# DATA ANALYTICS ASSIGNMENT


NIVEDITHA C U

PES1201701640

SECTION 'C'

RACHANA H S

PES1201701726

SECTION 'E'


## PROBLEM STATEMENT:

APPLY MOVING AVERAGE AND WEIGHTED MOVING AVERAGE AND EXPONENTIAL METHODS ON A TIME SERIES DATASET.

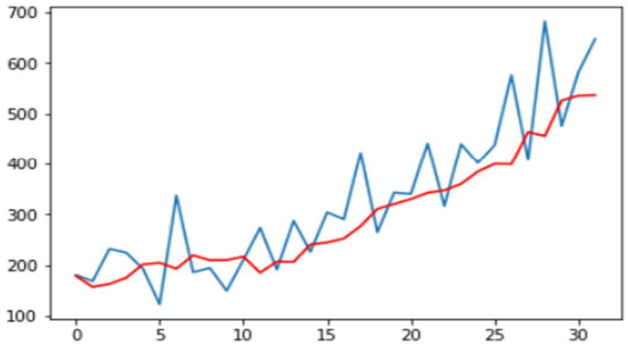
## SCREENSHOTS

MOVING AVERAGE:

In [2]:
```
from math import sqrt
from pandas import read_csv
from numpy import mean
from sklearn.metrics import mean_squared_error
from matplotlib import pyplot
series = read_csv('shampoo.csv', header=0, index_col=0)
X = series.values
window = 4
history = [X[i] for i in range(window)]
test = [X[i] for i in range(window, len(X))]
predictions = list()
for t in range(len(test)):
    length = len(history)
    yhat = mean([history[i] for i in range(length-window,length)])
    obs = test[t]
    predictions.append(yhat)
    history.append(obs)
    print('predicted=%f, expected=%f' % (yhat, obs))
error = mean_squared_error(test, predictions)
rmse=sqrt(error)
print('Test RMSE: %.3f' % rmse)

pyplot.plot(test)
pyplot.plot(predictions, color='red')
pyplot.show()
```

```
predicted=178.575000, expected=180.300000
predicted=157.150000, expected=168.500000
predicted=162.800000, expected=231.800000
predicted=174.975000, expected=224.500000
predicted=201.275000, expected=192.800000
predicted=204.400000, expected=122.900000
```

```
predicted=206.075000, expected=287.000000
predicted=240.450000, expected=226.000000
predicted=244.425000, expected=303.600000
predicted=252.000000, expected=289.900000
predicted=276.625000, expected=421.600000
predicted=310.275000, expected=264.500000
predicted=319.900000, expected=342.300000
predicted=329.575000, expected=339.700000
predicted=342.025000, expected=440.400000
predicted=346.725000, expected=315.900000
predicted=359.575000, expected=439.300000
predicted=383.825000, expected=401.300000
predicted=399.225000, expected=437.400000
predicted=398.475000, expected=575.500000
predicted=463.375000, expected=407.600000
predicted=455.450000, expected=682.000000
predicted=525.625000, expected=475.300000
predicted=535.100000, expected=581.300000
predicted=536.550000, expected=646.900000
Test RMSE: 79.942
```



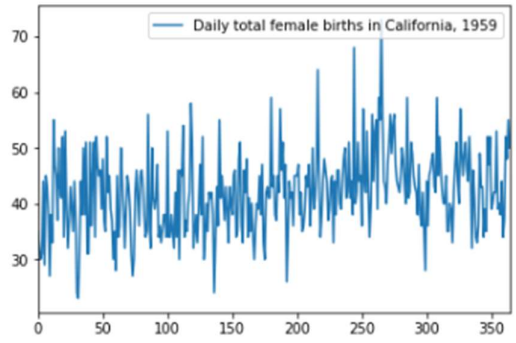

In [ ]:

WEIGHTED MOVING AVERAGE:

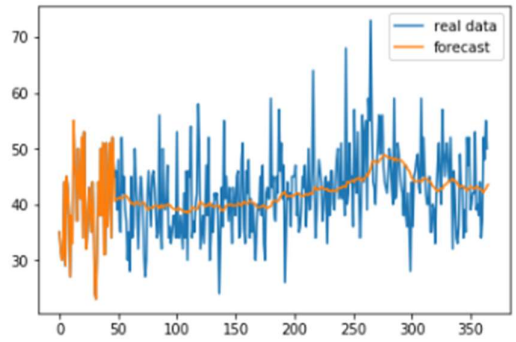
In [15]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from math import sqrt

csv_dataset = pd.read_csv("daily_total_female_births_in_cal.csv")
csv_dataset.dtypes
csv_dataset.plot()
plt.show()

optimal_n = None
best_mse = None
db = csv_dataset.iloc[:, :].values.astype('float64')
mean_results_for_all_possible_n_values = np.zeros(int(len(db) / 2 - 2))
for n in range(3, int(len(db) / 2 + 1)):
    mean_for_n = np.zeros(len(db) - n)
    for i in range(0, len(db) - n):
        weight = 1
        divider = 0
        result = 0
        for data in db[:, 0][i:i+n]:
            result += data * weight
            divider += weight
            weight += 1
        obs = result / divider
        mean_for_n[i] = np.power(obs - db[i + n][0], 2)
    mean_results_for_all_possible_n_values[n - 3] = np.mean(mean_for_n)
optimal_n = np.argmin(mean_results_for_all_possible_n_values) + 3
best_mse = np.min(mean_results_for_all_possible_n_values)
print("Best MSE = %s" % best_mse)
best_rmse=sqrt(best_mse)
print("Best RMSE = %s" % best_rmse)
print("Optimal n = %s" % optimal_n)

weight = 1
divider = 0
result = 0
for data in db[:, 0][len(db) - optimal_n: len(db)]:
    result += data * weight
    divider += weight
    weight += 1
next_observation =  result / divider
print("MA = %s" % next_observation)
```

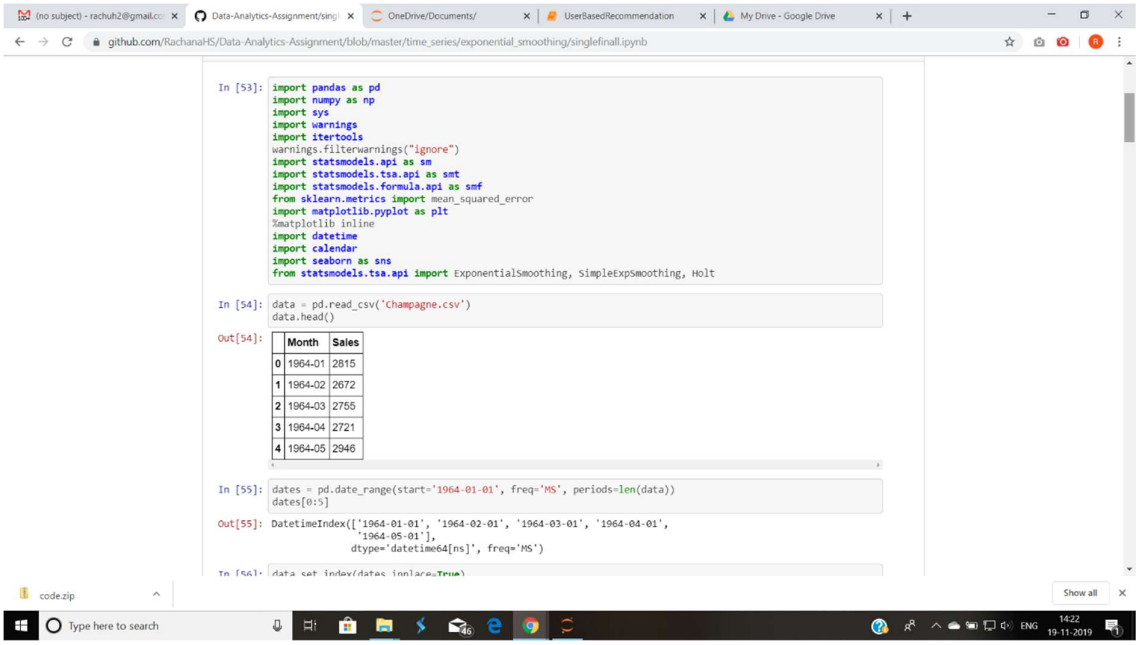
```
forecast = np.zeros(len(db) + 1)
for i in range(0, optimal_n):
    forecast[i] = db[i][0]
for i in range(0, len(db) - optimal_n + 1):
        weight = 1
        divider = 0
        result = 0
        for data in db[:, 0][i: i + optimal_n]:
            result += data * weight
            divider += weight
            weight += 1
        forecast[i+optimal_n] = result / divider
plt.plot(db[:, 0],label = 'real data')
plt.plot(forecast, label = 'forecast')
plt.legend()
plt.show()
```




```
Best MSE = 47.17514578536712
Best RMSE = 6.868416541341033
Optimal n = 47
MA = 43.54609929078014
```




SINGLE EXPONENTIAL



In [53]:
```
import pandas as pd
import numpy as np
import sys
import warnings
import itertools
warnings.filterwarnings("ignore")
import statsmodels.api as sm
import statsmodels.tsa.api as smt
import statsmodels.formula.api as smf
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
%matplotlib inline
import datetime
import calendar
import seaborn as sns
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt
```

In [54]:
```
data = pd.read_csv('Champagne.csv')
data.head()
```

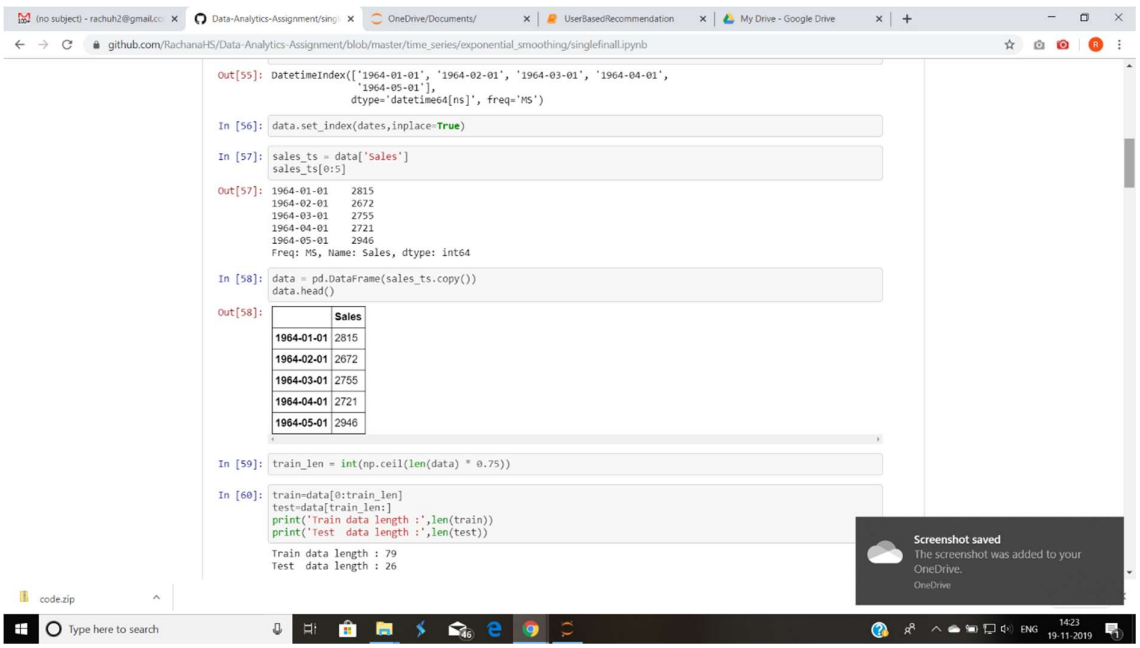
Out[54]:

| | Month | Sales |
|---|---|---|
| 0 | 1964-01 | 2815 |
| 1 | 1964-02 | 2672 |
| 2 | 1964-03 | 2755 |
| 3 | 1964-04 | 2721 |
| 4 | 1964-05 | 2946 |

In [55]:
```
dates = pd.date_range(start='1964-01-01', freq='MS', periods=len(data))
dates[0:5]
```

Out[55]:
```
DatetimeIndex(['1964-01-01', '1964-02-01', '1964-03-01', '1964-04-01',
               '1964-05-01'],
              dtype='datetime64[ns]', freq='MS')
```

In [56]:
```
data.set_index(dates,inplace=True)
```



In [57]:
```
sales_ts = data['Sales']
sales_ts[0:5]
```

Out[57]:
```
1964-01-01    2815
1964-02-01    2672
1964-03-01    2755
1964-04-01    2721
1964-05-01    2946
Freq: MS, Name: Sales, dtype: int64
```

In [58]:
```
data = pd.DataFrame(sales_ts.copy())
data.head()
```

Out[58]:

| | Sales |
|---|---|
| 1964-01-01 | 2815 |
| 1964-02-01 | 2672 |
| 1964-03-01 | 2755 |
| 1964-04-01 | 2721 |
| 1964-05-01 | 2946 |

In [59]:
```
train_len = int(np.ceil(len(data) * 0.75))
```

In [60]:
```
train=data[0:train_len]
test=data[train_len:]
print('Train data length :',len(train))
print('Test  data length :',len(test))
```

```
Train data length : 79
Test  data length : 26
```

Screenshot saved
The screenshot was added to your OneDrive.
OneDrive

```
data['alpha=0.8'] = exp_smooth(0.8)
```

In [64]:
```
data
```

Out[64]:

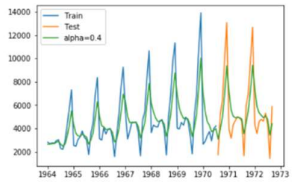
| | Sales | alpha=0.2 | alpha=0.4 | alpha=0.6 | alpha=0.8 |
|---|---|---|---|---|---|
| **1964-01-01** | 2815 | 2639.000000 | 2639.000000 | 2639.000000 | 2639.000000 |
| **1964-02-01** | 2672 | 2645.600000 | 2652.200000 | 2658.800000 | 2665.400000 |
| **1964-03-01** | 2755 | 2667.480000 | 2693.320000 | 2716.520000 | 2737.080000 |
| **1964-04-01** | 2721 | 2678.184000 | 2704.392000 | 2719.208000 | 2724.216000 |
| **1964-05-01** | 2946 | 2731.747200 | 2801.035200 | 2855.283200 | 2901.643200 |
| **1964-06-01** | 3036 | 2792.597760 | 2895.021120 | 2963.713280 | 3009.128640 |
| **1964-07-01** | 2282 | 2690.478208 | 2649.812672 | 2554.685312 | 2427.425728 |
| **1964-08-01** | 2212 | 2594.782566 | 2474.687603 | 2349.074125 | 2255.085146 |
| **1964-09-01** | 2922 | 2660.226053 | 2653.612562 | 2692.829650 | 2788.617029 |
| **1964-10-01** | 4301 | 2988.380842 | 3312.567537 | 3657.731860 | 3998.523406 |
| **1964-11-01** | 5764 | 3543.504674 | 4293.140522 | 4921.492744 | 5410.904681 |
| **1964-12-01** | 7312 | 4297.203739 | 5500.684313 | 6355.797098 | 6931.780936 |
| **1965-01-01** | 2541 | 3945.962991 | 4316.810588 | 4066.918839 | 3419.156187 |
| **1965-02-01** | 2475 | 3651.770393 | 3580.086353 | 3111.767536 | 2663.831237 |
| **1965-03-01** | 3031 | 3527.616314 | 3360.451812 | 3063.307014 | 2957.566247 |
| **1965-04-01** | 3266 | 3475.293052 | 3322.671087 | 3184.922806 | 3204.313249 |
| **1965-05-01** | 3776 | 3535.434441 | 3504.002652 | 3539.569122 | 3661.662650 |
| **1965-06-01** | 3230 | 3474.347553 | 3394.401591 | 3353.827649 | 3316.332530 |
| **1965-07-01** | 3028 | 3385.078042 | 3247.840955 | 3158.331060 | 3085.666506 |
| **1965-08-01** | 1759 | 3059.862434 | 2652.304573 | 2318.732424 | 2024.333301 |
| **1965-09-01** | 3595 | 3166.889947 | 3029.382744 | 3084.492970 | 3280.866660 |





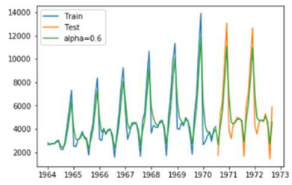
```
plt.plot(test['Sales'], label='Test')
plt.plot(data['alpha=0.4'], label='alpha=0.4')
plt.legend(loc='best')
```

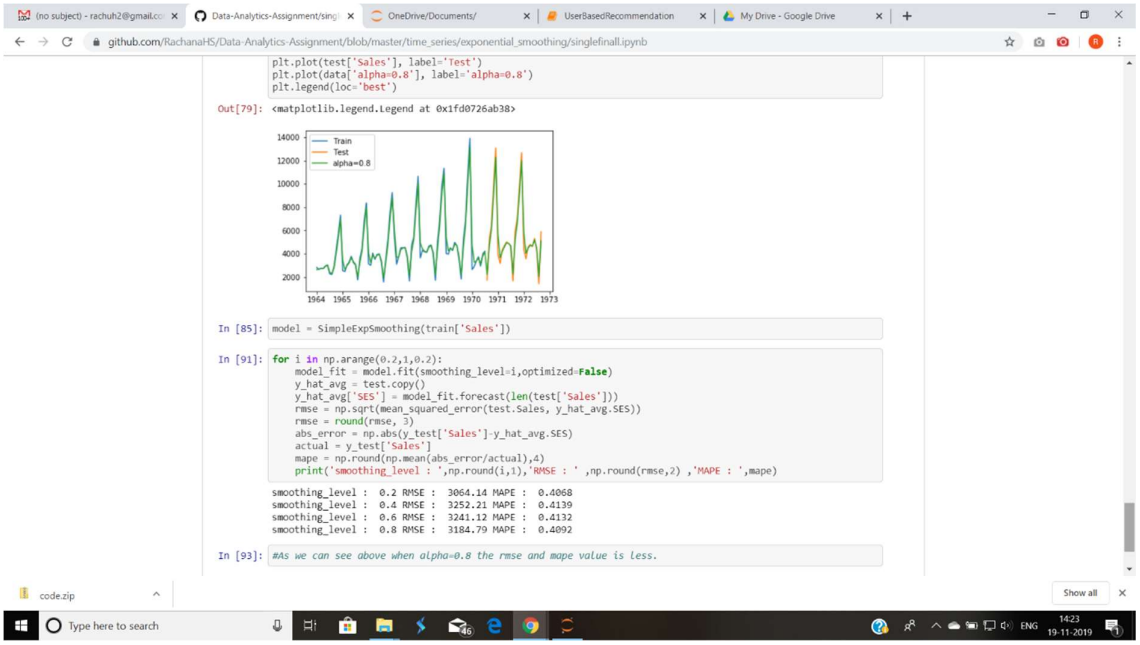
Out[76]:
```
<matplotlib.legend.Legend at 0x1fd07101860>
```




In [78]:
```
plt.plot(train['Sales'], label='Train')
plt.plot(test['Sales'], label='Test')
plt.plot(data['alpha=0.6'], label='alpha=0.6')
plt.legend(loc='best')
```

Out[78]:
```
<matplotlib.legend.Legend at 0x1fd071ecf98>
```





```
plt.plot(test['Sales'], label='Test')
plt.plot(data['alpha=0.8'], label='alpha=0.8')
plt.legend(loc='best')
```

Out[79]: <matplotlib.legend.Legend at 0x1fd0726ab38>



In [85]:
```
model = SimpleExpSmoothing(train['Sales'])
```

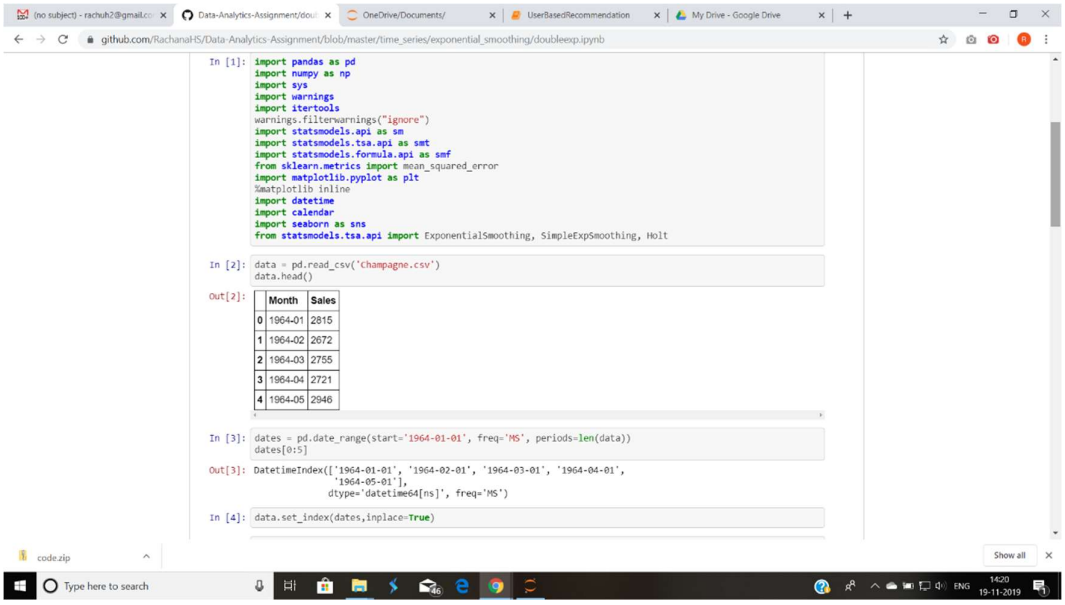
In [91]:
```
for i in np.arange(0.2,1,0.2):
    model_fit = model.fit(smoothing_level=i,optimized=False)
    y_hat_avg = test.copy()
    y_hat_avg['SES'] = model_fit.forecast(len(test['Sales']))
    rmse = np.sqrt(mean_squared_error(test.Sales, y_hat_avg.SES))
    rmse = round(rmse, 3)
    abs_error = np.abs(y_test['Sales']-y_hat_avg.SES)
    actual = y_test['Sales']
    mape = np.round(np.mean(abs_error/actual),4)
    print('smoothing_level : ',np.round(i,1),'RMSE : ' ,np.round(rmse,2) ,'MAPE : ',mape)
```

```
smoothing_level :  0.2 RMSE :  3064.14 MAPE :  0.4068
smoothing_level :  0.4 RMSE :  3252.21 MAPE :  0.4139
smoothing_level :  0.6 RMSE :  3241.12 MAPE :  0.4132
smoothing_level :  0.8 RMSE :  3184.79 MAPE :  0.4092
```

In [93]:
```
#As we can see above when alpha=0.8 the rmse and mape value is less.
```



## DOUBLE EXPONENTIAL



In [1]:
```
import pandas as pd
import numpy as np
import sys
import warnings
import itertools
warnings.filterwarnings("ignore")
import statsmodels.api as sm
import statsmodels.tsa.api as smt
import statsmodels.formula.api as smf
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
%matplotlib inline
import datetime
import calendar
import seaborn as sns
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt
```

In [2]:
```
data = pd.read_csv('Champagne.csv')
data.head()
```

Out[2]:

| | Month | Sales |
|---|---|---|
| 0 | 1964-01 | 2815 |
| 1 | 1964-02 | 2672 |
| 2 | 1964-03 | 2755 |
| 3 | 1964-04 | 2721 |
| 4 | 1964-05 | 2946 |

In [3]:
```
dates = pd.date_range(start='1964-01-01', freq='MS', periods=len(data))
dates[0:5]
```

Out[3]:
```
DatetimeIndex(['1964-01-01', '1964-02-01', '1964-03-01', '1964-04-01',
               '1964-05-01'],
              dtype='datetime64[ns]', freq='MS')
```

In [4]:
```
data.set_index(dates,inplace=True)
```

github.com/RachanaHS/Data-Analytics-Assignment/blob/master/time_series/exponential_smoothing/doubleexp.ipynb

In [2]:
```
data = pd.read_csv('Champagne.csv')
data.head()
```

Out[2]:

| | Month | Sales |
|---|---|---|
| 0 | 1964-01 | 2815 |
| 1 | 1964-02 | 2672 |
| 2 | 1964-03 | 2755 |
| 3 | 1964-04 | 2721 |
| 4 | 1964-05 | 2946 |

In [3]:
```
dates = pd.date_range(start='1964-01-01', freq='MS', periods=len(data))
dates[0:5]
```

Out[3]:
```
DatetimeIndex(['1964-01-01', '1964-02-01', '1964-03-01', '1964-04-01',
               '1964-05-01'],
              dtype='datetime64[ns]', freq='MS')
```

In [4]:
```
data.set_index(dates,inplace=True)
```

In [5]:
```
sales_ts = data['Sales']
sales_ts[0:5]
```

Out[5]:
```
1964-01-01    2815
1964-02-01    2672
1964-03-01    2755
1964-04-01    2721
1964-05-01    2946
Freq: MS, Name: Sales, dtype: int64
```

In [6]:
```
data = pd.DataFrame(sales_ts.copy())
data.head()
```

Out[6]:

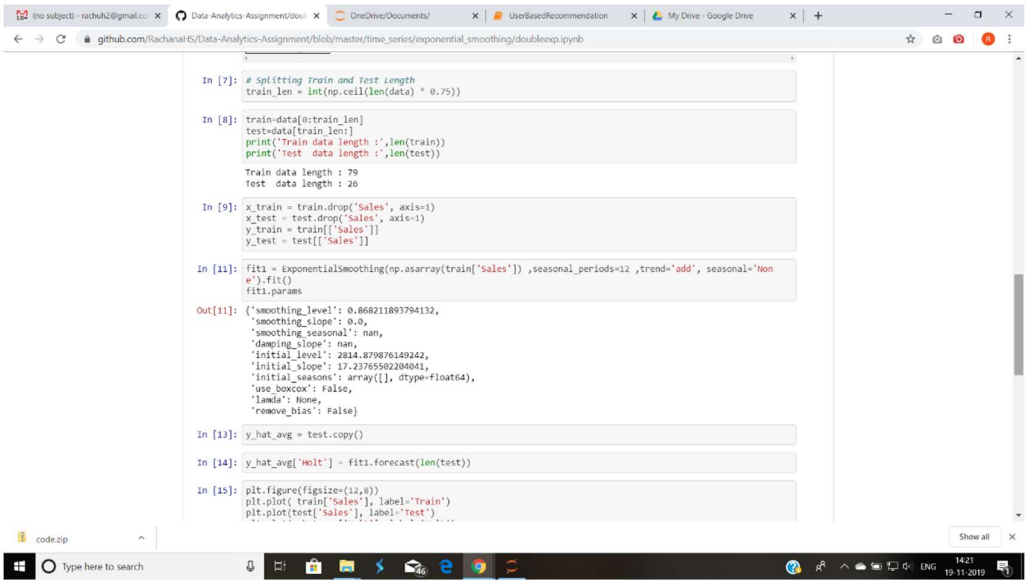
| | Sales |
|---|---|
| 1964-01-01 | 2815 |

code.zip

Show all

Type here to search

ENG 14:21 19-11-2019



github.com/RachanaHS/Data-Analytics-Assignment/blob/master/time_series/exponential_smoothing/doubleexp.ipynb

In [7]:
```
# Splitting Train and Test Length
train_len = int(np.ceil(len(data) * 0.75))
```

In [8]:
```
train=data[0:train_len]
test=data[train_len:]
print('Train data length :',len(train))
print('Test  data length :',len(test))
```

```
Train data length : 79
Test  data length : 26
```

In [9]:
```
x_train = train.drop('Sales', axis=1)
x_test = test.drop('Sales', axis=1)
y_train = train[['Sales']]
y_test = test[['Sales']]
```

In [11]:
```
fit1 = ExponentialSmoothing(np.asarray(train['Sales']) ,seasonal_periods=12 ,trend='add', seasonal='Non
e').fit()
fit1.params
```

Out[11]:
```
{'smoothing_level': 0.868211893794132,
 'smoothing_slope': 0.0,
 'smoothing_seasonal': nan,
 'damping_slope': nan,
 'initial_level': 2814.879876149242,
 'initial_slope': 17.237655022040411,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [13]:
```
y_hat_avg = test.copy()
```

In [14]:
```
y_hat_avg['Holt'] = fit1.forecast(len(test))
```

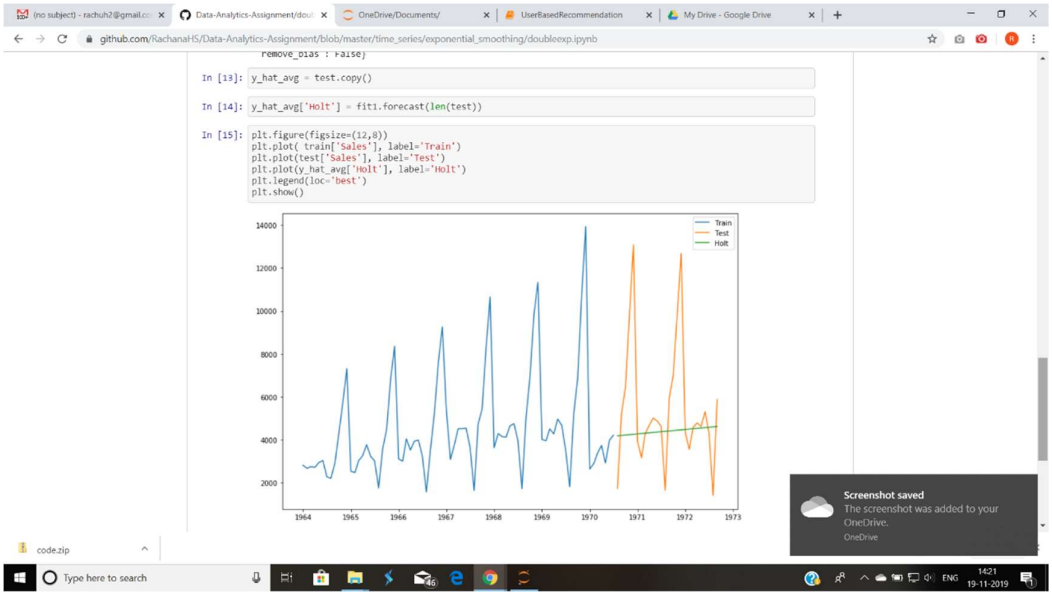
In [15]:
```
plt.figure(figsize=(12,8))
plt.plot( train['Sales'], label='Train')
plt.plot(test['Sales'], label='Test')
```

code.zip

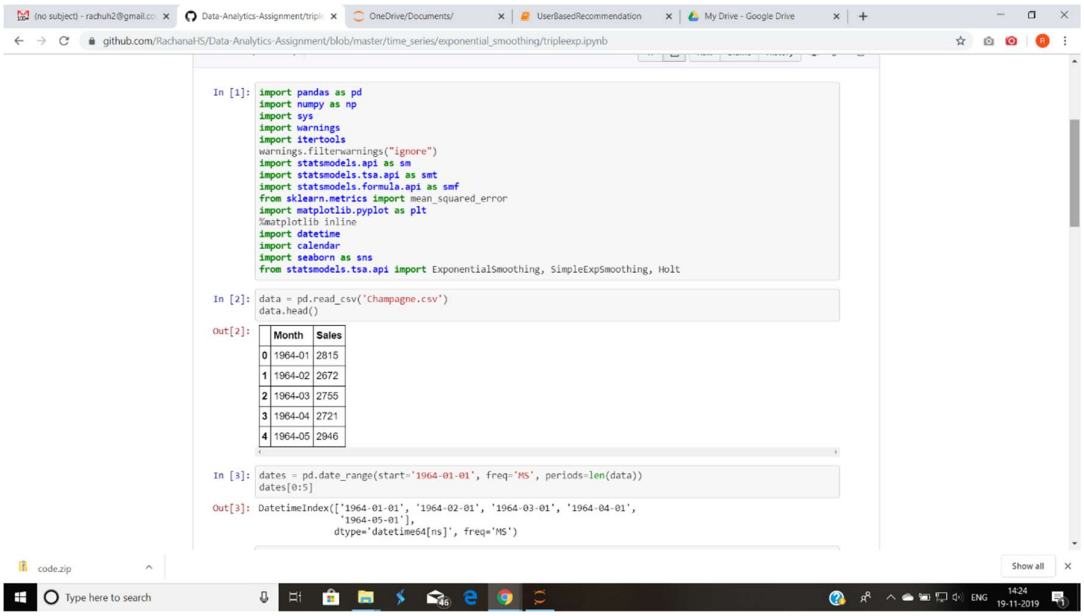
Show all

Type here to search

ENG 14:21 19-11-2019




# TRIPLE EXPONENTIAL



```
1964-05-01    2946
Freq: MS, Name: Sales, dtype: int64
```

In [6]:
```
data = pd.DataFrame(sales_ts.copy())
data.head()
```

Out[6]:

| | Sales |
|---|---|
| **1964-01-01** | 2815 |
| **1964-02-01** | 2672 |
| **1964-03-01** | 2755 |
| **1964-04-01** | 2721 |
| **1964-05-01** | 2946 |

In [7]:
```
# Splitting Train and Test Length
train_len = int(np.ceil(len(data) * 0.75))
```

In [8]:
```
train=data[0:train_len]
test=data[train_len:]
print('Train data length :',len(train))
print('Test  data length :',len(test))
```

```
Train data length : 79
Test  data length : 26
```

In [9]:
```
x_train = train.drop('Sales', axis=1)
x_test = test.drop('Sales', axis=1)
y_train = train[['Sales']]
y_test = test[['Sales']]
```

In [10]:
```
y_hat_avg = test.copy()
```

In [11]:
```
fit1 = ExponentialSmoothing(np.asarray(train['Sales']) ,seasonal_periods=12 ,trend='add', seasonal='add').
fit()
fit1.params
```

Out[11]:
```
{'smoothing_level': 0.05574986291458101,
```

code.zip

Show all

Type here to search

ENG 14:25 19-11-2019

(no subject) - rachuh2@gmail.co | Data-Analytics-Assignment/tripl | OneDrive/Documents/ | UserBasedRecommendation | My Drive - Google Drive

github.com/RachanaHS/Data-Analytics-Assignment/blob/master/time_series/exponential_smoothing/tripleexp.ipynb

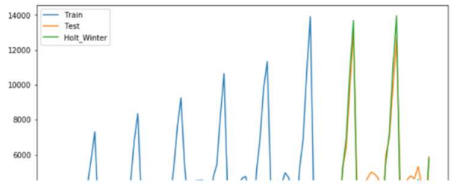
In [11]:
```
fit1 = ExponentialSmoothing(np.asarray(train['Sales']) ,seasonal_periods=12 ,trend='add', seasonal='add').
fit()
fit1.params
```

Out[11]:
```
{'smoothing_level': 0.05574986291458101,
 'smoothing_slope': 1.443911683751581e-12,
 'smoothing_seasonal': 0.9053845872345992,
 'damping_slope': nan,
 'initial_level': 3339.9105862162864,
 'initial_slope': 22.2445016461074,
 'initial_seasons': array([ -630.65411606,  -773.57658123,  -687.91626227,  -723.9294304 ,
        -501.55137492,  -421.01244496, -1174.51840923, -1257.20818584,
        -539.11320431,   832.32560022,  2298.17467999,  3838.99033183]),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [12]:
```
y_hat_avg['Holt_Winter'] = fit1.forecast(len(test))
```

In [13]:
```
plt.figure(figsize=(10,6))
plt.plot( train['Sales'], label='Train')
plt.plot(test['Sales'], label='Test')
plt.plot(y_hat_avg['Holt_Winter'], label='Holt_Winter')
plt.legend(loc='best')
```

Out[13]:
```
<matplotlib.legend.Legend at 0x2776b3fa978>
```




Screenshot saved
The screenshot was added to your OneDrive.
OneDrive

code.zip

Type here to search

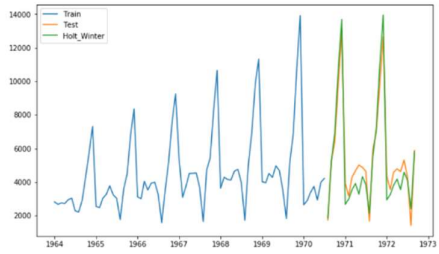
ENG 14:25 19-11-2019

(no subject) - rachuh2@gmail.co | Data-Analytics-Assignment/triple | OneDrive/Documents/ | UserBasedRecommendation | My Drive - Google Drive

github.com/RachanaHS/Data-Analytics-Assignment/blob/master/time_series/exponential_smoothing/tripleexp.ipynb

In [13]:
```
plt.figure(figsize=(10,6))
plt.plot( train['Sales'], label='Train')
plt.plot(test['Sales'], label='Test')
plt.plot(y_hat_avg['Holt_Winter'], label='Holt_Winter')
plt.legend(loc='best')
```

Out[13]: <matplotlib.legend.Legend at 0x2776b3fa978>




In [14]:
```
# Root Mean Square Error (RMSE)
rmse = np.sqrt(mean_squared_error(test.Sales, y_hat_avg.Holt_Winter))
rmse = round(rmse, 3)
```

In [15]:
```
# Mean Absolute Percentage Error
abs_error = np.abs(y_test['Sales']-y_hat_avg.Holt_Winter)
actual = y_test['Sales']
mape = np.round(np.mean(abs_error/actual),4)
```

In [16]:
```
print(rmse)
```

code.zip

Show all

Type here to search

ENG 14:25 19-11-2019