PROJECT APPENDIX

Predictive Analytics for IoT data streams

Saran Sai C AM.EN.U4CSE17017
Abhinav D AM.EN.U4CSE17023
Dheeraj M AM.EN.U4CSE17022

Department of Computer Science and Engineering,
Amrita Vishwa Vidyapeetham,
Amritapuri Campus, Kollam

Guide: Simi.S **Co-ordinator:Greeshma Sarath**

INDEX

Softwares tools and Platforms

1. IDE:
    a. Jupyter Notebook
    b. Kaggle

2. Python Packages:
    a. Pandas
    b. Numpy
    c. Sklearn
        i. linear_model
        ii. DecisionTreeRegressor
        iii. RandomForestRegressor
        iv. StandardScaler
        v. SGDRegressor
        vi. SVR

Steps to Run

1. Download and Install Jupyter Notebook.

2. Download all required Python packages.

3. Import all the required packages

4. Place all the CSV files(data sets) in appropriate locations.

5. Run the cells one by one.

Code Base: Complete code for the project can be found here:
https://github.com/abhinavdv/Predictive_Analytics_For_Solar_power_gen

Algorithms Used

1) AMWR algorithm

This is a prediction model which utilizes a moving window of data for training the model; and Once new data arrives, it calculates an error and retrains the model accordingly. AMWR can be implemented using the following 3 steps:

a)Selection of regression algorithm; ⇒ SVM has been chosen here

b) Finding optimum training window size; ⇒ a window size of 30 is chosen

c) Size of the prediction horizon.==> 1(as we are only predicting the next value)

Algorithm 1: Adaptive Prediction Window Size

```
Function PREDICTIONWINDOW(yact, ypred):
    MAPE = mean(abs((yact  ypred)/yact)  100);
    if MAPE ≥ x% then
        Prediction Window = Prediction Window + 1;
    end
    else if MAPE ≤ y% then
        Prediction Window = Prediction Window - 1;
    end
    else
        Prediction Window = Prediction Window;
    end
    return Prediction Window;
    end function
```

2) We have used SVM with the RBF kernel. RBF kernel is a function whose value depends on the distance from the origin or from some point. Gaussian Kernel is of the following format:

$$K(X_1, X_2) = exponent(-\gamma \|X_1 - X_2\|^2)$$

||X1 — X2 || = Euclidean distance between X1 & X2

DataSets

1. Temperature Readings: IoT Devices(Main dataset)
    - Temperature readings of an enterprise building room (admin), both inside and outside. This was recorded at random intervals. The recording speed was per second
    - https://www.kaggle.com/atulanandjha/temperature-readings-iot-devices

2. Madrid Traffic reading: IoT Devices(reference dataset)
    - Traffic readings of the city of Madrid.
    - https://github.com/adnanakbr/PredictiveAnalytics/blob/master/03-2014.zip

Screenshots of code

```
#main function
if __name__ == '__main__':
    MAPE_list = list()
    date=datetime.datetime(2020, 5, 25, 6, 15)
    dataprocess(df,date)
    print(MAPE_list)
    MAPE_npArray = np.array(MAPE_list)
    print(MAPE_npArray[:,0])
    mape = np.mean(np.abs((MAPE_npArray[:,0] - MAPE_npArray[:,1])/MAPE_npArray[:,0]))*100
    print("MAPE is: "+ str(mape))
```

executed in 641ms, finished 13:51:45 2021-03-27

```
[[5951.0, 5001.349273357164], [6507.428571, 5314.106613749326], [6520.285714, 5324.146679710648], [6311.0, 5204.504597943816],
[6447.142857, 5246.926925260154], [6482.142857, 5295.952421133673], [6388.714286, 5260.934964068658], [6383.285714, 5223.726140
74077], [6235.285714, 5127.083185720567], [6600.857143, 5363.0920782623425], [5829.142857, 4933.92332268176], [6337.0, 5194.784
50438123], [5655.857143, 4735.675131332344], [6259.714286, 5112.425899817844], [6264.125, 5151.030559757643], [6282.142857, 516
7.174161880497], [6321.142857, 5162.472531164948], [6371.428571, 5204.672972631726], [6337.571429, 5201.7494185161395], [6018.7
14286, 5014.948750760932]]
[5951.       6507.428571 6520.285714 6311.       6447.142857 6482.142857
 6388.714286 6383.285714 6235.285714 6600.857143 5829.142857 6337.
 5655.857143 6259.714286 6264.125    6282.142857 6321.142857 6371.428571
 6337.571429 6018.714286]
MAPE is: 17.708557447865406
```

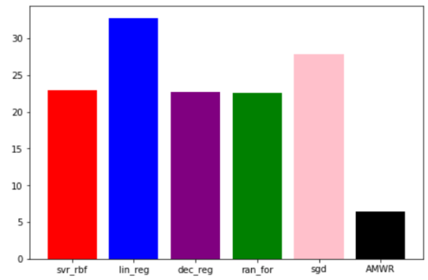
```
def pred(x, y, x_test):
    svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)
    y_rbf = svr_rbf.fit(x, np.ravel(y,order='C'))
    return svr_rbf.predict(x_test)



def AMWR(df):
    x_total = df[["AMBIENT_TEMPERATURE","MODULE_TEMPERATURE","IRRADIATION"]].to_numpy().tolist()
    y_total = df[["DAILY_YIELD"]].to_numpy().tolist()
    x = x_total[:len(df)-1]
    y = y_total[:len(df)-1]
    x_test = x_total[len(df)-1:]
    y_test = y_total[len(df)-1:]
    y_pred = pred(x,y,x_test)
    if(y_test!=0):
        MAPE_list.append([y_test[0][0],y_pred[0]])
```

executed in 7ms, finished 14:07:42 2021-03-27

```
AMWR              :6.447033486278069
```

```
[22]: import matplotlib.pyplot as plt
      fig = plt.figure()
      ax = fig.add_axes([0,0,1,1])
      algos = ['svr_rbf', 'lin_reg', 'dec_reg', 'ran_for', 'sgd','AMWR']
      mape_errors = TotalError[0]+[6.447033486278069]
      ax.bar(algos,mape_errors,color=['red', 'blue', 'purple', 'green', 'pink','black'])
      plt.show()
```

executed in 130ms, finished 20:48:42 2020-12-25



References

- https://www.researchgate.net/publication/317375359_Predictive_Analytics_for_Complex_IoT_Data_Streams
- https://www.researchgate.net/publication/275072928_A_Proactive_Complex_Event_Processing_Method_for_Large-Scale_Transportation_Internet_of_Things
- https://ieeexplore.ieee.org/abstract/document/8861915
- https://www.researchgate.net/publication/260322786_Predictive_Complex_Event_Processing_A_conceptual_framework_for_combining_Complex_Event_Processing_and_Predictive_Analytics
- https://www.researchgate.net/publication/317375359_Predictive_Analytics_for_Complex_IoT_Data_Streams
- https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3862
- https://ieeexplore.ieee.org/document/7389075
- https://ieeexplore.ieee.org/document/8756194
- https://ieeexplore.ieee.org/document/8230000