**Assignment 1**

***Description of the forecasting problem***

Weather data for the city of Szeged in Hungary, from 2006 to 2016, is available. This forecasting problem deals with predicting the **Apparent Temperature** value given a value of **Humidity** on that day.

Y = Apparent Temperature

X = Humidity

Where X is the independent variable/attribute, also known as predictor variable. Based on the values of X, we will predict the values of Y which is our variable of interest.

***Description of the available data***

|  |  |  |
| --- | --- | --- |
| **Column name** | **Type** | **Description** |
| Formatted Date | DateTime | Date and Time of the day |
| Summary | String | Short Summary of the day |
| Precip type | String | Type of precipitation |
| Temperature | Numeric | Actual Temperature |
| Apparent Temperature | Numeric | Temperature perceived by humans |
| Humidity | Numeric | Value of humidity |
| Wind Speed | Numeric | Speed of Wind |
| Wind Bearing | Numeric | Direction of the Wind |
| Visibility | Numeric | Distance at which an object or light can be clearly discerned |
| Loud Cover | Numeric | Total cover |
| Pressure | Numeric | Value of atmospheric pressure |
| Daily Summary | String | Overall summary for the day |

**Attributes used:**

Y (Variable of interest) = Apparent Temperature

X = Humidity

***Short overview of the selected algorithms***

**Linear Regression**

Regression – an approach for modeling the relationship between a dependent variable and independent variables

Linear regression – linear relationship between a dependent variable and independent variables

In simple linear regression, we predict scores on one variable from the scores on a second variable. The variable we are predicting is called the *criterion variable* and is referred to as Y. The variable we are basing our predictions on is called the *predictor variable* and is referred to as X. When there is only one predictor variable, the prediction method is called *simple regression*. If there are more than one predictor variables, the prediction is called *multivariate linear regression.*

**ARIMA (Time series)**

In statistics and econometrics and in particular in time series analysis an **autoregressive integrated moving average (ARIMA)** model is a generalization of an autoregressive moving average (ARMA) model. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting) ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity.

The AR part of ARIMA indicates that the evolving variable of interest is regressed on its own lagged (i.e., prior) values. The MA part indicates that the regression error is actually a linear combination of error terms whose values occurred contemporaneously and at various times in the past. The I (for "integrated") indicates that the data values have been replaced with the difference between their values and the previous values (and this differencing process may have been performed more than once). The purpose of each of these features is to make the model fit the data as well as possible.

***Specifics about how algorithms were applied and the evaluation procedure***

I am interested in finding the relationship between **Apparent Temperature** and **Humidity.** After closely observing the data, it becomes clear that there is an inverse relationship between both. To mathematically prove it, I calculate the co-relation value in MS Excel and it comes out to be negative i.e. -0.73618961, which indicates a strong inversely proportional relationship.

*CORREL (<Apparent Temp Values>, <Humidity values) = -0.73618961*

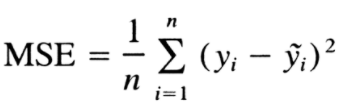
Note: correlation value approaching 1 indicates strong directly proportional dependency whereas a correlation value approaching -1 indicates a strong inversely proportional dependency.

Since a casual relationship between two parameters (Apparent Temperature and Humidity) needs to be calculated, the Linear Regression Algorithm should most likely be the best fit for this kind of problem statement. However, algorithms like Support Vector Regression and Decision trees are also used.

The dataset is divided into **training** and **test** data in ratio 4:1 respectively. The training data is used to train the model and the test data is used to check the accuracy of the model.

**Evaluation Procedure**: Mean Squared Error

The mean squared error tells us how close a regression line is to a set of points. It does this by taking the distances from the points to the regression line (these distances are the “errors”) and squaring them. The squaring is necessary to remove any negative signs. It also gives more weight to larger differences. It’s called the **mean**squared error as we are finding the average of a set of errors.



Where, n = number of data points

Yi =Actual value for data point i

Yi~ =Predicted value for data point i

**The smaller the mean squared error, the closer you are to finding the line of best fit. It means the algorithm which gives us the smallest value of MSE, will be our algorithm of choice.**

***Accuracy comparison***

|  |  |
| --- | --- |
| **Algorithm** | **MSE** |
| Linear Regression | 10.899382981520409 |
| Support Vector Regression | 11.979722416103748 |
| Decision Tree Analysis | 12.157561528442072 |

Since *Mean Square Error* is least for Linear regression, it implies that Linear regression, for this particular problem statement and dataset, is a better algorithm than Support vector regression and Decision tree to model the data at hand and make more accurate predictions.

***Code in Python***

**Linear Regression**

1. # Importing the libraries
2. **import** numpy as np
3. **import** matplotlib.pyplot as plt
4. **import** pandas as pd
6. #Reading the .CSV data file
7. dataset = pd.read\_csv('//Users//muhbandtekamshuru//Downloads//SVR//HungaryWeather500.csv')
8. """dataset.head()"""
9. X = dataset.iloc[:, 5].values
10. y = dataset.iloc[:, 4].values
12. y=y.reshape(-1,1)
13. X=X.reshape(-1,1)
15. # Dividing Data into Training and Test
16. **from** sklearn.cross\_validation **import** train\_test\_split
17. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)
19. **from** sklearn.linear\_model **import** LinearRegression
20. regressor = LinearRegression()
21. regressor.fit(X\_train, y\_train)
23. # Predicting the Test set results
24. y\_pred = regressor.predict(X\_test)
26. **import** statsmodels.formula.api as smf
27. a=regressor.score
29. dataset.columns=['Formatted Date', 'Summary', 'Precip Type', 'Temperature (C)',
30. 'ApparentTemperature', 'Humidity', 'Wind Speed (km/h)',
31. 'Wind Bearing (degrees)', 'Visibility (km)', 'Loud Cover',
32. 'Pressure (millibars)', 'Daily Summary']
34. **import** statsmodels.formula.api as sm
36. model = sm.ols(formula='ApparentTemperature ~ Humidity', data=dataset)
37. fitted1 = model.fit()
38. fitted1.summary()
40. y\_pred=y\_pred.reshape(-1,1)
41. a=sum(np.square(y\_test-y\_pred))
43. **from** sklearn.metrics **import** mean\_squared\_error
44. mean\_squared\_error(y\_test, y\_pred)

**MSE:** 10.899382981520409

**Support Vector Regression**

1. **import** numpy as np
2. **import** matplotlib.pyplot as plt
3. **import** pandas as pd
5. dataset = pd.read\_csv('//Users//muhbandtekamshuru//Downloads//SVR//HungaryWeather500.csv')
7. X = dataset.iloc[:, 5].values
8. y = dataset.iloc[:, 4].values
10. X=X.reshape(-1,1)
11. y=y.reshape(-1,1)
13. **from** sklearn.cross\_validation **import** train\_test\_split
14. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)
16. **from** sklearn.svm **import** SVR
17. regressor = SVR(kernel = 'rbf')
19. regressor.fit(X\_train, y\_train)
21. # Predicting the Test set results
22. y\_pred = regressor.predict(X\_test)
23. # Predicting a new result
25. # Visualising the SVR results
26. plt.scatter(X, y, color = 'red')
27. plt.plot(X, regressor.predict(X), color = 'blue')
28. plt.title('Truth or Bluff (SVR)')
29. plt.xlabel('Humidity')
30. plt.ylabel('Apparent Temperature')
31. plt.show()
33. # Visualising the SVR results (for higher resolution and smoother curve)
34. X\_grid = np.arange(min(X), max(X), 0.01) # choice of 0.01 instead of 0.1 step because the data is feature scaled
35. X\_grid = X\_grid.reshape((len(X\_grid), 1))
36. plt.scatter(X, y, color = 'red')
37. plt.plot(X\_grid, regressor.predict(X\_grid), color = 'blue')
38. plt.title('Truth or Bluff (SVR)')
39. plt.xlabel('Humidity')
40. plt.ylabel('Apparent Temperature')
41. plt.show()
43. y\_pred=y\_pred.reshape(-1,1)
45. **import** numpy as np
46. a=sum(np.square(y\_test-y\_pred))
48. **from** sklearn.metrics **import** mean\_squared\_error
49. mean\_squared\_error(y\_test, y\_pred)

**MSE:** 11.979722416103748

**Decision Tree**

1. # Importing the libraries
2. **import** numpy as np
3. **import** matplotlib.pyplot as plt
4. **import** pandas as pd
6. dataset = pd.read\_csv('//Users//muhbandtekamshuru//Downloads//SVR//HungaryWeather500.csv')
7. X=X.reshape(-1,1)
9. y=y.reshape(-1,1)
11. **from** sklearn.cross\_validation **import** train\_test\_split
12. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)
14. # Fitting Decision Tree Regression to the dataset
15. **from** sklearn.tree **import** DecisionTreeRegressor
16. regressor = DecisionTreeRegressor()
18. regressor.fit(X\_train, y\_train)
20. # Predicting the Test set results
21. y\_pred = regressor.predict(X\_test)
22. # Predicting a new result
23. regressor.score
25. y\_pred=y\_pred.reshape(-1,1)
27. a=np.square(y\_pred-y\_test)
29. **from** sklearn.metrics **import** mean\_squared\_error
30. mean\_squared\_error(y\_test, y\_pred)

**MSE:** 12.157561528442072