Configuration Manual

MSc Research Project
Data Analytics

Apurv Dubey
Student ID: x21141495

School of Computing
National College of Ireland

Supervisor: Dr Giovani Estrada

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Apurv Dubey
Student ID:	x21141495
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Dr Giovani Estrada
Submission Due Date:	15/12/2022
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Apurv Dubey
Date:	1st February 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Apurv Dubey
x21141495

1 Introduction

The procedures followed to complete the Research Project: Optimal placement of ambulances to best serve emergency calls are outlined in this configuration manual, together with information on the system setup, software, and hardware used. The technical specifications for the software and hardware are presented in Section 2. The method of Reverse geocoding is broken out in Section 4. The models' designs and implementations are discussed in Section 5.

2 System Configuration

To better understand the system setup that was put into place for the project's execution, please refer to this section of the configuration documentation.

2.1 Hardware Requirements

Processor	intel i5 8600k
RAM	16 GB
Disk Space	1 GB

2.2 Software Requirements

Operating System	Windows 10
Programming Language	Python version 3.9
Web-Broser	Google Chrome
Other Softwares	Jupyter Notebook, Microsoft Excel

3 Environment Setup

Details on environment setup, data collection, and use can be found in this part of the configuration guide.

3.1 Jupyter Notebook Setup

When working with Python-based applications, PIP is the package manager of choice. Packages are kept in a massive "online repository" known as the Python Package Index (PyPI). pip's default for obtaining packages and their dependencies is the PyPI repository.

To install Jupyter using pip, we need to first check if pip is updated in our system. Use the following command to update pip:

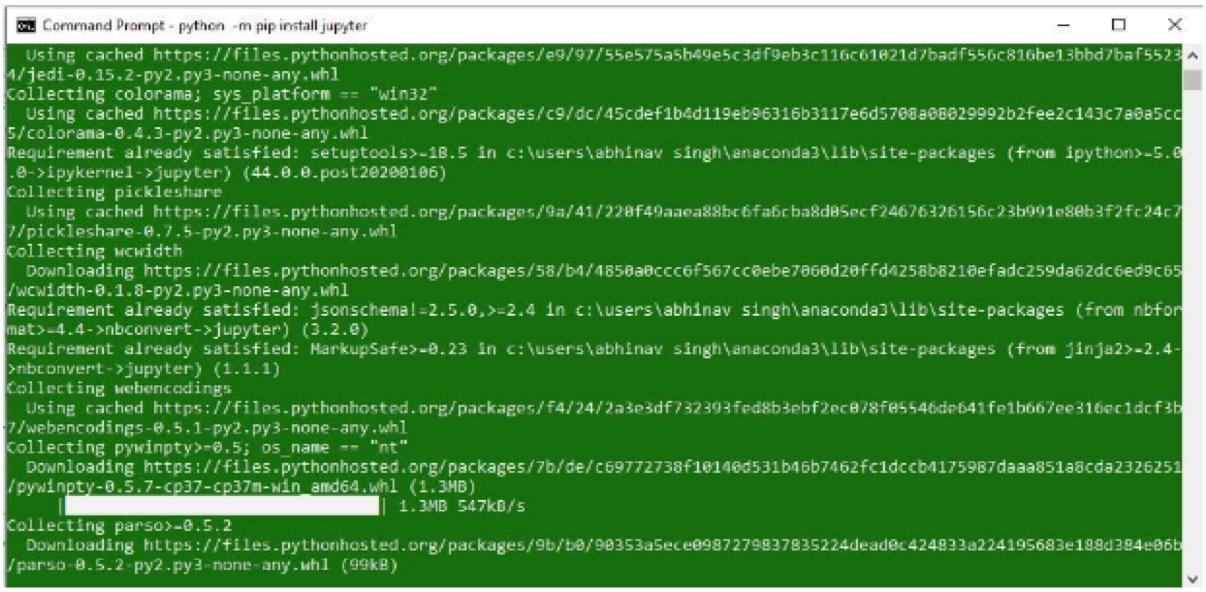
```
python -m pip install --upgrade pip
```

After updating the pip version, follow the instructions provided below to install Jupyter:

- **Command to install Jupyter:**

```
python -m pip install jupyter
```

- **Begin Installation:**



- **Launching Jupyter :** Use the following command in Command Prompt to open Jupyter.

```
jupyter notebook
```

3.2 Data Download

The data is obtained from UK.gov website and can be downloaded in csv format [1]

3.3 Libraries

You may see the library resources that will be required to carry out the research project in this area.

```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
%config InlineBackend.figure_format = 'svg'
plt.style.use('seaborn-dark-palette')
import pandas as pd
import seaborn as sns
sns.set_style('whitegrid')
import contextily
import itertools
```

```
import reverse_geocoder as rg
from scipy.spatial.distance import cdist
import math
from collections import defaultdict


from tqdm import tqdm
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import silhouette_score
from sklearn.datasets import make_blobs
from sklearn.neighbors import KNeighborsClassifier
from ipywidgets import interactive
import folium
import re
cols = ['#e6194b', '#3cb44b', '#ffe119', '#4363d8', '#f58231', '#911eb4',
        '#46f0f0', '#f032e6', '#bcf60c', '#fabebe', '#008080', '#e6beff',
        '#9a6324', '#fffac8', '#800000', '#aaffc3', '#808000', '#ffd8b1',
        '#000075', '#808080']*10
sns.set(style="white")

import plotly.express as px
```

4 Reverse Geocoding

This project tested two reverse geocoding algorithms and selected one based on speed of execution which will be shown in this section.

[1] https://www.gov.uk/government/statistics/reported%2Droad%2Dcasualties%2Dgreat%2Dbritain%2Dannual%2Dreport%2D2019.

4.1 Server Based Geocoding

In [5]:
```
from joblib import Parallel, delayed
```

In [6]:
```
from geopy.geocoders import Nominatim
def getLoc(loc, data):
  # locations = pd.DataFrame()
  geolocator = Nominatim(user_agent="geoapiExercises")
  lat= data.at[loc,'latitude']
  lng= data.at[loc,'longitude']
  location= geolocator.reverse(str(lat)+","+str(lng))
  address = location.raw['address']
  city= address.get('city','')
  print(city)
  return city
  # return locations.append({"city":city},ignore_index=True)
```

In [7]:
```
executor = Parallel(n_jobs=2)
tasks= (delayed(getLoc)(loc,df) for loc in range(100))
locations= executor(tasks)
```

4.2 Offline Geocoding

```
loc = df.iloc[:, 18:20]
tuples = [tuple(x) for x in loc.to_numpy()]
def getLoc1(df,x,y):
    coords=tuple(zip(df[x].iloc,df[y]))
    address= rg.search(coords)
    city= [x.get('admin2') for x in address]
    return city
```

```
city1 = []
city1 = getLoc1(df_clean,'latitude','longitude')
```

```
Loading formatted geocoded file...
```

5 Clustering

This section explains the different clustering techniques used.

5.1 Map Function

```
In [23]: def create_map(data,cluster_col):
    m = folium.Map(location=[data.latitude.mean(), data.longitude.mean()], zoom_start=9, tiles='openstreetmap')

    for _, row in data.iterrows():

        # get a colour
        if row[cluster_col]==-1:
            cluster_colour='black'
        else:
            cluster_colour = cols[row[cluster_col]]

        folium.CircleMarker(
            location=[row.latitude,row.longitude],
            radius=5,
            popup= row[cluster_col],
            color=cluster_colour,
            fill=True,
            fill_color=cluster_colour
        ).add_to(m)
    return m
```

5.2 K-means

```
In [24]: k_range=range(5,55,5)
kmeans_per_k=[]
for k in k_range:
    kmeans=KMeans(n_clusters=k,random_state=2, n_init = 300).fit(kmeans_coords)
    kmeans_per_k.append(kmeans)
```

5.2.1 Silhouette score

```
In [34]: silh_scores=[silhouette_score(kmeans_coords,model.labels_) for model in kmeans_per_k]
best_index = np.argmax(silh_scores)
best_k = k_range[best_index]
best_score = silh_scores[best_index]
print("best k value:",best_k)
print("silhouette score:",best_score)

plt.figure(figsize=(8, 3))
plt.grid(True)
plt.plot(k_range, silh_scores, "bo-")
plt.xlabel("k", fontsize=14)
plt.ylabel("Silhouette score", fontsize=14)
plt.plot(best_k, best_score, "rs")
plt.show()
```

```
best k value: 10
```

5.2.2 Inertia

In [35]:
```
inertias = [model.inertia_ for model in kmeans_per_k]
best_inertia = inertias[best_index]

plt.figure(figsize=(8, 3.5))
plt.grid(True)
plt.plot(k_range, inertias, "bo-")
plt.xlabel("$k$", fontsize=14)
plt.ylabel("Inertia", fontsize=14)
plt.plot(best_k, best_inertia, "rs")
plt.show()
```

5.2.3 Clustering for k= 20

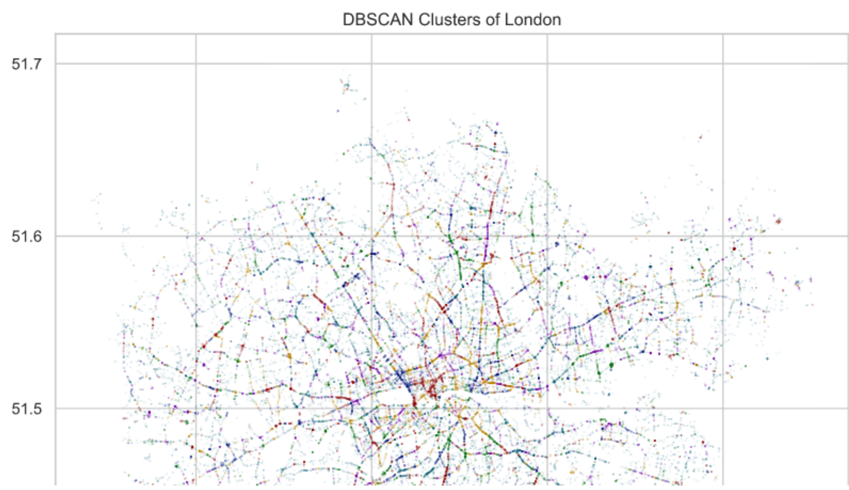
In [28]:
```
kmeans = KMeans(n_clusters=20, random_state=42, n_init = 300).fit(kmeans_coords)
kmeans_coords['label'] = kmeans.labels_


kmeans_coords = kmeans_coords.sample(80000)
plt.figure(figsize = (10,10))
for label in kmeans_coords.label.unique():
    plt.plot(kmeans_coords.longitude[kmeans_coords.label == label],kmeans_coords.latitude[kmeans_coords.label == label],
             '.', alpha = 0.3, markersize = 0.3)

plt.title('Clusters of London k=20')
plt.show()
```

5.2.4 Clustering for k=40

```
kmeans_coords1 = pd.DataFrame()
kmeans_coords1['longitude'] = longitude_lon
kmeans_coords1['latitude'] = latitude_lon
kmeans = KMeans(n_clusters=40, random_state=42, n_init = 300).fit(kmeans_coords1)
kmeans_coords1['label'] = kmeans.labels_

kmeans_coords1 = kmeans_coords1.sample(80000)
plt.figure(figsize = (10,10))
for label in kmeans_coords1.label.unique():
    plt.plot(kmeans_coords1.longitude[kmeans_coords1.label == label],kmeans_coords1.latitude[kmeans_coords1.label == label],'.',

plt.title('Clusters of London k=40')
plt.show()
```

5.3 DBSCAN

In [32]:
```
db = DBSCAN(eps=0.001, min_samples=5, algorithm='ball_tree', metric='haversine').fit(db_coords)
db_coords['label'] = db.labels_

db_coords = db_coords.sample(80000)
plt.figure(figsize = (10,10))
for label in db_coords.label.unique():
    plt.plot(db_coords.longitude[db_coords.label == label],db_coords.latitude[db_coords.label == label],
            '.', alpha = 0.3, markersize = 0.3)

plt.title('DBSCAN Clusters of London')
plt.show()
```



In [64]:
```
print(f'Number of clusters found: {len(np.unique(class_predictions))}')
print(f'Number of outliers found: {len(class_predictions[class_predictions==-1])}')

print(f'Silhouette ignoring outliers: {silhouette_score(db_coords[class_predictions!=-1], class_predictions[class_predictions!=-1

no_outliers = 0
no_outliers = np.array([(counter+2)*x if x==-1 else x for counter, x in enumerate(class_predictions)])
print(f'Silhouette outliers as singletons: {silhouette_score(db_coords, no_outliers)}')
plt_map=create_map(df0,'Clusters_dbscan')
plt_map.save('DBSCAN_map.html')
plt_map
```

```
Number of clusters found: 10
Number of outliers found: 24
Silhouette ignoring outliers: 0.8135760336584489
Silhouette outliers as singletons: 0.8003583908677331
```