

DDA LAB Report -5

Task : Time Series Forecasting

Data : Air Passengers Data set and Melbourne Data set

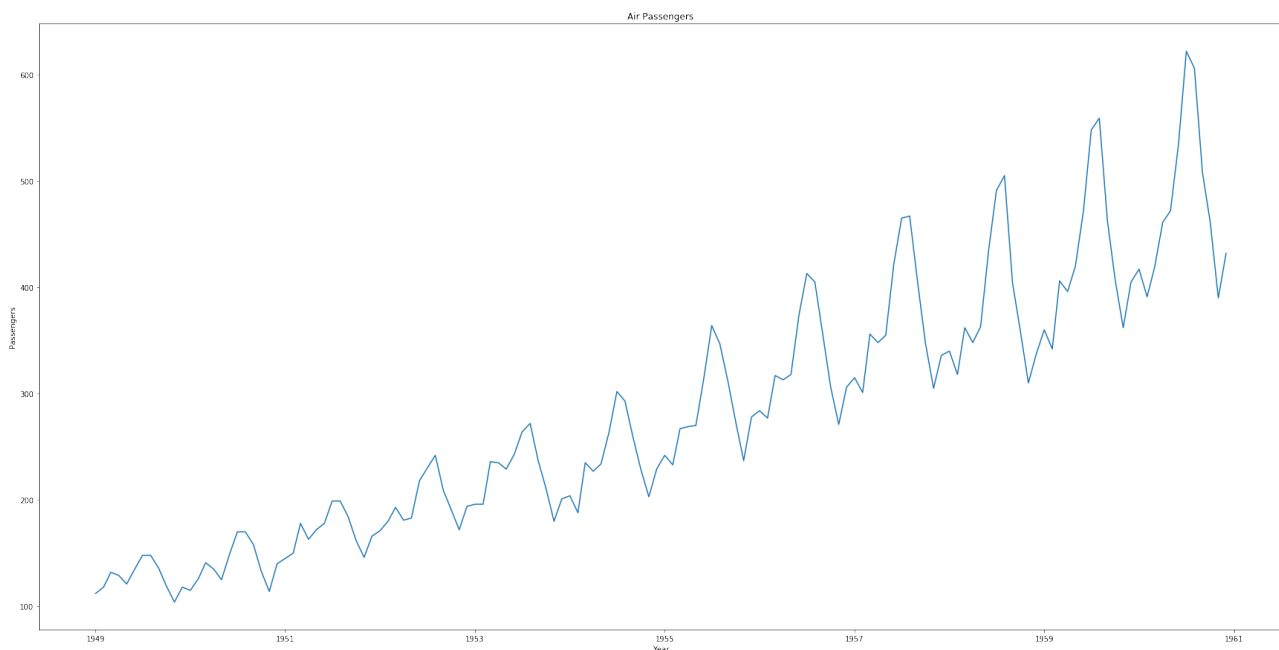
Exercise 1A: Stationary and Arima

A time series is said to be stationary if doesn't have any seasonality or trend. We can check this in many ways. One of the ways to perform DF test. If we get the p-value less than 0.5 then the series is stationary else it is not.

And calculating the mean after dividing the data set. If there is a big change in the means then it is not stationary else it is stationary.

Basing these two assumptions the task is solved.

First the Air Passengers Data set is loaded. The time series is plotted. The plot obtained is



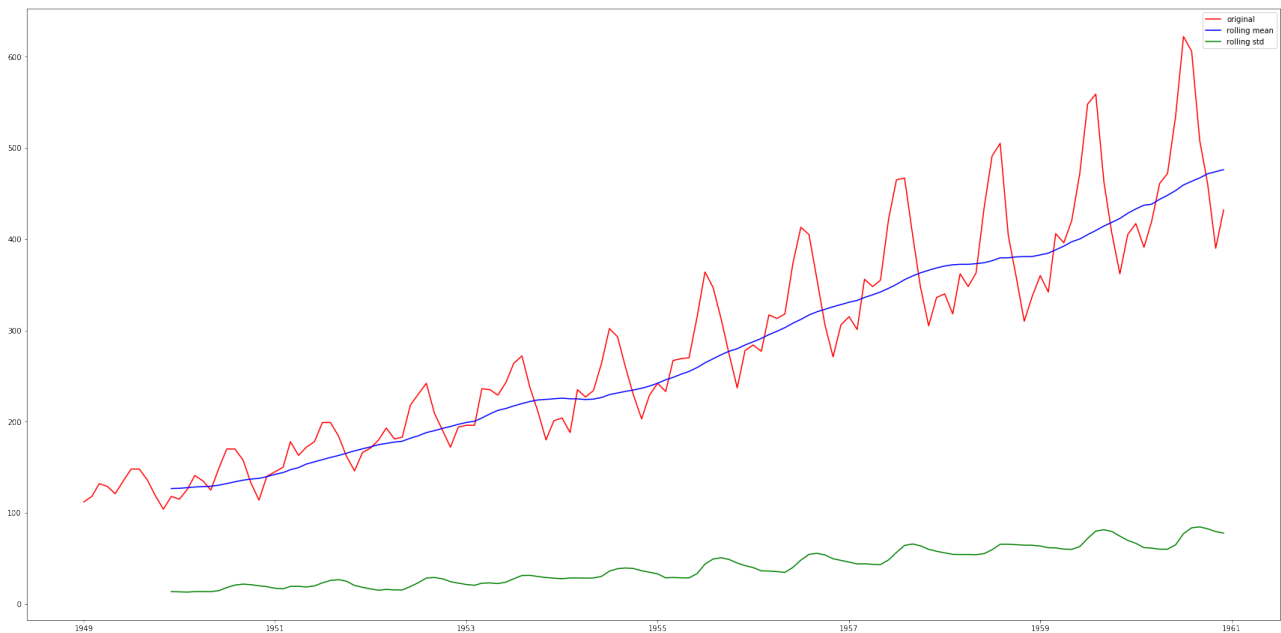
The mean is calculated on 2 halves and then the DF test is performed the result is

```

Mean of first part of data: 182.902777778
Mean of second part of data 377.694444444
Standard deviation of first part of data: 47.371803534
Standard deviation of second part of data 85.8368347002
Test Statistic          0.815369
p-value                0.991880
#Lags Used               13.000000
Number of Observations Used 130.000000
Critical Value (1%)      -3.481682
Critical Value (5%)      -2.884042
Critical Value (10%)     -2.578770

```

The plot is made with the rolling mean and the standard deviation. The plot is obtained like this

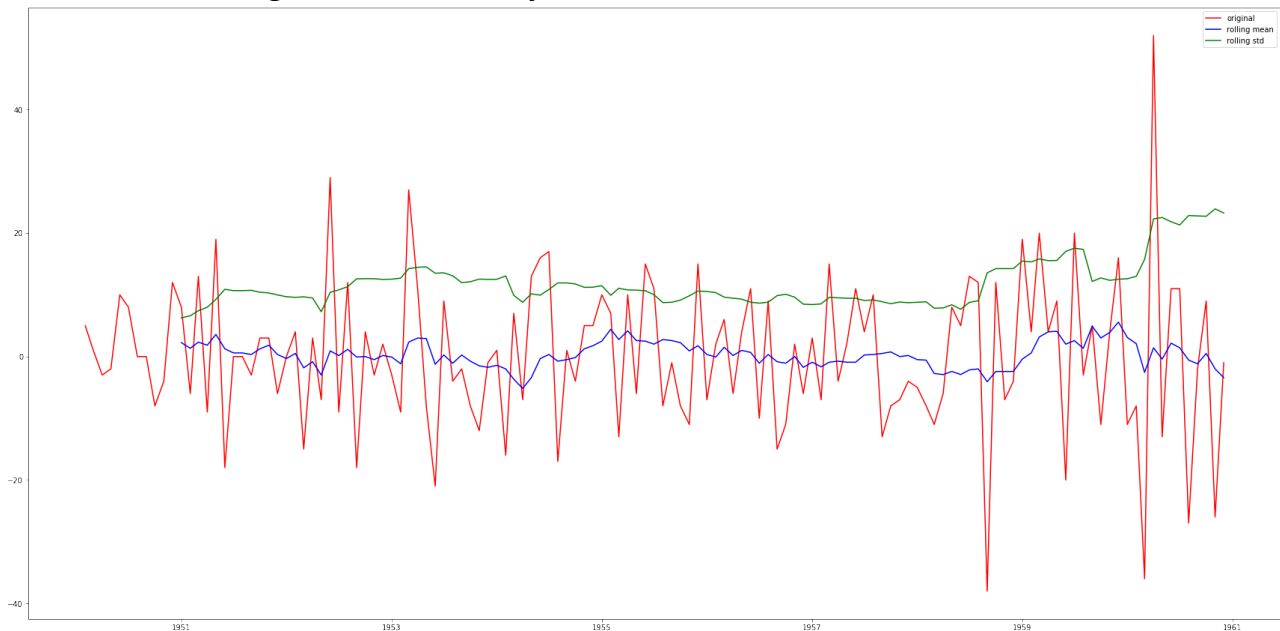


If we observe the plot and the test results we can say that the series is not stationary. To make it stationary we have to remove the trend and seasonality. After making the series stationary the DF test results are

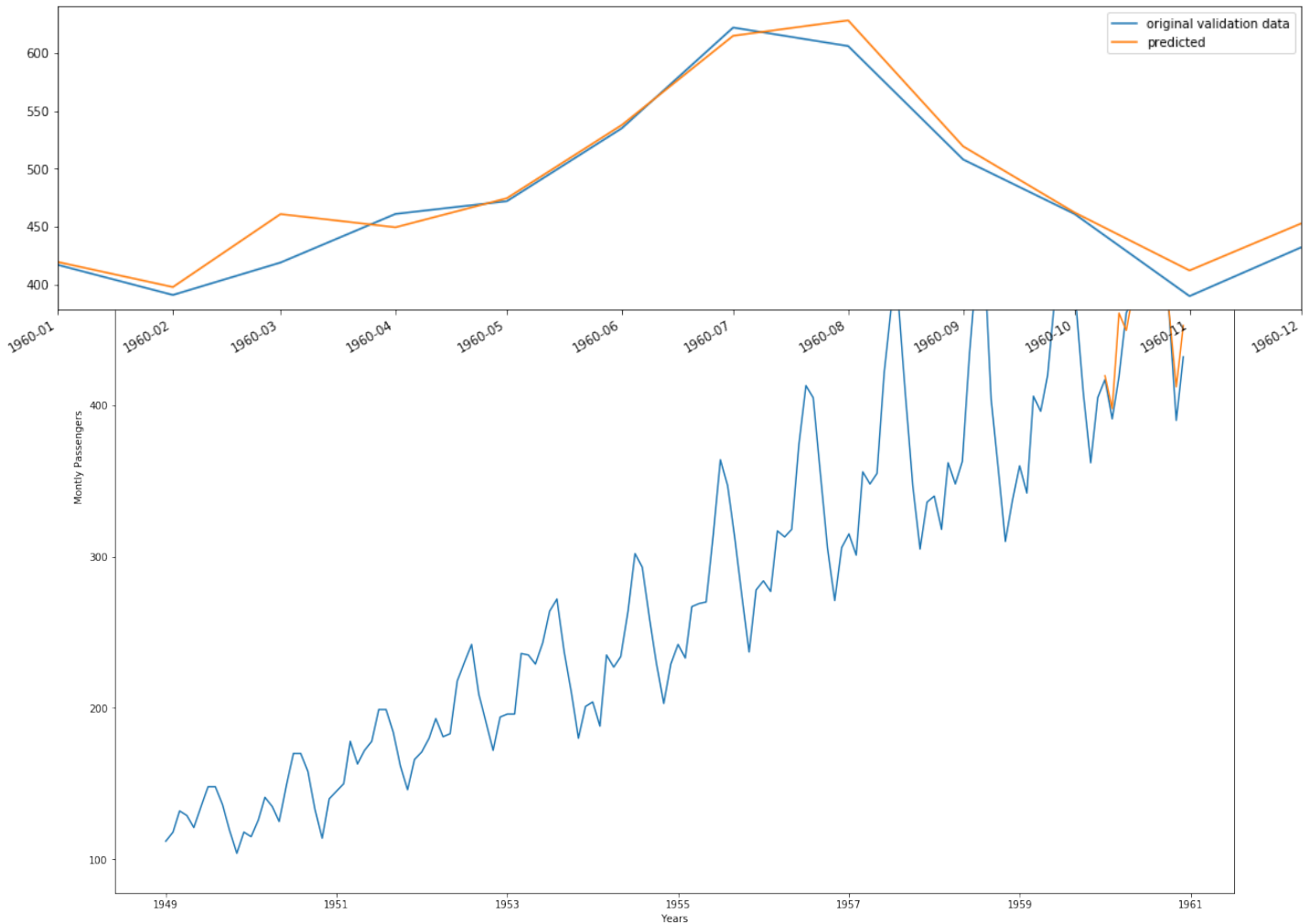
```

Mean of first part of data: 0.738461538462
Mean of second part of data -0.363636363636
Standard deviation of first part of data: 10.4267067197
Standard deviation of second part of data 13.895318989
Test Statistic -1.559562e+01
p-value 1.856512e-28
#Lags Used 0.000000e+00
Number of Observations Used 1.300000e+02
Critical Value (1%) -3.481682e+00
Critical Value (5%) -2.884042e+00
Critical Value (10%) -2.578770e+00
    
```

If we observe the P-value it is below 0.5. So we can say that the series is stationary now. The plot obtained after making the series stationary is



Since the Air Passenger has seasonality **SARIMAX** is used. For the parameters a grid search is made and the best parameters are used for training the model and then the forecast is made and the rmse is calculated. A graph is plotted for the forecasted data and the original data.

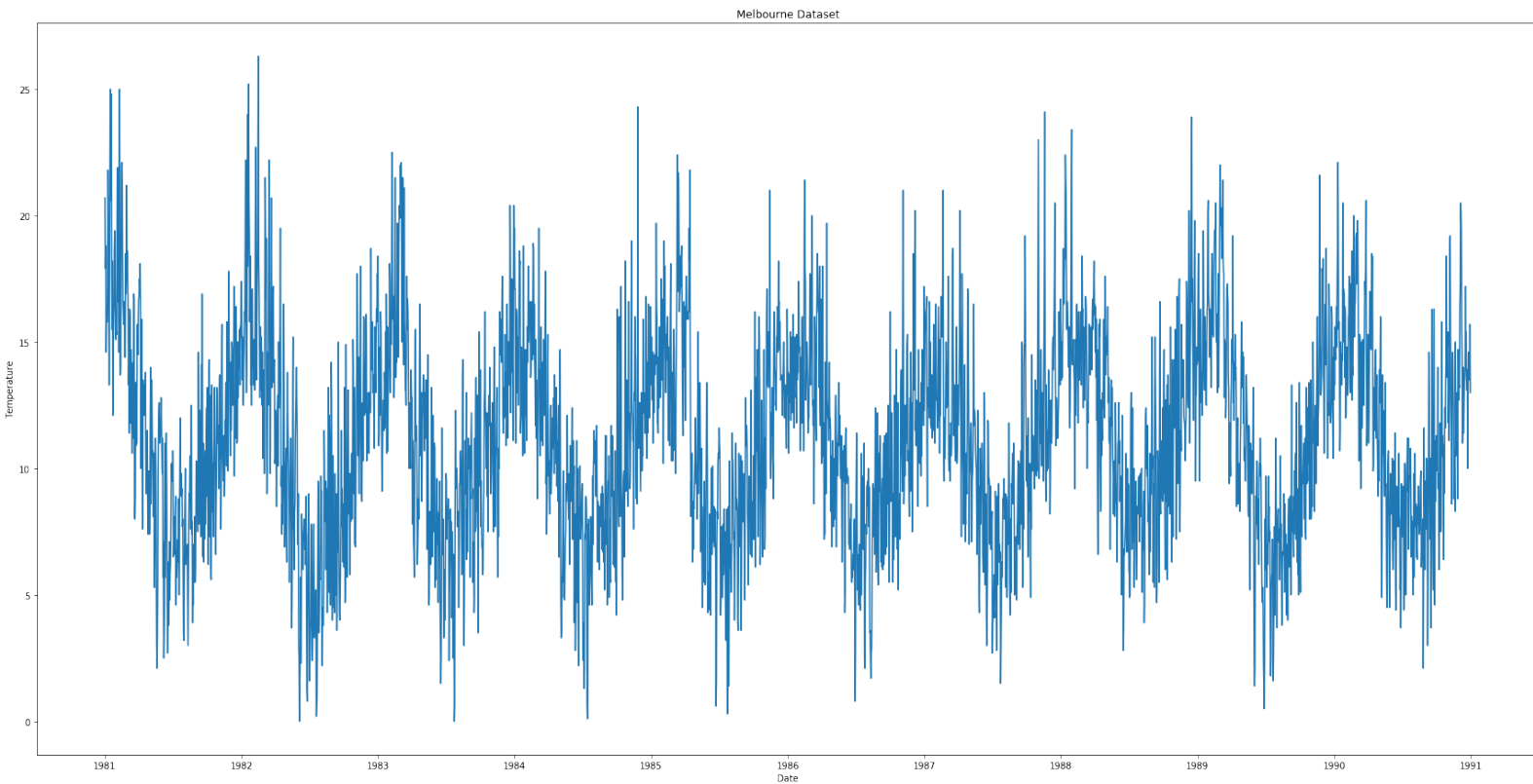


The RMSE is calculated for forecasts and it is 17.187245320475341.

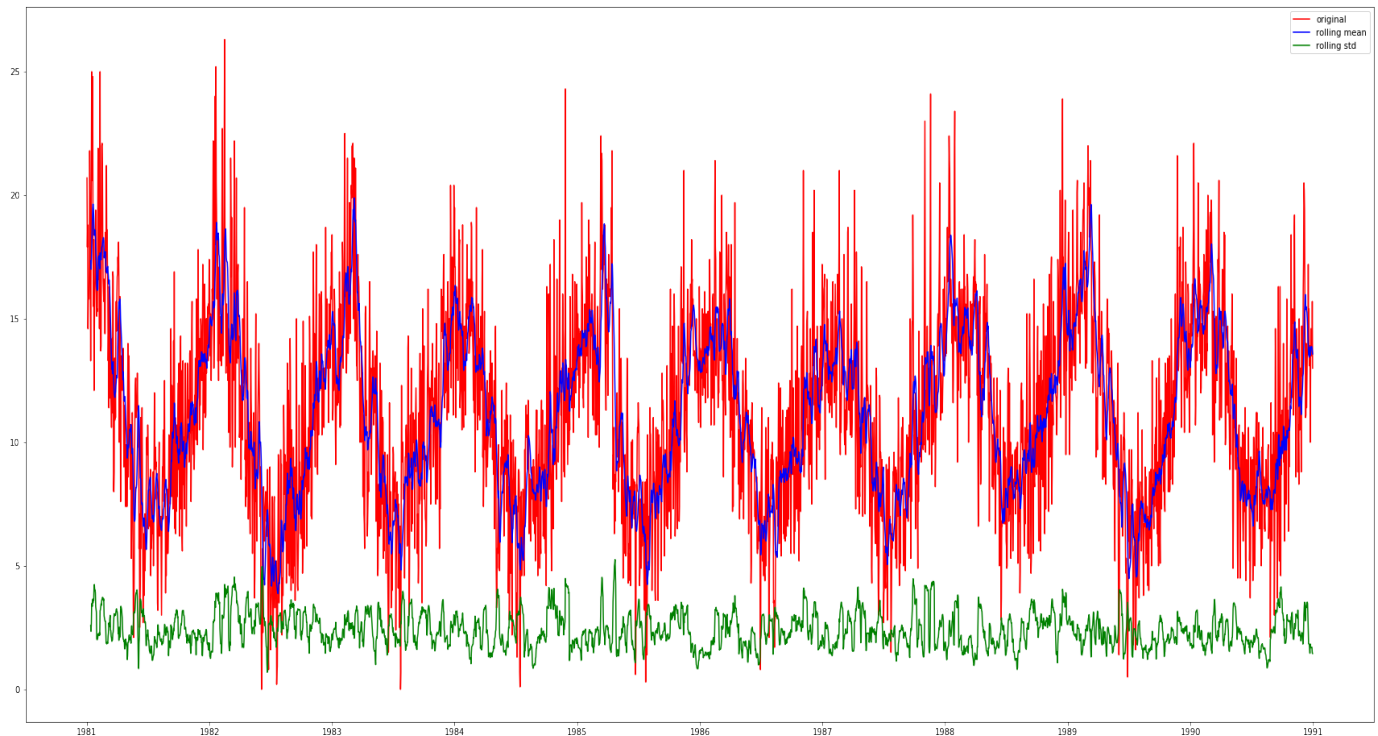
Mohana Nyamahahalli Venkatesha 276833

Melbourne Data set

The other data set which is given for the task is melbourne data set. First the data set is plotted.



The plot after plotting the mean and the standard deviation is



The results for the DF test are given below

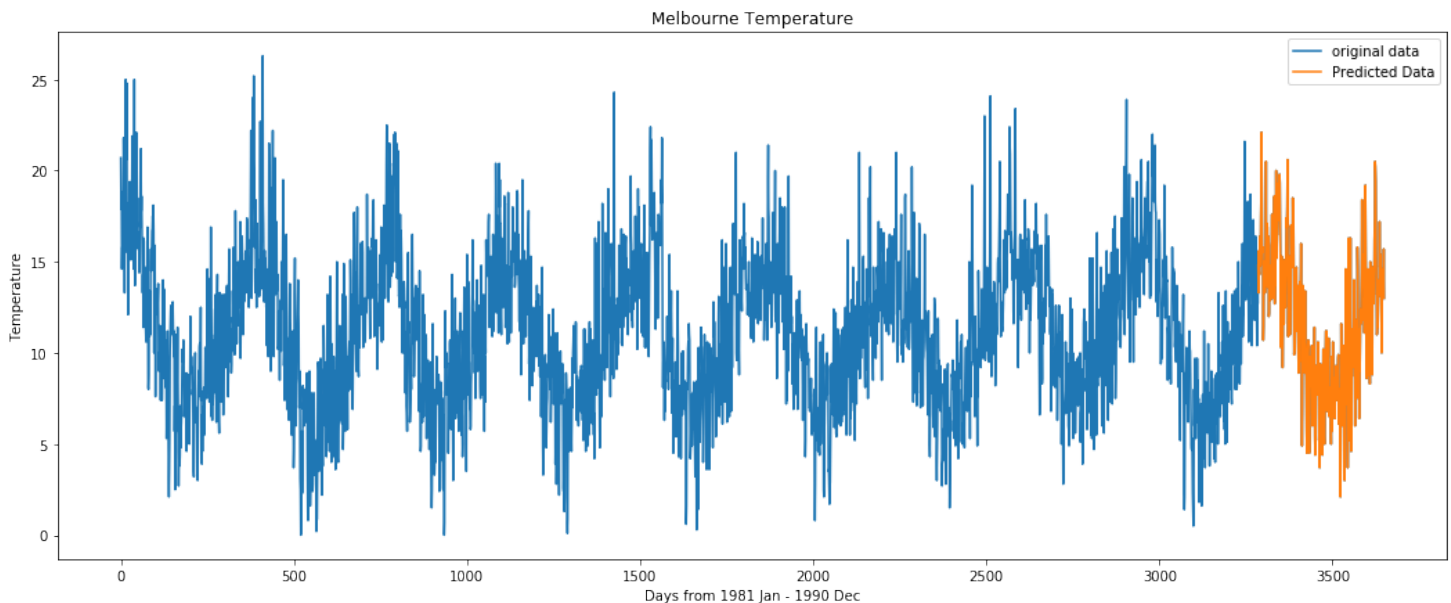
```

Mean of first part of data: 11.0435068493
Mean of second part of data 11.312
Standard deviation of first part of data: 4.26155194293
Standard deviation of second part of data 3.86700884016
Test Statistic -4.444805
p-value 0.000247
#Lags Used 20.000000
Number of Observations Used 3629.000000
Critical Value (1%) -3.432153
Critical Value (5%) -2.862337
Critical Value (10%) -2.567194
    
```

If we observe the p-value it is less than 0.5 and the difference in the means are less when compared. So the given time series is stationary and we can apply ARIMA for forecasting.

The parameters for the ARIMA are found by grid search and then the parameters are used for training the model and then the model is used to make the forecasts and then the RMSE is calculated and the plots are made for the forecasted data.

The RMSE obtained is 3.69205616048



Task 2: Tensor Flow

Exercise : Logistic Regression on the Olivetti faces data set

1. The **olivetti faces** data set is a data set that contains a set of face images taken between April 1992 and April 1994 at AT&T Laboratories Cambridge.
2. The data set has ten different images of each of 40 distinct subjects. The image is quantized to 256 grey levels and stored as unsigned 8 bit integers, the loader will convert these to floating point values on the interval [0,1].
3. The target has 0-39 indicating identity of person.
4. The data set consists of 400 rows with 4096 features.
5. The 3D data set consists of 400*64*64 is converted into 400*4096 and the targets are encoded by using Label binarizer which is provided by sklearn package.
6. The data is split into train and test with 90 and 10 percentage.
7. The train data is used for training the model.
8. By using the tensor flow place holders the weights and the biases are defined . Then cross entropy is used as the cost function for the model and the accuracy is also defined for the model. All these are done by the tensor flow variables.
9. The tensor board variables is defined with the tensor.summary.scalar. And the graphs as well.
10. When the session is initiated and till it finishes the following process occurs the training data is divided into batches of size 100 and fed to the tensorflow model and the logistic regression is implemented on the batch.
11. The cost is calculated for each epoch and the accuracy for the training set is calculated at the end. The cost is plotted and at the end the test accuracy is also calculated.
12. Once the program is run the tensor board is initiated and the graphs are observed.

```

Anaconda Prompt - tensorflow --logdir H:/tmp/tensorflow_logs/example2

(base) H:\DDA\DDA_LAB\EX5>python ex2.py
H:\lab\lib\site-packages\h5py\_init_.py:36: FutureWarning: Conversion of the second argument of 'issubdtype' from `float` to `np.floating` is deprecated. In future, it will
be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Epoch: 0 cost = 3.6771202087402344
Epoch: 1 cost = 3.6496806144714355
Epoch: 2 cost = 3.6267143090565996
Epoch: 3 cost = 3.604364474614461
Epoch: 4 cost = 3.5822153091430664
Epoch: 5 cost = 3.5602136452992754
Epoch: 6 cost = 3.5383502642313642
Epoch: 7 cost = 3.516623814900716
Epoch: 8 cost = 3.4950332641601562
Epoch: 9 cost = 3.4735772609710693
Epoch: 10 cost = 3.4522554874420166
Epoch: 11 cost = 3.431066830952962
Epoch: 12 cost = 3.410010814666748
Epoch: 13 cost = 3.389086564381917
Epoch: 14 cost = 3.3682934443155927
Epoch: 15 cost = 3.3476304213205967
Epoch: 16 cost = 3.3270970185597735
Epoch: 17 cost = 3.306692679723104
Epoch: 18 cost = 3.2864163716634116
Epoch: 19 cost = 3.2662679354349766
Epoch: 20 cost = 3.2462461789449057
Epoch: 21 cost = 3.2263504664103184
Epoch: 22 cost = 3.2065804799397792
Epoch: 23 cost = 3.1869354248046875
Epoch: 24 cost = 3.1674150625864668
Epoch: 25 cost = 3.1480179627736415
Epoch: 26 cost = 3.1287444432576494
Epoch: 27 cost = 3.109593311945597
Epoch: 28 cost = 3.0905641714731846
Epoch: 29 cost = 3.071656147638957
Epoch: 30 cost = 3.0528695583343506
Epoch: 31 cost = 3.0342028935750327
Epoch: 32 cost = 3.0156560738881426
Epoch: 33 cost = 2.9972287019093833
Epoch: 34 cost = 2.9789196650187177
Epoch: 35 cost = 2.960728963216146

```

```

Anaconda Prompt - tensorflow --logdir H:/tmp/tensorflow_logs/example2

Epoch: 31 cost = 3.0342028935750327
Epoch: 32 cost = 3.0156560738881426
Epoch: 33 cost = 2.9972287019093833
Epoch: 34 cost = 2.9789196650187177
Epoch: 35 cost = 2.960728963216146
Epoch: 36 cost = 2.9426561196645102
Epoch: 37 cost = 2.9247001012166343
Epoch: 38 cost = 2.9068611462910967
Epoch: 39 cost = 2.889179833221436
Epoch: 40 cost = 2.8715306917826338
Epoch: 41 cost = 2.85403839747111
Epoch: 42 cost = 2.8366610209147134
Epoch: 43 cost = 2.8193975289662676
Epoch: 44 cost = 2.802248001098633
Epoch: 45 cost = 2.7852116425832114
Epoch: 46 cost = 2.768287976582845
Epoch: 47 cost = 2.751476764678955
Epoch: 48 cost = 2.734777371088664
Epoch: 49 cost = 2.718189318974813
Epoch: 50 cost = 2.701712131500244
Epoch: 51 cost = 2.68534525235494
Epoch: 52 cost = 2.66908852259318
Epoch: 53 cost = 2.6529413064320884
Epoch: 54 cost = 2.6369031270345054
Epoch: 55 cost = 2.620973507563273
Epoch: 56 cost = 2.605151891708374
Epoch: 57 cost = 2.589438120524089
Epoch: 58 cost = 2.5738311608632407
Epoch: 59 cost = 2.558331410090129
Epoch: 60 cost = 2.5429375171661377
Epoch: 61 cost = 2.5276495615641275
Epoch: 62 cost = 2.5124666690826416
Epoch: 63 cost = 2.4973888397216797
Epoch: 64 cost = 2.482415517171224
Epoch: 65 cost = 2.4675459067026777
Epoch: 66 cost = 2.4527796109517412
Epoch: 67 cost = 2.4381163120269775
Epoch: 68 cost = 2.423555374145508
Epoch: 69 cost = 2.409096558888753
Epoch: 70 cost = 2.3947390715281167
Epoch: 71 cost = 2.3804827531178794

```


Mohana Nyamahahalli Venkatesha 276833

```

Anaconda Prompt - tensorboard --logdir H:/tmp/tensorflow_logs/example2
Epoch: 167 cost = 1.4025463660558064
Epoch: 168 cost = 1.395601789156596
Epoch: 169 cost = 1.3887080351511638
Epoch: 170 cost = 1.3818649053573608
Epoch: 171 cost = 1.3750718832015991
Epoch: 172 cost = 1.3683284918467202
Epoch: 173 cost = 1.361634339284263
Epoch: 174 cost = 1.3549889723459878
Epoch: 175 cost = 1.348392128944397
Epoch: 176 cost = 1.3418432076772053
Epoch: 177 cost = 1.3353420893351238
Epoch: 178 cost = 1.328880983988443
Epoch: 179 cost = 1.3224807977676392
Epoch: 180 cost = 1.3161199857593586
Epoch: 181 cost = 1.3098052740007046
Epoch: 182 cost = 1.303536057472229
Epoch: 183 cost = 1.2973122994105022
Epoch: 184 cost = 1.2911332845687866
Epoch: 185 cost = 1.2849987347920737
Epoch: 186 cost = 1.2789083321889243
Epoch: 187 cost = 1.2728615204493205
Epoch: 188 cost = 1.2668582598368328
Epoch: 189 cost = 1.2608978350957234
Epoch: 190 cost = 1.254980206489563
Epoch: 191 cost = 1.2491046984990437
Epoch: 192 cost = 1.2432710727055867
Epoch: 193 cost = 1.2374790509541829
Epoch: 194 cost = 1.231728156407674
Epoch: 195 cost = 1.2260181903839111
Epoch: 196 cost = 1.2203486363093057
Epoch: 197 cost = 1.2147192160288491
Epoch: 198 cost = 1.2091295719146729
Epoch: 199 cost = 1.2035794655481973
Training accuracy 98.8888597488403
Test accuracy 89.9999761581421
Run the command line:
--> tensorboard --logdir H:/tmp/tensorflow_logs/example2

(base) H:\DDA\DDA_LAB\EX5>tensorboard --logdir H:/tmp/tensorflow_logs/example2
H:\lab\lib\site-packages\h5py\_init_.py:36: FutureWarning: Conversion of the second argument of 'issubdtype' from `float` to `np.floating` is deprecated. In future, it will
be treated as `np.float64 == np.dtype(float).type`.

```

Learning Rate=0.01

Training Epochs=200

Batch size=100

The training accuracy obtained is 98.8 %

The test accuracy obtained is 89.9 %

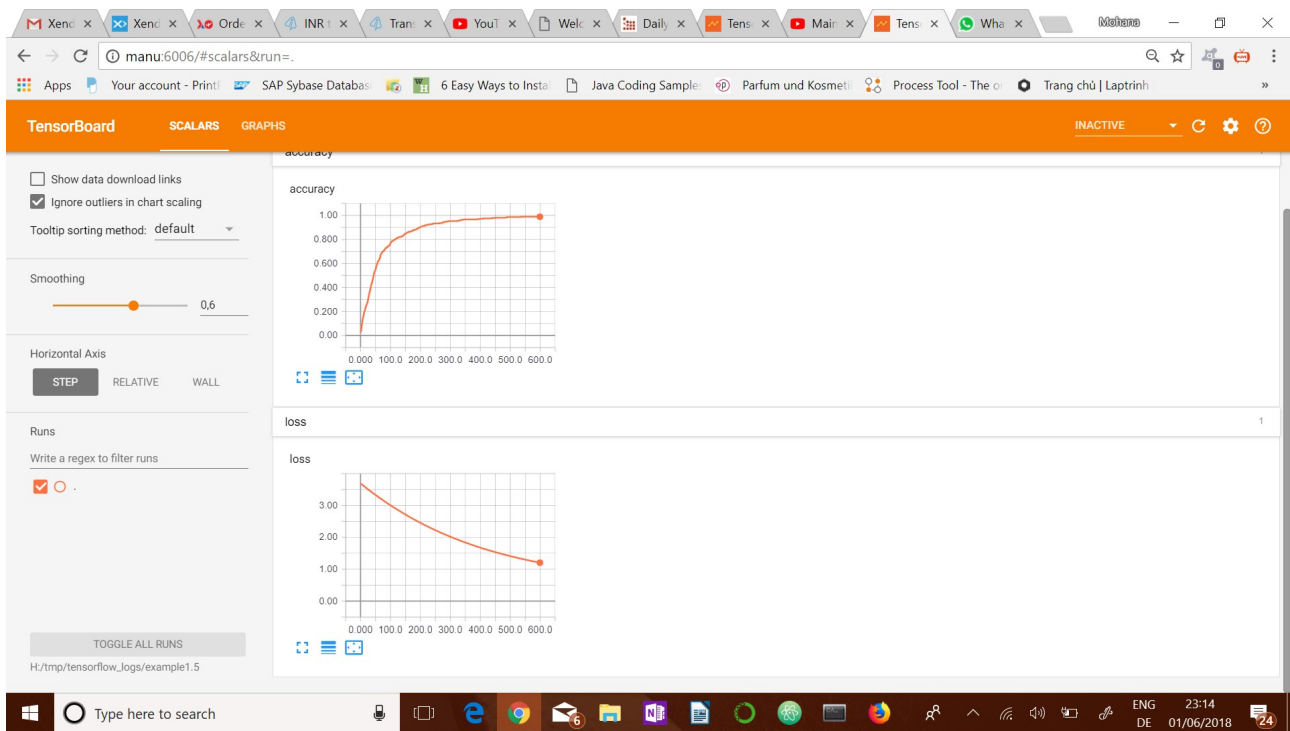
After deploying the tensor board and deploying it we can see it by

Run the command line:

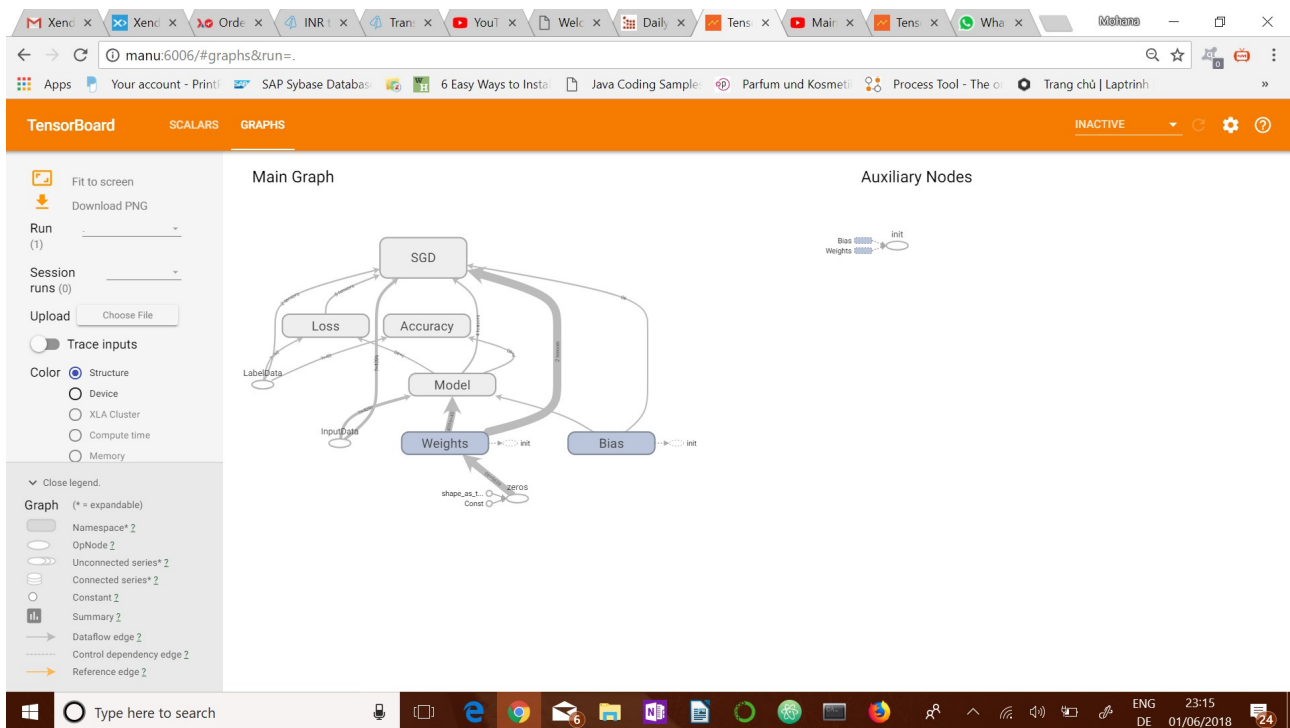
\$tensorboard --logdir= /tmp/tensorflow_logs/example1.6/

Then open <http://sunny:6006> into your web browser

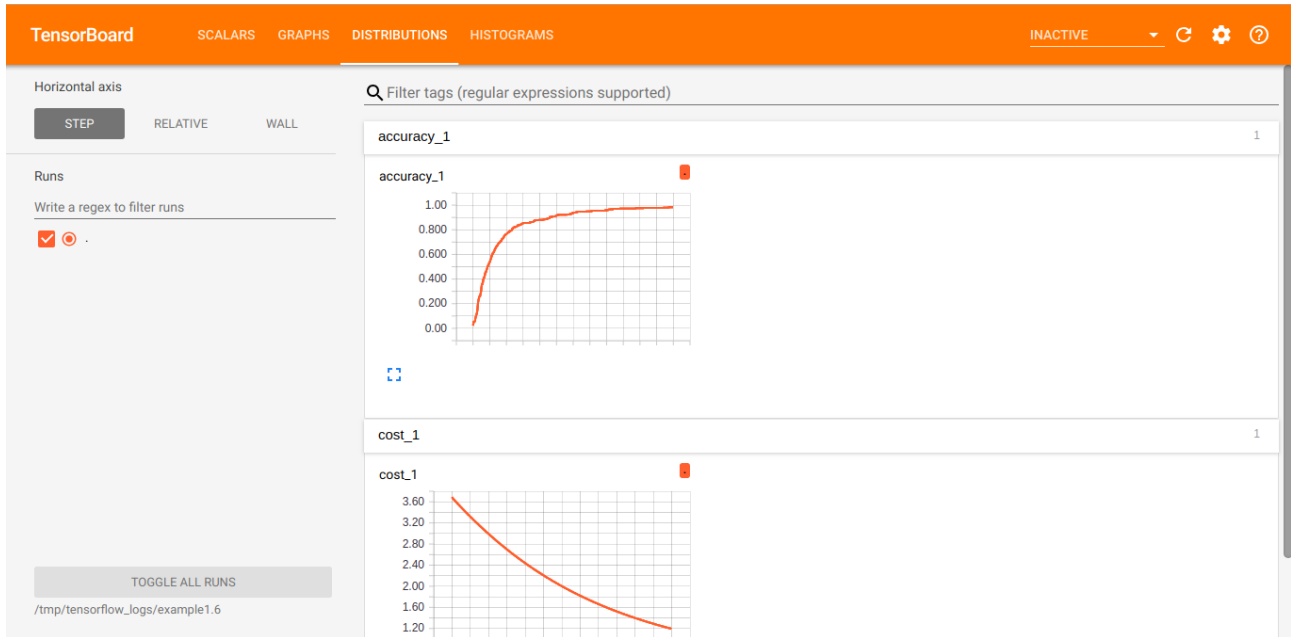
Scalars



Graph



Distributions



Histogram

