There are a few publicly available datasets that can be employed for credit card fraud detection. Commonly used Datasets Dataset from Kaggle Credit Card Fraud Detection Description:

* The dataset contains transactions made by European cardholders in the month of September 2013. 284,807 transactions 492 frauds detected Link: (https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud) Kaggle Credit Card Fraud Detection IEEE-CIS Fraud Detection Dataset : Description: Same thing as 2nd dataset, but with 2 more features; transaction details and identity of the user.
* IEEE-CIS Fraud Detection- Cup of Data- Link Financial Datasets with Synthetic data for Fraud Detection: link-(https://www.kaggle.com/c/ieee-fraud-detection).
* Synthesized data based on financial fraud scenarios Synthetic Financial Datasets Link-(https://www.kaggle.com/ntnu-testimon/paysim1)
* In this example, we will utilize the first dataset (Kaggle Credit Card Fraud Detection). In this iPython notebook tutorial, we will implement a Python program to train and evaluate the performance of a simple logistic regression model as well as generate a confusion matrix.

Algorithm:

1. Load the dataset.
2. If necessary preprocess the data.
3. Split the dataset into training and testing sets.
4. Train a logistic regression model.
5. Evaluate the mode and output the confusion matrix.

Python program:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix, classification\_report, precision\_recall\_curve, auc

# Load the dataset

try:

df = pd.read\_csv('C:/Users/Hemashree H/Desktop/sample/creditcard.csv')

except FileNotFoundError:

print("The dataset file 'creditcard.csv' was not found. Please ensure it is in the working directory.")

exit()

# Explore the dataset (optional)

print(df.head())

# Separate features and target variable

X = df.drop(columns=['Class'])

y = df['Class']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42, stratify=y)

# Train a logistic regression model

model = LogisticRegression(max\_iter=1000, solver='liblinear')

model.fit(X\_train, y\_train)

# Predict on the test set

y\_pred = model.predict(X\_test)

y\_pred\_proba = model.predict\_proba(X\_test)[:, 1]

# Evaluate the model

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

class\_report = classification\_report(y\_test, y\_pred)

# Print the results

print("Confusion Matrix:")

print(conf\_matrix)

print("\nClassification Report:")

print(class\_report)

# Plot confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Fraud', 'Fraud'], yticklabels=['Not Fraud', 'Fraud'])

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

# Plot Precision-Recall curve

precision, recall, thresholds = precision\_recall\_curve(y\_test, y\_pred\_proba)

pr\_auc = auc(recall, precision)

plt.figure(figsize=(8, 6))

plt.plot(recall, precision, marker='.')

plt.xlabel('Recall')

plt.ylabel('Precision')

plt.title(f'Precision-Recall Curve (AUC = {pr\_auc:.2f})')

plt.show()

Output:





