IAML – INFR11182 (LEVEL 11): Assignment #2

Due on Friday, November 15, 2019 @ 16:00

NO LATE SUBMISSIONS

IMPORTANT INFORMATION

N.B. *This document is best viewed on a screen as it contains a number of (highlighted) clickable hyperlinks.*
It is very important that you read and follow the instructions below to the letter: you will be deducted marks for not adhering to the advice below.

Good Scholarly Practice: Please remember the University requirement as regards all assessed work for credit. Details about this can be found at:

http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct

Specifically, this coursework must be **your own work**. We want you to be able to discuss the class material with each other, but the coursework you submit must be your own work. You are free to form study groups and discuss the **concepts related to, and the high-level approach to,** the coursework. **You may never share code, or share writeups. It is also not permitted to discuss this coursework on Piazza.** The only exception is that if you believe there is an error in the coursework, you may ask a **private** question to the instructors, and if we feel that the issue is justifed, we will send out an announcement.

General Instructions

N.B.: This is the Level 11 version of the course (INFR11182): if you are an undergraduate student, then you should go to the Level 10 version.

- This assignment consists of two parts each of which is a self-contained set of tasks. The first deals with the 20 Newsgroups Dataset which we have already encountered

in Assignment 1. The second makes use of the Bristol Air Quality Dataset. We provide (multiple) curated versions of each for your use: **MAKE SURE to use the version asked for in the particular question.** All the data can be found in the Assignment Repository, alongside with other tools.

- You should use python for implementing your solutions as this will standardise the output and also provide a consistent experience with the labs. Set up your environment as specified in the Labs **but use the provided requirements file in this repository!** This is important because it specifies **updated** versions of the libraries with features that you will need to implement the assignment questions.
- In order to further help with some of the implementations, we have provided a set of tools for you. These can be found in the `mpctools` repository. You will mostly use the modules within the `mpctools.extensions` package. Download the library and then follow the instructions in the repository to install it within your virtual environment.
- This assignment accounts for 25% of the Mark for this course and is graded based on a written report (compiled from a latex template which we provide) which you are to submit via Gradescope (see below). The actual assignment is marked out of 175 and we will normalise this to the required score.
- The criteria on which you will be judged include the quality of the textual answers and/or any plots asked for. While code will be needed to generate the results, this is not a programming assignment, and you are not expected to provide code unless explicitly requested.
- Read the instructions carefully, answering what is required and only that. Keep your answers brief and concise. Specifically, **for textual answers**, the size of the text-box in the latex template will give you an idea of the **maximum** length of your answer: you do **not need** to fill in the whole text-box but you will be **penalised** if you go over. This does not apply to figure-based answers.
- For answers involving figures, make sure to clearly label your plots and provide legends where necessary. You will be penalised if the visualisations are not clear.
- For answers involving numerical values and especially tables, use correct units where appropriate and format floating point values to a reasonable number of decimal places.

Submission Mechanics

Important: ***You must submit this assignment by Friday 15/11/2019 at 16:00. We do not accept Late Submissions for this coursework, except in the case of mitigating circumstances. Please refer to the ITO Website for further details.***

- We will use the Gradescope submission system for uploading PDF assignments, similar to the process employed in Assignment 1.

- You should clone or download the Assignment Repository from https://github.com/michael-camilleri/INFR11182-2019. Apart from a copy of these instructions, this contains:

  1. conda.req: This is the conda requirements file. You should set up a new environment with this version of the libraries.
  2. The data you will need for the assignment under the Data directory. This is organised by Assignment Parts.
  3. Two `tex` files, `Assignment_2.tex` and `style.tex`. These provide the template for you to fill out the assignment questions. In particular, the template forces your answers to appear on separate pages and also controls the length of textual answers.

- You should modify the template `Assignment_2.tex` **only** by:

  1. Uncommenting and specifying your student number in Line 42 (compilation will automatically fail if you forget to do this), and
  2. Filling in the answers in the provided `\answerbox` environment.

  DO NOT modify anything else in the template and certainly do not tinker with the style file. **We reserve the right to not mark assignments which do not adhere to the template**.

- For example, to fill in textual answers, you would modify the corresponding `answerbox`:

  ```
  \answerbox{6em}{
  Your answer here
  }
  ```

  with your answer (replacing 'Your answer here'):

  ```
  \answerbox{6em}{
  Steam locomotives were first developed in the United Kingdom
  during the early 19th century and used for railway transport
  until the middle of the 20th century. Richard Trevithick
  built the first steam locomotive in 1802.
  }
  ```

  which, when compiled gives:

  > Steam locomotives were first developed in the United Kingdom during the early 19th century and used for railway transport until the middle of the 20th century. Richard Trevithick built the first steam locomotive in 1802.

  Alternatively, for filling in an image, you would use:

```
\answerbox{20em}{
\begin{center}
\includegraphics[width=0.9\textwidth]{stock_image.jpg}
\end{center}
}
```

which can be compiled to:

- Once you have filled in all the answers, compile the latex document (you can use your favourite latex editor or just run `pdflatex Assignment_2.tex` twice) to generate the pdf which can then be uploaded to Gradescope.
- Use the same procedure to upload the document and specify the locations of your answers as for Assignment 1. Make sure that when asked whether you wish to SUBMIT IMAGES or SUBMIT PDF, you pick **SUBMIT PDF**.
- It is important that after you upload your submission, you specify the page corresponding to each question correctly, as this streamlines the job of the markers allowing us to give you feedback in good time. We reserve the right to not mark answers whose locations are incorrectly specified.
- The following links may be useful:
  - Gradescope Submission System
  - Gradescope help page

PART A: 20-NEWSGROUPS [77 POINTS]

This dataset is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups, each corresponding to a different topic. If you look at the topics, you may notice that they group together under some general headings.

In order to make the problem more manageable, we will only use 8 out of the full 20 different groups. These are numbered **sequentially**:

0. *comp.sys.ibm.pc.hardware*,
1. *comp.sys.mac.hardware*,
2. *rec.autos*,
3. *rec.motorcycles*,
4. *sci.crypt*,
5. *sci.electronics*,
6. *soc.religion.christian*, and
7. *talk.religion.misc*.

In contrast to Assignment 1, we have opted to use Term-Frequency/Inverse-Document Frequency TF-IDF weights for each word instead of the frequency counts. These weights represent the importance of a word to a document, with respect to a collection of documents. The importance increases proportionally to the number of times a word appears in the document, and is inversely proportional to the number of times the word appears in the whole corpus.

Question 1 : (10 points) Exploratory Analysis

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

We have preprocessed the data for you, and in addition split it randomly into a training `20ng_train.csv` and testing `20ng_test.csv` set. These are stored in csv file format (compressed for space) within `Data/PartA`: note that the first line in each file contains the header (column names). We also provide you with a file, `20ng_labels.csv` which contains the class labels in the correct order (i.e. where integer label n corresponds to the n^{th} class name): note that this contains a header like a dataframe but is in reality a pandas series. The additional file `20ng_train_hierarchical` will be used for the last question in this section.

Load the training and testing datasets into two pandas dataframes. You may use `pandas .read_csv()`: remember to set the compression flag to 'bz2'. Note that each file contains both the features as well as the target, as the last column labelled 'class'. You should extract the features and targets into separate dataframes/arrays.

Question 1 continued on next page. . .

[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?

[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?

[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?

Question 2 : (24 points) Unsupervised Learning

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

We will be focusing mainly on K-Means clustering. In what follows we will use **ONLY** the training data.

[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.

[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?

[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.*

[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?

[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:(6) align with the true class labels? Does it conform to your observations in Question2:(5)?

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).

In most cases, we will need to validate hyper-parameters: to this end, we will use K-Fold cross validation. Using the provided SKLearn implementation for StratifiedKFold, create a 10-Fold **stratified split** of the data. Make sure to set the random state to 0 for reproducibility, and turn on shuffling. We will use the same splits provided by this throughout this question.

[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?

[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`.

[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.*

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.

[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}.*

[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:(5) *Hint: Refer to the documentation for C in the LogisticRegression page on SKLearn.*

[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.

Question 4 : (17 points) Hierarchical Classification

We will now leverage the structure of the target labels to try out hierarchical classification.

Specifically recall that the 8 labels of interest may be grouped into 4 pairs as follows:

computing (0)	**recreational** (1)	**science** (2)	**religion** (3)
`comp.sys.ibm.pc.hardware`	`rec.autos`	`sci.crypt`	`talk.religion.misc`
`comp.sys.mac.hardware`	`rec.motorcycles`	`sci.electronics`	`soc.religion.christian`

We have provided groupings for the training dataset in `20ng_train_hierarchical.csv` but not for the testing set. This file contains the original features, the original classes, but also the super-labels under the heading `super`, which are numbered in order 0 through 3 in the order given above. We will thus attempt to train a hierarchical classifier by first training a base classifier on the super-labels, and then training individual binary classifiers on each of the sub-groups.

Load the training data from the file `20ng_train_hierarchical.csv`. We will use the same 10-fold Cross validation as before. Also make sure that wherever there is an element of randomness, you set the seed to 0 to ensure replicable results.

Question 4 continued on next page...

[4.1] (2 Marks) What aspects of the data may lend to better classification in such a hierarchical fashion?

[4.2] (3 points) First train a Logistic Regressor on the high-level classes (i.e. the four super-labels): use the same setup as before (Question 3), optimising the regularisation parameter C over the folds. Report the best validation-set F1-score (average over folds) together with the optimal value of C. *Hint: Remember to keep track of the* ***best*** *classifier trained on the* ***entire*** *dataset (i.e. training data with no folds).*

Can we compare this result to the previous ones?

[4.3] (2 points) We will now train individual binary classifiers for each of the groups. Why should we use the true super-class targets and not the ones predicted from the above classifier?

[4.4] (7 points) Train four independent Logistic Regression (binary) classifiers on the two classes within each super-group. That is, for each super-group, extract only the samples corresponding to that super-group using the true group label, then optimise a Logistic Regression classifier on the data with the targets being the two classes in that super-group. Report in each case the regularisation value which gives the best validation-set score as well as the associated F1-score. *Hint: This would look best in a table. Also, remember to keep track of the best classifier trained on the entire group in each case.*

[4.5] (3 points) Using the trained classifiers in a hierarchical fashion, evaluate the resulting model on the testing set for the full 8-way classification. *Hint: You will need to first generate the group-level predictions using the base classifier, and then, conditioned on this, predict the individual labels. Make sure to construct the indices correctly.* How does this compare to the original single-layer classifier?

PART B: BRISTOL AIR-QUALITY [98 POINTS]

For the next part of the assignment we will use a subset of the Bristol Air-Quality Dataset. This dataset deals with monitoring several pollutants in and around the city of Bristol. These are identified by latitute/longitude positions which are further assigned a unique `SiteID`. Readings are taken at hourly intervals from across the sites.

In order to make the analysis manageable we will again provide you with partially curated versions of the dataset, stored as compressed csv files under the `Data/PartB` directory. Again, the files contain a header within the first row. Note that we provide multiple versions of the data for different questions, and in some cases, we anonymise aspects of the data: make sure to use the version that is appropriate for the question.

Question 5 : (30 Points) Exploratory Analysis

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

Load the dataset `BristolAir_Exploratory.csv` into a pandas dataframe. Again, you should use `pandas.read_csv()` with the compression flag 'bz2': remember also that the first line is the header.

[5.1] (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.*

[5.2] (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.*

[5.3] (4 points) Briefly summarise how the sites compare in terms of number of samples and amount of problematic samples.

Question 5 continued on next page. . .

[5.4] (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.

[5.5] (2 points) Comment on the level of correlation between each pair of pollutants.

Another interesting analysis is to look at the correlations betweens sites. To help you with this, we have again preprocessed the data and provided you with a new dataset in which the data is organised in individual columns per-site, thus aligning together readings from the same time-point. We also only kept columns for which we have a significant number of overlapping time-points, i.e. for sites 1, 2, 4, 6, 7, 10, 12, 14, 16 and 17. Load the file `BristolAir_SiteAligned.csv`. Note that the first two lines are header rows (Pollutant and SiteID respectively) and hence you need to set `header=[0, 1]`: also remember to enforce `compression='bz2'`. Finally, the first column is the time/date of the reading (*Hint: tell* `pandas` *about this by setting* `parse_dates=[0]`). Explore the data and its structure and then answer the following questions.

[5.6] (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`.

[5.7] (4 points) Comment briefly on your observations from Question 5:(6): 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.

Question 6 : (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.

We have provided you with a new version of the dataset, `BristolAir_DayAligned.csv`, in which the data is organised by daily readings. That is the first two-columns indicate the SiteID (for which we have selected a reduced subset to make the problem more manageable), and the Date of the readings. The remaining 72 (24×3) columns (numbered 0 through 71) are the 24-hour readings for each of the 'NOx', 'NO2' and 'NO' pollutants respectively. Load the data into a pandas dataframe. Remember that the first line is the header, and that the index columns should be 'SiteID' and 'Date': also, this is again compressed using 'bz2'. Look at the first few lines and familiarise yourself with the data.

Question 6 continued on next page. . .

[6.1] (1 point) Plot the first 5 lines of data (plot each row as a single line-plot).

You may have noticed that it is very difficult to interpret the data in this manner, especially since there is also a lot of variation between different days. We will address this by employing Principal Component Analysis (PCA), but in this case, we will do this to capture the major sources of variability in the data and help us look at it in more detail. Given that we know about the correlation in the data, we also expect that this will help in dimensionality reduction.

[6.2] (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.*

[6.3] (2 points) Interpret and summarise the above plot.

[6.4] (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.*

[6.5] (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.*

Question 7 : (49 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.

To do this we will again make use of the time-aligned data in `BristolAir_SiteAligned`. Reload this dataset and extract only the NOx values. Then split this into a feature-set (X) and target value for site 17 (y).

Question 7 continued on next page. . .

[7.1] (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.

[7.2] (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.

[7.3] (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.

[7.4] (2 points) Another way of evaluating the applicability of a linear model is to analsye the residuals (errors). The Linear Regression model assumes a Gaussian form for the residuals. Fit a Gaussian to the errors (in the validation data) and report the mean and standard deviation.

[7.5] (4 points) Plot a histogram of both the residuals and the fitted Gaussian. The easiest way to generate a histogram of the Gaussian, is to generate a large number of samples ($\approx 10\times$ the amount of samples in the data) from the distribution (*Hint: refer to norm.rvs from `scipy.stats`*), and feed them to the hist method together with the residuals: i.e. calling `plt.hist(x=[residuals, samples])`. Use 50 bins in the range [-250, 250] and visualise a density plot rather than raw counts.

[7.6] (2 points) By referring to the plot in Question 7:(5), comment on whether your assumption in Question 7:(4) is valid.

[7.7] (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?

Question 7 continued on next page...

[7.8] (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 5:(6), and why do you think that is?

[7.9] (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?

[7.10] (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`.

[7.11] (1 points) What is the best-case RMSE performance on the validation set for KNN?

[7.12] (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`.

[7.13] (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?

[7.14] (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.*