Fundamental of Data Science
Anomaly & Outlier Detection - A Case Study

Nguyen Quang Huy - 22BI13195
Nguyen Tuan Khai - 22BI13202
Nguyen Hai Dang - 22BI13073
Le Linh Long - 22BI13262
Pham Thai Son - 22BI13397

University of Science and Technology of Hanoi

Abstract

Anomalies can be understood as deviations that supersede the prevailing patterns in a dataset. The discernment of anomalies hold paramount significance in various domains such as cybersecurity, machine learning, finance, and healthcare. In this study, the focal point lies on detecting and highlighting anomalies within a given dataset, and we delineated our research into 2 separate segments. First, we explore the employment of clustering-based methodologies of outlier detection on multivariate datasets; and following that we research the application of alternative techniques suited for outlier detection in univariate time series. Our endeavor delves profoundly into the evaluation of the effectiveness of algorithms and techniques developed for anomaly identification within datasets. Specifically, we scrutinize the utilization of methodologies including Isolation Forest, Local Outlier Factor, One-Class SVM, among others.

1 Introduction

Anomaly detection is the identification of observations, events or data points that deviate from what is usual, standard or expected, making them inconsistent with the rest of the dataset [1]. In this regard, anomalies are often indicative of significant events or changes which would often require context-specific understanding. A subdomain of anomaly detection involves the presence of outliers, which are traditionally identified from a statistical point of view as lie outside the expected range of data, signifying irregularities in the pattern. Since these terms are of similar meaning and are often used interchangeably in data science literature [2], our report will also use them correspondently in different contexts. These practices has been a long standing field of research in the domains of statistics and machine learning, where researchers and analysts would leverage statistical tools, algorithms, and more recently artificial intelligence, in identifying unanticipated changes in a dataset's normal behaviour. Anomalous data can signify critical problems like compromised data infrastructures or security breach, making their identification a pivotal necessity in various fields. The inherent challenge of the task has to do with the nature of outliers, as they are often rare and require a thorough understanding and capture of the normal data's distribution and expected behaviours. Particularly, in the field of data science, anomaly detection can be crucial to the training process for machine learning and deep learning models, where a single outlier can lead to significant skew in the overall distribution, thus prompting the need for a way to deal with them.

Traditionally, anomaly and outlier detection was based on discretionary techniques that heavily relied on domain expertise and subjective judgment. The process was conducted manually, with analysts resorting to visual inspection or simple statistical thresholds, drawing heavily on their understanding of the dataset. Although this methodology sufficed in some contexts, it was neither reliable nor replicable, and proved especially difficult to scale as data volumes grew. Traditional methods were susceptible to human fallibility and objectivity, and thus consistency when applied across diverse datasets.

With the emergence of data science, however, it has become more systematic and data-oriented. Modern techniques employ machine learning algorithms and statistical models to automatically detect outliers, providing much better accuracy and scalability. We now have data science tools which can help find intricate patterns in huge data sets but can also detect subtle variations from normality which may never be discovered through traditional methods, making this approach more robust for anomaly detection.

Isolation Forest

Developed by Tony Liu et al. in 2008 [3], isolation forest detects anomalies in data using binary trees, based on the assumption that the outliers are few in number and different from other data. Given the premise that anomalous data points are easier to separate, it recursively generates partitions on the sample by randomly selecting an attribute and then randomly selecting a split value between the minimum and maximum values allowed for that attribute. The recursive partitioning can be represented as an Isolation Tree, as follows:

Algorithm 1 Isolation Tree

1: Let $X = \{x_1, \ldots, x_n\}$ be a set of d-dimensional points and $X' \subset X$
2: For each node T in the Tree, T is either an external-node with no child, or an internal-node with one "test" and exactly two child nodes (T_l and T_r)
3: A test at node T consists of an attribute q and a split value p such that the test $q < p$ determines the traversal of a data point to either T_l or T_r.
4: The algorithm recursively divides X' by randomly selecting an attribute q and a split value p, until either

1. the node has only one instance, or
2. all data at the node have the same values.

Anomaly detection with Isolation Forest is done by first using the training dataset to build some number of iTrees, then for each data point in the test set, it is passed through all the iTrees, counting the path length for each tree. An "anomaly score" [4] is assigned to the instance, then if the score is greater than a predefined threshold, it is labelled as an anomaly.

Local Outlier Factor

Local Outlier Factor (LOF) is another unsupervised ML algorithm that identifies outliers with consideration of the local neighborhood as opposed to the data distribution in its entirety. Introduced by Breunig et al. [5], LOF is fundamentally a density-based technique that uses the nearest neighbor search to identify the anomalous points. For instance, when using the LOF technique, neighbors of each point are identified and compared against the density of the neighboring points, as follows:

Algorithm 2 Local Outlier Factor

1: Calculate distance between point P and all the given points using a defined distance function (e.g Euclidean, Manhattan, etc.).
2: Find the k (k-nearest neighbor) closest point. For example, if K = 3, find the third nearest neighbor's distance.
3: Find local reachability [6] density using the following equation:

$$\mathrm{lrd}_k(O) = \frac{\|N_k(O)\|}{\sum_{O' \in N_k(O)} \mathrm{reachdist}_k(O' \leftarrow O)}$$

where $N_k(O)$ refers to the number of neighbors, and reachable distance can be calculated as:

$$\mathrm{reachdist}_k(O' \leftarrow O) = \max\{\mathrm{dist}_k(O), \mathrm{dist}(O, O')\}$$

4: Lastly, we calculate the outlier score as follows:

$$\mathrm{LOF}_k(O) = \frac{\sum_{O' \in N_k(O)} \frac{\mathrm{lrd}_k(O')}{\mathrm{lrd}_k(O)}}{\|N_k(O)\|}$$

Mahalanobis distance

The Mahalanobis distance is a measure of the distance between a point P and a distribution D, introduced by P. C. Mahalanobis in 1936 [7]. Based on the presupposition of multivariate Gaussian distribution within the dataset, the distance of an observation x_i to the mode of the distribution is calculated as

$$d_{(\mu,\Sigma)}(x_i)^2 = (x_i - \mu)^T \Sigma^{-1} (x_i - \mu)$$

where μ and Σ are the mean and the covariance of the underlying Gaussian distributions. However, in most cases, only an estimate of the covariance within the dataset is calculated, leading to some subjectivity of the overall method. One way to estimate the covariance matrix Σ is via maximum likelihood estimate, given by:

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu})(x_i - \hat{\mu})^T$$

where $\hat{\mu}$ is the sample mean and n is the number of observations. This estimator is sensitive to outliers, which can significantly affect the covariance estimate and consequently the Mahalanobis distance calculated from it. Hence, a more robust method for estimating the covariance matrix that is designed to be less sensitive to outliers was devised. Introduced by P.J.Rousseuw in 1984 [8], the Mininum Covariance Determinant estimator works to estimate the covariance matrix as follows:

Algorithm 3 Minimum Covariance Determinant estimator

1: Choose a subset of h observations from the total n observations, where h is typically set to be greater than $\frac{n+p+1}{2}$ (with p being the number of dimensions).

2: For the selected subset, calculate the sample mean $\hat{\mu}$ and the sample covariance matrix S:

$$\hat{\mu} = \frac{1}{h}\sum_{i=1}^{h} x_i$$

$$S = \frac{1}{h-1}\sum_{i=1}^{h}(x_i - \hat{\mu})(x_i - \hat{\mu})^T$$

3: The goal is to find the subset that minimizes the determinant of the covariance matrix. The covariance matrix $\hat{\Sigma}_{\text{MCD}}$ is taken from the subset that yields the smallest determinant and used to calculate the Mahalanobis distance.

$$\hat{\Sigma}_{\text{MCD}} = \arg\min_{S} |\det(S)|$$

One-class SVM

One-class Support Vector Machines (SVMs) are a type of outlier detection method based off of the core principles of SVM. The objective behind using one-class SVM is to identify instances that deviate significantly from the norm. Unlike other traditional Machine Learning models, one-class SVM is not used to perform binary or multiclass classification tasks but to detect outliers or novelties within the dataset. The first formulation of one-class SVM, introduced by Schölkopf et al. [9] involves using a hyperplane (a plane in n-dimensions) for the decision boundary. Later iterations introduced by Tax and Duin saw the implementation of a hypersphere for the decision boundary, but for our purposes we will focus on the first. The primary goal of one-class SVM model is to estimate a function f that is positive on a subset of the input space (the normal data) and negative on its complement (the outliers). To this end, we opted for one-class SVM with Stochastic Gradient Descent with the application of the Nystroem method, which works as following: Let ϕ be the feature mapping function associated with the kernel of the SVM. Given a set of landmark points L = l_1, l_2, ..., l_m chosen by the Nystroem method, the approximate kernel matrix $\hat{K}$ is calculated as:

$$\hat{K} = \phi(X)\phi(L)^T$$

where X are the original data points. During training, the algorithm minimizes the following objective function:

$$\frac{1}{2}||w||^2 = \frac{1}{vn}\sum_{i=1}^{n} max(0, -<w, \phi(x)>)$$

where w is the weight vector, v is the parameter controlling the proportion of outliers (initialized as nu), ρ is the offset parameter, and n is the number of data points. After training, the decision function for predicting new samples involves computing the decision scores as the dot product between the weight vector w and the feature representation of the test samples with respect to the landmark points:

$$f(x) = <w, \phi(x_{test})\phi(L)^T> - \rho$$

Kernel Density Estimation and interpolation

We propose a spatial approach for detecting anomalies in time series using Kernel Density Estimation (KDE) and interpolation. Spatial methods evaluate the likelihood of a data point based on its location

within that parameter space, aiming to characterize the distribution of data in said space and outline the anomalies. Common techniques for spatial anomaly detection include histogram and Kernel Density Estimation (KDE). By segmenting the data range into bins and constructing a histogram, we can quantify the relative density of the data points. We begin by using the Fast Fourier Transform (FFT) method, similar to the approach used in seasonality analysis, to quickly approximate the density by reverse transforming the bins into a signal. Following that, we use the kernel density function with Epanechnikov kernel, which is mathematically represented as:

$$K(u) = \begin{cases} \frac{3}{4}(1-u^2) & \text{if } |u| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K(u)$$

where u represents the standardized distance between a point x where the density is being estimated and an observed point x_i, calculated as $\frac{x-x_i}{h}$ with h being the bandwidth controlling the smoothness of the density estimate. The Epanechnikov kernel function $K(u)$, also known as the parabolic kernel, serve to assign weights to each data point based on its distance from the point x where the density is estimated. The kernel function determines how much influence each data point x_i has on the final density estimate, which is denoted as $\hat{f}(x)$. This results in a smooth curve that approximates the underlying distribution of the data, from which the anomalies can be discerned. Particularly, an interp1d function from scipy.interpolate is used to interpolate the standardized density values over the range of the original data [10]. From calculating the density value through the interpolation function, a threshold is placed where if the density value for an observation falls below a set point, it is deemed an anomaly.

Prophet

In forecasting time series for anomaly detection, one of the powerful tools we implement is Prophet, developed by Facebook. Based upon an additive decomposition, it models time series data as a combination of 3 main components as follows:

$$y(t) = g(t) + s(t) + h(t) + \epsilon(t)$$

where $y(t)$ is the forecasted value at time t, $g(t)$ is the trend component, $s(t)$ is the seasonal component, $h(t)$ is the holiday effect, and $\epsilon(t)$ is the error term representing noise in the data. For our applications, we assumed no prior knowledge of the data being forecasted, meaning not specifying the seasonality component of the time series and disregarding the occurrence of holidays. This leave us with a model based on a piecewise linear trend component, which can be expressed mathematically as:

$$g(t) = m(t) \cdot t + b(t)$$

where $m(t)$ is the slope and $b(t)$ is the intercept. The function $g(t)$ is defined as linear segments that connect at changepoints, where the underlying trend of the time series data changes. Prophet automatically detects these changepoints during the fitting process by specifying a large number of potential changepoints and applying a sparse prior on the magnitudes of the rate changes. The error term, denoted $\epsilon(t)$, is simply represents the difference between the observed values and the predicted values from the model.

$$\epsilon(t) = y(t) - \hat{y}(t)$$

where $y(t)$ is the actual observed value at time t and $\hat{y}(t)$ is the predicted value from the model.

2 Application

2.1 Clustering-based outlier detection

For the task of clustering based anomaly detection, we opted to go with the Yeast dataset from the UCI Machine Learning Repository [11]. By identifying proteins with atypical localization patterns, businesses in biotechnology or pharmaceuticals can pinpoint unique or rare protein behaviors that may

lead to novel drug targets or biomarker discoveries. Since its last update, this dataset has 10 features and 1484 instances, created with the purpose of predicting the Cellular Localization Sites of proteins.

Variable Name	Role	Type
Sequence_Name	ID	Categorical
mcg	Feature	Continuous
gvh	Feature	Continuous
alm	Feature	Continuous
mit	Feature	Continuous
erl	Feature	Continuous
pox	Feature	Continuous
vac	Feature	Continuous
nuc	Feature	Continuous
localization_site	Target	Categorical

Table 1: Variables Table in Yeast Dataset

As seen in table 1, 2 features Sequence_Name and localization_site are categorical, and thus unviable for our specific task and will be discarded. The remaining features range from 0 to 1, and to further understand the distribution we created a simple box plot and a histogram as follows:

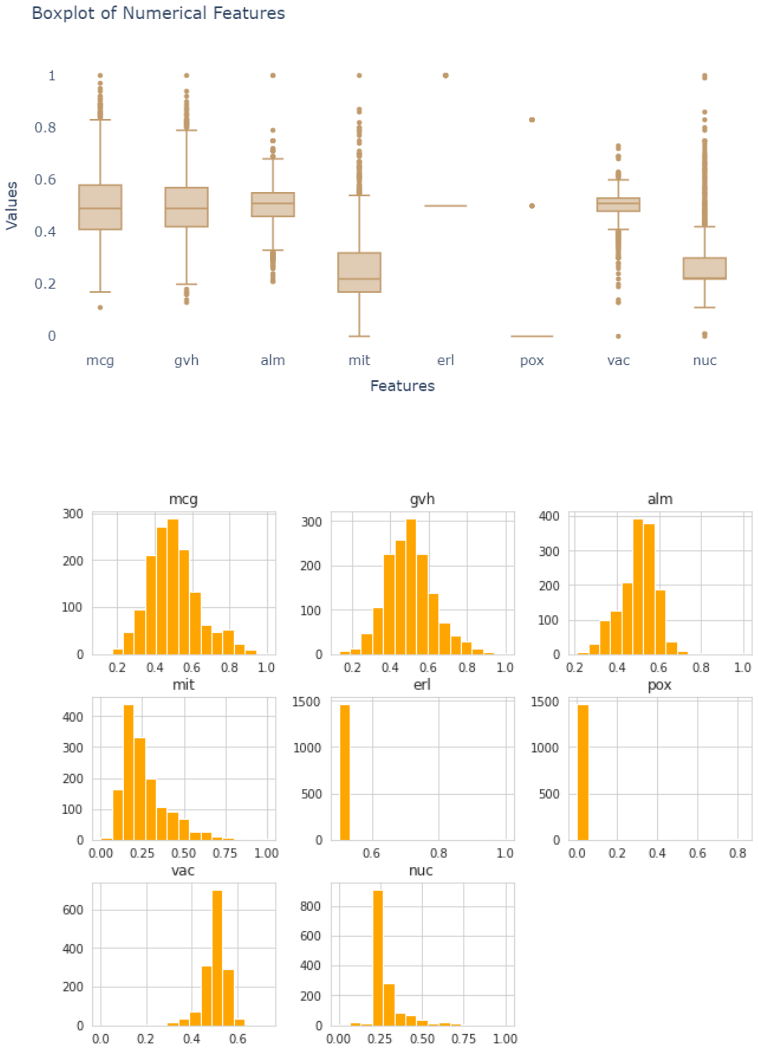


Figure 1: Feature-wise Histogram

As can be discerned from the box plot within the figure, the 2 features erl and pox are of little value to us, since they have virtually zero variance. The histogram in the figure also reflects this with these 2 features predominantly occupying the 0 and 0.5 bin. Later on, when we perform Principal Component analysis to reduce dimensions we will discard these 2 features.

Subsequently, we followed by testing for multivariate normality of our data, which is achieved through 1. testing for feature independence and 2. validating the hypothesis of feature-wise normality. The former is conducted using Pearson correlation analysis. We proceeded by employing the Shapiro-Wilk test [12] to assess the normality of the data distribution. This statistical test is used to evaluate whether the dataset conforms to a normal distribution, which is a fundamental assumption in some anomaly detection methods. This step will help us understand the results in the context of our dataset later on.

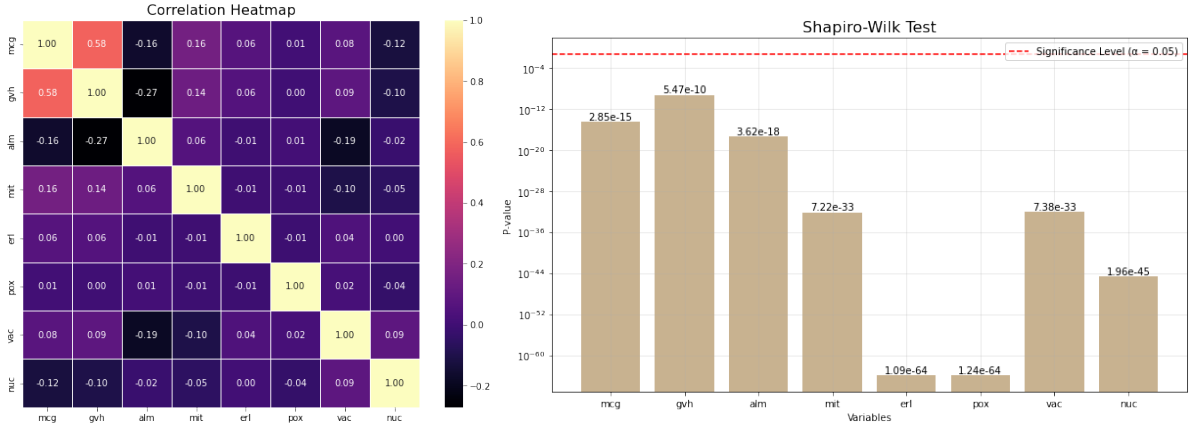


Figure 2: Testing for multivariate normality

Although the features demonstrate independence, the outcomes of the Shapiro-Wilk test reveal that each individual feature yields a p-value below the significance threshold of 0.05. Consequently, the null hypothesis positing feature-wise normality is rejected, and as a result our dataset does not demonstrate adherence to the presupposition of multivariate normal distribution. Lastly, we perform Principal Component analysis to reduce our data to 2 feature vectors to better accommodate our task. Following this, we were able to capture 51.21% of the explained variance, which while not reflecting the original dataset in its entirety we still effectively retained the intrinsic relationships between inliers and outliers in the original.

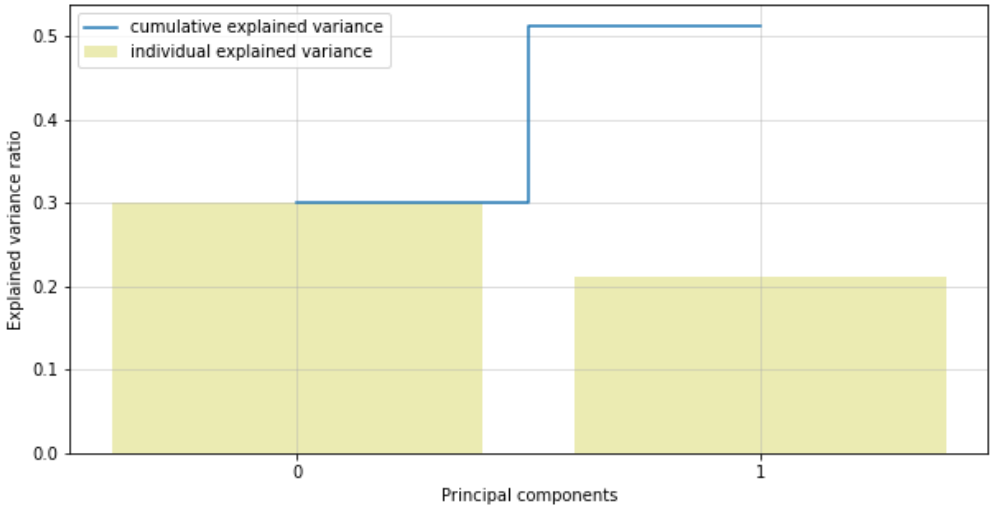


Figure 3: Cumulative Explain Variance Ratio following Principal Component Analysis

To set a baseline from antiquated outlier detection techniques, we implemented a simple z-score threshold in Python. It calculates the Z-score as a measure of a feature-wise mean and standard deviation. We then flagged data points with Z-scores of 1, 2, and 3 as possible outliers, setting different thresholds at these levels. For the implementation, we plotted results on scatter plots and marked outliers in red for each threshold. This approach is straightforward and interpretable for anomaly detection.

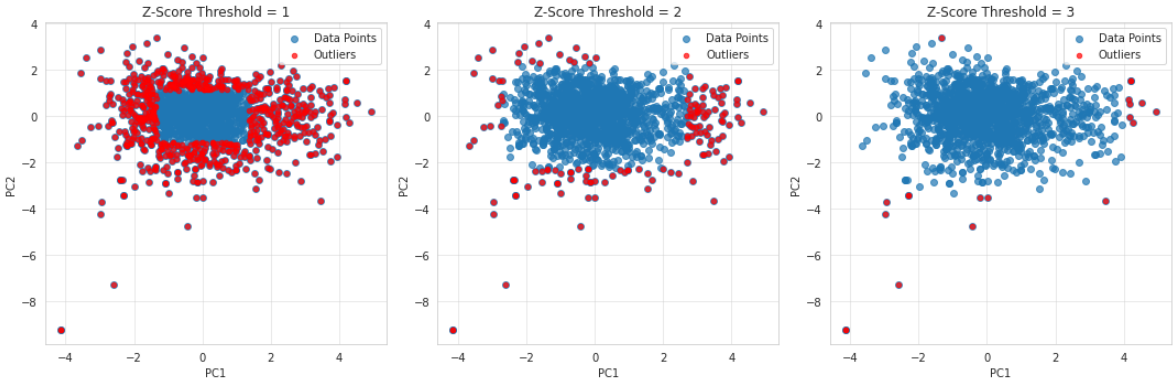


Figure 4: Z-score threshold to detect outliers

2.1.1 Local Outlier Factor

With the Local Outlier Factor method, we implemented our own simple structure based upon sklearn's Nearest Neighbors with a defined number of neighbors to consider when calculating LOF for a sample. We also define a specified contamination level representing the expected proportion of outliers in the dataset. After obtaining the LOF scores, the threshold for the model to label an instance as an outlier is set to be $100 * (1 - contamination)$ percentile of the distribution of LOF score, giving us a cutoff point above which points are considered outliers. Adjusting the threshold allow us to control the sensitivity of the model, but in our experimentation we assumed no prior knowledge of the data and chose an arbitrary figure. We arrived at the following results:

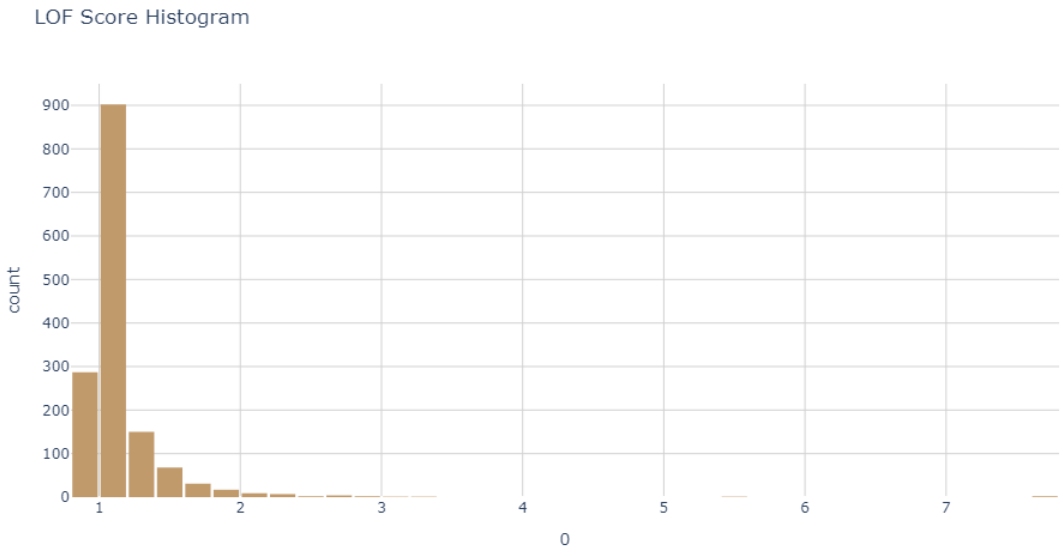


Figure 5

As seen in figure 5, this method evaluates the outlier score of each instance based on its relative

distance to the main cluster. We are thus able to detect many of the more prominent outliers in the dataset, but with outliers that are closer to the main cluster the model fail to effectively highlight them. The efficacy of outlier identification is influenced by the speculated percentage of outliers in the dataset, a factor contingent upon subjective interpretation.

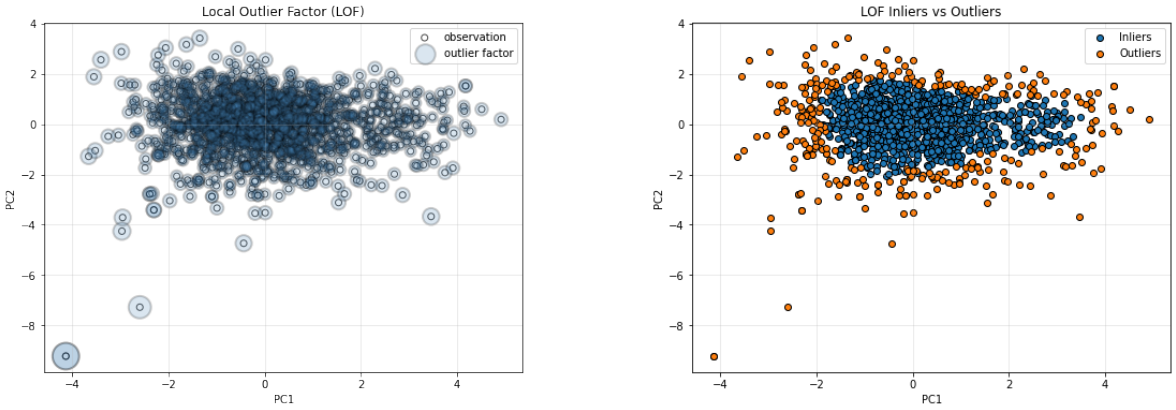


Figure 6: Local Outlier Factor output

2.1.2 Isolation Forest

We implement an instance of Isolation Forest with bootstrap sampling, mimicking the behaviour of Random Forests which introduces diversity to the training process. After which, we first visualize the path length decision boundary of the forest, and secondly we define the innermost level as the learned boundary that separates between inliers and outliers. Creating a meshgrid and visualizing the decision function, we have the following:

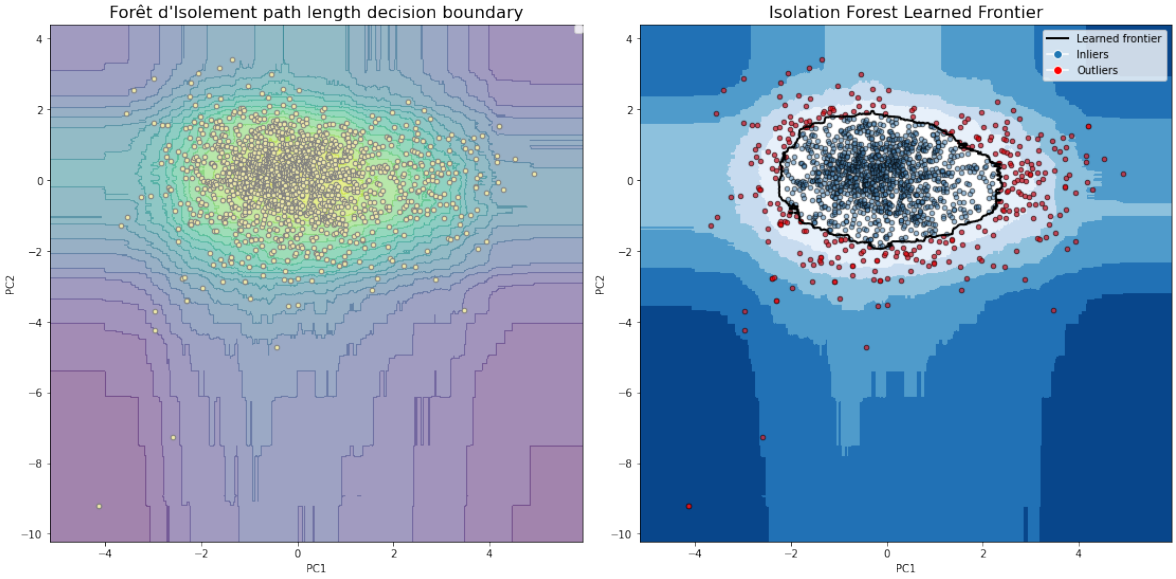


Figure 7: Isolation Forest output

To better understand the output of the model and how it is determining outliers, we need to delve deeper into the architecture of the first isolation tree in the forest. As previously established, isolation trees are binary tress constructed through recursive partitioning of the input space. The trees are build by randomly selective a feature and the choosing a split value for that feature withing the range

of the data. Outliers are expected to be isolated and lie in sparser regions of the feature space, and when scrutinizing the structure of the tree we expect that regular data points to have longer average path lengths from the root to the terminal node. This corresponds to the tree shown below, where each leaf node with a shorter path length is expected to be an outlier, with a higher anomaly score. This hearkens back to the output we obtained earlier, where regions with darker colors contain data points that have higher anomaly scores and are more likely to be outliers.

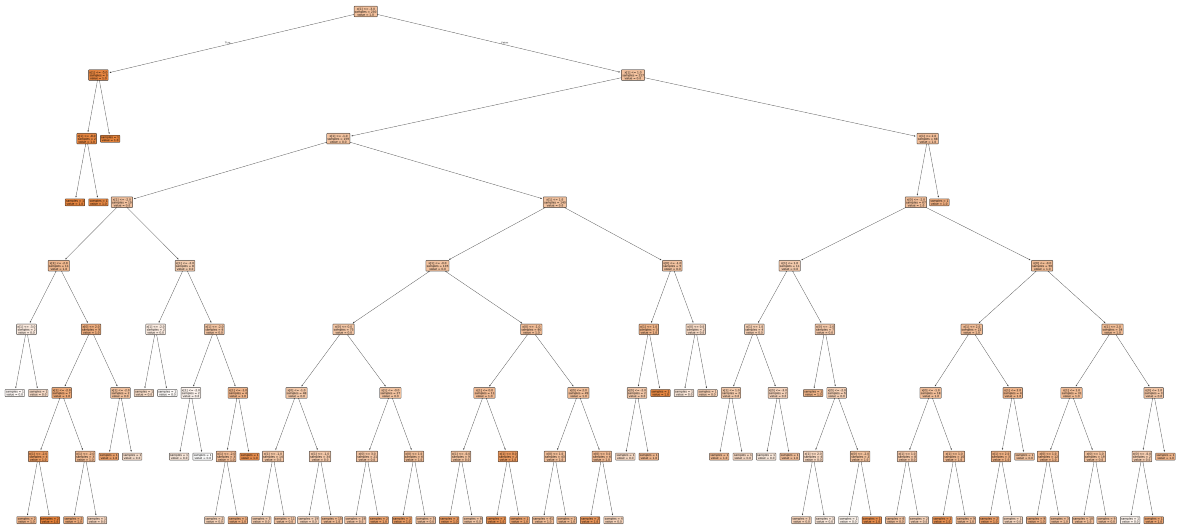


Figure 8: First tree in our Isolation Forest

2.1.3 One-class SVM with Stochastic Gradient Descent

For this section, we applied a machine learning pipeline to the original dataset using a combination of techniques. Initially, we utilized the Nystroem method with gamma value of 2 for transformation. Subsequently, a Stochastic Gradient Descent-based One-class SVM classifier was instantiated with nu value of 0.05 and tolerance of 1e-4. After fitting this pipeline to the dataset and obtaining predictions, to visualize the decision boundaries produced by the classifier, we generated a plot with a meshgrid for contouring. The decision boundary display was overlaid on the scatter plot of features, resulting in the following illustration:

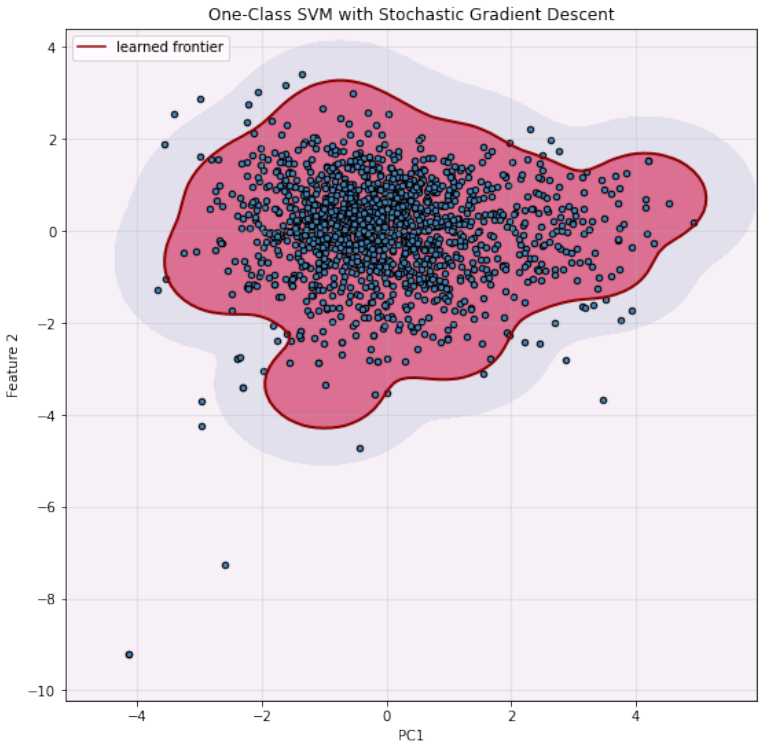


Figure 9: SGDOneClassSVM trained on the entire dataset

The main problem with this initial result is that the classifier one-class SVM is that the model has the unique characteristic of not being designed for binary detection, but rather for detecting a specific class. In this case, we train it on the entire dataset, so its decision boundary is acclimatized to both the inlier and outlier data points, leading to poor boundary formation. In order to model the inlier data's distribution and create a boundary to separate them, we will need to leverage the results from another method, in this case Isolation Forest, to label the inliers and outliers in the data.

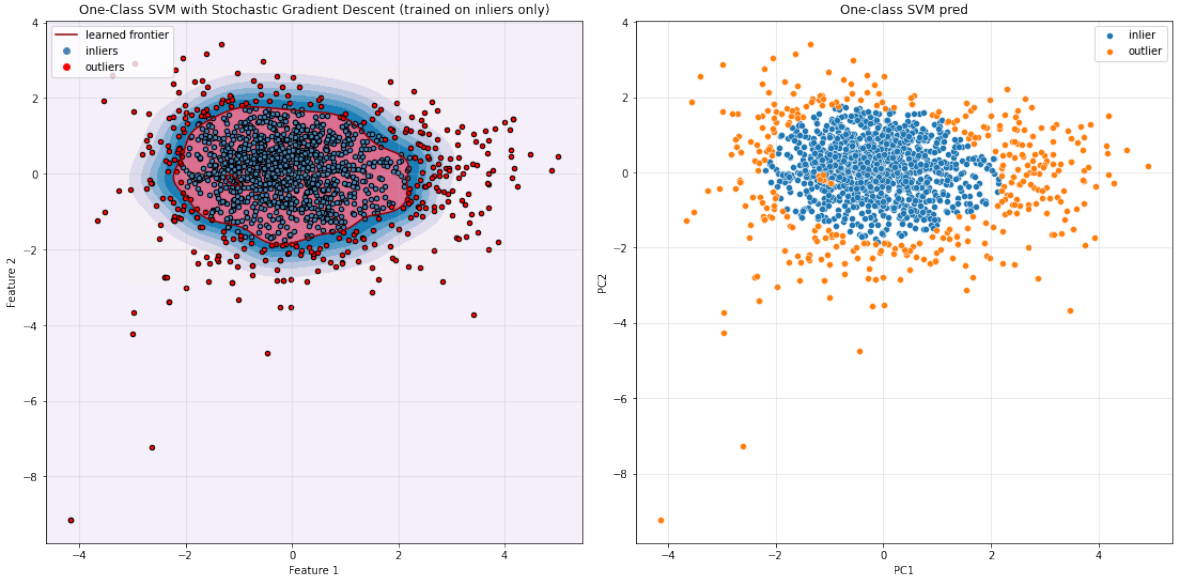


Figure 10: One-class SVM with Stochastic Gradient Descent output

By integrating the results from the Isolation Forest algorithm to identify and label inliers to train, a more refined and accurate representation of the dataset was achieved. This modification significantly enhanced the model's performance, enabling a more robust and discerning boundary formation that better captures the underlying structure of the inliers. Despite a minor misclassification within the main cluster, this outcome can still be deemed largely successful.

2.1.4 Mahalanobis Distance

The last individual method we seek to implement is to utilize Mahalanobis distance based on maximum likelihood estimate and minimum covariance determinant estimate respectively for the covariance matrix.

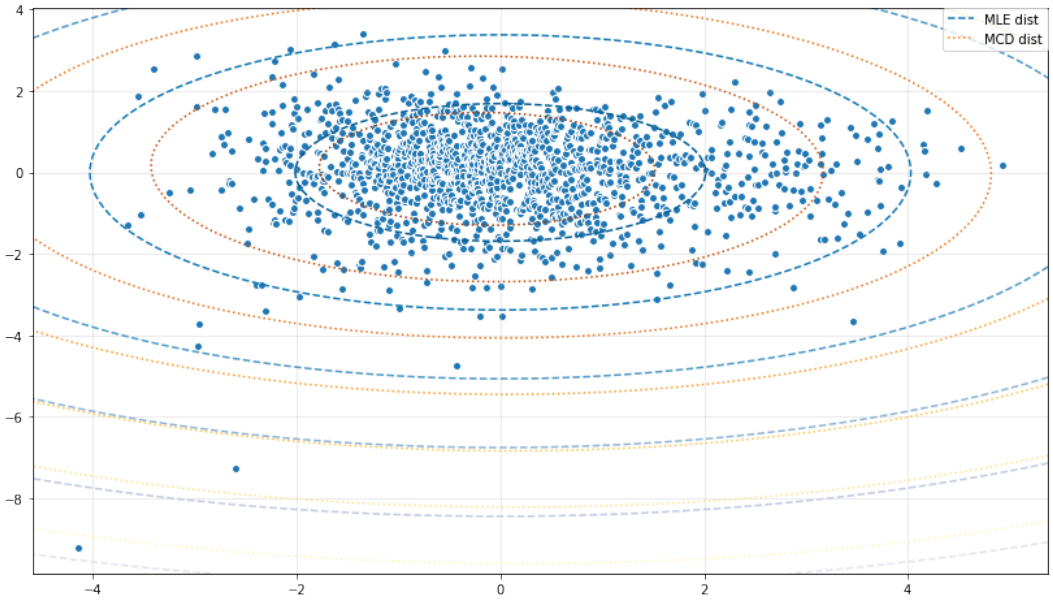
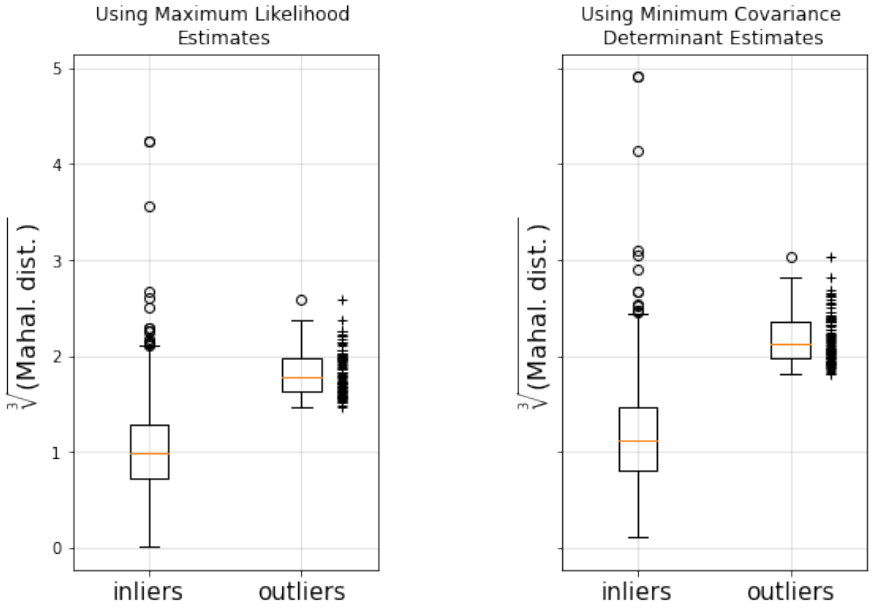


Figure 11: Mahalanobis Distance



Between using non-robust and robust estimates, we observe the following:

1. Non-robust: In the inliers distribution, the median distance is lower and it has a tighter interquartile range (IQR), but exhibits some outliers. For the outliers distribution the overall distribution of distance seems to be higher but still has some overlap with the inliers distribution, meaning some outlier may be misclassified.

2. Robust: Between inliers and outliers distribution of distance is a more notable difference, meaning we are better able to distinguish the 2. This is the superior method in this as well as most cases.

That being said, in detecting outliers using Mahalanobis distance with an elliptical boundary, the main presupposition is that our data follows a multivariate Gaussian distribution, meaning all feature vectors are independent of one another and follow a normal distribution. This is not the case for our dataset since no feature is normally distributed, leading to compromised results. This method is ineffective for this particular dataset.

2.1.5 Plurality Voting Ensemble

Finally, to bolster the reliability of our findings, we implemented a plurality voting scheme to aggregate predictions from multiple models we've implemented before. This approach involves combining the outputs of individual classifiers to arrive at a final decision, thereby enhancing the overall confidence in the results. By leveraging this ensemble method we aimed to mitigate individual model biases and uncertainties, reinforcing our analytical outcomes. We opted to forego the Mahalanobis distance method since it is not particularly effective for our dataset, thus our remaining framework is as follows:

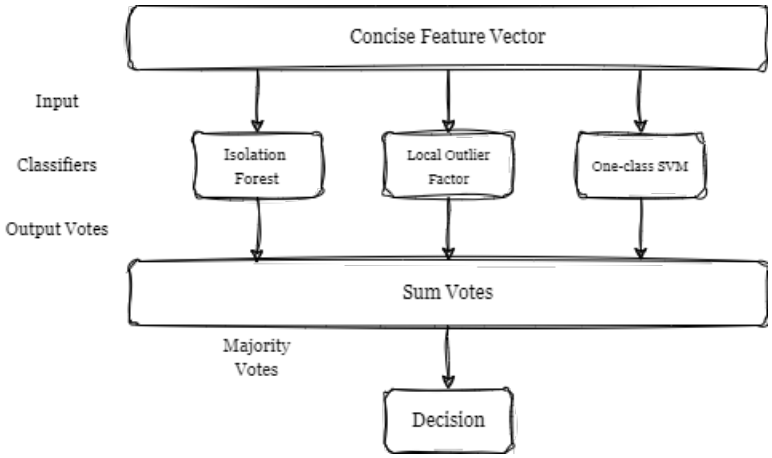


Figure 12: Plurality Voting Ensemble Framework

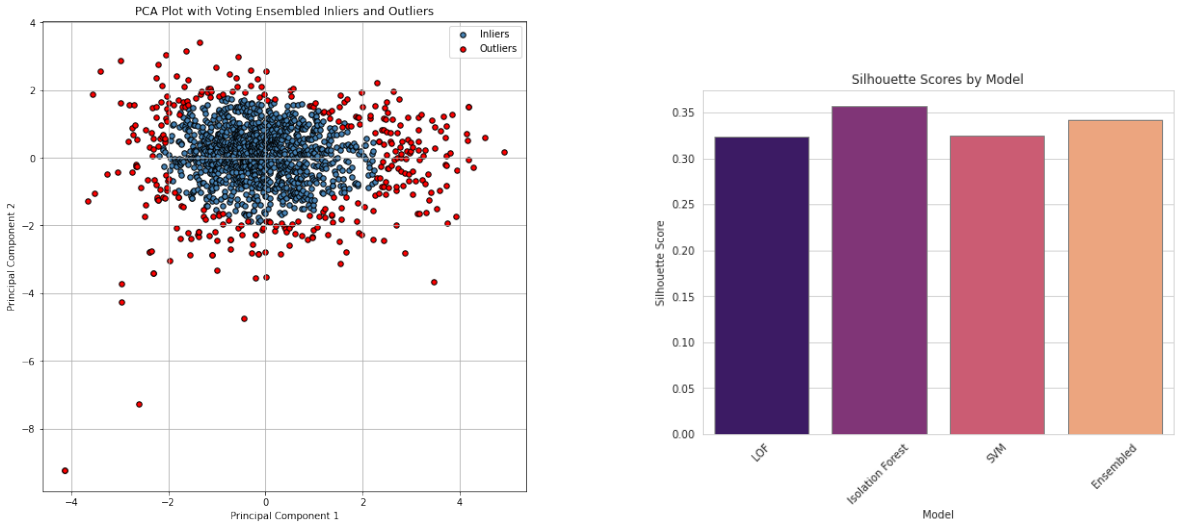


Figure 13: Ensembled output/Model evaluation

To evaluate the overall performance of our models, we chose the Silhouette Score metric, which allows us to quantitatively assess how well they are able to distinguish outliers from normal data points through in-cluster cohesion and separation between outliers and inliers. When observing the output metrics for each model on the right, we see that while Isolation Forest has the best cluster formation among the models we deployed, the voting ensembling of their results actually performed worse than its best constituent. This phenomenon is more commonly known as "ensemble divergence" or "ensemble disagreement", occurring when there is a lack of complementary strength for larger number of models. It may be the case that as more and more learners are ensembled, similar or correlated mistakes appear more often. It is also possible that the conflict in decision making based on the different algorithms is a cause for this problem. Overall, this is only a small difference, and still resulted in a successful attempt at devising an approach to separate outliers from a dataset.

2.2 Univariate time series anomaly detection

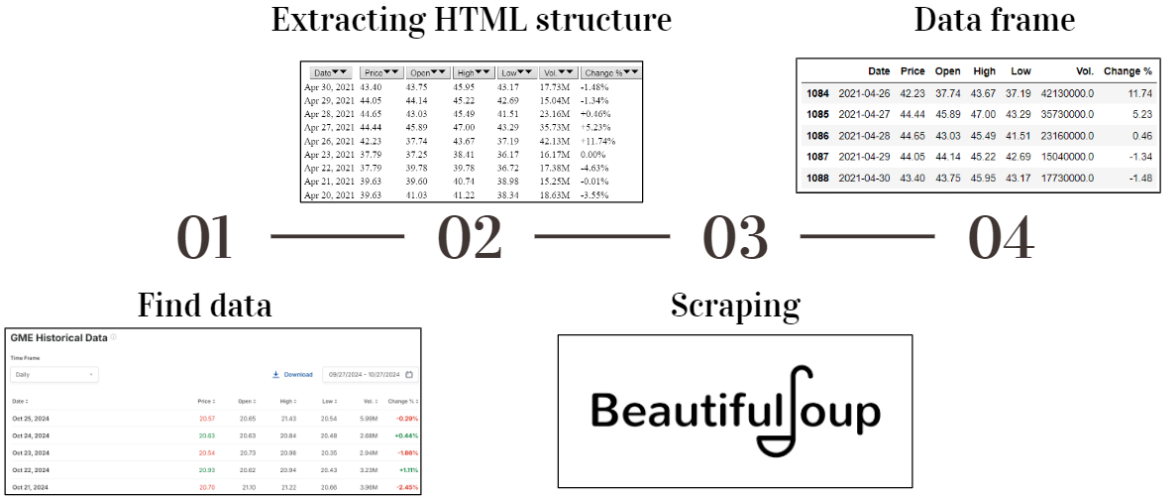
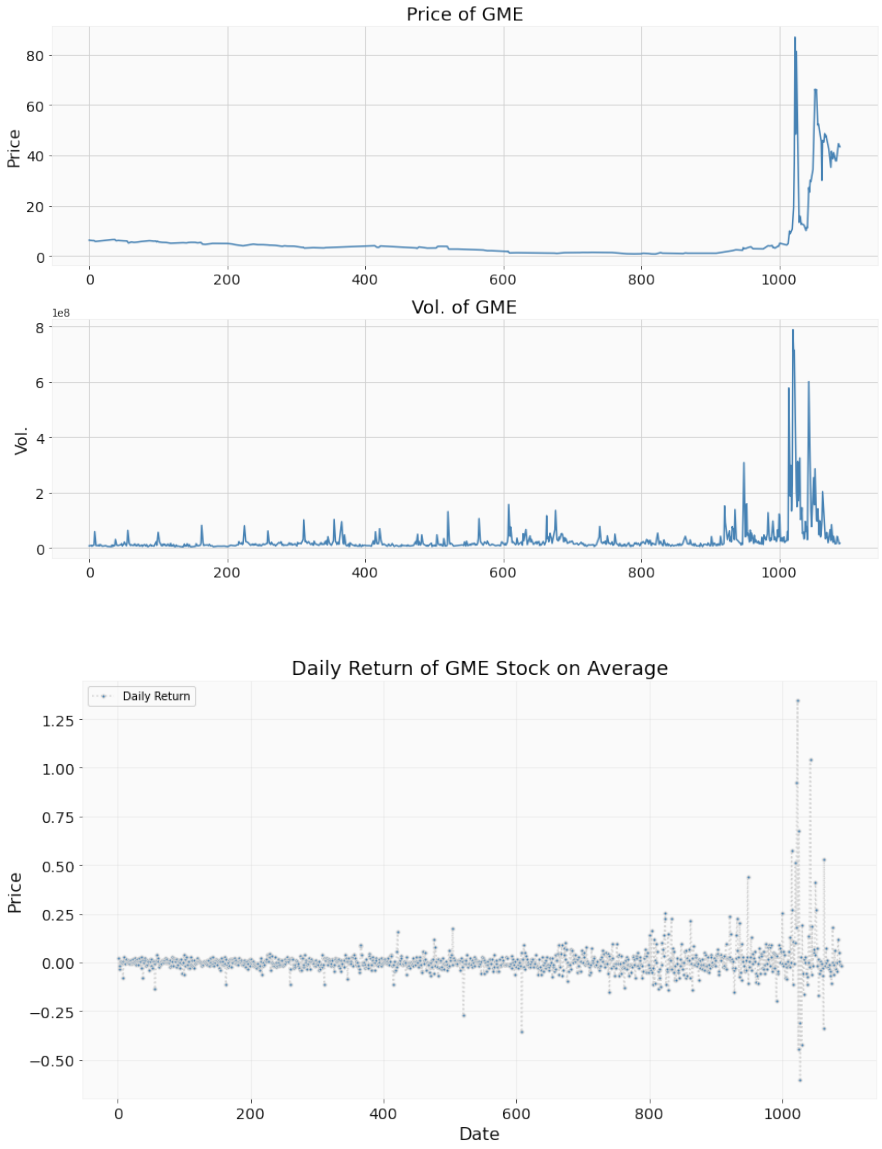


Figure 14: Data Crawling process

For this task, we decided upon using the GME stock price historical data from January 3rd 2017 to April 26th 2021 from which we would implement our methods to detect outliers. Our process begins

with extracting the raw HTML from investing.com and parsing it using BeautifulSoup to extract table data, then converting the extracted data into a dataframe. Since the original data contains Price, Open, High and Low, which are attributes pertaining to the stock price throughout the day and are closely correlated, we will only focus on the closing price to analyze for anomaly detection. This leaves us with the data as seen below:

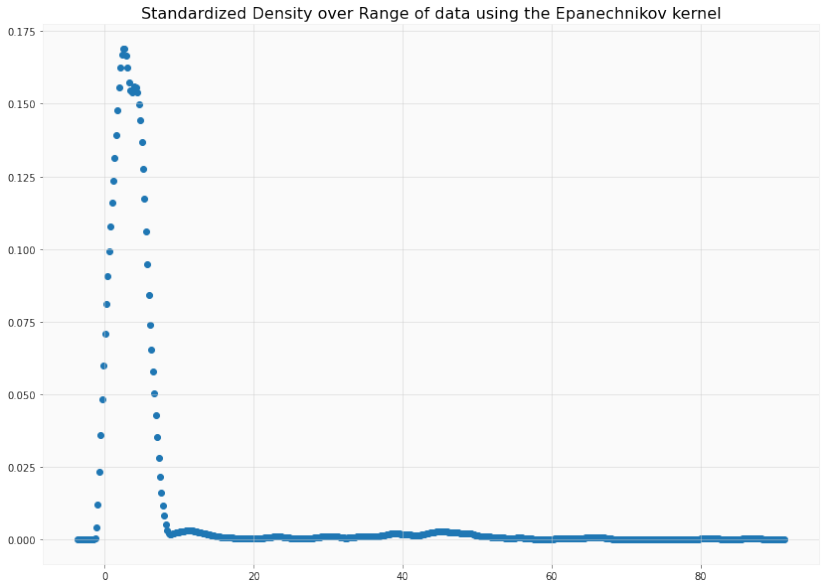


To give more context surrounding the data, the price of GameStop stocks (GME) experienced a significant and unprecedented surge that occured in late January 2021 [13]. Prior to the surge, GameStop was facing financial difficulties, with declining sales and a business model that struggled to adapt to the digital gaming landscape. By 2020, it had become one of the most shorted stocks one the market, with many institutional investors betting against it. In January 2021, retail investors on the WallStreetBets forum began to buy shares of GameStop en masse, leading to a rapid bull run that raised the price from $20 at the beginning of January to an all-time high of $483 on January 28th, 2021. Going back to our work centering around anomaly detection, this will be the event that we try to identify.

2.2.1 Spatial Approach

As explained before, this approach involves mapping the density of the data using Epanechnikov kernel. When visualizing the density of values in the historical data, we can observe that the majority

distribution is skewed toward the left; for much of the timeframe we're analyzing GME price did not change.



After the density estimation of data points is evaluated on a grid of 500 points, the obtained density values are normalized using MinMaxScaler. We transform the density values to a range from 0 to 1 for consistent comparison. Following this, we create an interpolation model to estimate density values at different points on the grid that were not directly calculated by the Kernel Density Estimation. From this step, anomalies are identified based on the threshold of 0.01. Points in the GME data where the estimated density falls below this threshold are flagged as anomalies, and we arrive at the following output:

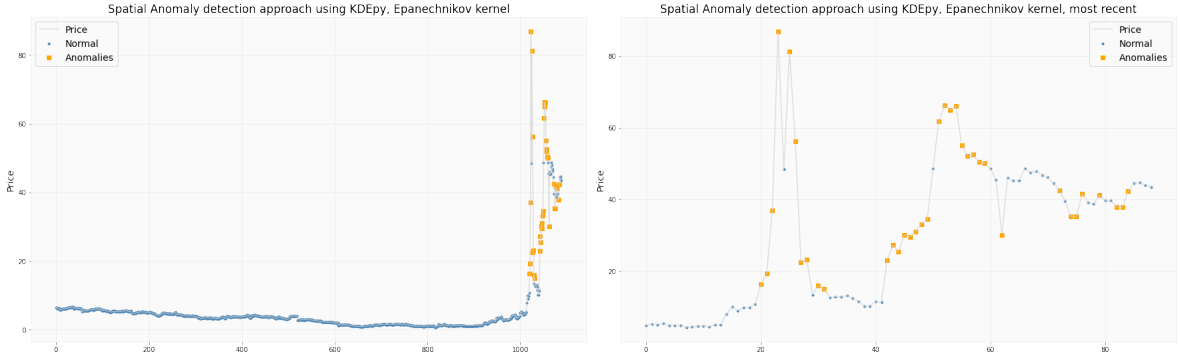


Figure 15: Results of Spatial Approach

With this method, we're able to detect sudden surges and falls in price with relative accuracy, which could be value for monitoring volatility and predicting future behaviour. We also see signs of potential price manipulation earlier on, but this method is not without its faults as some anomalies were not correctly identified.

2.2.2 Prophet

Our last method to implement is to use an instance of the Prophet model with the following configurations:

- Daily, weekly, and yearly seasonality components are turned off.

- Seasonality mode is set to 'additive'.
- Prediction interval width is set to 95%.
- The range of potential changepoints is set to 90% of the data.
- The growth type is linear.

From the forecasted prediction interval, anomalies are identified based on whether the actual value falls outside the interval. For data points that are labeled anomalies, the importance score provides a relative measure of the impact of each anomaly detected. A higher importance score indicates a more significant deviation of the actual value from the forecasted range.

- For anomalies where the actual value exceeds the upper bound, importance is calculated as the ratio of the difference between tha actual value and the upper bound to the actual value.

$$Importance = \frac{ActualValue - UpperBound}{ActualValue}$$

- For anomalies where the actual value falls below the lower bound, importance is calculated as the ratio of the difference between the lower bound and the actual value to the actual value.

$$Importance = \frac{LowerBound - ActualValue}{ActualValue}$$

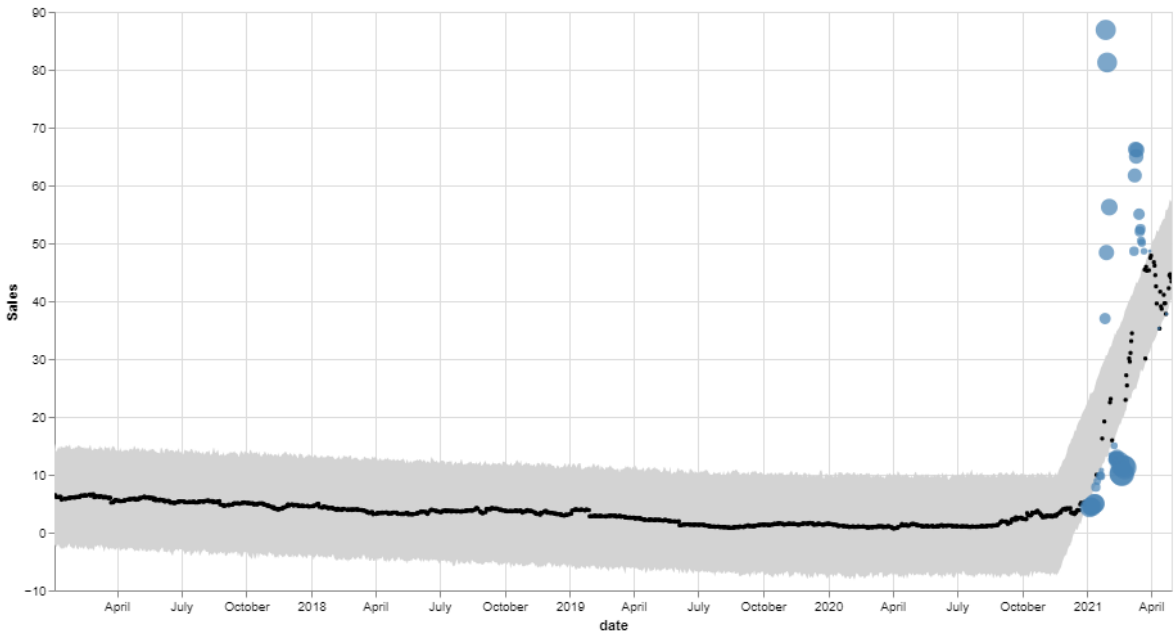


Figure 16: GME Anomaly Detection using Prophet

As a whole, the model effectively adapts to the range of the stock price throughout the time frame. One of the most common problem with using Prophet for discretionary portfolio managers in the financial sector is that Prophet has a hard time adapting to sudden and abrupt changes, but this attribute actually works in our favor as points that veer too far off from the predicted interval are labeled as outliers, with markedly high accuracy.

3 Conclusion

In conclusion, our study on anomaly detection has shed light on the efficacy as well as the shortcomings of the proposed methods, in both clustering-based approaches for multivariate datasets and in univariate time series. Our research underscores the effectiveness of these algorithms in pinpointing anomalies across different data structures, but more importantly highlighting their limitations in different scenarios (as were the case with Local Outlier Factor's emphasis on knowledge of data's contamination and one-class SVM's reliance on defined inlier distribution). As many state-of-the-art models are being discovered and implemented to tackle real world problems, novel methodologies are being explored, we will be able to advance anomaly detection capabilities and fortify data analytics practices across every industry.

References

[1] Joel Barnard, Cole Stryker (2023). *What is Anomaly Detection.*

[2] Aggarwal, Charu C. Outlier Analysis. Springer New York (2017) *An Introduction to Outlier Analysis*

[3] Fei Tony Liu; Kai Ming Ting; Zhi-Hua Zhou (2008). *"Isolation Forest". 2008 Eighth IEEE International Conference on Data Mining. pp. 413–422.*

[4] Shaffer, Clifford A. (2011). *Data structures & algorithm analysis in Java* (3rd Dover ed.). Mineola, NY: Dover Publications.

[5] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, Jörg Sander (2000) *LOF: Identifying Density-Based Local Outliers*

[6] https://www.sciencedirect.com/topics/engineering/reachability

[7] Sankhya A. (1936) *"Reprint of: Mahalanobis, P.C. "On the Generalised Distance in Statistics." "*

[8] P. J. Rousseeuw. Least median of squares regression. J. Am Stat Ass, 79:871, 1984.

[9] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylort, John Platt (2000) *Support Vector Method for Novelty Detection*

[10] https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.interp1d.html

[11] https://archive.ics.uci.edu/dataset/110/yeast

[12] S. S. Shapiro and M. B. Wilk (1965) *An Analysis of Variance Test for Normality (Complete Samples)*

[13] Beatrice Titus - *Inside the GameStop Saga: A Lesson in Narrative Attacks and Market Manipulation*