In [335…

```
import pandas as pd #Loading pandas library
import numpy as np #Loading numerical array library
import matplotlib.pyplot as plt #Loading level graph plotting library
pd.options.display.float_format = '{:.2f}'.format #for removing e values and showning i
%matplotlib inline #to graphs inline
```

```
UsageError: unrecognized arguments: #to graphs inline
```

For Office

In [217…

```
office = pd.read_excel('office-1.xlsx') #Loading office excel file
```

In [218…

```
office.info() #concise summary
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   month                  8760 non-null   int64  
 1   day                    8760 non-null   int64  
 2   hour                   8760 non-null   int64  
 3   Drybulb Temperature    8760 non-null   float64
 4   Wetbulb Temperature    8760 non-null   float64
 5   Relative Humidity      8760 non-null   int64  
 6   Wind Speed             8760 non-null   float64
 7   Wind Direction         8760 non-null   int64  
 8   Solar Radiation        8760 non-null   int64  
 9   Sky Clearness          8760 non-null   float64
 10  Total Electric Demand  8760 non-null   float64
 11  HVAC Electric Demand   8760 non-null   float64
dtypes: float64(6), int64(6)
memory usage: 821.4 KB
```

In [219…

```
office.describe() #summary of statistics pertaining
```

Out[219…

	month	day	hour	Drybulb Temperature	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation
count	8760.00	8760.00	8760.00	8760.00	8760.00	8760.00	8760.00	8760.00	8760.00
mean	6.53	15.72	11.50	27.14	20.66	60.65	3.64	203.73	261.98
std	3.45	8.80	6.92	7.39	4.69	21.54	2.25	111.86	343.13
min	1.00	1.00	0.00	5.00	5.00	6.00	0.00	0.00	0.00
25%	4.00	8.00	5.75	21.50	17.16	44.00	2.10	100.00	0.00
50%	7.00	16.00	11.50	27.00	20.81	63.00	3.10	200.00	0.00
75%	10.00	23.00	17.25	32.50	24.24	78.00	5.10	310.00	605.00
max	12.00	31.00	23.00	47.00	30.76	100.00	24.20	360.00	975.00

For Apartment

In [220…

```
apartment = pd.read_excel('apartment-1.xlsx') #Loading apartment excel file
```

In [221…

```
apartment.info() #concise summary
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   month                  8760 non-null   int64
 1   day                    8760 non-null   int64
 2   hour                   8760 non-null   int64
 3   Drybulb Temperature    8760 non-null   float64
 4   Wetbulb Temperature    8760 non-null   float64
 5   Relative Humidity      8760 non-null   int64
 6   Wind Speed             8760 non-null   float64
 7   Wind Direction         8760 non-null   int64
 8   Solar Radiation        8760 non-null   int64
 9   Sky Clearness          8760 non-null   float64
 10  Total Electric Demand  8760 non-null   float64
 11  HVAC Electric Demand   8760 non-null   float64
dtypes: float64(6), int64(6)
memory usage: 821.4 KB
```

In [222…

```
apartment.describe() #summary of statistics pertaining
```

Out[222…

	month	day	hour	Drybulb Temperature	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation
count	8760.00	8760.00	8760.00	8760.00	8760.00	8760.00	8760.00	8760.00	8760.00
mean	6.53	15.72	11.50	27.14	20.66	60.65	3.64	203.73	261.98
std	3.45	8.80	6.92	7.39	4.69	21.54	2.25	111.86	343.13
min	1.00	1.00	0.00	5.00	5.00	6.00	0.00	0.00	0.00
25%	4.00	8.00	5.75	21.50	17.16	44.00	2.10	100.00	0.00
50%	7.00	16.00	11.50	27.00	20.81	63.00	3.10	200.00	0.00
75%	10.00	23.00	17.25	32.50	24.24	78.00	5.10	310.00	605.00
max	12.00	31.00	23.00	47.00	30.76	100.00	24.20	360.00	975.00

In [ ]:

1- Compare the total and HVAC electricity consumption of the two buildings on a

monthly and annual basis. (a bar chart is preferred)

MONTHLY TOTAL ELECTIC DEMAND OFFICE VS APARTMENT COMPARISON

In [223…

```
TED = office[["month", "Total Electric Demand"]] #only taking month and total electric
TED_sum = office.groupby(['month']).agg({'Total Electric Demand':['sum']}) #aggrigating

apartment_TED = apartment[["month", "Total Electric Demand"]] #only taking month and to
apartment_TED_sum = apartment_TED.groupby(['month']).agg({'Total Electric Demand':['sum

TED_sum['Apartment Total Electric Demand'] = apartment_TED_sum['Total Electric Demand']
TED_sum
```

Out[223…

	Total Electric Demand	Apartment Total Electric Demand
	sum	
month		
1	44875590.82	16820731.99
2	40684551.46	15193346.36
3	46842264.51	16740075.03
4	42510976.71	15952436.27
5	45089119.66	16501839.51
6	44475965.77	15978919.31
7	43124130.60	16475356.47
8	46220496.51	16515081.03
9	42836664.71	16070290.27
10	44446274.82	16665378.99
11	44097908.30	16346024.79
12	44204822.60	16866417.47

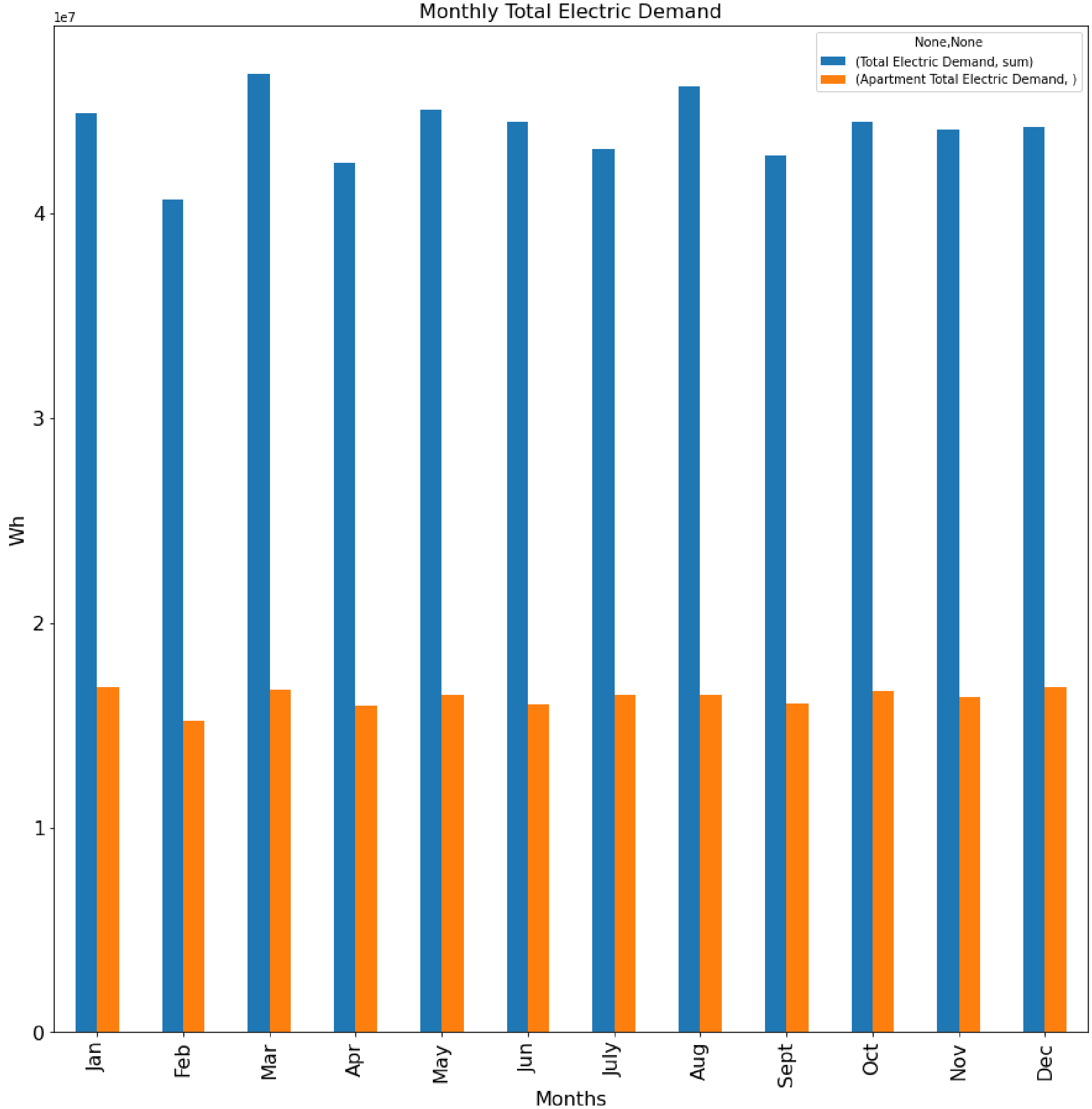
In [224…

```
TED_bar = TED_sum.plot.bar(figsize=(15,15), fontsize=16)
plt.title('Monthly Total Electric Demand', fontsize=16)
plt.xlabel('Months', fontsize=16)
plt.ylabel('Wh', fontsize=16)
TED_bar.set_xticklabels(('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'July', 'Aug', 'Sept
#plotting Office vs Apartment total electric demand graph
```

Out[224…

```
[Text(0, 0, 'Jan'),
 Text(1, 0, 'Feb'),
 Text(2, 0, 'Mar'),
 Text(3, 0, 'Apr'),
 Text(4, 0, 'May'),
 Text(5, 0, 'Jun'),
```

```
Text(6, 0, 'July'),
Text(7, 0, 'Aug'),
Text(8, 0, 'Sept'),
Text(9, 0, 'Oct'),
Text(10, 0, 'Nov'),
Text(11, 0, 'Dec')]
```



In comparison, office total electric demand is 266% more than the apartment total electric consumption. Also, found that summer and winter enrgy demand is almost same in both the buildings.

In [ ]:

In [ ]:

In [ ]:

MONTHLY HVAC ELECTRIC DEMAND OFFICE VS APARTMENT COMPARISON

In [225…

```
HVACED = office[["month", "HVAC Electric Demand"]] #only taking month and HVAC electric
HVACED_sum = HVACED.groupby(['month']).agg({'HVAC Electric Demand':['sum']}) #aggrigati

apartment_HVACED = apartment[["month", "HVAC Electric Demand"]] #only taking month and 
apartment_HVACED_sum = apartment_HVAC.groupby(['month']).agg({'HVAC Electric Demand':['

HVACED_sum['Apartment HVAC Electric Demand'] = apartment_HVACED_sum['HVAC Electric Dema
HVACED_sum
```

Out[225…

	HVAC Electric Demand	Apartment HVAC Electric Demand
	sum	
month		
1	9684090.83	3332421.81
2	10689019.00	4128330.05
3	16055533.29	6112962.84
4	19488421.88	8679103.99
5	29024002.83	14245235.38
6	35957892.47	18279150.64
7	40951356.13	22252593.81
8	44139636.81	22647171.40
9	35298221.78	19295250.59
10	25935376.39	13089089.55
11	19322451.83	8250116.26
12	11883342.90	4512345.58

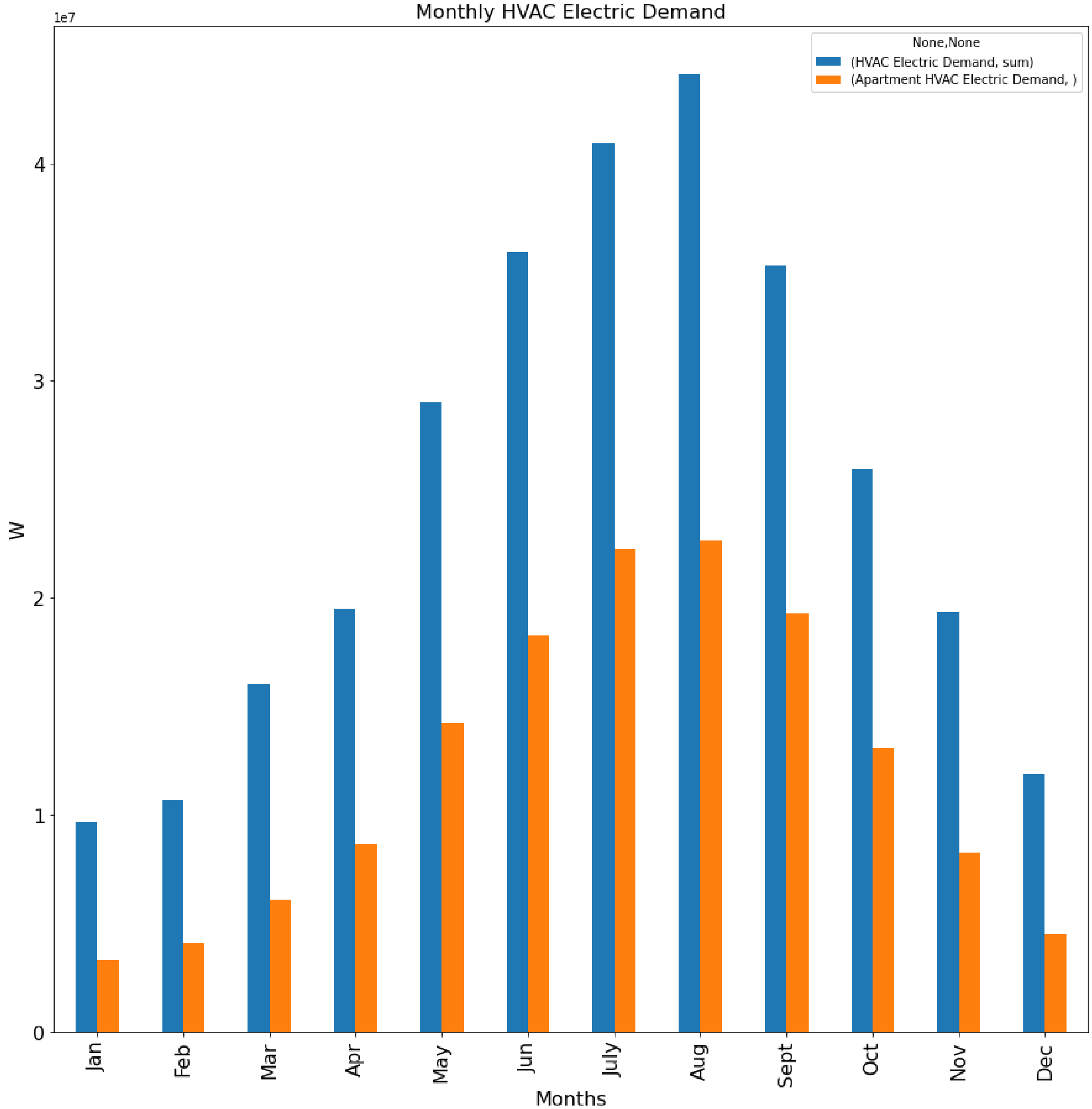
In [226…

```
HVACED_bar = HVACED_sum.plot.bar(figsize=(15,15), fontsize=16)
plt.title('Monthly HVAC Electric Demand',fontsize=16)
plt.xlabel('Months', fontsize=16)
plt.ylabel('W',fontsize=16)
HVACED_bar.set_xticklabels(('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'July', 'Aug', 'S

#plotting Office vs Apartment total electric demand graph
```

Out[226…

```
[Text(0, 0, 'Jan'),
 Text(1, 0, 'Feb'),
 Text(2, 0, 'Mar'),
 Text(3, 0, 'Apr'),
 Text(4, 0, 'May'),
 Text(5, 0, 'Jun'),
 Text(6, 0, 'July'),
 Text(7, 0, 'Aug'),
 Text(8, 0, 'Sept'),
```

```
 Text(9, 0, 'Oct'),
 Text(10, 0, 'Nov'),
 Text(11, 0, 'Dec')]
```



In comparison, office HVAC electric demand is 290% more than the apartment HVAC electric demand.

Office winter HVAC demand is 455% less than the summer HVAC demand.

Apartment winter HVAC demand is 679% less than the summer HVAC demand.

In [ ]:

In [ ]:

In [ ]:

ANNUAL TOTAL ELECTIC DEMAND OFFICE VS APARTMENT COMPARISON

In [227…

```
Office_annual_TED = office['Total Electric Demand'].sum() #Taking annual sum of total e
Office_annual_TED
```

Out[227… 529408766.4447

In [228…

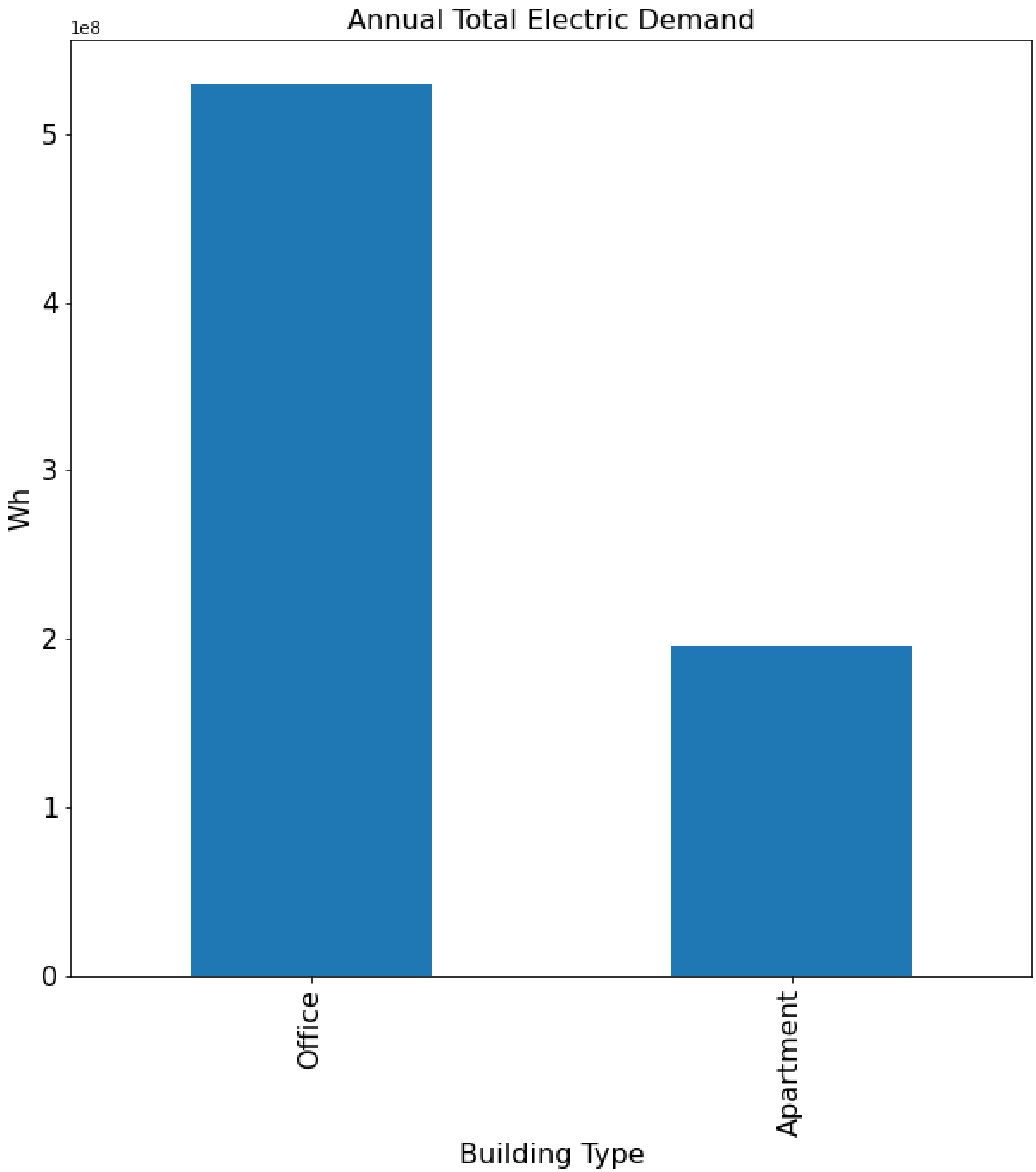
```
apartment_annual_TED = apartment['Total Electric Demand'].sum() #Taking annual sum of t
apartment_annual_TED
```

Out[228… 196125897.48569003

In [229…

```
Annual_TED = TED_sum.sum().plot.bar(figsize=(10,10),fontsize=16)
plt.title('Annual Total Electric Demand',fontsize=16)
plt.xlabel('Building Type',fontsize=16)
plt.ylabel('Wh',fontsize=16)
Annual_TED.set_xticklabels(('Office', 'Apartment'))

#Annual total electric demand comparison graph
```

Out[229… [Text(0, 0, 'Office'), Text(1, 0, 'Apartment')]



In [ ]:

In [ ]:

In [ ]:

ANNUAL HVAC ELECTIC DEMAND OFFICE VS APARTMENT COMPARISON

In [230…

```
Office_annual_HVAC = office['HVAC Electric Demand'].sum() #Taking annual sum of HVAC el
```

```
Office_annual_HVAC
```

Out[230… 298429346.1456469

In [231…

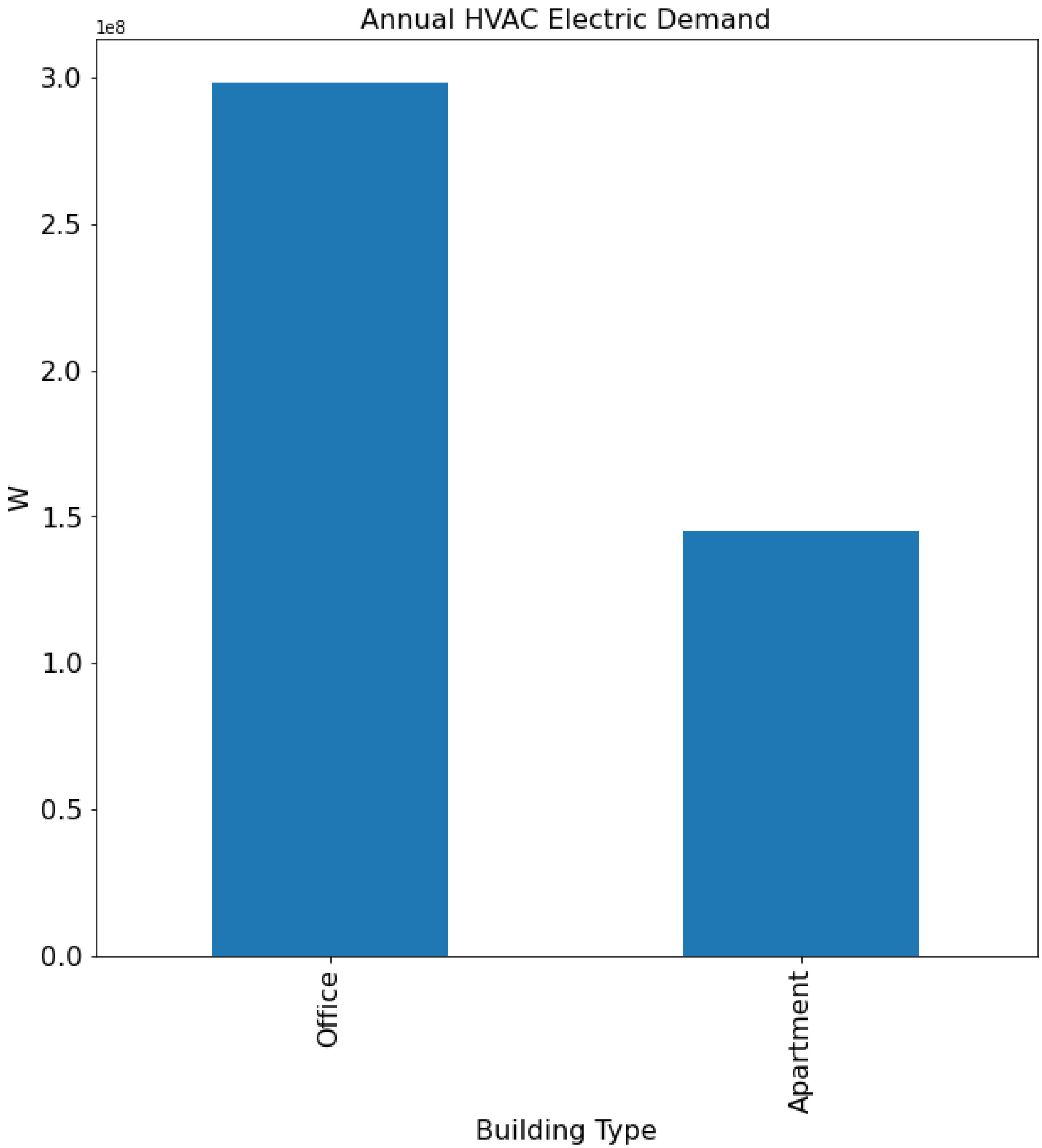
```
apartment_annual_HVAC = apartment['HVAC Electric Demand'].sum() #Taking annual sum of H
apartment_annual_HVAC
```

Out[231… 144823771.8943781

In [232…

```
Annual_HVAC = HVACED_sum.sum().plot.bar(figsize=(10,10),fontsize=16)
plt.title('Annual HVAC Electric Demand',fontsize=16)
plt.xlabel('Building Type',fontsize=16)
plt.ylabel('W',fontsize=16)
Annual_HVAC.set_xticklabels(('Office', 'Apartment'))

#Annual HVAC electric demand comparison graph
```

Out[232… [Text(0, 0, 'Office'), Text(1, 0, 'Apartment')]



In [ ]:

In [ ]:

In [ ]:

2- Plot and describe the distribution of the weather data.

For Office

In [233…

```
Office_weather_data = office[["Drybulb Temperature", "Wetbulb Temperature", "Relative H
Office_weather_data
```

Out[233…

	Drybulb Temperature	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation	Sky Clearness
0	16.00	13.71	78	0.50	190	0	0.00
1	15.60	13.76	82	2.10	120	0	0.00
2	15.10	13.60	85	2.10	120	0	0.00
3	14.80	13.51	87	2.10	140	0	0.00
4	14.40	13.23	88	1.00	150	0	0.00
...	...	...	...	...	...	...	...
8755	17.90	14.18	67	3.60	290	0	0.00
8756	17.50	14.06	69	3.10	270	0	0.00
8757	17.20	14.03	71	2.60	260	0	0.00
8758	16.80	13.89	73	3.10	260	0	0.00
8759	16.50	13.84	75	3.60	270	0	0.00

8760 rows × 7 columns

In [234…

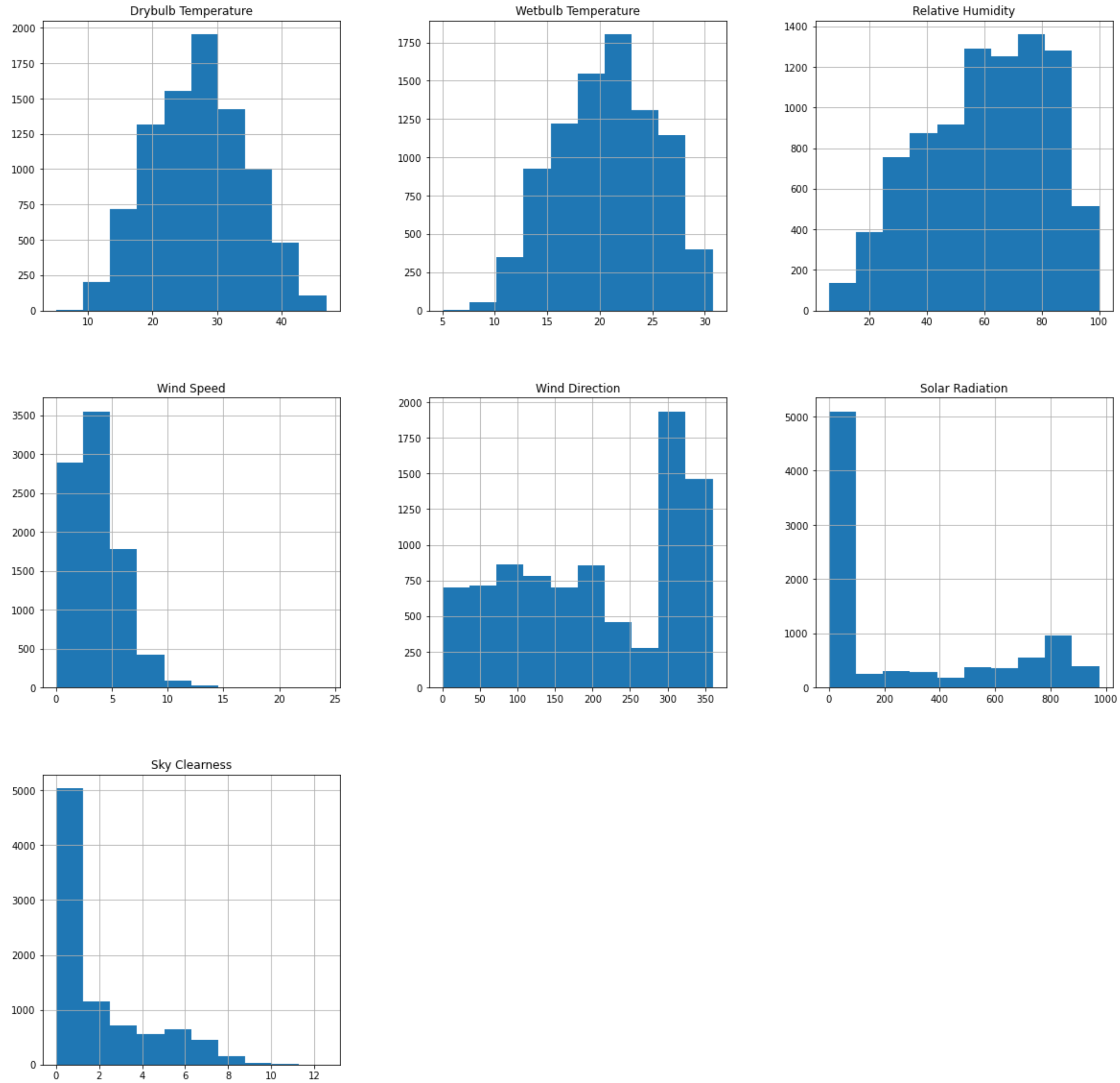
```
Office_weather_data.describe()
```

Out[234…

	Drybulb Temperature	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation	Sky Clearness
count	8760.00	8760.00	8760.00	8760.00	8760.00	8760.00	8760.00
mean	27.14	20.66	60.65	3.64	203.73	261.98	1.81
std	7.39	4.69	21.54	2.25	111.86	343.13	2.38
min	5.00	5.00	6.00	0.00	0.00	0.00	0.00
25%	21.50	17.16	44.00	2.10	100.00	0.00	0.00
50%	27.00	20.81	63.00	3.10	200.00	0.00	1.00
75%	32.50	24.24	78.00	5.10	310.00	605.00	3.10
max	47.00	30.76	100.00	24.20	360.00	975.00	12.56

In [235…

```
Office_weather_data.hist(figsize=(20,20))
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

For Apartment

In [236…

```
apartment_weather_data = apartment[["Drybulb Temperature", "Wetbulb Temperature", "Rela
apartment_weather_data
```

Out[236…

	Drybulb Temperature	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation	Sky Clearness
0	16.00	13.71	78	0.50	190	0	0.00

	Drybulb Temperature	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation	Sky Clearness
1	15.60	13.76	82	2.10	120	0	0.00
2	15.10	13.60	85	2.10	120	0	0.00
3	14.80	13.51	87	2.10	140	0	0.00
4	14.40	13.23	88	1.00	150	0	0.00
...	...	...	...	...	...	...	...
8755	17.90	14.18	67	3.60	290	0	0.00
8756	17.50	14.06	69	3.10	270	0	0.00
8757	17.20	14.03	71	2.60	260	0	0.00
8758	16.80	13.89	73	3.10	260	0	0.00
8759	16.50	13.84	75	3.60	270	0	0.00

8760 rows × 7 columns

In [237…

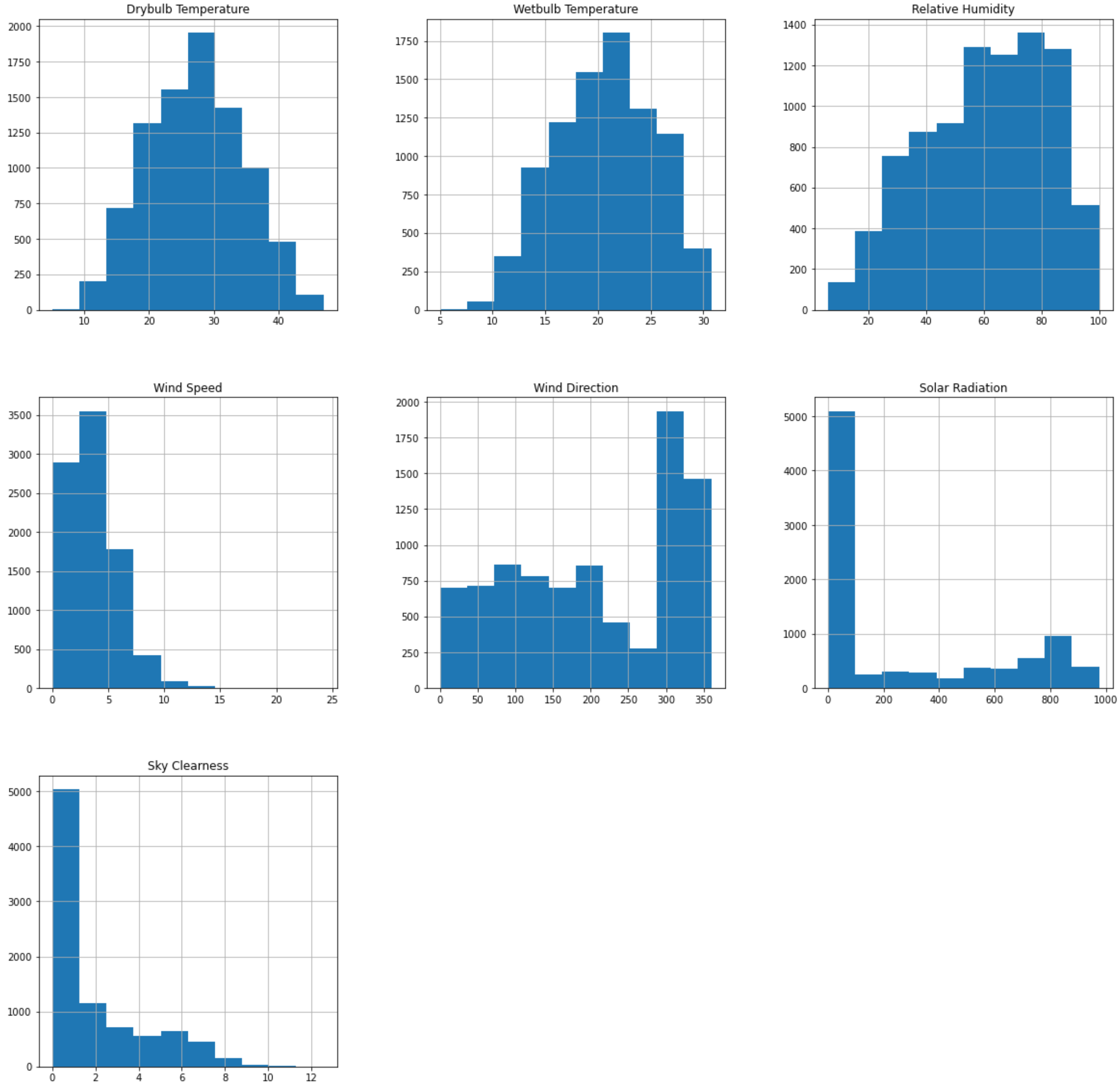
```
apartment_weather_data.describe()
```

Out[237…

	Drybulb Temperature	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation	Sky Clearness
count	8760.00	8760.00	8760.00	8760.00	8760.00	8760.00	8760.00
mean	27.14	20.66	60.65	3.64	203.73	261.98	1.81
std	7.39	4.69	21.54	2.25	111.86	343.13	2.38
min	5.00	5.00	6.00	0.00	0.00	0.00	0.00
25%	21.50	17.16	44.00	2.10	100.00	0.00	0.00
50%	27.00	20.81	63.00	3.10	200.00	0.00	1.00
75%	32.50	24.24	78.00	5.10	310.00	605.00	3.10
max	47.00	30.76	100.00	24.20	360.00	975.00	12.56

In [238…

```
apartment_weather_data.hist(figsize=(20,20))
plt.show()
```



From the Office weather data graph and Apartment weather data graph we can say that the location of both building are same.

Drybulb mean temprature is 27.14 degree

Wetbulb mena temprature is 20.66 degree

Relative Humidity mean is 60.65

wind speed mean is 3.64

wind direction is North South Facing

In [ ]:

In [ ]:

In [ ]:

In [ ]:

3 Report the correlations between weather conditions and HVAC demand for each building.

In [239…

```
office.corr()['HVAC Electric Demand'][3:-2] #finding correclation between HVAC and weat
```

Out[239…

```
Drybulb Temperature    0.69
Wetbulb Temperature    0.43
Relative Humidity     -0.57
Wind Speed             0.40
Wind Direction         0.30
Solar Radiation        0.58
Sky Clearness          0.55
Name: HVAC Electric Demand, dtype: float64
```

In comparison of HVAC and weather condition for office, we can see that the dry bulb has mazimum corrilation with the HVAC demand. After dry bulb, solar radiation, sky clearness and wetbulb are the wather factors which can incease HVAC demand.

In [ ]:

In [ ]:

In [ ]:

In [240…

```
apartment.corr()['HVAC Electric Demand'][3:-2] #finding correclation between HVAC and w
```

Out[240…

```
Drybulb Temperature    0.94
Wetbulb Temperature    0.85
Relative Humidity     -0.50
Wind Speed             0.30
Wind Direction         0.22
Solar Radiation        0.32
Sky Clearness          0.30
Name: HVAC Electric Demand, dtype: float64
```

In comparison of HVAC and weather condition for Apartment, we can see that the dry bulb and wetbulb has mazimum corrilation with the HVAC demand.

In [ ]:

In [ ]:

In [ ]:

4- Create a scatter plot of the weather conditions vs HVAC demand and explain what you can learn from these associations for each building.

For Office

In [241…

```
Office_wc_HVAC = office[["Drybulb Temperature", "Wetbulb Temperature", "Relative Humidi
Office_wc_HVAC
```

Out[241…

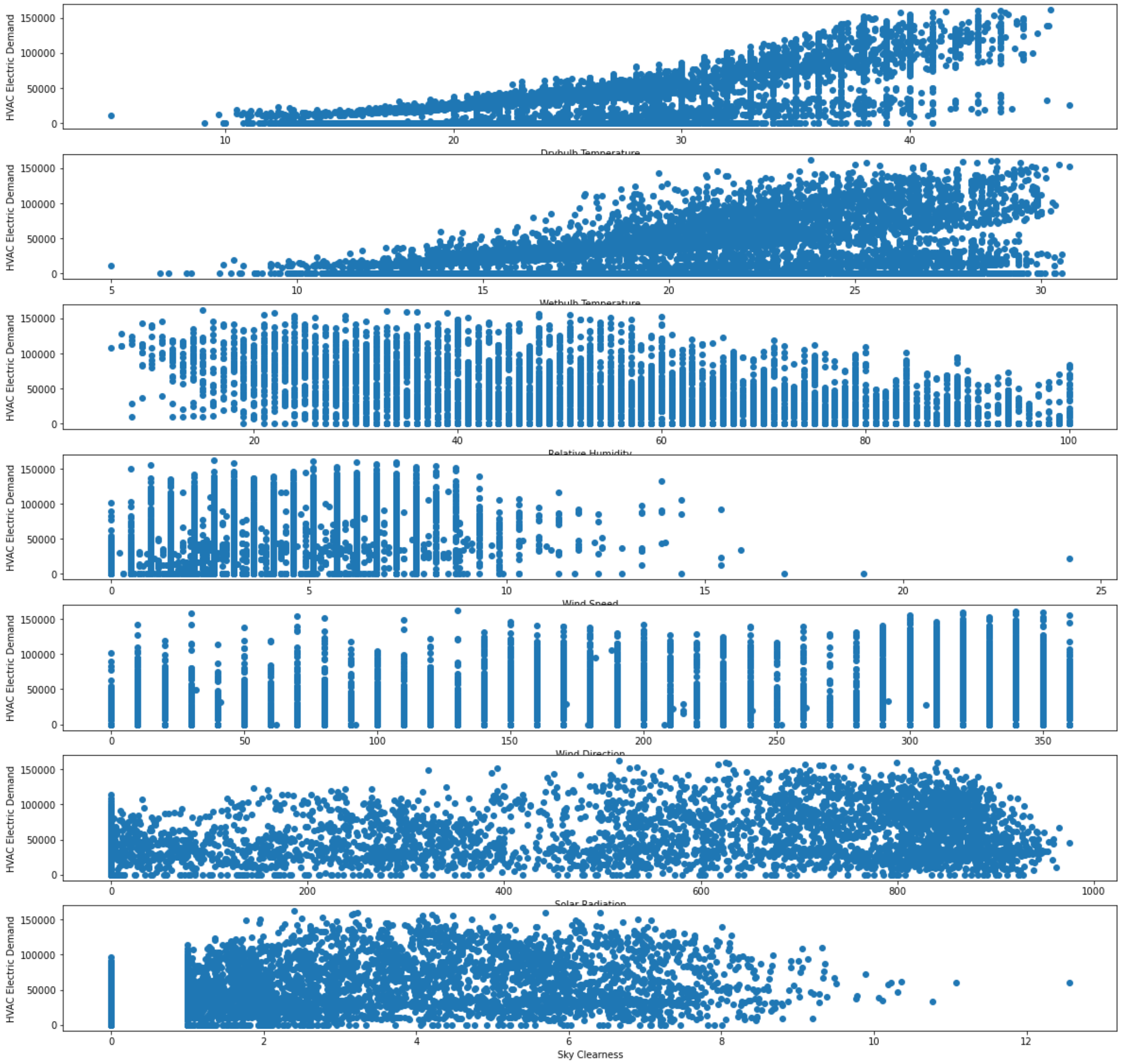
	Drybulb Temperature	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation	Sky Clearness	HVAC Electric Demand
0	16.00	13.71	78	0.50	190	0	0.00	8.44
1	15.60	13.76	82	2.10	120	0	0.00	8.44
2	15.10	13.60	85	2.10	120	0	0.00	8.44
3	14.80	13.51	87	2.10	140	0	0.00	8.44
4	14.40	13.23	88	1.00	150	0	0.00	8.44
...	...	...	...	...	...	...	...	...
8755	17.90	14.18	67	3.60	290	0	0.00	8.44
8756	17.50	14.06	69	3.10	270	0	0.00	8.44
8757	17.20	14.03	71	2.60	260	0	0.00	8.44
8758	16.80	13.89	73	3.10	260	0	0.00	8.44
8759	16.50	13.84	75	3.60	270	0	0.00	8.44

8760 rows × 8 columns

In [242…

```
fig, ax = plt.subplots(7, figsize=(20, 20))

ax[0].scatter(x = Office_wc_HVAC['Drybulb Temperature'], y = Office_wc_HVAC['HVAC Elect
ax[0].set_xlabel("Drybulb Temperature")
ax[0].set_ylabel("HVAC Electric Demand")

ax[1].scatter(x = Office_wc_HVAC['Wetbulb Temperature'], y = Office_wc_HVAC['HVAC Elect
ax[1].set_xlabel("Wetbulb Temperature")
```

```
ax[1].set_ylabel("HVAC Electric Demand")

ax[2].scatter(x = Office_wc_HVAC['Relative Humidity'], y = Office_wc_HVAC['HVAC Electri
ax[2].set_xlabel("Relative Humidity")
ax[2].set_ylabel("HVAC Electric Demand")

ax[3].scatter(x = Office_wc_HVAC['Wind Speed'], y = Office_wc_HVAC['HVAC Electric Deman
ax[3].set_xlabel("Wind Speed")
ax[3].set_ylabel("HVAC Electric Demand")

ax[4].scatter(x = Office_wc_HVAC['Wind Direction'], y = Office_wc_HVAC['HVAC Electric D
ax[4].set_xlabel("Wind Direction")
ax[4].set_ylabel("HVAC Electric Demand")

ax[5].scatter(x = Office_wc_HVAC['Solar Radiation'], y = Office_wc_HVAC['HVAC Electric
ax[5].set_xlabel("Solar Radiation")
ax[5].set_ylabel("HVAC Electric Demand")

ax[6].scatter(x = Office_wc_HVAC['Sky Clearness'], y = Office_wc_HVAC['HVAC Electric De
ax[6].set_xlabel("Sky Clearness")
ax[6].set_ylabel("HVAC Electric Demand")

plt.show()
```



In comparison of HVAC and weather condition for office, we can see that the dry bulb has mazimum corrilation with the HVAC demand. After dry bulb, solar radiation, sky clearness and wetbulb are the wather factors which is incease HVAC demand.

In [ ]:

In [ ]:

In [ ]:

For Apartment

In [243…

```
apartment_wc_HVAC = apartment[["Drybulb Temperature", "Wetbulb Temperature", "Relative
apartment_wc_HVAC
```

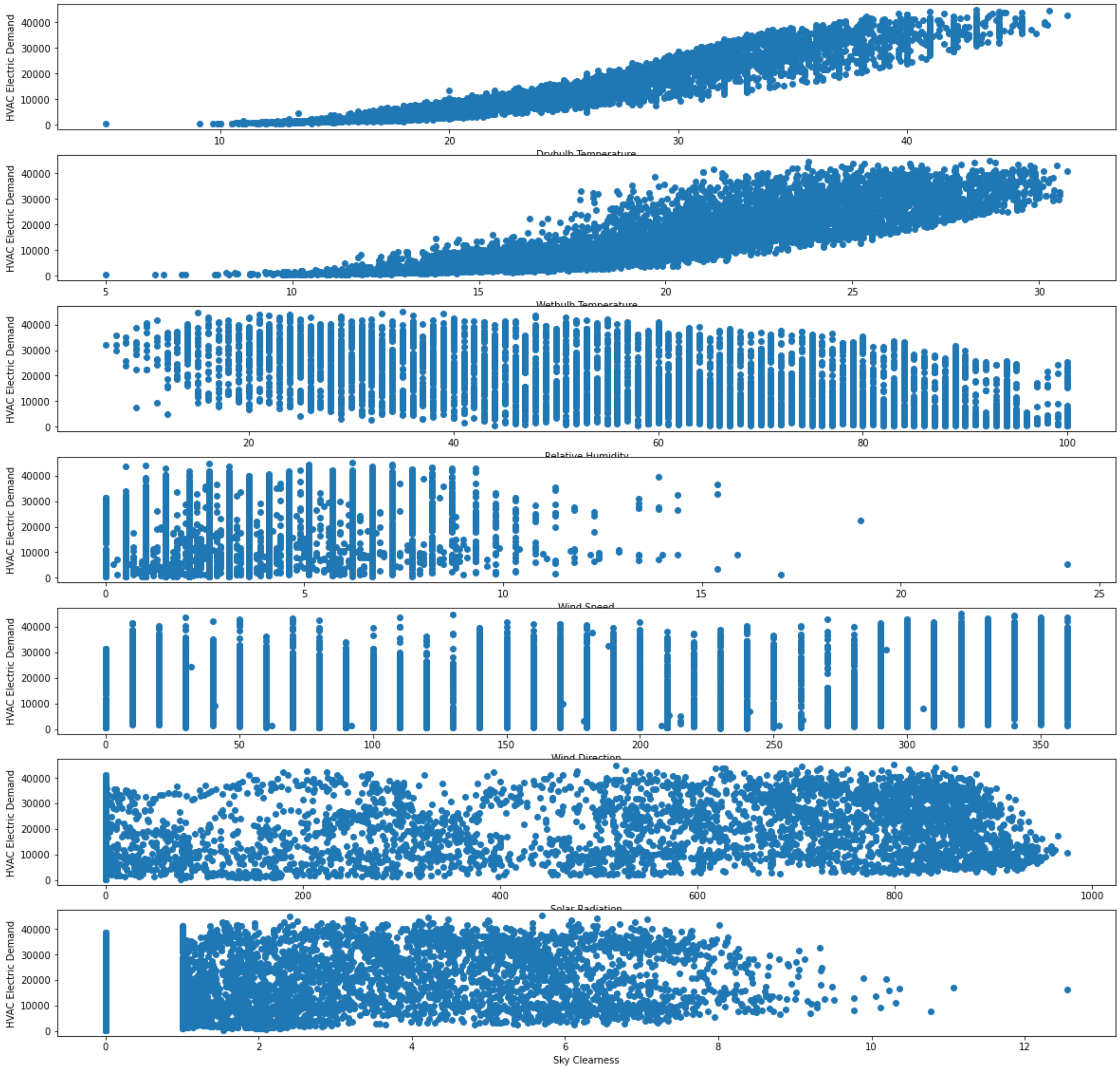
Out[243…

	Drybulb Temperature	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation	Sky Clearness	HVAC Electric Demand
0	16.00	13.71	78	0.50	190	0	0.00	2564.34
1	15.60	13.76	82	2.10	120	0	0.00	1954.42
2	15.10	13.60	85	2.10	120	0	0.00	1600.69
3	14.80	13.51	87	2.10	140	0	0.00	1385.73
4	14.40	13.23	88	1.00	150	0	0.00	1245.54
...	...	...	...	...	...	...	...	...
8755	17.90	14.18	67	3.60	290	0	0.00	5645.49
8756	17.50	14.06	69	3.10	270	0	0.00	5061.28
8757	17.20	14.03	71	2.60	260	0	0.00	4435.61
8758	16.80	13.89	73	3.10	260	0	0.00	3394.96
8759	16.50	13.84	75	3.60	270	0	0.00	2505.03

8760 rows × 8 columns

In [244…

```
fig, ax = plt.subplots(7, figsize=(20, 20))

ax[0].scatter(x = apartment_wc_HVAC['Drybulb Temperature'], y = apartment_wc_HVAC['HVAC
ax[0].set_xlabel("Drybulb Temperature")
ax[0].set_ylabel("HVAC Electric Demand")

ax[1].scatter(x = apartment_wc_HVAC['Wetbulb Temperature'], y = apartment_wc_HVAC['HVAC
ax[1].set_xlabel("Wetbulb Temperature")
ax[1].set_ylabel("HVAC Electric Demand")

ax[2].scatter(x = apartment_wc_HVAC['Relative Humidity'], y = apartment_wc_HVAC['HVAC E
ax[2].set_xlabel("Relative Humidity")
ax[2].set_ylabel("HVAC Electric Demand")

ax[3].scatter(x = apartment_wc_HVAC['Wind Speed'], y = apartment_wc_HVAC['HVAC Electric
ax[3].set_xlabel("Wind Speed")
ax[3].set_ylabel("HVAC Electric Demand")

ax[4].scatter(x = apartment_wc_HVAC['Wind Direction'], y = apartment_wc_HVAC['HVAC Elec
ax[4].set_xlabel("Wind Direction")
ax[4].set_ylabel("HVAC Electric Demand")

ax[5].scatter(x = apartment_wc_HVAC['Solar Radiation'], y = apartment_wc_HVAC['HVAC Ele
ax[5].set_xlabel("Solar Radiation")
ax[5].set_ylabel("HVAC Electric Demand")

ax[6].scatter(x = apartment_wc_HVAC['Sky Clearness'], y = apartment_wc_HVAC['HVAC Elect
ax[6].set_xlabel("Sky Clearness")
ax[6].set_ylabel("HVAC Electric Demand")

plt.show()
```



In comparison of HVAC and weather condition for Apartment, we can see that the dry bulb and wetbulb has mazimum corrilation with the HVAC demand.

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

Question 5: Split the data into training and test with a ratio of 0.2 as the test data.

6- Create a linear regression model and train it based on the training data using weather conditions as the feature set and HVAC demand as the label for each building. 1) Before training, do not forget to standardize your input. 2) Report the MSE value for the training and test data for both buildings.

For Office

In [245…

```
office1 = office.drop(columns = ['month','day','hour','Total Electric Demand']) #Drop a
office1
```

Out[245…

	Drybulb Temperature	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation	Sky Clearness	HVAC Electric Demand
0	16.00	13.71	78	0.50	190	0	0.00	8.44
1	15.60	13.76	82	2.10	120	0	0.00	8.44
2	15.10	13.60	85	2.10	120	0	0.00	8.44
3	14.80	13.51	87	2.10	140	0	0.00	8.44
4	14.40	13.23	88	1.00	150	0	0.00	8.44
...	...	...	...	...	...	...	...	...
8755	17.90	14.18	67	3.60	290	0	0.00	8.44
8756	17.50	14.06	69	3.10	270	0	0.00	8.44
8757	17.20	14.03	71	2.60	260	0	0.00	8.44
8758	16.80	13.89	73	3.10	260	0	0.00	8.44
8759	16.50	13.84	75	3.60	270	0	0.00	8.44

8760 rows × 8 columns

Data split : Test size = 20 %

In [246…

```
from sklearn.model_selection import train_test_split #Split data
train_set, test_set = train_test_split(office1, test_size = 0.2, random_state = 42) #tr
```

In [247…

```
train_set = train_set.reset_index(drop = True) #reset index
test_set = test_set.reset_index(drop = True) #reset index
```

In [248…

```
data = train_set.copy() #create train set as data
```

In [249…
```
corr_mat = data.corr() #corrilation matrix with respect to HVAC electric demand
corr_mat["HVAC Electric Demand"].sort_values(ascending=False)
```

Out[249…
```
HVAC Electric Demand    1.00
Drybulb Temperature     0.69
Solar Radiation         0.58
Sky Clearness           0.55
Wetbulb Temperature     0.44
Wind Speed              0.40
Wind Direction          0.31
Relative Humidity      -0.56
Name: HVAC Electric Demand, dtype: float64
```

In [250…
```
y = data.pop('HVAC Electric Demand') #separating target and data for train
X = data
```

In [251…
```
from sklearn.preprocessing import StandardScaler #standardizing the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled
```

Out[251…
```
array([[-0.0221998 , -1.03900228, -1.42730733, ..., -0.12649305,
         1.30081093,  1.08809622],
       [ 1.630992  ,  0.35412383, -1.7537163 , ..., -0.48558499,
         1.04524704,  0.5198727 ],
       [-1.10626   , -0.89227966,  0.81092561, ..., -1.29354183,
        -0.76693327, -0.76588453],
       ...,
       [ 1.33287545,  1.50655789, -0.49471027, ...,  1.04055573,
         1.54475828,  1.55459752],
       [-1.10626   , -0.7664174 ,  1.04407488, ...,  1.04055573,
        -0.76693327, -0.76588453],
       [-0.42872237,  0.10192865,  0.81092561, ..., -1.29354183,
        -0.76693327, -0.76588453]])
```

In [252…
```
from sklearn.linear_model import LinearRegression #load liner rigression model and trai
lin_reg = LinearRegression()
lin_reg.fit(X_scaled, y)
```

Out[252…
```
LinearRegression()
```

In [253…
```
from sklearn.metrics import mean_squared_error #Calculating mean sqare error for train
y_predicted = lin_reg.predict(X_scaled)
lin_mse = mean_squared_error(y, y_predicted)

print("MSE train error value for Office is :")
print(lin_mse)
```

```
MSE train error value for Office is :
548663025.5788134
```

In [254…
```
y_test = test_set.pop('HVAC Electric Demand') #separating target and data for test
X_test = test_set
```

In [255…

```
X_test_scale = scaler.transform(X_test) #calculationg mean sqare error for test data
y_predicted_test = lin_reg.predict(X_test_scale)
lin_mse_test = mean_squared_error(y_test, y_predicted_test)

print("MSE test error value for Office is :")
print(lin_mse_test)
```

```
MSE test error value for Office is :
516290613.1803748
```

For Apartment

In [256…

```
apartment1 = apartment.drop(columns = ['month','day','hour','Total Electric Demand']) #
apartment1
```

Out[256…

	Drybulb Temperature	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation	Sky Clearness	HVAC Electric Demand
0	16.00	13.71	78	0.50	190	0	0.00	2564.34
1	15.60	13.76	82	2.10	120	0	0.00	1954.42
2	15.10	13.60	85	2.10	120	0	0.00	1600.69
3	14.80	13.51	87	2.10	140	0	0.00	1385.73
4	14.40	13.23	88	1.00	150	0	0.00	1245.54
...	...	...	...	...	...	...	...	...
8755	17.90	14.18	67	3.60	290	0	0.00	5645.49
8756	17.50	14.06	69	3.10	270	0	0.00	5061.28
8757	17.20	14.03	71	2.60	260	0	0.00	4435.61
8758	16.80	13.89	73	3.10	260	0	0.00	3394.96
8759	16.50	13.84	75	3.60	270	0	0.00	2505.03

8760 rows × 8 columns

Data split : Test size = 20 %

In [257…

```
from sklearn.model_selection import train_test_split #Split data
train_set, test_set = train_test_split(apartment1, test_size = 0.2, random_state = 42)
```

In [258…

```
train_set = train_set.reset_index(drop = True) #reset index
test_set = test_set.reset_index(drop = True) #reset index
```

In [259…

```
data = train_set.copy() #create train set as data
```

In [260…

```
corr_mat = data.corr() #corrilation matrix with respect to HVAC electric demand
corr_mat["HVAC Electric Demand"].sort_values(ascending=False)
```

Out[260…

```
HVAC Electric Demand    1.00
Drybulb Temperature     0.93
Wetbulb Temperature     0.85
Solar Radiation         0.33
Sky Clearness           0.30
Wind Speed              0.30
Wind Direction          0.23
Relative Humidity      -0.49
Name: HVAC Electric Demand, dtype: float64
```

In [261…

```
y = data.pop('HVAC Electric Demand') #separating target and data for train
X = data
```

In [262…

```
from sklearn.preprocessing import StandardScaler #standardizing the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled
```

Out[262…

```
array([[-0.0221998 , -1.03900228, -1.42730733, ..., -0.12649305,
         1.30081093,  1.08809622],
       [ 1.630992  ,  0.35412383, -1.7537163 , ..., -0.48558499,
         1.04524704,  0.5198727 ],
       [-1.10626   , -0.89227966,  0.81092561, ..., -1.29354183,
        -0.76693327, -0.76588453],
       ...,
       [ 1.33287545,  1.50655789, -0.49471027, ...,  1.04055573,
         1.54475828,  1.55459752],
       [-1.10626   , -0.7664174 ,  1.04407488, ...,  1.04055573,
        -0.76693327, -0.76588453],
       [-0.42872237,  0.10192865,  0.81092561, ..., -1.29354183,
        -0.76693327, -0.76588453]])
```

In [263…

```
from sklearn.linear_model import LinearRegression #load liner rigression model and trai
lin_reg = LinearRegression()
lin_reg.fit(X_scaled, y)
```

Out[263…

```
LinearRegression()
```

In [264…

```
from sklearn.metrics import mean_squared_error #Calculating mean sqare error for train
y_predicted = lin_reg.predict(X_scaled)
lin_mse = mean_squared_error(y, y_predicted)

print("MSE train error value for Apartment is :")
print(lin_mse)
```

```
MSE train error value for Apartment is :
9776844.844944552
```

In [265…

```
y_test = test_set.pop('HVAC Electric Demand') #separating target and data for test
X_test = test_set
```

In [266…

```
X_test_scale = scaler.transform(X_test) #calculationg mean sqare error for test data
y_predicted_test = lin_reg.predict(X_test_scale)
lin_mse_test = mean_squared_error(y_test, y_predicted_test)
```

```
print("MSE test error value for Apartment is :")
print(lin_mse_test)
```

```
MSE test error value for Apartment is :
9744123.440380758
```

Question 7 : Incorporate the role of season and time of day into your regression model by introducing two sets of categorical variables:

1. First, explain how to add categorical variables into a regression model through OneHotEncoder in sklearn and what OneHotEncoder is (we did not cover this in our lecture and this is defined as an assignment for you.)
2. Second, use OneHotEncoder object and transform 'month' column and concatenate it to your weather conditions input.
3. Third, use pandas map method and convert the 'hour' column values as follows:
   a. {0,1,2,3,4,5}-->value=0,
   b. {6,7,8,9}-->value=1,
   c. {10,11,12}-->value=2,
   d. {13,14,15,16}-->value=3,
   e. {17,18,19}-->value=4,
   f. {20,21,22,23}-->value=5.
4. Fourth, apply OneHotEncoder on this new column and concatenate it to your input.

8- Repeat question 6 with the new dataset for both buildings and report any improvement you see in training and test MSE values.

Answer 7

Categorical data are variables that contain label values rather than numeric values. Many machine learning algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric. This means that categorical data must be converted to a numerical form. If the categorical variable is an output variable, you may also want to convert predictions by the model back into a categorical form in order to present them or use them in some application.

A one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

For Office

In [267...

```
office
```

Out[267...

	month	day	hour	Drybulb Temperature	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation	Sky Clearness
0	1	1	1	16.00	13.71	78	0.50	190	0	0.00
1	1	1	2	15.60	13.76	82	2.10	120	0	0.00
2	1	1	3	15.10	13.60	85	2.10	120	0	0.00
3	1	1	4	14.80	13.51	87	2.10	140	0	0.00
4	1	1	5	14.40	13.23	88	1.00	150	0	0.00
...	...	...	...	...	...	...	...	...	...	...
8755	12	31	20	17.90	14.18	67	3.60	290	0	0.00
8756	12	31	21	17.50	14.06	69	3.10	270	0	0.00
8757	12	31	22	17.20	14.03	71	2.60	260	0	0.00
8758	12	31	23	16.80	13.89	73	3.10	260	0	0.00
8759	12	31	0	16.50	13.84	75	3.60	270	0	0.00

8760 rows × 12 columns

In [268...

```
month_column = pd.get_dummies(office['month'])
month_column = month_column.drop(5, axis = 1)
month_column
```

Out[268...

	1	2	3	4	6	7	8	9	10	11	12
0	1	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
8755	0	0	0	0	0	0	0	0	0	0	1
8756	0	0	0	0	0	0	0	0	0	0	1
8757	0	0	0	0	0	0	0	0	0	0	1

	1	2	3	4	6	7	8	9	10	11	12
8758	0	0	0	0	0	0	0	0	0	0	1
8759	0	0	0	0	0	0	0	0	0	0	1

8760 rows × 11 columns

In [269…

```
month_column.columns = ["Jan", "Feb", "March", "Apr", "June", "July", "Aug", "Sep", "Oc
```

In [270…

```
target = office1.pop('HVAC Electric Demand')
target
```

Out[270…

```
0       8.44
1       8.44
2       8.44
3       8.44
4       8.44
        ...
8755    8.44
8756    8.44
8757    8.44
8758    8.44
8759    8.44
Name: HVAC Electric Demand, Length: 8760, dtype: float64
```

In [271…

```
Office_weather_data
```

Out[271…

	Drybulb Temperature	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation	Sky Clearness
0	16.00	13.71	78	0.50	190	0	0.00
1	15.60	13.76	82	2.10	120	0	0.00
2	15.10	13.60	85	2.10	120	0	0.00
3	14.80	13.51	87	2.10	140	0	0.00
4	14.40	13.23	88	1.00	150	0	0.00
...	...	...	...	...	...	...	...
8755	17.90	14.18	67	3.60	290	0	0.00
8756	17.50	14.06	69	3.10	270	0	0.00
8757	17.20	14.03	71	2.60	260	0	0.00
8758	16.80	13.89	73	3.10	260	0	0.00
8759	16.50	13.84	75	3.60	270	0	0.00

8760 rows × 7 columns

In [272…

```
office_wm = pd.concat([month_column, Office_weather_data], axis=1)
office_wm
```

Out[272...

	Jan	Feb	March	Apr	June	July	Aug	Sep	Oct	Nov	Dec	Drybulb Temperature	Wetbulb Temperature	Rela Humi
0	1	0	0	0	0	0	0	0	0	0	0	16.00	13.71	
1	1	0	0	0	0	0	0	0	0	0	0	15.60	13.76	
2	1	0	0	0	0	0	0	0	0	0	0	15.10	13.60	
3	1	0	0	0	0	0	0	0	0	0	0	14.80	13.51	
4	1	0	0	0	0	0	0	0	0	0	0	14.40	13.23	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8755	0	0	0	0	0	0	0	0	0	0	1	17.90	14.18	
8756	0	0	0	0	0	0	0	0	0	0	1	17.50	14.06	
8757	0	0	0	0	0	0	0	0	0	0	1	17.20	14.03	
8758	0	0	0	0	0	0	0	0	0	0	1	16.80	13.89	
8759	0	0	0	0	0	0	0	0	0	0	1	16.50	13.84	

8760 rows × 18 columns

In [273...

```
mapping = {0: 0, 1:0, 2:0, 3:0, 4:0, 5:0, 6:1,7:1,8:1,9:1,10:2,11:2,12:2,13:3,14:3,15:3
```

In [274...

```
office['hour'] = office['hour'].map(mapping)
```

In [275...

```
office['hour']
```

Out[275...

```
0       0
1       0
2       0
3       0
4       0
       ..
8755    5
8756    5
8757    5
8758    5
8759    0
Name: hour, Length: 8760, dtype: int64
```

In [276...

```
hour_column = pd.get_dummies(office['hour'])
hour_column = hour_column.drop(0, axis = 1)
hour_column
```

Out[276...

	1	2	3	4	5
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0

	1	2	3	4	5
3	0	0	0	0	0
4	0	0	0	0	0
...	...	...	...	...	...
8755	0	0	0	0	1
8756	0	0	0	0	1
8757	0	0	0	0	1
8758	0	0	0	0	1
8759	0	0	0	0	0

8760 rows × 5 columns

In [277…

```
office_wmh = pd.concat([hour_column, office_wm], axis=1)
office_wmh
```

Out[277…

	1	2	3	4	5	Jan	Feb	March	Apr	June	...	Oct	Nov	Dec	Drybulb Temperature	Wetbulb Temperature	F Hu
0	0	0	0	0	0	1	0	0	0	0	...	0	0	0	16.00	13.71	
1	0	0	0	0	0	1	0	0	0	0	...	0	0	0	15.60	13.76	
2	0	0	0	0	0	1	0	0	0	0	...	0	0	0	15.10	13.60	
3	0	0	0	0	0	1	0	0	0	0	...	0	0	0	14.80	13.51	
4	0	0	0	0	0	1	0	0	0	0	...	0	0	0	14.40	13.23	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8755	0	0	0	0	1	0	0	0	0	0	...	0	0	1	17.90	14.18	
8756	0	0	0	0	1	0	0	0	0	0	...	0	0	1	17.50	14.06	
8757	0	0	0	0	1	0	0	0	0	0	...	0	0	1	17.20	14.03	
8758	0	0	0	0	1	0	0	0	0	0	...	0	0	1	16.80	13.89	
8759	0	0	0	0	0	0	0	0	0	0	...	0	0	1	16.50	13.84	

8760 rows × 23 columns

In [278…

```
X_train_ofc, X_test_ofc, y_train_ofc, y_test_ofc = train_test_split(office_wmh, target,
```

In [279…

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled_ofc = scaler.fit_transform(X_train_ofc)
X_scaled_ofc
```

Out[279…

```
array([[-0.44468413, -0.3821444 ,  2.22692247, ..., -0.12649305,
```

```
         1.30081093,  1.08809622],
       [-0.44468413,  2.61681187, -0.44905021, ..., -0.48558499,
         1.04524704,  0.5198727 ],
       [-0.44468413, -0.3821444 , -0.44905021, ..., -1.29354183,
        -0.76693327, -0.76588453],
       ...,
       [-0.44468413, -0.3821444 ,  2.22692247, ...,  1.04055573,
         1.54475828,  1.55459752],
       [-0.44468413, -0.3821444 , -0.44905021, ...,  1.04055573,
        -0.76693327, -0.76588453],
       [-0.44468413, -0.3821444 , -0.44905021, ..., -1.29354183,
        -0.76693327, -0.76588453]])
```

In [280…

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X_scaled_ofc, y_train_ofc)
```

Out[280…

```
LinearRegression()
```

In [281…

```
from sklearn.metrics import mean_squared_error
y_predicted = lin_reg.predict(X_scaled_ofc)
lin_mse = mean_squared_error(y_train_ofc, y_predicted)
lin_mse
```

Out[281…

```
451892943.2194083
```

In [282…

```
X_test_scale = scaler.transform(X_test_ofc)
y_predicted_test = lin_reg.predict(X_test_scale)
lin_mse_test = mean_squared_error(y_test_ofc, y_predicted_test)
lin_mse_test
```

Out[282…

```
426749307.2308087
```

MSE value has been decrease from 548663025.5788134 to 451892943.2194083 for train and 516290613.1803748 to 426749307.2308087 for apartment.

In [ ]:

In [ ]:

In [ ]:

For Apartment

In [283…

```
apartment
```

Out[283…

	month	day	hour	Drybulb Temperature	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation	Sky Clearness
0	1	1	1	16.00	13.71	78	0.50	190	0	0.00
1	1	1	2	15.60	13.76	82	2.10	120	0	0.00
2	1	1	3	15.10	13.60	85	2.10	120	0	0.00
3	1	1	4	14.80	13.51	87	2.10	140	0	0.00
4	1	1	5	14.40	13.23	88	1.00	150	0	0.00
...	...	...	...	...	...	...	...	...	...	...
8755	12	31	20	17.90	14.18	67	3.60	290	0	0.00
8756	12	31	21	17.50	14.06	69	3.10	270	0	0.00
8757	12	31	22	17.20	14.03	71	2.60	260	0	0.00
8758	12	31	23	16.80	13.89	73	3.10	260	0	0.00
8759	12	31	0	16.50	13.84	75	3.60	270	0	0.00

8760 rows × 12 columns

In [284…

```
month_column = pd.get_dummies(apartment['month'])
month_column = month_column.drop(5, axis = 1)
month_column
```

Out[284…

	1	2	3	4	6	7	8	9	10	11	12
0	1	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
8755	0	0	0	0	0	0	0	0	0	0	1
8756	0	0	0	0	0	0	0	0	0	0	1
8757	0	0	0	0	0	0	0	0	0	0	1
8758	0	0	0	0	0	0	0	0	0	0	1
8759	0	0	0	0	0	0	0	0	0	0	1

8760 rows × 11 columns

In [285…

```
month_column.columns = ["Jan", "Feb", "March", "Apr", "June", "July", "Aug", "Sep", "Oc
```

In [286…

```
target = apartment1.pop('HVAC Electric Demand')
```

In [287…

```
target
```

Out[287…

```
0       2564.34
1       1954.42
2       1600.69
3       1385.73
4       1245.54
         ...
8755    5645.49
8756    5061.28
8757    4435.61
8758    3394.96
8759    2505.03
Name: HVAC Electric Demand, Length: 8760, dtype: float64
```

In [288…

```
apartment_weather_data
```

Out[288…

	Drybulb Temperature	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation	Sky Clearness
0	16.00	13.71	78	0.50	190	0	0.00
1	15.60	13.76	82	2.10	120	0	0.00
2	15.10	13.60	85	2.10	120	0	0.00
3	14.80	13.51	87	2.10	140	0	0.00
4	14.40	13.23	88	1.00	150	0	0.00
...	...	...	...	...	...	...	...
8755	17.90	14.18	67	3.60	290	0	0.00
8756	17.50	14.06	69	3.10	270	0	0.00
8757	17.20	14.03	71	2.60	260	0	0.00
8758	16.80	13.89	73	3.10	260	0	0.00
8759	16.50	13.84	75	3.60	270	0	0.00

8760 rows × 7 columns

In [289…

```
apartment_wm = pd.concat([month_column, apartment_weather_data], axis=1)
apartment_wm
```

Out[289…

	Jan	Feb	March	Apr	June	July	Aug	Sep	Oct	Nov	Dec	Drybulb Temperature	Wetbulb Temperature	Rela Humi
0	1	0	0	0	0	0	0	0	0	0	0	16.00	13.71	
1	1	0	0	0	0	0	0	0	0	0	0	15.60	13.76	
2	1	0	0	0	0	0	0	0	0	0	0	15.10	13.60	
3	1	0	0	0	0	0	0	0	0	0	0	14.80	13.51	

	Jan	Feb	March	Apr	June	July	Aug	Sep	Oct	Nov	Dec	Drybulb Temperature	Wetbulb Temperature	Rela Humi
4	1	0	0	0	0	0	0	0	0	0	0	14.40	13.23	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8755	0	0	0	0	0	0	0	0	0	0	1	17.90	14.18	
8756	0	0	0	0	0	0	0	0	0	0	1	17.50	14.06	
8757	0	0	0	0	0	0	0	0	0	0	1	17.20	14.03	
8758	0	0	0	0	0	0	0	0	0	0	1	16.80	13.89	
8759	0	0	0	0	0	0	0	0	0	0	1	16.50	13.84	

8760 rows × 18 columns

In [290…

```
mapping = {0: 0, 1:0, 2:0, 3:0, 4:0, 5:0, 6:1,7:1,8:1,9:1,10:2,11:2,12:2,13:3,14:3,15:3
```

In [291…

```
apartment['hour'] = apartment['hour'].map(mapping)
```

In [292…

```
apartment['hour']
```

Out[292…

```
0       0
1       0
2       0
3       0
4       0
       ..
8755    5
8756    5
8757    5
8758    5
8759    0
Name: hour, Length: 8760, dtype: int64
```

In [293…

```
hour_column = pd.get_dummies(apartment['hour'])
hour_column = hour_column.drop(0, axis = 1)
hour_column
```

Out[293…

	1	2	3	4	5
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
...	...	...	...	...	...
8755	0	0	0	0	1

	1	2	3	4	5
8756	0	0	0	0	1
8757	0	0	0	0	1
8758	0	0	0	0	1
8759	0	0	0	0	0

8760 rows × 5 columns

In [294…

```
apartment_wmh = pd.concat([hour_column, apartment_wm], axis=1)
apartment_wmh
```

Out[294…

	1	2	3	4	5	Jan	Feb	March	Apr	June	...	Oct	Nov	Dec	Drybulb Temperature	Wetbulb Temperature	F Hu
0	0	0	0	0	0	1	0	0	0	0	...	0	0	0	16.00	13.71	
1	0	0	0	0	0	1	0	0	0	0	...	0	0	0	15.60	13.76	
2	0	0	0	0	0	1	0	0	0	0	...	0	0	0	15.10	13.60	
3	0	0	0	0	0	1	0	0	0	0	...	0	0	0	14.80	13.51	
4	0	0	0	0	0	1	0	0	0	0	...	0	0	0	14.40	13.23	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8755	0	0	0	0	1	0	0	0	0	0	...	0	0	1	17.90	14.18	
8756	0	0	0	0	1	0	0	0	0	0	...	0	0	1	17.50	14.06	
8757	0	0	0	0	1	0	0	0	0	0	...	0	0	1	17.20	14.03	
8758	0	0	0	0	1	0	0	0	0	0	...	0	0	1	16.80	13.89	
8759	0	0	0	0	0	0	0	0	0	0	...	0	0	1	16.50	13.84	

8760 rows × 23 columns

In [295…

```
X_train_apt, X_test_apt, y_train_apt, y_test_apt = train_test_split(apartment_wmh, targ
```

In [296…

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled_apt = scaler.fit_transform(X_train_apt)
X_scaled_apt
```

Out[296…

```
array([[-0.44468413, -0.3821444 ,  2.22692247, ..., -0.12649305,
         1.30081093,  1.08809622],
       [-0.44468413,  2.61681187, -0.44905021, ..., -0.48558499,
         1.04524704,  0.5198727 ],
       [-0.44468413, -0.3821444 , -0.44905021, ..., -1.29354183,
        -0.76693327, -0.76588453],
       ...,
       [-0.44468413, -0.3821444 ,  2.22692247, ...,  1.04055573,
```

```
       1.54475828,  1.55459752],
      [-0.44468413, -0.3821444 , -0.44905021, ...,  1.04055573,
       -0.76693327, -0.76588453],
      [-0.44468413, -0.3821444 , -0.44905021, ..., -1.29354183,
       -0.76693327, -0.76588453]])
```

In [297…

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X_scaled_apt, y_train_apt)
```

Out[297…

```
LinearRegression()
```

In [298…

```
from sklearn.metrics import mean_squared_error
y_predicted = lin_reg.predict(X_scaled_apt)
lin_mse = mean_squared_error(y_train_apt, y_predicted)
lin_mse
```

Out[298…

```
4208786.121867774
```

In [299…

```
X_test_scale = scaler.transform(X_test_apt)
y_predicted_test = lin_reg.predict(X_test_scale)
lin_mse_test = mean_squared_error(y_test_apt, y_predicted_test)
lin_mse_test
```

Out[299…

```
4050179.4859527615
```

MSE value has been decrease from 9776844.844944552 to 4208786.121867774 for train and 9744123.440380758 to 4050179.4859527615 for apartment.

In [ ]:

In [ ]:

In [ ]:

Question 9 : Explain what regularization is in supervised learning and repeat step 8 using sklearn Ridge and Lasso classes

1. Use Ridge and report and plot test MSE for alpha={0, 0.005, 0.05,0.1,1}

In [300…

```
from sklearn.linear_model import Ridge
Ridge_model = Ridge()
```

For Office

In [301…

```
X_scaled_ofc = scaler.fit_transform(X_train_ofc)
```

In [302…

```
Ridge_model.fit(X_scaled_ofc, y_train_ofc)
```

Out[302…

```
Ridge()
```

In [303…

```
y_predict_ofc_train = Ridge_model.predict(X_scaled_ofc)
Ridge_MSE_ofc = mean_squared_error(y_train_ofc, y_predict_ofc_train)
print("Mean Sqaured Error for Train set is :")
print(Ridge_MSE_ofc)
```

```
Mean Sqaured Error for Train set is :
451898103.27265155
```

In [304…

```
X_scaled_ofc_test = scaler.fit_transform(X_test_ofc)
y_predict_ofc_test = Ridge_model.predict(X_scaled_ofc_test)
Ridge_MSE_ofc_test = mean_squared_error(y_test_ofc, y_predict_ofc_test)
print("Mean Sqaured Error for Test set is :")
print(Ridge_MSE_ofc_test)
```

```
Mean Sqaured Error for Test set is :
427639517.23101014
```

In [305…

```
MSE_ofc_array = []
alpha_office = [0,0.005,0.05,0.1,1]

for alfa in alpha_office:
    Ridge_model_continuous = Ridge(alpha=alfa)
    X_scaled_ofc = scaler.fit_transform(X_train_ofc)
    Ridge_model_continuous.fit(X_scaled_ofc, y_train_ofc)

    X_scaled_ofc_test = scaler.fit_transform(X_test_ofc)
    y_predict_ofc_test = Ridge_model_continuous.predict(X_scaled_ofc_test)
    MSE_test = mean_squared_error(y_test_ofc, y_predict_ofc_test)
    MSE_ofc_array.append(MSE_test)


MSE_ofc_array
```

Out[305…

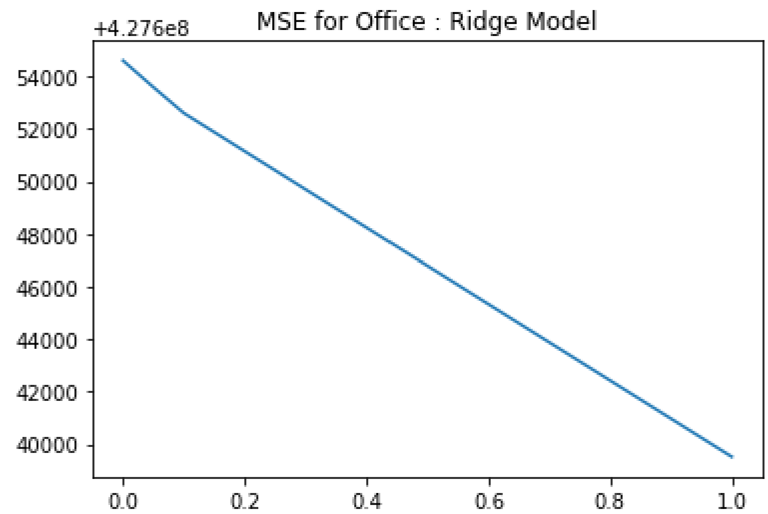
```
[427654606.38101596,
 427654503.56020284,
 427653590.8214128,
 427652603.3091867,
 427639517.23101014]
```

In [306…

```
plt.plot(alpha_office, MSE_ofc_array)
plt.title("MSE for Office : Ridge Model")
```

Out[306…

```
Text(0.5, 1.0, 'MSE for Office : Ridge Model')
```



MSE values has been decreased with respect to alpha. It means model is performing better when alpha increases gradually from zero to one.

In [ ]:

In [ ]:

In [ ]:

For Apartment

In [307…
```
X_scaled_apt = scaler.fit_transform(X_train_apt)
```

In [308…
```
Ridge_model.fit(X_scaled_apt, y_train_apt)
```

Out[308… Ridge()

In [309…
```
y_predict_apt_train = Ridge_model.predict(X_scaled_apt)
Ridge_MSE_apt = mean_squared_error(y_train_apt, y_predict_apt_train)
print("Mean Sqaured Error for Train set is :")
print(Ridge_MSE_apt)
```

```
Mean Sqaured Error for Train set is :
4208917.40206579
```

In [310…
```
X_scaled_apt_test = scaler.fit_transform(X_test_apt)
y_predict_apt_test = Ridge_model.predict(X_scaled_apt_test)
Ridge_MSE_apt_test = mean_squared_error(y_test_apt, y_predict_apt_test)
print("Mean Sqaured Error for Test set is :")
print(Ridge_MSE_apt_test)
```

```
Mean Sqaured Error for Test set is :
```

```
4104399.445155013
```

In [311…

```
MSE_apt_array = []
alpha_apartments = [0,0.005,0.05,0.1,1]

for alfa in alpha_apartments:
    Ridge_model_continuous = Ridge(alpha=alfa)
    X_scaled_apt = scaler.fit_transform(X_train_apt)
    Ridge_model_continuous.fit(X_scaled_apt, y_train_apt)

    X_scaled_apt_test = scaler.fit_transform(X_test_apt)
    y_predict_apt_test = Ridge_model_continuous.predict(X_scaled_apt_test)
    MSE_test = mean_squared_error(y_test_apt, y_predict_apt_test)
    MSE_apt_array.append(MSE_test)


MSE_apt_array
```

Out[311…

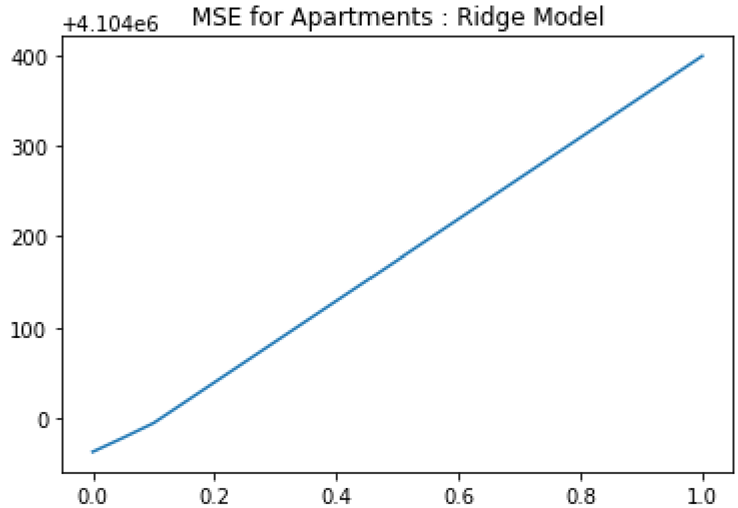
```
[4103962.444624918,
 4103963.96928798,
 4103977.996618928,
 4103994.2256815922,
 4104399.445155013]
```

In [312…

```
plt.plot(alpha_apartments, MSE_apt_array)
plt.title("MSE for Apartments : Ridge Model")
```

Out[312…

```
Text(0.5, 1.0, 'MSE for Apartments : Ridge Model')
```



MSE values has been increased with respect to alpha. It means model is not performing better when alpha increases gradually from zero to one.

In [ ]:

In [ ]:

In [ ]:

2. Use Lasso and report and plot test MSE for alpha={0, 0.005, 0.05,0.1,1}.

In [313…
```
from sklearn.linear_model import Lasso
Lasso_model = Lasso()
```

For Office

In [314…
```
Lasso_model.fit(X_scaled_ofc, y_train_ofc)
```

Out[314…
```
Lasso()
```

In [315…
```
y_predict_ofc_train = Lasso_model.predict(X_scaled_ofc)
Lasso_model_MSE_ofc = mean_squared_error(y_train_ofc, y_predict_ofc_train)
print("Mean Sqaured Error for Train set is :")
print(Lasso_model_MSE_ofc)
```

```
Mean Sqaured Error for Train set is :
451893413.33383167
```

In [316…
```
X_scaled_ofc_test = scaler.fit_transform(X_test_ofc)
y_predict_ofc_test = Lasso_model.predict(X_scaled_ofc_test)
Lasso_model_MSE_ofc_test = mean_squared_error(y_test_ofc, y_predict_ofc_test)
print("Mean Sqaured Error for Test set is :")
print(Lasso_model_MSE_ofc_test)
```

```
Mean Sqaured Error for Test set is :
427641239.0283439
```

In [317…
```
Lasso_MSE_ofc_array = []
alpha_office = [0,0.005,0.05,0.1,1]

for alfa in alpha_office:
    Lasso_model_continuous = Lasso(alpha=alfa)
    X_scaled_ofc = scaler.fit_transform(X_train_ofc)
    Lasso_model_continuous.fit(X_scaled_ofc, y_train_ofc)

    X_scaled_ofc_test = scaler.fit_transform(X_test_ofc)
    y_predict_ofc_test = Lasso_model_continuous.predict(X_scaled_ofc_test)
    MSE_test = mean_squared_error(y_test_ofc, y_predict_ofc_test)
    Lasso_MSE_ofc_array.append(MSE_test)


Lasso_MSE_ofc_array
```

```
<ipython-input-317-a14e8f934915>:7: UserWarning: With alpha=0, this algorithm does not c
onverge well. You are advised to use the LinearRegression estimator
  Lasso_model_continuous.fit(X_scaled_ofc, y_train_ofc)
C:\Users\CHINTAN SHAH\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_desce
nt.py:530: UserWarning: Coordinate descent with no regularization may lead to unexpected
results and is discouraged.
  model = cd_fast.enet_coordinate_descent(
C:\Users\CHINTAN SHAH\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_desce
```

```
nt.py:530: ConvergenceWarning: Objective did not converge. You might want to increase th
e number of iterations. Duality gap: 1583432873040.8047, tolerance: 939823802.8505596
  model = cd_fast.enet_coordinate_descent(
```

Out[317…

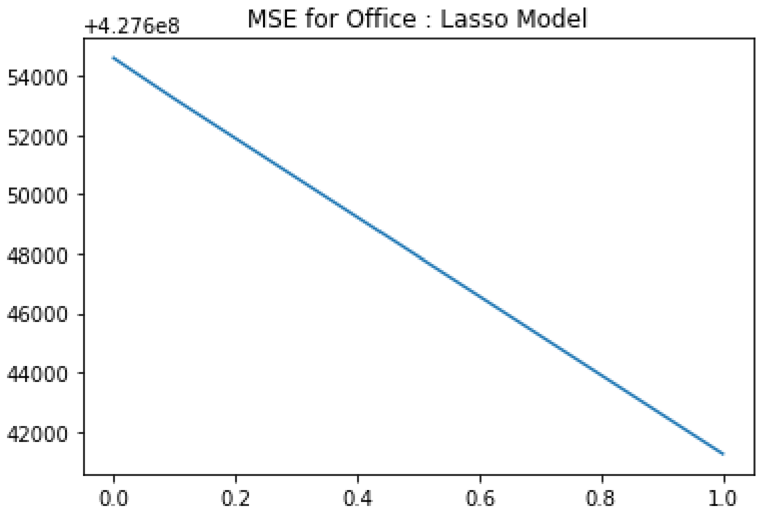
```
[427654606.3810098,
 427654537.1104744,
 427653914.7781717,
 427653225.6120857,
 427641239.0283439]
```

In [318…

```
plt.plot(alpha_office, Lasso_MSE_ofc_array)
plt.title("MSE for Office : Lasso Model")
```

Out[318…

```
Text(0.5, 1.0, 'MSE for Office : Lasso Model')
```



MSE values has been decreased with respect to alpha. It means model is performing better when alpha increases gradually from zero to one.

In [ ]:

In [ ]:

In [ ]:

For Apartment

In [319…

```
Lasso_model.fit(X_scaled_apt, y_train_apt)
```

Out[319…

```
Lasso()
```

In [320…

```
y_predict_apt_train = Lasso_model.predict(X_scaled_apt)
Lasso_model_MSE_apt = mean_squared_error(y_train_apt, y_predict_apt_train)
print("Mean Sqaured Error for Train set is :")
print(Lasso_model_MSE_apt)
```

```
Mean Sqaured Error for Train set is :
4209189.509423733
```

In [321…

```
X_scaled_apt_test = scaler.fit_transform(X_test_apt)
y_predict_apt_test = Lasso_model.predict(X_scaled_apt_test)
Lasso_model_MSE_apt_test = mean_squared_error(y_test_apt, y_predict_apt_test)
print("Mean Sqaured Error for Test set is :")
print(Lasso_model_MSE_apt_test)
```

```
Mean Sqaured Error for Test set is :
4103525.561304995
```

In [322…

```
Lasso_MSE_apt_array = []
alpha_apartments = [0,0.005,0.05,0.1,1]

for alfa in alpha_apartments:
    Lasso_model_continuous = Lasso(alpha=alfa)
    X_scaled_apt = scaler.fit_transform(X_train_apt)
    Lasso_model_continuous.fit(X_scaled_apt, y_train_apt)

    X_scaled_apt_test = scaler.fit_transform(X_test_apt)
    y_predict_apt_test = Lasso_model_continuous.predict(X_scaled_apt_test)
    MSE_test = mean_squared_error(y_test_apt, y_predict_apt_test)
    Lasso_MSE_apt_array.append(MSE_test)


Lasso_MSE_apt_array
```

```
<ipython-input-322-1d294d530f5c>:7: UserWarning: With alpha=0, this algorithm does not c
onverge well. You are advised to use the LinearRegression estimator
  Lasso_model_continuous.fit(X_scaled_apt, y_train_apt)
C:\Users\CHINTAN SHAH\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_desce
nt.py:530: UserWarning: Coordinate descent with no regularization may lead to unexpected
results and is discouraged.
  model = cd_fast.enet_coordinate_descent(
C:\Users\CHINTAN SHAH\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_desce
nt.py:530: ConvergenceWarning: Objective did not converge. You might want to increase th
e number of iterations. Duality gap: 14747586571.024668, tolerance: 87035546.24758647
  model = cd_fast.enet_coordinate_descent(
```

Out[322…

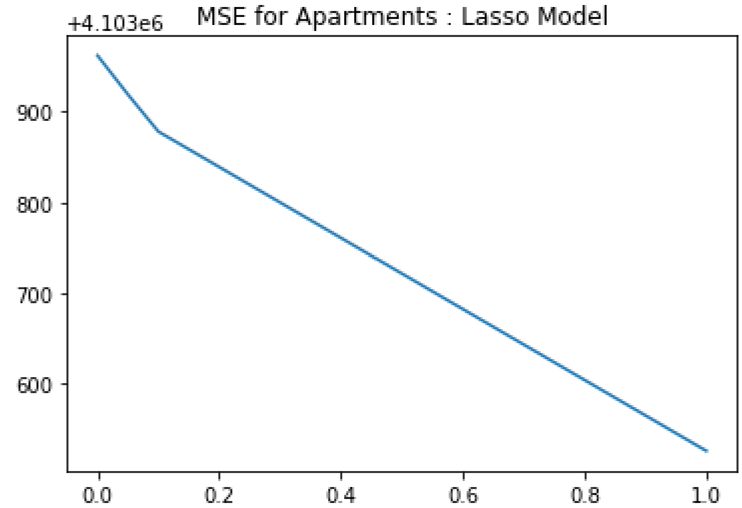
```
[4103962.444624951,
 4103958.032190498,
 4103919.283560474,
 4103878.280615343,
 4103525.561304995]
```

In [323…

```
plt.plot(alpha_apartments, Lasso_MSE_apt_array)
plt.title("MSE for Apartments : Lasso Model")
```

Out[323…

```
Text(0.5, 1.0, 'MSE for Apartments : Lasso Model')
```



MSE values has been decreased with respect to alpha. It means model is performing better when alpha increases gradually from zero to one.

In [ ]:

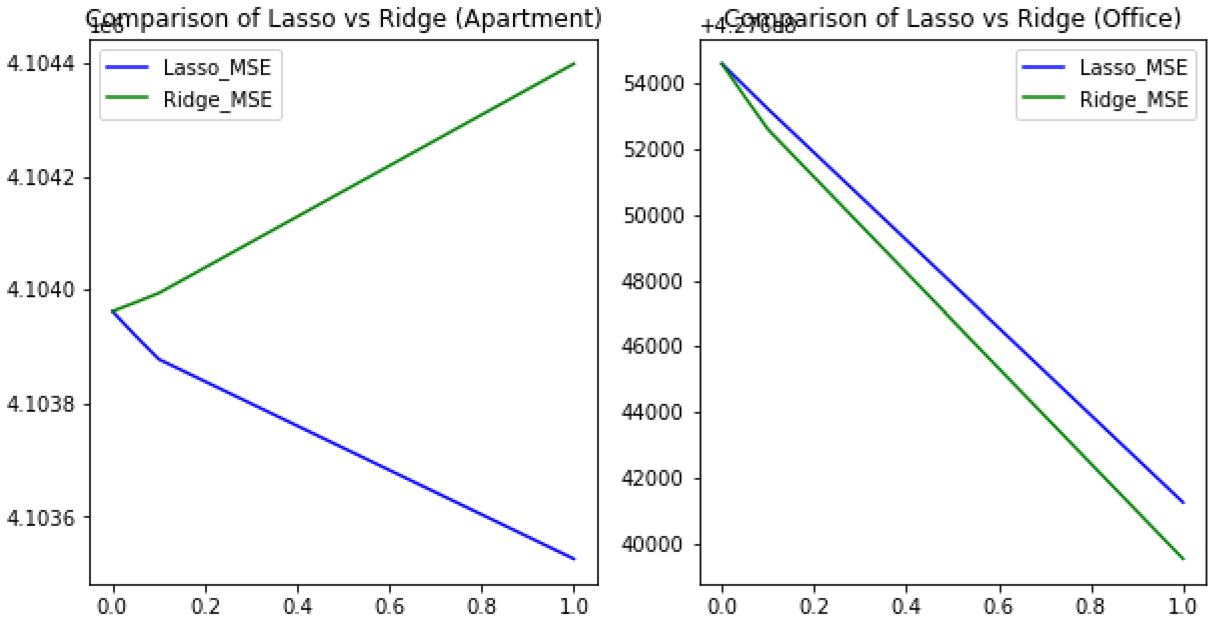
In [ ]:

In [ ]:

Comparison of Ridge and Lasso

In [324…

```
alpha_ = [0,0.005,0.05,0.1,1]

figure = plt.figure(figsize= (10,5))

figure.add_subplot(1,2,1)
plt.plot(alpha_, Lasso_MSE_apt_array, color ='b', label = "Lasso_MSE")
plt.plot(alpha_, MSE_apt_array, color ='g', label ="Ridge_MSE")
plt.title("Comparison of Lasso vs Ridge (Apartment)")
plt.legend();

figure.add_subplot(1,2,2)
plt.plot(alpha_, Lasso_MSE_ofc_array, color ='b', label = "Lasso_MSE")
plt.plot(alpha_, MSE_ofc_array, color ='g', label ="Ridge_MSE")
plt.title("Comparison of Lasso vs Ridge (Office)")
plt.legend();
```



In comparison to our model for ridge vs lasso, we can say that our model is more compitant with lasso regression method.

In [ ]:

In [ ]:

In [ ]:

Question 10 : Use the following sklearn regressors and compare the training and test MSE values and report the model with the best generalization (do not change the default values for these objects):

1. AdaBoostRegressor
2. BaggingRegressor
3. SVR
4. RandomForestRegressor

In [325…

```
from sklearn import tree
from sklearn.svm import SVR
from sklearn.ensemble import BaggingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import AdaBoostRegressor
import seaborn as sns
```

In [326…

```
Regressor_names = ["AdaBoostRegressor", "BaggingRegressor", "SVR", "DecisionTreeRegress
```

For Office

In [327…

```
MSE_train_list = []
MSE_test_list = []
for reg in Regressor_names:
    regressor_model =eval(reg)()

    regressor_model.fit(X_scaled_ofc, y_train_ofc)

    y_predicted_train = regressor_model.predict(X_scaled_ofc)
    y_predicted_test = regressor_model.predict(X_scaled_ofc_test)

    mse_train =mean_squared_error(y_train_ofc, y_predicted_train)
    mse_test =mean_squared_error(y_test_ofc, y_predicted_test)

    MSE_train_list.append(mse_train)
    MSE_test_list.append(mse_test)
```

In [328…

```
Regressors = pd.DataFrame({"Train MSE":MSE_train_list, "Test MSE":MSE_test_list, "Regre
Regressors
```

Out[328…

	Train MSE	Test MSE	Regressor
0	526391919.02	507206514.96	AdaBoostRegressor
1	68026031.90	368747631.34	BaggingRegressor
2	1429222294.12	1417953040.49	SVR
3	145857.81	649441665.08	DecisionTreeRegressor

In [329…

```
regressor = pd.melt(Regressors, id_vars = ['Regressor'] ,value_vars = ["Train MSE", "Te
regressor.columns = ['Regressor Type', "Train/Test", "MSE"]
regressor
```

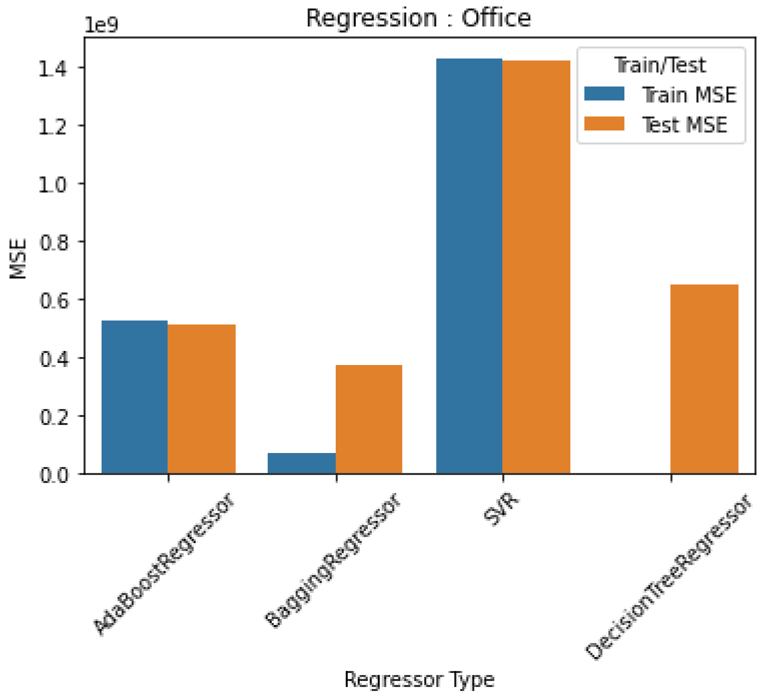
Out[329…

	Regressor Type	Train/Test	MSE
0	AdaBoostRegressor	Train MSE	526391919.02
1	BaggingRegressor	Train MSE	68026031.90
2	SVR	Train MSE	1429222294.12
3	DecisionTreeRegressor	Train MSE	145857.81
4	AdaBoostRegressor	Test MSE	507206514.96
5	BaggingRegressor	Test MSE	368747631.34
6	SVR	Test MSE	1417953040.49
7	DecisionTreeRegressor	Test MSE	649441665.08

In [330…

```
Regressors = pd.DataFrame({"Train MSE":MSE_train_list, "Test MSE":MSE_test_list, "Regre
regressor = pd.melt(Regressors, id_vars = ['Regressor'] ,value_vars = ["Train MSE", "Te
regressor.columns = ['Regressor Type', "Train/Test", "MSE"]
sns.barplot(x= "Regressor Type", y="MSE", hue = "Train/Test", data =regressor)
```

```
plt.xticks(rotation = 45)
plt.title("Regression : Office");
```



For office, aBagging Regressor method performs better than others.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

For Apartment

In [331…

```
MSE_train_list = []
MSE_test_list = []

for reg in regressor_names:
    regressor_model =eval(reg)()

    regressor_model.fit(X_scaled_apt, y_train_apt)

    y_predicted_train = regressor_model.predict(X_scaled_apt)
    y_predicted_test = regressor_model.predict(X_scaled_apt_test)

    mse_train = mean_squared_error(y_train_apt, y_predicted_train)
    mse_test = mean_squared_error(y_test_apt, y_predicted_test)
```

```
    MSE_train_list.append(mse_train)
    MSE_test_list.append(mse_test)
```

In [332…

```
Regressors = pd.DataFrame({"Train MSE":MSE_train_list, "Test MSE":MSE_test_list, "Regre
Regressors
```

Out[332…

	Train MSE	Test MSE	Regressor
0	5510953.95	5611615.92	AdaBoostRegressor
1	285251.71	1701251.91	BaggingRegressor
2	121129347.05	122964393.26	SVR
3	615.13	3171525.48	DecisionTreeRegressor

In [333…

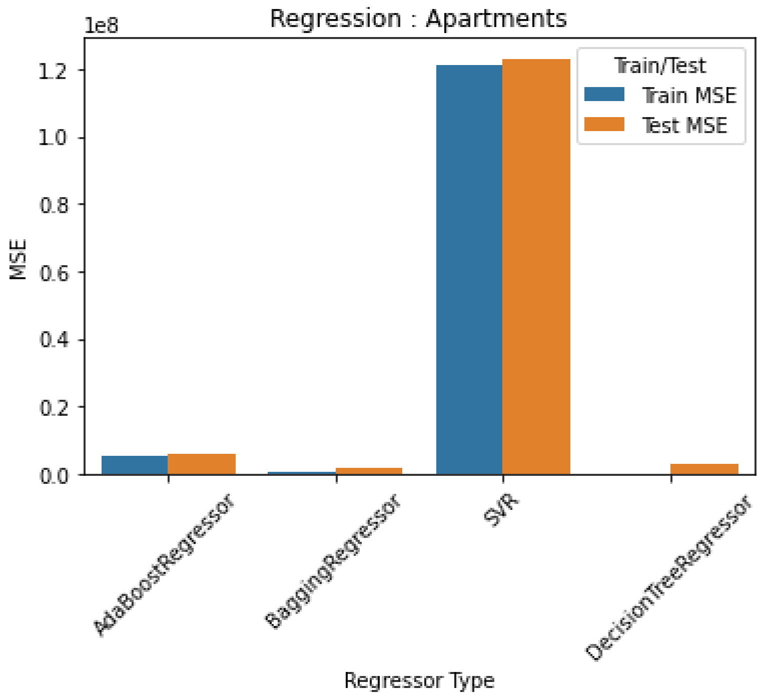
```
regressor = pd.melt(Regressors, id_vars = ['Regressor'] ,value_vars = ["Train MSE", "Te
regressor.columns = ['Regressor Type', "Train/Test", "MSE"]
regressor
```

Out[333…

	Regressor Type	Train/Test	MSE
0	AdaBoostRegressor	Train MSE	5510953.95
1	BaggingRegressor	Train MSE	285251.71
2	SVR	Train MSE	121129347.05
3	DecisionTreeRegressor	Train MSE	615.13
4	AdaBoostRegressor	Test MSE	5611615.92
5	BaggingRegressor	Test MSE	1701251.91
6	SVR	Test MSE	122964393.26
7	DecisionTreeRegressor	Test MSE	3171525.48

In [334…

```
sns.barplot(x= "Regressor Type", y="MSE", hue = "Train/Test", data =regressor)
plt.title("Regression : Apartments")
plt.xticks(rotation = 45);
```



For office, aBagging Regressor method performs better than others.

In [ ]:

In [ ]: