**FT Algorithms**

**Objective**

The algorithm should screen for anomaly or outlier or deviant or unusual data point. (An outlier is a data point that deviates so much from other data points as to arouse suspicion that it was generated by not following accepted practices/rules).

**Requirement**

-Definition of anomaly based on historical/future expected activity

-Definition of what is considered normal

-Training/evaluation and test data

-Ground truth of established labels and future considerations

-Accuracy metrics (Precision-recall, ROC, F1 score, threshold for alert, confusion matrix, kappa etc.)

-Strategy to handle false positives, false negatives, etc.

**Approach**

The type of algorithm can be supervised, unsupervised, semi-supervised, rule based, statistical or using deep learning technique or any combinations. The goal is to construct a right data model & algorithm for Vanguard Group to separate outliers from noise and normal data.

All anomaly detection programs can fall into these broad categories or any of its combinations.

* Rule Based

-used for smaller/medium sized data sets

* Statistical

-for smaller/medium sized data sets)

* Deep learning

-for very complex, large data sets)

Depending on how labels are defined, the algorithms can also be categorized as:

* Unsupervised(train on all data without labels)

**-**preferred for evaluating new and unknown anomalies.

**-**Used for validating or exploratory analysis of existing labels.

* Semi-supervised (train on filtered ‘normal class’ data only)

-preferred for novelty detection.

-Use when normal classes are well defined and other data points that do not belong to normal class are considered anomaly.

* Supervised (train on data with labels)

Preferred when there are no new or unknown anomalies and all anomalies are known and captured by the labels.

**Challenges with Outlier Identification**

1. **Class Imbalance**: Proportion of anomalous trades among total trades could be extremely small. In some cases, these issues can be tackled by resampling (under-sampling, over-sampling), or by generating synthetic samples. Resampling can be done with or without replacement.

**1.1 Oversampling:**

Oversampling can be achieved by adding similar instances of underrepresented class to balance the skewed class ratio. The most common technique to address oversampling is known as SMOTE (Synthetic Minority Over-sampling Technique).

**1.2 Undersampling:**

Undersampling can be achieved by removing the instance from the overrepresented class. The most common technique selects the K-Nearest Neighbors (KNN’s) and discards the remaining data.

**1.3 Synthetic Sampling:**

When relevant data is scare, synthetic sampling may help. These datasets mirror the distribution of the original sample. SMOTE and ADASYN techniques are commonly used for generating synthetic data.

1. **Accuracy Paradox:**

When trained on an imbalanced data, the model will most likely predict the dominant class. Therefore, the following metrics are used in evaluating anomaly.

**Confusion Matrix:** A breakdown of predictions into a table showing correct predictions (the diagonal) and the types of incorrect predictions made (the classes incorrect predictions were assigned)

**Precision:** A measure of a classifiers exactness

**Recall:** A measure of a classifier completeness

**F1 Score:** A weighted average of precision and recall

**Kappa:** Classification accuracy normalized by the imbalance of the classes in the data.

**ROC Curves:** Like precision and recall, accuracy is divided into sensitivity and specificity and models can be chosen based on the balance thresholds of these values. The Receiver Operating Characteristic (ROC) curve is a standard technique for summarizing classifier performance over a range of tradeoffs between true positive and false positive error rates. ROC curves can be thought of as representing the family of best decision boundaries for relative costs of true positives and false positives.

**AUC Curves:** The Area Under Curve (AUC) is an accepted traditional performance metric for a ROC curve.

1. **Class Overlap:** anomalous trades could be disguised as normal trades through various schemes.
2. **Class Drift:** tactics for anomalous trade evolves over time and the labels from current or historical data may not capture it.
3. **Clustering algorithms:**

**5.1 Optimized for Cluster Identification:**

Most of the clustering algorithms are optimized to find clusters rather than outliers. Accuracy of outlier detection depends on how good the clustering algorithm captures the structure of clusters. A set of many abnormal data objects that are similar to each other would be recognized as a cluster rather than noise or outlier.

**5.2 Assumptions affecting the results:**

Clustering algorithms factor in various assumptions like

-data is generated by one generating mechanism or one statistical process.

-anomalies are rare observations etc.

These assumptions deeply affect the results if one doesn’t take the time to understand the data and how it is generated.

In the last few years, plethora of algorithms have been developed to tackle various assumptions.

1. **High dimensionality**

Outliers can be **univariate** or m**ultivariate.** Univariate outliers can be found when looking at a distribution of values in a single feature space. Multivariate outliers can be found in a

n-dimensional space (of n-features). Looking at distributions in n-dimensional spaces can be very difficult for the human brain, therefore we use algorithms.

Most outlier detection techniques operate in the observational space, which is often associated with information redundancy, high correlations, measurement errors and noise. Also, due to the usually high dimensionality of the observational space, the anomalies may or may not be visible or observed. Even when detected in high dimensional space, they are difficult to explain. Therefore, for complex and high dimensional data, latent variables (e.g latent variable like ‘neighborhood’ - in housing data containing proximity to parks, school district or crime rate etc.) are extracted in the lower dimensional space to observe outliers.

1. **Feature Selection**

Feature selection methods are used to identify and remove unneeded, irrelevant and redundant attributes (columns) to reduce overfitting, improve accuracy and reduce training time. Feature selection is different from dimensionality reduction. Both methods seek to reduce the number of attributes in the dataset, but a dimensionality reduction method do so by creating new combinations of attributes.

There are three classes of algorithms for feature selections:

Filter Methods

This method applies a statistical measure to assign a score to each feature. It is often univariate and consider the feature independently, or with regard to the dependent variable. Some examples include the Chi squared test, information gain and correlation coefficient scores.

Wrapper Methods

It is a search method where different combinations of features are prepared, evaluated and compared to other combinations. The search process could be best-first or stochastic ( a random hill climbing algorithm, heuristics like forward and backward passes)

Embedded Methods

It learns features that best contribute to the accuracy while the model is being created. One such common method is called regularization. Examples of regularization algorithms are LASSO, Elastic Net and Ridge Regression.

1. **Labelling or Scoring**

For supervised method, which requires labels,

-data objects are labelled as normal or outlier (integer or one hot encoding)

The labels can be

1. pre-determined or known.

2. determined through other unsupervised algorithm and then used for prediction.

3. determined though other unsupervised algorithm and then validated with ground truth and then refined.

For unsupervised method (which has no labels) involve

1. exploratory determination of labels through various algorithms

2. finding an outlier score (e.g. the probability for being an outlier). Data objects can be sorted according to scores to focus on top outliers. Further, these scores can be converted into binary output with a threshold value.

**8.1 Recommended Labelling Approach 1**

Step 1

Use unsupervised or semi-supervised clustering algorithms to evaluate labels (outlier identification).

Step 2

Compare and refine the labels with ground truth and use supervised algorithms for screening on an ongoing basis.

**Project Stages**

**Step 1**:

Evaluate various clustering algorithms for outlier identification and its relevancy for Vanguard’s use case. Perform exploratory data analysis.

**Step 2**:

Perform in-depth analysis of FT data. Identify features for Vanguards use case. And prepare the data for algorithms.

**Step 3**:

Develop various relevant algorithms

.

**Step 4**:

Run various algorithms with data and understand the nature of the anomaly from the data.

**Step 5**:

Compare the identified outliers from the algorithms with the ground truth

**Step 6**:

Refine or reconstruct labels for outliers in the training/evaluation/testing data set

**Step 7**:

Build a new algorithm to screen for anomaly on an ongoing basis

**Step 8:**

Train with the new algorithm, evaluate and fine tune parameters

**Step 9:**

Test against unseen data

**Step 10:**

Continue the iterative process until the accuracy metrics are met.

**Types of algorithms**

1. **Distance Based**

K-nearest neighbors

K-Means

K-Means++

Mini Batch K-Means

K-Medians

Fuzzy-K-Means

K-Medoids

Hierarchical Clustering

Agglomerative Clustering

Divisive Clustering

Regression Hyperplane Distance

1. **Density Based**

Density Based Spatial Clustering Algorithm with Noise (DBSCAN)

Hierarchical DBSCAN (HDBSCAN)

Local Outlier Factor (LOF)

Mean Shift

1. **Tree Based**

Isolation Forest (IF)

1. **Parametric**

Gaussian Mixture Model (GMM)

Single/One Class SVMs

Extreme Value Theory

Z-score

Mean/Median Absolute Deviation (MAD)

1. **Graph based**

Affinity Propagation

Spectral Clustering

Visual Graphical Outlier Inspection

1. **Deep Learning**

Auto-encoders with tensorflow

Principle Component Analysis (PCA) with tensorflow

Non-Negative Matrix Factorization (NMF) or Bayesian factorization with tensorflow

Self-Organizing Map (SOM) and Particle Swarm Optimization(PSO)

1. **Other**

Other variations or combinations of the above

-Principle Component Analysis (PCA) with Logistic Regression

-Define labels using clustering algorithms (e.g K-means) and then use Auto-encoders with tensorflow for screening in production

1. **Distance Based Algorithms**

Assumptions:

-Normal data objects are closer in a neighborhood

-Outliers are far apart from their neighbors

**1.1 K-Nearest Neighbors (KNN)**

Algorithm:

KNN is a simple, non-parametric lazy learning technique used to classify data based on similarities in distance metrics such as Eucledian, Manhattan, Minkowski or Hamming distance.

To find outliers:

-compute the average distance to its KNN for each point.

-sort the values

-choose the biggest values as outliers

Majority of the voting class determines the cluster. When data is skewed, it presents a challenge for anomaly detection. Normal classes will dominate any new points. One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its K-nearest neighbors. The class of each of the K-nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point. Another way to overcome skew is by abstraction in data representation. As we discuss later in this document, in the self-organizing map’ (SOM), each node is a representative (a center) of a cluster of similar points, regardless of their density in the original training data. KNN can then be applied to the SOM.

**1.2 K-Means**

Algorithm:

It groups all data points to each one of the ‘K’ clusters. These clusters have similar variance and they are derived by minimizing a specific cost function: *sum of squares distance within the cluster*.

1. Number of clusters are chosen
2. Randomly guess K cluster center locations
3. Each data point finds out which center it is closest to
4. Each center finds the centroid of the points in owns
5. K cluster center jumps to the new centroid and repeats steps 3 and 4
6. Step 5 is repeated until it is terminated

The k-means algorithm makes all-or-nothing assignments of data points to clusters, and these hard decisions can often lead to poor solutions corresponding to local minima. Also it is very sensitive to initialization therefore prone to local minima.

**1.3 K-Means++**

Algorithm:

Randomly chooses one centroid from data points using weighted probability distribution proportional to the distance square.

Speeds up convergence.

**1.4 Mini Batch K-Means**

Algorithm:

Variant of K-means where mini-batches are used to reduce computation time.

Speeds up convergence but the quality of the results are compromised.

**1.5 K-Medians**

Algorithm:

Variation of K-means where the centroid is calculated using median instead of mean. This technique minimizes error over all clusters with respect to 1-norm distance metric as opposed to the square of the 2-norm distance as in K-means.

**1.6 Fuzzy-K-Means**

Algorithm:

It is a soft clustering technique where data points can potentially belong to multiple clusters. Membership grades are assigned to each of the data points. These membership grades indicate the degree to which data points belong to each cluster. Thus, points on the edge of a cluster, with lower membership grades, may be in the cluster to a lesser degree than points in the center of cluster

**1.7 K-Medoids**

Algorithm:

It is related to the K-means algorithm and the medoid-shift algorithm. A medoid can be defined as the object of a cluster whose average dissimilarity to all the objects in the cluster is minimal. i.e. it is a most centrally located point in the cluster. In contrast to the K-means algorithm, K-medoids chooses data points as centers (medoids or exemplars) and works with a generalization of the L1 Manhattan Norm to define distance between data points instead L2 like in K-means. It is more robust to noise and outliers as compared to K-means because it minimizes a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances.

**1.7.1 Partitioning Around Medoids(PAM)**

The most common realization of K*-*medoid clustering is the Partitioning Around Medoids algorithm and is as follows:

1. Initialize: randomly select ‘k’ of the ‘n’ data points as the medoids
2. **Assignment step**: Associate each data point to the closest medoid.
3. **Update step**: For each medoid ‘*m’* and each data point ‘*o’* associated to ‘*m’* swap ‘*m*’ and ‘*o*’ and compute the total cost of the configuration (that is, the average dissimilarity of ‘*o*’ to all the data points associated to ’*m’*).

Select the medoid ‘*o*’ with the lowest cost of the configuration.

Repeat alternating steps 2 and 3 until there is no change in the assignments.

**Pros of K-Means type of algorithms:**

It is a sort of simple partitioning algorithm rather than a clustering focussed algorithm. With right tricks and optimizations, it can be made efficient for exploratory data analysis.

For a large dataset, K-Means gives the best performance over other clustering algorithms with limited accuracy.

**Cons of K-Means type of algorithms:**

All K-means with L2 distance based algorithms are incapable of handling noise data and outliers as all data points become part of a cluster.

Work around to detect outlier is to establish a distance threshold (average distance for each cluster) for judging whether a data point is an outlier or not. However, this technique will not work if there are too many outliers.

Need to choose right ‘K’ thus affecting cluster quality. In supervised method, when ground truth is available in training data, check for homogeneity (cluster contains only member of single class) and Completeness (all members of a given class are assigned to the same cluster). In unsupervised method when ground truth is not available, use Silhouette Coefficient to evaluate the compactness and separation of the clusters.

K-means is incapable of handling non-spherical data. As a distance based algorithm, it assumes data distribution is spherical or isotropic (equal variance in all directions). Performance of K-means will degrade with non-spherical or anisotropic data.

**1.8 Hierarchical Clustering**

Algorithm:

Hierarchical clusters are good for finding outlier groups. Outliers take longer to join other groups during iteration.

* + 1. **Agglomerative Clustering**:

It is a bottoms up approach. It groups similar data points based on distance-matrix or raw data into clusters. It starts by treating each data points as a separate cluster.

1. Identifies two clusters that are together.
2. Merges the two most similar clusters.
3. Continue step 1 and step 2 until all the clusters are merged.

**1.8.1.1 Single link clustering:**

It is a basic version of Agglomerative Clustering. It chooses the closest cluster to merge based on the ranked distance of the tree when the clusters merged or split.

**1.8.1.2 Complete/Complex link Clustering**

More complex versions use mean or maximum distance between clusters or distance between cluster centroid etc. to determine which cluster to merge. Once the cluster hierarchy is determined, usually a cut or level is chosen to prune the tree based on the ‘number of clusters’ parameter. A dendrogram view of cluster will reveal how the clusters are formed.

**1.8.1.3 Ward’s Clustering**:

When clusters are merged in a hierarchical clustering, sometimes Ward’s method is used to evaluate the merging cost in terms of the distance between two clusters. It starts with zero (because every point is in its own cluster) and then grows as clusters merge. Ward’s method aims to keep this growth small by preferring to merge with clusters with smaller merging cost. Ward’s method can indirectly be used to evaluate the number of clusters by stopping the merging when costs go too high.

**Pros of Agglomerative Clustering:**

No assumption on the number of clusters. It is a repeatable and stable over many runs.

It is a partitioning algorithm therefore includes noise and outliers within the clusters

Dendrogram can be used to view how parameters affect cluster formation.

**Cons of Agglomerative Clustering:**

Higher complexity (quadratic). Not good for circular or hyper-spherical clusters.

Bottom-up methods make clustering decisions based on local patterns without initially taking into account the global distribution.

* + 1. **Divisive Clustering**:

It is a top down approach. Initially, all data points are in the same cluster, and the largest cluster is split until every data point is separate. It chooses the data points with the maximum average dissimilarity and then moves all data points to this cluster that are more similar to the new cluster than to the remainder. Many of the algorithms discussed previously in agglomerative clustering can be used for divisive clustering also with slight modification.

**Pros of Divisive Clustering:**

Top-down clustering benefits from complete information about the global distribution when making top-level partitioning decisions.

**Cons of Divisive Clustering:**

For estimating large numbers of clusters, this approach is both slow (due to all observations starting as one cluster, which it splits recursively) and statistically ill-posed.

**1.9 Regression Hyperplane Distance**

Algorithm:

In this case, any regression algorithm can be used to determine the hyperplane. Points that are further away from the regression hyperplane could be categorized as an outlier.

**Cons of Regression Hyperplane Distance:**

Noise and outlier data points could affect the formation of the regression plane. Therefore, it is not a robust technique for screening outliers.

1. **Density Based Algorithms**

Assumptions:

-Normal data objects have similar density around its neighbors.

-Outliers density is considerably different around its neighbors.

**2.1 DBSCAN (Density Based Spatial Clustering Algorithm with Noise)**

Algorithm:

DBSCAN works by greedily agglomerating points that are close to each other. Clusters with few points in them are considered outliers.

* a parameter ‘ε’ that specifies a distance threshold under which two points are considered to be close; and
* ‘MinPts’: the minimum number of points that have to be within a point’s ‘ε’-radius before that point can start agglomerating.
* Randomly chooses a point ‘p’.
* Retrieves all points density-reachable from ‘p’ with respect to ‘ε’ and ‘MinPts’
* If ‘p’ is a core point, a cluster is formed
* If ‘p’ is a border point, no points are density-reachable from ‘p’, then visit the next point.
* Repeat the process until all the data points have been processed.

**Pros of DBSCAN:**

DBSCAN is good at finding outliers or noise without requiring any hacks. No need to specify the number of clusters a priori. Capable of handling non-spherical data. or clusters of any shape.

No assumption on distribution of the data. Works well with multidimensional data for non-sparse data sets. Visualizing the results are easy and it is very intuitive.

It is stable over many runs if parameters are not changed. Can work with a very large dataset.

**Cons of DBSCAN:**

It is very sensitive to parameters ‘ε’ and ‘MinPts’

Sparse data sets may not work well and may lose efficiency on higher dimensional data. Values in the feature space need to be scaled. It is an unsupervised model, needs to be recalibrated each time with newer dataset.

**2.2 HDBSCAN (Hierarchical DBSCAN)**

Algorithm:

An evolved version of DBSCAN, which attempts to allow for clusters of differing variances and densities. HDBSCAN requires only ‘minimum cluster size’ as a parameter. Like in agglomerative clustering, every data point starts as part of its own cluster and iteratively clusters with the next nearest data points until all data points are clustered together. The minimum cluster size parameter tosses out clusters that fall below the threshold.

**Pros of HDBSCAN:**

Do not require number of clusters to be set a priori and allows for noisy data

**Cons of HDBSCAN:**

Relies on the density of data points

**2.3 Mean Shift**

Algorithm:

Mean shift clustering using a flat kernel aims to discover “blobs” in a smooth density of samples. It is a centroid-based algorithm, which works by updating candidates for centroids to be the mean of the points within a given region. These candidates are then filtered in a post-processing stage to eliminate near-duplicates to form the final set of centroids. Seeding is performed using a binning technique for scalability

It is a non-parametric feature-space analysis technique for locating the maxima of a density function.

1. Initialize random seed and window
2. Calculate center of gravity(the mean) of the window
3. Shift the search window to the mean
4. Repeat step 2 until convergence

**2.4 Local Outlier Factor (LOF)**

Algorithm:

The LOF is based on a concept of a local density, where locality is given by KNN, whose distance is used to estimate the density. By comparing the local density of a data point to the local densities of its neighbors, one can identify regions of similar density, and points that have a substantially lower density than their neighbors. These are considered outliers. The local density is estimated by the typical distance at which a point can be ‘reached’ from its neighbors. The definition of ‘reachability distance’ used in LOF is an additional measure to produce more stable results within clusters.

The LOF algorithm uses a user-defined parameter, MinPts, that determines the local neighborhood used for computing the outlier factor for each data point. The outcome of the algorithm strongly depends on this setting. One of the disadvantages of the LOF algorithm is that it is hard to tune the MinPts parameter. ILOF (Incremental Local Outlier Factor) is a modification of LOF that can handle large data streams and compute local outlier factors on-the-fly.

Local outlier detection approaches, however, are bound to overlook outliers when the data is a mixture of high-dimensional data points drawn from different data distributions. In those specific cases, a global neighborhood approach is taken to enable the discovery of outliers that would otherwise be left undetected. The purpose of using global neighborhoods is to assess the degree of outlier-ness of a given data point relative to other data points belonging to the same mixture component, avoiding the possibility that outliers can hide among members of other components of the mixture distribution.

1. **Tree Based algorithms**

Assumption:

Without any assumption on the data distribution, Isolation-Forest based approach measures how isolated a point is from others. Its basic principle is that outliers are few and far from the rest of the observations.

**3.1 Isolation Forest (IF)**

Algorithm:

Isolation Forest explicitly identifies anomalies instead of profiling normal data points. Isolation Forest, like any tree ensemble method, is built on the basis of decision trees. In these trees, partitions are created by first randomly selecting a feature and then selecting a random split value between the minimum and maximum value of the selected feature. In principle, outliers are less frequent than regular observations and are different from them in terms of values (they lie further away from the regular observations in the feature space). That is why by using such random partitioning they should be identified closer to the root of the tree (shorter average path length, i.e., the number of edges an observation must pass in the tree going from the root to the terminal node), with fewer splits necessary. Isolation Forest can be scaled up to handle large and high dimensional datasets.

Each observation is given an anomaly score and the following decision can be made on its basis:

Score close to 1 indicates anomalies

Score much smaller than 0.5 indicates normal observations

If all scores are close to 0.5 than the entire sample does not seem to have clearly distinct anomalies

**Pros of Isolation Forest**

-This method is optimized for outlier detection rather than cluster formation. No need to scale the values in the feature space. No assumption on data distributions. It has fewer parameters, this makes this method fairly robust and easy to optimize.

-Traditional anomaly detection techniques are resource intensive, computationally complex and expensive. Isolation Forest based techniques have the advantage of low complexity. Therefore, an ideal fit for large scale datasets residing in cloud based infrastructure.

-it employs a concept of isolation rather than any density or distance metric.

**Cons of Isolation Forest**

-Algorithm is available only in development version of scikit-learn

-Visualizing result is complicated

-if not correctly optimized, training time can be very long and computationally expensive.

1. **Parametric Algorithms**

Assumption:

Distribution of the data follow a probabilistic or statistical model.

**4.1 Gaussian Mixture Model (GMM)**

Algorithm:

GMM takes the dataset and tries to fit a number of possible normal distributions by estimating their parameters using ‘Expectation Maximization’. It is iteratively performed in two steps.

1. Expectation:

Generative a number of distributions with reasonable parameters (mean, variance and covariance). Then determine how each data point will fit into those distributions. One needs to specify the number of clusters. This is called ‘components ’ a priori.

1. Maximization:

Iterate and update distributions parameters (mean, variance and covariance) to maximize the likelihood that more data points can be assigned to a probable distributions (cluster).

The result of this is that each cluster is associated not with a hard-edged sphere, but with a smooth Gaussian model. Just as in the *k*-means expectation–maximization approach, this algorithm can sometimes miss the globally optimal solution, and thus in practice multiple random initializations are used.

**Gaussian mixture models** are a probabilistic model for representing [normally distributed](https://brilliant.org/wiki/multivariate-normal-distribution/) subpopulations within an overall population. [Mixture models](https://brilliant.org/wiki/mixture-model/) in general don't require knowing which subpopulation a data point belongs to, allowing the model to learn the subpopulations automatically. Since subpopulation assignment is not known, this constitutes a form of [unsupervised learning](https://brilliant.org/wiki/unsupervised-learning/).

**Pros of GMM:**

GMM allows data to vary anisotropically and provides probability estimates of cluster membership rather than “hard labelling” data points like K-Means.

**Cons of GMM:**

GMM still assumes normal distributions across dimensions and requires the number of clusters are specified a priori.

Both K-Means and GMMs include every data point in a cluster no matter how far away it is from the nearest centroid. Therefore, it is hard to find noise or outliers.

When one has insufficiently many points per mixture, estimating the covariance matrices becomes difficult, and the algorithm is known to diverge and find solutions with infinite likelihood unless one regularizes the covariances artificially.

This algorithm will always use all the components it has access to, needing held-out data or information theoretical criteria to decide how many components to use in the absence of external cues.

Expectation maximization (EM) of GMM involves learning the co-variances (instead of fixing them), learning mixing weights on class priori (instead of assuming), and to account for cluster assignment uncertainty by using the Expectation-Maximization (EM) algorithm and representing it with a simple distribution. Cluster assignments can be determined by minimizing the Kullback-Leibler divergence.

For high-dimensional input data, learning the full covariance matrix involves scalar parameters, which can be cumbersome and potentially be a cause of overfitting. For this reason, the Gaussians are often assumed to have diagonal covariance matrices (in which case the off-diagonal elements of covariance matrix are zeroed during updates) or even isotropic covariances. Class assignments can be easily read from the Q-distribution.

**4.2 Single/One Class SVMs(Support Vector Machines)**

Algorithm:

An SVM model is based on dividing the training sample points into separate categories by as wide a gap as possible, while penalizing training samples that fall on the wrong side of the gap. The SVM model then makes predictions by assigning points to one side of the gap or the other.

Single/One class SVM is trained on one class which is the ‘normal’ class. However, since SVM decision boundaries are soft, it can be used for unsupervised as well. It is able to detect outliers with higher accuracy where enough outlier data is not available. It is great for novelty detection too. If outliers are included in the training data, it does not perform well for outlier detection.

**4.3 Hidden Markov (HMM)**

Algorithm:

The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multidimensional) probability distribution. Transitions among the states are governed by a set of probabilities called ‘transition probabilities’. In a particular state an outcome or observation can be generated, according to the associated probability distribution. It is only the outcome, not the state visible to an external observer and therefore states are ``hidden'' to the outside.

Sometimes Hidden Markov Model is used to improve performance over One/Single class SVM in latent anomaly detection. In one of the method employing HMM, a behavioral distance between two states is computed. If it is greater than the threshold value, then it is considered an outlier. The behavioral distance between two processes is a measure of the deviation of their behaviors.

The hidden Markov model can be represented as the simplest dynamic Bayesian network.

There are three fundamental problems for HMMs:

-Given the model parameters and observed data, estimate the optimal sequence of hidden states.

-Given the model parameters and observed data, calculate the likelihood of the data.

-Given just the observed data, estimate the model parameters.

The first and the second problem can be solved by the dynamic programming algorithms known as the Viterbi algorithm and the Forward-Backward algorithm, respectively. The last one can be solved by an iterative Expectation-Maximization (EM) algorithm, known as the Baum-Welch algorithm.

In high dimensional space, noise or measurement error can pollute outliers. Evaluating in multiple subspace will make genuine outliers stand out and they can be explained easily.

**4.4 Extreme Value Theory**

Algorithm:

It seeks to assess, from a given ordered sample of a given random variable, the probability of events that are more extreme than any previously observed. The extreme value distribution is often used to model extreme events. The extreme values of any distribution have nearly the same distribution (called Extreme Value Distribution). EVT is known for finding anomalies in a streaming univariate time series data.

Extreme value distribution has the probability density function which is dependent on location and scale parameter. Successful version of Extreme Value Theory do not always require to set thresholds and do not make assumptions on the distribution. The main parameter is the risk thereby controlling the number of false positives.

**4.5 Z-score**

Algorithm:

Z-score is a measure of how many standard deviations below or above the mean. Higher z-score can be a potential outlier. However, it is only applicable if data points approximate a normal distribution and are assumed to fit a probabilistic model. It is recommended to generate a normal probability plot of data before applying an outlier test. Modified Z-score is a variation of Z-score adjusted for MAD (Median absolute deviation).

Pros of Z-score

It is only recommended for smaller sample sizes where the data is uni-modal(single peak).

Cons of Z-Score

Outliers in the data affect the mean and standard deviations, therefore this technique is highly unreliable. In some cases, sequentially removing worst outliers or using quantiles can mitigate the reliability issues.

**4.6 Mean/Median Absolute Deviation (MAD)**

Algorithm:

Similar to Z-score and like z-score, it is applicable to only one-dimensional data with normal distribution. Outliers in the data affect the mean and standard deviation, therefore this technique is highly unreliable

1. **Graph based algorithms**

Assumption:

The notion of similarity exists between all pairs of data points. Thresholds for similarity can be used to graph similarities.

**5.1 Affinity Propagation**

Algorithm:

It is a graph based approach that lets points ‘vote’ on their preferred ‘exemplar’. The end result is a set of cluster ‘exemplars’ from which we derive clusters by essentially doing what K-Means does by assigning each point to a cluster of its nearest exemplar.

In many situations, data is better and more easily characterized by a measure of pairwise similarities rather than defaulting to some distance criteria and in this case, clusters can instead be represented by an “exemplar” data point rather than domain-specific parameters. The affinity propagation clustering algorithm defines messages that are exchanged between data points indicating the ‘affinity’ each point has for another to act as its exemplar.

Affinity propagation is based on the concept of "message passing" between data points.Similar to *K*-medoids, affinity propagation finds "exemplars", members of the input set that are representative of clusters. But unlike K-medoids, Affinity propagation simultaneously considers all data points as possible exemplars, exchanging real-valued messages between them until a high-quality set of exemplars (and corresponding clusters) emerges.

As an input, the algorithm requires us to provide two sets of data:

1. ***Similarities*** between data points, representing how well-suited a point is to be another one’s exemplar. If there’s no similarity between two points, as in they cannot belong to the same cluster, this similarity can be omitted or set to -Infinity depending on implementation.
2. ***Preferences***, representing each data point’s suitability to be an exemplar. We may have some a priori information which points could be favored for this role, and so we can represent it through preferences.

Both similarities and preferences are often represented through a single matrix, where the values on the main diagonal represent preferences. Matrix representation is good for dense datasets. Where connections between points are sparse, it is more practical not to store the whole n x n matrix in memory, but instead keep a list of similarities to connected points. Behind the scene, ‘exchanging messages between points’ is the same thing as manipulating matrices, and it’s only a matter of perspective and implementation.

The algorithm then runs through a number of iterations, until it converges. Each iteration has two message-passing steps:

1. Calculating responsibilities***: Responsibility*** r(i, k) reflects the accumulated evidence for how well-suited point k is to serve as the exemplar for point i, taking into account other potential exemplars for point i. Responsibility is sent from data point i to candidate exemplar point k.
2. Calculating availabilities: ***Availability*** a(i, k) reflects the accumulated evidence for how appropriate it would be for point i to choose point k as its exemplar, taking into account the support from other points that point k should be an exemplar. Availability is sent from candidate exemplar point k to point i.

In order to calculate responsibilities, the algorithm uses original similarities and availabilities calculated in the previous iteration (initially, all availabilities are set to zero). Responsibilities are set to the input similarity between point i and point k as its exemplar, minus the largest of the similarity and availability sum between point i and other candidate exemplars. The logic behind calculating how suitable a point is for an exemplar is that it is favored more if the initial a priori preference was higher, but the responsibility gets lower when there is a similar point that considers itself a good candidate, so there is a ‘competition’ between the two until one is decided in some iteration.

Calculating availabilities, then, uses calculated responsibilities as evidence whether each candidate would make a good exemplar. Availability a(i, k) is set to the self-responsibility r(k, k) plus the sum of the positive responsibilities that candidate exemplar k receives from other points.

Finally, we can have different stopping criteria to terminate the procedure, such as when changes in values fall below some threshold, or the maximum number of iterations is reached. At any point through Affinity Propagation procedure, summing Responsibility (r) and Availability (a) matrices gives us the clustering information we need: for point i, the k with maximum r(i, k) + a(i, k) represents point i’s exemplar. Or, if we just need the set of exemplars, we can scan the main diagonal. If r(i, i) + a(i, i) > 0, point i is an exemplar.

Pros of Affinity Clustering:

Affinity Propagation has some advantages over K-Means. First of all the graph based exemplar voting means that the user doesn’t need to specify the number of clusters. Second, due to how the algorithm works under the hood with the graph representation it allows for non-metric dissimilarities

By adjusting the preferences we can lower or raise the number of clusters. Usual practice is to set all preferences to the median similarity for a medium to large number of clusters, or lowest similarity for a moderate number of clusters.

Unlike, other algorithms, it considers all data points as potential prototypes and passes soft information around until a subset of data points “win” and become exemplars.

Cons of Affinity Clustering:

As in K-means, it includes all points like a partition algorithm, therefore, nose and outliers skew cluster formation.

It can be applied only when there is way to measure or precompute a number (similarity) for each pair of data points. Time and memory requirements scale linearly with the number of similarities.

Suitability of the algorithm depends on the ability to identify a subset of data points as exemplars and using them to best account for all other data points.

Hierarchical Affinity Propagation is a variant of the algorithm that deals with quadratic complexity by splitting the dataset into a couple of subsets, clustering them separately, and then performing the second level of clustering.

**5.2 Spectral Clustering**

Algorithm:

Spectral clustering has become increasingly popular due to its simple implementation and promising performance in many graph-based clustering. It often outperforms traditional algorithms such as the k-means algorithm.

-Compute a similarity graph between data objects to cluster.

-Compute the first k eigenvectors of its Laplacian matrix to define a feature vector for each object.

-Run k-means on these features to separate objects into k classes.

Spectral Clustering does a low-dimension embedding of the affinity matrix between samples, followed by a K-Means in the low dimensional space. Spectral Clustering is used for image segmentation. It requires the number of clusters to be specified. It works well for a small number of clusters. In practice Spectral Clustering is very useful when the structure of the individual clusters is highly non-convex or more generally when a measure of the center and spread of the cluster is not a suitable description of the complete cluster.

**5.2 Visual Graphical Outlier Inspection**

Algorithm:

Plot a graph for a univariate+ or combinations of various variables of data for visual inspection (box, scatter, array plot, PCA, etc.) and look for outliers.

Pros of Visual Graphical

Very easy for exploratory analysis

Cons of Visual Graphical

Simplistic and unreliable

1. **Deep Learning Algorithms**

**6.1 Auto-encoders with Deep Learning (Tensorflow)**

Algorithm:

Outliers in some cases are hard to observe in higher dimensional space but are easier to identify in lower dimensional space. Autoencoders map the data they are fed to a lower dimensional space by combining the data’s most important features. It encodes the original data into a more compact representation. It also decides how the data is combined, hence the auto in Autoencoder. A latent variable is essentially an implicit feature of some data. It’s a variable that isn’t observed or measured directly. Autoencoders aim to capture latent variables.

The encoder maps an input vector ‘x’ to a hidden representation ‘h’. The decoder maps ‘h’ back in to the original input space as a reconstruction ‘z’ by the same transformation as the encoder. The difference between ‘x’ and ‘z’ is called reconstruction error. An autoencoder learns to minimize this reconstruction error.

Pros of Autoencoder:

-Able to learn non-linear feature representations, Reduce dimensionality

Cons of Autoencoder:

-Computationally expensive to train

-Extremely uninterpretable

-The underlying math is more complicated

-Prone to overfitting, though this can be mitigated via regularization

Deep Learning tend to overfit the data. By performing autoencoding, we restrict input data to relevant lower dimensional space which makes it easier to identify outliers

**6.1.1 De-noising Autoencoders**

De-noising autoencoders uses a noisy input ˆx (noise added to original input ‘x’) as the input vector. The difference between the resulting output, the reconstruction of the noisy input, and the original input is used as the reconstruction error. This allows the autoencoder to be robust to data with white noise and capture only meaningful patterns of the data

**6.2 PCA with Deep Learning (Tensorflow)**

Algorithm:

Principle Component Analysis is a dimensionality reduction technique. PCA works by projecting input data onto the eigenvectors of the data’s covariance matrix. The covariance matrix quantifies the variance of the data and how much each variable varies with respect to one another.

Eigenvectors are simply vectors that retain their span through a linear transformation, that is, they point in the same direction before and after the transformation. The covariance matrix transforms the original basis vectors to be oriented in the direction of the covariance between each variable. In simpler terms, the eigenvector allows us to re-frame the orientation of the original data to view it at a different angle without actually transforming the data. We are essentially extracting the component of each variable that leads to the most variance when we project the data onto these vectors. We can then select the dominant axes using the eigenvalues of the covariance matrix because they reflect the magnitude of the variance in the direction of their corresponding eigenvector.

We want principal components to be oriented in the direction of maximum variance because greater variance in attribute values can lead to better forecasting abilities.

Pros of PCA:

Reduces dimensionality

Interpretable

Fast run time

Cons of PCA:

Incapable of learning non-linear feature representations

Deep Learning tend to overfit the data. By performing PCA, we restrict input data to relevant lower dimensional space which makes it easier to identify outliers

**6.3 Non-Negative Matrix Factorization (NMF) or Bayesian factorization and then Deep Learning (tensorflow)**

Algorithm:

The challenge in outlier detection is the difficulty in separating true outliers from those data points that are due to noise/measurement errors. In high-dimensional space most of the features tend to be correlated. If a data point is a true outlier, it should be visible in several features. If we take a subspace approach, then a genuine outlier will show up rather than an accidental outlier. The challenge in pursuing a subspace approach is that the space of subspaces are exponential to the number of features and thus are impractical to explore. One way to address this issue is to reduce the dimensionality of the original space. This can be carried out by using matrix factorization approaches. Factorization is a principled approach of simultaneously aggregating correlated features into a reduced number of “meta-feature”.

Techniques like NMF are highly sensitive to outliers. Both mean and the variance-covariance matrix are extremely sensitive to the presence of even one extreme value(outlier) and their use for outlier detection will often mask the discovery of genuine outliers. Thus, we first have to modify NMF to make them more robust against outliers. The R-NMF algorithm is an analogous extension of the recently proposed “Kmeans- -” algorithm.

**6.4 Self-Organizing Map(SOM) and Particle Swarm Optimization(PSO)**

**6.4.1 Self-Organizing Map(SOM)**

Algorithm:

SOMs are unsupervised neural network approach to clustering. It is a feed forward single layer network arranged into a rectangular array. When an input pattern is presented, each neuron calculates how similar the input is to its weights. The neurons whose weights are most similar (minimal distance, d in input space) is declared the winner of the competition for the input pattern, and the weights of the winning neuron are straightened to reflect the outcome. The winning neuron receives the most learning at any stage, with neighbors receiving less, the further away they are from the winning neuron. Typical SOM steps are as follows:

1. Initialize hidden neuron weights to small random values or use PCA weight initialization
2. Feed data row ***x*ᵢ**to input layer
3. Iterate through each neuron in hidden layer and find the Best Matching Unit (BMU), i.e. the neuron that has the smallest Euclidean distance or metric with the data row ***x*ᵢ**
4. Apply a weight update to the BMU and its neighboring neurons. The BMU’s neighbor neurons are calculated using a neighborhood function Φ
5. Shrink the neighborhood function Φ
6. Repeat steps 2 to 5 until iteration limit reached or convergence (the average codebook vector distance between all neurons and all data is smaller than some threshold)

SOMs belong to a class of techniques called [Non-Linear Dimensionality Reduction (NLDR)](https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction). Clustering techniques such as K-means can be applied to the fitted SOM grid to discover hidden patterns of non-linearity.

Pros of SOMs:

-Working with high dimensional data sets is difficult; the SOM reduces information while preserving the most important topological relationships of the data elements on the two-dimensional plane, so that information from different sources can be efficiently fused.

-SOMs are trained using unsupervised learning, i.e.no prior knowledge is available and no assumptions are made about the class membership of data.

-The SOM algorithm is very efficient in handling large datasets. The SOM algorithm is also robust even when the data set is noisy

Cons of SOMs:

-The number of clusters needs to be specified

-A user has to either do manual inspection or apply traditional algorithms, like hierarchical or partitive, to find the cluster boundaries.

**6.4.2 Particle Swarm Optimization(PSO)**

Algorithm:

PSO was discovered from simplified social model simulation. In PSO physical position is not an important factor. The member that is called particle is initialized by assigning random positions and velocities. During each iteration, every particle is accelerated towards its own personal best, as well as in the direction of the global best position. This is achieved by calculating a new velocity term for each particle, based on the distance from its personal best, as well as its distance from the global best position, which will in turn affect the next position of the particle during the next epoch.

#### Pros of PSO:

-PSO is effective in nonlinear optimization problems.

-It is easy to implement.

-Only a few input parameters need to be adjusted in PSO.

-Because the update process in PSO is based on simple equations, PSO can be efficiently used on large data sets.

#### Cons of PSO:

-it tends to be trapped in a local optimum under some initialization conditions.

**6.5 General Adversarial Network(GAN)**

Algorithm:

Generative adversarial network (GANs) are deep neural network architectures comprised of two neural networks, pitting one against the other. GANs learn to mimic any distribution of data, thus enabling the identification of outliers even in high dimensionality. Unlike discriminative algorithms, GANs predict features given a certain label.

Pros of GAN:

No clear objective function to optimize

Cons of GAN:

Finicky to train

**6.5.1 Variational Autoencoders (VAE)**

Algorithm:

VAE is a probabilistic graphical model that combines variational inference with deep learning. Variational autoencoders (VAE) represent a popular, flexible form of deep generative model that can be stochastically fit to samples from a given random process using an information theoretic variational bound on the true underlying distribution. Because VAE reduces dimensions in a probabilistically sound way, theoretical foundations are firm. The anomaly detection task is conducted in a semi-supervised framework, using only data of normal instances for training the VAE. The probabilistic encoder fφ and decoder gθ both parameterizes an isotropic normal distribution in the latent variable space and the original input variable space, respectively. For testing, a number of samples are drawn from the probabilistic encoder of the trained VAE. For each sample from the encoder, the probabilistic decoder outputs the mean and variance parameter. Using these parameters, the probability of the original data generating from the distribution is calculated using Monte Carlo integration. The average probability is used as an anomaly score and is called the reconstruction probability. Data points with high reconstruction probability is classified as anomalies.

Pros of VAE:

The advantage of a VAE over an autoencoder and a PCA is that it provides a probability measure rather than a reconstruction error as an anomaly score, which we will call the reconstruction probability. Probabilities are more principled and objective than reconstruction errors and does not require model specific thresholds for judging anomalies.

Cons of VAE:

It is harder to implement.

The generalized samples are much more blurred(noisy) than GAN.

1. **Other**

**7.1 Variations or combination of the above**

**7.1.1 PCA with Logistic Regression**

Algorithm:

The PCA does an unsupervised dimensionality reduction, while the logistic regression does the prediction. It is not a robust technique for anomaly detection as discussed previously in regression hyperplane distance.

* + 1. **Kmeans to define labels and then use Auto-encoders with tensorflow**