**Credit Card Fraud Detection**

**NAME:RAMYA**

**REGISTER NUMBER:112021104025**

**PHASE 5: Project Documentation and Submission**



* **Problem statements:**
* **Objective:** the goal is to develop an efficient credit card fraud detection system using a kaggle dataset .the dataset consists of credit card transactions labeled as fraudulent transactions in real time to minimize financial losses and maintain trust among credit card users.
* **Challenges:**
* Imbalanced dataset with a significantly low number of fraudlent transactions.
* Developing a model that can accurately distinguish between legitimate and fraudulent transactions.
* Ensuring the model can adapt to evolving fraud patterns.
* Maintaining low false positive rates to minimize inconveniences for legitimate cardholders.
* **Design thinking process:**
* **Empathize:** understand the pain points and challenges associates with credit card fraud. Analyze user experiences and the impact of fraudulent activities on both customers and financial institutions.
* **Define:** clearly define the problem, considering the available dataset from kaggle, which includes transaction information.define the metrics for success, such as accuracy , precision, recall, and F1 score.
* **Ideate:** brainstorm various approaches to tackle credit card fraud detection using machine learning techniques. Consider feature engineering , different algorithms such as logistic regression, random forests, neural networks , and ensemble methods to enhance the model’s accuracy.
* **Prototype:** develop and test multiple machine learning models using the Kaggle dataset . this involves data preprocessing , feature selection, model development, and evaluation. Experiment with various algorithms and hyper parameters to optimize the model’s performance.
* **Test:** evaluate the models using metrics like precision, recall, F1 score and ROC curves to choose the best-performing model. Test the model on unseen data to validate its effectiveness.
* **Implement:** deploy the chosen model for real-time monitoring of credit card transactions. Ensure continuous monitoring and updates to adapt to new fraud patterns.
* **Phases of development :**
* **Data exploration and understanding:** acquire the kaggle datasset, explore its features , and understand the data distribution . Address any missing values or anomalies in the dataset.
* **Data preprocessing and feature engineering:** clean the dataset , handle missing values , normalize feataures, and engineer new relevant features . address the class imbalance issue by oversampling, undersampling , or using techniques like SMOTE(Synthetic Minority Over-sampling Techinque).
* **Model development:** train various machine learning models on the preprocessed data. Experiment with different algorithms and ensemble methods. Tune hyperparameters to enhance model performance.
* **Model evaluation:** evaluate models using metrics such as accuracy, precision , recall, F1 score , and ROC curves. Select the model that best balances between identifying fraudulent transactions and minimizing false positives.
* **Model deployment and monitoring:** integrate the chosen model into the credit card transaction system for real-time monitoring. Continuously monitor the model’s performance and make necessary updates as new fraud patterns emerge.
* **Documentation and reporting:** document the entire development process, including methodologies, challenges faced, model performance, and implementation details. Provide a comprehensive report for stakeholders, compliance, and further research.
* **Feedback iteration:** gather feedback from the implemented system, assess any shortcomings or new patterns of fraudulent activities, and iterate on the model to improve its accuracy and adaptability.
* **Dataset Explanation:**
* Credit card fraud detection is the collective term for the policies, tools, methodologies, and practices that credit card companies and financial institutions take to combat identity fraud and stop fraudulent transactions.
* This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.
* It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data.
* Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'.
* Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset.
* The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning.
* Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.
* The dataset will contain various features or attributes related to each transaction.
* Transaction amount
* Transaction date and time
* Merchant information
* Customer account details
* Location information
* Card type
* Transaction category
* **Data preprocessing steps:**
* **Data loading and inspection:** load the dataset and inspect the data to understand the features, their distributions ,and the class imbalance.
* **Handling missing values:** check for missing or null values in the dataset, if there are any, impute or drop these values based on the context of the data.
* **Scaling ‘Time’ and ‘Amount’:** Scale the’Time’ and ‘Amount’ features to bring them to a similar scale as the other features in the dataset. This can be done using standard scaling methods .eg.,Min-Max Scaling or Standard Scaling.
* **Dealing with imbalanced data:** address the class imbalance issue by using techniques like oversampling the minority class, undersampling the majority class, or utilizing synthetic data generation techniques like SMOTE(Synthetic Minority Over-Sampling Technique).
* **Choice of Machine Learning Algorithm:**
* **Logistic Regression:** logistic regression is a common choice due to its simplicity and interpretability. It’s a good starting point for binary classification problem like fraud detection.
* **Advantages:** logistic regression is a commonly used algorithm in fraud detection due to its interpretability and ability to provide probabilities for binary classification.
* **Consideration:** Despite its simplicity, logistic regression might struggle with highly complex nonlinear relationships in the data.
* **Random Forests: random forests are an ensemble learning method that can handle imbalanced data sets and complex relationship between features.**
* Advantages**: random forests can handle imbalanced data , are less prone to overfitting , and can capture nonlinear relationships in the data.**
* Consideration**: while they generally perform well, they might be computationally more expensive and less interpretable than logistic regression.**
* Advantages**: logistic regression is a commonly used algorithm in fraud detection due to its interpretable and ability to provide probabilities for binary classification problems . It’s relatively fast and can handle datasets.**
* Consideration**: despite its simplicity , logistic regression might struggle with highly complex nonlinear relationship in the data.**
* **Gradient boosting algorithms : These algorithms perform well in dealing with imbalanced datasets and can often yield accuracy by combining weak learnears into a strong model.**
* **Suport Vector Machine(SVM): SVMs work well in high-dimensional spaces and are effective for binary classification tasks like fraud detection,**
* **Neural networks: deep learnings, particulary neural networks, can learn intricate patterns but might require more data and computational resources.**

**The choice of algorithms often involves experimentation and comparison based on how well each model can handle the imbalanced data, its ability to distinguish fraudulent transactions, computational efficiency, and the interpretability of the results.**

Evaluation Metrics:

Given the class imbalance in the dataset , standard accuracy alone may not be a reliable metric to assess model performance . instead , a combination of several evaluation metrics is often used to gauge the model’s effectiveness in credit card fraud detection . key evaluation metrics include.

* Precision and recall :

Precision measures the accuracy of the model in identifying truly fraudulent transactions among all the instances flagged as fraud.

Recall measures the proportion of acal fraudulent transactions that are correctly identified by the model.

* F1score: the F1 score is the harmonic mean of precision and recall. It gives a balanced view of precision and recall , particularly suitable for imbalanced datasets.
* ROC Curve and AUC-ROC:

Receive Operating Charateristic(ROC) curve shows the trade-off between true positive rate and false positive rate at various threshold settings.

Area Under the Curve(AUC) of the curve summarizes the model’s performances across various threshold values. A higher AUC indicates better model performance.

* Confusion matrix :

Provides a detailed table showing true positives , true negatives , false positives , and false negatives , offering insights into model performance.

In credit card fraud detection . the emphasis is usually on minimizing false negatives while controlling false positives to avoid inconveniencing legitimate cardholders. Therefore ,a good model needs to strike a balance between precision and recall while keeping false positive rates low. The evaluation metrics help in assessing this balance and guiding the selection of the most suitable model for fraud detection on the kaggle dataset.

**Phase 5: Project Documentation & Submission**

In this part you will document your project and prepare it for submission.

 Document the credit card fraud detection project and prepare it for

.

**Dataset Link:**[**https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud**](https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)

**Credit card fraud detection:**

# # import the required packages

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import sys

import scipy

# # load the dataset using pandas

data = pd.read\_csv(r'C:\Users\Ravi\Downloads\creditcardfraud\creditcard.csv')

# # dataset exploring

print(data.columns)

# # Print the shape of the data

data = data.sample(frac=0.1, random\_state = 1)

print(data.shape)

print(data.describe())

# V1 - V28 are the results of a PCA Dimensionality reduction to protect user identities and sensitive features

# # Plot histograms of each parameter

data.hist(figsize = (20, 20))

plt.show()

# # Determine number of fraud cases in dataset

Fraud = data[data['Class'] == 1]

Valid = data[data['Class'] == 0]

outlier\_fraction = len(Fraud)/float(len(Valid))

print(outlier\_fraction)

print('Fraud Cases: {}'.format(len(data[data['Class'] == 1])))

print('Valid Transactions: {}'.format(len(data[data['Class'] == 0])))

# # Correlation matrix

corrmat = data.corr()

fig = plt.figure(figsize = (12, 9))

sns.heatmap(corrmat, vmax = .8, square = True)

plt.show()

# # Get all the columns from the dataFrame

columns = data.columns.tolist()

# # Filter the columns to remove data we do not want

columns = [c for c in columns if c not in ["Class"]]

# # Store the variable we'll be predicting on

target = "Class"

X = data[columns]

Y = data[target]

# # Print shapes

print(X.shape)

print(Y.shape)

from sklearn.metrics import classification\_report, accuracy\_score

from sklearn.ensemble import IsolationForest

from sklearn.neighbors import LocalOutlierFactor

# # define random states

state = 1

# # define outlier detection tools to be compared

classifiers = {

"Isolation Forest": IsolationForest(max\_samples=len(X),

contamination=outlier\_fraction,

random\_state=state),

"Local Outlier Factor": LocalOutlierFactor(

n\_neighbors=20,

contamination=outlier\_fraction)}

plt.figure(figsize=(9, 7))

n\_outliers = len(Fraud)

for i, (clf\_name, clf) in enumerate(classifiers.items()):

# # fit the data and tag outliers

if clf\_name == "Local Outlier Factor":

y\_pred = clf.fit\_predict(X)

scores\_pred = clf.negative\_outlier\_factor\_

else:

clf.fit(X)

scores\_pred = clf.decision\_function(X)

y\_pred = clf.predict(X)

# # Reshape the prediction values to 0 for valid, 1 for fraud.

y\_pred[y\_pred == 1] = 0

y\_pred[y\_pred == -1] = 1

n\_errors = (y\_pred != Y).sum()

# # Run classification metrics

print('{}: {}'.format(clf\_name, n\_errors))

print(accuracy\_score(Y, y\_pred))

print(classification\_report(Y, y\_pred))

# output:

Index(['Time', 'V1', 'V2', ..., 'V28', 'Amount', 'Class'], dtype='object')

(28481, 31)

Time V1 V2 ... V28 Amount Class

count 28481.000000 28481.000000 28481.000000 ... 28481.000000 28481.000000 28481.000000

mean 94838.202258 0.000081 -0.000273 ... -0.002567 88.291022 0.001720

std 47484.015786 1.959448 1.651309 ... 0.332070 250.105092 0.041443

min 0.000000 -25.162799 -72.715728 ... -15.430084 0.000000 0.000000

25% 54230.000000 -0.920373 -0.598746 ... -0.015428 5.650000 0.000000

50% 84605.000000 0.018111 0.065486 ... 0.031245 22.000000 0.000000

75% 139676.000000 1.315642 0.803892 ... 0.077577 77.050000 0.000000

max 172792.000000 2.454930 22.057729 ... 33.847808 25691.160000 1.000000

(28481, 30)

(28481,)

Isolation Forest: 71

0.99750711000316

precision recall f1-score support

0 1.00 1.00 1.00 28432

1 0.28 0.29 0.28 49

accuracy 1.00 28481

macro avg 0.64 0.64 0.64 28481

weighted avg 1.00 1.00 1.00 28481

Local Outlier Factor: 97

0.9965942207085425

precision recall f1-score support

0 1.00 1.00 1.00 28432

1 0.02 0.02 0.02 49

accuracy 1.00 28481

macro avg 0.51 0.51 0.51 28481

weighted avg 1.00 1.00 1.00 28481