Practical Introduction to Data Science 2020/21

Assessment 3

Done by: Si Han Ang

Introduction:

For the assessment, I used the two datasets specified in the instructions.

For Dataset 1 (UK weather data), wrote python codes to automatically download each station's weather data from https://www.metoffice.gov.uk/research/climate/maps-and-data/historic-station-data into separate text files and another set of python codes to parse and pre-process it and analyse it.

For Dataset 2, I manually downloaded the data sets (years 2011 - 2014) from http://www.ons.gov.uk/peoplepopulationandcommunity/wellbeing/datasets/personalwellbeingestimatesgeographicalbreakdown, pre-processed and analysed the data using python code. I will provide more details in the subsections that follow.

This introduction section has the following sub-sections:

- Folder Structure of the Data and Codes
- Walkthrough of the automatic download codes for UK weather data
- Pre-processing codes for UK weather data

Folder Structure of the Data and Codes

The materials for assessment submissions are organised into a folder structure as follows:

```
├──proc_data
│   ├──happiness
│   └──weather
├──python_notebooks
│   └──.ipynb_checkpoints
├──raw_data
│   ├──weather
│   └──wellbeing
└──supp_data
```

Diving into each of the folders:

The **raw_data** folder is intended to contain raw data files (i.e. data before pre-processing) contains two sub-folders:

- **weather folder**: it contains weather data that is downloaded from Met Office of UK for each station
- **wellbeing** folder: it contains the wellbeing data 2011/2012 data to 2014/2015 data downloaded from

https://www.ons.gov.uk/peoplepopulationandcommunity/wellbeing/datasets/personalwellbeingestimatesgeographicalbreakdown

The **proc_data** folder is intended to contain pre-processed data files. This is the folder where the pre-processed and cleaned weather data and happiness data is housed. This folder again contains two sub-folders:

- **weather** folder: it contains cleaned and pre-processed weather data after such cleaning is done on the raw data.
- **happiness** folder: it contains cleaned and pre-processed happiness data after such cleaning is done on the raw data.

The **python_notebooks** folder contains all the jupyter notebooks that are used for this assessment.

The **supp_data** folder contains the supplementary data provided: regions.txt and stations.txt files.

Walkthrough of the automatic download codes for UK weather data

For dataset 1, the UK weather data for each station can be downloaded automatically by means of writing a programme. I used python for this automatic downloading of UK weather data and the codes are in "Automatic Download of UK Weather Data.ipynb" in the python_notebooks folder.

The downloading is done by using the python library called urllib.request. The logic is as follows: I wrote a function called "retrieve_stn_data" that allows one to download a particular station's data into the **weather** sub-folder of the **raw_data** folder as text file. This function takes in the station name appends '.txt' to it and pre-pends it by a base URL 'https://www.metoffice.gov.uk/pub/data/weather/uk/climate/stationdata/' to be accessed programmatically. Next, I imported the "stations.txt" file as a pandas dataframe object and converted it to a list object. Subsequently, I looped through each element of this list (corresponding to a station name) and pass this station name into the "retrieve_stn_data" function to download.

For more details, please refer to "Automatic Download of UK Weather Data.ipynb" in the python_notebooks folder.

Pre-processing codes for UK weather data

Before I can start to do the analysis in the questions below, I would need to first read in all these station's weather data from the **weather** sub-folder of the **raw_data** folder, this would be followed by extracting the relevant portions of the data to be used, transforming the data extracted, consolidating them and cleaning it. These steps are done in the codes found in "UK Weather Data Preprocessing Script.ipynb" in the **python_notebooks** folder.

The "UK Weather Data Preprocessing Script.ipynb" notebook is split into 3 sections: 1) ETL and Data Consolidation, and 2) Clean each column, and 3) Export for Further Analysis

In the first section, I wrote 3 functions:

- get_lastpos_header: this function takes in two arguments: "headers_lst" (list) and "line" (string); and gets the last characters' position of the headers (which are elements of "headers_lst") in "line" which is a string containing all these headers.

- match_headerlines: this function takes in 4 arguments: “header1_lst”, “header1_lastpos_lst”, “header2_lst”, “header2_lastpos_lst”. All of them are expected to be list-like objects. It matches the last positions of the headers in the first header line with the last positions of the headers in the 2nd header line; for those that could be matched, it implies that the header in the 2nd header line actually a wrapped text from the corresponding header in the first header line. Thus, I concatenate these headers and return them.
- load_and_proc_weather_file: this function loads and processes the weather file by taking the weather file name as input. I loop through each line within the weather data file to: 1) extract the longitude and latitude by matching a regular expression. This comes from the observation that latitude and longitude are always after “Lat” and “Lon” respectively in the files. 2) check if the first header line is found, if so, I call get_lastpos_header function on both first and 2nd header line (assumed to be one line below the first header line) and call match_headerlines to find out which of the headers has wrapped text in the 2nd header line and combine to get a full set of headers. 3) Once the header is found, I would split the subsequent rows (skipping the 2nd header line) using space as a delimiter and store the first N elements where N is the number of headers found in a list 4) Finally, I convert this list containing the data into a pandas dataframe object and put in station names, latitude and longitude as separate columns and return this dataframe object.

The first section involves running “load_and_proc_weather_file” function for each station’s weather data file and concatenating these resulting pandas dataframe together row-wise.

In the “Clean each column” section, I cleaned year column of the dataset of the “Site” entries by removing these rows with such entries and subsequently by making sure the year column contains 4 digits character and converted it to integer type. I cleaned the month column to ensure that is of 1 to 2 digits, convert it to integer type and assign nan when it is not like this. For all the numeric columns of the weather data, I used regex to make sure that the data is either integer or float format by requiring digits to be mandatory followed by optional decimal point and optional digits behind decimal points and converted it to float type; I assigned nan values if it is not numeric.

In the last section “Export for Further Analysis”, I exported the processed and combined weather data as a csv file to the **weather** sub-folder of the **proc_data** folder.

Question 1:

The codes for this section could be found in the “Part 1 Codes and Analysis.ipynb” notebook in the **python_notebooks** folder. There are two main sections in the notebook: 1) Exploratory Data Analysis and 2) Clustering.

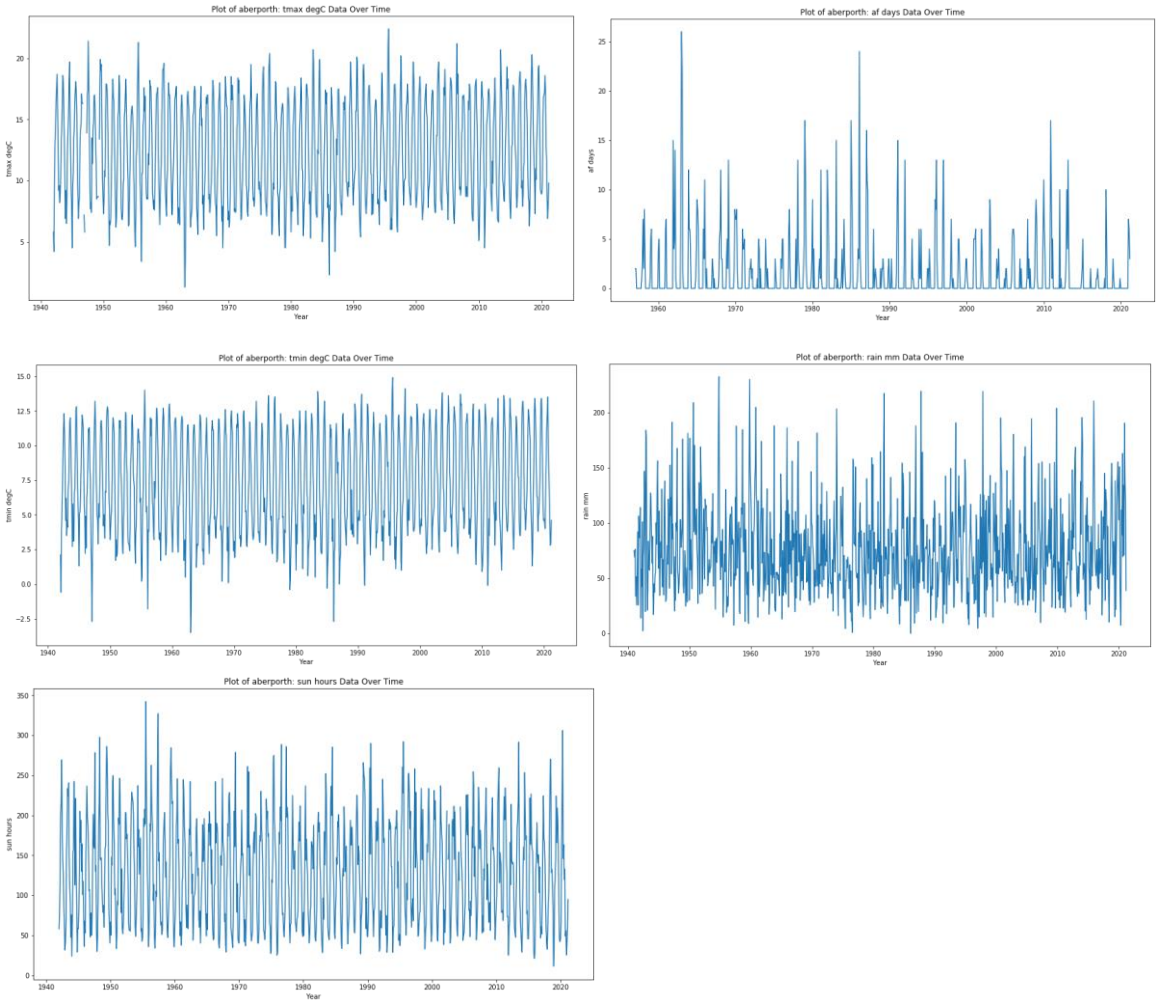
This question requires us to use a clustering algorithm to see if these stations can be clustered into groups that have “similar” weather without considering the latitude and longitude. The following details my approach to the question.

Exploratory Data Analysis

First, after importing the cleaned weather data from the **weather** sub-folder of the **proc_data** folder, I conducted a series of exploratory data analysis (please refer to the Exploratory Data Analysis section of the codes for details) to understand the empirical univariate distributions of each of the weather

variables: max temperature (tmax), minimum temperature (tmin), air frost days (af days), rainfall (rain) and sun hours.

The data comprises monthly weather data across multiple years for each station and one needs to determine at which level of granularity to analyse the data at. To answer this, I decided to first look at the time series plots for each station's weather data (please refer to the notebook for all graphs) to see if there are some temporal trends in the time series. I used matplotlib.pyplot plot function to do this and noted that the weather data don't appear to have much trends in general and there is seasonality in the data with occasional spikes. For instance, the weather data for Aberporth below illustrates this point:



Next, I checked the mean, standard deviation and coefficient of variation (CV) for each of these weather variables for each station. I do this by using the groupby method in pandas and taking mean, standard deviation and CV (i.e. standard deviation divided by mean) with the detailed results in the notebook. The main finding was that the CV was quite large for af days (being ~1.41) while others are quite contained below 1 shown in the table below. Under the assumption that there is not much of a time trend, this leads me to the factor in the seasonality component.

Coefficient of Variation	tmax degC	0.385022
	tmin degC	0.694023
	af days	1.413218
	rain mm	0.541046
	sun hours	0.522892

Another side note is that the mean of each station's weather variable has quite some variation (giving me some confidence that clustering might be fruitful). I also tried looking at the median but similar results as mean.

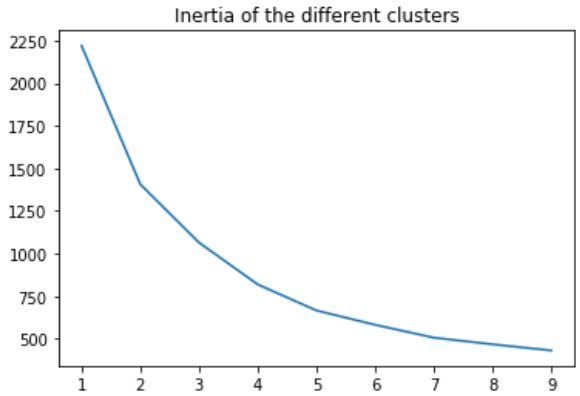
The next step I did was to look at the averages for each month for each station's weather data again using pandas groupby function on the station and month. The sizable differences in the weather data in the months corresponding to the 4 seasons are apparent (please refer to notebook).

To err on the safe side of caution, I decided to do two approaches, (A1) one to prepare the averages for each month for each station's weather data again using pandas groupby function on the station and month followed by pivoting the months for each weather data to become columns of the new dataset (for e.g. sun hours 1, sun hours 2 refer to the sun hours for Jan and Feb respectively) ; the other (A2) is to prepare the mean of each station's weather variable across all time, again by groupby followed by mean. Next, I will standardise the features to ensure that the different magnitudes and ranges for the weather features do not "unfairly" overweigh or under-weigh the importance of each feature. I used the StandardScaler function in sci-kit learn.

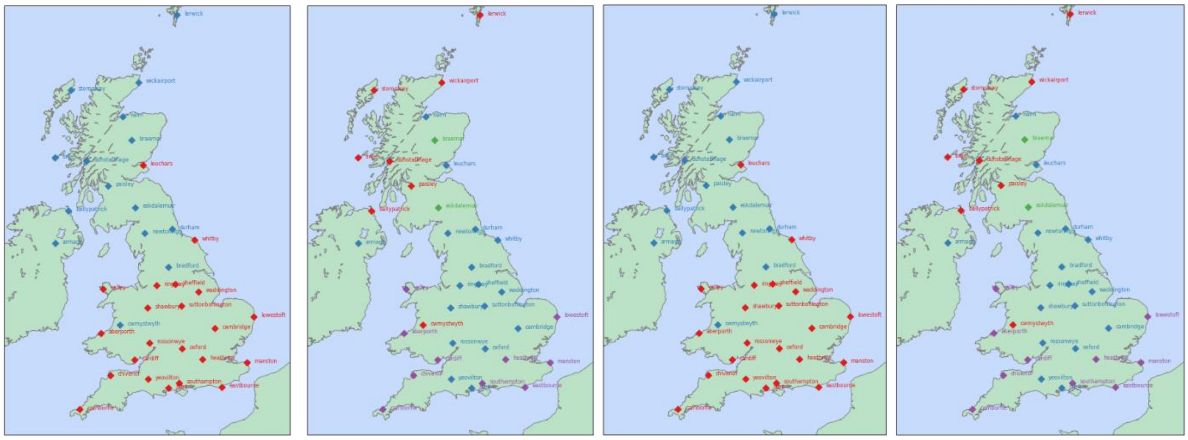
Clustering

I conducted 2 types of clustering algorithms: K-means and Gaussian Mixture Model (GMM). The reason for trying these two types is because K-means is a form of hard clustering algorithm based on distances while GMM is probabilistic in nature; and I wanted to have a competitor model kind of approach in this exercise. For the implementation, I used the relevant libraries in sci-kit learn and wrote two functions: kmeans_clustering and gmm_clustering to do the clustering up to a pre-defined number of K clusters and plots the relevant metrics for each clustering algorithm for me to discern which number of clusters to pick (please refer to the code for details on implementation). For K-means clustering, I plotted the inertia also known as within-cluster sum-of-squares criterion (which measures how much variation is there within the clusters) and used the elbow method to determine the "best" K to use. As for GMM, I used BIC to pick the number of clusters corresponding to the lowest BIC and silhouette score to pick the number of clusters corresponding to the highest score for evaluation.

For the K-means clustering on the data from A1 (approach 1), I think the elbow could be at either 2 or 4 clusters from the inertia plot below:



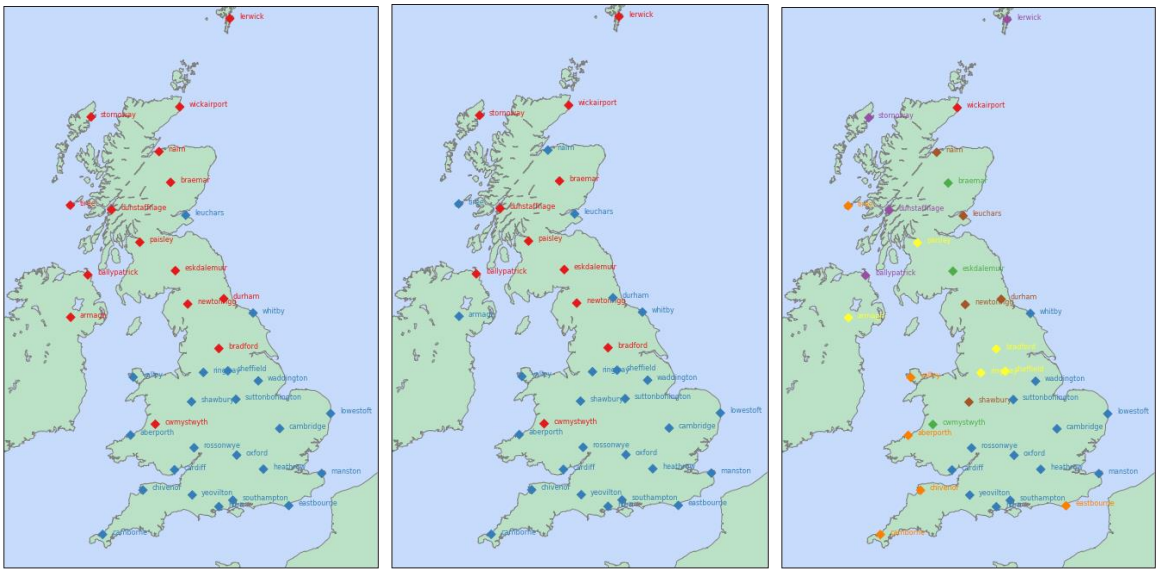
Next, I get the labels and plot them on a map using basemap. I created a user-defined function for this plotting called plot_ukmap. The results that follow provide an interpretable result for 2-clusters as it almost splits the stations into northern and southern UK, while the 4-clusters result appear to be having clusters roughly like (red- northern coastal UK, blue – inland UK, green- inland north UK, purple- southern coastal UK):



K-means 2-cluster results (A1) | K-means 4-cluster results (A1) | K-means 2-cluster results (A2) | K-means 4-cluster results (A2)

The notion I had was that the closer the stations, the more "similar" the weather should be and it appears the 2-cluster results are more interpretable. I repeated this with simpler approach (A2) and got the same results as shown above (interestingly the elbow method pointed me to use 2-clusters and 4-clusters too)

Next, I repeated all a similar exercise as the steps above on the two approaches but this time using the GMM clustering algorithm. The metrics plot for (A1) pointed me to 2-clusters while those in (A2) if based on the silhouette score, we still go with 2 clusters but with BIC, we will pick 7 clusters. The results of the clustering are as follows:



GMM 2-cluster results (A1) | GMM 2-cluster results (A2) | GMM 7-cluster results (A2)

The GMM results for 2-cluster (A1) is the same as those from K-means clustering but those for 2-cluster (A2) are a bit different. As for GMM 7-clusters, it is really difficult to interpret it and could potentially be too many clusters. In conclusion, I would still go with K-means 2-cluster under approach of A1 and the results are consistent with 2-cluster K-Means (A2) and 2-cluster GMM (A1).

Question 2:

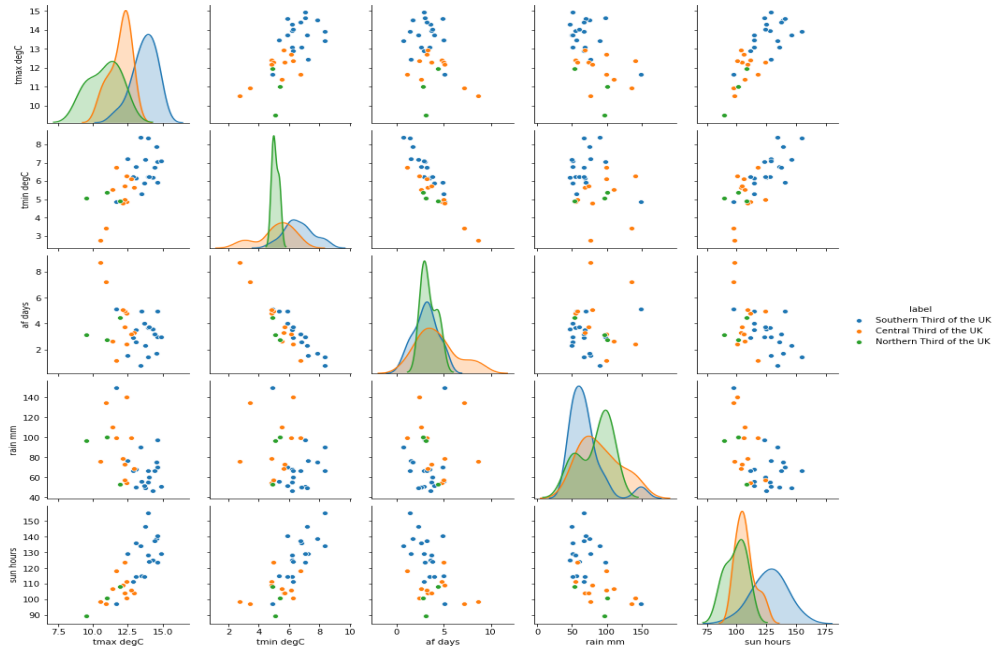
The codes for this section could be found in the "Part 2 Codes and Analysis.ipynb" notebook in the **python_notebooks** folder. These are the main sections in the notebook: 1) Pre-processing Data 2) Exploratory Data Analysis, 3) Feature Engineering, 4) Naïve Baye Classifier, 5) Decision Tree Classifier, 6) Random Forest Classifier, 7) Logistic Regression

I start out by importing the cleaned weather data from the **weather** sub-folder of the **proc_data** folder. According to the question, we need to first classify the stations based on latitude into Northern Third of the UK, Central Third of the UK or Southern Third of the UK. I wrote a function classify_lat in the "Pre-processing Data" section of the notebook which splits equally in the range of 49.9 and 60.9 and classify which label to assign to for a given latitude. Next, I aggregate the weather data by grouping by the stations and taking the mean (using the groupby function in pandas) so that the level of granularity is at station level only. This ignores the temporal effects which could complicate the analysis. Next, I segregated the last 5 stations (in alphabetical order) and keep them as test set. I further define training data set: X_train and y_train, and test data set: X_test and y_test.

In the EDA section, I explore the mean, standard deviation, coefficient of variation, minimum and maximum of weather data for each label. I do this by using the groupby function and grouped by labels while applying the relevant metrics (please refer to notebook for detailed results).

Some observations are as follows: It seems like there is quite a bit of distinction between weather data for each of the labels in terms of the mean. The coefficient of variation indicates that within each of the labels the standard deviation of the weather data relative to the mean is still quite contained. From the mean for each labels, it seems that Northern Third of UK tends to be the coldest, followed by Central Third of UK and Southern Third of UK. Generally, the number of AF days is highest in Central Third of UK, and relatively similar number of AF days for Northern Third of UK and Southern Third of UK. More rain in Central Third of UK followed by Northern Third of UK and Southern Third of UK. Most sun hours in Southern Third of UK followed by Central Third of UK and Northern Third of UK.

As part of the EDA, I also plotted the pairplots for each label. I did this using the pairplot function in seaborn library in python and the visualisation is as follows:



From the pair plots, there appears to be some linear relationship between some of the variables:

- tmax and tmin (positive)
- tmax and rain (negative)
- tmax and sun hours (positive)
- tmin and af days (negative)
- tmin and rain (positive)
- tmin and sun hours (positive)
- af days and sun hours (negative)
- rain and sun hours (negative)

There appears to be some distinction between the values under the different labels with some differences in distributions. However, it seems like some transformation might be required to allow for a better classifier.

In the "Feature Engineering" section, I decided to use two sets of features for modelling: original set of features and the principal components (PCs) of the features as another set of features to consider. To derive the number of PCs to use, I look at the explained variance ratio for each of the PCs (please refer to codes for details) and noted that percentage variance explained by first 2 PCs is 99.7%. Based on this, I decided to use the first 2 PCs as another set of features.

For the next few sections on the modelling (except of Gaussian Naïve Bayes), I used hyperparameter tuning on the training set to get the best set of hyperparameters for the models: Decision Tree Classifier, Random Forest Classifier and Logistic Regression via 5-fold cross-validation using the accuracy as the score function. The aforementioned hyperparameter tuning is done by using the sci-kit learn function "GridSearchCV" and since there are 3 classes in the labels, I used the "OneVsRestClassifier" function (for details, please refer to "Decision Tree Classifier", "Random Forest Classifier", "Logistic Regression" sections for the respective user-defined functions I wrote to implement hyperparameter tuning). Subsequently, I used the "best" set of hyperparameters and re-fit it to the entire training set and evaluated the training accuracy and test accuracy. For Naïve Bayes, I just fit directly to the entire training set and evaluated the training accuracy and test accuracy as there wasn't really hyperparameters that needed to be tuned.

I consolidated the results as follows (for comparison of predicted labels and actual labels for each model, please refer directly to the respective sections of the notebook):

Model	Set of Features	Training Accuracy	Test Accuracy
Gaussian Naive Bayes	Original	0.90625	0.8
	First 2 PCs	0.75	0.6
Decision Tree	Original	0.96875	0.8
	First 2 PCs	1	0.6
Random Forest	Original	1	0.6
	First 2 PCs	0.84375	0.6
Logistic Regression	Original	0.84375	0.8
	First 2 PCs	0.8125	0.6

From this table, it seems that using the original set of features tend to give better test accuracy for a given model likely due to nonlinear relationships with y that are not accounted for. Furthermore, a more complicated model like random forest tends to perform worse in test accuracy likely due to the small samples we have for model training. The simpler models like Naïve Bayes, Decision Tree and

Logistic Regression performs equally well (using original features) based on their test accuracy. I would be more inclined towards using Gaussian Naïve Bayes as the final model as decision tree's training accuracy is very high compared to test accuracy leading me to be suspicious if there is overfitting to the training set. As for logistic regression, there could be potential risk of underfitting to the data. However, the sample size is too small for us to further analyse and make conclusions using the learning curves.

To answer the question, the predictions for Naïve Bayes on the last 5 stations would be:

	station	predictions	actual
0	valley	Southern Third of the UK	Southern Third of the UK
1	waddington	Southern Third of the UK	Southern Third of the UK
2	whitby	Southern Third of the UK	Central Third of the UK
3	wickairport	Northern Third of the UK	Northern Third of the UK
4	yeovilton	Southern Third of the UK	Southern Third of the UK

Question 3:

The codes for this section could be found in the "Part 3 Codes and Analysis.ipynb" notebook in the **python_notebooks** folder. The two main sections in the notebook: 1) Pre-processing Happiness Data, 2) Exploratory Data Analysis and Modelling.

In this question, we are supposed to answer whether weather affects how happy we are. To do so, I will join the weather station data with the happiness data in a coarse-grained manner.

Pre-processing Happiness Data

In the "Pre-processing Happiness Data" section, I wrote a function called "extract_happiness" which takes in a file name and reads this file name from the **wellbeing** sub-folder of the **raw_data** folder. The output is a pandas dataframe object which is a cleaned happiness dataset extracted from the file. The logic for pre-processing in the "extract_happiness" function is as follows: I read in the sheet labelled "Happiness" and drop the empty rows (all NaN). Next, I extract the year by using regular expression to locate the first occurrence of the year after "April". Next, I get the last column index by locating the "Average rating" column using keyword search of "Average". Then I get the index of the last header row by finding the index with "Codes" in the first column and do a forward fill. Next, I get the index of the first row of the "appendix" and I will determine the row above to be the last row to cut off. Lastly, I do the relevant index slicing to get the data of interest and add in the year into a new column called "year". Please refer to the codes for details on implementation.

After defining this "extract_happiness" function, all I needed to do is to read in all the wellbeing files present in the **wellbeing** sub-folder of the **raw_data** folder; appending them to a list and concatenating them into a pandas dataframe. Next, I load in the regions.txt file from the **supp_data** folder and do an inner join of the "Area names" to the happiness data. This is to put in the latitude and longitude of each area that was initially not present in the happiness data. This set of cleaned happiness data is exported to the **happiness** sub-folder of the **proc_data** folder for further analysis in question 4.

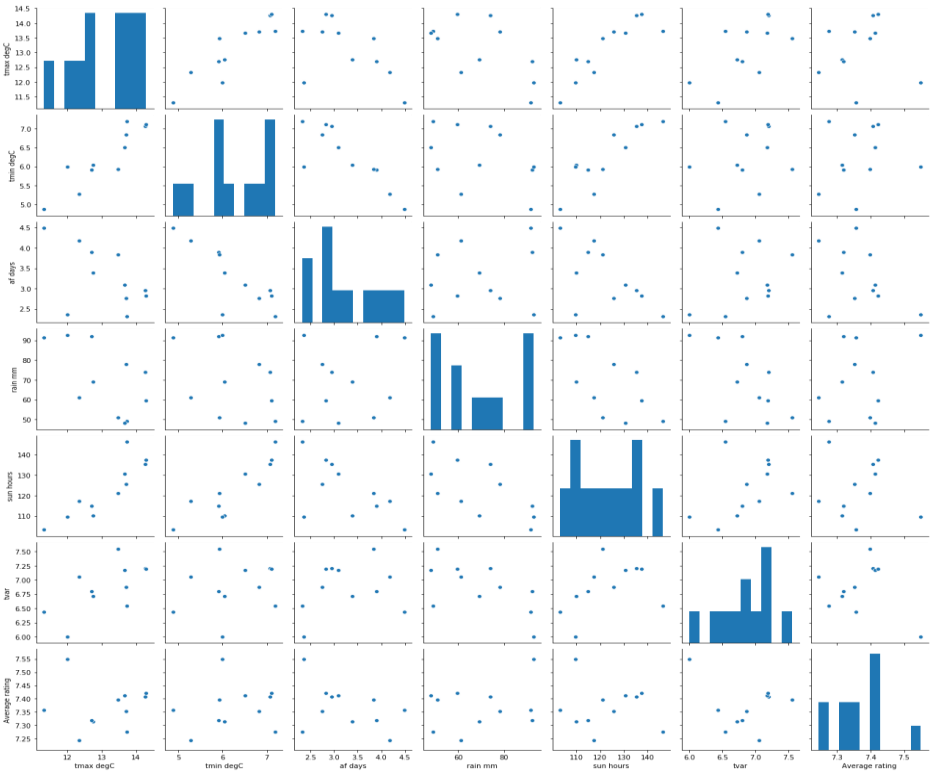
After getting this set of data ready, I would use the assumption that to a first approximation the general climate at a given weather stations is fairly constant over time and that happiness also does not change too much from year to year. This means that I would just need to consider the average happiness ratings across all years for each area and the average weather conditions across all years for each station. I use the groupby function and take the mean to do this. Next, I loaded cleaned

weather data from **weather** sub-folder in the **proc_data** folder. I grouped by station and take the mean weather conditions for each of the features across the years.

The next step is to join the stations' weather data with the happiness data. I did this by assigning the nearest region/area to the station based on the latitude and longitude and merge to weather data. I wrote a function "compute_distance" using a haversine formula and this is adapted from the source: https://stackoverflow.com/questions/19412462/getting-distance-between-two-points-based-on-latitude-longitude. I wrote another function called "assign_nearest_region_merge" to assign nearest region/area to the station using the distance computed by the "compute_distance" function and subsequently merge to weather data. Please refer to the notebook for the details of the dataframe after the assignment.

Exploratory Data Analysis and Modelling

to answer whether weather affects how happy we are, I first conducted EDA to understand relationships between the variables. But even before that, I would need to make a decision of whether I should explore this relationship at the level of granularity of station or region/area. I decided to group by the region and take the mean for each region and use it as the most granular level. The reason for this is because the happiness data is the same for stations within the same region and I think it appropriate for me to analyse at that level for the weather data. After preparing the data, I plotted pair plot (using seaborn library) and the following shows the results:



Focussing on the last row of charts: It does seem like higher max-temperature tends to be associated with higher average happiness index. Furthermore, a temperature variation seems to have a positive association with average happiness index. It does appear that the data point from Northern Ireland seems like a likely candidate for outlier which should be checked.

Next, I computed the correlation matrix for the data (please refer to the notebook for correlation matrix details). I have kept the mean percentage of votes within different happiness rating ranges just

to see if there are any discernible patterns and correlations: it appears that there seems to be higher percentage of votes in 0-4 and 9-10 rating ranges in regions where tmax and tmin are lower. This seems to reflect strong opinions in happiness in such regions with lower tmax and tmin. Similar case as observed for sun hours where people tend to have strong opinions in regions where sun hours is low. For af days, there is a positive correlation with percentage of votes in 0-4, 5-6 ranges and a negative correlation with percentage of votes in 7-8 and 9-10 ranges reflecting lower happiness ratings when af days go up. Rain on the other hand, doesn't seem to give very logical interpretation with positive correlation with the percentages of votes in 9-10 ranges and negative ones with the other ranges. Looking just at the weather variables with the average happiness ratings, I got the following results:

	Weather Variables	Correlation
0	tmax degC	0.034473
1	tmin degC	0.152970
2	af days	-0.423255
3	rain mm	0.266494
4	sun hours	-0.101754
5	tvar	-0.192671

Positive relationship between happiness and variables: tmax, tmin, rain; and negative relationship between af days, sun hours and temperature variation (note that this is because of the point on the top left that affected the relationship).

Clearly, such bivariate comparisons are not adequate as the weather variables might have some correlation between them and it would better to look into multiple linear regression for such partial effects and "control" for effects of other explanatory variables on the dependent variable.

Before fitting the multiple linear regression model, I conducted outlier tests on both happiness data and the weather variables. For happiness data, I conducted the one-sided Grubbs test to check for outliers for y at alpha = 0.05 both sides. For weather variables, I checked if any of the points fall below 1.5 * Inter-Quartile Range from 1st quartile or above 1.5 * Inter-Quartile Range from 3rd quartile. It seems that all data points passed these tests. Next, I normalised the weather data.

I decided to use backward elimination method and wrote a function for it named "backward_elim" (please refer to it for details). Using the normalised weather features (all excluding temperature variation which is tmax- tmin) and conducting backward elimination on it using a significance level threshold of 0.1, I got the following model:

OLS Regression Results

Dep. Variable:	Average rating	R-squared:	0.706
Model:	OLS	Adj. R-squared:	0.510
Method:	Least Squares	F-statistic:	3.602
Date:	Fri, 23 Apr 2021	Prob (F-statistic):	0.0792
Time:	20:25:48	Log-Likelihood:	18.959
No. Observations:	11	AIC:	-27.92
Df Residuals:	6	BIC:	-25.93
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	8.5531	0.438	19.549	0.000	7.482	9.624
tmax degC	36.6556	13.008	2.818	0.030	4.825	68.486
tmin degC	-63.7935	26.750	-2.385	0.054	-129.248	1.661
af days	-37.9642	12.475	-3.043	0.023	-68.489	-7.440
sun hours	-1.0367	0.453	-2.288	0.062	-2.145	0.072

Omnibus:	0.571	Durbin-Watson:	1.299
Prob(Omnibus):	0.752	Jarque-Bera (JB):	0.571
Skew:	-0.257	Prob(JB):	0.752
Kurtosis:	2.010	Cond. No.	2.39e+03

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.39e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

From the above regression model summary output, I conclude that a higher max temperature (tmax) has a positive relationship with happiness index and min temperature (tmin) appears to have a negative relationship with happiness index. This indirectly is reflected by the observed association between larger temperature variation and higher happiness index in the pair plots. Furthermore, lower af (air frost) days is associated with higher happiness index which makes sense as places with many af days would mean very cold and harsh weather conditions which might take a toll on happiness. Interestingly, the number of sun hours is also negatively related with happiness index but the magnitude of the coefficient is much less. It is also worth noting that if one were to use a more stringent criteria of 0.05 significance level as a cut-off, sun hours and tmin would be deemed as statistically insignificant.

I repeated similar steps using with temperature variation in place of max and min temperatures (i.e. normalisation before backward elimination model fitting):

OLS Regression Results

Dep. Variable:	Average rating	R-squared:	0.748
Model:	OLS	Adj. R-squared:	0.579
Method:	Least Squares	F-statistic:	4.443
Date:	Fri, 23 Apr 2021	Prob (F-statistic):	0.0521
Time:	20:25:48	Log-Likelihood:	19.799
No. Observations:	11	AIC:	-29.60
Df Residuals:	6	BIC:	-27.61
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	13.5510	2.636	5.140	0.002	7.100	20.002
af days	-24.5730	6.213	-3.955	0.007	-39.777	-9.369
rain mm	-2.9001	1.356	-2.138	0.076	-6.219	0.419
sun hours	-6.5437	2.518	-2.599	0.041	-12.705	-0.382
tvar	29.7433	9.169	3.244	0.018	7.306	52.180

Omnibus:	1.594	Durbin-Watson:	1.203
Prob(Omnibus):	0.451	Jarque-Bera (JB):	0.778
Skew:	0.010	Prob(JB):	0.678
Kurtosis:	1.698	Cond. No.	952.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Using an alternative formulation by considering temperature variation instead of tmax and tmin. I obtained the above output from the regression model. All variables are statistically significant at 0.05 significance level and the following are some conclusions that could be drawn on the relationships between weather and happiness:

- Fewer af days are associated with higher happiness
- Lower rainfall is associated with higher happiness
- Fewer sun hours are associated with higher happiness
- Larger temperature variation is associated with higher happiness.

I would prefer the second model over the first due to the higher adjusted R-squared of 0.579 compared with 0.51 in the first. However, even for the 2nd model, the probability for F-test is 0.0521 which leads us to not reject the null hypothesis that model with no independent variables fits the data just as well at 0.05 significance level; albeit borderline. Hence, it does seem to suggest that the effect of weather on happiness probably exists but is not really statistically significant at 0.05 level of significance.

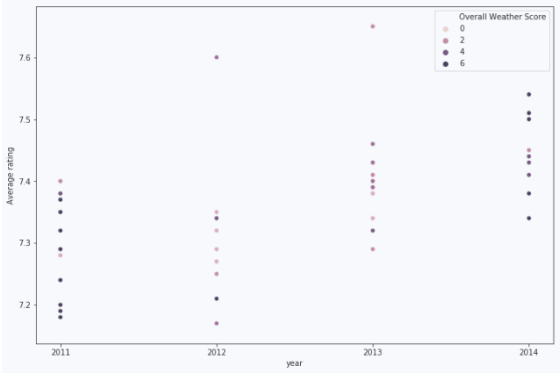
Question 4a:

The codes for this section could be found in the "Part 4 Codes and Analysis.ipynb" notebook in the **python_notebooks** folder. These are the main sections in the notebook: 1) Look for any temporal effects that we neglected earlier: Are people happier in years when the weather is better? 2) Investigation of whether better weather results in higher happiness index after accounting for the temporal effects and the regional effects

This question allows us to explore the data in any way that we want and I chose to focus on the first suggested question with additional analyses.

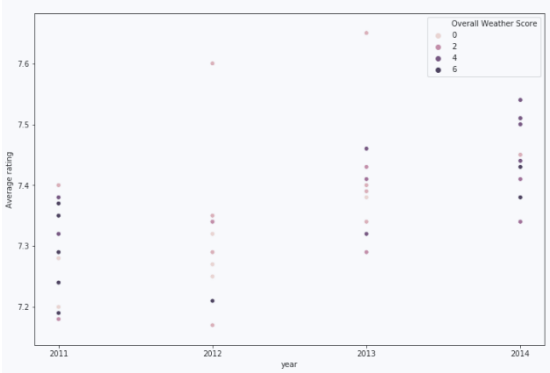
To answer this question of whether people are happier in years when the weather is better, I look at happiness data and the weather data from 2011 to 2014. I would first need to define what is a better weather and chose to define better weather as less rain, more sun hours, less af days, warmer tmax, warmer tmin. Using the mean as the threshold, I would "score" each of these variables as 1 if it falls on the "better" side and 0 otherwise. The score will range from 0 to 5 with higher score indicating better weather. I would use data visualisation to inspect the patterns in the data then based on summary statistics and gather conclusions (please refer to the notebook for details of the implementation). The pre-processing steps largely involves similar steps as in question 3, by aggregating weather data to the station and year level and happiness data to the region/area and year level before doing a join based on the assigned nearest region (a similar function called “assign_nearest_region”). In this question, I kept the level of granularity at the station and year level first to check on the data visualisation and summary statistics for comparison with the version where level of granularity is at the region and year level to see if there are any differences in conclusion. Results are as follows:

Results of Average Rating against Years with Overall Weather Score (Granularity Level at Station)



	Overall Weather Score		Average rating	
	mean	std	mean	std
year				
2011	3.400000	1.914854	7.295600	0.070064
2012	1.875000	1.569062	7.327083	0.095666
2013	2.166667	1.372610	7.412500	0.083835
2014	3.454545	1.503243	7.460000	0.052554

Results of Average Rating against Years with Overall Weather Score (Granularity Level at Region)



	Overall Weather Score		Average rating	
	mean	std	mean	std
year				
2011	3.272727	2.101947	7.290909	0.079932
2012	1.636364	1.689540	7.316364	0.111020
2013	1.818182	1.328020	7.407273	0.094455
2014	3.800000	1.229273	7.451000	0.064023

From the summary tables above, it does appear that in years when weather is better, the average happiness ratings are not really affected by it. For instance, 2011 and 2014 which saw better weather conditions in general saw a mixed result in average happiness rating. Happiness rating in 2011 was the lowest among all years, and 2014 was higher among all 4 years.

The natural question would then be whether better weather results in higher happiness index after accounting for the temporal effects and the regional effects. To answer this, I modelled this using the random effects model of panel regression. I initially wanted to do two-factor ANOVA but realise that

it would not be able to provide any guidance on the direction of the relationships. For the random effects model, I used “RandomEffects” function from “linearmodels.panel” library. The results are as follows:

```
                        RandomEffects Estimation Summary
================================================================================
Dep. Variable:         Average rating   R-squared:                        0.6986
Estimator:                RandomEffects   R-squared (Between):             -0.1742
No. Observations:                  43   R-squared (Within):               0.1114
Date:                Sun, Apr 25 2021   R-squared (Overall):             -0.0377
Time:                        19:14:29   Log-likelihood                    46.807
Cov. Estimator:            Unadjusted
                                        F-statistic:                      22.016
Entities:                          11   P-value                           0.0000
Avg Obs:                       3.9091   Distribution:                    F(4,38)
Min Obs:                       3.0000
Max Obs:                       4.0000   F-statistic (robust):             0.5938
                                        P-value                           0.6692
Time periods:                       4   Distribution:                    F(4,38)
Avg Obs:                       10.750
Min Obs:                       10.000
Max Obs:                       11.000

                             Parameter Estimates
================================================================================================
                        Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------------------------
const                      7.2924     0.0607     120.06     0.0000      7.1694      7.4153
Overall Weather Score      0.0130     0.0099     1.3205     0.1946     -0.0069      0.0330
year.2012                  0.0549     0.0647     0.8480     0.4018     -0.0761      0.1858
year.2013                  0.0489     0.0696     0.7028     0.4865     -0.0919      0.1897
year.2014                  0.0639     0.0748     0.8537     0.3986     -0.0876      0.2154
================================================================================================
```

Looking at the p-value of the overall weather score, at 0.05 level of significance, it is not statistically significant. However, it provides some indication that the weather score has a positive relationship with the happiness rating albeit. On average, 1 score higher in the overall weather score (better weather) is associated with a 0.013 increase in the average happiness rating after “controlling” for region-specific and temporal effects. Notably, the year dummy variables all have positive coefficients and increasing in magnitude (except 2013) and this suggests that for a given region, the happiness rating tends to be higher from base year of 2011 and appears to be increasing albeit very minutely.

References

1. Hastie, T., Tibshirani, R., & Friedman, J. (2021). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)* (2nd ed.). Springer.