

STEP 1: IMPORT LIBRARIES

```
In [1]: 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.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.dummy import DummyClassifier

import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

STEP 2: LOAD DATASET

```
In [3]: df = pd.read_csv(
    r"C:\Users\yashw\OneDrive\Desktop\Credit Card Fraud Detection using Machine Le
)

df.head()
```

Out[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	'
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.0986
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.0851
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.2476
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.3774
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.2705

5 rows × 31 columns



STEP 3: BASIC DATA CHECKS

```
In [4]: df.shape
df.info()
df.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Time     284807 non-null   float64
 1   V1       284807 non-null   float64
 2   V2       284807 non-null   float64
 3   V3       284807 non-null   float64
 4   V4       284807 non-null   float64
 5   V5       284807 non-null   float64
 6   V6       284807 non-null   float64
 7   V7       284807 non-null   float64
 8   V8       284807 non-null   float64
 9   V9       284807 non-null   float64
 10  V10      284807 non-null   float64
 11  V11      284807 non-null   float64
 12  V12      284807 non-null   float64
 13  V13      284807 non-null   float64
 14  V14      284807 non-null   float64
 15  V15      284807 non-null   float64
 16  V16      284807 non-null   float64
 17  V17      284807 non-null   float64
 18  V18      284807 non-null   float64
 19  V19      284807 non-null   float64
 20  V20      284807 non-null   float64
 21  V21      284807 non-null   float64
 22  V22      284807 non-null   float64
 23  V23      284807 non-null   float64
 24  V24      284807 non-null   float64
 25  V25      284807 non-null   float64
 26  V26      284807 non-null   float64
 27  V27      284807 non-null   float64
 28  V28      284807 non-null   float64
 29  Amount    284807 non-null   float64
 30  Class     284807 non-null   int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

```
Out[4]: Time      0  
          V1       0  
          V2       0  
          V3       0  
          V4       0  
          V5       0  
          V6       0  
          V7       0  
          V8       0  
          V9       0  
          V10      0  
          V11      0  
          V12      0  
          V13      0  
          V14      0  
          V15      0  
          V16      0  
          V17      0  
          V18      0  
          V19      0  
          V20      0  
          V21      0  
          V22      0  
          V23      0  
          V24      0  
          V25      0  
          V26      0  
          V27      0  
          V28      0  
          Amount    0  
          Class     0  
          dtype: int64
```

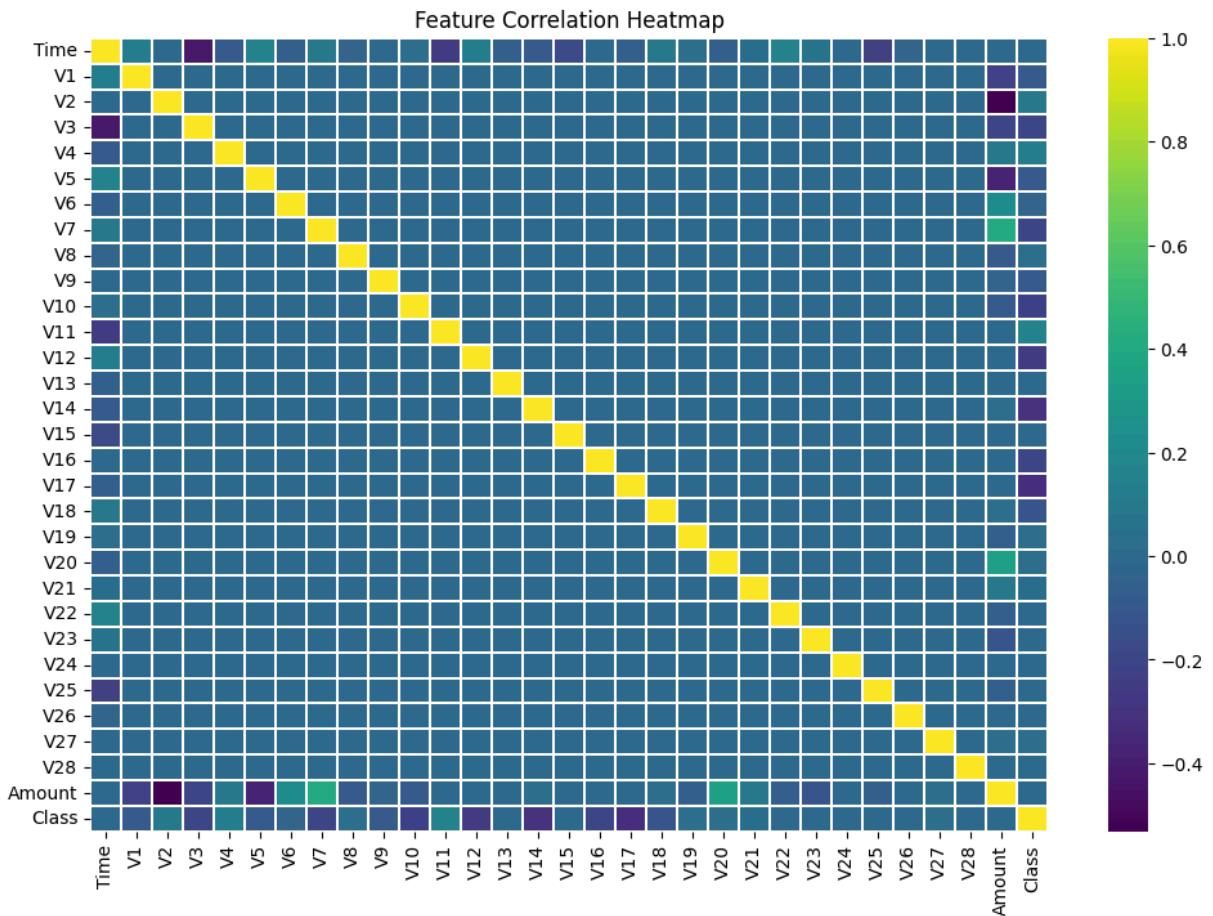
STEP 4: TARGET VARIABLE ANALYSIS

```
In [5]: df['Class'].value_counts()
```

```
Out[5]: Class  
0    284315  
1     492  
Name: count, dtype: int64
```

STEP 5: EDA

```
In [6]: plt.figure(figsize=(12,8))  
sns.heatmap(df.corr(), cmap='viridis', linewidths=0.1)  
plt.title("Feature Correlation Heatmap")  
plt.show()
```



STEP 6: FEATURE-TARGET SPLIT

```
In [7]: X = df.drop("Class", axis=1)
y = df["Class"]
```

STEP 7: TRAIN-TEST SPLIT

```
In [8]: X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.25,
    random_state=42,
    stratify=y
)
```

STEP 8: BASELINE MODEL

```
In [9]: dummy = DummyClassifier(strategy="most_frequent")
dummy.fit(X_train, y_train)
dummy_pred = dummy.predict(X_test)

print("Dummy Accuracy:", accuracy_score(y_test, dummy_pred))
```

Dummy Accuracy: 0.9982725204348193

STEP 9: LOGISTIC REGRESSION

```
In [10]: lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)

lr_pred = lr.predict(X_test)

print("Accuracy:", accuracy_score(y_test, lr_pred))
print("Precision:", precision_score(y_test, lr_pred))
print("Recall:", recall_score(y_test, lr_pred))
print("F1:", f1_score(y_test, lr_pred))
```

Accuracy: 0.9990309260975815

Precision: 0.75

Recall: 0.6585365853658537

F1: 0.7012987012987012

STEP 10: RANDOM FOREST

```
In [11]: rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

rf_pred = rf.predict(X_test)

print("Accuracy:", accuracy_score(y_test, rf_pred))
print("Precision:", precision_score(y_test, rf_pred))
print("Recall:", recall_score(y_test, rf_pred))
print("F1:", f1_score(y_test, rf_pred))
```

Accuracy: 0.9995786635206876

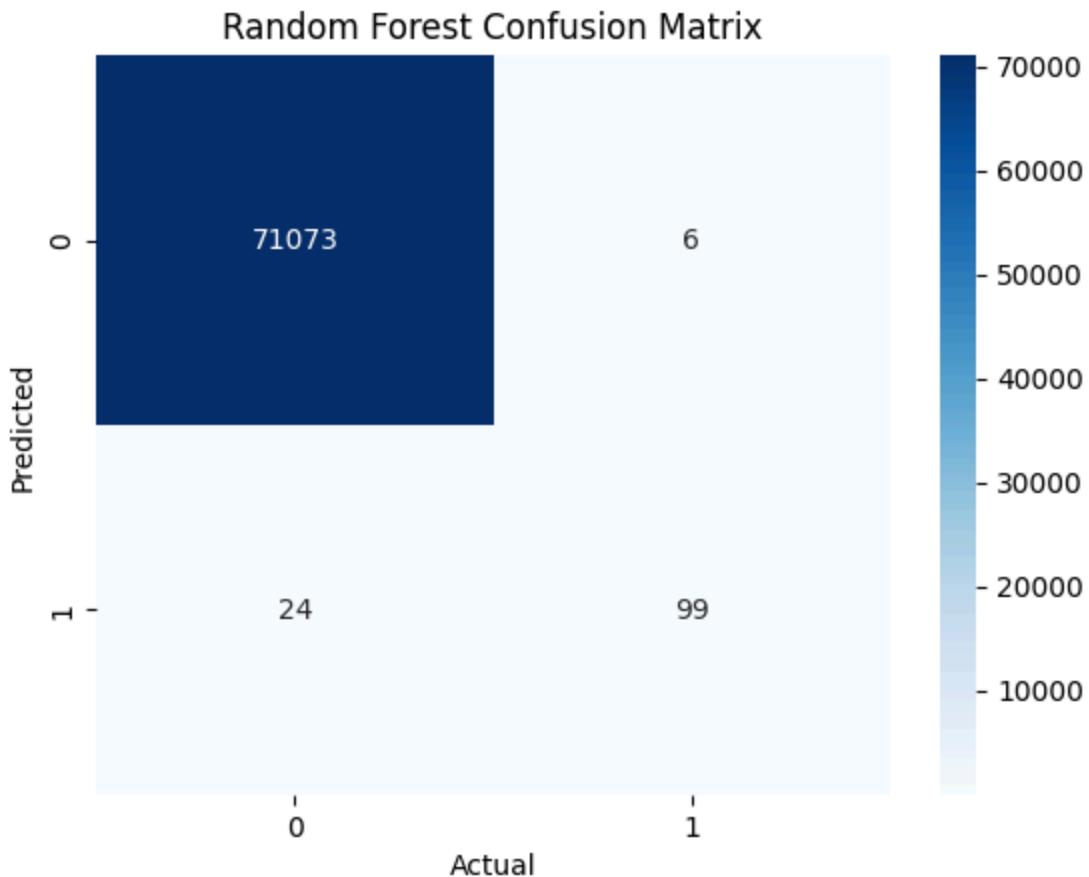
Precision: 0.9428571428571428

Recall: 0.8048780487804879

F1: 0.868421052631579

STEP 11: CONFUSION MATRIX

```
In [12]: cm = confusion_matrix(y_test, rf_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.title("Random Forest Confusion Matrix")
plt.show()
```



STEP 12: NEW / UNSEEN TRANSACTION PREDICTION

```
In [13]: new_transaction = X_test.iloc[[0]]

prediction = rf.predict(new_transaction)
probability = rf.predict_proba(new_transaction)

if prediction[0] == 1:
    print("⚠ Fraudulent Transaction")
else:
    print("✓ Legitimate Transaction")

print("Fraud Probability:", probability[0][1])
```

✓ Legitimate Transaction
Fraud Probability: 0.0

STEP 13: MODEL SAVING

```
In [15]: import os
import joblib

# create models folder if it doesn't exist
os.makedirs("models", exist_ok=True)

# save model
joblib.dump(rf, "models/fraud_model.pkl")
```

```
print("✓ Model saved successfully!")
```

✓ Model saved successfully!