**MACHINE LEARNING**

**MINI PROJECT REPORT**

***Title : Credit Card Fraud Detection using Machine Learning***



Submitted By: Submitted To:

**Nonihal Choudhary Mrs. Pawandeep Sharma**

UID:24MCI10100

Branch: MCA (AI/ML)

Section: 24MAM-4A

Semester: 2nd

**Aim:** To develop a Credit Card Fraud Detection system using Machine Learning algorithms such as Logistic Regression and Random Forest, compare their performance, and visualize evaluation metrics to identify fraudulent transactions effectively.

**Task To Be Done:**

* Load and explore the credit card transaction dataset.
* Preprocess the data using feature scaling and balancing techniques.
* Train two classification models: Logistic Regression and Random Forest (with hyperparameter tuning).
* Evaluate both models using metrics like Accuracy, Confusion Matrix, AUC Score, ROC Curve, and Precision-Recall Curve.
* Compare the models and visualize their performance.
* Display a summary table comparing the results.
* Visualize the fraud vs non-fraud distribution using a pie chart.

**Steps for Project:**

* **Data Loading:** Load the `creditcard.csv` dataset and display key features.
* **Exploratory Data Analysis (EDA):** Plot a pie chart to show the distribution of fraud and non-fraud transactions.
* **Preprocessing:** Scale `Time` and `Amount` features using `RobustScaler`, drop original columns, and balance the dataset using undersampling.
* **Train-Test Split:** Split the balanced dataset into training and testing sets (80/20 split).
* **Model Training & Evaluation:** Train Logistic Regression and evaluate. Tune Random Forest using GridSearchCV, then train and evaluate.
* **Feature Importance:** Plot the top important features for Random Forest.
* **ROC Curve & Precision-Recall Curve:** Visualize ROC and PR curves.
* **Model Comparison Table:** Generate and display comparison of Accuracy and AUC Scores.

**Code of the Project:**

**import pandas as pd**

**import numpy as np**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**import time**

**from sklearn.model\_selection import train\_test\_split, GridSearchCV**

**from sklearn.metrics import (**

**classification\_report, confusion\_matrix, accuracy\_score,**

**roc\_auc\_score, roc\_curve, precision\_recall\_curve**

**)**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.ensemble import RandomForestClassifier**

**from sklearn.preprocessing import RobustScaler**

**from sklearn.utils import shuffle**

**from tabulate import tabulate**

**import joblib**

**print("\nDataset:-\n\n")**

**df = pd.read\_csv('creditcard.csv')**

**print(df[['V1', 'V2', 'V3', 'Amount', 'Time', 'Class']].head())**

**print(f"\nDataset shape: {df.shape[0]} rows × {df.shape[1]} columns")**

**print( "="\*150)**

**print("\n")**

**plt.figure(figsize=(5, 5))**

**df['Class'].value\_counts().plot.pie(labels=['Non-Fraud', 'Fraud'],**

**autopct='%1.2f%%', colors=['skyblue', 'salmon'])**

**plt.title('Class Distribution')**

**plt.ylabel('')**

**plt.show()**

**print("="\*150)**

**rbs = RobustScaler()**

**scaled = rbs.fit\_transform(df[['Time', 'Amount']])**

**scaled\_df = pd.DataFrame(scaled, columns=['scaled\_time', 'scaled\_amount'])**

**df = pd.concat([df, scaled\_df], axis=1)**

**df.drop(['Time', 'Amount'], axis=1, inplace=True)**

**non\_fraud = df[df['Class'] == 0].sample(n=492, random\_state=1)**

**fraud = df[df['Class'] == 1]**

**new\_df = pd.concat([non\_fraud, fraud])**

**new\_df = shuffle(new\_df, random\_state=2)**

**X = new\_df.drop('Class', axis=1)**

**y = new\_df['Class']**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**def evaluate\_model(model, name):**

**print("\n\n" + "^"\*40)**

**print(f"{name} Evaluation")**

**print("^"\*40)**

**start = time.time()**

**model.fit(X\_train, y\_train)**

**end = time.time()**

**pred = model.predict(X\_test)**

**print(f"Training Time: {end - start:.2f} seconds")**

**print("\nClassification Report:\n\n", classification\_report(y\_test, pred))**

**print("\nConfusion Matrix:\n", confusion\_matrix(y\_test, pred))**

**print("\n"f"Accuracy: {accuracy\_score(y\_test, pred) \* 100:.2f}%")**

**print("\n"f"AUC Score: {roc\_auc\_score(y\_test, pred):.2f}")**

**print("\n")**

**sns.heatmap(confusion\_matrix(y\_test, pred), annot=True, fmt="d", cmap="Purples")**

**plt.title(f"{name} - Confusion Matrix")**

**plt.xlabel("Predicted")**

**plt.ylabel("Actual")**

**plt.tight\_layout()**

**plt.show()**

**return model, pred**

**lr\_model, lr\_pred = evaluate\_model(**

**LogisticRegression(class\_weight='balanced', max\_iter=1000),**

**"Logistic Regression")**

**print("\n\n" + "="\*150)**

**print("\nPerforming Hyperparameter Tuning on Random Forest:\n")**

**param\_grid = {**

**'n\_estimators': [50, 100],**

**'max\_depth': [None, 10, 20],**

**'min\_samples\_split': [2, 5],**

**}**

**grid = GridSearchCV(RandomForestClassifier(class\_weight='balanced'), param\_grid, cv=3, scoring='accuracy')**

**grid.fit(X\_train, y\_train)**

**print("Best RF Parameters:", grid.best\_params\_)**

**best\_rf = grid.best\_estimator\_**

**rf\_model, rf\_pred = evaluate\_model(best\_rf, "Random Forest (Tuned)")**

**print("\n\n" + "="\*150)**

**print("\n")**

**importances = best\_rf.feature\_importances\_**

**features = X.columns**

**indices = np.argsort(importances)[::-1]**

**plt.figure(figsize=(10, 5))**

**sns.barplot(x=importances[indices], y=features[indices], color='blue')**

**plt.title("Feature Importance (Random Forest)")**

**plt.xlabel("Importance")**

**plt.ylabel("Features")**

**plt.tight\_layout()**

**plt.show()**

**print("\n\n" + "="\*150)**

**print("\n")**

**rf\_probs = best\_rf.predict\_proba(X\_test)[:, 1]**

**fpr, tpr, \_ = roc\_curve(y\_test, rf\_probs)**

**plt.figure(figsize=(6, 4))**

**plt.plot(fpr, tpr, label='Random Forest', color='green')**

**plt.plot([0, 1], [0, 1], 'k--')**

**plt.xlabel("False Positive Rate")**

**plt.ylabel("True Positive Rate")**

**plt.title("ROC Curve - Random Forest")**

**plt.legend()**

**plt.grid(True)**

**plt.tight\_layout()**

**plt.show()**

**print("\n\n" + "="\*90)**

**print("\n")**

**precision, recall, \_ = precision\_recall\_curve(y\_test, rf\_probs)**

**plt.figure(figsize=(6, 4))**

**plt.plot(recall, precision, marker='.')**

**plt.xlabel("Recall")**

**plt.ylabel("Precision")**

**plt.title("Precision-Recall Curve - Random Forest")**

**plt.grid(True)**

**plt.tight\_layout()**

**plt.show()**

**print("\n\n" + "="\*90)**

**print("\n")**

**results = {**

**"Model Name": ["Logistic Regression", "Random Forest (Tuned)"],**

**"Accuracy (%)": [accuracy\_score(y\_test, lr\_pred), accuracy\_score(y\_test, rf\_pred)],**

**"AUC Score": [roc\_auc\_score(y\_test, lr\_pred), roc\_auc\_score(y\_test, rf\_pred)]**

**}**

**summary\_df = pd.DataFrame(results)**

**summary\_df.columns = [**

**"\033[1mModel Name\033[0m",**

**"\033[1mAccuracy (%)\033[0m",**

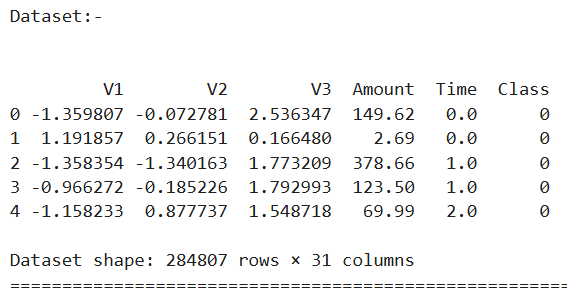
**"\033[1mAUC Score\033[0m"**

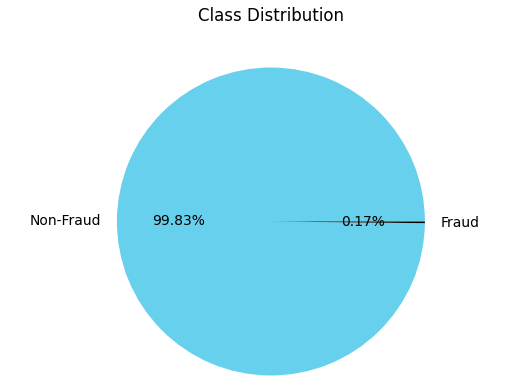
**]**

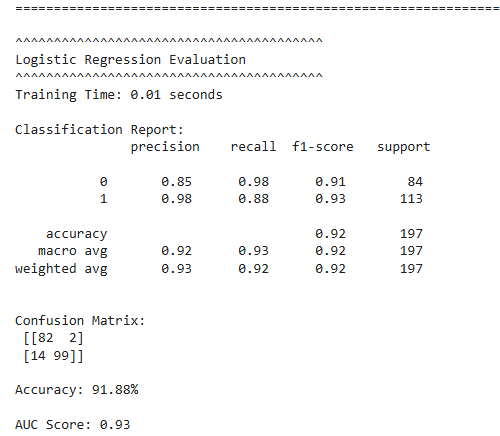
**print("\n\n\U0001F4CA Model Comparison Summary:\n\n")**

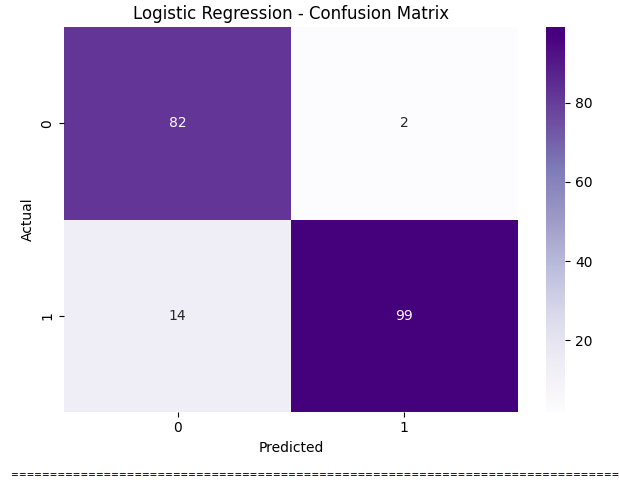
**print(tabulate(summary\_df, headers='keys', tablefmt='fancy\_grid', showindex=False))**

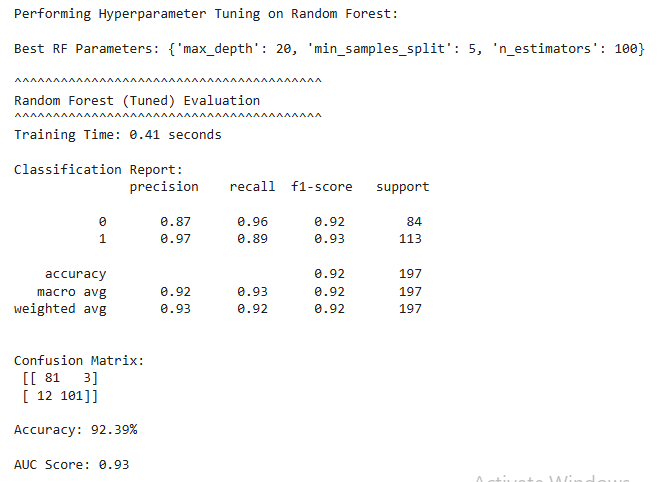
**Output:**

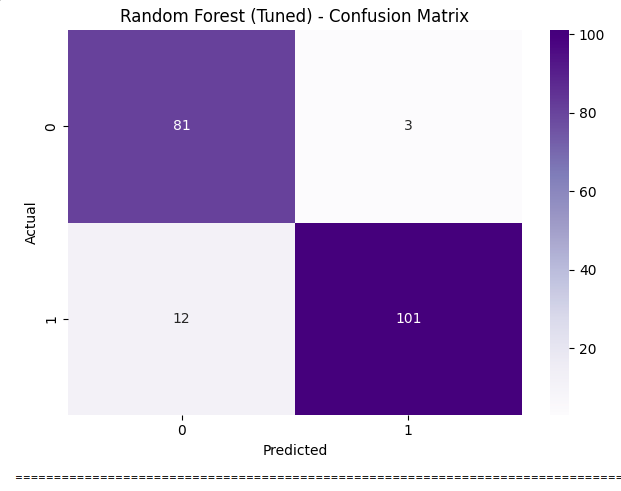
****

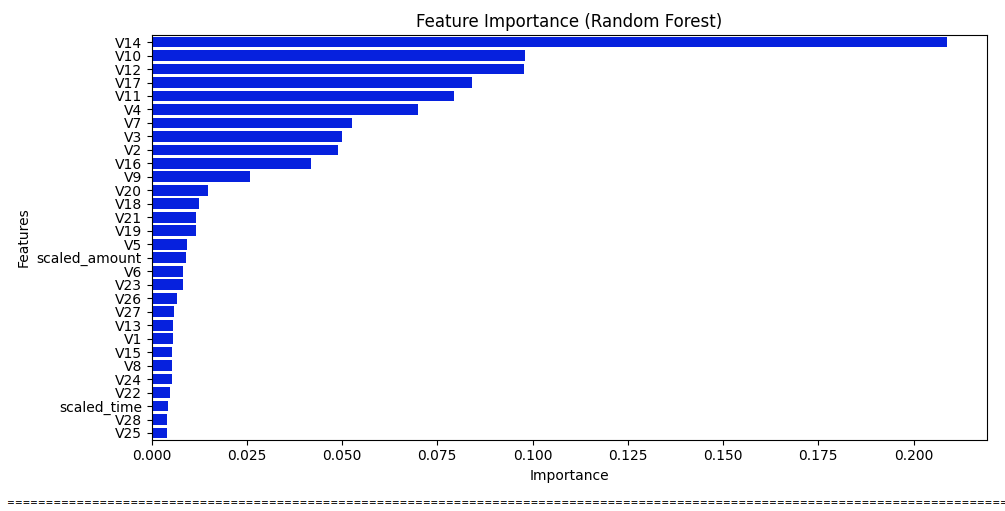
****

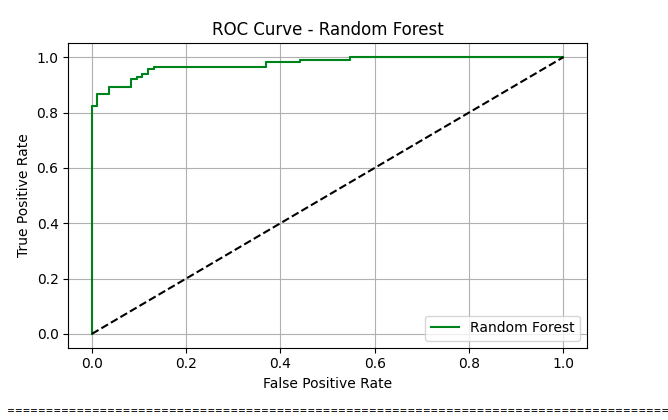
****

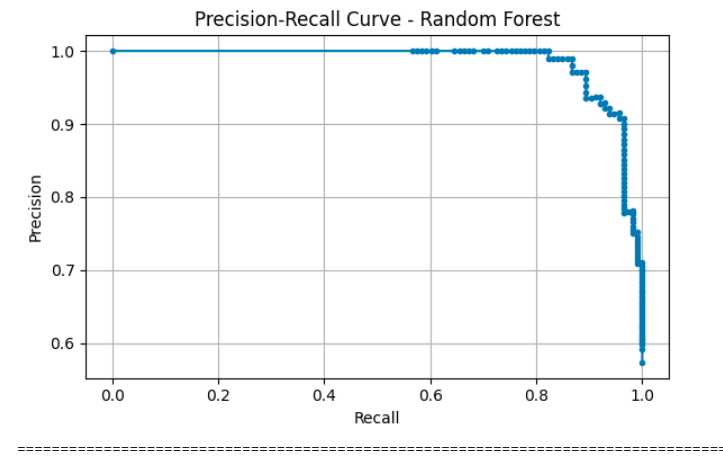
****

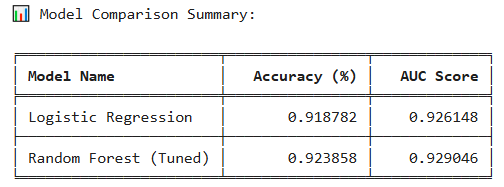
****

****

****

****

****

****