Python predictive model

The model developed in python is able to predict the amount of wind and solar power produced in Northern Ireland over a five-day period.

Firstly, the wind power model was developed. The python library ‘pandas’ was used to read in the provided excel list of all wind turbines in Northern Ireland.

```
#read in the wind turbine date

wind = pd.read_excel('turbine_list_student (1).xlsx')
wind.head(5)
```

	Name	Address	t_capacity[MW]	Num_turbine	tot_capacity[MW]	latd	longd
0	Abbey Road, 209 Turbine 1	305 East of 209 Abbey Road, Newtownards	0.25	1	0.25	54.5921	-5.54138
1	Abbey Road, 209 Turbine 2	580m NE of 209 Abbey Road, Millisle	0.25	1	0.25	54.5941	-5.53866
2	Aghaboy Road, 24a	460m SW of 24a Aghaboy Road, Omagh, BT79 7QY	0.25	1	0.25	54.6794	-7.13111
3	Aghalougher Road, 28	450m South of 28 Aghalougher Road, Castlederg,...	0.25	1	0.25	54.6623	-7.70331
4	Aghincurk Road, 3	400m north west of 3 Aghhincurk Road Newtownha...	0.25	1	0.25	54.2108	-6.56825

```
#assign tables values to variables

lat = wind['latd'].values
lon = wind['longd'].values
capacity = wind['tot_capacity[MW]']
```

```
len(lat)
```

```
643
```

Table 1: Turbine data

The crux of the code is two “for” loops. The first loop gathers forecasted weather data from Open Weather API for each turbine. Data is collected every three hours for five days, for the 643 turbines given, meaning there are more than 25,000 data points.

```
#initialize total mwhr

total_MWhr=0
```

```
#The first loop gets a response for weather data every 3 hours fpr 5 days in the future at a certain wind turbine.
#The second loop then converts the wind speed to MWhr and outputs a total MWhr



for farm in range (0,len(wind['latd'].values)):
    lat1=lat[farm]
    lon1=lon[farm]
    args = {'lat': lat1, 'lon': lon1, 'appid': '6a849af5aa9413d2533ad0acd144b513'}
    response = requests.get('http://api.openweathermap.org/data/2.5/forecast', params=args)

    time.sleep(1.5) #tells the programme to wait 1.5 seconds so we don't go over the api limit

    res=requests.get(response.url)
    data = res.json()

    count=data['cnt']



    max_capacity=capacity[farm]
```

Figure 3: Collecting the data from Open Weather

The second “for” loop takes each data point and converts the wind speed into MWhr based on the maximum capacity of the turbine farm and the wind speed. A final total is collected, based on the assumption of a cut in speed of 4ms -1 and a cut off speed of 25ms -1 . This total is then output, as shown in the code below:

```
# Assume a cut in speed of 4 m/s and linear to 12 m/s with a cut off at 25 m/s
# Also assume wind at 80m is 40% higher than that at 10m which the api predicts


    for x in range(0, count):
        wind_speed=data['list'][x]['wind']['speed']

        wind_speed=wind_speed*1.4

        if wind_speed < 4 or wind_speed > 25:
            y = 0
        if wind_speed >= 4 and wind_speed <= 12:
            y = (1/8)*wind_speed-0.5
        if wind_speed > 12 and wind_speed <= 35:
            y = 1
        output=y*max_capacity*3/1000 #each time step is 3 hours and capactiy is in kW, so this is the MW.hr produced
        total_MWhr+=output




print("total_MWhr = ", total_MWhr)
```

Figure 4: Code used to generate wind MWh

Next, the solar part of the model could be developed from data provided for the solar farms in Northern Ireland. This data was first converted into python class variables to increase accessibility.

```
class Solar:
    def __init__(self,lat,long,capacity):

        self.lat=lat
        self.long=long
        self.capacity=capacity
        self.efficiency=0.086 #average effiency in clear sky from the experimental data
        self.area=(capacity*10**6)/100 #based on 1KW solar panel= 10m^2


Dunmanbridge= Solar(54.629270,-5.912440,5)
Belfast_city_airport=Solar(54.624741,-5.854350,4.83)
Ballygarvey=Solar(54.885239,-6.237280,8)
Greater_Belfast=Solar(54.597286,-5.930120,200)

solar_lst=[Dunmanbridge,Belfast_city_airport,Ballygarvey,Greater_Belfast]


Belfast = pytz.timezone('Europe/Belfast')
```

Figure 5: Development of solar class

Some assumptions were made at this point, namely the area of a solar panel and the average efficiency. From literature it was seen a reasonable estimate for the solar panel area was 10m 2 per
kilowatt produced. (Simhan, 2014) The efficiency value was taken from the supporting materials given on Canvas.
Once again, the crux of the code is two “for” loops. The first loop obtains forecasted weather data
from Open Weather API for each solar farm. Data is collected in the same manner as for the wind
turbines; every three hours for five days.

```
for solar in solar_lst:
    lat1=solar.lat
    lon1=solar.long
    args = {'lat': lat1, 'lon': lon1, 'appid': '6a849af5aa9413d2533ad0acd144b513'}
    response = requests.get('http://api.openweathermap.org/data/2.5/forecast', params=args)

    time.sleep(1.5) #tells the programme to wait 1.5 seconds so we don't go over the api limit

    res=requests.get(response.url)
    data = res.json()

    count=data['cnt']
    date=data['list'][0]['dt_txt']
```

Figure 6: Collecting solar data

The second “for” loop finds the solar incidence for each data point. To do this, pysolar is used, which requires altitude. This is obtained from the get_altitude function and the time data. An issue was encountered at this point as the time data from Open Weather is in the wrong format and must be adjusted to be usable. The code seen below converts the time from a unix timestamp to a datetime object that is localized with the Belfast time zone.

```
for x in range(0, count):
    loc_dt=data['list'][x]['dt_txt']

    yr=loc_dt[0:2]
    mnth=loc_dt[5:7]
    day=loc_dt[8:10]
    hour=loc_dt[11:13]
    minte=loc_dt[14:16]

    dt=datetime.strptime(f"{day}/{mnth}/{yr} {hour}:{minte}", "%d/%m/%y %H:%M")

    loc_dt2 = Belfast.localize(dt)
```

Figure 7: Adjusting the time format

Once pysolar has calculated the solar incidence, power is calculated using Equation 1.

Power = (rad) · (efficiency) · (area)

Equation 1

The cloud cover percentage is then taken from the Open Weather API data. The total output poweris then calculated using Equation 2, which takes the cloud cover into account.

P out (P max , Cloud) = P max (1 − 0.75 · Cloud 3)

Equation 2

The final power is then converted into MWhr.

```
    for x in range(0, count):
        loc_dt=data['list'][x]['dt_txt']

        yr=loc_dt[0:2]
        mnth=loc_dt[5:7]
        day=loc_dt[8:10]
        hour=loc_dt[11:13]
        minte=loc_dt[14:16]

        dt=datetime.strptime(f"{day}/{mnth}/{yr} {hour}:{minte}", "%d/%m/%y %H:%M")

        loc_dt2 = Belfast.localize(dt)

        alt = get_altitude(lat1, lon1,loc_dt2)
        rad =(radiation.get_radiation_direct(loc_dt2, alt))

        cloud=data['list'][x]['clouds']['all']/100
        power=rad*(solar.efficiency)*(solar.area)

        output=power*(1-(0.75*cloud**3))

        total_MWhr+=(output/1000000)*3



print("total_MWhr = ", total_MWhr)
```

Figure 8: Total solar power

The next stage was to run and test the model. The script was cleaned up and made into a single function so it could run as one script. A problem was encountered when running the script because API restricts the number of data points that can be obtained per minute to 60. This makes it challenging to test the model against historical data due to the length of time it would take, so It was decided to group the turbines by location to speed this process up.

To create the clustering algorithm,we started with an equation from literature. (Boeing, 2018) This enabled us to specify a maximum distance apart the individual wind farms could be to be considered a cluster.

Spatial data clustering with DBSCAN

[7]:
```
kms_per_radian=6371.0088
epsilon=4.6/kms_per_radian
db=DBSCAN(eps=epsilon,min_samples=1,algorithm='ball_tree',metric='haversine').fit(np.radians(coords))
cluster_labels=db.labels_
num_clusters=len(set(cluster_labels))
clusters=pd.Series([coords[cluster_labels==n] for n in range(num_clusters)])
print(f'Number of clusters:{num_clusters}')


#the value for epsilon was made to give a good middle ground between lots of clusters and long program run time. i.e lots of clusters=lots of api calls
```

```
Number of clusters:75
```

Figure 9: Clustering code

The middle point of each cluster was found to produce wind data for each cluster.

[8]:
```
def get_centermost_point(cluster):
    centroid = (MultiPoint(cluster).centroid.x, MultiPoint(cluster).centroid.y)
    centermost_point = min(cluster, key=lambda point: great_circle(point, centroid).m)
    return tuple(centermost_point)

centermost_points = clusters.map(get_centermost_point)
```

[19]:
```
lats, lons = zip(*centermost_points)
rep_points = pd.DataFrame({'lon':lons, 'lat':lats})

rep_points
```

[19]:

	lon	lat
0	-5.52905	54.5485
1	-7.03046	54.5744
2	-7.71028	54.6503
3	-6.49004	54.2384
4	-5.92373	54.2459
...	...	...
70	-7.06361	54.8822
71	-7.33500	54.3044
72	-6.72249	54.8753
73	-6.64750	54.7689
74	-6.73801	54.3253

75 rows × 2 columns

Figure 10: Finding midpoints

The clusters were then plotted to check the validity of the groupings.

```
colours = ['black','gray','rosybrown','lightcoral','indianred','brown','firebrick','maroon','darkred','red','tomato','darksalmon','coral','orangered','sienna','chocolate','sadd
vectorizer = np.vectorize(lambda x: colors[x % len(colors)])
```

```
fig, ax = plt.subplots(figsize=[10, 6])

for counter,cluster in enumerate (clusters):
    colour=colours[counter]

    for point in cluster:
        ax.scatter(point[1], point[0],c=colour)


ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
plt.show()
```

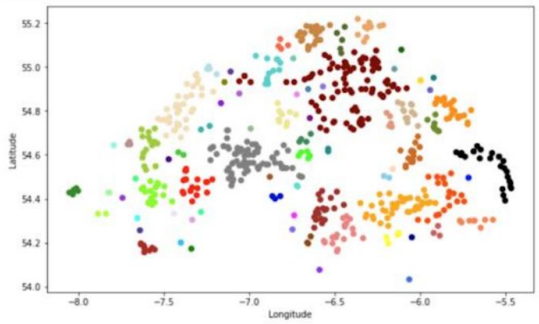

Figure 11: Map of clusters

The data frame could then be created for the midpoints of all the clusters.

creating a data frame of the centremost points

```
]: rs = rep_points.apply(lambda row: wind[(wind['latd']==row['lat']) & (wind['longd']==row['lon'])].iloc[0], axis=1)

   rs.head(5)
```

]:

	Name	Address	t_capacity[MW]	Num_turbine	tot_capacity[MW]	latd	longd
0	Tullykevin Road, 16	150m northwest of 16 Tullykevin Road, Newtowna...	0.225	1	0.225	54.5485	-5.52905
1	Inishative Road, 17	117m North of 17 Inishative Road, Carrickmore,...	0.150	1	0.150	54.5744	-7.03046
2	Church Hill	Church Hill, Meenamullan Road, Killeter, Castl...	2.300	8	18.400	54.6503	-7.71028
3	Tullyherron Road, 19	815m north west of No 19 Tullyherron Road, Cor...	0.225	1	0.225	54.2384	-6.49004
4	Dundrinne Road, 63	285m SW of 63 Dundrinne Road, Castlewellan, BT...	0.225	1	0.225	54.2459	-5.92373

Figure 12: Data frame

Total_capacity[MW] was updated to represent the total capacity of the cluster, rather than just the capacity of the centre point in the cluster.

```
capacity_d={}

for counter,cluster in enumerate(clusters):
    total_capacity=0
    for point in cluster:

        c=wind[(wind['latd']==point[0])& (wind['longd']==point[1])]['tot_capacity[MW]']
        lst=c.tolist()
        intg=lst[0]

        total_capacity+=intg



    rs.at[counter,'tot_capacity[MW]']=total_capacity
```

```
rs
```

	Name	Address	t_capacity[MW]	Num_turbine	tot_capacity[MW]	latd	longd
0	Tullykevin Road, 16	150m northwest of 16 Tullykevin Road, Newtowna...	0.225	1	6.350	54.5485	-5.52905
1	Inishative Road, 17	117m North of 17 Inishative Road, Carrickmore,...	0.150	1	192.975	54.5744	-7.03046
2	Church Hill	Church Hill, Meenamullan Road, Killeter, Castl...	2.300	8	67.250	54.6503	-7.71028
3	Tullyherron Road, 19	815m north west of No 19 Tullyherron Road, Cor...	0.225	1	4.450	54.2384	-6.49004
4	Dundrinne Road, 63	285m SW of 63 Dundrinne Road, Castlewellan, BT...	0.225	1	0.625	54.2459	-5.92373
...	...	...	...	...	...	...	...
70	Straid Road, 41	Lands 585m east of 41 Straid Road, Claudy	0.250	1	0.250	54.8822	-7.06361
71	Teiges Mountain	County Fermanagh	2.500	5	12.500	54.3044	-7.33500
72	Tirkane Road, 150	290 Metres South West of 150 Tirkane Road, Mag...	0.225	1	0.225	54.8753	-6.72249
73	Tobermore Concrete Products	Lisnamuck Road, Tobermore	0.250	1	0.250	54.7689	-6.64750
74	Tullycallidy Road, 12	367m east of no.12 Tullycallidy Road, BT60 4NE	0.250	1	0.250	54.3253	-6.73801

75 rows × 7 columns

Figure 13: Clusters

The wind power section of the model was then ready to be tested. As there are now 75 clusters, the run time is quicker. Running the programme gave a prediction for the day of 456 MWh.

Comparing this to the values taken from EirGrid Group's live wind prediction software, it can be seen that the actual wind generation for the test day (13 th November) was 500 MWh. This value was found by calculating a mean of the wind generated for each hour. It can be seen from the data that EirGrid predicted 486 MWh would be generated on average.

Their forecast was 2.8% less than the actual wind generated, and this python model was 8.8% away from the real value. This is an acceptable level of disparity and it is therefore verified that this model is accurate. (EirGrid, 2020)

```
#The first loop gets a response for historical weather data for every farm at one hour intervals for the current day
#The second loop then converts the wind speed to MWhr and outputs a total MWhr.
#total MWhr is then converted to an average

for farm in range (0,len(wind['latd'].values)):
    lat1=lat[farm]
    lon1=lon[farm]
    args = {'lat': lat1, 'lon': lon1, 'dt':time_now, 'appid': '6a849af5aa9413d2533ad0acd144b513'}
    response = requests.get('http://api.openweathermap.org/data/2.5/onecall/timemachine', params=args)

    time.sleep(1.1) #tells the programme to wait 1.1 seconds so we don't go over the api limit

    res=requests.get(response.url)
    data = res.json()

    max_capacity=capacity[farm]


    for x in range(0, count):

        #only read the hourly json data. if the program is run at 16:46, 16 data points will be returned at every hour from 00:00

        if x==0:
            continue

        else:
            wind_speed=data['hourly'][x-1]['wind_speed']


        wind_speed=wind_speed*1.4

        if wind_speed < 4 or wind_speed > 25:
            y = 0
        if wind_speed >= 4 and wind_speed <= 12:
            y = (1/8)*wind_speed-0.5
        if wind_speed > 12 and wind_speed <= 35:
            y = 1
        output=y*max_capacity*1 #this is the MW produced

        total_MWhr+=output


avg_MWhr=total_MWhr/count #convert mw to mw/hr
```

```
n_t=datetime.fromtimestamp(data['hourly'][count-1]['dt'])
f_t=datetime.fromtimestamp(data['hourly'][0]['dt'])
print("total MWhr = ", total_MWhr)
print("avg_MWhr = ", avg_MWhr)
print(f'from {n_t} to {f_t}')
```

```
total_MWhr =  6390.705451249997
avg_MWhr =  456.4789608035712
from 2020-11-14 13:00:00 to 2020-11-14 00:00:00
```

Figure 14: Testing Wind

Running the data without clustering gives a value of 447, which is approximately 2% different to the value using cluster. This shows that clustering is not the main reason for the difference between the values generated by the model and those seen in literature. This difference may be attributed to the power curve that was used, as it was not specific to each turbine.

To test the solar part of the model, it was suggested to use data from a PV site in Newquay. Using data from the PV panel manufacturer's website gave suitable data to test the model. (Renugen, 2020) The total capacity of the solar panels is 3995W and the average efficiency is 0.163.

When the model is run, the solar MWh is considerably larger than that seen in the Newquay results. This is because the data is not specific to Northern Ireland, but this verifies that the model runs and works. The panel angle also has an impact, but this is beyond the scope of this model.

In conclusion, the model runs successfully and the wind data, is particular, is extremely accurate. The solar data is less accurate, but this is due to a lack of relevant historical data.

Bibliography

Boeing, G., 2018. Clustering to Reduce Spatial Data Set Size.

EirGrid, 2020. Wind Generation. [Online]
Available at: http://smartgriddashboard.eirgrid.com/#all/wind
[Accessed 13 November 2020].

Renugen, 2020. Suntech STP250S-20/Wd 250 Watt Solar Panel Module. [Online]
Available at: https://www.renugen.co.uk/suntech-stp250s-20-wd-250-watt-solar-panel-module-discontinued/
[Accessed 14 November 2020].

Simhan, N., 2014. Area Required for Solar PV Power Plants. [Online]
Available at: http://www.suncyclopedia.com/en/area-required-for-solar-pv-power-plants/
[Accessed 03 November 2020].