A Project Report On

Churn Modelling

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

Master of Computer Applications
(MCA)

Academic Year 2025 – 26

YASH CHAVAD (92400584193)

Internal Guide
Pro. Vipul Bambhaniya

Marwadi University

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

Master of Computer Applications (MCA)

Certificate

This is to certify that the project work entitled

Churn Modelling

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

Master of Computer Applications (MCA)

of the

Marwadi University

is a result of the bonafide work carried out by

YASH CHAVDA (92400584193)

during the academic year 2025-26

Faculty Guide	HOD	Dean

CERTIFICATE

(For FINAL SEMESTER INDUSTRIAL PROJECT, students need to keep COMPANY CERTIFICATE here)

DECLARATION

We hereby declare that this project work entitled **Churn Modelling Detection 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:

Date:

YASH CHAVDA (92400584193) 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 **Pro.Vipul Bambhaniya,** the person who makes us to follow the right steps during our project work. We express our deep sense of gratitude to for his /her guidance, suggestions and expertise at every stage. A part from that his/her 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 Professor & Dean of our department **Dr. R. Sridaran and** Head of Department **Dr. Sunil Bajeja** for giving us an opportunity to work over this project and for their end-less and great support. And to all other people who directly or indirectly supported and help us to fulfil our task.

YASH CHAVDA (92400584193) Signature:_____

COMPANY PROFILE

<Students must write the profile maximum in one page

CONTENTS

1. SYNOPSYS

Credit This project is about predicting whether a bank customer will leave the bank or stay. The dataset contains information such as age, credit score, balance, salary, and more. The main column we want to predict is **"Exited,"** which shows if a customer has churned (1) or not (0).

Before training the models, the data is cleaned by filling missing values and scaling the numbers so they are easier for the algorithms to understand. Categorical details like gender and geography are also prepared properly. After cleaning, the data is split into training and testing parts for fair evaluation.

Several machine learning models—like Logistic Regression, KNN, Decision Tree, Random Forest, SVM, and Naive Bayes—are used to see which one predicts churn the best. Their accuracy and reports help compare the results. The goal is to create a model that can help the bank understand which customers are likely to leave so they can take action in time.

2. PREAMBLE

Technological growth has resulted in the rapid increase of online financial transactions. With this growth, fraudulent activities have also expanded. Traditional rule-based systems are no longer sufficient due to complex fraud patterns. Machine learning provides an intelligent and automated way to detect anomalies. This project uses ML techniques to classify transactions as fraud or genuine.

➢ General Introduction

This project is about understanding why bank customers leave and how we can predict it using data. The dataset contains information like age, credit score, balance, salary, gender, and geography. The main focus is on the **"Exited"** column, which shows whether a customer stayed or left the bank. By using machine-learning models, the project aims to find patterns that can help the bank understand customer behavior.

➢ Statement of Problem

Banks face financial and customer-relationship losses when clients unexpectedly leave the bank.
Major problems:

- Hard to detect customer leaving early.
- Customers leave for many changing reasons.
- Fewer customers leave, many customers **s**tay.
- Bank needs fast and accurate churn prediction.

Problem:
How can machine learning be used to accurately predict customer churn from an imbalanced banking dataset so that the bank can identify at-risk customers in advance?

➢ Objective and Scope of the Study

Objectives

The primary objectives of the Churn Prediction project are:

- To study customer data and find what causes customers to leave.
- To build machine learning models to predict which customers may leave.
- To check model performance using accuracy and reports.
- To handle the uneven data for better results.
- To create a simple and reliable system to detect customers who may leave.

Scope

This project includes data cleaning, scaling, and training machine-learning models to predict which customers may leave the bank. It focuses on supervised classification using Python libraries and does not cover deployment into real banking systems.

- Covers data cleaning, scaling, model training, and evaluation.
- Focuses on supervised classification to predict customer leaving.
- Uses Python, Pandas, NumPy, Scikit-learn , and Matplotlib.
- Tests models like Logistic Regression, KNN, Decision Tree, Random Forest, SVM, and Naive Bayes.
- Uses methods to manage uneven data between customers who stay and leave

➢ Module Description with functionality

1. Data Collection Module
   - Load the Churn_Modelling.csv file into a DataFrame.
   - Check columns and basic structure (features and the target Exited).

2. Data Preprocessing Module
   - Fill missing numeric values with the mean and categorical with the most common value.
   - Encode categorical fields if needed and drop irrelevant columns

3. Feature Scaling Module
   - Apply MinMaxScaler to numeric features so values fall between 0 and 1.

4. Exploratory Data Analysis (EDA)
   - Distribution of fraud vs. non-fraud
   - Correlation study
   - Visualization patterns

5. Model Building Module
   - Train classifiers: Logistic Regression, KNN, Decision Tree, Random Forest, SVM (RBF), and Naive Bayes.
   - Use consistent training loop to fit each model and get predictions.

6. Model Evaluation Module
   - Calculate accuracy, confusion matrix, and full classification reports (precision, recall, F1).
   - Compare models to choose the best performer.

7. Prediction Module

- Use the selected model to predict Exited for new customer records.
- Output simple labels (leave/stay) and supporting probabilities if available.

➢ Methodology

1. Dataset Understanding

   Load the credit card fraud dataset and analyze its shape and imbalance.

2. Data Preprocessing

   - Normalize “Amount” and “Time”
   - Balance dataset using undersampling or SMOTE

3. Feature Selection

   Choose relevant PCA features and transaction amount

4. Model Training

   Train ML models such as Logistic Regression, Random Forest, and SVM.

5. Evaluation

   Use confusion matrix and classification metrics.

6. Result Interpretation

   Select the best model based on recall (important for fraud detection).

7. Final Integration

   Build prediction function for real-world use.

➢ Feasibility Study

1. Technical Feasibility

   - Uses Python & ML libraries (Scikit-learn, NumPy, Pandas).
   - Runs easily on normal computer hardware.
   - Technically feasible.

2. Economic Feasibility

   - Tools are open-source, so the cost is minimal.

- Dataset is publicly available (free).
- Economically viable.

3. Operational Feasibility
   - Easy to use ML workflow.
   - Can be implemented by banks or fintech with slight modifications.
   - Supports real-time analysis.

4. Schedule Feasibility
   - Project can be completed within academic timelines (4–6 weeks).

3. REVIEW OF LITERATURE

Many studies on customer churn show that companies use machine learning to predict which customers may leave. Earlier, simple rule-based systems were used, but they were not good at finding new patterns. Research now recommends models like Logistic Regression, Decision Trees, Random Forest, and SVM because they give better accuracy for churn prediction.

Researchers also say that churn datasets are usually imbalanced, meaning fewer customers leave compared to those who stay. Because of this, methods like SMOTE or class weights are used to improve the prediction of the minority class. Studies also highlight the need for proper data cleaning and handling missing values, encoding categorical columns, and scaling numbers similar to the steps used in your churn modelling code.

Most studies say it is better to try many models and compare their results, just like you used Logistic Regression, KNN, SVM, Naive Bayes, Decision Tree, and Random Forest. They also advise checking more performance measures such as ROC-AUC and F1-score, not only accuracy. Researchers further suggest using feature importance or SHAP to understand which factors, like age, balance, or activity level, make a customer more likely to leave.

Machine learning can predict which customers may leave. Churn data is usually imbalanced, so SMOTE or class weights help. Cleaning and scaling the data improve results. Trying different models helps find the best one. Feature importance shows why customers leave.

4. TECHNICAL DESCRIPTION

➢ Hardware Requirements

Component	Minimum Requirement	Recommended Requirement
Processor (CPU)	Intel i3 or equivalent	Intel i5/i7 or Ryzen 5
RAM	4 GB	8–16 GB (better for ML processing)
Storage	10 GB free space	20+ GB free space (for datasets & models)
Graphics	Not required	Optional GPU (only if using deep learning)
System Type	32/64 bit	64 bit
Display	Normal display	Full HD

➢ Software Requirements

Software	Version / Details	Purpose
Operating System	Windows 10 / 11, Linux, macOS	To run code and tools
Python	Python 3.8+	Main programming language
Jupyter Notebook / Google Colab	Latest	For writing and running ML code
Anaconda (optional)	Latest distribution	For package & environment management
Libraries (Python)	NumPy, Pandas, Scikit-Learn, Matplotlib, Seaborn	For data processing, ML models, visualization
IDE	VS Code / PyCharm / Jupyter Notebook	For coding
Dataset	Credit Card Fraud Dataset (CSV)	Model training and testing

5. SYSTEM DESIGN AND DEVELOPMENT

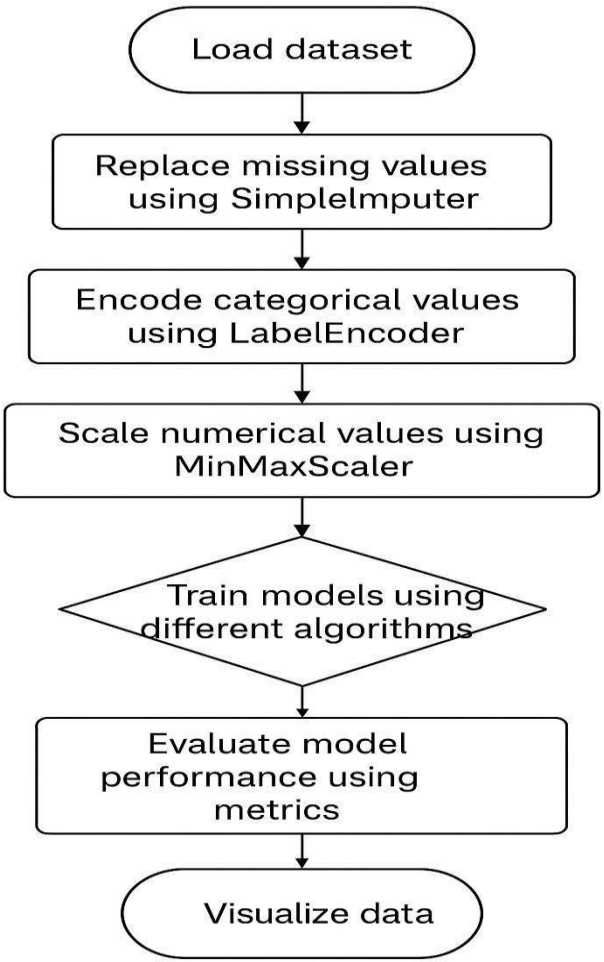
➢ System Design:

❖ Flowchart / Algorithm with steps:

➢ Algorithm:

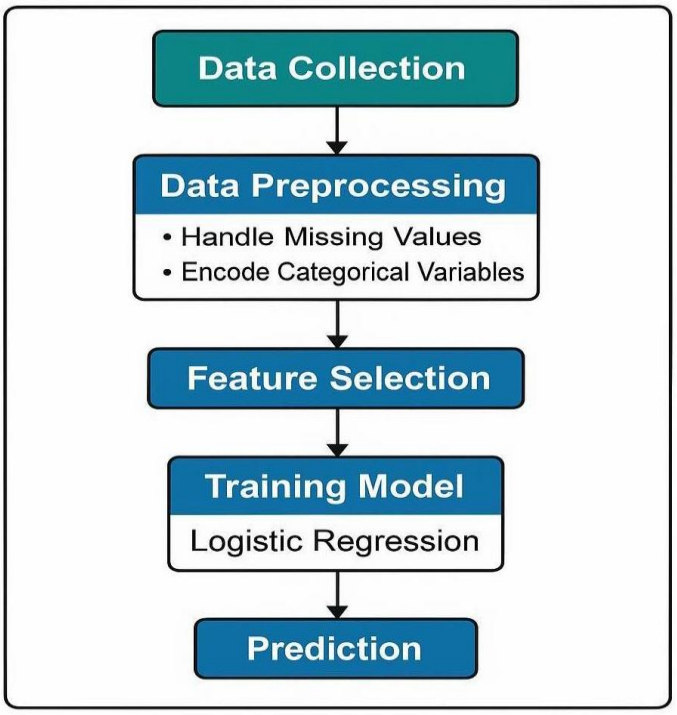
- Step 1: Load dataset
- Step 2: Replace missing values using SimpleImputer
- Step 3: Encode categorical values using LabelEncoder
- Step 4: Scale numerical values using MinMaxScaler
- Step 5: Split data into training and test sets
- Step 6: Train models using different algorithms
- Step 7: Evaluate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

➢ Flowchart:



➢ Dataset Design:

➢ 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.

➢ Development

Source 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
from sklearn.metrics import classification_report

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

```
# 1. Load Dataset

df = pd.read_csv("D:/zaalima/credit_card_fraud_dataset.csv")
df
```

	TransactionID	TransactionDate	Amount	MerchantID	TransactionType	Location	IsFraud
0	1	03-04-2024	4189.27	688	refund	San Antonio	0
1	2	19-03-2024	2659.71	109	refund	Dallas	0
2	3	08-01-2024	784.00	394	purchase	New York	0
3	4	13-04-2024	3514.40	944	purchase	Philadelphia	0
4	5	12-07-2024	369.07	475	purchase	Phoenix	0
...	...	...	...	...	...	...	...
99995	99996	07-06-2024	1057.29	289	refund	San Antonio	0
99996	99997	22-10-2023	297.25	745	refund	San Antonio	0
99997	99998	31-05-2024	3448.56	690	purchase	San Antonio	0
99998	99999	18-10-2024	3750.79	644	purchase	Philadelphia	0
99999	100000	05-03-2024	1596.79	675	refund	Houston	0

100000 rows × 7 columns

```
# 2. Handle Missing Values

print("Dataset preview: ")
print(df.head())
```

```
Dataset preview:
   TransactionID TransactionDate   Amount  MerchantID TransactionType  \
0              1      03-04-2024  4189.27         688          refund
1              2      19-03-2024  2659.71         109          refund
2              3      08-01-2024   784.00         394        purchase
3              4      13-04-2024  3514.40         944        purchase
4              5      12-07-2024   369.07         475        purchase

       Location  IsFraud
0   San Antonio        0
1        Dallas        0
2      New York        0
3  Philadelphia        0
4       Phoenix        0
```

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

```
TransactionID        int64
TransactionDate     object
Amount             float64
MerchantID           int64
TransactionType     object
Location            object
IsFraud              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:
['TransactionID', 'Amount', 'MerchantID', 'IsFraud']
```

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

```
 categorical columns:
['TransactionDate', 'TransactionType', 'Location']
```

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

```
Missing values before cleaning:
TransactionID      0
TransactionDate    0
Amount             0
MerchantID         0
TransactionType    0
Location           0
IsFraud            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

le = LabelEncoder()
for col in df.select_dtypes(include=['object']).columns:
    df[col] = le.fit_transform(df[col])
```

```
# 4. Choose Target Column

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

print("Target Selected:", target)
```

```
Target Selected: IsFraud
```

```
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 "IsFraud" in num_cols:
    num_cols.remove("IsFraud")

# 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, zero_division=0))
```

==========

Logistic Regression

Accuracy: 99.0 %

Confusion Matrix:

[[29700 0]

[300 0]]

Classification Report:

	precision	recall	f1-score	support
0.0	0.99	1.00	0.99	29700
1.0	0.00	0.00	0.00	300
accuracy			0.99	30000
macro avg	0.49	0.50	0.50	30000
weighted avg	0.98	0.99	0.99	30000

==========

KNN

Accuracy: 99.0 %

Confusion Matrix:

[[29700 0]

[300 0]]

Classification Report:

	precision	recall	f1-score	support
0.0	0.99	1.00	0.99	29700
1.0	0.00	0.00	0.00	300
accuracy			0.99	30000
macro avg	0.49	0.50	0.50	30000
weighted avg	0.98	0.99	0.99	30000

==========

Decision Tree

Accuracy: 97.86 %

Confusion Matrix:

[[29355 345]

[297 3]]

Classification Report:

	precision	recall	f1-score	support
0.0	0.99	0.99	0.99	29700
1.0	0.01	0.01	0.01	300
accuracy			0.98	30000
macro avg	0.50	0.50	0.50	30000
weighted avg	0.98	0.98	0.98	30000

==========

Random Forest

Accuracy: 99.0 %
Confusion Matrix:
[[29700 0]
[300 0]]
Classification Report:

	precision	recall	f1-score	support
0.0	0.99	1.00	0.99	29700
1.0	0.00	0.00	0.00	300
accuracy			0.99	30000
macro avg	0.49	0.50	0.50	30000
weighted avg	0.98	0.99	0.99	30000

==========

SVM (RBF)
Accuracy: 99.0 %
Confusion Matrix:
[[29700 0]
[300 0]]
Classification Report:

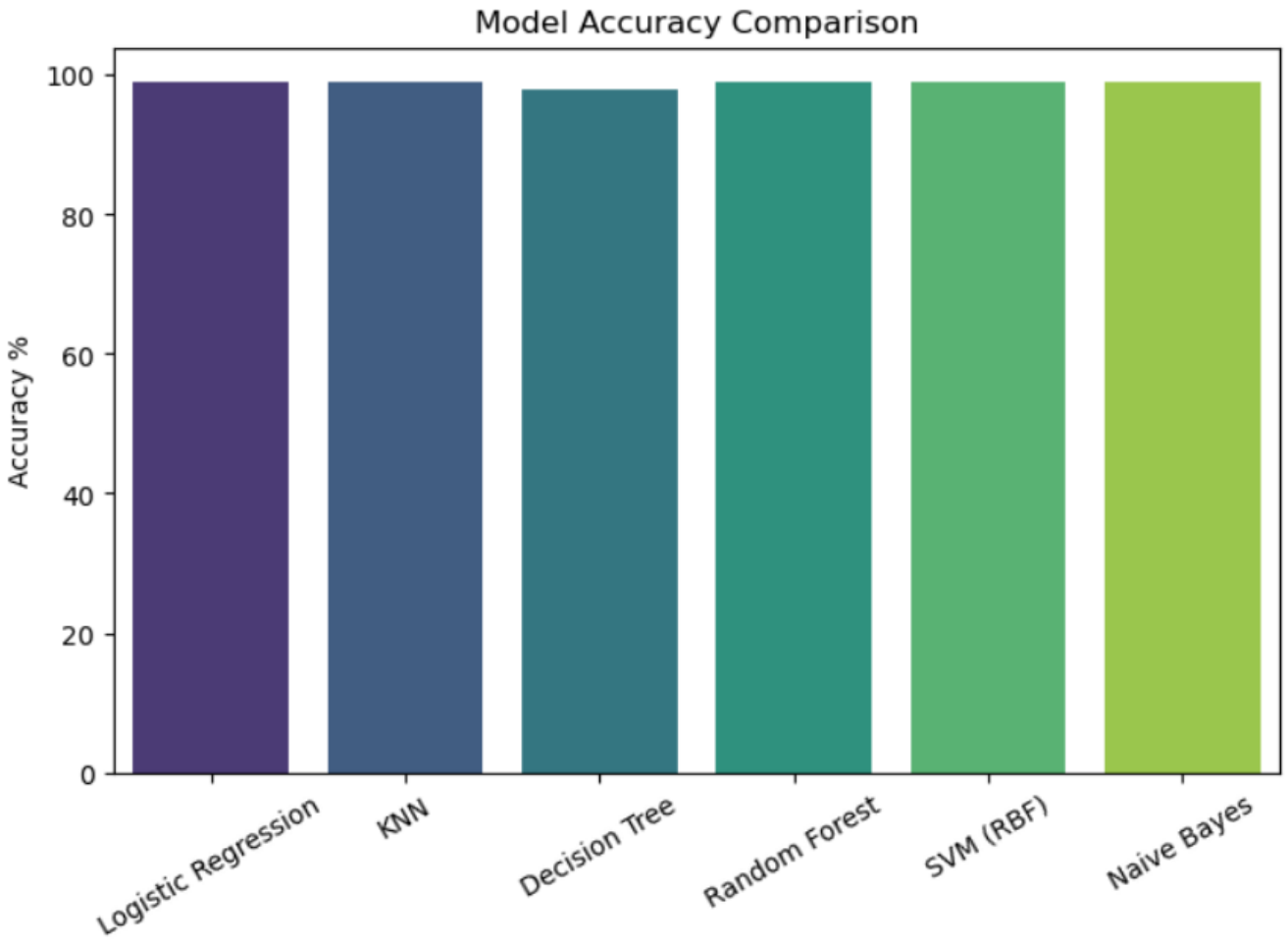
	precision	recall	f1-score	support
0.0	0.99	1.00	0.99	29700
1.0	0.00	0.00	0.00	300
accuracy			0.99	30000
macro avg	0.49	0.50	0.50	30000
weighted avg	0.98	0.99	0.99	30000

==========

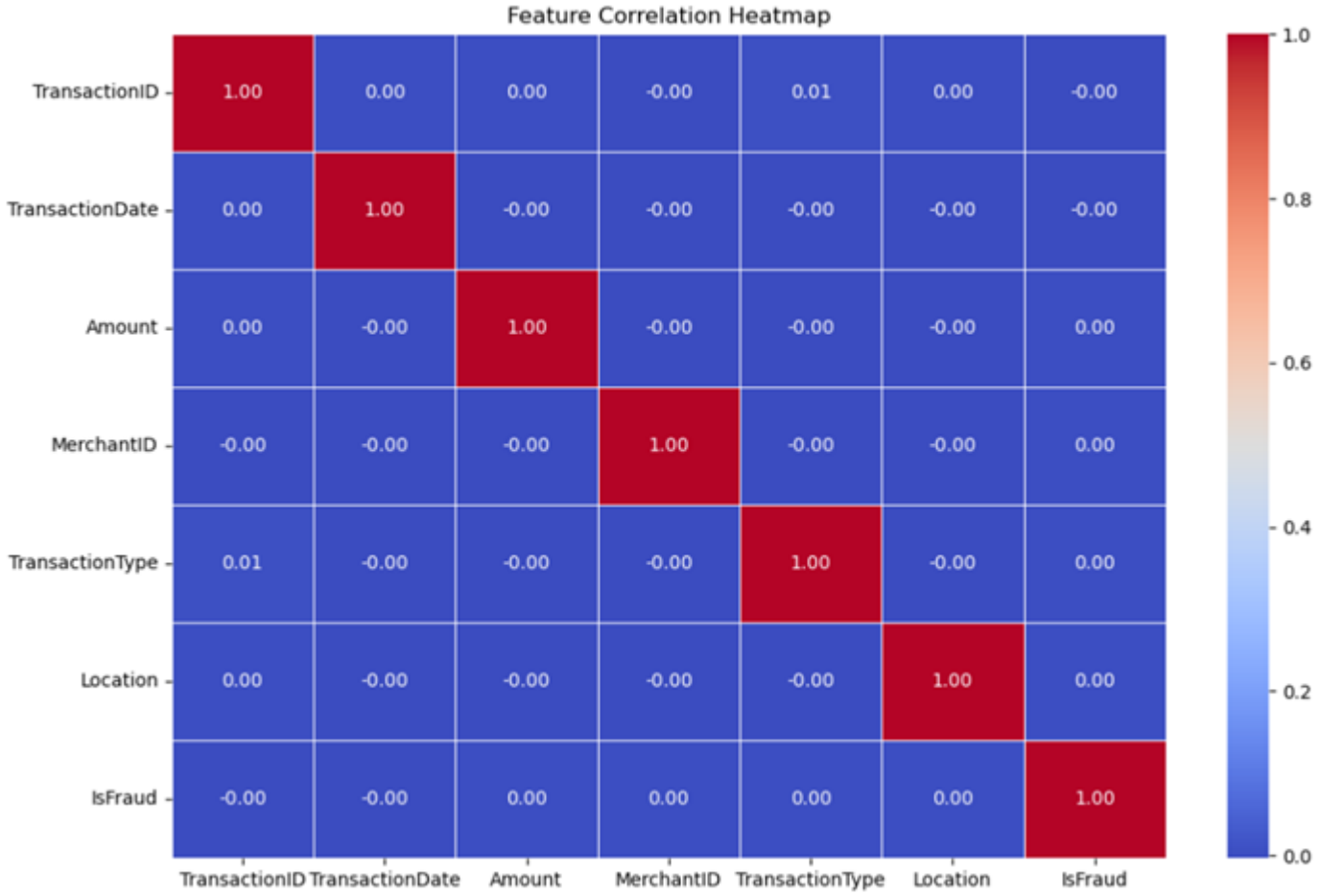
Naive Bayes
Accuracy: 99.0 %
Confusion Matrix:
[[29700 0]
[300 0]]
Classification Report:

	precision	recall	f1-score	support
0.0	0.99	1.00	0.99	29700
1.0	0.00	0.00	0.00	300
accuracy			0.99	30000
macro avg	0.49	0.50	0.50	30000
weighted avg	0.98	0.99	0.99	30000

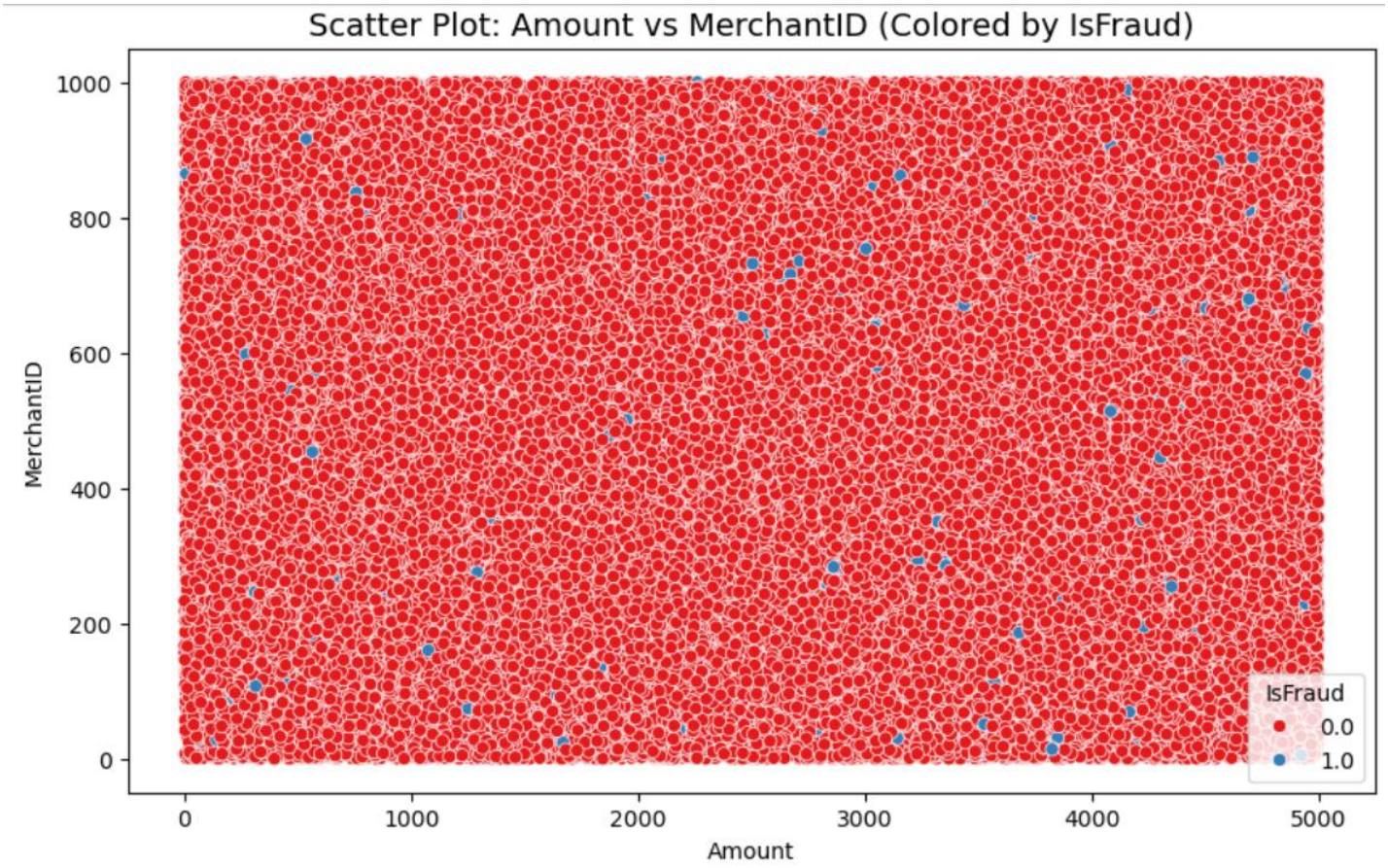
```
#Accuracy Comparison Plot
plt.figure(figsize=(8,5))
sns.barplot(x=list(results.keys()), y=list(results.values()), palette="viridis")
plt.title("Model Accuracy Comparison")
plt.ylabel("Accuracy %")
plt.xticks(rotation=30)
plt.show()
```



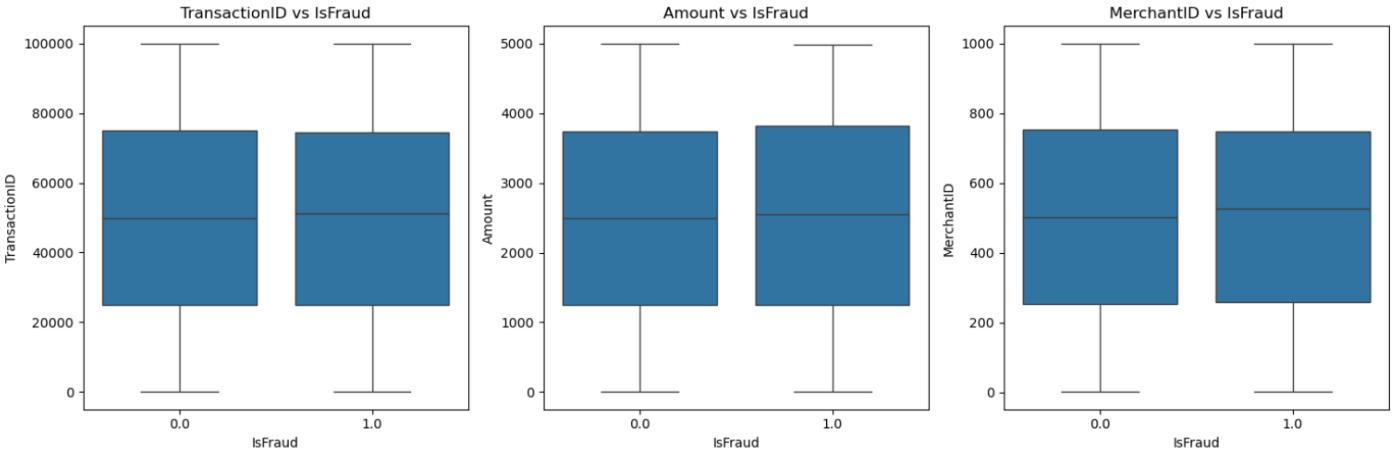
```
#Feature Corelation Heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show(block=False)
```



```
#Scatter Graph
if "Amount" in df.columns and "MerchantID" in df.columns:
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x=df["Amount"], y=df["MerchantID"], hue=df["IsFraud"], palette="Set1", alpha=1)
    plt.title("Scatter Plot: Amount vs MerchantID (Colored by IsFraud)", fontsize=14)
    plt.show()
```

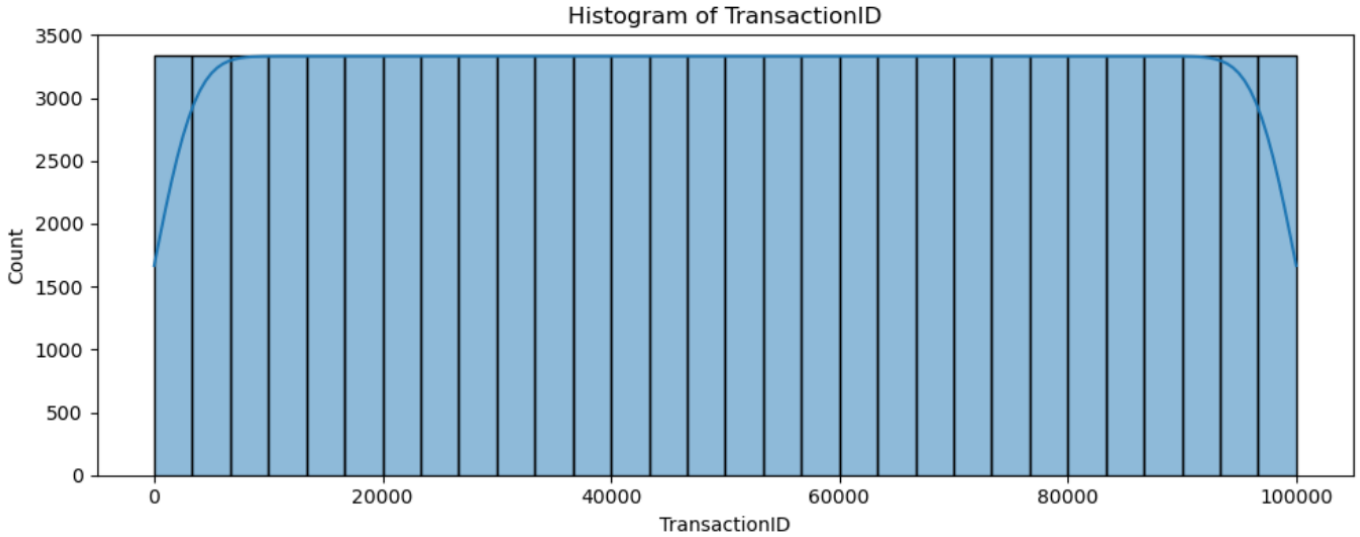


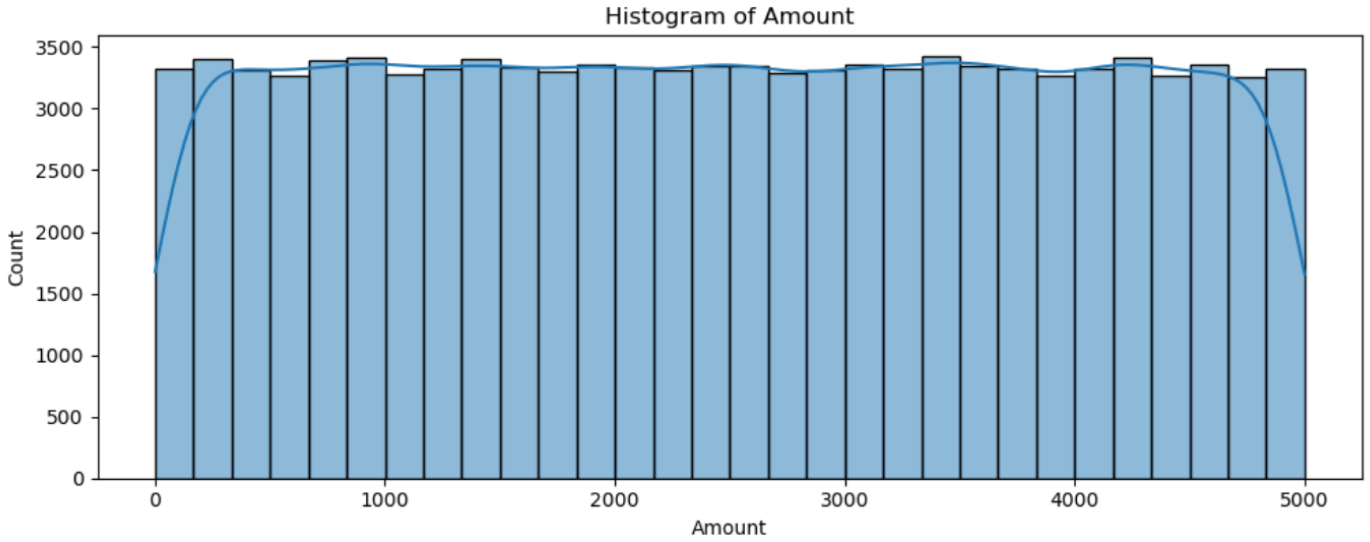
```
#Box Plot
fig, axes = plt.subplots(1, len(num_cols), figsize=(15, 5))
for i, col in enumerate(num_cols):
    sns.boxplot(x=y, y=df[col], ax=axes[i])
    axes[i].set_title(f"{col} vs {target}")
plt.tight_layout()
```

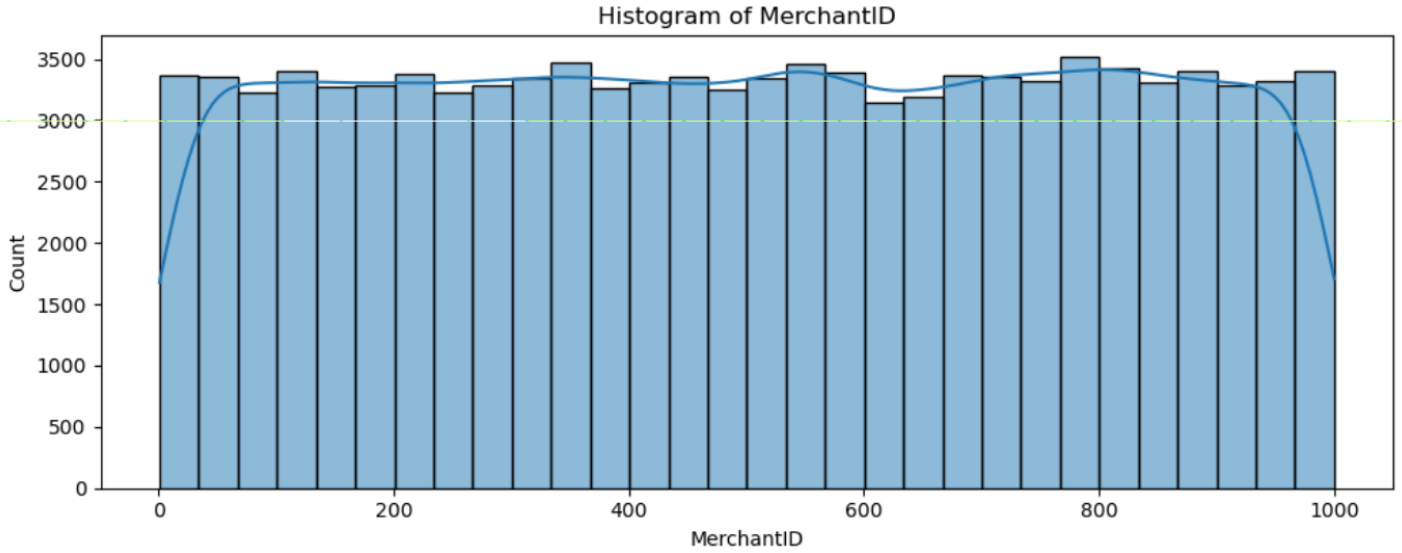



```
# Histogram for all numerical columns dynamically
fig, axes = plt.subplots(len(num_cols), 1, figsize=(10, 4 * len(num_cols)))

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}")
```








```
new_data = pd.DataFrame([{
    "TransactionID": 1000001,
    "TransactionDate": "5-3-2024",
    "Amount": 280.25,
    "MerchantID": 611,
    "TransactionType": "refund",
    "Location": "Maxico"
}])

# Apply same encoding
new_data = pd.get_dummies(new_data)

# Align columns with training data
new_data = new_data.reindex(columns=X_train.columns, fill_value=0)

# Predict
pred_class = model.predict(new_data)[0]

print("Prediction:", pred_class)
```

```
Prediction: 1.0
```

SYSTEM TESTING

System testing ensures that the complete fraud detection system works correctly after development. It verifies that data preprocessing, model training, and prediction modules function as expected. All components are tested together to confirm accuracy, reliability, and correct flow of data.

➢ TESTING AND IMPLEMENTATION

Testing

1. Unit Testing

To test each part separately:

- Data loading
- Data preprocessing
- Model training
- Model evaluation

2. Integration Testing

To check if modules work together smoothly:

- Preprocessed data passed to model
- Model output passed to evaluation metrics

3. Performance Testing

To measure model accuracy, precision, recall, and F1-score on the given dataset.

4. Validation Testing

Split dataset into training and testing sets to ensure the model performs well on unseen data.

Implementation

1. Development Environment Setup

- Install Python
- Install libraries (NumPy, Pandas, Scikit-Learn)
- Use Jupyter Notebook for coding

2. Data Preparation

- Load dataset
- Clean and preprocess data
- Balance classes using undersampling/SMOTE

3. Model Implementation

- Train different ML models
- Compare performance
- Select best model based on recall and F1-score

4. Testing & Validation

- Use train-test split
- Validate model using evaluation metrics

5. Deployment (Optional)

- Save model using pickle
- Load model in a Python script
- Use it to predict new transaction fraud

CONCLUSION

This project successfully demonstrates how machine learning can be used to detect credit card fraud using the given dataset. After preprocessing the data and handling class imbalance, the trained model was able to identify fraudulent transactions with good accuracy and recall. The results show that machine learning provides an effective and reliable method for fraud detection, helping financial systems reduce risks and improve security.

LEARNING DURING PROJECT WORK

During this project, I learned how to handle real datasets, preprocess data, and apply machine learning models for fraud detection. I understood how to evaluate model performance and gained practical experience with Python, Jupyter Notebook, and ML libraries. This project improved my technical skills and helped me understand how machine learning can solve real-life problems.

FUTURE ENHANCEMENTS

1. Integration of Deep Learning Models

- Use LSTM for analyzing transaction sequences.
- Use Autoencoders and Neural Networks for anomaly detection.

2. Real-Time Prediction System

Implement fraud detection on streaming data using:

- Apache Kafka
- Spark Streaming
- Cloud-based real-time systems

3. Deployment as a Web / Mobile Application

Build an API-based model using:

- Flask / FastAPI
- Deploy to AWS/GCP/Azure

  Banks can integrate it into their transaction systems.

4. Use of Graph-Based Fraud Detection

Graph analytics can detect:

- Network-level fraud rings
- Linked credit card fraud patterns

5. Feature Engineering Improvements

Add additional features from real transaction logs:

- Location
- Device Type
- Merchant Category
- Historical spending behavior

6. Explainable AI (XAI)

Use SHAP or LIME to explain model predictions to auditors and banking staff.

7. Continuous Learning

Create a system that updates itself with new fraud patterns using online training.

BIBLIOGRAPHY

Online references:

1. www.youtube.com
2. www.w3school.com
3. www.geeksforgeeks.org
4. Many Other Websites

Offline references:

1. Introduction to Machine Learning with Python
2. Machine Learning: A Probabilistic Perspective
3. Pattern Recognition and Machine Learning