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

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, RobustScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    classification_report, confusion_matrix,
    roc_auc_score, roc_curve,
    precision_recall_curve, average_precision_score
)
from imblearn.over_sampling import SMOTE
import joblib
import gradio as gr

# Load dataset
df = pd.read_csv("creditcard.csv")
print("Dataset shape:", df.shape)
print(df['Class'].value_counts())

# Preprocessing
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(window=1, min_periods=1).mean())
df.drop(['Time', 'Amount'], axis=1, inplace=True)

# Train-test split
X = df.drop('Class', axis=1)
y = df['Class']
X_train,    X_test,    y_train,    y_test    =    train_test_split(X,    y,    test_size=0.3,
random_state=42, stratify=y)

# 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:
", classification_report(y_test, y_pred))
print("Confusion Matrix:
", confusion_matrix(y_test, y_pred))
print("ROC AUC Score:", roc_auc_score(y_test, y_proba))

# Confusion matrix heatmap
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()

# ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_proba)
roc_auc = roc_auc_score(y_test, y_proba)
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 Curve
precision, recall, _ = precision_recall_curve(y_test, y_proba)
avg_precision = average_precision_score(y_test, y_proba)
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()

# Save model
joblib.dump(clf, 'fraud_model.pkl')
```