Data Mining Report: Assessing and implementing clustering and classification techniques

Freddy Marten
School of Computing and Communications, Lancaster University
SCC403: Data Mining

***Abstract*—This report first concerns Climate Data from Basel, Switzerland in the Summer and Winter seasons between 2010 and 2019. We also look at a video stream of a police car chasing a motorbike. We pre-process both datasets to ready them for data mining. I will cluster the climate data, discerning the typical days of weather which form Basel's climate. By clustering with k-means, DBSCAN and OPTICS I show a broad and then more detailed picture. We then train a few classifiers capable of discerning the car from the motorbike. After tuning hyper-parameters we assess the performance of both our clustering and classification results.**

I. Introduction

A. Pre-processing Techniques Considered

1) Extreme/Missing Observations: We will search for such observations using histograms and summary statistics. Any outliers are treated based on the context of the data.

2) Feature Selection: Before using any formal techniques I considered which variables are reasonable to remove from both the data sets. With both data sets I selected pertinent columns to simplify the problem with minimal loss of information. For the Climate Data, I used the variance threshold technique from the sci-kit learn package on the remaining features to reduce redundancy even further [8], and avoid an imbalanced data set. With the Data Stream I will split some rows, to ensure our data set is more balanced when training classifiers.

3) Feature Scaling: For many subsequent methods it might be important to use min-max scaling and z-score standardisation so that they function effectively. This is especially important to the climate data since our variables are on vastly different scales.

4) Principal Component Analysis: Principal Component Analysis (PCA) is a dimensionality reduction technique which uses linear algebra to create new descriptive variables (principal components) which are linear combinations of the features. Crucially we can visualise our high dimensional data set using PCA with a measurable amount of variance explained with just 2 principal components.

B. Clustering Techniques Considered

Clustering algorithms come in a few different broad forms. I chose to consider clustering methods with significantly differences to increase the chance of identifying the most prominent clusters in the data. I rejected Hierarchical methods outright due to their sensitivity to noise and outliers [6]. Furthermore, given our variables of all numeric rather than categorical, it can be hard to follow the tree downward intuitively. For larger data sets, hierarchical algorithms suffer with high computational complexity, $O(n^2)$ as a minimum. For comparison, k-means, which also has the benefit of being straightforward to interpret has $O(n)$ computational complexity. We split this section into the three methods/orderings; K-means, DBSCAN and OPTICS.

1) K-means: K-means is essentially an optimization problem with the goal of minimizing the sum of squared errors within a pre-determined number of groups, k [3]. The algorithm provides no guarantee of a global optimum and is susceptible to reaching local optima, especially with certain data structures [3]. Despite this, k-means' straightforward interpretability is useful as a foundation to understand other methods from. Furthermore, given the complexity of climate data in general, and the fact that only the summer and winter seasons are included in our climate data, I expect the transition between the summer and winter seasons to be quite abrupt. This was the main reason why I have not considered an algorithm like fuzzy k-means - I would have employed this if the spring and autumn seasons were included.

2) DBSCAN: DBSCAN [5] was proposed as a relatively easy to use clustering algorithm that worked efficiently on large spatial databases to identify clusters of a wide range of forms. [2] claims DBSCAN's strength is in identifying clusters whilst not being too sensitive to noise. Furthermore, in the paper introducing a generalised DBSCAN, [18] highlights that it succeeds in being easy to use, with a single hyperparameter, $\text{Eps} > 0$ which sets minimum number of points surrounding a point for it to not be considered as noise. This fits our Basel climate data, we have more than two dimensions and a large number of data points. Weather is obviously not completely independent of the day previous. Despite this, classifying some days as noise may help us identify the strange attractors in our chaotic weather data [9]. However, [2] adds that DBSCAN struggles to identify multiple clusters if they are on different clustering scales. There has been evidence, [17], that DBSCAN can perform worse than K-means when performed on Weather Data. Though, [17], only used rainfall data and only from areas that can receive huge quantities of rainfall around the equator whereas our data is from much further north in Basel. Regardless, using OPTICS [1] with DBSCAN could improve performance above that of k-means.

3) OPTICS: Optics [1], is an extension of the DBSCAN algorithm which is motivated by the problem of clustering a dataset with clusters of data points at different scales. So it is not a clustering algorithm itself. For instance, in many cases if we look at an arbitrary 2-dimensional scatter plot of all of the data we may only identify two large clusters. Then zooming in to these clusters we might notice a few different clusters within these. Zooming in again on one of these clusters we might notice another set of clusters. OPTICS aims to order the dataset in a particular way such that information regarding each of these zooms or *clustering levels* as [1] calls them, is contained. Given the complex relationships that the features have with one another in our climate data, it seems possible we will have nested clusters within our different clustering levels. Accordingly, I try to provide a broader picture with k-means alongside a more complex multi-level picture with DBSCAN and OPTICS.

C. Classification Techniques Considered

When performing a classification task we first want to check whether our data is linearly separable or if a more complex non-linear classification technique might necessary. When considering the scatter plot of the full data stream, 14, we see that the our stream is linearly separable though with a few data points close to the line separating the Car and Motorbike frames. Using a simple linear support vector machine to create a significant margin between the two datasets therefore seemed appropriate.

1) Support Vector Classifier: Our support vector classifier [4] is aimed at creating an optimal hyperplane and corresponding margin. Note that the hyperplane of 2-dimensional space is simply a straight line. [15] defines a hyperplane in N-dimensional space, with $\mathbf{w} \in \mathbb{R}^N$ and $b \in \mathbb{R}$, is defined (1). With the canonical margins we simply translate the hyperplane by 1 unit in either direction giving (2).

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{1}$$

$$\mathbf{w} \cdot \mathbf{x} + b = \pm 1 \tag{2}$$

2) Random Forest: I chose a random forest because I thought it paired better with our SVM than other linear classifiers. I strongly considered using a real time classifier since I felt this would have more practical applications but had trouble implementing such techniques. A random forest has the advantage of being easy to interpret but also robust. From their discovery, [12], they have been shown to be susceptible to overfitting. However in our case I feel the linear separable nature of the data 14, allows us to get highly accurate classifiers that might be appropriate for future videos of car chases or simply identifying vehicles in traffic.

3) Performance Measures: In this report the performance measures: Accuracy, F1 Score, Recall and Precision will all be used for Classification. We have, from [16]:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

$$F_1 = \frac{2}{\text{Recall}^{-1} + \text{Precision}^{-1}} \tag{6}$$

Using this combination rather than Accuracy alone, gives us a more detailed picture of where classifiers might be under-performing. For instance with our label set - Car and Motorbike we might be able to see which vehicle is being classified more accurately in a more detailed way. Our F_1 Score is the harmonic mean of Recall and Precision, giving us a useful measure of both Recall and Precision combined.

II. Pre-processing

A. Climate Basel Data

Our climate data initially had 18 features with 1763 rows. It is very difficult to visualise data of this many dimensions clearly. Therefore we will have to use dimensionality reduction at some stage or stages in order to get to a point where we can visualise our clustering.

1) Extreme/Missing Observations: When we consider outliers in our climate data [8] highlights that we should not only consider the outliers of the single variable, but the relationships between the features in a multi-dimensional setting. This is a much more difficult thing to do especially in our case of the Basel Climate Data where it makes sense for a variable like snowfall to spike at low temperatures. Extreme weather exists and whilst it may make our data more difficult to cluster by including such outliers, the results we produce by excluding such observations might not hold as much weight. As [8] notes, though a row may be extreme it still could belong to the overarching distribution of the data. Searching and reporting and data that is not just extreme but seems like a mistake or corrupted [8] is important though data entry errors are unlikely given the data is gathered automatically, [14].

2) Feature Selection: Before performing any pre-processing I selected the features most pertinent to describing the environment. So I first removed all the minimum or maximum versions of the features, only keeping the means. I also removed wind gust as it showed very strong correlation with wind speed of 0.93. Following this we were left with 7, where we have quite low correlation across all the features. Note that mean humidity and temperature do have some significant correlation, -0.55, which is to be expected given these are related variables.

3) Feature Scaling: For both our clustering methods we standardise and normalise our data. We do this for multiple reasons. For one, our data is measured on different scales; pascals, millimetres, kilometers per hour ect., but our techniques do not account for this. Since we are utilising distance measures it is important the distance results we have are not simply stemming from the different of our features. Mathematically, min-max feature scaling (or normalisation) is performed by dividing the variables by the minimum and maximum value from each feature. So for a column x, transformed value, x' and the value at the ith row, x_i, we would have [11]:

$$x'_i = \frac{x_i - \min\{x\}}{\max\{x\} - \min\{x\}} \tag{7}$$

This ensures all our data fits within a set over the interval $[0, 1]$. Performing z-score standardisation [11] does not give our data a $\text{Normal}(0, 1)$ distribution as another name for the scaling technique, *Normalisation* might indicate. Z-score standardisation simply ensures we have a transformed mean, $\mu' = 0$ and transformed standard deviation, $\sigma' = 1$.

$$x'_i = \frac{x_i - \mu}{\sigma} \tag{8}$$

4) Dimensionality Reduction: Before clustering our Climate Dataset, I used PCA to make our five dimensional dataset more interpretable in less dimensions. Visualisation is an essential aspect of clustering and we cannot do this if our dataset has too many dimensions. Though our 1st and 2nd Principal Components are less easy to understand than say, temperature and rainfall, they are still a linear combination of these features. Our 1st Principal Component accounts for a large proportion of the variance within our data at over 42.6% as seen in 1. Adding the 2nd component we have over 65% of variance explained. So the 2d projections of principal components we display later when clustering describe 65% of the variance of the data.

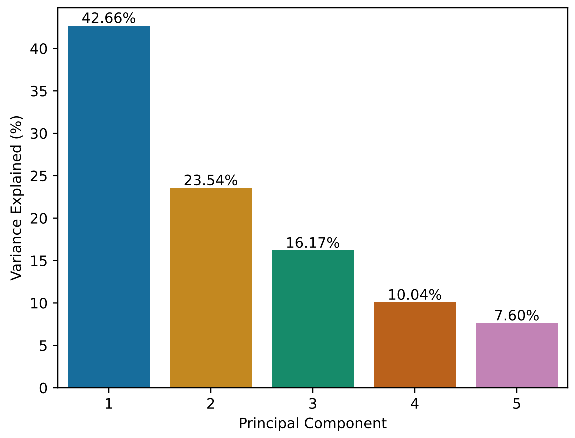

Fig. 1. Variance Explained by Principal Components

B. *Car chase data stream*

First I combined the WLA and label csv files by column. Then I looked for outliers across this full, combined dataset.

1) Extreme/Missing Observations: To detect any outliers I used a box-plot of the data shown in figure 13. Note that we account for relationships [8] between our fundamental features - Length and Width by using *area* in figure 13. I also considered these same factors with the car chase data stream, though here it is much easier to identify outliers given the smaller size of the dataset and since we have the mp4 file aswell I cross-referenced by eye and saw no signs of corruption or outliers.

2) Feature Selection: From here, I removed the first 17 rows of the dataset before the police car is visible. I split the dataset into train and test datasets, with 80% in the training dataset and 20% into the test dataset. The first 17 rows were then added to the test set so now we are not overly training our models on the motorbike object but can test models on additional data.

3) Feature Scaling: I used the min-max scaler (see 7) to scale both values down. This was the extent of feature scaling - I felt we should take advantage of the middle ground we have here between interpretability of our features and results.

III. Clustering

A crucial consideration when using clustering algorithms is the specification of the number of clusters by the user [6]. Often we use some distance measure to determine closeness of data points. Euclidean distance is a standard approach, and the one I have used in K-means, DBSCAN and OPTICs clustering. For n-dimensional vectors a and b Eucliden distance is defined:

$$d_e(a, b) = \sqrt{(a_1 - b_1)^2 + \cdots + (a_n - b_n)^2} \tag{9}$$

A. *k-means*

1) Hyper-parameter Tuning: In this report k-means uses Scikit-learn's greedy *k-means++* algorithm to choose our centers. Therefore, whilst k-means has a computational complexity of $O(n)$ [6], when using greedy k-means++ we have an approximate complexity of $O(\ell^3 \log^3 k)$[10], for candidate samples, ℓ. So the only hyper-parameter we need to be careful tuning is the number of clusters, k. When choosing the optimal number of clusters I both considered the appearance of the scatter plots 9, 2 and 10. Elbow plots of the data using both the inertia and distortion and Silhouette Scores were also useful. Of note in figure 9 is that the data is more sparse in the area around the division line. This indicates that there weaker clustering in this area which is desirable. Compared to figure 2, which does have the key areas of density in separate clustered but the separation between the dark blue and yellow clusters is less pronounced. For inertia and distortion of point p, p_{inert} and p_{distort} respectively we have (10)

$$p_{\text{inert}} = \left(\sum_{k=2}^{n} d_e(p, p_{\text{centroid}}) \right)^2 \tag{10}$$

$$p_{\text{distort}} = \frac{p_{\text{inert}}}{n} \tag{11}$$

Where n is the largest number of clusters we consider - in this case 7. So in figure 8, we are simply averaging out these two measures across all data points for different values of k.

Note that Distortion penalises greater k values more than Inertia due to the $1/n$ penalty coefficient. Increasing the value of k always decreases our average distances because we are adding new cluster points. Therefore we want to find a k which causes the remainder of the graph to be linear and approaching. For both Inertia and Distortion it appears our elbow point is either at 3 or 4 clusters. We consider both and try to distinguish our optimal value of k using 2-d projections of our clustering.

2) Results: In $k = 3$ 2 our clusters look much more convincing than with $k = 4$ in 10. In particular our cluster on the left is quite clearly divided from the others. There is some uncertainty in the divide between our yellow and dark blue clusters on the far right of the plot. Our silhouette scores were very similar for both $k = 3$ and $k = 4$, shown in table III.

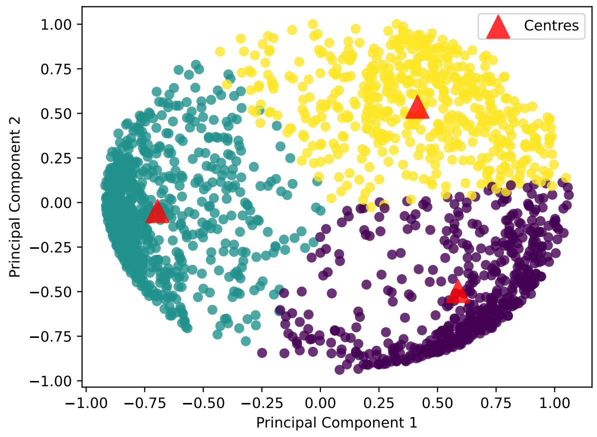

Fig. 2. 2-d projection of k-means with $k = 3$ using 1st & 2nd Principal components

B. DBSCAN

1) Hyper-parameter Tuning: We first check how DBSCAN performs prior to using OPTICs. DBSCAN is very dependent on our value of Eps, which is the parameter the algorithm is really only dependent on [5]. I used some rules of thumb for tuning the parameters outlined in [18] such as using $k = 2d+1$ for our d-dimensional dataset. So since our dataset has 5-dimensions we used $k = 11$ initially. Performing k-nearest-neighbours [7], on this data and sorting by ascending distance we produced an elbow like plot 11,[5]. We use this to choose our Eps value at the part of the elbow plot with the most curvature, approximately marked by the dashed grey line in 11 at Eps = 0.25.

2) Results: The first thing we notice is quite how many data points have been classified as noise. This leaves us with a clear clusters labelled 0 and 1 on the left and bottom right respectively. There is some overlap between the remaining

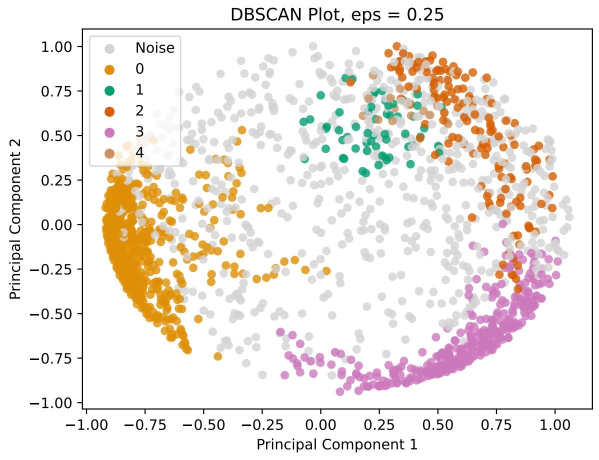

Fig. 3. DBSCAN

clusters in the top right of the plot. Note that whilst some noise is also overlapping in this area this is because our plot is a 2-dimensional projection of the PCA transformed data and therefore the other components for these points mean their euclidean distance is further from the clusters.

C. DBSCAN with OPTICS

1) Hyper-parameter Tuning: OPTICs is founded with allowing us to have a logical way of choosing our hyper-parameters with our reachability plot, 4. This is similar to 11 but now using OPTICS to order our data rather than doing some pre-processing using K-NN. Our reachability plot has multiple valleys with the bottoms of these valleys corresponding to the centers of our clusters [1]. These clusters are included in the noise if their valley is below Eps. There are also some points regarded as noise regardless of our value of Eps. With all this considered multiple different DBSCAN plots were created using OPTICS, which we examined based on the 2-d visualisation we get due to our PCA.

2) Results: First we consider the automatic OPTICS plot in figure 5. Much like k-means and our optimal DBSCAN plot, a strong cluster is identified on the left of the plot. In figure 5 this stretches to cover much of the left half. We see little noise in this region with most the noise in the middle right of the plot. There was another string cluster labelled 1 on the bottom right of the plot which was similarly identified in figure 3. The key difference between figures 5 and 3 is the much larger region of noise in figure 3. Our silhouette scores did not exceed k-means with the closest to one being 0.3 for 0.45 - this plot is shown in figure 12.

We also consider how DBSCAN changes at different epsilon cuts when using OPTICS ordering.

IV. Classification

A. Support Vector Machine

Given the linear separability of our data as discovered in our pre-processing it comes as no surprise that our support

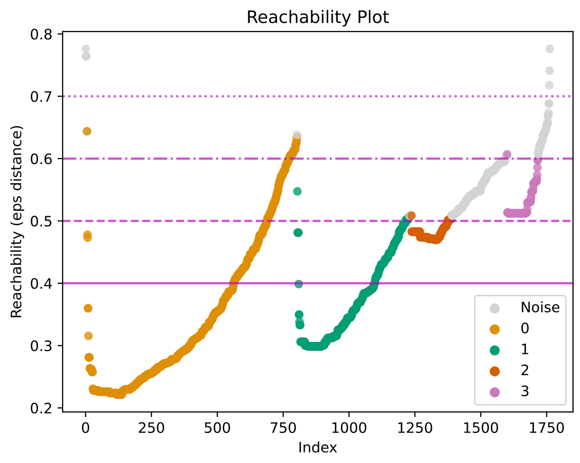

Fig. 4. Reachability Plot

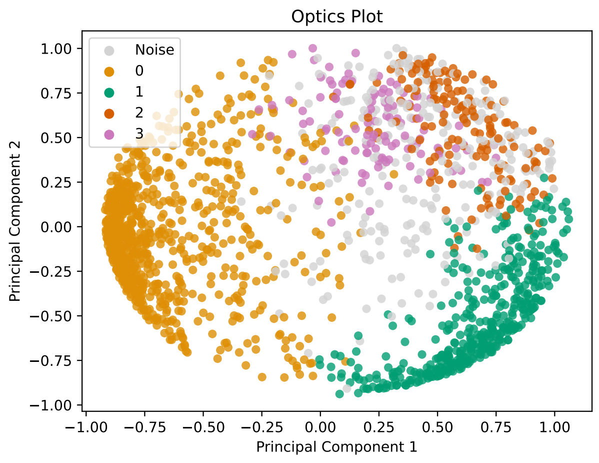

Fig. 5. Automatic OPTICS Clustering

vector machine achieves a good classification of the data.

1) Hyper-parameter Tuning: Our Support Vector Classifier did not require much parameter tuning - our F_1 Scores were 1 for all values of cost parameter $c \in [0, 50]$. By 6 notice that we could quickly check if all of our performance measures equal 1 by checking only F_1 Scores. Only when we have measures not equal to 1 do we have to consider the more detailed view provided by the four metrics together. I chose 10 since I felt the margin in figure 6 separated the data in a neat and logical way. SVM required minimal tuning to classify our data in an accurate way.

2) Results: Since we only have two dimensions we produce a similar scatter-plot to 14 but now with our margin and decision boundaries. This took quite a lot of manipulation. The form our hyperplane, 1, is not initially very useful. We manipulate this into a more useful form now.

Measure	SVM Score
Accuracy	1.000
F1 Score	1.000
Recall	1.000
Precision	1.000

TABLE I
SVM PERFORMANCE FOR ALL COST PARAMETERS BETWEEN 1 AND 50

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{12}$$

$$\begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b = 0 \tag{13}$$

$$w_1 x_1 + w_2 x_2 + b = 0 \tag{14}$$

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2} \tag{15}$$

Now from this straightforward manipulation we have an equation in a familiar linear form. Scikit learn will calculate w_1, w_2 and b so we can now form program our separating hyperplane. The only thing left is to calculate the distance of the margin is from the hyperplane

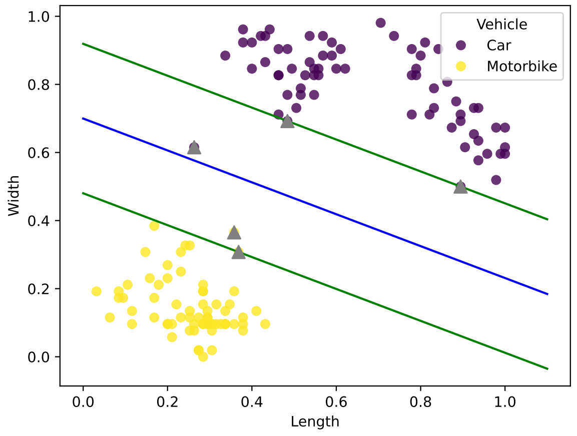

Fig. 6. Support Vector Machine margin marked in green, and separating hyperplane in blue

B. Random Forest

I initially fit a random forest with 5 estimators finding one false positive and one false negative. This is already quite a strong fit as shown in table II.

1) Hyper-parameter Tuning: Given the presence of Type I and Type II errors, I used cross-validation of the random search to adjust our parameters. I used 10-fold Randomized Cross-Validation to test tune the hyper-parameters. Note that this is quite a time-intensive process [13] even for our training data with only 136 rows it took 8.105 seconds to perform this. Therefore, another technique might be necessary if using a larger data stream sample due to long processing times.

Following this cross-validation we did have improved results II. It appears, [12]'s findings have been found again that random forests have a tendancy to ovefit.

Measure	Untuned	Tuned
Accuracy	0.981	0.942
F1 Score	0.977	0.933
Recall	1.000	1.000
Precision	0.955	0.875

TABLE II
RANDOM FOREST PERFORMANCE BEFORE AND AFTER TUNING HYPER-PARAMETERS

2) Results: Our hyper-parameter tuning decreased our scores slightly. It appears our model has overfitted to cars in this case with three motorbikes misclassified as cars. Accordingly Recall remains at 100% since in our dataset no cars have been misclassified (these are the *positive* results in our case.

V. CONCLUSION

With Basel Climate data, we have considered the clustering approaches, k-means, DBSCAN, and OPTICs and for DBSCAN and OPTICS with a range of different Eps parameters giving vastly different results. For a broad understanding of this data set, I would recommend k-means with 3 clusters- this had the lowest silhouette score and a visually appealing 2-d visualization. If a more detailed look at less noisy results is desired, the automatic OPTICS clustering is powerful - highlighting the major points of k-means but with some additional complexity. For the data stream, a linear support vector machine classifier is definitely advised based on my findings. A random forest is quite accurate, but seems less suited to this dataset. Results and implementation were easier and better with our svc.

REFERENCES

[1] Mihael Ankerst et al. "OPTICS". eng. In: *SIGMOD record* 28.2 (1999), pp. 49–60. ISSN: 0163-5808.

[2] Panthadeep Bhattacharjee and Pinaki Mitra. "A survey of density based clustering algorithms". In: *Frontiers of Computer Science* 15 (2021), pp. 1–27.

[3] Yizong Cheng. "Mean shift, mode seeking, and clustering". In: *IEEE transactions on pattern analysis and machine intelligence* 17.8 (1995), pp. 790–799.

[4] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". eng. In: *Machine learning* 20.3 (1995), pp. 273–297. ISSN: 0885-6125.

[5] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise". In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[6] Absalom E. Ezugwu et al. "A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects". In: *Engineering Applications of Artificial Intelligence* 110 (2022). Cited by: 158. DOI: 10.1016/j.engappai.2022.104743.

[7] Evelyn Fix and Joseph Lawson Hodges. "Discriminatory analysis: Nonparametric discrimination: Small sample performance". In: (1952).

[8] Karina Gibert, Miquel Sànchez-Marrè, and Joaquín Izquierdo. "A survey on pre-processing techniques: Relevant issues in the context of environmental data mining". eng. In: *Ai communications* 29.6 (2016), pp. 627–663. ISSN: 0921-7126.

[9] James Gleick. *Chaos making a new science*. eng. New York, N.Y.: Open Road Integrated Media, 2011. ISBN: 9781453210475.

[10] Christoph Grunau et al. *A Nearly Tight Analysis of Greedy k-means++*. 2022. arXiv: 2207.07949 [cs.DS].

[11] Jiawei Han, Micheline Kamber, and Jian Pei. "3 - Data Preprocessing". In: *Data Mining (Third Edition)*. Ed. by Jiawei Han, Micheline Kamber, and Jian Pei. Third Edition. The Morgan Kaufmann Series in Data Management Systems. Boston: Morgan Kaufmann, 2012, pp. 83–124. ISBN: 978-0-12-381479-1. DOI: https://doi.org/10.1016/B978-0-12-381479-1.00003-4.

[12] Tin Kam Ho. "Random decision forests". In: *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE. 1995, pp. 278–282.

[13] Sergey Kucheryavskiy et al. "Procrustes Cross-ValidationA Bridge between Cross-Validation and Independent Validation Sets". eng. In: *Analytical chemistry (Washington)* 92.17 (2020), pp. 11842–11850. ISSN: 0003-2700.

[14] Michaelaschloegl. *Weather archive Basel*. Dec. 2023. URL: https://www.meteoblue.com/en/weather/historyclimate/weatherarchive/basel_switzerland_2661604.

[15] Mehryar Mohri. *Foundations of machine learning*. eng. Second edition. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2018. ISBN: 9780262039406.

[16] David L Olson and Dursun Delen. *Advanced data mining techniques*. Springer Science & Business Media, 2008, p. 138.

[17] G C Pamuji and H Rongtao. "A Comparison study of DBScan and K-Means Clustering in Jakarta rainfall based on the Tropical Rainfall Measuring Mission (TRMM) 1998-2007". In: *IOP Conference Series: Materials Science and Engineering* 879.1 (July 2020), p. 012057. DOI: 10.1088/1757-899X/879/1/012057. URL: https://dx.doi.org/10.1088/1757-899X/879/1/012057.

[18] Jörg Sander et al. "Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications". eng. In: *Data mining and knowledge discovery* 2.2 (1998), pp. 169–194. ISSN: 1384-5810.

ACKNOWLEDGEMENTS

The following were all very useful.

- SCC403 labs and lectures

- Base Python (version 3.11) was used, documentation https://docs.python.org/3.11/library/index.html
- Sci-kit learn documentation https://scikit-learn.org/stable/user_guide.html
- Pandas, https://pandas.pydata.org/docs/user_guide/index.html#user-guide
- https://www.geeksforgeeks.org/pandas-tutorial/?ref=shm
- Seaborn, https://seaborn.pydata.org/tutorial.html
- Numpy, https://numpy.org/doc/1.26/user/index.html#user
- Matplotlib https://matplotlib.org/stable/users/index.html was extremely useful
- Variance thresholds Towards Data Science Various for feature selection
- For optics, https://www.geeksforgeeks.org/ml-optics-clustering-implementing-using-sklearn/
- For svm, How to plot a decision boundary with margins in 2d space
- Determining Epsilon and MinPts parameters of DBSCAN clustering https://sefidian.com/2022/12/18/how-to-determine-epsilon-and-minpts-parameters-of-dbscan-clustering/
- Scipy, https://docs.scipy.org/doc//scipy/tutorial/index.html#user-guide
- For accuracy measures https://proclusacademy.com/blog/practical/precision-recall-f1-score-sklearn/whenasessingaccuracy
- 17 clustering algorithms used in data science when initially choosing my clustering algorithms.
- For developing my understanding of PCA Making sense of PCA

Appendix

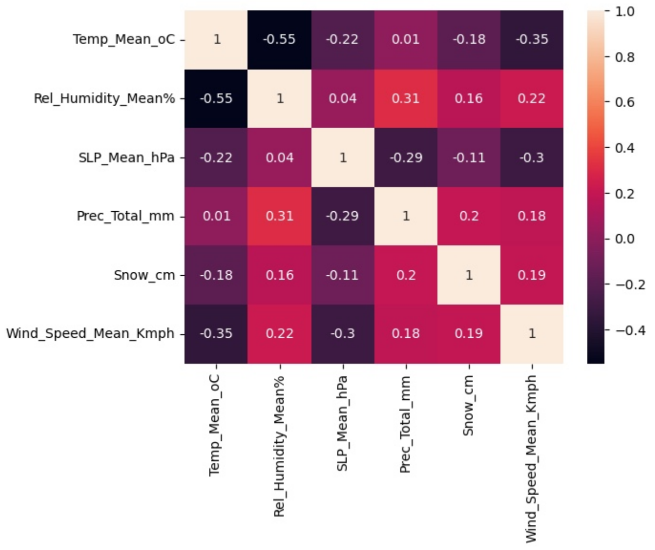

Fig. 7. Correlation Matrix

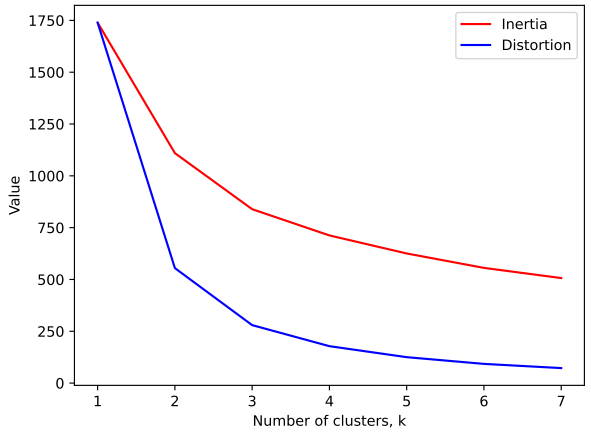

Fig. 8. Elbow Plot

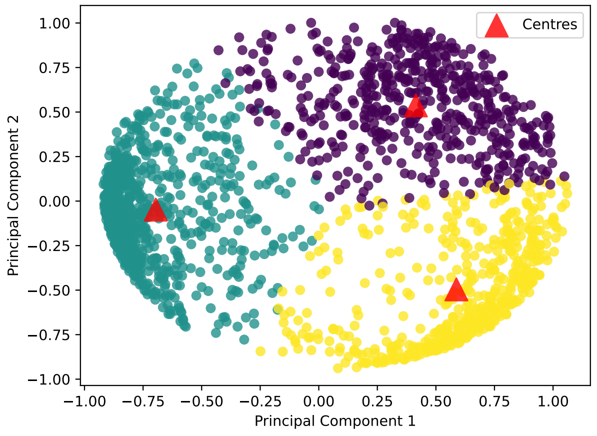

Fig. 9. 2-d projection of k-means with $k = 2$ using 1st & 2nd Principal components

k	Silhouette Score
4	0.362
3	0.362
2	0.347
6	0.320

TABLE III
SILHOUTTE SCORES FOR K-MEANS IN DESCENDING ORDER

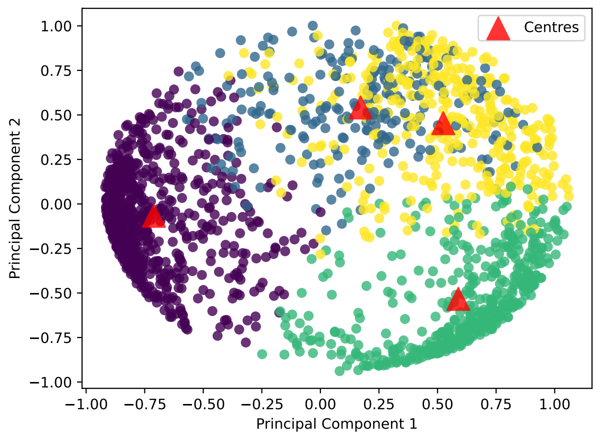

Fig. 10. 2-d projection of k-means with $k = 4$ using 1st & 2nd Principal components

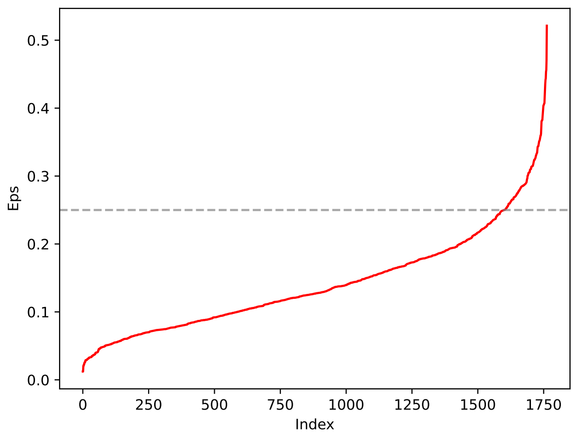

Fig. 11. Elbow Plot for DBSCAN using K-Nearest-Neighbour ordering. Approximate threshold point marked in grey.

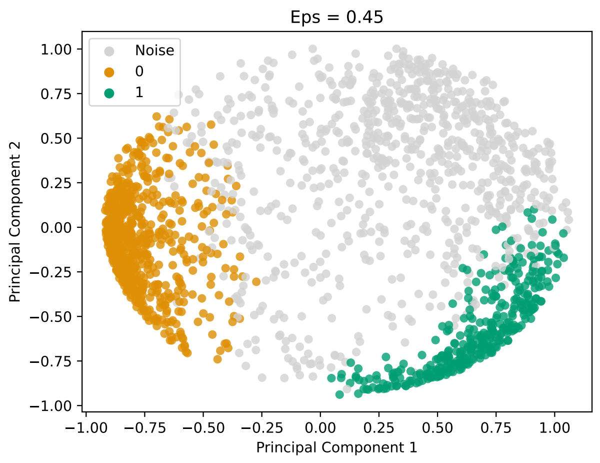

Fig. 12. DBSCAN Epsilon cutoff at Eps=0.45 OPTICS Clustering

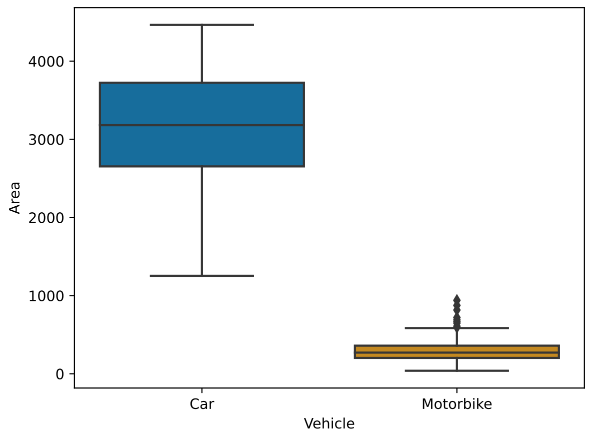

Fig. 13. Box Plot of Full Data Stream Dataset (incl. first 16 rows)

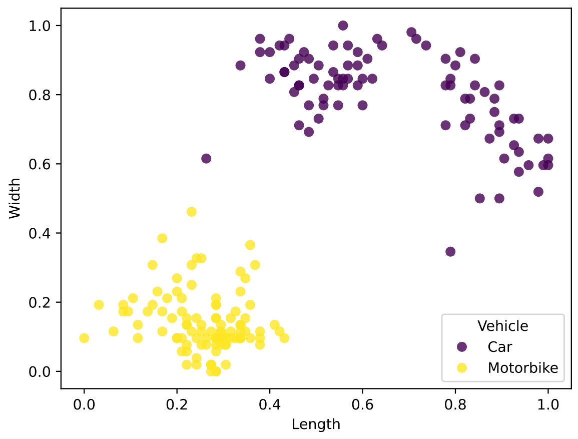

Fig. 14. Scatter Plot of Full Data Stream Dataset (incl. first 16 rows)