Weather Data Reading and Pre-processing

This section explains how the weather data for the stations were obtained and cleaned to be used for data exploration, visualization, and analysis. The code is found in reading_and_pre_processing.py

Reading weather data into a dataframe

Using the stations.txt file downloaded from the learn page, a list of the stations was created. Then, a helper function, *get_station_data* was created, which takes a station name, makes a request to this url: (https://www.metoffice.gov.uk/pub/data/weather/uk/climate/stationdata/) and returns the data for that station in its raw format.

Using the list of stations and the *get_station_data* function, another function called *get_all_data_and_preprocess* was defined, which retrieves the necessary bit of the data (i.e. all the weather data including their latitude, longitude, and station name) for each of the station employing some cleaning steps like

- lines.index('yyyy mm tmax tmin af rain sun') : to obtain the index of that string to retrieve the weather data
- using a regex which a pattern, r'Lat\s+(-?\d+\.\d+)\s+Lon\s+(-?\d+\.\d+)' : to retrieve the latitude and longitude values
- data_ = [x[0:7] if len(x) > 7 else x +((['None'] * (7-len(x)))) for x in data] : to drop any unnecessary column I wouldn't need in the weather data i.e. any other columns after 'sun' like provisional and extending any data which does not have 7 values with 'None' values as they are 7 features 'yyyy mm tmax tmin af rain sun'.
- storing the data (weather data, station name, longitude, and latitude) in a dataframe then converting string of the form, num* and num# to just num string, where number represent a float in a string using df = df.applymap(lambda x: re.findall(r'\d+\.\d+|\d+', str(x))[0] if (re.findall(r'\d+\.\d+|\d+', str(x))) else x)

and finally, concatenating the dataframes for each station into one dataframe called full_data, which is then returned.

Further cleaning/pre-processing

This stage ensured that the data was consistent across all the stations and ready to be used by employing further cleaning like putting the data in their types for each column, using interpolation techniques to handle missing data and aligning the data for the stations.

It was divided into the steps which are explained below:

Further cleaning:

Because there were some rows in the data whose year and month were not integers, for instance, station: Cwmystwyth with row data [year: Site, month: closed, tmax: 0, tmin: 0, af: 0, rain: 0, sun: 0]) , these were removed from the data with the help of a function, *clean_further,* which when the dataframe from *get_all_data_and_preprocess* is passed does that. Also, the function converts some columns to their appropriate type like year and month to int and the temperatures, rainfall, and sun values to float.

Extension

As some stations were closed in earlier years (like Cwmystwyth closed in 2011), there was a chunk of data missing from the time they were closed to the March 2023. Thus, it was necessary to extend the data, filling each column with Nan values for each station such that they can be predicted using some techniques and to ensure that all stations have data to March 2023. To do this, a function called *extend_data* was created which takes the dataframe from the cleaning process and extends them.

Interpolation

As there was a lot of missing data, which may be due to instrument failure, communication issues, human errors, or site closing too soon, simply dropping rows or columns in the data with missing values will lead to the loss of a lot of critical data which could potentially affect the analysis of the data. Thus, the missing data needs to be handled and could be handled using various interpolation methods based on some assumptions because interpolation methods allow for the estimation of missing values by utilizing the spatial and temporal relationships between data points [1]. When interpolating the weather data, spatial and temporal interpolations using linear interpolation as a tool to estimate values were used. Spatial interpolation is a method of estimating values at unsampled locations within an area based on the values of sampled locations. Temporal interpolation is a method of estimating values at unsampled times within a time series based on the values of sampled times . [2][3]

Linear interpolation was used as the tool to drive the spatial and temporal interpolations As weather data typically exhibits a high degree of temporal correlation, meaning that nearby observations are likely to be highly correlated. This temporal correlation makes linear interpolation a good method for estimating missing values, as it can accurately capture the trends and patterns in the data. Overall, linear interpolation can be a good approach to fill missing weather data as it can accurately estimate missing values by using nearby observed data points to form a linear function that approximates the missing data. It is a simple and computationally efficient method that does not require complex statistical models or large amounts of data.[4]

The following can be said after looking at the characteristics of the weather data:

- Temperature (tmax(degC) , tmin(deg(C)) is a variable that varies in both space and time. Generally, temperature decreases with increasing altitude and distance from the coast, therefore spatial interpolation can help account for these geographical differences. On the other hand, temperature also exhibits a distinct seasonal pattern,

with warmer temperatures in the summer and colder temperatures in the winter. As a result, temporal interpolation can aid in accounting for these temporal changes. [5]

- Rainfall (rain(mm)) is another variable influenced by both spatial and temporal fluctuations. Rainfall can vary greatly based on location (for example, coastal vs. inland locations) and topography (for example, mountains vs. plains), hence spatial interpolation can help account for these spatial changes. Rainfall has distinct seasonal patterns, with more falling in the winter and less in the summer, therefore temporal interpolation can help account for these temporal fluctuations. [6][8]

- Variables such as air frost (af(days)) and sunshine (sun(hours)) are predominantly affected by seasonal variations, and they have a high degree of temporal variability but little spatial variability. As a result, temporal interpolation may be preferable for certain variables. [7][8]

For rainfall and temperature features, a combination to spatial and temporal interpolation was used to account for spatial and temporal changes and then temporal interpolation for sunshine hours and air frost. The function, *interpolate_extended_data*, does this.
The spatiotemporal interpolation performed on the columns *'tmax(degC)', 'tmin(degC)', 'rain(mm)'* is done by creating a grid of points based on the longitude, latitude, year, and month columns of the data passed to it, and interpolates any missing data using the *griddata* function from the NumPy library with a linear method. Also, the temporal interpolation performed on the columns *'af(days)', 'sun(hours)'* using the *interpolate* function from pandas with a linear method to handle the missing columns (refer to *interpolate_extended_data*).

Some assumptions for this approach:

- Nearby stations have similar weather patterns, which appears to be a reasonable assumption because weather patterns are typically spatially and temporally correlated.
- The data can be interpolated linearly in both space and time.
- The missing values can be predicted based on the linear trend of the available data over time.
- The data has missing values the needs to be interpolated which are represented by nan values.

Some Implications:

- The missing values are replaced with estimated values from the interpolation, which may affect the accuracy of the data as the estimated values are not the values.
- The interpolated values may not be exact and may have some degree of error.
- The interpolated values may be influenced by the values of nearby stations and times, which may introduce some bias.
- Should there be non-linear trends or strong seasonal patterns in the data, the above will fail or produce incorrect results.
- Extrapolation is not performed, so the interpolated values are only valid within the range of observed values.

Some Improvements:

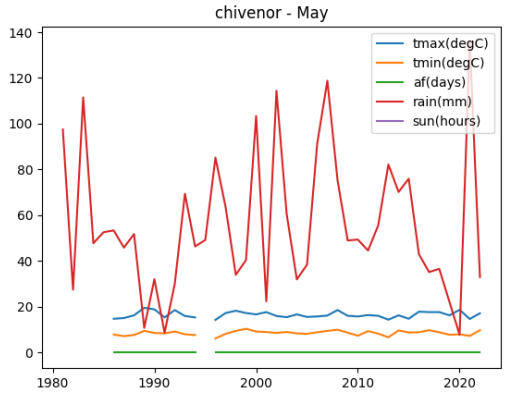
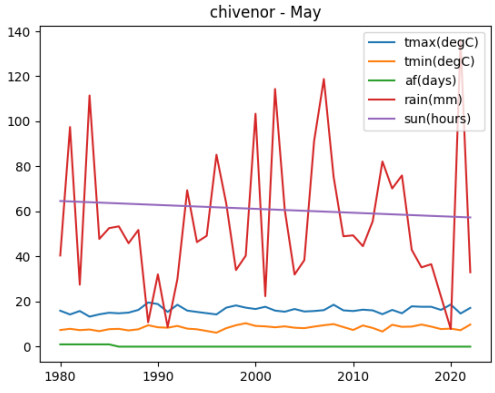
- Performing additional data quality checks before interpolation to identify any data that may be unreliable or erroneous like removing outliers across the stations' data.
- Incorporating additional data sources, such as satellite data or ground-based measurements as this will improve the accuracy and coverage of the interpolation.
- Using a different interpolation method that does not assume linearity, such as a spline or kriging and/or experimenting with different interpolation methods to find the most appropriate one for the data.

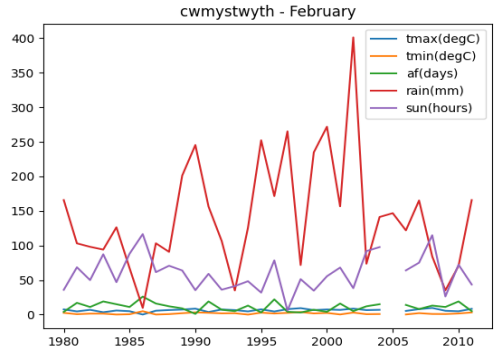
Alignment

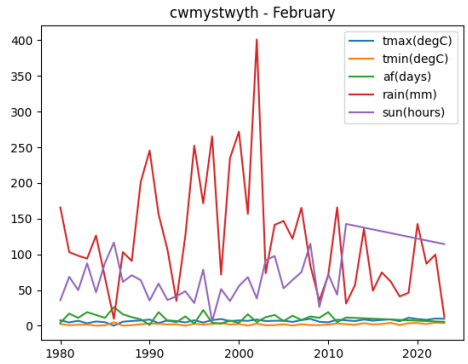
Since all the stations started from different years, aligning the data was the next step after attempting to handle the missing data. As clustering and classification will be performed on the weather data, it makes sense to align the data to include only common years for all the different stations, ensuring that the data is consistent and comparable. This makes it possible to spot weather data patterns and trends that are that are consistent across different stations. Additionally, it aids in reducing the dimensionality of the data, which facilitates analysis and results interpretation. After aligning, I checked to the data to make that for all the stations, every year had data for 12 months except for March 2023 which had 3 months. The function, *align_data* does this.

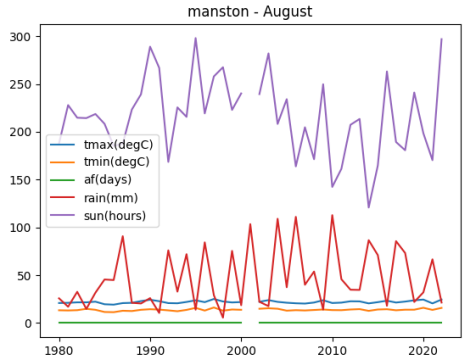
The data is then saved in a csv file called, cleaned_aligned_data, which is used for this assessment. The data consists of weather data from Jan 1980 to March 2023 for each station.

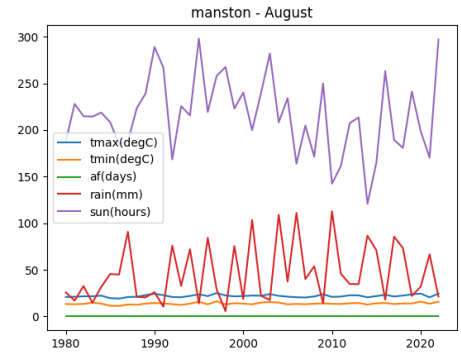
The data was then visualised to see the result of the full pre-processing steps applied. Below shows some plotted graphs for some stations according to months before and after the cleaning using same time period.

Before pre-processing	After full preprocessing
	









Looking at the first graph, i.e. chivenor May, it can be seen that from 1980-2023, there was no recorded sun hours (left), but in the image on the right, it can be also be seen that the interpolated step attempted to predict what the sun hours would look like using data from nearby stations and other available information, which is a linear graph, however, that would not be the case in the real life as sunshine hours does not depict linear trend over time.

Overall, from the above images, it could be seen that the interpolation step does a relatively fair job filling in estimating the values of missing data for the various features especially when the missing data are sandwiched between known data points and using data from nearby stations and does poorly otherwise. It is also difficult to tell whether these estimates are closer or farther from the what the actual (looking at satellite data or other data sources) would have been if those assumptions made above did not hold.

Because of the abovementioned, it would have been great to use a different interpolation method that does not assume linearity, such as a spline or kriging and/or experimenting with different interpolation methods to find the most appropriate one for the data or incorporating additional data sources such as satellite data to ensure that the missing values are as close to their true values.

Task 1: Clustering

In this section, the task was to look only at the weather data excluding the latitude/longitude of the weather station and see if a clustering algorithm can be used to group the stations into groups that have similar weather?

The code for this section is in cluster.py. Hence, to attempt to group the stations based on their similar weather conditions, the average data for each station across the same period was used.

Using the average of the weather readings for each station helps us to obtain a representative value that summarizes the overall weather pattern for that station and that taking the average helps to mitigate the effect of outliers or extreme values that may occur in some individual readings and provides a more stable and reliable measure of the station's weather conditions. Moreover, using the average makes it easier to compare and cluster the stations based on their similar weather patterns so they can be grouped based on their similarities which can provide insights into larger-scale weather patterns and climate trends. Although using average give a general overview of the weather pattern for a particular station, it may not capture variations or extreme events.

Hierarchical clustering was used to cluster the stations. Hierarchical clustering is a clustering algorithm that groups similar data points into clusters based on their similarity or distance. It can create more complex shaped clusters that were not possible with other clustering algorithms like in Gaussian mixture models (GMM) where the clusters are assumed to be Gaussian, which may not be appropriate for all datasets which in this case, our weather data may not necessary be Gaussian [9][10]. In addition, it does not make any assumptions about the shape of the clusters and does not need the number of clusters to be defined beforehand, unlike KMeans clustering which assumes that the shape of the clusters is spherical, and the number of clusters needs to be specified [11]. Hierarchical clustering assumes that the data points are continuous and that the distance between them can be measured using a distance metric.

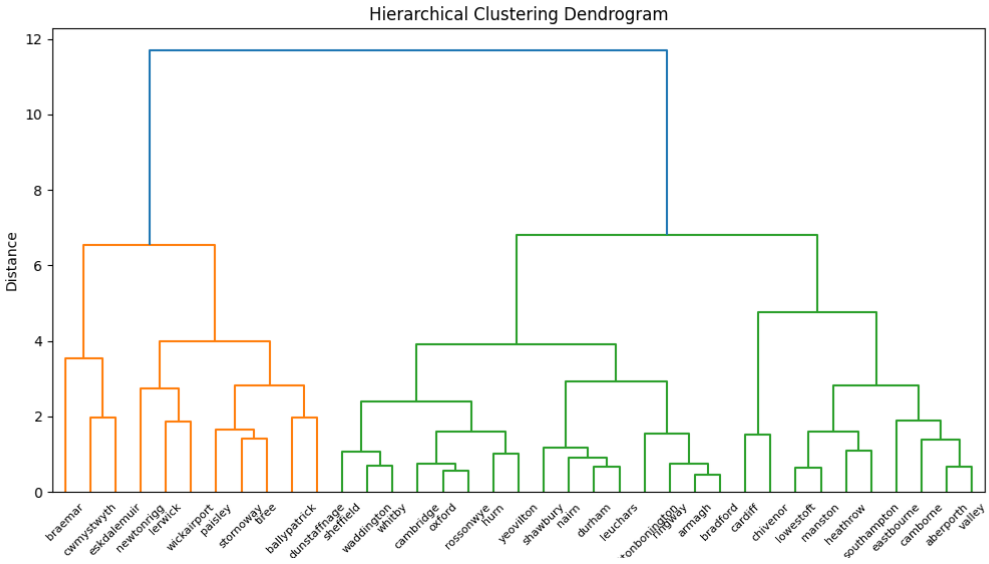
One of the main advantages of this algorithm is that it provides detailed information about which observations are most similar to each other and this level of detail can be useful in exploratory data analysis. Another advantage is that it allows for the visualization of results using dendrograms or tree-like structures which are somewhat easy to interpret and understand [10]. However, hierarchical clustering can be computationally expensive and may not be suitable for large datasets due to high time and space complexity. Additionally, there the results from a hierarchical clustering can be sensitive to the choice of distance metric and linkage method chosen even for the same dataset.[9][10]

Before performing the clustering, the temperatures, rainfall, air frost and sunshine hours columns of the data were selected and were standardised as these features had different units. This is because hierarchical clustering can be sensitive to scale, because it is based on distance measures between data points. And when the variables or features have different scales, the distance measure can be dominated by the variables with larger scales, leading to biased results. Therefore, standardizing helps by scaling the variables to have zero mean

and unit variance, which puts all variables on the same scale and ensures that no variable dominates the distance measure [13]. The clustering was then performed on the data using the Ward method with Euclidean metric distance to minimize the variance of the clusters.

Observations and Analysis

Performing the clustering on the entire data for all the stations revealed the following:



From the above, there are two main clusters, which is then broken down into smaller clusters revealing how similar these stations are based on the weather data.

Considering the two main clusters:

The first consist of stations such as Ballypatrick, Braemar, Cwmystwyth, Dunstaffnage, Eskdalemuir, Lerwick, Paisley, Stornoway, Wick Airport, Tiree, and Newton Rigg. These stations are mostly located in Northern Ireland, Scotland, and Northern England. First cluster center values are [tmax=11.5907229 , tmin=5.59855092, air frost= 3.95037844, rain=106.92196181, sun hours = 88.41635966].

The second cluster includes Aberporth, Armagh, Bradford, Camborne, Cambridge, Cardiff, Chivenor, Durham, Eastbourne, Heathrow, Hurn, Leuchars, Lowestoft, Manston, Nairn, Oxford, Ringway, Ross-on-Wye, Shawbury, Sheffield, Southampton, Sutton Bonington, Valley, Waddington, Whitby, and Yeovilton. These stations are located in Southern England, Wales, and Northern Ireland. Second cluster center values: [tmax = 13.85491918, tmin= 6.79883437, air frost= 2.91840818, rain= 63.44122437, sun hours=120.54876403].

Looking at the cluster center values, it can be seen that stations in the first cluster have relatively higher air frost and higher rain values with relatively low temperature and sun hours values than those in second cluster. This implies that these stations tend to which tend to have colder and wetter weather with mild sunshine compared those in the second.

Those in the second, on the other hand, tend to have relatively higher temperature and sun hours values with relatively low air frost and rain values than those in the first cluster. This implies that the stations in this cluster experience warmer and sunnier weather conditions than the stations in the first cluster.

Overall, the clustering results using the Ward method appears to be consistent with the weather patterns and geographical locations of the stations in the UK, as the first clusters mostly has station located in the Northern part of the UK which tends to have colder and wet weather conditions and the second has stations located in the Southern part of the UK which tends to have milder and warmer weather conditions.

Task 2: Classification

In this section, the task was to attempt to predict whether 5 stations fall in the Northern Third of the UK, Central Third of the UK or Southern Third of the UK using only the weather data.

The code for this part is in classification.py. To attempt this task, a K-fold validation was performed on three classifiers and the best model based on the accuracy and f1 score was selected and tested on the 5 stations data.

Before performing the K-Fold and the prediction, a couple of things were done which are listed below.

Because the data was to be predicted on some targets, those were created with the help of the mostly northerly latitude and the most southerly latitude values given in the question and a function called *divide_UK_stations_into_regions*. This function takes the number of the regions we have to divide UK into, the labels to be given to each region and the dataframe to append the region for each instance based on its latitude. In this function, it find the difference between northerly latitude and the most southerly latitude values and divide that by the number of regions to get the intervals, then it creates a list of latitude bands and with labels and pandas cut function, the dataset is divided into regions.

The next thing is to get out test set, so the last five stations and their weather data were obtained. This was done with the help of a function called *divide_station_data_into_two*, which takes the data and divide it into two datasets, one with the last 5 stations data and the other is the dataset for every other station excluding the last five ones.

The next step was to perform the K_fold and make some predictions on the test set. In this case, the function *perform_K_fold* does that. The function, *perform_K_fold***,** performs K-fold cross-validation using three classifiers: Logistic Regression, Support Vector Machines (SVM), and Random Forest on a given training dataset, evaluates each classifier's performance using accuracy and F1 score, and then applies the classifiers to a test dataset to predict the target variable's classes. The logistic regression has parameters:

solver=sag, multi_class=multinomial, random_state=0, max_iter=5000 because when the max_iter was less than 5000, it converged quikly so the coe and the sag solver because it can handle large datasets.
All the three classifiers had a random_state of 0 and setting the random state to 0 will ensure that the same sequence of random numbers is used to split the data into training and testing sets, as well as to initialize the weights of the model. It can be useful for comparing the performance of different classifiers or for reproducing a specific result.

First, the function selects the columns that are going to be used as features and extracts the values of the features and target variable from the input training and test dataframes. Then, it encodes the target variable using *LabelEncoder* because the target variable, region, is categorical. Many machine learning models, including logistic regression, SVM and random forest, require the target variable to be in numerical form to learn from the data so encoding the target variable converts the categories into numeric labels to allows the models to learn from the data [14]. Features of the data are also standardardized using *StandardScaler* because they are measured on different scales, more weight will be given to the features with larger values. Thus, standardizing the data puts all the features on the same scale, so that they are treated equally by the models [13].

Next, the function defines the three classifiers and creates a K-fold cross-validation object using the specified number of folds. For each fold, it trains the classifiers on the training set and evaluates them on both the training and validation sets using accuracy and F1 score, which are stored in lists for each classifier. After all folds are completed, the function calculates the mean accuracy and F1 score for each classifier on the training and validation sets.
Finally, the function applies the trained classifiers to the test dataset to predict the target variable's classes and prints the accuracy and F1 score for each classifier.

Observations and Analysis

Note: The data was shuffled and the f1_score was calculated based on the weighted average as there the regions were highly imbalanced.

Using a K-value of 10 for the K-fold, From the results from the K-Fold validation and the prediction on test set, the following observations can be made:

LR = Logistic Regression, SVM =Support Vector Machines, and RF=Random Forest

Training mean accuracy: LR:0.594, SVM:0.607, RF:1.000
Training mean f1 score: LR:0.528, SVM:0.531, RF:1.000
Validation mean accuracy: LR:0.593, SVM:0.606, RF:0.661
Validation mean f1 score: LR:0.527, SVM:0.530, RF:0.644

predicting on test set
Test accuracy: LR: 0.200
Test f1 score: LR: 0.067
Test accuracy: SVM: 0.239

Test f1 score: SVM: 0.175
Test accuracy: RF: 0.600
Test f1 score: RF: 0.450

- The training mean accuracy and f1 scores for LR and SVM are relatively low compared to the RF classifier, indicating that they may not fit the training data as well as RF.
- RF has a perfect mean training accuracy and f1 score, which may suggest overfitting to the training data.
- The validation mean accuracy and f1 score for LR and SVM are slightly worse than that of RF, indicating that RF may perform better in generalization to unseen data, at least as measured by the validation set.
- Based on the above, RF may have better generalization performance than LR and SVM on unseen data though it appears to be overfitting the dataset.

  Predicting each classifier on the test set

- The test accuracy and f1 scores for LR and SVM are very low, indicating that these models do not perform well on the test set. In contrast, RF has a much higher test accuracy and f1 score, suggesting that it is the best model for this test set.

Overall, it seems that RF has the best performance on this dataset, but it might be overfitting the training data. LR and SVM have lower performance, but they may generalize better on unseen data. However, with their low test performance, this suggests that they may not be suitable for this test set.

Task 3: Happines and weather data

In this section, the task was to try to answer the question as to whether the weather affects how happy people are using the weather data and the happiness data from http://www.ons.gov.uk/peoplepopulationandcommunity/wellbeing/datasets/personalwellbeingestimatesgeographicalbreakdown. The code can be found in weather_and_happiness_data.py with some helper function defined in helper_q3.py.

The first I did was to grab the data from the link above and explore what it has. The following happiness data were obtained: April_2011-March 2012, April_2012-March 2013, April_2013-March 2013, April_2014-March 2015. Looking at the data, the column of interest was the average(mean) rating for the happiness scale as it gives a single value that represents the overall happiness level for each area and moreover not all the label columns [low, high, and so on] had values.

Then next thing that was done was to find a way to grab the important aspects of the data needed. And for this, a function, *clean_years_happiness_data* was defined. This function takes a path to the excel file which consists of the well-being data and then grabs the happiness data from the happiness sheet into a dataframe. After that, the data is being cleaned where there is the renaming of columns, unnecessary rows and columns are

dropped, important columns (the area names, the county and county town and average happiness rating] are obtained. Finally, the function returns the cleaned and processed data.

As we were to merge the happiness data with the weather data, with the help of a couple searches, a list of the weather stations, their regions and county names were obtained and complied into a excel sheet called stations_counties.xlsx. Using this compiled data and a helper function, *get_station_and_happiness_data_for_a_particular_year,* it was possible to compile the list of weather stations and the happiness data for a particular year and month. For instance, if the happiness data for April 2011 to March 2012 dataframe is passed to the function. It will do the following: it reads excel file containing station and county data and returns a new dataframe with the station and corresponding happiness data April 2011 to March 2012 year. It first reads in the station and county data and fills in any missing values with the string None. It then iterates over each row of the station and county data, extracts the region and county names, and searches for corresponding happiness data in the happiness data dataframe. If a match is found, it extracts the corresponding data and stores it in the current row and column of the station and county dataframe. If a match is not found, it stores a null value in the current row and a new column with col_ prefix and the original column name. Finally, it returns the station and county dataframe with the corresponding happiness data.

With 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, a function called *full_happiness_data*, which takes the a dataframe consisting of the station and happiness data for a time period like April 2011 to March 2012, and the start year and end year which in this case will 2011 and 2012 respectively, the happiness data for each station was extend on a monthly basis. So, the data now had values April 2011 to March 2012 for each station.

With the help of all above helper functions and the assumption, a full happiness data from Jan 1980 to March 2023 was obtained, where in the data, Jan 1980 to March 2011 used value of April 2011 to March 2012, and values from April 2015 to March 2023 uses value of April 2011 to March 2012 for each station. Having this data, it was easier to merge the weather data and the happiness data as they both common column, station and started from Jan 1980 to March 2023.

With the merged weather and happiness data, a correlation matrix was plotted to see how the weather and average happiness mean rating correlates which could help to answer whether the weather affects how happy people are.

General observations and conclusion

Plotting a correlation matrix for the features of the data(all instances from Jan 1980 to March 2023) which are tmax(degC), tmin(degC), af(days), rain(mm), sun(hours), average happiness rating. This is correlation matrix.

Weather Variables	tmax(degC)	tmin(degC)	af(days)	rain(mm)	sun(hours)	average_happiness_rating
tmax(degC)	1.000000	0.929999	-0.688465	-0.206636	0.609547	-0.069548
tmin(degC)	0.929999	1.000000	-0.700333	-0.097409	0.505949	-0.053552
af(days)	-0.688465	-0.700333	1.000000	-0.003713	-0.369917	0.007726
rain(mm)	-0.206636	-0.097409	-0.003713	1.000000	-0.370023	0.088171
sun(hours)	0.609547	0.505949	-0.369917	-0.370023	1.000000	-0.108457
average_happiness_rating	-0.069548	-0.053552	0.007726	0.088171	-0.108457	1.000000

From the correlation matrix, the following can be deduced between the weather data and happiness of people:

Temperature variables (tmax and tmin) have a strong positive correlation with each other, with a correlation coefficient of 0.93. This suggests that when the maximum temperature is high, the minimum temperature tends to be high as well.

Temperature variables have a strong negative correlation with the number of air frost days (af), with correlation coefficients of -0.69 and -0.70. This indicates that when temperatures are high, the number of air frost days tends to be low.

Sunshine hours have a moderately strong positive correlation with the maximum temperature (0.61) and a moderate negative correlation with the number of air frost days (-0.37). This suggests that when there are more sunshine hours, temperatures tend to be higher, and the number of air frost days tends to be lower.

Rainfall has a weak negative correlation with the number of air frost days (-0.003) and a weak positive correlation with the average happiness rating (0.09). This suggests that rainfall may have a minimal impact on air frost days, and that people may be slightly happier on days with more rainfall. This might be true because in some cases and for some people because of the cozy feeling of being indoors during a rainy day.

There is a moderately strong negative correlation (-0.11) between sunshine hours and the average happiness rating. This suggests that people may be less happy when there are more hours of sunshine. This is unexpected as usually people are happier when there are more sun hours.

There is a weak negative correlation between temperature variables and the average happiness rating, with correlation coefficients of -0.07 and -0.05. This suggests that as temperatures increase, people may feel slightly less happy on average. However, since the magnitude of the correlation is quite small, so strong conclusions cannot really be made from the data.

Overall, these observations suggest that weather can have a small to moderate effect on people's happiness, but it is only one of many factors that can influence happiness. Other

factors such as personal circumstances, social relationships, and economic factors can also play a significant role in determining people's happiness.

Task 4: Running the codes

Important note: because the data used in all 3 tasks uses the interpolated and aligned dataset from the pre-processing steps, all the interpolation assumes still holds.

Explanations for what code does has been explained in each tasks section and the codes also content comments to aid in understanding how they work should their description in the report not be somewhat clear.

The python used for this assignment was 3.10.8. Kindly install any package that is necessary for the code for run.

1. Make sure all the additional files are in the same place specifies in the code or else change the paths in the code so you can run. The python files and the datasets and other files are in the same directory and/or folder as specified in the code.

2. For the clustering.py file, the number of clusters, k can be change. The only thing is to ensure that the cleaned_aligned_data.csv file is in the same directory as the cluster.py file.

3. For the classification.py, the num_regions and the labels at the end of the file can be changed, however, ensure that the num_regions values and the length of the labels are the same. The divide_UK_stations_into_regions functions created the latitude bands for labelling in the ascending order so ensure that the labelling is done as such else different values will be obtained. Th number of folds can also be change in the perform_K_fold function.

4. Provided all the files are in the same directory as specified in the code, then running python3 <name of the file> in the terminal should either print graphs or results at the terminal.

References

[1] *A hybrid method for interpolating missing data in heterogeneous spatio ...* (no date). Available at: https://www.researchgate.net/publication/293804041_A_Hybrid_Method_for_Interpolating_Missing_Data_in_Heterogeneous_Spatio-Temporal_Datasets/fulltext/56db8d6c08aee73df6d2ba10/A-Hybrid-Method-for-Interpolating-Missing-Data-in-Heterogeneous-Spatio-Temporal-Datasets.pdf

[2] *Interpolation* (no date) *Interpolation - an overview | ScienceDirect Topics*. Available at: https://www.sciencedirect.com/topics/earth-and-planetary-sciences/interpolation.

[3] Ujaval (2022) *Temporal gap-filling with linear interpolation in gee*, *Spatial Thoughts*. Available at: https://spatialthoughts.com/2021/11/08/temporal-interpolation-gee/

[4] *(PDF) filling missing values in weather data banks - researchgate* (no date). Available at: https://www.researchgate.net/publication/224703245_Filling_missing_values_in_weather_data_banks

[5] *Interpolation methods for climate data - knmi* (no date). Available at: https://cdn.knmi.nl/knmi/pdf/bibliotheek/knmipubIR/IR2009-04.pdf

[6] *A hybrid method for interpolating missing data in heterogeneous spatio ...* (no date). Available at: https://www.researchgate.net/publication/293804041_A_Hybrid_Method_for_Interpolating_Missing_Data_in_Heterogeneous_Spatio-Temporal_Datasets/fulltext/56db8d6c08aee73df6d2ba10/A-Hybrid-Method-for-Interpolating-Missing-Data-in-Heterogeneous-Spatio-Temporal-Datasets.pdf

[7] *State of the UK climate 2020 - kendon - wiley online library* (no date). Available at: https://rmets.onlinelibrary.wiley.com/doi/10.1002/joc.7285

[8] Di Cecco, G.J. and Gouhier, T.C. (2018) *Increased spatial and temporal autocorrelation of temperature under climate change*, *Nature News*. Nature Publishing Group. Available at: https://www.nature.com/articles/s41598-018-33217-0/

[9] *What is hierarchical clustering?* (no date) *KDnuggets*. Available at: https://www.kdnuggets.com/2019/09/hierarchical-clustering.html

[10] Pai, P. (2021) *Hierarchical clustering explained*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/hierarchical-clustering-explained-e59b13846da8

[11] Dabbura, I. (2022) *K-means clustering: Algorithm, applications, evaluation methods, and drawbacks*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

[12] Jovana *et al.* (2018) *Agglomerative hierarchical clustering*, *Datanovia*. Available at: https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/

[13] Serafeim Loukas, P.D. (2023) *How scikit-learn's standardscaler works*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/how-and-why-to-standardize-your-data-996926c2c832

[14] Gong, D. (2022) *Top 6 machine learning algorithms for classification*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/top-machine-learning-algorithms-for-classification-2197870ff501