Practical Introduction to Data Science – Assessment 3

Brian Mukeswe

University of Edinburgh School of Informatics

CONTENTS

INTRODUCTION

This report summarizes the analyses performed as part of Assessment 3. The report includes a description of the steps taken, with explanations for the different choices that were made when processing and analyzing the data. Additionally, the report summarizes the results and conclusions that were obtained from the analysis.

DATA PROCESSING – DATASET 1

Dataset 1 consists of monthly weather data from 37 weather stations in the UK with several weather attributes recorded at each station. Table 1 shows the weather attributes included in the data set.

Table 1: Summary of Weather Attributes Contained in Dataset 1.

Feature	Description	Type	Units	Range	Mean
"sun_hours"	The number of sunshine hours recorded at a station within a given month.	numerical	hours	2.8 to 350.3	118.4
"af_days"	The number of frost days recorded at a station within a given month	numerical	days	0 to 31	3.5
"rain_mm"	The quantity of rainfall in millimeters (mm) recorded at a given station within a given month.	numerical	millimeters	0 to 568.8	72.8
"tmin_degC"	The minimum temperature in degrees centigrade that is recorded at a station during a given month.	numerical	Degrees Centigrade	-8.6 to 17	5.9
"tmax_degC":	The maximum temperature in degrees centigrade that is recorded at a station during a given month.	numerical	Degrees Centigrade	-0.9 to 28.3	12.7

Downloading the data

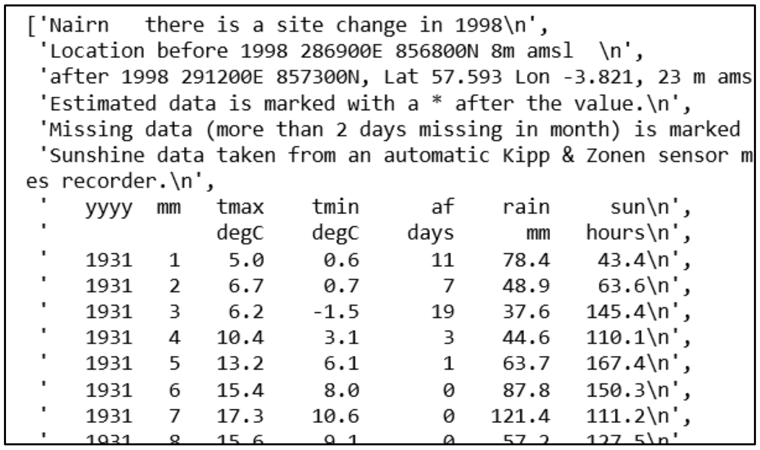
- I used a Python script to download all the weather data into text files, because automating the file download with a script saved time. The script generated a log so that I could easily determine if any file downloads were unsuccessful.
- I used Python's "requests" library to retrieve bytes of data from each URL, and then stored the data into a text file (.txt).
- The Python code used to download can be found in following notebook:
  - "*download_weather_data.ipynb*"

Cleaning the data

I used the Python code in the notebook "*store_weather_data.ipynb*" to perform the data cleaning and standardization steps described below:

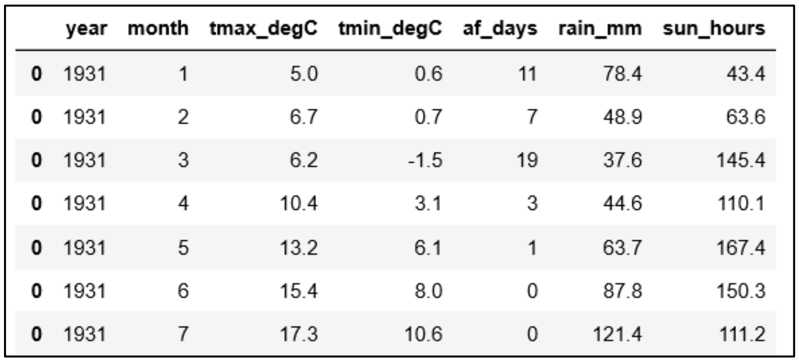
- Stripped off headers, extracted weather data from all text files and merged all the weather data into a pandas data frame. I loaded all the data into a single data frame so that I could uniformly apply the same cleaning and standardization actions to the data.
- Removed non-numeric characters from the data including “*”, “#”, ”$”, “---“ and other text characters that were mixed among numeric entries. I used 1000 as a numeric placeholder for missing values instead of the string “---“. 1000 was used because it is far outside the range of any valid weather data entries.
- All data entries were initially stored as strings, so I used the pandas “astype” method to cast all data attributes to numerical types.
- Figure 1 and
- Figure 2 show a few rows of the data before and after performing the cleaning steps described above.

Figure 1: Sample of Lines Read from the Downloaded TXT file Containing Weather Data



```
['Nairn   there is a site change in 1998\n',
 'Location before 1998 286900E 856800N 8m amsl  \n',
 'after 1998 291200E 857300N, Lat 57.593 Lon -3.821, 23 m ams
 'Estimated data is marked with a * after the value.\n',
 'Missing data (more than 2 days missing in month) is marked
 'Sunshine data taken from an automatic Kipp & Zonen sensor m
es recorder.\n',
 '   yyyy  mm   tmax    tmin      af    rain     sun\n',
 '              degC    degC    days      mm   hours\n',
 '   1931   1    5.0     0.6      11    78.4    43.4\n',
 '   1931   2    6.7     0.7       7    48.9    63.6\n',
 '   1931   3    6.2    -1.5      19    37.6   145.4\n',
 '   1931   4   10.4     3.1       3    44.6   110.1\n',
 '   1931   5   13.2     6.1       1    63.7   167.4\n',
 '   1931   6   15.4     8.0       0    87.8   150.3\n',
 '   1931   7   17.3    10.6       0   121.4   111.2\n',
 '   1931   8   15.6     9.1       0    57.2   127.5\n'
```

Figure 2: Data Frame Showing a Few Rows of Cleaned Weather Data



	year	month	tmax_degC	tmin_degC	af_days	rain_mm	sun_hours
0	1931	1	5.0	0.6	11	78.4	43.4
0	1931	2	6.7	0.7	7	48.9	63.6
0	1931	3	6.2	-1.5	19	37.6	145.4
0	1931	4	10.4	3.1	3	44.6	110.1
0	1931	5	13.2	6.1	1	63.7	167.4
0	1931	6	15.4	8.0	0	87.8	150.3
0	1931	7	17.3	10.6	0	121.4	111.2

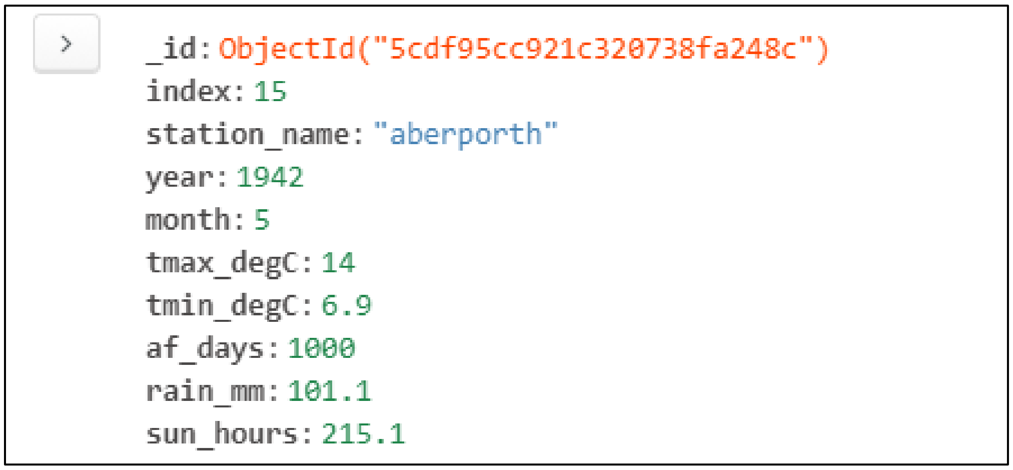
Storing the data

The notebook “*store_weather_data.ipynb*” also contains code that was used to store all the weather data in a Mongodb database.

- I decided to store the cleaned data in a database so that I could easily access specific parts of the data without having to load the full data set into memory. Using a database also meant that the cleaning steps did not have to be repeatedly executed whenever I restarted my computer.

- Moreover, Mogodb is opensource and Mongodb queries are easily integrated into Python syntax. Additionally, Mongodb supports joins/aggregations of different datasets, a feature that is necessary in part 3 of this assignment.
- I stored each row of weather data as a single document, such as the one shown in Figure 3, within the a collection, thereby creating the flexibility to filter the data using every single attribute i.e. (year, month, sun_hours, af_days, rain_mm, tmax_degC, tmin_degC).

Figure 3: Sample Mogodb Document Containing a Single Line of Weather Data



```
_id: ObjectId("5cdf95cc921c320738fa248c")
index: 15
station_name: "aberporth"
year: 1942
month: 5
tmax_degC: 14
tmin_degC: 6.9
af_days: 1000
rain_mm: 101.1
sun_hours: 215.1
```

DATA PROCESSING – DATASET 2

Dataset 2 consists of survey data about how people from 12 regions in the UK rate their own level happiness, with each region represented by a unique area code. Happiness was rated on a scale of 0 to 10 and the results were categorized as shown in Table 2. For each region, the percentage of the population that falls within a given category was provided. Additionally, the coordinates (i.e. latitudes and longitudes) representing the center of each region were provided.

Table 2 Happiness Level Categorization:

Happiness Rating	Category
0 to 4	Low
5 to 6	Medium
7 to 8	High
9 to 10	Very High

Downloading the data

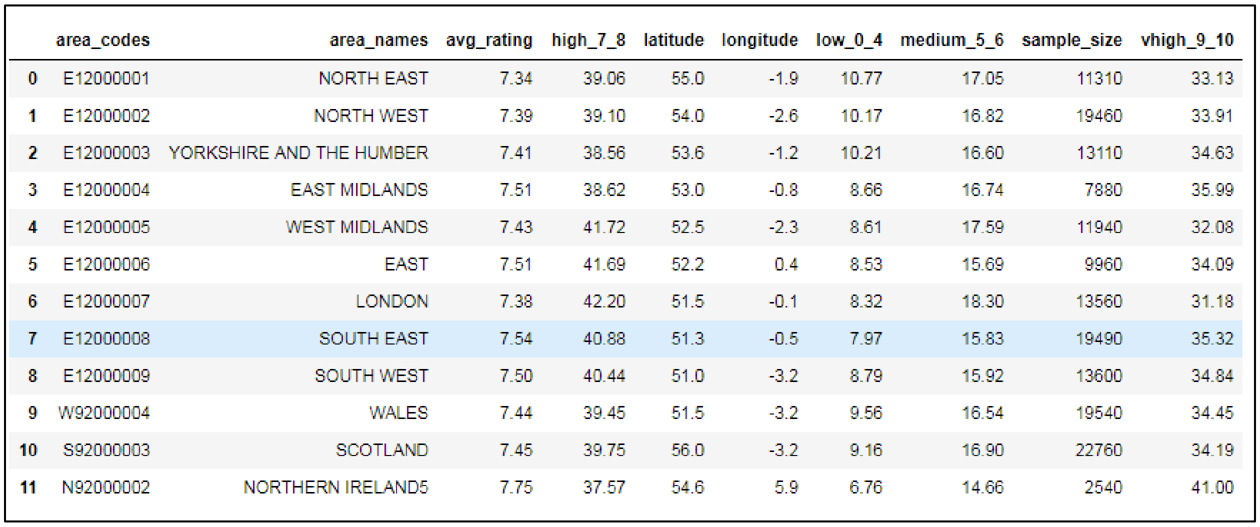
I downloaded the most recently available data (2014 to 2015) as spreadsheet that I stored on my computer's file system. ("*geographicbreakdownreferencetable_tcm77-417203.xls*")

Cleaning the data

The relevant data was embedded within a larger dataset contained in the “happiness” tab of the spreadsheet. In order to extract the relevant data, I performed the following steps using the code in the notebook “*read_happiness_data.ipynb*”:

- I loaded the “Happiness” tab of the spreadsheet into a data frame and dropped all the extra sections of the dataset (e.g. confidence intervals, counties etc.), keeping only the data shown in Figure 4. I loaded the area codes and coordinates corresponding to the relevant regions from the provided text file “regions.txt” into a Pandas data frame, and used the area codes to identify and merge the relevant happiness data into a single data frame shown in Figure 4.

Figure 4: Data Frame Showing Cleaned Happiness Data for Each Area

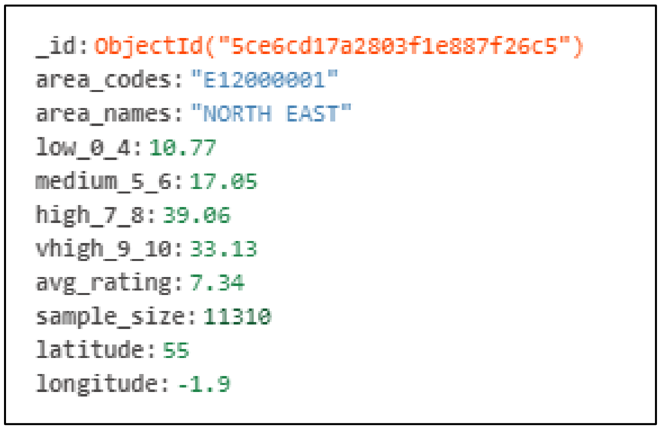


	area_codes	area_names	avg_rating	high_7_8	latitude	longitude	low_0_4	medium_5_6	sample_size	vhigh_9_10
0	E12000001	NORTH EAST	7.34	39.06	55.0	-1.9	10.77	17.05	11310	33.13
1	E12000002	NORTH WEST	7.39	39.10	54.0	-2.6	10.17	16.82	19460	33.91
2	E12000003	YORKSHIRE AND THE HUMBER	7.41	38.56	53.6	-1.2	10.21	16.60	13110	34.63
3	E12000004	EAST MIDLANDS	7.51	38.62	53.0	-0.8	8.66	16.74	7880	35.99
4	E12000005	WEST MIDLANDS	7.43	41.72	52.5	-2.3	8.61	17.59	11940	32.08
5	E12000006	EAST	7.51	41.69	52.2	0.4	8.53	15.69	9960	34.09
6	E12000007	LONDON	7.38	42.20	51.5	-0.1	8.32	18.30	13560	31.18
7	E12000008	SOUTH EAST	7.54	40.88	51.3	-0.5	7.97	15.83	19490	35.32
8	E12000009	SOUTH WEST	7.50	40.44	51.0	-3.2	8.79	15.92	13600	34.84
9	W92000004	WALES	7.44	39.45	51.5	-3.2	9.56	16.54	19540	34.45
10	S92000003	SCOTLAND	7.45	39.75	56.0	-3.2	9.16	16.90	22760	34.19
11	N92000002	NORTHERN IRELAND5	7.75	37.57	54.6	5.9	6.76	14.66	2540	41.00

Storing the data

I stored the happiness data in a different collection within the same Mongodb database as the weather data. Co-locating the datasets in the same database facilitated aggregation of the datasets in part 3. Each row of the data shown in Figure 4 was stored a document in the collection.

Figure 5: Sample Mongodb Document Containing a Single Line of Happiness Data



```
_id: ObjectId("5ce6cd17a2803f1e887f26c5")
area_codes: "E12000001"
area_names: "NORTH EAST"
low_0_4: 10.77
medium_5_6: 17.05
high_7_8: 39.06
vhigh_9_10: 33.13
avg_rating: 7.34
sample_size: 11310
latitude: 55
longitude: -1.9
```

PART 1 - CLUSTERING

The code used to perform clustering and the necessary pre-processing steps on the weather data is in the notebook: *"explore_weather_data.ipynb"*

Part 1 - Approach

I chose to use the most recent data. However, 2019 data was considered to be only provisional and only included 4 months of weather data. As a result, I used 2018 data instead. Using only 2018 data is deemed to be sufficient under the initial assumption that the climate at a given station is fairly constant over time.

I noticed that 10 stations did not report any sun_hours data for 2018. For each station that was missing sun_hours data, I filled in the missing data by averaging historical data. I averaged the values from the most recently available data from three fill years. Again, this is assuming that the average number of sun hours recorded at a given station in a given month is stable over time. Filling in the missing sun_hours data was accomplished using three functions:

- *get_fill_years(station, years_to_check)*: identifies the three most recent years that recorded sun hours data at a given station.
- *get_fill_values(station, fill_years)*: calculates the average of sun hours for each month over a specified number set of fill years.
- *fill_sun_hours()*: fills in missing sun hours data for 2018 using historical average data

The fill years used for each of the stations with missing 2018 sun_hours data are listed in Table 3.

Table 3: Years Used to Calculate Average "sun_hours" Values to Fill in Missing 2018 Data

Station	**Most Recent Years with sun_hours Data ("Fill Years")**		
	Year 1	**Year 2**	**Year 3**
Ballypatrick	1990	1989	1988
Braemar	2004	2003	2002
Cambridge	2009	2008	2007
Cardiff	1995	1994	1993
Chivenor	1970	1968	1967
Dunstaffnage	2001	2000	1999
Durham	1998	1997	1996
Newtonrigg	1980	1979	1978
Suttonbonington	1999	1998	1997
Wickairport	1992	1991	1990

I also noticed that three stations did not report any weather data for 2018 because they were already closed i.e. cwmystwyth, ringway, Southampton. I decided to drop the three stations out of the analysis instead of filling them in with averaged historical data for simplicity, and assuming that adding the stations using averaged historical data will not change the analysis results materially.

For the purposes of clustering I decided to reduce the 2018 data set by taking averages over a specific season and also over the whole year for each station. I wanted to see if seasonal weather patterns were better than annual weather patterns for identifying distinct clusters of stations with similar weather. This was accomplished mainly using the following functions:

- *get_season_average (months)*: Reduces 2018 weather data by calculating the average of each weather attribute over a specified set of months. This function can also be used to calculate the annual average weather data by specifying all 12 months in the input parameter.
- *get_months_clusters (num_clusters, months)*: Performs KMeans clustering using reduced 2018 weather data, whereby the weather data is reduced by taking averages only across the specified set of months.

Additionally, I visualized 2018 weather data, prior to clustering, using pair plots from Python's seaborn library as shown in Figure 6 and Figure 7, before and after reducing the data respectively. Some observed patterns include:

- In Figure 6, the seasonal variations in each weather attribute is evident when plotted against the months. For example, is can be seen that the summer months generally have higher temperature recordings than the winter months, and the winter months have a higher number of frost days compared to the summer months.
- In both Figure 6 and Figure 7, strong correlations between the minimum and maximum temperatures are evident. This is not surprising since one would generally expect the stations that record lower minimum temperatures to also record lower maximum temperatures because both minimum and maximum temperature recordings depend on the same confounding variable i.e. the temperature at the station; Stations that are located in an area with a hot climate (with generally high average temperatures) tend to have higher maximum temperature and higher minimum temperatures.
- A strong correlation can also be observed between the number of frost days and the minimum temperature in both Figure 6 and Figure 7. This is consistent with the notion that very low temperatures are associated with higher frost levels and, of course, lower minimum temperatures.
- The reduced data in Figure 7, shows two imbalanced clusters that are mainly driven by the distribution of sun_hours values. The larger cluster consists of stations that receive on average more than 100 sun hours a month, and the smaller cluster consists of stations that receive on average less than 50 sun hours a month.
- The distributions of all the weather attributes approximate a normal distribution in the raw 2018 data in Figure 6, with the distributions of af_days, rain_mm and sun_hours being skewed normal distributions. However, the distribution of sun_hours in the reduced data (i.e. Figure 7) does not appear to be a normal distribution.

Figure 6: Pair Plot of Raw 2018 Weather Data

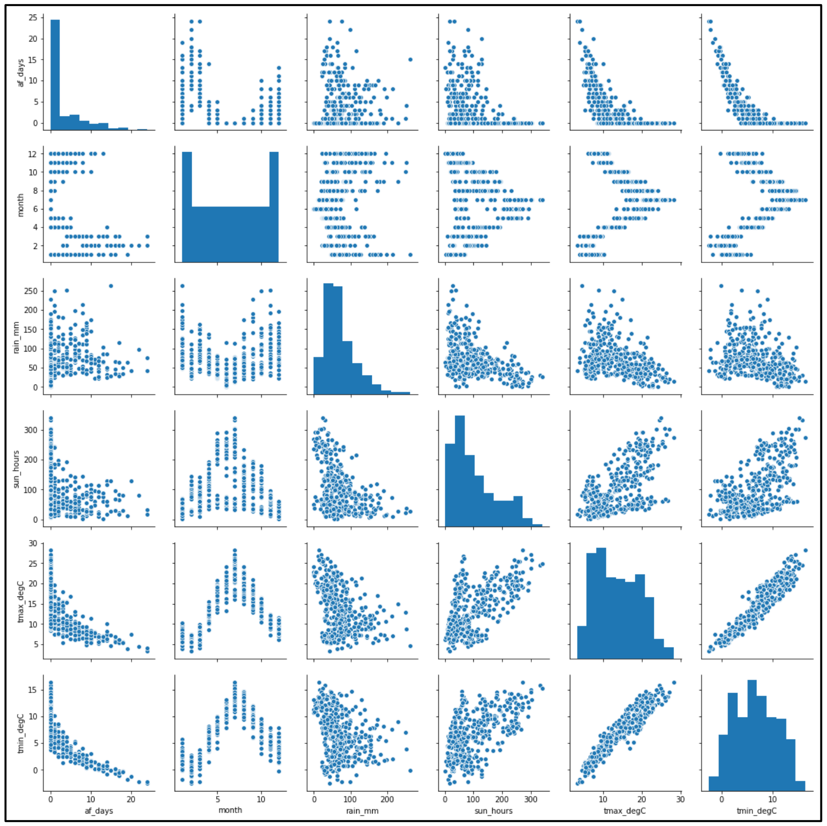
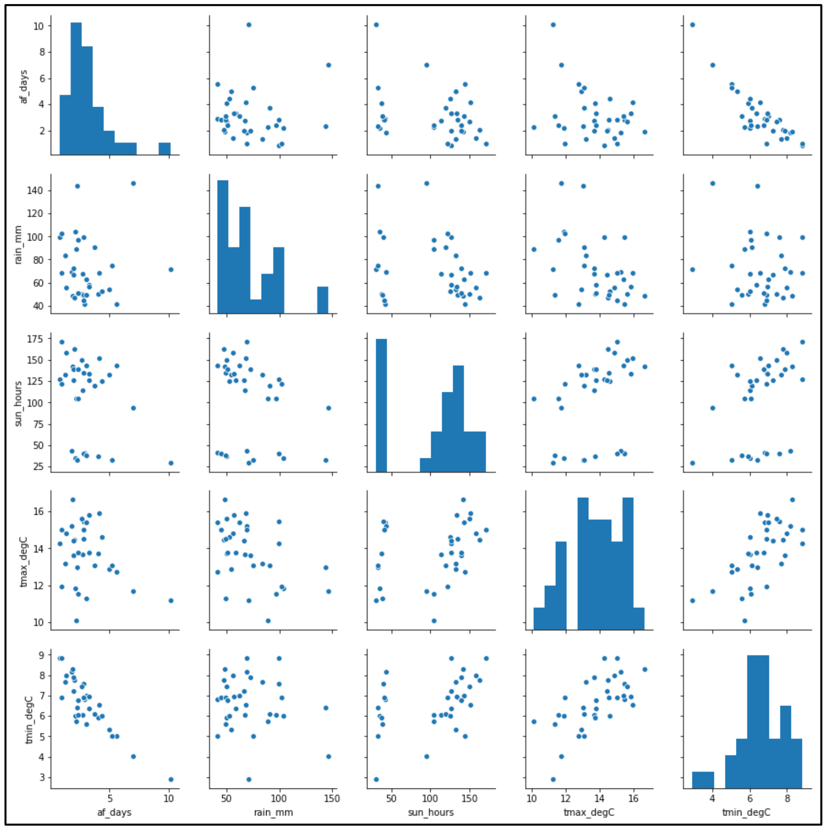


Figure 7: Pair Plot of Reduced 2018 Weather Data (Averaged over 12 months)



I performed clustering on the weather data using the KMeans algorithm implemented in Python's scikit-learn library. I decided to use KMeans because the output of the KMeans algorithm are easier to interpret compared to more complicated clustering algorithms like the Gaussian Mixture Models algorithm. KMeans is a simpler model than the Gaussian Mixture Models algorithm because the KMeans algorithm results in clearly defined cluster assignments, in contrast with the Gaussian Mixture Models that results in soft clusters with probabilities assigned to each datapoint for each cluster. In the event that I was not able to identify meaningful clusters within the data using KMeans, I would have considered a more complicated approach.

Before fitting the KMeans model, I normalized the data so that the distances calculated during KMeans clustering are not dominated by features with large numerical values e.g. rain. Normalization was performed using the MinMaxScaler in Python's scikit-learn library to transform all the weather features to a scale of 0 to 1.

I initially attempted to use three as the value of K corresponding to three weather clusters i.e. mild, moderate and extreme weather, but then I decided to use two as the value of K because the resulting clusters were relatively more distinct and easier to interpret when using two clusters.

I also performed KMeans clustering using all the recorded weather attributes. However, based on the observations from Figure 7 showing that the sun_hours attributes always created two distinct clusters when plotted against any of the other features, I decided to try KMeans clustering while omitting the sun_hours attribute. This is because I wanted to see if there are any other similarities, besides number of sun hours, with which stations could be clustered.

Part 1 - Results and Conclusions

With K=2, averaging data over specific seasons and averaging data over the whole year both resulted in the same composition of the discovered clusters. This implies that specific seasonal weather patterns are not any different from annual weather patterns for the purposes of identifying stations that have similar weather.

Table 4: Discovered clusters using all Weather Features

Cluster 0		Cluster 1
aberporth	nairn	ballypatrick
armagh	oxford	braemar
bradford	paisley	cambridge
camborne	rossonwye	cardiff
eastbourne	shawbury	chivenor
eskdalemuir	sheffield	dunstaffnage
heathrow	stornoway	durham
hurn	tiree	newtonrigg
lerwick	valley	suttonbonington
leuchars	waddington	wickairport
lowestoft	whitby	
manston	yeovilton	

The discovered clusters are not balanced, with significantly more stations being assigned to cluster 0 than to cluster 1. To further investigate the discovered clusters, a seaborn pair plot in Figure 8 is used.

As shown in Figure 8, the discovered clusters are mainly driven by the sun_hours feature. Removing the sun_hours feature results in the clusters being more distinct across all the features as shown in Figure 9. The cluster assignments obtained before and after dropping the sun_hours attribute are shown in Table 5.

Table 5: Discovered Clusters Before and After Dropping sun_hours Feature.

Using All Weather Features			**After Dropping sun_hours Feature**	
Cluster 0		**Cluster 1**	**Cluster 0**	**Cluster 1**
Key Similarities • High number of sun hours i.e. on average more than 100 sun hours recorded per month.		**Key Similarities** • Low number of sun hours i.e. on average less than 50 sun hours recorded per month.	**Key Similarities** • Relatively higher temperatures. • Relatively fewer number of frost days.	**Key Similarities** • Relatively lower temperatures. • Relatively higher number of frost days.
Members		**Members**	**Members**	**Members**
aberporth	nairn	ballypatrick	aberporth	ballypatrick
armagh	oxford	braemar	armagh	bradford
bradford	paisley	cambridge	camborne	braemar
camborne	rossonwye	cardiff	cambridge	durham
eastbourne	shawbury	chivenor	cardiff	eskdalemuir
eskdalemuir	sheffield	dunstaffnage	chivenor	lerwick
heathrow	stornoway	durham	dunstaffnage	leuchars
hurn	tiree	newtonrigg	eastbourne	nairn
lerwick	valley	suttonbonington	heathrow	newtonrigg
leuchars	waddington	wickairport	hurn	paisley
lowestoft	whitby		lowestoft	shawbury
manston	yeovilton		manston	stornoway
			oxford	wickairport
			rossonwye	
			sheffield	
			suttonbonington	
			tiree	
			valley	
			waddington	
			whitby	
			yeovilton	

Figure 8: Discovered Clusters Using all Weather Features

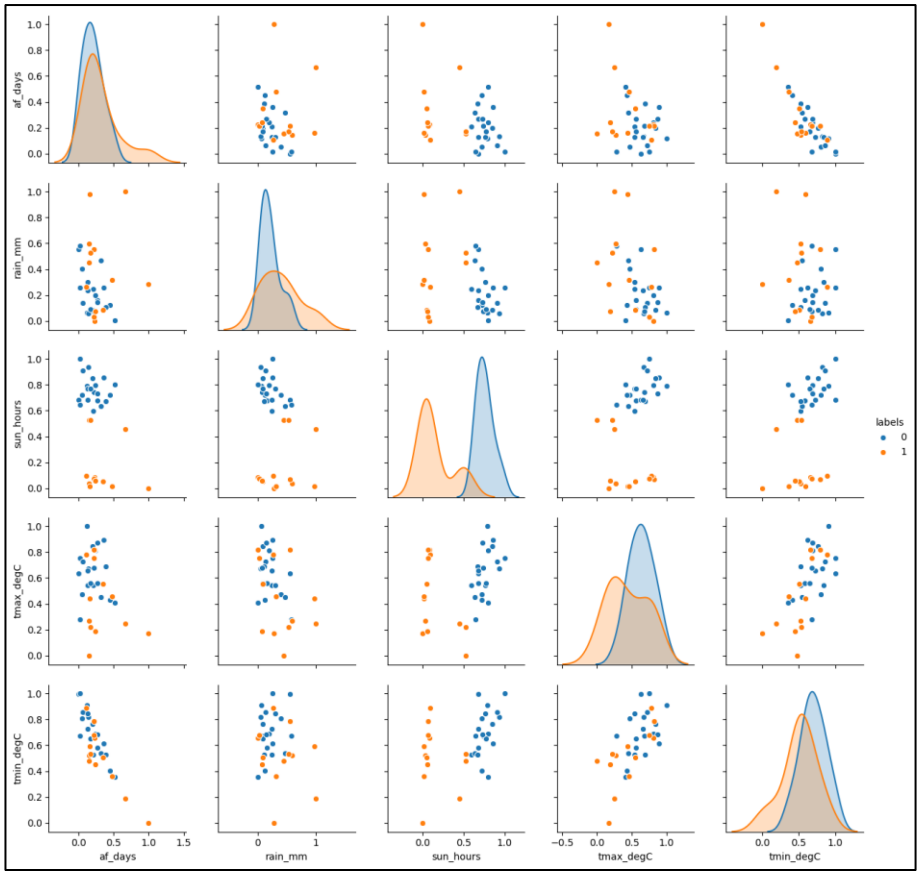
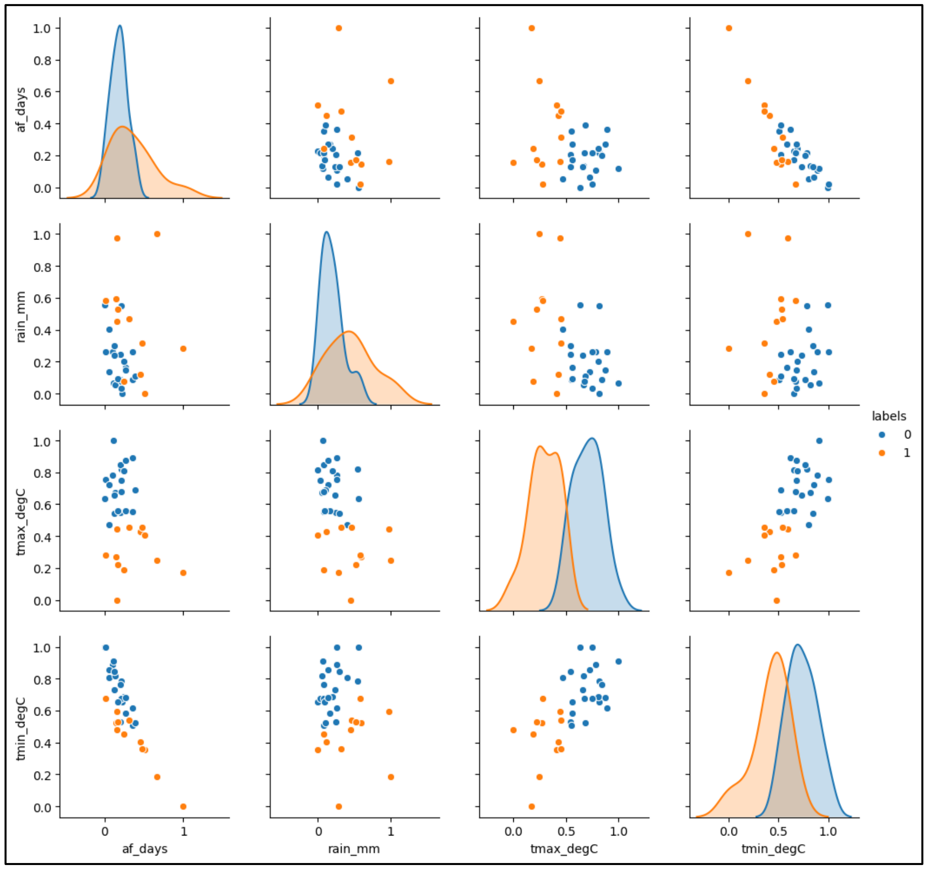


Figure 9: Discovered Clusters after Dropping the sun_hours Feature.



PART 2 - CLASSIFICATION

The code used to implement the steps described in part 2 can be found in the following notebooks:

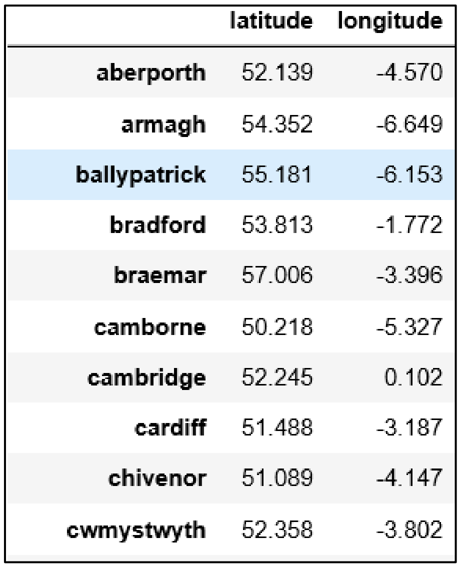
- *"explore_weather_data.ipynb"*
- *"read_store_lat-long_data.ipynb*"

Part 2 - Approach

Similarly to part 1, I reduced the weather data by selecting only 2018 data and averaging feature data over 12 months for each station.

Using the code in "*read_store_lat-long_data.ipynb*", I parsed the headings in the downloaded text files containing weather data to obtain the station coordinates. The coordinates of each station were then loaded into the data frame depicted in Figure 10. For stations like Nairn that changed locations at some point in time, I used whichever location that was provided in the format "Lat xx, Lon xx" in the station's text file header.

Figure 10: Data Frame showing a few Rows of Station Coordinate Data



	latitude	longitude
aberporth	52.139	-4.570
armagh	54.352	-6.649
ballypatrick	55.181	-6.153
bradford	53.813	-1.772
braemar	57.006	-3.396
camborne	50.218	-5.327
cambridge	52.245	0.102
cardiff	51.488	-3.187
chivenor	51.089	-4.147
cwmystwyth	52.358	-3.802

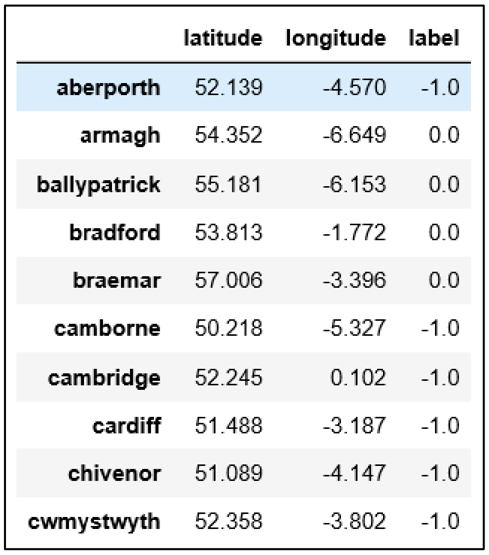
I then created a class label for each station based on the latitude using numerical labels shown in Table 6.

Table 6: Criteria Used to Create True Class Labels

Region	Latitude (L)	Class Label
Northern Third	L > 57.233	1
Central Third	53.567 < L < 57.233	0
Southern Third	L < 53.567	-1

The resulting labelled data shown in Figure 11 was then stored in collection that is co-located in the same Mongodb database as the weather data.

Figure 11: Data Frame showing a few Rows of Labeled Station Coordinate Data.

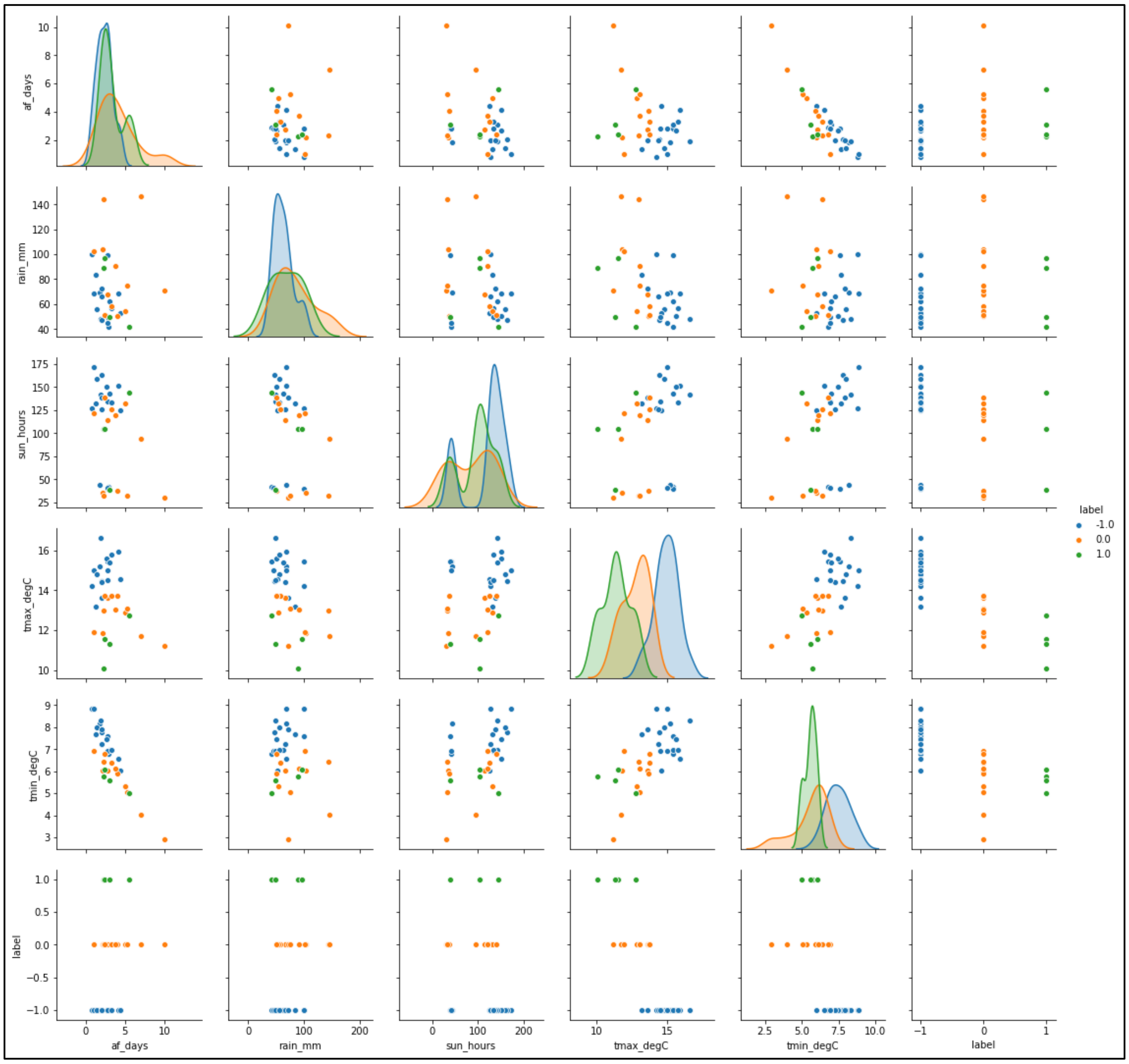


	latitude	longitude	label
aberporth	52.139	-4.570	-1.0
armagh	54.352	-6.649	0.0
ballypatrick	55.181	-6.153	0.0
bradford	53.813	-1.772	0.0
braemar	57.006	-3.396	0.0
camborne	50.218	-5.327	-1.0
cambridge	52.245	0.102	-1.0
cardiff	51.488	-3.187	-1.0
chivenor	51.089	-4.147	-1.0
cwmystwyth	52.358	-3.802	-1.0

I then joined the labelled coordinate data with the reduced 2018 weather data, and visualized the labelled dataset prior to training and testing a classification algorithm.

Figure 12 shows the weather data with class labels assigned to each station. Based on Figure 12, it is evident that the stations located in the lower third of the UK record relatively higher temperatures than the stations in the central and northern thirds. It is also noticeable that the stations in the southern third of the UK generally correspond to the Cluster 0 in Figure 9, while the stations in the northern and central thirds correspond to the Cluster 1 in Figure 9.

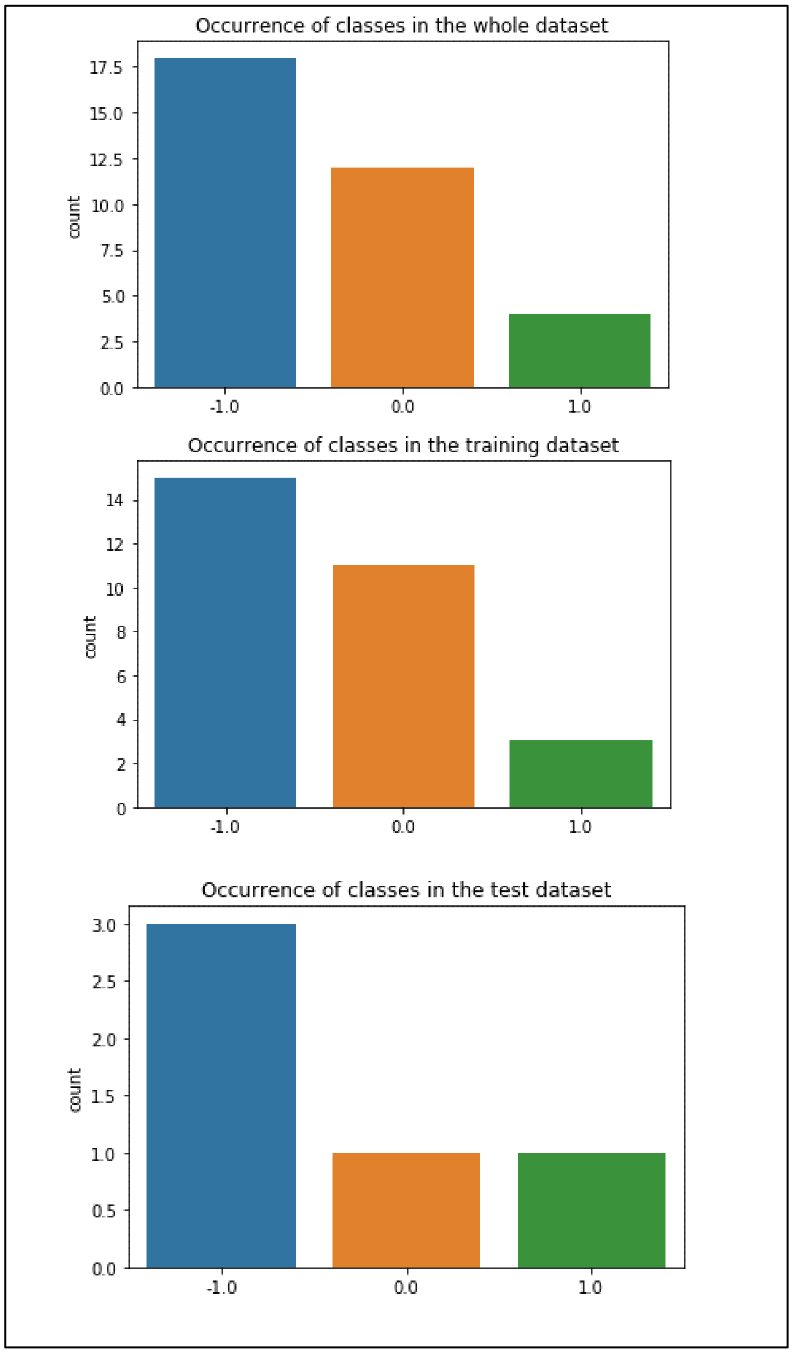
Figure 12: Pair Plot of 2018 Weather Data Showing Stations in the Northern, Central and Southern Thirds of the UK.



I decided to use a Gaussian Naïve Bayes classifier in Python's scikit-learn library to model the weather data. I chose to use a Gaussian Naïve Bayes model because the distributions of each feature, besides sun_hours, in the reduced weather data approximates a normal distribution. I also considered using the Random Forest Classifcation algorithm in the event that the results obtained from using the Gaussian Naïve Bayes classifier were not satisfactory.

Using the train_test_split method in scikit-learn, I randomly set aside 5 stations to be used in evaluating the accuracy performance of the Naïve Bayes classifier. Accuracy is deemed to be a reasonable metric for evaluating the classifiers on this data set because we are interested in building a classier that correctly predicts the location of a station given the station's weather data. Since the accuracy score is sensitive to the proportions of the different classes within the dataset, I used seaborn's count plot method to visualize the relative proportion of the three classes in the test set, training set and the whole data set a shown in Figure 13. The goal was to ensure that the class composition of the selected test set was approximately reflective of the class composition in the original dataset. If the class composition in the test set was, for example, such that it only contains a single class, the classifier would then trivially obtain an accuracy score of 100% by always predicting the same class.

Figure 13: Occurrence of Classes in the Labelled Weather Dataset



Part 2 - Results and Conclusions

First, I trained the Naive Bayes classifier using all the features of the reduced weather data and obtained the following accuracy performance:

- Training set accuracy – 86.21 %
- Testing set accuracy – 60.00 %

The classifier performed fairly well on the test set but was not able to generalize well to the training set. I then decided to investigate the effect of excluding each feature in-turn on the performance of the Naïve Bayes classifier. I hypothesized that excluding some features may allow the Naïve Bayes classifier to fit the data better. This is because some features, for example tmin_degC and tmax_degC, are strongly correlated hence invalidating the independence assumption of Naïve Bayes. The results of this investigation are summarized in Table 7.

Table 7: Naive Bayes Classifier Performance in Different Scenarios.

Scenario #	Excluded Feature	Training Set Accuracy (%)	Test Set Accuracy (%)	Comments
1	None	86.20	60.00	• Classifier performs fairly well on the training set, but poorly on the test set. • Classifier over fits the training data and fails to generalize.
2	af_days	89.66	60.00	• Removing af_days leads to a slight improvement in classification accuracy on the training set compared to scenario-1, however performance on the test set remains poor.
3	rain_mm	89.66	80.00	• Removing rain_mm also leads to a slight improvement in classification accuracy compared to scenario-1. • Over-fitting is also reduced compared to scenario-1 since accuracy performance on the test set improves by 20%. However, the model still fails to generalize because the accuracy performance on the training set exceeds that on the test set.
4	sun_hours	89.66	100.0	• Removing sun_hours leads to a significant improvement in accuracy performance on both the training set and the test set compared to scenario-1. • Moreover, the model does not over fit the training set and is able to generalize. The model achieves a perfect accuracy score on the test set. • The improvement in the model's performance after removing the sun_hours feature is due to the better fit to the model when the sun_hours feature is excluded. As discussed in PART 1 - Clustering, the sun_hours data is not normally distributed, yet the Gaussian Naïve Bayes classifier assumes that all the features are

Scenario #	Excluded Feature	Training Set Accuracy (%)	Test Set Accuracy (%)	Comments
				normally distributed. Consequently, the classier models the data better when the sun_hours feature is removed.
5	tmax_degC	75.86	60.00	• Removing tmax_degC results in a decline in the accuracy performance of the classifier on the training data. This implies that tmax_degC is important for modelling the underlying structure in the data, and excluding the feature results in a poor model of the dataset. • However, removing tmax_degC does not affect the accuracy performance on the test set. Perhaps this is because achieving a 60% accuracy performance is trivial and can be achieved even by a poor model of the data i.e. always predict the dominant class (southern third).
6	tmin_degC	82.76	80.00	• Removing tmin_degC also results in a small decline in the accuracy performance on the training set. However, over-fitting is reduced compared to scenario-1 since accuracy performance on the test set improves by 20%. • Nonetheless, the model still fails to generalize because the accuracy performance on the training set exceeds that on the test set. • Perharps the decline in performance that in scenario-6 is smaller than that in scenario-5 because of the conditional independence assumption made by the Naïve Bayes classifier. As discussed in PART 1 - Clustering, tmin_dgeC is strongly correlated with both tmax_degC and af_days. However, the Naïve Bayes classifier assumes that all the weather futures are independent. Therefore, by removing tmin_degC, the validity of the Naïve Baes assumption increases.

Based on the results discussed in Table 4, the model in scenario-4 was selected as a satisfactory classifier for the purpose specified in this assessment.

PART 3 - JOINING DATASETS

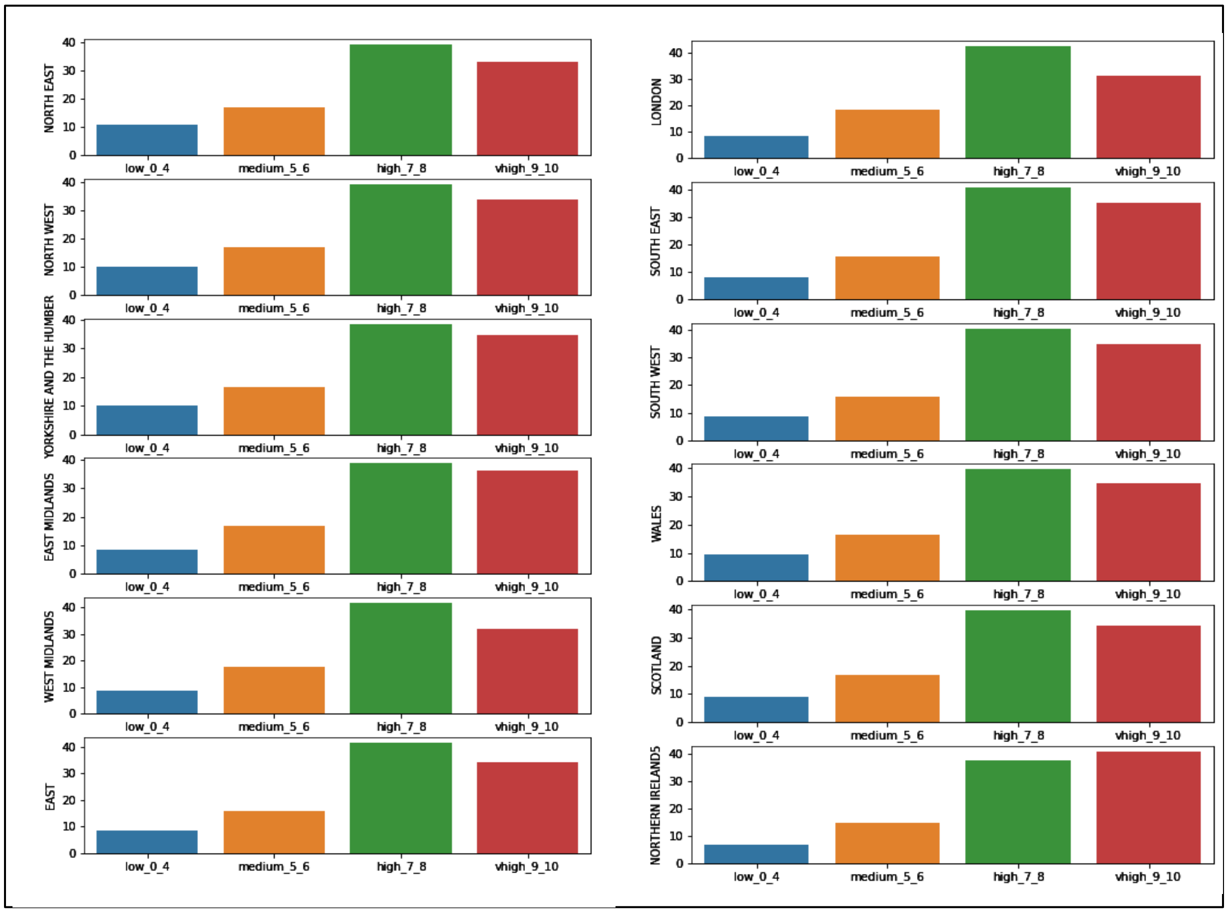
The code for implementing the steps described in Part 3 can be found in the following notebooks:

- *"join_happiness_and_weather_data.ipynb"*
- *"visualize_locations.ipynb"*

Part 3 - Approach

Before joining the happiness data with the weather data, I visualized the happiness data as shown in Figure 14. It is evident that the happiness data is biased towards the high end of the 0 - 10 scale. Based on the data, either people in the UK are generally happy or people tend to overestimate their own level of happiness. I am of the opinion that the latter is more likely.

Figure 14: Plot Showing Proportion People in each Happiness Category for Each Area.



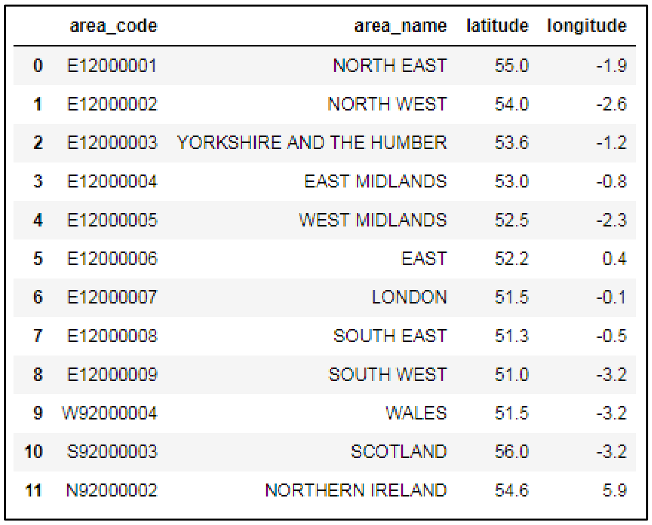
For the purposes of assessing how weather affects happiness, I reduced the happiness data by combining the low and medium categories into a single category i.e. "unhappy", and the high and very high categories into a "happy" category. Reducing the happiness data made it easier to visually compare the levels of happiness associated with specific weather stations using only three intuitive happiness-metrics i.e.:

1. Average happiness rating ("avg_rating")
2. Percentage of population that is happy ("happy")
3. Percentage of population that is unhappy ("unhappy")

Using the station coordinates (Figure 10) and the area center coordinates (Figure 15), for each station, I identified the closest area center using Euclidean distance as the metric for determining how close a station is from the center of an area. The resulting pairing of stations to area codes is shown in Figure

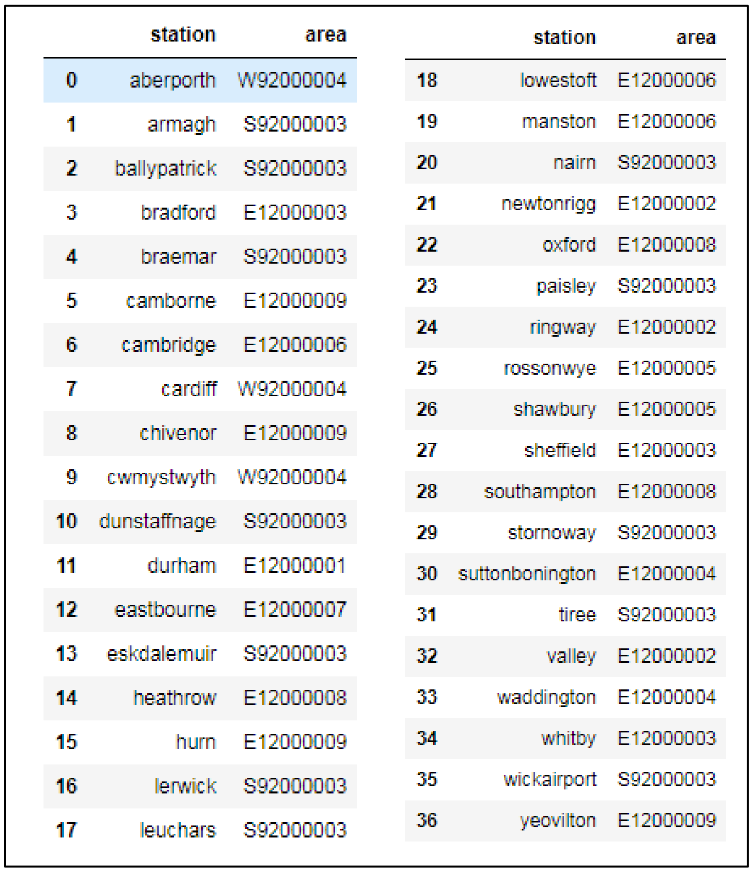
16. I also created a Voronoi diagram (Figure 17) showing how the UK was partitioned in order to assign each station to the closest area center.

Figure 15: Data Frame Showing Area Codes and the Coordinates Corresponding to the Center of Each Area



	area_code	area_name	latitude	longitude
0	E12000001	NORTH EAST	55.0	-1.9
1	E12000002	NORTH WEST	54.0	-2.6
2	E12000003	YORKSHIRE AND THE HUMBER	53.6	-1.2
3	E12000004	EAST MIDLANDS	53.0	-0.8
4	E12000005	WEST MIDLANDS	52.5	-2.3
5	E12000006	EAST	52.2	0.4
6	E12000007	LONDON	51.5	-0.1
7	E12000008	SOUTH EAST	51.3	-0.5
8	E12000009	SOUTH WEST	51.0	-3.2
9	W92000004	WALES	51.5	-3.2
10	S92000003	SCOTLAND	56.0	-3.2
11	N92000002	NORTHERN IRELAND	54.6	5.9

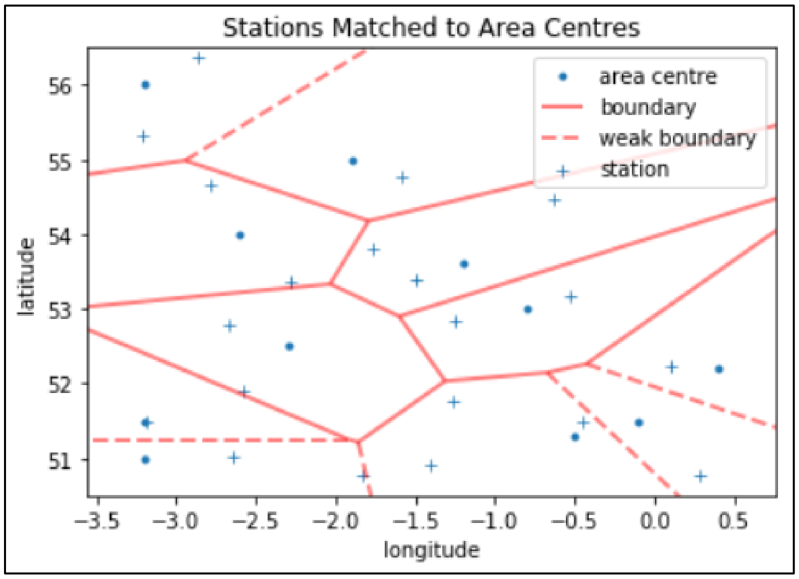
Figure 16: Data Frames Showing Stations Names Assigned to the Closest Areas



	station	area
0	aberporth	W92000004
1	armagh	S92000003
2	ballypatrick	S92000003
3	bradford	E12000003
4	braemar	S92000003
5	camborne	E12000009
6	cambridge	E12000006
7	cardiff	W92000004
8	chivenor	E12000009
9	cwmystwyth	W92000004
10	dunstaffnage	S92000003
11	durham	E12000001
12	eastbourne	E12000007
13	eskdalemuir	S92000003
14	heathrow	E12000008
15	hurn	E12000009
16	lerwick	S92000003
17	leuchars	S92000003

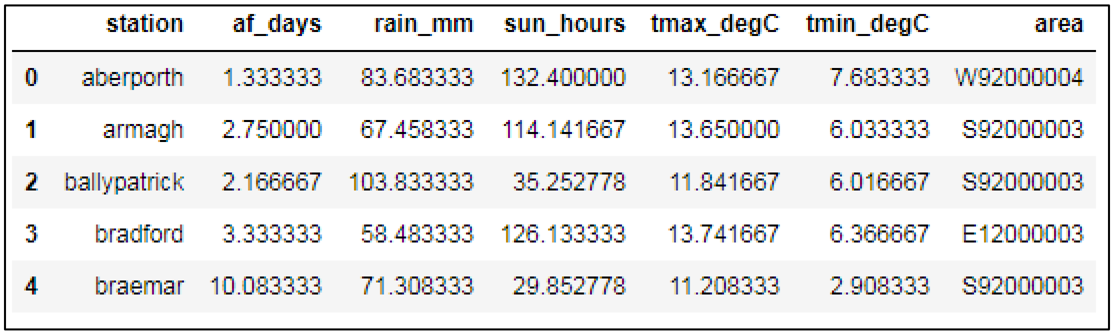
	station	area
18	lowestoft	E12000006
19	manston	E12000006
20	nairn	S92000003
21	newtonrigg	E12000002
22	oxford	E12000008
23	paisley	S92000003
24	ringway	E12000002
25	rossonwye	E12000005
26	shawbury	E12000005
27	sheffield	E12000003
28	southampton	E12000008
29	stornoway	S92000003
30	suttonbonington	E12000004
31	tiree	S92000003
32	valley	E12000002
33	waddington	E12000004
34	whitby	E12000003
35	wickairport	S92000003
36	yeovilton	E12000009

Figure 17: A Close-Up Voronoi Diagram Showing the Partitioning of the UK Using the Provided Area Centers.



I then assigned the area codes to the reduced 2018 weather data using the pairs listed in Figure 16 to create a data frame such as the one depicted in Figure 18. The weather data containing area codes was then stored in a new collection in the same Mongodb database where the happiness data was stored.

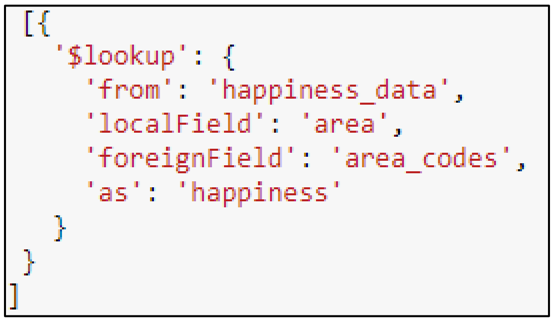
Figure 18: Data Frame Showing A Few Rows of Area Codes Joined with Weather Data



	station	af_days	rain_mm	sun_hours	tmax_degC	tmin_degC	area
0	aberporth	1.333333	83.683333	132.400000	13.166667	7.683333	W92000004
1	armagh	2.750000	67.458333	114.141667	13.650000	6.033333	S92000003
2	ballypatrick	2.166667	103.833333	35.252778	11.841667	6.016667	S92000003
3	bradford	3.333333	58.483333	126.133333	13.741667	6.366667	E12000003
4	braemar	10.083333	71.308333	29.852778	11.208333	2.908333	S92000003

Using the aggregation shown in Figure 19, I joined the weather data with the happiness data based on area codes. The resulting joined dataset is shown in Table 8.

Figure 19: Aggregation Used to Join Weather Data and Happiness Data



```
[{
  '$lookup': {
    'from': 'happiness_data',
    'localField': 'area',
    'foreignField': 'area_codes',
    'as': 'happiness'
  }
 }
]
```

Table 8: Joined Weather and Happiness Data

#	area_codes	latitude	longitude	area_names	station	station_lat	station_long	low_0_4	medium_5_6	high_7_8	vhigh_9_10	avg_rating	af_days	rain_mm	sun_hours	tmax_degC	tmin_degC
0	W92000004	51.5	-3.2	WALES	aberporth	52.139	-4.57	9.56	16.54	39.45	34.45	7.44	1.33	83.68	132.40	13.17	7.68
1	S92000003	56	-3.2	SCOTLAND	armagh	54.352	-6.649	9.16	16.9	39.75	34.19	7.45	2.75	67.46	114.14	13.65	6.03
2	S92000003	56	-3.2	SCOTLAND	ballypatrick	55.181	-6.153	9.16	16.9	39.75	34.19	7.45	2.17	103.83	35.25	11.84	6.02
3	E12000003	53.6	-1.2	YORKSHIRE AND THE HUMBER	bradford	53.813	-1.772	10.21	16.6	38.56	34.63	7.41	3.33	58.48	126.13	13.74	6.37
4	S92000003	56	-3.2	SCOTLAND	braemar	57.006	-3.396	9.16	16.9	39.75	34.19	7.45	10.08	71.31	29.85	11.21	2.91
5	E12000009	51	-3.2	SOUTH WEST	camborne	50.218	-5.327	8.79	15.92	40.44	34.84	7.5	0.83	99.68	126.88	14.24	8.83
6	E12000006	52.2	0.4	EAST	cambridge	52.245	0.102	8.53	15.69	41.69	34.09	7.51	2.92	41.48	41.83	15.43	6.80
7	W92000004	51.5	-3.2	WALES	cardiff	51.488	-3.187	9.56	16.54	39.45	34.45	7.44	2.83	99.30	39.82	15.43	7.58
8	E12000009	51	-3.2	SOUTH WEST	chivenor	51.089	-4.147	8.79	15.92	40.44	34.84	7.5	1.83	69.12	43.77	15.20	8.18
9	W92000004[1]	51.5	-3.2	WALES	cwmystwyth	52.358	-3.802	9.56	16.54	39.45	34.45	7.44	-	-	-	-	-
10	S92000003	56	-3.2	SCOTLAND	dunstaffnage	56.451	-5.439	9.16	16.9	39.75	34.19	7.45	2.33	143.94	32.31	12.98	6.43
11	E12000001	55	-1.9	NORTH EAST	durham	54.768	-1.585	10.77	17.05	39.06	33.13	7.34	4.08	50.38	37.36	13.71	5.91
12	E12000007	51.5	-0.1	LONDON	eastbourne	50.762	0.285	8.32	18.3	42.2	31.18	7.38	1.00	68.80	171.62	15.02	8.84
13	S92000003	56	-3.2	SCOTLAND	eskdalemuir	55.311	-3.206	9.16	16.9	39.75	34.19	7.45	7.00	146.31	94.38	11.71	4.02
14	E12000008	51.3	-0.5	SOUTH EAST	heathrow	51.479	-0.449	7.97	15.83	40.88	35.32	7.54	1.92	48.33	142.07	16.63	8.30
15	E12000009	51	-3.2	SOUTH WEST	hurn	50.779	-1.835	8.79	15.92	40.44	34.84	7.5	4.17	68.73	151.48	15.92	6.56
16	S92000003	56	-3.2	SCOTLAND	lerwick	60.139	-1.183	9.16	16.9	39.75	34.19	7.45	2.25	88.88	104.53	10.08	5.74
17	S92000003	56	-3.2	SCOTLAND	leuchars	56.377	-2.861	9.16	16.9	39.75	34.19	7.45	5.00	54.19	132.19	12.88	5.32
18	E12000006	52.2	0.4	EAST	lowestoft	52.483	1.727	8.53	15.69	41.69	34.09	7.51	2.08	47.40	162.93	14.48	7.78
19	E12000006	52.2	0.4	EAST	manston	51.346	1.337	8.53	15.69	41.69	34.09	7.51	1.42	55.93	158.57	14.83	7.99
20	S92000003	56	-3.2	SCOTLAND	nairn	57.593	-3.821	9.16	16.9	39.75	34.19	7.45	5.58	41.78	143.70	12.75	5.03
21	E12000002	54	-2.6	NORTH WEST	newtonrigg	54.67	-2.786	10.17	16.82	39.1	33.91	7.39	5.25	74.87	32.34	13.07	5.04
22	E12000008	51.3	-0.5	SOUTH EAST	oxford	51.761	-1.262	7.97	15.83	40.88	35.32	7.54	2.67	50.52	150.14	15.62	7.45
23	S92000003	56	-3.2	SCOTLAND	paisley	55.846	-4.43	9.16	16.9	39.75	34.19	7.45	3.75	90.43	120.02	13.05	6.12
24	E12000002[2]	54	-2.6	NORTH WEST	ringway	53.356	-2.279	10.17	16.82	39.1	33.91	7.39	-	-	-	-	-
25	E12000005	52.5	-2.3	WEST MIDLANDS	rossonwye	51.911	-2.584	8.61	17.59	41.72	32.08	7.43	3.08	62.54	142.77	15.39	6.98
26	E12000005	52.5	-2.3	WEST MIDLANDS	shawbury	52.794	-2.663	8.61	17.59	41.72	32.08	7.43	4.42	52.77	125.16	14.58	6.03
27	E12000003	53.6	-1.2	YORKSHIRE AND THE HUMBER	sheffield	53.381	-1.49	10.21	16.6	38.56	34.63	7.41	2.00	66.52	126.48	14.39	7.23
28	E12000008[3]	51.3	-0.5	SOUTH EAST	southampton	50.898	-1.408	7.97	15.83	40.88	35.32	7.54	-	-	-	-	-
29	S92000003	56	-3.2	SCOTLAND	stornoway	58.214	-6.318	9.16	16.9	39.75	34.19	7.45	2.42	96.82	104.47	11.54	6.06
30	E12000004	53	-0.8	EAST MIDLANDS	suttonbonington	52.833	-1.25	8.66	16.74	38.62	35.99	7.51	2.83	44.68	40.40	15.00	6.92
31	S92000003	56	-3.2	SCOTLAND	tiree	56.5	-6.88	9.16	16.9	39.75	34.19	7.45	1.00	102.48	121.43	11.93	6.91
32	E12000002	54	-2.6	NORTH WEST	valley	53.252	-4.535	10.17	16.82	39.1	33.91	7.39	2.00	72.77	138.88	13.64	7.91
33	E12000004	53	-0.8	EAST MIDLANDS	waddington	53.175	-0.522	8.66	16.74	38.62	35.99	7.51	2.83	49.48	134.77	14.52	6.91
34	E12000003	53.6	-1.2	YORKSHIRE AND THE HUMBER	whitby	54.481	-0.624	10.21	16.6	38.56	34.63	7.41	2.42	51.28	139.25	13.74	6.78
35	S92000003	56	-3.2	SCOTLAND	wickairport	58.454	-3.088	9.16	16.9	39.75	34.19	7.45	3.08	49.55	38.24	11.30	5.58
36	E12000009	51	-3.2	SOUTH WEST	yeovilton	51.006	-2.641	8.79	15.92	40.44	34.84	7.5	3.33	56.75	133.30	15.79	6.96

[1] cwmystwyth station was closed before 2018 so there was no weather data recorded for 2018.
[2] ringway station was closed before 2018 so there was no weather data recorded for 2018.
[3] Southampton station was closed before 2018 so there was no weather data recorded for 2018.

After joining the weather data with the happiness data, I decided to look for patterns that might point to a relationship between the weather and happiness levels. The investigation mainly entailed three steps:

- Plotting the weather attributes against each of the three happiness-metrics to see if there are any clear correlations (see Table 9).
- Superimposing happiness metrics onto the weather station locations, and then visualizing the spatial variation of the surveyed happiness levels (see Table 10).
- Comparing the observed spatial variations in happiness levels with spatial variations in the different weather attributes, with the goal of identifying correlations and patterns (see Table 10 and Table 11).

Part 3 - Results and Conclusions

As shown in Table 9, maximum and minimum temperature has the have the strongest correlation with happiness compared to the other weather attributes. Moreover, this observation is true for all the three happiness-metrics.

Table 9: Pearson Correlation Coefficients between Weather Attributes and Happiness Metrics.

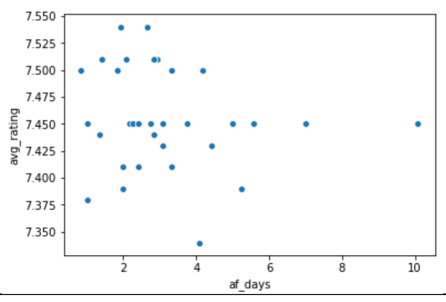
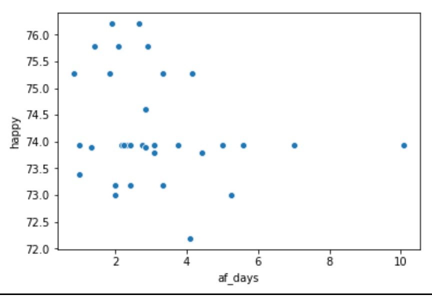
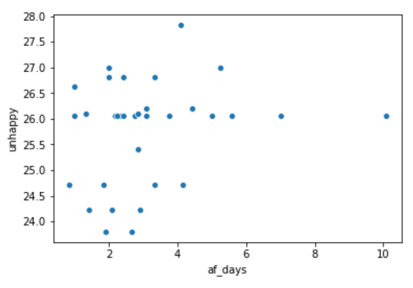
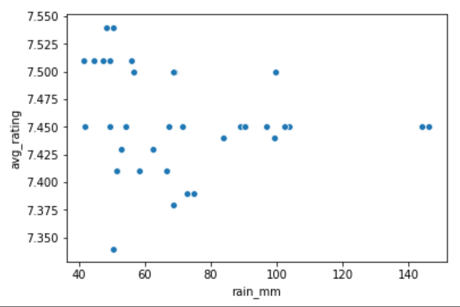
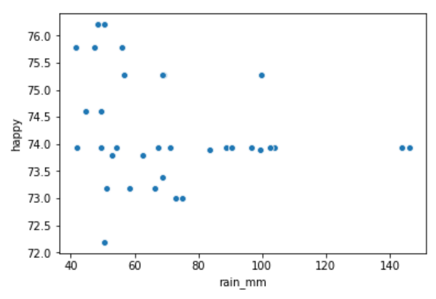
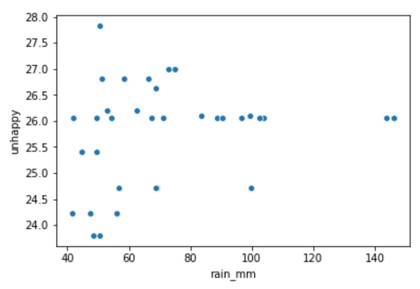
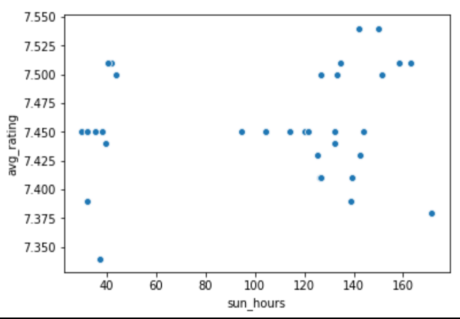
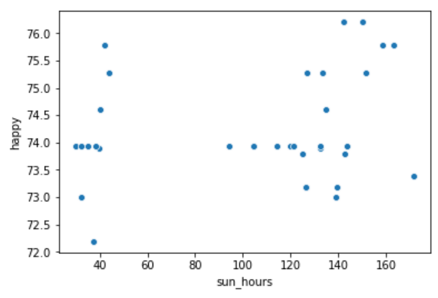
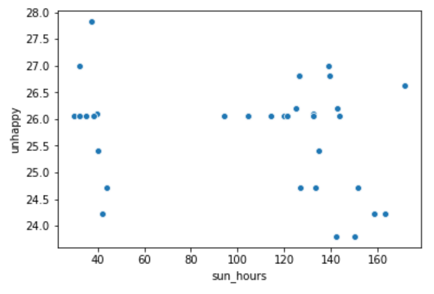
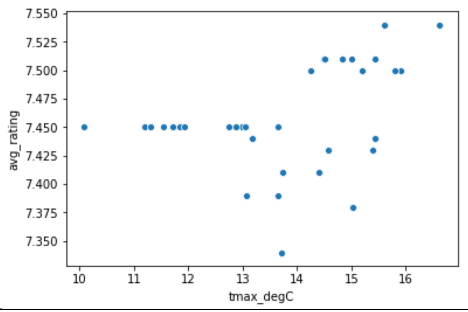
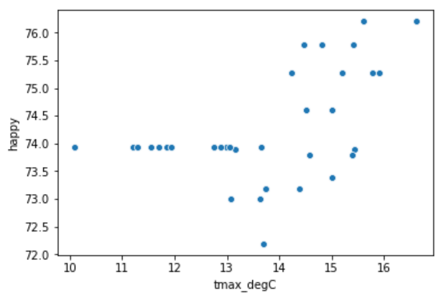
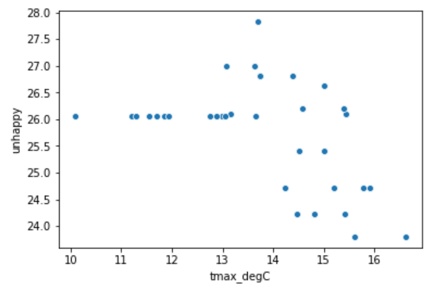
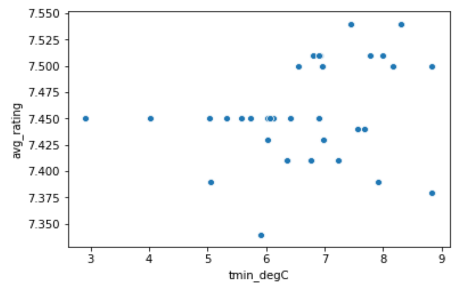
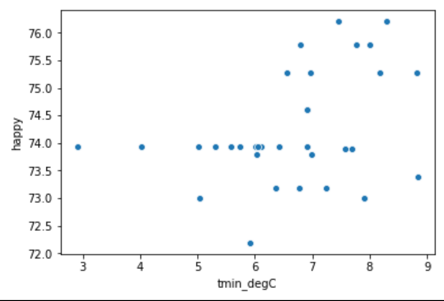
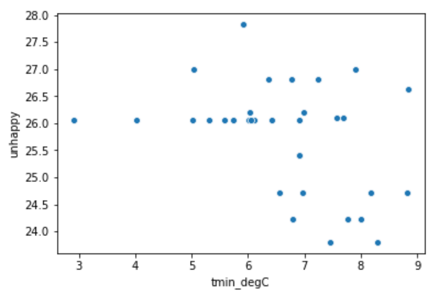
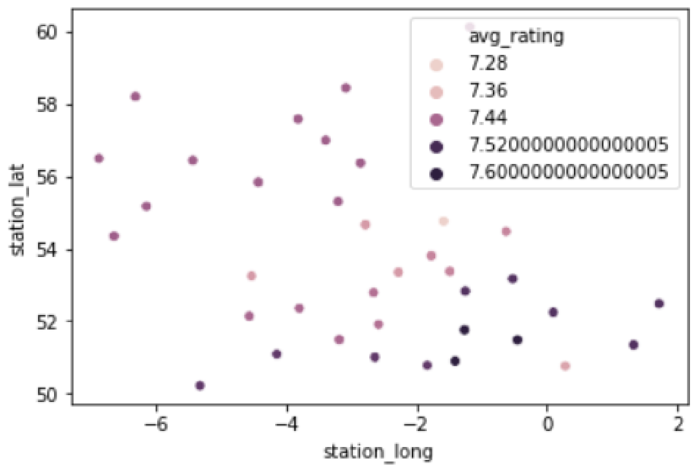
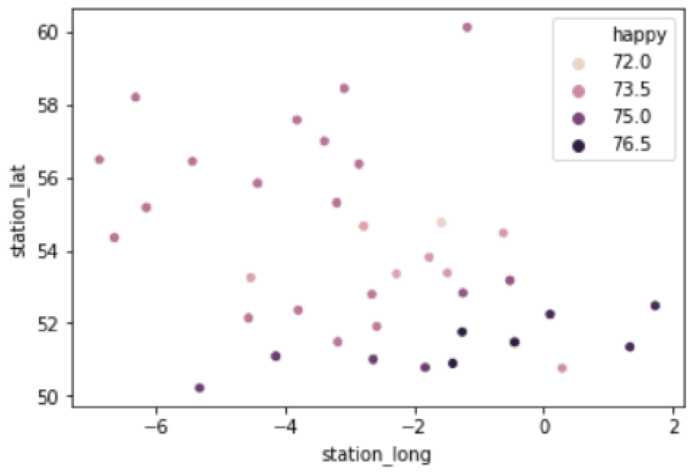
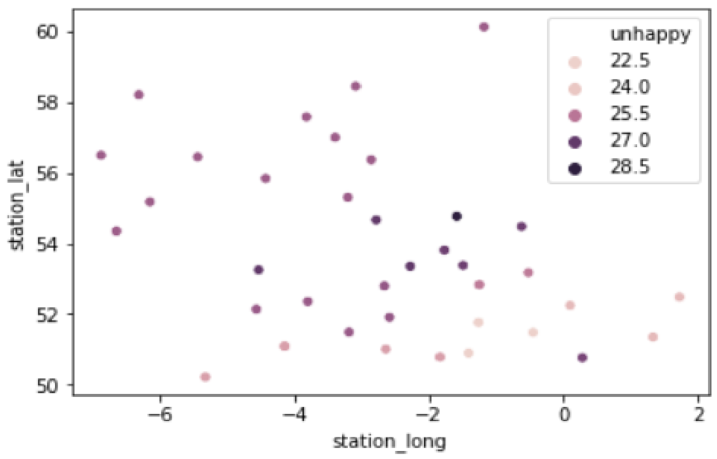
	Happiness Metric		
Weather Attribute	**Average Happiness Rating**	**Percentage of Population that are Happy**	**Percentage of population that are Unhappy**
af_days	Coefficient = -0.1511 	Coefficient = -0.2142 	Coefficient = 0.2143 
rain_mm	Coefficient = -0.1641 	Coefficient = -0.2128 	Coefficient = 0.2116 
sun_hours	Coefficient = 0.1535 	Coefficient = 0.2284 	Coefficient = -0.2289 
tmax_degC	Coefficient = 0.3795 	Coefficient = 0.4942 	Coefficient = -0.4942 
tmin_degC	Coefficient = 0.264 	Coefficient = 0.3857 	Coefficient = -0.3861 

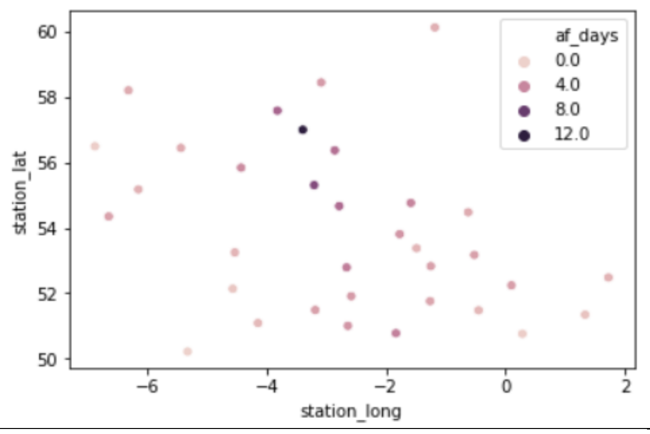
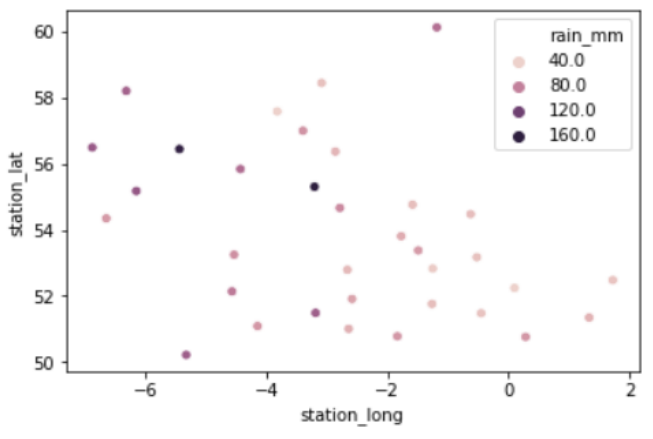
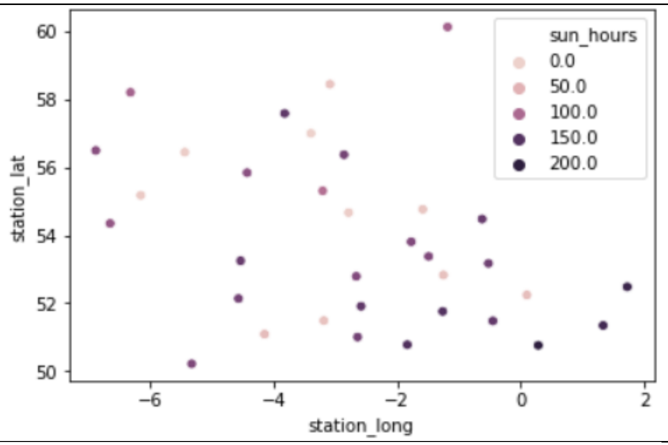
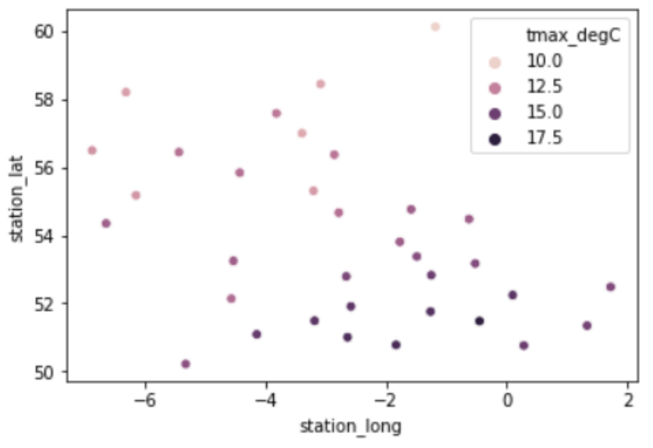
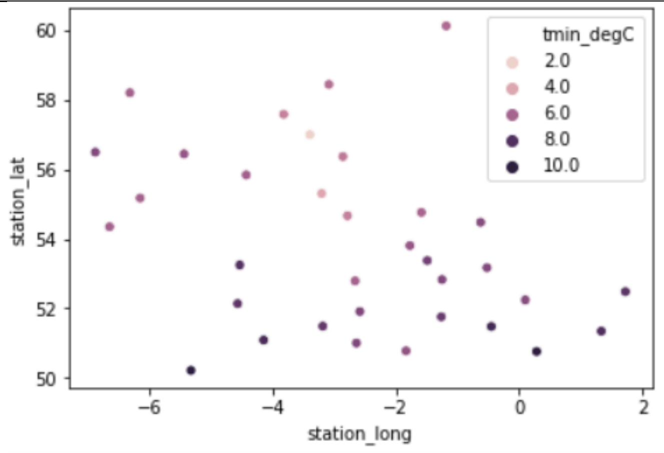
Table 10 summarizes how the observed happiness varies across different weather station locations considering the three happiness-metrics.

Table 10: Spatial Variation of Happiness Ratings

Spatial Plot (latitude vs longitude)	Observation Remarks
	• Stations in the southern part of the UK are associated with relatively higher average happiness ratings, compared to stations in northern part.
	• In the southern parts of the UK, a larger percentage of the population classify themselves as happy compared to the northern parts.
	• In the northern parts of the UK, a larger percentage of the population classify themselves as unhappy compared to the southern parts.

In spite of the bias in the happiness data (i.e. people generally over estimating their own happiness levels), one may conclude that on average, people in the southern parts of the UK are generally happier than the people in the north based on the results in Table 10. In light, of the observed happiness patterns, Table 11, highlights the spatial patterns in weather attributes and how the patterns may point to a relationship between happiness and weather.

Table 11: Spatial Variation of Weather Attributes

Spatial Plot (latitude vs longitude)	Observation Remarks
	• On average, stations in the northern-central parts of the UK record higher numbers of frost days compared to other parts of the UK. • The observed pattern does not indicate a possible relationship between the number of frost days and happiness. This is because there is no clear correlation between the recorded number of frost days and happiness ratings.
	• On average, stations in the northwestern parts of the UK record relatively higher amounts of rainfall compared to stations in the eastern and southern parts. • The observed pattern indicates a very weak negative correlation between the amount of rain and happiness. The southern parts that receive less rainfall also report higher happiness levels compared to the northern parts that receive more rainfall. However, it should be noted that this observation is not sufficient to infer causation i.e. that low amounts of rainfall cause high levels of happiness or vice versa.
	• There are stations with a low number of sun hours located all over the UK. However, there are more stations in the southern parts of the UK with a high number of sun hours compared to the northern part. This implies that overall, places in southern parts of the UK are likely to receive more sun hours than places in northern parts. • The observed pattern indicates a very weak positive correlation between the number of sun hours received in an area and happiness. The southern areas that record a higher number of sun hours also report higher levels of happiness. However, it should be noted that this observation is not sufficient to infer causation i.e. that a high number of sun hours causes higher levels of happiness.
	• Stations in the southern parts of the UK on average record higher maximum temperatures compared to stations in the northern parts. • The observed pattern indicates a relatively strong positive correlation between the maximum temperature recorded in an area and happiness. The southern areas that record higher maximum temperatures also report higher levels of happiness. However, it should be noted that this observation is not sufficient to infer causation i.e. that high maximum temperatures cause higher levels of happiness. • Of all the weather attributes, maximum-temperature has the strongest correlation with happiness. The observed pattern is also consistent with the Pearson correlation coefficients shown in Table 9.
	• Stations in the northern parts of the UK on average record lower minimum temperatures compared to stations in the southern parts. This is expected given the correlation that was already observed between minimum-temperature and maximum-temperature. • The observed pattern indicates a relatively strong positive correlation between the minimum temperature recorded in an area and happiness. The northern areas that record lower minimum temperatures also report lower levels of happiness. However, it should be noted that this observation is not sufficient to infer causation i.e. that low minimum temperatures cause lower levels of happiness.

PART 4B – AUTOMATION

I automated the steps described above using a number of scripts written in Python. The main scripts for each part can be executed directly on the command line i.e. (python script_name.py). Each of the scripts makes use of custom utility scripts where most of the code resides. The input parameters can be changed by directly editing the main script code as indicated by the comments included in the file.

Preprocessing

Automation scripts:

- *processing.py*: This script performs the following actions:
    - Downloads weather data for a set of stations specified either as a list contained in a text file "stations.txt".
    - Cleans the weather data for all the stations by removing non-numerical characters. All the downloaded weather data is loaded into a single data frame.
    - Stores all the cleaned weather data in a Mongodb database.

- *utility_scripts\processing_utils.py*: Contains utility functions that are used by *processing.py*

Part 1

Automation scripts:

- *part1_clustering.py:* This script performs clustering of weather data using a set of parameters that are specified by the user. The user is required to specify the following parameters:
    - **sample_years:** list of years which should be included in the sample data.
    - **season:** list of months that should be included when calculating a reduced weather dataset.
    - **num_clusters:** the number of clusters to be discovered
    - **features:** list of weather attributes to be used for clustering.
- *utility_scripts/clustering_utils.py:* Contains utility functions that are used by *part1_clustering.py.*

Part 2

Automation scripts:

- *label_station_data.py*: This script assigns labels to all the weather stations and stores the labeled data set in a Mongodb database. The labels are created depending on the number of regions specified by the user. e.g. for three regions, the labels {0, 1, 2} will be used, and for five regions, the labels {0, 1, 2, 3, 4} will be used.
- *part2_classification.py:* This script reports the classification accuracy score using a sample dataset that can be specified by the user. The user can specify the following parameters when running the script:
    - **test_size:** The number of data points to be set aside for evaluating classification accuracy expressed as a proportion of the total number of datapoints in the dataset.
    - **sample_years:** list of years which should be included in the sample data.
    - **season:** list of months that should be included when calculating a reduced weather dataset.
    - **features:** list of weather attributes to be used for classification.

    - **num_regions**: the number of parts (classes) into which to divide the UK for classification purposes.
- *utility_scripts/classification_utils.py*: Contains utility functions that are used by *part2_classification.py.*
- *utility_scripts/clustering_utils.py:* Also contains some (shared) utility functions that are used by *part2_classification.py.*

Part 3

Automation Scripts:

- *store_happiness_data.py:* This script reads happiness data from a source spreadsheet that was downloaded from the Office for National Statistics website. (source url: http://www.ons.gov.uk/peoplepopulationandcommunity/wellbeing/datasets/personalwellbeingestimatesgeographicalbreakdown). The script parses the spreadsheet and only selects data that corresponds to a specified set of area codes. The data is then stored in a Mongodb database.
- *part3_joining_data.py:* This script a joins a specified sample of weather data with surveyed happiness data based on area codes. The also script reports correlations between each weather attribute and happiness metric, and writes the joined dataset to a specified file format i.e. (either csv, xls, xlsx or printing to standard output). The user can specify the following parameters:
    - **output:** the format for reporting the joined dataset
    - **metrics:** the metrics to use for correlation. options include the following: ("avg_rating", "happy", "unhappy", "low_0_4", "medium_5_6", "high_7_8", "vhigh_9_10"
    - **features:** The weather attributes to be included when reporting correlations.
- *utility_scripts/happiness_data_utils.py:* Contains utility functions that are used by *store_happiness_data.py* and *part3_joining_data.py.*