|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |
| --- |
| Time Series Analysis |
| Report |

|  |  |  |
| --- | --- | --- |
| By |  |  |
| Safaa |  |  |
| 331215795 |  |  |

|  |  |
| --- | --- |
| Table of Contents | |
|  | Page No. |
| Introduction of Time Series Analysis …...……………….. | 3 |
| Toturial ……………………………………………..………. | 3 |
| Code of Holt Winter ………….………………………….. | 8 |
| Code of Holt Winter After Changes ……………………. | 13 |

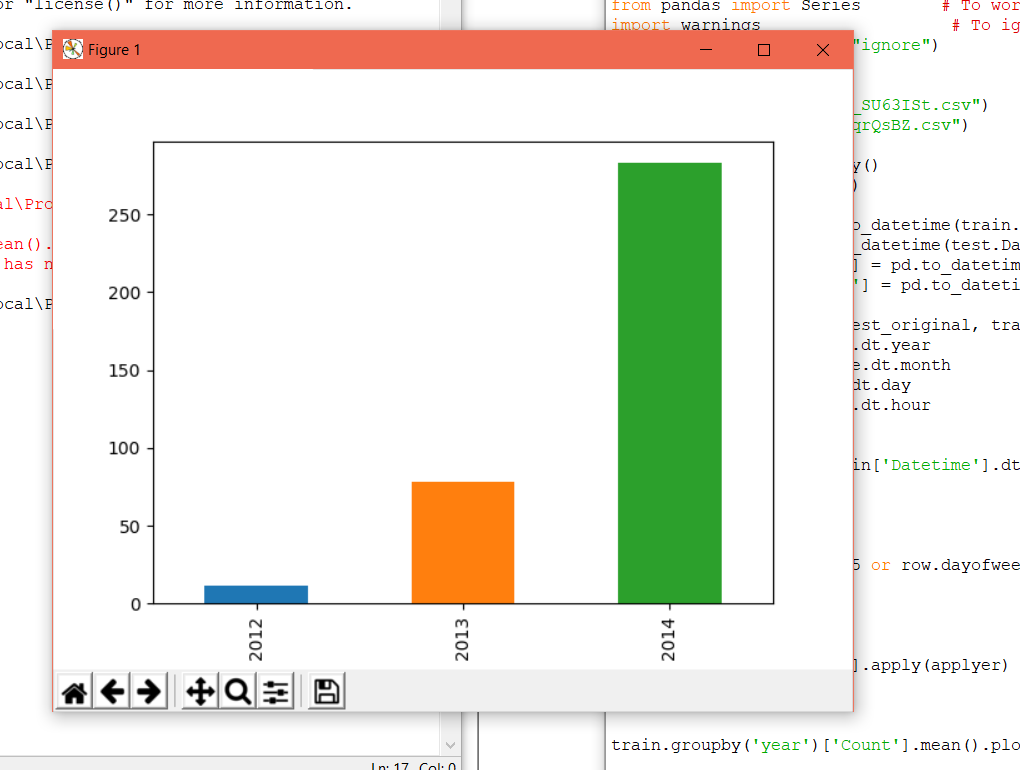
**Introduction of Time Series Analysis**

Time Series is generally data which is collected over time and is dependent on it. A series of data points collected in time order is known as a time series. Most of business houses work on time series data to analyze sales number for the next year, website traffic, count of traffic, number of calls received, etc. Data of a time series can be used for forecasting. Not every data collected with respect to time represents a time series.

**Toturial**

* Hypothesis generation.

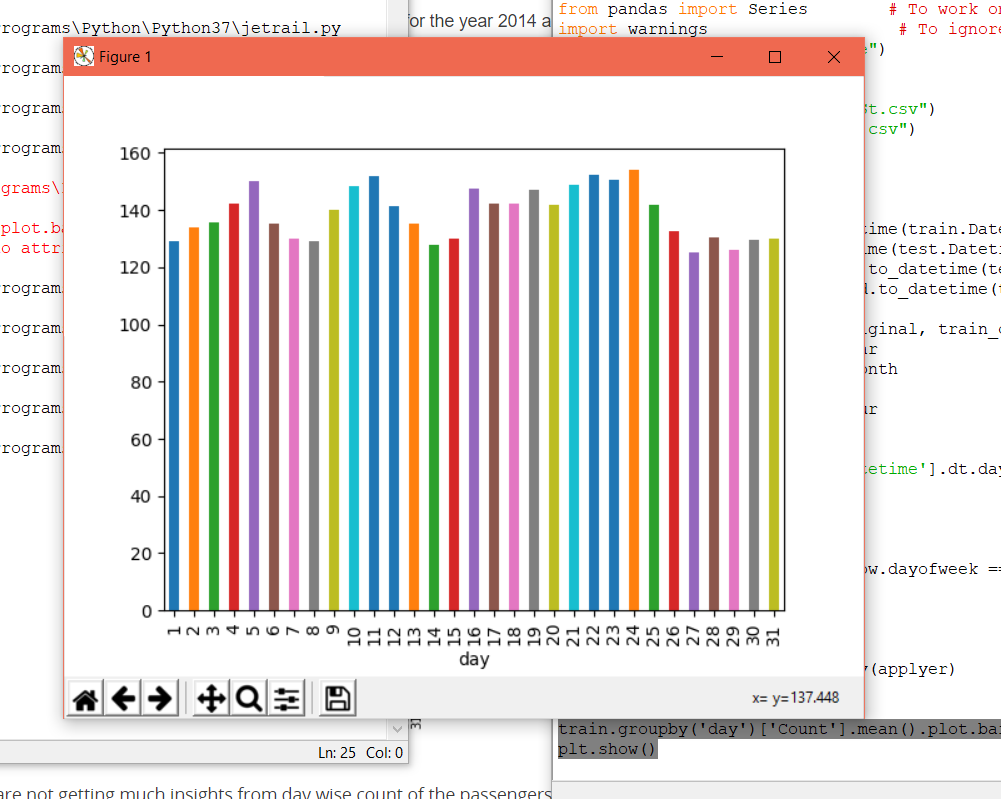
1. Traffic will increase as the years pass by.



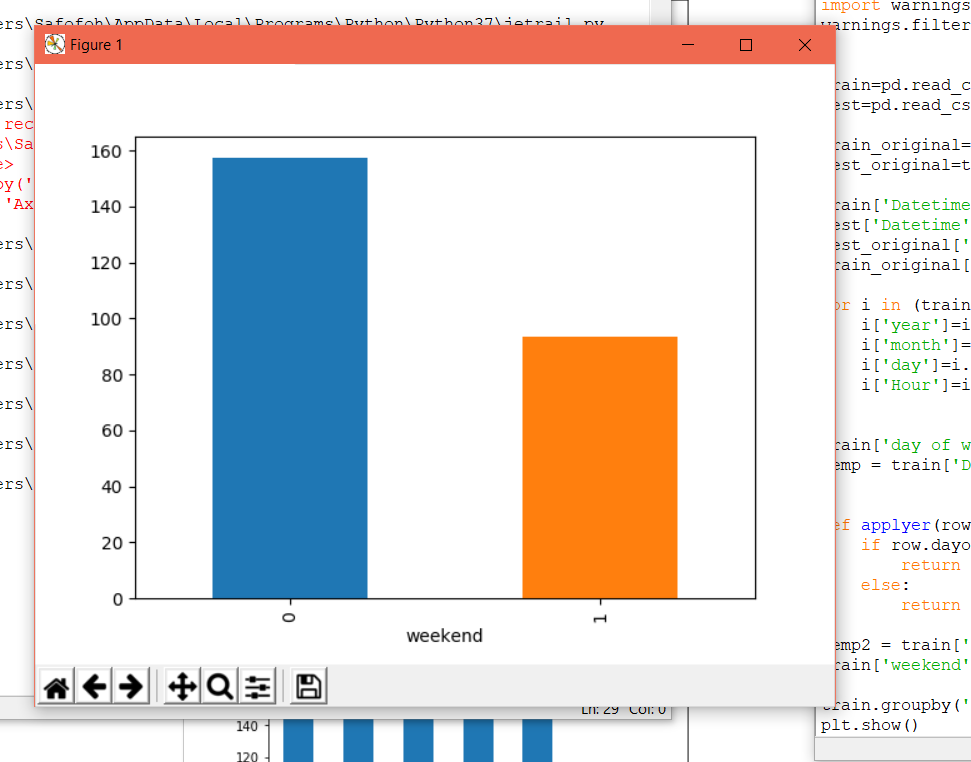
1. Traffic will increase from May to October.



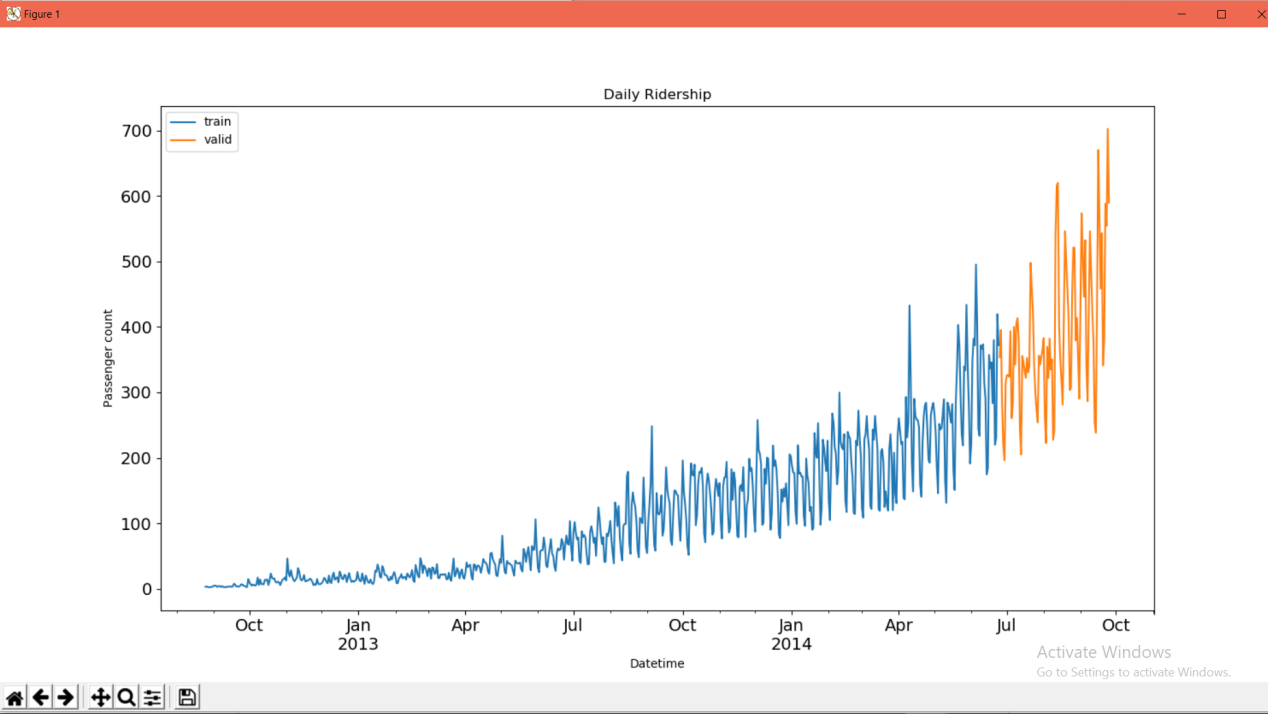
1. Traffic will be more during peak hours.



1. Traffic will be more on weekdays.

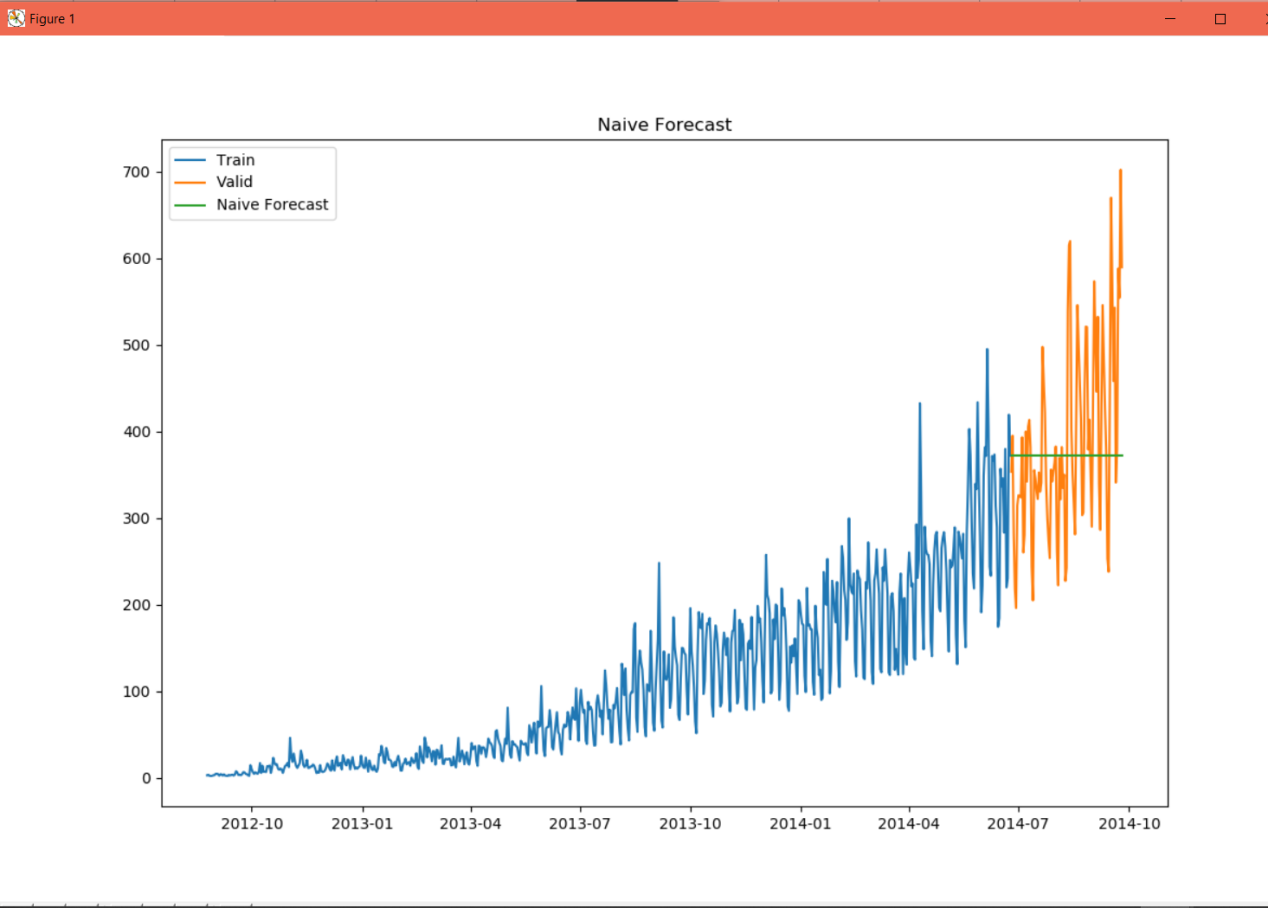


* Spiltting the data into training and validation set. Last 3 months as the validation data and the rest for training data.

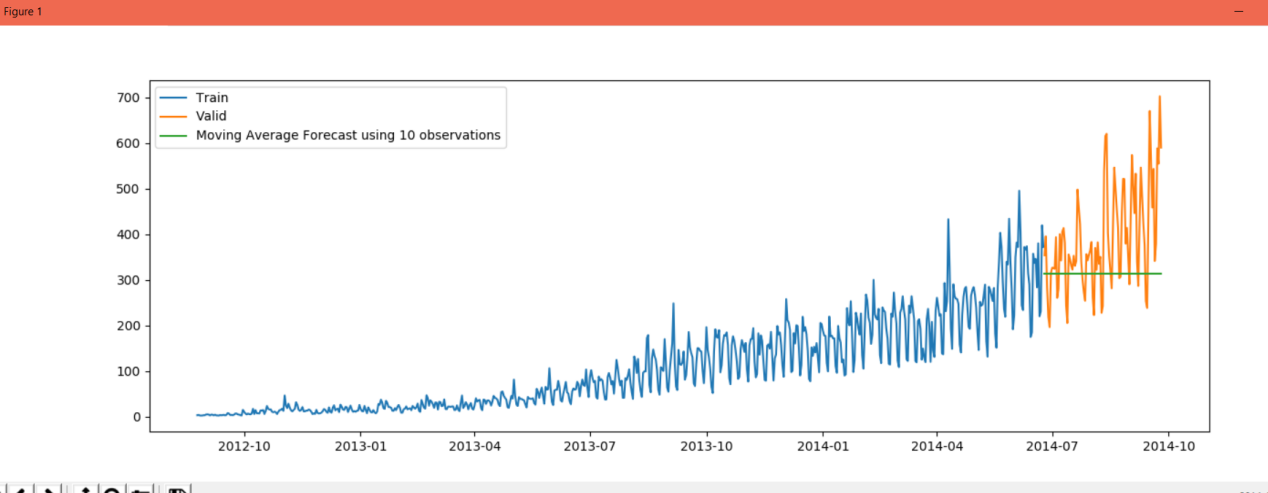


* Modeling Techniques.

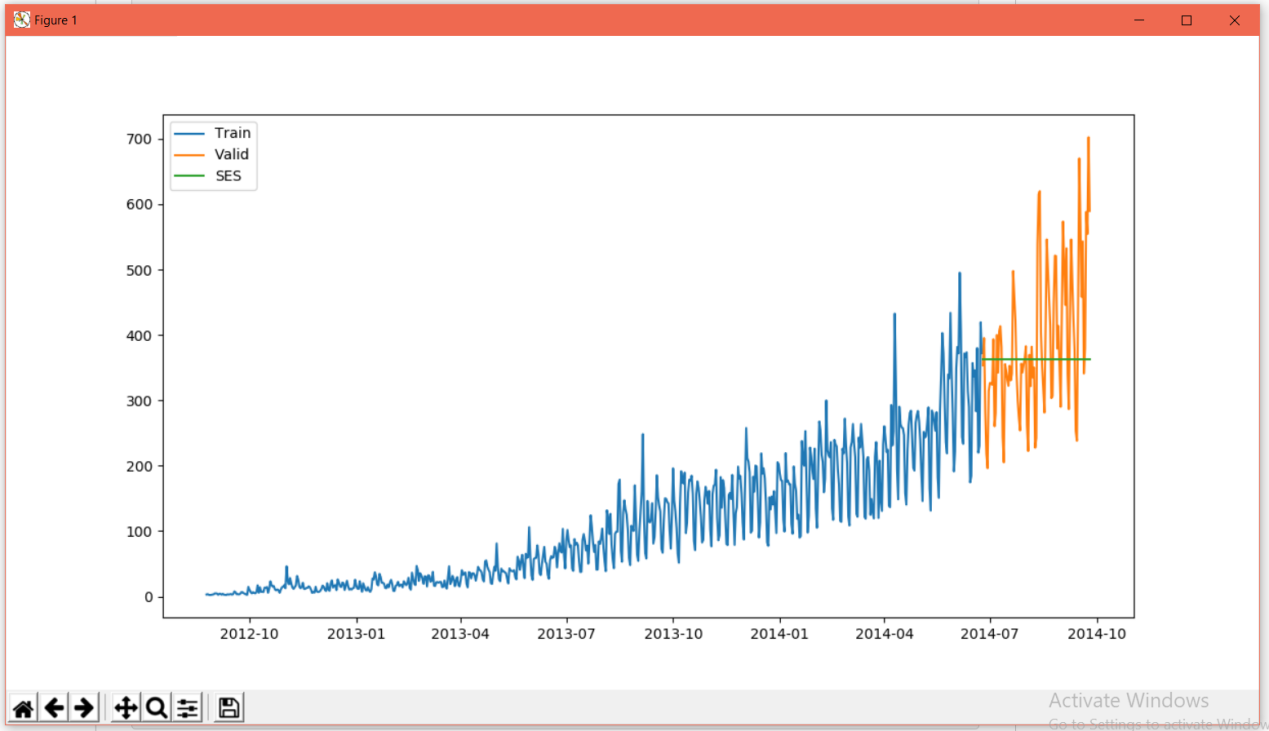
1. Naive Approach.



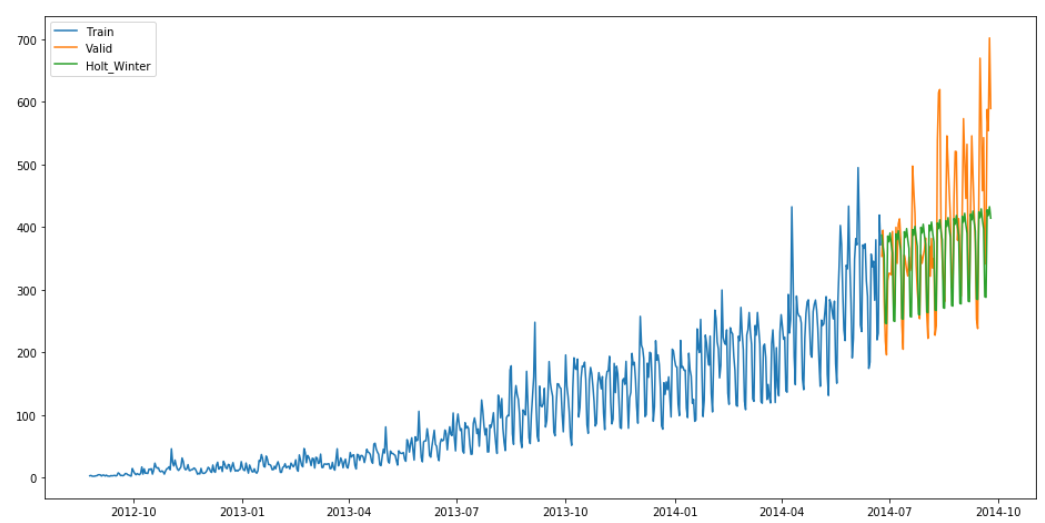
1. Moving Average.



1. Simple Exponential Smoothing.



1. Holt Winter Model.



**Code of Holt Winter**

import pandas as pd

import numpy as np # For mathematical calculations

import matplotlib.pyplot as plt # For plotting graphs

from datetime import datetime # To access datetime

from pandas import Series # To work on series %matplotlib inline

from sklearn.metrics import mean\_squared\_error

from math import sqrt

from statsmodels.tsa.holtwinters import ExponentialSmoothing

import warnings # To ignore the warnings

warnings.filterwarnings("ignore")

train=pd.read\_csv("Train\_SU63ISt.csv")

test=pd.read\_csv("Test\_0qrQsBZ.csv")

train\_original=train.copy()

test\_original=test.copy()

train['Datetime'] = pd.to\_datetime(train.Datetime,format='%d-%m-%Y %H:%M')

test['Datetime'] = pd.to\_datetime(test.Datetime,format='%d-%m-%Y %H:%M')

test\_original['Datetime'] = pd.to\_datetime(test\_original.Datetime,format='%d-%m-%Y %H:%M')

train\_original['Datetime'] = pd.to\_datetime(train\_original.Datetime,format='%d-%m-%Y %H:%M')

for i in (train, test, test\_original, train\_original):

i['year']=i.Datetime.dt.year

i['month']=i.Datetime.dt.month

i['day']=i.Datetime.dt.day

i['Hour']=i.Datetime.dt.hour

train['day of week']=train['Datetime'].dt.dayofweek

temp = train['Datetime']

def applyer(row):

if row.dayofweek == 5 or row.dayofweek == 6:

return 1

else:

return 0

temp2 = train['Datetime'].apply(applyer)

train['weekend']=temp2

train=train.drop('ID',1)

train.Timestamp = pd.to\_datetime(train.Datetime,format='%d-%m-%Y %H:%M')

train.index = train.Timestamp

# Hourly time series

hourly = train.resample('H').mean()

# Converting to daily mean

daily = train.resample('D').mean()

# Converting to weekly mean

weekly = train.resample('W').mean()

# Converting to monthly mean

monthly = train.resample('M').mean()

test.Timestamp = pd.to\_datetime(test.Datetime,format='%d-%m-%Y %H:%M')

test.index = test.Timestamp

# Converting to daily mean

test = test.resample('D').mean()

train.Timestamp = pd.to\_datetime(train.Datetime,format='%d-%m-%Y %H:%M')

train.index = train.Timestamp

# Converting to daily mean

train = train.resample('D').mean()

Train=train.ix['2012-08-25':'2014-06-24']

valid=train.ix['2014-06-25':'2014-09-25']

y\_hat\_avg = valid.copy()

fit1 = ExponentialSmoothing(np.asarray(Train['Count']) ,seasonal\_periods=7 ,trend='add', seasonal='add',).fit()

y\_hat\_avg['Holt\_Winter'] = fit1.forecast(len(valid))

plt.figure(figsize=(16,8))

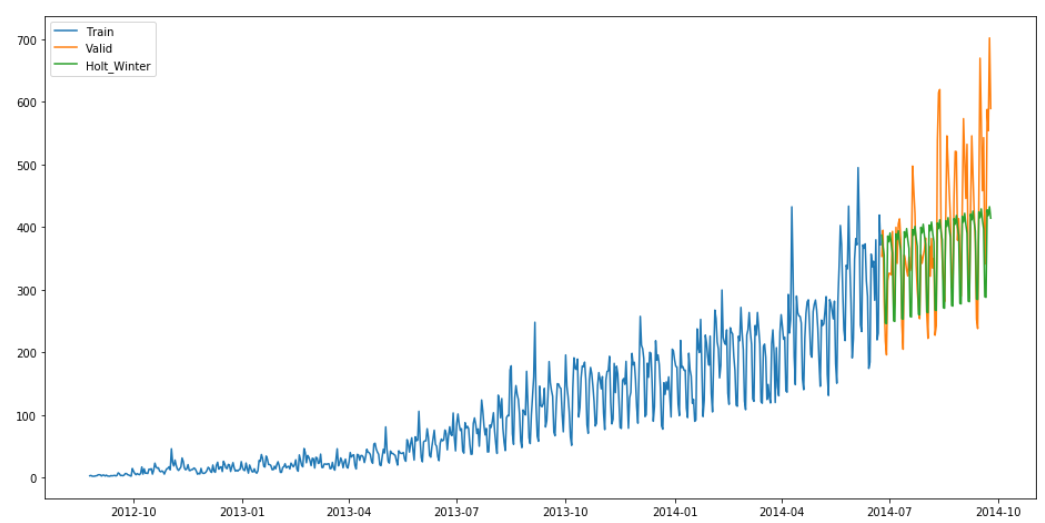
plt.plot( Train['Count'], label='Train')

plt.plot(valid['Count'], label='Valid')

plt.plot(y\_hat\_avg['Holt\_Winter'], label='Holt\_Winter')

plt.legend(loc='best')

plt.show()



**Code of Holt Winter After Changes**

import pandas as pd

import numpy as np # For mathematical calculations

import matplotlib.pyplot as plt # For plotting graphs

from datetime import datetime # To access datetime

from pandas import Series # To work on series %matplotlib inline

from sklearn.metrics import mean\_squared\_error

from math import sqrt

from statsmodels.tsa.holtwinters import ExponentialSmoothing

import warnings # To ignore the warnings

warnings.filterwarnings("ignore")

train=pd.read\_csv("Train\_SU63ISt.csv")

test=pd.read\_csv("Test\_0qrQsBZ.csv")

train\_original=train.copy()

test\_original=test.copy()

train['Datetime'] = pd.to\_datetime(train.Datetime,format='%d-%m-%Y %H:%M')

test['Datetime'] = pd.to\_datetime(test.Datetime,format='%d-%m-%Y %H:%M')

test\_original['Datetime'] = pd.to\_datetime(test\_original.Datetime,format='%d-%m-%Y %H:%M')

train\_original['Datetime'] = pd.to\_datetime(train\_original.Datetime,format='%d-%m-%Y %H:%M')

for i in (train, test, test\_original, train\_original):

i['year']=i.Datetime.dt.year

i['month']=i.Datetime.dt.month

i['day']=i.Datetime.dt.day

i['Hour']=i.Datetime.dt.hour

train['day of week']=train['Datetime'].dt.dayofweek

temp = train['Datetime']

def applyer(row):

if row.dayofweek == 5 or row.dayofweek == 6:

return 1

else:

return 0

temp2 = train['Datetime'].apply(applyer)

train['weekend']=temp2

train=train.drop('ID',1)

train.Timestamp = pd.to\_datetime(train.Datetime,format='%d-%m-%Y %H:%M')

train.index = train.Timestamp

# Hourly time series

hourly = train.resample('H').mean()

# Converting to daily mean

daily = train.resample('D').mean()

# Converting to weekly mean

weekly = train.resample('W').mean()

# Converting to monthly mean

monthly = train.resample('M').mean()

test.Timestamp = pd.to\_datetime(test.Datetime,format='%d-%m-%Y %H:%M')

test.index = test.Timestamp

# Converting to daily mean

test = test.resample('D').mean()

train.Timestamp = pd.to\_datetime(train.Datetime,format='%d-%m-%Y %H:%M')

train.index = train.Timestamp

# Converting to daily mean

train = train.resample('D').mean()

Train=train.ix['2012-08-25':'2014-07-24']

valid=train.ix['2014-07-25':'2014-09-25']

y\_hat\_avg = valid.copy()

fit1 = ExponentialSmoothing(np.asarray(Train['Count']) ,seasonal\_periods=70 ,trend='add', seasonal='add',).fit()

y\_hat\_avg['Holt\_Winter'] = fit1.forecast(len(valid))

plt.figure(figsize=(16,8))

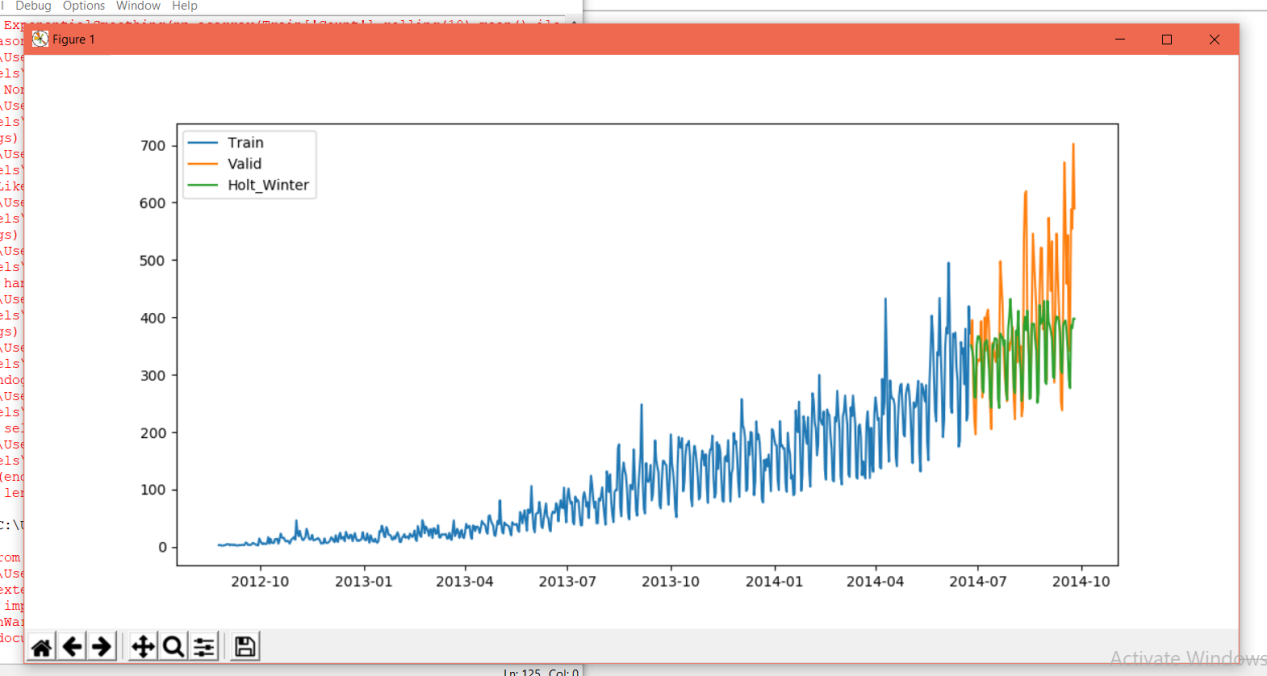
plt.plot( Train['Count'], label='Train')

plt.plot(valid['Count'], label='Valid')

plt.plot(y\_hat\_avg['Holt\_Winter'], label='Holt\_Winter\_Edited')

plt.legend(loc='best')

plt.show()

****