As you can see, the schema of this data contains the date and time variables and weather conditions as the input (X vector) along with total and HVAC electricity demand (in W) as the target variables (labels).

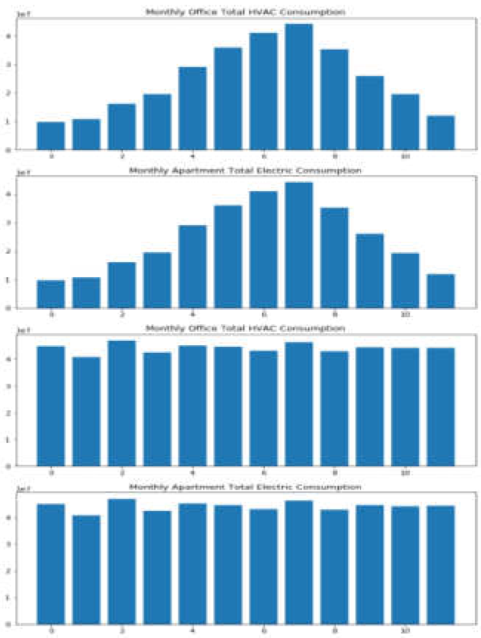
Assume that the electricity demand is the average hourly demand and it can be interpreted as the hourly consumption in Wh.

Answer the following questions (each answer needs full explanation and plotting would not suffice):

1- Compare the total and HVAC electricity consumption of the two buildings on a monthly and annual basis. (a bar chart is preferred)

```
df_group1 = df_office.groupby(['month']).sum() #sum
df_group2 = df_apt.groupby(['month']).sum()

x = np.arange(df_group1.shape[0])

y1 = df_group1.iloc[:,-1] #office HVAC
y2 = df_group1.iloc[:,-1] # apt HVAC
y3 = df_group1.iloc[:,-2] # Office Total elec
y4 = df_group1.iloc[:,-2] # apt total elec


plt.figure(figsize=(10,20)) # monthly

plt.subplot(411)
plt.bar(x,y1)
plt.title('Monthly Office Total HVAC Consumption')

plt.subplot(412)
plt.bar(x,y2)
plt.title('Monthly Apartment Total Electric Consumption')

plt.subplot(413)
plt.bar(x,y3)
plt.title('Monthly Office Total Electric Consumption')

plt.subplot(414)
plt.bar(x,y4)
plt.title('Monthly Apartment Total Electric Consumption')


plt.show()
```

1) Monthly / Total and HVAC consumption in both building

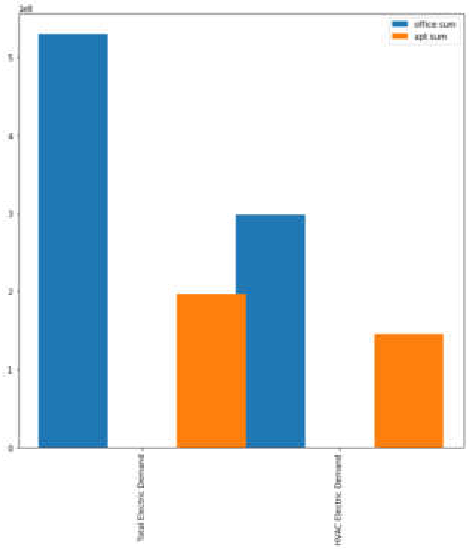


Project 2_ Homework_ Hosung Lee

->

Monthly Sum of HVAC and Total Elec consumption are distributed similarly. Also for Total HVAC and Electrical consumption are similarly. I can assume HVAC is the major consumption for both apartment and office.

```
x = np.arange(len(y_sum_off))
width =0.35
label = y_apt_ann.index

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

plt.bar(x-width,y_sum_off, width, label='office sum')
plt.bar(x+width, y_sum_apt, width, label='apt sum')
plt.legend() #mean,std

plt.xticks(x, label, rotation='vertical')
plt.show()
```



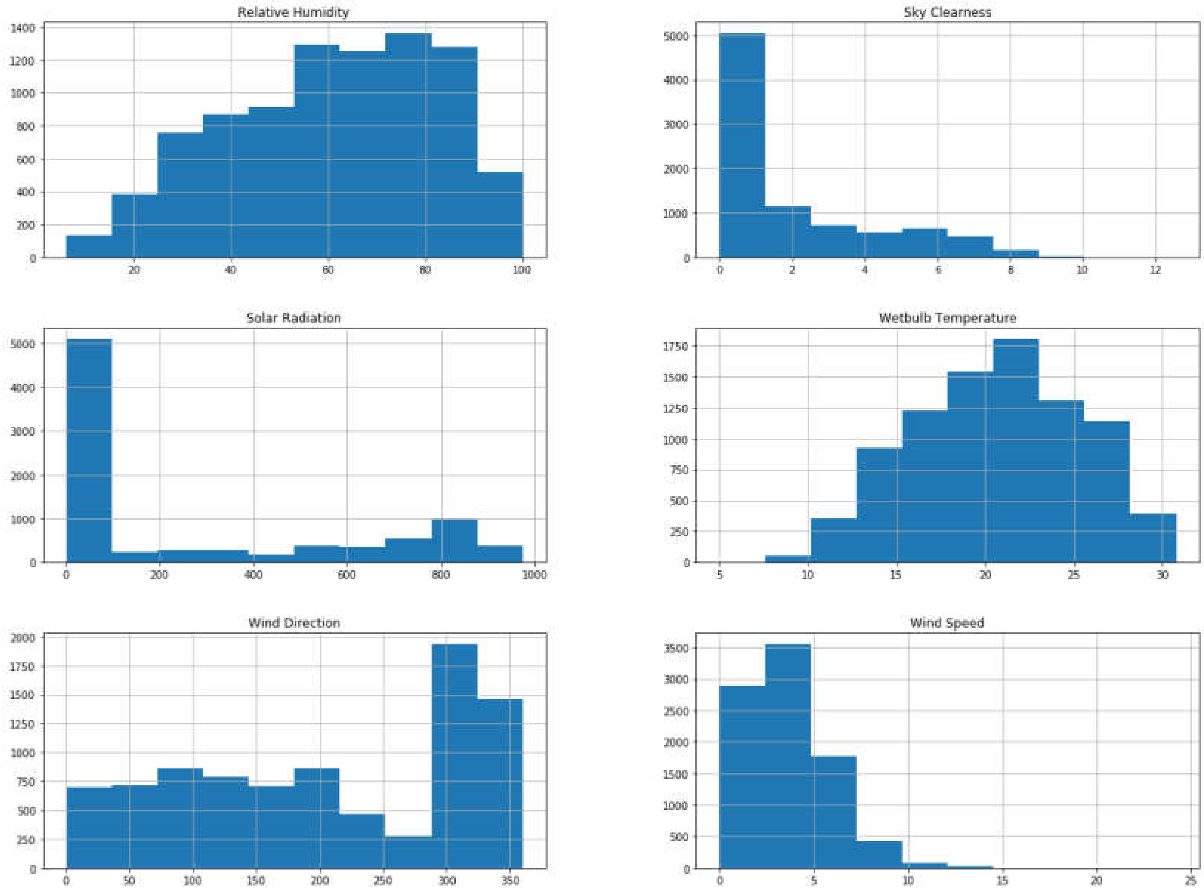
->

Annual consumption for office is highest in Total electricity demand for office.

Apartment's total electricity consumption is lower than those of office's.

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

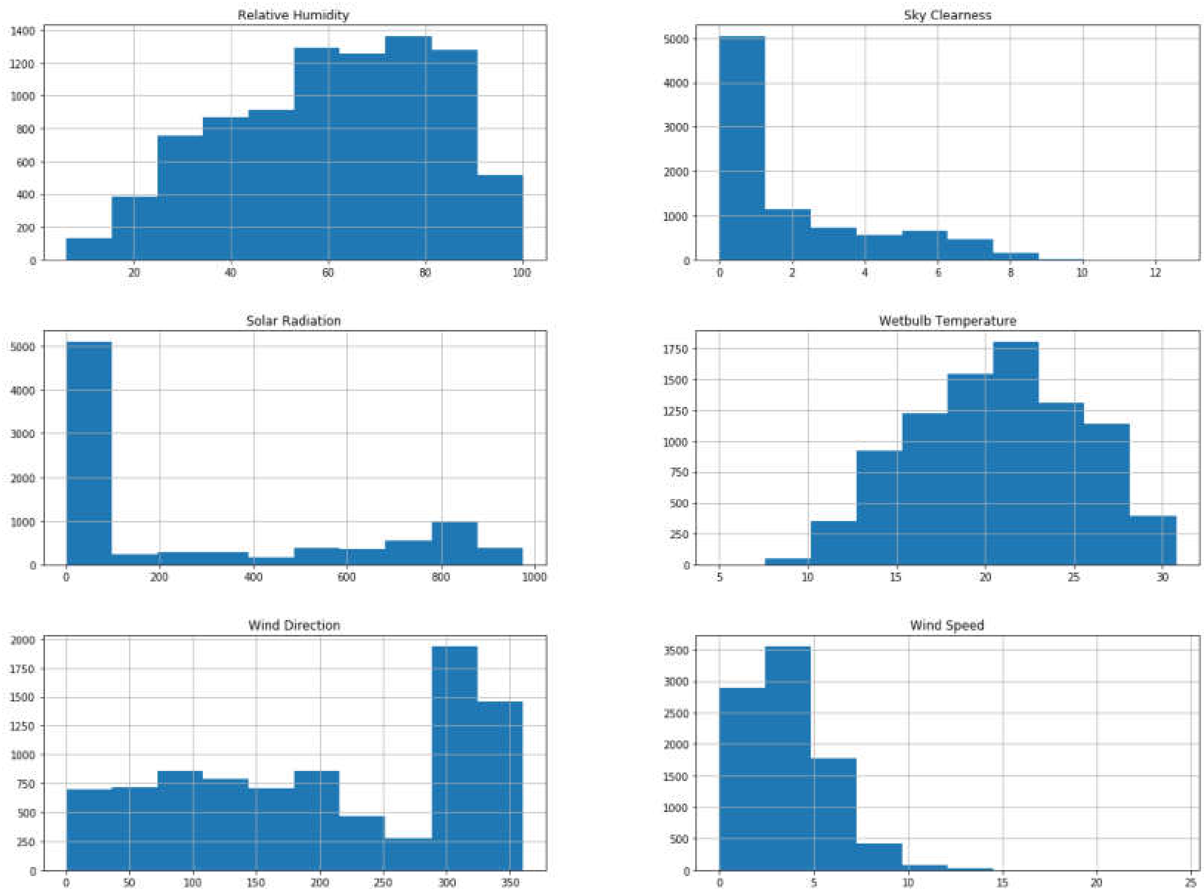
1) Officer Weather Data. Plot



Description: Relative humidity and Wetbulb Temperature are distributed normally. Sky clearness, Solar Radiation and Windspeed shows concentration in earlier year. Wind Direction distributed mostly in 300 to 350 degree.

2) Apartment Weather Data

Project 2_ Homework_ Hosung Lee



Description: (Same as Office) Relative humidity and Wetbulb Temperature are distributed normally. Sky clearness, Solar Radiation and Windspeed shows concentration in earlier year. Wind Direction distributed mostly in 300 to 350 degree.

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

```
9]: df_apt_weather_corr = df_apt.iloc[:,4:]
    df_apt_weather_corr_mat = df_apt_weather_corr.corr()
    df_apt_weather_corr_mat["Total Electric Demand"].sort_values(ascending=False)
```

```
9]: Total Electric Demand     1.000000
    Wind Direction            0.178016
    HVAC Electric Demand      0.152915
    Wind Speed                0.135558
    Wetbulb Temperature       0.070646
    Relative Humidity         0.056831
    Sky Clearness            -0.291045
    Solar Radiation          -0.305266
    Name: Total Electric Demand, dtype: float64
```

```
0]: df_apt_weather_corr_mat["HVAC Electric Demand"].sort_values(ascending=False)
```

```
0]: HVAC Electric Demand      1.000000
    Wetbulb Temperature       0.847445
    Solar Radiation           0.324191
    Wind Speed                0.301006
    Sky Clearness             0.300573
    Wind Direction            0.224513
    Total Electric Demand     0.152915
    Relative Humidity        -0.495784
    Name: HVAC Electric Demand, dtype: float64
```

<Apartment building [Total Electric Demand] is not strongly correlated with any weather condition>
<Apartment building [HVAC Electric Demand] is highly correlated with Wetbulb Temperature>

```
df_office_weather_corr = df_office.iloc[:,4:]
df_office_weather_corr_mat = df_office_weather_corr.corr()
df_office_weather_corr_mat["Total Electric Demand"].sort_values(ascending=False)
```

```
Total Electric Demand     1.000000
HVAC Electric Demand      0.681183
Solar Radiation           0.537996
Sky Clearness             0.504569
Wind Speed                0.287623
Wind Direction            0.209546
Wetbulb Temperature       0.032890
Relative Humidity        -0.392646
Name: Total Electric Demand, dtype: float64
```

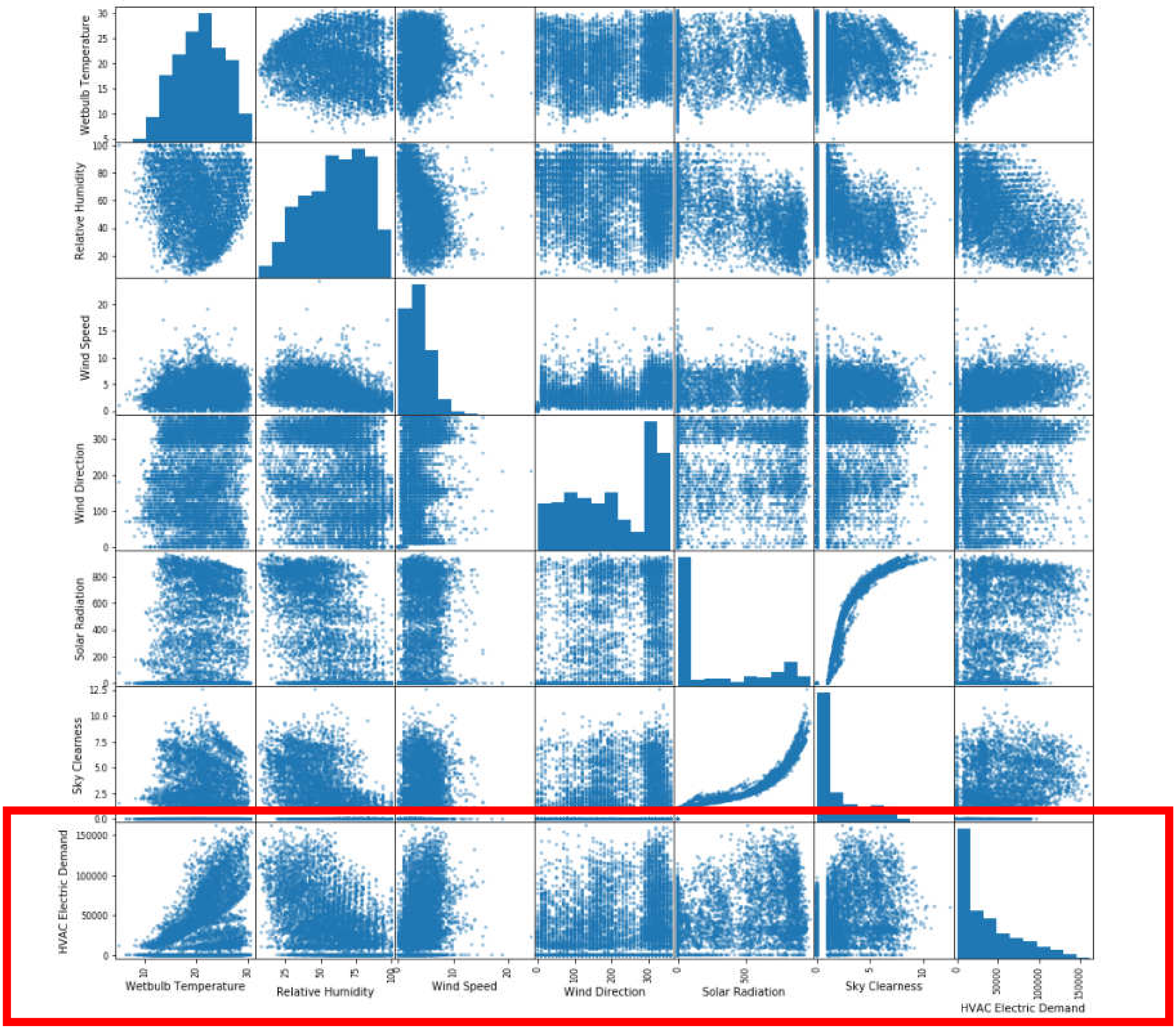
```
df_office_weather_corr_mat["HVAC Electric Demand"].sort_values(ascending=False)
```

```
HVAC Electric Demand      1.000000
Total Electric Demand     0.681183
Solar Radiation           0.583861
Sky Clearness             0.553459
Wetbulb Temperature       0.432901
Wind Speed                0.396214
Wind Direction            0.301195
Relative Humidity        -0.567162
Name: HVAC Electric Demand, dtype: float64
```

<Office building [Total Electric Demand] is strongly correlated with [solar radiation] and [Sky Clearness]>
<Office building [HVAC Electric Demand] is also highly correlated with[Solar Radiation], [Sky Clerarness] >

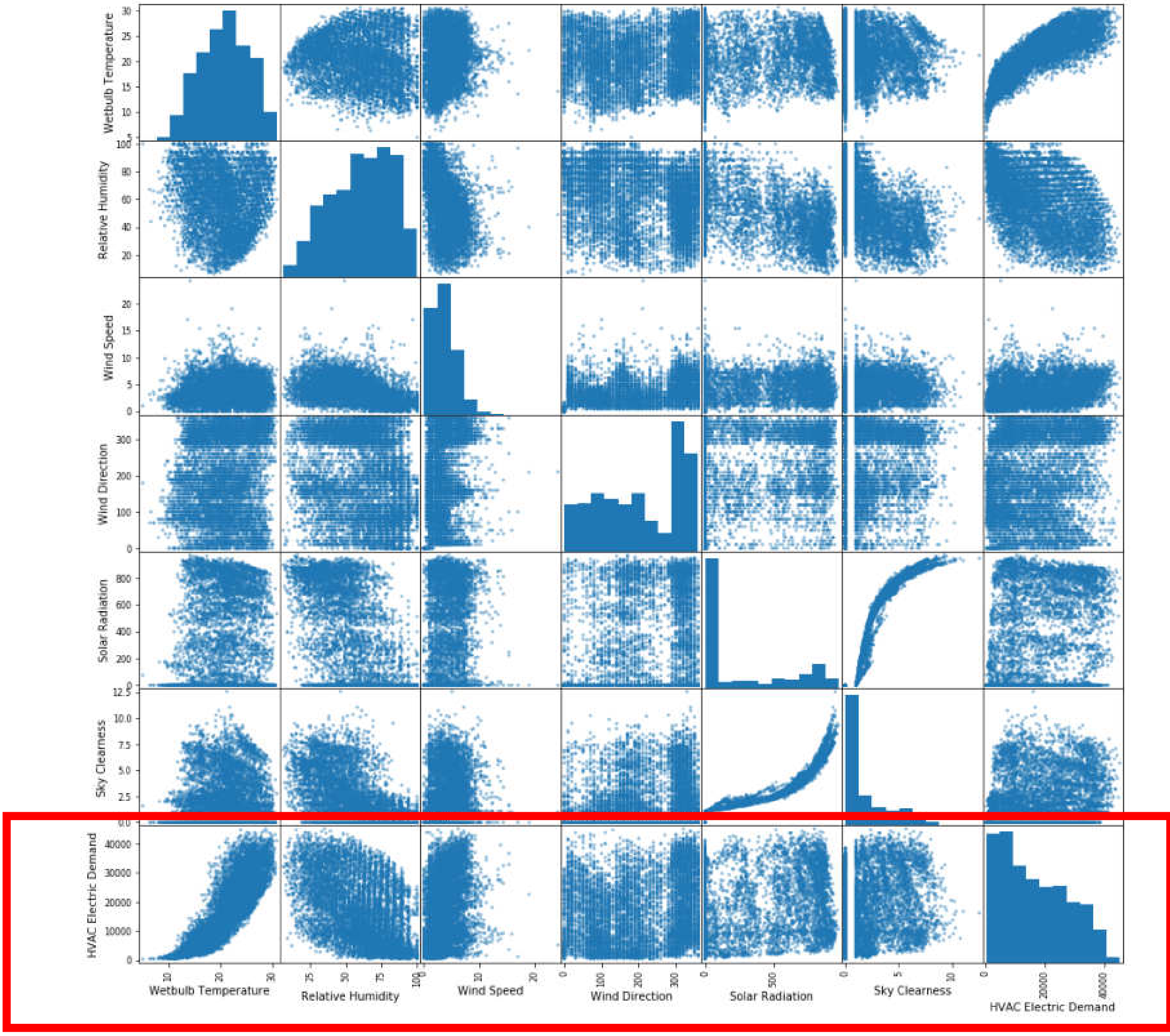
4- Create a scatter plot of the weather conditions vs HVAC demand and explain what you can learn from these associations for each building.

```python
df_office_weather = df_office.iloc[:,4:12]
del df_office_weather["Total Electric Demand"]


pd.plotting.scatter_matrix(df_office_weather, figsize=(16,16))
plt.show()
```



< We could verify that HVAC and. Wetbulb are highly correlated in Office building>

Project 2_ Homework_ Hosung Lee



< We could also verify that HVAC and Wetbulb are highly correlated positively,
Relative humidity however, negatively correlated>

5- Split the data into training and test with a ratio of 0.2 as the test

5- Split the data into training and test with a ratio of 0.2 as the te data.

```
df_office = pd.read_excel("office-1.xlsx")
df_apt = pd.read_excel("apartment-1.xlsx")
```

```
from sklearn.model_selection import train_test_split
train_set_1, test_set_1 = train_test_split(df_office, test_size=0.2, random_state=42)
train_set_2, test_set_2 = train_test_split(df_apt, test_size=0.2, random_state=42)
```

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.

- Before training, do not forget to standardize your input.
- Report the MSE value for the training and test data for both buildings.

o Before training, do not forget to standardize your input. o Report the MSE value for the training and test data for both buildings.

```
data = train_set_1.copy()
#data = train_set_2.copy()
#data = test_set_1.copy()
#data = test_set_2.copy()
```

```
y = data.pop('HVAC Electric Demand')
X = data.iloc[:,4:10]


from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X_scaled, y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
from sklearn.metrics import mean_squared_error
y_predicted = lin_reg.predict(X_scaled)
lin_mse = mean_squared_error(y, y_predicted)
lin_rmse = np.sqrt(lin_mse)
lin_rmse
```

```
24520.33719052414
```

1) Train Data [Office] , (MSE ,RMSE) = (601246935.9370012, 24520.33719052414)
2) Train Data [Apartment] , (MSE ,RMSE) = (13133338.825460138, 3623.9948710587514)
3) Test Data [Office] , (MSE ,RMSE) = (568999703.9683799, 23853.714678606764)
4) Test Data [Apartment] , (MSE ,RMSE) = (13085962.638976622, 3617.4525068031817)

Regression	Train_Office	Test_Office	Train_Apartment	Test_Apartment
6. Linear	548663025	512750777.49	9776844.84	9723236.742

MSE Values are so large in this prediction problem. So, we move to next question to handle input data.

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

- 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.)
- Second, use OneHotEncoder object and transform 'month' column and concatenate it to your weather conditions input.
- Third, use pandas map method and convert the 'hour' column values as follows:
  - {0,1,2,3,4,5}-->value=0
  - {6,7,8,9}-->value=1
  - {10,11,12}-->value=2
  - {13,14,15,16}-->value=3
  - {17,18,19}-->value=4
  - {20,21,22,23}-->value=5
- Fourth, apply OneHotEncoder on this new column and concatenate

First)

One-Hot Encoding is used to transform categorical data into numerical data. However, when the data is not continuous (i.e. 3.6 means March or April?), One-Hot Encoding can be effectively transforms the data since only '1' is placed one and only place within its row.
Example.

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

→

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

Second)

But in this project the month is already changed into numerical values. Therefore, I can assume that I can perform only concat function to add month to weather inputs.

Project 2_ Homework_ Hosung Lee

```
df_concat_month=pd.concat([months,X1], axis=1)
df_concat_month
```

	1	2	3	4	5	6	7	8	9	10	11	12	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation	Sky Clearness
0	1	0	0	0	0	0	0	0	0	0	0	0	13.705041	78	0.5	190	0	0.0
1	1	0	0	0	0	0	0	0	0	0	0	0	13.758291	82	2.1	120	0	0.0
2	1	0	0	0	0	0	0	0	0	0	0	0	13.595604	85	2.1	120	0	0.0
3	1	0	0	0	0	0	0	0	0	0	0	0	13.512457	87	2.1	140	0	0.0
4	1	0	0	0	0	0	0	0	0	0	0	0	13.227824	88	1.0	150	0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8755	0	0	0	0	0	0	0	0	0	0	0	1	14.182659	67	3.6	290	0	0.0
8756	0	0	0	0	0	0	0	0	0	0	0	1	14.062586	69	3.1	270	0	0.0
8757	0	0	0	0	0	0	0	0	0	0	0	1	14.025331	71	2.6	260	0	0.0
8758	0	0	0	0	0	0	0	0	0	0	0	1	13.889234	73	3.1	260	0	0.0
8759	0	0	0	0	0	0	0	0	0	0	0	1	13.839001	75	3.6	270	0	0.0

8760 rows × 18 columns

Third] Range

column values as follows:

[279]:
```
import pandas as pd
from pandas import Series, DataFrame

hour  = month.iloc[:,2]
df_hour = Series(hour)

def hourchange(hour):

    if   hour <= 5:
         hourRange = '0'

    elif hour <= 9 :
         hourRange ='1'

    elif hour <= 12 :
         hourRange = '2'

    elif hour <= 16 :
         hourRange = '3'

    elif hour <= 19 :
         hourRange = '4'

    elif hour <= 23 :
         hourRange = '5'
    else:
        hourRange =pd.np.nan
    return hourRange
```

[278]:
```
srhourRange = hour.map(hourchange)
srhourRange
```

[278]:
```
0       0
1       0
2       0
3       0
4       0
       ..
8755    5
8756    5
8757    5
```

Project 2_ Homework_ Hosung Lee

Fourth]

Fourth, apply OneHotEncoder on this new column and concatenate it to your input.

```
hours = pd.get_dummies(srhourRange.hour)
hours
```

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

8760 rows × 6 columns

```
df_concat_month_hour=pd.concat([hours,df_concat_month], axis=1)
df_concat_month_hour
weather_condition = df_concat_month_hour
weather_condition
```

	0	1	2	3	4	5	1	2	3	4	...	9	10	11	12	Wetbulb Temperature	Relative Humidity	Wind Speed	Wind Direction	Solar Radiation	Sky Clearness
0	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0	13.705041	78	0.5	190	0	0.0
1	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0	13.758291	82	2.1	120	0	0.0
2	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0	13.595604	85	2.1	120	0	0.0
3	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0	13.512457	87	2.1	140	0	0.0
4	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0	13.227824	88	1.0	150	0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8755	0	0	0	0	0	1	0	0	0	0	...	0	0	0	1	14.182659	67	3.6	290	0	0.0
8756	0	0	0	0	0	1	0	0	0	0	...	0	0	0	1	14.062586	69	3.1	270	0	0.0
8757	0	0	0	0	0	1	0	0	0	0	...	0	0	0	1	14.025331	71	2.6	260	0	0.0
8758	0	0	0	0	0	1	0	0	0	0	...	0	0	0	1	13.889234	73	3.1	260	0	0.0
8759	1	0	0	0	0	0	0	0	0	0	...	0	0	0	1	13.839001	75	3.6	270	0	0.0

8760 rows × 24 columns

New Weather condition as input features are available for both building.

```
df_concat_month_hour_off=pd.concat([hours_office,df_concat_month_office], axis=1)
df_concat_month_hour_apt=pd.concat([hours_apt,df_concat_month_apt], axis=1)
df_concat_month_hour_apt
```

	0	1	2	3	4	5	1	2	3	4	...	9	10	11	12	Wetbulb Temperature	Relative Humidity	Wind Speed	Win Directio
0	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0	13.705041	78	0.5	19
1	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0	13.758291	82	2.1	12
2	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0	13.595604	85	2.1	12
3	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0	13.512457	87	2.1	14
4	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0	13.227824	88	1.0	15
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
8755	0	0	0	0	0	1	0	0	0	0	...	0	0	0	1	14.182659	67	3.6	29
8756	0	0	0	0	0	1	0	0	0	0	...	0	0	0	1	14.062586	69	3.1	27
8757	0	0	0	0	0	1	0	0	0	0	...	0	0	0	1	14.025331	71	2.6	26
8758	0	0	0	0	0	1	0	0	0	0	...	0	0	0	1	13.889234	73	3.1	26
8759	1	0	0	0	0	0	0	0	0	0	...	0	0	0	1	13.839001	75	3.6	27

8760 rows × 24 columns

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

<MSE Value change>

Regression	Train_Office	Test_Office	Train_Apartment	Test_Apartment
6. Linear	548663025	512750777.49	9776844.84	9723236.742
8. Weather_added	498836951	470168938.31	6375746.048	6254283.070

The overall performance in MSE and RMSE has improved but it can not be said it is good regression performance.

9- Explain what regularization is in supervised learning and repeat step 8 using sklearn Ridge and Lasso classes based on the below instruction:

Regularization is supervised learning is used to prevent regression model from overfitting.
Using L1-norm and L2-norm, Lasso and Ridge Regression is used to its regularization.

Using Ridge Regression) MSE for alpha={0, 0.005, 0.05,0.1,1}

<MSE Value Change>

Regression	Train_Office	Test_Office	Train_Apartment	Test_Apartment
6. Linear	548663025	512750777.49	9776844.84	9723236.742
8. Weather_added	498836951	470168938.31	6375746.048	6254283.070
9.1 Ridge alpha=0	499184061	471717732	6375441.70	6239333.79
9.1 Ridge alpha=0.005	499184061 .	466359199	6376312	6371859
9.1 Ridge alpha=0.05	498750453	466359205	6375438	6239277
9.1 Ridge alpha=0.1	498750454	466359220	6375439	6239278
9.1 Ridge alpha=1	498750566	466361238	6375441	6239333
9.2 Lasso , alpha=0	498750453	466359199	6375438	6239277
9.2 Lasso, alpha=0.05	498750453	466359199	6375438	6239277
9.2 Lasso, alpha=0.05	498750453	466359200	6375439	6239277
9.2 Lasso, alpha=0.1	498750454	466359200	6375439	6239277
9.2 Lasso, alpha=1	498750550.	466359200	6375534	6239278

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):

Regression	Train_Office	Test_Office	Train_Apartment	Test_Apartment
6. Linear	548663025	512750777.49	9776844.84	9723236.742
8. Weather_added	498836951	470168938.31	6375746.048	6254283.070
9.1 Ridge alpha=0	499184061	471717732	6375441.70	6239333.79
9.1 Ridge alpha=0.005	499184061	466359199	6376312	6371859
9.1 Ridge alpha=0.05	498750453	466359205	6375438	6239277
9.1 Ridge alpha=0.1	498750454	466359220	6375439	6239278
9.1 Ridge alpha=1	498750566	466361238	6375441	6239333
9.2 Lasso , alpha=0	498750453	466359199	6375438	6239277
9.2 Lasso, alpha=0.05	498750453	466359199	6375438	6239277
9.2 Lasso, alpha=0.05	498750453	466359200	6375439	6239277
9.2 Lasso, alpha=0.1	498750454	466359200	6375439	6239277
9.2 Lasso, alpha=1	498750550.	466359200	6375534	6239278
Adaboost	510662770	479125250	6675025.140	6323352.056
Bagging with SVR	1028	1028.10902	1015.4115244	1017.30571
RandomForest(depth=10)	262	262.728	14.134890	14.541529859

Using Ensemble Method the RandomForest with depth 10 shows the best performance in MSE.