IAML – INFR10069 (LEVEL 10): Assignment #2

s1712653

PART A: 20-NEWSGROUPS [60 POINTS]

Question 1 : (10 points) Exploratory Analysis

We will begin by exploring the Dataset to get some insight about it.

[1.1] (5 points) Focusing first on the training set, summarise the key features/observations in the data: focus on the dimensionality, data ranges, feature and class distribution and report anything out of the ordinary. What are the typical values of the features like?

The training set consists of 5648 observations, each with a dimensionality of 1000. Each attribute of an observation corresponds to a word, and its value is the TF-IDF weight of that word, which ranges from 0 to 1 in this particular data set. The average values of each attribute range from 0.0001 to 0.0252. The most common attribute value is 0 as not all words appear in a document at once. There is an uneven class distribution as class 7 is almost half as frequent as all of the other classes, this will impact our model classification accuracy as words that belong to class 7 might be misclassified. The distribution of the mean value of each attribute resembles a positively skewed normal distribution with mean roughly around 0.0025.

Question 1 continued on next page. . .

[1.2] (3 points) Looking now at the Testing set, how does it compare with the Training Set (in terms of sizes and feature-distributions) and what could be the repurcussions of this?

The testing set consists of 1883 observations with a similar attribute value range to the training set. The class distribution also closely resembles the training set where class 1 to 6 have similar frequencies while class 7 is almost half as frequent as the other classes. The distribution of the mean value of each attribute also resembles the distribution observed on the training set, this supports the assumption for machine learning where the training and testing sets should be sampled from the same distribution, thus lowering the amount of bias from the testing set.

Question 1 continued on next page. . .

[1.3] (2 points) Why do you think it is useful to consider TF-IDF weights as opposed to just the frequency of times a word appears in a document as a feature?

The desired model should take into account how relevant each word is to each document, that is, the words which would most help a human better understand a document without reading it all. Relevant words do not necessarily mean the most frequent words, for example, stopwords which are frequent while not offering much meaning. Thus it is important to not only consider the document but the dataset as a whole, as we want to distinguish words that are meaningful to a particular document and not words that appear the most, which TF-IDF tries to measure.

Question 2 : (24 points) Unsupervised Learning

We will now explore the documents in some detail by way of clustering.

[2.1] (2 points) The K-Means algorithm is non-deterministic. Explain why this is, and how the final model is selected in the SKLearn implementation of KMeans.

The non-deterministic nature of the K-Means algorithm comes from its random selection of starting centroids, this means that depending on the choice of the starting centroids the algorithm could give rise to entirely different models of the data. The SKLearn implementation of K-Means selects the final model based on which one minimizes Inertia the most.

Question 2 continued on next page…

[2.2] (1 point) One of the parameters we need to specify when using k-means is the number of clusters. What is a reasonable number for this problem and why?

The observations in the data set each belong to 1 of 8 different classes. This could imply that the observations from each class have similar values in the feature space, thus a reasonable number for this data set could be the number of classes: 8.

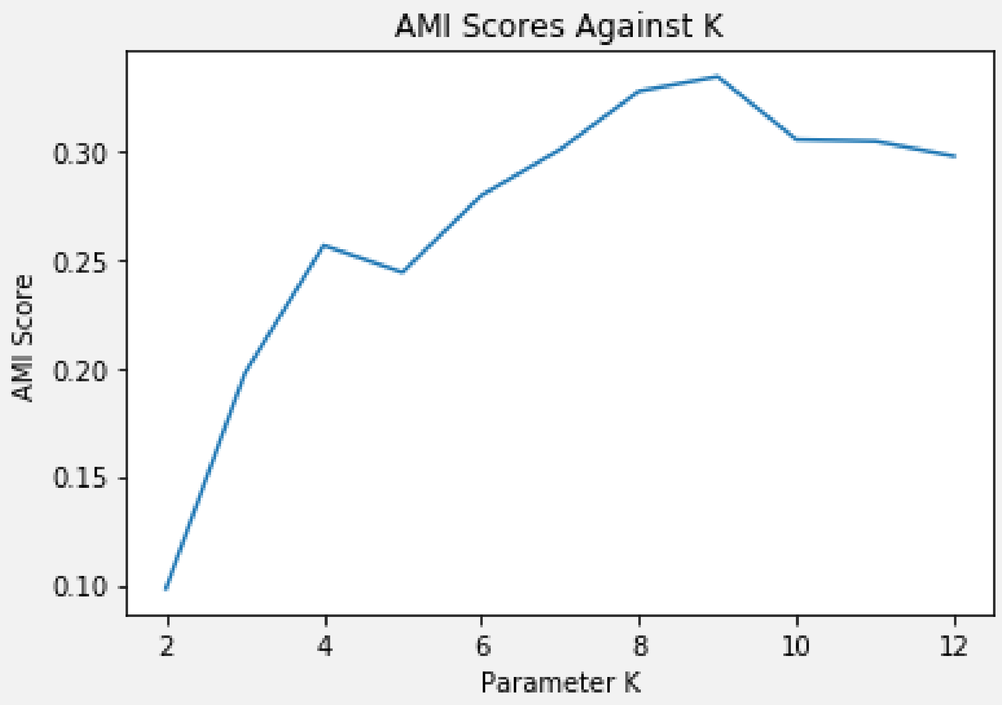
Question 2 continued on next page...

[2.3] (5 points) We will use the Adjusted Mutual Information (AMI) i.e. `adjusted_mutual_info_score` between the clusters and the true (known) labels to quantify the performance of the clustering. Give an expression for the MI in terms of entropy. In short, describe what the MI measures about two variables, why this is applicable here and why it might be difficult to use in practice. *Hint: MI is sometimes referred to as Information Gain: note that you are asked only about the standard way we defined MI and not the AMI which is adjusted for the size of the domain and for chance agreement.*

Mutual information, MI, of known variables U and V can be calculated as $\mathrm{MI}(U, V) = \mathrm{H}(U) - \mathrm{H}(U|V)$, where $\mathrm{H}(X)$ is the entropy of variable X. Mutual information measures how much one variable tells us about the other, thus it is the reduction in uncertainty about variable X after having observed variable Y. Mutual Information is applicable to this problem as we are trying to measure how much information is shared, and therefore the accuracy, between predicted labels and the ground truth of the data regardless of the ordering of the clusters produced by the algorithm. In practice it is difficult to use as it requires for the true labels of the data to be known, which might not be available in practice or requires them to be manually assigned.

Question 2 continued on next page. . .

[2.4] (4 points) Fit K-Means objects with `n_clusters` ranging from 2 to 12. Set the random seed to 1000 and the number of initialisations to 50, but leave all other values at default. For each fit compute the adjusted mutual information (there is an SKLearn function for that). Set `average_method='max'`. Plot the AMI scores against the number of clusters (as a line plot).



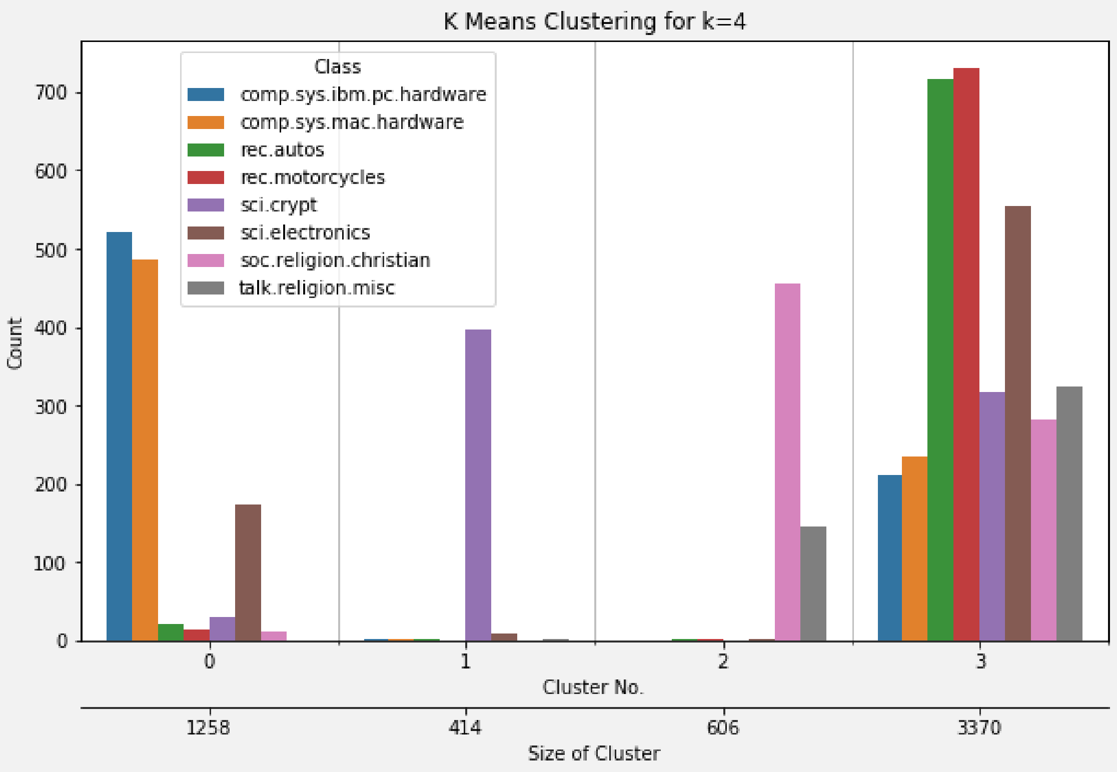
Question 2 continued on next page…

[2.5] (3 points) Discuss any trends and interesting aspects which emerge from the plot. Does this follow from your expectations?

The AMI seems to grow larger as K increases up to a point. Contrary to what is expected the highest AMI is not observed when $K = 8$ (the number of classes), but instead from $K = 9$, which could indicate that the data set is best modeled by 9 different classes. The AMI starts trending down for $K > 9$. There also seems to be an unexpected decrease in AMI from $K = 4$ to $K = 5$.

Question 2 continued on next page...

[2.6] (6 points) Let us investigate the case with four (4) clusters in some more detail. Using seaborn's `countplot` function, plot a bar-chart of the number of data-points with a particular class (encoded by colour) assigned to each cluster centre (encoded by position on the plot's x-axis). As part of the cluster labels, include the total number of data-points assigned to that cluster.



[2.7] (3 points) How does the clustering in Question2:(f) align with the true class labels? Does it conform to your observations in Q 2(e)?

The clustering for $K = 4$ is not very accurate as all observations belonging to a particular true class label are not all in 1 cluster. Most observations belonging to a class seem to be mostly spread out over two different clusters with the exception of *rec.motorcycles* and *rec.autos* which are mostly assigned to a single cluster. The cluster sizes are also very uneven with the smallest clusters only containing 1 or 2 classes on the majority, while the largest cluster has almost 6 times the samples as the smallest cluster and also contains significant amounts of samples from all classes. The result is mostly expected as the AMI score for $K = 4$ indicates that the data is not well modelled by 4 clusters as observed in the previous questions.

Question 3 : (26 points) Logistic Regression Classification

We will now try out supervised classification on this data. We will focus on Logistic Regression and measure performance in terms of the F1 score (familiarise yourself with this score which is related to the precision and recall scores that we learnt about in class).

[3.1] (3 points) What is the F1-score, and why is it preferable to accuracy in our problem? How does the macro-average work to extend the score to multi-class classification?

The F1 Score is the weighted average of the precision and recall of a model. It takes into account both false positives and false negatives, as well as taking into account class imbalance, which is the case for this data set, thus it is prefered over accuracy. The macro-average extends the score to handle multiple classes by calculating the metrics for each label and then finding their unweighted mean as per the SKlearn documentation for the function f1_score.

Question 3 continued on next page...

[3.2] (2 points) As always we start with a simple baseline classifier. Define such a classifier (indicating why you chose it) and report its performance on the **Test** set. Use the 'macro' average for the `f1_score`.

The baseline classifier used will be one that classifies all test samples as the class that occured most often in the training set, *i.e* the class with the highest prior probability. This is a useful baseline as it allows us to compare it to more complex models. The F1 Score of this model on the test set is: 0.029.

Question 3 continued on next page…

[3.3] (3 points) We will now train a LogisticRegression Classifier from SKLearn. By referring to the documentation, explain how the Logistic Regression model can be applied to classify multi-class labels as in our case. *Hint: Limit your explanation to methods we discussed in the lectures.*

To extend Logistic Regression for multi-class problems, first a weight vector w_k is created for each class, then the softmax function is used to calculate the probability that a feature vector belongs to a class, $P(y = k|x)$, thus assigning each feature vector to a class. The model is then optimized to minimize the log loss, in this case the L-BFGS algorithm will be used as the optimization method.

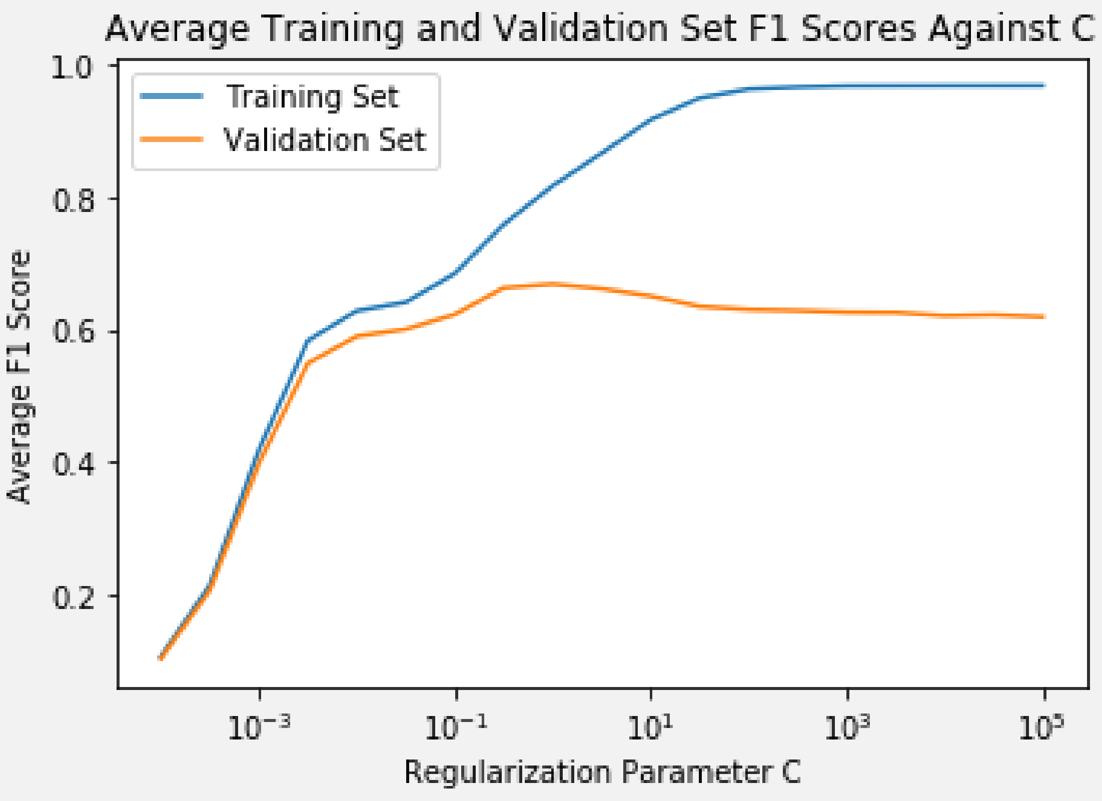
Question 3 continued on next page...

[3.4] (4 points) Train a Logistic Regressor on the training data. Set `solver='lbfgs'`, `multi_class='multinomial'` and `random_state=0`. Use the Cross-Validation object you created and report the average validation-set F1-score as well as the standard deviation. Comment on the result.

The average of the F1 Scores calculated from the 10 folds of the training set is: 0.669, while the standard deviation of the F1 Scores is: 0.017. The logistic regression model trained on each fold has a significantly higher average F1 Score than the baseline classifier, while having a decently high average F1 Score itself. The F1 Scores calculated also have a low standard deviation indicating that the trained model is precise.

Question 3 continued on next page...

[3.5] (5 points) We will now optimise the Regularisation parameter C using cross-validation. Train a logistic regressor for different values of C: in each case, evaluate the F1 score on the training and validation portion of the fold. That is, for each value of C you must provide the training set and validation-set scores per fold and then compute (and store) the average of both over all folds. Finally plot the (average) training and validation-set scores as a function of C. *Hint: Use a logarithmic scale for C, spanning 19 samples between 10^{-4} to 10^{5}.*



Question 3 continued on next page. . .

[3.6] (7 points) What is the optimal value of C (and the corresponding score)? How did you choose this value? By making reference to the effect of the regularisation parameter C on the optimisation, explain what is happening in your plot from Question 3:(e) *Hint: Refer to the documentation for C in the LogisticRegression page on SKLearn.*

The optimal value of C is 1. This value was chosen as it gives rise to the highest average F1 Score on the validation set. As per the SKlearn documentation, C is the inverse regularization parameter defined as $C = \frac{1}{\lambda}$, where λ is the lambda regulator. The effect of changing the value of C can be seen in the plot above, where regularizing too little causes the convergence parameters to not be acertained and therefore leads to a low average F1 Score for the training and validation set, while regularizing too much leads to overfitting of the training data, leading to a higher average F1 Score for the training set while lowering the F1 Score for new data.

Question 3 continued on next page. . .

[3.7] (2 points) Finally, report the score of the best model on the test-set, after retraining on the entire training set (that is drop the folds). *Hint: You may need to set `max_iter = 200`.* Comment briefly on the result.

The F1 Score for the best model, logistic regression classifier with regularization parameter $C = 1$, on the test set is: 0.675. This result is significantly larger compared to the baseline classifier, and is a decently large F1 Score itself, which indicates that it is classifying the test set well.

PART B: BRISTOL AIR-QUALITY [90 POINTS]

Question 4 : (30 Points) Exploratory Analysis

We will begin by exploring the Dataset to familiarise ourselves with it.

[4.1] (6 points) Summarise the key features/observations in the data: describe the purpose of each column and report (briefly) also on the dimensionality/ranges (ballpark figures only, and how they compare across features) and number of sites, and identify anything out of the ordinary/problematic: i.e. look out for missing data and negative values. Why are the latter unreasonable in such a dataset? *Hint: Refer to the documentation for how to interpret the pollutant values.*

The data set contains 1306757 observations, each corresponting to a reading of 3 pollutants (NO_x, NO_2, NO), with values ranging from -31.08 to 2164.25 across all pollutants, at one of 18 sites, labeled from 0 to 17, along with the latitude, ranging from 51.43 to 51.49 degrees, and logitude, ranging from -2.69 to -2.54 degrees, of that site and the date, ranging from the year 2007 to 2019, and time, ranging from 0AM to 12PM in 1 hour intervals, the reading was made. The data has a dimension of 7, however it contains observations with missing values, depicted as NaN, as well as negative values for the pollutant readings. Having pollutant reading with negative values in this dataset is unreasonable as there is no real world interpretation of a negative reading of a pollutant and therefore doesn't have any meaning.

Question 4 continued on next page. . .

[4.2] (6 points) Repeat the same analysis but this time on a per-site basis. Provide a table with the number of samples and percentage of problematic samples (negative and missing) in each site. To report numbers, count a row which has at least one missing entry as having missing data, and similarly for negative entries. *Hint: Pandas has a handy method, `to_latex()`, for generating a latex table from a dataframe.*

SiteID	0	1	2	3	4	5
Count	6446	163111	62990	25464	74787	113952
% Missing	1.613404	6.290195	4.348309	77.332705	2.068541	8.828279
% Negative	0	0	0.004763	0.777568	0.005349	0

SiteID	6	7	8	9	10	11
Count	142141	115162	43824	22071	96407	20693
% Missing	7.444017	4.194960	21.056955	5.301074	3.589988	1.904026
% Negative	0.002814	0.277869	0	0	0.004149	0.086986

SiteID	12	13	14	15	16	17
Count	45240	12423	113951	2712	154331	91053
% Missing	17.484527	51.461000	10.53172	100.0	6.530768	6.271073
% Negative	0	0.016099	0	0	0.013607	0.002197

Question 4 continued on next page...

[4.3] (4 points) Briefly summarise how the sites compare in terms of number of samples and amount of problematic samples.

The number of samples belonging to each of the 18 sites varies greatly with the smallest number of samples in site 15, with 2712 samples, and the largest number of samples in site 1, with 163111 samples. All sites contain problematic samples, making up less than 10% of the majority of all sites, however sites 3, 8, 13 and 15 contain significantly more problematic samples, with site 15 being made up of only problematic samples. Out of the 2 types of problematic samples, missing samples occur significantly more than negative samples.

Question 4 continued on next page...

[4.4] (3 points) Given that the columns are all oxides of nitrogen and hence we expect them to be related, we will now look at correlations in our data. This will also be useful in determining how well we can predict any one of the readings from the other two. Remove the data from sites 3 and 15 and compute the **Pearson** correlation coefficient between each of the three pollutant columns on the remaining data. Visualise the coefficients between each pair of columns in a table.

	NOx	NO2	NO
NOx	1.000000	0.878016	0.988019
NO2	0.878016	1.000000	0.807853
NO	0.988019	0.807853	1.000000

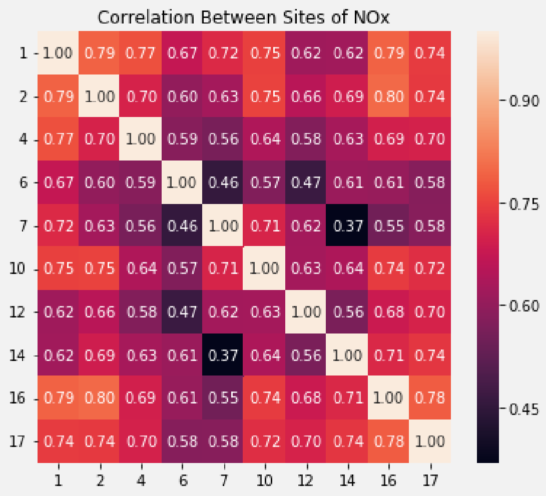
Question 4 continued on next page. . .

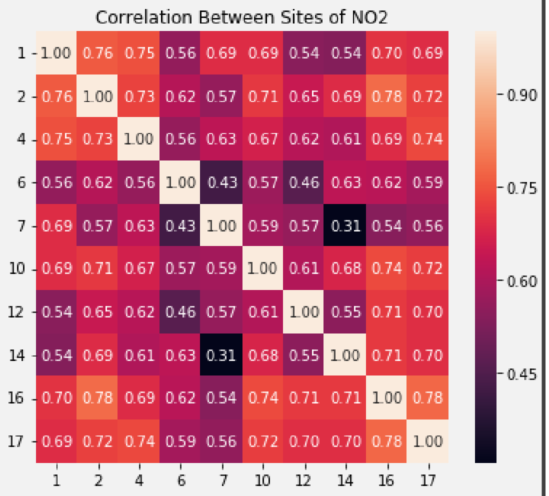
[4.5] (2 points) Comment on the level of correlation between each pair of pollutants.

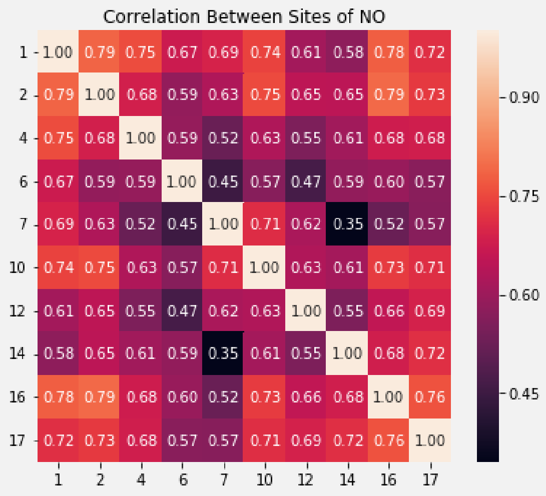
From the data set, each pollutant seems to have a significant positive correlation with every other pollutant. Out of all the pairs NO_x and NO exhibit the strongest positive correlation, while NO_x and NO_2 exhibit the least strongest positive correlation. This could indicate that all 3 pollutants have a positive correlation with each other.

Question 4 continued on next page...

[4.6] (5 points) For each of the three pollutants, compute the Pearson correlation between sites. *Hint: You will need to remove the 'Date Time' column and then group by the first level of the columns.* Then plot these as three heatmaps: show the values within the figures. *Hint: Use the method* `plot_matrix()` *from* `mpctools.extensions.mplext`.







Question 4 continued on next page...

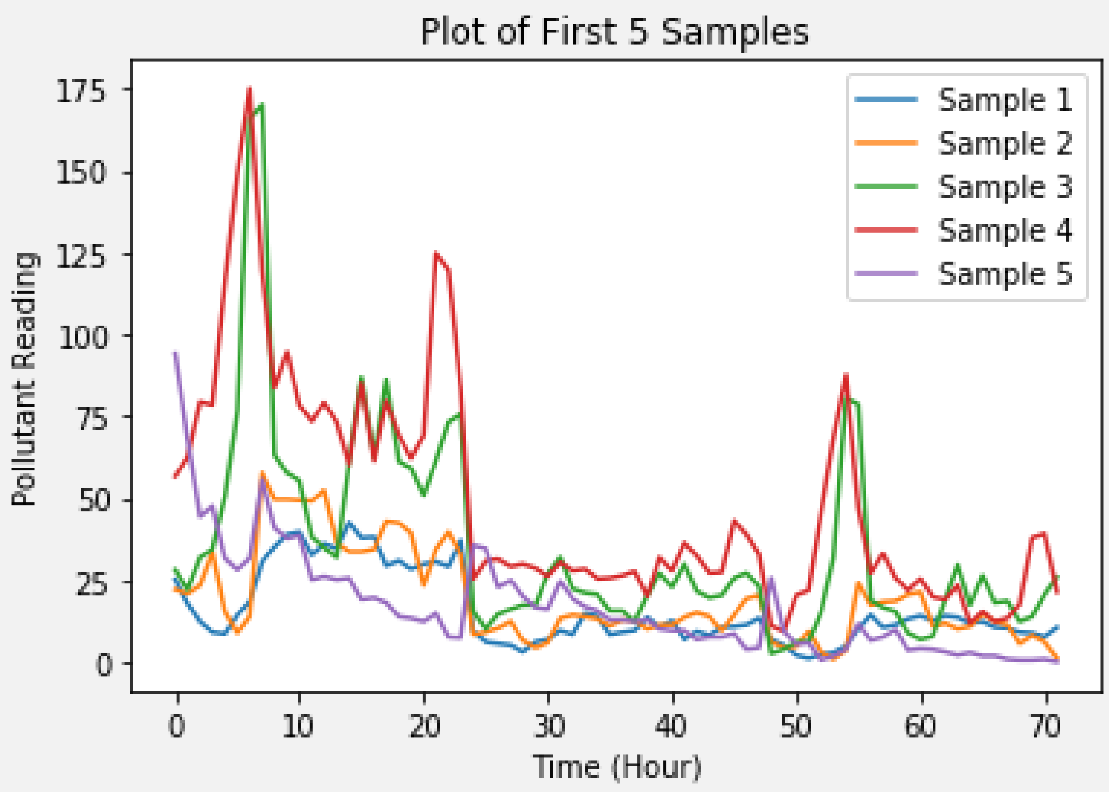
[4.7] (4 points) Comment briefly on your observations from Question 4:(f): start by summarising the results from the NO gas and then comment on whether the same is observed in the other gases or if there is something different.

As seen from the heatmap plot for NO from the last question the pearson correlation coefficient between each pair of sites are all greater than 0, meaning that they all exhibit positive correlation. Some sites, however, show a much weaker positive correlation when paired with all the other sites, particularly sites 6, 7, 12 and 14, with the smallest correlation coefficient arising from the pair 14 and 7. The other 2 heatmaps show similar patterns, however in the heatmap for NO_2, the correlation between sites 10 and 7 is much lower compared to the other two plots.

Question 5 : (19 Points) Principal Component Analysis

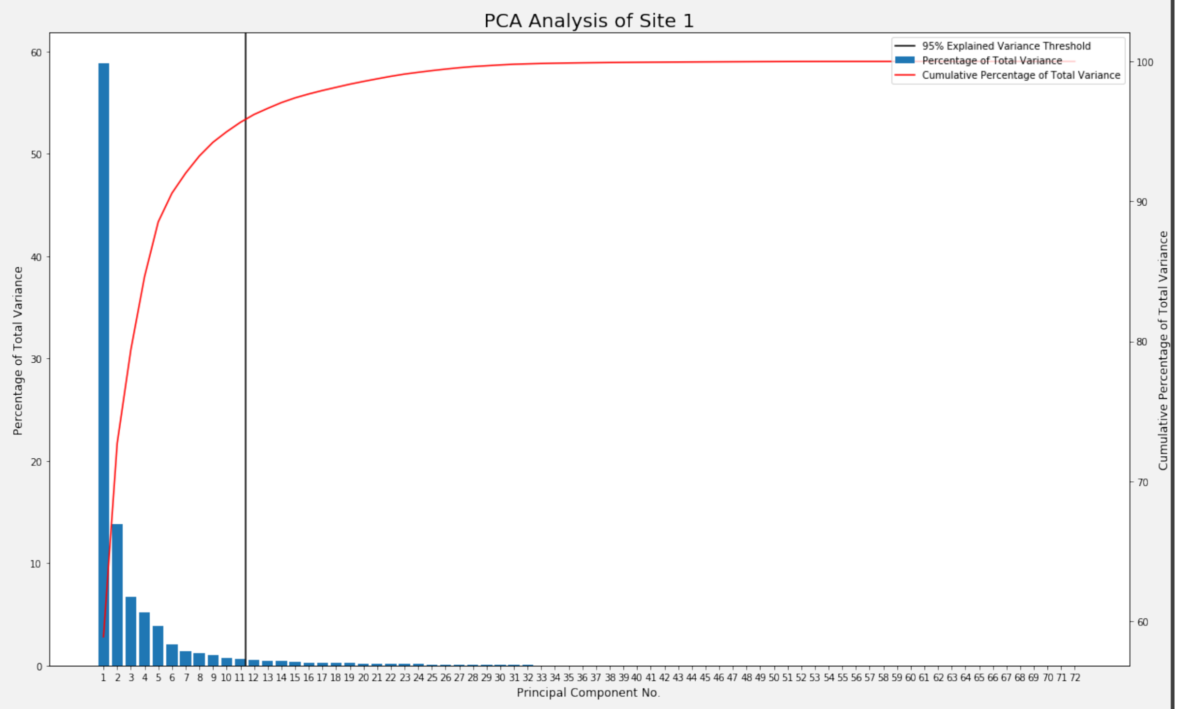
One aspect which we have not yet explored is the temporal nature of the data. That is, we need to keep in mind that the readings have a temporal aspect to them which can provide some interesting insight. We will explore this next.

[5.1] (1 point) Plot the first 5 lines of data (plot each row as a single line-plot).



Question 5 continued on next page. . .

[5.2] (5 points) We will focus first on data solely from Site 1. Extract the data from this site, and run PCA with the number of components set to 72 for now. Set the `random_state=0`. On a single graph plot: (i) the percentage of the variance explained by each principal component (as a bar-chart), (ii) the cumulative variance (line-plot) explained by the first n components: (*Hint: you should use* `twinx()` *to make the plot fit*), *and*, (iii) mark the point at which the number of components collectively explain at least 95% of the variance (using a vertical line). *Hint: Number components starting from 1.*



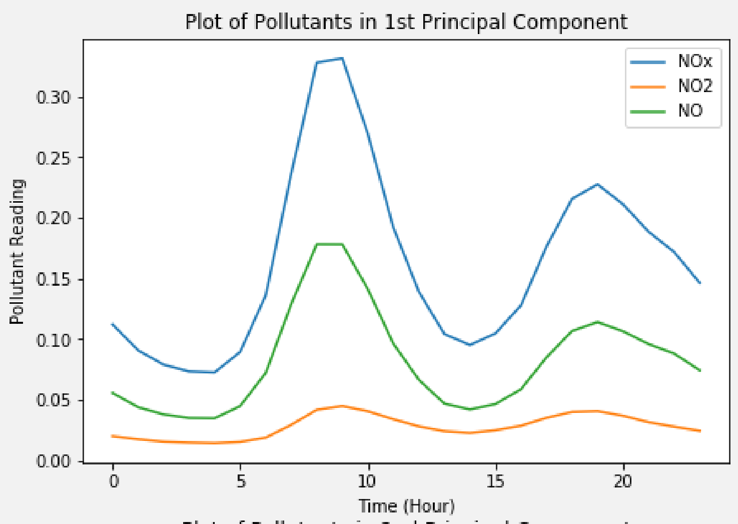
Question 5 continued on next page...

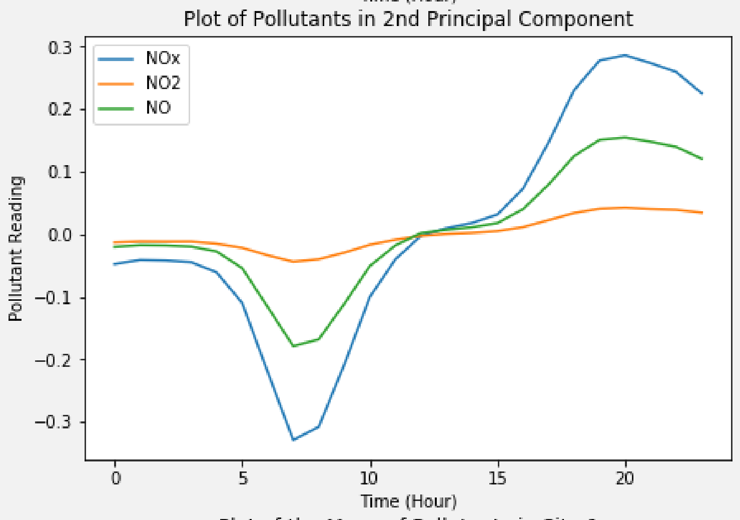
[5.3] (2 points) Interpret and summarise the above plot.

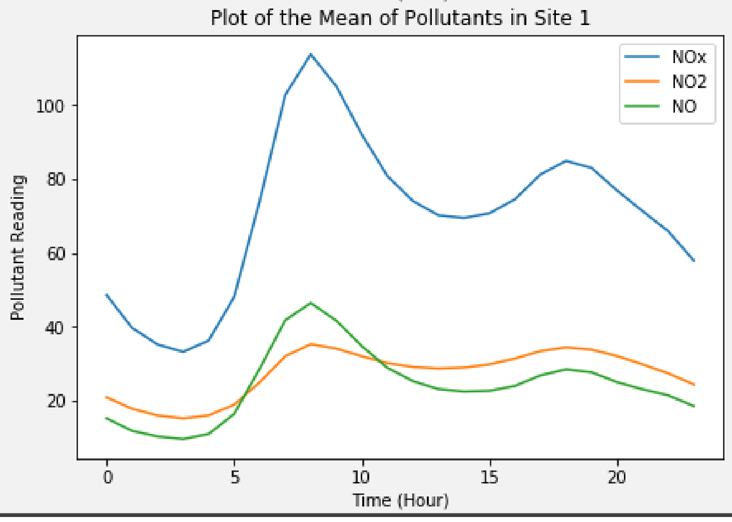
As seen from the vertical bars in the plot from the previous question, the first principal component explains more of the total variance than all of the other components combined, thus only 11 principal components are needed to explain 95% of the total variance. The red line shows the cumulative percentage of the total variance over all 72 principal components, with the cumulative sum being the total variance of the original data.

Question 5 continued on next page. . .

[5.4] (5 points) Generate three figures, one for the mean and one for each of the first 2 principal components: in each, plot the mean/component as three lines, one for each pollutant throught one day cycle. *Hint: You will need to reshape the components appropriately.*







Question 5 continued on next page. . .

[5.5] (6 points) Focusing on the mean and first principal component, are there any significant patterns which emerge throughout the day? *Hint: Think about car usage throughout the day.* What is different when interpreting the mean versus the first component? *Hint: Do peaks signify the same thing in both cases?* Looking at the principal components only, are there any significant differences between the pollutants? Why could this be happening? *Hint: You can refer to one of the limitations of PCA.*

Comparing the plot of the mean and the 1st principal component, a pattern can be seen on the amount of pollutants throughout the day, namely pollutant readings increase significantly around 9AM before decreasing, and then increase significantly again around 7PM, before decreasing again. This could be due to pollution from cars, as there is an increase in use during rush hours. The issue with comparing the mean and the 1st principal component, is that the values in the mean correspond to pollutant readings, whilst in the 1st principal component they corresponds to the direction of greatest variance in the feature space of site 1, thus they don't correspond to the same thing. There are no significant differences between pollutants when comparing the the first 2 principal components, this could be due to the fact that PCA is scale variant and thus is affected by the scale of the data. This can be solved by normalizing the data beforehand, however this was not done.

Question 6 : (41 points) Regression

Given our understanding of the correlation between signals and sites, we will now attempt to predict the NOx level for Site 17 given the value at the other sites. We will evaluate our models using the Root Mean Squared Error (RMSE) i.e. the square root of the mean_squared_error score by sklearn.

[6.1] (2 points) First things first: since we are dealing with a supervised task, we will need to split our data into a training and testing set. Furthermore, since some of our regressors will involve hyper-parameter tuning, we will also need a validation set. Use the `multi_way_split()` method from `mpctools.extensions.skext` to split the data into a Training (60%), Validation (15%) and Testing (25%) set: use the ShuffleSplit object from sklearn for the `splitter`. Set the random state to 0. *Hint: The method gives you the indices of the split for each set, which can then be applied to multiple matrices.* Report the sizes of each dataset.

Size of training set: 8937, size of validation set: 2234, size of testing set: 3724. All sets have a dimension of 9.

Question 6 continued on next page...

[6.2] (4 points) Let us start with a baseline. By using only the y-values, what baseline regressor can you define (indicate what it does)? Implement it and report the RMSE on the training and validation sets. Interpret this relative to the statistics of the data.

The baseline regressor will use the y-values to select the average value for site 17 to use as its prediction. This is done by taking the mean of the y-values and using it as the predicted value for the training and validation set. RMSE of the training set: 79.714, RMSE of the validation set: 80.211. The values gives us the standard deviation of the residuals, which are the prediction errors, and in this case there is a large standard deviation for both data sets indicating a poor performance.

Question 6 continued on next page...

[6.3] (3 points) Let us now try a more interesting algorithm: specifically, we will start with LinearRegression. Train the regressor on the training data and report the RMSE on the training and validation set, and comment on the relative performance to the baseline.

RMSE of the training set: 39.835, RMSE of the validation set: 41.127. When compared to the baseline regressor it can be seen that the RMSE values for the linear regression algorithm are almost half of those reported by the baseline predictor for both data sets, indicating a smaller standard deviation of the residuals and thus a much better performance when the linear regression algorithm is used.

Question 6 continued on next page...

[6.4] (5 points) We want to explore further what the model is learning. Explain why in Linear Regression, we cannot just blindly use the weights of the regression coefficients to evaluate the relative importance of each feature, but rather we have to normalise the features. By referring to the documentation for the LinearRegression implementation in SKLearn, explain what the normalisation does and how it helps in comparing features. Will this affect the performance of the Linear Regressor?

The magnitude of the coefficients indicates how a change in the independent variables affects the target variable. Thus it depends on their magnitudes, and if they are not of the same scale comparison is meaningless. To make comparisons meaningful normalization is used to get a uniform scale of the data. By the SKLearn documentation, the input data is normalized by subtracting the mean and dividing by the L2-*norm*, which allows for the features to be comapared by looking at the coeffiecients. Normalizing has no effect on the performance of the linear regressor.

Question 6 continued on next page…

[6.5] (5 points) Retrain the regressor, setting `normalize=True` and report (in a table) the ratio of the relative importance of each feature. Which is the most/least important site? How do they compare with the correlation coefficients for Site 17 as computed in Question 4:(f), and why do you think that is?

Site	1	2	4	6	7	10	12	14	16
Coefficient	0.115556	0.009947	0.134139	-0.006799	0.059388	0.079779	0.12911	0.094507	0.161828

As seen from the table above, the feature with the greatest relative importance is site 16 as it has the largest magnitude while site 6 has the smallest one and thus has the least importance. When compared with the correlation coefficients for Site 17, it can be seen that the greater the relative importance of that site, the greater its correlation coefficient relative to the others. This is observed due to the fact that linear regression attempts to create a linear model of the independent variables against the target variable, thus if there is a stronger positive correlation between a feaure and the target variable, then it will have a greater weight on the linear regression model as there is a stronger linear relationship.

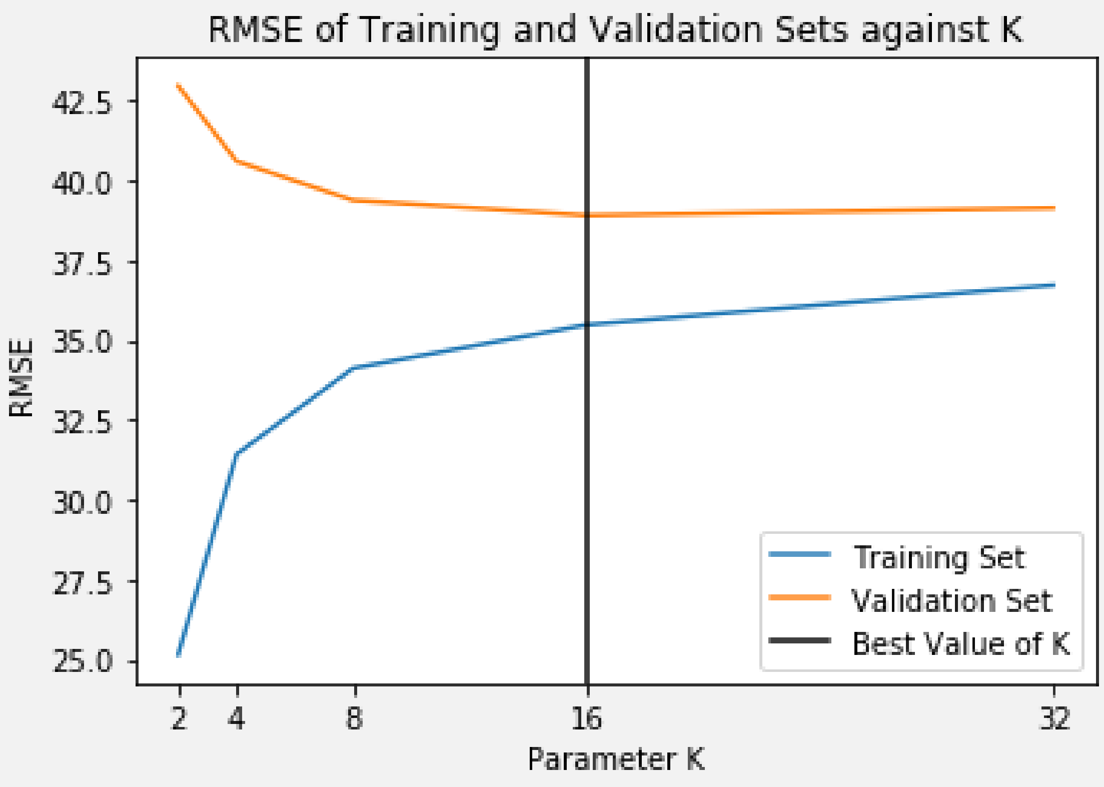
Question 6 continued on next page...

[6.6] (5 points) It might be that with non-linear models, we may get better performance. Let us try to use K-Nearest-Neighbours. Train a KNN regressor with default parameters on the training set and report performance on the training and validation set. *Hint: it might be beneficial to set `n_jobs=-1` to improve performance.* How does it compare with Linear Regression in terms of performance on both sets? What is a limitation of the KNN algorithm for our dataset?

RMSE of the training set: 32.436, RMSE of the validation set: 40.307. When compared to the linear regression performance it can be seen that the KNN regressor performs marginally better on both sets as it outputs lower RMSE values. One of the limitations of the KNN algorithm to the data set is that it can't deal with missing data or outliers such as the negative pollutant readings.

Question 6 continued on next page. . .

[6.7] (4 points) The KNN regression allows setting a number of hyper-parameters. We will optimise only one: the number of neighbours to use. By using the validation set, find the optimal value for the `n_neighbours` parameter out of the values [2, 4, 8, 16, 32]. Plot the training/validation RMSE and indicate (for example with a line) the best value for `n_neighbours`.

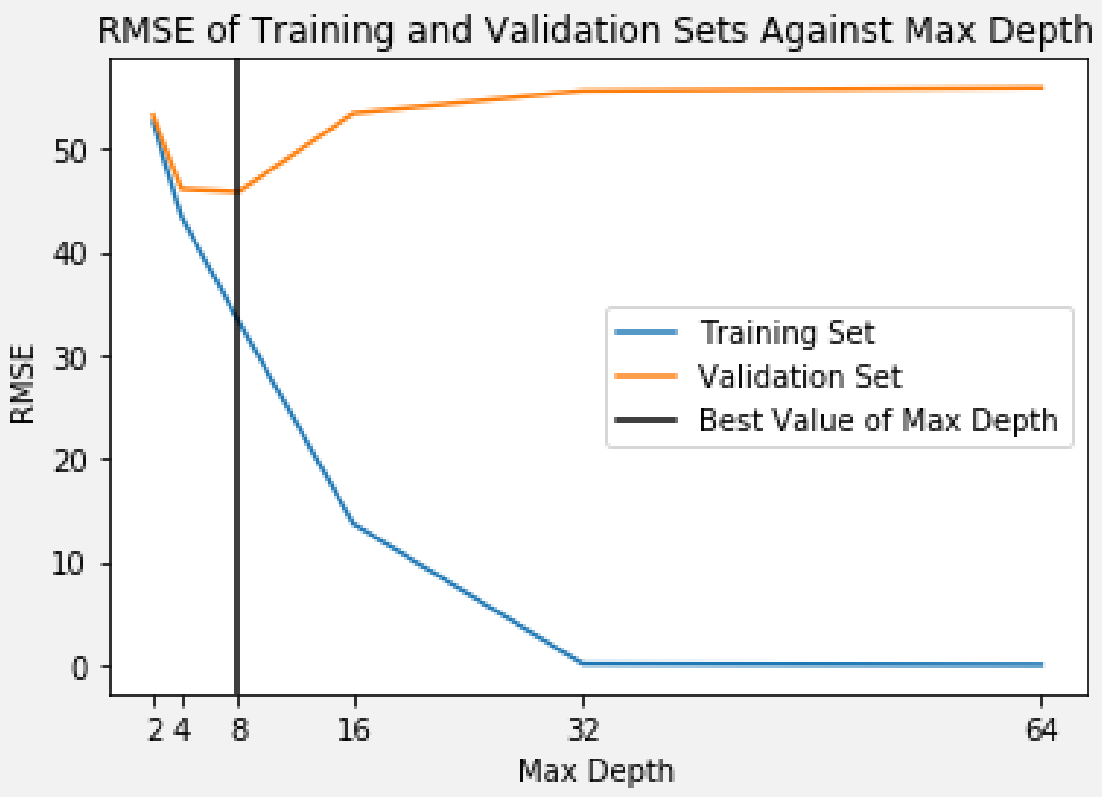


[6.8] (1 points) What is the best-case RMSE performance on the validation set for KNN?

The best case performance on the validation set is when $K = 16$, with an RMSE value of 38.902.

Question 6 continued on next page...

[6.9] (4 points) Let us try one last regression algorithm: we will now use DecisionTreeRegressor. Again, the algorithm contains a number of hyper-parameters, and we will optimise the depth of the tree. Train a series of Decision Tree Regressors, optimising (over the validation set) the `max_depth` over the values [2, 4, 8, 16, 32, 64]. Set `random_state=0`. Plot the training/validation RMSE and indicate (as before) the best value for `max_depth`.



Question 6 continued on next page. . .

[6.10] (3 points) What is the best-case RMSE performance on the validation set? What do you notice from the plot about the performance of the Decision Tree Regressor?

The best case performance on the validation set is when max_depth = 8, with an RMSE value of 45.844. As the max depth increases the decision tree model starts to overfit to the training data more and more. This causes poor predictions on unseen data, which can be seen from the RMSE value of the validation set increasing.

Question 6 continued on next page. . .

[6.11] (5 points) To conclude let us now compare all the models on the testing set. Combine the training and validation sets and retrain the model from each family on it: in cases where we optimised hyper-parameters, set this to the best-case value. Report the testing-set performance of each model in a table *Hint: You should have 4 values.*

	Baseline Regressor	Linear Regression	KNN Regression	Decision Tree Regressor
RMSE of Testing Set	78.937312	40.509027	37.984965	43.068871