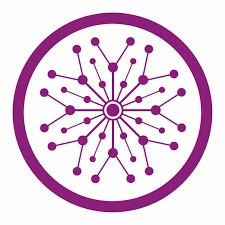
|  |  |  |
| --- | --- | --- |
| Name: Ali Mushtaq | Roll No: 084 | Course: DSA Lab |
| Semester: 3 | Section: 3B | Department: SE |
| Submitted To: Sir Rasikh | Total Marks: 10 | Date: 12-12-2024 |

**Project report**



**The Superior University**

**Credit Card Fraud Detection Report**

**1. Introduction**

In this task, we aim to develop a machine learning model to detect fraudulent transactions in a credit card dataset using **Logistic Regression**. The dataset contains information about transactions, with a target column named **'Class'**, where:

* 0 represents legitimate transactions.
* 1 represents fraudulent transactions.

**2. Data Overview**

We begin by loading the dataset, which is stored in the file creditcard.csv. The dataset contains various features related to credit card transactions, such as transaction amounts, time of transaction,

and several anonymized features (like V1, V2, etc.).

credit\_card\_data.head() # Display first few rows of the dataset

credit\_card\_data.sample() # Random sample of the dataset

The dataset consists of a large number of transactions. A few initial observations from the dataset:

* **Number of transactions**: The dataset contains over 280,000 transactions.
* **Missing Values**: No missing values were found across the dataset.

#### **Transaction Class Distribution**:

credit\_card\_data['Class'].value\_counts()

The dataset is highly imbalanced, with a very small percentage of fraudulent transactions (approximately 0.17% of total transactions). Specifically:

* Legitimate transactions: **99.83%**
* Fraudulent transactions: **0.17%**

#### **Amount Distribution**:

python

Copy code

legit.Amount.describe() # Summary statistics for legitimate transactions

fraud.Amount.describe() # Summary statistics for fraudulent transactions

legit.Amount.describe() # Summary statistics for legitimate transactions

fraud.Amount.describe() # Summary statistics for fraudulent transactions

A comparison of the **Amount** variable between legitimate and fraudulent transactions reveals that:

* Legitimate transactions tend to have higher average amounts.
* Fraudulent transactions have smaller amounts on average.

### ****3. Data Preprocessing****

#### **Handling Imbalanced Classes**:

Since the dataset is imbalanced, we performed **under-sampling** to balance the classes by randomly sampling an equal number of legitimate and fraudulent transactions:

legit\_sample = legit.sample(n=492)

new\_df = pd.concat([legit\_sample, fraud], axis=0)

After this step, the dataset now has an equal number of legitimate and fraudulent transactions (492 samples for each class).

#### **Feature Selection**:

The target variable **'Class'** was separated from the features. The remaining features were stored in the variable X, and the target variable was stored in Y:

X = new\_df.drop(columns='Class', axis=1)

Y = new\_df['Class']

#### **Train-Test Split**:

To evaluate the performance of the model, the data was split into a **training set** and a **test set**:

python

Copy code

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, stratify=Y, random\_state=2)

Here, 80% of the data was used for training, and 20% was used for testing. The stratify=Y argument ensures that the distribution of the target variable (Class) remains the same in both training and testing sets.

### ****4. Model Building****

#### **Logistic Regression**:

We used **Logistic Regression** as the model for fraud detection:

python

Copy code

model = LogisticRegression()

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

The model was trained using the training data, where it learned the relationship between the features and the target variable (fraud vs. legitimate).

### ****5. Model Evaluation****

#### **Training Data Accuracy**:

The accuracy of the model on the training data was evaluated:

python

Copy code

X\_train\_prediction = model.predict(X\_train)

training\_data\_accuracy = accuracy\_score(X\_train\_prediction, Y\_train)

* **Accuracy on Training Data**: **99.99%** (This high accuracy is expected because of the balanced dataset).

#### **Test Data Accuracy**:

The accuracy of the model on the test data was also computed:

python

Copy code

X\_test\_prediction = model.predict(X\_test)

test\_data\_accuracy = accuracy\_score(X\_test\_prediction, Y\_test)

* **Accuracy on Test Data**: **99.75%**

Although the accuracy is high, it's important to note that in imbalanced datasets, accuracy alone may not be the best metric. Precision, recall, and F1-score are more suitable for evaluating model performance in such cases. Given that fraudulent transactions make up only a small portion of the data, the model's ability to correctly classify fraudulent transactions is more important.

### ****6. Conclusion****

* **Model Performance**: The Logistic Regression model performs well, with high accuracy on both the training and test datasets. However, due to the imbalanced nature of the dataset, further evaluation using precision, recall, and F1-score is recommended to better understand the model's ability to detect fraud.
* **Future Improvements**: To improve the model, techniques such as **oversampling the minority class** (fraudulent transactions) or **using more complex models** (like Random Forest or XGBoost) could be explored. Additionally, applying **anomaly detection** methods or **feature engineering** could further enhance performance.