

Task 3: Fraud Detection System

Description:

Develop a fraud detection system using a dataset like the Credit Card Fraud Dataset.

Steps:

1. Data Preprocessing:

○ Handle imbalanced data using techniques like SMOTE or undersampling.

2. Model Training:

○ Train a Random Forest or Gradient Boosting model to detect fraudulent transactions.

3. Model Evaluation:

○ Evaluate the system's precision, recall, and F1-score.

4. Testing Interface:

○ Create a simple interface (e.g., a command-line input) to test the fraud detection system.

Outcome:

● A Python script capable of detecting fraudulent transactions with evaluation metrics and a testing interface.

▼ **Installation and Import Dataset and Libraries**

```
from google.colab import files
uploaded = files.upload()
import pandas as pd
df = pd.read_csv('creditcard.csv')
```

Choose Files | creditcard.csv

creditcard.csv(text/csv) - 150828752 bytes, last modified: 5/28/2025 - 100% done

Saving creditcard.csv to creditcard.csv

```
import pandas as pd
import numpy as np
import random
import warnings
warnings.filterwarnings("ignore")

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

from imblearn.over_sampling import SMOTE

import matplotlib.pyplot as plt
import seaborn as sns
```

```
df.head()
```

	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	
	0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110
	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.10
	1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.90
	0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.19
	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.13

1. Data Preprocessing:

○ Handle imbalanced data using techniques like SMOTE or undersampling.

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

```
Class
0    284315
1       492
Name: count, dtype: int64
```

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

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_scaled, y)

print("After SMOTE:")
print(pd.Series(y_resampled).value_counts())
```

```
After SMOTE:
Class
0    284315
1    284315
Name: count, dtype: int64
```

2. Model Training:

○ Train a Random Forest or Gradient Boosting model to detect fraudulent transactions.

```
X_train, X_test, y_train, y_test = train_test_split(
    X_resampled, y_resampled, test_size=0.2, random_state=42
)
```

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

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

3. Model Evaluation:

○ Evaluate the system's precision, recall, and F1-score.

```
y_pred = model.predict(X_test)
```

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
```

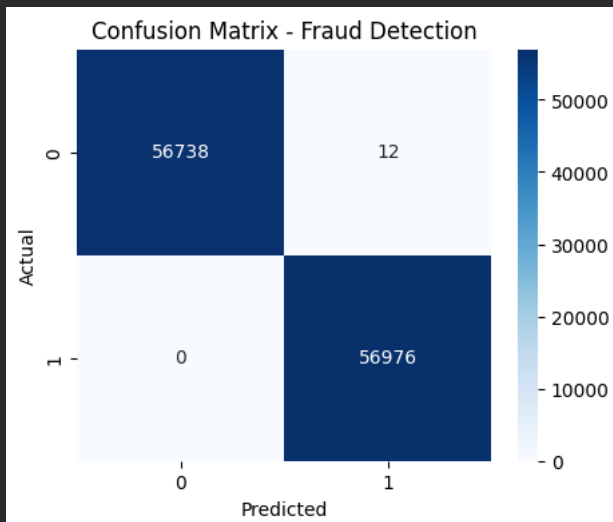
```
Accuracy: 0.9998944832316269
```

```
Classification Report:
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56750
1	1.00	1.00	1.00	56976
accuracy			1.00	113726
macro avg	1.00	1.00	1.00	113726
weighted avg	1.00	1.00	1.00	113726

```
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Fraud Detection")
plt.show()
```



4. Testing Interface:

○ Create a simple interface (e.g., a command-line input) to test the fraud detection system.

```
def test_random_transaction():
    idx = random.randint(0, len(df)-1)
    transaction = df.iloc[idx]

    X_input = transaction.drop("Class").values.reshape(1, -1)
    X_input_scaled = scaler.transform(X_input)

    prediction = model.predict(X_input_scaled)[0]

    print("Transaction Index:", idx)
    print("Actual:", "FRAUD" if transaction["Class"] == 1 else "LEGIT")

    if prediction == 1:
        print("Model Prediction: Fraudulent Transaction Detected!")
    else:
        print("Model Prediction: Legitimate Transaction")
```

```
test_random_transaction()
```

```
Transaction Index: 219809
Actual: LEGIT
Model Prediction: Legitimate Transaction
```

```
def test_smart_transaction():
    # 50% chance fraud, 50% legit
```

```
if random.random() < 0.5:
    transaction = df[df["Class"] == 1].sample(1)
    actual = "FRAUD"
else:
    transaction = df[df["Class"] == 0].sample(1)
    actual = "LEGIT"

X_input = transaction.drop("Class", axis=1)
X_input_scaled = scaler.transform(X_input)

prediction = model.predict(X_input_scaled)[0]

print("Actual Transaction:", actual)

if prediction == 1:
    print("Model Prediction: Fraudulent Transaction Detected!")
else:
    print("Model Prediction: Legitimate Transaction")
```

```
test_smart_transaction()
```

```
Actual Transaction: FRAUD
Model Prediction: Fraudulent Transaction Detected!
```

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.