Load the dataset

In [1]:
```
import pandas as pd
```

In [2]:
```
# Load the dataset
data  = "creditcard.csv"
data = pd.read_csv(data)
```

In [3]:
```
# Display basic information about the dataset
print(data.info())

# Display the first few rows of the dataset
print(data.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
None
   Time        V1        V2        V3        V4        V5        V6        V7  \
0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
```

```
1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4   2.0 -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
1  0.125895 -0.008983  0.014724    2.69      0
2 -0.139097 -0.055353 -0.059752  378.66      0
3 -0.221929  0.062723  0.061458  123.50      0
4  0.502292  0.219422  0.215153   69.99      0

[5 rows x 31 columns]
```

Data Preprocessing

In [4]:

```
from sklearn.preprocessing import StandardScaler

# Standardize the 'Amount' column
scaler = StandardScaler()
data['Amount'] = scaler.fit_transform(data['Amount'].values.reshape(-1, 1))

# Drop unnecessary columns (if any)
# For example, if 'Time' is not relevant, you can drop it
data = data.drop(['Time'], axis=1)
```

In [5]:

```
pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in c:\users\h s computer s hyd\anaconda3
\lib\site-packages (0.11.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\h s computer s hyd\anaco
nda3\lib\site-packages (from imbalanced-learn) (2.2.0)
Requirement already satisfied: joblib>=1.1.1 in c:\users\h s computer s hyd\anaconda3\li
b\site-packages (from imbalanced-learn) (1.3.2)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\h s computer s hyd\anacon
da3\lib\site-packages (from imbalanced-learn) (1.3.2)
Requirement already satisfied: scipy>=1.5.0 in c:\users\h s computer s hyd\anaconda3\lib
\site-packages (from imbalanced-learn) (1.7.1)
Requirement already satisfied: numpy>=1.17.3 in c:\users\h s computer s hyd\anaconda3\li
b\site-packages (from imbalanced-learn) (1.22.4)
Note: you may need to restart the kernel to use updated packages.
```

Handle Class Imbalance

In [6]:

```
# Count the number of fraudulent and genuine transactions
fraud_count = data['Class'].sum()
genuine_count = len(data) - fraud_count

# Display class distribution
```

```
print(f"Fraudulent transactions: {fraud_count}")
print(f"Genuine transactions: {genuine_count}")
```

```
Fraudulent transactions: 492
Genuine transactions: 284315
```

Split the dataset

In [7]:

```
from sklearn.model_selection import train_test_split

# Separate features (X) and target variable (y)
X = data.drop('Class', axis=1)
y = data['Class']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4
```

In [8]:

```
from sklearn.utils import resample

# Combine X_train and y_train for resampling
combined_data = pd.concat([X_train, y_train], axis=1)

# Separate majority and minority classes
majority_class = combined_data[combined_data['Class'] == 0]
minority_class = combined_data[combined_data['Class'] == 1]

# Upsample the minority class
minority_upsampled = resample(minority_class, replace=True, n_samples=len(majority_clas

# Combine the upsampled minority class with the majority class
upsampled_data = pd.concat([majority_class, minority_upsampled])

# Separate features (X_resampled) and target variable (y_resampled)
X_resampled = upsampled_data.drop('Class', axis=1)
y_resampled = upsampled_data['Class']

# Check the class distribution after upsampling
print("Class distribution after upsampling:")
print(y_resampled.value_counts())
```

```
Class distribution after upsampling:
0    227451
1    227451
Name: Class, dtype: int64
```

Build a Random Forest Classifier

In [9]:

```
from sklearn.ensemble import RandomForestClassifier

# Initialize and train the model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_resampled, y_resampled)
```

Out[9]:

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

Evaluate the model

In [10]:

```
from sklearn.metrics import classification_report, confusion_matrix

# Make predictions on the test set
y_pred = model.predict(X_test)

# Display classification report and confusion matrix
print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.99      0.80      0.88        98

    accuracy                           1.00     56962
   macro avg       0.99      0.90      0.94     56962
weighted avg       1.00      1.00      1.00     56962

Confusion Matrix:
[[56863     1]
 [   20    78]]
```