Credit Card Fraud Detection Using Machine Learning Algorithms

Jai Gupta

Grade 11, Step by Step School, Noida

Abstract: ***This study explores the application of machine learning algorithms for the detection of credit card fraud. Utilizing a comprehensive dataset of 284,807 credit card transactions, we implemented anomaly detection methods to identify fraudulent activities. The methods include autoencoders, which have shown effectiveness in handling imbalanced datasets. Our results demonstrate a high accuracy of the model, suggesting its potential in real-time fraud detection and prevention in financial systems. This study contributes to the evolving field of cybersecurity by providing a robust framework for credit card fraud detection. The purpose of this study is to develop and evaluate a machine learning-based model for the detection of credit card fraud, addressing the challenge of data imbalance and focusing on anomaly detection methods. The significance of this research lies in its potential to enhance the security of financial transactions and protect consumers from fraud. By leveraging advanced machine learning techniques, this study contributes to the development of more effective fraud detection systems.***

Keywords: Credit Card Fraud, Machine Learning, Anomaly Detection, Autoencoders, Cybersecurity

1. Introduction

In the year 2021, nearly 390,000 reports of credit fraud were filed to the Federal Trade Commission, making it by far one of the most common forms of fraud. Since more than half of the payments made today are online, this is highly alarming. Credit card fraud can cause people to lose millions of rupees and detecting it can be challenging. Many attempts have previously been made to try and put a stop to this fraud, but this practice remains widespread. In this research, we will use an anomaly detection method to detect fraud using a database of credit card transactions.

2. Literature Review

1) https://www.analyticsvidhya.com/blog/2023/05/anomaly-detection-in-credit-card-fraud/ [2]

a) Talks about how Google runs its anomaly detection, taking the example of logging in. Google checks when you log into your account, what your normal activity is when you log in, and other general information that it collects from cookies and your digital footprint. Using this large database it is able to detect any activity that may seem out of place or unusual.

b) Uses feature engineering (essentially picking up the most important data points from pre-existing data and then using that to make a prediction about the outcome, like when deciding how much to sell a car for we use the market value as a starting point), data preprocessing (cleaning, scanning and combing through data to make it suitable to use, SMOTE analysis and removal of outliers are good examples),

c) They start by checking if the data has any missing values which is a very good way of making sure that the data is not compromised.

d) They then tried to find outliers using 3 different methods, isolation forests (creates random decision trees), one class SVMs (plots everything on one plane, and the data points that are outlying are detected and it also makes use of Local Outlier Factor which essentially measures the density of all the data identifies the data that is more or less dense than the others. They gave us 3 different values. I believe that they should have made use of one of the outlier detecting techniques and then treated the outliers. I say this because by not treating the outliers they compromise the integrity of their model as their model may be working with skewed data which would be an issue as it would make it more difficult to generalise their work.

e) It makes use of stratified K fold cross-validation techniques to divide it's data (NOT SURE) while this is far better than simply random sampling I still believe that if the author believed that their data was that imbalance then they could have made use of methods such as SMOTE analysis to get better results on their models. They then used a decision tree and logistic regression with cross-validation and deployed only the random forest model. I believe that this could have been improved as if they used boosting techniques and tuned versions of the same models they may have gotten better yields.

2) https://towardsdatascience.com/detection-of-credit-card-fraud-with-an-autoencoder-9275854efd48 [3]

a) They start by checking for missing values which is good

b) The data is unbalanced but they do not correct it as they do not need to as they are using semi-supervised techniques

c) They then remove the fraudulent transactions as that is what they will train their autoencoder on and remove unnecessary variables. All of this is great in my opinion as they cover every base and use new methods that would give them better results. They could have done a little bit more correlation analysis and used Phi-K correlation techniques to make more pre-test predictions and perhaps remove more variables that are unnecessary or unreasonable to keep

d) In autoencoding they take the data, with the normal state and deconstruct that data to reconstruct it in a manner such that it closely resembles the normal state given before. This allows it to locate and understand what data points are the most relevant.
e) They then train and evaluate the model. Upon evaluation, they then also deploy other methods to try and get a better yield
f) All in all, I believe that this was a very well-rounded way of going about the entire process. However, a few points as mentioned above could have been worked upon. They could have also made use of unsupervised learning methods to perhaps have better yields.

3) https://medium.com/analytics-vidhya/credit-card-fraud-detection-c66d1399c0b7 [4]
a) Uses a very good dataset
b) Use very good methods to visualize the data (Catplots, histograms, etc.)
c) The data is very imbalance but nothing is done to treat that
d) No correlation analysis is done before running the models
e) They use isolation forest, One classSVM, DBSCAN, and local outlier factor, however, due to the nature of isolation forest I believe that using it with such imbalanced data can be unwise.
f) I believe that DBSCAN is a good model to use when we have imbalanced data that has not been treated as it deals with outliers already.

4) https://www.researchgate.net/profile/Mohammad-Mahdi-Rezapour-Mashhadi/publication/337788635_Anomaly_Detection_using_Unsupervised_Methods_Credit_Card_Fraud_Case_Study/links/5dede1c64585159aa46e8949/Anomaly-Detection-using-Unsupervised-Methods-Credit-Card-Fraud-Case-Study.pdf [5]
a) Made use of one class SVM but also discussed how it could have a high false positive rate. This paper found that when using one-class SVM, various Kernel types could be used out of which linear models worked best. The data, like in many other studies was divided into fraudulent and normal transactions as the data has to be trained on normal data.
b) Used autoencoders but tweaked it to have functions with 3 layers, input, hidden and output layer to optimise results. Similar to one class SVM, it was trained on the normal data and then tested on the fraudulent one.

5) https://www.hindawi.com/journals/misy/2022/8027903/ [6]
a) Uses SMOTE technique to fix oversampling in data
b) iForest is one of the techniques it used to do the anomaly detection as they found that it handled bigger datasets better. However, the method seemed a lot more volatile in comparison to other methods as it decreased the data distribution complexity and hence may give results that are worse. This paper tried to fix the same using the KCPA method.
c) Makes use of a traditional One class SVM method with Ada Boost
d) They also used a boosting method called AdaBoost. They used a One Class SVM as a base classifier to carry out their boosting.
e) It finds the OCSVM with Adaboost to give the best results.

6) https://arxiv.org/pdf/1904.10604.pdf [1]
a) The study used various methods to compare supervised and unsupervised learning in credit card fraud detection

AI models to detect credit card fraud can be broadly categorised into 2 categories, supervised and unsupervised learning. Supervised learning takes place when the data is already labelled as fraudulent or not. A study conducted compared various supervised and unsupervised learning models to detect credit card fraud [1] and found that supervised learning models perform slightly better at the expense of spending more time removing outliers and balancing the data, they used Area Under the Curve to measure the efficiency of each model for the same.

Another model made use of the Local Outlier Factor [2], which is another method that measures the density of all of the data identifiers of the data and sees which data is more or less dense than others. This method was used to detect outliers in the above study and was able to do so in such a manner that it did not compromise the data Along with this they used K-fold stratification to divide their data for training which helps us run models on data that could be imbalanced while still getting consistent results. These two methods can come together to be a good way of managing our data.

Autoencoders and their use in most AI models regarding the same have shown very consistent results as it is very proficient at dealing with imbalanced data too. This method has also shown great yield as it understands what data points to focus more on. Using simple autoencoding models [3] can prove to be a good way of detecting anomalies in credit card transactions.

Another paper used Artificial Neural Networks to try and do anomaly detection[4]. It used a neural network divided into 3 layers, the first being the input layer in which the transaction of each customer is inserted. There is then a hidden layer which consists of weights and bias and finally, an output layer indicating fraud. In the end, it had an accuracy of 99% which shows that such an unsupervised method may also be a great option.

Another interesting model that could have been tried was an iForest. However, a study found that while it was good at handling big datasets, it was very volatile. [6]

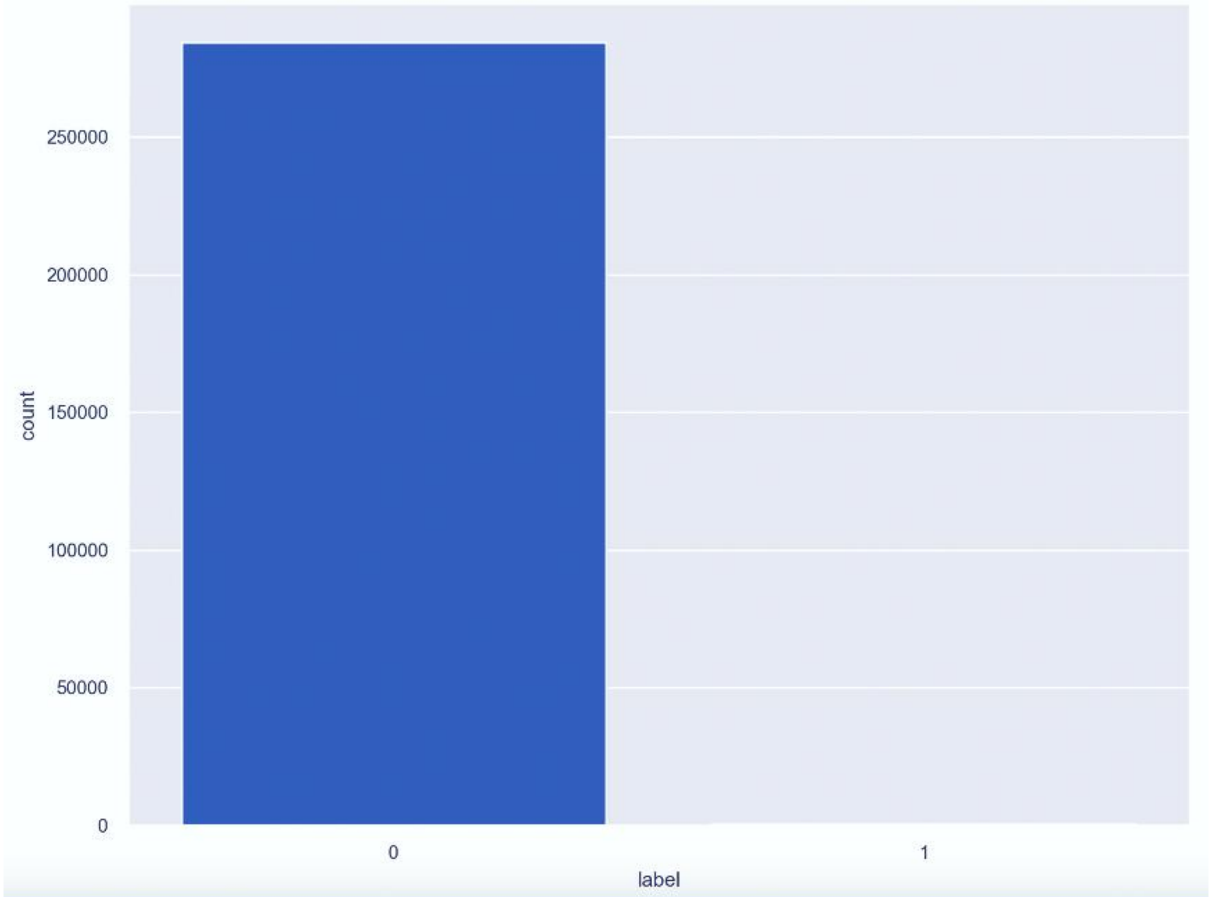
The One-Class SVM is a method that was found in most papers. It is very diverse, one study made use of a One Class

SVM along with Adaboosting which yielded very positive results [6]. Another method used various different versions of a traditional One-Class SVM model to see how results would vary [5] again yielding outstanding results. While a lot of studies are able to produce great results with the traditional version of the model.

3. Methods

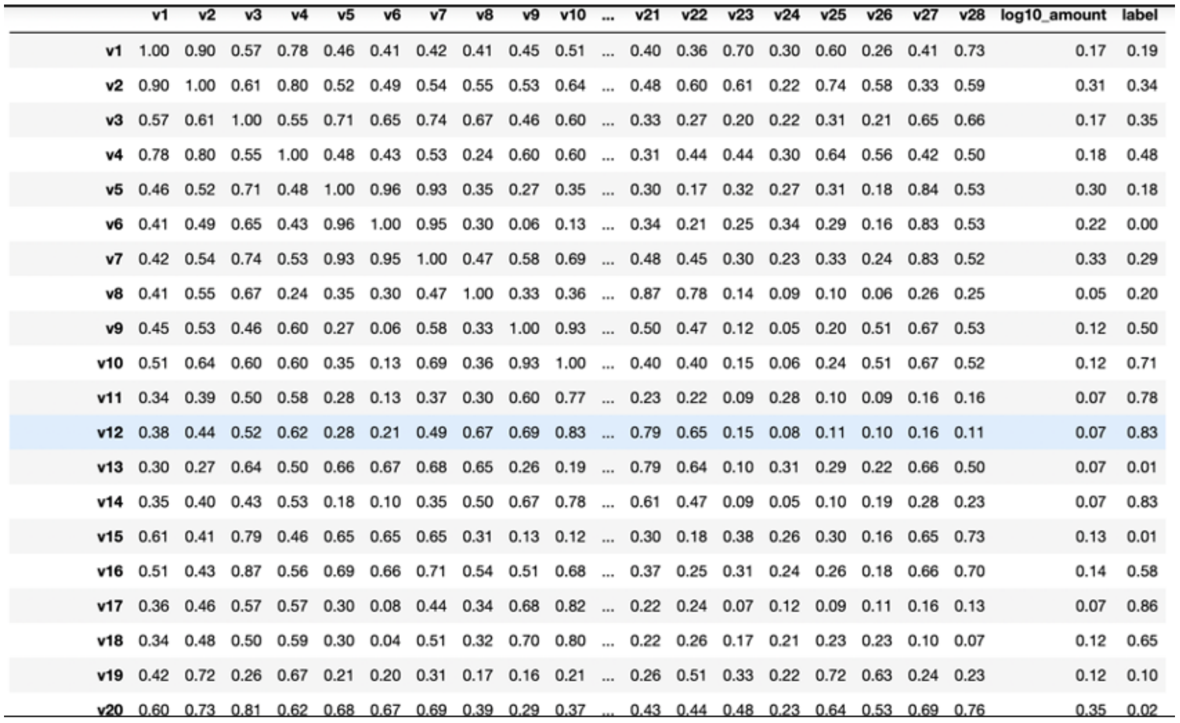
Dataset

In this study, a comprehensive dataset containing 284,807 instances of credit card transactions was used. Having these many instances enables the model to have abundant data for testing and training; hence making it a lot more accurate and precise. In order to better understand our dataset, we started by visually representing it using scatter plots. We use 'label" as our target variable which can give 2 outcomes, either fraudulent or clean transactions. The dataset that we used showed a substantial imbalance having merely 492 instances of fraudulent transactions amidst a majority of clean transactions; clearly, we cannot continue to train with this model. Therefore we took a more strategic approach by meticulously undersampling the clean transactions in a ratio of 15:1 (to fraud).



To show the disparity in between fraudulent and clean transactions in our data

After balancing our data we split it into testing and training data creating a robust environment for our model to learn. Once that is completed our data is ready; it is run through the model. The implications of our outcome are profound, using this model we can accurately predict the authenticity of any credit card transaction. This predictive capability of our model allows us to identify and block fraudulent transactions; hence protecting sensitive data and the financial resources of millions of people across the world.

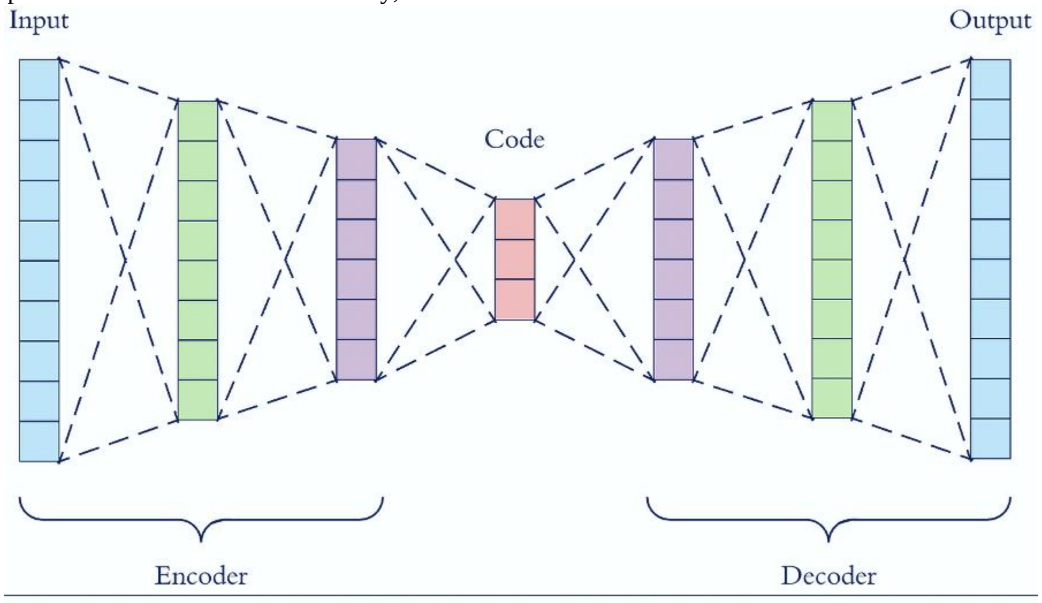


	v1	v2	v3	v4	v5	v6	v7	v8	v9	v10	...	v21	v22	v23	v24	v25	v26	v27	v28	log10_amount	label
v1	1.00	0.90	0.57	0.78	0.46	0.41	0.42	0.41	0.45	0.51	...	0.40	0.36	0.70	0.30	0.60	0.26	0.41	0.73	0.17	0.19
v2	0.90	1.00	0.61	0.80	0.52	0.49	0.54	0.55	0.53	0.64	...	0.48	0.60	0.61	0.22	0.74	0.58	0.33	0.59	0.31	0.34
v3	0.57	0.61	1.00	0.55	0.71	0.65	0.74	0.67	0.46	0.60	...	0.33	0.27	0.20	0.22	0.31	0.21	0.65	0.66	0.17	0.35
v4	0.78	0.80	0.55	1.00	0.48	0.43	0.53	0.24	0.60	0.60	...	0.31	0.44	0.44	0.30	0.64	0.56	0.42	0.50	0.18	0.48
v5	0.46	0.52	0.71	0.48	1.00	0.96	0.93	0.35	0.27	0.35	...	0.30	0.17	0.32	0.27	0.31	0.18	0.84	0.53	0.30	0.18
v6	0.41	0.49	0.65	0.43	0.96	1.00	0.95	0.30	0.06	0.13	...	0.34	0.21	0.25	0.34	0.29	0.16	0.83	0.53	0.22	0.00
v7	0.42	0.54	0.74	0.53	0.93	0.95	1.00	0.47	0.58	0.69	...	0.48	0.45	0.30	0.23	0.33	0.24	0.83	0.52	0.33	0.29
v8	0.41	0.55	0.67	0.24	0.35	0.30	0.47	1.00	0.33	0.36	...	0.87	0.78	0.14	0.09	0.10	0.06	0.26	0.25	0.05	0.20
v9	0.45	0.53	0.46	0.60	0.27	0.06	0.58	0.33	1.00	0.93	...	0.50	0.47	0.12	0.05	0.20	0.51	0.67	0.53	0.12	0.50
v10	0.51	0.64	0.60	0.60	0.35	0.13	0.69	0.36	0.93	1.00	...	0.40	0.40	0.15	0.06	0.24	0.51	0.67	0.52	0.12	0.71
v11	0.34	0.39	0.50	0.58	0.28	0.13	0.37	0.30	0.60	0.77	...	0.23	0.22	0.09	0.28	0.10	0.09	0.16	0.16	0.07	0.78
v12	0.38	0.44	0.52	0.62	0.28	0.21	0.49	0.67	0.69	0.83	...	0.79	0.65	0.15	0.08	0.11	0.10	0.16	0.11	0.07	0.83
v13	0.30	0.27	0.64	0.50	0.66	0.67	0.68	0.65	0.26	0.19	...	0.79	0.64	0.10	0.31	0.29	0.22	0.66	0.50	0.07	0.01
v14	0.35	0.40	0.43	0.53	0.18	0.10	0.35	0.50	0.67	0.78	...	0.61	0.47	0.09	0.05	0.10	0.19	0.28	0.23	0.07	0.83
v15	0.61	0.41	0.79	0.46	0.65	0.65	0.65	0.31	0.13	0.12	...	0.30	0.18	0.38	0.26	0.30	0.16	0.65	0.73	0.13	0.01
v16	0.51	0.43	0.87	0.56	0.69	0.66	0.71	0.54	0.51	0.68	...	0.37	0.25	0.31	0.24	0.26	0.18	0.66	0.70	0.14	0.58
v17	0.36	0.46	0.57	0.57	0.30	0.08	0.44	0.34	0.68	0.82	...	0.22	0.24	0.07	0.12	0.09	0.11	0.16	0.13	0.07	0.86
v18	0.34	0.48	0.50	0.59	0.30	0.04	0.51	0.32	0.70	0.80	...	0.22	0.26	0.17	0.21	0.23	0.23	0.10	0.07	0.12	0.65
v19	0.42	0.72	0.26	0.67	0.21	0.20	0.31	0.17	0.16	0.21	...	0.26	0.51	0.33	0.22	0.72	0.63	0.24	0.23	0.12	0.10
v20	0.60	0.73	0.81	0.62	0.68	0.67	0.69	0.39	0.29	0.37	...	0.43	0.44	0.48	0.23	0.64	0.53	0.69	0.76	0.35	0.02

Correlation of different variables

Experiment:

After preprocessing our dataset and dividing it into test and train, the next step is to train our model. In this study, we used autoencoders as after analyzing various other studies (more detail in the literary review section) we found that autoencoders were the most efficient for imbalanced datasets.



How do autoencoders work

Autoencoders work by encoding the data and then reconstructing the original data set. The encoder is used to compress the data into a latent space representation, a latent space representation is essentially the place where the model will mark input data. While we train the model, the encoders map the data onto a lower dimensional space (essentially a space with a lesser amount of variables) capturing the important data points in our dataset. The other part is the decoder, which uses the compressed data and tries to reconstruct the original data from it, through training it will learn how to map this compressed data back to the original form. Through training we minimise the difference between the input data and the reconstructed data, hence creating a more accurate model.

We used 'tensorboard' to visualise our neural network and see the architecture of our encoders and decoders to gain a better understanding of how our model functions. We also used tensorboard to monitor our training metrics and make sure that our model is learning effectively, allowing us to also track how well our model does in reconstructing the data. Another thing that we use tensorboard for in this case is seeing how our model is mapping the high-level data into a latent space representation which gives us an insight into how well our model is plotting the data.

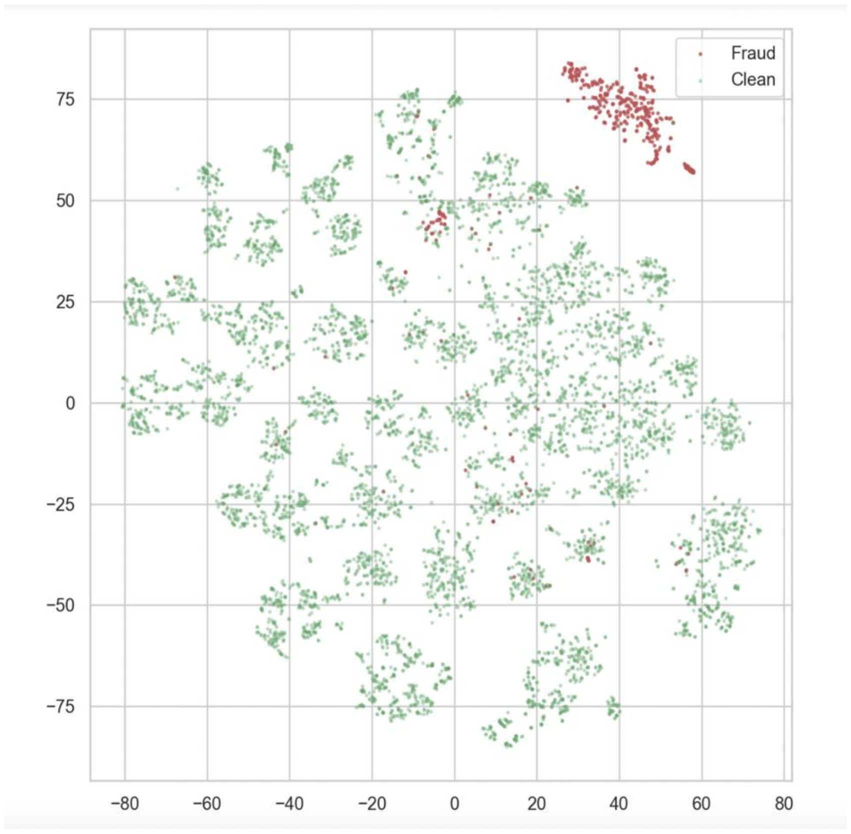
In short, we start by importing a lot of modules which allow us to better visualise and analyse our data (such as pandas, tensorflow, numpy, matplotlib and seaborn) We use pandas for its data structures and functions to manipulate and analyse our data. We use tensorflow for its high-level APIs (like Keras), to carry out high-level mathematical calculations, to help create the architecture for our autoencoder's neural network and for its powerful visualisation tools such as tensorboard. We use numpy because it helps us carry out high-level mathematical calculations on tables and arrays which is helpful here as our data is mostly in a tabulated format. We use matplotlib and seaborn for their powerful visualisation capabilities to help visualise and understand our data better as it is very large and vast. We then manipulate our data to make it easier to work with and define our manual variables and use t-SNE to help visualise the clean and fraudulent data. After that we split the model into a train and test set to get it ready for fraud detection and training of the model, we use various functions to ensure that our model will be able to reproduce results every time it is run; this ensures the fact that our model will have sufficient training and testing data and also ensures that our model will be reliable to produce the same results consistently. We then use a pipeline to transform our data, we normalise the features of our data using the 'Normalizer()' and then perform min-max scaling on these features using 'MinMixScaler()'; this ensures that the same transformers are applied both to the training and test data; so our pipeline has 2 steps in it, a normaliser and a scaler; hence preventing data leakage. We then set batch sizes and epochs to regulate the speed of the processing which also allows us to set how many times our model will through the data to adjust the model's weights and minimise training error. We then set up our autoencoder model using the tensorflow (tf.keras) library, reducing input features from 16 down to just 2 in the encoder and bringing it back up to 16 in the decoder section; this is all done using ELU activation which helps minimise reconstruction error. We then use the Mean Squared Error (MSE) loss function and the Adam Optimiser. The MSE loss function helps us by quantifying prediction errors, which essentially means that it will compute the difference between target values and actual values; the larger the difference, the larger the error; it also condenses all of the prediction errors into a single value; here it also guides our model to minimise our loss through the whole process, it also helps in adjusting our model's parameters which is imperative in any machine learning model. Then we also set up an early stopping mechanism using the tensorflow's keras API, which is essentially used to prevent the overfitting of our model to make it more accurate as it will save the model that has the best performance on the validation set allowing us to work with the best model tested. We then set up tensorboard to log our training metrics for visualisation. This data has information about batches, such as the batch size and is updated after every round of training; allowing us to monitor our model in real-time and allowing us to have a better understanding of our code yet again. We then carry out the reconstruction process and finally; our model has been trained.

4. Results and Discussions

Through our results, we observed that the model accuracy is around 99%. Also through the scatterplot below we can see that fraudulent transactions are not random; they all clearly have some sort of pattern to them. Hence, it is possible to create a consistent machine learning model that can efficiently point out these fraudulent transactions in real-time, preventing credit card theft and people's finances.

In the context of this study, there are a few things thatcould be done to improve the results and generalise the model. This includes adding more layers to the encoder and decoders, this would have provided our model with deeper learning capabilities. Another thing that we could have done is use other activation techniques rather than just ELU and tested results, changing the activation method could have helped boost results by a lot, however, out of the methods that we tried in this study we found ELU to be most effective. We could have also tried different learning rates or changing up the batch sizes as having smaller batch sizes would mean that there would be more noise that the model would have to sort through which may help improve results, but at the same time having batch sizes that are too small would cause issues in processing time.

All in all, this model with its regularisation techniques, complex data pre-processing, loss functions, activation methods, and more make it extremely accurate and efficient. We use a plethora of machine learning techniques and skills that allow this model to be one that can be generalised very easily, one that can duplicate results and one that works with maximum precision. The applications of this model are widespread and can help us in many ways. It can help boost customer trust in financial institutions and online merchants hence boosting their will to interact and work with such vendors and institutions. In the financial industry, it can help fraud prevention in the first place and it can also help banks regularise their protocols and policies on fraud as it would be easier to identify and determine its degree. This model can also be applied to other places such as insurance companies who can use a similar model to detect insurance fraud. It can also be used in the medical sector to identify fraudulent claims and billing practices or in tourism companies to detect fake bookings, credit card fraud, and other types of fraudulent activities. This model can be adapted in many ways not just in terms of fraud detection but also to achieve other goals which can be met with machine learning. This is what makes this model so special and unique, it's wide amount of usage just by changing up small parts of its code is what makes it so unique.



In conclusion, the study demonstrates the effectiveness of machine learning algorithms, particularly autoencoders, in detecting credit card fraud. The high accuracy of the model underscores its potential as a valuable tool in the financial industry for real-time fraud detection and prevention. Future work may explore the integration of other machine-learning techniques and larger datasets to further enhance the model's reliability and applicability.

References

[1] https://arxiv.org/pdf/1904.10604.pdf
[2] The Optimized Anomaly Detection Models Based on an Approach of Dealing with Imbalanced Dataset for Credit Card Fraud Detection
[3] Anomaly Detection using Unsupervised Methods: Credit Card Fraud Case Study
[4] https://medium.com/analytics-vidhya/credit-card-fraud-detection-c66d1399c0b7
[5] Detection of Credit Card Fraud with an Autoencoder | Towards Data Science
[6] Anomaly Detection in Credit Card Fraud
[7] ELU Explained | Papers With Code.
[8] Understanding Latent Space in Machine Learning | by Ekin Tiu | Towards Data Science.
[9] https://d1wqtxts1xzle7.cloudfront.net/56975530/Building_Autoencoders_in_Keras-libre.pdf?1531314557=&response-content-disposition=inline%3B+filename%3DBuilding_Autoencoders_in_Keras.pdf&Expires=1697909762&Signature=Rj6G-NwH4Fb7KDvyTSI-jOO8tB5Gh0q~29fI7u8PbcW5BqpwfBATL6O5n-cCEQVh9MQrvIX1BGiF~t59WN1faLAx9wilDq~~PZd7sOooIE~6oZt78qDvWJ9H6fi-wEz3nCYq-5W3c4WZhtE5gKMA6re68PJD-UmZKHFOgK4ZOVraY~UslQtJIb6V4x1604z59zsawAigbt2lGsxUZiVqVDWlOf7QIZBNVTCTu1k7~lB3xRMfaXErzy5-nm9De1Dfcer-OZJYgqGdQ53111YLQsD0OWV8-BXaZF3IP2y9TA9K7kZLl1oy9kLWRQh0JWsHkyKgWpA~YEK1La7oTr43Wg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA
[10] A Comprehensive Guide on Optimizers in Deep Learning.
[11] https://ojs.aaai.org/index.php/AAAI/article/view/10894
[12] https://arxiv.org/abs/1702.05659
[13] https://www.sciencedirect.com/science/article/abs/pii/S0893608005801078
[14] Production Machine Learning Pipelines | Proceedings of the 2021 International Conference on Management of Data
[15] Identifying strong lenses with unsupervised machine learning using convolutional autoencoder | Monthly Notices of the Royal Astronomical Society | Oxford Academic