STEP 1: Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import joblib
```

STEP 2: Load Dataset

```
df = pd.read_csv("/content/synthetic_credit_card_fraud_dataset.csv")
df.head()
```

	Time	Amount	V1	V2	V3	V4	V5	V6	
0	134034	59.561956	-1.349422	0.563467	-0.462246	1.808542	0.077719	1.683083	-1
1	33340	89.562922	-1.791851	-0.487429	0.865120	-0.443762	-0.262683	-0.039862	-1
2	108821	1.000000	0.881509	0.403441	-1.261326	0.119382	2.034923	0.990065	1
3	75209	69.697145	-0.354314	1.320858	-2.158034	-0.089611	-0.469750	-1.116201	2
4	164329	45.572244	-1.956442	1.174213	1.260515	1.480553	-0.297553	0.469755	-0

Next steps: Generate code with df | New interactive sheet

STEP 3: Check Class Imbalance

```
df["Class"].value_counts()
```

Class	count
0	9800
1	200

dtype: int64

STEP 4: Separate Features & Target

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

STEP 5: Stratified Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)
```

STEP 6: Handle Imbalance (Manual Oversampling)

```
train_data = pd.concat([X_train, y_train], axis=1)

fraud = train_data[train_data["Class"] == 1]
non_fraud = train_data[train_data["Class"] == 0]

fraud_oversampled = fraud.sample(len(non_fraud), replace=True, random_state=42)

balanced_train = pd.concat([non_fraud, fraud_oversampled])

X_train_bal = balanced_train.drop("Class", axis=1)
y_train_bal = balanced_train["Class"]

y_train_bal.value_counts()
```

Class	count
0	7840
1	7840

dtype: int64

STEP 7: Feature Scaling

```
scaler = StandardScaler()

X_train_bal = scaler.fit_transform(X_train_bal)
X_test_scaled = scaler.transform(X_test)
```

STEP 8: Baseline Model – Logistic Regression

```
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train_bal, y_train_bal)

lr_pred = lr.predict(X_test_scaled)

print("Logistic Regression Performance:\n")
print(classification_report(y_test, lr_pred))
```

```
Logistic Regression Performance:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1960
           1       0.97      0.97      0.97        40

    accuracy                           1.00      2000
   macro avg       0.99      0.99      0.99      2000
weighted avg       1.00      1.00      1.00      2000
```

STEP 9: Train Random Forest

```
rf = RandomForestClassifier(
    n_estimators=100,
    random_state=42
)

rf.fit(X_train_bal, y_train_bal)

rf_pred = rf.predict(X_test_scaled)

print("Random Forest Performance:\n")
print(classification_report(y_test, rf_pred))
```

```
Random Forest Performance:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1960
           1       1.00      0.80      0.89        40

    accuracy                           1.00      2000
   macro avg       1.00      0.90      0.94      2000
weighted avg       1.00      1.00      1.00      2000
```

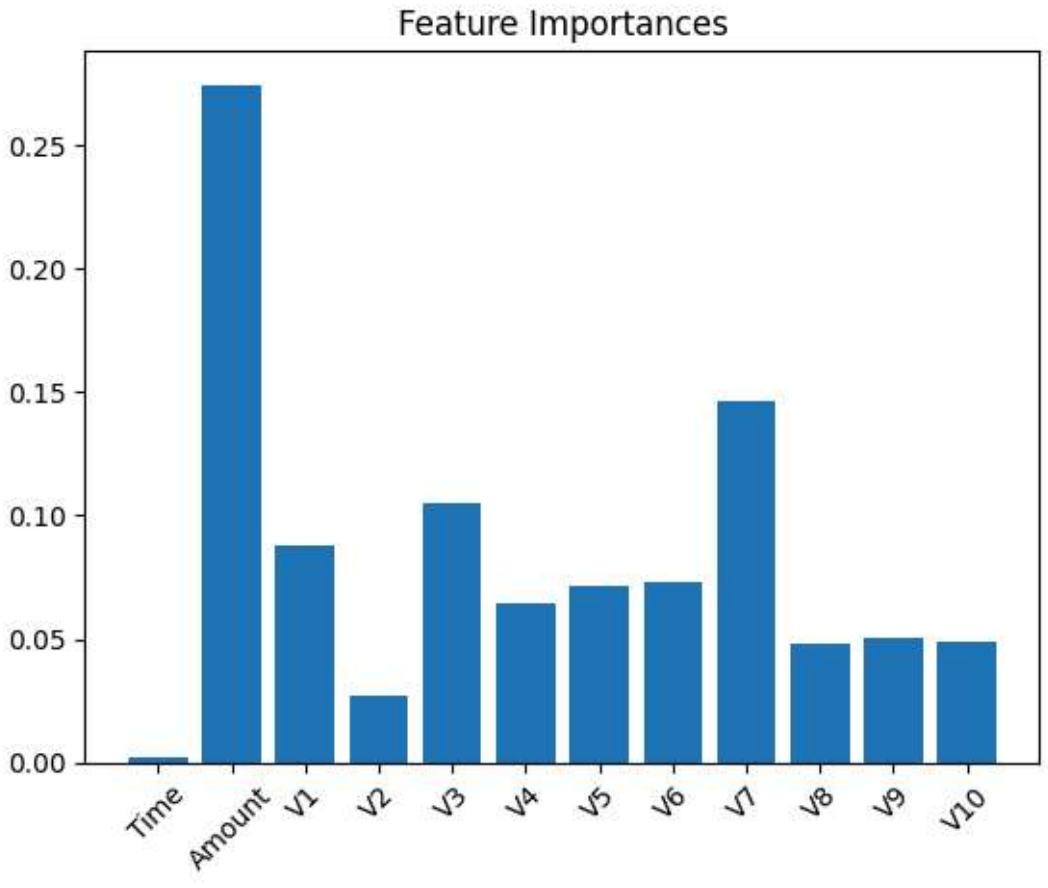
STEP 10: Confusion Matrix

```
cm = confusion_matrix(y_test, rf_pred)

print("Confusion Matrix:\n", cm)
```

```
Confusion Matrix:
 [[1960    0]
 [   8   32]]
```

STEP 11: Feature Importance

```
importances = rf.feature_importances_
feature_names = X.columns

plt.figure()
plt.bar(feature_names, importances)
plt.xticks(rotation=45)
plt.title("Feature Importances")
plt.show()
```



STEP 12: Save Model

```
joblib.dump(rf, "random_forest_fraud_model.pkl")
```

```
['random_forest_fraud_model.pkl']
```