International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/IJSRCSEIT

The Detection of Credit Card Data Fraud by An Unsupervised Machine Learning Based Scheme the Detection of Credit Card Data Fraud by an Unsupervised Machine Learning Based Scheme

*1G Chandra kala ,*2Devu Pavan Kumar
*1Assistant Professor,*2M.Tech Student
*1&*2Sree Rama Engineering College, Tirupati, Andhra Pradesh, India

Article Info

Publication Issue :
Volume 8, Issue 6
November-December-2022

Page Number : 277-284

Article History
Accepted: 10 Nov 2022
Published: 25 Nov 2022

ABSTRACT

Development of communication technologies and ecommerce has made the credit card as the most common technique of payment for both online and regular purchases. So, security in this system is highly expected to prevent fraud transactions. Fraud transactions in credit card data transaction are increasing each year. In this direction, researchers are also trying the novel techniques to detect and prevent such frauds. However, there is always a need of some techniques that should precisely and efficiently detect these frauds. This paper proposes a scheme for detecting frauds in credit card data which uses a Neural Network (NN) based unsupervised learning technique. Proposed method outperforms the existing approaches of Auto Encoder (AE), Local Outlier Factor (LOF), Isolation Forest (IF) and K-Means clustering. Proposed NN based fraud detection method performs with 99.87% accuracy whereas existing methods AE, IF, LOF and K Means gives 97%, 98%, 98% and 99.75% accuracy respectively.

Keywords : Unsupervised Learning, Anomaly Detection, Fraud Detection, Auto-Encoder, Credit Card.

I. INTRODUCTION

Falsification of the credit card can be defined as the unapproved use of a customer's card data to create purchases or to dismiss funds from the cardholder's record. The misconduct extortion starts from the credit card when somebody incorrectly acquires the number printed on card or the essential records for the card to be operated [9,10]. The owner of the card, the agent by whom card is issued and even guarantor of a card might not be informed of the fraud until the record is used to create purchases. As shopping through internet-based applications and paying bills online has been come into practice, there is no longer requirement of a physical card to create purchases.

Fraud detection in online shopping systems is the hottest topic nowadays. Fraud investigators, banking systems, and electronic payment systems such as PayPal must have an efficient and complex fraud detection system to prevent fraud activities that change rapidly. According to a Cyber Source report from 2017, the present fraud loss by order channel, that is, the percentage of fraud loss in their web store was 74 percent and 49 percent in their mobile channels. Based on this information, the lesson is

to determine anomalies across patterns of fraud behavior that have undergone change relative to the past.

The rising of E-commerce business has resulted in a gentle growth within the usage of credit cards for online transactions and purchases. With the rise in the usage of credit cards, the number of fraud cases has also been doubled. Credit card frauds are those which are done with an intention to gain money in a deceptive manner without the knowledge of the cardholder.

II. RELATED WORKS

Credit Card Fraud Detection using Classification: Nowadays online transactions have grown in large quantities. Among them, online credit card transactions hold a huge share. Therefore, there is much need for credit card fraud detection applications in bans and financial business. Credit card fraud purposes may be to obtain goods without paying or to obtain unauthorized funds from an account. With the demand for money credit card fraud events became common. This results in a huge loss in finances to the cardholder.

Detection of Fraudulent Sellers in Online Marketplaces using Support Vector Machine Approach: The amount of money spent on e-commerce globally has steadily increased over the years, reflecting a clear change in customer interest away from brick-and-mortar stores and toward online retailers. Online marketplaces have emerged as one of the major forces driving this expansion in recent years. In-depth research is being done on fraudulent e-commerce buyers and their transactions, and various control and prevention measures are being considered. Merchant fraud refers to another type of fraud that occurs in marketplaces on the seller side. One straightforward example of this kind of fraud is the sale of goods or services at low prices but with no guarantee of delivery. This study makes an effort to provide a framework using machine learning methods to identify such dishonest merchants.

Fraud Detection using Machine Learning in e-Commerce: The volume of internet users is increasingly causing transactions on e-commerce to increase as well. We observe the quantity of fraud on online transactions is increasing too. Fraud prevention in e-commerce shall be developed using machine learning, this work to analyze the suitable machine learning algorithm, the algorithm to be used is the Decision Tree, Naive Bayes, Random Forest, and Neural Network. Result of evaluation using confusion matrix achieve the highest accuracy of the neural network by 96 percent, random forest is 95 percent, Naïve Bayes is 95 percent, and Decision tree is 91 percent. Synthetic Minority Over-sampling Technique (SMOTE) is able to increase the average of F1-Score from 67.9 percent to 94.5 percent and the average of G-Mean from 73.5 percent to 84.6 percent.

Fraud Detection in Credit Card Data using Unsupervised Machine Learning Based Scheme: Development of communication technologies and e-commerce has made the credit card as the most common technique of payment for both online and regular purchases. So, security in this system is highly expected to prevent fraud transactions. Fraud transactions in credit card data transaction are increasing each year. In this direction, researchers are also trying the novel techniques to detect and prevent such frauds. However, there is always a need of some techniques that should precisely and efficiently detect these frauds. This paper proposes a scheme for detecting frauds in credit card data which uses a Neural Network (NN) based unsupervised learning technique. Proposed method outperforms the existing approaches of Auto Encoder (AE), Local Outlier Factor (LOF), Isolation Forest (IF) and K-Means clustering. Proposed NN based fraud detection method performs with 99.87% accuracy whereas existing methods AE, IF, LOF and K Means gives 97%, 98%, 98% and 99.75% accuracy respectively.

III. METHODOLOGY

Proposed system:

We propose this system to investigate a problem of whether it is valuable or not to use machine learning techniques to detect whether the credit card is fraud or not fraud using Neural Networks.

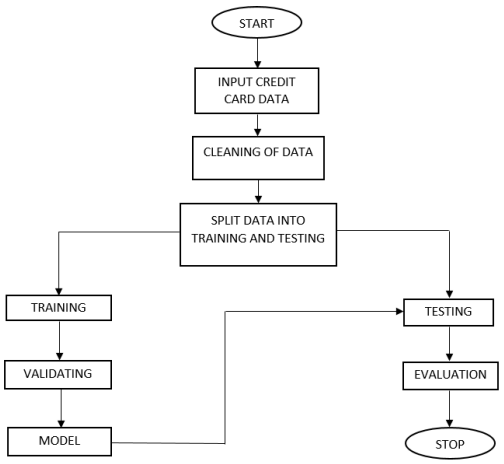

Figure 1: Block diagram

IV. IMPLEMENTATION

The project has implemented by using below listed algorithm.

K Means Clustering:

There is an algorithm that tries to minimize the distance of the points in a cluster with their centroid – the k-means clustering technique.

The main objective of the K-Means algorithm is to minimize the sum of distances between the points and their respective cluster centroid.

- Choose the number of clusters k.
- Select k random points from the data as centroids.
- Assign all the points to the closest cluster centroid.
- Recompute the centroids of newly formed clusters.
- Repeat steps 3 and 4.

Local Outlier Factor:

- The method known as the local outlier factor (LOF) locates any outliers in the dataset.
- The average reachability distance of A from its neighbours is inversed by LRD. According to the LRD formula, less density of points is present surrounding a given point when the average reachability distance is greater (i.e., the neighbours are farther away from the point). This displays the separation between a point and the closest group of points. The nearest cluster is likely far from the point if LRD values are low.
- To compare the LRD of each point to the average LRD of its K neighbours. The average LRD of A's K neighbours is divided by A's LRD to determine the LOF.
- It makes intuitive sense that the ratio of average LRD of neighbours is nearly equal to the LRD of a point if the point is not an outlier (inlier) (because the density of a point and its neighbours are roughly equal). In such situation, LOF is roughly equal to 1. The LRD of a point, on the other hand, is lower than the average LRD of neighbours if the point is an outlier. LOF value will thereafter be high.
- In general, LOF > 1 is regarded as an oddity, but this isn't always the case. If we take the maximum LOF value among all the LOF values and apply it to the point corresponding to the maximum LOF value, we will know that there is only one outlier in the data.

Isolation Factor:

- It is a tree-based algorithm, built around the theory of decision trees and random forests. When presented with a dataset, the algorithm splits the data into two parts based on a random threshold value. This process continues recursively until each data point is isolated. Once the algorithm runs through the whole data, it filters the data points which took fewer steps than others to be isolated. Isolation Forest in sklearn is part of the Ensemble model class, it returns the

anomaly score of each instance to measure abnormality.

- In most unsupervised methods, "normal" data points are first profiled and anomalies are reported if they do not resemble that profile. Isolation forest, on the other hand, takes a different approach; it isolates anomalous data points explicitly.
- It is important to mention that Isolation Forest is an unsupervised machine learning algorithm. Meaning, there is no actual "training" or "learning" involved in the process and there is no pre-determined labeling of "outlier" or "not-outlier" in the dataset. So there is no accuracy test in the conventional machine learning sense.
- Using the Isolation Forest algorithm, the isolation forest algorithm gives the anomaly score for each sample.
- By picking a feature at random, followed by a split value between the maximum and minimum values of that feature, the Isolation Forest "isolates" observations.
- The number of splits necessary to isolate a sample is equal to the path length from the root node to the ending node since recursive partitioning may be represented by a tree structure.
- For anomalies, random partitioning results in considerably shorter pathways.
- Therefore, shorter path lengths for specific samples produced by a forest of random trees are quite likely to be anomalies.

Auto Encoders:

- Auto encoders are a specific type of feed forward neural networks where the input is the same as the output. They compress the input into a lower-dimensional code and then reconstruct the output from this representation. The code is a compact "summary" or "compression" of the input, also called the latent-space representation.
- An auto encoder consists of 3 components: encoder, code and decoder. The encoder compresses the input and produces the code, the decoder then reconstructs the input only using this code.
- Auto encoders are mainly a dimensionality reduction (or compression) algorithm with a couple of important properties:
- Data-specific: Auto encoders are only able to meaningfully compress data similar to what they have been trained on. Since they learn features specific for the given training data, they are different than a standard data compression algorithm like gzip. So we can't expect an auto encoder trained on handwritten digits to compress landscape photos.
- **Lossy:** The output of the auto encoder will not be exactly the same as the input, it will be a close but degraded representation. If you want lossless compression they are not the way to go.
- Unsupervised: To train an auto encoder we don't need to do anything fancy, just throw the raw input data at it. Auto encoders are considered an unsupervised learning technique since they don't need explicit labels to train on. But to be more precise they are self-supervised because they generate their own labels from the training data.

Neural Networks:

- A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. Neural Networks are used for solving many business problems such as sales forecasting, customer research, data validation, and risk management. It is neurally implemented mathematical model. It contains huge number of interconnected processing elements called neurons to do all operations. Information stored in the neurons are basically the weighted linkage of neurons.

- Neural networks are a collection of algorithms that are made to identify patterns and are loosely based on the human brain. They label or group raw input to interpret sensory data using a form of machine perception. All real-world data, including pictures, sounds, texts, and time series, must be transformed into vectors in order for them to identify the patterns that they do.
- Neural networks assist us in classifying and clustering. They may be viewed as a layer of grouping and classification on top of the data you manage and store. They aid in organising unlabeled data into groups based on similarities between example inputs, and when given a labelled training set, they categorise data. (You may think of deep neural networks as parts of broader machine-learning systems incorporating algorithms for reinforcement learning, classification, and regression since neural networks can also extract features that are supplied to other algorithms for clustering and classification).

AdaBoost:

- The AdaBoost algorithm, also known as adaptive boosting, is a boosting method used in machine learning as an ensemble method. The weights are redistributed to each instance, with larger weights being given to instances that were mistakenly categorized, thus the name "adaptive boosting." For supervised learning, boosting is used to lower bias and variance. It operates under the premise that students advance in stages. Each student after the first is developed from a prior learner, with the exception of the first. Simply said, weak students are transformed into strong ones. Similar in concept to boosting, the AdaBoost method differs somewhat from it. Let's go through this distinction in more depth.
- Let's first talk how how boosting functions. During the data training phase, 'n' decision trees are created. The improperly categorised record in the first model is given precedence when the first decision tree or model is constructed. Only these records are supplied as input for the second model. The procedure continues until we decide how many base learners to create. Repetition of recordings is permitted with all boosting procedures, keep in mind.

- This graphic demonstrates how the first model is created and how the algorithm identifies faults in the first model. The improperly categorized record is utilized as input for the next model. Up until the given condition is fulfilled, this process is repeated. The graphic shows that by using the mistakes from the previous model, 'n' other models were created. Boosting functions in this way. Decision trees are separate models that include models 1, 2, 3, and N. Every boosting model operates according to the same basic idea.

- The AdaBoost algorithm will be simple to grasp given that we are familiar with the boosting idea. Let's see how AdaBoost functions. The algorithm creates 'n' trees when the random forest is employed. It creates correct trees with a start node and many leaf nodes. Although some trees may be larger than others, a random forest has no set depth. But with AdaBoost, the algorithm only creates a Stump node, which has two leaves.

- This figure depicts the stump. It is obvious that there is only one node with two leaves. These stumps are poor students, and boosting methods favour this. In AdaBoost, the order of the stumps matters a lot. The initial stump's mistake affects how subsequent ones are created. An illustration will help you comprehend this.

- The output in this sample dataset, which only contains three characteristics, is categorical. The dataset is actually represented in the image. Because of the output's binary/categorical nature, it is now a classification issue. In reality, the dataset may contain any number of characteristics and records. For the purposes of explanation, let's look at 5 datasets. The result is categorical and is shown below as Yes or No. A sample weight will be given to each of these records. The formula utilised for this is 'W=1/N' where N is the number of records. Since there are only 5 entries in this dataset, the sample weight is initially set at 1. The weight of each record is the same. This time, it is 1/5.

- Discover the AdaBoost Model from Data
- Ada Boosting, which is based on binary classification issues, is best utilised to improve the performance of decision trees.
- The author first referred to AdaBoost as AdaBoost.M1. Discrete Ada Boost is a more modern name for it. Due to the fact that categorization rather than regression is the intended application.
- Any machine learning algorithm's performance may be improved with AdaBoost. It works best with reluctant students.

V. Results and Discussion

The following screenshots are depicted the flow and working process of project.

Data Loading: This is the page having dataset of credit card fraud detection using machine learning.

```
In [1]: import numpy as np
        import pandas as pd
        pd.set_option("display.max_columns", 50)

In [2]: df=pd.read_csv('creditcard.csv')

In [3]: df.shape
Out[3]: (284807, 31)

In [4]: df=df.sample(frac=0.3)

In [5]: df.head()
```

Out[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
65599	51660.0	1.341125	-0.623387	0.266612	-0.752746	-0.973363	-0.558557	-0.698878	0.106931	-0.846834	0.849140	1.481860	-0.926999	-2.135813
100689	67590.0	1.448854	-1.107643	0.389966	-1.286769	-1.641257	-1.026508	-0.904255	-0.223546	-1.754978	1.442340	-0.583566	-1.210821	-0.123435
260838	159746.0	-0.991059	1.457985	0.319896	-0.681177	0.858103	0.278567	0.630172	0.420629	-0.962712	-0.449144	-0.319297	0.737241	1.005241
84444	60307.0	0.804001	-0.769965	0.900562	0.741939	-0.757199	0.816300	-0.508076	0.295429	0.666500	-0.293944	0.908810	1.697421	0.813203
106696	70080.0	1.391282	-0.370237	0.082736	0.582143	-0.580665	-0.177171	-0.421567	-0.009174	-0.416592	0.788066	-2.054547	-1.121096	-1.540978

Information about Dataset: Here we can see the information about dataset.

```
In [8]: df.info()#3
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 85442 entries, 65599 to 58941
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Time    85442 non-null  float64
 1   V1      85442 non-null  float64
 2   V2      85442 non-null  float64
 3   V3      85442 non-null  float64
 4   V4      85442 non-null  float64
 5   V5      85442 non-null  float64
 6   V6      85442 non-null  float64
 7   V7      85442 non-null  float64
 8   V8      85442 non-null  float64
 9   V9      85442 non-null  float64
 10  V10     85442 non-null  float64
 11  V11     85442 non-null  float64
 12  V12     85442 non-null  float64
 13  V13     85442 non-null  float64
 14  V14     85442 non-null  float64
 15  V15     85442 non-null  float64
 16  V16     85442 non-null  float64
 17  V17     85442 non-null  float64
 18  V18     85442 non-null  float64
 19  V19     85442 non-null  float64
 20  V20     85442 non-null  float64
 21  V21     85442 non-null  float64
 22  V22     85442 non-null  float64
 23  V23     85442 non-null  float64
 24  V24     85442 non-null  float64
 25  V25     85442 non-null  float64
 26  V26     85442 non-null  float64
 27  V27     85442 non-null  float64
 28  V28     85442 non-null  float64
 29  Amount  85442 non-null  float64
 30  Class   85442 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 20.9 MB
```

Local Outlier Factor Model Building: This is the for model building with local outlier factor.

```
from sklearn.neighbors import LocalOutlierFactor
```

```
lof=LocalOutlierFactor(n_neighbors=20, contamination=anomalies_fraction)
y_pred=lof.fit_predict(X)
```

Classification Report: Here we can see the classification result for local outlier factor.

```
]: print(classification_report(y,y_pred, digits=6))
```

```
              precision    recall  f1-score   support

           0   0.998053  0.998053  0.998053     85273
           1   0.017751  0.017751  0.017751       169

    accuracy                       0.996114     85442
   macro avg   0.507902  0.507902  0.507902     85442
weighted avg   0.996114  0.996114  0.996114     85442
```

Isolation Forest Model Building: This is the for model building with Isolation forest.

```
']: from sklearn.ensemble import IsolationForest
```

```
i]: Iso_Forest=IsolationForest(contamination=anomalies_fraction, max_samples=len(X))
    Iso_Forest.fit(X)
```

Classification Report: Here we can see the classification result for isolation forest.

```
print(classification_report(y,y_pred2,digits=6))
```

```
              precision    recall  f1-score   support

           0   0.998745  0.998745  0.998745     85273
           1   0.366864  0.366864  0.366864       169

    accuracy                       0.997495     85442
   macro avg   0.682805  0.682805  0.682805     85442
weighted avg   0.997495  0.997495  0.997495     85442
```

K-Means Model Building: This is the for model building with K-Means clustering.

```
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
```

```
kmeans = KMeans(n_clusters=2, init='k-means++',n_jobs = -1)
```

```
model = kmeans.fit(X)
```

```
C:\Users\YMTS0356\AppData\Roaming\Python\Python36\site-packages\skle
recated in version 0.23 and will be removed in 1.0 (renaming of 0.25
  " removed in 1.0 (renaming of 0.25).", FutureWarning)
```

Classification Report: Here we can see the classification result for K-Means clustering.

```
: print(classification_report(y,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     85273
           1       0.01      0.08      0.01       169

    accuracy                           0.98     85442
   macro avg       0.50      0.53      0.50     85442
weighted avg       1.00      0.98      0.99     85442
```

Auto Encoders: This is the for model building with auto encoders.

```
from keras.models import Model, load_model
from keras.layers import Input, Dense
from keras.callbacks import ModelCheckpoint, TensorBoard
from keras import regularizers
```

```
# df = df.drop(['Time'],axis =1)
X_train, X_test = train_test_split(df, test_size=0.2, random_sta
X_train = X_train[X_train.Class == 0]
X_train = X_train.drop(['Class'], axis=1)
y_test = X_test['Class']
X_test = X_test.drop(['Class'], axis=1)
X_train = X_train.values
X_test = X_test.values
X_train.shape
```

```
(68212, 29)
```

```
input_dim = X_train.shape[1]
encoding_dim = 14
input_layer = Input(shape=(input_dim, ))
encoder = Dense(encoding_dim, activation="tanh",
                activity_regularizer=regularizers.l1(10e-5))(inp
encoder = Dense(int(encoding_dim / 2), activation="relu")(encode
decoder = Dense(int(encoding_dim / 2), activation='tanh')(encode
decoder = Dense(input_dim, activation='relu')(decoder)
autoencoder = Model(inputs=input_layer, outputs=decoder)
```

```
nb_epoch = 20
batch_size = 32
autoencoder.compile(optimizer='adam',
                    loss='mean_squared_error',
                    metrics=['accuracy'])
# checkpointer = ModelCheckpoint(filepath="model.h5",
#                                verbose=0,
#                                save_best_only=True)
# tensorboard = TensorBoard(log_dir='./logs',
#                           histogram_freq=0,
#                           write_graph=True,
#                           write_images=True)
history = autoencoder.fit(X_train, X_train,
                    epochs=nb_epoch,
                    batch_size=batch_size,

                    validation_data=(X_test, X_test)).history
```

Classification Report: Here we can see the classification result for auto encoder.

```
print(classification_report(error_df.true_class, y_pred3))
```

```
              precision    recall  f1-score   support

           0       1.00      0.92      0.96     17061
           1       0.02      0.79      0.03        28

    accuracy                           0.92     17089
   macro avg       0.51      0.85      0.49     17089
weighted avg       1.00      0.92      0.96     17089
```

Neural Networks: This is the for model building with neural networks.

```
from keras.models import Sequential
from keras.layers import Dense, Dropout
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# We are transforming data to numpy array to implementing with keras
X_train = np.array(X_train)
X_test = np.array(X_test)
y_train = np.array(y_train)
y_test = np.array(y_test)
```

```
X_train.shape
```

```
(59809, 29)
```

```
model = Sequential([
    Dense(units=20, input_dim = X_train.shape[1], activation='relu'),
    Dense(units=24,activation='relu'),
    Dropout(0.5),
    Dense(units=20,activation='relu'),
    Dense(units=24,activation='relu'),
    Dense(1, activation='sigmoid')
])
model.summary()
model.compile(loss='binary_crossentropy',optimizer='adam',metrics = ['accuracy'])
nb_epoch = 50
model.fit(X_train,y_train,epochs=nb_epoch,batch_size = batch_size)
```

Classification Report: Here we can see the classification result for neural networks.

```
print(classification_report(y_test,pred1.round()))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     25584
           1       0.90      0.78      0.84        49

    accuracy                           1.00     25633
   macro avg       0.95      0.89      0.92     25633
weighted avg       1.00      1.00      1.00     25633
```

VI. Conclusion

In this application, we have successfully created unsupervised ML models to detect whether the credit card is fraud or not fraud. We noticed that out of Local Outlier Factor, Isolation Forest, K-Means Clustering, Neural Networks and Auto Encoders Neural Networks performs well with accuracy score of 99% and precision and recall scores of 85%.

VII. REFERENCES

[1]. Taha, Altyeb & Malebary, Sharaf. (2020). An Intelligent Approach to Credit Card Fraud Detection Using an Optimized Light Gradient Boosting Machine. IEEE Access. 8. 25579-25587.

[2]. Assaghir, Zainab & Taher, Yehia & Haque, Rafiqul & Hacid, Mohand-Said & Zeineddine, Hassan. (2019). An Experimental Study With Imbalanced Classification Approaches for Credit Card Fraud Detection. IEEE Access.

[3]. L. Meneghetti, M. Terzi, S. Del Favero, G. A Susto, C. Cobelli, “DataDriven Anomaly Recognition for Unsupervised Model-Free Fault Detection in Artificial Pancreas”, Ieee Transactions On Control Systems Technology, (2018) pp. 1-15

[4]. F. Carcillo, Y.-A. Le Borgne and O. Caelen et al., “Combining unsupervised and supervised learning in credit card fraud detection”, Information Sciences, Elsevier (2019), pp. 1-15.

[5]. Ashphak, Mr. & Singh, Tejpal & Sinhal, Dr. Amit. (2012). A Survey of Fraud Detection System using Hidden Markov Model for Credit Card Application Prof. Amit Sinhal. 1.

[6]. Renjith, Shini. (2018). Detection of Fraudulent Sellers in Online Marketplaces using Support Vector Machine Approach. International Journal of Engineering Trends and Technology. 57. 48-53. 10.14445/22315381/IJETT-V57P210.

[7]. Saputra, Adi & Suharjito, Suharjito. (2019). Fraud Detection using Machine Learning in e-Commerce. 10.14569/IJACSA.2019.0100943.

[8]. A. K. Rai and R. K. Dwivedi, "Fraud Detection in Credit Card Data using Unsupervised Machine Learning Based Scheme," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2020, pp. 421-426, doi: 10.1109/ICESC48915.2020.9155615.

[9]. John O. Awoyemi, Adebayo O. Adetunmbi, Samuel A. Oluwadare et al., “Credit card fraud detection using Machine Learning Techniques: A Comparative Analysis”, IEEE, 2017.

[10]. Rajendra Kumar Dwivedi, Sonali Pandey, Rakesh Kumar “A study on Machine Learning Approaches for Outlier Detection in Wireless Sensor Network” IEEE International Conference Confluence, (2018).

Cite this article as :

ShadanG Chandra Kala, Devu Pavan Kumar, "The Detection of Credit Card Data Fraud by An Unsupervised Machine Learning Based Scheme the Detection of Credit Card Data Fraud by an Unsupervised Machine Learning Based Scheme", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 8 Issue 6, pp. 277-284, November-December 2022.
Journal URL : https://ijsrcseit.com/CSEIT228638