Machine Learning Approaches for Credit Card Fraud Detection
Final Project Report

Warren Liu

School of STEM, Computer Science & Software Engineering

University of Washington Bothell

CSS581: Machine Learning

Dr. Afra Mashhadi

5th March 2023

Abstract

The rise of credit card fraud has posed a significant threat to sellers in eCommerce. It occurs when someone uses another person's credit card or credit card information to make unauthorized purchases or withdraw money. Credit card fraud can happen in various ways, including skimming, phishing, and data breaches. Sellers suffer from the loss of both the product and the payment if the cardholder disputes the transaction. Therefore, early detection of fraudulent transactions is crucial to prevent financial loss. In this project, we use machine learning to develop models for fraud detection. The problem can be addressed through both outlier detection and classification approaches. We explore unsupervised learning methods such as clustering, LOF, and Autoencoder Neural Network for outlier detection, and supervised learning methods such as KNN, Decision Tree, and Balanced Random Forest for classification. Our experiments show that the classification models outperform the outlier detection models on the dataset we used.

1. Introduction

Credit card fraud is a significant issue that merchants and cardholders face, as scammers steal credit card information to make unauthorized transactions. When the cardholder identifies the fraudulent activity, they ask the bank to cancel the transaction and refund the money, leaving the merchant at a loss if they have already shipped the product. Therefore, detecting fraudulent transactions at the moment of purchase is critical for protecting merchants from financial losses. In this project, we address the problem of credit card fraud detection using machine learning techniques.

We approach fraud detection as both an outlier detection problem and a classification problem. In the outlier detection approach, we identify transactions that deviate significantly from the normal behavior of the system, indicating possible fraudulent activity. We use unsupervised learning methods such as Clustering models, LOF models, and Autoencoder Neural Network for this approach. In the classification approach, we train a model to predict whether a given transaction is fraudulent or not. We use supervised learning methods such as KNN models, decision tree models, and balanced random forest models for this approach.

Our study focuses on a specific dataset collected by Narayanan [1] and published on Kaggle that provides features related to transactions, and we use machine learning models to identify fraudulent transactions. The results of our experiments indicate that the classification models outperform the outlier detection models for the specific dataset we used. However, we believe that the Autoencoder Neural Network in the outlier detection approach can achieve better results with more features.

However, our proposed approach has several limitations. Firstly, the dataset used in the experiments only includes 7 features, which may limit the performance of unsupervised learning models. It is possible that we could notice an increase in the performance of unsupervised learning models by adding more important features. However, we are unsure if banks are willing to provide such information. Secondly, in the real world, banks may not provide detailed information regarding the transaction and cardholder, which could hinder the application of these models in the business. Moreover, the bank is not posting new datasets continuously, which limits the generalizability of the proposed solution. Therefore, close collaboration with banks is essential to collect and use data effectively for fraud detection.

2. Related Work

Fraud detection has been a widely researched topic in the field of machine learning, and two main approaches have been taken to address this problem: outlier detection and classification.

Due to the difficulty of labeling fraud data in real-time, unsupervised learning models such as UNISIM for Medicare claim fraud by Tang et al. [2] and Community Detection Algorithm for Bank Fraud by Sarma et al. [3] are used as outlier detection to detect anomalous behavior.

However, some types of fraud such as paycheck fraud can be labeled once the owner reports the fraud to the bank. For these types of fraud, supervised learning models, such as Random Forest, Logistic Regression, and AdaBoost used by Jain et al., [4], and the Markov Model built by Bhusari [5], might be a better approach. Nevertheless, some unsupervised learning models such as Hmm and Clustering [6] are also used for credit card fraud detection.

For the dataset we use, several classification models such as DNN, SVM, and Decision Tree built by Hjorth [7], have been applied in previous studies, and their performances are shown in Table 1.

Model	Accuracy	TP	FP	TN	FN
DNN	0.9890	~16,000	1,100	~180,000	83
SVM	0.998415	~17,000	190	~180,000	120
DT	0.999975	~17,000	5	~180,000	0
LR	0.95801	~10,000	~7,100	~180,000	~1,300

Table 1 Previous Studies on This Dataset

3. Data

The dataset [1] used in this project contains one binary target and seven feature: distance from home, distance from last transaction, ratio to median purchase price, repeat retailer, used chip, used pin number, and online order. The last four features are Boolean values, while the first three features are numerical values that range from 1.82 to 11,851.10. To account for this variance, we applied the `StandardScaler()` to scale the first three features.

To validate the dataset, we checked for `NaN` values and duplicates and found that the dataset is well-maintained and cleaned. The dataset contains a total of one million records, of which about 87 ,000 are fraudulent transactions and over 900,000 are normal transactions, as illustrated in Figure 1. Despite the dataset being unbalanced, our analysis has shown that 80% of the fraudulent transaction records are sufficient to train models that capture fraud patterns. In the unsupervised learning approach, we didn't balance the data since we removed the target. However, in the supervised learning approach, we used both unbalanced and balanced data to train models and compared their performance.

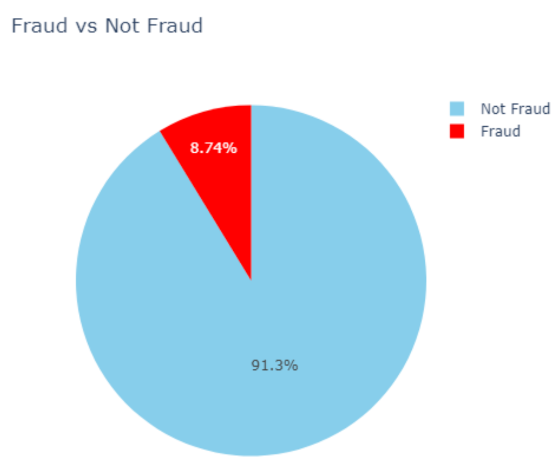


Figure 1 Data Distribution

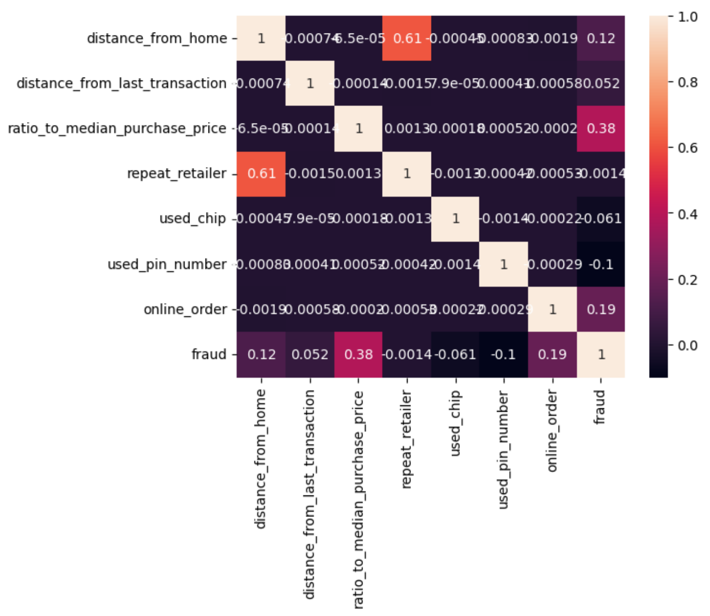


Figure 2 Features Pairwise Correlation

The pairwise correlation of features was explored and presented in the form of a plot. The results revealed that out of the seven features, three features namely distance from home, ratio to median purchase, and online order exhibited a high degree of correlation with the target variable. Conversely, distance from the last transaction showed a weak correlation with the target variable, while the remaining three features demonstrated a negative correlation. The findings are illustrated in Figure 2.

Principal Component Analysis (PCA) was conducted to examine the variability in the dataset for subsequent analysis. The results revealed that six components could account for 95% of the total variance, with the first three components explaining the majority of the variance, as illustrated in Table 2.

distance_from_home	27.48%
distance_from_last_transaction	27.42%
ratio_to_median_purchase_price	27.36%
repeat_retailer	6.24%
used_chip	6.23%
used_pin_number	2.80%
online_order	2.48%

Table 2 Data Variability

4. Experiments

4.1 Model Performance Evaluation

In detecting fraud, accuracy is crucial. However, when two models show similar accuracy, the confusion matrix was used to evaluate their performance. The goal was to detect as many fraudulent transactions as possible. If one model had a lower false positive rate than another, it was even possible to forgive some inaccuracy and a higher false negative rate. In practice, false negative decisions (detecting normal transactions as fraud) can still be rectified by confirming the transaction with the customer, while false positive decisions (detecting fraudulent transactions as normal) should be avoided at all costs.

4.2 Outlier Detection Approach

4.2.1 HDBSCAN

We utilized the HDBSCAN model as our initial approach for fraud detection. HDBSCAN is a density-based hierarchical clustering method that determines the final clustering results based on a multi-level implementation of clustering selection [8]. Following some parameter tuning, we obtained the best performance result as illustrated in Figure 3. The model exhibited a significant number of false positive decisions, which were approximately three times greater than the true positive decisions. As a result, the precision score was 0.26, the recall score was 0.41, and the f1-score was 0.32. These scores indicate poor model performance.

Subsequently, we visualized 10% of the clustering data and compared it with the input data. The results revealed that this model was almost incapable of identifying fraudulent transactions. See Figure 4.

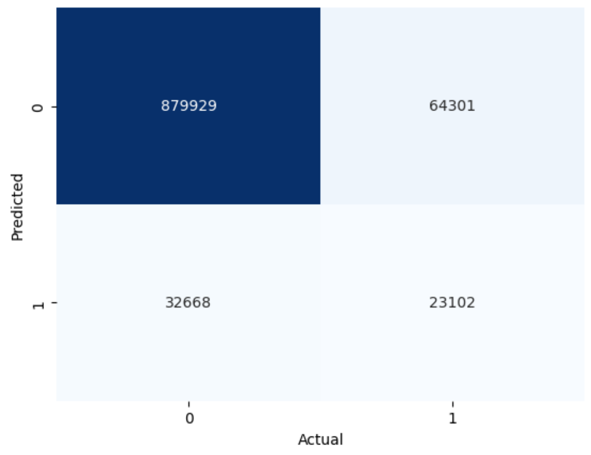


Figure 3 Confusion Matrix of HDBSCAN Model

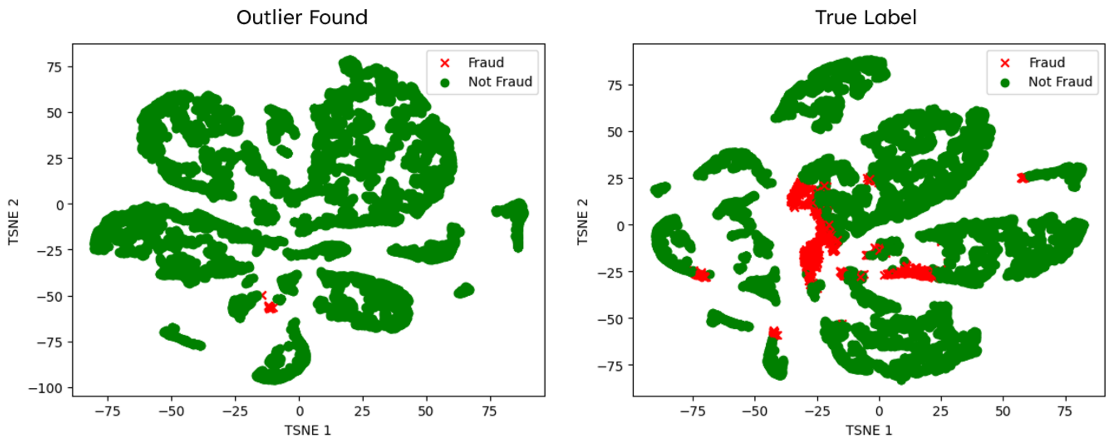
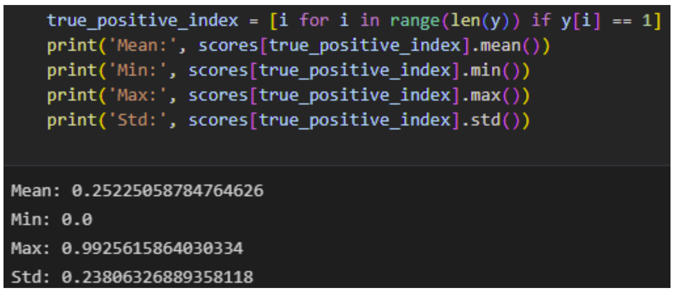


Figure 4 HDBSCAN Clustering Visualization

4.2.2 HDBSCAN + Threshold

We considered the possibility that the label generated by the HDBSCAN model may not be accurate enough. To address this, we manually set a threshold to determine the final label of the clustering. We extracted the scores of true fraudulent data generated by the model and calculated the mean and standard deviation of these scores. We then set the mean as the threshold value, which was determined to be 0.245. See Figure 5.

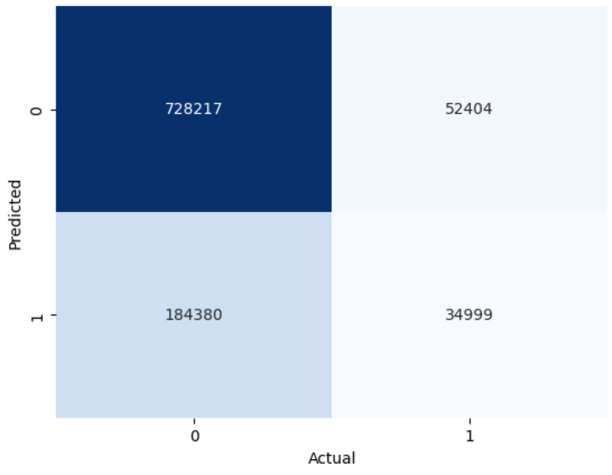


```
    true_positive_index = [i for i in range(len(y)) if y[i] == 1]
    print('Mean:', scores[true_positive_index].mean())
    print('Min:', scores[true_positive_index].min())
    print('Max:', scores[true_positive_index].max())
    print('Std:', scores[true_positive_index].std())

Mean: 0.25225058784764626
Min: 0.0
Max: 0.9925615864030334
Std: 0.23806326889358118
```

Figure 5 Find Threshold for HDBSCAN

After the threshold was set, it was observed that the false positive rate decreased by about 19%. However, this was accompanied by a significant increase in the false negative rate, which rose from 32,668 to 184,380 as indicated in Figure 6. Subsequently, the clusters were plotted again and displayed in Figure 7, revealing that the model was making an excessive number of false negative decisions.



Predicted \ Actual	0	1
0	728217	52404
1	184380	34999

Figure 6 Confusion Matrix of HDBSCAN + Threshold Model

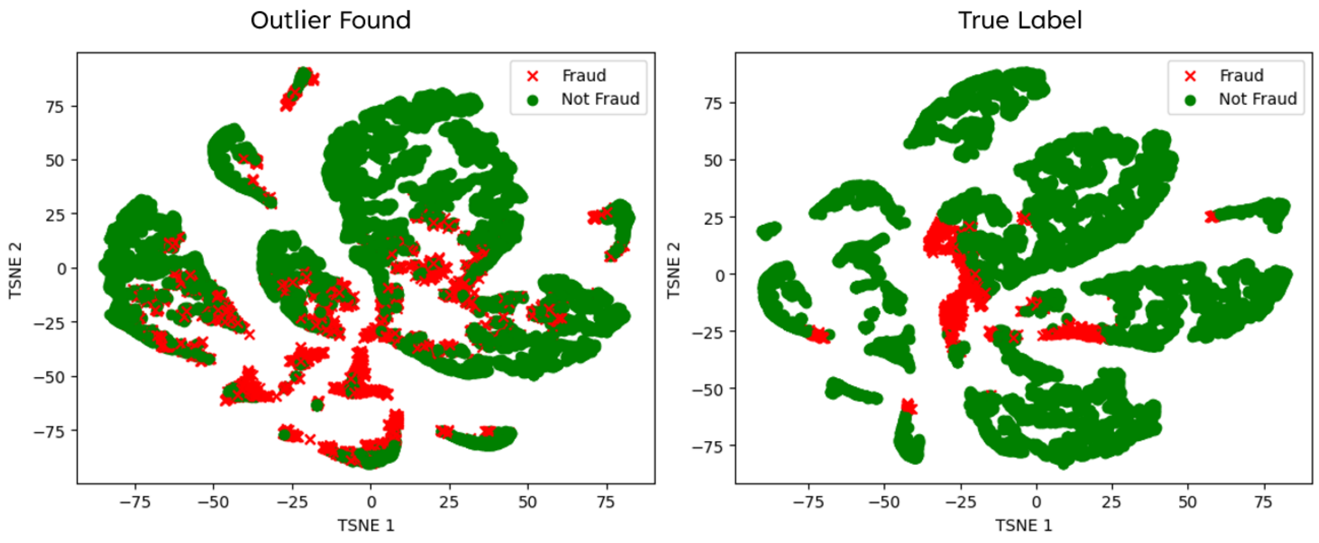


Figure 7 HDBSCAN + Threshold Clustering Visualization

4.2.3 HDBSCAN + PCA

As previously determined, six components were capable of explaining 95% of the variance, with the first three features accounting for the majority. In an effort to reduce data dimensionality prior to feeding into the HDBSCAN model, we employed PCA. Our best results were obtained by reducing the dimensionality to three. With this approach, the false positive rate decreased by about 70%. However, the cost was an increase in the false negative rate from 32,668 to 572,477, as shown in Figure 8.

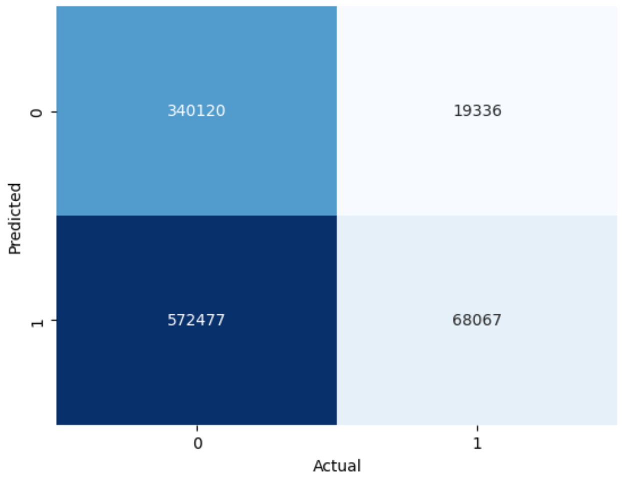


Figure 8 Confusion Matrix of HDBSCAN + PCA Model

4.2.4 Local Outlier Factor (LOF)

The Local Outlier Factor (LOF) algorithm as shown in Figure 9 is a type of unsupervised anomaly detection method that calculates the local density deviation of a given data point in relation to its neighbors. This method does not distinguish an outlier from the dataset directly, but using LOF to indicate outlier-ness degree which quantifies how outlying an object is [9]. We employed this model to detect anomalies in the dataset, but despite our attempts at tuning, none of the combinations yielded satisfactory results. The model was essentially making random guesses, with a nearly 50-50 split between correct and incorrect predictions, as depicted in Figure 10.

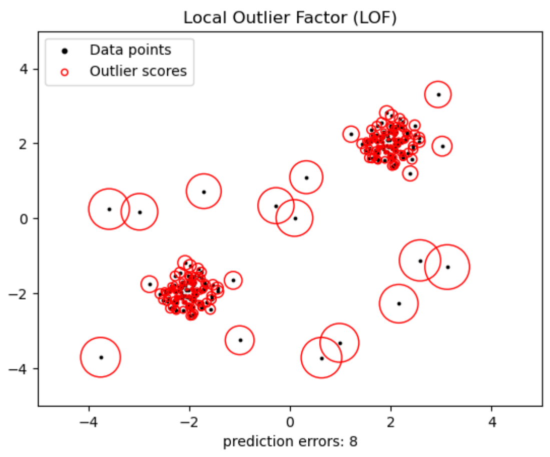


Figure 9 LOF [10]

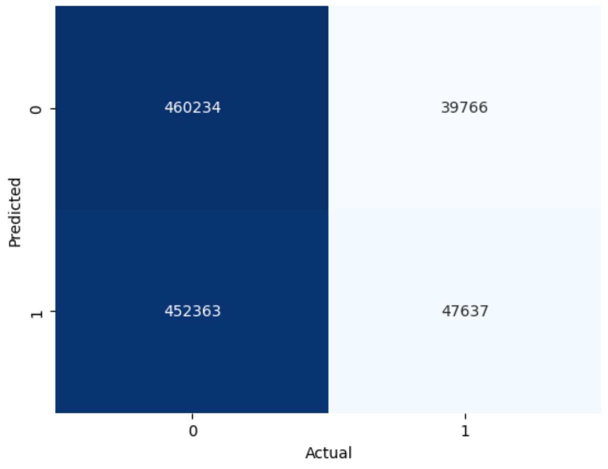


Figure 10 Confusion Matrix of LOF Model

4.2.5 Autoencoder Neural Network

An autoencoder as shown in Figure 11 is a type of artificial neural network used to learn efficient codings of unlabeled data (unsupervised learning). An autoencoder learns two functions: an encoding function that transforms the input data, and a decoding function that recreates the input data from the encoded representation [11] [12]. The primary concept underlying our approach for applying autoencoder as an outlier detector is to train the network solely on normal transactions, enabling it to learn the normal transaction patterns for encoding and decoding. Subsequently, during testing, we feed the network with both normal and fraudulent transactions. Theoretically, if the input is a normal transaction, the output and the input's reconstruction error should be minimal. On the other hand, if the input is a fraudulent transaction, the output and input error will be considerably higher, allowing us to identify the fraudulent transaction or outlier.

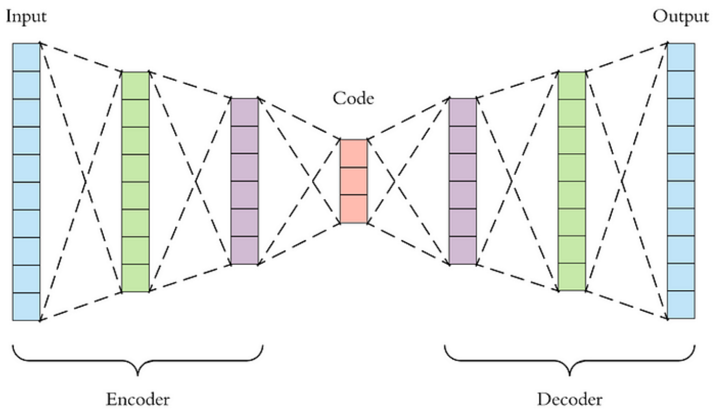
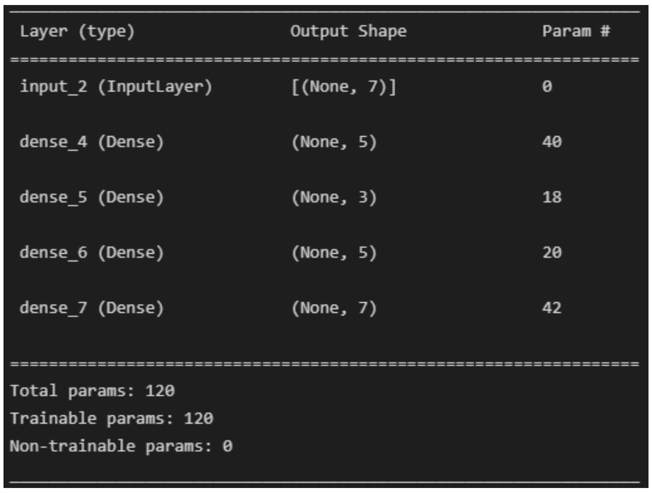


Figure 11 Autoencoder Neural Network [13]

After tuning the autoencoder model, we found that the simplest model, as shown in Figure 12, performed the best. This may be due to the low complexity of the dataset, which allowed for the use of a simpler model. The highest reconstruction accuracy achieved was 0.6741, indicating that the model was not able to reconstruct the input from the code very well. As a result, the performance of the model was not good. Compared to the HDBSCAN model, the false positive rate decreased by approximately 24%, but the false negative rate increased by about 5%. These results are depicted in Figure 13.



```
Layer (type)                Output Shape              Param #
=================================================================
input_2 (InputLayer)        [(None, 7)]               0

dense_4 (Dense)             (None, 5)                 40

dense_5 (Dense)             (None, 3)                 18

dense_6 (Dense)             (None, 5)                 20

dense_7 (Dense)             (None, 7)                 42

=================================================================
Total params: 120
Trainable params: 120
Non-trainable params: 0
```

Figure 12 Autoencoder Model

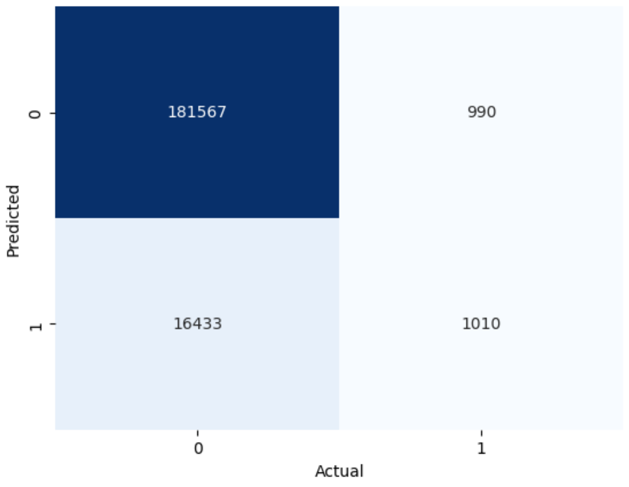


Figure 13 Confusion Matrix of Autoencoder Model

4.3 Reasons for Bad Performance

After evaluating the performance of various outlier detection models, we began to question the reason for their poor performance. We considered whether it was due to inadequate tuning of the models or the characteristics of the dataset itself.

Upon further analysis, we observed that the fraudulent transactions were not distinctly separated from normal transactions but formed clusters. As shown in Figure 14, these fraudulent transactions were often found at the edges of the clusters of normal transactions, indicating their similarity to normal transactions. We calculated the similarity and pairwise distance between normal and fraudulent transactions to other transactions and found that their mean similarity and pairwise distance were very close to each other. See table 3.

Based on these observations, we hypothesized that fraudulent transactions in this dataset behaved similarly to normal transactions and differed only in one or a few features. Hence, the difference between them was minimal, making it challenging for outlier algorithms to detect them. Outlier algorithms were not sensitive to small differences in a few features, which might have contributed to their poor performance.

Therefore, we decided to shift our focus toward classification models.

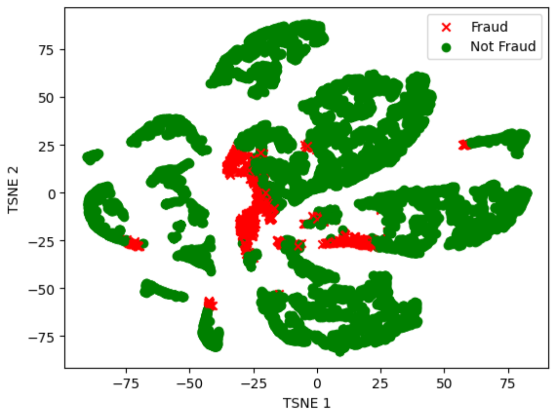


Figure 14 Clusters of Input Data

Mean of similarity of normal transactions to others	0.5502531
Mean of similarity of fraudulent transactions to others	0.5494259
Mean of pairwise distances of normal transactions to others	1.8151133
Mean of pairwise distances of fraudulent transactions to others	1.9935986

Table 3 Mean of Similarity and Pairwise Distance

4.4 Classification Approach

4.4.1 K-nearest Neighbors

Upon reviewing Figure 14, it was observed that while normal and fraudulent transactions were dispersed throughout the plot, they also formed small clusters. Thus, we hypothesized that the KNN algorithm may perform well in this scenario.

After tuning the model and selecting the best K value of 4, we fed the model with unbalanced data and found that it performed exceptionally well. Figure 15 illustrates that the false positive rate and false negative rate were below 1%, with an accuracy of 0.99874. We then trained and tested the model using balanced data through Random Under Sampling and Random Over Sampling techniques, and the performance improved even further, as illustrated in Figure 16 and Figure 17 respectively. The performance of all models is listed in Table 44.

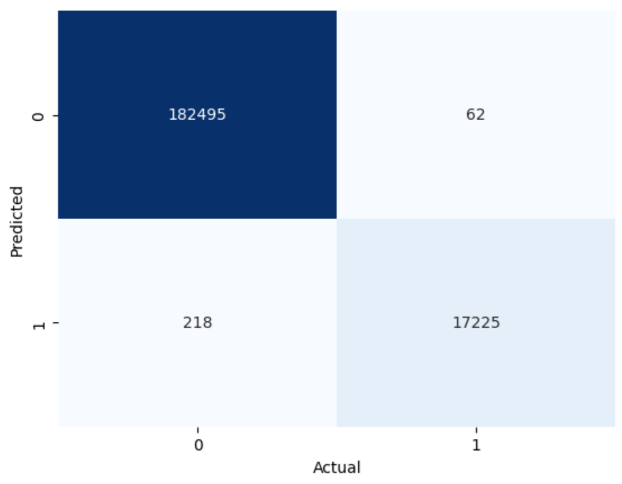


Figure 15 KNN using Unbalanced Data

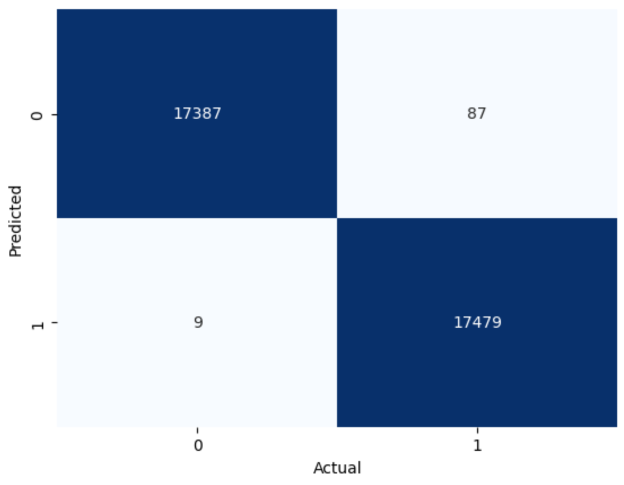


Figure 16 KNN using Random Under Sampling Data

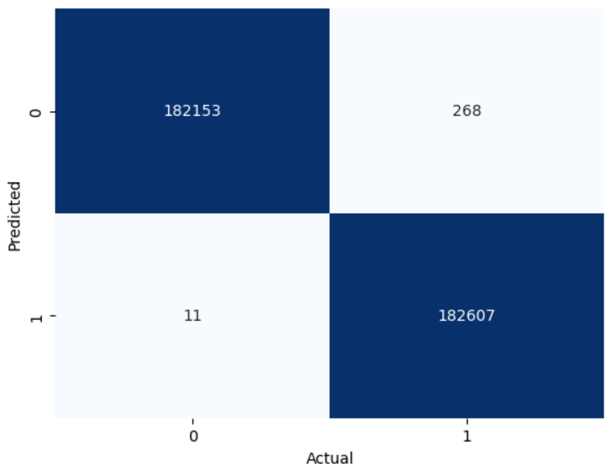


Figure 17 KNN using Over Sampling Data

Model	Accuracy	FPR	FNR
KNN (Unbalanced)	0.99874	0.6828%	0.0728%
KNN (Under Sampling)	0.99725	0.4953%	0.0517%
KNN (Over Sampling)	0.99923	0.1465%	0.0060%

Table 4 Performance of All KNN Models

4.4.2 Decision Tree

In our classification approach, we next built a Decision Tree model, which is known for its simplicity and fast training time. With the unbalanced data, as shown in Figure 18, the model achieved an accuracy close to 1 without any bias towards the majority class. This proved that training the model on 80% of the minority class was sufficient to capture the pattern of transactions despite the dataset's unbalanced nature.

After applying Random Under Sampling to the data, as shown in Figure 19, we found that the model's performance was perfect. However, we considered that the data removed during the under-sampling process may contain important information that the model could incorrectly predict. To test this hypothesis, we evaluated the model on all data, and the accuracy dropped to 0.99989, confirming our suspicion. Hence, we excluded this result from our conclusion.

Using Random Over Sampling, as shown in Figure 20, we found that the model was perfect, with a false positive rate of 0 and only 4 false negative predictions that we could ignore. The performance of all models is listed in Table 5.

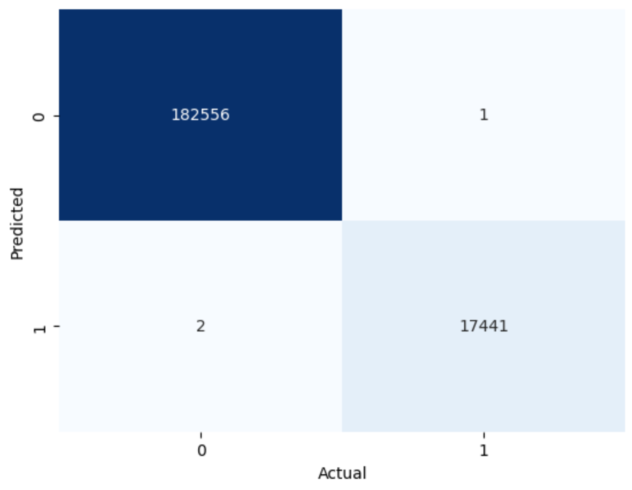


Figure 18 Decision Tree using Unbalanced Data

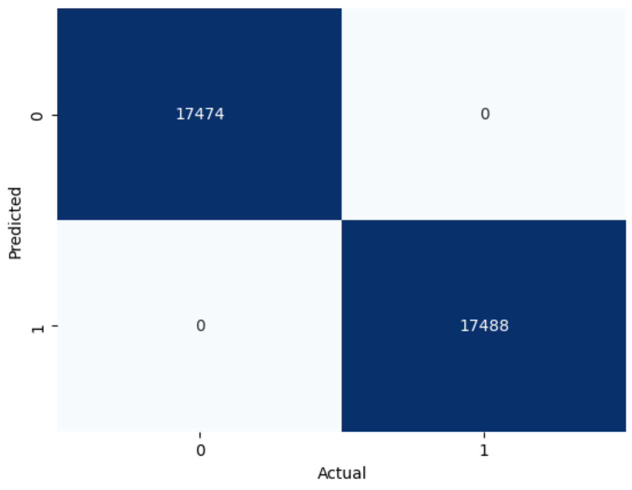


Figure 19 Decision Tree using Under Sampling Data

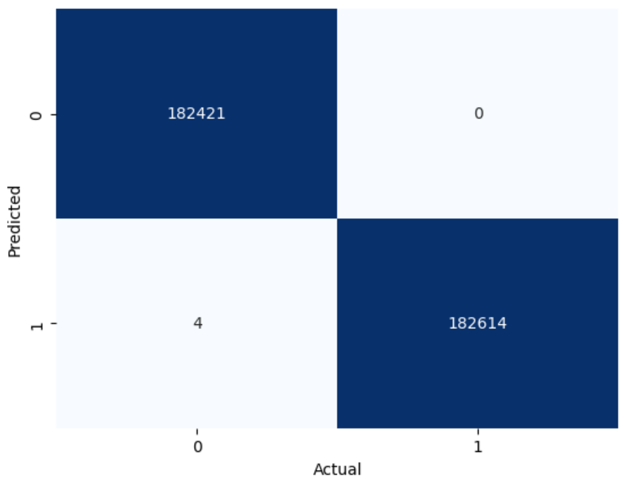


Figure 20 Decision Tree using Over Sampling Data

Model	Accuracy	FPR	FNR
DT (Unbalanced)	0.999985	0.00573%	0.00109%
DT (Over Sampling)	0.999989	0.00%	0.00219%

Table 5 Performance of All DT Models

4.4.3 Validation of Decision Tree Models for Overfitting

To assess whether the Decision Tree models were overfitting, we conducted several tests. Firstly, we plotted the decision tree, as shown in Figure 21, and examined its patterns. From this analysis, we concluded that the model had captured every pattern of transactions.

Secondly, we used Cross Validation to evaluate the model's performance. We set the number of folds to 5 and found that the accuracies were [0.997, 0.9995, 0.9998, 0.9998, 1.0], with a mean accuracy of 0.99985.

Lastly, we plotted the accuracy vs max depth graph, as shown in Figure 22, and observed that the test score did not decrease as the train set score increased. Based on these results, we concluded that the models did not overfit.

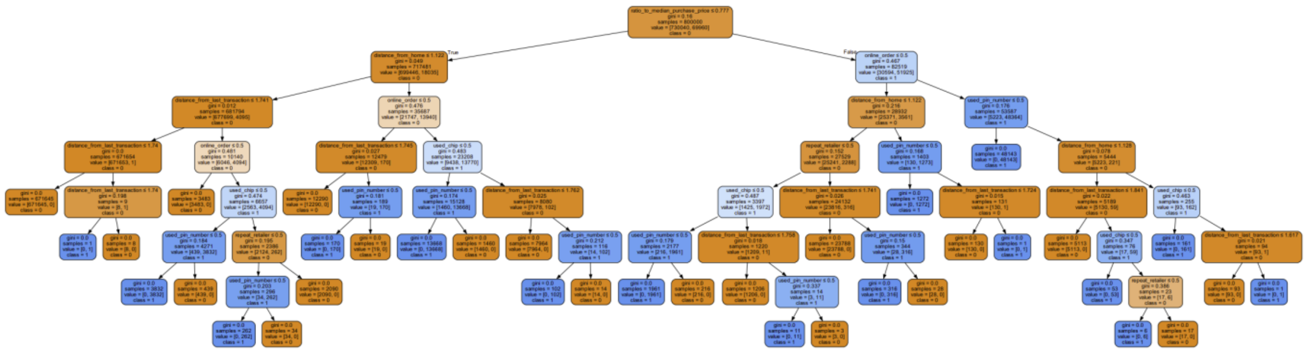


Figure 21 Decision Tree

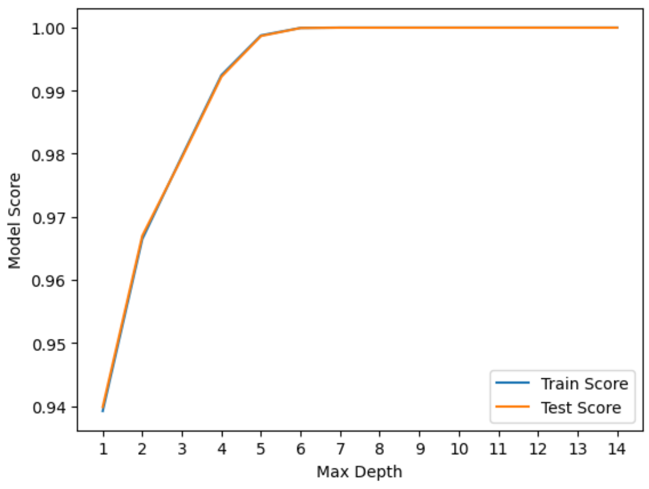


Figure 22 score vs. max_depth plot of Decision Tree

4.4.4 Balanced Random Forest

Balanced Random Forest was the last model we built, and it proved to be a strong contender for detecting fraudulent transactions. This model uses an ensemble of decision trees that are trained on bootstrapped subsets of the data, and each tree is trained on a randomly selected subset of features. During prediction, the model averages the predictions of all the trees to come up with the final classification. In our experiments, we did not have to balance the data for this model, which made it easier to use. As shown in Figure 23, although this model was not as good as the Decision Tree in terms of accuracy, it still performed extremely well and outperformed the outlier detection models we tested earlier.

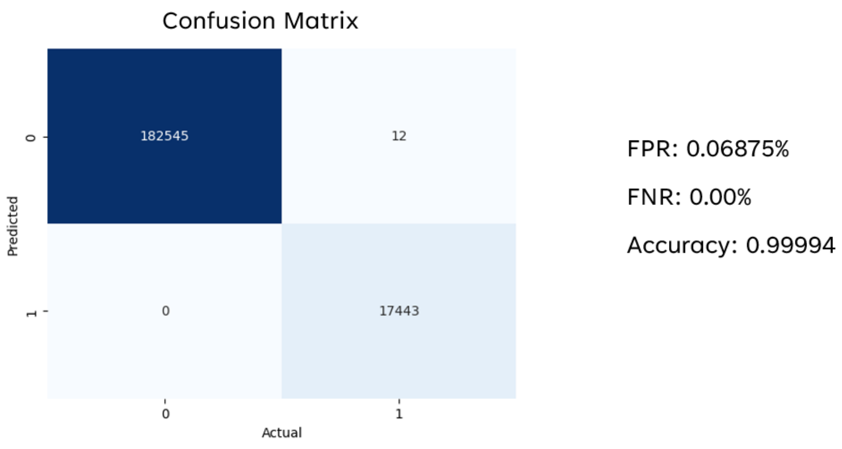


Figure 23 Confusion Matrix of using Balanced Random Forest Model

5. Conclusion

The results of our experiments, as presented in Table 6, demonstrate that the outlier detection models performed poorly while all classification models achieved nearly perfect performance. The Balanced Random Forest achieved a zero false positive rate, while the oversampling Decision Tree achieved the highest accuracy and a zero false positive rate. We believe that this dataset was well-maintained, well-cleaned, and well-labeled, with no noise in the dataset, which made it an excellent choice for classification models. Furthermore, although the data was relatively unbalanced, it provided enough data on the minority class. We also considered that the fraudulent and normal transactions behaved almost identically, with only a tiny difference between them, which made it difficult for outlier detection algorithms to work effectively.

Our project has shown that classification models outperformed outlier detection models for fraud detection on this particular dataset. However, there are limitations to our approach. Firstly, it is challenging to determine if the Decision Tree model is overfitting. Secondly, classification models such as Decision Tree may not be able to handle new kinds of fraudulent transactions as they may not have the splitting rules for unseen data, resulting in unpredictable decisions. Furthermore, the dataset used in our experiments only includes seven features, which may limit the performance of unsupervised learning models. Lastly, effective collaboration with banks is crucial for collecting and using data for fraud detection, as these data are confidential and banks may not provide them to us in real-time, which could limit the performance of our models.

Model	**Accuracy**	**FPR**	**FNR**
	Unsupervised		
HDBSCAN	0.903031	73.75%	3.58%
HDBSCAN (Threshold)	0.763216	59.96%	21.37%
HDBSCAN (PCA)	0.408187	22.12%	64.86%
LOF	0.507871	45.50%	49.35%
Autoencoder	0.917885	49.5%	8.30%
	Supervised		
KNN (Unbalanced)	0.99874	0.68%	0.07%
KNN (Oversampling)	0.99923	015%	0.006%
DT (Unbalanced)	0.999985	0.0057%	0.0011%
DT (Oversampling)	**0.999989**	**0.00%**	**0.0022%**
BRF	0.994	0.069%	0.00%

Table 6 All Models' Performance

6. Future Work

In our project, we have found that Autoencoder is a promising approach for fraud detection due to its ability to handle new types of fraudulent transactions. The algorithm of Autoencoder is capable of detecting unusual behaviors even when the fraud is a new kind of fraud, as long as the fraudulent behavior is different from normal transactions. The Autoencoder can achieve this by identifying the outliers through the reconstruction error. However, we have observed that in our experiment, the performance of the Autoencoder was limited by the complexity of the dataset.

We believe that if the features of the dataset were more complex, such as if the IP address used for the transaction matched the shipping address, if the transaction was made using a web proxy, or if the cardholder's devices such as phones or tablets were nearby the place where the transaction was made, the Autoencoder could achieve similar or even better performance than the classification models in practice. However, due to the simplicity of the dataset used in our experiments, we could not draw a definitive conclusion on the effectiveness of Autoencoder for fraud detection.

Overall, while Autoencoder shows promise for detecting new kinds of fraud, further research is needed to determine its effectiveness on more complex datasets and in real-world scenarios.

References

[1] "Credit Card Fraud," Kaggle, May 2022. [Online]. Available: https://www.kaggle.com/datasets/dhanushnarayananr/credit-card-fraud. [Accessed 1 Feburary 2023].

[2] M. Tang, B. S. U. Mendis, D. W. Murray, Y. Hu and A. Sutinen, "Unsupervised fraud detection in Medicare Australia," in *CONF-CDS 2021: The 2nd International Conference on Computing and Data Science*, 2011.

[3] D. Sarma, W. Alam, I. Saha, M. N. Alam, M. J. Alam and S. Hossain, "Bank Fraud Detection using Community Detection Algorithm," in *IEEE*, Coimbatore, India, 2020.

[4] V. Jain, H. Kavitha and S. M. Kumar, "Credit Card Fraud Detection Web Application using Streamlit and Machine Learning," in *IEEE*, Hassan, India, 2022.

[5] V. Bhusari and S. Patil, "Study of Hidden Markov Model in credit card fraudulent detection," in *IEEE*, Coimbatore, India, 2016.

[6] K. Janvitha, C. R. S. Vasavi, A. Sruthi, K. Praharshitha and D. K. Anguraj, "Survey on Detection of Credit Card Frauds using Hmm and various Clustering Approaches," in *IEEE*, Coimbatore, India, 2021.

[7] Z. HJORTH, "DNN, SVM, and DT for Fraud Detection," Kaggle, August 2022. [Online]. Available: https://www.kaggle.com/code/zwhjorth/dnn-svm-and-dt-for-fraud-detection. [Accessed March 2023].

[8] J. Liu, S. Sun and C. Chen, "Big data Analysis of Regional Meteorological Observation Based : on Hierarchical Density Clustering Algorithm HDBSCAN," in *IEEE*, Shanghai, China, 2021.

[9] O. Alghushairy, R. Alsini and X. Ma, "An Efficient Local Outlier Factor for Data Stream Processing: A Case Study," in *IEEE*, Las Vegas, NV, USA, 2020.

[10] "Outlier detection with Local Outlier Factor (LOF)," scikit-learn, [Online]. Available: https://scikit-learn.org/stable/auto_examples/neighbors/plot_lof_outlier_detection.html. [Accessed March 2023].

[11] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal,* vol. 37, pp. 233-243, 1991.

[12] M. A. Kramer, "Autoassociative neural networks," *Computers & Chemical Engineering,* vol. 16, pp. 313-328, 1992.

[13] A. Dertat, "Applied Deep Learning - Part 3: Autoencoders," Towards Data Science, 3 October 2017. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798. [Accessed March 2023].