Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Statistics and Machine Learning
Autumn term 2020 | LIU-IDA/STAT-A--21/008--SE

Non-supervised classification of illegally produced alcohol using chemical fingerprinting

Vinay Bengaluru Ashwath Narayan Murthy

Supervisor: Anders Nordgaard
Examiner: Bertil Wegmann

LINKÖPINGS UNIVERSITET

Abstract

Most of the illegally produced batches of ethanol are home-made or semi-industrial, and therefore characterized by a number of by-products called fusel alcohols. The amounts of fusel alcohols found in the chemical investigation of seized alcohol form a kind of 'finger mark' of the production method conditions used to produce alcohol, the finger mark can be utilized in drawing conclusions about possible connections between different cases of alcohol seizures. To be able to draw such conclusions, efficient methods of classification are needed, and ideally it should be possible to give probabilistic or deterministic statements about the method or conditions used for producing the seized material. Classifiers need to be built on historical data and ideally found from applying supervised learning algorithms, but a problem is that the ground truth is not known in the historical cases. This problem is addressed using Cluster analysis, an unsupervised machine learning technique that is used for the discovery of groups among historical cases with respect to by-products and building the ground truth for the historical cases. Multiple clustering methods, based on different clustering approaches such as partitioning approach, hierarchical approach and density-based approach, are used for the discovery of groups among the historical cases of alcohol seizures. The clustering solution obtained by DBSCAN, which uses the cluster ordering obtained as an output from OPTICS, seems to be promising. Classification, a supervised machine learning technique, is used to build a classifier that best classifies the future samples into specific classes which in turn enables us to draw conclusions about the possible connections between different cases of alcohol seizures. Three classification methods K-nearest neighbors, Random Forests and Adaptive Boosting are implemented. Random Forests seems to be the most appropriate method to be used for classifying new samples, though the accuracies of K-nearest neighbors and Random Forests are similar.

Acknowledgements

I would first like to express my gratitude to Anders Nordgaard who played the role as my internal supervisor, external supervisor and the Project Commissioner at The National Forensic Centre of The Swedish Police Authority, without whom this study would not have been possible.

I would like to thank my examiner Bertil Wegmann and my opponent Andreas Christopoulos Charitos for their valuable comments and feedback.

A special thanks to Oleg Sysoev and all the teaching staff for helping us throughout our journey in this Master's programme.

I would also like to thank my family and friends for their moral and financial support throughout this journey of mine.

Table of Contents

1. Introduction 1

1.1 Background 1

1.2 Aim 2

1.3 Related Work 2

2. Data 3

3. Theory 5

3.1 Unsupervised Learning 5

3.1.1 Cluster Analysis 5

3.1.1.1 K-means clustering 6

3.1.1.2 Partition Around Medoids (PAM) 7

3.1.1.3 Agglomerative Hierarchical Clustering 8

3.1.1.4 Density Based Spatial Clustering of Applications with Noise (DBSCAN) 9

3.1.1.5 Ordering Points to Identify the Clustering Structure (OPTICS) 10

3.1.2 Clustering Tendency 12

3.1.2.1 Visual Assessment of Tendency (VAT) 12

3.1.2.2 Hopkins Statistic 13

3.1.3 Optimal Number of Clusters 13

3.1.4 Cluster Validation 15

3.2 Supervised Learning 15

3.2.1 Classification 15

3.2.1.1 K-Nearest Neighbors (k-NN) 16

3.2.1.2 Decision Trees 17

3.2.1.3 Random Forests 18

3.2.1.4 Adaptive Boosting (AdaBoost) 19

3.3 Dimensionality Reduction 20

3.3.1 Principal Component Analysis (PCA) 20

3.3.2 t-Distributed Stochastic Neighbor Embedding (t-SNE) 21

3.3.3 PCA vs t-SNE 23

3.4 Cross-Validation 23

3.4.1 Stratified Cross-Validation 23

3.4.2 Nested Cross-Validation 24

3.5 Distance Metric 24

4. Method 26

4.1 Data Pre-processing 26

4.2 Exploratory Data Analysis 26

4.3 Dimensionality Reduction 27

4.3.1 PCA 27

4.3.2 t-SNE 27
4.4 Cluster Analysis 27
4.4.1 Clustering Tendency 27
4.4.1.1 Visual Assessment of Tendency (VAT) 27
4.4.1.2 Hopkins Statistic 28
4.4.2 Optimal Number of Clusters 28
4.4.3.1 K-means Clustering 28
4.4.3.2 Partition Around Medoids (K-medoids) 28
4.4.3.3 Agglomerative Hierarchical Clustering 28
4.4.3.4 OPTICS 28
4.4.4 Cluster Validation 29
4.5 Classification 29
4.5.1 K-Nearest Neighbors 29
4.5.2 Random Forests 29
4.5.3 Adaptive Boosting 30
5. Results and Discussion 31
5.1 Exploratory Data Analysis 31
5.2 PCA 34
5.3 t-SNE 35
5.4 VAT 39
5.5 Hopkins Statistic 40
5.6 Optimal Number of Clusters 40
5.7 K-Means Clustering 40
5.8 Partition Around Medoids 41
5.9 Agglomerative Hierarchical Clustering 41
5.10 OPTICS 42
5.11 Cluster Validation 44
5.12 K-Nearest Neighbors 45
5.12 Random Forests 45
5.13 Adaptive Boosting 46
6. Conclusion 48
7. Bibliography 49

List of Figures

Figure 1: Alcohol dataset 3
Figure 2: Sample chromatogram 3
Figure 3: Using K-means to find three clusters in the dataset [4] 6
Figure 4:Basic idea of K-medoids [5] 7
Figure 5: Total swapping cost [5] 8

Figure 6: Dendrogram [4] 8
Figure 7:Directly density-reachable [5] Figure 8:Density-reachable[5] Figure 9:Density-connected[5] 9
Figure 10: DBSCAN clustering interpretation [5] 10
Figure 11: OPTICS terminology [5] 11
Figure 12: Overview of the indices implemented in the NbClust package [9] 14
Figure 13: Nearest neighbors [4] 16
Figure 14: A simple decision tree [5] 17
Figure 15: An illustration of Random Forest's idea [10] 18
Figure 16: An illustration of AdaBoost's idea [11] 20
Figure 17: Principal components [5] 21
Figure 18: Measuring pairwise similarities in high-dimensional space [12] 21
Figure 19: Gaussian vs Student t-distribution [12] 22
Figure 20: Swiss Roll dataset. t-SNE (solid line) vs PCA [12] 23
Figure 21: Outlier detection 26
Figure 22: Acetaldehyd 31
Figure 23: Etylacetat 32
Figure 24: N-Propanol 32
Figure 25: Isobutanol 32
Figure 26: Acetal 33
Figure 27: 3-Metyl-1-Butanol 33
Figure 28: 3-Metyl-1-Butanol 33
Figure 29: PCA 2-dimensional plot 35
Figure 30: PCA 3-dimensional plot 35
Figure 31: t-SNE 2d plot. Perplexity=10 36
Figure 32: t-SNE 3d plot. Perplexity=10 36
Figure 33: t-SNE 2d plot. Perplexity=25 37
Figure 34: t-SNE 3d plot. Perplexity=25 37
Figure 35: t-SNE 2d plot. Perplexity=45 38
Figure 36: t-SNE 3d plot. Perplexity=45 38
Figure 37: Visual assessment of clustering tendency 39
Figure 38: K-means clustering data distribution. 40
Figure 39: K-medoids clustering data distribution 41
Figure 40: Hierarchical clustering data distribution 42
Figure 41:Reachability plot, epsilon=1 42
Figure 42: Reachability plot, epsilon=0.5 43
Figure 43: Reachability plot, epsilon=1.5 43
Figure 44: DBSCAN data distribution 44

List of Tables

Table 1: Substances monitored in chromatographic analysis of seized alcohol 4
Table 2: Characteristics of features 34
Table 3: Percentage of variation explained by first three principal components. 34
Table 4: Optimal number of clusters. 40
Table 5: K-means clustering data distribution. 40
Table 6: K-medoids clustering data distribution. 41
Table 7: Hierarchical clustering data distribution. 42
Table 8: DBSCAN data distribution. 44
Table 9: Accuracy and optimized parameter value for k-nearest neighbor classifier. 45
Table 10: Accuracy and optimized parameter value for Random Forest classifier. 46
Table 11: Accuracy and optimized parameter values for AdaBoost classifier. 47

1. Introduction

Supervised learning is a Machine Learning technique which involves input variables and an output variable, an algorithm is used to learn the mapping function from the input to the output. The objective is to approximate the mapping function as precisely as possible so that the output variable can be predicted for new input data. Supervised learning is further categorized into Classification and Regression problems. In a classification problem the output is a categorical variable such as objects, colors etc. In a regression problem the output is usually a real valued continuous variable such as measurements of weights, heights, currency.

Unsupervised learning is a Machine Learning technique which involves just input variables and no output variable. The objective is to model the underlying data structure or distribution in order to get insights about the data. Unsupervised learning is further categorized into Clustering and Association problems, among others. Clustering problems involve the discovery of inherent groupings in the data, such as grouping cities by its population. Association problems involve discovery of rules that describe portions of data, such as people that buy alcohol also tend to buy cigarettes.

The dataset of interest is a collection of historical cases of seized alcohol samples as rows/observations and the cells along the columns, which represent co-variates/variables, contain the amounts of different by-products/fusel alcohols/impurities detected, measured using the unit milligram per liter, in the corresponding sample observations.

This thesis is a combination of Supervised and Unsupervised learning. Specifically, it is a combination of Clustering and Classification. At first, Clustering is used to find inherent groupings in the given dataset i.e. building the ground truth class labels for the observations in the dataset using suitable methods, since the ground truth or the gold standard is not available. Later, Classification is used to build a robust model using suitable methods, that learns precise mapping function using training dataset and best predicts the labels for test dataset. The entire dataset is split into training set which is used to build a classification model and test set in order to determine how well the classification model works on previously unseen data. The classification model built can be used to classify future samples of Alcohol seizures into different groups as precisely as possible. This would help in connecting different cases of alcohol seizures at different time points and/or locations.

1.1 Background

At the National Forensic Centre (NFC) of the Swedish Police Authority there is casework concerning illegal production of ethanol (to be distributed to consumers of alcoholic beverages). This kind of casework is rare from a European (or World-wide) perspective but is due to the legislation in Sweden and other Nordic countries on the production and distribution of alcohol.

The production of alcohol can be industrial with high control on various so-called impurities (by-products). However, most of the illegally produced batches of ethanol are home-made or semi-industrial, and therefore characterized by a number of by-products called fusel alcohols. In the chemical investigation of seized alcohol, suspected to have been illegally produced, several substances are found from gas-chromatographic analysis (GC-FID) and in particular the presence and amounts of fusel alcohols. One may say that these amounts form a kind of “finger mark” of the production method conditions used when producing, and it can be utilized in drawing conclusions about possible connections between different cases or seizures.

However, to be able to draw such conclusions, efficient methods of classification are needed, and ideally it should be possible to give statements about the method or conditions used for producing the seized material. Classifiers need to be built on historical data and ideally found from applying supervised learning algorithms, but a problem is that the ground truth is not known in the historical cases. This is so since the statement from the laboratory is used in the preliminary

investigation and presented in court for the court to make a decision, but the decision made is not to be taken as ground truth.

1.2 Aim

The first and most important part of the project is to investigate whether there are homogeneous groups in the data set with respect to substances apart from ethanol and in particular the so-called fusel oils. The usage of such grouping would be to be able to either link future materials with respect to production methods if they seem to belong to the same group i.e. to classify future materials to a same group, or to separate future materials if they appear to belong to different groups. This part also includes an exploratory stage of data visualization, using suitable techniques, to check for the presence of clear clustering patterns in the dataset.

The second part of the project is, if it seems to be possible to obtain a sensible grouping, to investigate whether a model, to classify the future samples precisely, could be learnt.

1.3 Related Work

A study by C.G.G. Aitken and D. Lucy [1] similar to this project involved the implementation of five methods of assessment for the value of evidence for multivariate data. The data used for the study was the measurements of elemental composition of glass. The study was based on the stated fact that "The evaluation of measurements on characteristics of trace evidence found at a crime scene and on a suspect is an important part of forensic science" [1]. Two out of five methods were based on significance tests and three methods were based on the evaluation of likelihood ratios. The likelihood ratio compares the probability of the measurements on the evidence assuming a common source for the crime scene and suspect evidence with the probability of the measurements on the evidence assuming different sources for the crime scene and suspect evidence. Multivariate data is reduced into univariate data by projecting the original data onto the first principal component in one of the likelihood ratio approach. The two other versions of likelihood ratio account for correlation among the variables and for two levels of variation i.e. the variation between sources and the variation within sources. One version assumes that between source variability is modelled by a multivariate normal distribution, the other version models the variability with a multivariate kernel density estimate [1].

Based on the above explanation, the study by C.G.G. Aitken and D. Lucy [1] was about comparison of seized material for which the ground truth was available unlike in this study. Due to the unavailability of ground truth for sample observations in this study, there is a need to create the ground truth first which is achieved using Cluster Analysis. The second part of this study involves implementation of Classification models to classify new samples, into groups discovered by Cluster Analysis, as accurately as possible. In conjunction, a comparison between new samples/seized materials can be done by using their class labels which is similar to the intended outcome of the study performed by C.G.G. Aitken and D. Lucy [1].

2. Data

Prov	Datum	Filename	acetaldehyd	metanol	etanol	etylacetat	n-propanol	isobutanol	acetal	3-metyl-1-butanol	2-metyl-1-butanol	Material
587	070616	017B1801.D	69.25635		36.52834		60.45432	61.82315		37.31786		401402070215
588	070616	018B1901.D			38.59493	63.1012	172.29456	287.22155		274.97697	76.71908	401402019815
589	070616	019B2001.D	33.94765		39.56812	59.05337	180.72835	292.77599		247.50393	67.6458	401402019817
590	070616	020B2101.D	60.19644		39.74748	92.76569	102.28287	268.49965		237.69853	64.03704	401402019819
591	070616	021B2201.D	79.38066		39.58498	132.62883	188.35881	351.11258		590.01206	192.36074	401402019821
592	070616	022B2301.D	77.35616		39.55334	127.67264	188.18011	350.8034		587.68532	192.20254	401402019823
593	070616	023B2401.D	84.21502		40.39361	138.81514	192.89911	365.92757		608.11447	197.60896	401402019825
594	070616	024B2501.D			38.25617	67.05114	170.62205	287.11085		287.5381	80.98391	401402019827
599	070619	002B0201.D	107.59289		40.16218	204.46453	300.48138	175.07684		311.89773	88.50275	401401795415
600	070619	003B0301.D	153.71567		41.07916	68.29614	283.80541	171.48267		95.32468	20.10093	401401795417
601	070619	004B0401.D	129.12377		40.42090	22.31945	278.98613	149.25679		37.74507	7.5214	401401795419
602	070619	005B0501.D	105.47992		37.86010	164.20412	305.91837	171.33307		202.818	56.2293	401401795421
603	070619	006B0601.D			39.01351		240.47795	102.63405		209.88137	246.64574	401401754215
604	070619	007B0701.D	51.52195		34.04832	56.7245	231.54685	104.46537		78.21993	15.70265	401401818015

Figure 1: Alcohol dataset

Each row in the data set represents one analysis, made on one sample of material, and the column 'Material' is a coded material number where the first 10 digits represent the case in which analyses of the seized material are made. There can be several materials in one case, so the last two digits represent the material number (i.e. from 01 to 99). Moreover, there may be instances of replicate measurements, which means that the value in this column will be the same for two or more rows. To form a unique ID for each analysis the value in column 'Material' can be combined with the value in column 'Filename', which is just an internal code.

Data comes from analyses of seized samples of liquid suspected to be home-made alcohol. The analytical methods used are Gas Chromatography with Flame Ionization Detector (GC-FID) and High-Performance Liquid Chromatography (HPLC) [2]. The output from such analyses are so-called chromatograms. The image of a chromatogram from an analysis of a seized sample is not used here since no rights has been obtained for using it. To illustrate, a chromatogram from another analysis is shown in Figure 2.

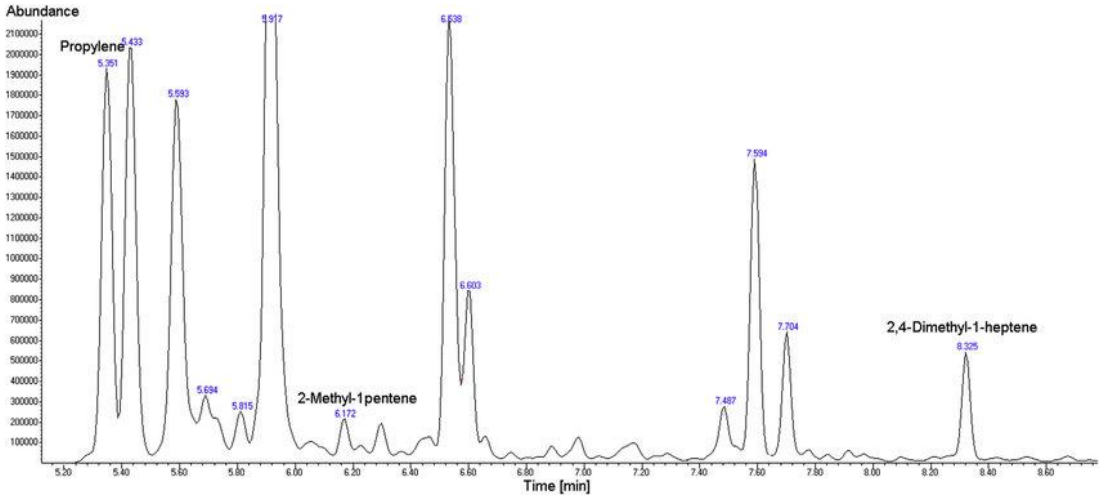

Figure 2: Sample chromatogram

Each peak in the chromatogram represents a chemical compound and the numbers above the peaks are the corresponding peak areas, an integration of the peak is done. The higher the area the more of the compound in question, in relation to other substances.

The substances monitored and stored in the database from which the dataset is taken from are shown in Table 1.

Table 1: Substances monitored in chromatographic analysis of seized alcohol

acetaldehyd:	acetaldehyde (by-product of manufacturing)
metanol:	methanol (by-product of manufacturing - bad distillation)
etanol:	ethanol (target of production)
etylacetat:	ethyl acetate (by-product of manufacturing)
n-propanol:	n-propanol (a so-called fusel oil, by-product typical for home-made ethanol, expected to have discriminate power to separate batches produced with different methods)
isobutanol:	isobutanol (another fusel oil, also expected to have discriminative power to separate batches produced with different methods)
acetal:	acetal (by-product of manufacturing)
3-metyl-1-butanol:	3-Methyl-1-butanol (another fusel oil, also expected to have discriminative power to separate batches produced with different methods)
2-metyl-1-butanol:	2-Methyl-1-butanol (another fusel oil, also expected to have discriminative power to separate batches produced with different methods)

In the data set, all values except those for ethanol are peak areas in the actual chromatogram. The values for ethanol are concentrations in percent. The normal percentages of alcohol excluding wines or beers, for consumption sold in shops in Sweden varies between 30 and 60 %. The data to be used are those for which there is at least one substance besides ethanol. The file has been cleaned to remove analyses where only the concentration of ethanol is stored. These would correspond to seizures of alcohol that is not home made, but industrially produced (so-called technical spirits). However, there is no guarantee that all such cases have been discovered, so an extra step of data cleaning has been recommended.

There are some entries where the concentration of ethanol is missing. This does not imply that the concentration is zero but could be due to the concentration was not calculated. However, it could also be the case that the material did not contain any ethanol. For the other substances, a missing value should be interpreted as the substance could not be detected. Chemically this means that its peak in the chromatogram did not exceed the limit of detection i.e. the threshold for noise. Hence, there should not be any missing values for substances except ethanol, when the field is blank it can be replaced by zero.

3. Theory

3.1 Unsupervised Learning

Unsupervised learning is a type of Machine learning that is used to detect underlying patterns in datasets that do not contain pre-existing labels or classes. Unsupervised machine learning tasks are explanatory in nature. Minimum human supervision is needed in unsupervised learning. Unlike supervised learning which makes use of labeled data unsupervised learning enables to model the probability densities over the input data. The applications of unsupervised learning include density estimation, domains involving summarizing and explaining data features. The importance of unsupervised learning lies in the purpose to discover previously unknown patterns in the data. The best situation where unsupervised learning can be used is when there is no data on the desired outcome. Since the desired outcomes are not known there is no possible way to determine how accurate a particular unsupervised learning method is. Unsupervised learning is often broadly classified into Cluster analysis and Association analysis also among others such as Anomaly detection, Density estimation, Latent variable models etc. [3].

3.1.1 Cluster Analysis

Cluster analysis is used to divide data into groups that are either meaningful or useful or both. Cluster analysis includes applications where it is used as a stand-alone tool to get insight into the data distribution and as a pre-processing step for other algorithms. If the objective is to find meaningful groups, then the clusters should capture the natural underlying structure of the data.

Data objects are grouped together based on the information found in the data that describes the data objects and the relation between the data objects. The goal of Clustering is that the data objects within a group should be similar to each other and the objects in different groups should be different from one another. Clustering is more distinct when the similarity or homogeneity within a group is greater i.e. small variance within a group and dissimilarity between the groups is greater i.e. large variance between the groups. Clustering can also be considered as a form of classification since it creates labels for data objects with the group labels that the objects are clustered into. In contrast, Classification which is included under Supervised learning, is used to assign class labels to new unlabeled data objects using a model trained from data objects with pre-existing class labels. Cluster analysis is also referred to as Unsupervised classification sometimes due to this reason [4].

The notion of degree of similarity between data objects being clustered is central to all the goals of cluster analysis. Clustering methods attempt to cluster the data objects based on the definition of similarity supplied to it [5].

Requirements for clustering applications include scalability, the ability to deal with different attribute types, discovery of clusters with arbitrary shapes, minimal requirement for domain knowledge to determine input parameters, the ability to deal with noise and outliers, insensitivity to the order of input records, the ability to deal with high dimensional data, incorporation of user-specified constraints, interpretability and usability [5].

The types of data involved in clustering analysis include interval-scaled variables - continuous measurements such as weight and temperature, binary variables - variables with two states such as on/off and yes/no, nominal variables - a generalization of the binary variable in that it can take more than two states such as red, blue, green, yellow, ordinal variables - where ranking is important such as gold, silver, bronze medals, ratio variables - a positive measurement on a nonlinear scale such as growth, variables of mixed types [5].

There are several clustering approaches, some of the major approaches are discussed here. Partitioning approach-based methods construct various partitions and then evaluate them by some criterion like minimizing the sum of squares. Typical partitioning approach-based methods include K-means, PAM, CLARANS. Hierarchical approach-based methods create a

hierarchical decomposition of the set of data objects using some criterion. Typical methods include Agglomerative clustering (Agnes), Divisive clustering (Diana), BIRCH, ROCK, CHAMELEON. Density-based approach is based on connectivity and density functions. Typical density-based methods include DBSCAN, OPTICS, DenClue. Grid-based approach is based on multiple-level granularity structure. Typical grid-based methods are STING, WaveCLuster, CLIQUE. Model-based approaches are based on hypothesizing a model for each of the clusters and trying to find the best fit of the model to each other. Typical methods include EM, SOM, COBWEB. Frequent pattern-based approach is based on the analysis of frequent patterns. Typical method is pCluster. User-guided or constraint-based approaches cluster the data objects by considering user-specified or application-specific constraints. Typical methods include COD (obstacles), constrained clustering [5].

3.1.1.1 K-means clustering

K-means is a prototype-based clustering technique where a prototype is defined in terms of a centroid. A centroid in K-means is the mean of a group of points in a continuous multi-dimensional space. A centroid is almost never an actual data object. At first K initial centroids are chosen, where K corresponds to the number of clusters desired and is a user specified parameter. Each data object in the dataset is then assigned to the closest centroid, and each group of points assigned to the closest centroid form a cluster. The centroids of these newly formed clusters are computed and updated. The assignment and update steps are repeated until no point switches clusters, or until the centroids do not change [17].

Algorithm: Basic K-means clustering algorithm [4]

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid based on mean values of objects in the cluster.
4: Recompute the centroid of each cluster i.e. update cluster means.
5: **until** Centroids do not change

The functioning of K-means is depicted in Figure 3. It depicts how three final clusters are formed in four consecutive assignment-update steps, starting from three initial centroids.

In the first step, shown in Figure 3(a), all the initial centroids lie within the bigger group of points and each data object is assigned to one of the initial centroids. Mean is used as the centroid here. The centroid is updated after the data objects are assigned to it. The figure for each iteration indicates the centroid at the beginning of that particular step and the assignment of points to those centroids. In the second step, points are assigned to the updated centroids, again the centroids are updated. In iteration 2, 3, and 4, as depicted in Figures 3 (b), (c), and (d), respectively, two of the centroids move to the smaller groups of points at the bottom. The K-means algorithm terminates in Figure 3(d) when no more changes occur, and the centroids would have identified the natural grouping of points [4].

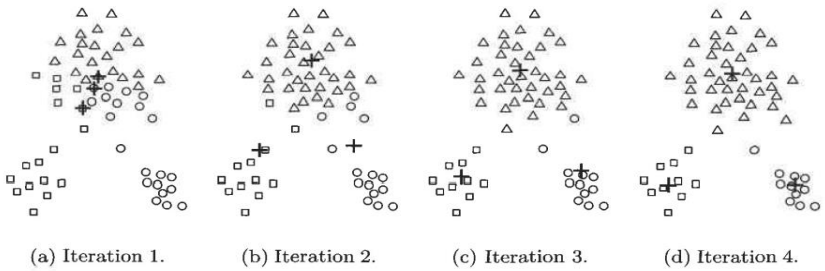

Figure 3: Using K-means to find three clusters in the dataset [4]

A proximity measure that quantifies the notion of closeness for the data under consideration is necessary to assign data objects to their closest centroids. Often, Euclidean distance is used for data objects in Euclidean space. However, several types of proximity measures may be suited for a given data type. In this project, Canberra distance [Section 3.5] is used as a proximity measure in cluster analysis of the given dataset. The reason for choosing Canberra over Euclidean distance as a proximity measure will be discussed at a later part in this report.

Consider data whose proximity measure is Canberra distance. For the objective function, which measures the quality of clustering, the error of each data object is calculated, i.e., its Canberra distance to the closest centroid, and then compute the total sum of errors. Different runs of K-means produce different sets of clusters. The set of clusters with the smallest error is preferred since it is an indication of the prototype being a better representation of the objects in their cluster [4].

$$\text{Error} = \sum_{i=1}^{K} \sum_{x \epsilon Ci} \text{dist}(ci, x)$$

where dist = Canberra distance between two data objects, x = an object, Ci = the i^{th} cluster, ci = centroid of cluster Ci, K = number of clusters.

The centroid i.e. the mean of the i^{th} cluster is calculated as follows

$$c_i = \frac{1}{mi} \sum_{x \epsilon Ci} x$$

where mi = the number of objects in the i^{th} cluster.

K-means is simple, efficient and applicable to a wide variety of data types. However, it is not suited for all types of data such as non-globular clusters or clusters of different densities and sizes. K-means is also bad at handling data with outliers. Application of K-means is also restricted to data with has a notion of center or centroid [17].

3.1.1.2 Partition Around Medoids (PAM)

The major difference between K-means and K-medoids or PAM is that here the center for a cluster is restricted to be one of the data objects assigned to the cluster. PAM is more robust than K-means in the presence of outliers and noise since a medoid is less influenced by outliers or extreme values than a mean. PAM starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering. The algorithm is summarized below. The basic idea of K-medoids algorithm is depicted in Figure 4.

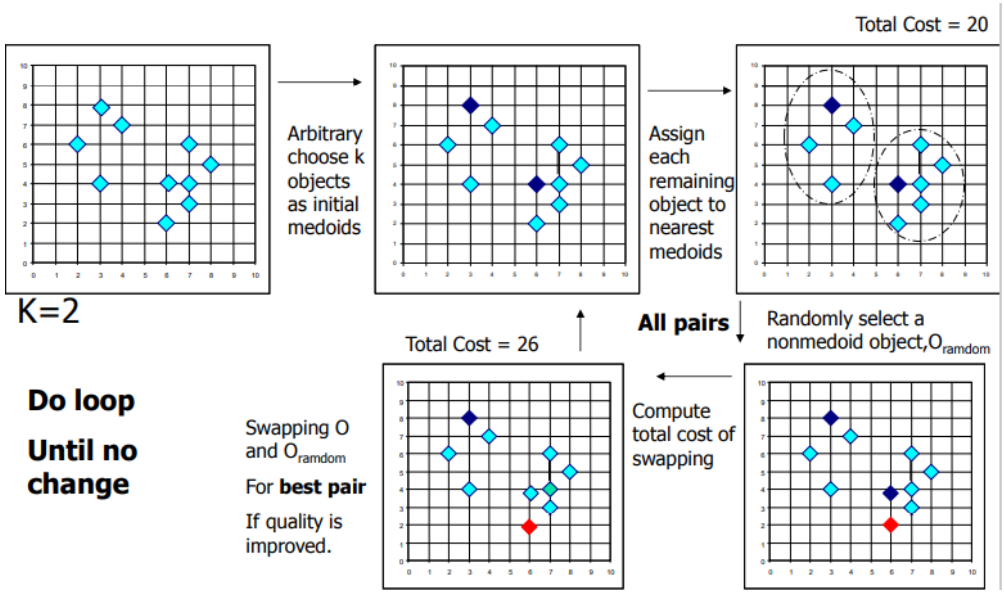

Figure 4:Basic idea of K-medoids [5]

Figure 5 depicts how the total swapping cost is calculated. $TC_{ih} = \sum_j C_{jih}$, d-distance metric

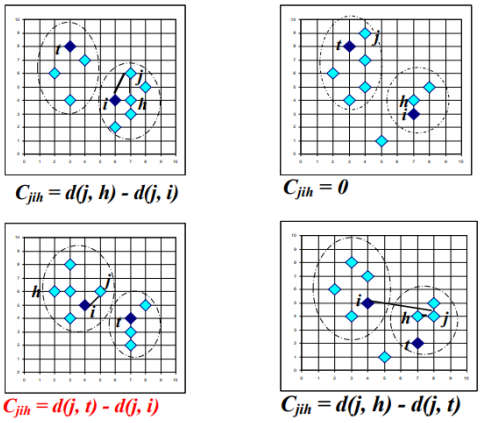

Figure 5: Total swapping cost [5]

Algorithm: Basic PAM clustering algorithm [5]

1: Select K representative objects arbitrarily
2: For each pair of non-selected object h and selected object i, calculate the total swapping cost TC_{ih}
3: Select a pair i and h, which corresponds to the minimum swapping cost
   I. If $TC_{ih} < 0$, i is replaced by h
   II. Then assign each non-selected object to the most similar representative object
4: Repeat steps 2-3 until there is no change

3.1.1.3 Agglomerative Hierarchical Clustering

Agglomerative hierarchical clustering starts with each data object as individual clusters and merges the closest or most similar pair of clusters in every successive step until only one cluster remains. A notion of defining cluster proximity is necessary. Generally, hierarchical clustering is graphically displayed using dendrograms, if the dataset contains a small number of data objects. Dendrograms display cluster-subcluster relationships and the order in which the clusters are merged [18].

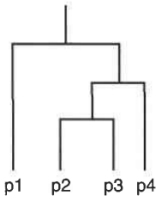

Figure 6: Dendrogram [4]

Algorithm: Basic agglomerative hierarchical clustering algorithm [4]

1: Compute the proximity matrix, if necessary.
2: **repeat**
3: Merge the closest two clusters.
4: Update the proximity matrix to reflect the proximity between the new cluster and the original clusters.
5: **until** Only one cluster remains.

Agglomerative hierarchical clustering techniques are differentiated based on how the cluster proximity is defined. Agglomerative hierarchical clustering techniques such as single link, complete link and group average hierarchical clustering come from graph-based view of clusters. In a prototype-based view of clusters, the proximity is generally defined as the proximity between cluster centroids [4]. An example for prototype-based view of clusters is Ward's method which will be discussed in the Method part of this report.

One issue with agglomerative hierarchical clustering is that it cannot be viewed as globally optimizing an objective function. It instead decides locally using various criteria, at each step the clusters to be merged. Since the information about the pairwise similarity of all points is used, there is a tendency to make good local decisions about combining two clusters. However, once two clusters are merged, it cannot be undone at a later point of time. This approach prevents the local optimization criterion from becoming a global optimization criterion [4].

3.1.1.4 Density Based Spatial Clustering of Applications with Noise (DBSCAN)

In density-based clustering, the density of an object is measured by the number of objects close to it. Core objects are the objects with dense neighborhoods. DBSCAN finds core objects and connects them to their neighborhoods to form dense regions as clusters [6].

DBSCAN quantifies the neighborhood of an object by a user-specified parameter $\varepsilon > 0$, it is used to specify radius of every object's neighborhood. DBSCAN uses a second user-specified parameter 'MinPts', to determine whether a neighborhood is dense or not [6].

Given a set of objects 'D', an object is a core object if a minimum of MinPts objects are contained in its ε-neighborhood. An object 'p' is directly density-reachable from a core object 'q' with respect to ε and MinPts in D, if p is within the ε neighborhood of q. All objects can be brought into a dense region by a core object, from its ε neighborhood, using this directly density-reachable relation. An object p is density-reachable from q with respect to ε and MinPts in D, if there is a chain of objects $p_1,\ldots,p_n$ such that $p_1=q$, $p_n=p$ and p_{i+1} is directly density-reachable from p_i with respect to ε and MinPts. Density-reachability is not a symmetric relation. Two objects p and q are density connected with respect to ε and MinPts if there is an object o such that p and q are density reachable from o with respect to ε and MinPts. Density-connectedness is a symmetric relation. The notion of directly density-reachable, density-reachable, density-connectedness is depicted pictorially in the Figure 7, 8 and 9 respectively [5] [6].

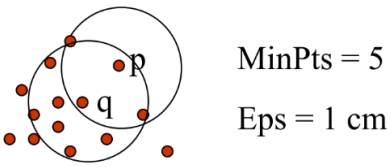

Figure 7:Directly density-reachable [5]

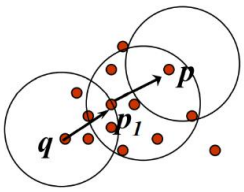

Figure 8:Density-reachable[5]

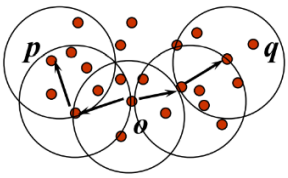

Figure 9:Density-connected[5]

A subset $C \subseteq D$ is a cluster if (1) for any two objects $o_1,o_2 \in C$, o_1 and o_2 are density-connected and (2) there does not exist an object $o \in C$ and another object $o' \in (D - C)$ such that o and o' are density-connected [5]. A cluster is simply a maximal set of density connected points. A simple interpretation of DBSCAN clustering is shown Figure 10.

Input:

- D: dataset containing n objects
- ε: the radius parameter
- MinPts: the neighborhood density threshold

Output:

- A set of density-based clusters.

Algorithm: DBSCAN [5]

```
1. mark all objects as unvisited;
2. do
3.       randomly select an unvisited object p;
4.       mark p as visited;
5.       if the ε -neighborhood of p has at least MinPts objects
6.             create a new cluster C, and add p to C;
7.             let N be the set of objects in the ε -neighborhood of p;
8.             for each point p' in N
9.                   if p' is unvisited
10.                        mark p' as visited;
11.                        if the ε -neighborhood of p' has at least MinPts points, add those
                           points to N;
12.                  if p' is not yet a member of any cluster, add p' to C;
13.            end for
14.            output C;
15.      else mark p as noise;
16. until no object is unvisited;
```

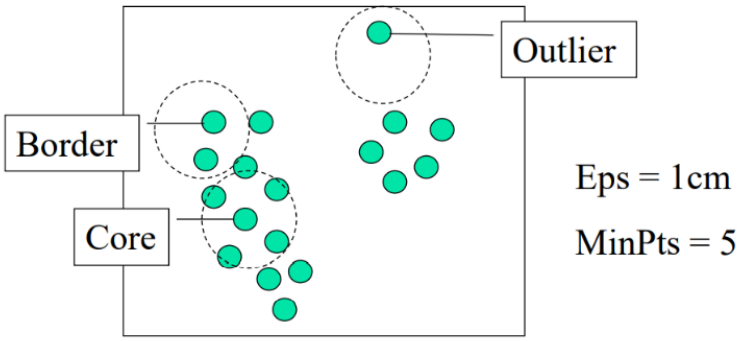

Figure 10: DBSCAN clustering interpretation [5]

3.1.1.5 Ordering Points to Identify the Clustering Structure (OPTICS)

DBSCAN requires the user to input two parameters i.e. ε and MinPts. These parameters are usually set empirically and are difficult to determine. Slight changes in the input parameters lead to different clustering results. Real world, high dimensional datasets are often highly skewed such that their intrinsic clustering structure may not be characterized by a single set of global density parameters [7].

OPTICS is used to overcome aforementioned problems. OPTICS produces a cluster ordering rather than the clustering itself. Cluster ordering is a linear list of all data objects and represents the density-based clustering structure of the dataset. Objects in a denser cluster are put closer to each other in the cluster ordering. OPTICS does not require the user to specify the density threshold ε. The intrinsic clustering structure and the visualization of the clustering can be obtained from the cluster ordering [7].

Different clusterings are constructed simultaneously by processing the objects in an order which selects an object that is density-reachable with respect to the lowest ε value. This way clusters with higher density i.e. lower ε will be finished first. OPTICS needs two properties for every object based on this [5].

"Core-distance: The core-distance of an object p is the smallest value ε' such that the ε'-neighborhood of p contains at least MinPts objects. That is, ε' is the minimum distance threshold that makes p a core object. If p is not a core object with respect to ε' and MinPts, the core-distance of p is undefined." [5]

"Reachability-distance: The reachability-distance to object p from q is the minimum radius value that makes p density-reachable from q. According to the definition of density-reachability, q has to be a core object and p must be in the neighborhood of q. Therefore, the reachability-distance from q to p is max{core-distance(q), dist(p, q)}. If q is not a core object with respect to ε and MinPts, the reachability-distance to p from q is undefined." [5]

"An object p may be directly reachable from multiple core objects. Therefore, p may have multiple reachability-distances with respect to different core objects. The smallest reachability-distance of p is of particular interest because it gives the shortest path for which p is connected to a dense cluster." [5]

Figure 11 illustrates the concepts of core-distance and reachability-distance. Consider ε = 6 mm and MinPts = 5. The core-distance of p is the distance, ε', between p and the fourth closest data object from p. The reachability-distance of q_1 from p is the core-distance of p (i.e., ε' = 3mm) because this is greater than the distance from p to q_1. The reachability-distance of q_2 with respect to p is the distance from p to q_2 because this is greater than the core-distance of p [5].

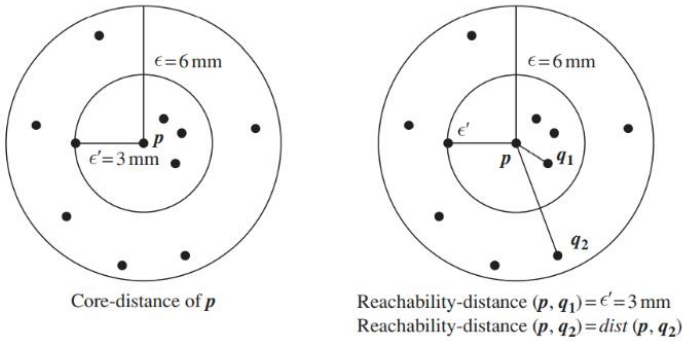

Figure 11: OPTICS terminology [5]

Input:

- DB: dataset containing n objects
- ε: the radius parameter
- MinPts: the neighborhood density threshold

Output:

- Cluster ordering of the database.

Algorithm: OPTICS [7]

```
   #reapeat for all objects in the database
1: for each point pt of DB
       #initializing reachability distance of selected point
2:     pt.reachable_dist = UNDEFINED
3: for each unprocessed point pt of DB
       #extract neighbors of selected point with respect to ε and MinPts in DBSCAN
4:     Nbrs = getNbrs(pt, eps)
5:     mark pt as processed
6:     output pt to the ordered list
       #check if the selected point is not a noise point
7:     if (core_dist(pt, eps, Minpts) != UNDEFINED)
           #initialize a priority queue to extract the closest point
            With respect to ε and MinPts
8:         Seeds = empty priority queue
```

```
                #call the update function
9:              update(Nbrs, pt, Seeds, eps, Minpts)
                #repeat the process for next closest point
10:             for each next q in Seeds
11:                 Nbrs' = getNbrs(q, eps)
12:                 mark q as processed
13:                 output q to the ordered list
14:                 if (core_dist(q, eps, Minpts) != UNDEFINED)
15:                     update(Nbrs', q, Seeds, eps, Minpts)


9.1: update (Nbrs, pt, Seeds, eps, MinPts)
        #Calculate the core distance for the given point
9.2:    coredist = core_dist(pt, eps, MinPts)
        #Update the Reachability distance for each neighbour of p
9.3:    for each obj in Nbrs
9.4:        if (obj is not processed)
9.5:            new_reach_distance = max(coredist, dist(pt, obj))
                #Check if the neighbor point is in seeds
9.6:            if (obj.reachable_dist == UNDEFINED)
                    #update step
9.7:                obj.reachabled_dist = new_reach_distance
9.8:                Seeds.insert(obj, new_reach_distance)
9.9:            else
9.10:               if (new_reach_distance < obj.reachable_dist)
                        #update step
9.11:                   o.reachable_dist = new_reach_distance
9.12:                   Seeds.move-up(obj, new_reach_distance)
```

3.1.2 Clustering Tendency

It is important to check if a dataset contains any meaningful clusters in it before applying any clustering methods on it. If the dataset contains clusters, it is helpful to know the number of clusters present. Clustering methods are ought to produce clusters even if the dataset does not contain inherent groupings, because of its very purpose. The statistical and visual method for assessing the clustering tendency is presented further.

3.1.2.1 Visual Assessment of Tendency (VAT)

VAT is an algorithm used to create visualization of the dataset which is useful to obtain insights on the number of clusters and the cluster hierarchy in the dataset. The pairwise distances between data objects are displayed. The darker the shade, the lesser is the distance between two data objects. A way to interpret the image is to count the number of black squares along the diagonal which would represent the number of clusters in the dataset [8]. The algorithm for visual assessment of clustering tendency approach is as follows.

Algorithm: Visual assessment of cluster tendency [8]

1: Compute the similarity matrix or dissimilarity matrix between the objects in the dataset using a distance metric.
2: Reorder the similarity matrix or dissimilarity matrix so that similar objects are close to each other i.e. create an ordered similarity matrix or ordered dissimilarity matrix.

3: The ordered similarity matrix or ordered dissimilarity matrix is displayed as an ordered similarity image or an ordered dissimilarity image.

3.1.2.2 Hopkins Statistic

Clustering tendency assessment using statistical methods determines if the dataset has a non-random structure, which leads to meaningful clusters. Clustering algorithms may return clusters for the datasets that do not have any non-random structure and do not contain any clusters, such as a set of uniformly distributed points in the data space. Such clusters are random and not meaningful [5].

The intuition behind statistical ways of assessing clustering tendency is that we can try to measure the probability that the dataset is generated by a uniform data distribution. Hopkins statistic is a spatial statistic used to test the spatial randomness of data as distributed in space [5].

Algorithm: Hopkins statistic [4]

1: Generate n points that are randomly distributed across the data space.
2: Sample n actual data points.
3: For both sets of points find the distance to the nearest neighbor in the original dataset.
Let ui be the nearest neighbor distances of the artificially generated points.
Let wi be the nearest neighbor distances of the sample of points from the original dataset.
4: Hopkins statistic (H) is calculated as $H = \frac{\sum_{i=1}^{n} wi}{\sum_{i=1}^{n} wi + \sum_{i=1}^{n} ui}$

If the data points in the original dataset are uniformly distributed, the nearest neighbor distances for randomly (artificially) generated points and the sample of data points are roughly the same and the value of H will be close to 0.5. If the original dataset is highly skewed, the nearest neighbor distances for the sample of data points will be substantially smaller than that of randomly generated points and the value of H will be closer to 0 [5].

The null hypothesis is that D is uniformly distributed hence it does not contain meaningful clusters. The alternative hypothesis is that D is not uniformly distributed hence contains clusters. A threshold value of 0.5 is used to reject the alternative hypothesis i.e. if $H > 0.5$, then it is unlikely that the dataset contains statistically significant clusters. In other words, if H is at most 0.5 the null hypothesis is rejected [5].

3.1.3 Optimal Number of Clusters

All partitioning and hierarchical clustering approaches requires the number of clusters as an input parameter to perform clustering. NbClust, a package in R which provides 30 indices, is used to determine the optimal number of clusters. The best number of clusters is decided using majority rule. The overview of the different indices implemented in the NbClust package is shown in Figure 12 [9].

	Name of the index in **NbClust**	Optimal number of clusters
1.	`"ch"` (Calinski and Harabasz 1974)	Maximum value of the index
2.	`"duda"` (Duda and Hart 1973)	Smallest number of clusters such that index > criticalValue
3.	`"pseudot2"` (Duda and Hart 1973)	Smallest number of clusters such that index < criticalValue
4.	`"cindex"` (Hubert and Levin 1976)	Minimum value of the index
5.	`"gamma"` (Baker and Hubert 1975)	Maximum value of the index
6.	`"beale"` (Beale 1969)	Number of clusters such that critical value >= alpha
7.	`"ccc"` (Sarle 1983)	Maximum value of the index
8.	`"ptbiserial"` (Milligan 1980, 1981)	Maximum value of the index
9.	`"gplus"` (Rohlf 1974; Milligan 1981)	Minimum value of the index
10.	`"db"` (Davies and Bouldin 1979)	Minimum value of the index
11.	`"frey"` (Frey and Van Groenewoud 1972)	Cluster level before index value < 1.00
12.	`"hartigan"` (Hartigan 1975)	Maximum difference between hierarchy levels of the index
13.	`"tau"` (Rohlf 1974; Milligan 1981)	Maximum value of the index
14.	`"ratkowsky"` (Ratkowsky and Lance 1978)	Maximum value of the index
15.	`"scott"` (Scott and Symons 1971)	Maximum difference between hierarchy levels of the index
16.	`"marriot"` (Marriot 1971)	Max. value of second differences between levels of the index
17.	`"ball"` (Ball and Hall 1965)	Maximum difference between hierarchy levels of the index
18.	`"trcovw"` (Milligan and Cooper 1985)	Maximum difference between hierarchy levels of the index
19.	`"tracew"` (Milligan and Cooper 1985)	Max. value of second differences between levels
20.	`"friedman"` (Friedman and Rubin 1967)	Maximum difference between hierarchy levels of the index
21.	`"mcclain"` (McClain and Rao 1975)	Minimum value of the index
22.	`"rubin"` (Friedman and Rubin 1967)	Minimum value of second differences between levels
23.	`"kl"` (Krzanowski and Lai 1988)	Maximum value of the index
24.	`"silhouette"` (Rousseeuw 1987)	Maximum value of the index
25.	`"gap"` (Tibshirani *et al.* 2001)	Smallest number of clusters such that criticalValue >= 0
26.	`"dindex"` (Lebart *et al.* 2000)	Graphical method
27.	`"dunn"` (Dunn 1974)	Maximum value of the index
28.	`"hubert"` (Hubert and Arabie 1985)	Graphical method
29.	`"sdindex"` (Halkidi *et al.* 2000)	Minimum value of the index
30.	`"sdbw"` (Halkidi and Vazirgiannis 2001)	Minimum value of the index

Figure 12: Overview of the indices implemented in the NbClust package [9]

3.1.4 Cluster Validation

Cluster validation is the process of determining how good a clustering algorithm is based on the application of different cluster validation statistics to the clustering solution. Hence clustering algorithms can be compared to each other based on the clustering solution it produces. Two categories of cluster validation statistics are discussed here.

Internal cluster validation uses the internal information of the clustering process such as sum of squared errors in case of K-means clustering. Indices related to internal cluster validation are Silhouette coefficient, Dunn index [9]. Internal cluster validation can only be used to compare same clustering algorithms with different parameter values or clustering algorithms from the same family. For example, two K-means clustering models with different K values can be compared with each other to find out the better model. Two clustering algorithms such as a K-means algorithm and a K-medoids algorithm, which are two different algorithms but from the same family. Partition-based approach can also be compared to each other but only if the objective function for clustering is the same.

External cluster validation uses external information such as the true class labels of the samples to check the goodness of a clustering solution. The clustering accuracy of clustering algorithms can be determined by finding the proportion of samples that are clustered correctly. Hence clustering algorithms can be compared with each other irrespective of their type and their objective function for clustering.

3.2 Supervised Learning

In supervised machine learning a dataset contains a set of variables which are measured and can be denoted as inputs. These input variables have some influence on one or more output variables. The goal of supervised learning is to use the input variables to predict the values for output variables. Input variables are also called as predictors, independent variables or features. Output variables are also called as response variables or dependent variables.

Supervised learning is a type of machine learning where the task is to learn a function that maps an input to an output based on example input-output pairs. A function is deduced from labeled training data. Each observation in the training dataset consists of an input, which is typically a vector, and a desired output. A supervised learning algorithm produces an inferred function by analyzing the training data, which can be used for mapping new observations. In an optimal scenario the algorithm will determine the class labels for previously unseen data correctly.

Supervised learning can be categorized into two categories, Classification and Regression. Classification is used when the target variable is of categorical type e.g. ethnicity of people, if a mail is a spam or not a spam. Regression is used when the target variable is of continuous type e.g. heights of individuals, sales forecast of a company.

3.2.1 Classification

Classification is a two-step process. The first step is the learning step which is the construction of a classification model and the second step is the classification step where the model is used to predict class labels for given data.

In the first step, a classifier describing a predetermined set of data classes is built. In this learning step, the classification algorithm builds the classifier by analyzing and learning from a training set of observations and their pre-existing class labels. The class label of each training observation is available, and the learning of the classifier is supervised, this is the reason classification falls under supervised learning. This step is also viewed as the learning of a mapping function $y=f(X)$, that can predict the class label of a new observation X.

In the second step, the classifier is used for classification. First, the classifier's accuracy of prediction is estimated. The entire dataset will be split into training set and test set, and the classifier is built in the first step using the training set. The test set which is independent of the

training set and not used during the training of the classifier, is used to estimate the predictive accuracy. The accuracy is calculated as the percentage of test set observations that are correctly classified. If the accuracy of the classification algorithm is considered acceptable, it can be used to classify future data or previously unseen data [5].

3.2.1.1 K-Nearest Neighbors (k-NN)

K-Nearest Neighbor classification technique is a lazy learner because it employs the strategy of delaying the process of modelling the training data until it is needed to classify the test data. K-Nearest Neighbor finds all the observations in the training set that are relatively similar to the attributes of the observation of the test set for which class label is to be predicted. These training set observations is used to predict the class label of the test set observation. Given a test set observation, its proximity to the rest of the observations in training set is calculated using a proximity measure.

Figure 13 illustrates the 1-, 2-, and 3-nearest neighbors of a data object located at the center of each circle. The data object is classified based on the class labels of its neighbors. The test class data object is assigned to the majority class of its nearest neighbors when the neighbors have more than one label. In Figure 13 (a), the 1-nearest neighbor of the data point is a negative class label, so the data point is assigned to the negative class. As shown in Figure 13 (c), out of the three nearest neighbors if two are positive labelled and one is negative labelled, using the majority voting scheme, the data object is assigned to positive class. One of the classes is randomly chosen to classify the data point in case there is a tie between the nearest neighbor classes [4].

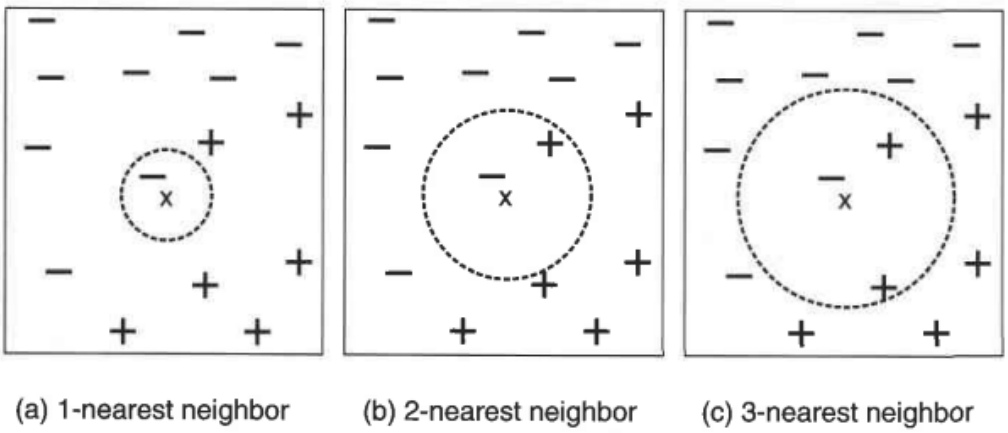

Figure 13: Nearest neighbors [4]

Algorithm: K-Nearest Neighbor classification [4]

1: Let k be the number of nearest neighbors and D be the set of training samples.
2: for each test sample z=(x',y') do
3: Compute d(x',x), the distance between z and every example, (x,y) ∈ D.
4: Select $D_z \subseteq D$, the set of k closest training samples to z
5: $y' = \text{argmax}_v \sum_{(xi,yi) \in Dz} I(v=y_i)$
6: end for

The distance between each test sample z=(x',y') and all the training samples (x,y) ∈ D is computed, to determine the test sample's nearest neighbor list D_z. After obtaining the nearest-neighbor list, the class label for test sample is determined based on the majority class of its nearest neighbors.

Majority voting: $y' = \text{argmax}_v \sum_{(xi,yi) \in Dz} I(v=y_i)$

Where ν is a class label, y_i is the class label for one of the nearest neighbors and $I(\cdot)$ is an indicator function that returns the value 1 if its argument is true and 0 otherwise [4].

3.2.1.2 Decision Trees

A decision tree is a flowchart-like tree structure as shown Figure 14. It contains nodes and branches. The internal nodes or the non-leaf nodes denotes tests on features or attributes of the dataset. Branches in decision trees denotes outcomes of tests of their respective nodes. The leaf nodes are the terminal nodes and hold class labels. The very first node in a decision tree is called the root node [5].

An example of a simple decision tree is depicted in Figure 14. It indicates whether a customer at an electronic store is likely to purchase a computer. The root node and internal nodes denoted by rectangles represent the attributes, the branches represent the splitting conditions and the leaf nodes denoted by ovals represent the outcomes. This is an example of a binary tree since all the internal nodes branches to only two nodes. Non-binary trees contain nodes that branches to more than two nodes. Figure 14 is self-explanatory [5].

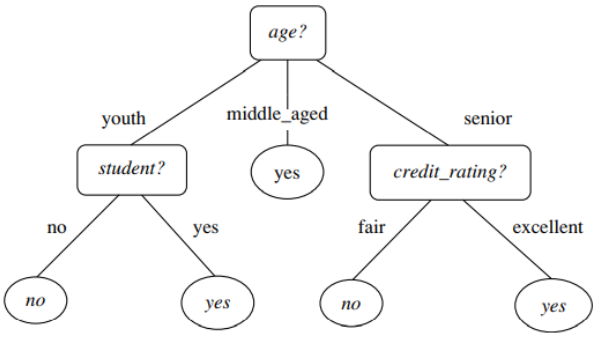

Figure 14: A simple decision tree [5]

Using decision trees for classification: Consider an n-dimensional observation for which the class label is to be determined. The feature values of the observation are tested against the decision tree. A path is traced from the root to the leaf node, which holds the class prediction for the n-dimensional observation. A basic algorithm for generating a decision tree from the training data is depicted below [5].

Input:

- D: Training data with class labels for each observation
- attribute_list: Set of attributes or features
- Attribute_selection_method: A procedure to determine the splitting criterion that best classifies the observation in the training dataset into individual classes. The criterion consists of a splitting_attribute and either a split-point or splitting subset.

Output:

- A decision tree.

Algorithm: Decision Tree [5]

1: Create a node N;
2: **if** observations in D are all of the same class, C, **then**
3: return N as the leaf node labeled with the class C;
4: **if** attribute_list is empty **then**
5: return N as a leaf node labeled with the majority class in D; //majority voting
6: apply **Attribute_selection_method**(D, attribute list) to **find** the "best" *splitting_criterion*;
7: label node N with *splitting_criterion*;
8: **if** splitting attribute is discrete-valued **and** multiway splits allowed **then**
9: *attribute_list* ← *attribute_list – splitting_attribute*; // remove *splitting_attribute*
10: **for each** outcome j of *splitting_criterion*

// partition the tuples and grow subtrees for each partition
11: let D_j be the set of data tuples in D satisfying outcome j; // a partition
12: **if** D_j is empty **then**
13: attach a leaf labeled with the majority class in D to node N;
14: **else** attach the node returned by **Generate_decision_tree**(D_j, *attribute_list*) to node N;
End for
15: return N;

3.2.1.3 Random Forests

Random Forests is an ensemble method built from Decision Trees. Decision Trees are easy to build, use and interpret but in practice suffers from inaccuracy. Decision Trees work well with the training dataset that is used to create them, but they are not flexible when classifying new samples i.e. the test dataset. Random Forests combine the simplicity of Decision Trees with flexibility resulting in improvement in accuracy [10].

The first step is creating a bootstrapped dataset, which is of the same size as the original dataset, by randomly sampling the observations from the original dataset with replacement i.e. the same observation can be sampled more than once. The second step involves creating a decision tree using the bootstrapped dataset. A random subset of features or all the features can be used as candidates at each step for finding a split at each node. The procedure for constructing decision trees is discussed in section 3.2.1.2. The first and second step is repeated for a user specified number of times resulting in a wide variety of decision trees which makes random forests more effective [10].

Consider an observation, for which the class label is to be determined. The observation is run down through all the trees built in the random forest and a class label is obtained from each tree. The class that receives the majority votes will be assigned as the class label for the observation. This technique of bootstrapping the data and using the aggregate to determine class labels is called Bagging [10].

Consider a particular tree created in the random forest and an observation from the training dataset that was not included in the bootstrapped dataset due to sampling with replacement, the group of observations excluded from the bootstrapped dataset are called Out-of-Bag dataset. Since the Out-of-Bag observation is not used to create the tree, it can be used to check if the tree classifies this observation correctly. This Out-of-Bag observation is run through all the other trees that were built without it. Since the label with the majority votes wins, it is the label assigned to the out-of-bag observation. This is repeated for all the other out-of-bag observations for all the trees. The accuracy of the random forest can be measured by the proportion of the out-of-bag samples that are correctly classified by the random forest [10].

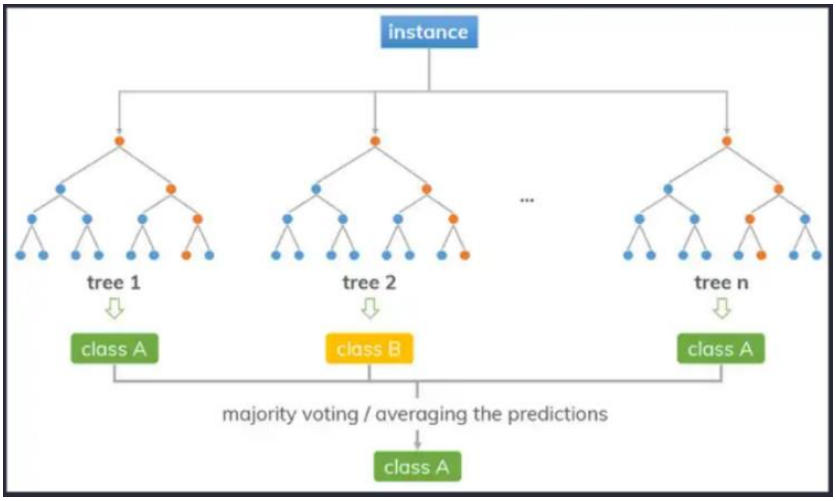

Figure 15: An illustration of Random Forest's idea [10]

3.2.1.4 Adaptive Boosting (AdaBoost)

In a Random Forest each time a Decision Tree is created, a full-sized tree is created. Some of the trees might be bigger than the others, but there is no pre-determined maximum depth. The trees in a forest of trees created with AdaBoost are usually just a node and two leaves. These trees with just one node and two leave are called Decision Stumps, so the forest of trees obtained from AdaBoost is a forest of stumps in actual. Decision Stumps, also referred to as weak learners, are much less accurate than fully grown trees since a stump can use only one variable to make a decision unlike a full-sized tree where all the variables are made use of [11].

In Random Forests, each Decision Tree has an equal vote on the final classification. In AdaBoost, some Decision Stumps get more say than the others in the final classification. The Decision Trees in a Random Forest is created independent of each other. The Decision Stumps created using AdaBoost is dependent on each other since the creation of a stump is influenced by the errors made by its preceding stump [11]. A basic AdaBoost algorithm is shown below.

Input:

- D, a set of *d* class-labeled training tuples
- *k*, the number of rounds (one classifier is generated per round)
- a classification learning scheme

Output:

- A composite model

Algorithm: Adaptive Boosting [5]

1: initialize the weight of each tuple in D to 1/d ;
2: **for** i = 1 to k **do** // for each round:
3: sample D with replacement according to the tuple weights to obtain D_i ;
4: use training set D_i to derive a model, M_i ;
5: compute error(M_i), the error rate of M_i
6: **if** error(Mi) > 0.5 **then**
7: go back to step 3 and try again;
8: **end if**
9: **for** each tuple in D_i that was correctly classified **do**
10: multiply the weight of the tuple by error(M_i)/(1 − error(M_i)); // update weights
11: normalize the weight of each tuple;
12: **end for**

To use the ensemble to classify tuple, X:

1: initialize weight of each class to 0;
2: **for** i = 1 to k **do** // for each classifier:
3: $w_i = \log \frac{1-\text{error(Mi)}}{\text{error(Mi)}}$; // weight of the classifier's vote
4: c = M_i(X); // get class prediction for X from M_i
5: add w_i to weight for class c
6: **end for**
7: return the class with the largest weight;

Where: $\text{error}(M_i) = \sum_{j=1}^{d} w_j * \text{err}(X_j)$

w_j is the weight for the j'th observation, err(X_j) is the misclassification error of tuple X_j, if the tuple (the sample observation) was misclassified then err(X_j) is 1, otherwise it is 0.

A basic idea of Adaptive Boosting is illustrated in Figure 16 which is self-explanatory.

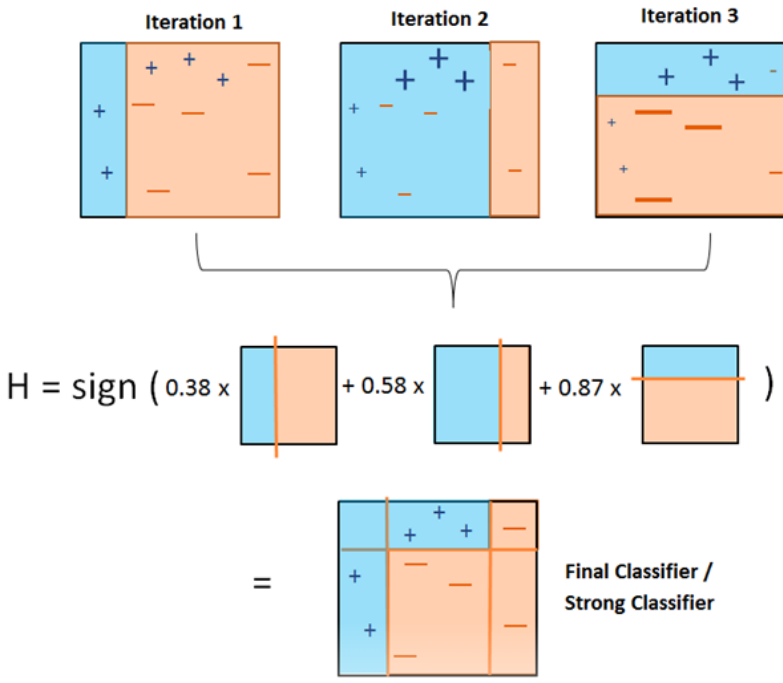

Figure 16: An illustration of AdaBoost's idea [11]

3.3 Dimensionality Reduction

Dimensionality reduction is one of the data reduction strategies that is used to obtain a reduced representation of a dataset. Besides being much smaller in volume, the reduced representation maintains the integrity of the original data closely. Models built on the reduced data should be more efficient yet produce the results similar to that of the original data. Dimensionality reduction is the process of reducing the number of random variables or features in the dataset under consideration. Dimensionality reduction can be performed either as an initial step to other machine learning algorithms or just to aid the visualization of high dimensional data in a lower dimensional space like two or three dimensions which humans can perceive [5].

3.3.1 Principal Component Analysis (PCA)

Consider the data to be reduced to contain n number of dimensions or variables. PCA finds n-dimensional orthogonal base vectors in order of how much of the variance in data is explained. The user can then select k number of principal components to retain that is best capable of representing the original data, where $k \leq n$. Thus the original data is projected onto a much smaller dimensional space, resulting in dimensionality reduction [5].

The basic procedure of PCA is as follows

1. The input data is normalized to make each variable in the dataset to fit in the same range. This ensures that a variable with a large range does not dominate over a variable with a relatively smaller range of values.
2. To provide a basis for the normalized input data, k orthonormal vectors are computed by PCA. These vectors called principal components, are unit vectors each pointing in a direction perpendicular to each other. Input data is a linear combination of these principal components.
3. The principal components are sorted in order of decreasing significance i.e. the amount of variation in data it captures along a particular dimension. The principal components serve as a new set of axes for the data, providing information about variance along their respective axes. The principal components are sorted such that the first component captures the highest variance in the data, the second component captures the second

highest variance in the data and so on. Figure 17 illustrates first two principal components Y_1 and Y_2, for the given dataset that is originally mapped onto axes X_1 and X_2.

4. Since the principal components are sorted in decreasing order of significance, by eliminating the components that captures lower percentage of variance the data size can be reduced in terms of dimensions. It should be possible to reestablish a good approximation of the original data using the strongest principal components [5].

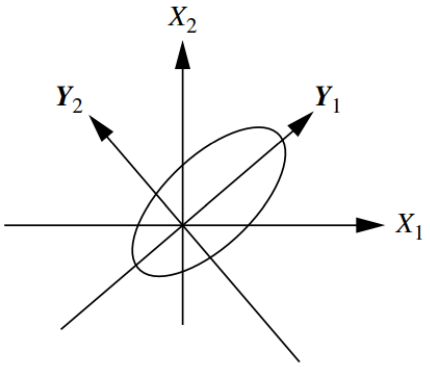

Figure 17: Principal components [5]

3.3.2 t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a nonlinear dimensionality reduction technique, meaning it allows the separation of data that cannot be separated by any straight line. It is primarily used for visualizing high-dimensional data and helps in giving an intuition of data distribution in high-dimensional space. The t-SNE algorithm measures the proximity between pairs of data objects in original space and a low dimensional space. The two proximities are then optimized using a cost function [12].

The basic procedure of t-SNE is as follows:

1. Measure proximity between data objects in original space. Consider data objects in a 2-dimensional space which is the original space as shown in Figure 18. A Gaussian distribution is centered over each data point x_i. The density of all points x_j under that Gaussian distribution is measured. To obtain a set of probabilities P_{ij} for all points, which will be proportional to proximities, renormalization is performed for all points. If for instance, data objects x_1 and x_2 have equal density values under a particular Gaussian distribution, then their proximities from the point under consideration are equal indicating the existence of local similarities in the structure of original space. The variance of the Gaussian distribution which basically is the number of nearest neighbors is controlled using the perplexity parameter. The normal values used for perplexity parameter range from 5 to 50 [12].

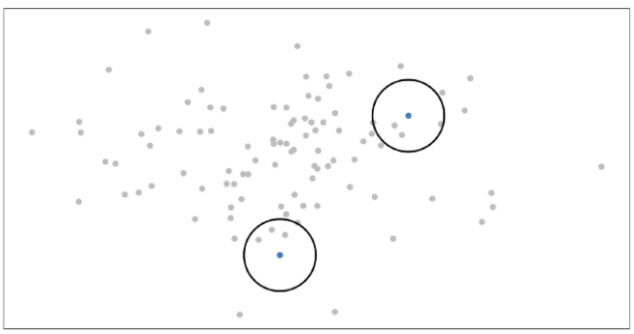

Figure 18: Measuring pairwise similarities in high-dimensional space [12]

2. The second step follows the same procedure as the first step except that a Cauchy or t-distribution with one degree of freedom is used instead of a Gaussian distribution. Another

set of probabilities Q_{ij} for the reduced space is obtained. Compared to a Gaussian distribution, Cauchy distribution has heavier tails which enables it to capture the data objects that are far away better [12].

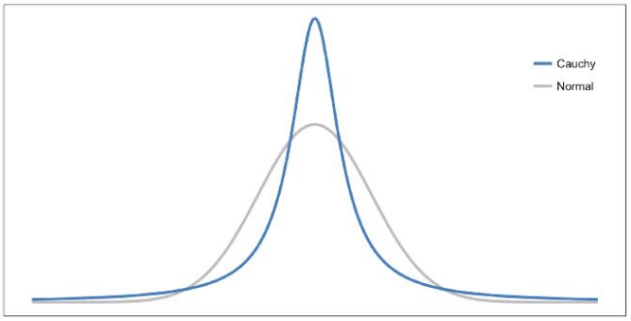

Figure 19: Gaussian vs Student t-distribution [12]

3. Q_{ij} should reflect P_{ij} to the best of its ability. The difference between the probability distributions in the original and reduced spaces is measured using Kullback-Liebler divergence. Finally, gradient decent is used to minimize Kullback-Liebler cost function [12].

The pseudocode for a simple version of t-SNE algorithm is provided below [12]. The notations used in the pseudocode are a bit different from those used in the basic procedure above.

Algorithm : Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for t=1 **to** T **do**
    compute low-dimensional affinities q_{ij}
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

Where:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}$$

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}$$

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)\left(1 + \|y_i - y_j\|^2\right)^{-1}$$

3.3.3 PCA vs t-SNE

PCA is a linear dimension reduction technique. It seeks to maximize variance and preserves large pairwise distances. Data objects that are different in the higher dimensional space will be far apart in the reduced lower dimensional space [5]. Since real world datasets may follow non-linear manifold structures, this results in the quality of visualization being poor in lower dimensional space.

t-SNE preserves only small pairwise distances or local similarities [12] and PCA differs from this. Figure 20 illustrates the difference between PCA and t-SNE using the swiss roll dataset. The figure depicts that due to the non-linear structure of the dataset and PCA preserving only large distances, PCA results in incorrect representation of data in reduced dimensional space.

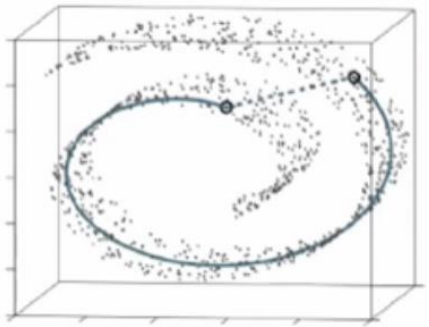

Figure 20: Swiss Roll dataset. t-SNE (solid line) vs PCA [12]

3.4 Cross-Validation

Cross validation is a statistical method used to estimate the ability of machine learning models on new data. It is used to compare models and select a model for a given predictive modelling problem. Cross validation results in performance estimates that usually have a lower bias or are less optimistic than those produced by other methods such as just train/test split. It is often used to evaluate machine learning models when the sample data available is limited. It has a single input parameter k that is used to specify the number of folds the dataset is to be divided into.

The general k-fold cross-validation procedure is as follows:

1. The entire dataset is shuffled in a random manner.
2. The entire dataset is divided into k groups or folds.
3. For each fold:
    I. The fold is taken as test data. It is also referred to as hold-out data.
    II. The remaining folds are taken as training data.
    III. A model is fit on the training dataset and evaluated on the hold-out data.
    IV. The evaluation score such as accuracy is stored, and the model is discarded.
4. The performance of the model is summarized using the sample of the evaluation scores [3].

3.4.1 Stratified Cross-Validation

Model evaluation is done by dividing the dataset into a training set and a test set, then a model is fit using the training set and the model's performance is estimated when making predictions on the test set. The training and test sets must be adequately large and representative of the true under lying distribution to ensure that the model performance estimate obtained is not too biased.

The commonly used approaches for model evaluation are train/test split and k-fold cross-validation. They work well for datasets where the distribution of data objects, with respect to their class labels, among training and test set is balanced. When the distribution of data objects, with respect to their class labels, are not balanced among training and test set, some folds will contain

too little or no data samples from the minority class. This results in the model predicting correctly only for the samples that belong to the majority class and the model evaluations to be misleading.

The solution is to split the dataset in a random manner as usual but in such a way that the same class distribution is maintained in each fold. This technique of using the target variable to control the sampling process is called stratification or stratified sampling. Stratified k-fold cross-validation is a version of k-fold cross-validation that ensures the distribution of data objects, with respect to their class labels, in each split of data will match the distribution of data objects, with respect to their class labels, of the complete training dataset [5].

3.4.2 Nested Cross-Validation

In k-fold cross-validation the performance of machine learning models is estimated by making predictions on the holdout data i.e. the data that was not used to fit the model during training. In addition to using it when comparing and selecting a model for the dataset, it can also be used for optimizing the hyperparameters of a model. The hyperparameter optimization procedure can be nested under the model selection procedure. The model selection procedure will contain model hyperparameter optimization procedure nested inside it. Hence the name Nested cross-validation.

In k-fold cross-validation, a model is fit on all folds but one and the fit model is evaluated on the holdout fold. Here all the folds used to fit a model, put together is the training set and the one hold-out fold is the test set. Each training set is then passed onto the inner model hyperparameter optimization procedure from which the optimal set of model hyperparameters are obtained. The inner k-fold cross-validation procedure splits the training set provided to it by the outer procedure into k folds and aids the evaluation of each set of hyperparameters. Since the hyperparameter optimizing procedure is exposed to only a subset of data provided by the outer cross-validation procedure, it does not have an opportunity to overfit the entire dataset.

The configuration and fitting of the final model are performed using the procedure applied internally to the outer loop as follows.

1. Model selection is done based on the model's performance on the outer k-fold cross-validation procedure.
2. The inner k-fold cross-validation procedure is applied to the entire dataset.
3. A final model is configured using the hyperparameters found during this final search.
4. The final model is fit on the entire dataset and the model can be used to make predictions on new data [13].

3.5 Distance Metric

Euclidean distance is probably the most well-known distance metric used for numerical data. Methods using Euclidean distance metric to define similarity or proximity between data objects work well only when datasets contain compact and isolated clusters [14]. The feature with the largest range will dominate the other features. Euclidean distance does not work well on multidimensional and sparse data. The dataset under consideration does not contain negative entries since the amounts of substances found in an alcohol sample cannot be negative, hence Euclidean distance is generally used for datasets with negative entries in it. Euclidean distance compensates for the negative distances obtained between data objects along different dimensions by squaring the distances between data objects along all dimensions. Hence there is no need for Euclidean distance here.

Canberra distance metric suits our application since it is one of the distance metrics generally used in the analysis of chemical data. It results in accurate outcome for high-dimensional datasets using the K-medoids algorithm. Applications of Canberra distance metric include all kinds of partitional and hierarchical clustering algorithms [14]. Since the alcohol dataset does not contain any negative entries and Canberra distance, which is somewhat equivalent to using absolute distance between data objects, suits our application. Canberra distance metric will be used for the other clustering methods as well.

The Canberra distance **d** between vectors **p** and **q** in an n-dimensional real vector space is given as follows:

$$d(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i| + |q_i|}$$

where $\mathbf{p} = (p_1, p_2, \ldots p_n)$ and $\mathbf{q} = (q_1, q_2, \ldots q_n)$ are vectors.

4. Method

4.1 Data Pre-processing

The terms samples, observations, data objects and data points are used interchangeably throughout the report. The terms features, variables, dimensions and covariates are also used interchangeably throughout the report.

The dataset is checked for duplicate observations and the duplicates are removed. The features acetaldehyd, etylacetat, n-propanol, isobutanol, acetal, 3-metyl-1-butanol, 2-metyl-1-butanol are used from here on. The variable methanol is neglected since it does not occur frequently in non-industrially produced alcohol and it does not have much discriminative power to differentiate between different groups of observations in the dataset. Observations with NA's i.e. empty for all the variables are removed. All empty fields in the remaining observations are replaced by the value zero.

4.2 Exploratory Data Analysis

Exploratory analysis approach, for analysing the dataset to summarize the main characteristics of each feature, is performed. The main characteristics of each feature used are Minimum value, Maximum value, Mean, Median, Skewness, zero count and Outlier count. Skewness is a measure of the asymmetry of a density curve about its mean. The standard procedure for outlier detection using box plot is depicted in Figure 21.

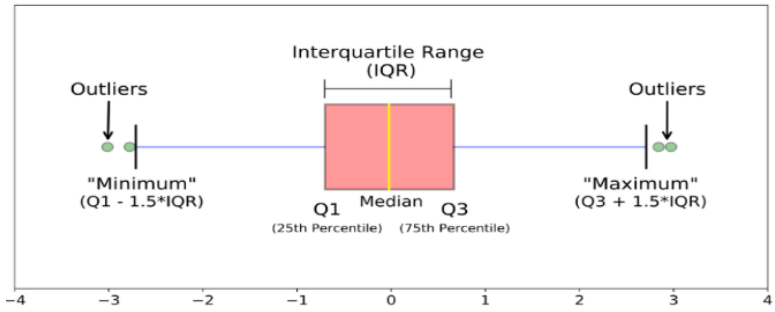

Figure 21: Outlier detection

Box plots are used to determine how the values in each variable are distributed. Boxplots display the distribution of each feature based on five characteristics i.e. median (Q2/50th percentile) the middle value of a feature, first quartile (Q1/25th percentile) the middle value between the smallest number (not the "minimum") and the median of the dataset, third quartile (Q3/75th percentile) the middle value between the median and the highest value (not the "maximum") of the dataset, maximum: Q3+1.5*IQR and minimum: Q1-1.5*IQR. Box plots also display interquartile range (IQR) 25th to 75th percentile, whiskers (shown in blue Figure 21) and outliers (shown as green circles in Figure 21) [15].

All the functions mentioned below are with respect to Python. Histograms are used to see the distribution of individual variables. The figures in results section 5.1 depict Seaborn's version of histogram, known as density plot. It provides a nice density curve over the entire distribution of data. The function distplot is used to plot the histograms. Seaborn's distplot function combines matplotlib's hist function with seaborn's kdeplot and rugplot functions. Matplotlib's hist function computes and draws the histogram of features with automatic calculation of a good default bin size. Seaborn's kdeplot function plots univariate distributions using kernel density estimation.

Kernel density estimate (KDE) is used to represent each feature using a continuous probability density curve in one dimension. Rather than using discrete bins like histogram, KDE plot smooths observations with a Gaussian kernel, producing a continuous density estimate. Seaborn's rugplot plots marginal distributions by drawing ticks along the x and y axes.

4.3 Dimensionality Reduction

The sole purpose of using dimensionality reduction technique here is to visualize the high-dimensional alcohol dataset in lower dimensions. Since a visual representation of a dataset in low-dimensional space helps us in getting an idea about how the data is distributed in the actual high-dimensional space, the seven-dimensional alcohol dataset is visualized in reduced two and three-dimensional space which we can perceive. This part of the study is instructive, especially for chemists, who often use PCA alone to reveal groups [16].

4.3.1 PCA

PCA is implemented using Scikit learn package in Python. The data is not scaled before fitting the PCA model to it. The first three principal components i.e. principal components that capture the three largest variations are extracted. Plots of the data in a reduced two-dimensional and a three-dimensional space is obtained by plotting the data using the first two and three principal components, respectively. Python's Seaborn package is used for the purpose of visualization.

4.3.2 t-SNE

t-SNE is implemented using Scikit learn package in Python. The first three components or dimensions of the embedded are extracted. Plots of the data in a reduced two-dimensional and a three-dimensional space is obtained by plotting the data using the first two and three dimensions of the embedded space respectively. Python's Seaborn package is used for the purpose of visualization.

4.4 Cluster Analysis

The following workflow is used for the cluster analysis of the given dataset for individual clustering methods used and for the clustering as a whole process.

I. Assessing the clustering tendency
II. Determining the optimal number of clusters for different clustering methods
III. Application of clustering methods
IV. Evaluation of clusters or Cluster validation

4.4.1 Clustering Tendency

A visual method Visual Assessment of Tendency (VAT) and a statistical method Hopkins Statistic is used for assessing the clustering tendency of the data and is presented further.

4.4.1.1 Visual Assessment of Tendency (VAT)

Visual Assessment of Tendency method is implemented using a package called pyclustertend in Python. A better version of the VAT algorithm iVAT which provides a better visualization of the ordered similarity matrix is used. Current implementation of iVAT works only with Euclidean distance as a distance metric, the source code was altered to use Canberra distance as the distance metric instead.

4.4.1.2 Hopkins Statistic

The statistical method to check for clustering tendency Hopkins Statistic is implemented using a package called pyclustertend in Python. Current implementation of Hopkins Statistic function works only with Euclidean distance as a distance metric, the source code was altered to use Canberra distance as the distance metric instead.

4.4.2 Optimal Number of Clusters

The optimal number of clusters for K-means clustering, Partition Around Medoids and Agglomerative Hierarchical Clustering is determined using a package called NbClust in R. This also means finding the best or optimum model for the three methods mentioned above. The output obtained is a result of the majority voting over 30 different indices shown in section 3.1.3.

4.4.3.1 K-means Clustering

K-means clustering is implemented using Pyclustering package in Python. Canberra distance metric is used to compute the similarity between data objects. K-means++ method is used for initialization of centroids. It selects initial cluster centers for K-means clustering in a smart way to speed up convergence. The value for the input parameter K i.e. the number of clusters to be used is already determined in section 4.4.2.

4.4.3.2 Partition Around Medoids (K-medoids)

K-medoids clustering is implemented using Scikit-learn-extra package in python. Canberra distance metric is used to compute the similarity between data objects. K-means++ method is used for initialization. It selects initial cluster centers for k-means clustering in a smart way to speed up convergence. The value for input parameter K i.e. the number of clusters to be used is already determined in section 4.4.2.

4.4.3.3 Agglomerative Hierarchical Clustering

Agglomerative hierarchical clustering is implemented in R using stats package. A distance matrix is first generated on the Alcohol dataset using Canberra distance metric, hierarchical clustering is then applied to the distance matrix to obtain a dendrogram. The cluster analysis method used is Ward method. “Ward method minimizes the total within-cluster variance. At each step, the pair of clusters with minimum cluster distance are merged. To implement this method, at each step find the pair of clusters that leads to minimum increase in total within-cluster variance after merging” [9]. The value for K i.e. the number of clusters to be used to obtain an optimum clustering solution is already determined in section 4.4.2. The dendrogram is cut using an R function at a particular place to obtain K clusters.

4.4.3.4 OPTICS

OPTICS is implemented using Scikit learn package in Python. Canberra distance metric is used to compute the similarity between data objects. The input parameter, MinPts is set using a heuristic method which says the Minimum points should be set to two times the number of features or dimensions of the dataset involved. Following this heuristic, we get 14 as the value for Minimum points parameter for the alcohol dataset. Some of the other heuristic methods include setting Minimum points parameter to the natural logarithm of the number of observations in the dataset or number of dimensions plus one. The reason for choosing the first heuristic method is that the value obtained by first method coincides with the range of values, i.e. 10 to 20, suggested in the original paper on OPTICS [7] to find the best results. DBSCAN clustering method is used to perform the actual clustering. The value for the second input parameter i.e. epsilon ε is chosen by visual method. The reachability plot, obtained as an output from OPTICS algorithm (shown in the results

section 5.10), is visually examined for selecting the epsilon value and value of 1 is chosen in this case since it seems to produce a better solution than other values.

4.4.4 Cluster Validation

Cluster validation among different models of the same clustering algorithm i.e. K-means with K=2 and 3 for instance, is already performed inherently during the process of determining the optimal number of clusters for K-means, K-medoids and Hierarchical clustering. The models trained are the ones with the best parameters.

Cluster validation and comparison among different clustering algorithms, that belong to different families, i.e. K-means, K-medoids, Hierarchical clustering which belongs to partition based approach and DBSCAN or OPTICS which belong to density based approach, cannot be performed since the very notion of clusters is different in these methods and the objective function for clustering is completely different from one another.

4.5 Classification

After building the ground truth for all the observations in the alcohol dataset using clustering techniques in the first part of this project, classification is used in the second part of this project to build a robust model that learns a mapping function that maps the independent variables to the target variable. The classifier built will be used for the classification of future samples.

4.5.1 K-Nearest Neighbors

K-nearest neighbor classification is implemented using Scikit learn library in Python. The only hyperparameter required is k, which defines the number of nearest neighbors for a given data point. Canberra distance metric is used to measure the proximity between data objects. A combination of nested and stratified cross-validation is used for model selection and hyperparameter optimization. All other input parameters to the Scikit learn's K-nearest neighbor classifier function were used with default settings.

The hyperparameter search space is set as k = 1,2,3…. 20. The search strategy used is grid search. For the outer stratified cross-validation i.e. for the model selection loop, the number of splits is set to 13 since the class with least number of data objects has 13 data objects in it and setting the number of splits to 13 ensures that at least and at most 1 data object from this particular class is included in each fold. For the inner stratified cross-validation i.e. the hyperparameter optimization loop, the number of splits is set to 12 since only the training set is available for this loop and the training set will contain only 12 data objects from the minority class i.e. one data object from the minority class in each fold. The remaining 1 data object from the minority class will be in the outer loop's hold out set.

4.5.2 Random Forests

Random Forests which is an ensemble method is implemented using Scikit learn library in Python. The base estimators used are Decision tree classifiers. The hyperparameter, number of trees to be built in the random forest is determined using the hyperparameter search space. Gini coefficient is the function used to measure the quality of a split. The nodes are expanded until all the leaves are pure or until all leaves contain less than two samples. All the features are considered as candidates when looking for the best split at each node. Bootstrapped samples are used to build trees. The out-of-bag samples are used to estimate the generalization accuracy. The number of samples drawn from the original dataset to create a bootstrapped dataset to train each base classifier, is equal to the number of observations in the original dataset. A combination of nested and stratified cross-validation is used for model selection and hyperparameter optimization. All other input parameters to the Scikit learn's Random Forest classifier function were used with default settings.

The hyperparameter search space is set as n_estimators = 50,55,60,65….150. The search strategy used is grid search. For the outer stratified cross-validation i.e. for the model selection loop, the number of splits is set to 13 since the class with least number of data objects has 13 data objects in it and setting the number of splits to 13 ensures that at least and at most 1 data object from this particular class is included in each fold. For the inner stratified cross-validation i.e. the hyperparameter optimization loop, the number of splits is set to 12 since only the training set is available for this loop and the training set will contain only 12 data objects from the minority class i.e. one data object from the minority class in each fold. The remaining 1 data object from the minority class will be in the outer loop's hold out set.

4.5.3 Adaptive Boosting

Adaptive Boosting is implemented using Scikit learn library in Python. The base estimators used are Decision tree classifiers. The hyperparameters, number of base estimators to be built and the learning rate for adaptive boosting is determined using the hyperparameter search space. All other input parameters to the Scikit learn's AdaBoost classifier function were used with default settings.

The hyperparameter search space is set as n_estimators = 1,2,3… 20 and learning_rate = 0.1,0.2,0.3….. 1. The search strategy used is grid search. For the outer stratified cross-validation i.e. for the model selection loop, the number of splits is set to 13 since the class with least number of data objects has 13 data objects in it and setting the number of splits to 13 ensures that at least and at most 1 data object from this particular class is included in each fold. For the inner stratified cross-validation i.e. the hyperparameter optimization loop, the number of splits is set to 12 since only the training set is available for this loop and the training set will contain only 12 data objects from the minority class i.e. one data object from the minority class in each fold. The remaining 1 data object from the minority class will be in the outer loop's hold out set.

5. Results and Discussion

5.1 Exploratory Data Analysis

The histograms below show us the distribution i.e. frequency of each value in each of the variables involved in the analysis. It is evident that all the variables are highly positively skewed. The box plots indicate the outliers calculated using the standard procedure as explained in section 4.2 and Figure 21. The plots are obtained just to get an idea about how the data is distributed along different dimensions. The plots illustrate the marginal distributions of all the random variables involved. Though smooth curves are not an absolute necessity in this case, it gives us a good understanding of the marginal distributions sometimes. The smoothing function used is Gaussian kernel. We see the marginal distributions of variables extending beyond 0 and into the negative x-axis as a result of addition of multiple Gaussian kernels at x=0 due substantial amount of 0's in all the variables. Ideally, the part of the density curve that extends beyond 0 towards the negative x-axis should be truncated and ignored resulting in the distribution's mode falling at x=0, similar to that of the current plots. The current implementation of Seaborn library's distplot function in Python which does not include truncation of the density curve results in the error we see along negative x-axis.

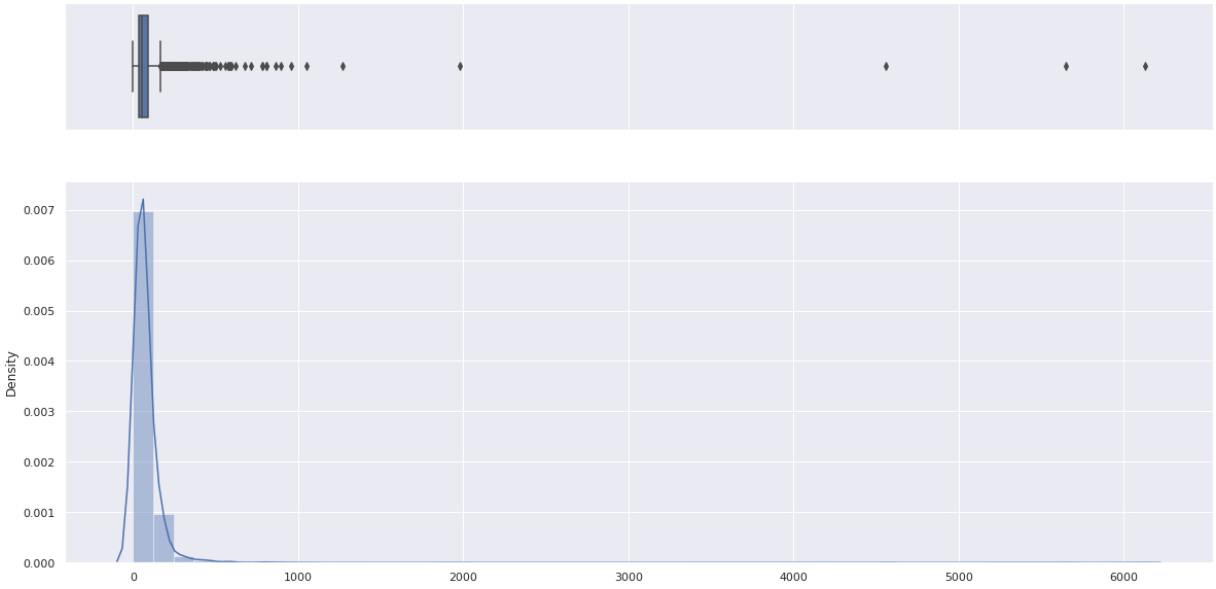

Figure 22: Acetaldehyd

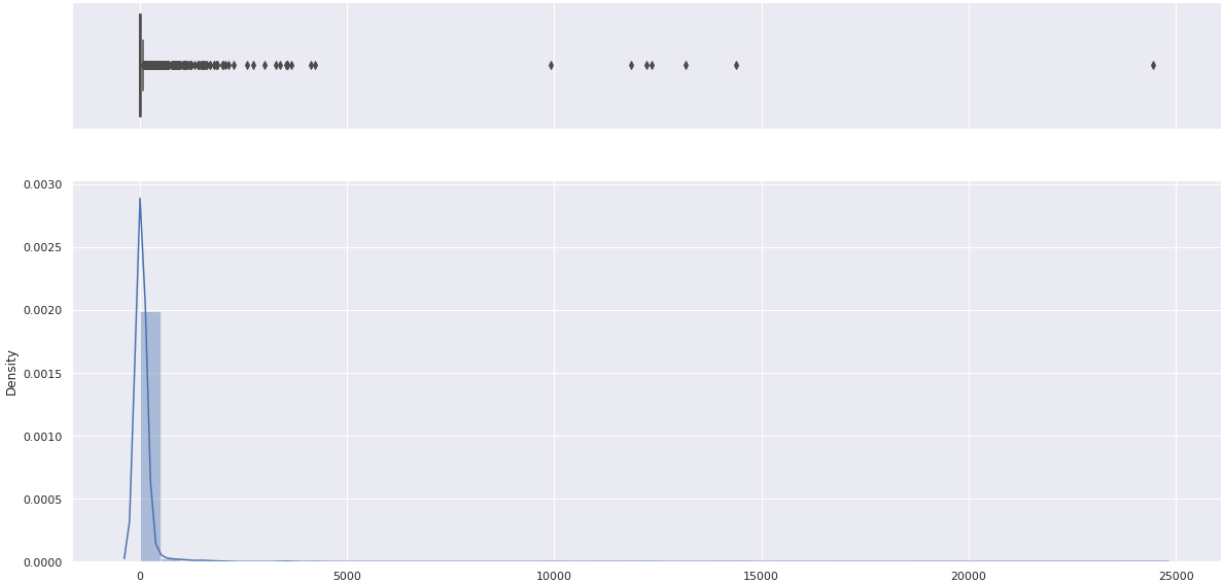

Figure 23: Etylacetat

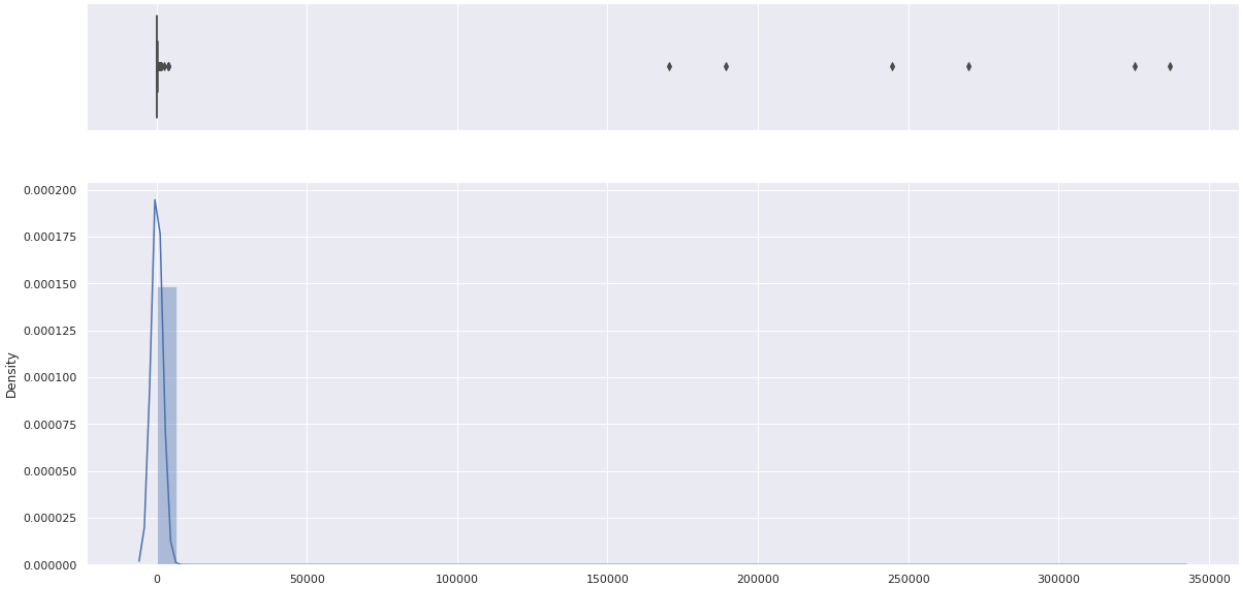

Figure 24: N-Propanol

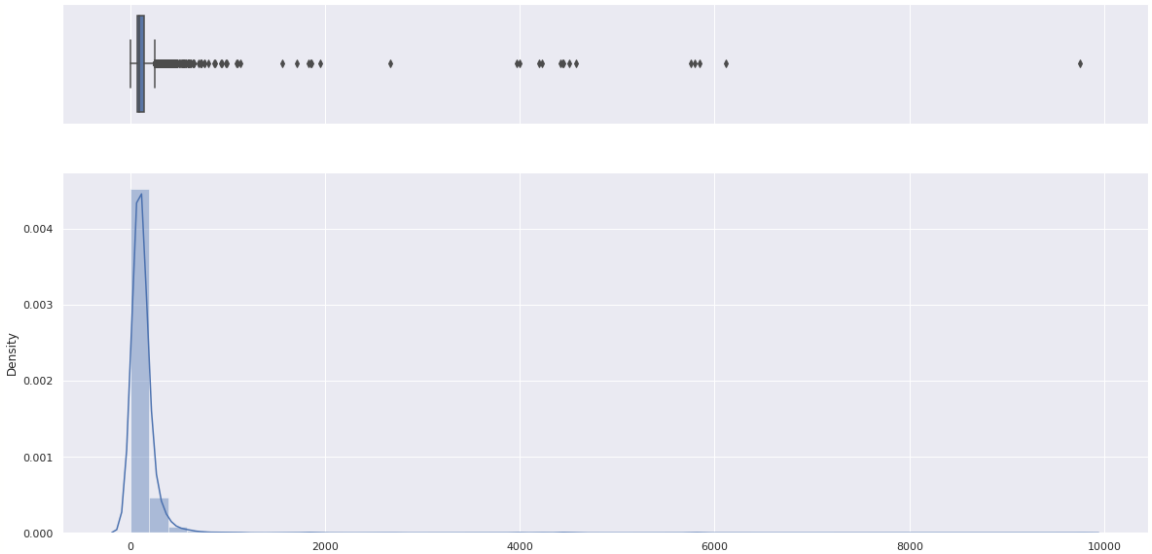

Figure 25: Isobutanol

Figure 26: Acetal

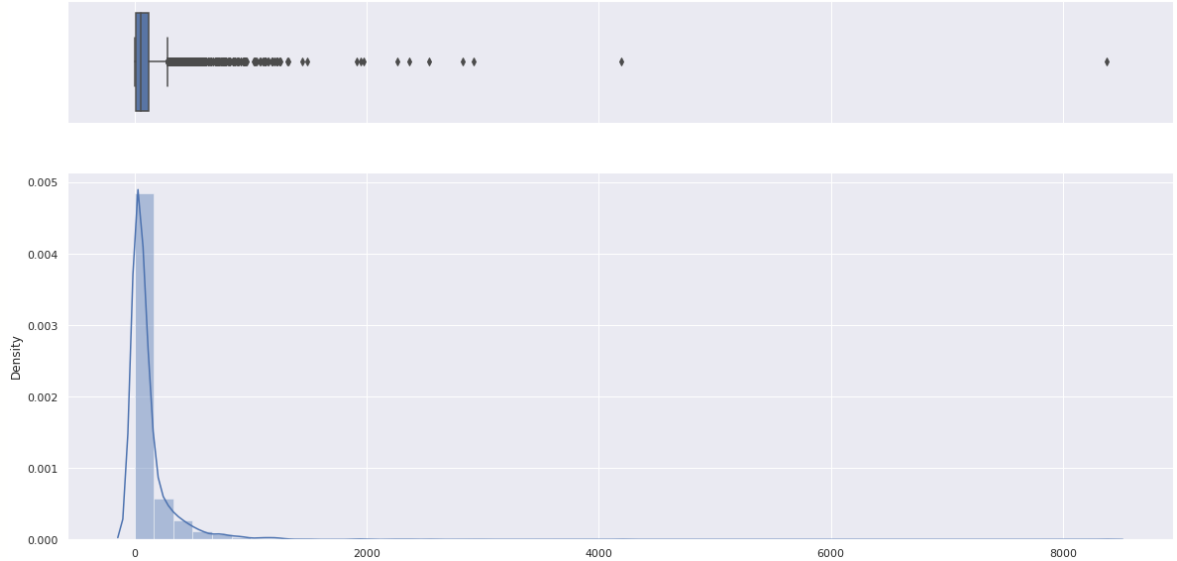

Figure 27: 3-Metyl-1-Butanol

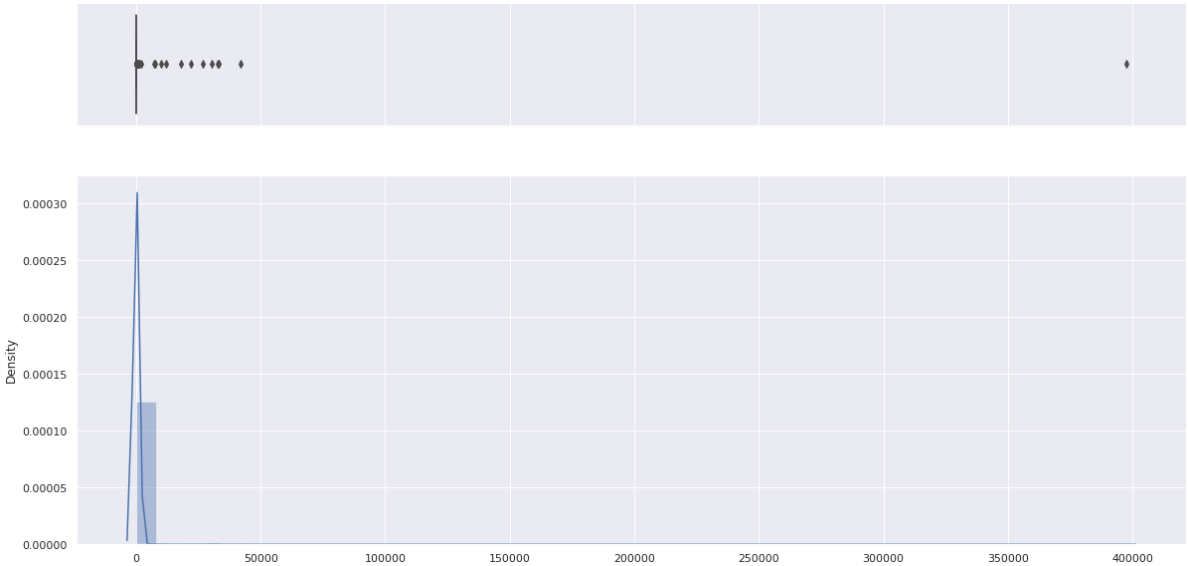

Figure 28: 3-Metyl-1-Butanol

Table 2 depicts some of the main characteristics of the features involved in the analysis. The unit of measurement for all the features is milligram per liter (mg/L).

Table 2: Characteristics of features

Feature	Min	Max	Mean	Median	Skewness	Zero count	Outliers
Acetaldehyd	0	6126.49	77.12	55.78	25.05	632	282
Etylacetat	0	24452.25	88.25	10.24	22.52	1937	477
N-propanol	0	336825	538.41	126.55	28.03	162	294
Isobutanol	0	9754.78	140.93	98.15	16.05	93	332
Acetal	0	5299.57	20.82	0	19.22	3215	846
3-metyl-1-butanol	0	8372.09	121.54	52.55	12.34	750	460
2-metyl-1-butanol	0	397629	185.85	10.61	60.03	1734	460

By examining the characteristics in Table 2, it is evident that:

1. The range (Max-Min) of all the variables is quite large.
2. The mean and the median values are very small compared to the range as a result of large number of zeros. Hence mean and median is not very helpful as a measure of central tendency here.
3. The Skewness values indicate that all marginal distributions are highly positively skewed.
4. The zero count gives us an idea about how sparse the data is.
5. There are substantial number of outliers that are calculated using the standard procedure as explained in section 4.2 and Figure 21.

5.2 PCA

Table 3 shows the variation percentage explained by each of the first three principal components. The first two components capture 99.5% of the variance cumulatively. In an ideal scenario we should be able to get an idea about the data distribution in the high dimensional space by plotting the reduced data using just the first two principal components.

Table 3: Percentage of variation explained by first three principal components.

Principal Component	Percentage of explained variance
First	71.3
Second	28.2
Third	0.3

Figure 29 depicts the distribution of the data in a reduced two-dimensional space. It is evident that this particular plot is not much of a use to get an intuition of how the data is distributed in the actual seven-dimensional space for the alcohol dataset.

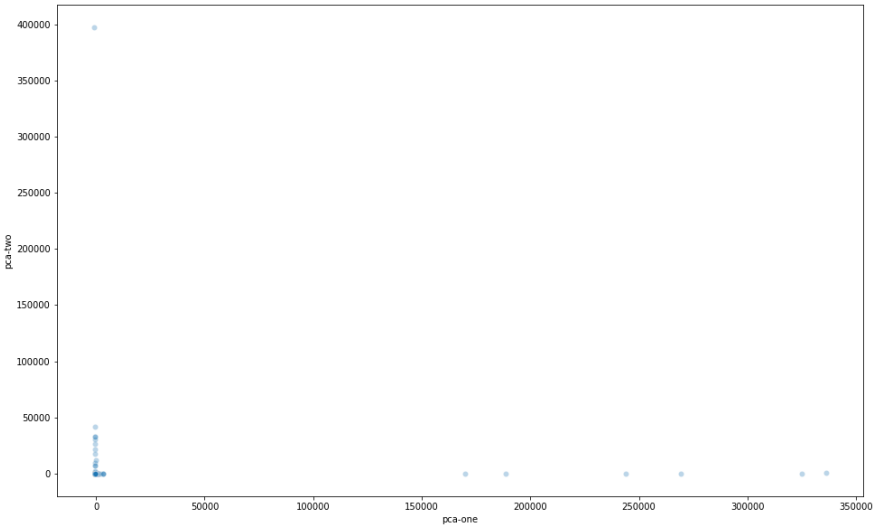

Figure 29: PCA 2-dimensional plot

Figure 30 depicts the distribution of the data in a reduced three-dimensional space. Though the first two principal components explain the essential part of the total variance, due to a large number of zeros and large range of variables it does not seem to be possible to obtain an informative plot. The effect of large range of variables on PCA can be taken care of by using transformations such as log or square root, but the choice was made to not transform the data and use the original data for visualization. It is evident that this plot in Figure 30 is not much of a use, similar to the previous plot in Figure 29, to get an intuition of how the data is distributed in the actual seven-dimensional space for the alcohol dataset.

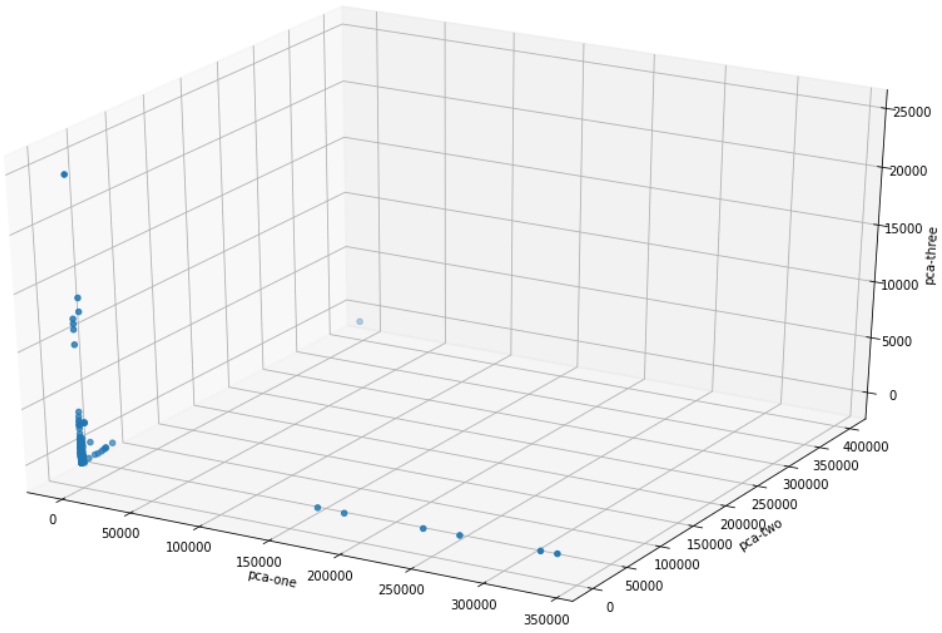

Figure 30: PCA 3-dimensional plot

5.3 t-SNE

The plots below depict the distribution of data in reduced two and three-dimensional space for different values of perplexity parameter. The reason for choosing perplexity parameter value to be 10,25 and 45 respectively is that the original paper [12] suggests a perplexity value in the range 5 to 50 to obtain good results.

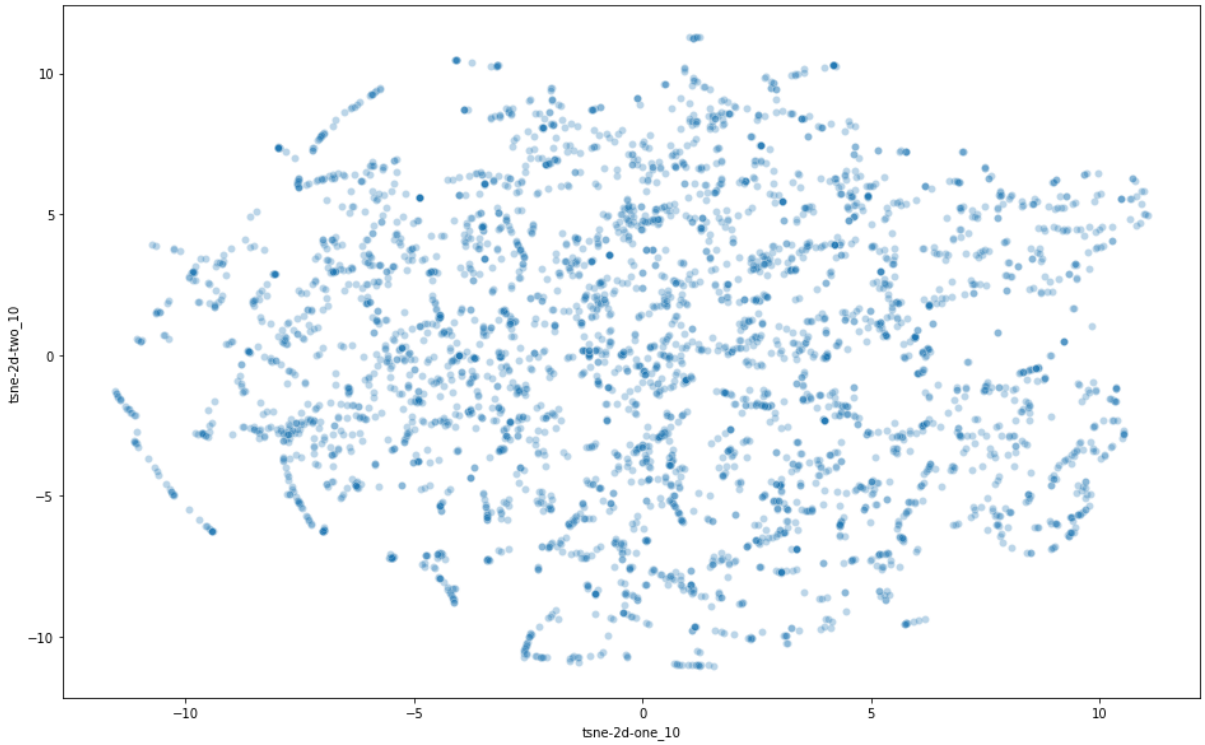

Figure 31: t-SNE 2d plot. Perplexity=10

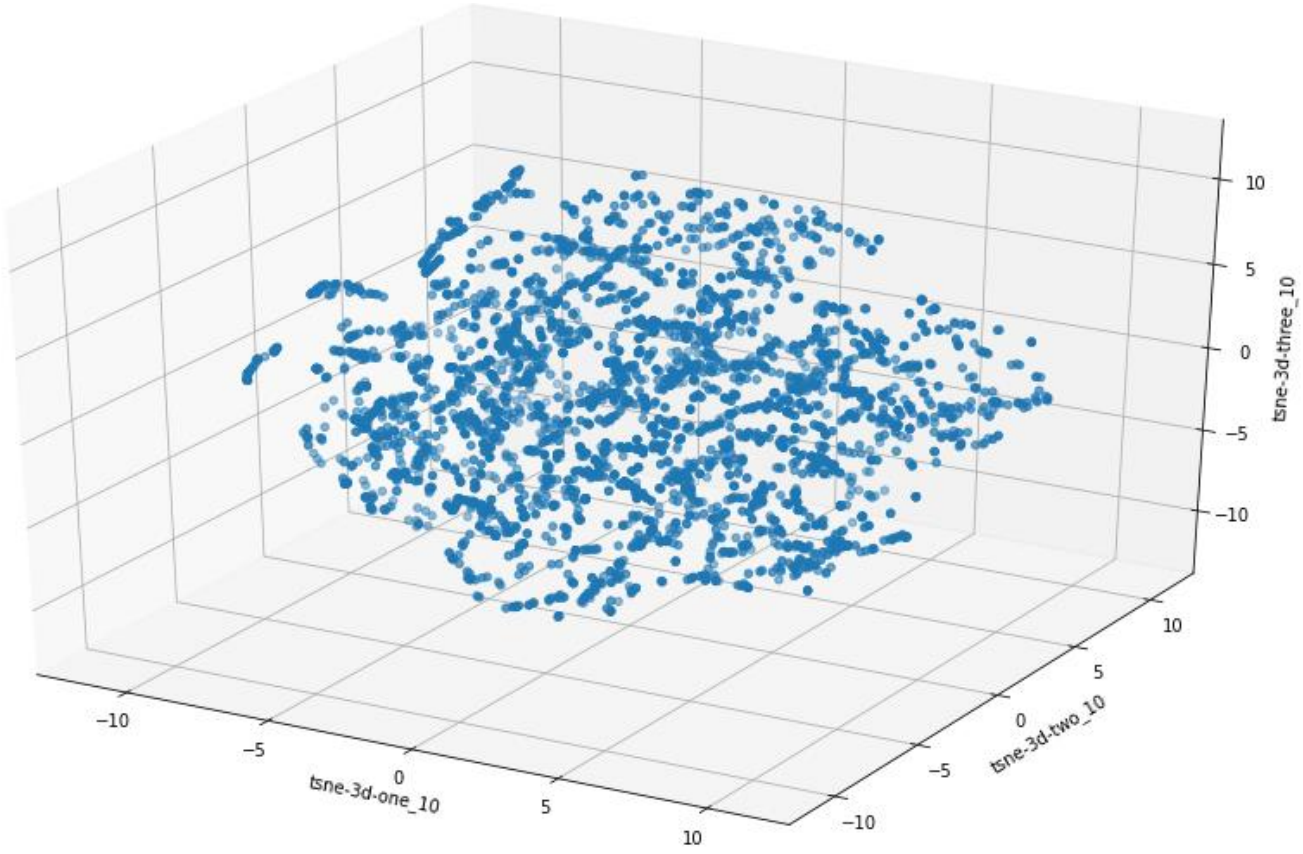

Figure 32: t-SNE 3d plot. Perplexity=10

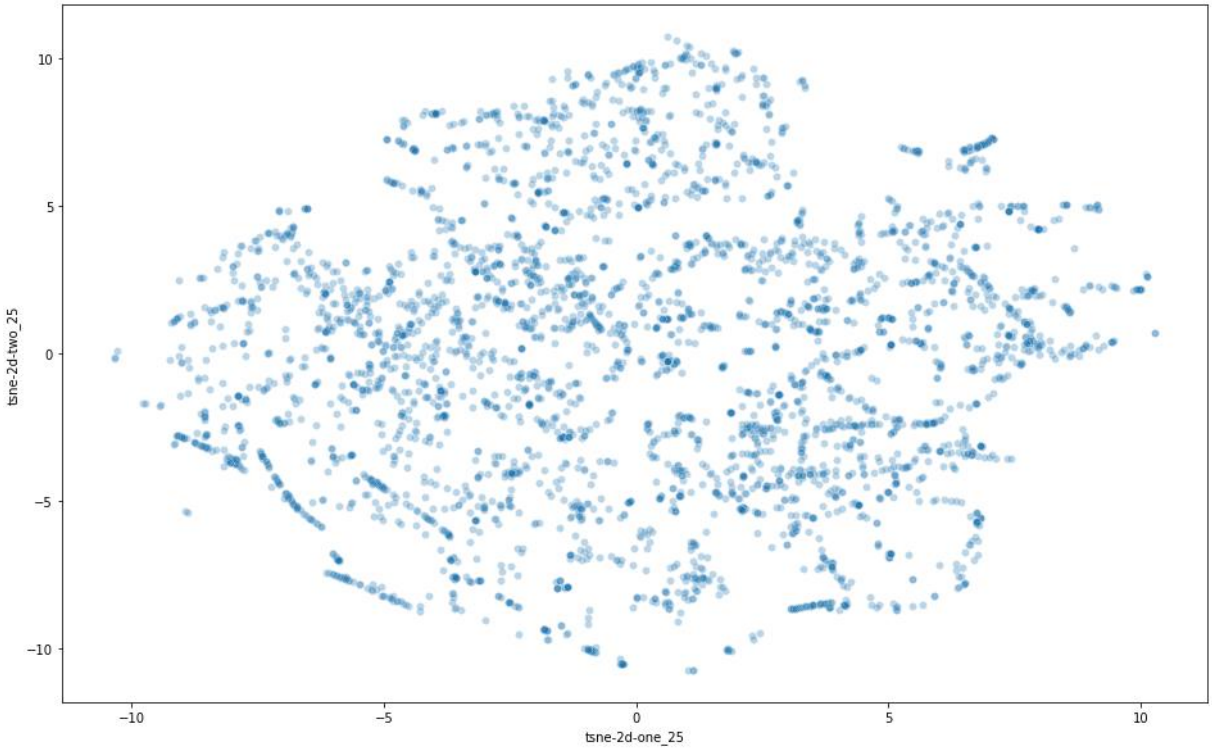

Figure 33: t-SNE 2d plot. Perplexity=25

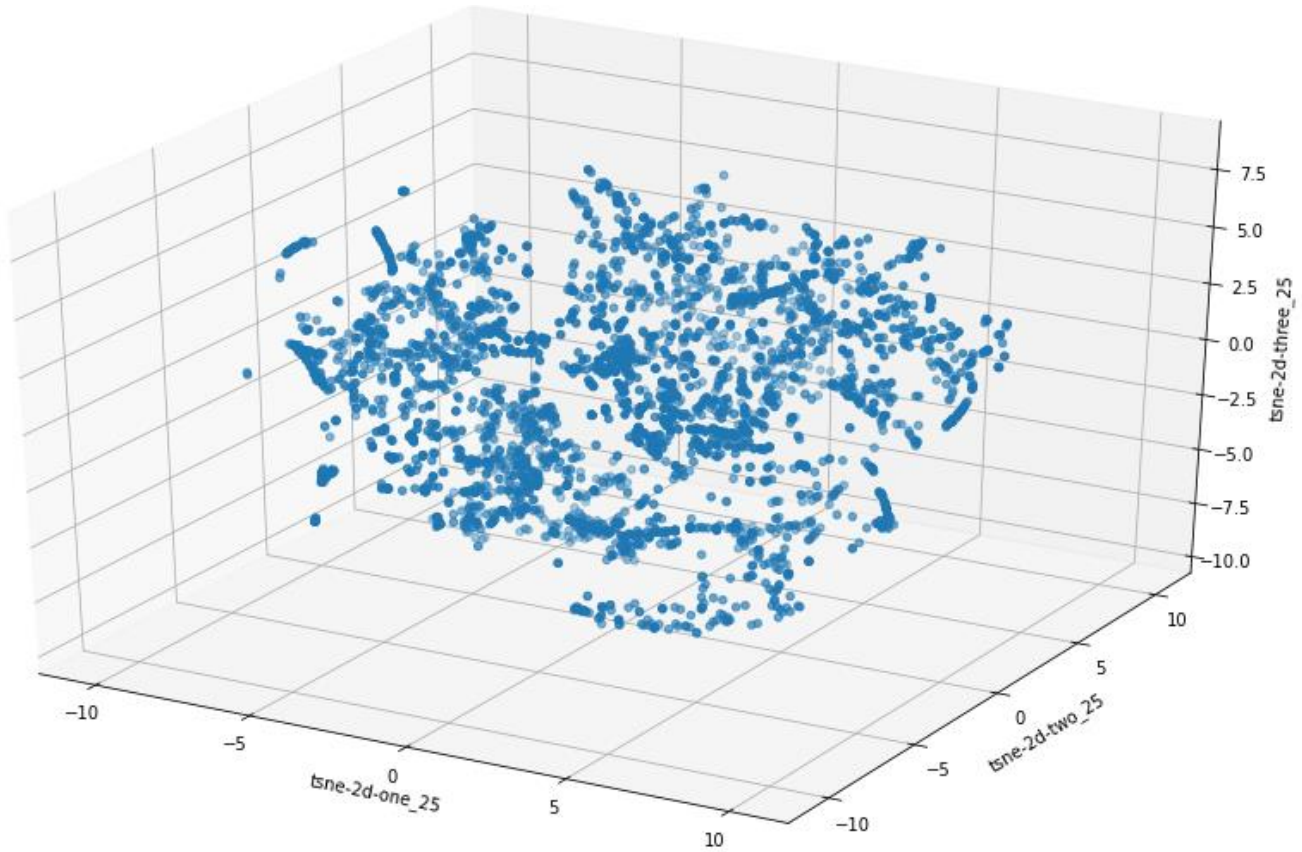

Figure 34: t-SNE 3d plot. Perplexity=25

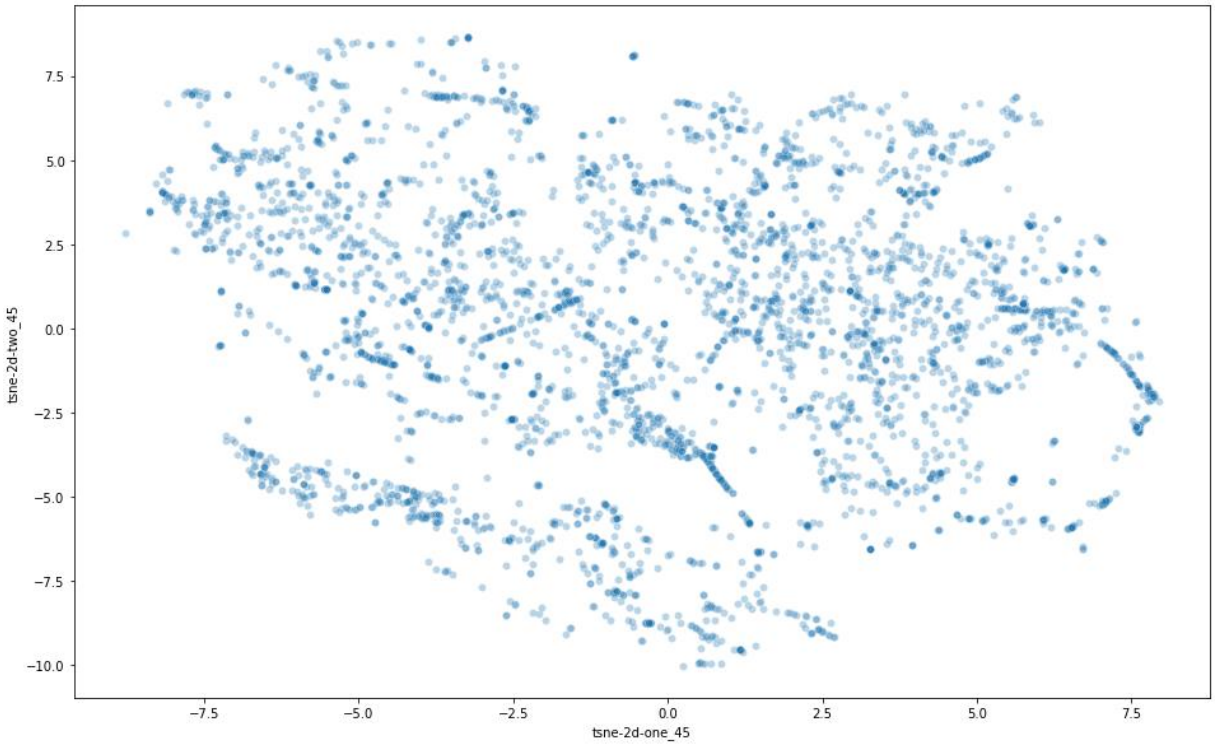

Figure 35: t-SNE 2d plot. Perplexity=45

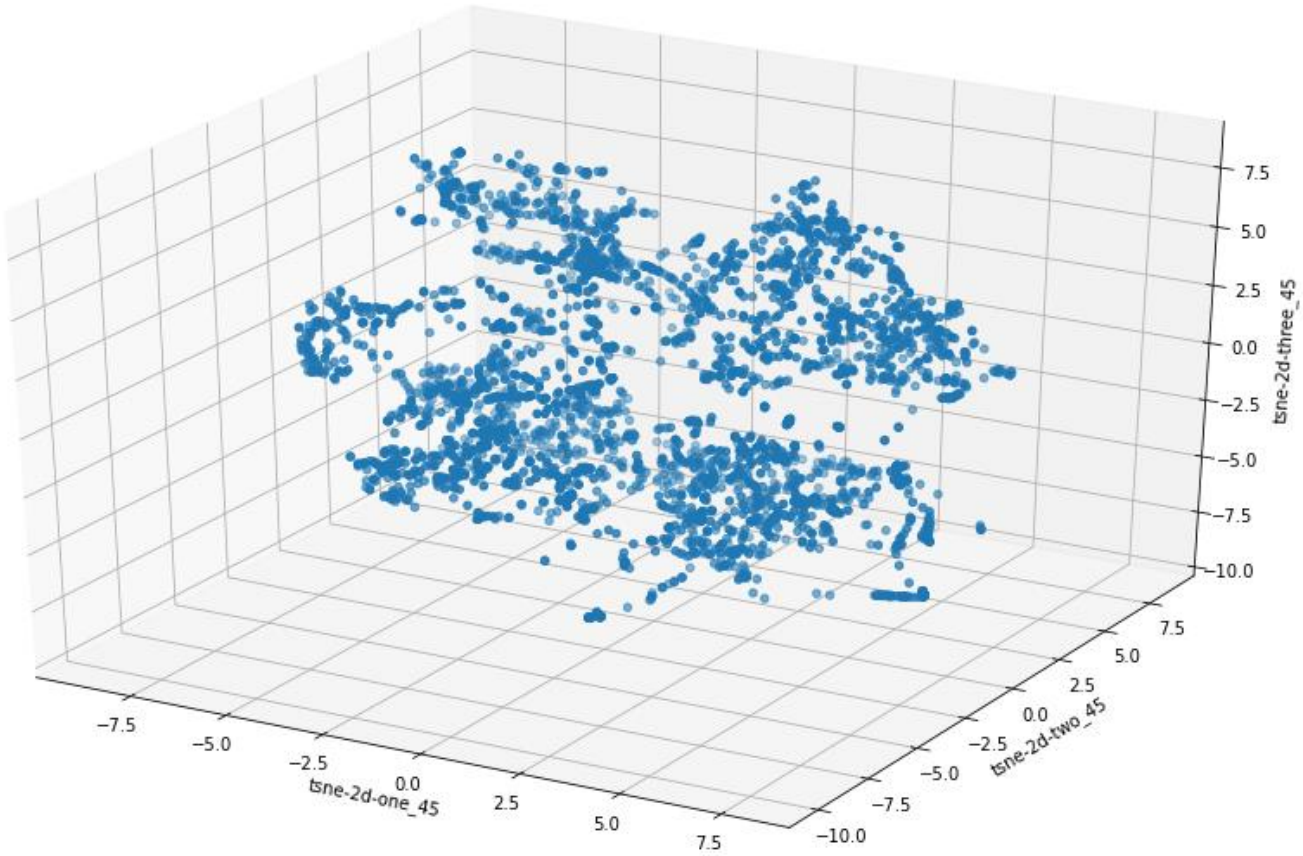

Figure 36: t-SNE 3d plot. Perplexity=45

Considering the following facts that:

1. The hyperparameters are important.
2. Cluster sizes in a t-SNE plots do not necessarily reflect the true cluster sizes.
3. Distances between clusters in t-SNE plots might not reflect the true distances between clusters in original high-dimensional space.
4. Random noise in original space might not look random in the t-SNE reduced space.
5. True distributions can be seen sometimes.
6. For topology, more than one plot may be needed.

Visual inspection of multiple plots with different hyperparameters is necessary to get an intuition of the distribution of the data in the actual high-dimensional space [12]. By examining the above t-SNE plots, though it is not possible to find obvious cluster patterns, at least the fact that the data is not randomly or uniformly distributed in the actual high-dimensional space comes to light. The data not being randomly or uniformly distributed conversely mean that it is clusterable.

5.4 VAT

Figure 37 depicts the output for the iVAT algorithm, which is an improved version of the VAT algorithm. The iVAT algorithm produces a more precise image at a cost of heavier computing. It is evident from Figure 37 that several black squares with varying sizes are seen along the diagonal indicating the presence of clusters in the alcohol dataset. In addition, the plot shows us how big the clusters are. Close examination of the plot also reveals the presence of dense clusters nested inside less denser clusters. The aim of an optimum clustering solution would be to capture a number of clusters as close as possible to the number of black squares seen along the diagonal.

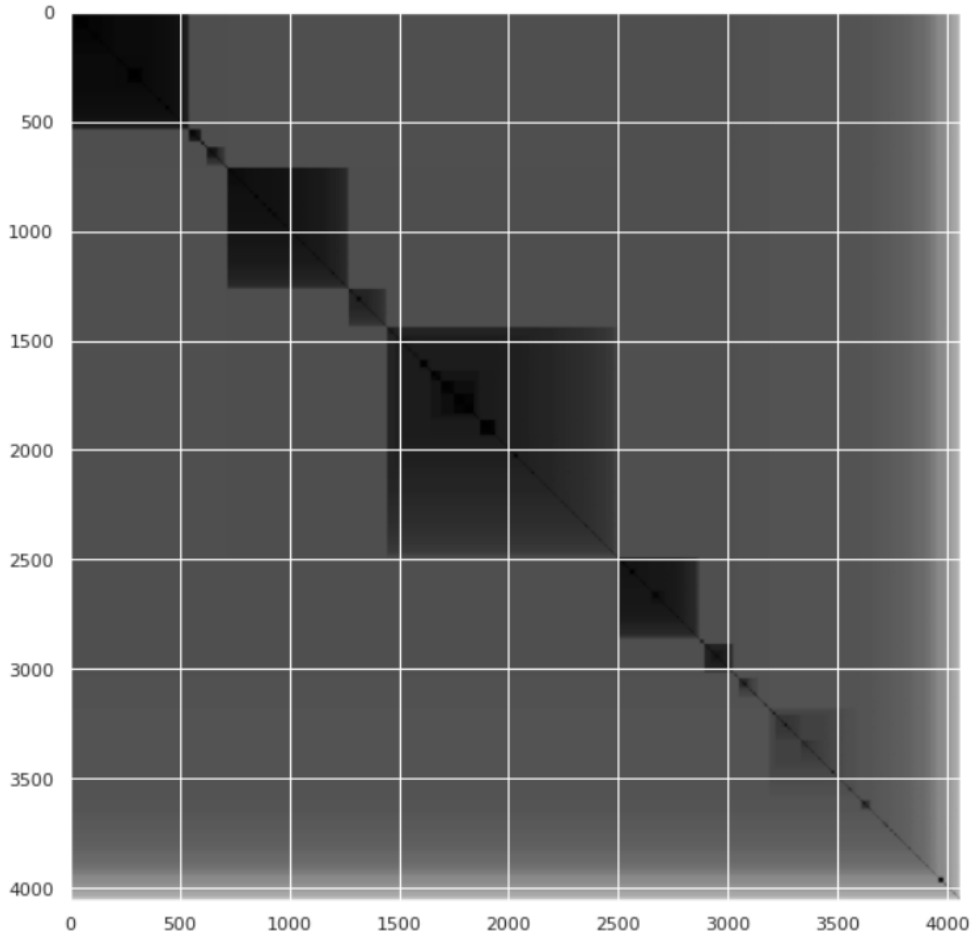

Figure 37: Visual assessment of clustering tendency

5.5 Hopkins Statistic

A Hopkins statistic value of 0.057 was obtained for the Alcohol dataset indicating that it is likely the dataset contains statistically significant clusters.

5.6 Optimal Number of Clusters

As discussed in section 3.1.3 the optimum number of clusters is obtained for different clustering methods for the alcohol dataset by majority voting of 30 indices, shown in Figure 12, using the package NbClust in R.

Table 4: Optimal number of clusters.

Method	Number of clusters
K-means	3
Partition Around Medoids	2
Hierarchical clustering	4

5.7 K-Means Clustering

The distribution of data objects among different clusters, after K-means Clustering, is shown in Figure 38 and Table 5. K-means clustering was able to partition the data into just three groups based on the optimum K-means model that exists i.e. it captures just tree clusters with approximately 34% of the observations in cluster 0, 53% of the observations in cluster 1 and 13% of the observations in cluster 2. The cluster names 0, 1, 2 mean nothing in specific and is used just for the purpose of notation for different clusters. The clustering solution obtained from K-means can be compared to the output obtained from VAT in section 5.4 to check for the expected number of clusters and their sizes. This is not exactly the kind of clustering result that is expected to be achieved, so we continue trying other clustering methods on the alcohol dataset.

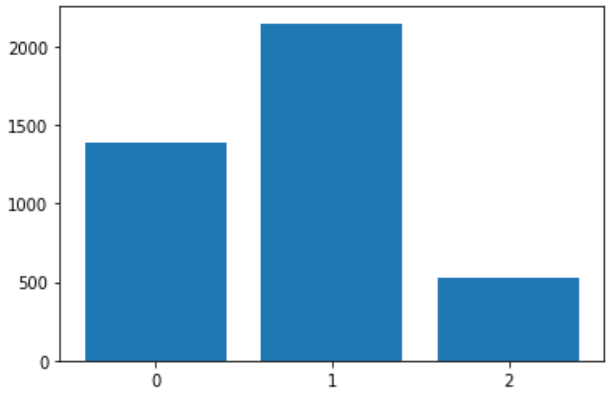

Figure 38: K-means clustering data distribution.

Table 5: K-means clustering data distribution.

Cluster	Number of data objects
0	1388
1	2145
2	528

5.8 Partition Around Medoids

The distribution of data objects among different clusters, after Partition Around Medoids clustering, is shown in Figure 39 and Table 6. PAM clustering was able to partition the data into just four groups based on the optimum PAM model that exists i.e. it captures just four clusters with approximately 19% of the observations in cluster 2, 14% of the observations in cluster 1, 38% of the observations in cluster 3 and 29% of the observations in cluster 0. The cluster names 0, 1, 2, 3 mean nothing in specific and is used just for the purpose of notation for different clusters. The clustering solution obtained from PAM can be compared to the output obtained from VAT in section 5.4 to check for the expected number of clusters and their sizes. Similar to K-means clustering, this is not exactly the kind of clustering result that is expected to be achieved, so we continue trying other clustering methods on the alcohol dataset.

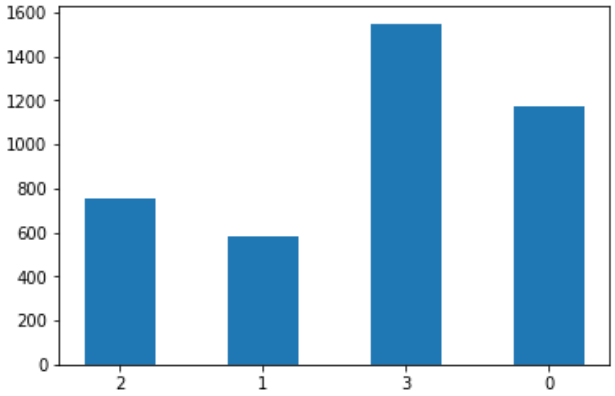

Figure 39: K-medoids clustering data distribution

Table 6: K-medoids clustering data distribution.

Cluster	Number of data objects
2	756
1	584
3	1549
0	1172

5.9 Agglomerative Hierarchical Clustering

The distribution of data objects among different clusters, after Agglomerative Hierarchical Clustering, is shown in Figure 40 and Table 7. Agglomerative Hierarchical Clustering was able to partition the data into just two groups based on the optimum Agglomerative Hierarchical Clustering model that exists i.e. it captures just two clusters with approximately 57% of the observations in cluster 1 and 43% of the observations in cluster 2. The cluster names 1 and 2 mean nothing in specific and is used just for the purpose of notation for different clusters. The clustering solution obtained from Agglomerative Hierarchical Clustering can be compared to the output obtained from VAT in section 5.4 to check for the expected number of clusters and their sizes. Similar to K-means clustering and PAM, this is not exactly the kind of clustering result that is expected to be achieved, so we continue trying other clustering methods on the alcohol dataset.

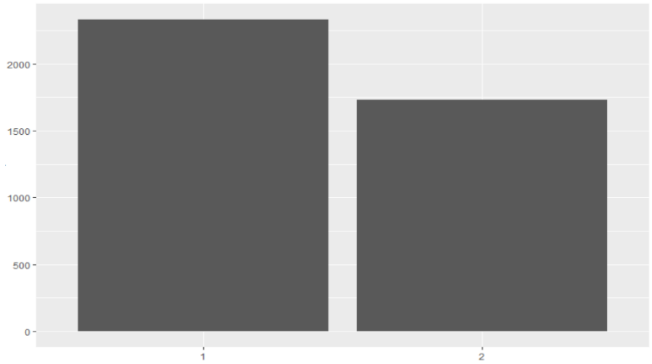

Figure 40: Hierarchical clustering data distribution

Table 7: Hierarchical clustering data distribution.

Cluster	Number of data objects
1	2332
2	1729

5.10 OPTICS

In the reachability plot in Figure 41, curves of different colors indicate different clusters. All data objects marked as black dots are the points classified as noise. The clustering solution, the data objects grouped into clusters and the noise points change with the change in epsilon parameter as demonstrated in Figure 42 and Figure 43. The clustering solution of interest is the one in Figure 41. Figures 42 and 43 are used just for the illustrative purpose.

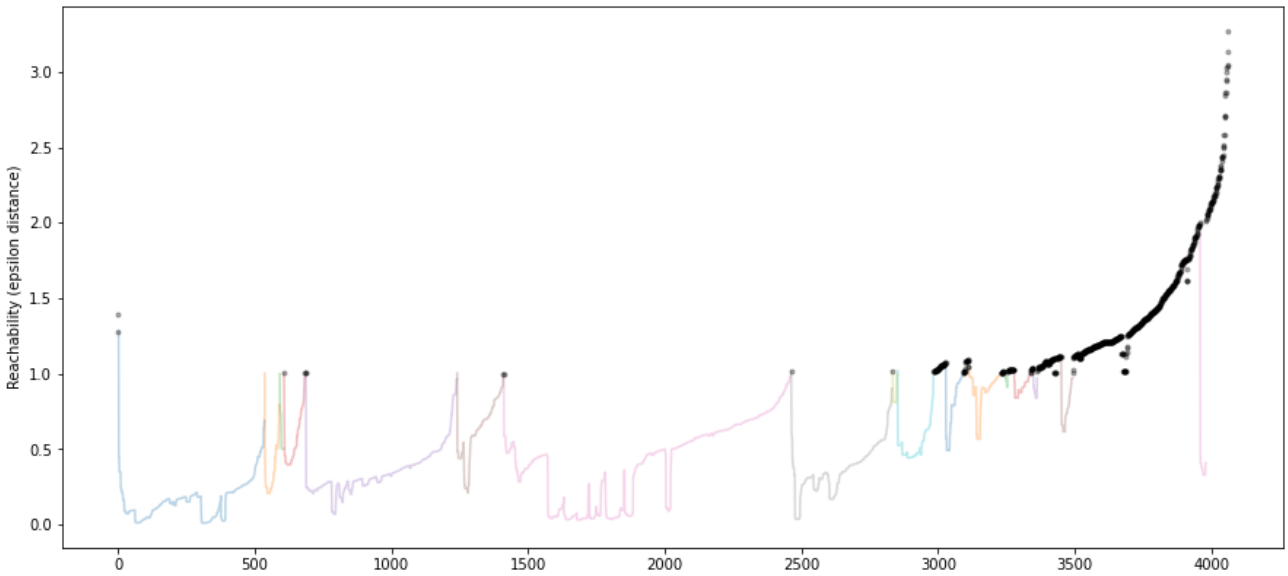

Figure 41:Reachability plot, epsilon=1

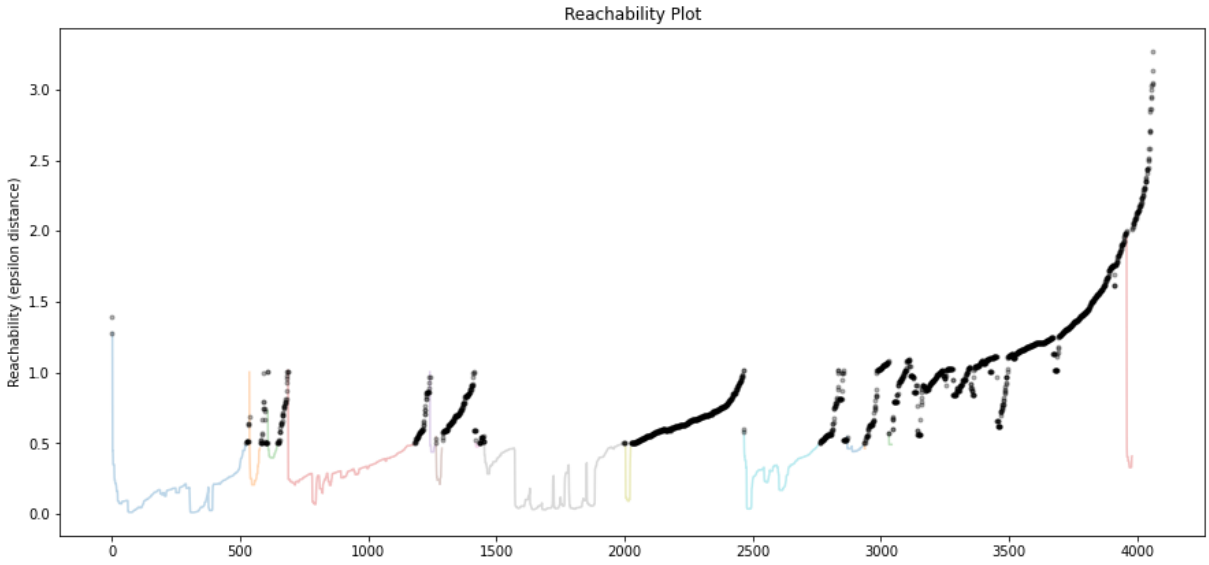

Figure 42: Reachability plot, epsilon=0.5

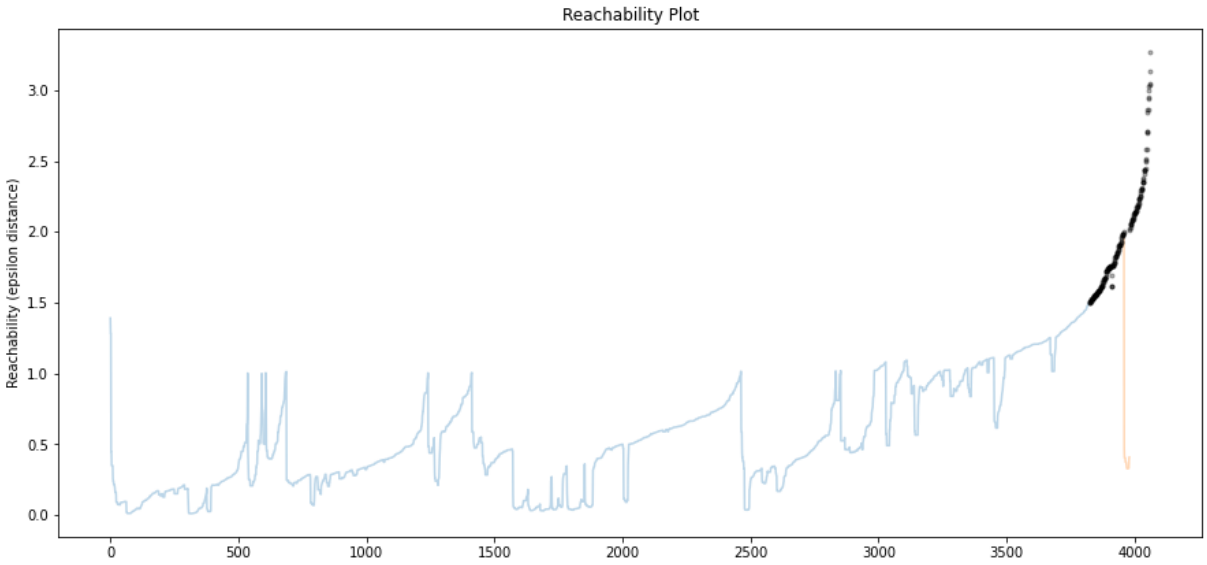

Figure 43: Reachability plot, epsilon=1.5

The distribution of data objects among different clusters, after clustering using DBSCAN on the cluster ordering obtained by OPTICS, is shown in Figure 44 and Table 8. The cluster name -1 indicates noise points and the cluster names 0,1,2…16 mean nothing in specific and is used just for the purpose of notation for different clusters. The clustering solution obtained from a combination of OPTICS and DBSCAN can be compared to the output obtained from VAT in section 5.4 to check for the expected number of clusters and their sizes. In comparison to K-means clustering, PAM and Agglomerative Hierarchical clustering this method seems to be the best since it captures a number of clusters as close as possible to the number of dark squares seen along the diagonal in the VAT plot in section 5.4. This also indicates that the clusters in the dataset is arbitrarily shaped which is possible to be captured only by this method and not the previous three methods.

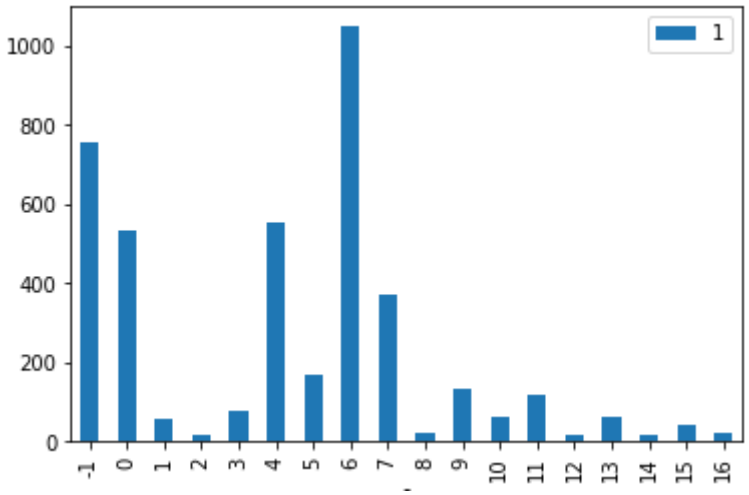

Figure 44: DBSCAN data distribution

Table 8: DBSCAN data distribution.

Cluster	Number of data objects
-1	757
0	534
1	55
2	15
3	77
4	553
5	169
6	1050
7	369
8	19
9	132
10	63
11	117
12	13
13	59
14	15
15	42
16	22

5.11 Cluster Validation

Though it is not possible to compare different clustering algorithms such as partition-based methods and density-based methods due to different notions of clusters and different objective functions, in this particular case it is possible to compare the clustering results obtained from different algorithms using the output from VAT in section 5.4. The output from VAT indicates presence of multiple clusters which is captured only by OPTICS as shown in section 5.10.

5.12 K-Nearest Neighbors

Table 9 shows the accuracy and optimized parameter value for K, i.e. the number of nearest neighbors, obtained for different folds of the dataset which aggregates to the entire dataset. Using cross-validation for model selection helps avoid biased or overly optimistic performance estimates of the model. Using nested cross-validation ensures hyperparameter tuning and output of optimized hyperparameter values for each fold of the dataset. The classification accuracy is calculated as the proportion of the samples correctly classified. The accuracies over different folds remain fairly constant without a lot of variation which is a result of stratified cross-validation. The optimum parameter value for the number of nearest neighbors to be considered for classification of a new data point remain constant at a value of 2 over all the folds. Based on the following observations, a K-nearest neighbor classification model with K=2 nearest neighbors can be trained using the entire dataset and used for predicting the classes of new samples. One of the issues with implementing this model is that a parameter value of 2 for the number of nearest neighbors may be seen as a sign of overfitting and the model might not be able to generalize well on new data.

Table 9: Accuracy and optimized parameter value for k-nearest neighbor classifier.

Accuracy	Nearest neighbors
0.978	2
0.974	2
0.958	2
0.978	2
0.968	2
0.974	2
0.984	2
0.968	2
0.962	2
0.974	2
0.971	2
0.974	2
0.984	2

A mean accuracy of 0.973 with a standard deviation of 0.007 was obtained.

5.12 Random Forests

Table 10 shows the accuracy and optimized parameter value for n_estimators, i.e. the number of trees to be built in the Forest, obtained for different folds of the dataset which aggregates to the entire dataset. Using cross-validation for model selection helps avoid biased or overly optimistic performance estimates of the model. Using nested cross-validation ensures hyperparameter tuning and output of optimized hyperparameter values for each fold of the dataset. The classification accuracy is calculated as the proportion of the samples correctly classified. The accuracies over different folds remain fairly constant without a lot of variation which is a result of stratified cross-validation. The optimum parameter value for the number of trees to be built in the Forest changes over different folds.

This may be due to the way Random Forests are designed to work wherein Decision Trees are built using the bootstrapped samples. Some of the samples excluded in the bootstrapped set i.e. the out-of-bag samples might be very different from those in the bootstrapped set. This might lead to construction of more trees to be able to capture the structure of the out-of-bag samples. A second reason might be that the variables are chosen at random to be the candidates at each node for each tree leading to a selection of different variable each time at the same node of the same tree.

Based on the following observations, a Random Forest classifier with n_estimators = 145 can be trained using the entire dataset and used for predicting the classes of new samples.

Table 10: Accuracy and optimized parameter value for Random Forest classifier.

Accuracy	n_estimators
0.974	70
0.965	75
0.971	125
0.962	125
0.965	85
0.978	80
0.990	145
0.974	125
0.962	130
0.974	95
0.962	70
0.971	70
0.981	75

A mean accuracy of 0.971 with a standard deviation of 0.008 was obtained.

5.13 Adaptive Boosting

Table 11 shows the accuracy and optimized parameter value for learning_rate, i.e. the amount by which the contribution of each classifier must be shrunk, and n_estimators, i.e. the maximum number of estimators/weak learners at which boosting is terminated after a perfect fit is achieved, obtained for different folds of the dataset which aggregates to the entire dataset. Using cross-validation for model selection helps avoid biased or overly optimistic performance estimates of the model. Using nested cross-validation ensures hyperparameter tuning and output of optimized hyperparameter values for each fold of the dataset. The classification accuracy is calculated as the proportion of the samples correctly classified. The accuracies over different folds remain fairly constant without a lot of variation which is a result of stratified cross-validation. The optimum parameter value for learning_rate remain constant at a value of 0.1 for all the folds whereas it changes for n_estimators for 3 out of the 13 folds.
Based on the following observations, though the optimum value for n_estimators has two different values of 14 and 16, a Random Forest classifier with learning_rate = 0.1 and n_estimators = 14 can be considered as the best AdaBoost model that exists for this data since majority of the folds indicated 14 as the optimum parameter value for n_estimators.

This method cannot be used for classification of new samples since k-NN and Random Forest proved to provide higher classification accuracies.

Table 11: Accuracy and optimized parameter values for AdaBoost classifier.

Accuracy	learning_rate	n_estimators
0.837	0.1	14
0.827	0.1	14
0.831	0.1	14
0.818	0.1	14
0.824	0.1	14
0.821	0.1	16
0.824	0.1	14
0.811	0.1	16
0.821	0.1	14
0.821	0.1	14
0.821	0.1	14
0.827	0.1	14
0.827	0.1	16

A mean accuracy of 0.824 with a standard deviation of 0.006 was obtained.

6. Conclusion

Partition based methods – K-means clustering, Partition Around Medoids, Agglomerative Hierarchical clustering and Density based methods – OPTICS, to obtain cluster ordering, and DBSCAN, to obtain clustering solution using the cluster ordering obtained from OPTICS, were used to cluster the samples in the Alcohol dataset. It was not possible to obtain a meaningful clustering solution using K-means, Partition Around Medoids and Agglomerative Hierarchical clustering. It was possible to obtain a meaningful clustering solution using OPTICS and DBSCAN. Ground truth was built using the clusters discovered by DBSCAN.

Partition based approaches such as K-means clustering and Partition Around Medoids (PAM) is designed based on the notion of prototypes which are the representative points of clusters. The objective function involves reducing the sum of distances of data points from the prototypes in all the clusters. This very notion of a prototype and the objective function inhibits K-means and PAM from capturing non-convex or arbitrary shaped clusters.

Agglomerative Hierarchical Clustering decides locally using criteria such as Ward's method, at each step the cluster to be merged. Due to the availability of pairwise similarities of all points, there is a tendency to make good local decisions about combining two clusters, but it does not necessarily lead to a good global solution.

OPTICS and DBSCAN which is based on the notion of density of a data point, measured by the number of data points close to it, is designed to capture arbitrary shaped clusters. The results obtained is an indication that the observations in the dataset under consideration lie in arbitrary shaped clusters and the reason why the clustering solution obtained using a combination OPTICS and DBSCAN produce meaningful clustering solution.

Three classification algorithms- K-nearest neighbor, Random Forests and Adaptive Boosting were used to train the models and tested using test data set. Out of the three classification methods implemented K-nearest neighbor and Random Forests seem to be promising with accuracies of 97.3% and 97.1% respectively. Here, Random Forest is considered to be a better method to classify a new sample since the optimum parameter value of 2 obtained for number of neighbors (k) in k-NN might be seen as a sign of overfitting and that the model would not be able to generalize well on previously unseen or new data.

7. Bibliography

[1] Aitken, C., & Lucy, D. (2004). Evaluation of trace evidence in the form of multivariate data. *Journal Of The Royal Statistical Society: Series C (Applied Statistics)*, *53*(1), 109-122. doi: 10.1046/j.0035-9254.2003.05271.x

[2] Grob, R., & Barry, E. (2004). *Modern practice of gas chromatography* (4th ed.). Hoboken, N.J.: Wiley.

[3] Hastie, T., Tibshirani, R., & Friedman, J. *The elements of statistical learning*.

[4] Tan, P., Steinbach, M., Karpatne, A., & Kumar, V. *Introduction to data mining*.

[5] Han, J., Kamber, M., & Pei, J. *Data mining*.

[6] Ester, M., Kriegel, H.P., Sander, J., Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise

[7] Ankerst, M., Breunig, M., Kriegel, H., & Sander, J. (1999). OPTICS. *ACM SIGMOD Record*, *28*(2), 49-60. doi: 10.1145/304181.304187

[8] Bezdek, J., & Hathaway, R. (2002). VAT: A tool for visual assessment of (cluster) tendency. *Proceedings Of The 2002 International Joint Conference On Neural Networks*, *3*. doi: 10.1109/IJCNN.2002.1007487

[9] Charrad, M., Ghazzali, N., Boiteau, V., & Niknafs, A. (2014). NbClust: AnRPackage for Determining the Relevant Number of Clusters in a Data Set. *Journal Of Statistical Software*, *61*(6). doi: 10.18637/jss.v061.i06

[10] Breiman, L. (2001). Random Forests. *Machine Learning*, *45*, 5-32. doi: 10.1023/a:1017934522171

[11] Freund, Y., & Schapire, R. (1999). A Short Introduction to Boosting. *Journal Of Japanese Society For Artificial Intelligence*, *14*(5), 771-780. doi: 10.1023/a:1007662407062

[12] Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. J. Mach. Learn. Res. 2008, 9, 2579–2625.

[13] Cawley, G., & Talbot, N. (2010). On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *Journal Of Machine Learning Research*.

[14] Shirkhorshidi, A., Aghabozorgi, S., & Wah, T. (2015). A Comparison Study on Similarity and Dissimilarity Measures in Clustering Continuous Data. *PLOS ONE*, *10*(12). doi: 10.1371/journal.pone.0144059

[15] Williamson, D., Parker, R., & Kendrick, J. (1989). The Box Plot: A Simple Visual Method to Interpret Data. *Annals Of Internal Medicine*, *110*(11), 916-921. doi: 10.7326/0003-4819-110-11-916

[16] Guideline for the use of Chemometrics in Forensic Chemistry. DWG-CFGC-001. European Network of Forensic Science Institutes – Drugs Working Group, 2020

[17] Hartigan, J. (1975). *Clustering algorithms*. New York: Wiley.

[18] Kaufman, L., & Rousseeuw, P. (1990). *Finding Groups in Data* (3rd ed.). New York: Wiley.