Assessment 3

This assessment is a practical exercise. You should undertake the activities described below and write these up in a report.

Since you only have a few weeks to work on this problem and since several of the ideas covered here could have been new to you on this course we are going to work on some fairly straightforward data and try to answer some general questions without worrying *too* much about statistical rigour. In particular, the datasets are not particularly big, which has the advantage that they are manageable to work with, but it has the disadvantage that in some cases, the machine learning algorithms will not have so much data to work with, and so the results that you obtain may not be particularly statistically significant.

You are going to work with two publicly available data sets:

1. Historic weather/climate data from the UK, and
2. Self-reported happiness statistics from the UK

This assessment is fairly open-ended: I encourage you to explore these datasets and see what interesting things you can find. You may use Python, R, or any other analysis tools that you are comfortable with. There is no one "right" answer. In writing up your work, you should try to justify *why* you did things the way you did as well as just describing the steps that you took and the results that you obtained. The process that you go through is as important as the final result, so you should make sure that you clearly describe the steps that you have taken in your report. It is the nature of this kind of work that you will have to make some assumptions and approximations. These should be stated and, where possible, justified.

The assessment is split into four parts. Parts 1, 2 and 3 make up 80% of the mark and are compulsory. Part 4 is split into two options (a) and (b). Please attempt **either** Part 4a **or** Part 4b. The percentages in brackets in the headings below indicate the weight associated with each part.

After each of Parts 1 to 4 have been described below, there are some general hints, which might help you undertake some of the parts of the assessment.

The Data

This is real life data. You will have to do a fair amount of tidying and preprocessing of this data before it can be fed into a learning algorithm. This is an important part of the process and you will get credit for doing this part of the procedure.

Dataset 1 can be obtained from https://www.metoffice.gov.uk/research/climate/maps-and-data/historic-station-data.

Note that if your browser blocks pop-up windows, the drop-down menu of weather stations will do nothing, and you'll have to click on the red dots on the map to get the data for each weather station. If you want to automate the download process, you might find the data in stations.txt (available from Learn) helpful. This file contains a list of strings which correspond to each of the stations, and they can be used in URLs to go directly to the data. For example, for the station "nairn", you can find the data at: http://www.metoffice.gov.uk/pub/data/weather/uk/climate/stationdata/nairndata.txt.

Dataset 2 can be obtained from http://www.ons.gov.uk/peoplepopulationandcommunity/wellbeing/datasets/personalwellbeingestimatesgeographicalbreakdown. This contains four sub-datasets (life satisfaction, worthwhile, happiness and anxiety). For the purposes of Part 3 below, you should use the ***happiness*** data. If you choose to do part 4a, then you could look at the other sub-datasets too. This dataset includes data at the level of local government regions, but that is probably more detail than we need for this exercise. It is sufficient to use data at the level of "Region". The file regions.txt (available from Learn) lists the regions and gives a latitude and longitude which can be used to describe their locations. The regions are identified by the Area Code.

Part 1 (25%)

Look at the weather data for each of the weather stations.

Looking only at the weather data (that is, don't include the latitude/longitude of the weather station) can you use a clustering algorithm to see if these stations can be clustered into groups that have "similar" weather? Note that you will have multiple weather readings for each station. You could pick the most recent, an average, an extreme, or you could consider all of the points individually and look for clusters in the individual observations. You should try to justify your choice.

Part 2 (25%)

Now let's turn this into a classification problem.

You should exclude 5 of the weather stations from this set (You could do this by picking the 5 last stations alphabetically).

Can you predict whether they fall in the Northern Third of the UK, Central Third of the UK or Southern Third of the UK using only the weather data? You have latitude data for all the weather stations, so that can give you the classification for each of the weather stations in your training set. To determine the latitude of the lines dividing the UK into three, you should note that the most northerly point has latitude 60.9 and the most southerly point has latitude 49.9.

Part 3 (30%)

The Office for National Statistics[1] collects and publishes data on how happy people think they are. This is self-reported data, so it may well have some inherent biases, but it is good enough for us to draw some general conclusions. Happiness is measured on a scale of 0-10.

Let's try to answer the question as to whether the weather affects how happy we are.

Try to join the weather station data with the happiness data. You will only be able to do this in a coarse-grained way, because the places where there are weather stations do not correspond directly to the census areas used in the happiness surveys. You could use bands of latitude to do this, or you could put a grid over the county and look at which grid cell a weather station or census region falls in. Don't worry too much about the fact that weather data and census data don't cover exactly the same time periods, we can assume 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.

For the final part of this assessment you can try *either* Part 4a *or* Part 4b below.

Part 4a (20%)

Explore this data further in any way that you want.

You might want to:

1. Look for any temporal effects that we neglected earlier: Are people happier in years when the weather is better?
2. Perform clustering or classification on the happiness data on its own.
3. Join these data sets to any other publicly available data set and look for any patterns.
4. Look at an alternative geographical decomposition of the UK from Part 2. What happens if you split the country East to West? What happens if you choose your three regions to have an equal number of weather stations in each region?

[1] https://www.ons.gov.uk

Part 4b (20%)

Try to automate parts 1 to 3 as much as possible. Write scripts and/or programs that follow each of the steps you did to find your answers to parts 1 to 3. Try to parameterise the scripts so that the whole process could be easily re-run with a small difference, say, by dividing the UK into four equal regions instead of three.

Submit your programs/scripts in a zip file (or similar) and include in your report a brief description of how they can be run and what they do.

Some Hints

You almost certainly have more fine-grained data here than you need to answer some of the questions. You will probably want to reduce the data in some way before using it for clustering or classification. You could do this by taking averages, taking extreme values (like maxima) or by selecting a (hopefully) representative subset of the data.

You can pre-process your data in several ways. You can do this manually, but to speed things up, you will probably want to use Python, R, a database system, or the command line. Here are some potentially useful command line examples.

Some Useful Command Line Functions

You can use the **curl** command to download data from the command line, e.g.:

```
curl https://www.metoffice.gov.uk//pub/data/weather/uk/climate/stationdata/aberporthdata.txt >
aberporthdata.txt
```

You can loop over a number of files with a command like

```
for place in aberporth armagh ballypatrick; do curl
'https://www.metoffice.gov.uk//pub/data/weather/uk/climate/stationdata/'$place'data.txt' >
$place'data.txt'; done
```

You can remove every occurrence of the character '*' from a file with a command like:

```
sed 's/*//g' armaghdata.txt
```

You can use the grep command to match certain patterns. The following command matches every line where the first column contains only numerical digits and the second column (space separated) contains the value '6':

```
grep '^ *[0-9]\+ \+6' armaghdata.txt
```

You can remove the first 7 lines of a file with a command like

```
awk ' NR>7 ' oldfile.txt > newfile.txt
```

Practical Introduction to Data Science
Assignment III

Exam # - B066003

May 10, 2020

Part 1

As the datasets required preliminary processing and cleaning, this section is much longer than Part 2 of the report. Cleaned data from this section is used in the next section as well. In order to get the weather datasets, the file **stations.txt** was used. The following code was run:

```
import requests

with open('stations.txt', 'r') as f:
    for l in f:
        url_to_download =
        ↪  "http://www.metoffice.gov.uk/pub/data/ /
        ↪  weather/uk/climate/stationdata/{}data.txt"
        .format(l.strip())
        myfile = requests.get(url_to_download)
        open("{}.txt".format(l.strip()),
        ↪  'wb').write(myfile.content)
```

The 37 data files were downloaded onto my local directory. Then, they were inspected and several symbols were removed - the # and * symbols, the $ sign, and the word "Provisional". This was done with the following 'sed' command, run 4 times (once for each of the 3 symbols and the 1 word):

```
for place in aberporth armagh ballypatrick bradford braemar
 ↪  camborne cambridge cardiff chivenor cwmystwyth dunstaffnage
 ↪  durham eastbourne eskdalemuir heathrow hurn lerwick
 ↪  leuchars lowestoft manston nairn newtonrigg oxford paisley
 ↪  ringway rossonwye shawbury sheffield southampton stornoway
 ↪  suttonbonington tiree valley waddington whitby wickairport
 ↪  yeovilton; do sed -i 's/#//g' ''$place'.txt'; done
```

The $ sign had to be escaped.
Then, the 37 datasets were uploaded to my *Google Drive* and I used *Google Colab* for additional preprocessing. First, I printed the first 10 lines of each file to check for the consistency of the header format. I wanted to put everything into 1 *Pandas Dataframe*, appending 3 columns for each row (in addition to the weather 5 columns) - the station name, the latitude, and the longitude for the station. This would mean that the station name, latitude and longitude are the same and repeated, but would allow for easier grouping and work using only *Pandas*. I noticed that the latitude, longitude could be extracted using the Python *split()* function and I used the file name for the station name. Upon doing that and saving the dataframe, I performed data validation, and saw that the number of columns is higher than expected. I cleaned up some dirty entries from the original files, which were split into more than 5 columns, on my local computer, reuploaded the data and reran the code. Additionally, the code removed the last line "Site Closed" present in each of the 37 *.txt* files. Code used:

```
data_complete = []

for f in os.listdir('/content/gdrive/My
 ↪  Drive/pitds3/weather_data'):
  station_name = "---"
  latitude = "---"
  longitude = "---"
  latitude_seen = False
  longitude_seen = False
  deg_seen = False
  with open('/content/gdrive/My Drive/pitds3/weather_data/' +
   ↪  f) as myfile:
    for line in myfile:
```

```
      line = line.lower().strip()
      if deg_seen and line != 'site closed':
        data_complete.append(line.split() + [latitude,
        ↪  longitude, f.split('.')[0]])
      else:
        line = line.split()
        if line[0] == 'degc':
          deg_seen = True
        if not latitude_seen:
          if 'lat' in line:
            latitude_seen = True
            for i, el in enumerate(line):
              if el == 'lat':
                latitude = line[i+1]
                break
        if not longitude_seen:
          if 'lon' in line:
            longitude_seen = True
            for i, el in enumerate(line):
              if el == 'lon':
                longitude = line[i+1]
                break
    latitude_seen = False
    longitude_seen = False
    deg_seen = False
```

The *data_complete* list of lists was turned to a *Dataframe* with a *Pandas* function. The columns were renamed:

```
dfs_renamed = df.rename(columns={0: "year", 1: "month",
↪  2:"t_max", 3:"t_min", 4:"af_days",
                                 5:"rain_mm", 6:"sun_hours",
                                 ↪  7:"lat", 8:"long",
                                 ↪  9:"station"})
dfs_renamed.head()
```

	year	month	t_max	t_min	af_days	rain_mm	sun_hours	lat	long	station
0	1947	1	3.3	-0.7	---	34.0	60.0	53.175	-0.522,	waddington
1	1947	2	-0.6	-3.8	---	19.0	24.7	53.175	-0.522,	waddington
2	1947	3	6.2	0.0	---	95.0	73.7	53.175	-0.522,	waddington
3	1947	4	12.6	4.7	---	43.0	161.7	53.175	-0.522,	waddington
4	1947	5	19.8	8.1	---	13.0	209.4	53.175	-0.522,	waddington

There were still some commas left in the *long* column. Those were removed with the Pandas *replace* function.

Next, I had to decide how to perform the clustering. I decided to do it on the mean "weather", i.e. take the mean of every column after grouping by station, and cluster using the resulting 5-dimensional data. This would give us of a general idea of where the weather is similar, based on a variety of factors, not just one. However, I decided that filtering had to be done before the clustering. In particular, the starting and ending point for the measurements for different stations were different. Therefore I filtered the data date range from year 1979 (incl.) to year 1999 (incl.). The means for the different stations would be inconsistent and incomparable with each other if the filtering is not performed, in case climate changes over time. I found the lower bound for the years to be 1979, by grouping the data by stations, and taking the maximum of the earliest (minimum) years to get the starting point for which all stations have data. The year found was 1978 for the Camborne station, however, after inspecting the camborne.txt file, the measurements started from September, therefore I chose the year 1979. I found the year 1999 by grouping by station again, and taking the minimum of the latest (maximum) years, which was year 2000. Finally, the data was grouped by station and the means for the groups was taken:

station	t_max	t_min	af_days	rain_mm	sun_hours	lat	long	t_avg
aberporth	12.301190	7.061508	1.698413	73.168651	128.881349	52.139	-4.570	9.681349
armagh	12.906349	5.833730	3.388889	67.738889	98.476984	54.352	-6.649	9.370040
ballypatrick	11.201770	5.339189	3.230337	104.559848	107.738194	55.181	-6.153	8.255963
bradford	12.426190	5.793651	3.750000	73.244048	101.493254	53.813	-1.772	9.109921
braemar	10.355556	2.815873	8.730159	77.550000	100.346586	57.006	-3.396	6.585714
camborne	13.206746	8.085714	0.869048	91.169841	136.414458	50.218	-5.327	10.646230
cambridge	14.082937	6.136508	3.619048	46.900000	123.485317	52.245	0.102	10.109722
cardiff	14.315476	6.841667	3.126984	95.348810	125.095652	51.488	-3.187	10.578571
chivenor	14.704403	7.833974	1.711538	76.259914	NaN	51.089	-4.147	11.283974
cwmystwyth	11.618952	4.989069	5.024291	152.641129	96.934274	52.358	-3.802	8.306883
dunstaffnage	12.070161	6.037500	2.518072	145.009504	100.353000	56.451	-5.439	9.053831
durham	12.565476	5.126984	4.527778	54.155159	117.720080	54.768	-1.585	8.846230
eastbourne	13.698413	8.385317	1.408730	66.963492	154.348016	50.762	0.285	11.041865

Figure 1: Part of the grouped-by-station dataframe (37 entries in total).

The average temperature was appended to the previous dataframe (calculated by t_min + t_max, divided by 2), which is why it also appears in the grouping. The mean latitudes and longitudes denote the real latitude and longitudes as they are the same for each entry per station. This data was used to perform the clustering. One entry (chivenor) had a NaN value for the sun_hours column, as chivenor's measurements between 1979 and 1999 for the sun_hours were all "- - -"s. This was replaced by the mean of sun_hours for the remaining 36 entries. The columns used for the clustering were the t_max, t_min, af_days, rain_mm and sun_hours columns - 5D data. The average temperature wasn't used as this information was already captured in t_max and t_min.

To perform the clustering the *K-Means* clustering algorithm was used, using a loop between 1 and 15 cluster centers to determine the best value. Sklearn's *StandardScaler* was used on the data to reduce the effect of difference in units(temperature vs rainfall, for example). 5 clusters were chosen based on the graph denoting the sum of squared distances between the points and the center, which showed significant decrease in gain after the 5th cluster center:

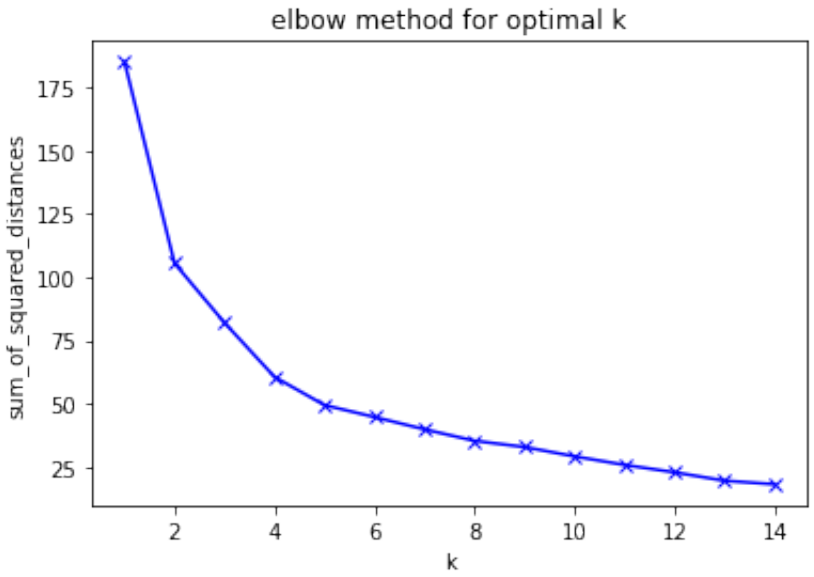

Figure 2: Sum of squared distances between the 37 data points and their closest cluster centers, per number of centers.

Thereafter, the predictions were appended to the dataframe:

station	t_max	t_min	af_days	rain_mm	sun_hours	lat	long	t_avg	pred
aberporth	12.301190	7.061508	1.698413	73.168651	128.881349	52.139	-4.570	9.681349	4
armagh	12.906349	5.833730	3.388889	67.738889	98.476984	54.352	-6.649	9.370040	3
ballypatrick	11.201770	5.339189	3.230337	104.559848	107.738194	55.181	-6.153	8.255963	1
bradford	12.426190	5.793651	3.750000	73.244048	101.493254	53.813	-1.772	9.109921	3
braemar	10.355556	2.815873	8.730159	77.550000	100.346586	57.006	-3.396	6.585714	2
camborne	13.206746	8.085714	0.869048	91.169841	136.414458	50.218	-5.327	10.646230	4
cambridge	14.082937	6.136508	3.619048	46.900000	123.485317	52.245	0.102	10.109722	0
cardiff	14.315476	6.841667	3.126984	95.348810	125.095652	51.488	-3.187	10.578571	0
chivenor	14.704403	7.833974	1.711538	76.259914	118.108037	51.089	-4.147	11.283974	4
cwmystwyth	11.618952	4.989069	5.024291	152.641129	96.934274	52.358	-3.802	8.306883	1
dunstaffnage	12.070161	6.037500	2.518072	145.009504	100.353000	56.451	-5.439	9.053831	1
durham	12.565476	5.126984	4.527778	54.155159	117.720080	54.768	-1.585	8.846230	3
eastbourne	13.698413	8.385317	1.408730	66.963492	154.348016	50.762	0.285	11.041865	4

Figure 3: Part of the grouped-by-station dataframe.

Finally, in order to make sense of the clusters, the Google Maps *gmaps* library was used to show the clusters based on the provided latitude and longitude. Unfortunately (and strangely) *Google Colab* did not support the *gmaps* library, so I had to install it on my local computer. The 5 found clusters were as follows (denoted by different colors):

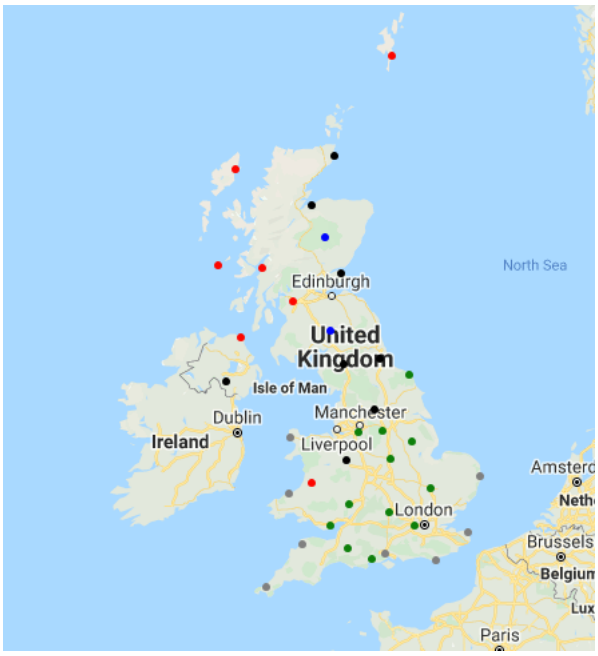

Figure 4: Weather data from 1979 to 1999, grouped by city and clustered by mean average minimum temperature, mean average maximum temperature, mean average days of frost, and mean total sunshine duration, per month.

Interestingly, the clustered data resembles the following chart from the Met Office for the winter temperatures from roughly the same period (1971 - 2000) with my clustered cities falling into different colorings of the map:

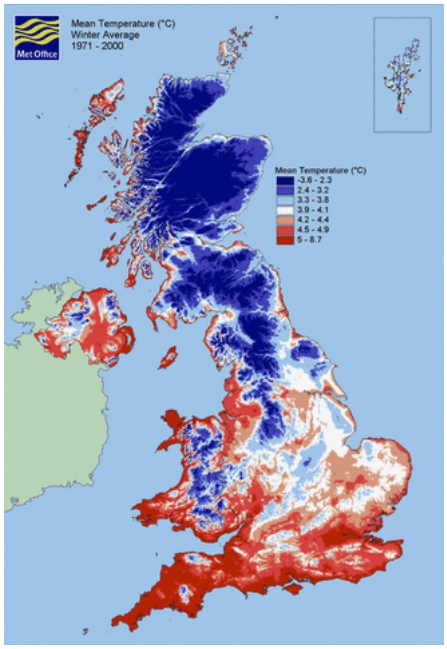

Figure 5: Image taken from http://thebritishgeographer.weebly.com/the-climate-of-the-british-isles.html

However, the mapping is not one-to-one as I've not used only winter data, and I've clustered based on factors, other than mean temperature as well. Additionally, K-Means may not be an ideal clustering method. However, this would be a good way to validate the method by choosing matching data (i.e. only winter data and clustering by mean temperature from 1971 until 2000, although again, some stations don't have measurements before 1978).

Part 2

For this part, I've decided to work with the data I've been transforming throughout Part 1. Hence, I had 2 datasets - 1 "big" dataset from 1978 until 1999 of 9312 entries (much smaller than the full dataset due to the year filtering) and 1 "small" dataset of aggregated data of 37 entries for the 37

stations. The null values of both datasets were not replaced by the means as the data was not yet split into training and test sets. By using the dissecting lines' coordinates, I mapped the station labels from 49.9 to 53.56 degrees latitude to 0 (south), 53.56 to 57.23 to 1 (center), and from 57.23 to 60.9 to 2 (north) and appended the labels as a new column to both the big and the small datasets. After this operation, I explored the number of stations in each area using the small dataset of 37 entries. There were 21 stations in the south area, 12 stations in the center, and 4 in the north.

I chose to split the data in a stratified way manually to ensure similar class distribution within the train and test set and forbid a random sampler to miss underrepresented classes (such as north), and hence the classifier would learn nothing about them. Therefore, I used 16 stations from the south, 9 stations from the center and 3 stations from the north for training (total of 28 stations), and 5 stations from the south, 3 stations from the center and 1 station from the north for testing (total of 9 stations). Hence, training-test split was around 0.75 for each of the 3 classes (0.76 for south, and 0.75 for both center and north).

As I had 2 datasets, I first decided to experiment with the larger one (the one with 9312 entries). I filtered the dataset using the training stations to create a training set of 7044 rows, and a test set of 2268 rows. As each station had approximately the same number of measurements (due to being taken for the same period from 1978 until 1999), this ensured that the proportion of the classes from training to testing for the "big" dataset was also around 0.75 (validated with counts after station groupby on the unsplit data, there were small variations, but the majority of stations had the same number of measurements).

After the splitting was complete, the null values from the training set were replaced with the averages of their respective columns and the null values from the testing set were replaced with the same averages calculated from the training set, in order to avoid looking at test data.

	Unnamed: 0	Unnamed: 0.1	year	month	t_max	t_min	af_days	rain_mm	sun_hours	lat	long	station	t_avg	area
0	1512	6519	1979-01-01	1	2.4	-3.5	23.0	49.6	60.600000	54.768	-1.585	durham	-0.55	1
1	1513	6520	1979-01-01	2	2.8	-1.1	18.0	30.6	49.200000	54.768	-1.585	durham	0.85	1
2	1514	6521	1979-01-01	3	6.4	0.9	13.0	166.0	90.600000	54.768	-1.585	durham	3.65	1
3	1515	6522	1979-01-01	4	10.0	3.4	1.0	44.7	92.500000	54.768	-1.585	durham	6.70	1
4	1516	6523	1979-01-01	5	12.8	4.6	4.0	96.5	150.300000	54.768	-1.585	durham	8.70	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7039	9307	37417	1999-01-01	8	16.4	10.6	0.0	74.6	117.965828	55.181	-6.153	ballypatrick	13.50	1
7040	9308	37418	1999-01-01	9	16.2	10.6	0.0	212.2	117.965828	55.181	-6.153	ballypatrick	13.40	1
7041	9309	37419	1999-01-01	10	12.2	7.8	0.0	78.4	117.965828	55.181	-6.153	ballypatrick	10.00	1
7042	9310	37420	1999-01-01	11	9.4	4.8	1.0	136.6	117.965828	55.181	-6.153	ballypatrick	7.10	1
7043	9311	37421	1999-01-01	12	6.5	1.9	2.0	300.2	117.965828	55.181	-6.153	ballypatrick	4.20	1

7044 rows × 14 columns

Figure 6: Training data from the "large" dataset.

	Unnamed: 0	Unnamed: 0.1	year	month	t_max	t_min	af_days	rain_mm	sun_hours	lat	long	station	t_avg	area
0	0	384	1979-01-01	1	2.1	-3.6	26.0	48.7	61.8	53.175	-0.522	waddington	-0.75	0
1	1	385	1979-01-01	2	2.7	-1.4	25.0	56.0	57.7	53.175	-0.522	waddington	0.65	0
2	2	386	1979-01-01	3	7.6	1.2	7.0	68.8	87.7	53.175	-0.522	waddington	4.40	0
3	3	387	1979-01-01	4	11.4	3.9	2.0	42.8	124.3	53.175	-0.522	waddington	7.65	0
4	4	388	1979-01-01	5	14.6	5.9	3.0	102.1	189.4	53.175	-0.522	waddington	10.25	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2263	5791	24673	1999-01-01	8	19.1	11.3	0.0	45.4	153.4	55.846	-4.430	paisley	15.20	1
2264	5792	24674	1999-01-01	9	17.8	11.5	0.0	143.7	118.7	55.846	-4.430	paisley	14.65	1
2265	5793	24675	1999-01-01	10	13.1	7.6	0.0	59.6	82.1	55.846	-4.430	paisley	10.35	1
2266	5794	24676	1999-01-01	11	10.0	4.5	0.0	184.2	61.9	55.846	-4.430	paisley	7.25	1
2267	5795	24677	1999-01-01	12	6.4	0.9	9.0	297.1	40.1	55.846	-4.430	paisley	3.65	1

2268 rows × 14 columns

Figure 7: Test data from the "large" dataset.

The relevant feature columns were selected (i.e. the columns t_max, t_min, af_days, rain_mm and sun_hours) and the "area" column was used as the label. The Sklearn *StandardScaler* was fitted on the training set and the training and test sets were transformed appropriately using the fitted scaler.

The Sklearn out-of-the-box *MLPClassifier* was used for fitting on the training and predictions on the test set, as this classifier is able to find non-linear decision boundaries and is a relatively simple neural network architecture with predefined and well-performing hyperparameters. After fitting on the training set, I used the classifier to make predictions on the test set and got an accuracy of only around 53% on the 2268 entries.

My first idea to bring the accuracy up was to find the most common label per station and use this to make the prediction (i.e. majority voting). This would've brought up the accuracy if the classifier were generally correct most of the time per station (i.e. it predicted the correct label for the individual measurements more often than every incorrect label). I found the most common labels by appending the already extracted predictions to the testing dataframe, grouping by station, and selecting the most common predicted area element per station. However, the most common label turned out to be 0 (or south) for all of the stations. Therefore, I considered other approaches as well. Because the classes were unbalanced, I thought about adding to the "north" class by using more measurements for the 4 "north" stations (i.e. include data outside of the 1978 - 1999 window). I thought about trying to use other types of scalers and classifiers as well.

However, before doing any of these things I tried to use the "small" dataset first to fit a classifier and make predictions despite the training set consisting of only 28 entries (1 entry per station, with the features denoting the means of the measurements) and the test set consisting of only 9 entries (selected stations for training and testing were the same as in the "large" dataset - a stratified split, those stations were selected using the "small" dataset in the first place).

	Unnamed: 0	station	t_max	t_min	af_days	rain_mm	sun_hours	lat	long	t_avg	area
0	0	aberporth	12.301190	7.061508	1.698413	73.168651	128.881349	52.139	-4.570	9.681349	0
1	1	armagh	12.906349	5.833730	3.388889	67.738889	98.476984	54.352	-6.649	9.370040	1
2	2	ballypatrick	11.201770	5.339189	3.230337	104.559848	107.738194	55.181	-6.153	8.255963	1
3	3	bradford	12.426190	5.793651	3.750000	73.244048	101.493254	53.813	-1.772	9.109921	1
4	4	braemar	10.355556	2.815873	8.730159	77.550000	100.346586	57.006	-3.396	6.585714	1
5	5	camborne	13.206746	8.085714	0.869048	91.169841	136.414458	50.218	-5.327	10.646230	0
6	6	cambridge	14.082937	6.136508	3.619048	46.900000	123.485317	52.245	0.102	10.109722	0
7	7	cardiff	14.315476	6.841667	3.126984	95.348810	125.095652	51.488	-3.187	10.578571	0
8	8	chivenor	14.704403	7.833974	1.711538	76.259914	117.098832	51.089	-4.147	11.283974	0
9	9	cwmystwyth	11.618952	4.989069	5.024291	152.641129	96.934274	52.358	-3.802	8.306883	0
10	10	dunstaffnage	12.070161	6.037500	2.518072	145.009504	100.353000	56.451	-5.439	9.053831	1
11	11	durham	12.565476	5.126984	4.527778	54.155159	117.720080	54.768	-1.585	8.846230	1
12	12	eastbourne	13.698413	8.385317	1.408730	66.963492	154.348016	50.762	0.285	11.041865	0

Figure 8: Training data from the "small" dataset (average temperature was never used).

	Unnamed: 0	station	t_max	t_min	af_days	rain_mm	sun_hours	lat	long	t_avg	area
0	23	paisley	12.572222	5.955556	3.257937	105.525794	102.270635	55.846	-4.430	9.263889	1
1	28	southampton	14.739286	7.429762	2.483871	63.473016	137.521818	50.898	-1.408	11.084524	0
2	30	suttonbonington	13.540476	5.809921	4.150794	50.744444	116.215476	52.833	-1.250	9.675198	0
3	31	tiree	11.521032	6.586508	1.293651	107.866667	116.738492	56.500	-6.880	9.053770	1
4	32	valley	13.043651	7.441667	1.690476	69.175397	133.913492	53.252	-4.535	10.242659	0
5	33	waddington	13.150397	5.923016	3.662698	50.072619	129.573413	53.175	-0.522	9.536706	0
6	34	whitby	12.049798	6.098394	2.752000	46.548594	124.950813	54.481	-0.624	9.066397	1
7	35	wickairport	10.489243	4.872400	3.868000	68.961905	101.495480	58.454	-3.088	7.672800	2
8	36	yeovilton	14.372619	6.034524	4.468254	59.466270	127.541237	51.006	-2.641	10.203571	0

Figure 9: Test data from the "small" dataset.

The same procedures were applied as before (i.e. null values replacement, scaling, fitting and prediction). This time the accuracy using the same procedures, classifier, and hyperparameters, was 88% (8/9 correct) even predicting the single "south" data point in the test set correctly, despite the very small training set that was being used by the network for learning.

It's possible that the many data points of individual measurements per station overlap with the individual measurements from other stations in the

5D space, making the decision boundaries complex and difficult to learn. However, by taking averages, the measurements are smoothed out and the decision boundaries are more simple, however, more robust due to using aggregated data. To confirm this hypothesis, ideas for further investigation would be to, for example, use the means from more years, select different subsets of stations (i.e. stratified cross-validation), or confirm that using a "large" dataset is more ineffective in general for this case (i.e. use other classifiers, try other scaling types, select data from more years etc.).

Part 3

For this task, the data from the 4 happiness excel files was extracted at the smallest geographical level of cities/districts, excluding the larger 2 levels. This resulted in 351 entries per dataset for years 2011-2012, 2012-2013, and 2013-2014, and in 336 entries for dataset 2014-2015. The columns related to confidence intervals were discarded. The datasets were cleaned manually, saved as csvs and were uploaded to *Google Drive*: Next, a public dataset con-

Codes	Area	Low	Medium	High	Very High	Average	Sample size
E08000020	Gateshead	11.4	17.38	40.45	30.78	7.25	860
E08000021	Newcastle upon Tyn	10.73	17.79	44.72	26.75	7.2	930
E08000022	North Tyneside	10.47	17.12	43.07	29.35	7.26	810
E08000023	South Tyneside	13.7	16.81	36.68	32.81	7.2	970
E08000024	Sunderland	12.61	18.72	32.28	36.39	7.27	880
E07000026	Allerdale	x	x	x	x	8.06	170
E07000027	Barrow-in-Furness	x	x	x	x	7.19	130
E07000028	Carlisle	x	x	x	x	7.37	230
E07000029	Copeland	x	x	x	x	7.61	120
E07000030	Eden	x	x	x	x	7.57	110
E07000031	South Lakeland	x	x	x	x	7.75	200
E08000001	Bolton	11.84	14.65	38.31	35.19	7.41	810
E08000002	Bury	8.35	16.51	41.61	33.53	7.53	680
E08000003	Manchester	10.2	17.71	40.54	31.56	7.31	970

Figure 10: Head of the happiness data for year 2014-2015.

taining the codes and coordinates of 391 local authority districts was downloaded from *https://public.opendatasoft.com/explore/dataset/united-kingdom-local-authority-districts-december-2018*.

The dataset was downloaded as JSON and relevant information was selected using the Python *json* module and was saved as a csv:

	Codes	lat	long
0	E07000085	51.07465983	-0.9550706569
1	E07000090	50.84848726	-0.9905934154
2	E07000102	51.66098291	-0.4605850039
3	E07000126	53.7279726	-2.708716835
4	E07000144	52.70128582	1.30985846
5	E07000235	52.18295323	-2.345025467
6	E08000030	52.59925905	-1.966329162
7	S12000013	57.92617092	-6.851839299
8	S12000018	55.90118767	-4.743644861
9	W06000020	51.69863792	-3.053468726
10	E07000038	53.22592252	-1.448310362

Figure 11: Local authority district codes and coordinates.

Next, the 5 datasets (the 4 happiness datasets and the dataset with the code-coordinate mapping) were joined based on the **Codes** field, resulting in a dataset of 317 rows (where each area is unique) and 31 columns denoting the codes, the coordinates, duplicate area name columns for the 4 happiness datasets, and the happiness scores for the 4 years. A new column **Global Average** was created by calculating the mean of the 4 **Average** columns, which denoted the average happiness per year for an area. The global average denoted the average happiness for an area for the 4 years.

Finally, this merged dataset was joined with the temperature dataset. The used temperature dataset was the grouped-by-station dataset from Part 1, denoting the mean measurements covering the period 1979-1999 - a dataset of 37 entries, 1 entry per station (the census periods don't match, but I'm working with the assumption that the climate stays relatively constant over time as per the task specification):

station	t_max	t_min	af_days	rain_mm	sun_hours	lat	long	t_avg
aberporth	12.301190	7.061508	1.698413	73.168651	128.881349	52.139	-4.570	9.681349
armagh	12.906349	5.833730	3.388889	67.738889	98.476984	54.352	-6.649	9.370040
ballypatrick	11.201770	5.339189	3.230337	104.559848	107.738194	55.181	-6.153	8.255963
bradford	12.426190	5.793651	3.750000	73.244048	101.493254	53.813	-1.772	9.109921
braemar	10.355556	2.815873	8.730159	77.550000	100.346586	57.006	-3.396	6.585714
camborne	13.206746	8.085714	0.869048	91.169841	136.414458	50.218	-5.327	10.646230
cambridge	14.082937	6.136508	3.619048	46.900000	123.485317	52.245	0.102	10.109722
cardiff	14.315476	6.841667	3.126984	95.348810	125.095652	51.488	-3.187	10.578571
chivenor	14.704403	7.833974	1.711538	76.259914	NaN	51.089	-4.147	11.283974
cwmystwyth	11.618952	4.989069	5.024291	152.641129	96.934274	52.358	-3.802	8.306883
dunstaffnage	12.070161	6.037500	2.518072	145.009504	100.353000	56.451	-5.439	9.053831
durham	12.565476	5.126984	4.527778	54.155159	117.720080	54.768	-1.585	8.846230
eastbourne	13.698413	8.385317	1.408730	66.963492	154.348016	50.762	0.285	11.041865

Figure 12: Part of the grouped-by-station dataframe.

The joining was done by looping over the 317 areas of the merged dataset and finding the closest weather station from the smaller dataset coordinate-wise for each area. This was done with a Python function *cdist* from the module *scipy.spatial.distance*, as I wasn't sure whether the distances would be correctly sorted when using a custom generic Euclidean Distance function, due to not being sure whether a degree of longitude is larger or smaller than a degree of latitude distance-wise, as the Earth is not perfectly spherical (it's more of a disc shape, ha). When the closest station was found, the 5 weather readings of this station and the column $t\_avg = (t\_min + t\_max)/2$ were appended to the merged dataset for the relevant area.

This resulted in a dataset, which was used to answer the question of whether weather (ha) affects happiness. This was done by taking pairwise correlations between the column **Global Average** (denoting the average happiness rating for years 2011-2015) and the 6 weather reading columns (the 5 default reading columns + the column *t_avg*). This was done using the function *pearsonr* from the *scipy.stats* module, which, in addition to the correlation, returns the level of significance of the correlation:

```
print(pearsonr(df_merged['t_avg'], df_merged['Global
↪  Average']))
print(pearsonr(df_merged['t_min'], df_merged['Global
↪  Average']))
print(pearsonr(df_merged['t_max'], df_merged['Global
↪  Average']))
print(pearsonr(df_merged['rain_mm'], df_merged['Global
↪  Average']))
print(pearsonr(df_merged['sun_hours'], df_merged['Global
↪  Average']))
print(pearsonr(df_merged['af_days'], df_merged['Global
↪  Average']))
```

The results were as follows:

- Correlation with *t_avg* - -0.107, p-value - 0.055
- Correlation with *t_min* - -0.054, p-value - 0.331
- **Correlation with *t_max* - -0.140, p-value - 0.012**
- Correlation with *rain_mm* - 0.046, p-value - 0.412
- Correlation with *sun_hours* - -0.009, p-value - 0.865
- Correlation with *af_days* - 0.015, p-value - 0.783

Although most results are not statistically significant, we observe that surprisingly, the maximum temperature column correlates negatively with happiness levels with statistical significance ($p < 0.05$). Interestingly, the literature suggests that weather doesn't correlate highly with happiness, and moreover, counterintuitively, there may even exist a slight correlation between warm weather and negative moods (however, the correlations may be non-causal). The 2 listed articles seem to support the findings from our dataset, contain references to scientific literature and discuss the links between weather and happiness levels in more detail:

https://www.bbc.co.uk/news/uk-18986041

https://www.psychologytoday.com/gb/blog/the-social-self/201604/does-warmer-weather-really-make-you-happier

Part 4

In order to take temporal effects into account and determine whether weather affects happiness year by year, one needs to take the mean temperature over all stations over a given year and correlate this with the mean happiness levels in the entirety of UK for the particular year. However, since these operations would result in only 4 datapoints (for the 4 years), I've decided not to draw correlations between the global weather and the global happiness levels due to not having enough data, which would increase the probability of "by-chance" correlations. Additionally, as established from available research, the link between weather and happiness is not strong.

Therefore, I've decided to visualize the distribution of happiness across the UK using the dataset created in Part 3. Since the dataset contained the matched regions along with their coordinates and their happiness levels (in addition to the mean weather measurements of the closest station for the 1979-1999 period), I first created several heatmaps, which were used to motivate further investigation. I focused mainly on the distribution of the low happiness scores, since high concentrations of very unhappy people will (obviously) correlate more with higher rates of depression and suicide. Therefore, more resources could be put into well-being in these areas. Additionally, it could be argued that the gain of 1 point of happiness in the very low levels is significantly higher than the gain of 1 point in the very high levels, since the utility of happiness grows with a higher rate up until a point of "contentment" (similar to the idea for the utility of money).

The first point of investigation was to plot a heatmap of happiness based on the number of unhappy people in the regions from the dataset. This was done by dividing the **Very Low** column by 100 (to get percentages) and multiplying the percentages by the **Sample size** column to get the number of unhappy people per region. This operation was done for the 4 years and the mean number of very unhappy people per region for the 4 years was taken for the heatmap point weighing:

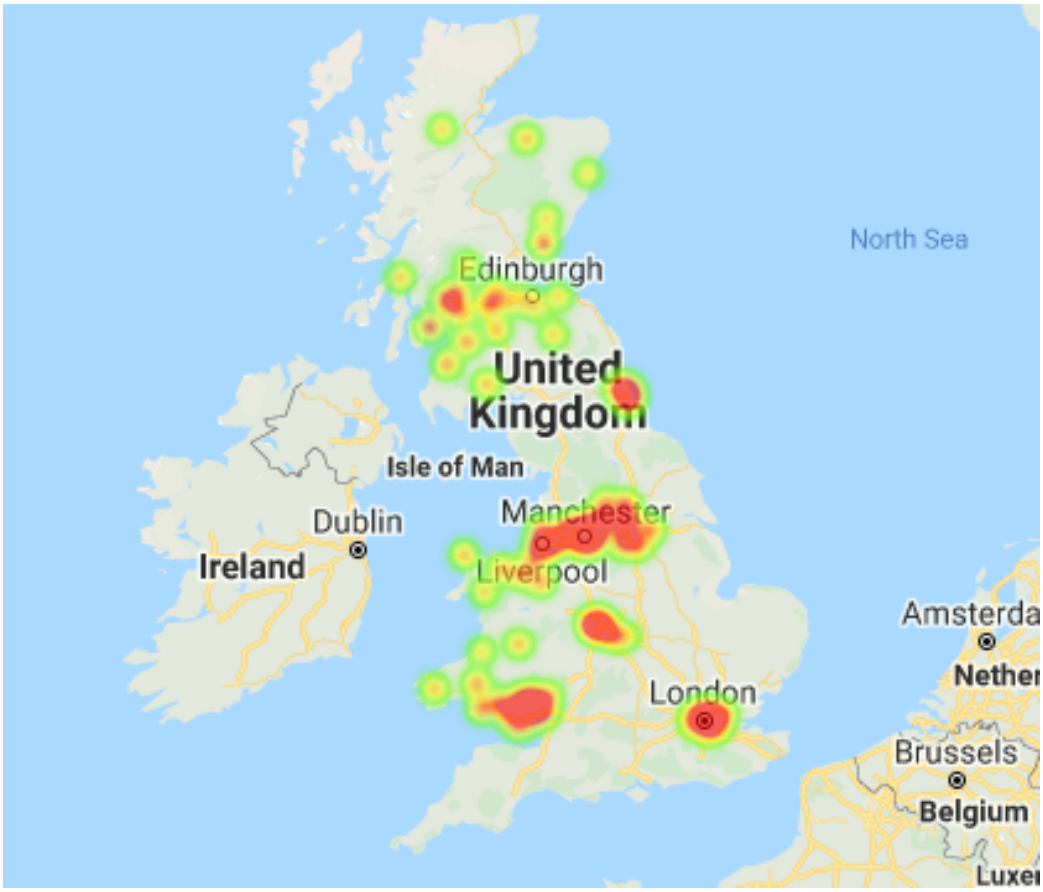

Figure 13: Heatmap of the average number of unhappy people in the UK for the years 2011-2015.

Although this map is not fully popululated due to the high number of nulls in the **Very Low** column, it shows a clear trend of high number of unhappy people in the big cities. However, because a check was not performed whether the sample size is proportional to the population of a region, I decided that this may simply be due to there being more people in the big cities.

Therefore, I took the average of the percentages for the 4 years as well (this wouldn't have been done if I noticed a trend of high number of unhappy people in underpopulated areas):

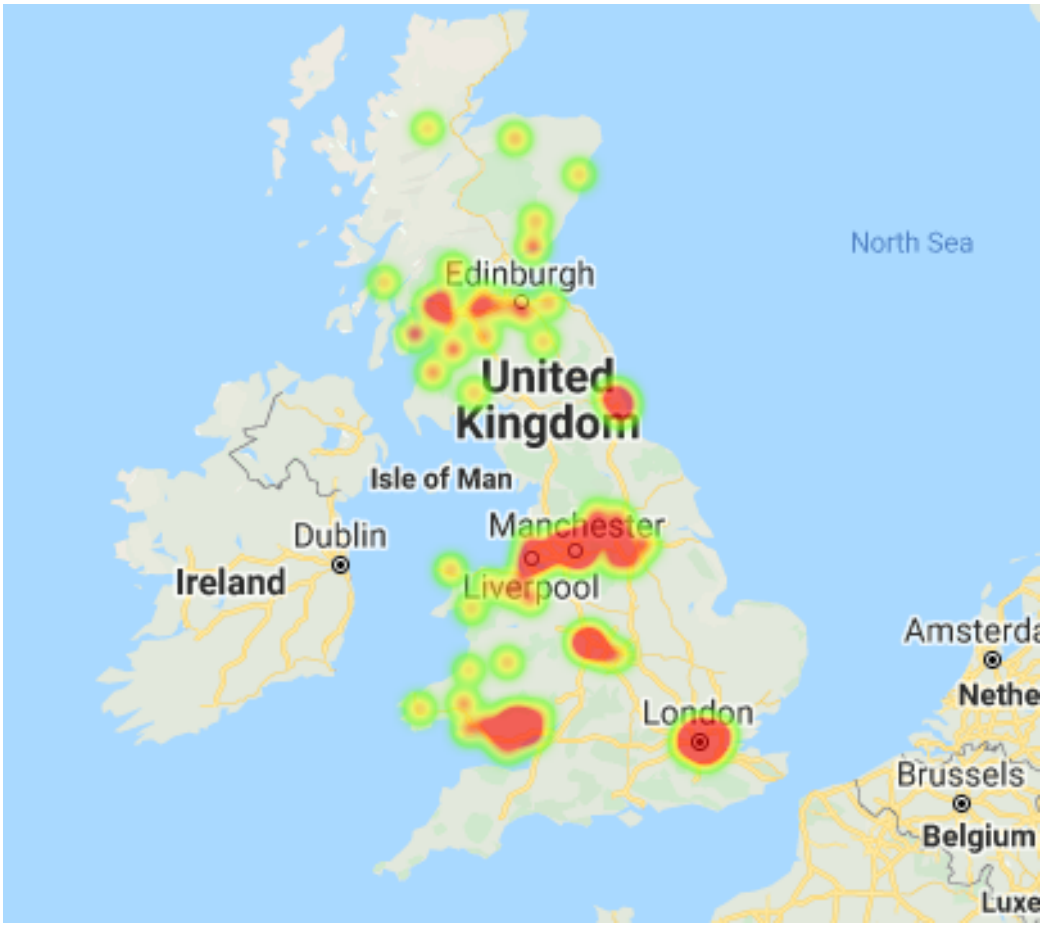

Figure 14: Heatmap of the average percentage of very unhappy people in the UK for the years 2011-2015.

This shows that there is clear trend of the percentages of very unhappy people being higher in big cities.

In order to investigate whether this trend affects the average happiness distribution significantly, I also created a heatmap of average happiness per region:

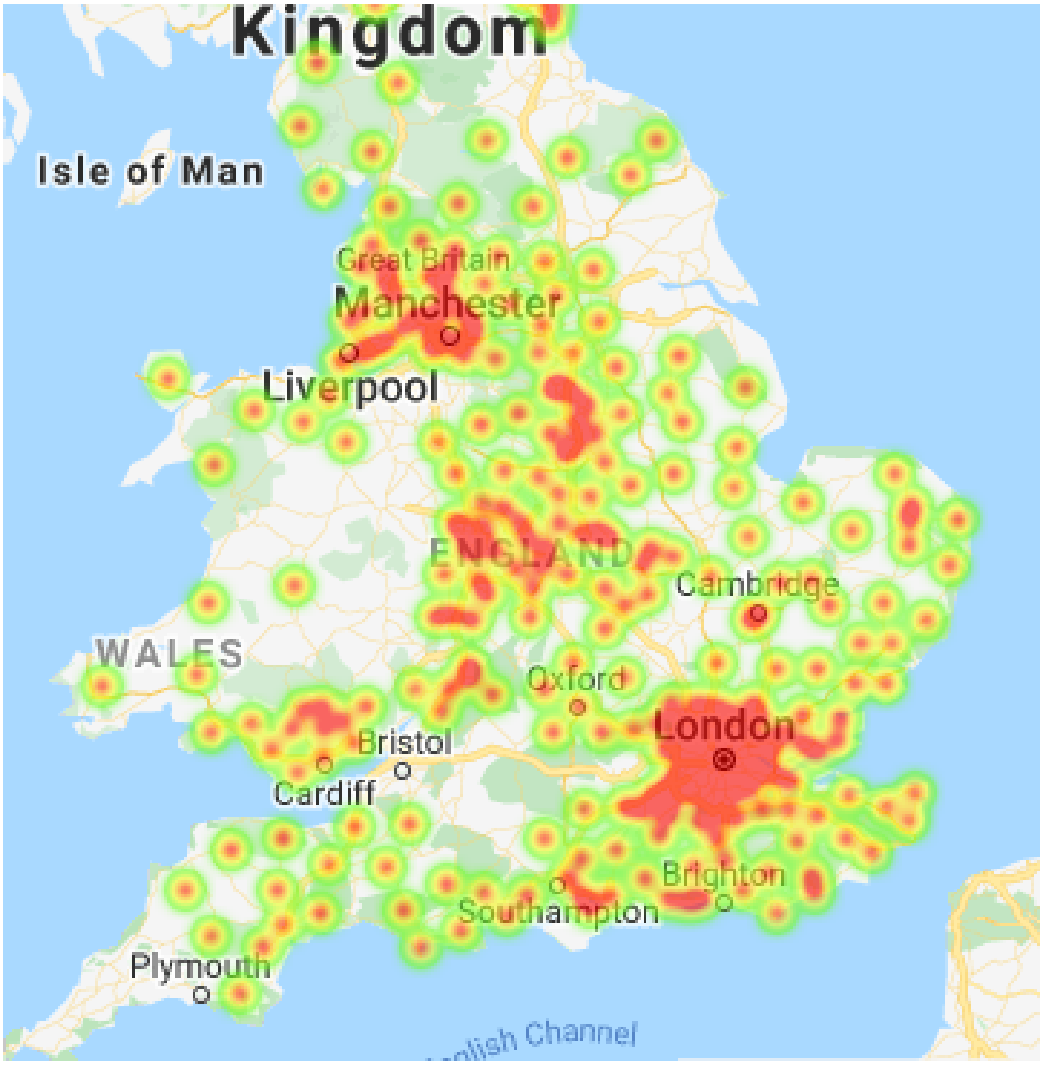

Figure 15: Heatmap of the average happiness ratings in the UK for the years 2011-2015.

This showed that despite the high number of unhappy people, the average happiness level around the big cities remained high which made me suspicious. I then found out that by using the heatmaps I've been plotting concentrations of the measurements instead of the distribution of the rating values, despite using the weighted version of the *gmaps* heatmap (using the ratings as weights). I found this by zooming out the map, setting the circle radius to very small such that circles didn't overlap, and multiplying the

weights progressively (by 1000, 10000, and so on). The radiai of the dots were not increasing and I further found that *gmaps* was not the appropriate library to plot distribution of values. Thus this made the observations from Part 4 so far, invalid.

Upon further investigation, I found that choropleths are the more appropriate type of plot to show the distribution of values by region. The Python library *Folium* was used to create the plots. Because the link - (*https://public.opendatasoft.com/explore/dataset/united-kingdom-local-authority-districts-december-2018*)
- from which I extracted the coordinates for the regions in the happiness data also contained the shapes of the regions, it had the option to be downloaded as a file with *.geojson* extension. I used this *.geojson* along with the transformed and merged for all years pandas dataframe, to create 3 choropleths - 1 with average rating for the regions for years 2011-2015, 1 with the mean number of very unhappy people (lowest rating) for years 2011-2015, and 1 with the mean percentages of people that are very unhappy for years 2011-2015. The shaded areas represent areas where data was not available (i.e. where values were null):

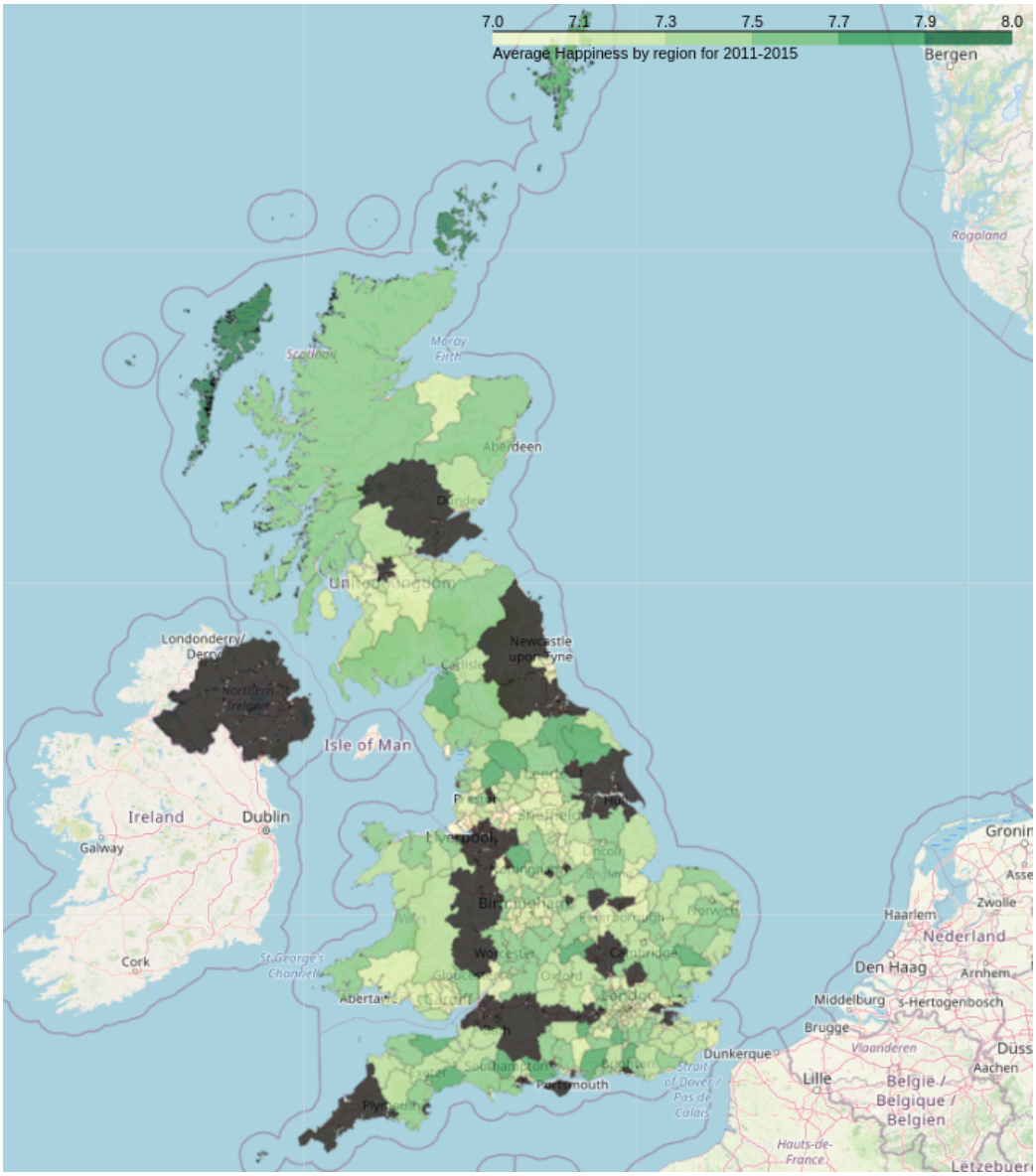

Figure 16: Choropleth of the mean average happiness ratings in the UK for the years 2011-2015.

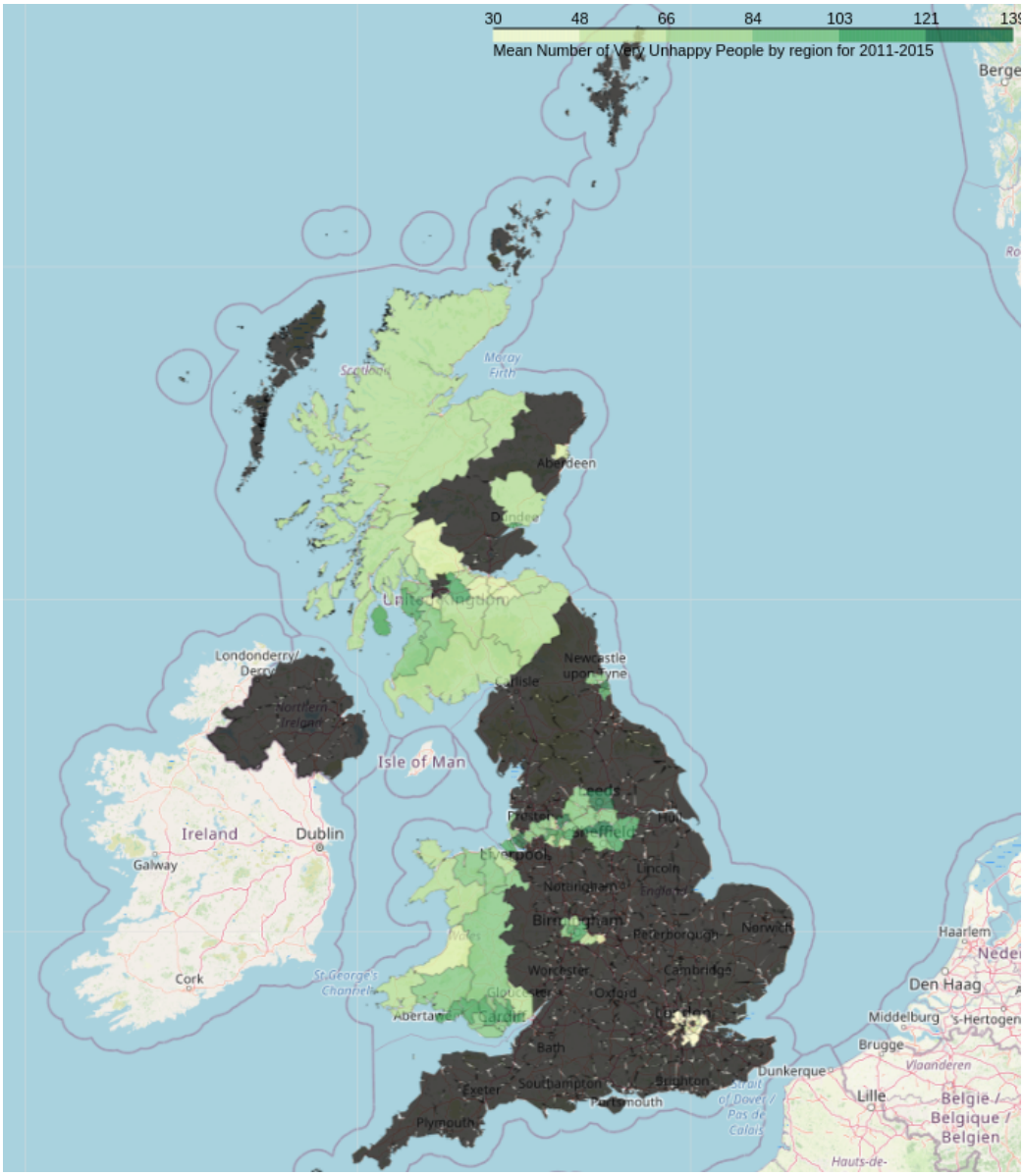

Figure 17: Choropleth of the mean number of very unhappy people in the UK for the years 2011-2015.

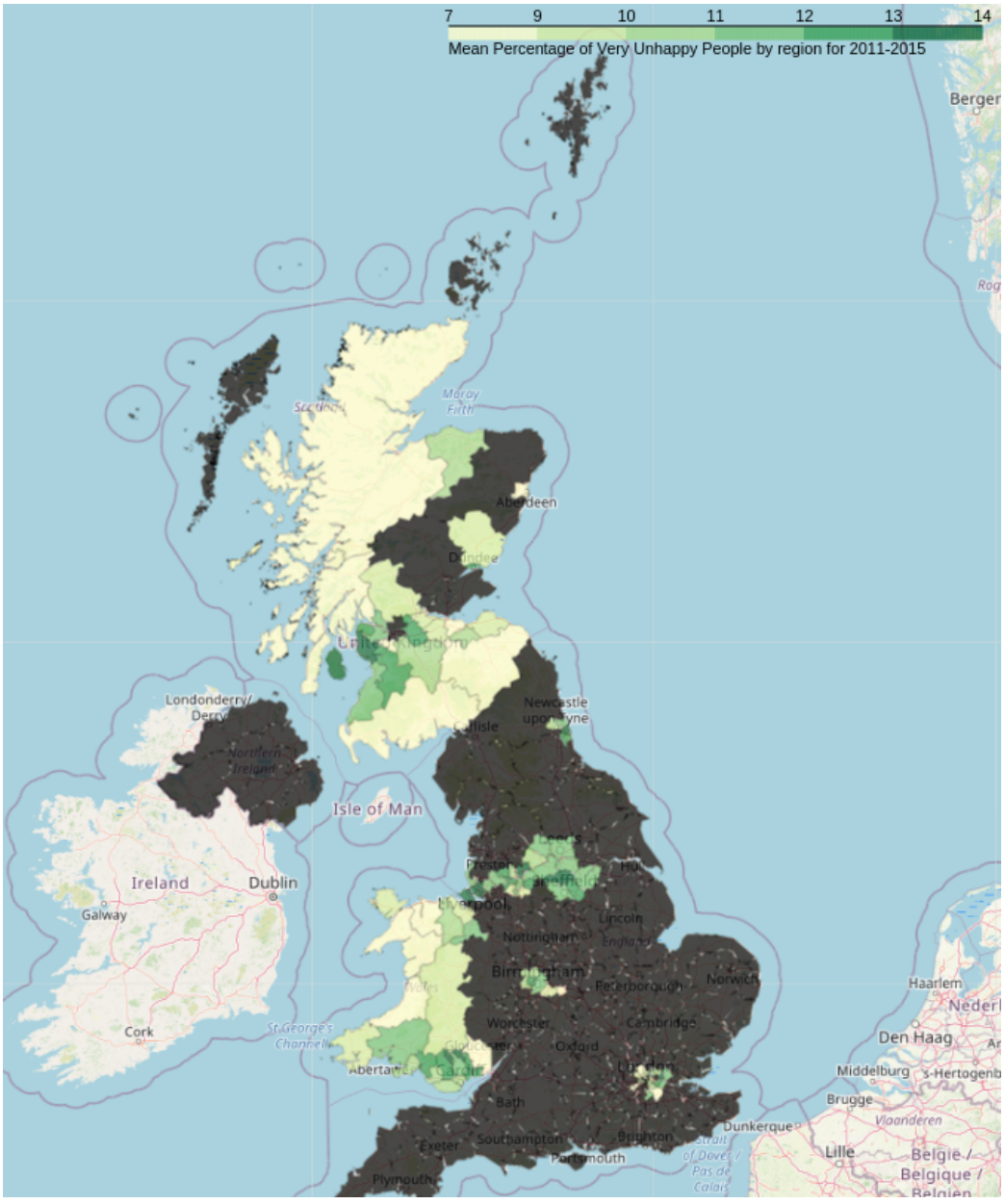

Figure 18: Choropleth of the mean percentage of very unhappy people in the UK for the years 2011-2015.

By investigating the choropleths we can see a common trend - the areas in and around big cities exhibit lower average happiness rating, i.e. London, Cardiff, Manchester and Glasgow experience a happiness "dip" compared to their surrounding areas. This is confirmed by the 2 choropleths denoting the number of very unhappy people and the percentages of very unhappy people. Although much data is unavailable for the second and third plot, the available data shows that the aforementioned areas (London, Manchester, Glasgow, Cardiff) are darker than their surroundings. Thus, the maps show that living in more "natural" rural areas, may contribute to a more positive outlook (although the scientific literature is conflicting and findings vary from country to country).