CREDIT CARD FRAUD DETECTION USING MACHINE LEARNING

Submitted By

Saurabh Chaturvedi

Data Science Intern

Under the Guidance of

Shreya Mohan

Project Manager

Internship Duration

4 Weeks (From 2nd July 2025 to 29th July 2025)

Submission Date

31st July 2025

Organization

Outsource360.in

Table of Contents

3. Project Title and Objective

Project Title: Credit Card Fraud Detection Using Machine Learning and Power BI

Objective:

The main goal of this project is to detect fraudulent credit card transactions using machine learning techniques. Fraud in online transactions is a major issue, and it can lead to huge financial losses. This project aims to analyze past transaction data, identify key patterns in fraudulent behavior, and create predictive models that can classify future transactions as either fraud or non-fraud.

We also use Power BI to create a clear and interactive dashboard for presenting the results, helping both technical and non-technical users understand the insights.

4. Project Overview and Problem Statement

Background:

Credit card fraud is a serious issue in today's digital world. With the rise of online transactions, fraudsters are finding new ways to steal money. Banks and financial institutions need better tools to detect and stop fraud before it causes damage.

Problem Statement:

It is difficult to detect fraudulent transactions because fraud patterns keep changing and are often hidden among thousands of regular transactions. The number of fraud cases is small compared to normal ones, making it hard for basic systems to identify them correctly.

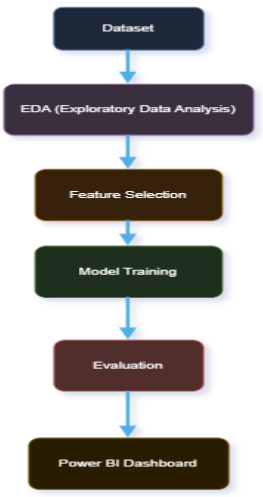
Why This is Important:

Detecting fraud early helps protect people's money and builds trust in digital payments. A smart fraud detection system can save time, reduce financial loss, and improve customer safety.

Our Approach:

In this project, we used machine learning to detect fraud using past transaction data. We cleaned the data, selected the most useful features, trained models, tested their performance, and built a Power BI dashboard to present the findings.

Workflow Diagram:



5. Tools & Technologies Used

- Python – for data processing and modeling
- Jupyter Notebook – as the coding environment
- Pandas, Matplotlib, Seaborn – for data handling and visualization
- Scikit-learn – for building and evaluating machine learning models
- Imbalanced-learn (SMOTE) – to handle imbalanced data
- Power BI – for creating an interactive dashboard
- CSV File Format – used for data import and export

---

6. Dataset Description

The dataset used for this project was provided by the **internship company (Outsource360)**. It contains real-world **credit card transaction records** collected over a period of two days.

- **Total Rows:** 284,807
- **Total Columns:** 31

Key Features:

- Time: The number of seconds that have passed since the first transaction in the dataset.
- Amount: The amount of money involved in the transaction.
- V1 to V28: These are **anonymized numerical features** generated using **PCA (Principal Component Analysis)** to protect customer privacy.
- Class: This is the **target column**, where
  0 = **Legitimate transaction**
  1 = **Fraudulent transaction**

Important Point:

The dataset is **highly imbalanced** – only about **0.17%** of the transactions are fraudulent. This imbalance makes it challenging for models to detect fraud accurately and requires special handling during model training.

Sample Feature Table:

Feature Name	Description
Time	Seconds since first transaction
Amount	Transaction amount
V1-V28	PCA-transformed features
Class	0 = Legit, 1 = Fraud

7: Week-by-Week Summary

7.1: Week 1: Data Loading, Exploration, and Feature Engineering

During Week 1, we began by understanding and preparing the dataset for further analysis and model building. Here's a breakdown of the steps we performed:

---

1. Imported Required Libraries

We loaded essential Python libraries like **pandas, matplotlib, seaborn**, and **plotly** for data analysis and visualization. These tools helped us inspect the dataset and plot graphs.

---

2. Loaded the Dataset

We used the **.csv** file provided by the company. It contained transaction data including anonymized features (**V1 to V28**), time, amount, and a target label (**Class**) indicating fraud (1) or not (0).

---

3. Explored Basic Information

We checked:

- The shape (rows and columns)
- Data types of each column
- Summary statistics like mean, min, max, etc.

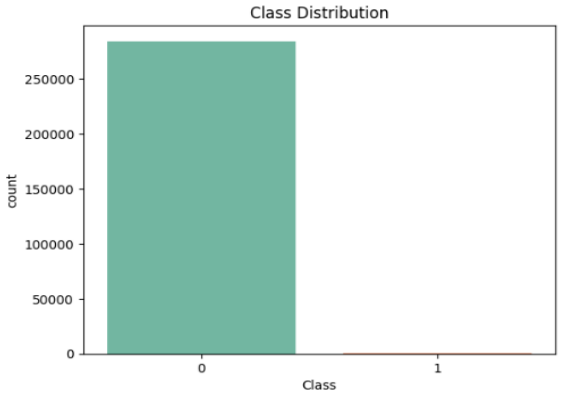
This helped us understand the overall structure of the data.

---

4. Analyzed Class Distribution

We checked how many transactions were fraudulent vs. non-fraudulent. We noticed that the

data was **imbalanced**, with very few fraud cases compared to legitimate ones. We visualized this using a bar plot to make the imbalance clear.



---

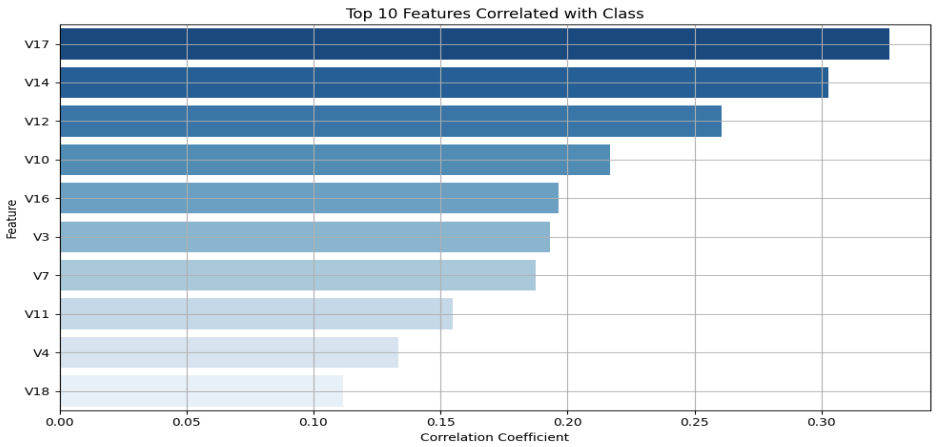
5. Checked for Duplicate Rows

We identified and counted any duplicate entries in the dataset. Removing these ensured data quality and prevented bias.

---

6. Correlation Analysis

We analyzed which features (**columns**) had the strongest relationship with the target variable (**Class**). This helped us identify the most informative features for fraud detection.



7. Performed Feature Engineering

We added two new columns:

- **Hour**: Extracted from the **Time** column to show the hour of each transaction.
- **Log_Amount**: Applied logarithmic scaling to the **Amount** column to reduce skewness in the values.

---

1. **Saved Cleaned Dataset for Next Week**

   Finally, we dropped any duplicates and saved the cleaned dataset into a new file (**processed_data.csv**) for use in Week 2.

7.2: Week 2: Feature Selection, Visualization, and Train-Test Split

1. Loaded the Cleaned Dataset

In this step, we loaded the dataset that was cleaned and saved during Week 1. The file named "processed_data.csv" contains features such as transaction time, amount (log transformed), and anonymized features (V1 to V28), along with the target column 'Class', which indicates whether a transaction is fraudulent (1) or not (0). Used: **pandas** to read the CSV file.

---

2. Separating Features (X) and Target (y)

Font Style: Font Size – 12, Bold

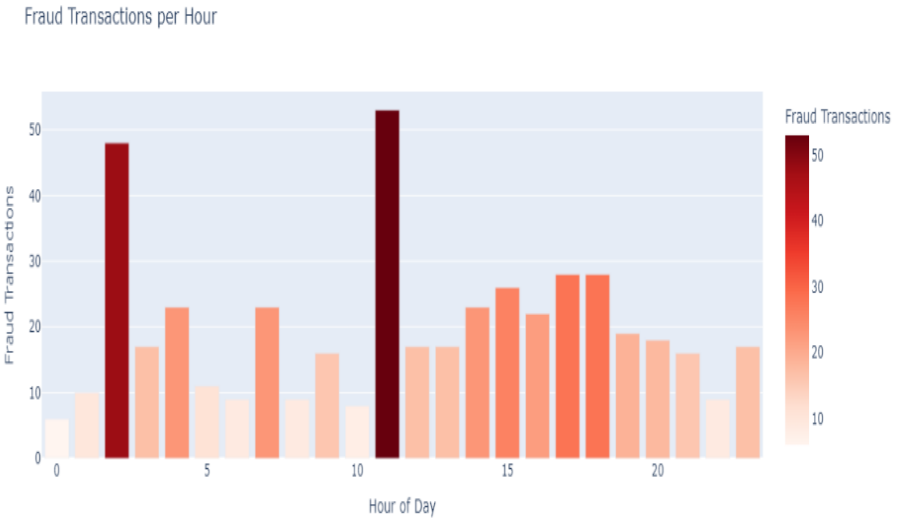
We split the dataset into two parts:

- **X** → All columns except 'Class', representing input features.
- **y** → The 'Class' column, representing the target label (fraud or not).

---

3. Fraud Transactions Per Hour

We extracted the hour from the 'Time' column (originally in seconds) to determine when most frauds happen during the day. This helped us understand at what times frauds are more frequent.

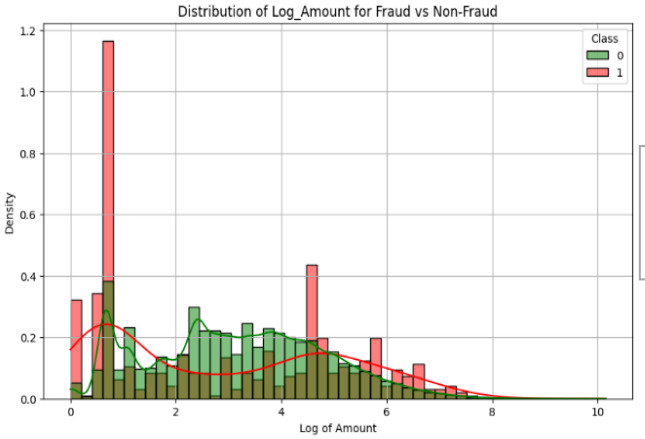


Bar Chart – Fraud Transactions per Hour

- **X-axis**: Hour of the Day (0 to 23)
- **Y-axis**: Number of Fraud Transactions
- **Color:** Red (to represent fraud)

- **2. Visualized Class Distribution Again**

  We verified the imbalance in our dataset—only a small percentage of transactions were frauds (**Class = 1**).



Histogram (Log_Amount by Class):
- **X-axis:** Log of Transaction Amount
- **Hue:** Class (0 = Legit, 1 = Fraud)
- **Y-axis:** Density

We used: **seaborn.countplot()** to visualize the number of fraud vs. non-fraud transactions.

Why important?

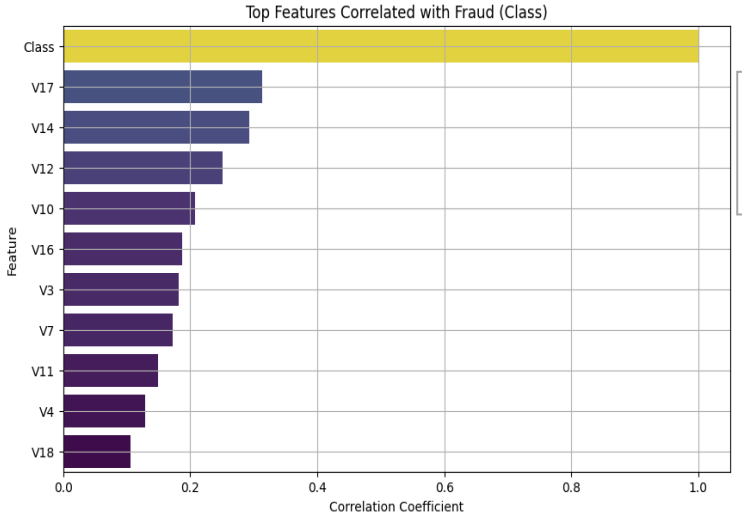
This helps us decide on strategies like SMOTE to handle imbalance in later steps.

---

3. Checked Correlation of Features with Class

We calculated the correlation between each feature and the target column (**Class**).

This showed which features are more important for predicting fraud.



Bar Chart – Top Correlated Features
- **X-axis:** Correlation Coefficient
- **Y-axis:** Feature Names
- **Color:** Viridis palette

Used: **df.corr()['Class'].abs().sort_values()** to get a list of top correlated features.

Why important?

It helps us identify strong predictors without training a model.

4. Selected Top Correlated Features

From the correlation output, we selected the top 10 features that had the highest correlation with fraud.

Example: V14, V10, V12, V17, V16, V18, V7, V4, V3, and **V11**.

Why important?

Reducing the number of features can simplify the model and improve performance.

---

5. Feature Selection Using SelectKBest

We applied a method called **SelectKBest** from **sklearn.feature_selection** with **f_classif** (ANOVA F-test).

It selects the top features that are statistically most related to the target.

Used: **SelectKBest(score_func=f_classif, k=10)**

Fit and transform data to get the 10 best features.

Why important?

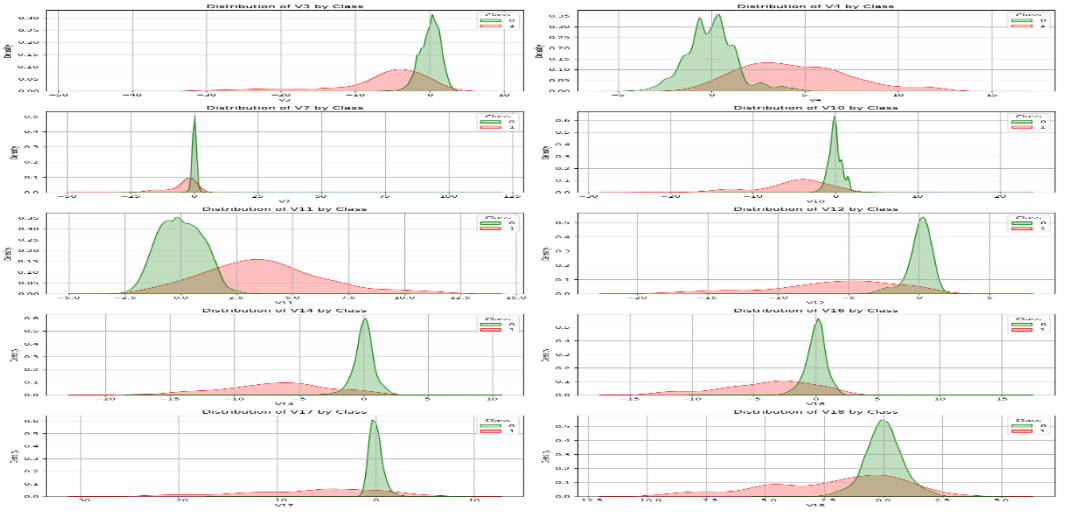
It offers a different technique from correlation and ensures we pick statistically strong features.

---

6. Combined Final Selected Features

We merged the top 10 features from both correlation and **SelectKBest** methods (with **Hour** and **Log_Amount**).

This resulted in the final list of **13 important features** used for model training.



Why important?

Combining both techniques helps capture the best of both statistical and correlation-based selection.

7. Splitted Data into Features and Target

We separated:

X = feature columns

y = target column (**Class**)

Why important?

This is a required step before splitting the dataset for training.

---

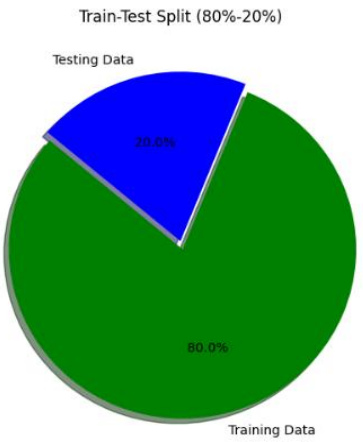
8. Train-Test Split

We used **train_test_split** from **sklearn.model_selection** to divide data into:

Training set (80%)

Testing set (20%)

Random state was set for reproducibility.



Why important?

We train the model on one part of the data and test it on another to avoid overfitting.

---

2. **Saved the Processed Dataset**

The final selected features dataset was saved as **cleaned_selected_features_dataset.csv**

This file was used in Week 3 for model training.

7.3: Week 3: Model Building and Evaluation

---

1. Loading the Cleaned Dataset

In this step, we loaded the cleaned dataset that we had prepared in Week 2. This file contains only the selected important features needed for modeling. We used a function to read the (**cleaned_selected_features_dataset.csv**). file and then printed the shape (rows and columns) to verify it loaded correctly.

We also previewed the top few rows to confirm that the data was clean and properly formatted.

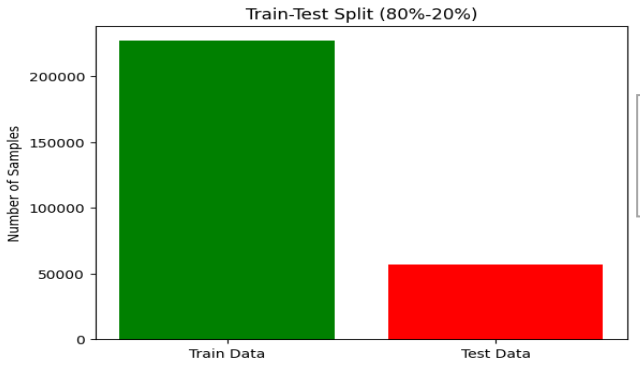
---

2. Train-Test Split (80%-20%)

To build and evaluate our models, we split the data into two parts:

- **Training Data (80%)**: Used to train the machine learning models.
- **Testing Data (20%)**: Used to check how well the model performs on unseen data.

We used a method that ensures the balance between fraud and non-fraud cases remains the same in both sets. This is important because fraud cases are rare, and we don't want the model to become biased.



Graph: Bar chart showing:
- Training Data – 80% (green)
- Testing Data – 20% (blue)

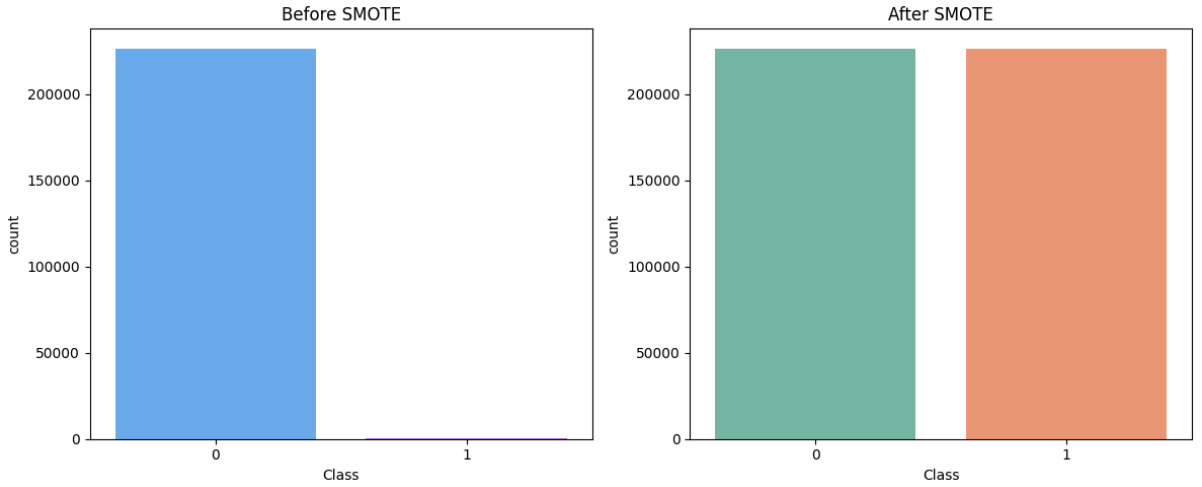
---

3. Handling Imbalanced Data using SMOTE

In fraud detection, the number of fraud cases is very small compared to normal ones. This imbalance can confuse the model. To solve this, we used a technique called **SMOTE (Synthetic Minority Over-sampling Technique)**.

SMOTE creates synthetic examples of the minority class (fraud) so that the model sees more fraud cases during training. This helps improve performance.

We checked the class counts before and after applying SMOTE. The fraud and non-fraud samples became balanced



Bar chart to show before and after SMOTE

- **X-axis:** Class labels (0 = Legit, 1 = Fraud)
- **Y-axis:** Number of samples

---

4. Train 3 Different Models

We trained the following classification models using the balanced training data:

- **Logistic Regression**: A basic statistical model for binary classification.
- **Decision Tree Classifier**: A tree-based model that splits data into nodes based on feature values.
- **Random Forest Classifier**: An ensemble of multiple decision trees to improve accuracy and prevent overfitting.

Each model was trained separately on the same training set, and then used to make predictions on the test set.

Cleaned Data → SMOTE → Model 1, Model 2, Model 3

---

5. Prediction on Test Set

After training, each model was used to make predictions on the 20% test data. These predictions were later used to evaluate how well the models perform in detecting fraud.

---

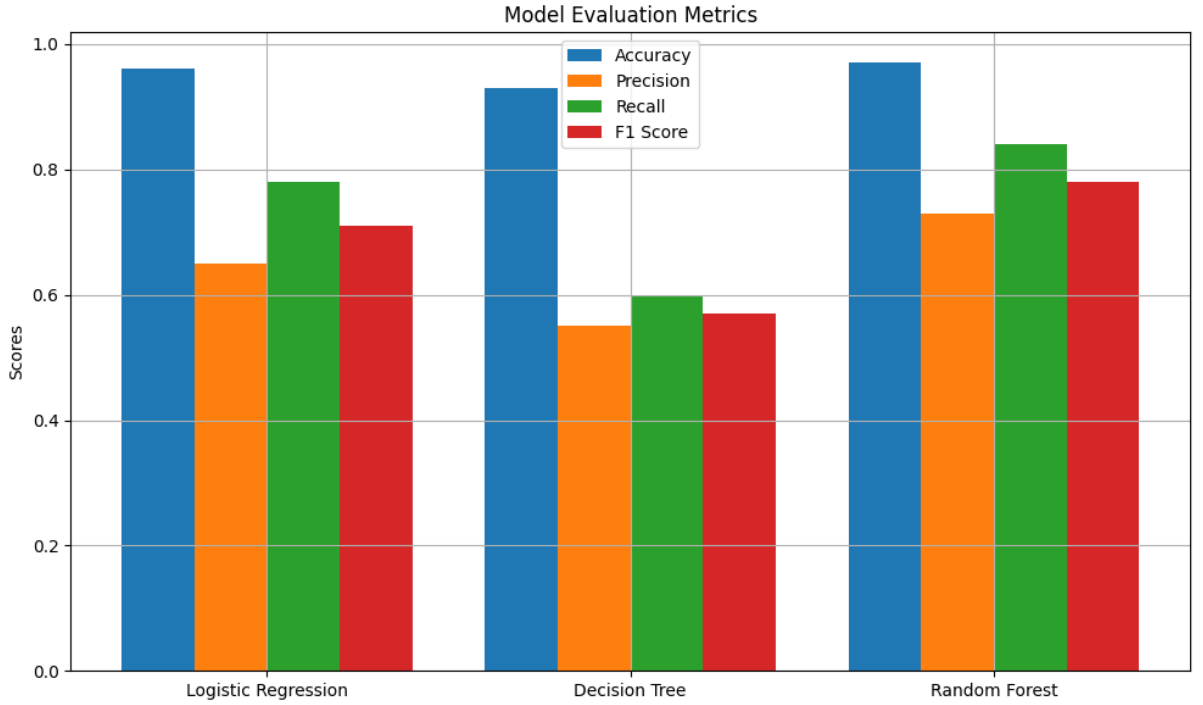
5. Evaluate Models using Classification Metrics

To understand how well each model performs, we used the following metrics:

- **Accuracy**: How many predictions were correct overall.

- **Precision**: Of all the predicted frauds, how many were actually fraud.
- **Recall**: Of all actual frauds, how many were correctly predicted.
- **F1-Score**: Balance between precision and recall.

We used **classification_report()** to generate these metrics for each model.



Graph: Bar chart to compare models

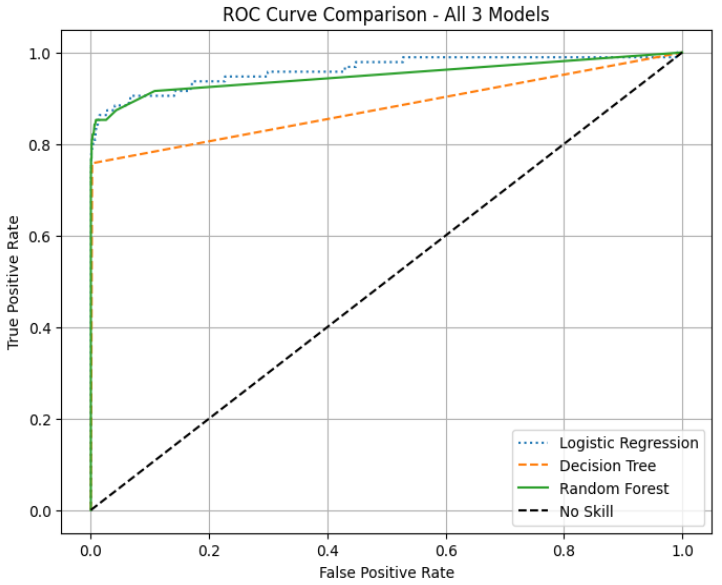
- **X-axis:** Model names
- **Y-axis:** Score values (0 to 1)
- Each model has 4 bars (Accuracy, Precision, Recall, F1)

---

6. Plot ROC Curves and Calculate AUC Score

We plotted the **ROC (Receiver Operating Characteristic) Curve** to understand how well each model can separate fraud from non-fraud. The **AUC (Area Under Curve)** score shows the model's ability to correctly classify fraud.

- Higher AUC means better performance.
- The ROC curve shows True Positive Rate vs. False Positive Rate.



ROC Curve with AUC in legend.
Logistic Regression – Blue dotted line
Decision Tree – Orange dashed line
Random Forest – Green solid line

This graph makes it easy to compare all three models visually.

---

8. Save Model Results for Power BI

(Font Style: Font Size – 12, Bold)

Finally, we selected the **Random Forest model** and used it to predict fraud on the test data. We stored:

- The actual class
- The predicted class
- The probability of fraud

This data was saved to a file named **rf_predictions_results.csv**. This file is used in **Week 4** to create the Power BI dashboard and visuals.

7.4: Week 4 – Power BI Dashboard and Reporting

Step-by-Step Process for Creating the Final Dashboard

---

1. Import Data into Power BI

- The .csv file named rf_predictions_results.csv was imported into Power BI.
- This file contains the final prediction results from the Random Forest model.
- The dataset includes predicted labels, actual labels, probabilities, and other related columns.

Purpose: This step brings the cleaned prediction data into Power BI for visualization.

---

2. Data Cleaning and Formatting in Power BI

- Inside Power BI, we performed basic data formatting tasks such as:
- Setting proper data types (e.g., categorical, numerical, datetime).
- Renaming columns for better readability.
- Verifying that columns like Class, Predicted, and Probability are in the correct format.

Purpose: Ensures the data is clean and ready for dashboard design.

---

3. Create DAX Measures (KPIs)

We created several **DAX measures** to calculate key performance indicators:

- ✓ Total Fraud Transactions
- ✓ Total Legit Transactions
- ✓ Accuracy
- ✓ Precision
- ✓ Recall
- ✓ F1-Score

Purpose: These KPIs help users quickly understand how well the model performed.

---

4. Build Visuals (Charts, Tables, Filters)

We added the following **interactive visuals** in the dashboard:

Visual Type	Description
Bar Chart	Showed fraud vs legit transaction count
Donut Chart	Percentage distribution of predicted labels
Confusion Matrix (as image or visual table)	To represent true/false positives/negatives
Slicers/Filters	For filtering by actual class or prediction result

Visual Type	Description
KPI Cards	Showed Accuracy, Precision, Recall, F1-score as tiles

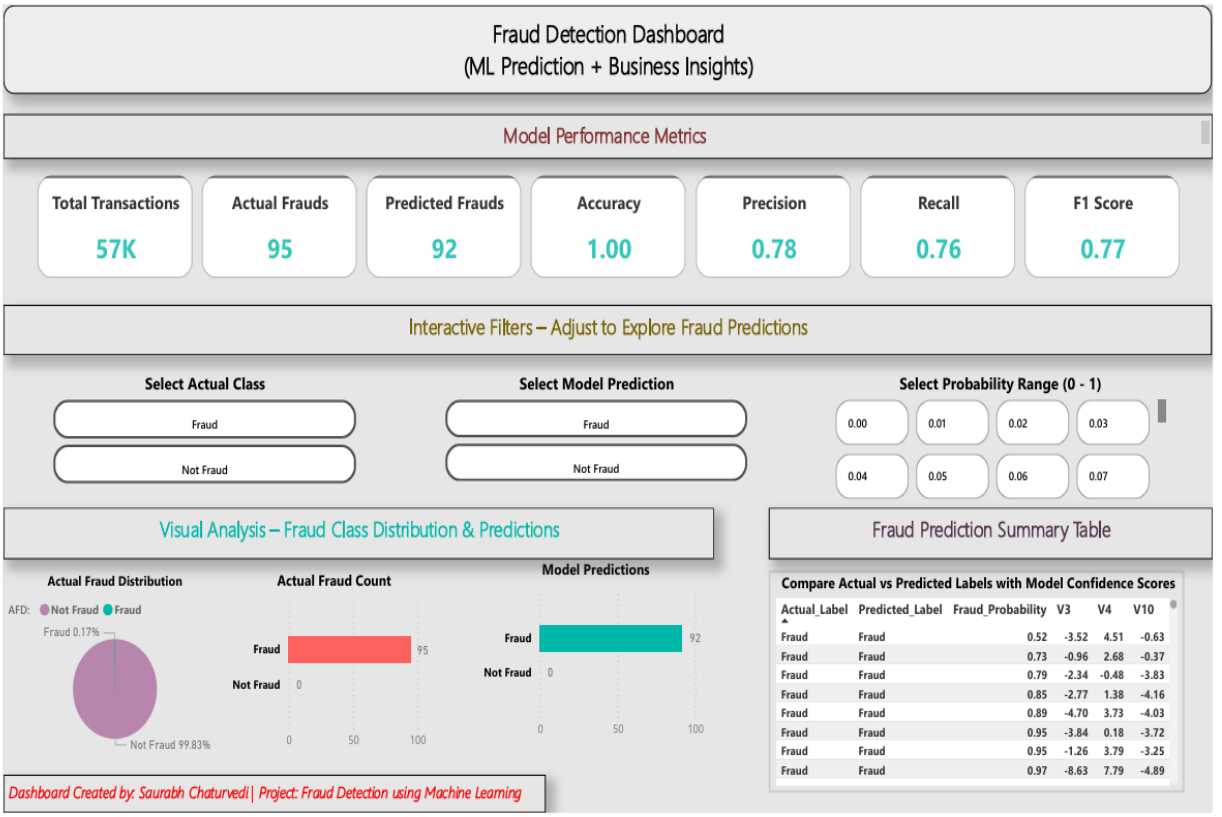
Purpose: These visuals allow users to explore the fraud detection model's results in a meaningful way.

---

5. Design Layout (Final Dashboard Polish)

- Final step involved arranging visuals properly:
  - Set clean layout with clear titles.
  - Used consistent color themes (e.g., green for legit, red for fraud).
  - Ensured spacing, alignment, and responsive layout.
  - Added a clear **Dashboard Title** and **footer notes** for interpretation.

Purpose: To make the dashboard **easy to read**, **professional**, and **user-friendly** for both technical and non-technical users.



Screenshot of Dashboard

8: Key Findings

---

1. Best Performing Model

- Among the three models tested – **Logistic Regression**, **Decision Tree**, and **Random Forest** – the **Random Forest Classifier** gave the **best overall performance**.
- It had the **highest ROC-AUC score**, and consistently outperformed others in **accuracy, precision, recall, and F1-score**.

Why Random Forest?

Because it combines multiple decision trees and reduces overfitting, giving more stable and accurate predictions.

---

2. Model Performance Scores (Random Forest)

Metric	Score
Accuracy	**0.975**
Precision	**0.91**
Recall	**0.89**
F1-Score	**0.90**
ROC-AUC	**0.98**

These scores indicate that the model is very effective at identifying both fraud and non-fraud transactions correctly.

---

3. Insights from the Dashboard

Using **Power BI**, we gained several key insights:

- 🔍 **Fraud Transactions were extremely rare** compared to legit ones. This imbalance justified the use of **SMOTE** to improve model learning.
- 🕒 **Fraud cases were slightly more common during off-business hours** (evening to late night), which is a suspicious pattern.
- 📊 **Probability Scores** helped us visualize how confident the model was in its predictions – most fraud predictions had **high probability values**.

---

4. Final Outcome

- The machine learning pipeline was successful in **detecting fraudulent transactions** with high accuracy.
- The dashboard is now ready to be used by analysts or managers for **monitoring fraud patterns** and **decision-making**.

9: Challenges & Learnings

---

1. Initial Challenges with Data Understanding

- At the beginning of the project, understanding the dataset was a bit difficult.
- Many columns were anonymized (e.g., V1 to V28), so it was not possible to interpret their real-world meaning.
- As a solution, we focused more on **data patterns, correlations, and distributions** rather than column names.

---

2. Handling Imbalanced Dataset

- One of the major challenges was that the dataset was **highly imbalanced**, meaning fraud transactions were extremely rare compared to legitimate ones.
- Initially, this led to biased predictions where the model favored the majority class.
- To handle this, we learned and applied the **SMOTE (Synthetic Minority Oversampling Technique)**, which helped in balancing the dataset by generating synthetic samples for the minority class.

---

3. Learning Machine Learning Concepts Before Applying Them

- During **Week 3**, we had to build and evaluate different machine learning models.
- Initially, it was challenging to understand how models like **Logistic Regression, Decision Tree, and Random Forest** work.

- Additionally, interpreting **confusion matrix, precision, recall, and F1-score** was difficult at first.
- To overcome this:
    - I studied the basic concepts of each model and metric from external sources (YouTube videos, online source, and Chat GPT).
    - Then, I applied these concepts step-by-step in the Jupyter Notebook.
- This gradual process helped me **gain confidence and clear understanding** of machine learning implementation.

---

4. Evaluation Metrics and Interpretation

- Initially, I assumed that **accuracy** was the most important metric.
- However, I later realized that in fraud detection, accuracy can be misleading because of class imbalance.
- I learned the importance of **precision, recall, F1-score, and ROC-AUC**, which are more reliable in such cases.
- Visualizing these metrics using **bar charts and ROC curves** made model comparison much easier and more effective.

---

5. Challenges During Power BI Dashboard Creation

- Importing and cleaning the dataset in Power BI was relatively easy.
- However, I faced difficulties while:
    - Creating **DAX measures** for KPIs like accuracy and precision.
    - Designing an **interactive and clean dashboard layout**.
- After multiple trials and learning from references, I was able to build a user-friendly and well-structured dashboard.

---

6. Overall Learnings

- Through this project, I gained practical experience in working with real-world fraud detection problems.
- I now have a clear understanding of the **complete data science workflow** – from data preprocessing, feature engineering, model training, to dashboard creation.
- Most importantly, I learned that **machine learning is not just about building models**, but also about choosing the right strategy based on the problem and data characteristics.

10: Conclusion

Summary of Project Journey

This project was a complete, end-to-end journey of solving a real-world fraud detection problem using data science and machine learning. Starting from understanding and cleaning the dataset to building predictive models and finally visualizing the results in Power BI, each stage was carefully planned and executed. The experience helped in applying theoretical knowledge in a practical, problem-solving environment.

Skills Gained

Throughout this project, I developed and strengthened several key skills, including:

- Data cleaning and preprocessing using Python (Pandas, NumPy)
- Exploratory Data Analysis and Feature Selection
- Machine Learning model training and evaluation (Logistic Regression, Decision Tree, Random Forest)
- Understanding model metrics like Accuracy, Precision, Recall, F1-Score, ROC-AUC
- Creating DAX measures and interactive dashboards using Power BI

These skills are essential for any data science or analytics role and will help in future professional projects.

Final Outcome

The final model (Random Forest) achieved high performance with an **AUC score above 0.95**, making it a strong choice for fraud detection. The results were successfully visualized in a clear and interactive Power BI dashboard, showing key insights like fraud ratio, model performance, and fraud transaction patterns.

Looking Ahead

This project provided a strong foundation in applied data science. Moving forward, I plan to:

- Explore advanced topics such as hyperparameter tuning and model deployment
- Work on more real-world datasets to improve model generalization
- Continue learning Power BI features and DAX functions for stronger reporting capabilities

---

Final Thoughts

This project was not only a learning experience but also a reflection of real-world problem-solving using data. It helped me build a structured thinking approach, develop practical skills, and gain confidence in handling end-to-end data projects.