ABSTRACT

Title of Thesis:	A COMPARATIVE STUDY OF OUTLIER DETECTION METHODS AND THEIR DOWNSTREAM EFFECTS
	Vikram Adipudi, Master of Science, 2024
Thesis Directed By:	Jeffrey W. Herrmann, Institute for Systems Research

When fitting machine learning models on datasets there is a possibility of mistakes occurring with overfitting due to outliers in the dataset. Mistakes can lead to incorrect predictions from the model and could diminish the usefulness of the model. Outlier detection is conducted as a precursor step to avoid errors caused by this and to improve performance of the model. This study compares how different outlier detection methods impact regression, classification, and clustering methods. To identify which outlier detection performs best in conjunction with different tasks. To conduct this study multiple outlier detection algorithms were used to clean datasets and the cleaned data was fed into the models. The performance of the model with and without cleaning was compared to identify trends. This study found that using outlier detection of any kind will have little impact on supervised tasks such as regression and classification. For the unsupervised task different clustering models had outlier detection and removal algorithms that made the most positive impact in the clustering. Most commonly IForest and PCA had the greatest impact on clustering methods.

A COMPARATIVE STUDY OF OUTLIER DETECTION METHODS AND THEIR DOWNSTREAM EFFECTS

by

Vikram Adipudi

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2024

Advisory Committee:
Professor Jeffrey W. Herrmann, Chair
Dr. Thomas Hedberg
Professor Michael Fu

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Outliers are observations that are deviations from what is normal. They are common occurrences in many datasets. Outliers can distort the information that is extrapolated from data. Outliers in a dataset can be caused by a simple measurement error, experiment variability, or other unique events. Outliers are also commonly referred to as anomalies or novelties, technically these words refer to different types of deviations: anomalies are instances that are a part of a different distribution, and novelties are instances that come from a new/evolving distribution (under the given distribution) [1]. For this thesis these three terms will be treated the same and referred to as outliers: instances that are low probability for the given distribution [1]. When using any dataset, outlier removal is a crucial preprocessing step to prevent the outliers from skewing data and effecting the results.

1.1 Outlier Detection

Outlier detection is used for many fields to identify novelties that need to be inspected more thoroughly, such as in medical imagery where an outlier could be a harmful condition or in fraud detection where outliers could be crimes that need to be inspected. Outlier detection and removal is also useful as a data preprocessing step for future downstream (DS) tasks with the data. These downstream tasks benefit from a cleaned dataset so they can function efficiently. For the sake of this thesis, I will focus on machine learning (ML) downstream (DS) tasks such as regression learning, classifying, and clustering. Because outliers can cause incorrect results for these ML tasks, removing outliers yields cleaner and more accurate results.

1.2 Downstream Tasks

Regression learning is a supervised learning algorithm that can be used to predict new values. This can be used for forecasting and accurate prediction when the dependent variable is continuous. They work by training a regression model on a dataset to predict future values. Inconsistencies in the training data will affect the model's accuracy. For example, the model tends to assume that the training dataset is distributed normally, and outliers violate these assumptions. Figure 1 shows an example in which Linear Regression was used to generate a regression model for a dataset with outliers and to generate a second regression model for the same dataset after using Local Outlier Factor to identify some outliers, which were removed.

Figure 1: Regression with outliers and cleaned data

Similar to regression learning, classification algorithms are supervised algorithms that yield a model that can predict the class the subject will be. The model is trained by a dataset that has defined class labels. Outliers will affect what the model considers a class, which can lead to class imbalances as well as affecting the decision boundaries for what the model considers for each class. The classifier may incorrectly classify certain datapoints if the boundary is skewed. Figure 2 shows an example of this phenomenon; when the outliers are included, Logistic Regression Classification generates very different decision boundaries for the classes.

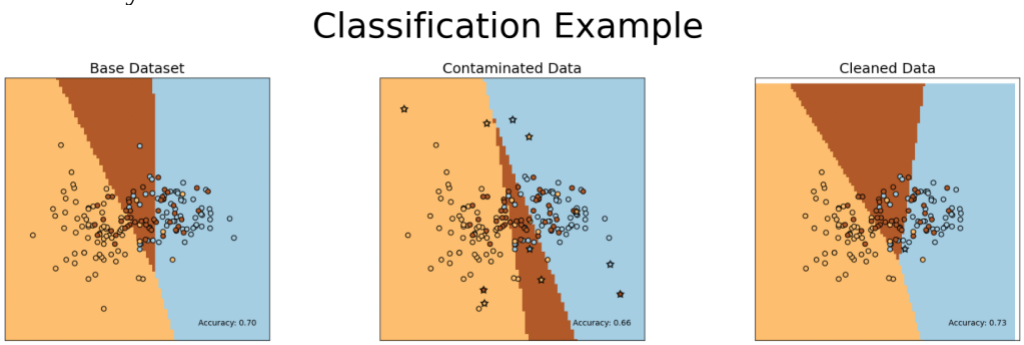

Figure 2: Classification with outliers and cleaned data

Clustering algorithms are unsupervised algorithms that group similar datapoints into clusters to find patterns in data. For Clustering tasks, outliers will yield distorted clusters and shift centroids. Figure 3 shows an example of this issue; where outliers affect the agglomerative clustering models output, and outliers removed with Local Outlier Factor fixes this issue.

Figure 3: Clustering with outliers and cleaned data

1.3 Research Problem

In all these tasks outlier detection and removal can vastly improve the robustness of the models. With many different outlier detection (OD) algorithms and many downstream algorithms and many different types of data sets, there is no established combination of these factors to ensure quality results. The purpose of this thesis will be to conduct a comparative analysis of the downstream effects of different outlier detection and removal methods on machine learning tasks.

To compare the outlier detection methods, the performance of downstream algorithms will be measured when trained on datasets that have been cleaned by removing the outliers that were detected with different algorithms. The accuracy and training time for the models will be the major metrics to consider for performance.

1.4 Thesis Outline

The remainder of the thesis includes the following chapters: Chapter 2 consists of the literature review. This includes the background on outlier detection models and their characteristics, and it also includes the current state of comparisons. Chapter 3 presents the approach that was used to compare the OD algorithms. It describes the specific datasets, DS algorithms and OD algorithms used as well as the parameters that were set for each algorithm. Chapter 4 contains the results for the regression, classification, and clustering algorithms. Finally, Chapter 5 will conclude the study.

Chapter 2: Literature Review

2.1 Outlier Detection Methods

Because of the importance of outlier detection (OD) in data analysis, the field has been greatly researched and many different OD algorithms with varying complexity and efficiency have been created. These OD algorithms can be classified into three groups: distance-based methods, probabilistic methods, and dimension reduction methods.

2.1.1 Distance-based Methods

Distance-based methods use the locations of the data points and compare them to surrounding datapoints to identify outliers. Often distance-based methods of outlier detection are used because they are easy to implement and computationally inexpensive. The most common distance-based algorithm is K-nearest neighbor. KNN calculates the distance of a test point to its K nearest neighbors, where K is a parameter [2]. If the total distance is greater than a threshold value, the test point is considered an outlier. KNN is dependent on the value of K chosen. KNN is an easy algorithm to implement, and it is generally good for low dimensional data however, for high dimensional data it is not effective [1]. The increase in dimensions increases the complexity of the distance formula and unique outliers can't be identified.

An outlier detection method that is effective for high-dimensional data is the Local Outlier Factor (LOF) algorithm. LOF takes the local density of a test point and compares it to the local density of nearby datapoints, if the density of the test point is different from the density of nearby points it is considered an outlier. The local density refers to the number of the local/nearby datapoints over the local space. LOF doesn't make any assumptions on the distribution of the data, therefore it can work for a dataset that isn't bound by a traditional probability density, or when the datasets have different distributions at different places in the dataspace.

The last method that is commonly used is an Isolation Forest (IForest). Unlike the previous algorithms IForest doesn't look at metrics between datapoints, therefore they aren't distance based in the same sense as KNN and LOF. However, they are based on the positions an outlier has in the dataset respective of the inliers, so they remain in this family. The idea of IForest is that if a dataset has an outlier, it will be easier to separate from the dataset [3]. IForest works by recursively splitting the data into branches. The data is split by randomly choosing a feature and randomly selecting a value in that feature space to split the data. The two splits of the data become child nodes in the tree. Each node is then split into two more nodes, this repeats until a node is left with a single data point. The outlier score for each datapoint is calculated by the depth in the tree for that datapoints node, the shallower nodes are considered outliers.

2.1.2 Probabilistic Methods

Another way of determining outliers is by estimating a probability distribution for the dataset and identifying datapoints that don't conform to the distribution as outliers. This method is referred to as probabilistic. One of the oldest OD methods is done by computing the Mahalanobis distance of a test point to the mean of the dataset: if the distance is greater than a certain threshold parameter, then the test point is considered an outlier. Given the dataset's covariance matrix $\mathbf{\Sigma}$ the Mahalanobis distance D_M for each point is the following. Where $\mathbf{x}$ is the datapoints vector and $\boldsymbol{\mu}$ is the datasets mean vector [4].

$$\mathrm{D_M}(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \mathbf{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}$$

The Mahalanobis method is good for multi-variate data because it considers the variance of the variables, so the distance relates to the relation between variables [5]. However, the Mahalanobis method only works on the assumption of a normal distribution in the dataset, the accuracy diminishes when this is not the case. Also, the Mahalanobis method requires matrix inversion therefore it can be computationally expensive for high dimensional datasets.

Another common probabilistic OD method is Kernel Density Estimation (KDE). KDE is a non-parametric method to estimate the PDF of the dataset. The PDF is created by taking the density of the data at sections and smoothing values with a kernel. KDEs can be used for outlier detection by finding datapoints that exist in low density positions within the estimated PDF [6]. An issue with the KDE is that it assumes the data can be described as a continuous density function.

Along with the different families of outlier detection there is also another distinction within probabilistic methods and that is whether or not an algorithm uses deep learning. Deep learning models utilize neural networks with multiple layers. Normalizing flows is a deep learning method that creates more accurate density functions. This method also creates a density estimation, but it works by starting with a base density function and transforming it with a neural network to create an invertible density [7,8]. This tends to be a more accurate representation of the data, but the method is also computationally expensive.

2.1.3 Dimension Reduction Method

The last family of outlier detection methods are dimension reduction methods. These involves identifying key features of the data so the data can be reduced to a lower dimensionality. The most common reconstruction method is Principal Component Analysis (PCA). PCA involves transforming data to a reduced dimensionality latent space where principal components and corresponding eigen values are established. In the reduced dimensionality a distance-based OD algorithm is used to identify outliers. This is useful when the distance-based method is expensive or inaccurate for high dimensions

Similar to Probabilistic family there are also dimension reduction method that utilize deep learning: Autoencoders (AE) and Variational Autoencoders (VAE). Autoencoder's are another method that is similar to PCA but AEs use a neural

network to reduce the dimensionality and reconstruct the data in the high dimension space [1]. The reconstruction methods can only reconstruct normal data points, abnormal points cannot be reconstructed therefore any data point that is not correctly reconstructed is classified as an outlier [1]. The last reconstruction method is a Variational Autoencoder (VAE), these work similar to AEs except when they are encoding the data, they don't directly map the datapoints to the new latent space. Instead, it maps the datapoint to a distribution in the latent space, this helps in preventing overfitting of the model. This also makes VAEs a probabilistic method along with being a reconstruction method.

2.2 Current state of Comparisons

Considering all the different OD techniques stated, along with multiple variations of these, it is difficult to know which algorithm is the best. One comparative study on outlier detection algorithms evaluated several outlier detection models and compared their performance among several different datasets with varying number of dimensions and number of anomalies [9]. That study compared many different factors such as the impact of dimensionality, dataset size, the training time, prediction time and memory usage. However, the most important performance metric is the accuracy, which is measured by calculating the area under the precision recall (PR) curve and the area under the receiver operator characteristic curve (ROC) for several datasets. Overall, the outlier detection with the best PR AUC metric was IForest, and Kernel Density Estimator had the best ROC AUC. This study doesn't investigate how these outlier detection methods are beneficial as a preprocessing step.

There are currently other works that investigate how a certain DS algorithm might benefit from a certain OD algorithm, conducted to see if a DS task benefits from outlier removal. Bramm et al. investigated how outlier removal along with other data preprocessing techniques can be used to improve a solar power plant forecasting regression model [10]. In this paper a simple box and whisker outlier analysis was used to identify datapoints to remove, with this the linear regression model had "reduced the mean absolute percentage error *(MAPE)* to 43.74%" [10]. This paper only focusses on the box and whisker outlier detection method and not more complicated OD algorithms. It also only studies the impact that preprocessing has on linear regression and not any other regression model. Almeida et al. improved hierarchical cluster analysis with outlier detection [11]. In this paper they used a modified density outlier detection method to remove outliers from a dataset before clustering the data with several different clustering algorithms. The outlier removal step improved some clustering algorithms but for other clustering algorithms outlier removal had little effect, or it would cause more clusters to form. Unfortunately, only datasets with 2 variables were considered in this paper, it doesn't test if the results are consistent for different types of datasets. This paper is good at examining how outlier removal can affect different clustering algorithms, but it doesn't check how different outlier detection algorithms affect clustering algorithms.

These are papers that narrowly look at specific cases of OD with DS tasks, but there is a lack of information on the best combinations: which OD methods improve which DS tasks the most?

The efficiency of the OD algorithms was not considered in these comparative studies. In order to find the best combination, it is also important to identify if a complicated deep learning OD method is necessary for a certain type of dataset, downstream algorithm, or use case. Spending time and effort to train a model for outlier removal might not be necessary for simple datasets or if a regression is also complex and robust enough to be not as affected by outliers.

Chapter 3: Approach

This study examines the downstream effects of using different outlier detection algorithms to remove outliers from datasets. Several different OD algorithms will be compared in order to consider all combinations of OD with downstream algorithms. The changes that different OD algorithms make in the performance of different regression, classification and clustering algorithms will be studied to determine what OD algorithm works best with each downstream algorithm.

3.1 Methodology

For the Supervised tasks each dataset was split into train and test sets. The training dataset was then cleaned by running an outlier detection algorithm and removing the outliers. Once the outliers were removed from the training data then the training data was used to train each Regressor or Classifier which resulted in a trained model. The model was then used to predict values for the test data.

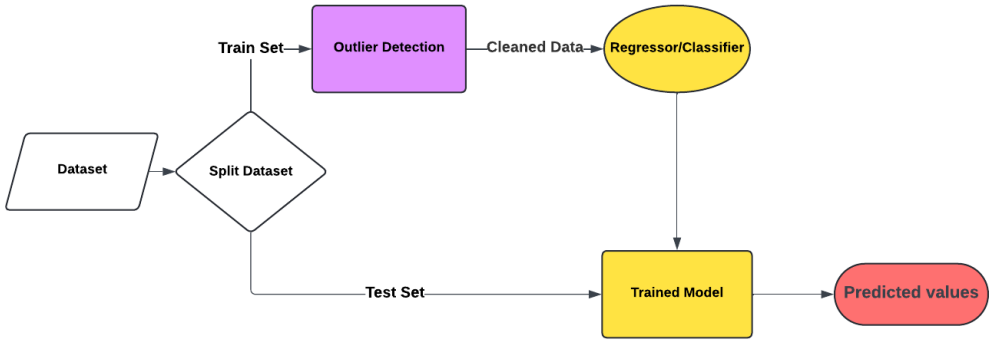

Figure 4: Methodology for Supervised tasks

For the Clustering task the path is similar with the exception of splitting the dataset. Outlier detection and removal was conducted on the dataset, and the cleaned data was fed into the clustering model to obtain labels for each data point.

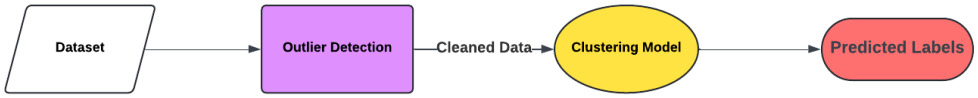

Figure 5: Methodology for Clustering

3.2 Datasets

The study used 13 unique datasets that have varying number of features and characteristics. These datasets come from papers that use machine learning models to assume the target feature. They are also commonly used for testing different machine learning models as well as being used in testing outlier detection methods. Domingues et al. uses some of the same datasets in their Outlier detection Summary [9]. All these data sets are publicly available from UCI [12] and OpenML [13]. There are 6 datasets for regression, 7 datasets for classifications, and 6 datasets for

clustering. The clustering and classification datasets are the same. this way the clustering tasks have a ground truth that can be compared to the labels assigned, otherwise there would be no way of telling if a clustering algorithm was accurate. In practice clustering would be performed on datasets where the ground truth is unknown.

Table 1: Datasets used

Datasets	Task	Features	Continuous Dims	Categ. Dims	Instances	Target Feature	Classes/ Labels	Characteristics
Abalone	Regression	8	7	1	4177	Age		
Ailerons	Regression	33	33	0	13750	goal		
Bike Share Demand	Regression	11	6	5	17379	count		Sensor, Time
diamonds	Regression	9	6	3	53940	cost		
Wine Quality	Regression	12	11	1	4898	Quality		
Sulfur	Regression	7	7	0	9368	gas concentrate		Sensor
covertype partial	Classification, Clustering	54	8	46	8000	cover type	7	
Dry Bean Dataset	Classification, Clustering	16	16	0	13611	bean type	7	
eeg_eye	Classification, Clustering	14	14	0	14980	eye open/closed	2	Sensor
Yeast	Classification	8	8	0	1484	localization site	10	
Run or walk info	Classification	6	6	0	88588	run or walk	2	Sensor
shuttle	Classification, Clustering	9	9	0	58000	class	7	

Along with the datasets above, variations were made to the datasets to create 24 new datasets. The new datasets are based on the original datasets except noise is added to simulate outliers. The noise added is impulse noise, this is done to simulate real world sensor data. Impulse noise happens in sensors that incorrectly registers values as the max or min of the possible values. The new datasets created are created by randomly choosing instances and randomly choosing a feature to add impulse noise to. For each dataset, two new datasets are created with a tenth and a fifth of the datapoints being noise.

The categorical dimensions include both nominal and ordinal values, these can pose problems for some OD algorithms and downstream algorithms, since getting distances from categorical values is a difficult task and would need a distance measure to be created. To avoid this, features have been encoded to numerical values. For the nominal features, one-hot encoding was applied. In the case of the abalone dataset, the abalone sex feature has three values male, female, and child, so it becomes three new features for sex_male, sex_female, and sex_child where a value of 1 in that feature indicates that that instance is that sex. For the categorical features that are ordinal the values are transformed to values of 1 to the number of distinct values in that feature. However, if the features are time-based then they are cyclically encoded, because for time features the largest value will loop around to the smallest. For example, in the bike_sharing dataset there is a Month feature, the values are from 1 to 12 but the distance from January to February is equivalent to the distance from

December to January, so cyclical encoding is needed. In cyclical encoding two new features are needed for feature 'X', Xsin = sin(2πX/max(X)) and Xcos = cos(2πX/max(X)) are created.

Since some OD algorithms are based on distance there is a bias against features that have a large range of values. The datasets features have all been feature scaled to prevent this bias, after normalization all features have a range from zero to one. Min-max normalization was the feature scaling method used, for each instance the value x in the feature X the new normalized value gets the following value x' = (x-min(X))/(max(X)-min(X)). Min-max normalization was chosen to keep the effect of outliers prevalent, min-max normalization is sensitive to outliers because the distance to outlier values is scaled down [14].

Regression and classification are supervised tasks, and they need to be trained on some data, so after the datasets are encoded and normalized, they are then split into training sets and testing sets. The training sets are split from each of the datasets by the values: 400, 600, 800, 1000,1200. Varying training sizes are used for each dataset to get a range of results for different amounts of training data. They are not separated by percent of the dataset since a large training set is correlated to improve supervised performance and the datasets are various sizes. The datasets are all split by the same incremental amounts for the training set, the testing set for each dataset will be different but since in regression and classification the metrics used are averages of the predicted values having inconsistent sizes.

3.3 Outlier Detection and Removal

After the dataset was normalized, outlier detection was performed on all data for clustering and the training data for regression and classification. For classifier training datasets when outlier detection occurred while ignoring the class label, the model would treat less abundant classes as outliers and remove them from the dataset entirely. This would cause the classifier to label the test set by a limited amount of classes. To avoid this class specific outlier detection occurred, the training data was divided up by the classes and then outlier detection was performed on each class. If the class had a small number of instances, then there was the possibility that no instances were considered outliers. After the outliers were removed from each class the data was combined in the order before separation.

Table 2: Outlier Detection algorithms used

Outlier Detection	Parameter A	Parameter B	SciKit-Learn Function
K Nearest Neighbor	Number of Neighbors	N/A	Sklearn.neighbors.NearestNeighbors
Local Outlier Factor	Number of Neighbors	N/A	Sklearn.neighbors.LocalOutlierFactor
Isolation Forest	Number of Estimators	Max Features	Sklearn.ensemble.IsolationForest
Principal Component analysis (with KNN)	Number of Neighbors	Number of Components	Sklearn.decomposition.IsolationForest
Kernel Density Estimation	N/A	N/A	Sklearn.neighbors.KernelDensity

Mahalanobis	N/A	N/A	N/A

The Distance based methods used are KNN, LOF and IForest. The dimension reduction method that was examined was PCA. For PCA KNN was used to find outliers in the reduced dimensionality. The probabilistic models used are KDE and Mahalanobis. Most of the outlier detection models were built from the SciKit-Learn API [15], specific functions used are shown in Table 2. For Mahalanobis, the distance equation was translated to python to get outlier scores.

Each of the models have different parameters to be considered so to keep it consistent the two most important parameters (param A, param B) for each algorithm were varied. Param A varied in values from 2-7. For KNN, LOF, and PCA param A was the number of neighbors to be looked at, PCA doesn't look at the number of neighbors but inside the PCA function KNN is used and that is where the number of neighbors matters. For Isolation Forest Param A was the number of estimators, or the amount of trees/splits. Parameter B was a range of values from: min, small, medium, large, and max the actual values changed based on the function. Min and max were the smallest and largest possible values for the parameter and small, medium, and large were 25%,50%, and 75% of the possible values respectively. Only two functions needed the second parameter, PCA and IForest. Param B in PCA represented the number of components to reduce the feature space to. Max value for PCA was one less than the number of features, so the dimensionality can still be reduced. In IForest Param B represented the max features to be considered when fitting each tree. If an algorithm required more than two parameter, constant values were used for the excess parameters.

Table 3: Parameter B values

Dataset	# Encoded Features	Min	Small	Medium	Large	Max - IForest	Max - PCA
Abalone	10	1	3	5	8	10	9
Ailerons	33	1	9	17	25	33	32
Bike Share Demand	16	1	4	8	12	16	15
diamonds	9	1	3	5	7	9	8
Wine Quality	12	1	3	6	9	12	11
Sulfur	6	1	2	3	5	6	5
covertype partial	54	1	14	27	41	54	53
Dry Bean Dataset	16	1	4	8	12	16	15
eeg_eye	14	1	4	7	11	14	13
Yeast	8	1	2	4	6	8	7
Run_or_walk_information	6	1	2	3	5	6	5
shuttle	9	1	3	5	7	9	8

To determine the effect of different amounts of outliers removed a contamination threshold is utilized. Each OD function returns an outlier score, a metric that identifies how much an instance (data point) is an outlier. The contamination threshold is a value from 0.05 to 0.35 it represents the percent of the

data that is to be treated as outliers. For example, with a contamination threshold of 0.05, the instances with outlier scores in bottom 5 percentile are classified as outliers.

3.4 Downstream Tasks

To get a comprehensive view of how machine learning tasks are dependent on outlier detection and removal, multiple algorithms were used to solve the tasks. For regression learning multiple linear, ridge, decision tree, random forest and support vector machine (SVM) regression models were used. For classification tasks the algorithms studied were decision trees, random forest, KNN, logistic regression and SVM. Regression and clustering tasks are very similar so there are overlap in the algorithms used. Finally, the clustering algorithms used were k-means, gaussian mixture, agglomerative, density-based spatial clustering of applications with noise (DBSCAN) and hierarchal density-based spatial clustering of applications with noise (HDBSCAN) clustering. The downstream algorithms were obtained from the Scikit-Learn API [15], specific functions are shown in Table 6.

Table 4: DS algorithms used

	Algorithm	SciKit-Learn Function
Regression	Multiple Linear Regression	linear_model.LinearRegression
	Ridge Regression	linear_model.Ridge
	Lasso Regression	linear_model.Lasso
	Elastic Net Regression	linear_model.ElasticNet
	Random Forest Regression	ensemble.RandomForestRegressor
	Support Vector Regression	svm.SVR
Classification	Logistic Regression Classifier	linear_model.LogisticRegression
	Decision Tree Classifier	tree.DecisionTreeClassifier
	Random Forest Classifier	ensemble.RandomForestClassifier
	Support Vector Classifier	svm.SVC
	KNN Classifier	neighbors.KNeighborsClassifier
	Gradient Boosting Classifier	ensemble.GradientBoostingClassifier
Clustering	Gaussian Mixture Clustering	mixture.GaussianMixture
	Agglomerative Clustering	cluster.AgglomerativeClustering
	K Means Clustering	cluster.Kmeans
	DBSCAN	cluster.DBSCAN
	HDBSCAN	cluster.HDBSCAN

3.5 Performance Metrics

The main performance metrics are different for each of the downstream tasks, but they essentially measure the accuracy of the models. For regression the root mean square error (rMSE), mean absolute error (MAE) and adjusted r-squared will be calculated from the predicted values and the actual test values. The rMSE and MAE will show how accurate the model is. R-squared metric shows how well the model fit the data. Adjusted r-squared is used instead of r-squared to adjust for the amount of

features in the dataset. Adjusted r-squared metric is equivalent to: $1 - \frac{(1-R^2)(n-1)}{n-p-1}$, where R^2 is the r-squared metric for the dataset.

For classification tasks the accuracy, precision, recall and F1-score were measured to provide insight on the models. Precision and recall are needed to show if the models are better for false positive and false negatives respectively. This is important for cases where only one is important and the other might be irrelevant. For example, when testing for diseases false positives are ok but false negatives can lead to dangerous results.

Finally for clustering the silhouette score and the adjusted rand index (ARI) is measured. The silhouette score is an average measure of how far each instance is from the center of its cluster. This is a good metric to show if the clusters are dense and uniform. However, silhouette score isn't a perfect measure of cluster accuracy, especially when the distribution of the cluster is more complex. The ARI is a better metric for that case because it compares the similarity between predicted labels with the actual labels. It is adjusted because it accounts for the similarity that can be achieved by randomness.

Since there are many different OD methods all with varying parameters along with different DS algorithms and Datasets, there will be many different metrics for each combination, and it will be difficult to identify trends in the data. To simplify this the metrics are grouped by averages or max/min together to help identify trends. With a Dataset i, with a train/test split value s, downstream algorithm j, OD algorithm k, threshold parameter t and parameters a and b we can get a performance metric $Y_{isjktab}$. If we average the values across the split value for every dataset, we get $Y_{\bar{\imath}jtab} = \frac{1}{n}\sum_s Y_{isjktab}$ where n is the number of split values. This metric is useful to show the values independent of the train/test split. Another generalization that is made is by using the best performance values across all possible values for parameters a and b for the OD algorithm, $Y'_{\bar{\imath}jk_t} = \max_{a,b} Y_{\bar{\imath}jk_{tab}}$. For rMSE however the equation used is $\min_{a,b} Y_{\bar{\imath}jk_{tab}}$, since for rMSE a lower value is better. The best value is used to show the potential of the outlier detection with ideal parameters. Since the study examines how OD improve the DS algorithms the relative change between control and OD needs to be measured. The control in this case would be the performance of the downstream algorithm with no OD. To measure the relative change Z the fraction of the OD metrics Y' to control metrics Y_c are measured, $Z = Y'/Y_c$. With the fraction of the control the improvements of the OD methods can be measured, the values can also be kept consistent across datasets where metrics might have different ranges.

Chapter 4: Results

To determine the ideal outlier detection algorithm to use for a downstream algorithm and dataset many different groupings of each factor were compared to find trends. However, because of the vast amount of different groupings it is difficult to get a universal solution.

4.1 Regression

For regression algorithms, there are two major features considered when determining the performance of the regressor: the accuracy of the model, and the goodness of fit of the data. These features can be measured with the root mean square error and the adjusted R-squared. In both cases the dataset considered has the most impact on the values followed by the regressor used, with outlier detection having the smallest impact.

4.1.1 Root Mean Squared Error

Outlier detection seems to have a very limited and negligible impact on the rMSE of the regressor. As seen in figure 6, with OD the error only goes as low as 90% of the control and that is only on one instance. In most cases the regressor produces more error as more data is removed from the train set. This is true for all models except support vector regression where KNN, PCA, and IForest reduced the error. However, when KNN, PCA or IForest is used with elastic net regression they increased the error more than the other OD methods. This shows how a certain OD algorithm could help or harm a regressor depending on the regressor.

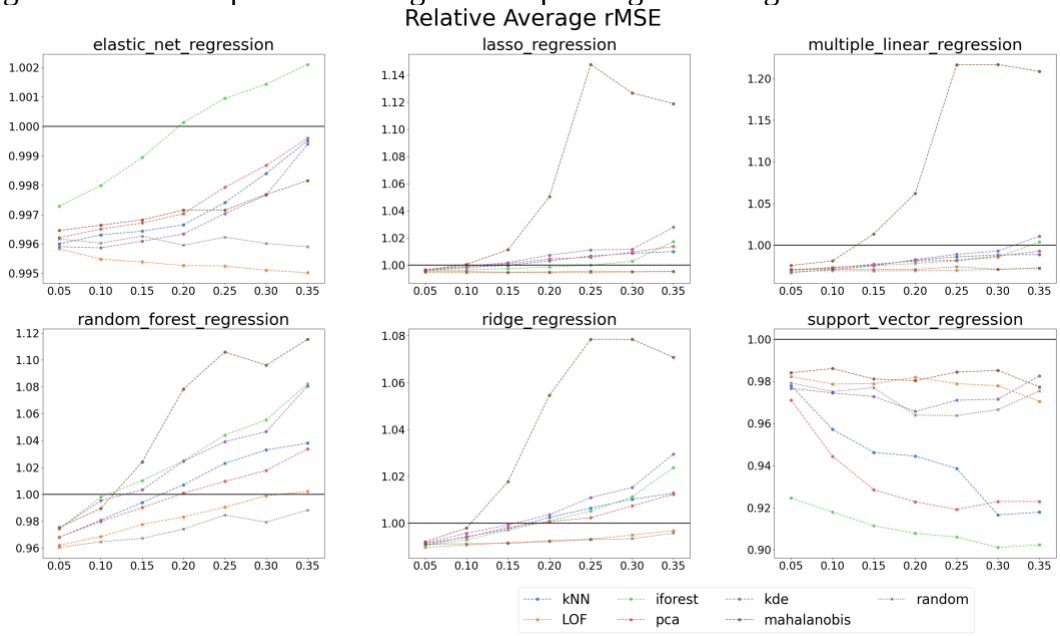

Figure 6: Relative root mean square error (rMSE) for regression models with outlier detection as a fraction of the control group. The best values for each parameter configuration at each threshold were averaged across the datasets

4.1.2 Adjusted R-Squared

The OD methods don't have a great impact on the adjusted r-squared metrics. As seen in Figure 7 the effect of the outlier detection would be minimal and would rarely be greater than the control. The main exception being when OD was performed on elastic net regression and lasso regression. For these regressors cleaning the data improves the adjusted r-squared, and as the contamination threshold increases the adjusted r-squared also increases. Especially when the data is cleaned with IForest. Adversely, the random forest regression model performs worse when the data is cleaned and gets significantly worse as the contamination threshold is increased.

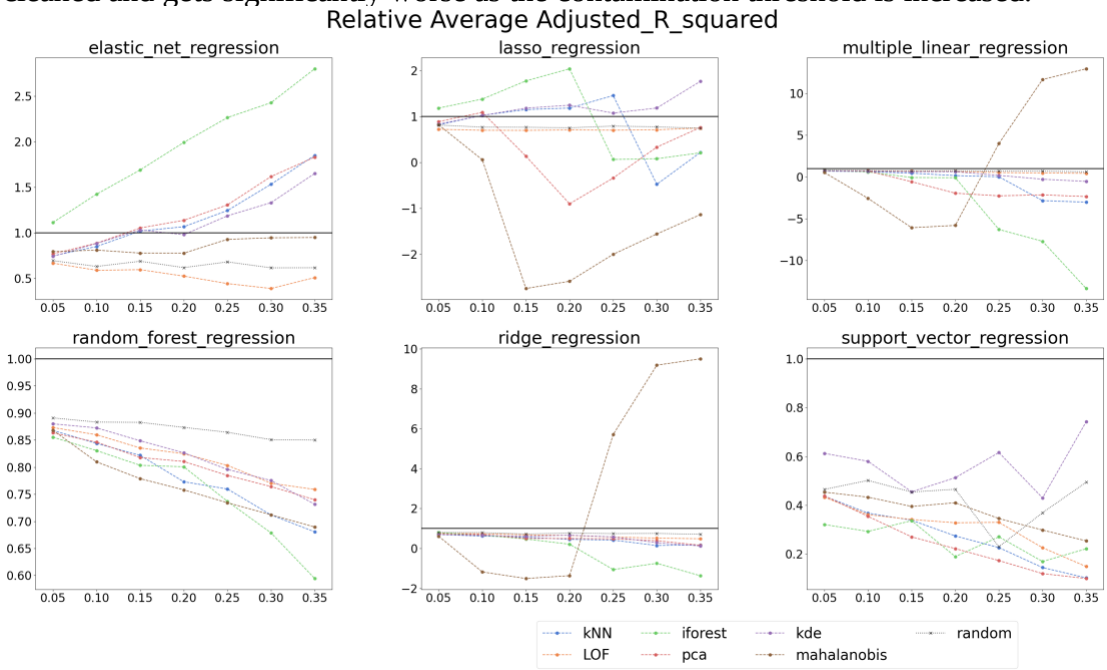

Figure 7: Relative Adjusted R-square value for regression models with outlier detection. The best values for each parameter configuration at each threshold were averaged across the datasets

4.2 Classification

For Classification tasks the outlier detection slightly improved the performance of the classifiers. However, it is a very small possibly negligible increase. As seen in Figure 8 the OD algorithms improves the accuracy of the models, especially at low threshold values. The improvement, however, is only around a two percent increase at most. There is also very little variations with which OD method is used, the only notable features are that KDE improves random forest, support vector, KNN and gradient boosting classifiers the least and LOF is the best OD method to use with support vector classifiers.

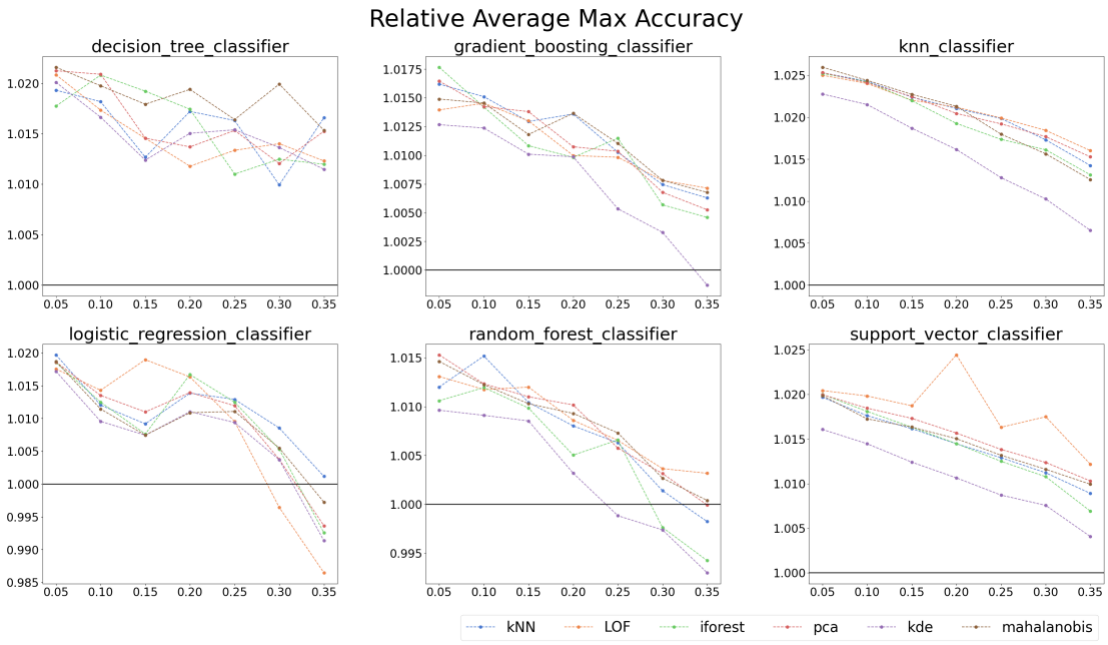

Figure 8: Relative average for classification models with outlier detection. The best values for each parameter configuration at each threshold were averaged across the datasets

Although the OD only improve the classifiers by a small amount it is important to note that the classifiers had more variations in the accuracy. Similar to regression the most important factor continues to be which classifier is used, as shown in Figure 9. Even though the variation in accuracy between the classifiers is very small, the OD methods can't improve a model to become better then another. Also, as more suspected outliers are removed then the performance continued to decrease as seen in Figure 8. Therefore, classification could be the most dependent on data points even if they are slight outliers. The removal of outliers might not change the classifier model enough to make it worth cleaning.

Max Accuracy

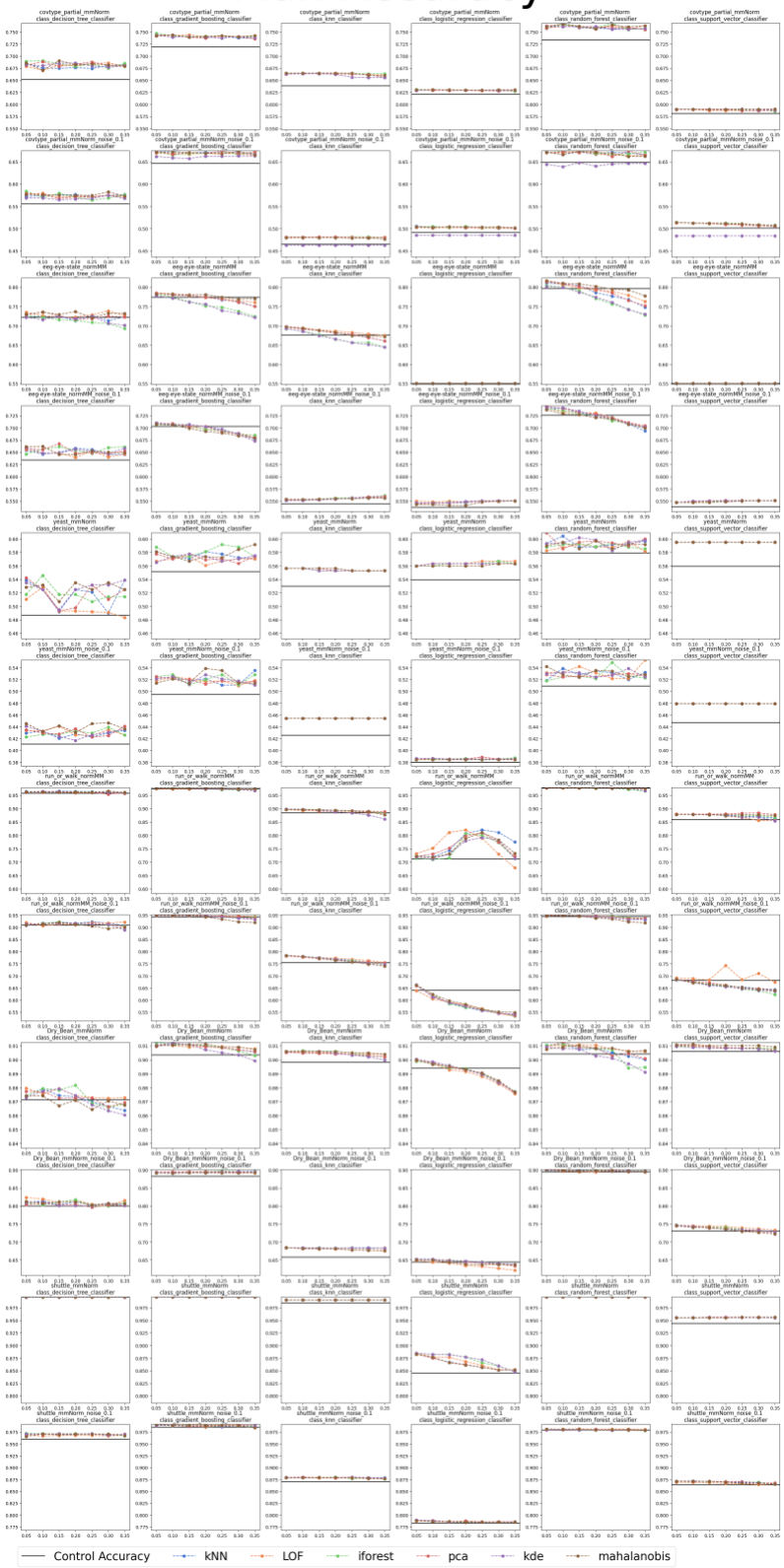

Figure 9: Accuracy of outlier detection for each dataset and classifier. Max Value across each parameter configuration.

4.3 Clustering

Similar to the supervised tasks the most important factor for clustering is still the clustering algorithm, however the OD algorithm used had more effect on the ARI and silhouette score for the clustering tasks. The impact on the silhouette score isn't as much as it is on the ARI, but it is still significant. In Figure 10 when keeping the range of y values to plot the same it is shown that the outlier detection can improve a clustering algorithm to be better than another algorithm. For example, when looking at the covtype partial dataset on the first row, we can see that the silhouette score for k means clustering and agglomerative clustering have a difference of 0.1 in their control values. But when IForest is used with k means clustering the score increases and is better than the control for agglomerative clustering.

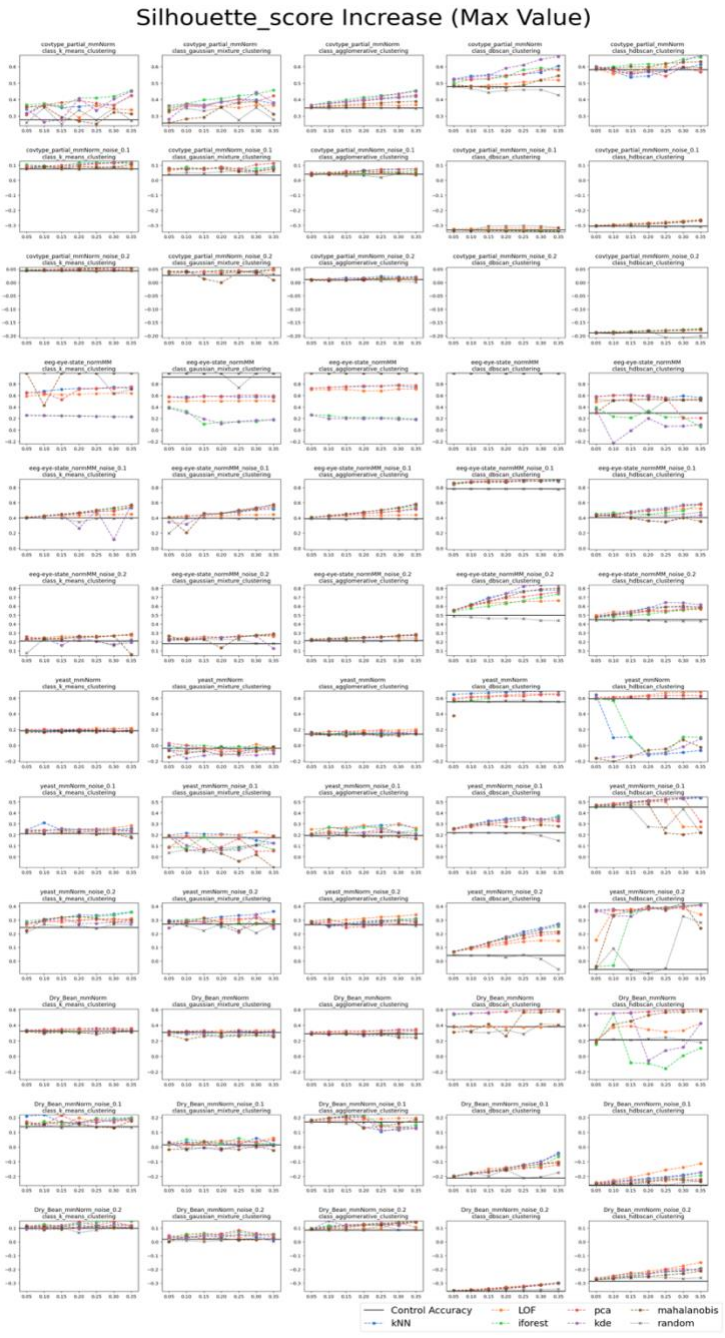

Figure 10: Silhouette Score of outlier detection for each dataset and classifier. Best values across each parameter configuration.

4.3.1 Performance vs Outliers Removed

As the number of outliers removed increased, the silhouette score also increased for many combinations. However, the silhouette score describes how close instances are to the cluster they are in, therefore the better scores may be caused by the removal of data points. With fewer data points it is less likely that instances are labeled outside of their clusters. To identify how much the removal of outliers affected the results, random data points were removed and then clustering was done on the randomly cleaned data.

As shown in Figure 11, the silhouette score for the covtype partial dataset with agglomerative clustering improved as more outliers were removed for every OD method. The random cleaning method however stays consistent with the control regardless of the amount removed. These trends are consistent with the other datasets and clustering algorithm to a varying extent and some exceptions. Therefore, because the silhouette scores increased as more outliers were removed, it appears that using OD benefits the formation of the clusters. However, the removal of any datapoint, outlier or inlier, in clustering is still detrimental. A principal use case of clustering is to predict the behavior of an instance based on how similar instances in the same cluster has behaved [16], for example in online shopping clustering can be used to predict how a buyer will act into the future based on similar buyers. If the instance that needs to be examined is treated as an outlier and removed then it is impossible to predict its behavior, but if the instance that needs to be examined isn't an outlier than there is no problem, and the removal of outliers is beneficial.

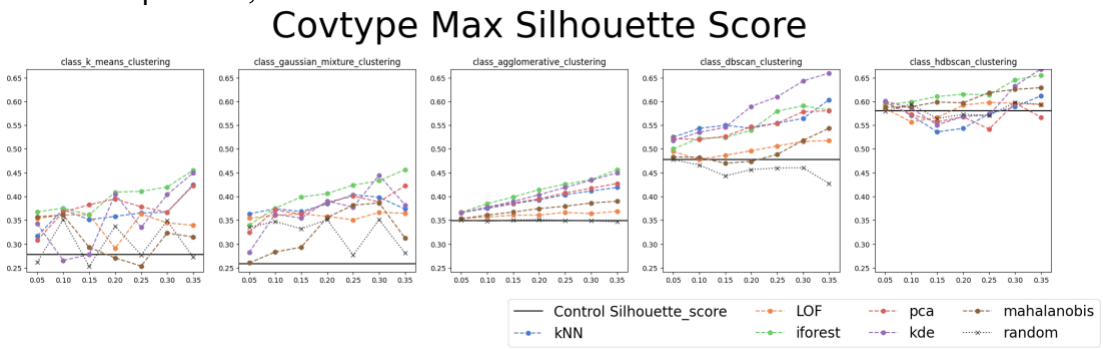

Figure 11: covtype partial silhouette score, max values across each parameter configuration.

4.3.2 Silhouette Score

When it comes to which OD algorithms benefit each clustering model the most, it is difficult to establish frontrunners because of all the variables to consider. To provide a simplification when looking at the values of a clustering model OD pair as a percent of the clustering model without OD on that dataset we can gather Figure 12.

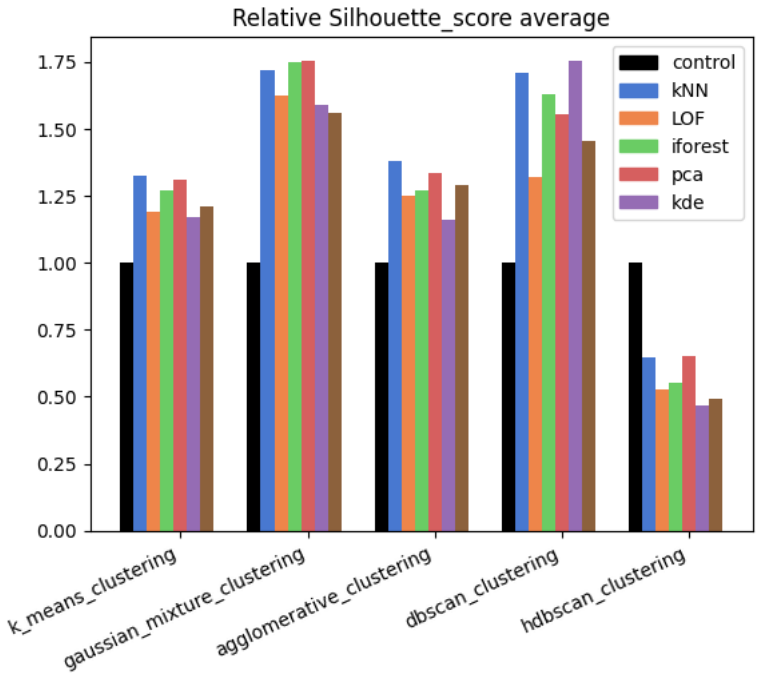

Figure 12: Relative Average Silhouette. Averaged across each dataset.

When comparing how much each OD algorithim improves relative to the control (removing no outliers) it is determined that KNN, IForest and PCA consistently improve the silhouette score the most.

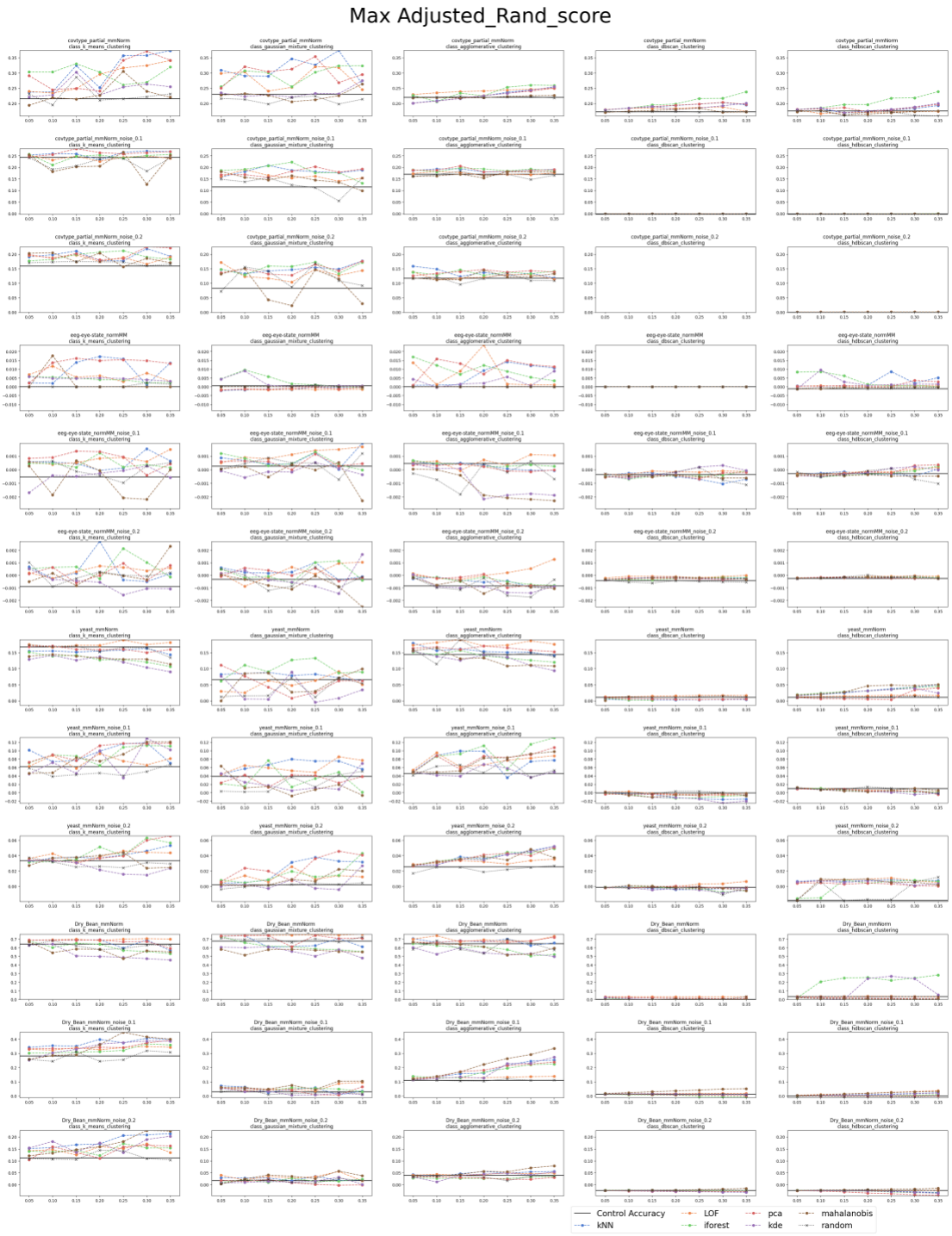

Figure 13: Adjusted rand index. max values across each parameter configuration.

4.3.3 Adjusted Rand Index

Silhouette score is a used when determining if the shape of a cluster is uniform, but it doesn't confirm whether a cluster is accurate. With the ARI the similarity between the true labels and the predicted labels can be measured.

Removing outliers affected the ARI less than it affected the silhouette score. Interestingly the ARI wasn't greatly impacted by the clustering algorithm used either. The ARI was significantly more sensitive to the clustering model used and slightly more sensitive to OD used.

When considering the ARI achieved from clustering with OD as a percent of the ARI achieved from clustering alone, it is possible to see how much each OD method improves each clustering model. Figure 13 shows this, but it isn't very clear what OD algorithms are overall best. Averaging these values across the datasets gives Figure 14.

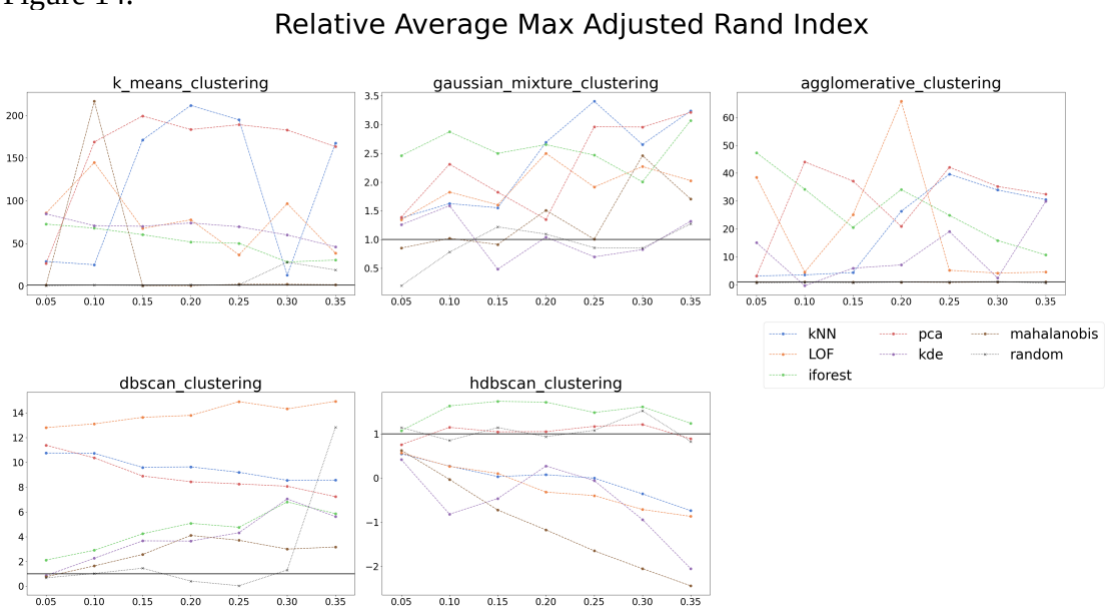

Figure 14: Adjusted rand index averaged across each dataset. Max values across each parameter configuration.

From Figure 14, we can see that, for k means clustering, PCA is the best way to improve the ARI, and KNN also improves the ARI, but it is more variable. From the gaussian mixture plot we can see that when the contamination is low then IForest is ideal. When the contamination reaches 20% it is more difficult to say what the ideal OD algorithm is. When agglomerative clustering is used, PCA seems to be the most consistently ideal cleaning method to use across all contamination thresholds. DBSCAN clustering graph shows that LOF, KNN and PCA are all good at improving ARI, but as more outliers need to be removed LOF stays consistent, but KNN and PCA decrease in performance. Finally, IForest outlier detection seems to be the only way to improv HDBSCAN. Apart from IForest and PCA, all other outlier detection models result in lowering ARIs of the clustering algorithm. They even perform worse than randomly cleaning the data. Therefore, when using HDBSCAN choosing an OD algorithm to clean the data might cause more damage.

Table 5: Summary of OD methods with Regression models, Green is the recommended OD method for the model and red is for worst OD method

OD algorithm	results	notes
Multiple Linear Regression	KNN	Slightly improved Error, Error slightly increased as threshold increased
	LOF	Little impact
	IForest	Slightly improved Error, Error slightly increased as threshold increased. Adjusted R squared sharply decreased at threshold 0.2
	PCA	Slightly Reduced Error and adjusted R squared. Error slightly increased as threshold increased. Adjusted R squared decreased as threshold increased
	KDE	Slightly improved error, Error slightly worsened as threshold increased
	Mahalanobis	error and adjusted R squared sharply rose as threshold reached ~ .2 for most datasets
Ridge Regression	KNN	Little impact
	LOF	Little impact
	IForest	Little impact
	PCA	Little impact
	KDE	Little impact
	Mahalanobis	Error Sharply increases with threshold.
Lasso Regression	KNN	Error increased. Adjusted R squared slightly improved when threshold is low
	LOF	Little impact
	IForest	Adjusted R squared improved when threshold is low
	PCA	Error increased. Adjusted R squared greatly decreased
	KDE	Error increased. Adjusted R squared slightly improved
	Mahalanobis	Error greatly increased
Elastic Net Regression	KNN	Adjusted R squared increased at high threshold
	LOF	Error decreased the most, Adjusted R squared slightly increased
	IForest	Largest increase in Adjusted R squared and rMSE
	PCA	Adjusted R squared increased at high threshold
	KDE	Adjusted R squared increased at high threshold
	Mahalanobis	Little Impact
Random Forest Regression	KNN	Error slightly improved at 0.05 threshold, Adjusted R Squared decreased
	LOF	Error slightly improved, Adjusted R Squared decreased
	IForest	Error slightly improved at 0.05 threshold, Adjusted R Squared decreased
	PCA	Error slightly improved at 0.05 threshold, Adjusted R Squared decreased
	KDE	Error slightly improved at 0.05 threshold, Adjusted R Squared decreased
	Mahalanobis	Error slightly improved at 0.05 threshold, Adjusted R Squared decreased
Support Vector Regression	KNN	Error greatly improved at high threshold, Adjusted R Squared greatly decreased
	LOF	Error Improved, Adjusted R Squared greatly decreased
	IForest	Error greatly improved, Adjusted R Squared greatly decreased
	PCA	Error greatly improved at high threshold
	KDE	Error Improved, Adjusted R Squared greatly decreased
	Mahalanobis	Error Improved, Adjusted R Squared greatly decreased

Table 6: Summary of OD methods with Classification models

OD algorithm	results	notes
Logistic Regression Classifier	KNN	Accuracy slightly improved, but improvement decreased as threshold grew
	LOF	Accuracy slightly improved, but improvement decreased as threshold grew
	IForest	Accuracy slightly improved, but improvement decreased as threshold grew
	PCA	Accuracy slightly improved, but improvement decreased as threshold grew
	KDE	Accuracy slightly improved, but improvement decreased as threshold grew
	Mahalanobis	Accuracy slightly improved, but improvement decreased as threshold grew
Decision Tree Classifier	KNN	Accuracy slightly improved
	LOF	Accuracy slightly improved
	IForest	Accuracy slightly improved
	PCA	Accuracy slightly improved
	KDE	Accuracy slightly improved
	Mahalanobis	Accuracy slightly improved
Random Forest Classifier	KNN	Accuracy slightly improved. Improvement decreased as threshold grew
	LOF	Accuracy slightly improved, but improvement decreased as threshold grew
	IForest	Accuracy slightly improved, but improvement decreased as threshold grew
	PCA	Accuracy slightly improved, but improvement decreased as threshold grew
	KDE	Accuracy slightly improved, but improvement decreased as threshold grew. Worst OD method
	Mahalanobis	Accuracy slightly improved, but improvement decreased as threshold grew
Support Vector Classifier	KNN	Accuracy slightly improved, but improvement decreased as threshold grew
	LOF	Accuracy slightly improved, but improvement decreased as threshold grew. Best OD method
	IForest	Accuracy slightly improved, but improvement decreased as threshold grew
	PCA	Accuracy slightly improved, but improvement decreased as threshold grew
	KDE	Accuracy slightly improved, but improvement decreased as threshold grew. Worst OD method
	Mahalanobis	Accuracy slightly improved, but improvement decreased as threshold grew
KNN Classifier	KNN	Accuracy slightly improved, but improvement decreased as threshold grew
	LOF	Accuracy slightly improved, but improvement decreased as threshold grew
	IForest	Accuracy slightly improved, but improvement decreased as threshold grew
	PCA	Accuracy slightly improved, but improvement decreased as threshold grew
	KDE	Accuracy slightly improved, but improvement decreased as threshold grew. Worst OD method
	Mahalanobis	Accuracy slightly improved, but improvement decreased as threshold grew

Table 7: Summary of OD methods with Clustering models

OD algorithm	results	notes
Gradient Boosting Classifier	KNN	Accuracy slightly improved, but improvement decreased as threshold grew
	LOF	Accuracy slightly improved, but improvement decreased as threshold grew
	IForest	Accuracy slightly improved, but improvement decreased as threshold grew
	PCA	Accuracy slightly improved, but improvement decreased as threshold grew
	KDE	Accuracy slightly improved, but improvement decreased as threshold grew. Worst OD method
	Mahalanobis	Accuracy slightly improved, but improvement decreased as threshold grew
K Means Clustering	KNN	Silhouette Score and ARI are improved. Silhouette Score increases as threshold increases
	LOF	Silhouette Score is improved. ARI is slightly improved
	IForest	Silhouette Score is improved. Silhouette Score increases as threshold increases. ARI is slightly improved
	PCA	Silhouette Score and ARI are improved. Silhouette Score increases as threshold increases
	KDE	Silhouette Score is improved
	Mahalanobis	Silhouette Score is slightly improved
Gaussian Mixture Clustering	KNN	Silhouette Score and ARI are improved. ARI increased as threshold is 0.20 and more
	LOF	Silhouette Score and ARI are improved
	IForest	Silhouette Score and ARI are improved
	PCA	Silhouette Score and ARI are improved. Performance increase as threshold gets over 0.25
	KDE	little to no improvements
	Mahalanobis	little to no improvements
Agglomerative Clustering	KNN	Greater improvement in ARI at higher threshold.
	LOF	Improvement in ARI
	IForest	Large improvement in ARI
	PCA	Large improvement in ARI and Silhouette score
	KDE	Little improvement in ARI and Silhouette score
	Mahalanobis	No improvement in ARI and slight improvement in Silhouette Score
DBSCAN	KNN	2nd largest improvement in ARI and silhouette score
	LOF	Largest improvement in ARI and smallest improvement in Silhouette score
	IForest	Small improvement in ARI
	PCA	Decent improvement in ARI and Silhouette score
	KDE	Small improvement in ARI and largest improvement in Silhouette score
	Mahalanobis	Low improvement in ARI and Silhouette Score
HDBSCAN	KNN	Large decrease in ARI and silhouette score
	LOF	Decrease in ARI, small improvement in silhouette score
	IForest	Improvement in ARI, Small decrease in silhouette score
	PCA	No change in ARI, decrease in silhouette score
	KDE	Large decrease in ARI and silhouette score
	Mahalanobis	Large decrease in ARI and silhouette score

Chapter 5: Conclusion

With many different OD algorithms choosing the right one for cleaning a dataset can depend on the downstream task as well as the dataset. With the results provided in this thesis it can be easier to determine the correct OD algorithm to use or whether OD is even necessary. Table 5,6,7 shows the recommended OD method for DS models in green. Mistakes can be avoided where certain OD algorithms are detrimental such as using outlier detection with random forest regression. Adequate improvements to machine learning task can be completed without sacrificing too much in time complexity for outlier detection.

Interesting discoveries derived from this study include the following. When completing a classification task, the model was more likely to be resistant to outliers, and the performance rarely improved when outliers were removed. The model would often suffer from outlier detection; therefore the model might have been reliant on the datapoints that were cleaned. For regression tasks the model would have similar performance with or without OD, unless the models are support vector regression or random forest regression in which case any OD method would help the regressor, and the IForest method would perform the best. With clustering tasks, the model would consistently perform better as outliers are removed. However, removing too many suspected outliers could be detrimental. With the clustering task different models had clear optimal OD methods to use to clean.

5.1 Future works

The work done was only a limited amount of outlier detection techniques, there are more OD algorithms to test. More complex regression clustering and classification models also need to be tested. Along with more types of datasets, images, audio, and more complex signal data needs to be examined.

The time complexity for each OD method needs to be studied as well. With the time complexity determining which method is better when they have similar results can be determined. There is no need to use a complex method when a simple one performs just as well.

Synthetic datasets will also need to be considered. With synthetic datasets it is possibly to identify how accurate the outlier detection methods are. Also, the true underlying model will be able to be compared to the predictions.

Bibliography

[1] L. Ruff *et al.*, “A unifying review of deep and shallow anomaly detection,” *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, May 2021, doi: 10.1109/jproc.2021.3052449.
[2] S. Ramaswamy, R. Rastogi, and K. Shim, “Efficient algorithms for mining outliers from large data sets,” *SIGMOD Record*, vol. 29, no. 2, pp. 427–438, May 2000, doi: 10.1145/335191.335437.
[3] F. T. Liu, K. M. Ting and Z. -H. Zhou, "Isolation Forest," 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 2008, pp. 413-422, doi: 10.1109/ICDM.2008.17. keywords: {Application software;Credit cards;Detectors;Constraint optimization;Data mining;Information technology;Laboratories;Isolation technology;Performance evaluation;Astronomy;anomaly detection;outlier detection;novelty detection;isolation forest;binary trees;model based},
[4] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, ‘The Mahalanobis distance’, *Chemometrics and Intelligent Laboratory Systems*, vol. 50, no. 1, pp. 1–18, 2000.
[5] R. Todeschini, D. Ballabio, V. Consonni, F. Sahigara, and P. Filzmoser, “Locally centred Mahalanobis distance: A new distance measure with salient features towards outlier detection,” *Analytica Chimica Acta*, vol. 787, pp. 1–9, Jul. 2013, doi: 10.1016/j.aca.2013.04.034.
[6] L. J. Latecki, A. Lazarević, and D. Pokrajac, “Outlier Detection with Kernel Density Functions,” in *Lecture notes in computer science*, 2007, pp. 61–75. doi: 10.1007/978-3-540-73499-4_6.
[7] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using Real NVP,” *arXiv (Cornell University)*, May 2016, doi: 10.48550/arxiv.1605.08803.
[8] N. Kumar, P. Hanfeld, M. H. Hecht, M. Bußmann, S. Gumhold, and N. Hoffmann, “InFlow: Robust outlier detection utilizing Normalizing Flows,” *arXiv (Cornell University)*, Jun. 2021, doi: 10.48550/arxiv.2106.12894.
[9] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, “A comparative evaluation of outlier detection algorithms: Experiments and analyses,” *Pattern Recognition*, vol. 74, pp. 406–421, Feb. 2018, doi: 10.1016/j.patcog.2017.09.037.
[10] A. Bramm, S. Eroshenko and A. Khalyasmaa, "Effect Of Data Preprocessing on the Forecasting Accuracy of Solar Power Plant," 2021 XVIII International Scientific Technical Conference Alternating Current Electric Drives (ACED), Ekaterinburg, Russia, 2021, pp. 1-5, doi: 10.1109/ACED50605.2021.9462288. keywords: {Measurement;Analytical models;Renewable energy sources;Linear regression;Machine learning;Predictive models;Data models;regression;forecasting;renewable energy source;accuracy metrics},
[11] J. Almeida, L. M. S. Barbosa, A. a. C. C. Pais, and S. J. Formosinho, “Improving hierarchical cluster analysis: A new method with outlier detection and automatic clustering,” *Chemometrics and Intelligent Laboratory Systems (Print)*, vol. 87, no. 2, pp. 208–217, Jun. 2007, doi: 10.1016/j.chemolab.2007.01.005.
[12] Markelle Kelly, Rachel Longjohn, Kolby Nottingham,

The UCI Machine Learning Repository, https://archive.ics.uci.edu
[13] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: networked science in machine learning. SIGKDD Explorations 15(2), pp 49-60, 2013.
[14] N. Vafaei, R. A. Ribeiro, and L. M. Camarinha-Matos, "Comparison of normalization techniques on data sets with outliers," *International Journal of Decision Support System Technology*, vol. 14, no. 1, pp. 1–17, Nov. 2021, doi: 10.4018/ijdsst.286184.
[15] F. Pedregosa *et al.*, 'Scikit-learn: Machine Learning in Python', *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[16] A. B. Dymnicki and D. B. Henry, "Use of Clustering Methods to Understand More about the Case," *Methodological Innovations Online*, vol. 6, no. 2, pp. 6–26, Aug. 2011, doi: 10.4256/mio.2010.0033.