Unsupervised Clustering of Swiss Climate Data with AutoML

Kieran B. A. Molloy
Faculty of Science and Technology
Lancaster University
UK
k.molloy@lancaster.ac.uk

***Abstract*—Using various dimensionality reduction algorithms; PCA, PFA and a custom PCA L1-Norm, to reduce a large dataset to perform unsupervised clustering and compare the performance of various clustering methods, including analysis of the optimal number of clusters and scoring of clustering. Then classifying the dataset using SVM as well as using the Auto-Sklearn library to efficiently evaluate the hyperparameters for the problem including creating an MLP extension and testing a zeroconf approach.**

***Index Terms*—PCA, PFA, K-Means, Birch, Spectral, Mini-Batch K-Means, SVM, SVC, ROC, HPO, SMAC, LDA, NB**

I. Introduction

Data mining is the process of pulling out patterns from large datasets using multiple statistical and computational methods. The future aims of data mining is to minimise the computational burden of high-dimensional data to allow increasingly large datasets to be investigated. Figure 1 demonstrates the proposed stages and methods that will be applied at those stages, additional emphasis will be applied to feature selection specifically the Principal Factor Analysis (PFA) algorithm to reduce the dimensionality of the problem based upon the information content. The initial pre-processing stage mainly looks at stanardisation, checking for (Not a Number) NaN values (or categorical values) and then outputting a clean dataset for further analysis. Then the second stage of pre-processing, a custom L1-Norm PCA algorithm is created and its benefits over a standard PCA (L2 PCA). Clustering looks at generating labels for similar data, the different methods can give very different results due to the way they calculate distance. Finally classification creates a model that can be used to label future information. The provided dataset is climate data collected in Basel, Switzerland and contains 1763 records from the summer and winter seasons from 2010 to 2019.

II. Pre-Processing

The initial step of a classification problem is to investigate the provided dataset, ensuring it is clean. Additionally some of the methods applied in later sections require data to be in a specific format, such as within an interval of $\mathbf{x} \in [0, 1]$. For the dataset at hand, the first step is to check for missing values of which there are none. Additionally there are no categorical values, hence no one-hot encoding or alterior methods are required. The second step is transforming the data to a known plane using normalisation, standardisation or equivalent techniques. Standardisation over the interval $[-1, 1]$ was attempted, but caused problems with other algorithms, as did normalisation, and so a simple standardisation over $[0, 1]$ resulting in column-wise, $\overline{x} = 0$ and $\sigma = 1$. Additionally, feature binarisation was considered for the snow, and rain indicators however this was later replaced by data subsetting (see Section III) as this was found to give better clustering performance. Furthermore, K-Bins discretisation for the sunshine variable was able to reduce variability and increase clustering performance too.

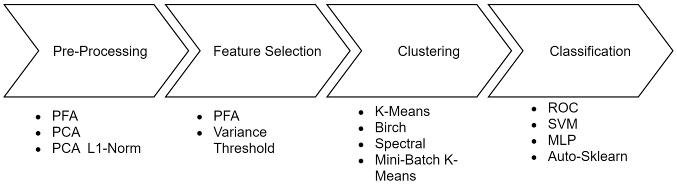


Fig. 1. Proposed Pipeline for analysis

III. Feature Selection

Data is often represented in high dimensional matrix form, but many features are often redundant, noisy or correlated, [1]. Which can result in poor performance or over fitting. This problem can be resolved by reducing the number of features being considered, there are various strategies, with varying results and speed [2]. As described in [3], feature selection, from a label availability perspective, can be categorised into Supervised Feature Selection and Unsupervised Feature Selection. The dataset considered requires unsupervised feature selection as there are no ground truth labels. Initially the dataset is split on multiple conditions, the first is snowfall. Splitting on the condition of $x > 0$ to create two datasets, df_standard and df_snowing. The standard subset is then split again for rainfall, on the condition $x > 0$ creating df_raining and df_standard.

A. Supervised Feature Selection (SFS)

Supervised Feature Selection methods, such as, [2] [4] , are generally able to efficiently output good features due to having access to training labels, which contain all information required to classify.

B. Un-supervised Feature Selection (UFS)

However in un-supervised feature selection, training labels are not available, which makes feature selection far more difficult. The most prominent methods are Unsupervised Discriminative Feature Selection (UDFS) [5], which aims to select the most discriminative features for data representation. And Non-negative Discriminate Feature Selection (NFDS) [6], which performs non-negative spectral analysis and feature selection simultaneously. Both of these methods ignore data cleanliness, outliers and noise. Which leads to the Robust Unsupervised Feature Selection (RUFS) [3], which performs robust clustering and robust feature selection simultaneously to determine which features hold the most information. In this problem Principal Feature Analysis (PFA) will be used [7]. It is chosen to be used due to its simple implementation, similar criteria to Principal Component Analysis (PCA) (described in further detail in Section IV), which maximises the variability of the features in the lower dimensional space and minimising the reconstruction error. The PFA method is briefly described below. Let X be a zero mean n-dimensional random feature vector. Let Σ be the covariance matrix of X. Let A be a matrix whose columns are the orthogonal eigenvectors of the matrix Σ

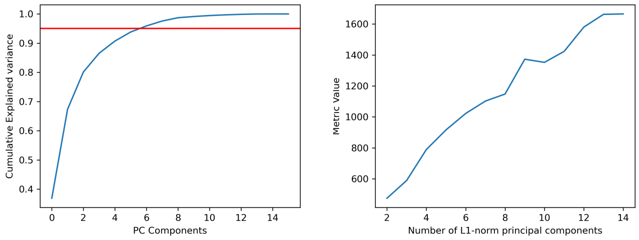
$$\Sigma = A \Lambda A^T$$

$$\text{where} \quad \Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \ddots & 0 & \\ & 0 & \ddots & \\ & & & \lambda_n \end{bmatrix}$$

$\lambda_1, \ldots, \lambda_n$ are the eigenvalues of Σ and $A^T A = I_n$. Let A_q be the first q columns of A and let $V_1, \ldots, V_n$ be the first rows of the matrix A_q. Each vector V_i represents the projection of the i'th feature of the vector X to the lower dimensional space. Creating a PFA algorithm from scratch, also using the standard PCA from sklearn (as the L1-Norm algorithm described above caused compatibility issues) it can be seen that when reducing the dataset to 6 dimensions, the highest influence dimensions are 0, 7, 2, 3, 10, 14 respectively.

IV. Principal Component Analysis

Principal Component Analysis (PCA) is often used during exploratory data analysis and for creating predictive models by leveraging dimensionality reduction by projecting data points into lower principal components. The principal components of a set of points in real space are a sequence of p direction vectors, where the i-th vector is the direction of a line that best fits the data whilst being orthogonal to the first $i-1$ vectors. It is notable that the principal components are eigenvectors of the covariance matrix, and as such can be computed using eigendecomposition. An additional benefit of using PCA is the low computing cost due to using singular-value decomposition (SVD). However in modern big data sets, faulty or corrupted data, often referred to as 'outliers', can cause sensitivity problems. This is due to the L2-PCA places squared emphasis on the magnitude of each data point coordinate, which over-emphasises peripheral points (the outliers that are easier to detect and remove). For this reason, instead of using a standard PCA or L2-PCA, the L1-Norm formulation places linear emphasis on the data point coordinates which is more robust, and invariant to rotations [8]. The L1-Norm is more computationally expensive, with $\mathcal{O}(2^{NK})$ where K is the principal component rank, and $X \in \mathcal{R}$ for a DxN matrix. This can be minimised to $\mathcal{O}(N^{DK-K+1})$ when $d = \text{rank}(X)$ and $K < D$. However, using a novel algorithm utilising bit flipping the L1-Norm calculation can be reduced to something closer to standard L2-Norm PCA calculations $\mathcal{O}\big(ND\text{min}(N,D) + N^2K^2(K^2+d)\big)$ for all $K < d$. This algorithm is demonstrated to be superior to standard L2-Norm PCA for characterising potentially faulty data [9]. The algorithm described in this paper is implemented from scratch using just SVD function from scipy package as this benefits from the optimisation in C.



(a) a

(b) b

Fig. 2. df_standard (a) Explained Variance for increasing pc components, with 95% marked in red (b) L1-Norm metric scores

Figure 2a shows the explained variance curve as a cumulative sum of explained variance ratios for increasing i principal components, for a 90% level 5 components are sufficient and for 95% 6 components are sufficient, 6 PC's will be used for later analysis.

V. Clustering

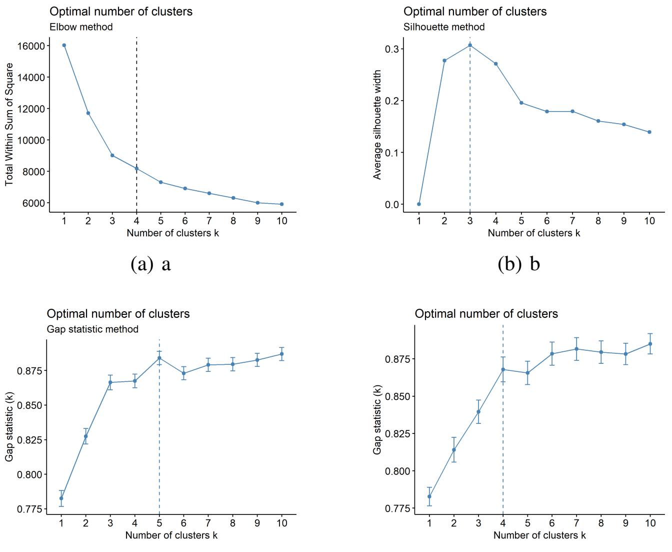
As the dataset contained no ground truth labels, unsupervised clustering must be performed and evaluated.

A. Cluster Analysis

One of the fundamental problems with clustering is determining the optimal, k, number of clusters. There are a number of methods for measuring similarities and estimating the optimal clusters. Generally these fall into two categories:

1) Statistical Methods : Comparing evidence vs null hypothesis
2) Direct Methods : Optimising a criterion

The first method described is the elbow method, which looks at the within-cluster sum of square (WSS), and attempts to select a k value whereby an additional cluster improves little. The second method is the silhouette approach, which computes the average silhouette for different k values and seeks to maximise [10]. The third method which is generally considered the most accurate, the gap statistic compares the



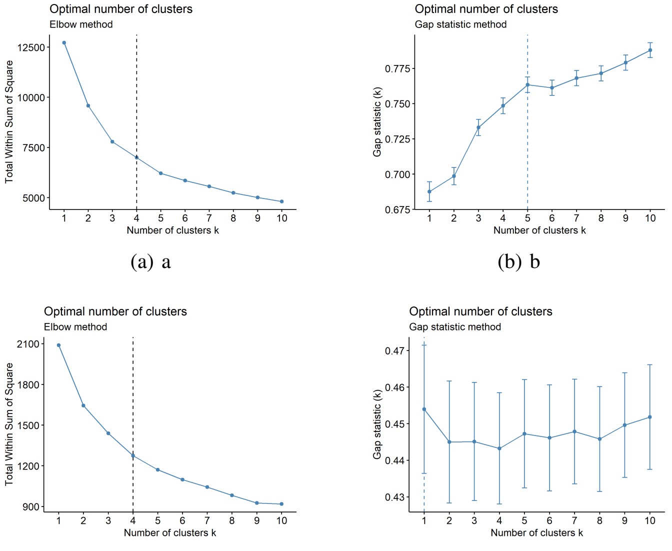
(a) a

(b) b

(c) c

(d) d

Fig. 3. df_standard cluster analysis; (a) demonstrates the elbow method (b) demonstrates the silhouette method (c) demonstrates the gap statistic method with $ns = 25$ and $nboot = 500$ (d) demonstrates the hierarchical clustering method with $kmax = 10$ and $nboot = 500$



(a) a

(b) b

(c) c

(d) d

Fig. 4. (a), (b) Consider df_raining for the elbow and gap statistic method (c),(d) Consider df_snowing for the elbow and gap statistic method. N.B the gap statistic method uses $ns = 25$ and $nboot = 500$

total within intra-cluster variation for different values of k with their expected values under null reference distribution of the data. The estimate of the optimal clusters will be the value that maximises the gap statistic. [11]. As described in Section III the data is split into 3 datasets, labelled df_raining, df_snowing and df_standard.

Fig 3 demonstrates various algorithms for the standard dataset. The elbow method is suggesting 4 clusters is appropriate, the silhouette and hierarchical method is suggesting 3 where as the gap statistic is suggesting 5. All 3 of these values will be tested for accuracy. Fig 4 demonstrates the elbow and gap statistic methods for the raining and snowing subsets, the gap statistic for the snow subset suggests using 1 cluster is sufficient, and for the raining subset 5 is recommended. The elbow method suggests 4 for both, which is an expected result due to how its calculated. Additionally the silhouette method suggests 2 for both snowing and raining subsets. Hence 5 clusters will be used for the raining subset, and 1 cluster will be used for the snowing subset and will be given the label ”snowing” and no longer considered for clustering.

B. Cluster Evaluation

Evaluating the performance of a clustering algorithm can be complicated, especially without ground truth labels, and as such the only comparisons that can be made are using the model itself. This paper considers 3 scoring methods; Silhoette Coefficient (CS) , Calinksi-Harabasz Index (CHI) and Davies Bouldin Index (DBI). The silhouette coefficient s for a single sample is given as

$$s = \frac{b-a}{\max(a,b)}$$

where a is the mean distance between a sample and all other points in the same class, and b is the mean distance between a sample and all other points in the nearest cluster [12]. The score is bounded between $[-1, 1]$ for incorrect clusters to high dense clusters, with higher scores when clusters are dense and well separated. However sometimes it can misperform when using density based clusters (such as DBSCAN). The Calinkski-Harabasz index (Variance Ratio Criterion), score s for dataset E of size n_E which has been clustered into k clusters.

$$s = \frac{\mathrm{tr}(B_k)}{\mathrm{tr}(W_k)} \cdot \frac{n_E - k}{k-1}$$

where $\mathrm{tr}(B_k)$ is thr trace between the group dispersion matrix and $\mathrm{tr}(W_k)$ is the trace of the within-cluster dispersion matrix defined by

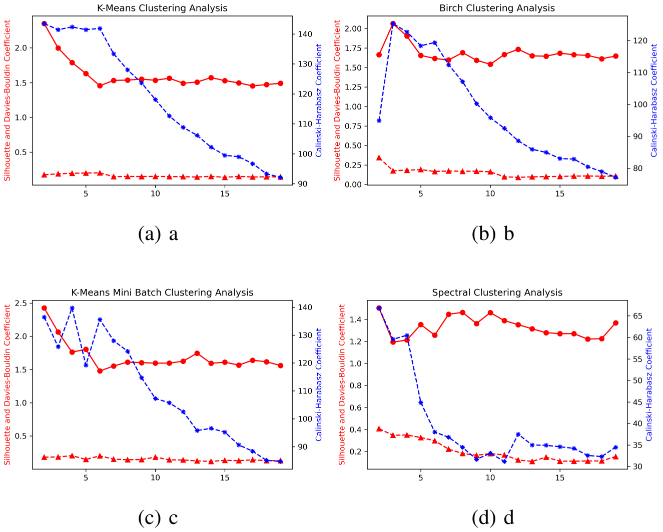
$$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T$$

and

$$B_k = \sum_{q=1}^{k} n_q (c_q - c_E)(c_q - c_E)^T$$

where C_q is the set of points in cluster q, c_q is the centre of cluster q, c_E is the center of E, and n_q is the number of points in q. A high Calinkski-Harabasz score generally relates to dense and well seperated clusters and is fast to compute, however it suffers the same fate as the silhouette score and can misperform with convex clusters such as density based clustering [13]. The final evaluation metric is the Davies-Bouldin Index s for

$$s = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{ij}$$



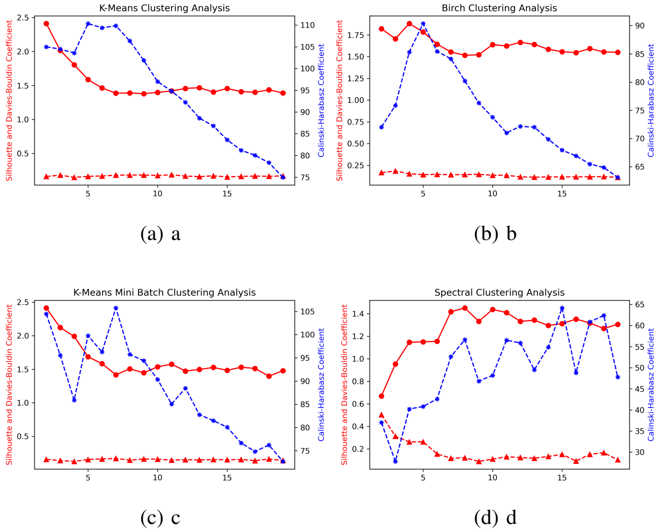
(a) a

(b) b

(c) c

(d) d

Fig. 5. df_standard ; Silhouette Score (red dashed with triangles), Davies-Bouldin (red with circles) and Calinski-Harabasz (blue dashed with stars)(a) K-Means (b) Birch (c) K-Means Mini Batch $k = 200$ (d) Spectral Analysis



(a) a

(b) b

(c) c

(d) d

Fig. 6. df_raining ; Silhouette Score (red dashed with triangles), Davies-Bouldin (red with circles) and Calinski-Harabasz (blue dashed with stars)(a) K-Means (b) Birch (c) K-Means Mini Batch $k = 200$ (d) Spectral Analysis

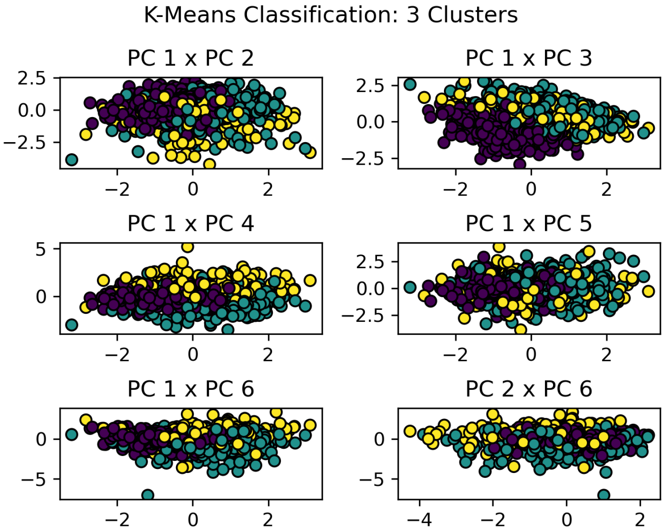


Fig. 7. df_standard : K-Means Clustering algorithm with $k = 3$ clusters, plotting PC 1 vs PC n

where

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

. s_i is the average distance between each point of cluster i and the centroid as of that cluster, d_{ij} is the distance between cluster centroids i and j. A condition implied is that R_{ij} is non-negative and symmetric. The Davies-Bouldin index is the average similarity between clusters, and 0 is the lowest possible score which relates to well defined clusters and it is fast to compute but it suffers the same fate as the previous two with convex clusters and misperforms. Another large drawback for this method is the limitation to euclidean space due to the usage of centroid distances [14] [15].

This paper will compare 4 different clustering methods:

1) K-Means
2) Birch
3) Mini Batch K-Means
4) Spectral Analysis

Figures 5 and 6 demonstrate the performance of the clustering algorithms using the evaluation methods defined above for a varying number of clusters. Figure 5 shows the metric scores for the standard dataset, with a clear downward trend for all methods as n increases. K-Means Mini Batch has irregular scores for lower cluster numbers, which is expected due to the low number of samples. These metrics reinforce the clustering strategy outlines from the previous analysis for the standard dataset. Figure 6 shows the metric scores for the raining dataset, K-Means and Birch follow a similar curve however the CHI score are $y - 20$. Spectral performs noticeably poorly for both the standard and raining dataset

The maxi-min average scores across the metrics for the standard dataset suggested K-Means with 3 clusters performed the strongest, and so will be used to predict labels. Graphically it can be seen in Figure 7, with the first PC plotted against the others and each label coloured in its respective label, it does not look accuratley labelled but this is due to difficulties plotting in dimensions beyond 3 and this data is 6 dimensional. The raining subset suggested Birch with 5 clusters performed the strongest, and this can be seen in Figure 8

These two clustering methods are used to predict labels based upon the data X, and then this is saved and assigned to the original dataset. As noted previously, the snowing subset are all given the label 8. Clusters are not given names as it would increase interpretability for classification algorithms however cluster names are presented in Table I

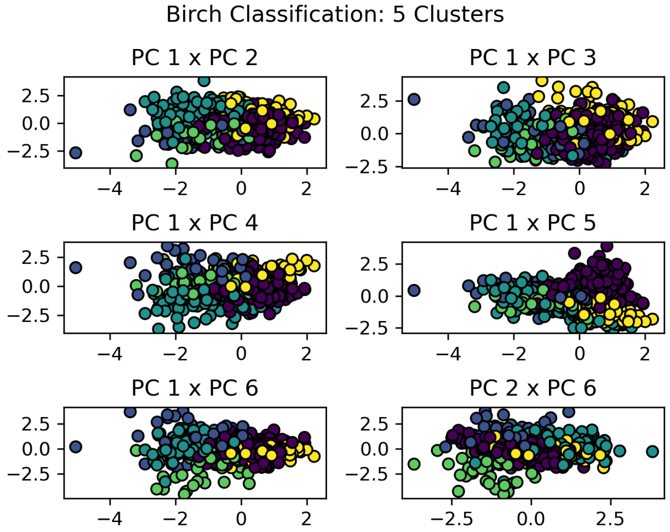


Fig. 8. df_raining : Birch Clustering algorithm with $k = 5$ clusters, plotting PC 1 vs PC n

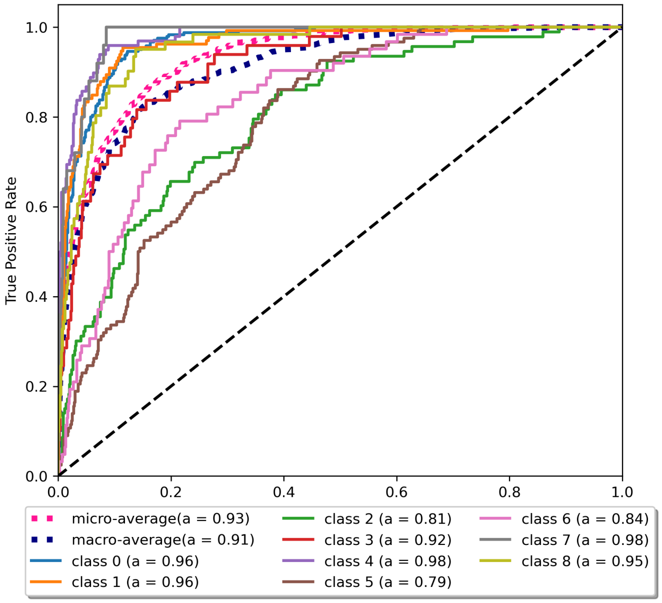


Fig. 9. ROC Plot for Support Vector Machine classification with micro-average and macro-average

VI. Classification

Once the dataset has been clustered, and labels have been applied a classifying model can be created to predict which label future values should get. Starting by splitting the dataset into testing and training data to ensure testing data is unseen, for most tests a 95/5 split will be used unless noted otherwise.

A. Support Vector Machines

The first method that will be performed is Support Vector Classification (SVC) which constructs a hyper-plane (or set of hyper-planes) in an infinte dimensional space. A good seperation is achieved by the hyper-plane which has the largest distance to the nearest training data points of any class and generally a large margin correlates to lower classifier errors [16]. Support Vector Machines (SVM), which SVC is a type of, use Sequential Minimal Optimisation (SMO) to solve very large Quadratic Programming (QP) optimisation problems. SMO breaks a large QP problem into a series of smaller QP problems qhich can be solved analytically, instead of computationally-heavier numerical solutions. The primal optimisation problem is to minimise:

$$r(\mathbf{w}, \mathbf{e}) = \frac{1}{2}||\mathbf{w}||^2 + \frac{c}{\ell}\sum_i e_i$$

subject to

$$y_i \cdot ((\mathbf{x}_i \cdot \mathbf{w}) + b) \geq 1 - e_i, e_i \geq 0.$$

Equation VI-A is a re-parameterisation of the Support Vector Regression (SVR) algorithm [17], [18]. A simple method to evaluate the performance of a classifier is by using a Receiver Operating Characteristic (ROC) curve, which illustrates the diagnostic performance of a binary classifier. The curve is created by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. (These are also known as precision and recall). To use this is a multi-class setting, a series of 1 v many operations are made, this is shown in Figure 9. This is a computationally costly due to ROC space having $c(c-1)$ dimensions for c classes. The labels include the area under the curve (AUC), which when normalised, the AUC is equal to the probability that a classifier will rank a positive instance higher than a negative instance, this statistic was previously used often to compare models [19]. The AUC values for this SVC model are relatively strong, especially for labels 0, 1, 4 and 7. However performs weakly for labels 2, 5 and 6.

TABLE I
Cluster Labels

label	Description
0	low sun
1	medium sun
2	high sun
3	low rain
4	medium rain + low wind
5	medium rain + medium wind
6	medium rain + high wind
7	high rain
8	snowing

B. AutoML

AutoML provides methods and processes such as Hyper-parameter Optimisation (HPO), Neural Architecture Search (NAS) and Meta-Learning, and makes them available to non-experts [20]. This is an initative between the ML Freiburg and ML Hannover. The main interest of this study is HPO, mainly the provided packages Auto-Sklearn (ASKL) and SMAC (Sequential Model-Based Algorithm Configuration). Auto-Sklearn provides supervised machine learning hyperparameter optimisation and bayesian optimisation based upon the scikit learn machine learning library, it does this by searching for an appropriate learning algorithm and optimises its hyperparameters [21] [22]. This leverages the seperate inititative, SMAC, which is a tool for optimising algorithm parameters which at first was designed to speed up local and tree search algorithm (very successfully [23]). Additionally it was recently found to be very effective for optimising hyperparameters of machine learning algorithms, scaling better to high dimensions than other algorithms [24] [25] [26] [27]. Auto-Sklearn leverages python libraries specific to unix environments, so the following experiments are executed on WSL2 (Windows Subsystem for Linux). The first experiment with ASKL extends the provided API to implement a sklearn library function, Multi-layer Perceptron (MLP) Classifier. The script created adds some HPO but is largely standard code. Using the standard sklearn testtrain splitting method, the labelled dataset is split and trained using variant hyperparameters for MLP classification. A time limit of 60 seconds returns 14 ensembles giving an overall accuracy of 90.29% and the best ensemble a metric of 0.4000. The peak validation score is 0.876430 with 23 attempted runs, 22 successful runs with 1 failure due to exceeding memory allocation. The full run details can be found in the additional file. The second experiment implements common classifying algorithms including Bernouilli, Multinomial and Gaussian Naive Bayes, standard decision trees, extra trees, adaboost, gradient boosting, k-NN, LDA, liblinear SVC, passive aggressive and SGD. Additionally implemented are samplers, scalers, imputers (14 feature processing methods, and 3 data preprocessing methods, giving rise to a structured hypothesis space with 100+ hyperparameters). Starting by splitting the labelled data into test and train using 6733 split then trained on each of the models individually and scoring them based on micro versions of f1-score and recall. The best ensemble resulted in an accuracy of 87% with a precision value of 87%, recall 87% and F1 0.87. Additionally, the resulting confusion matrix is shown below.

$$C = \begin{bmatrix} 158 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 87 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 5 & 49 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 33 & 6 & 1 & 3 & 6 & 0 \\ 0 & 0 & 0 & 1 & 57 & 3 & 4 & 3 & 0 \\ 0 & 0 & 0 & 1 & 1 & 15 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 10 & 0 & 53 & 1 & 0 \\ 0 & 0 & 0 & 1 & 9 & 0 & 3 & 19 & 7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 37 \end{bmatrix}$$

Furthermore, it returns 1636 baseline positives from 582 overall, giving a 281.1% baseline rate. This baseline rate suggests there is far more analysis to be performed in this area to further optimise the results, especially including more computation time and a higher memory allocation.

VII. Conclusion

This paper investigated the viability of using PFA to analyse the dimensions that contain the most information, which lead to little success. Where as the custom L1-Norm algorithm, to prevent outliers being over-weighted, performed far stronger so they were used in conjunction, however the PFA is not required. Splitting the dataset required significant work logistically as the 3 subsets differed significantly, perhaps this maintains a good reason to split again however the clustering algorithms potentially could perform as strong with the single dataset. The performance of various clustering algorithms were evaluated objectively on this dataset using metrics such as the David-Bouldin Index. Finally, when classifying the SVC algorithm performed strongly giving strong results, as in the ROC 9. But the analysis from AutoML / Auto-Sklearn provided far better insight and provides much investigation for the future.

VIII. Acknowledgements

This paper heavily relies on the sklearn library [28] [29] and the work of AutoML to provide Auto-Sklearn as well as github user Motorrat for the base of the second AutoML experiment.

References

[1] H. Liu, X. Wu, and S. Zhang, "Feature selection using hierarchical feature clustering," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, (New York, NY, USA), p. 979–984, Association for Computing Machinery, 2011.

[2] T. Phyu and N. Oo, "Performance comparison of feature selection methods," *MATEC Web of Conferences*, vol. 42, p. 06002, 01 2016.

[3] M. Qian and C. Zhai, "Robust unsupervised feature selection," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, p. 1621–1627, AAAI Press, 2013.

[4] S. Zheng and C. Ding, "A group lasso based sparse knn classifier," *Pattern Recognition Letters*, vol. 131, pp. 227 – 233, 2020.

[5] Y. Yang, H. Shen, Z. Ma, Z. Huang, and X. Zhou, "l2, 1-norm regularized discriminative feature selection for unsupervised learning," in *IJCAI*, 2011.

[6] Z. Li, Y. Yang, J. Liu, X. Zhou, and H. Lu, "Unsupervised feature selection using nonnegative spectral analysis," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, p. 1026–1032, AAAI Press, 2012.

[7] Y. Lu, I. Cohen, X. S. Zhou, and Q. Tian, "Feature selection using principal feature analysis," in *Proceedings of the 15th ACM International Conference on Multimedia*, MM '07, (New York, NY, USA), p. 301–304, Association for Computing Machinery, 2007.

[8] N. Kwak, "Principal component analysis based on l1-norm maximization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 9, pp. 1672–1680, 2008.

[9] P. P. Markopoulos, S. Kundu, S. Chamadia, and D. A. Pados, "Efficient l1-norm principal-component analysis via bit flipping," *IEEE Transactions on Signal Processing*, vol. 65, no. 16, pp. 4252–4264, 2017.

[10] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Statistics, Wiley, 2009.

[11] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[12] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53 – 65, 1987.

[13] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics*, vol. 3, no. 1, pp. 1–27, 1974.

[14] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[15] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of Intelligent Information Systems*, vol. 17, pp. 107–145, Dec 2001.

[16] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization, advances in kernel methods," 1999.

[17] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Computation*, vol. 12, no. 5, pp. 1207–1245, 2000.

[18] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, Sep 1995.

[19] J. A. Hanley and B. J. McNeil, "A method of comparing the areas under receiver operating characteristic curves derived from the same cases.," *Radiology*, vol. 148, no. 3, pp. 839–843, 1983.

[20] F. Hutter, L. Kotthoff, and J. Vanschoren, eds., *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2018. In press, available at http://automl.org/book.

[21] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 2962–2970, Curran Associates, Inc., 2015.

[22] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter, "Auto-sklearn 2.0," *arXiv:2006.???? [cs.LG]*, 2020.

[23] F. Hutter, H. H. Hoos, K. Leyton-Brown, and K. P. Murphy, "Time-bounded sequential parameter optimization," in *Proceedings of the conference on Learning and Intelligent OptimizatioN (LION 4)*, Jan. 2010.

[24] F. Hutter, H. Hoos, and K. Leyton-Brown, "An evaluation of sequential model-based optimization for expensive blackbox functions," in *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '13 Companion, (New York, NY, USA), p. 1209–1216, Association for Computing Machinery, 2013.

[25] A. Zela, A. Klein, S. Falkner, and F. Hutter, "Towards automated deep learning: Efficient joint neural architecture and hyperparameter search," in *ICML 2018 AutoML Workshop*, July 2018.

[26] S. Falkner, M. Lindauer, and F. Hutter, "Spysmac: Automated configuration and performance analysis of sat solvers," in *Proceedings of the International Conference on Satisfiability Solving (SAT'15)*, pp. 1–8, Aug. 2015. To appear.

[27] M. Feurer and F. Hutter, "Towards further automation in automl," in *ICML 2018 AutoML Workshop*, July 2018.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[29] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.