Department of Computer Science and Engineering (Data Science)

Subject: Machine Learning – I (DJ19DSC402)

AY: 2021-22

Experiment 10 (Mini Project)

Regression Analysis For Prediction Of Energy Efficiency In Residential Structures

- Dyuti Vartak (60009200060)

- Pramit Bhatia (60009200065)

Abstract

The HVAC (Heating, Ventilation and Cooling) industry describes the amount of conditioning homes need as heating and cooling loads. Heating loads refer to the amount of heat energy required to be added to an area to maintain the temperature in an adequate range. Cooling loads refer to the amount of heat energy that needs to be removed from an area to maintain the temperature in an adequate range.

This project aims to provide a machine-learning derived solution to predict the effects of the given attributes, i.e. Relative Compactness, Surface Area, Wall Area, Roof Area, Overall Height, Orientation, Glazing Area, Glazing Area Distribution, on two output variables, i.e Heating Load and Cooling Load for residential buildings.

Data Description

In the procured data-set, we have 768 records and for each, 8 attributes (denoted by X1 - X8), and 2 outputs (denoted by Y1 and Y2). The aim is to create a machine learning model to accurately predict these 2 outputs using the given attributes.

Specifically:

X1 Relative Compactness

X2 Surface Area

X3 Wall Area

X4 Roof Area

X5 Overall Height

X6 Orientation

X7 Glazing Area

X8 Glazing Area Distribution

To Predict:

Y1 Heating Load

Y2 Cooling Load

Data-Set

X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
0.98	514.5	294	110.25	7	2	0	0	15.55	21.33
0.98	514.5	294	110.25	7	3	0	0	15.55	21.33
0.98	514.5	294	110.25	7	4	0	0	15.55	21.33
0.98	514.5	294	110.25	7	5	0	0	15.55	21.33
0.9	563.5	318.5	122.5	7	2	0	0	20.84	28.28
0.9	563.5	318.5	122.5	7	3	0	0	21.46	25.38
0.9	563.5	318.5	122.5	7	4	0	0	20.71	25.16
0.9	563.5	318.5	122.5	7	5	0	0	19.68	29.6
0.86	588	294	147	7	2	0	0	19.5	27.3

Data Pre-processing

+ Code   + Text

Machine Learning Mini Project

Authors:
Pramit Bhatia -- 60009200065
Dyuti Vartak -- 60009200060

Importing the libraries

```
[4] import numpy as np
    import matplotlib.pyplot as plt
    import pandas as pd
```

Importing the dataset

```
[60] dataset = pd.read_csv('/content/ENB2012_data.csv')
```

Taking care of missing data

```
[8] dataset.isnull().sum()
```

```
X1    0
X2    0
X3    0
X4    0
X5    0
X6    0
X7    0
X8    0
Y1    0
Y2    0
dtype: int64
```

There is no missing data present.

Preprocessing Function

```
[ ] def preprocessing(dataset):
      X = dataset.iloc[:, :-1].values
      y = dataset.iloc[:, -1].values

      # Splitting the dataset into the Training set and Test set
      from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)

      # Feature Scaling
      from sklearn.preprocessing import StandardScaler
      sc = StandardScaler()
      X_train[:, :] = sc.fit_transform(X_train[:, :])
      X_test[:, :] = sc.transform(X_test[:, :])


      return X_train, X_test, y_train, y_test
```

Exploratory Data Analysis

EDA

```
[9] dataset
```

	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
0	0.98	514.5	294.0	110.25	7.0	2	0.0	0	15.55	21.33
1	0.98	514.5	294.0	110.25	7.0	3	0.0	0	15.55	21.33
2	0.98	514.5	294.0	110.25	7.0	4	0.0	0	15.55	21.33
3	0.98	514.5	294.0	110.25	7.0	5	0.0	0	15.55	21.33
4	0.90	563.5	318.5	122.50	7.0	2	0.0	0	20.84	28.28
...	...	...	...	...	...	...	...	...	...	...
763	0.64	784.0	343.0	220.50	3.5	5	0.4	5	17.88	21.40
764	0.62	808.5	367.5	220.50	3.5	2	0.4	5	16.54	16.88
765	0.62	808.5	367.5	220.50	3.5	3	0.4	5	16.44	17.11
766	0.62	808.5	367.5	220.50	3.5	4	0.4	5	16.48	16.61
767	0.62	808.5	367.5	220.50	3.5	5	0.4	5	16.64	16.03

768 rows × 10 columns

```
dataset.info()
```

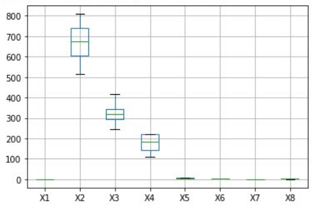
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   X1      768 non-null    float64
 1   X2      768 non-null    float64
 2   X3      768 non-null    float64
 3   X4      768 non-null    float64
 4   X5      768 non-null    float64
 5   X6      768 non-null    int64
 6   X7      768 non-null    float64
 7   X8      768 non-null    int64
 8   Y1      768 non-null    float64
 9   Y2      768 non-null    float64
dtypes: float64(8), int64(2)
memory usage: 60.1 KB
```

```
[11] dataset.describe()
```

	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
count	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000	768.000000	768.00000	768.000000	768.000000
mean	0.764167	671.708333	318.500000	176.604167	5.25000	3.500000	0.234375	2.81250	22.307201	24.587760
std	0.105777	88.086116	43.626481	45.165950	1.75114	1.118763	0.133221	1.55096	10.090196	9.513306
min	0.620000	514.500000	245.000000	110.250000	3.50000	2.000000	0.000000	0.00000	6.010000	10.900000
25%	0.682500	606.375000	294.000000	140.875000	3.50000	2.750000	0.100000	1.75000	12.992500	15.620000
50%	0.750000	673.750000	318.500000	183.750000	5.25000	3.500000	0.250000	3.00000	18.950000	22.080000
75%	0.830000	741.125000	343.000000	220.500000	7.00000	4.250000	0.400000	4.00000	31.667500	33.132500
max	0.980000	808.500000	416.500000	220.500000	7.00000	5.000000	0.400000	5.00000	43.100000	48.030000

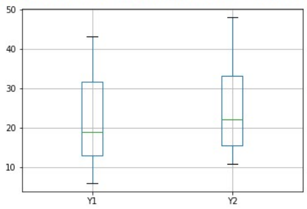
```
dataset.boxplot(column=["X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff45eca5ad0>
```




```
[35] dataset.boxplot(column=["Y1", "Y2"])
```

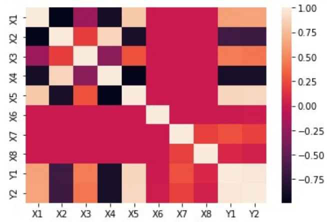
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff45ec77a50>
```



Correlation Analysis

```
import seaborn as sns
sns.heatmap(dataset.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff45f415190>
```




```
[55] (dataset[{"X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8"}]).corr() > .9
```

	X4	X3	X7	X2	X1	X6	X5	X8
X4	True	False	False	False	False	False	False	False
X3	False	True	False	False	False	False	False	False
X7	False	False	True	False	False	False	False	False
X2	False	False	False	True	False	False	False	False
X1	False	False	False	False	True	False	False	False
X6	False	False	False	False	False	True	False	False
X5	False	False	False	False	False	False	True	False
X8	False	False	False	False	False	False	False	True

```
(dataset[{"X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8"}]).corr() < -.9
```

	X4	X3	X7	X2	X1	X6	X5	X8
X4	False	False	False	False	False	False	True	False
X3	False	False	False	False	False	False	False	False
X7	False	False	False	False	False	False	False	False
X2	False	False	False	False	True	False	False	False
X1	False	False	False	True	False	False	False	False
X6	False	False	False	False	False	False	False	False
X5	True	False	False	False	False	False	False	False
X8	False	False	False	False	False	False	False	False

X1 and X2 are related
X4 and X5 related

```
[58] dataset[{"X1", "X5", "X2", "X4"}].corr()
```

	X2	X1	X5	X4
X2	1.000000	-0.991901	-0.858148	0.880720
X1	-0.991901	1.000000	0.827747	-0.868823
X5	-0.858148	0.827747	1.000000	-0.972512
X4	0.880720	-0.868823	-0.972512	1.000000

Here, we perform Correlation Analysis on the given data. We find that the fields X1 and X2, i.e. relative compactness and surface area are related. Also, fields X4 and X5, i.e roof area and overall height are related. Thus, we create 4 datasets with different combinations of these two sets, i.e one with X1/X4, one with X2/X4, one with X1/X5,

Let us consider dataset2 and drop X1 / X4

```
dataset2 = dataset.drop(columns={"X1", "X4"})
```

Let us consider dataset3 and drop X1 / X5

```
[82] dataset3 = dataset.drop(columns={"X1", "X5"})
```

Let us consider dataset4 and drop X2 / X4

```
[83] dataset4 = dataset.drop(columns={"X2", "X4"})
```

Let us consider dataset5 and drop X2 / X5

```
[84] dataset5 = dataset.drop(columns={"X2", "X5"})
```

Selecting A Machine Learning Model

To find out the most suitable machine learning model, we have made different functions which take X_train, X_test, y_train and y_test to create a classifier and find out the r2 score.

Finding Best Model

```
[19] def multiplelinearregression(X_train, X_test, y_train, y_test):
       from sklearn.linear_model import LinearRegression
       regressor = LinearRegression()
       regressor.fit(X_train, y_train)
       y_pred = regressor.predict(X_test)
       from sklearn.metrics import r2_score
       return r2_score(y_test, y_pred)
```

```
[20] def polynomialregression(X_train, X_test, y_train, y_test):
       from sklearn.preprocessing import PolynomialFeatures
       from sklearn.linear_model import LinearRegression
       poly_reg = PolynomialFeatures(degree = 4)
       X_poly = poly_reg.fit_transform(X_train)
       regressor = LinearRegression()
       regressor.fit(X_poly, y_train)
       y_pred = regressor.predict(poly_reg.transform(X_test))
       from sklearn.metrics import r2_score
       return(r2_score(y_test, y_pred))
```

```
[21] def decisiontree(X_train, X_test, y_train, y_test):
       from sklearn.tree import DecisionTreeRegressor
       regressor = DecisionTreeRegressor(random_state = 0)
       regressor.fit(X_train, y_train)
       y_pred = regressor.predict(X_test)
       from sklearn.metrics import r2_score
       return r2_score(y_test, y_pred)
```

```
[22] def randomForest(X_train, X_test, y_train, y_test):
       from sklearn.ensemble import RandomForestRegressor
       regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
       regressor.fit(X_train, y_train)
       y_pred = regressor.predict(X_test)
       from sklearn.metrics import r2_score
       return r2_score(y_test, y_pred)
```

```
[23] def svrs(X_train, X_test, y_train, y_test):
       from sklearn.preprocessing import StandardScaler
       sc_X = StandardScaler()
       sc_y = StandardScaler()
       X_train = sc_X.fit_transform(X_train)
       y_train = sc_y.fit_transform(y_train.reshape(-1,1))
       from sklearn.svm import SVR
       regressor = SVR(kernel = 'rbf')
       regressor.fit(X_train, y_train.ravel())
       y_pred = sc_y.inverse_transform(regressor.predict(sc_X.transform(X_test)).reshape(-1, 1))
       from sklearn.metrics import r2_score
       return r2_score(y_test, y_pred)
```

```
[74] def findbestmodel(X_train, X_test, y_train, y_test):
       result = []
       result.append([multiplelinearregression(X_train, X_test, y_train, y_test), "Multiple Linear"])
       result.append([polynomialregression(X_train, X_test, y_train, y_test), "Polynomial"])
       result.append([decisiontree(X_train, X_test, y_train, y_test), "Decision Tree"])
       result.append([randomForest(X_train, X_test, y_train, y_test), "Random Forest"])
       result.append([svrs(X_train, X_test, y_train, y_test), "Support Vector"])

       return max(result)
```

Algorithm

```
X_train, X_test, y_train, y_test = preprocessing(dataset)
print(findbestmodel(X_train, X_test, y_train, y_test))

X_train, X_test, y_train, y_test = preprocessing(dataset2)
print(findbestmodel(X_train, X_test, y_train, y_test))

X_train, X_test, y_train, y_test = preprocessing(dataset3)
print(findbestmodel(X_train, X_test, y_train, y_test))

X_train, X_test, y_train, y_test = preprocessing(dataset4)
print(findbestmodel(X_train, X_test, y_train, y_test))

X_train, X_test, y_train, y_test = preprocessing(dataset5)
print(findbestmodel(X_train, X_test, y_train, y_test))
```

```
[0.9787408395464385, 'Decision Tree']
[0.9811414231066395, 'Decision Tree']
[0.978718572714163, 'Decision Tree']
[0.9786238131121481, 'Decision Tree']
[0.9798032576402889, 'Decision Tree']
```

As we can see, the best result is obtained using Decision Tree Regression.
We tested the original dataset as well as other datasets obtained after correlation analysis.
The best accuracy was obtained on dataset2 where we dropped columns X1 and X4.
Accuracy Obtained: **0.9811414231066395**

As we can see, the best result is obtained using Decision Tree Algorithm for all the datasets. The highest accuracy was obtained when we selected columns X1 and X4 after correlation analysis.

Now, given that we know decision tree is the most suitable model, we can calculate the max depth of the decision tree as well.
The max depth returned here is 18 for all datasets.

```
18
[0.9787408395464385, 'Decision Tree']
18
[0.9811414231066395, 'Decision Tree']
18
[0.978718572714163, 'Decision Tree']
18
[0.9786238131121481, 'Decision Tree']
18
[0.9798032576402889, 'Decision Tree']
```

When we vary the max depths across the range(1, 18), the result that we get for dataset2 is:

```
def decisiontree(X_train, X_test, y_train, y_test, i):
  from sklearn.tree import DecisionTreeRegressor
  regressor = DecisionTreeRegressor(random_state = 0, max_depth = i)
  regressor.fit(X_train, y_train)
  y_pred = regressor.predict(X_test)
  print(regressor.tree_.max_depth)
  from sklearn.metrics import r2_score
  return r2_score(y_test, y_pred)
```

```
[29] X_train, X_test, y_train, y_test = preprocessing(dataset2)
     for i in range(1, 18):
       print(decisiontree(X_train, X_test, y_train, y_test, i))
```

```
1
0.7869390967491975
2
0.8998723641431341
3
0.9417408661816823
4
0.9567461524183587
5
0.9645505486878869
6
0.9733838011158988
7
0.9739985314379452
8
0.976758678789575
9
0.9763379994203787
10
0.9758537914300365
11
0.9760666105218394
12
0.9787172397389474
13
0.9803964826887537
14
0.9785098426626211
15
0.9800778801610648
16
0.9791086381570343
17
0.9785731267702577
```

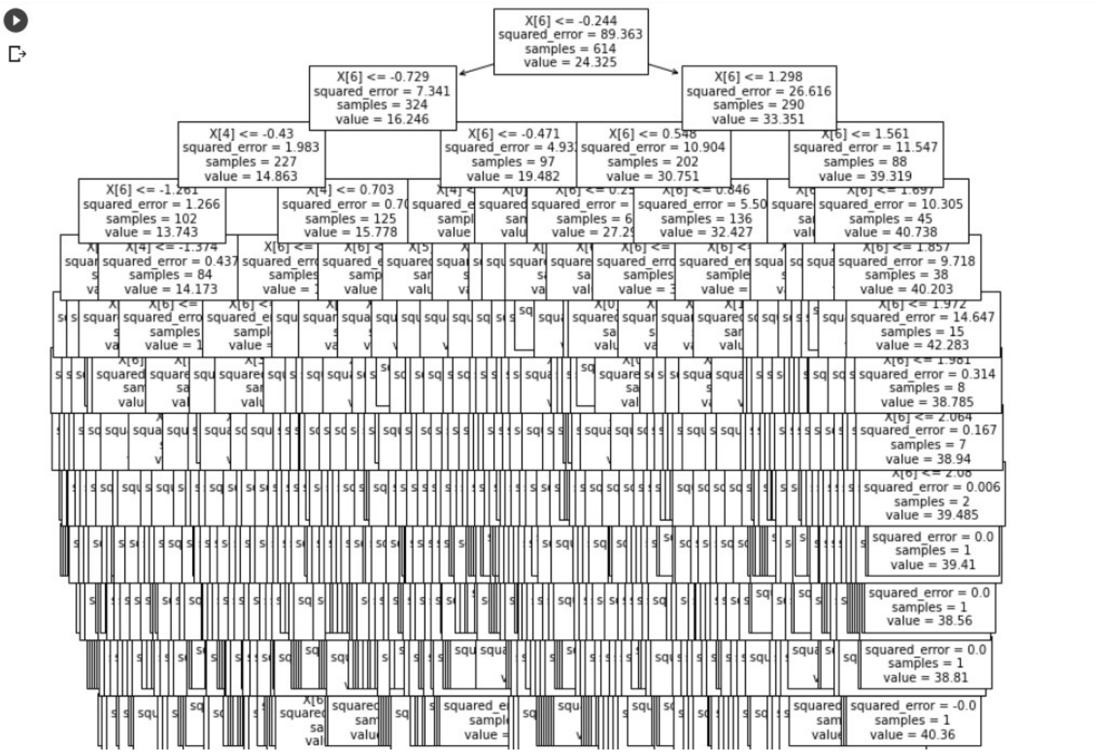
Thus, we find that the max accuracy was observed with max depth of 13.

Result Analysis

We found that the highest accuracy was observed with max depth of 13, and with the dataset that included the following features:

{X1, X3, X4, X6, X7, X8}

The decision tree is printed below:



Conclusion And Future Scope

To conclude, we figured out that not all the features from the given dataset impact the end result. The features “Surface Area” and “Overall Height” are closely related to the features “Relative Compactness” and “Roof Area” respectively. Thus, we can construct a machine learning model to predict the Heating Load and Cooling Load without these features.

Specifically:	To Predict:
X1 Relative Compactness	Y1 Heating Load
X2 Surface Area	Y2 Cooling Load
X3 Wall Area	
X4 Roof Area	
X5 Overall Height	
X6 Orientation	
X7 Glazing Area	
X8 Glazing Area Distribution	

The optimal machine learning algorithm to predict was found out via testing several algorithms such as Multiple Linear Regression, Decision Tree Regression, Polynomial Regression, SVM using Kernel and Random Forest. The highest accuracy was obtained using Decision Tree Regression in all the datasets that were tested. Accuracy: **0.9811414231066395**

The future scope for this project would be to use multiple machine learning algorithms and hyperparametize all of them to figure out the best **ensemble** model for our dataset. We can also use Boosting, i.e AdaBoost or XGBoost to improve the accuracy.