Loading and Preparing Data

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
data = pd.read_csv('creditcard.csv')
print(data.head())
print("\nMissing values in each column:")
print(data.isnull().sum())
data = data.fillna(0)
```

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

         V8        V9  ...       V21       V22       V23       V24       V25  \
0  0.098698  0.363787  ... -0.018307  0.277838 -0.110474  0.066928  0.128539
1  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288 -0.339846  0.167170
2  0.247676 -1.514654  ...  0.247998  0.771679  0.909412 -0.689281 -0.327642
3  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321 -1.175575  0.647376
4 -0.270533  0.817739  ... -0.009431  0.798278 -0.137458  0.141267 -0.206010

        V26       V27       V28  Amount  Class
0 -0.189115  0.133558 -0.021053  149.62    0.0
1  0.125895 -0.008983  0.014724    2.69    0.0
2 -0.139097 -0.055353 -0.059752  378.66    0.0
3 -0.221929  0.062723  0.061458  123.50    0.0
4  0.502292  0.219422  0.215153   69.99    0.0

[5 rows x 31 columns]

Missing values in each column:
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       1
V14       1
V15       1
V16       1
V17       1
V18       1
V19       1
V20       1
V21       1
V22       1
V23       1
V24       1
V25       1
V26       1
V27       1
V28       1
Amount    1
Class     1
dtype: int64
```

Understanding the Data

```
# The 'Class' column tells us if it's fraud (1) or not (0)
X = data.drop('Class', axis=1)
y = data['Class']
fraud_cases = y.value_counts()[1]
```

```
normal_cases = y.value_counts()[0]
print(f"\nNormal transactions: {normal_cases}")
print(f"Fraud transactions: {fraud_cases}")
print(f"Fraud percentage: {fraud_cases/(normal_cases+fraud_cases)*100:.2f}%")
```

```
Normal transactions: 7948
Fraud transactions: 25
Fraud percentage: 0.31%
```

Splitting the Data (Training and Testing)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(f"\nTraining set size: {len(X_train)}")
print(f"Testing set size: {len(X_test)}")
```

```
Training set size: 6378
Testing set size: 1595
```

We choose to Random Forest Model So we need to Train it

```
model = RandomForestClassifier(n_estimators=50, random_state=42)
model.fit(X_train, y_train)
print("\nModel trained successfully!")
```

```
Model trained successfully!
```

Evaluating the model performances

```
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"\nModel accuracy: {accuracy*100:.2f}%")
fraud_predictions = predictions[y_test == 1]
correct_fraud = sum(fraud_predictions == 1)
total_fraud = len(fraud_predictions)
print(f"Caught {correct_fraud} out of {total_fraud} fraud cases")
```

```
Model accuracy: 100.00%
Caught 2 out of 2 fraud cases
```

Our Model is fully Accurate we can save it!

```
import joblib
joblib.dump(model, 'fraud_detection_model.joblib')
print("\nModel saved as 'fraud_detection_model.joblib'")
```

```
Model saved as 'fraud_detection_model.joblib'
```

trying another differnt Model (Logistic Regression) training it and then we can evaluate the performance.

```
from sklearn.linear_model import LogisticRegression
logreg_model = LogisticRegression(max_iter=1000)
logreg_model.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000)
```

```
y_pred = logreg_model.predict(X_test)
print("Model Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
```

```
print("\nSample predictions:")
print(results.head(20))
```

```
Model Accuracy: 0.9993730407523511

Confusion Matrix:
[[1592    1]
 [   0    2]]

Sample predictions:
      Actual  Predicted
7560     0.0        0.0
1405     0.0        0.0
5196     0.0        0.0
2087     0.0        0.0
3337     0.0        0.0
1302     0.0        0.0
5030     0.0        0.0
737      0.0        0.0
7528     0.0        0.0
7391     0.0        0.0
2132     0.0        0.0
6683     0.0        0.0
7016     0.0        0.0
3333     0.0        0.0
7866     0.0        0.0
518      0.0        0.0
6516     0.0        0.0
7088     0.0        0.0
3620     0.0        0.0
5962     0.0        0.0
```

Start coding or generate with AI.