Analyzing Credit Card Fraud with Anomaly Detection Algorithms

Luis Park LPARK4@SWARTHMORE.EDU
Neil Zhu NZHU1@SWARTHMORE.EDU

Abstract

This paper explores the precision performance of a Hierarchical Agglomerative Clustering Model, Mahalanobis Distance Model, Local Outlier Factor Model, and Isolation Forest in identifying fraudulent transactions in the Credit Card Fraud Kaggle Dataset. To show a statistically robust precision performance for Isolation Forest and Local Outlier Factor, a Stratified K-Fold Cross-Validation and Grid Search pipeline was implemented and employed. Because of the Agglomerative Clustering Model's temporal inability to utilize the pipeline, a repeated hold-out method was used to examine performance. Given the lack of parameters for Mahalanobis Distance, only Stratified K-Fold Cross-Validation was used. The pipeline showed a strong performance in precision scores for both the Isolation Forest and the Mahalanobis Distance model with average test precision scores of 0.66 and 0.24 accordingly. Our results for Local Outlier Factor and Hierarchical Agglomerative Clustering were extremely poor with both having an average precision score of 0.

This paper shows that the Mahalanobis Distance Model and Isolation Forest are viable models to identify fraudulent transactions in the Credit Card Fraud Kaggle Dataset and that the Local Outlier Model and Agglomerative Clustering Model are not.

1. Introduction

Credit card fraud is defined as a form of identity theft that involves an unauthorized taking of another's credit card information for the purpose of charging purchases to the account or removing funds from it ("Credit Card Fraud"). We were motivated to research the subject because recent inventions within machine learning like Chat-GPT have opened up the potential for machine learning to be used in fraudulent ways. For example, Chat-GPT could be used to make phishing emails more convincing which would allow bad actors to engage in malicious activity with personal data. We wanted to see if machine learning could be used to combat fraudulent activities, so we picked credit card fraud because of its occurrence every day and relevance to society. To provide context on how much credit card fraud occurs, the Federal Trade Commission (FTC) received 1.108 million reports of credit card fraud in 2022 (Caporal). Furthermore, credit card fraud isn't going away anytime soon as the Nilson Report, a report about the payments industry, projects that over $165.1 billion in losses will be incurred in the US over the next 10 years (Confessore). Experiencing credit card fraud can lead to significant monetary losses, identity theft, and poor financial records. Additionally, financial institutions need to devote considerable resources to combating credit card fraud through implementing anti-fraud solutions and reimbursing victims. Thus, we arrive at our central question for our machine learning project: How accurate are anomaly detection models at predicting if a given transaction is fraudulent? Based on this question, our hypothesis is that the algorithms with the most amount of representational power will perform the best given the high number of features within the dataset. We believe that Isolation Forests will perform the best as it has the most amount of representational power out of the four algorithms that we implemented.

We will address our hypothesis by testing our implementation of four algorithms anomaly detection models through the "Credit Card Fraud" dataset. The dataset we worked with was from Kaggle which contained 284,807 transactions made by European cardholders ina two-day period in September 2013. Only 492 or 0.172% of the transactions were fraudulent which led us to utilize unsupervised anomaly detection algorithms as we concluded that there wasn't going to be enough positive data to effectively use supervised algorithms. This was done through implementing four algorithms: Hierarchical Agglomerative Clustering, Isolation Forests, Local Outlier Factor, and Mahalanobis distance. Hierarchical Agglomerative Clus-

CPSC 66 Machine Learning Proceedings, Swarthmore College, Spring 2021.

tering essentially works by having each data point start out as its own cluster and merging ”similar” clusters together until convergence. Isolation Forests work as an ensemble method in which datapoints are randomly divided and placed into trees in which the depth of a datapoint in a tree is used to determine outliers. Local Outlier Factor works by taking a given data point and comparing the density of points surrounding neighboring points to the density of points surrounding the given point. Mahalanobis distance works by find the distance between a point and a distribution of points. We referenced online resources to create our anomaly detection algorithms from stratch. After writing the algorithms, we used metrics such precision, recall, and F1 score to examine the performance of our algorithms.

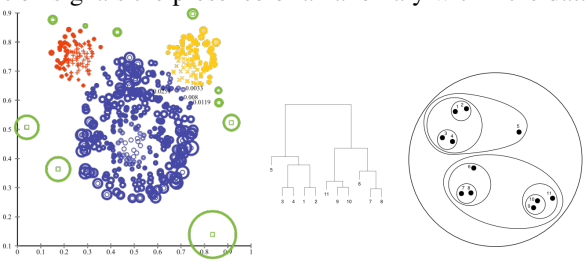
2. Methods

Hierarchical Agglomerative Clustering

Another model that was used to identify fraudulent instances was the Agglomerative Clustering model. At a high level, this model utilizes the continuous merging of ”similar” clusters until convergence. Then, the clusters that have only one data point should indicate a high likelihood of an anomaly. This model is initialized by setting each datapoint in a given dataset as a cluster. Then, the model identifies the two clusters that have the highest degree of similarity. The similarity of two clusters is defined by the distance between the two clusters. There are three distinct methodologies for calculating the distance between two clusters: single linkage, complete linkage, average linkage. Single linkage is defined by the smallest distance between two points in each cluster. Complete linkage is defined by the greatest distance between two points in each cluster. Average linkage is defined as the average distance between all points in each cluster. Additionally, there are two different metrics of calculating the distance between two points: Euclidean distance and Chebyshev distance. The Chebyshev metric is used because it measures the maximum distance between two points across all feature dimensions, thus making it a strong metric to capture anomalies with certain extreme feature values.

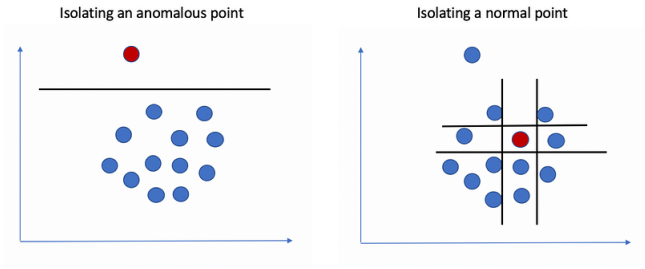
After identifying the two clusters that are most similar given both the linkage and distance metric, the two clusters are merged, and the merging process is continued iteratively until the distance between the two clusters no longer meets a specified threshold value. The threshold distance is crucial because eventually, two widely separated clusters will be considered the closest, making it unnecessary to merge dissimilar clusters. The threshold distance value ensures that such situations are avoided.

After all necessary clusters are merged, each cluster will be observed, and clusters with the size of one data point indicate a strong presence of an anomaly point. This is because if a single data point cluster has not merged with any other cluster throughout the entire process, the single data point cluster must possess extreme feature values. This observation signals the presence of an anomaly within the dataset.



Isolation Forests

A model that was used to identify fraudulent instances was an Isolation Forest. The Isolation Forest utilizes a forest of isolation trees that are each constructed recursively. The key principle in developing each tree is by selecting a random feature and a random value of that feature to quickly isolate anomalous data points. In more detail, for each tree, both a random feature “f” and a corresponding random value “r” are selected. Then, the data is partitioned by sending data points with a feature value less than “r” to the left child node, and the rest of the data points to the right child node. This is recursively repeated until the current node only contains one data point, thus isolating every data point in the data set. The rationale behind this methodology is that homogenous data points should have similar feature values. This consequently requires many splits to isolate each point of the homogenous data(resulting in deeper depth leaf nodes). Conversely, fraudulent instances will tend to have extreme feature values which consequently require very few splits to isolate the anomalous data(resulting in shallow depth leaf nodes). This indicates that a homogenous testing example will traverse down an isolation tree at great depths, whereas an anomalous testing example will traverse down an isolation tree at small depths. In result, depth will be the main metric to calculate the anomaly score.

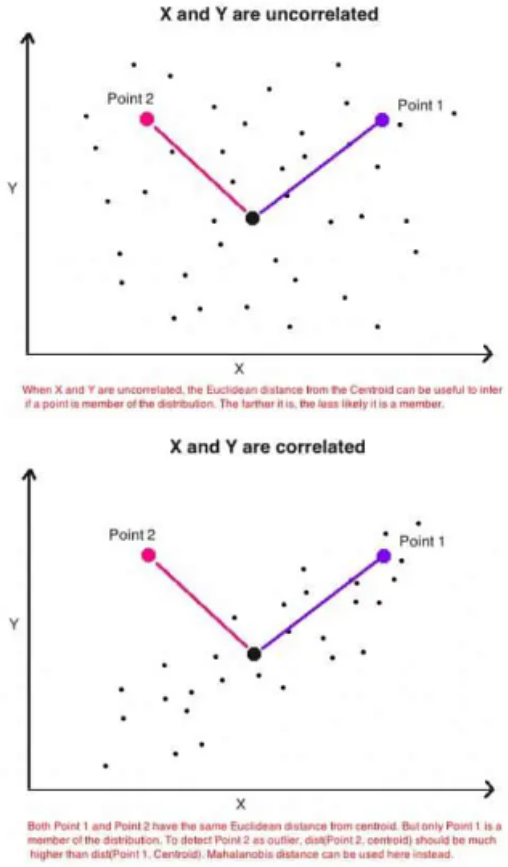


The calculation of an anomaly score will be denoted as $S(x, n) = 2^{-(c(n)/h(x))}$. In this equation, “x” represents the testing example, and “n” represents the number of leaf nodes. The function $c(n)$ expresses the average of

the average path length from the root to all leaves across all trees within the isolation forest. The function $h(x)$ expresses the average path length from root to leaf node given the testing example "x" across all trees within the isolation forest. Intuitively, an anomaly testing example should have a very low $h(x)$ value compared to the $c(n)$ output. This indicates that the average depth of an anomaly point should be minute compared to the average depth of all trees. As $h(x)$ decreases and $c(n)$ increases (as example becomes more anomalous), $-(c(n)/h(x))$ will converge to -infinity, thus making $S(x,n) = 2^{-(c(n)/h(x))}$ converge to 0. As $c(n)$ decreases and $h(x)$ increases (as example becomes more homogenous), $-(c(n)/h(x))$ will converge to 0 thus making $S(x,n) = 2^{-(c(n)/h(x))}$ converge to 1. To summarize, a score closer to 0 indicates a higher likelihood of anomalous behavior, while a higher score close to 1 indicates a low likelihood of anomalous behavior. The utilization of $2^{-(c(n)/h(x))}$ ensures a scalable and interpretable score that is conducive to effective anomaly identification. To test the isolation forest after training, each data point in the testing set is used to calculate the anomaly score and is stored in an array. Because the anomaly scores follow a normal distribution after confirmation of a Q-Q plot, a z-score was utilized as a cut-off method where if the z-score of an anomaly score lied beyond the threshold z score, it was labeled as an anomaly point. The appropriate z-score threshold should be identified when conducting a grid search.

Mahalanobis Distance

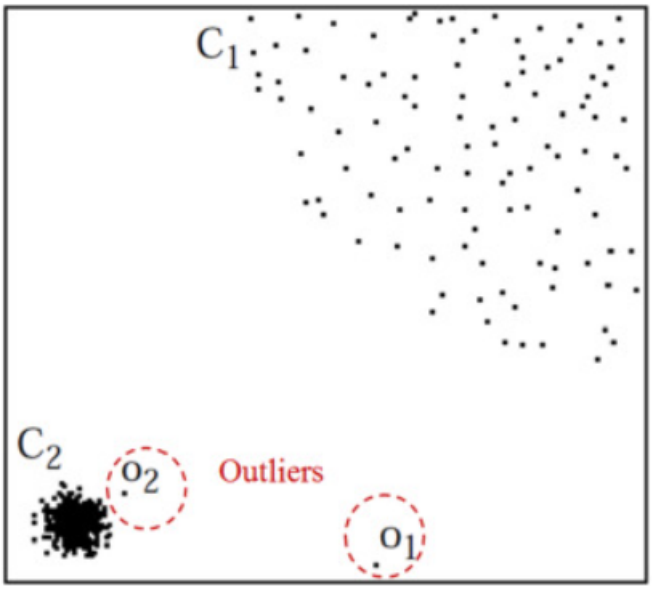
Mahalanobis distance is a statistical method that can be applied within an anomaly detection sphere. The method has to do with finding the distance between a point and a distribution of points. This is different from traditional distance metrics which seek to find a distance between two individual points. However, the true power of Mahalanobis distance comes from how it removes correlation between features in data. Thus, clusters which result from correlated data that can have cluster points farther away from a query point than an outlier point will normalize to look more like a normal, "round" cluster. This is can be seen in the graphs below from Machine Learning +.



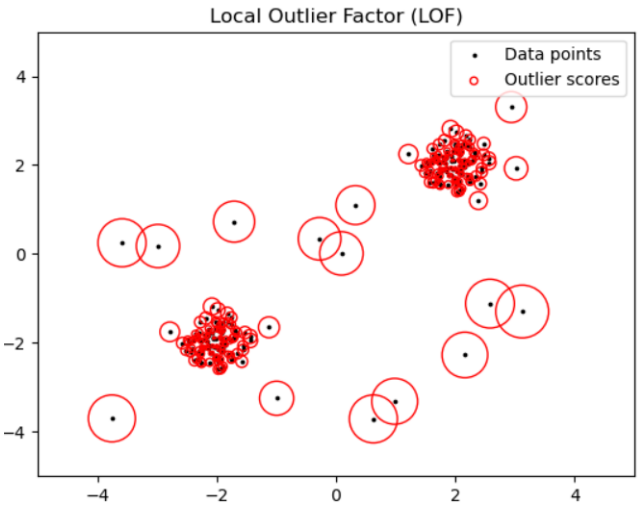
This method can help with data that has many different features that may correlate with one another to cause oddly shaped cluster spaces which is why we picked this method for the credit card dataset which has 30 features.

Local Outlier Factor

Local Outlier Factor is an unsupervised anomaly detection algorithm that is related to the k-nearest neighbors algorithm in that it utilizes the distances to neighboring points to create a score which reflects how much of an outlier it is. The key difference between a standard KNN anomaly detection algorithm and local outlier factor is how each algorithm deals with density. A standard KNN anomaly detection algorithm will not take into account the relative density of points that form a cluster. Thus, if there are different densities of clusters within an anomaly detection dataset, a KNN algorithm may misclassify a point due to the outliers being different distances from their nearest cluster. This can be seen in the graph below from ScienceDirect.



Local Outlier Factor essentially works by taking in a parameter k and finding calculating the distances for the k nearest neighbors of every point. Then, the algorithm finds a score, or Local Outlier Factor, for each data point. To find the Local Outlier Factor for a point, the algorithm finds the ratio between the average density of the points surrounding each neighbor of the point and the density of the point itself. Thus, if a Local Outlier Factor is low, the point has a similar density of points surrounding it as do its neighboring points which would indicate an inlier. Conversely, if a Local Outlier Factor is high, the point has a lower density of points surrounding it as do its neighboring point which would indicate an outlier. This can be seen in the graph below from Scikit Learn.



We thought that this algorithm would be better than a traditional KNN anomaly detection algorithm because it would be less affected by the lack of postively labeled data.

3. Experiments and Results

Hierarchical Aggomerative Clustering

Calculating the performance metric of the Hierarchical Agglomerative Clustering Model showed great difficulty for several reasons. The first being the run time complexity of the model. Because the clustering model needs to calculate the closest two clusters among all other clusters, the runtime complexity is O(n^2). Furthermore, the model needs to keep repeating the process of finding the two closest clusters until a certain threshold is met. This is an additional O(n) time complexity layered on top of the model, thus making the model run in O(n^3) time. The second main issue is the Credit Card Fraud Dataset. There are only about 0.1 percent of positive labels which shows a significant imbalance of labels. This then requires the model to take in a very large subset of the data to have a sufficient amount of positive labels to identify. For example, if the subset size is only 1000 examples, there is an estimate of 1 positive label. Observing the performance of the model with only one label to catch would be a poor metric of performance, thus a much larger subset is needed. With a sufficient subset of 20,000 examples and a time complexity of O(n^3), the runtime results in being extremely slow. To improve runtime, the model utilized python library "joblib" to run the model in ten CPU cores in parallel to significantly improve compute time. Despite the improvement in runtime, running the model through a K-Fold Cross Validation and a Grid Search is not within scope. After running the model in the pipeline for over 80 hours, the model still did not finish identifying the anomaly points. Because of the constraints of the model, the only reasonable way to test the model was using a repeated hold-out validation method where each test passed in a subset of 10,000 datapoints with different sets of hyperparameters. With this method, the model took approximately 18 hours to finish merging the clusters. For each hold-out validation, the precision score was 0 regardless of the hyperparameters.

Isolation Forest

The performance of the Isolation Forest is calculated using a pipeline that utilizes a Stratified K-Fold Cross-Validation and a Grid Search to ensure a statistically robust performance metric.

The pipeline function is first fed 20,000 random examples from the Credit Card Fraud Dataset and is then partitioned into two folds: a training and testing fold. The pipeline training fold is then sent to the Grid Search function that further partitions the pipeline training fold into two different folds: a training and development fold. The development fold is utilized to search for the best combination of hyperparameters for the Grid Search training fold. There are three distinct hyperparameters being observed for the

model: percent of features used, percent of data used, and the z-score cutoff threshold. In total, there are 48 different combinations of hyperparameters, and each will be tested for both folds in the Grid Search function in search for the "best" performing combination of hyperparameters. The performance metric for the hyperparameters is defined by the precision score. Anomaly detection models for credit card transactions usually optimize for precision score because credit card companies strictly want to avoid falsely accusing their customers of fraudulent transactions as this would result in losing clients.

After identifying the hyperparameters that produce the highest precision score from the Grid Search Function for the given training fold in the pipeline function, the pipeline function will use both the identified hyperparameters and the pipeline training fold to train the Isolation Forest. The precision of the trained Isolation Forest is then calculated using the pipeline testing fold. This entire process of finding the best hyperparameters and applying it to the pipeline training fold is done in two iterations; each pipeline fold being a training and testing set once. Thus, two precision scores and two combinations of hyperparameters should be produced.

Using this pipeline function, the isolation forest displayed two precision scores of 1.0 and 0.3158 with parameters: percentOfFeatures': 0.25, 'percentOfData': 0.1, 'z-score': -4.4, precision: 1.0 and 'percentOfFeatures': 0.1, 'percentOfData': 0.1, 'z-score': -4.2, 'precision': 0.3518

Mahalanobis Distance

We analyzed the performance of the Mahalanobis distance through Stratified K-Fold Cross-Validation while incorporating training and testing sets to ensure consistent results. We didn't do a grid search for the best parameters because there aren't a significant amount of hyperparameters that are worth searching for within for Mahalanobis distance. Within our Stratified K-Fold Cross-Validation run, we utilized precision as our main metric to find the best threshold value and to rate the performance of our training and testing sets with 4 folds. Furthermore, due to computational restrictions having to do with memory constraints, we could only run 80,000 examples out of the 284,807 available examples in cross-validation. Additionally, since our data is ordered by time, we selected 80,000 random examples out of the available examples to ensure we weren't picked data specific to a certain time. In addition, we ensure there were a sufficient number of postive examples within our randomly as to not throw off our results.

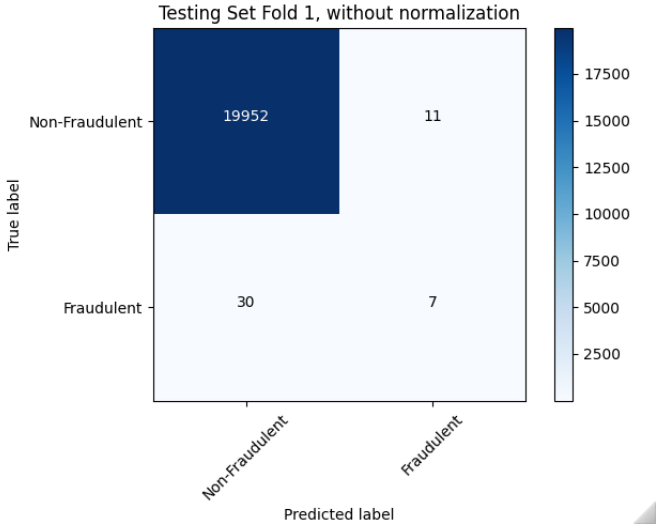
These were our results:

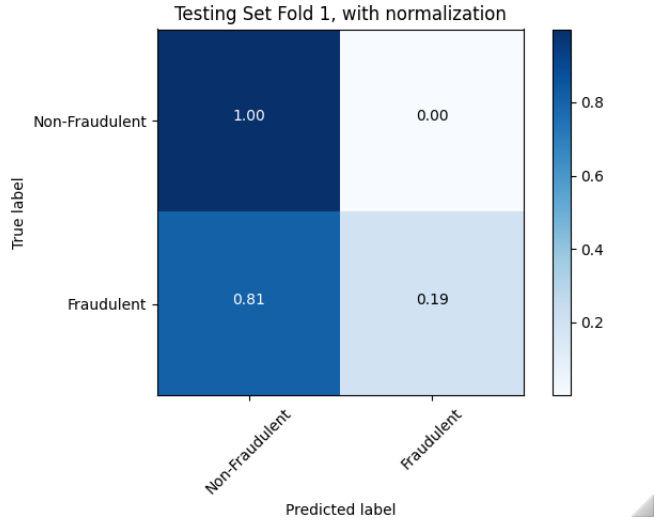
Number of positive examples in cross-validation: 150/492

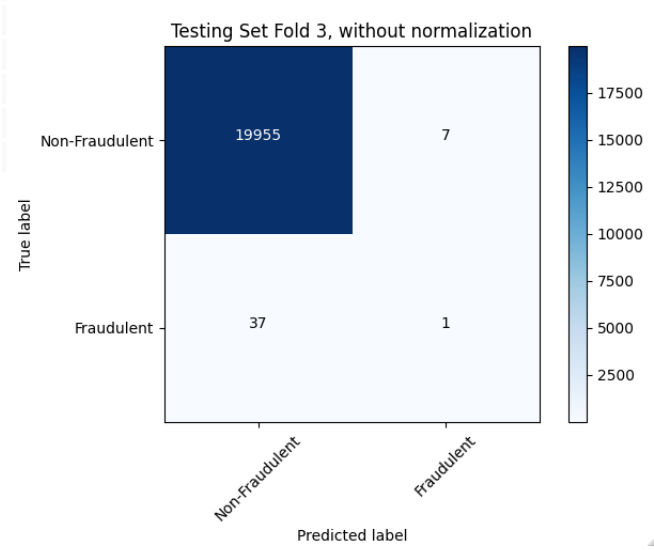
Fold	Training Precision	Training Recall	Training F1 Score
1	0.38	0.15	0.22
2	0.36	0.21	0.27
3	0.39	0.14	0.21
4	0.42	0.19	0.26
Mean	0.3875	0.1725	0.24

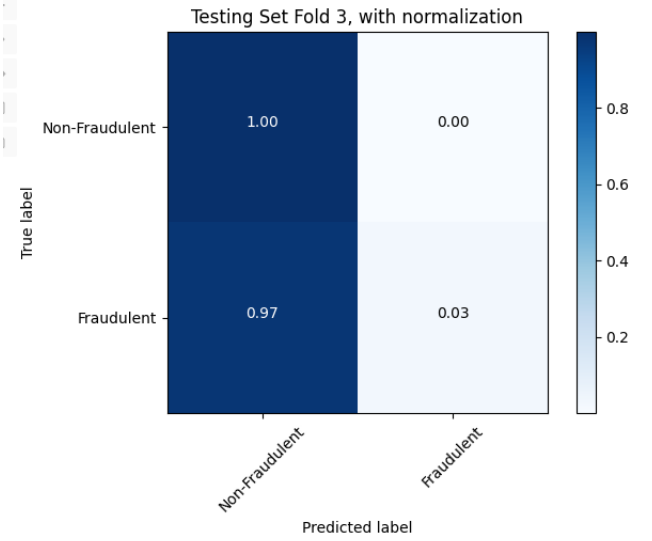
Fold	Testing Precision	Testing Recall	Testing F1 Score
1	0.39	0.19	0.25
2	0.21	0.11	0.14
3	0.13	0.03	0.04
4	0.22	0.11	0.14
Mean	0.2375	0.11	0.1425

Additionally, we created confusion matrices based on our testing set data. Here are confusion matrices for the best and worst performing folds for test sets in terms of precision.









Local Outlier Factor

We analyzed the performance of the Local Outlier Factor through a pipeline that utilized Stratified K-Fold Cross-Validation to ensure consistent results and a Grid Search to find the best performance parameters. We ran cross-validation with 4 folds on only 80,000 examples out of 284,807 available examples as this algorithm was computationally expensive and we didn't want to take up too much time finding the best parameters. Additionally, we randomly selected examples from the dataset as it was ordered by time. We used a Grid Search in order to find the best parameters for our Local Outlier Factor algorithm and tested two different distance metrics, Euclidean and Manhattan distance, and three different k values, 5, 10, and 25. In the end, we found that Euclidean distance with a k value of 5 performed the best for a majority of the folds in terms of testing precision.

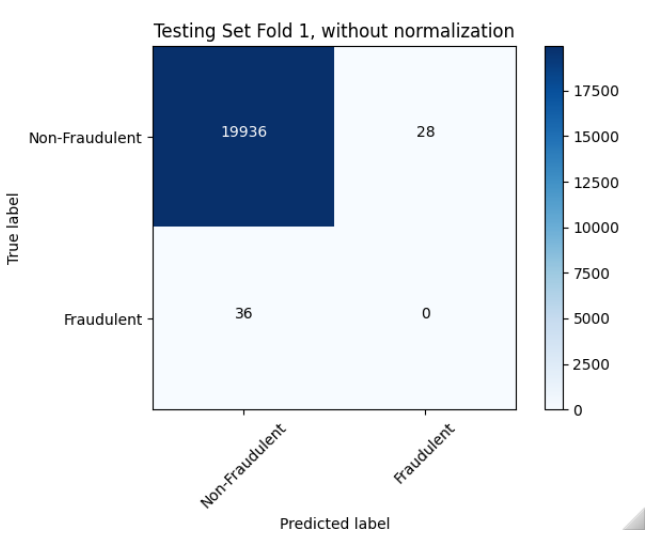
Then, we ran a Stratified K-Fold Cross-Validation again on 80,000 randomly selected examples with 4 folds using the hyperparameters that we found to more closely examine the data. The results are listed below.

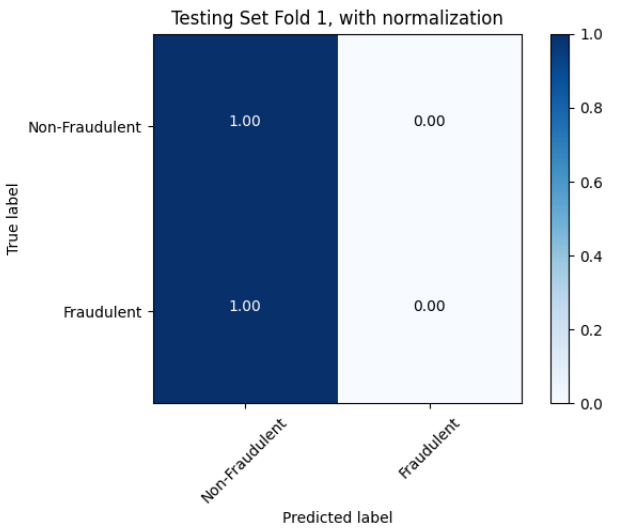
Number of positive examples in cross-validation: 145/492

Fold	Training Precision	Training Recall	Training F1 Score
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
Mean	0	0	0

Fold	Testing Precision	Testing Recall	Testing F1 Score
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
Mean	0	0	0

Despite our poor results for Local Outlier Factor, we wanted to share some confusion matrices of our results for one of the folds to display how the algorithm was classifying examples.





Also, we took a look at the difference in Local Outlier Factors between fraudulent credit card transactions and normal transactions. These averages were from a random sample of 20,000 examples. Here are our results.

Number of positive examples: 37/492

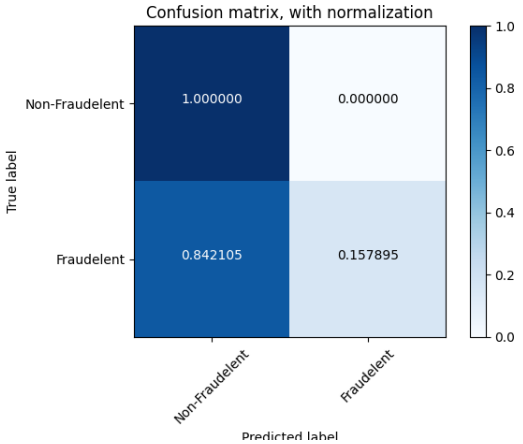
Average Local Outlier Factor for Normal Examples 1.17

Average Local Outlier Factor for Fraudulent Examples 1.21

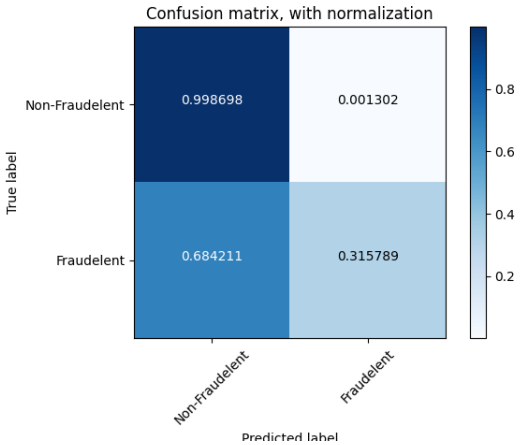
4. Discussion

Isolation Forest

The first set of identified best hyperparameters for the Isolation Forest showed a strong favor in a low percentage use of both features(25%) and data(10%) when training the Isolation Forest. This is because if each Isolation Tree utilizes a small number of features and data, each Isolation Tree will have high variance. Thus, the Isolation Forest becomes highly diverse and robust. The identified z-score threshold was at -4.4 which indicates that anomaly scores within the 0.05% percentile can only be identified as an anomaly. This z-score is satisfactory because there are 0.172% fraudelent transactions within the Credit Card Fraud Kaggle Dataset. With these hyperparameters, the Isolation Forest displayed a phenomenal precision score of 1 which indicates a perfect score of all predicted positive label to be true positive labels.



The second set of identified best hyperparameters for the Isolation Forest also showed a strong favor in a low percentage use of both features(10%) and data(10%). The z-score threshold was set at -4.2 which indicates anomaly scores within the 0.13% percentile are labeled fraudelent. With these hyperparameters, the Isolation Forest displayed a sub-par performance of a 0.3518 precision score. Because the z-score threshold was slightly higher, more examples were classified as an anomaly, thus leading to more False Positive classifcations which led to a lower performance of precision score. This is our conjecture as to why the Isolation Forest model was performing poorly with its assigned hyperparameters.



With two key componenets, the Isolation Forest model displayed good precision performance in catching fraudulent transactions. The first component is using a low percentage of features and data to train the model. The second component is setting the z-score threshold low enough to prevent large amounts of False Positive classifications.

Hierarchical Agglomerative Clustering

After extensive testing of the Hierarchical Agglomerative Clustering Model, the model showed an average precision score of 0, displaying an extremely poor performance of

the model. One plausible reason for these results are potentially noisy features from the dataset. Because 28 features are anonymized, munging noisy features is difficult because a qualitative analysis of the contribution of each feature can not be done. This results in the Agglomerative Clustering model having to take account of all features when calculating the distance across clusters, and some of these potential noisy features may be having erroneous contributions in the distance metric, thus leading the model to perform poorly.

Mahalanobis Distance

Our results for Mahalanobis distance were somewhat indicative of success. When optimizing our threshold for precision we were about to get an average precision of 0.3875 for our training data and an average precision of 0.2375 for our testing data. Given how 28 out of the 30 features have been altered with PCA, we can't say for sure what features might be causing this discrepancy between our training precision and testing precision. However, a reason that we've come up with to explain why our testing precision has been lower than the training precision is that the PCA modification of the original data only keeps the global trends of the original data. Thus, our algorithm wasn't able to capture all of the nuance the original data retained. Therefore, our Mahalanobis distance model was most likely only capturing the most obvious outliers in the data and thus, wasn't able to capture some more of the outliers and have better performance. This can be seen in how our training recall was 0.1725 and how our testing recall was 0.11 when optimizing for precision. Additionally, with thirty features, the model is working within 30 dimensional space which increases the potential that some features will create noise that will throw off distance calculations. Furthermore, since most of the features don't have names, we weren't be able to drop features that would seem to create noise in the dataset. Despite the worse average performance for our testing sets, there was great variation in the testing precision. The testing precision for fold 1 similar to its training score while the testing precision for fold 3 was 75% worse than its training precision. Due to computational restrictions, each fold was only able to receive about 37 outliers for each 20,000 sample testing set. Some of the variation in the testing precision could be attributed to the small number of outliers used in the testing set. Overall, our results for Mahalanobis distance were decent considering the modifications and dimensionality of the dataset.

Local Outlier Factor

Our results for Local Outlier Factor were disappointing. When optimizing our threshold for precision we weren't able to catch any instances of fraudulent cases across 4 folds of Stratified K-Fold Cross-Validation. This is most likely due to the fact that Local Outlier Factor has a tough time differentiating fraud instances from normal instances. As seen in the previous section, the average Local Outlier Factor for fraudulent cases is 1.21 and for normal cases is 1.17. Thus, a threshold value that can easily differentiate between the cases is hard to find. We conjecture that the distance metrics that Local Outlier Factor relies on couldn't accurately represent the dataset which contained 30 features. Unlike Mahalanobis distance, the clusters for Local Outlier Factor aren't altered so that clustering is in a "round" shape. Thus, we believe that the actual cluster space was too oddly shaped for the algorithm to differentiate between inliers and outliers. Additionally, the large number of features increases the likelihood that some features are introducing noise into the dataset which tend to severely affect the performance of algorithms similar to KNN like Local Outlier Factor as they use every feature within their distance calculations.

5. Social Implications

The implications for our work in the project extend to many groups such as credit card holders, financial institutions, merchants, and fraudsters. Credit card holders are impacted because they are the ones facing the consequences of fraud as they could face a loss of money, poor financial records, and a loss of personal information. Financial institutions are impacted because they need to maintain a good reputation and credit card fraud will impact that. Additionally, they will need to spend money compensating victims and combatting credit card fraud. Merchants are affected because fraudulent credit card transactions are often canceled, so they experience a loss of revenue. Fraudsters will obtain free products, money, or personal information at the expense of credit card holders.

Our project will help credit card holders, financial institutions, and merchants fight credit card fraud while limiting the ability of fraudsters to commit fraud. For example, if a credit card company implements algorithms similar to ours, they will be able to flag suspicious transactions so that credit card holders can be aware of any wrongdoing with their cards. This will prevent downstream impacts on financial institutions and merchants. As a result, fraudsters will lose power relvative to legitimate users of the credit card transaction system and stakeholders will have more trust within the system. However, due to the fact that financial institutions need to collect personal information to detect fraudulent transactions, this will empower financial institutions and the companies they share this information with as they will know more about their customers.

The two main concerns of how this may be used in the real-world are the collection of personal information and misidentification of fraud. Machine learning algorithms includiang anomaly detection algorithms are very data hun-

gry and companies will need to collect massive amounts of data in order to train and test effective models. This may provide a convenient excuse to these companies to collect data from their clients that may or may not be necessary. Additionally, it could lead to a loss of personal information as more institutions will be keeping large amount of their clients' data on hand which increases the risk that they will be targets of hacking. Additionally, fraud can be misidentified which can lead to legitimate users of credit cards having their accounts unnecessarily locked which can cause a lot of inconvenience for legimate users. As a result, companies will need to tune their algorithms to only identify transactions as fraudulent if they have a high degree of confidence.

Depending on one's perspective, the implementation of anomaly detection algorithms can either be good or bad. From a purely economic perspective, the implementation of anomaly detection machine learning algorithms by financial institutions is good because it reduces monetary damage from fraud. As stated in the introduction, $165.1 billion in losses are projected to be incurred in the US over the next 10 years (Confessore), so the economic gains from anomaly detection algorithms are significant. On the other hand, from a moral perspective, the implementation of anomaly detection algorithms might not be good as it encourages businesses to collect data about their clients. This feeds into existing business models in which companies collect data about their clients to sell to other companies. This data can be used for questionable purposes such as influencing politics (think about the Facebook-Cambridge Analytica scandal). Our work could fuel the normalization of data collection and erode the idea that personal information should be kept private.

6. Conclusions

In line with our hypothesis, Isolation Forests performed the best as it had the most capability to represent the cluster space of our dataset. Additionally, Mahalanobis distance produced viable results as it was able to reshape the cluster space to separate outliers from inliers. Unfortunately, our Hierarchical Aggolomerative Clustering and Local Outlier Factor models produced poor results. Hierarchical Agglomerative Clustering produced bad results because of the complexity of the clustering as simple distance metrics such as Euclidean and Chebyshev distance cannot account for the 30 dimensional space that we were working in. Similarly, Local Outlier Factor performed poorly as its Manhattan and Euclidean distance metrics couldn't work well within such a high dimensional space. Through our hands-on implementation of our algorithms, we also learned how to make our algorithms more efficient through reducing big O runtime and parallelization as our algorithms weren't particularly fast to train and test, and due to the class imbalance of the dataset, we needed to work with large sample sizes in order to produce statistically robust models. We were hoping to run our algorithms on larger datasets, but our limited resources in terms of computational power and time contrained us to smaller datasets. In the future, we could extend the project by incorporating supervised learning algorithms such as logistic regression into our testing as the Credit Card Fraud dataset had labels. Despite how much we learned through the project, we still have unanswered questions. One question that arised from this project was how much does PCA affect machine learning algorithms. After seeing the poor performance of some of our algorithms on the dataset, we were curious as to whether the PCA done to the dataset was negatively impacting our results. Another question that remains for us arised from the poor performance of Hierarchical Agglomerative Clustering and Local Outlier Factor models which displayed poor performance in a high dimensional dataset. In the future, we would love to know if they would perform better within a lower dimensional space which we could achieve by dropping some features. In reflection, despite our mixed results, we both learned a lot about anomaly detection and the process of implementing and experimenting with machine learning algorithms. We hope this project was an insight into how anomaly detection algorithms work with credit card fraud detection.

Acknowledgments

Akshara_416. "Anomaly Detection Using Isolation Forest - a Complete Guide." Analytics Vidhya, 13 Sept. 2023, www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/.

Belyadi, Hoss, and Alireza Haghighat. "Local Outlier Factor." ScienceDirect, ScienceDirect, 2021, www.sciencedirect.com/topics/computer-science/local-outlier-factor#: :text=The%20local%20outlier%20factor%20of,restricted%2

Caporal, Jack. "Identity Theft and Credit Card Fraud Statistics for 2023." The Ascent, The Motley Fool, 20 Nov. 2023, www.fool.com/the-ascent/research/identity-theft-credit-card-fraud-statistics/.

Confessore, Nicholas. "Cambridge Analytica and Facebook: The Scandal and the Fallout so Far." The New York Times, The New York Times, 4 Apr. 2018, www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html.

"Credit Card Fraud Detection." Kaggle, Machine Learning Group - ULB, 23 Mar. 2018, www.kaggle.com/datasets/mlg-ulb/creditcardfraud.

"Credit Card Fraud." Legal Information Institute, Cornell

University, www.law.cornell.edu/wex/credit_card_fraud. Accessed 15 Dec. 2023.

Jayaswal, Vaibhav. "Local Outlier Factor (LOF) - Algorithm for Outlier Identification." Medium, Towards Data Science, 30 Aug. 2020, towardsdatascience.com/local-outlier-factor-lof-algorithm-for-outlier-identification-8efb887d9843.

"Outlier Detection with Local Outlier Factor (LOF)." Scikit Learn, scikit-learn.org/stable/auto_examples/neighbors/plot_lof_outlier_detection.html. Accessed 15 Dec. 2023.

Mazarbhuiya, Fokrul. "Anomaly Detection Using Agglomerative Hierarchical Clustering Algorithm." Albaha University, Albaha, KSA.

Prabhakaran, Selva. "Mahalanobis Distance - Understanding the Math with Examples (Python)." Machine Learning Plus, Machine Learning Plus, 1 Mar. 2022, www.machinelearningplus.com/statistics/mahalanobis-distance/.