STEP - 1

Importing Libraries :-

Need to import libraries because they are pre-built tools that helps us with data analysis , visualization , machine learning.

Libraries are like pre-written code. Instead of writing code from scratch , we can use libraries.

```python
import pandas as pd  # Helps to handle and process data (like Excel)
import numpy as np  # Helps in numerical calculations
import matplotlib.pyplot as plt  # Helps in creating graphs and charts
import seaborn as sns  # Makes graphs look better
from sklearn.model_selection import train_test_split  # Helps to divide data into training an
from sklearn.ensemble import RandomForestClassifier  # A machine learning model used for frau
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report  # Helps
```

`pandas` helps us **load and analyze data.**

`numpy` helps in **mathematical calculations.**

`matplotlib` and `seaborn` help in **visualizing data** (graphs). `sklearn.model_selection` helps **split data** for training and testing.

`sklearn.ensemble.RandomForestClassifier` is **the machine learning model** that detects fraud.

`sklearn.metrics` helps to **check if our model is working correctly.**

STEP - 2

Loading the Dataset (Getting the Data)

```python
df = pd.read_csv('creditcard.csv')  # Load the dataset
print(df.head())  # Show the first 5 rows of the data
```

`pd.read_csv('creditcard.csv')` loads the data from a **CSV file** (like an Excel sheet).

`df.head()` shows **the first 5 rows** so we can understand what the data looks like.

💡 **Example of Data:**

Time	Amount	Class
0	150.00	0

1	20.00	0
2	500.00	1

`Class = 0` means **Not Fraud** (Genuine Transaction).
`Class = 1` means **Fraud** (Fake Transaction).

STEP - 3

Checking Data Information

Before training the model, we must **check if our data is complete** and understand its structure.

```python
print(df.info())  # Check for missing values and column details
print(df.describe())  # Show basic statistics of the data
```

`df.info()` shows **if any data is missing** and what type of data is present.

`df.describe()` gives **summary statistics** (like minimum, maximum, and average values).

STEP - 4

Checking Fraud vs Genuine Transactions

We check **how many fraud transactions** are in the dataset.

```python
print(df['Class'].value_counts())  # Count how many transactions are fraud vs genuine
```

```scss
0 → 284315 (Genuine transactions)
1 → 492 (Fraud transactions)
```

It means the dataset has **many more genuine transactions than fraud** (called an **imbalanced dataset**).
We need to **fix this imbalance** later, or the model may not detect fraud correctly!

STEP- 5

Splitting Data for Training and Testing

We separate the dataset into **features (X)** and **target (y)**.

- **Features (X)** → All columns except the "Class" column.
- **Target (y)** → The "Class" column (0 or 1).

Then, we split the data into **Training (80%)** and **Testing (20%)**.

```python
X = df.drop(columns=['Class'])  # Remove the target column, keep only features
y = df['Class']  # The target variable (fraud or not)

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

`X` contains **all transaction details except Class**.
`y` contains **only Class (fraud or not fraud)**.
`train_test_split(X, y, test_size=0.2)` → 80% data for **training**, 20% for **testing**.

STEP - 6

Training the Machine Learning Model

Now, we train a **Random Forest model**, which is a powerful algorithm for detecting fraud.

```python
model = RandomForestClassifier(n_estimators=100, random_state=42)  # Create the model
model.fit(X_train, y_train)  # Train the model with training data
```

`RandomForestClassifier(n_estimators=100)` creates a model with **100 decision trees**.
`model.fit(X_train, y_train)` trains the model using **training data**.

The model **learns patterns** from the data to identify fraud transactions.

STEP - 7

Making Predictions

After training, we **test the model** using our test data.

```python
y_pred = model.predict(X_test)  # Predict on test data
```

`model.predict(X_test)` uses the trained model to **predict fraud or genuine** for test data**.**

STEP - 8

Evaluating the Model (Checking Accuracy)

We check if the model **correctly predicts fraud transactions.**

```python
accuracy = accuracy_score(y_test, y_pred)  # Check accuracy
print(f'Accuracy: {accuracy * 100:.2f}%')

print(confusion_matrix(y_test, y_pred))  # Show confusion matrix
print(classification_report(y_test, y_pred))  # Show precision, recall, F1-score
```

`accuracy_score(y_test, y_pred)` → **Measures how well the model performed.**
`confusion_matrix(y_test, y_pred)` → **Shows correct and incorrect predictions.**
`classification_report(y_test, y_pred)` → **Shows detailed performance metrics.**

Higher **accuracy** means a better fraud detection model!

STEP - 9

Visualizing the Results

We create a **graph** to see where the model made mistakes.

```python
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

It creates a **heatmap** to visualize **how many fraud cases were correctly detected.**

Final Summary

1. **Loaded the credit card transactions dataset.**
2. **Checked for missing values and imbalances.**
3. **Divided the dataset into training (80%) and testing (20%).**
4. **Trained a Random Forest model to detect fraud.**
5. **Made predictions on test data.**
6. **Checked model accuracy** and **visualized results.**