Aniket Amar Thopte

Electric Load Forecasting using different techniques

Linear Regression

Introduction:

The problem statement deals with hourly load forecasting. We have been given dataset from 2002 to 2005 (i.e. 4 years data) on hourly basis. In our case, I have used **Simple Linear Regression** technique for forecasting the load for 2006 year on hourly basis.

Dataset:

The Dataset file is named **‘history.csv**’. The dataset provided has 4 variable columns namely ‘Date’ in dd/mm/yy (as per python date stamp), ‘Hour’ ranging from 1 to 24, ‘Temperature’ & ‘Load’. The total **number of observations given is 35064**. Another excel file containing the 2006 data is provided named ’**fcst.csv**’ in which load column is kept blank to be forecasted.

Methodology:

The coding was done on Python (Jupyter Notebook). The various functions used in the code are elaborated in the Jupyter notebook using ‘Markdown Feature’. The forecasting is done using a machine learning algorithm called Simple Linear Regression.

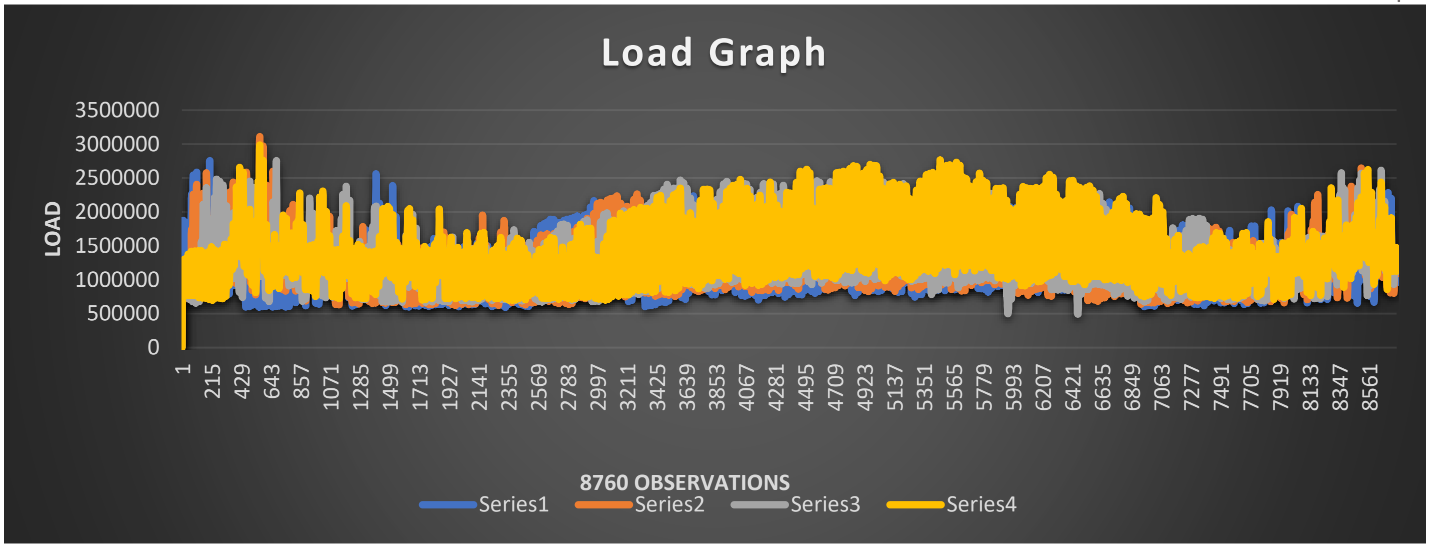
In statistics, simple linear regression is a linear regression model with a single explanatory variable. That is, it concerns two-dimensional sample points with one independent variable and one dependent variable (conventionally, the x and y coordinates in a Cartesian coordinate system) and finds a linear function (a non-vertical straight line) that, as accurately as possible, predicts the dependent variable values as a function of the independent variables. The adjective simple refers to the fact that the outcome variable is related to a single predictor.

In our case the outcome variable is Load and the predictor variable is Temperature.

Data-Preprocessing:

The first and foremost step taken is to delete the 24 observations for the 29th February timestamp in the leap year 2004. This is done as these values for the date 29th Feb are seen only once throughout the four years. Also, we have dropped the hours column during the entire forecasting.

In order to check if the data has any seasonality, trend or just in simple terms is following the same pattern we have plotted a line graph in Excel for years 2002(Series-1) to 2005(Series-4).



Hence, we can clearly see that the entire years load majorly (approximately) repeats itself. **So, we have taken average of temperature column and load for the 4 years and reduced the observation count from 35040 to 8760.** These 8760 observations are used for training and testing purpose in the ratio (70:30).

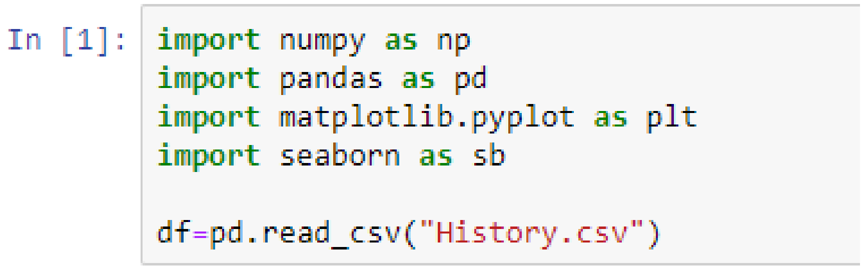
Code and Theory:

NumPy:
It is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Pandas:
In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

Matplotlib:
Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits.

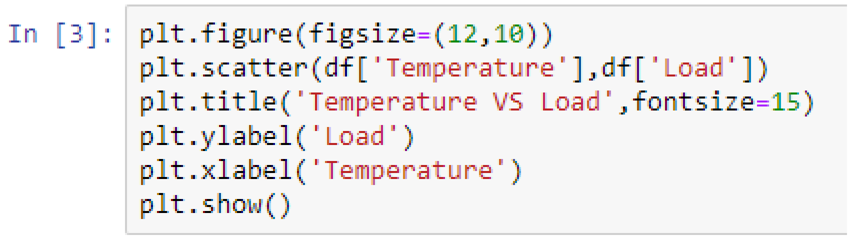
Seaborn:
Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.



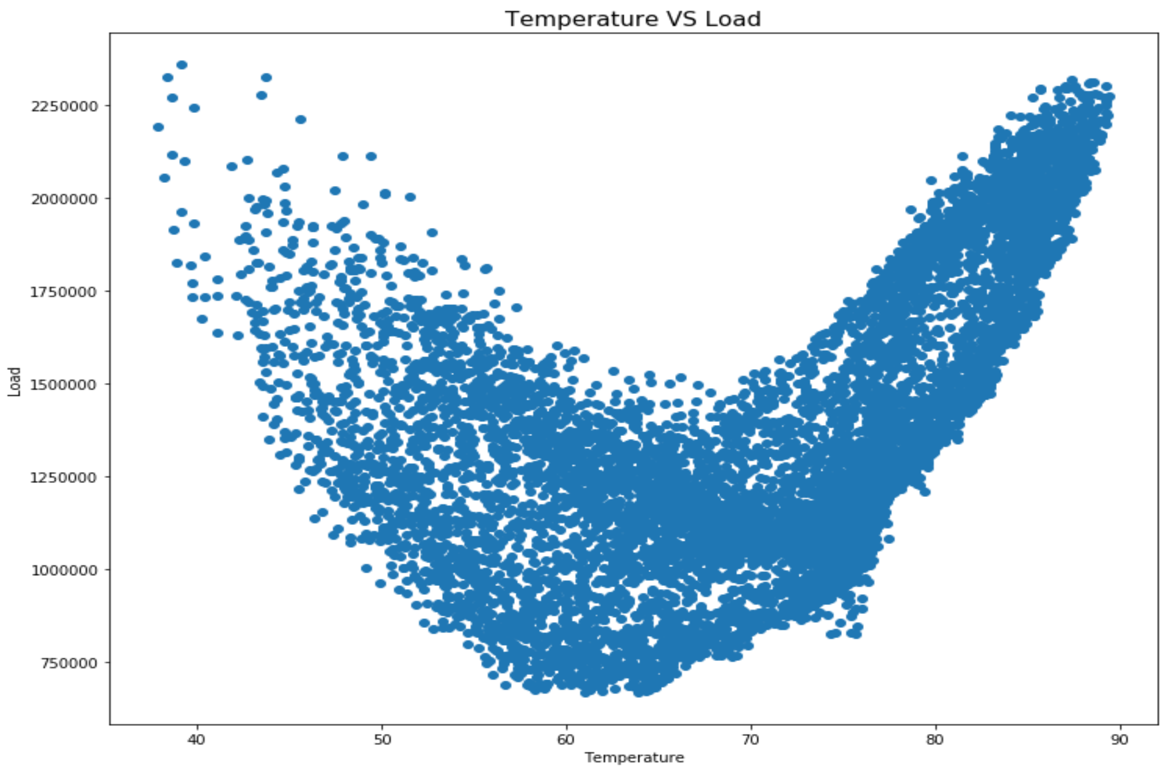
```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sb

        df=pd.read_csv("History.csv")
```

We have assigned the variable to the dataset file as ‘df’.

Plots for understanding & Preliminary Observation:



```
In [3]: plt.figure(figsize=(12,10))
        plt.scatter(df['Temperature'],df['Load'])
        plt.title('Temperature VS Load',fontsize=15)
        plt.ylabel('Load')
        plt.xlabel('Temperature')
        plt.show()
```



We can see majority of the load data is concentrated mostly in the temperature range of 65 to 90.

Applying Linear Regression Model

We set the data for training & testing purpose. We have selected all the rows and only Temperature & Load.

```
In [4]: X=df.iloc[:,1:3].values
        Y=df.iloc[:,3].values
```

```
In [18]: #Scikit-Learn is a free machine Learning Library for Python.
         #It features various algorithmsLike support vector machine, random forests, and k-neighbours.
         #Also supports Python numerical and scientific Libraries Like NumPy and SciPy.
```

```
In [5]: from sklearn.model_selection import train_test_split
        xtrain,xtest,ytrain,ytest=train_test_split(X,Y,test_size=1/3,random_state=0)
```

We recall Linear Regreesion function from sklearn library.

```
In [6]: from sklearn.linear_model import LinearRegression
        reg=LinearRegression()
        reg.fit(xtrain,ytrain)

Out[6]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

We give the command for testing and predicting.

```
In [7]: predicted=reg.predict(xtest)
```

```
In [8]: predicted[:10]

Out[8]: array([1193548.2055955 , 1016321.65744966, 1375776.68149244,
               1432149.4633588 , 1590573.41697048, 1483899.44806514,
               1562429.33800892, 1400283.00642703, 1591656.62898614,
               1088334.30052165])
```

```
In [9]: ytest[:10]

Out[9]: array([ 940522.75,  721025.25, 1383331.  , 1138872.  , 1718659.75,
               1133032.  , 1974775.25, 1364149.  ,  992579.25, 1464552.  ])
```

```
In [10]: xtrain

Out[10]: array([[ 7.  , 41.84],
                [15.  , 77.18],
                [ 8.  , 62.57],
                ...,
                [12.  , 85.97],
                [ 1.  , 72.43],
                [21.  , 70.16]])
```

Making the Forecast

We load the 'Fcst.csv' file where the outcome variable & predictor variable is there.

```
In [11]: df1=pd.read_csv("Fcst.csv")
```

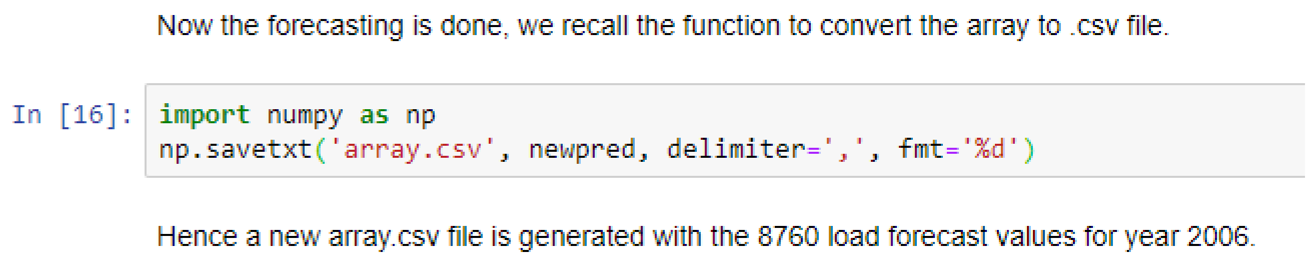
```
In [12]: dfnew=df1.iloc[:,1:3]
```

We set the variable 'newpred' for forecasting purpose.

```
In [13]: newpred=reg.predict(dfnew)
```

```
In [14]: newpred=newpred.astype(int)
```

```
In [15]: newpred

Out[15]: array([ 963098,  984605, 1006581, ..., 1579035, 1597256, 1615478])
```

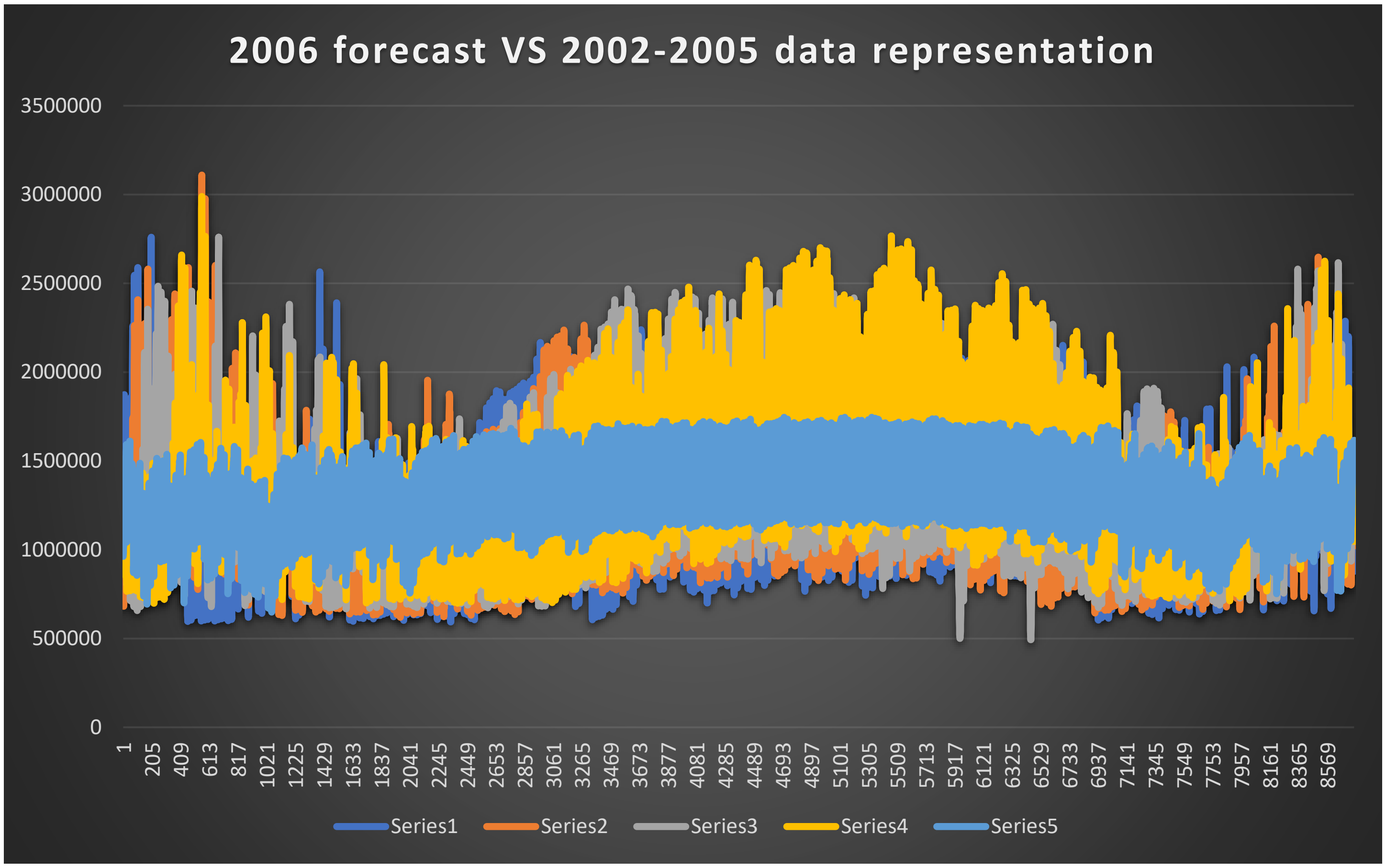


Now the forecasting is done, we recall the function to convert the array to .csv file.

```
In [16]: import numpy as np
         np.savetxt('array.csv', newpred, delimiter=',', fmt='%d')
```

Hence a new array.csv file is generated with the 8760 load forecast values for year 2006.

For visual purpose we will plot line graph of the 2002 to 2005(Series 1-4) load along with 2006(Series 5) forecast.



__Conclusion:__

Hence, we have used the simple linear regression model for forecasted year ahead hourly based load. Our model is working.

The Mean Average Percentage Error is **17.98%**