```
from google.colab import files
Upload=files.upload()
```

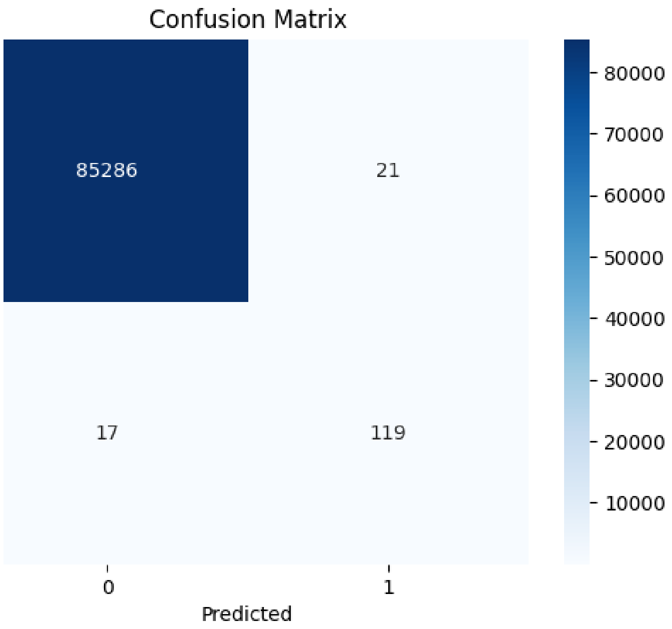
Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving creditcard.csv to creditcard.csv
```

```
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.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
from imblearn.over_sampling import SMOTE

# Load Dataset
df = pd.read_csv("creditcard.csv")

# Basic info
print("Dataset shape:", df.shape)
print(df['Class'].value_counts())

# Data preprocessing
X = df.drop(['Class', 'Time'], axis=1)
y = df['Class']

# Scale Amount
X['Amount'] = StandardScaler().fit_transform(X['Amount'].values.reshape(-1, 1))

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42

# Handle imbalance using SMOTE
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X_train, y_train)

print("After SMOTE:", y_res.value_counts())

# Train model
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_res, y_res)

# Predictions
y_pred = clf.predict(X_test)
y_proba = clf.predict_proba(X_test)[:, 1]

# Evaluation
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("ROC AUC Score:", roc_auc_score(y_test, y_proba))

# Plot confusion matrix
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
```

```
plt.ylabel("Actual")
plt.show()
```

```
: shape: (284807, 31)

;4315
 492
:ount, dtype: int64
;MOTE: Class
)9008
)9008
:ount, dtype: int64
Fication Report:
             precision    recall  f1-score   support

       0       1.00      1.00      1.00     85307
       1       0.85      0.88      0.86       136

:uracy                           1.00     85443
·o avg       0.92      0.94      0.93     85443
:d avg       1.00      1.00      1.00     85443

.on Matrix:
36    21]
'   119]]
: Score: 0.9766445188623236
```




```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Load dataset
```

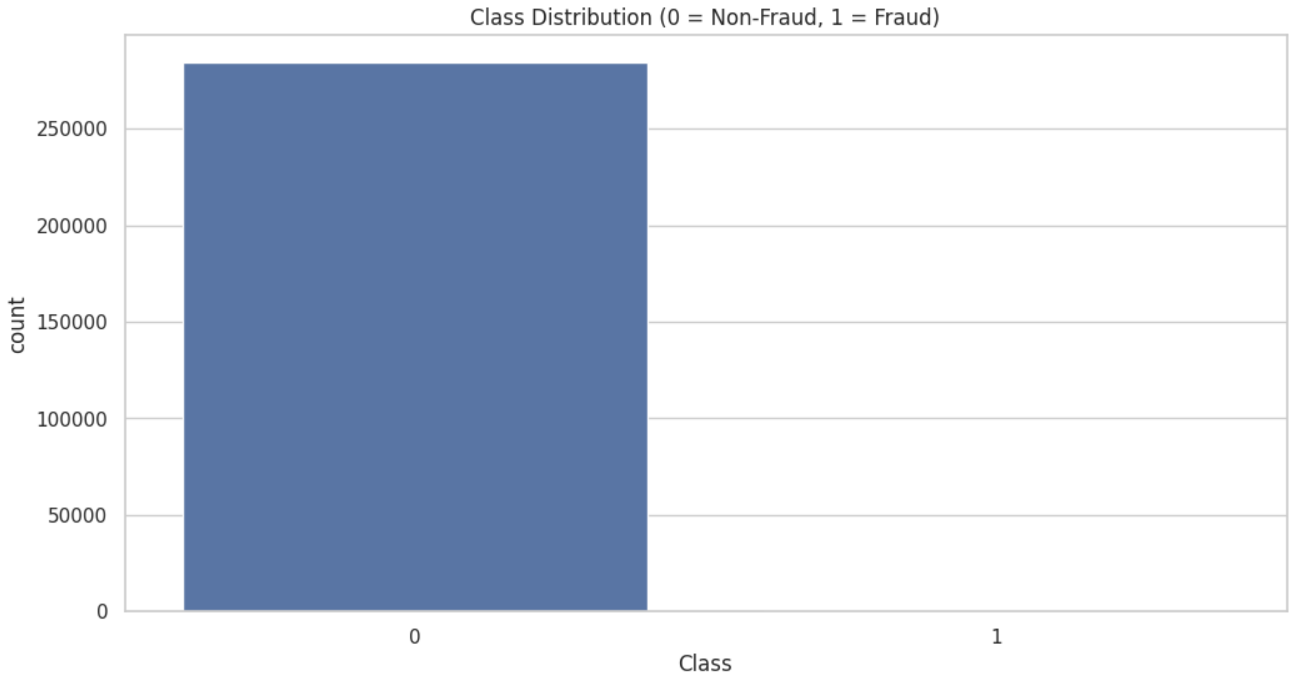
```python
df = pd.read_csv("creditcard.csv")

# View dataset shape and class balance
print("Dataset shape:", df.shape)
print("Class distribution:\n", df['Class'].value_counts())

# Drop 'Time' column (not useful for prediction)
df = df.drop(columns=['Time'])

# Scale the 'Amount' feature
scaler = StandardScaler()
df['Amount'] = scaler.fit_transform(df['Amount'].values.reshape(-1, 1))

# Define features and target
X = df.drop('Class', axis=1)
y = df['Class']

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

# Show results
print("Training data shape:", X_train.shape)
print("Testing data shape:", X_test.shape)
```

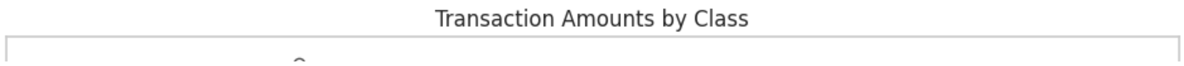
```
Dataset shape: (284807, 31)
Class distribution:
 Class
0    284315
1       492
Name: count, dtype: int64
Training data shape: (199364, 29)
Testing data shape: (85443, 29)
```

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

sns.set(style='whitegrid')
plt.rcParams['figure.figsize'] = (12, 6)


df = pd.read_csv('creditcard.csv')  # Update path if needed


print("Shape of data:", df.shape)
print(df.info())
print(df.describe())

print("Missing Values:\n", df.isnull().sum())


sns.countplot(x='Class', data=df)
plt.title('Class Distribution (0 = Non-Fraud, 1 = Fraud)')
plt.show()


sns.boxplot(x='Class', y='Amount', data=df)
plt.title('Transaction Amounts by Class')
```

```
plt.show()


df['Hour'] = df['Time'].apply(lambda x: np.floor(x / 3600) % 24)
sns.histplot(data=df, x='Hour', bins=24, hue='Class', multiple='stack', kde=True)
plt.title('Transaction Distribution by Hour')
plt.xlabel('Hour of the Day')
plt.show()

corr = df.corr()
sns.heatmap(corr, cmap='coolwarm', linewidths=0.5)
plt.title('Feature Correlation Matrix')
plt.show()


pca_features = df.iloc[:, 1:29]
pca_features.boxplot(rot=90)
plt.title('PCA Features Distribution')
plt.show()


fraud_count = df['Class'].value_counts()
fraud_percent = fraud_count[1] / fraud_count.sum() * 100
print(f"Fraudulent Transactions: {fraud_count[1]} ({fraud_percent:.4f}%)")
```

```
 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
```



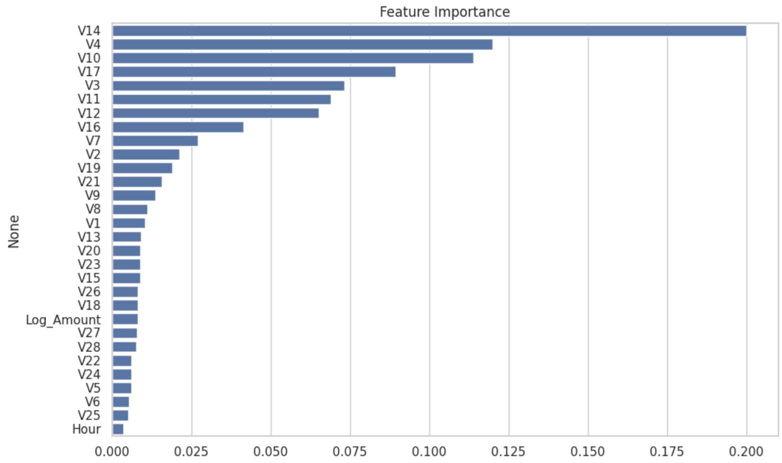


```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, RobustScaler


df = pd.read_csv('creditcard.csv')  # Update path if needed

df.dropna(inplace=True)




df['Hour'] = np.floor(df['Time'] / 3600) % 24


df['Log_Amount'] = np.log1p(df['Amount'])


rs = RobustScaler()
df['Scaled_Amount'] = rs.fit_transform(df['Amount'].values.reshape(-1, 1))

df['Rolling_Amount_1hr'] = df.groupby('Hour')['Amount'].transform(lambda x: x.rolling(wi


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

print("Transformed Dataset:\n", df.head())
print("\nFeature Columns:", df.columns.tolist())
print("Shape:", df.shape)
```

```
Transformed Dataset:
          V1        V2        V3        V4        V5        V6        V7  \
0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941

         V8        V9       V10  ...       V24       V25       V26       V27
0  0.098698  0.363787  0.090794  ...  0.066928  0.128539 -0.189115  0.133558
1  0.085102 -0.255425 -0.166974  ... -0.339846  0.167170  0.125895 -0.008983
2  0.247676 -1.514654  0.207643  ... -0.689281 -0.327642 -0.139097 -0.055353
3  0.377436 -1.387024 -0.054952  ... -1.175575  0.647376 -0.221929  0.062723
4 -0.270533  0.817739  0.753074  ...  0.141267 -0.206010  0.502292  0.219422

        V28  Class  Hour  Log_Amount  Scaled_Amount  Rolling_Amount_1hr
0 -0.021053      0   0.0    5.014760       1.783274          149.620000
1  0.014724      0   0.0    1.305626      -0.269825           76.155000
2 -0.059752      0   0.0    5.939276       4.983721          176.990000
3  0.061458      0   0.0    4.824306       1.418291          168.283333
4  0.215153      0   0.0    4.262539       0.670579          190.716667

[5 rows x 33 columns]

Feature Columns: ['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
Shape: (284807, 33)
```

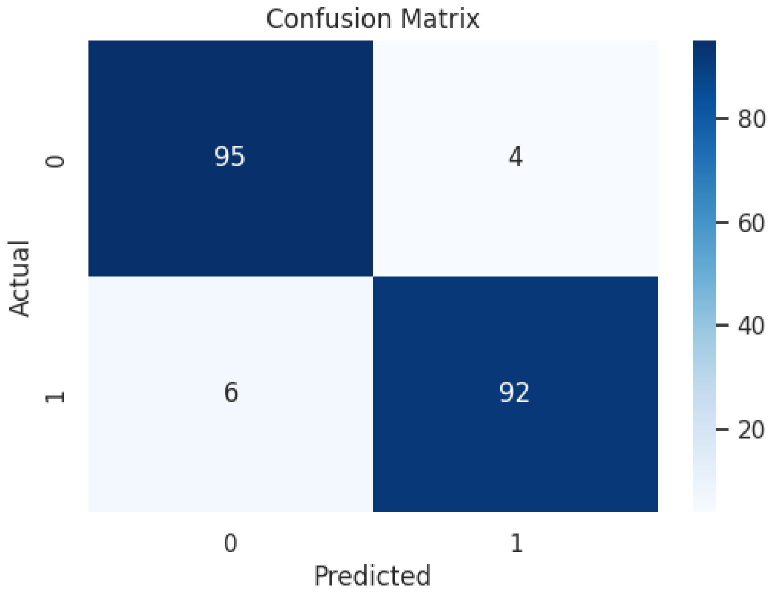
```
mport pandas as pd
mport numpy as np
rom sklearn.model_selection import train_test_split
rom sklearn.ensemble import RandomForestClassifier
rom sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, precis
rom sklearn.utils import resample


f = pd.read_csv('creditcard.csv')  # Replace with your processed data path


f['Hour'] = np.floor(df['Time'] / 3600) % 24
f['Log_Amount'] = np.log1p(df['Amount'])
f.drop(['Time', 'Amount'], axis=1, inplace=True)

raud = df[df['Class'] == 1]
on_fraud = df[df['Class'] == 0].sample(len(fraud), random_state=42)
alanced_df = pd.concat([fraud, non_fraud])


 = balanced_df.drop('Class', axis=1)
 = balanced_df['Class']


_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, 

odel = RandomForestClassifier(n_estimators=100, random_state=42)
odel.fit(X_train, y_train)


_pred = model.predict(X_test)
_proba = model.predict_proba(X_test)[:, 1]

rint("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
rint("\nClassification Report:\n", classification_report(y_test, y_pred))
rint("ROC AUC Score:", roc_auc_score(y_test, y_proba))


mport matplotlib.pyplot as plt
mport seaborn as sns

mportances = model.feature_importances_
eatures = X.columns
ndices = np.argsort(importances)[::-1]

lt.figure(figsize=(10, 6))
ns.barplot(x=importances[indices], y=features[indices])
lt.title("Feature Importance")
lt.tight_layout()
lt.show()
```

```
Confusion Matrix:
 [[95  4]
 [ 6 92]]

Classification Report:
              precision    recall  f1-score

           0       0.94      0.96      0.95
           1       0.96      0.94      0.95

    accuracy                           0.95
   macro avg       0.95      0.95      0.95
weighted avg       0.95      0.95      0.95

ROC AUC Score:  0.9888167388167388
```




```
from sklearn.metrics import (
    classification_report,
    confusion_matrix,
    roc_auc_score,
    roc_curve,
    precision_recall_curve,
    average_precision_score
)
import matplotlib.pyplot as plt
import seaborn as sns


cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

print("Classification Report:\n")
print(classification_report(y_test, y_pred))


fpr, tpr, _ = roc_curve(y_test, y_proba)
```

```
roc_auc = roc_auc_score(y_test, y_proba)

plt.figure(figsize=(6, 4))
plt.plot(fpr, tpr, label=f"ROC Curve (AUC = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend(loc="lower right")
plt.show()


precision, recall, _ = precision_recall_curve(y_test, y_proba)
avg_precision = average_precision_score(y_test, y_proba)

plt.figure(figsize=(6, 4))
plt.plot(recall, precision, label=f"AP = {avg_precision:.2f}")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve")
plt.legend()
plt.show()
```




```
Classification Report:
```

```
import joblib

joblib.dump(model, 'fraud_model.pkl')
```

```
['fraud_model.pkl']
      accuracy                          0.95
     macro avg       0.95      0.95      0.95
```

```
import gradio as gr
import pandas as pd
```