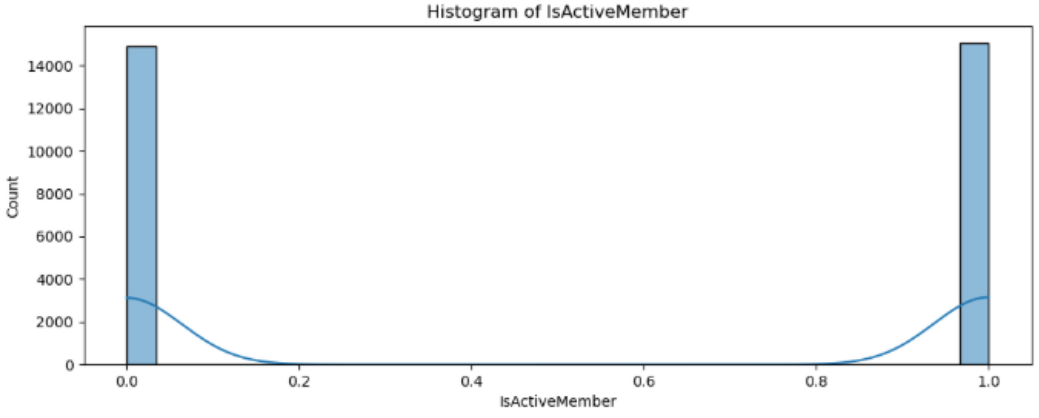

Histogram of Age
Count
Age
0
200
400
600
800
1000
1200
20
30
40
50
60
70
80
90

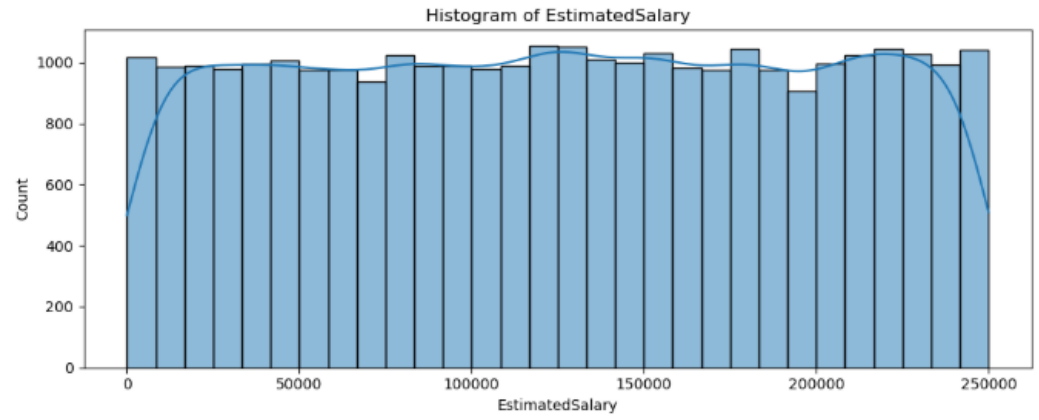

Histogram of Tenure
Count
Tenure
0
500
1000
1500
2000
2500
0
2
4
6
8
10


Histogram of Balance
Count
Balance
0
200
400
600
800
1000
0
50000
100000
150000
200000
250000

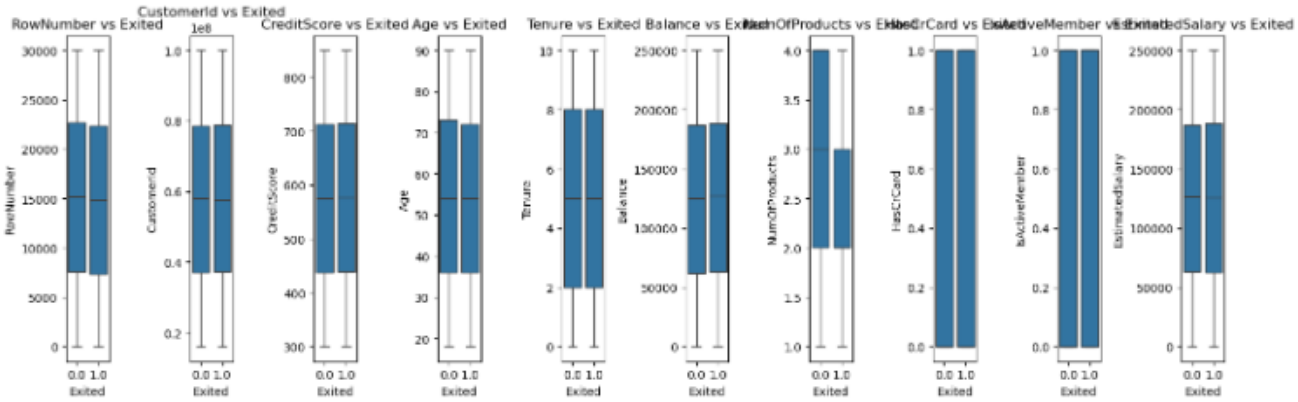

Histogram of NumOfProducts
Count
NumOfProducts
0
1000
2000
3000
4000
5000
6000
7000
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Histogram of HasCrCard
Count
HasCrCard
0
2000
4000
6000
8000
10000
12000
14000
0.0
0.2
0.4
0.6
0.8
1.0


Histogram of IsActiveMember
Count
IsActiveMember
0
2000
4000
6000
8000
10000
12000
14000
0.0
0.2
0.4
0.6
0.8
1.0


Histogram of EstimatedSalary
Count
EstimatedSalary
0
200
400
600
800
1000
0
50000
100000
150000
200000
250000

- **Box Plot :**



4.2 Test Reports (Accuracy, Classification Reports):

Each model was evaluated using Confusion Matrix, Accuracy Score, and Classification Report.

Results include precision, recall, F1-score, and overall accuracy. Below are some sample findings:

- **Logistic Regression:** Accuracy – 49.97 %
- **KNN :** Accuracy – 50.53%
- **Dicision Tree :** Accuracy – 49.06%
- **Random Forest :** Accuracy – 50.88%

5. Proposed Enhancements:

- The current project can be enhanced by integrating deep learning models such as Artificial Neural Networks for improved accuracy.
  Additional customer behavior data like transaction history, feedback, and social interactions can be included to strengthen predictions.
  Real-time churn prediction systems can be developed using streaming data for instant decision-making.
  Feature selection and hyperparameter tuning can be applied to further optimize model performance.
  Finally, deploying the model as a web or mobile application will make it easily accessible for business users.

6. Conclusion:

This project successfully applies machine learning algorithms to predict customer churn (Exited).
By identifying at-risk customers in advance, businesses can take proactive steps to improve retention.
Overall, the system helps reduce revenue loss and enhances customer satisfaction through data-driven insights.

7. Bibliography:

1. scikit-learn: Machine Learning in Python – https://scikit-learn.org/

2. pandas Documentation – https://pandas.pydata.org/

3. seaborn Documentation – https://seaborn.pydata.org/

4. matplotlib Documentation – https://matplotlib.org/

5. Heart Disease Dataset (original inspiration) – https://www.kaggle.com/