MSDS 411: Unsupervised Learning

Assignment 4

August 21, 2025

Anomaly detection algorithms for fraud detection

Yingyu Mao

Abstract

In this paper, I explored the use of unsupervised learning algorithms for fraud detection in a sales transaction dataset. Three algorithms were tested: DBSCAN, DBSCAN-LOF, and isolation forest. The best-performing model was DBSCAN, which achieved a high recall rate of over 90% for fraudulent events in the test set. The model's performance was evaluated using metrics like average precision, ROC-AUC, and F1-score. While DBSCAN performed well, the combined DBSCAN-LOF and Isolation Forest models showed poorer results, with average precision and ROC-AUC values of 0.28 and 0.79 for DBSCAN-LOF and 0.10 and 0.62 for Isolation Forest, respectively. The high recall rate of the DBSCAN model suggests it is effective at identifying fraudulent transactions in new data, though this comes at the cost of a higher rate of false positives. This trade-off is an important consideration for a business when deciding on an optimal cutoff for identifying fraud. The analysis also highlighted a limitation: the models struggled with out-of-range unseen data. In practice, this can be rectified by periodically retraining the model with newer data.

Keywords: DBSCAN, Local outlier factors, isolation forest, fraud detection

1.0 Introduction

Anomaly detection, or outlier detection, is the identification of data points that exhibit significant deviation from the majority of a dataset. These instances do not conform to a defined pattern of normal behavior and are thought to be generated by an alternative underlying process. Anomaly and novelty detection has applications where the outliers are of interest, such as detection of financial fraud, cyber-attacks in cybersecurity, suspicious behavior in video surveillance, or even as a standard step for data cleaning in any data analysis task.

The detection of fraudulent and anomalous transactions is a uniquely challenging domain for predictive modeling. The difficulty stems from two core issues: the inherent rarity of fraudulent events and the adaptive and evolving nature of malicious strategies. Since fraudulent patterns are not static and are often only identified retrospectively, the problem is not well-suited to traditional supervised learning, which relies on accurate labeling. Consequently, anomaly detection is approached as an unsupervised learning problem, where algorithms identify deviations from normal behavior without pre-labeled examples of fraud.

In this paper, I will explore different unsupervised learning algorithms for fraud detection in a sales transaction dataset.

2.0 Methods and Literature Review

Anomaly detection relies on effective learning of normal patterns. There are multiple approaches to learn the pattern of normal data points depending on the dimensionality of the dataset. Univariate data allows more traditional geometric and statistical techniques to separate the normal population and the outliers. Without assuming normal distribution or relying on the standard deviation, which are unreliable with highly skewed or multimodal data, the upper and lower bound of the data can be calculated as the third quartile plus 1.5 or 3 times the interquartile range and the first quartile minus 1.5 or 3 times the interquartile range, respectively.

With higher dimensional data, machine learning algorithms are used to identify the inherent groupings of the normal points. Density-based anomaly detection method assumes that normal points form tight and dense clusters while anomalies are outside of these clusters. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester, et al. 1996), a clustering algorithm that groups data points based on their density, can be used for this type of anomaly identification. Two key parameters affect the final output of DBSCAN: `eps` (epsilon or ϵ) and `min_samples`. `eps` is the maximum distance between two data points for them to be considered neighbors. A larger ϵ will consider more points as neighbors, potentially leading to larger clusters, while a smaller ϵ can leads to more and smaller clusters. `min_samples` is the minimum number of points in a point's ϵ-neighborhood required to be considered as a core point. The core points form the relatively dense regions of a dataset. Points that are not a core point but are reachable by the core points (within their ϵ-neighborhood) are border points. They do not have enough neighbors and form the less dense regions surrounding the core clusters. The points that are out of reach (not within the ϵ-neighborhood of any point) are consider outliers. When performing clustering, the cluster assignment of border points depends on the sequence of data points visited by the algorithm. Some suggest simply considering them as noise to avoid the borderline behavior

(Campello, et al. 2015). For anomaly detection, this group of data points could be an interesting area to focus on, especially if fraudulent events have evolved to mimic normal behavior.
DBSCAN assumes similar density across all clusters in a dataset. This is not usually the case, and the assumption could lead to misclassification of less clusters as outliers. Another density-based anomaly detection algorithm, LOF, addresses this issue by calculating a local outlier factor of each data point as its local reachability density with respect to the reachability density of its neighborhood (Breunig, et al. 2000). Regardless of the density of the neighborhood, the LOF value captures the degree of an object being an outlier within a local region. The outcome of the algorithm is sensitive to the parameter `n_neighbors`, as it determines how many data points to be included to compute local reachability density. Breunig et al. suggested using at least 10 neighbors for Gaussian and uniformly distributed clusters to stabilize the standard deviation of LOF values. The minimum of `n_neighbors` should also be at least the estimated size of a cluster, since if a cluster C contains fewer than `n_neighbors` objects, then the set of `n_neighbors` of each object in C will include a potential outlier p, and vice versa, resulting in incorrect LOF values.
Additionally, instead of relying on normal pattern/cluster identification to detect anomalies (which is usually computationally heavy and limited to low-dimension data and smaller datasets), another popular model-based method, isolation forest (iForest), uses isolation to directly isolate anomalies. The developers of the algorithm described anomalies as points that are minorities and have attribute values that differ a lot from the majority of normal instances. In hierarchical clustering, they can somehow be spotted as small or singular clusters that are at the top of the dendrogram. But instead of building the entire dendrogram for a dataset, which is also computationally heavy, isolation forest subsamples the dataset and builds partial models (iTrees), then uses an ensemble of iTrees to calculate the average path length of a point (steps needed to isolate a point) with respect to the average path length of the entire dataset. The shorter the path length, the higher the anomaly score. A ranking of anomaly scores is generated in the end, and a user can apply different cutoffs (parameter `contamination`) to predict the outliers. One other parameter to be specified is `n_estimators`, the number of iTrees to grow in an iForest. Depending on the complexity of the data (high dimensional with noisy attributes), ranking the attributes can be done for constructing iTrees. Most datasets tested converge with less than 100 iTrees.
The rest of this paper will describe the training process for these models on a sales transaction dataset and compare their performance on fraud detection.

3.0 Model Training and Results

3.1 Exploratory data analysis and data transformations

The dataset used in this paper contains systematic samples of sales observations from a case study reported by Torgo (Torgo 2017). The training set contains 133731 entries with no labeling of fraud or normal samples. The test set contains 15732 entries with "ok" label for normal samples and "fraud" label for fraudulent transactions. There are 5 columns of features for each set: `ID`, `Prod`, `Quant`, `Val`, `Insp`. The assignment instruction restricts the learning to be done without

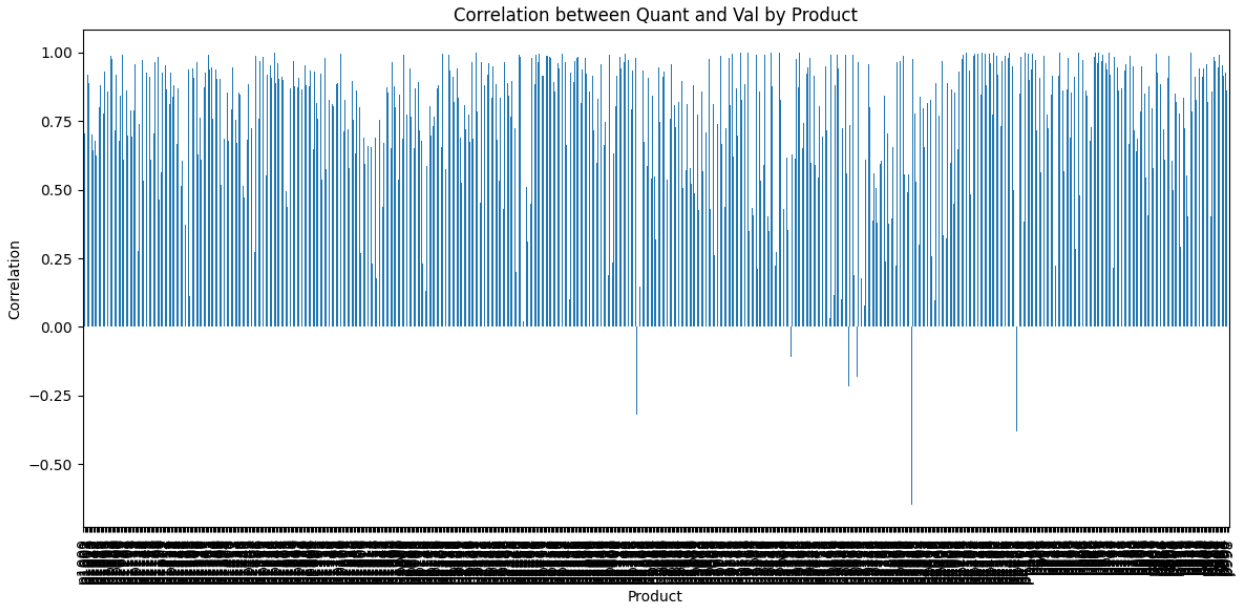
the `ID` column, which was excluded from further analysis. `Insp` column in the training set was also dropped. The labels of the test set in the `Insp` column were separated out as `y_test` to be used later for model evaluation. `Prod` contains information about the sales product category, a categorical datatype. `Quant` is the quantity of a product that the sales representative claims to have sold. `Val` is the value of the total sales that the sales representative claims to have sold. The latter two are numeric.

The training set and test set contain about 3.5% and 1.2% missing values, respectively. Neither is a significant amount. I simply dropped all entries with missing values. All test `Prod` categories can be found in the training set, which means predictions will not be made on new products. `Value_count` of `Prod` shows varying numbers for different product categories. Some products are rare, with less than 5 entries in the training set. These might pose challenges when establishing the statistical pattern of normal data.

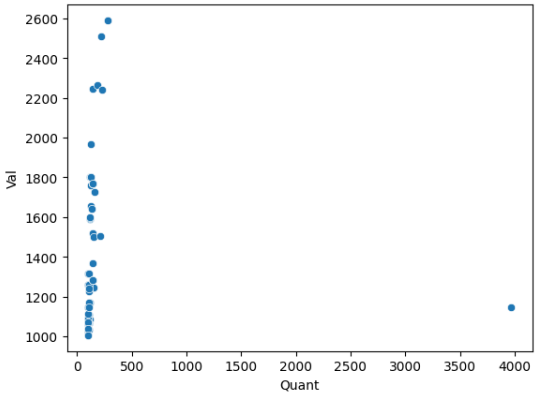
Distributions of `Quant` and `Val` are highly screwed. A log transformation was applied to these features followed by Min-Max scaling to preserve the positive values of `Quant` and `Val`. Since the products are different entities and assume different sales behavior, the scaler was fitted with the data within each product category instead of the entire dataset and then applied to the test data grouped by product category.

One can also choose to train unsupervised models with the entire dataset or build separate models for different product categories. To carry out training with all product categories at once, we need to code the categories in `Prod`. Since the categorical data is nominal instead of ordinal, usually one-hot encoding would be the mathematically correct way to encode it. However, this put the data in a much higher dimension and caused challenges for the learning of the computationally heavy DBSCAN. Isolation Forest also performed poorly with the high-dimensional binary data (data not shown). Another way to deal with the categorical data is to use Gower distance metric to precompute distance matrix of the mixed-type data for DBSCAN or LOF input. However, the large size of the training set makes it infeasible to calculate a 120k-by-120k distance matrix. This approach also makes the space-partitioning data structure of DBSCAN unavailable, which further increases the demand for memory. I eventually settled on label encoding for the `Prod` data. This resulted in a dataset with no added dimensions and all numeric data, which is easy to handle by all the algorithms used in this study. Additionally, if we remain the integer values in the `Prod` dimension (without scaling), the data points will be distributed discretely in this dimension (as clusters on parallel planes), and automatically far enough apart from each other as if they are analyzed separately when `Quant` and `Val` are both scaled between 0 and 1.

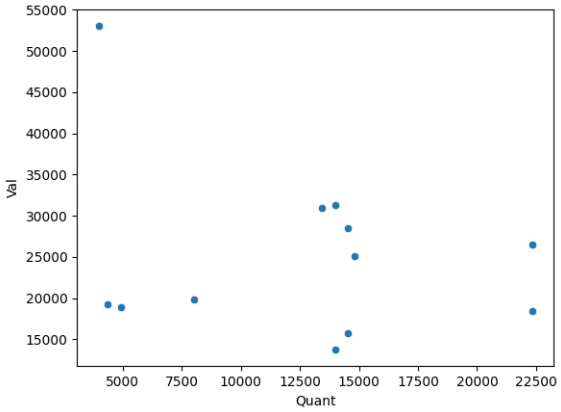
Logic suggests that `Quant` and `Val` should correlate well with each other. Their correlations within each product category are shown in the following figure. As expected, most of the correlations are > 0.5, with some < 0.5, and oddly several less than 0.



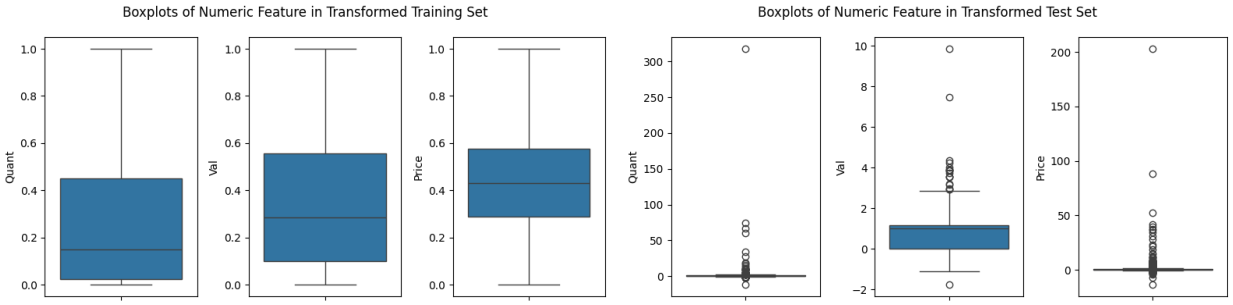
I picked out product `p2189` for further inspection by generating a scatter plot of `Quant` and `Val` as shown below. There was clearly a cluster of points on the left that follows a pattern, but one data point was found in the far-right corner of the graph, which is likely a fraudulent event. This event has high `Quant` but low `Val`, which means the per unit price deviates significantly from the normal pattern.



The scatter plot of another product category with negative `Quant`-`Val` correlation, `p4484`, is shown below. Here the normal pattern is less obvious, as there are not enough data points in this category. But by visual inspection, the data points at the bottom part of the graph still show some `Quant`-`Val` correlation. What distorts the correlation is most likely the data point at the top with low `Quant`, but high `Val`, which means the per unit price for this transaction significantly deviates from the normal price fluctuation range.



From the above exploratory data analysis, it appears that price captures another dimension of attributes of the sales transactions that are not directly shown by quantities and value. It might be reasonable to add a `Price` feature calculated as `Val/Quant` to perform downstream training. To show this feature has some predicting power, I used its interquartile range as a univariate outlier detection method. `Price` feature by itself was able to recall 58% of frauds in the test set, with an area under the ROC of 0.68 and a f1 score of 0.27. A quick comparison of a DBSCAN model trained only with `Quant` and `Val` verses with `Quant`, `Val`, and `Price`, the latter showed better performance: ROC-AUC 0.848 verses 0.789, average precision 0.364 verses 0.213. I added this feature with log transformation and Min-Max scaling for all model training in this paper. The final distributions of the three numerical features in the train and test sets are shown below.

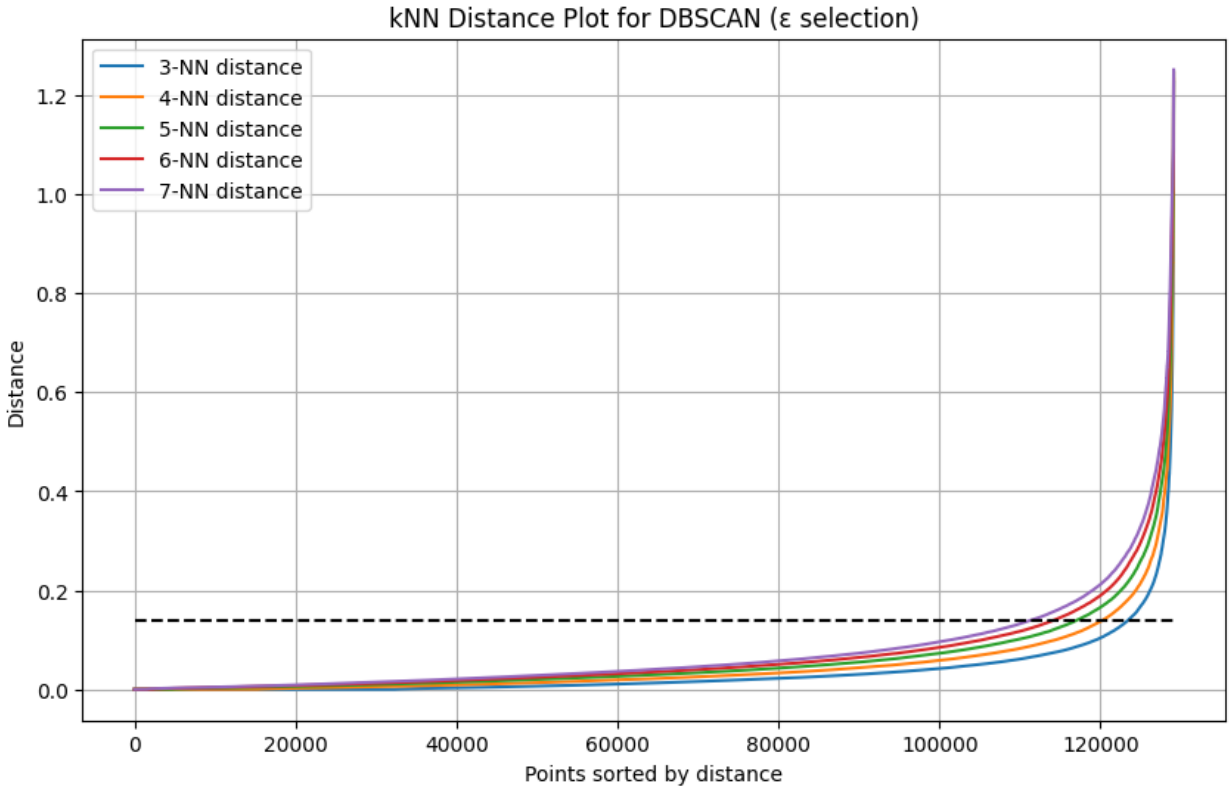


There are plenty of out-of-range data points in the test set as shown by the extreme values in the boxplot. However, this is the nature of machine learning: the training set cannot completely represent the test set. To truly test the robustness of a model, we need to know how it performs on unseen data that are in some cases out of the range of the training data.

3.2 DBSCAN

As mentioned in section 2.0 the two critical hyperparameters for DBSCAN model are `eps` and `min_samples`. If the training set contains labels, the approach to tune these parameters is to set up a parameter grid and select the sets of parameters that result in the highest adjusted Rand index (Hubert and Arabie 1985). With unlabeled data, internal clustering validity metrics such as silhouette score or heuristic methods are needed. Silhouette score compares the intra-cluster distances to the inter-cluster distances of all data points. It is more appropriate when the machine learning task is clustering instead of anomaly detection. In anomaly detection, the outlier “class” does not form a traditional sense of a tight cluster, which cannot be assessed with silhouette

score. The heuristic method for determining `min_samples` mentioned in Hahsler's paper (Hahsler, Piekenbrock and Doran 2019) is to use at least the number of dimensions of the dataset plus one. For `eps`, kNN distance plot of the data points in increasing order can be generated, and the knee in the plot, an estimate of distances of points within well-formed clusters vs the noise, can be used as `eps`, the maximum distance between two samples for one to be considered as in the neighborhood of the other. The kNN distance plot for different `min_samples` is shown below.

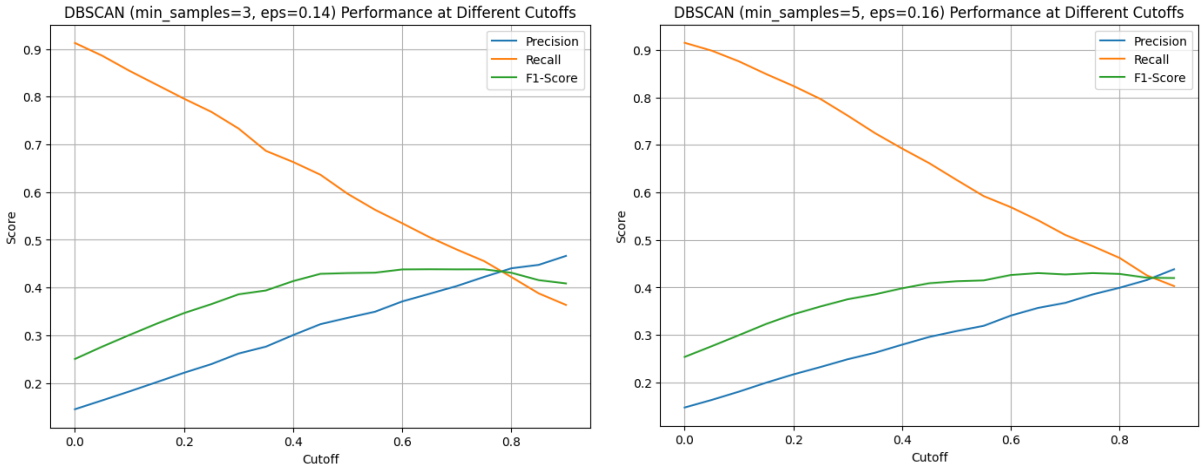


The 3-NN curve gave me the best elbow shape. I picked the `eps` based on this plot as 0.14. To compare different heuristic criteria for parameter selection, I trained the DBSCAN model on both `eps=0.14, min_samples=3`, and `eps=0.16, min_samples=5` (Hahsher's suggestion of using at least the number of data dimensions plus one). Since there is no prediction function in the Scikit-learn DBSCAN module, I defined a new function `dbscan_predict` to make predictions of the test set. First, the distances between each test data point and all core points in the train set were calculated, then the minimal distance was compared to `eps` of the model. If `min_dist <= eps`, then the test data point was determined to be a core point (probability of being an outlier was set to 0.0). Otherwise, the probability of being an outlier increased as the distance increased and was scaled between `eps` and the maximum distance in the training set (as shown in the kNN distance plot). The comparison of the two models is listed in the following table.

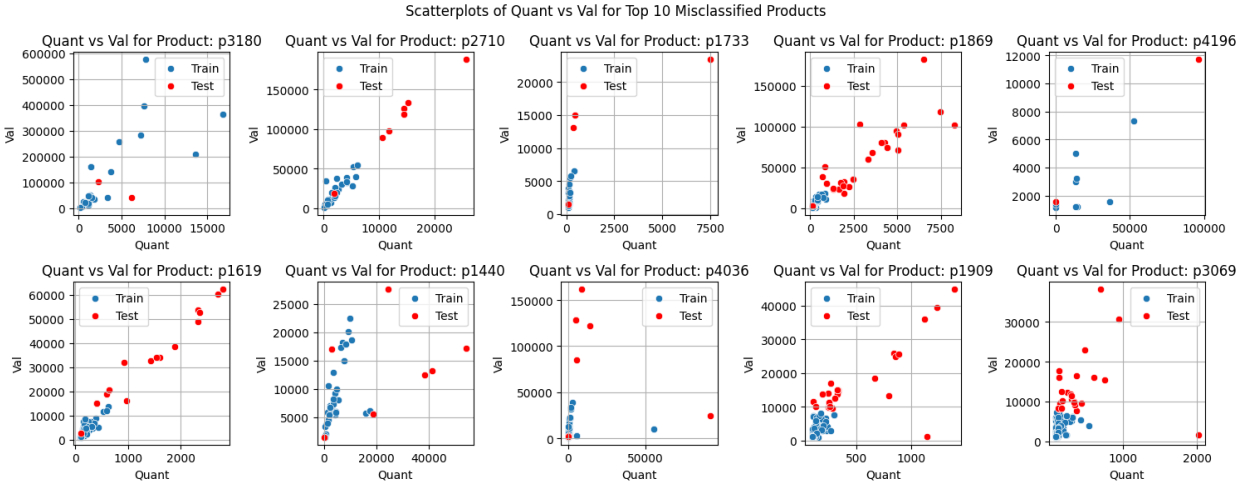
	Number of clusters	Number of outliers	Average Precision	ROC-AUC
`eps=0.14, min_samples=3`	2443	4550	0.363712	0.847932

`eps=0.16, min_samples=5`	1419	6826	0.349827	0.850254

Less `min_samples` resulted in more and smaller clusters, consequentially less detected outliers. The performance of the two models was similar, with some tradeoffs between the average precision of detecting the positive class and the overall accuracy of detecting both classes. Their precision-recall values at different cutoffs are also similar as illustrated below. These values suggest that at the 0.0 cutoff (only points within `eps` distance from the core points of the training set are inliers), 91% fraudulent events were recalled at the cost of 5.9 times events of false positives.



To get an idea of the misclassified transactions, I identified the top misclassified product categories and plotted the data points from both the train and test set. Interestingly, these are either rare product categories (p4196) or where test values are outside the normal range of the training set.



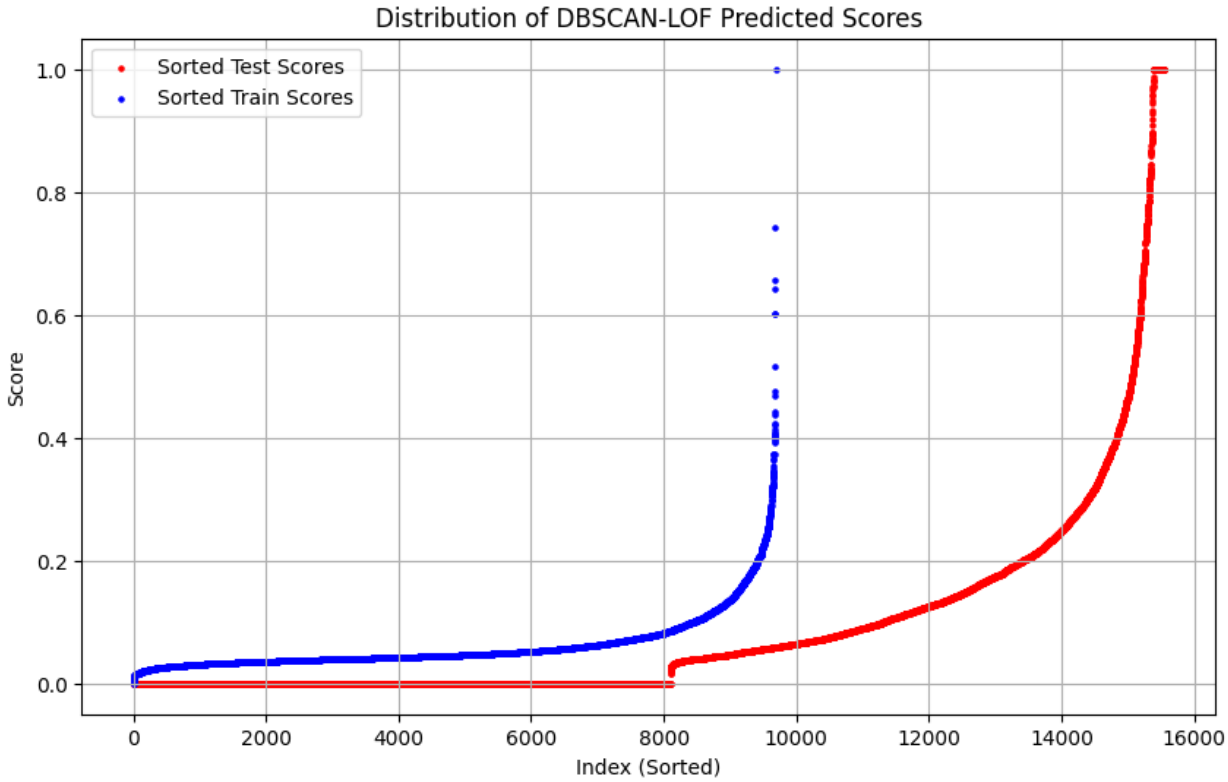
No further tuning was attempted to fit the test set as this could result in overfitting. As a matter of fact, simply scaling the training and test set with a scaler learned from both sets (this brings the out of range test data into range) resulted in a much higher average precision of 0.5 and higher

ROC-AUC of 0.863 (results in Python notebook). This does not mean that the model trained on both the train and test data will perform better in general but that it fits better on the test set than a model that has not seen the test set at all. However, it does show that the heuristic method for parameter estimation described above is a reliable approach to tune DBSCAN models for anomaly detection. Practically, depending on how the model is implemented, new data can be periodically incorporated into model training to update the model.

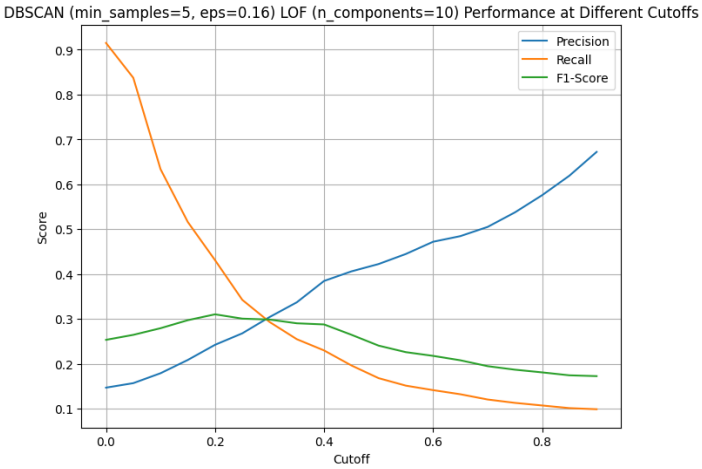
3.3 DBSCAN-LOF

DBSCAN model assumes similar density across all clusters, since the same `eps` and `min_samples` values are used to determine all clusters. This is, however, not necessarily the case as we have seen in Section 3.1 that one product category formed a denser cluster while the other is more scattered. The `eps` for product `p2189` can be set much smaller than that of `p4484`. To overcome this, one can train a separate DBSCAN model for each product category. However, this approach is impractical as the number of product categories grows. I attempted this approach using a universal `eps` cutoff at 0.04 based on the kNN distance plot of each product category. The training surprisingly only took 20s verses 2s with the single DBSCAN model. However, the resulting model's performance is worse than that trained on the entire dataset: average precision at 0.2, and ROC-AUC at 0.793. It did have a higher recall but at the cost of precision.

Another approach to address the varying density issue is to use the local outlier factor algorithm to reassess the low-density regions identified by DBSCAN. This could be just the outliers called by DBSCAN or outliers plus the border points. I used the outliers identified by DBSCAN (eps=0.16, min_samples=5) to train LOF, as this model identified tighter clusters and more outliers. Tuning of the hyperparameters, `n_neighbors` and `contamination`, was challenging without labels. I defaulted to the automatic method used in Breunig's paper to estimate the level of contamination and picked 10 neighbors for calculating LOF scores. Additionally, instead of relying on LOF prediction function, which is dependent on contamination, I estimated the probability of a test data point being an outlier using the minimum and maximum value of the LOF scores of the training set to scale the LOF scores of the test data, whose `min_dist` were larger than the `eps` of the DBSCAN model (probability was set to 0.0 if `min_dist <= eps`, as mentioned in the previous section). The higher the LOF score, the higher the probability of being an outlier. This resulted in the distribution of probability after ordering as shown below.



In the low-density region of the training set (blue dots), the probabilities of many points were heavily distributed below 0.1. There is a knee point around 0.1 where the LOF score goes up significantly, corresponding to the probability of being an outlier going up significantly. The knee point could potentially be used as the cutoff for converting the probability values to the class predictions of the test set. The test set in general has higher LOF scores, indicating the unseen patterns that are not represented by the training set. There is still a tail of data points with low LOF scores in the test set (low probability of being an outlier), potentially close to but outside the boundaries of DBSCAN clusters. Recall that at 0.0 cutoff (all points beyond the DBSCAN boundaries are all classified as outliers), the recall for the DBSCAN models was 0.915 but the precision was only 0.147, which means a lot of false positives. The hope is that LOF can discern the false positives with low LOF score and bring up precision value without affecting the recall. However, the DBSCAN-LOF model ended up having worse performance: average precision only at 0.28 and ROC-AUC at 0.79. Its precision and recall values at different cutoffs are graphed below. In comparison to the two DBSCAN models, its recall value dropped faster as the cutoff increased. This suggests that LOF trained on the training set could not effectively distinguish the outliers from the normal in the low-density regions of the test set.



3.4 Isolation Forest

Hyperparameter tuning for isolation forest is also challenging without labeled data. Tuning with labels in test set runs the risk of overfitting. However, the original iForest paper shows that most datasets converge with less than 100 trees (`n_estimators=100`). The anomaly rate (`contamination`) can be estimated with historical data in practice, but this assignment did not specify. I used the anomaly rate of the test set to set the parameter `contamination=0.08` to train the IF models. The resulting model's performance was worse than DBSCAN models: `AUC: 0.62, Precision: 0.10 Recall: 0.83, F1-Score: 0.19`. Training an iForest model for each product category only slightly increased the recall to 0.86.

4.0 Conclusions

Overall, the best model was the DBSCAN models built upon label-encoded product category feature, and log-transformed, Min-Max-scaled product quantity and sales value features, along with an engineered feature, product price. The highest recall rate of fraudulent events in unseen test data can reach over 90%, with 5 times more false positives than true positives. The management should consider the costs and benefits of recalling or missing a fraudulent transaction to determine the optimal cutoff in practice. Even after the model is implemented, the management should consider periodically retraining the model with newer data, especially when a drop in precision or recall is observed. Another limitation of the model seems to come from uncertainty with rare products (sparse regions in the data). Gathering more data points for these products to retrain the model or simply having manual inspections more frequently on the rare categories is also a viable next step.

References

Breunig, Markus M., Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. "LOF: Identifying Density-Based Local Outliers." *ACM SIGMOD Record* 29(2):93-104.

Campello, RJGB, D Moulavi, A Zimek, and J Sander. 2015. "Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection." *ACM Transactions on Knowledge Discovery from Data* 10(1), 5.

Ester, M., H. P. Kriegel, J. Sander, and X. Xu. 1996. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining.* Portland, OR: AAAI Press. 226-231.

Hahsler, Michael, Matthew Piekenbrock, and Derek Doran. 2019. "dbscan: Fast Density-Based Clustering with R." *Journal of Statistical Software* Volume 91, Issue 1. 1-30.

Hubert, L, and P Arabie. 1985. "Comparing Partitions." *Journal of Classification* 2(1), 193–218.

Torgo, Luis. 2017. *Data Mining with R: Learning with Case Studies (2nd ed.).* Boca Raton, Fla.: Chapman & Hall/CRC Press.