Python Programming Assignment

Plotting the Grow Dataset

Gejo Royce - 2542972@dundee.ac.uk

AC50002 - Programming languages for Data Engineering

University of Dundee

5 January 2024

Table of Content

1. TASK
Plotting the Grow Dataset

You are provided with:

- ***The Grow dataset Growlocations.csv. This file contains the locations of all the GROW sensors as Latitude and Longitude***
- ***A map of the UK from Openstreet map.***

You should create a Python program that can read the dataset into a dataframe and plot the locations of the sensors on the map provided. You can use online tutorials to do this (but mention them in comments section of your code. However, there are a number of errors with the dataset that you will need to fix in order to get the correct plot. These include

- ***Some location values are way outside the allowed values for latitude and Longitude.***
- ***Some locations are not on the map provided.***
- ***The labels of the columns have not been verified so may be incorrect.***

The bounding box for the map is as follows:

- ***Longitude Min -10.592***
- ***Longitude Max 1.6848***
- ***Latitude Min 50.681***
- ***Latitude Max 57.98***

2. DESIGN

The python code is designed to visualise the GROW sensor's location from a given data set (GrowLocations.csv) and overlay them on the map image (map.png) provided.

The design of the code is based on the following steps :

1. Upload and Reading File – Using the files.upload function in the Google Colab the provided 'GrowLocations.csv' file is been uploaded to the Google Colab. The Data in the uploaded CSV file is read to a dataframe via Pandas function.

2. Swapping Latitude & Longitude Values – The task sheet given a hint regarding incorrectness of the column name and by testing and validating the dataset on test environment, its understood there is incorrect naming of columns. So, the data over those columns are swapped.

3. Data Cleaning, Manipulating and Filtering – Valid Sensors are obtained to a DataFrame, 'clean_data' after filtering the sensor data with the bounding area in the map and avoiding invalid sensors other than this area. After cleaning the data, the sensors in clean_data is copied to cleaned_data.

4. Scattered Plotting of Sensors and Overlay the same to the map – Valid sensors are plot with Scattered Plot methods over the Longitude and Latitude values. Then these plots are mapped to the provided image of map.

5. Display and Saving the Mapped Image – The result is displayed and the output map is saved.

3. CODE

1. Libraries

```
from google.colab import files
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import display
from matplotlib.offsetbox import OffsetImage, AnnotationBbox
```

1. 'from google.colab import file' - module is used to enable file upload functionality which used to upload the sensor data to the google colab environment.
(Reference : https://saturncloud.io/blog/uploading-local-files-using-google-colab/ Step 2. I already tired Step 1, 3 in my previous assignments on machine learning and BIS so need to try other methods)

2. 'import pandas as pd' - pandas library is used for data analysis and this provide the dataframe structures to handle the dataset in tabular form which is easy for manipulate like in the database.

3. 'import matplotlib.pyplot as plt' - library for create illustrations and visualisation in python, here its used to plot sensor locations on the map.

4. 'from IPython.display import display' - the display function from the IPython.display is used to show tabs in much interactive format. The tables are used for cleaned_data table to validate the plots on map with the cleaned data. Also, distribution of grow sensors in the latitude and longitude is iterated in to a distribution chart so that the plotting can be validated.

5. 'from matplotlib.offsetbox import OffsetImage, AnnotationBbox' - this function is used to overlay the plots into the map images.
(Reference : https://medium.com/analytics-vidhya/how-to-plot-image-and-text-on-matplotlib-91d7f23a3043)

2. Modules

The code has four modules to perform overall operations and they are –

2.1. Upload & Read CSV File to the dataframe :

While running this module there will be a prompt to upload the CSV file from the local drive to the Google Colab and the data from the CSV uploaded will be copied to a dataframe – **'data'**, then this data frame is displayed to check the dataframe.

```
# Upload the 'Growlocations.csv' file
uploaded = files.upload()

# Read the data into a DataFrame
data = pd.read_csv(list(uploaded.keys())[0])

display(data)
```

2.2. Swapping the Latitude and Longitude Values :

This module swaps the data in the Latitude to the Longitude and vice-versa. This correct the incorrectness of column name which is hinted on the task as well as validated from testing the data in the test environment.
(https://colab.research.google.com/drive/1bl38E6p5xBKDpKc7Yc-BP-XY19qr3Ttu?usp=sharing)

```
# Swap 'Latitude' and 'Longitude' values
data['Latitude'], data['Longitude'] = data['Longitude'], data['Latitude']
```

2.3. Cleaning & Filtering Data:

The code filters the latitude and longitude in the desired range between the location frame bound with in the map provided. And the filtered rows are written into a new data-frame called - 'cleaned_data'. Then the cleaned data frame is displayed for validation and check the distribution. Also, while checking the format of the dataframe, we can understand the latitude is coming first in the original dataframe, but in the mapping longitude is in the x-axis so for the data manipulation we need consider longitude to first since those data come to x-axis and latitude data to y-axis.

```
# Function to filter and clean the data
def clean_data(data):
    # Filter rows out of desired latitude and longitude range
    valid_data = data[(data['Longitude'] >= -10.60 ) & (data['Longitude'] <= 1.70) &
                      (data['Latitude'] >= 50.60) & (data['Latitude'] <=58.0 )]
    return valid_data
# Clean the data
cleaned_data = clean_data(data)
print("\nCleaned Data:")
display(cleaned_data)
```

2.3.1. Distribution of Sensors & Its Frequencies:

This module is used to understand the relation between the longitude and latitude frequencies and how they can visualise with scattered plotting.

```
# Distribution of Longitude
from matplotlib import pyplot as plt
cleaned_data['Longitude'].plot(kind='hist', bins=20, title='Longitude')
plt.gca().spines[['top', 'right',]].set_visible(False)

#Distribution of Latitude
from matplotlib import pyplot as plt
cleaned_data['Latitude'].plot(kind='hist', bins=20, title='Latitude')
plt.gca().spines[['top', 'right',]].set_visible(False)

#Scattered Mapping of Sensors
from matplotlib import pyplot as plt
cleaned_data.plot(kind='scatter', x='Longitude',y='Latitude', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

2.4. Plotting the Scattered Mapping to the Given Map:

This module overlays the scattered mapping of sensor locations by longitude – latitude data into the provided map. The overlayed map is displayed and save as output map.

Reference : https://medium.com/analytics-vidhya/how-to-plot-image-and-text-on-matplotlib-91d7f23a3043

```
# Load the map image
map_image = plt.imread('map.png')
# Overlay the map image over the plotted sensor locations in given latitude and longitude
range
fig, ax = plt.subplots(figsize=(12,7))
ax.imshow(map image, extent=[ -10.592, 1.6848,50.681, 57.985])
# Plot the valid sensor locations in the cleaned data with scatter methods
ax.scatter(cleaned_data['Longitude'], cleaned_data['Latitude'], marker='o', color='blue', la-
bel='GROW Sensors')

# Display the map and sensor locations
plt.title('GROW Sensor Locations on Map')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.legend()
plt.grid(True)

# Save the output to a file named 'outputmap.png'
plt.savefig('outputmap.png')
plt.show()
```

4. EXECUTION & TESTING

The Code File and Executed Files are uploaded to GitHub repository - https://github.com/GejoRoyce/Python_Assignment_2

Also can be access via Google Colab
Final Code : https://colab.research.google.com/drive/1Xy_E_eL_ipLVzrZfGEr5MtVpyFkwI0lE?usp=sharing

Test Code : https://colab.research.google.com/drive/1bl38E6p5xBKDpKc7Yc-BP-XY19qr3Ttu?usp=sharing

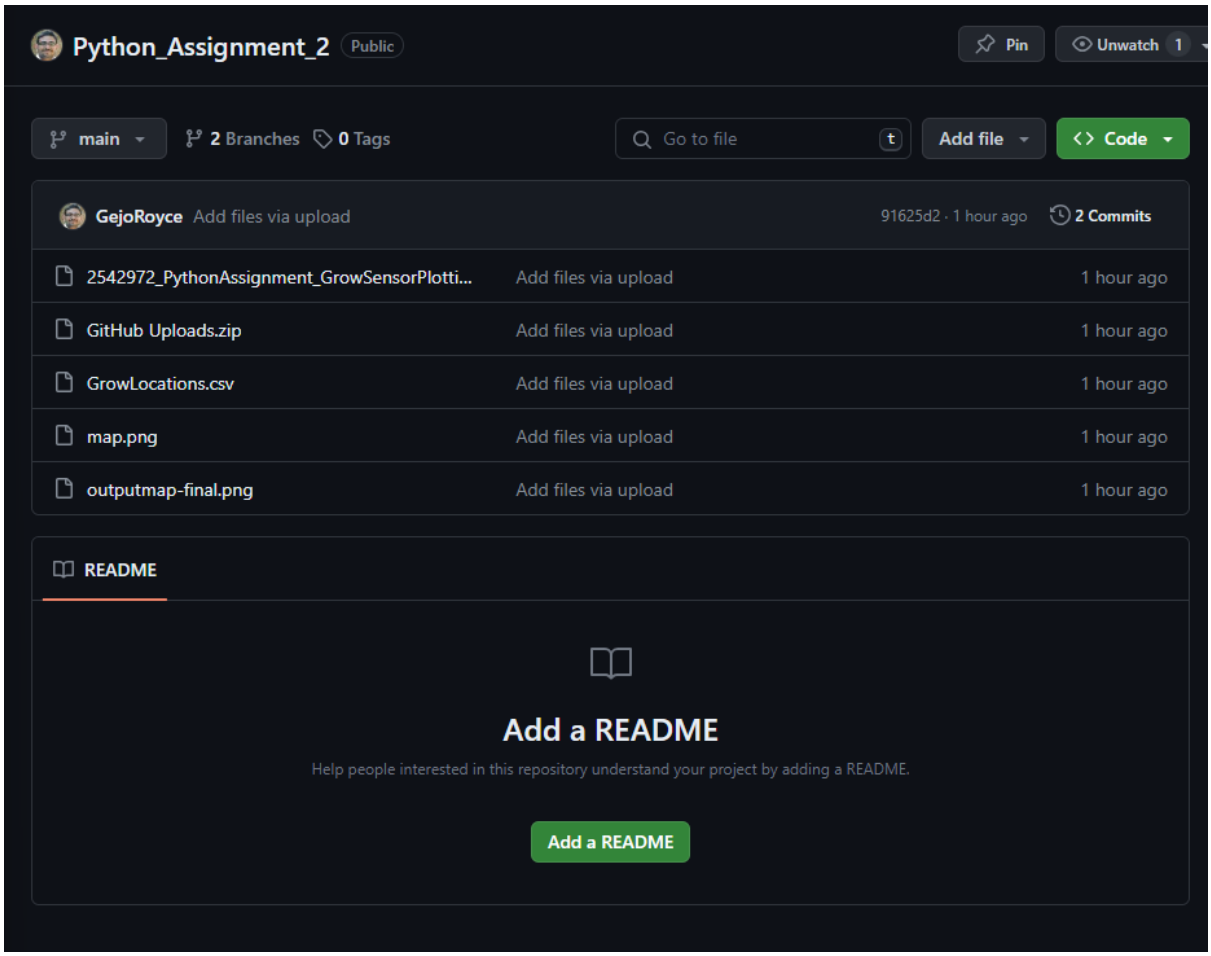
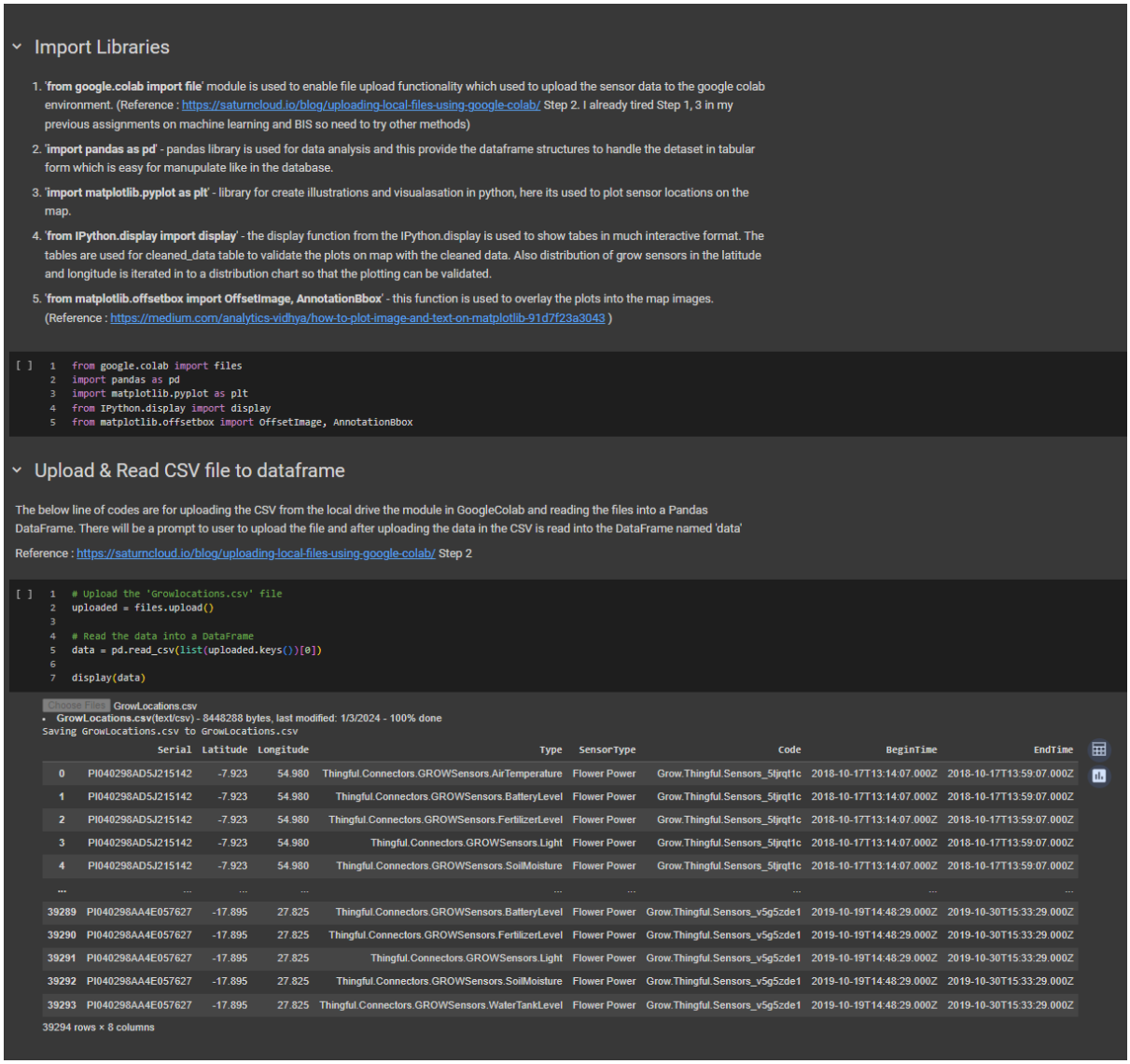


Figure 1 – File Uploads in Main Branch of Python Assignment 2 folder in GitHub Repository

In the repository there is another branch named test showing how the coding is evolved to swapping the column names and how the boundaries are trimmed to the fit the map.



Import Libraries

1. '**from google.colab import file**' module is used to enable file upload functionality which used to upload the sensor data to the google colab environment. (Reference : https://saturncloud.io/blog/uploading-local-files-using-google-colab/ Step 2. I already tired Step 1, 3 in my previous assignments on machine learning and BIS so need to try other methods)
2. '**import pandas as pd**' - pandas library is used for data analysis and this provide the dataframe structures to handle the detaset in tabular form which is easy for manupulate like in the database.
3. '**import matplotlib.pyplot as plt**' - library for create illustrations and visualasation in python, here its used to plot sensor locations on the map.
4. '**from IPython.display import display**' - the display function from the IPython.display is used to show tabes in much interactive format. The tables are used for cleaned_data table to validate the plots on map with the cleaned data. Also distribution of grow sensors in the latitude and longitude is iterated in to a distribution chart so that the plotting can be validated.
5. '**from matplotlib.offsetbox import OffsetImage, AnnotationBbox**' - this function is used to overlay the plots into the map images. (Reference : https://medium.com/analytics-vidhya/how-to-plot-image-and-text-on-matplotlib-91d7f23a3043)

```
[ ]  1  from google.colab import files
     2  import pandas as pd
     3  import matplotlib.pyplot as plt
     4  from IPython.display import display
     5  from matplotlib.offsetbox import OffsetImage, AnnotationBbox
```

Upload & Read CSV file to dataframe

The below line of codes are for uploading the CSV from the local drive the module in GoogleColab and reading the files into a Pandas DataFrame. There will be a prompt to user to upload the file and after uploading the data in the CSV is read into the DataFrame named 'data'

Reference : https://saturncloud.io/blog/uploading-local-files-using-google-colab/ Step 2

```
[ ]  1  # Upload the 'Growlocations.csv' file
     2  uploaded = files.upload()
     3
     4  # Read the data into a DataFrame
     5  data = pd.read_csv(list(uploaded.keys())[0])
     6
     7  display(data)
```

Choose Files GrowLocations.csv

- **GrowLocations.csv**(text/csv) - 8448288 bytes, last modified: 1/3/2024 - 100% done

Saving GrowLocations.csv to GrowLocations.csv

	Serial	Latitude	Longitude	Type	SensorType	Code	BeginTime	EndTime
0	PI040298AD5J215142	-7.923	54.980	Thingful.Connectors.GROWSensors.AirTemperature	Flower Power	Grow.Thingful.Sensors_5tjrqt1c	2018-10-17T13:14:07.000Z	2018-10-17T13:59:07.000Z
1	PI040298AD5J215142	-7.923	54.980	Thingful.Connectors.GROWSensors.BatteryLevel	Flower Power	Grow.Thingful.Sensors_5tjrqt1c	2018-10-17T13:14:07.000Z	2018-10-17T13:59:07.000Z
2	PI040298AD5J215142	-7.923	54.980	Thingful.Connectors.GROWSensors.FertilizerLevel	Flower Power	Grow.Thingful.Sensors_5tjrqt1c	2018-10-17T13:14:07.000Z	2018-10-17T13:59:07.000Z
3	PI040298AD5J215142	-7.923	54.980	Thingful.Connectors.GROWSensors.Light	Flower Power	Grow.Thingful.Sensors_5tjrqt1c	2018-10-17T13:14:07.000Z	2018-10-17T13:59:07.000Z
4	PI040298AD5J215142	-7.923	54.980	Thingful.Connectors.GROWSensors.SoilMoisture	Flower Power	Grow.Thingful.Sensors_5tjrqt1c	2018-10-17T13:14:07.000Z	2018-10-17T13:59:07.000Z
...	...	...	...	...	...	...	...	...
39289	PI040298AA4E057627	-17.895	27.825	Thingful.Connectors.GROWSensors.BatteryLevel	Flower Power	Grow.Thingful.Sensors_v5g5zde1	2019-10-19T14:48:29.000Z	2019-10-30T15:33:29.000Z
39290	PI040298AA4E057627	-17.895	27.825	Thingful.Connectors.GROWSensors.FertilizerLevel	Flower Power	Grow.Thingful.Sensors_v5g5zde1	2019-10-19T14:48:29.000Z	2019-10-30T15:33:29.000Z
39291	PI040298AA4E057627	-17.895	27.825	Thingful.Connectors.GROWSensors.Light	Flower Power	Grow.Thingful.Sensors_v5g5zde1	2019-10-19T14:48:29.000Z	2019-10-30T15:33:29.000Z
39292	PI040298AA4E057627	-17.895	27.825	Thingful.Connectors.GROWSensors.SoilMoisture	Flower Power	Grow.Thingful.Sensors_v5g5zde1	2019-10-19T14:48:29.000Z	2019-10-30T15:33:29.000Z
39293	PI040298AA4E057627	-17.895	27.825	Thingful.Connectors.GROWSensors.WaterTankLevel	Flower Power	Grow.Thingful.Sensors_v5g5zde1	2019-10-19T14:48:29.000Z	2019-10-30T15:33:29.000Z

39294 rows × 8 columns

Figure 2 - Output from the Code on Google Colab

Swapping the Latitude & Longitude Values

From the assumption from the tests done on the python notebook (https://colab.research.google.com/drive/1bl38E6p5xBKDpKc7Yc-BP-XY19qr3Ttu?usp=sharing) the incorrectness of column name is validated so swapping the Latitude and Longitude Values for getting desired result.

```
[ ]  # Swap 'Latitude' and 'Longitude' values
     data['Latitude'], data['Longitude'] = data['Longitude'], data['Latitude']
```

Cleaning & Filtering Data

The code filters the latitude and longitude in the desired range between the location frame bound with in the map provided. And the filtered rows are write into a new dataframe called - 'cleaned_data'. Then the cleaned data frame is displayed for validation and check the distribution. Also while checking the format of the dataframe we can understand the latitude is coming first, but in the mapping longitude is in the x-axis so for the data manipulation we need consider longitude to first since those data come to x-axis and latitude data to y-axis.

```
# Function to filter and clean the data
def clean_data(data):
    # Filter rows out of desired latitude and longitude range
    valid_data = data[(data['Longitude'] >= -10.60 ) & (data['Longitude'] <= 1.70) &
                      (data['Latitude'] >= 50.60) & (data['Latitude'] <=58.0 )]
    return valid_data

# Clean the data
cleaned_data = clean_data(data)

print("\nCleaned Data:")
display(cleaned_data)
```

```
Cleaned Data:
```

	Serial	Latitude	Longitude	Type	SensorType	Code	BeginTime	EndTime
0	PI040298AD5J215142	54.980	-7.923	Thingful.Connectors.GROWSensors.AirTemperature	Flower Power	Grow.Thingful.Sensors_5tjrqt1c	2018-10-17T13:14:07.000Z	2018-10-17T13:59:07.000Z
1	PI040298AD5J215142	54.980	-7.923	Thingful.Connectors.GROWSensors.BatteryLevel	Flower Power	Grow.Thingful.Sensors_5tjrqt1c	2018-10-17T13:14:07.000Z	2018-10-17T13:59:07.000Z
2	PI040298AD5J215142	54.980	-7.923	Thingful.Connectors.GROWSensors.FertilizerLevel	Flower Power	Grow.Thingful.Sensors_5tjrqt1c	2018-10-17T13:14:07.000Z	2018-10-17T13:59:07.000Z
3	PI040298AD5J215142	54.980	-7.923	Thingful.Connectors.GROWSensors.Light	Flower Power	Grow.Thingful.Sensors_5tjrqt1c	2018-10-17T13:14:07.000Z	2018-10-17T13:59:07.000Z
4	PI040298AD5J215142	54.980	-7.923	Thingful.Connectors.GROWSensors.SoilMoisture	Flower Power	Grow.Thingful.Sensors_5tjrqt1c	2018-10-17T13:14:07.000Z	2018-10-17T13:59:07.000Z
...	...	...	...	...	...	...	...	...
39259	PI040298AA3K026204	51.396	0.122	Thingful.Connectors.GROWSensors.BatteryLevel	Flower Power	Grow.Thingful.Sensors_vnbfmkdp	2019-10-30T11:17:34.000Z	2019-10-30T16:02:34.000Z
39260	PI040298AA3K026204	51.396	0.122	Thingful.Connectors.GROWSensors.FertilizerLevel	Flower Power	Grow.Thingful.Sensors_vnbfmkdp	2019-10-30T11:17:34.000Z	2019-10-30T16:02:34.000Z
39261	PI040298AA3K026204	51.396	0.122	Thingful.Connectors.GROWSensors.Light	Flower Power	Grow.Thingful.Sensors_vnbfmkdp	2019-10-30T11:17:34.000Z	2019-10-30T16:02:34.000Z
39262	PI040298AA3K026204	51.396	0.122	Thingful.Connectors.GROWSensors.SoilMoisture	Flower Power	Grow.Thingful.Sensors_vnbfmkdp	2019-10-30T11:17:34.000Z	2019-10-30T16:02:34.000Z
39263	PI040298AA3K026204	51.396	0.122	Thingful.Connectors.GROWSensors.WaterTankLevel	Flower Power	Grow.Thingful.Sensors_vnbfmkdp	2019-10-30T11:17:34.000Z	2019-10-30T16:02:34.000Z

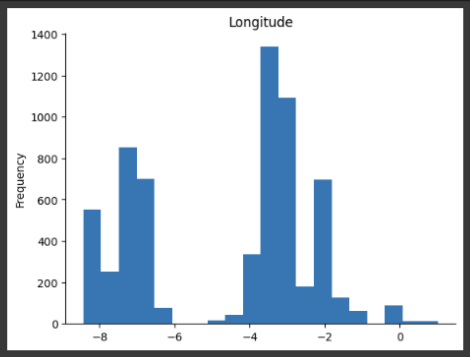
6438 rows × 8 columns

Figure 3 - Output from the Code on Google Colab

Distribution of Sensors & Its Frequencies

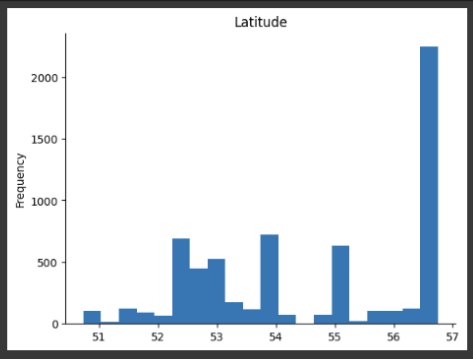
Distribution of Longitude

```
[ ]  1  from matplotlib import pyplot as plt
     2  cleaned_data['Longitude'].plot(kind='hist', bins=20, title='Longitude')
     3  plt.gca().spines[['top', 'right',]].set_visible(False)
```



Distribution of Latitude

```
[ ]  1  from matplotlib import pyplot as plt
     2  cleaned_data['Latitude'].plot(kind='hist', bins=20, title='Latitude')
     3  plt.gca().spines[['top', 'right',]].set_visible(False)
```



Scatted Mapping of Sensors

```
1  from matplotlib import pyplot as plt
2  cleaned_data.plot(kind='scatter', x='Longitude',y='Latitude', s=32, alpha=.8)
3  plt.gca().spines[['top', 'right',]].set_visible(False)
```

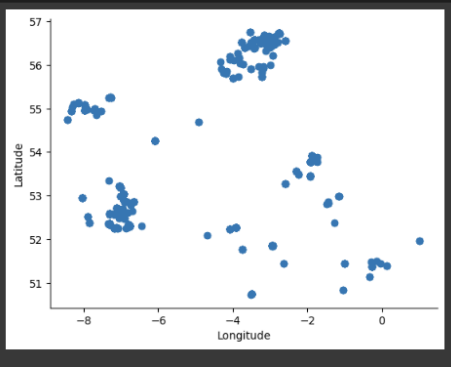


Figure 4 - Output from the Code on Google Colab

Plotting the Scatted Mapping to the Given Map

The below code loads the map image, create the plot of map and sensor location and plot the valid sensor location to the map and display the same and generate output_map.

Reference : https://medium.com/analytics-vidhya/how-to-plot-image-and-text-on-matplotlib-91d7f23a3043

```
# Load the map image
map_image = plt.imread('map.png')

# Overlay the map image over the plotted sensor locations in given latitude and longitude range
fig, ax = plt.subplots(figsize=(12,7))
ax.imshow(map_image, extent=[ -10.592, 1.6848,50.681, 57.985])

# Plot the valid sensor locations in the cleaned data with scatter methods
ax.scatter(cleaned_data['Longitude'], cleaned_data['Latitude'], marker='o', color='blue', label='GROW Sensors')

# Display the map and sensor locations
plt.title('GROW Sensor Locations on Map')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.legend()
plt.grid(True)

# Save the output to a file named 'outputmap.png'
plt.savefig('outputmap.png')
plt.show()
```

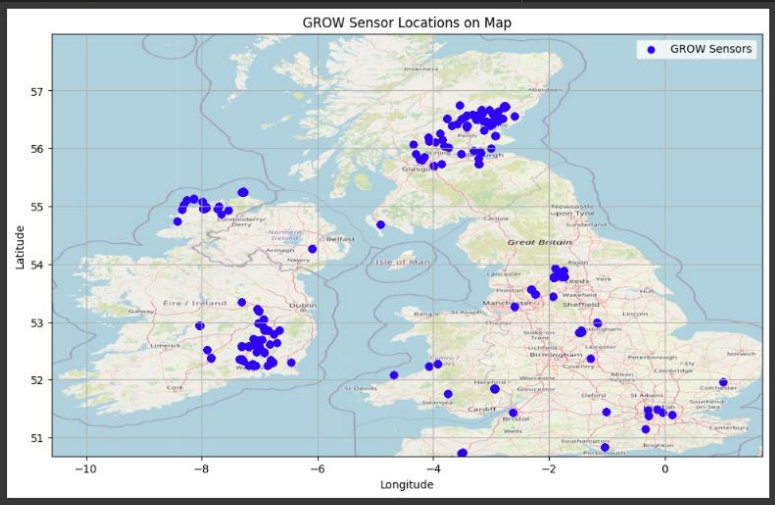
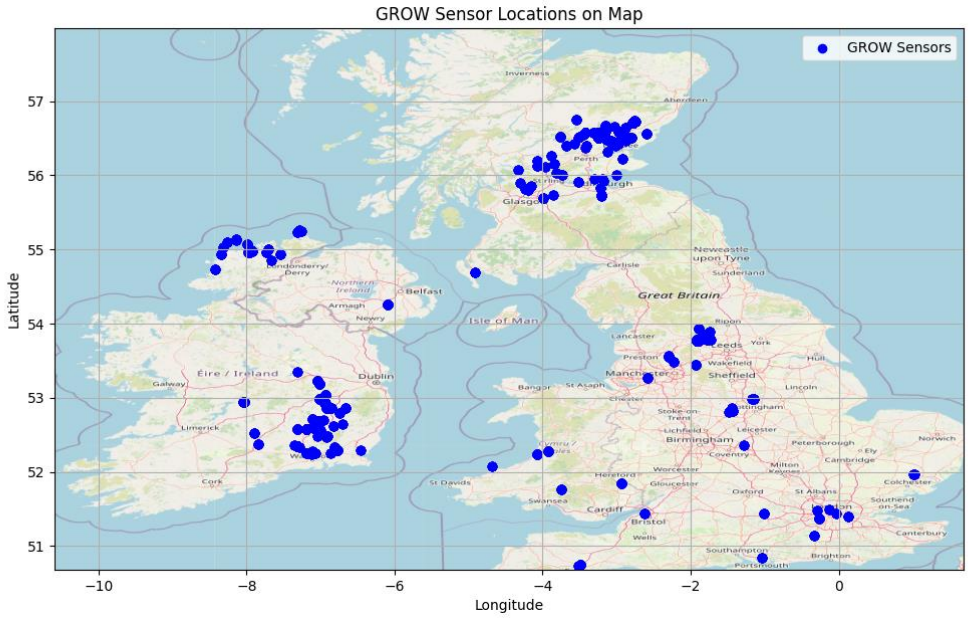
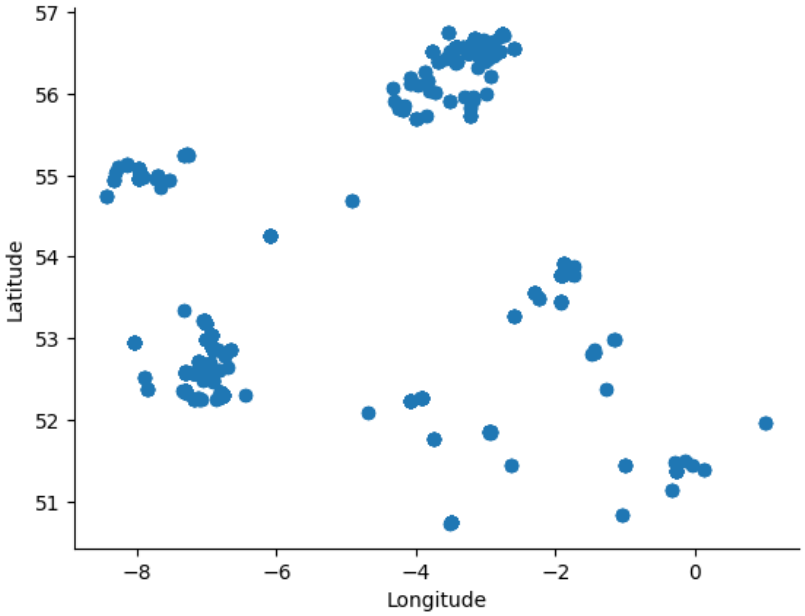
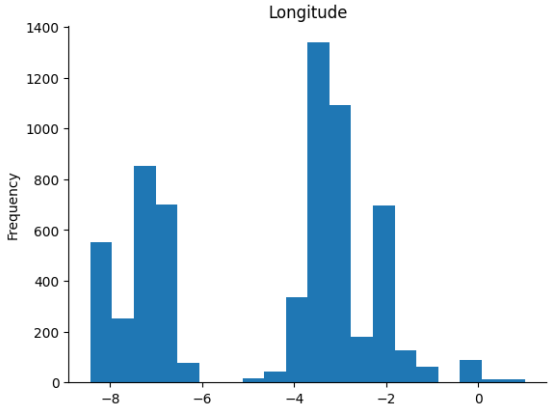


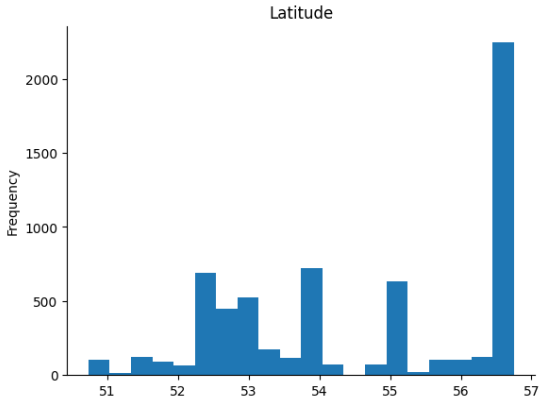
Figure 5 - Output from the Code on Google Colab

5. OUTPUTS


GROW Sensor Locations on Map
GROW Sensors
Latitude
Longitude
Great Britain
Éire / Ireland
Isle of Man


Latitude
Longitude


Longitude
Frequency


Latitude
Frequency

6. REFERENCES

1. Uploading local files using Google Colab (2023) Saturn Cloud Blog.
Available at: https://saturncloud.io/blog/uploading-local-files-using-google-colab/
(Accessed: 04 January 2024).

2. Shotin (2020)
How to plot image and text on Matplotlib? Medium.
Available at: https://medium.com/analytics-vidhya/how-to-plot-image-and-text-on-matplotlib-91d7f23a3043
(Accessed: 04 January 2024).

3. Zach (2022)
How to swap two columns in pandas (with example), Statology.
Available at: https://www.statology.org/swap-columns-pandas/
(Accessed: 04 January 2024).

4. Pierson, L. (2019)
Advance your python skills for data science learning path:
LinkedIn learning, formerly Lynda.com, LinkedIn.
Available at: https://www.linkedin.com/learning/paths/advance-your-python-skills-for-data-science?u=43337284
(Accessed: 04 January 2024).