IAML – INFR10069 (LEVEL 10): Assignment #2

s1703913

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 has 5648 rows (documents) and 1000 columns (word TF-IDF frequencies), with the data ranging between zero (min) and approx. 0.96 (max). Classes have generally around 740 instances each, with class 7 being the exception: 471 instances. Features are typically zero (zero-inflated model), which in turn shifts their mean (0.001708) and standard deviation (0.034360) to be very close to zero.

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 has 1883 rows (documents), which is a third of the size of the training set. Features are also typically zero (zero-inflated model) with a similar mean and standard deviation as the training set, and their range is between zero (min) and approx. 0.86 (max).

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 TF-IDF weights add to the importance of a word to a document (TF) by taking into consideration the word importance in the whole collection of documents (IDF). I think this is useful here because it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. They are also known as stop-words and are generally filtered. TF-IDF does something similar: it weighs them down by being inversely proportional to the number of times the word appears in the whole corpus.

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.

Given enough time, K-means will always converge, however this may be to a local minimum. This is highly dependent on the initialization of the centroids. As a result, the K-Means algorithm is non-deterministic. One method to help address this issue is the k-means++ initialization scheme, which is the default in the SKLearn implementation of KMeans. It initializes the centroids to be (generally) distant from each other, leading to provably better results than random initialization.

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?

A reasonable number of clusters for this problem is 8, as we have 8 different newsgroups in the dataset. Our intrinsic evaluation will report higher accuracy as less clusters risk being ignored in the alignment of system-to-reference clusters.

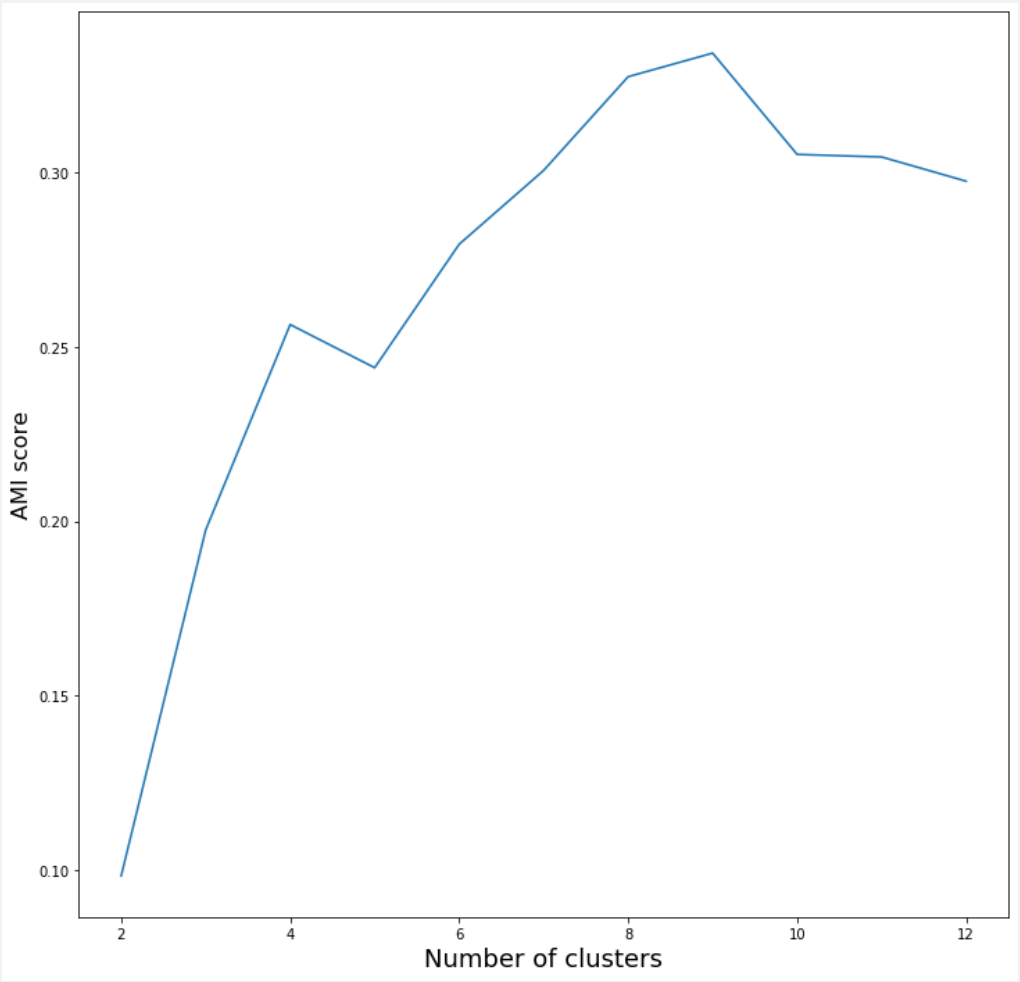
Question 2 continued on next page...

[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) is a function that measures the agreement of two independent label assignments for a dataset (true vs predicted labels). Its expression is given by MI(U, V) = H(U) + H(V) - H(U, V) = $\sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log\left(\frac{P(i,j)}{P(i)P'(j)}\right)$, where H(U) and H(V) are entropies of label assignments i.e. $H(U) = -\sum_{i=1}^{|U|} P(i)\log(P(i))$. This is applicable here because we need to quantify the performance of the clustering: MI gives us an intuitive interpretation and bounded scores (0.0 is as bad as it can be, 1.0 is a perfect score). These metrics might be difficult to use in practice because they require the knowledge of the ground truth classes, which are almost never available or require manual assignment by human annotators.

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).



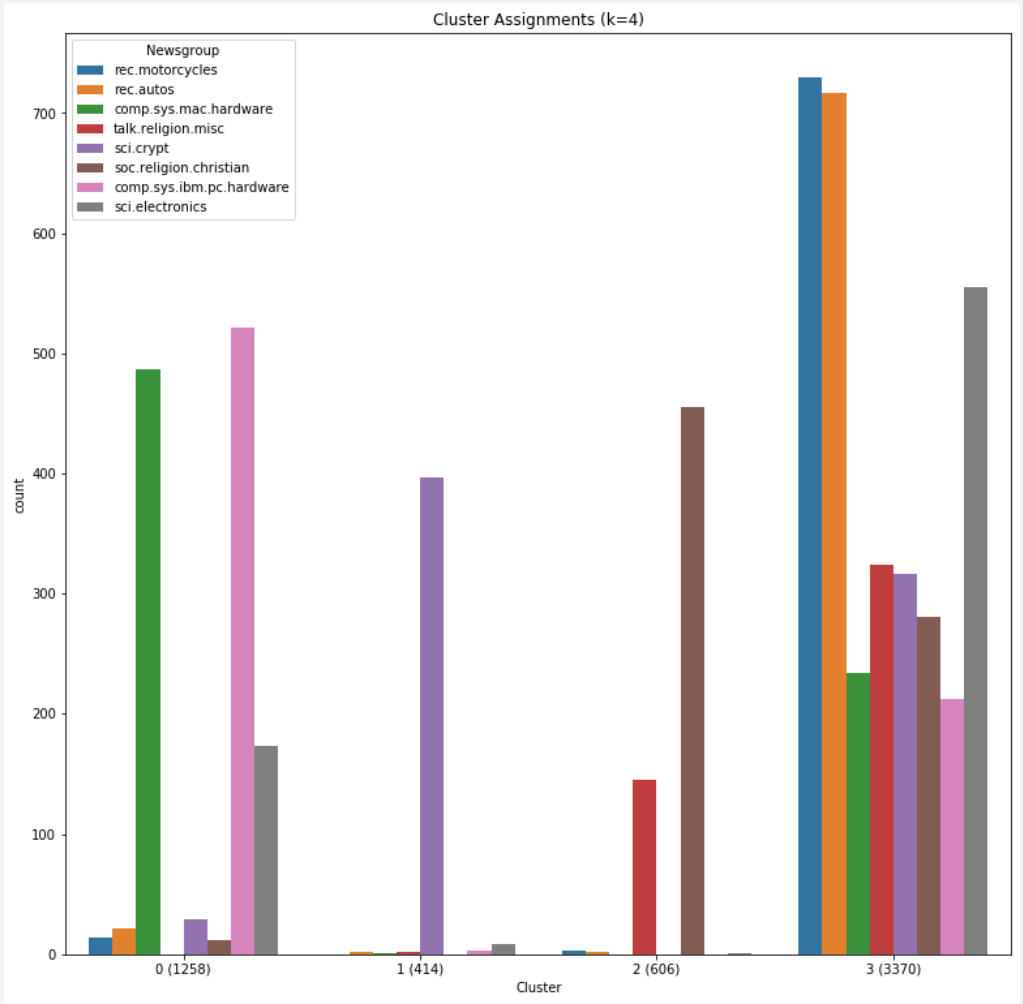
Question 2 continued on next page. . .

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

A trend which can be observed from the plot is how the performance keeps steadily increasing up to the point when the prediction uses 9 clusters, after which it keeps slowly decreasing. This suggests that peak performance is reached at around 9 clusters, which is very close to the true number of classes (8). Furthermore, we notice an additional local peak at 4 clusters, which corresponds to the idea that although we have 8 classes, there are 4 main topics: hardware, religion, mechanics and cryptography.

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.



Question 2 continued on next page. . .

[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)?

Clustering in Q 2(f) aligns fairly accurately with the true class labels, with the mention that Cluster 3 accumulates around 59% of all the data-points, while Cluster 1 is left with 7%. Cluster 0 has successfully grouped together two similar classes 'comp.sys.mac.hardware' and 'comp.sys.ibm.pc.hardware', while cluster 2 has also grouped together two similar classes:'talk.religion.misc' and 'soc.religion.christian'. Cluster 1 has isolated the independent class 'sci.crypt', and Cluster 3 has grouped together classes 'rec.motorcycles', 'rec.autos', and even 'sci.electronics'. This is in direct accordance with the observations at Q 2(e), where the AMI scores shows a general peak of performance close to 8 clusters as well as at 4 clusters.

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 harmonic mean of recall and precision: $\frac{2 \times precision \times recall}{precision \times recall}$, where recall is $\frac{TP}{TP+FP}$ and precision is $\frac{TP}{TP+FN}$. This is preferable to accuracy because we are interested in the FP and FN assignments: the F1-score penalizes extremes in those two categories. In multi-class classification, the final score (using the macro-average setting) is the unweighted mean of the F1-scores for each class.

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`.

Whenever we investigate an ML model, we want to know whether it performs better than the competition, especially if there is a simpler or more tractable approach (known as the baseline). I have chosen a Gaussian Naive Bayes as my baseline model because it is simple and also extremely useful in general as a system in modelling human languages. It obtains an F1-score of 0.56 on the Test set.

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.*

Logistic Regression can be applied to classify multi-class labels by calculating a different weight vector (model) w_k for each class which classifies new instances into either class K or not class K. Having c such vectors, we can apply the Softmax function $\frac{exp(w_k \times x)}{\sum_{j=1}^{c} exp(w_j \times x)}$ which raises the logits to the power of e to bring negative numbers into the range (0,∞) and then normalizes all the probabilities so they sum up to 1.

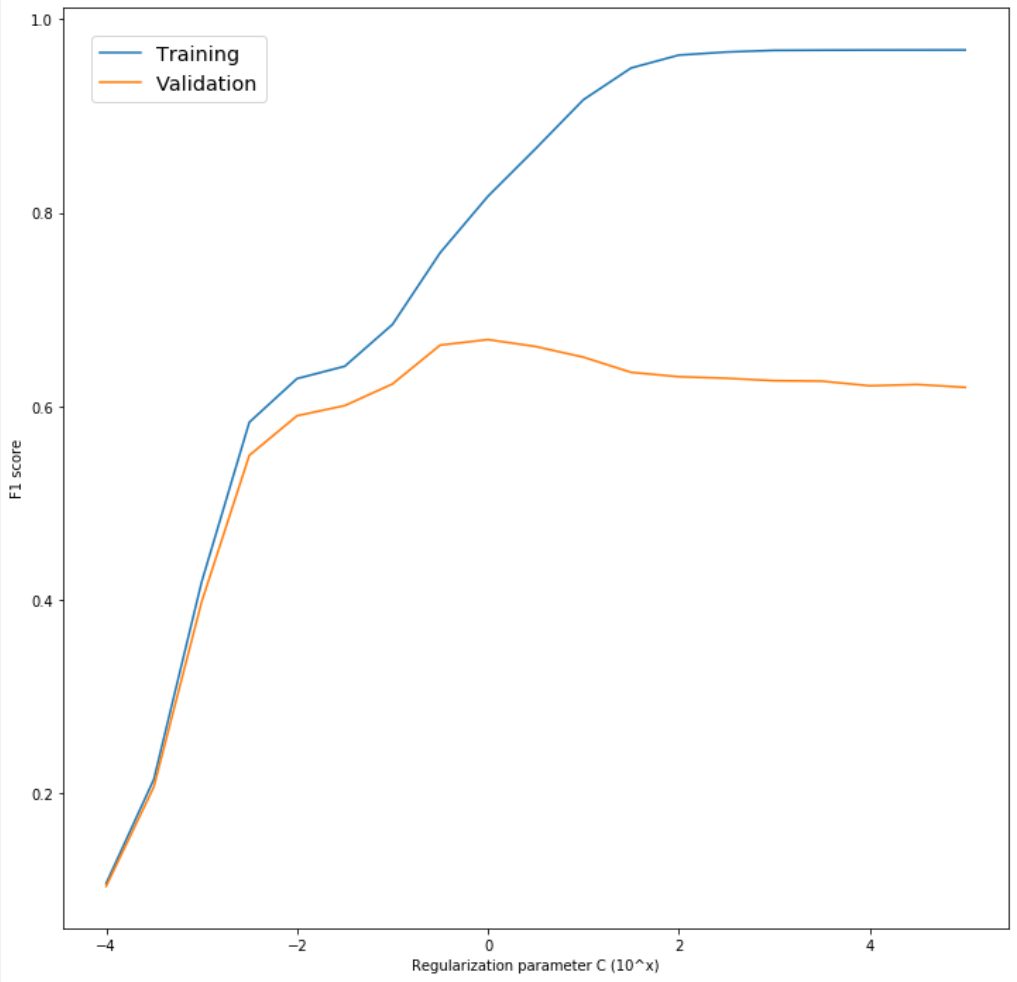
Question 3 continued on next page...

[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.

By training a Logistic Regressor on the training data, we obtain an average F1-score of 0.67 with a fairly small standard deviation: 0.017. This translates to an improvement over the baseline classifier (0.56) because of the consistently higher F1-scores obtained (as indicated by the small standard deviation).

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 can be found at the point where performance on the validation set is high, but performance on the training set is low enough not to indicate overfitting. This can be observed in the plot at Q 3(e) at around x=0 (corresponding to a C value of 1 and an F1-score of 0.67). The plot displays the evolution of the performance of classification as the penalty for increasing the magnitude of parameter values decreases (smaller values of C specify stronger regularisation). We notice how as regularisation weakens, overfitting starts to take place.

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.

After retraining the Logistic Regressor on the entire training set, the F1-score of the best model (C=1) on the test-set is 0.67. This is similar to the performance of the classification when training was done on 90% of the data, in the 10-Fold stratified split.

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 has 1306758 measurements (rows) with 7 attributes (columns) each: *SiteID*, *Loc.Lat* and *Loc.Long* identify the location, *Date Time* the moment of measurement and NO_x, NO_2 and NO are obtained from chemiluminescent analysers (API T200) in $\frac{\mu g}{m^3}$. NO_x has a higher range (up to 2164) when compared to NO_2 (576) and NO (1231). There are 18 sites, numbered 0 to 17. Missing data and negative values occur quite often. The latter are unreasonable here because they defy the logic of the data: pollutant values as measured by the API T200 analysers should never be negative (unless because of calibration errors or the corruption of the Auto Zero filter), as should any of the other columns.

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	Samples	Missing Samples %	Negative Samples %
0	6446.0	1.61	0.0000
1	163111	6.29	0.0000
2	62990	4.35	0.0048
3	25464	77.33	0.7776
4	74787	2.07	0.0053
5	113952	8.83	0.0000
6	142141	7.44	0.0028
7	115162	4.19	0.2779
8	43824	21.06	0.0000
9	22071	5.30	0.0000
10	96407	3.59	0.0041
11	20693	1.90	0.0870
12	45240	17.48	0.0000
13	12423	51.46	0.0161
14	113951	10.53	0.0000
15	2712	100.00	0.0000
16	154331	6.53	0.0136
17	91053	6.27	0.0022

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.

Sites are unevenly distributed in terms of number of samples: the mean is of 72598 samples, but the standard deviation is very high (53066). This translates to a high variability in the number of samples among sites e.g. sites 0 and 15 have 6446 and 2712 samples, while sites 1 and 16 have 163111 and 154331 samples. In terms of problematic samples, each site has, on average, around 5% missing samples, with a few exceptions: most notably, site 8 has 21%, site 13 has 51%, site 3 has 77% and site 15 has 100% (every entry is corrupted). Additionally, 10 sites have a small percentage of negative samples (<1%) and the rest of them have no such 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

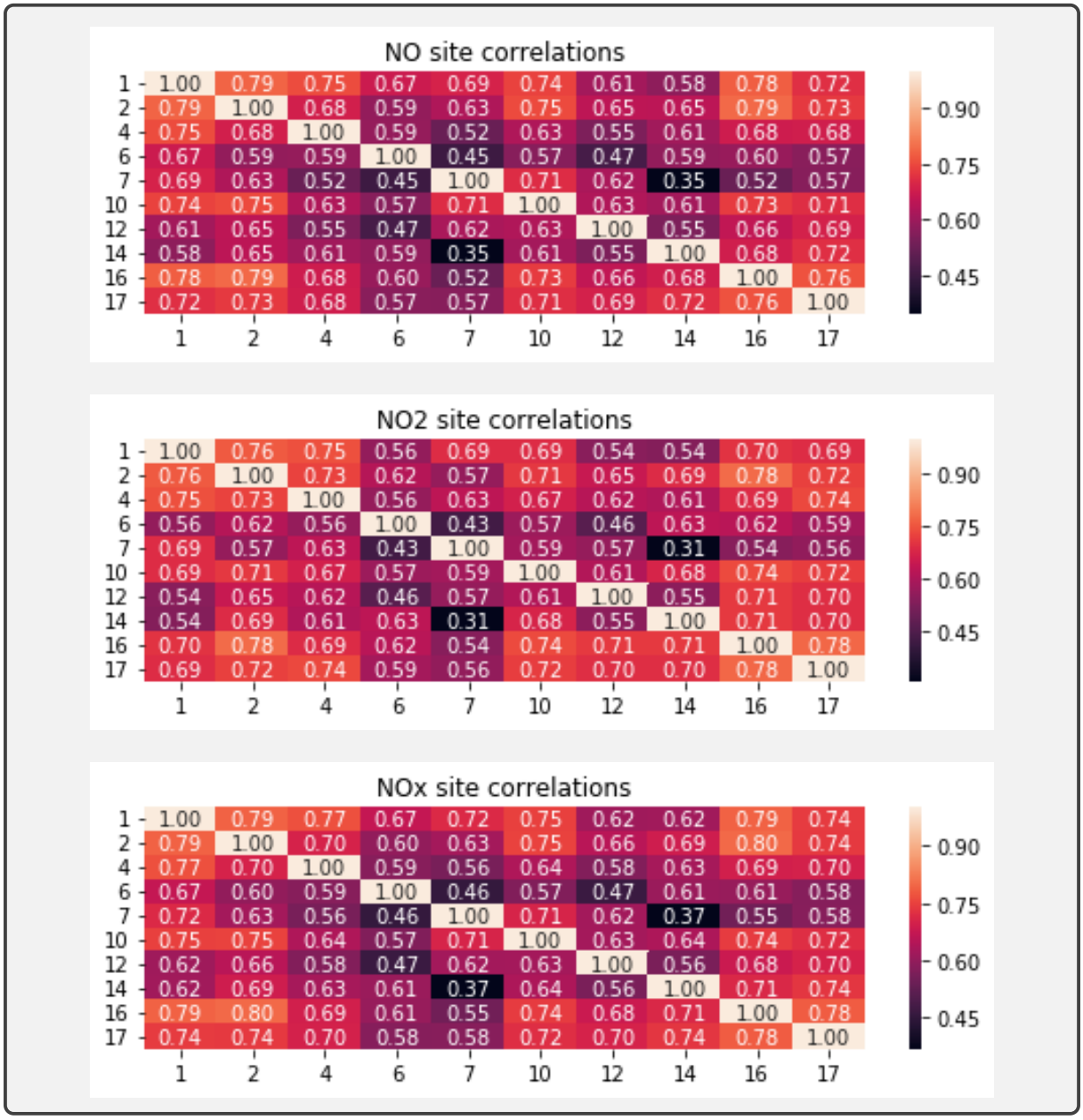
Question 4 continued on next page. . .

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

NO_x and NO have the highest correlation (0.99), followed closely by NO_x and NO_2 at 0.88, and NO and NO_2 at 0.8. We notice that all pairs of pollutants are generally very highly correlated.

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. . .

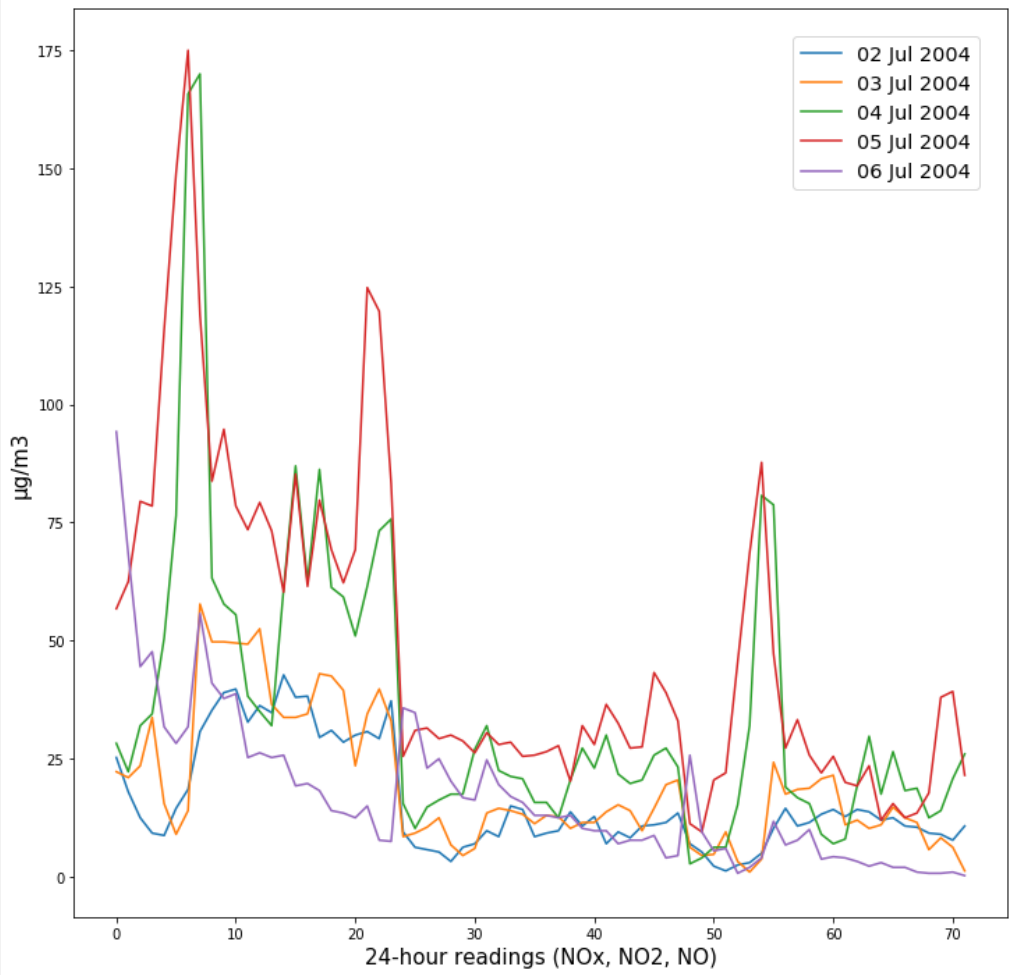
[4.7] (4 points) Comment briefly on your observations from Question 4:(f): start by summarising the results from the NO gas and then comment on whether the same is observed in the other gases or if there is something different.

The NO gas indicates correlations of over 0.5 overall, with three exceptions being between sites 14 and 7 (0.35), and between sites 7 and 6 (0.45) and 12 and 6 (0.47). The same exceptions are also observed in the NO2 and NOx heatmaps of correlations across sites.

Question 5 : (19 Points) Principal Component Analysis

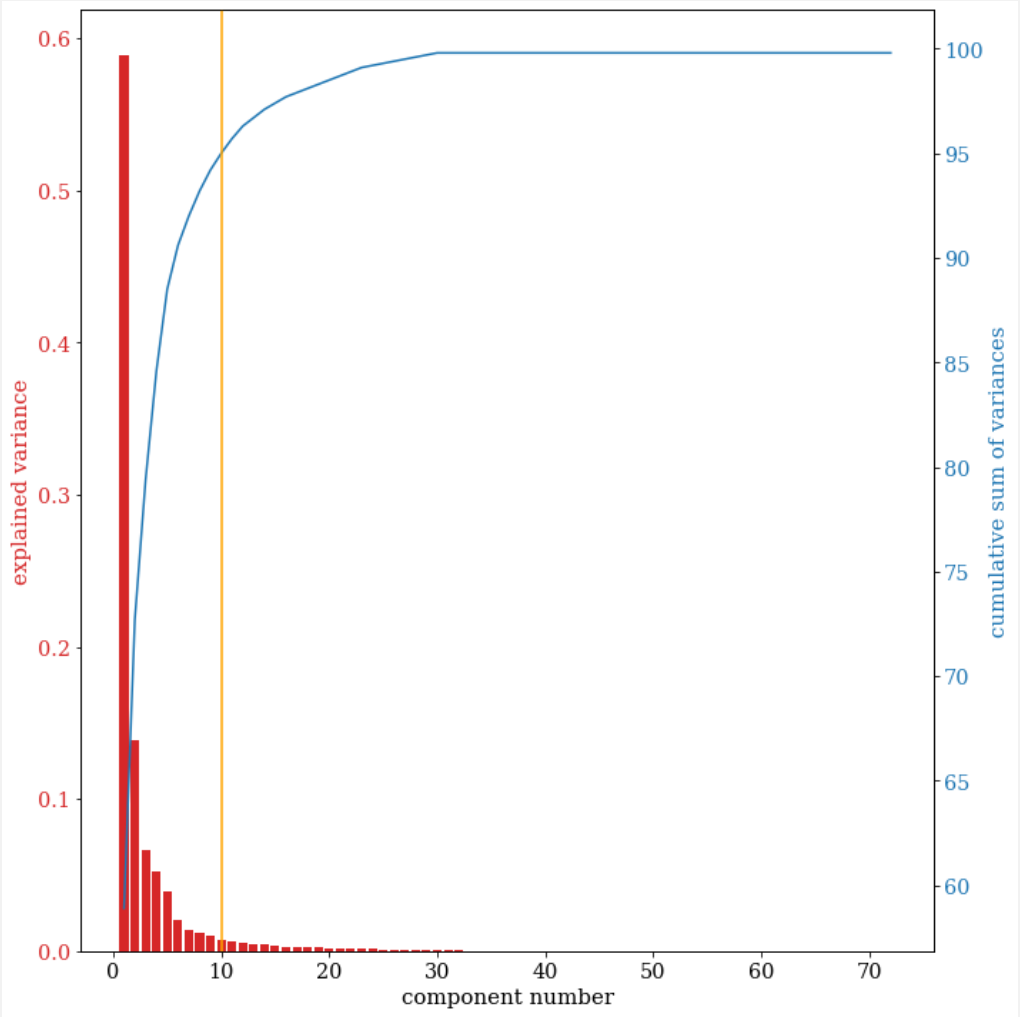
One aspect which we have not yet explored is the temporal nature of the data. That is, we need to keep in mind that the readings have a temporal aspect to them which can provide some interesting insight. We will explore this next.

[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.*



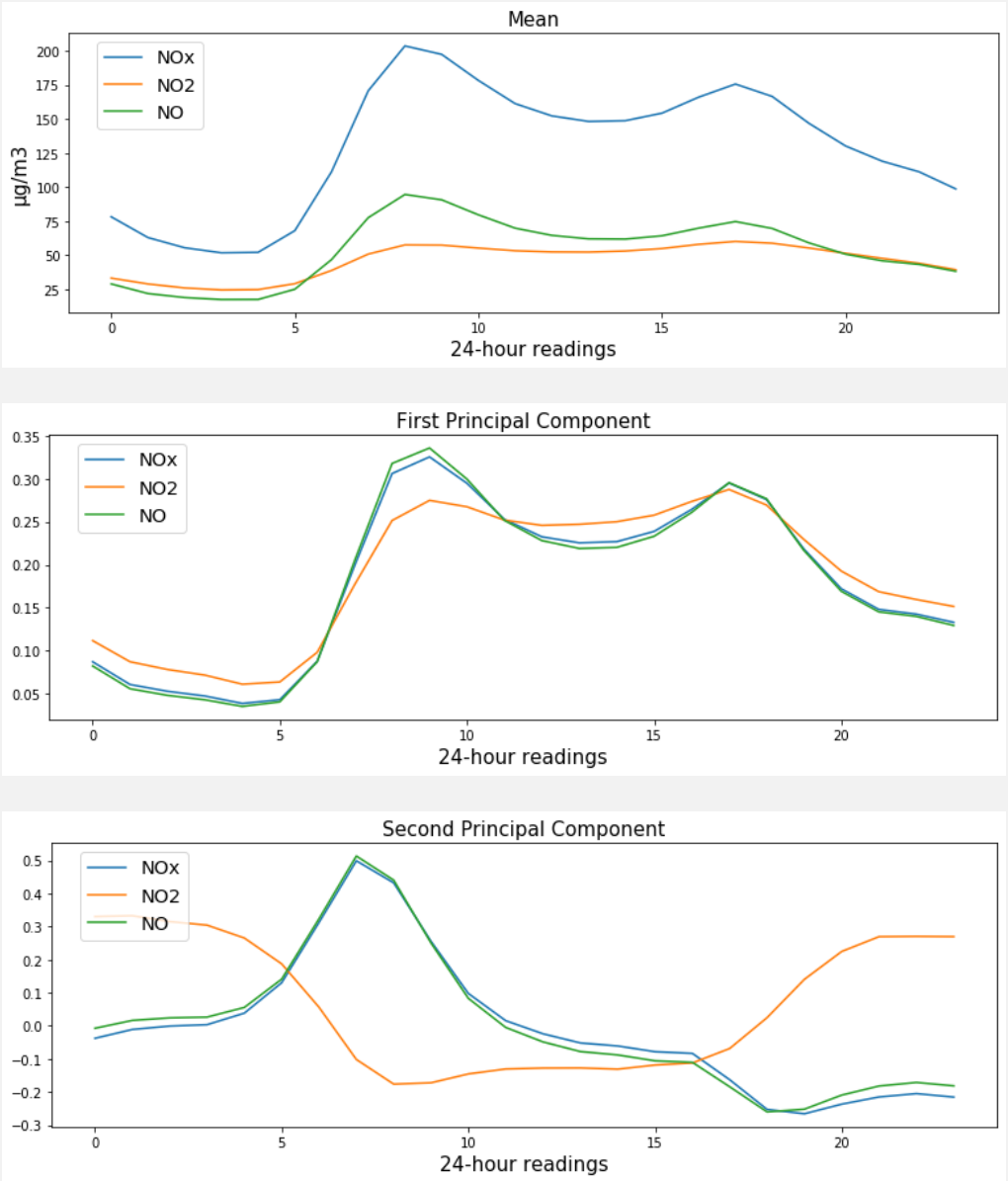
Question 5 continued on next page. . .

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

In the above plot we notice how the first 10 principal components collectively explain most (95%) of the variance in the data, with the first one covering 58.9%. The component-wise explained variance (red bar-chart) shows that the Principal Components have been sorted in decreasing order by their eigenvalues (as expected), and that the cumulative sum of variances (blue line-plot) is inversely proportional to the PCs because they are sequentially added to the sum, increasing it.

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.*

When analysing the 24-hour readings for the three pollutants (NOx, NO2, NO), we observe significant patterns emerging throughout the day which correspond to car usage during rush hours: values triple over a short period of time, starting from 6am and up until 10am, after which they slightly decrease but remain stable until around 5pm, when they start to rapidly decrease. The peak for the mean signifies the average hour when the measurement of $\frac{\mu g}{m^3}$ peaks, while the peak for the first principal component represents the hour when the measurements record the highest variability on different days i.e. pollutant levels vary significantly at that time on different days. In the second principal component, the NO2 measurements differ significantly from those of NOx and NO. This is linked to a limitation of the PCA technique: it assumes linearity in the data, while the readings are clearly non-linear.

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.

The training set has 8937 measurements, the validation set has 2234, and the testing set has 3724.

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.

I will use the SKLearn *DummyRegressor* as a baseline regressor, with the strategy set to "mean": always predicts the mean of the set's y-values. RMSE on the training set is 79.71, while RMSE on the validation set is 80.21 (in this case they are also the std. devs. of the y-values for the training and validation sets). Given that the mean of the data is 98.32 (training) and 99.89 (validation), and that the 90th percentile is 197.25 (training) and 195.20 (validation), RMSE is very large.

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.

After training a Linear Regressor on the training data, RMSE on the training set is 39.83 and RMSE on the validation set is 41.13. Performance relative to the baseline regressor has more than doubled, with an improvement of 59.48% on the training set and 58.82% on the validation set.

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?

As the magnitudes of the coefficients express how much a unit-change in the independent variable (feature vector) affects the dependent variable (target), we know that they depend on both those variables. To compare the coefficients, we need to make sure that the features are on the same scale by normalising them. In the SKLearn implementation, normalisation is done by subtracting the mean and dividing by the l2-norm (square root of the sum of the squared vector values). This will **not** affect 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 ID (feature)	% Relative importance
1	14.61
2	1.26
4	16.96
6	0.86
7	7.51
10	10.09
12	16.32
14	11.95
16	20.46

The most important site is site 16, and the least important site is site 6. This is in accordance with the correlation coefficients for Site 17 at Q 4(f), because we observe that the correlation between NOx measurements at site 17 and site 16 is highest (0.78), and correlation between site 17 and site 6 is lowest (0.58).

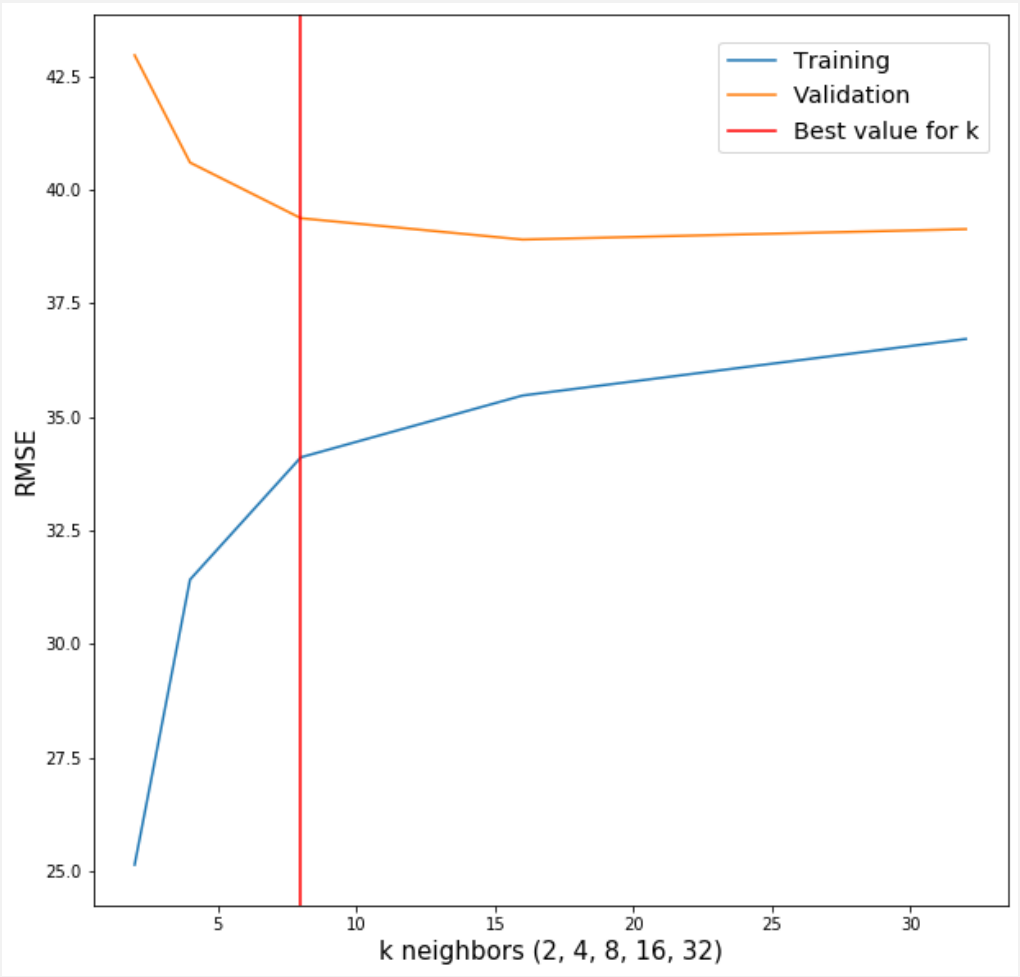
Question 6 continued on next page. . .

[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?

The KNN regressor has a RMSE of 32.44 on the training set, and a RMSE of 40.31 on the validation set. This brings a slight improvement on the training set (18.55%), but is insignificantly better on the validation set (1.99%). A limitation of the KNN algorithm for our unnormalised dataset is that it is completely necessary that features have the same scale, since absolute differences in features weight the same, i.e., a given distance in feature 1 must mean the same for feature 2.

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`.



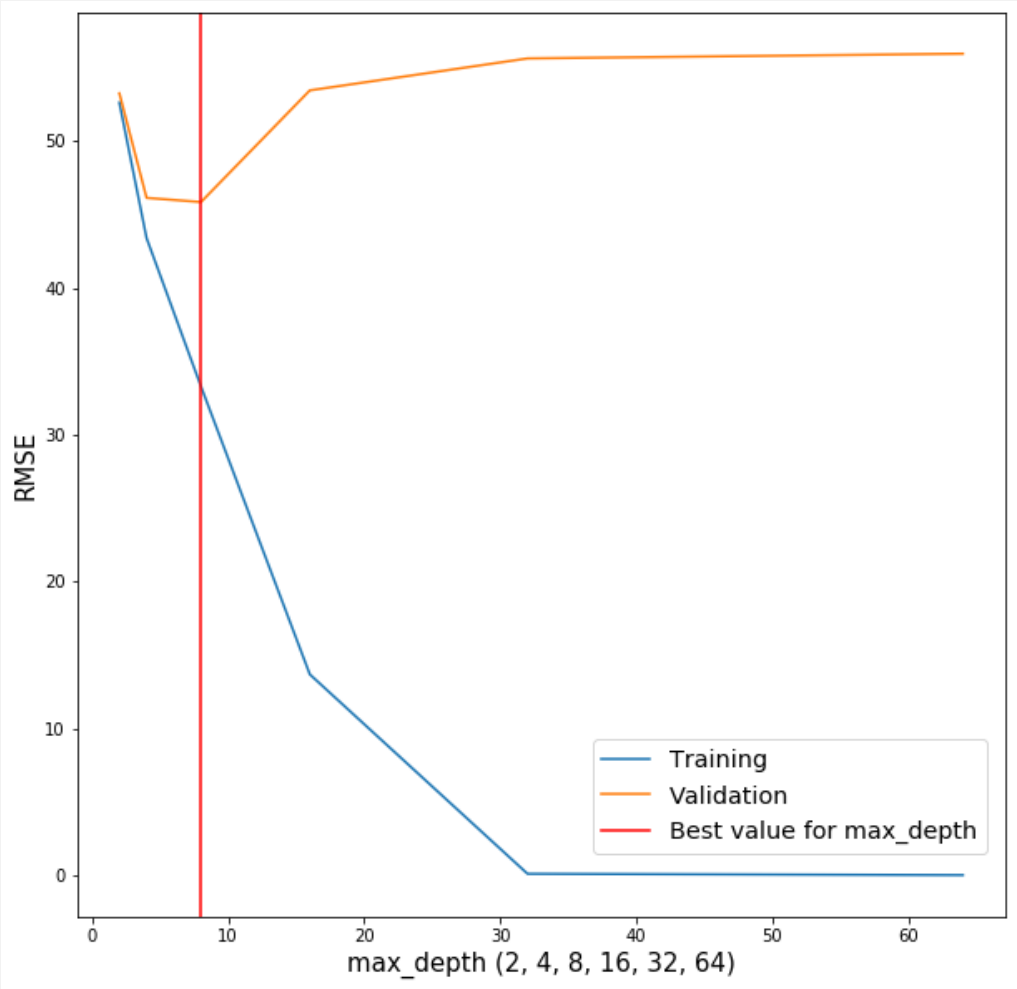
Question 6 continued on next page. . .

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

The best-case RMSE performance of the KNN regressor on the validation set is 38.90, for k=16.

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 RMSE performance on the validation set is 45.84, with k=8. From the plot we notice that as the depth of the tree increases, performance on the training set rapidly increases, but performance on the validation set decreases. This is because overfitting starts to take place (RMSE quickly approaches 0 on the training set).

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.*

Testing-set performances (RMSE):

Baseline	Linear Regressor	KNN Regressor	Decision Tree Regressor
79.24	41.16	39.52	42.61