Taxi! Taxi! –NYC Taxi Volume Anomaly Detection

Comparing DBSCAN, OCSVM, and Autoencoder Approaches to Time-Series Anomaly Detection

Nicholas Vastine
College of Engineering
& Applied Science
University of Colorado Boulder
Boulder, CO USA
nick.vastine@colorado.edu

ABSTRACT

This project compared three anomaly detection methods on unlabeled, time-series NYC Taxi reporting: unsupervised density-based clustering (DBSCAN), semi-supervised one class classification support vector machine (OCSVM), and an autoencoder method based on a threshold reconstruction loss. The models are assessed on two datasets: an hourly-aggregate binning taxi volume per hour, and a daily-aggregate binning taxi volume per day. Further feature engineering was performed to retain time-series context including rolling averages, rate of change, and cyclic encoding of time features. Model were tuned to achieve a qualitatively justified 10-15% anomaly rate, and visually evaluated using taxi volume vs. time plots and PCA/t-SNE visualizations. All three models performed similarly on the simpler daily-aggregated data. OCSVM was easier to interpret on hourly-aggregated data than the other methods. OCSVM labeled hourly-aggregated data anomalies grouped by day (ie entire days of hourly data was marked as anomalous more often than individual hours), while DBSCAN and the autoencoder appeared to mark anomalies primarily in high variance commute windows (ie harder to predict taxi volume at 8 AM due to higher variance). The differences in model performance was explained by evaluating feature importance using the resulting anomaly labels in a Random Forest classifier. The models had similar feature importance in daily-aggregates which explained their similar results. However, DBSCAN and the autoencoder used count more heavily than OCSVM in hourly-aggregates, perhaps explaining the hourly variation. OCSVM instead relied on daily rolling average and rate of change in the hourly-aggregates, explaining its ability to label entire days as anomalies in the hourly-aggregated data. Of the tested methods, all three appeared suitable on the simpler day-aggregated data. OCSVM seemed most robust on the more complex hour-aggregated data. Future work may consider other features from the dataset to differentiate anomalies, evaluate the methods on labeled anomaly data for easier comparison, and compare algorithm performance to long short-term memory (LSTM) recurrent neural networks (RNN) designed for time-series data.

INTRODUCTION

Problem Statement: Anomaly detection performs a critical role in identifying irregularities and deviations in data across many industries. These applications range from fraudulent charges in finance, machine or sensor faults in industrial operation, and can be used in business to leverage opportunities or reduce risk [1]. This project intends to evaluate various anomaly detection techniques on unlabeled, time-series New York City taxi volume data with the intent to identify anomalies of high volume and low volume. Identifying such anomalies are both practically valuable for operations (justifying hiring additional drivers, adjusting scheduling, implementing time-limited offerings) and theoretically valuable to compare anomaly detection methods and limitations (unlabeled time-series, changing 'normal conditions', controlling for variables like day of the week or seasonality).

Justification*:* Anomalies come in many shapes and sizes. Approaches vary to identify global outliers (single points deviating from all points), contextual outliers (outlier within specific scenario), and collective outliers (groups of objects which differ). Different methods are suitable for different datasets considering feature complexity, label availability, and allowable model complexity. The differences in approach and results become apparent by considering different approaches for the same dataset.

Limitations of Existing Solutions: As mentioned, there are many methods for anomaly detection. These methods are explored in more detail in *Related Work* since each approach has its own limitations to consider. Also note that far more comprehensive analyses on anomaly detection methods have been performed, such as Chandola et. al's "Anomaly detection: A survey" [1] as well as numerical examples based on generated data, such as scikit-learns' "Comparing anomaly detection algorithms for outlier detection on toy datasets" [2]. Thus, the intent of this project is not to revolutionize anomaly detection methodologies, but instead to strengthen my own understanding of time-series modeling in anomaly detection applications. My own understanding of these methods is the limitation this project seeks to overcome.

Related Problems*:* Many outlier detection approaches apply to classification problems, both supervised and unsupervised.

Time-series analysis is valuable to learn given the wide range of sequential data applications including sales variation, text, or audio processing. The proliferation with the Internet of Things (IoT) [1] also reveals how anomaly detection can integrate into these data streams for live monitoring while following changing 'normal' conditions.

This project will also consider neural network autoencoder implementation, valuable to integrate with other neural network structures such as recurrent neural networks (RNN) for time-series or convolutional neural networks (CNN) for images and video applications.

While this project focuses on anomaly detection methodologies, these methods serve as exposure to many related problems and topics as well.

***Potential Contribution**:* While this project will not revolutionize anomaly detection, it intends to provide a basic understanding of their implementation, strengths, weaknesses, and applications of various anomaly detection techniques. The project will also consider feature engineering approaches to represent time-series data for non-time-series anomaly detection methods.

RELATED WORK

***Topics**:* There is no shortage of anomaly detection applications across many industries. Some anomalies may be discrete, such as spam emails, fraudulent transactions, and disease detection. Other anomalies may be more contextual, for example a further drop in traffic on Christmas following reduced traffic in the preceding week. Many applications extend this contextual analysis for time-series data, relying on more complex structures such as RNN to contextualize the data. These time-series applications could capture machinery sensor faults, variations in traffic, or monitor server access in cybersecurity applications. Most interesting is the development of live time-series anomaly detection which can assess ongoing data streams to alert users to unusual activity.

***Methods**:* Anomaly detection can be performed using many of the machine learning methods learned through the DTSA5500 Data Mining courses and the larger CU Boulder MSDS degree. However, applying these techniques to anomaly detection reveal their own strengths, shortcomings, and trade-offs.

Binary classification methods such as decision trees, logistic regression, or SVM may suffice for labeled anomaly data. However, these approaches are limited by imbalanced data (few anomalies and many normal points), may not be robust under changing operating conditions (consider multiple "normal" operating conditions for a production line), and most importantly cannot apply to unlabeled data (such as sensor data). Previous work from a Supervised Machine Learning course explored several supervised methods in detail, and as such will not be the focus of this project.

Unsupervised methods or semi-supervised methods do not require labels to determine "normal" conditions. For example, clustering methods like DBSCAN use point density to identify outliers separate from a single or multiple 'normal' clusters. Semi-supervised methods including one-class classification (OCC) learn normal conditions by training exclusively on normal data. These classifiers then distinguish new points as normal or outlier using the learned decision boundaries [3].

The past decade of growth in deep learning and neural network applications too applies to anomaly detection. This project will consider an autoencoder structure trained on normal data which should exaggerate deviations from normal conditions. Autoencoders can be used independently using reconstruction loss to classify outliers or as preprocessing with other classification methods such as RNN.

***Previous Studies**:* There is a wealth of literature review comparing outlier analyses, listing approaches and their merits. [1]

The intent of *this* analysis, however, is to practice implementation rather than report on theory from other papers. By performing the implementation myself, not only do I build the necessary analysis skills, but also intimately familiarize myself with the methods. The literature reviews provide additional methods which could be assessed as future work beyond the scope of this project.

PROPOSED WORK

The intent of this project is to evaluate various anomaly detection methodologies to evaluate their ease of implementation and performance on unlabeled, time-series data. However, many anomaly detection techniques are not designed for time-series data, and as such this project also involves feature engineering to retain these time dependencies and understand their importance in detecting anomalous data.

This analysis will evaluate the chosen algorithms on hourly-aggregated data, and simpler daily-aggregated data to understand their ability and approach to identify anomalies.

***Data**:* This study is built around transportation data for New York City taxi cabs. The New York City Taxi & Limousine Commission (TLC) is responsible for licensing and regulation of the city's taxi cabs, for-hire vehicles, and more. The TLC estimates 200,000 TLC licensees complete 1,000,000 trips each day. [4] This study will focus on the quintessential 'yellow cab', called Medallion drivers.

The TLC provides monthly data, and annual aggregated data. This project will consider "2019 Yellow Taxi Trip Data" sourced from NYC Open Data [5] which avoids deviations from COVID.

Feature Engineering: This data includes 84.4M records with 18 features. These features include *trip_distance*, *fare_amount*, *tip_amount*, and most importantly timestamps for pickup.

The features provided create many opportunities for anomaly detection, such as outliers in trip distance, tip amount relative to fare, and even using Pickup/Dropoff locations to identify unusual trips within regions. The focus of this project, however, is variations in trip volume. Anomalies should include low travel on major holidays and consider the context of seasonal variation. The taxi data will be aggregated by summing the number of trip records in one-hour bins for hourly-aggregated data and in one-day bins for daily-aggregated data over the entire year of records.

Note that many anomaly detection methods do not model time-series dependencies directly, instead assuming each data point as independent. As such, new features to consider hour, day of the week, month, and season will be engineered to group similar samples. Likewise, features such as 1-week and 1-day rolling averages preserve smooth variation, while rate of change features exaggerate change. The specific features implemented include

rolling averages, rate of change, and cyclic encoding are found in the Feature Engineering section.

***Tools**:* This project will compare three anomaly detection methods:

Semi-supervised One-Class classification using Support Vector Machines (OCSVM): Note this study does not consider labeled data since labels are often rare or expensive to implement. Instead, the algorithm should learn normal conditions, then distinguish whether new data is normal or an anomaly. Semi-supervised OCC using SVM trains under a presumed normal dataset to develop a decision boundary. Scikit-learn provides a kernelized method and a stochastic gradient descent (SGD) method which scale quadratically and linearly with the number of samples respectively [2].

Unsupervised Density-Based Spatial Clustering of Applications with Noise (DBSCAN): DBSCAN is an unsupervised clustering method which relies on the density of similar points to identify outliers. DBSCAN is convenient as the number of clusters is not specified, and the clusters are instead dictated by the maximum distance between points.

Neural Network Autoencoders: Autoencoders are neural networks composed of an encoder and decoder. The encoder progressively reduces dimensions by reducing neurons in subsequent layers or performing convolutions. Data is reduced to the autoencoder bottleneck, often called the 'latent feature space'. Data is then decoded, expanding from the latent feature space back to the original feature set. If an autoencoder is designed and trained effectively, it should reconstruct the input information through this dimension encoding/decoding with relatively low reconstruction loss. If an anomaly is passed to the autoencoder trained primarily on normal data, the autoencoder will have a harder time reproducing the input and a higher reconstruction loss. Exceeding a set threshold reconstruction loss labels such points as anomalies.

The SVM and DBSCAN methods will be based on scikit-learn packages [6,7]. The autoencoder will be built in Keras using TensorFlow as a backend [8,9]. The model hyperparameters will be tuned to adjust the percentage of points labeled as anomalies. This tuning includes ε in DBSCAN which controls the maximum distance between points in a cluster, γ in OCSVM which controls the influence of points to smooth the decision boundary, and the threshold reconstruction loss in the autoencoder methods.

MAIN TASKS

Consider the following prioritized task list:

1. ***Data Preparation*** – Data sourcing. Binning volume data each hour (24*365 = 8760 points). Decomposing datetime into hour, weekday, day, month, season.
2. ***Exploratory Data Analysis*** – Volume distributions. Plot trends. Identify IQR outliers.
3. ***Feature Engineering*** – Develop time-series dependencies. Standardize / normalize features.
4. ***Modeling*** – Building and training algorithms: OCSVM, DBSCAN, Autoencoder.
5. ***Evaluation*** – Evaluate against statistical outliers. Visualize outliers on volume vs. time graph and PCA/ t-SNE based visualizations for qualitative evaluation.
6. ***Feature Importance*** – Evaluate importance of features in each model by applying Random Forest classifier to each model's labeled anomalies.

Additional work could consider feeding intentionally anomalous results, testing algorithms on other metrics from the dataset, and considering additional time-series features such as noting holidays or seasonal-trend decomposition.

DATA PREPARATION

The original dataset included 84.4M trip reports for NYC taxis in 2019. The 8 GB dataset took more than 45 minutes to download and was too large to process directly in Python using RAM. The dataset also included irrelevant features since the project focus is solely on trip volume.

A PostgreSQL database was implemented to load the data and query results. The SQL queries decomposed pickup timestamps for each record into hour, day, and month labels. The labeled data was then binned to aggregate total trips in each hour throughout the year (ex. 7-8AM on January 1). These binned records were further labeled with weekday and seasons to capture additional time-series dependencies. The dataset was reduced from 8 GB with 83.3M records to 258 KB with 8,760 records.

The final hourly-aggregated dataset included: count of trips, day, month, weekday, season, and a simplified datetime for plotting. Categorical variables were encoded using integers, for example weekdays were represented 0 to 6 and seasons as 0 to 3.

The separate daily-aggregated dataset was then constructed from the hourly-aggregated dataset to evaluate the algorithms on a simpler dataset.

EXPLORATORY DATA ANALYSIS

Time-Series Plotting: Initial analysis plotted the hourly aggregated data to understand trends. Figure 1 isolates the hourly aggregated data for the month of December to emphasize contextual variation in the data.

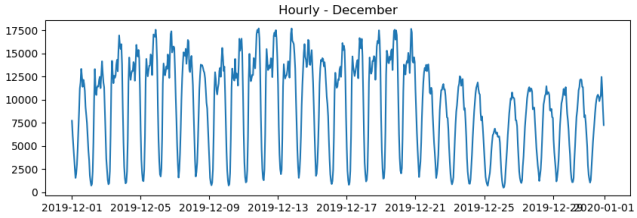

Figure 1: Hourly Aggregated Data for the month of December

Figure 1 demonstrates several key trends. Unsurprisingly, taxi volume is reduced in the middle of the night compared to midday, hence the rapid oscillation each day. Likewise, taxi volume is reduced on weekends without commuters, missing a distinct after-work spike in traffic. In the month of December, taxi volume is reduced during the week of Christmas, and even more so on Christmas day. The hourly plot highlights possible collective

outliers (the week of Christmas) and a contextual outlier (Christmas Day reduced relative the week of Christmas) which is severe enough to serve as a global outlier.

The December data exemplifies the importance of capturing context in time-series analysis. Figure 2 shows aggregated taxi volume per day (365 records) and by month (12 records) which reveal longer term trends not apparent in hourly analysis.

Figure 2: Hourly, Daily, and Monthly Aggregated Trips throughout 2019

By aggregating daily trips, the difference between weekdays and weekends is exaggerated. Anticipated holiday anomalies (Memorial Day in late May, Thanksgiving in late November, Christmas in late December) are also exaggerated in the daily aggregate plot.

Larger seasonal trends, such as reduced volume in the summer months and higher volume in spring months, are also more apparent in the monthly plot. These larger trends are crucial to establish relative 'normal' conditions to capture contextual outliers throughout the year.

Distribution Analysis: Consider that this data is composed of cycles, for example a daily 24-hour cycle, a weekly 7-day cycle, and an annual/seasonal cycle. Segmenting the data using these cycles can indicate distinct distributions of trip volume, for example higher day taxi volume than night and higher volume on weekdays than weekends.

Figure 3 presents the distribution of the hourly binned totals grouped by hour and weekday. Note the bi-modal distribution in each weekday representing midday and late-night travel.

Outliers: Outliers can be calculated using 1.5xIQR for various groupings. For example, hourly-aggregated outliers are flagged when the point is outside the 1.5xIQR for each hour of the day. Day-aggregated outliers are flagged for each weekday. These points are used as a basic evaluation metric for the algorithms. Figure 4 demonstrates how these outlier points flag many concerns but are not comprehensive, for example for collective outliers.

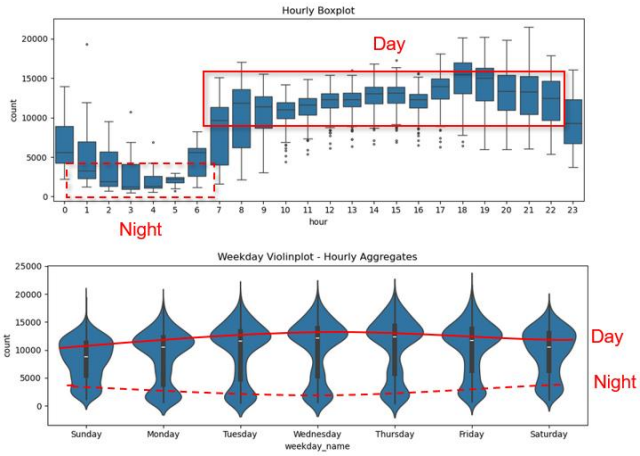

Figure 3: Distribution Plots reflecting distribution of binned hourly data grouped by hour and weekday.

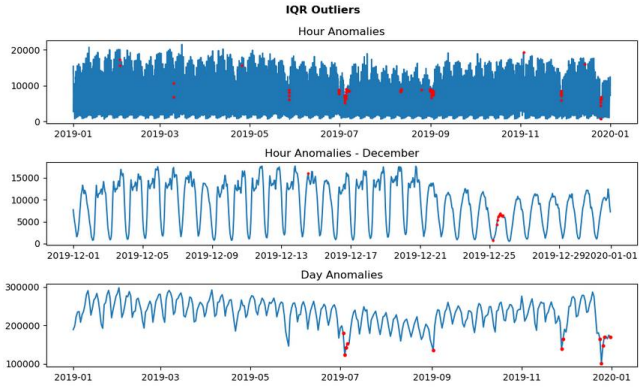

Figure 4: IQR-based Hourly Aggregate Outliers

FEATURE ENGINEERING

Recall the chosen anomaly detection methods are not designed for time-series data. As such, features must be engineered to capture these time-series relationships.

Rolling Average Features: Recall the additional insight gained by aggregating the hourly binned data into daily and monthly aggregates. These secondary aggregates reveal longer term variations such as the difference between weekdays and weekends or larger seasonal change.

A similar approach to aggregation is rolling average. For example, engineering the average volume over the previous 24 hours can reveal if one hour is below average, or if the entire day was below average. Likewise, rolling averages of longer periods (7-days, 14-days) capture longer term trends. This project implements 1-day, 7-day, and 14-day rolling averages, which are plotted in Figure 5.

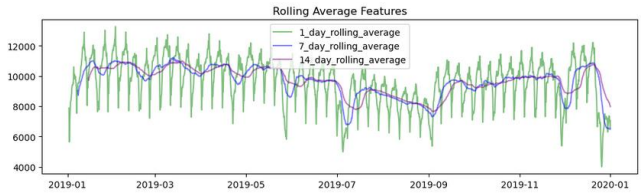

Figure 5: Rolling Average Features

Rolling averages smooth the rapid variation in the hourly aggregated data, capturing long term trends. For example, Figure 5 reveals dramatic dips during the holiday season.

Rate of Change: While rolling average features smooth rapid variation, rate of change features exaggerate rapid variation. These features evaluate the percentage change in the hourly- or daily-aggregate compared to one day ago and one week ago. For example, if a sample Monday were a holiday, a 7-day rate of change feature would show a dramatic decrease compared to the previous non-holiday Monday.

Figure 6 plots the 1-day and 7-day rate of change features for the hourly-aggregate data, which were also implemented for daily data analysis.

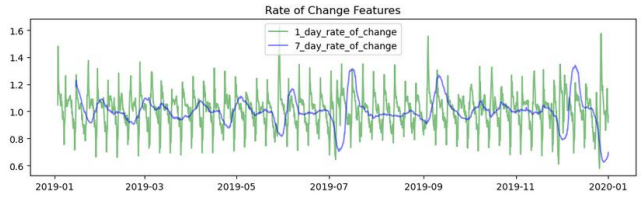

Figure 6: Rate of Change Features

Figure 6 reveals the value of these features to spot anomalies in the data. For example, consider the dramatic variation in 7_day_rate_of_change around July, Thanksgiving, and Christmas.

Cyclic Encoding: Time-series data can be problematic for machine learning because it is hard to encode time-dependent relationships.

Categorical variables such as weekday or season often require one-hot encoding which increase the feature space and disregard the relationship between categories (ex. Tuesday is closer to Monday than Saturday).

Similarly, directly encoded variables like hour or day of the month miss the cyclic nature of time. For example, hour 23 and hour 0 are very separate in direct encoding, while 11 pm and 12 am are very close in real time. Similarly, day 31 and day 1 encoded directly are far apart, but the 31st of the previous month and the 1st of the following month are very close.

Both issues are avoided by encoding cyclical variables to represent the cycle of time. These cyclical variables use sine or cosine functions with a period aligned with the cycle. For example, each function captures the daily 24-hour cycle, weekly 7-day cycle, variable monthly cycle, and annual cycle.

Figure 7 demonstrates these cyclic encodings which are individually encoded from 0 to 1 but shifted for the visualization.

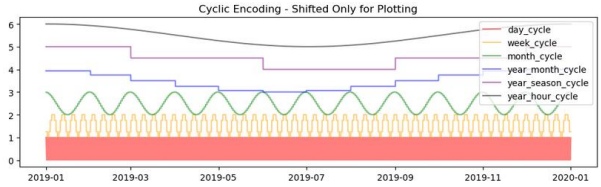

Figure 7: Cyclic Encodings

Final Data Modifications: The rolling average and rate of change features require past data to be calculated. Since our data only extends to the start of 2019, this means the first two weeks of data have null values in the 14-day rolling average. As such, the first two weeks are dropped to avoid negative effects on the anomaly detection algorithms. Ideally data would be used continuously across years unlike in this project's implementation.

Finally, the data will be normalized or standardized depending on the algorithm [10]. DBSCAN and OCSVM will use standardized variables with mean 0, while the autoencoder will use min-max normalized data from 0 to 1.

MODELING

Each model was implemented and tuned to achieve between 10-15% anomalies in the data. 10-15% anomalies was determined experimentally, judging how the models performed on both hourly and daily aggregated data.

OneClassSVM was implemented with the tuned γ hyperparameter. Recall γ represents the influence of each point on the decision boundary. By reducing γ, each point has less influence on the decision boundary and the model better generalizes the dataset. A sensitivity study on γ revealed very small values of γ reduced the anomaly percentage, though some models showed instability in this trend. The final hourly and daily models used γ values of 1E-9 and 1.5E-7 respectively, far smaller than the default value of the reciprocal of the # of features.

DBSCAN was implemented with a tuned ε hyperparameter. Recall ε represents the maximum distance between points in the same cluster. As such, increasing ε allows for less dense clusters, allowing more points to be classified in clusters rather than as outliers. ε was increased from the default 0.5 to 0.85 and 1.85 for the hourly and daily aggregates respectively. *min_samples* (the minimum number of samples to consider a cluster) can also be tuned to adjust the anomaly percentage. However, it is challenging to vary both ε and *min_samples* to achieve a desired anomaly % since the problem has too many degrees of freedom. While *min_samples* was explored at the chosen ε values, it could be examined in detail in additional studies.

The autoencoder was encoded using 3 dense layers (Input>64>32>latent dimension) then decoded using 3 dense layers and a sigmoid output layer (latent dimension>32>64>Input dimensions). The latent dimension at the bottleneck of the autoencoder as chosen based on the number of PCA components to capture 90% of variance, which was 6 and 4 for the hourly-aggregate and daily-aggregate data respectively The latent

dimension could be tuned as a hyperparameter, but was not explored in this study. The autoencoder is trained to reproduce its input by reducing reconstruction loss. Anomalous points will be harder to reproduce and therefore have a higher reconstruction loss. A threshold reconstruction loss value serves as a hyperparameter to label points as anomalies. The final autoencoder reconstruction losses are found in Figure 8 with the chosen threshold reconstruction loss values.

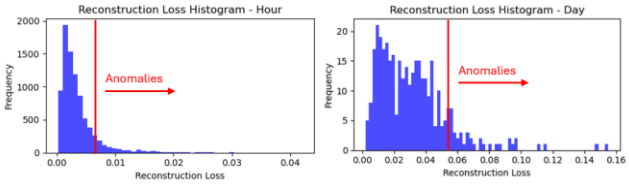

Figure 8: Reconstruction Loss from autoencoder with anomalies labeled above chosen threshold loss.

EVALUATION

The algorithms were evaluated in three ways:

1) Evaluate the ability to identify statistical outliers
2) Visually assess anomalies on volume vs. time plots
3) Visually assess anomalies on PCA/t-SNE plots.

Statistical Outliers: When an anomaly detection dataset includes labeled entries, these approaches could be evaluated using accuracy, recall, precision, and F1 score metrics. However, automatic anomaly detection is often an unsupervised or semi-supervised process from a raw data stream. This project will approach evaluation by first labeling statistical outliers occurring outside of the 1.5 IQR range. Hourly-aggregates mark outliers when grouped by hour of the day, while daily-aggregates mark outliers when grouped by day of the week.

Table 1 indicates what percentage of the 80 hour-outliers and 13 day-outliers each algorithm labeled as an anomaly. Notably, all algorithms identified more daily-aggregated outliers than hourly outliers. OCSVM also performed better on the hour-aggregated outliers.

Table 1: % of statistical outliers identified by each method.

	Model	Hour Outlier %	Day Outlier %
0	OCSVM	70.27	83.33
1	DBSCAM	14.86	91.67
2	Autoencoder	14.86	58.33

Volume vs. Time Plots: Since there are many types of outliers (global, contextual, collective), it is valuable to look at the labeled anomalies on the original dataset. This helps to understand how each method approaches the anomaly detection. The figures follow a similar format, plotting hourly-aggregates for the entire year, then for the month of December, then plotting the daily-aggregate anomalies.

OCSVM, as shown in Figure 9, appears to label the hourly anomalies by labeling entire days as anomalous. This is reflected in the plot as 'vertical' groupings, for example labeling the hourly points in the days surrounding Christmas as anomalous.

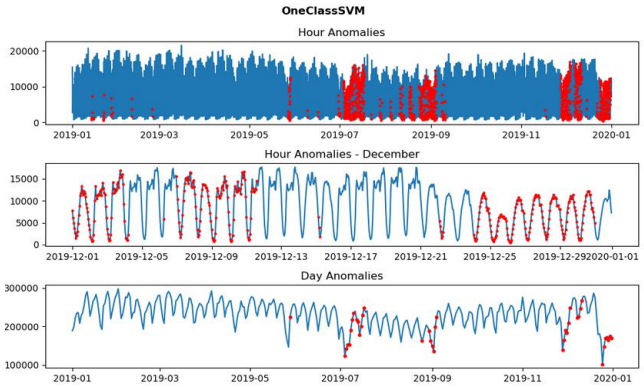

Figure 9: OCSVM Anomaly Visualization

DBSCAN is shown in Figure 10. The daily anomaly labeling appears similar to OCSVM labeling. However, DBSCAN is much harder to interpret for hour-aggregate anomalies. Notably DBSCAN picks up anomalies on the day of Christmas, but does not consider the anomalies on the surrounding days. This may occur if collective anomalies are labeled as a distinct 'normal' cluster.

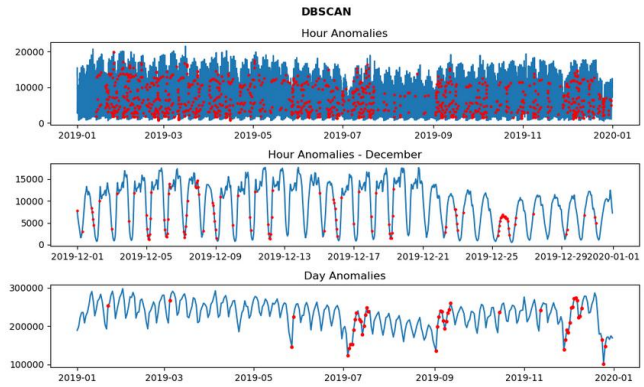

Figure 10: DBSCAN Anomaly Visualization

The autoencoder, presented in Figure 11, performs comparably in labeling daily-aggregate data. On hourly-aggregate data, it performs similar to DBSCAN but better captures some daily-anomaly trends like OCSVM such as surrounding Christmas. In the autoencoder anomalies, it appears a specific hour is commonly labeled as an anomaly. This specific hour appears as a horizontal group in the volume plot. This may be due to high variance hours from 7-9 AM and 6-10 PM.

Across all three plots, it appears the algorithms perform similarly on the day-aggregate anomaly detection, identifying variations like Thanksgiving and Christmas consistently. However, DBSCAN and the Autoencoder are harder to interpret for hourly data without clearly identifiable hourly anomalies while OCSVM appears to label entire days in the hour-aggregate data as anomalous.

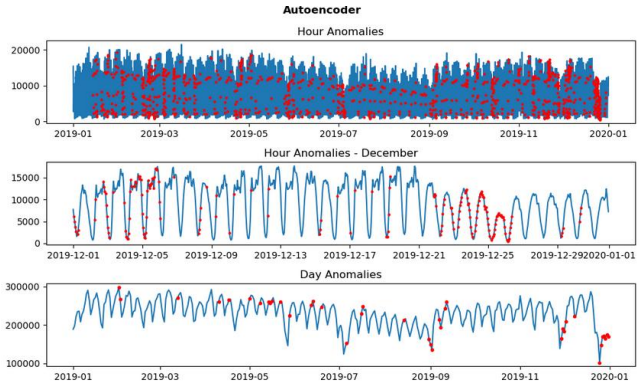

Figure 11: Autoencoder Anomaly Visualization

PCA and t-SNE Visualization: The hourly-aggregate and daily-aggregate anomalies can be hard to interpret, as demonstrated with DBSCAN and autoencoder hourly anomaly data. An alternative interpretation approach is to use PCA and t-SNE to decompose the feature space, allowing us to plot more complex feature relationships across 2 dimensions. These plots visualize similarities between points, such as those identified by clustering algorithms or grouped using decision boundaries developed by SVM methods.

Consider the PCA and t-SNE plots of the standardized data in Figure 12. The plots reveal possible groupings, outliers, and reveal the distribution of data when colored by month and by weekday.

The daily-aggregate plots have more distinct outliers than the hour-aggregate which may explain the similar performance on day-aggregate data.

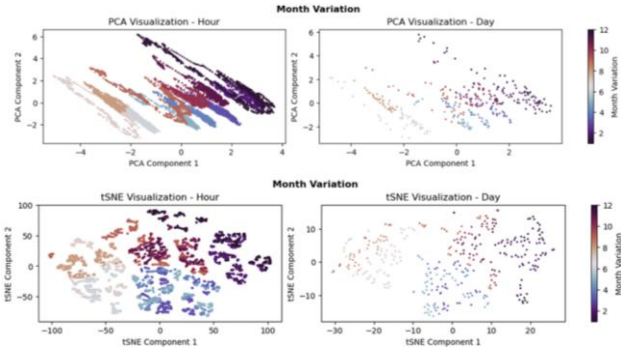

Figure 12: PCA and t-SNE visualization, colored by monthly variation. Highlights seasonal variation of points.

The following plots highlight the anomalies for the three algorithms, revealing their differences in approach.

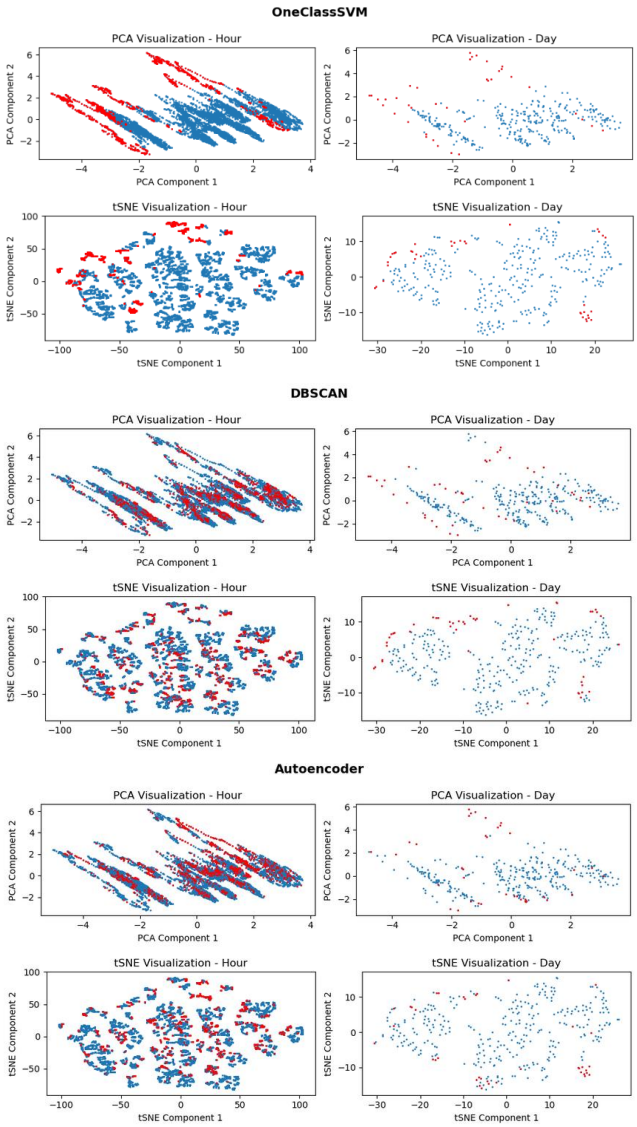

Figure 13: PCA and t-SNE visualizations of anomalies predicted by each algorithm approach.

These plots clarify the variation between approaches, for example OCSVM more clearly labeling groups in the hour-aggregate data as visualized with the t-SNE plot. The day-aggregate plots reveal the similarities among the daily-aggregate data with distinct outliers while the hourly-aggregate data has dramatic variation between the approaches.

FEATURE IMPORTANCE

One approach to understand these differences in algorithm anomalies is to perform supervised classification on each algorithm's anomalies and extract feature importance. This process was performed using Random Forest classification and plotted in Figure 14.

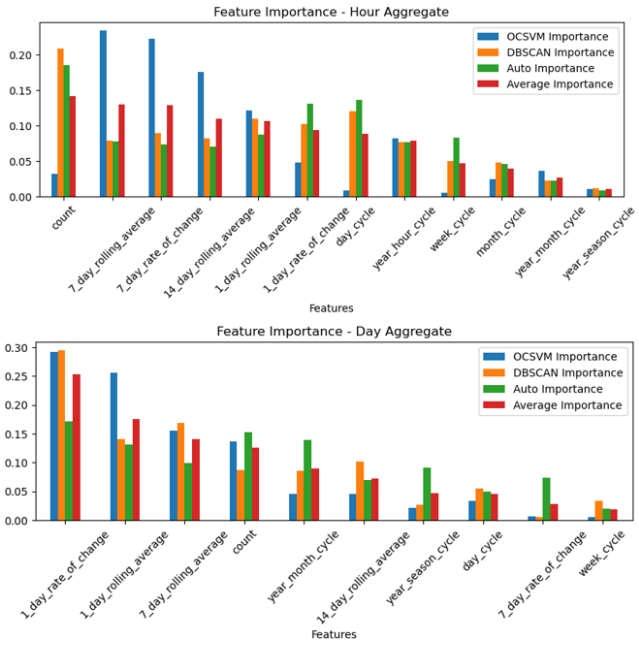

Figure 14: Feature Importance of each algorithm approach, for Hourly-Aggregate and Daily-Aggregate datasets.

Notably, the models use similar features in daily-aggregate anomaly detection, especially *1_day_rate_of_change* and rolling average features. However, the hourly-aggregate features are dramatically different, showing DBSCAN and autoencoder similarities using *count,* likely explaining their unexpected horizontal trends in the volume vs. time plots.

Evaluation Analysis: It appears the algorithms performed similarly on the easier-to-separate daily-aggregate data. The models relied on similar features and therefore identified similar anomalies. The variation between models may be explained by imprecise tuning, but all models detect the anticipated anomalies (July, Thanksgiving, Christmas).

The algorithms showed much more variation in the hourly-aggregate data, likely because the data is harder to parse into distinct anomalies as indicated by PCA and t-SNE plots. Of the models, OCSVM performance was easiest to interpret, effectively recreating the daily anomaly labeling on the hourly data. Both DBSCAN and the Autoencoder appear to rely on *count* variation, which may be beneficial to identify individual hour anomalies but are much harder to interpret on a larger scale.

These differences between methods are reflected in feature importance analysis. All three models used similar features during daily-aggregate anomaly detection, but OCSVM used 7-day and 1-day metrics more aggressively than DBSCAN and autoencoders in hourly-aggregate anomaly detection.

Given the choice of the three models, OCSVM performs most closely to expectations and is easiest to interpret its results.

DISCUSSION

Timeline: The Data Mining Project course was taken as an intensive over a few days based on the following schedule:

- Data Cleaning 1 hour
- Data Pre-Processing 3 hours
- Exploratory Data Analysis 3 hours
- Additional Processing 1 hour
- **Project Checkpoint** 2 hours – Total 10 hours
- Algorithm Implementation 2 hours
- Result Evaluation 4 hours
- Feature Importance 2 hours
- **Final Project** 2 hours – Total 10 hours

The project was projected to finish in two long days following the proposal. However, the project went over the timeline by two days due to several unexpected challenges and productivity below expectations.

Challenges: Surprisingly, many of the original challenges (building PCA/t-SNE visualizations, long training times, evaluating feature importance) were not problematic. The challenges which delayed the project were unexpected, and therefore hard to mitigate.

One time intensive challenge was implementing SQL to pre-process the large dataset. The report writing and slides took significantly longer than expected, including plot formatting and additional steps to neatly label data. In the effort of good presentation, the original code was re-written to utilize functions, improve organization, and correct unexpected anomaly indexing issues. Model hyperparameter tuning was also much scarier when performed initially, which is why the simpler daily-aggregate dataset was implemented outside of the original scope.

Future Risk Mitigation: For future similar projects, it is strongly recommended to simplify the problem, then simplify it even further (for example using the daily-aggregates instead of hour-aggregates to being). This incremental development eases processing of more complex data. It is also recommended to spend more time during planning, for example anticipating which plots and calculations will be performed repeatedly and investing in those functions.

CONCLUSION

Anomaly detection applies to many uses and industries, but can be challenging to implement issues without labeled data. The proliferation of time-series data also creates many opportunities to detect anomalies automatically from data streams. This report uses NYC Taxi volume data to compare three time-series anomaly detection methods: unsupervised density-based spatial clustering of applications with noise (DBSCAN), one-class classification using support vector machines (OCSVM), and a trained autoencoder using reconstruction loss to identify outliers. Analysis is performed on hourly-aggregated and daily-aggregated data to consider varying complexity in data.

The original 84.4M individual trip records were processed in SQL before being explored in Python. Exploratory analysis revealed

some expected trends, for example higher taxi volume during the day than in the late night, lower volume on weekends, and significant seasonal variation. Global outliers such as drops in traffic during Thanksgiving and Christmas are also apparent, as well as collective outliers such as the week leading to Christmas.

Feature engineering was required to capture the time dependencies of the model for the machine learning algorithms. These features included rolling averages which smooth rapid change, rate of change which exaggerate rapid change, and cyclic encoding which captures the cyclic nature of time (daily, weekly, monthly, and annual cycles).

The implemented models had their parameters hypertuned to achieve 10-15% anomaly, with their resulting anomalies visualized using volume vs. time data and PCA/t-SNE visualization.

The evaluation of these visualizations indicated OCSVM identified hourly-aggregated anomalies over entire days, while DBSCAN and autoencoder methods identified anomalies within specific hour ranges. OCSVM, as a result, was much easier to interpret. All three anomalies performed similarly on daily-aggregated anomalies, suggesting that the underlying separability of the dataset can affect each algorithm's performance. The differences in model performance on hourly-aggregated data were understood by assessing feature importance. Feature importance was generated from Random Forest classification on each model's anomalies. While OCSVM relied on daily rolling average and rate of change variables (explaining its ability to mark entire days as anomalous), DBSCAN and the autoencoder relied on direct count metrics which with more noise. This may explain their harder-to-interpret results.

The models showed similar feature importance on daily-aggregated data, relying primarily on rate of change and rolling averages, leading to similar anomaly classification.

While the algorithms performed similarly on daily-aggregated data, OCSVM was the easiest to interpret on the more complex hourly-aggregated data and reflected user expectations for anomaly detection.

***Key Findings**:*

- Pre-visualizing data using PCA or SVM helps identify the ease of detecting anomalies using visible outliers.
- Algorithm performance very dependent on hyperparameter tuning, requiring subjective anomaly evaluation.
- Algorithms performed similarly on simpler day-aggregated data, reflecting easier to separate dataset.
- OCSVM was easiest to interpret on more challenging hour-aggregated data, labeling days of anomalies rather individual hours.
- Feature importance analysis revealed similarities across day-aggregated algorithm performance, and stark differences between OCSVM and the DBSCAN / autoencoder methods.
- Rolling average and rate of change features appeared most importance to anomaly detection.

***Future Work**:* This algorithm comparison can be easily expanded to consider additional datasets and methods. For example, labeled anomalous data such as the Numenta Anomaly Benchmark could to more easily quantify algorithm performance. Other features from the NYC Taxi dataset such as geolocation could further inform anomaly labeling. Additional time-series features could be engineered, such as binary labeling holidays or using Seasonal Trend decomposition using LOESS (STL). While these algorithms are general for anomaly detection, they could be compared to time-series specific approaches, such as long short-term memory (LSTM) recurrent neural networks (RNN).

REFERENCES

[1] Chatterjee, A., & Ahmed, B. S. (2022). IoT anomaly detection methods and applications: A survey. *Internet of Things*, *19*, DOI: https://doi.org/10.1016/j.iot.2022.100568.

[2] scikit-learn. Comparing anomaly detection algorithms for outlier detection on toy datasets. https://scikit-learn.org/stable/auto_examples/miscellaneous/plot_anomaly_comparison.html

[3] Brownlee, J. (2020) One-Class Classification Algorithms for Imbalanced Datasets. *Machine Learning Mastery*. https://machinelearningmastery.com/one-class-classification-algorithms/

[4] City of New York. About TLC. https://www.nyc.gov/site/tlc/about/about-tlc.page

[5] Taxi and Limousine Commission (TLC). (2023). 2019 Yellow Taxi Trip Data. *NYC OpenData* https://data.cityofnewyork.us/Transportation/2019-Yellow-Taxi-Trip-Data/2upf-qytp

[6] scikit-learn. OneClassSVM. https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html

[7] scikit-learn. DBSCAN. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

[8] Chollet, F. (2016). Building Autoencoders in Keras. *Keras.* https://blog.keras.io/building-autoencoders-in-keras.html

[9] Pavithrasv. (2020). Timeseries anomaly detection using an Autoencoder. *Keras.* https://keras.io/examples/timeseries/timeseries_anomaly_detection/

[10] Shalbu, S. (2024) Normalization vs. Standardization: How to Know the Difference. *DataCamp*. https://www.datacamp.com/tutorial/normalization-vs-standardization