Identifying Credit Card Fraud With Anomaly Detection Algorithms

Eduardo Aguilar, Gyan Bains

CS66: Machine Learning

Abstract

Our project seeks to compare the effectiveness of three anomaly detection algorithms on a credit card transaction dataset. The three algorithms we chose were Isolation Forests, Local Outlier Factor, and a Gaussian Mixture Model. We used the Sci-Kit Learn implementation of the three algorithms then trained and tested them on a kaggle dataset consisting of 300,000 real credit card transactions. Next, we completed 30 iterations of repeated holdout testing. When choosing which evaluation metrics to focus on, we had to consider that too high of a false negative rate would mean that our algorithm is ineffective at what it is meant to do, leading to serious financial repercussions for consumers, financial institutions, and merchants. On the other hand, too many false positives could create a nuisance for the involved parties, leading to financial losses as well. With these factors in mind, we chose to focus on F2 score to place more weight on false negatives than false positives, and we do this by giving more weight to recall than to precision. We found our Isolation Forest to have a F2 score of 0.44, our Local Outlier Factor model had a F2 of 0.06, and finally, our Gaussian Mixture Model generated an F2 score of 0.81.

Introduction

Credit card fraud occurs when someone uses stolen credit card information to make unauthorized purchases or take cash advances out on an account that does not belong to them. It is the most common form of identity theft and plagues consumers and financial institutions worldwide. According to Bankrate, in 2021, “the FTC fielded nearly 390,000 reports of credit card fraud, making it one of the most common kinds of fraud in the U.S.” (2023). Zippia found “47% of Americans falling victim to [credit card fraud] within the past 5 years” (2022). Being a victim of credit card fraud can lead to monetary losses, credit score decreases, and personal information theft. Additionally, financial institutions incur costs associated with detecting, researching, and reimbursement of fraudulent transactions to cardholders. Thus, arriving at the central hypothesis of this project: How successfully can machine learning algorithms identify fraudulent credit card transactions?

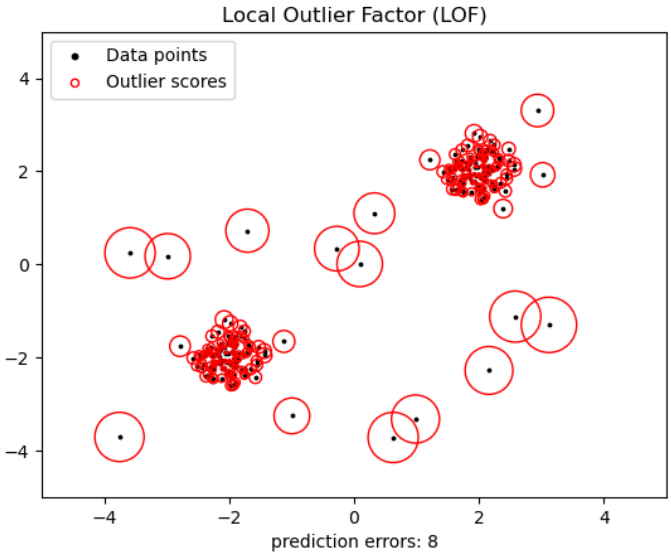
We aim to address the problem of credit card fraud by applying anomaly detection algorithms that can identify fraudulent transactions with high accuracy. The solution will involve use of the “credit card fraud detection” dataset, which is based on a month of real transactions by european credit card users, consisting of 300,000 transactions in total. We evaluate and compare the performance of three algorithms. We will know we *solved* our problem when we find an algorithm that produces a high recall and F2 score. Consumers with credit cards, merchants, and financial institutions would stand to benefit from a solution to this problem.

The three anomaly detection algorithms we will compare are an Isolation Forest, Local Outlier Factor, and Gaussian Mixture Model. Isolation forests work by creating several decision trees to partition the data. Points that can be partitioned with fewer splits are labeled as anomalies since they are unlikely to belong to the same distribution as the majority of the data. At a high level, The Local Outlier Factor algorithm works by calculating the density around a given point and marking points with a significantly lower density as anomalies. Finally, the Gaussian Mixture Model functions by fitting several Gaussian distributions to the data with the goal of representing the underlying probability distribution. Points with a low probability of belonging to one of the distributions are then labeled as anomalies. We used the Sci-kit Learn implementations of these three algorithms and compared their varying degrees of effectiveness at identifying fraudulent credit card transactions.

Algorithms

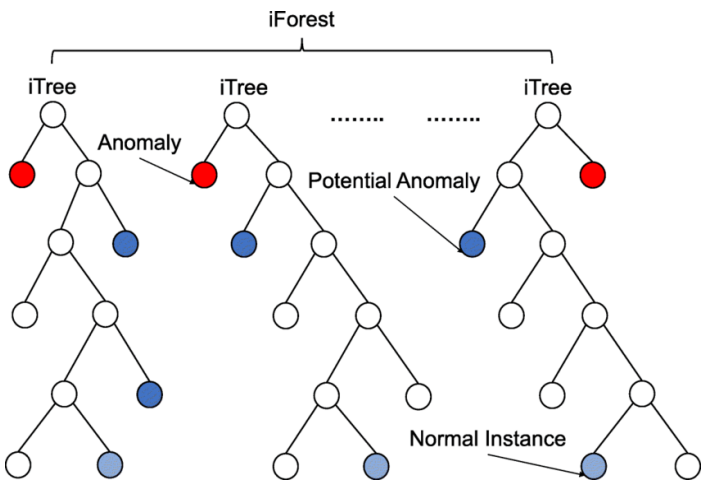
Local Outlier Factor

The Local Outlier Factor (LOF) algorithm is an unsupervised algorithm that is somewhat similar to k-Nearest Neighbors. The algorithm takes a parameter k and computes the sum of the distances between a given point and its k nearest neighbors. The algorithm then takes the average of the ratio of the density of the point and the density of the k nearest neighbors to assign the example a score, which is called the Local Outlier Factor. The LOF score essentially represents the density around a point, per the image (SciKit). If the density is high, then a point is less likely to be an anomaly. If the density is low, then a point is unique in some way and will be flagged as an outlier.



Isolation Forest

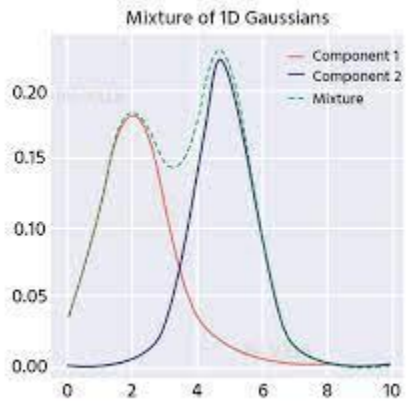
The Isolation Forest algorithm functions in a similar way to a Random Forest. The algorithm works by creating a forest of trees, with each tree randomly selecting a feature then recursively splitting the data on randomly selected values of that feature. The basic idea is that examples that can be more easily split out from the data are outliers since they have an unusual value for that



given feature, as seen in the image on the right (Regaya). The degree of ease that an example can be separated with can be measured by the number of partitions to isolate it. The algorithm identifies these outliers by taking an average of how many partitions it takes to isolate an example across several random trees.

Gaussian Mixture Model

Gaussian Mixture Model GMM) is an unsupervised algorithm that assumes that the data is generated by a mixture of gaussian distributions. The model has to estimate the parameters of a gaussian mixture distribution such as its own mean and variance. GMMs are powerful because they can combine several functions together to model complex data spaces, in a similar way as the image (Nixus) shows. Thus, we can use a GMM to try and model the dataset. The algorithm can then identify anomalies as the data points that do not fit the learned distribution.



Relevance of Models

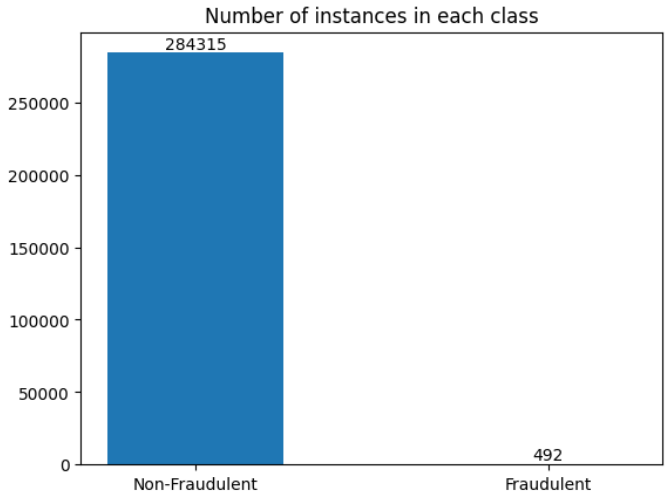
The three algorithms we used represent three unique ways to identify anomalies in a dataset. To identify fraud in a dataset of credit card transactions, the only reasonable strategy is to try to find the outliers. Most people have somewhat routine credit card spending habits in terms of location, goods purchased, transaction size, etc., so the strategy to identify fraudulent activity has to begin with finding the transactions that don't fit the pattern since fraudsters won't always know or be able to mimic the spending patterns of the identities they are trying to assume. Thus, anomaly detection algorithms can have some success in identifying credit card fraud.

Experimental Results

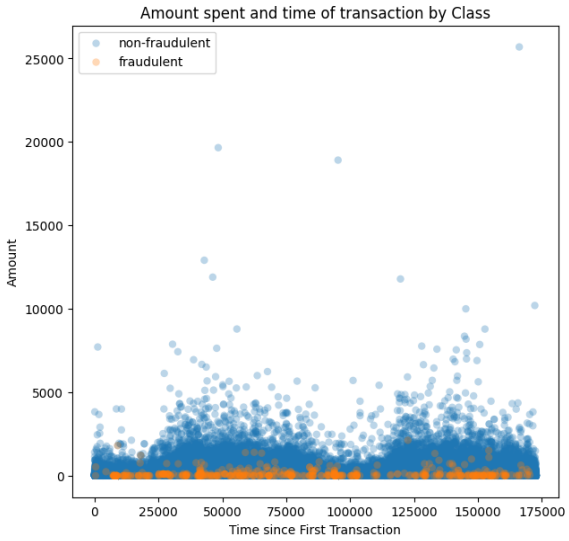
Dataset

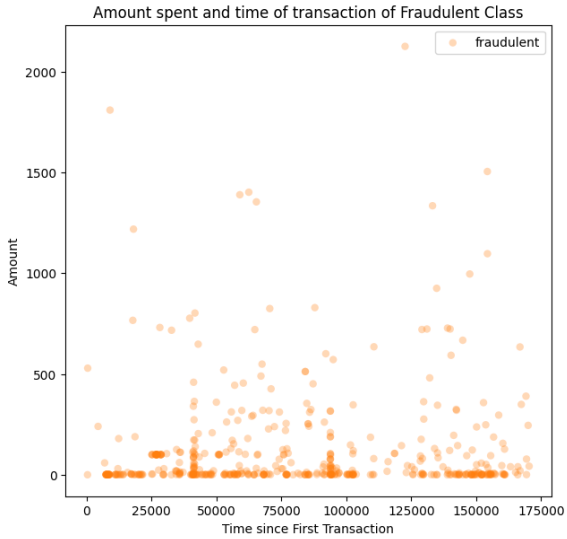
Our dataset consists of 492 frauds out of 284,807 total transactions completed by European Credit Card holders in September 2013. The data contains 31 features with "time", "amount", and "fraud" being the only three features that are not principal components. The names of the remaining 28 features principal components are not disclosed due to privacy concerns. Taking the features with labels, we performed some explanatory analysis in the beginning of the project to gain insight into the underlying data and inform future decision making.

Explanatory Analysis



This histogram demonstrates the class imbalance, clearly showing this is going to be a difficult problem to tackle due to the severity of the imbalance. This has implications for the performance of machine learning models that we will be using, as it may be the case that a particular model is not well suited to handle this severe class imbalance. Additionally, it is also an indication that sampling techniques could prove useful in this situation, however, we did not have time to do any.





These graphs give us two interesting findings. First, the distribution of fraudulent transactions across time is not uniformly distributed. Perhaps an indication that criminals aim to perform fraudulent transactions at specific times, in hopes that the transaction will go unnoticed. Secondly, the amount of fraudulent transactions are much lower than we anticipated. We theorize that criminals' main goal is for transactions to go unnoticed, by taking small amounts of money over long periods of time rather than making one large transaction which hypothetically has a higher probability of being noticed.

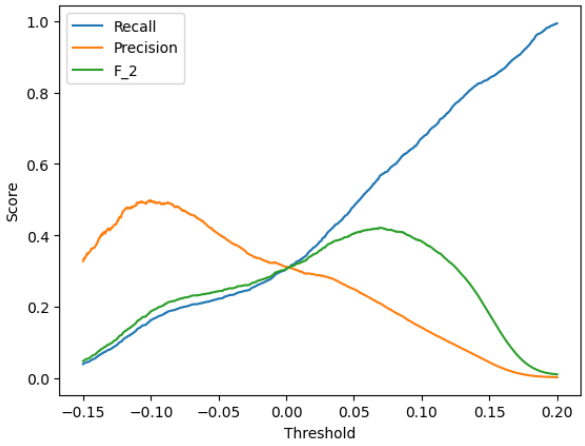
Testing Procedure

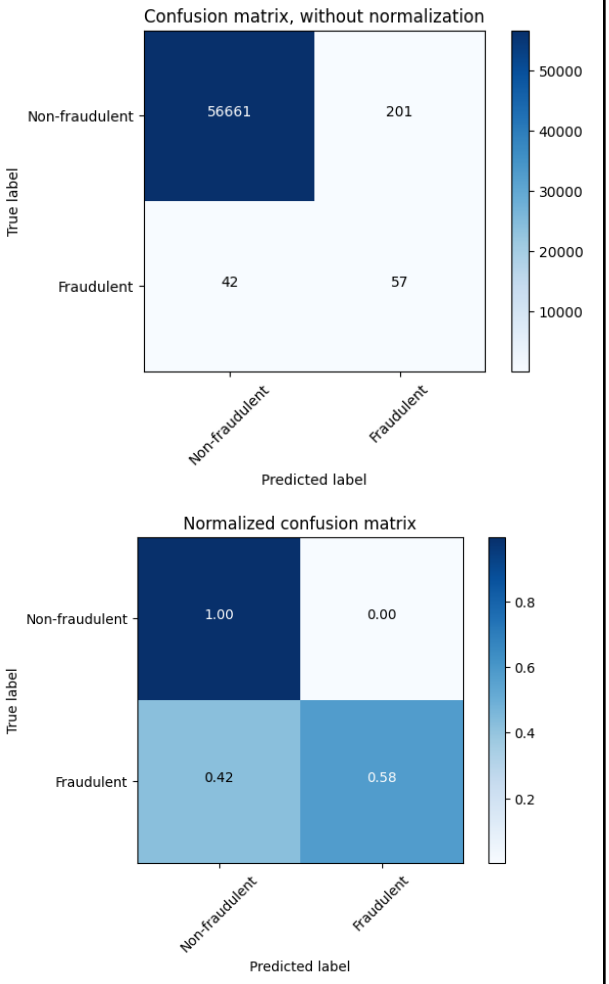
We ran repeated holdout validation for each of our three algorithms. Given time constraints, we decided to run more iterations on the algorithms that showed more promising preliminary results. Additionally, we made some modifications to our testing pipeline as needed for each algorithm. The specific modifications and results for each algorithm are described in detail below.

Isolation Forest

IF (30 iterations/ repeated hold-out validation)

Mean F2 =0.43515491774697834,
variance=0.0014291356396904113



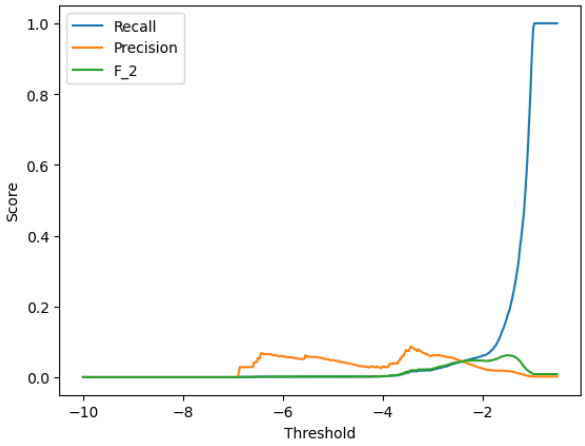


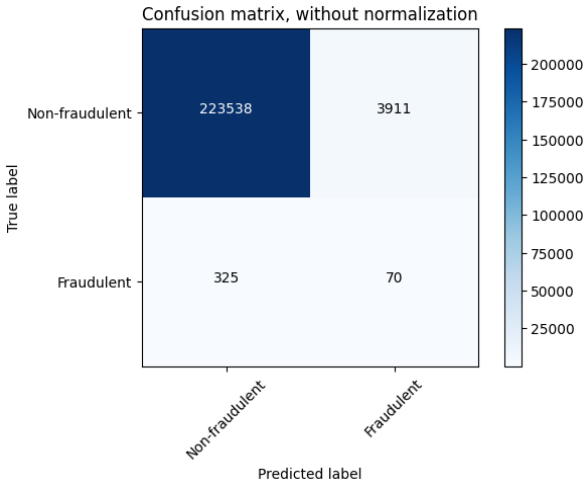
Our Isolation Forest found some moderate success in identifying credit card fraud. For testing, we ran repeated hold out validation. The plot above shows the average precision, recall, and F2 for each of 1000 threshold levels between -0.15 and 0.20. The threshold is a normalized score that an example needs in order to be classified as an outlier. The optimal threshold is about 0.07, where the F2 score is about 0.435. The most alarming part of the results is the high false positive rate. 201 examples were falsely labeled as fraudulent, which could be a potential problem given the context of the problem. We were pleased with the low false negative rate, which is even more important than false positives for credit card fraud detection.

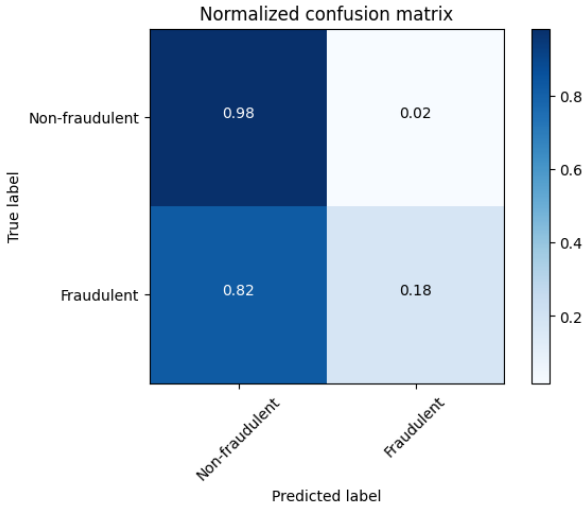
Local Outlier Factor

LOF (10 iterations)

Mean F2=0.06360002411499134,
variance=8.175308703509792e-06





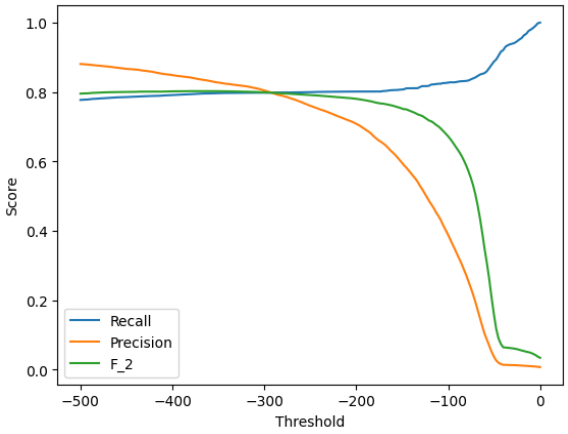


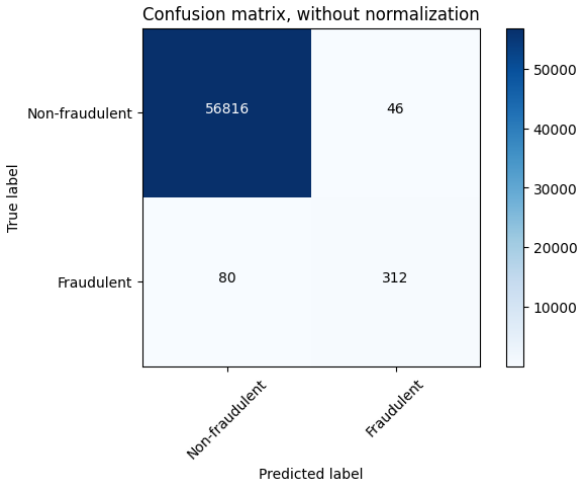
The Local Outlier Factor Algorithm was by far the least successful of the three algorithms. Given its lack of promise in early tests, we only ran 10 iterations of repeated hold out testing. We then plotted the average Recall, Precision, and F2 scores for 500 points between a threshold of -10 and -0.5. The results that we found were particularly disappointing. Both the false positive and false negative rates were higher than the other two algorithms. Our attempts at hyperparameter tuning did very little to alleviate the problem, so we decided that LOF might not be the best fit for this problem.

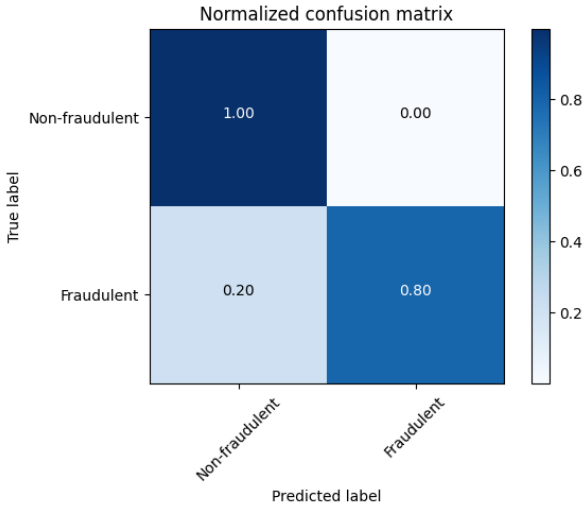
Gaussian Mixture Model

GMM (50 iterations/train only normal data)

Mean F2=0.8096195614866094,
variance=5.380901463413736e-05







The Gaussian Mixture Model was the most successful of our three algorithms. We trained the model on a subset of data consisting entirely of non fraudulent transactions. We then used the rest of the data to run repeated holdout validation and plotted the average precision, recall, and F2 scores across 200 threshold levels between -500 and 0. At the optimal threshold level, we found a mean F2 score of about 0.810. We were very pleased with the high recall and precision as well and the false positive rate was the lowest we observed amongst the three algorithms.

Discussion

The fact that our Local Outlier Factor model and Gaussian Mixture Model performed so differently was surprising given that both algorithms rely on underlying patterns in the data. The main difference between LOF and the other two algorithms is that it functions on a local basis in the sense that it only uses neighbors to determine if a point is an outlier. Perhaps that trait was detrimental to its success in this scenario since the other two global algorithms performed far better. We also could have spent some additional time tuning the hyperparameters of the LOF algorithm but instead opted to focus more on our other two algorithms since they produced higher baseline F2 scores. With regards to the Gaussian Mixture Model, we hypothesized that it did so well because the features in our dataset are approximately distributed. Surely enough, when we went back and plotted each of the Principal Components in our dataset, we found several of them to be approximately normally distributed. This strong match between algorithm and dataset was vital to its success in our opinion. This leads us to conclude that using the right algorithm for your data is a vital step in successfully tackling a problem using machine learning.

Social Implications

Credit card fraud transaction algorithms are used to analyze transactions both before and after they are authorized. For the purposes of analyzing social implications of a credit card transaction, we will consider both scenarios.

There are four key stakeholders in this situation; credit hard holders, financial institutions, merchants, and fraudsters.

Consumers can be impacted both positively and negatively by the algorithm. If the algorithm fails to identify fraudulent transactions, the consumer may incur financial losses. The consumer could be liable to pay for the transaction. Furthermore, if the fraudulent transaction creates a negative balance in the consumer's account, they may also be responsible for the associated fees. In severe cases, fraud can impact the credit score of a consumer. Poor credit can lead to higher interest rates and fewer loan options. It can also make finding a job more difficult. Through its effect on credit rating, the algorithm can have an effect on the long term financial well being of a consumer.

Consumers may also be inconvenienced by the algorithm. When fraud is detected, the consumer's credit card will be canceled. This can be particularly annoying if the algorithm falsely detects fraud. From the perspective of a consumer, having a credit card transaction declined because it has been incorrectly labeled as fraudulent can be embarrassing and frustrating. In the event that the consumer doesn't have another form of payment, it can even prevent them from making a purchase.

Even if the algorithm accurately identifies fraudulent activity on a consumer's credit card and cancels the card, it is an inconvenience for the consumer to wait for a replacement to arrive in the mail. However, this is a relatively small inconvenience for consumers compared to having fraudulent transactions on their statements.

In summary, consumers have a lot to gain from a credit card fraud detection algorithm but can be left in a vulnerable position if the algorithm fails to detect fraud, and can be inconvenienced if the algorithm falsely identifies valid transactions as fraudulent.

The algorithm also has mixed impacts on credit card issuers, which can be thought of as financial institutions. To begin with, there are significant costs associated with developing and maintaining a fraud detection algorithm. Not only do they need to cover hardware costs, but they will need to pay developers to create an algorithm and continually update it as fraud tactics progress.

Not only do they have to worry about tangible costs, but these financial institutions also have to balance their reputations with customer satisfaction. On one end of the spectrum, if an issuer fails to identify fraud on their credit cards, their reputation will be damaged and they might stand to lose consumers. The issuer might also stand to lose even more money since they can't expect a credit card holder to pay for fraud on their card. On the other hand, if an issuer falsely identifies fraud too frequently, consumers might grow annoyed and instead opt for a different card.

Financial institutions have to make significant investments to develop and maintain fraud detection algorithms, but they have to walk a careful line between protecting themselves and protecting the interests of their credit card holders.

The third impacted group consists of merchants that accept credit cards as a form of payment. They will naturally be interested in accepting credit cards because it will increase their customer base and sales. However, they are also affected by the same tradeoff that financial institutions face. If the algorithm fails to detect fraud, the merchants can be responsible for chargebacks, which is when the merchant is responsible for the cost of a fraudulent transaction that took place at their store or on their website. To avoid chargebacks, a merchant might require multi factor authentication or take other measures to prevent fraud, but these tactics come at the risk of irritating and potentially losing customers. The merchants will also be affected if the algorithm identifies legitimate transactions as fraudulent because that might prevent transactions from being completed, leading to the merchant losing sales.

In summary, merchants are at the mercy of the algorithm to some extent. They can take some measures of their own in an effort to prevent fraud, but face a tradeoff between protecting their financial interests and keeping customers happy.

The final group that the algorithm impacts is fraudsters. Unlike any of the previous groups, they do not stand to gain anything from the implementation of fraud detection algorithms. A successful algorithm will identify fraudulent activities and prevent them from taking place, which reduces profitability for fraudsters. They may also face legal repercussions if their identities are revealed by the algorithm. However, if a fraudster is able to learn how an algorithm works, they might become an adversarial threat and commit fraud in such a way that the algorithm will not detect it. That said, fraudsters can only hope that financial institutions and merchants are willing to concede some fraud in order to keep their customers happy.

In conclusion, having a fraud detection algorithm is a prerequisite for any credit card issuer. Customers, financial institutions, and merchants are all affected by the balance between false negatives and false positives while fraudsters stand with nothing to gain.

Future work and Conclusion

Given the severe class imbalance in the dataset, it is clear that machine learning models could potentially have lots to benefit from sample balancing. As mentioned above, keeping this severe imbalance through training and testing could hurt performance. “To improve the accuracy of anomaly detection, the number of abnormal samples should be expanded to ensure the balance of the dataset” (Shangguan et al., 2021). For these reasons, we would have liked to try some sampling techniques and see how that affected performance. Undersampling, oversampling, or maybe a mix of the two. Plus, this project was aimed at getting experience in different areas of machine learning, and this is one we wish we had time to research.

Regarding the refinement of our algorithms, our general approach was to get an algorithm working and then focus on tuning hyperparameters, tailoring our data pre-processing, etc. This approach worked well since we started with our Local Outlier Factor Model then immediately moved on once we recognized that it was not well suited for the problem. That said, it would have been nice to experiment further with hyperparameter tuning and other improvements even though it might never produce the F2 scores that our other models can.

Our best model was clearly the Gaussian Mixture. Since it did so well, we are naturally curious to explore how we can further improve its performance. One area we would like to explore is limiting which features we allow it to use. The underlying assumption of the data is that the features are approximately normally distributed but only some of our features actually were. It would be interesting to see how only allowing the model to use the normally distributed features might affect its performance.

Lastly, we would have liked to explore the performance of some other anomaly detection algorithms on this dataset. The next models on our list were Robust Covariance and One-Class SVM.

In conclusion, we were pleased with the results of our three algorithms, particularly the Gaussian Mixture Model. We got the opportunity to explore and understand the nuances of anomaly detection and learned that using an algorithm that is well suited for the dataset is vital to finding success.

References

A. Shangguan, G. Xie, L. Mu, R. Fei and X. Hei, "Abnormal samples oversampling for anomaly detection based on uniform scale strategy and closed area," in *IEEE Transactions on Knowledge and Data Engineering*, doi: 10.1109/TKDE.2021.3130595.

Egan, John. “Credit Card Fraud Statistics.” Edited by Liz Bingler, *Bankrate*, 12 Jan. 2023, https://www.bankrate.com/finance/credit-cards/credit-card-fraud-statistics/.

Flynn, Jack. “30+ Credit Card Statistics [2023]: Credit Card Debt, Fraud, Usage, and Ownership Facts.” *Zippia*, 20 Dec. 2022, https://www.zippia.com/advice/credit-card-statistics/.

R. Yousra, F. Fodil, A. Abbes. “Point-Denoise: Unsupervised Outlier Detection for 3D Point Clouds.” *MultiMedia Tools and Applications*, Springer Nature, July 2021, https://www.researchgate.net/publication/352017898_Point-Denoise_Unsupervised_outlier_detection_for_3D_point_clouds_enhancement.

"Credit Card Fraud Detection." *Kaggle*, 23 Mar. 2018, https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud.

"Outlier Detection with Local Outlier Factor (LOF)." *Scikit Learn*, https://scikit-learn.org/stable/auto_examples/neighbors/plot_lof_outlier_detection.html.

Dfnt. "Gaussian Mixture Model." *Nixus*, https://nixustechnologies.com/gaussian-mixture-model/.