TIME SERIES MODELING & ANALYSIS

Instructor Name: Reza Jafari

Lab#: 4

Submitted by Dinesh Kumar Padmanabhan

Date: 09-Oct-2020

ABSTRACT

In LAB #4, we learned concepts of different forecast methods namely, Average , Naïve, Drift, simple exponential smoothing, Holts linear trend and Holts winter seasonal methods. Using these forecast methods we performed one-step-ahead prediction and h-step prediction. With the help of simple datasets, we plotted various time series plots and made a comparison with respect to forecast accuracy.

INTRODUCTION

Some forecasting methods are very simple and surprisingly effective. We will use the following forecasting methods as benchmarks throughout this lab.

Average method

Here, the forecasts of all future values are equal to the average (or "mean") of the historical data. If we let the historical data be denoted by y1,…,yTy1,…,yT, then we can write the forecasts as

$$\hat{y}_{T+h|T} = \frac{y_1 + y_2 + \ldots + y_T}{T}$$

Naïve method

For naïve forecasts, we simply set all forecasts to be the value of the last observation. That is,

$$\hat{y}_{T+h|T} = y_T$$

Drift method

The variation on the naiıve method is to allow the forecast to increase or decrease over time, where the amount of change over time (called the drift) is set to be the average change seen in the historical data. Formally, the forecast for time T + h is written as :

$$\hat{y}_{T+h|T} = y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1}) = y_T + h\left(\frac{y_T - y_1}{T-1}\right)$$

Simple Exponential Smoothing

Simple exponential smoothing is calculated using weighted averages where the weights decrease exponentially as observations come from further in the past, the smallest weights are associated with the oldest observations. Simple exponential smoothing is between the two extremes: naive and average.

$$\boxed{\hat{y}_{t+1|t} = \alpha y_t + (1-\alpha)\hat{y}_{t|t-1}}$$

where 0 ≤ α ≤ 1 is called the damping factor.

Holts Linear Trend Method

Holt's (1957) extended simple exponential smoothing to allow the forecasting of data with trend. This method involves a forecast equation and two smoothing equations (one for level and one for the trend):

$$\begin{aligned} &(\textit{Forecast equation}) & \hat{y}_{t+h|t} &= \ell_t + hb_t \\ &(\textit{Level equation}) & \ell_t &= \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\ &(\textit{Trend equation}) & b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \end{aligned}$$

Holt-Winter seasonal method

Holt-Winter seasonal method comprises the forecast equation and three smoothing equations:

1 Level ℓt

2 Trend bt

3 Seasonal st

$$\begin{aligned} \hat{y}_{t+h|t} &= \ell_t + hb_t + s_{t+h-m(k+1)} \\ \ell_t &= \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\ b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \\ s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \end{aligned}$$

METHOD, THEORY & PROCEDURES

Method:

1. Programming Language: Python

Libraries used: Some basic libraries used for analysis & model building are mentioned below

library(Numpy) - large collection of high-level mathematical functions to operate on these arrays.

library (Pandas) – For Data manipulation and analysis

library(Matplotlib) – is a system for declaratively creating graphics

library(Math) –To Compute mathematical calculations

library (statsmodels) – Import statistical models

Theory:

To Plot the forecast accuracy of above-mentioned methods for the given data set and determine which method performs better.

Procedure:

I shall be looking at the results of various forecast accuracy methods and time series plots and infer about it in my analysis. And through my exploration I shall try to identify which methods perform better and draw inferences.

The Dataset will be explored in following stages:

1. **Data Exploration (EDA)** – looking at different forecast methods and making inferences about the data.
2. **Data Visualization** – Plotting different time series plots for the forecast methods and forecast accuracy.
3. **Testing** – Running ACF to identify the correlation between errors.

ANSWERS TO QUESTIONS

File - unknown

```
C:\ProgramData\Anaconda3\python.exe "C:\Program Files\
JetBrains\PyCharm 2019.3.1\plugins\python\helpers\pydev\
pydevconsole.py" --mode=client --port=52099

import sys; print('Python %s on %s' % (sys.version, sys.
platform))
sys.path.extend(['C:\\Users\\nsree_000\\Desktop\\Python-
Quiz', 'C:/Users/nsree_000/Desktop/Python-Quiz'])

Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915
64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more
information
IPython 7.8.0 -- An enhanced Interactive Python. Type '?'
for help.
PyDev console: using IPython 7.8.0

Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915
64 bit (AMD64)] on win32
In[2]: runfile('C:/Users/nsree_000/Desktop/Python-Quiz/TIME
 SERIES/LAB4.py', wdir='C:/Users/nsree_000/Desktop/Python-
Quiz/TIME SERIES')


--------------------FORECAST METHOD| AVERAGE
--------------------
Average method 1-step prediction:  [112.0, 115.0, 120.
66666666666667, 122.75, 122.4, 124.5, 127.85714285714286,
128.875]
Average method h-step prediction:  [127.77777778 127.
77777778 127.77777778 127.77777778 127.77777778]
Mean Square Error of prediction errors for Average method
:  159.04233648667798
Mean Square Error of forecast errors for Average method:
200.44938271604934
Mean of prediction errors for Average method:  7.
993898809523808
Variance of prediction errors for Average method:  95.
13991830977182
Variance of forecast errors for Average method:  151.76
[0.046811024090599146, -0.21182764494224582, -0.
04682758913462302, 0.4002377834160776, -0.
023101138293326388, -0.6757791485049354, 0.
010486713368453861, 1.0, 0.010486713368453861, -0.
6757791485049354, -0.023101138293326388, 0.4002377834160776
, -0.04682758913462302, -0.21182764494224582, 0.
```

Page 1 of 3

```
046811024090599146]
Q value for Average method:  6.000897787361459


--------------------FORECAST METHOD| NAIVE --------------------
Naive method 1-step prediction:  [112, 118, 132, 129, 121, 135, 148, 136]
Naive method h-step prediction:  [119. 119. 119. 119. 119.]
Mean Square Error of prediction errors for Naive method:  137.875
Mean Square Error of forecast errors for Naive method:  155.0
Mean of prediction errors for Naive method:  0.875
Variance of prediction errors for Naive method:  137.109375
Variance of forecast errors for Naive method:  151.76
Q value for Naive method:  5.85083686901892


--------------------FORECAST METHOD| DRIFT --------------------
Drift method 1-step prediction:  [112, 124.0, 142.0, 134.66666666666666, 123.25, 139.6, 154.0, 139.42857142857142]
Drift method h-step prediction:  [119.875, 120.75, 121.625, 122.5, 123.375]
Mean Square Error of prediction errors for Drift method:  175.71585104875277
Mean Square Error of forecast errors for Drift method:  125.271875
Mean of prediction errors for Drift method:  -3.8681547619047585
Variance of prediction errors for Drift method:  160.7532297867063
Variance of forecast errors for Drift method:  124.59124999999999
Q value for Drift method:  7.385119910860086


--------------------FORECAST METHOD| SIMPLE EXPONENTIAL METHOD--------------------
SES method 1-step prediction with damping factor 0.5:  [112, 115.0, 123.5, 126.25, 123.625, 129.3125, 138.65625, 137.328125]
SES method h-step prediction with damping factor 0.5:  [128.1640625 128.1640625 128.1640625 128.1640625 128.1640625]
Mean Square Error of prediction errors for SES method:  150
```

```
.55020141601562
Mean Square Error of forecast errors for SES method:  205.98941650390626
Mean of prediction errors for SES method:  4.041015625
Variance of prediction errors for SES method:  134.22039413452148
Variance of forecast errors for SES method:  151.76
Q value for SES method:  5.172049247582443


--------------------COMPARISION OF FORECAST METHODS--------------------
                                 Q_val    MSE_pred  MSE_forecast  Mean_pred  \
Method


Average                       6.000898  159.042336    200.449383   7.993899
Naive                         5.850837  137.875000    155.000000   0.875000
Drift                         7.385120  175.715851    125.271875  -3.868155
Simple Exponential Smoothing  5.172049  150.550201    205.989417   4.041016

                              variance_pred  variance_forecast
Method

Average                           95.139918          151.76000
Naive                            137.109375          151.76000
Drift                            160.753230          124.59125
Simple Exponential Smoothing     134.220394          151.76000

```

```
#%%===========================================================================
# 1: Let suppose a time series dataset is given as below. Without a help of Python
and using the average forecast method perform one-step ahead prediction and fill
out the table. To perform the correct cross-validation, start with two
observations {y1, y2} and predict y3 (you can now generate the first error).
Add observations by one {y1, y2, y3} and predict y4 (you can now generate the
second error). Continue this through the dataset. Then calculate the MSE of the 1-
step prediction and MSE of h-step forecast.
# %%---------------------------------------------------------------------------
```

$\boxed{\text{AVERAGE METHOD}}$

TRAINING SET 1- STEP AHEAD PREDICTION

t	y_t	$\hat{y}_{t+1 \mid t}$	e	e^2
1	112	-	-	-
2	118	112	6	36
3	132	115	17	289
4	129	120.67	8.33	69.39
5	121	122.75	-1.75	03.06
6	135	122.4	12.6	158.76
7	148	122.5	23.5	552.25
8	136	127.86	8.14	66.26
9	119	128.88	-9.88	97.61

Mean square Error 1-step:

$$= \frac{36 + 289 + 69.39 + 3.06 + 158.76 + 552.25 + 66.26 + 97.61}{8}$$

$$= 159.04$$

TESTING SET h-step ahead ~~Prediction~~ forecast

t	y_t	$\hat{y}_{t+h \mid t}$	e	e^2
1	104	127.78	-23.78	565.49
2	118	127.78	-9.78	95.65
3	115	127.78	-12.78	163.33
4	126	127.78	-1.78	3.17
5	141	127.78	13.22	174.77

$$MSE_{h\text{-step}} = \frac{565.49 + 95.65 + 163.33 + 3.17 + 174.77}{5}$$

$$= 200.48$$

$\boxed{\text{NAIVE METHOD}}$

1- step ahead Prediction (Training set)

t	y_t	$\hat{y}_{t+1\|t}$	e	e^2
1	112	-	-	-
2	118	112	6	36
3	132	118	14	196
4	129	132	-3	9
5	121	129	-8	64
6	135	121	14	196
7	148	135	13	169
8	136	148	-12	144
9	119	136	-17	289

$$MSE_{1\text{-step}} : \frac{36+196+9+64+196+169+144+289}{8}$$

$$= 137.88$$

h- step ahead forecast (Testing set)

t	y_t	$\hat{y}_{t+h\|t}$	e	e^2
1	104	119	-15	225
2	118	119	-1	1
3	115	119	-4	16
4	126	119	7	49
5	141	119	22	484

$$MSE_{h\text{-step}} : \frac{225+1+16+49+484}{5}$$

$$= 155$$

DRIFT METHOD

1- step ahead Prediction (training set)

t	y_t	$\hat{y}_{t+1\|t}$	e	e^2
1	112	-	-	-
2	118	112	6	36
3	132	124	8	64
4	129	142	-13	169
5	121	134.67	-13.67	186.87
6	135	123.25	11.75	138.06
7	148	139.6	8.4	70.56
8	136	154	-18	324
9	119	139.43	-20.43	417.38

$$\hat{y}_{t+1|t} = y_t + h\left(\frac{y_t - y_1}{t-1}\right)$$

$$\hat{y}_{2|1} = 112$$

$$\hat{y}_{3|2} = 118 + \left(\frac{118-112}{2-1}\right) = 124$$

$$\hat{y}_{4|3} = 132 + \left(\frac{132-112}{3-1}\right) = 142$$

$$\hat{y}_{5|4} = 129 + \left(\frac{129-112}{4-1}\right) = 134.67$$

$$\hat{y}_{6|5} = 121 + \left(\frac{121-112}{5-1}\right) = 123.25$$

$$\hat{y}_{7|6} = 135 + \left(\frac{135-112}{6-1}\right) = 139.6$$

$$\hat{y}_{8|7} = 148 + \left(\frac{148-112}{7-1}\right) = 154$$

$$\hat{y}_{9|8} = 136 + \left(\frac{136-112}{8-1}\right) = 139.43$$

$$MSE_{1\,step} = \frac{36 + 64 + 169 + 186.67 + 138.06 + 70.56 + 324 + 417.38}{8}$$

$= 175.73$

h-step ahead forecast (Testing set)

h	y_{t+h}	$\hat{y}_{t+h\|t}$	e	e^2
1	104	119.88	-15.88	252.17
2	118	120.75	-2.75	7.56
3	115	121.63	-6.63	43.96
4	126	122.5	3.5	12.25
5	141	123.38	17.62	310.46

$$\hat{y}_{t+h|t} = y_t + h\left(\frac{y_t - y_1}{t-1}\right)$$

$$\hat{y}_{10|9} = 119 + 1\left(\frac{119-112}{9-1}\right) = 119.88$$

$$\hat{y}_{11|9} = 119 + 2\left(\frac{119-112}{9-1}\right) = 120.75$$

$$\hat{y}_{12|9} = 119 + 3\left(\frac{119-112}{9-1}\right) = 121.63$$

$$\hat{y}_{13|9} = 119 + 4\left(\frac{119-112}{9-1}\right) = 122.5$$

$$\hat{y}_{14|9} = 119 + 5\left(\frac{119-112}{9-1}\right) = 123.28$$

$$MSE_{hstep} = \frac{252.17 + 7.56 + 43.96 + 12.25 + 310.46}{5}$$

$$= 125.28$$

SIMPLE EXPONENTIAL SMOOTHING METHOD

1-step ahead prediction (Training set)

t	y_t	$\hat{y}_{t+1\|t}$	e	e^2
1	112	–	–	–
2	118	112	6	36
3	132	115	17	289
4	129	123.5	5.5	30.25
5	121	126.25	-5.25	27.56
6	135	123.63	11.37	129.28
7	148	129.32	18.68	348.94
8	136	138.66	-2.66	7.08
9	119	137.33	-18.33	335.99

$\hat{y}_{t+1|t} = \alpha y_t + (1-\alpha)\,\hat{y}_{t|t+1}$

$\hat{y}_{2|1} = (0.5)112 + (1-0.5)\,112 = 112$

$\hat{y}_{3|2} = (0.5)118 + (1-0.5)\,112 = 115$

$\hat{y}_{4|3} = (0.5)132 + (1-0.5)115 = 123.5$

$\hat{y}_{5|4} = (0.5)129 + (1-0.5)123.5 = 126.25$

$\hat{y}_{6|5} = (0.5)121 + (1-0.5)126.25 = 123.63$

$\hat{y}_{7|6} \quad (0.5)135 + (1-0.5)123.63 = 129.32$

$\hat{y}_{8|7} \quad (0.5)148 + (1-0.5)129.32 = 138.66$

$\hat{y}_{9|8} \quad (0.5)136 + (1-0.5)138.66 = 137.33$

$\hat{y}_{10|9} \quad (0.5)119 + (1-0.5)137.33 = 128.17$

$$MSE_{1\text{-step}} = \frac{36 + 289 + 30.25 + 27.56 + 129.28 + 348.94 + 7.08 + 335.99}{8}$$

$= 150.59$

h-step ahead forecast (testing set)

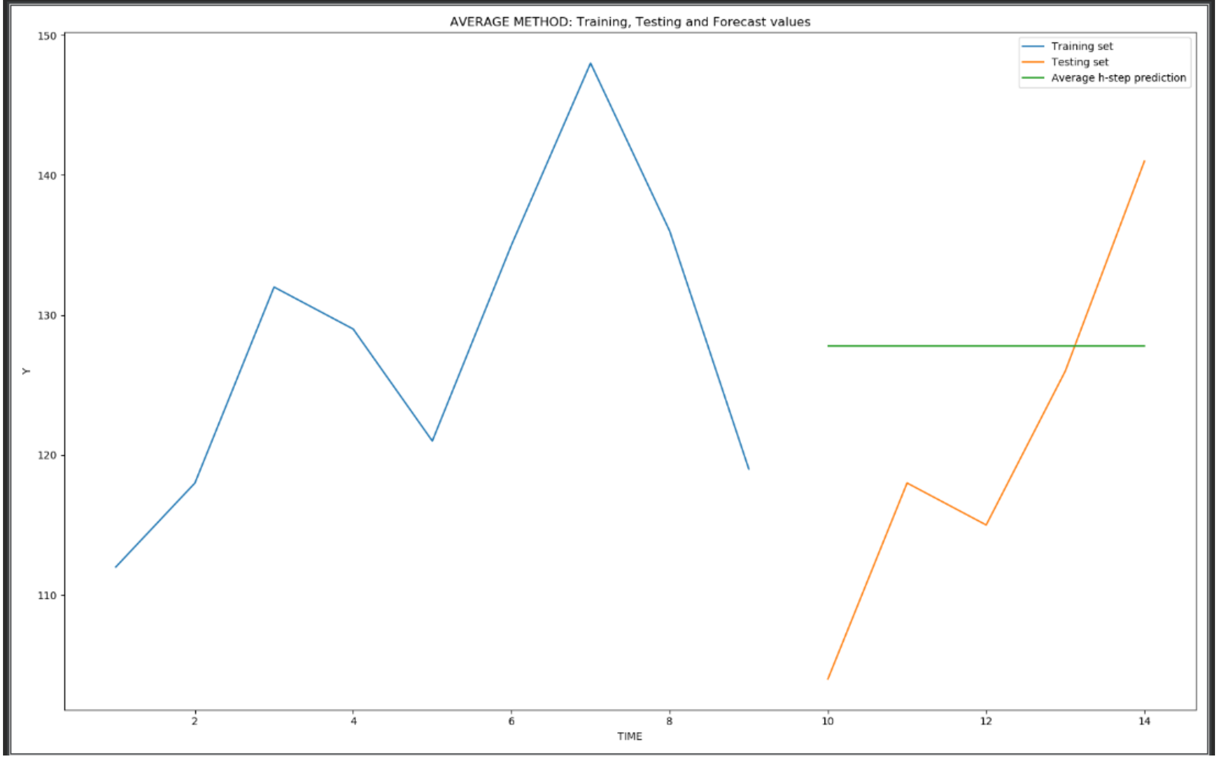
h	y_{t+h}	$\hat{y}_{t+h\|t}$	e	e^2
1	104	128.17	-24.17	584.19
2	118	128.17	-10.17	103.43
3	115	128.17	-13.17	173.45
4	126	128.17	-2.17	4.71
5	141	128.17	12.83	164.61

$\cancel{\hat{y}_{10+1} = (0.5)(119) + (1-0.5)}$

$$MSE_{hstep} = \frac{584.19 + 103.43 + 173.45 + 4.71 + 164.61}{5}$$

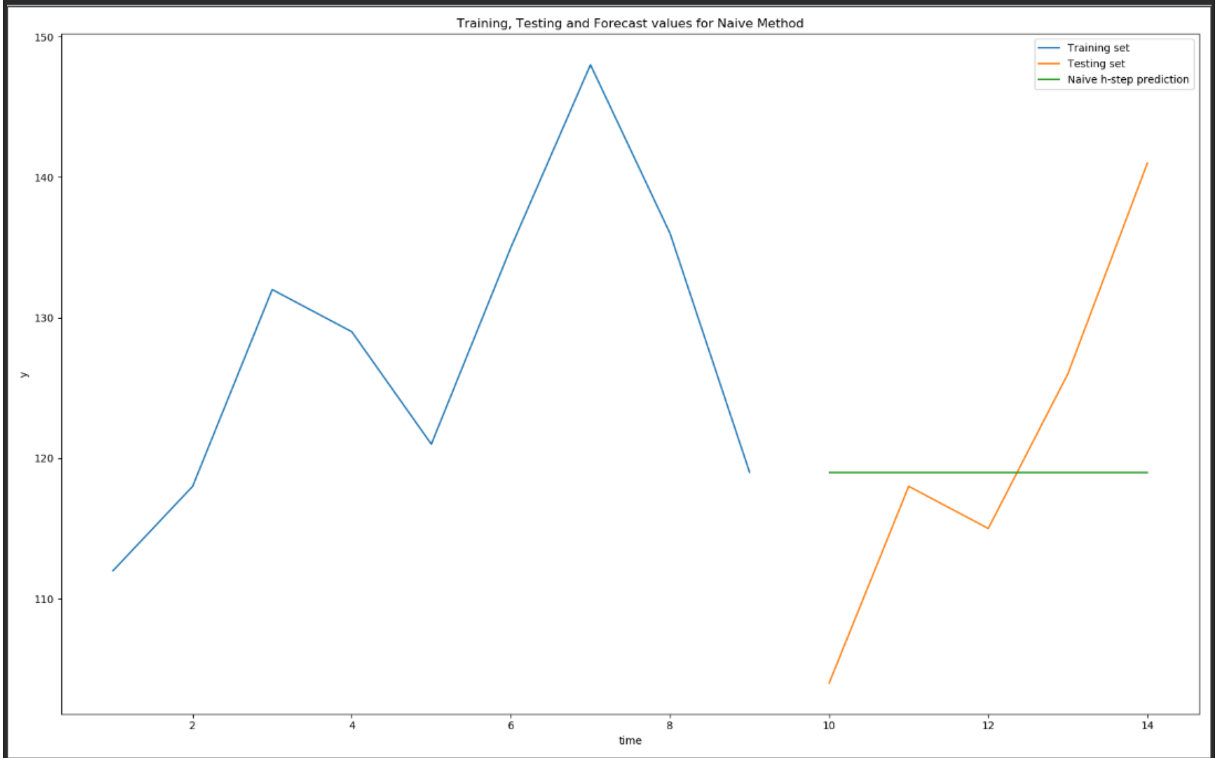
$$= 206.08$$

```
#%%=========================================================================
# 2: Write a python code that perform the task in step 1. Plot the test set,
training set and the h-step forecast in one graph with different marker/color.
# Add an appropriate title, legend, x-label, y-label to each graph.
# No need to include the 1-step prediction in this graph.
#%%=========================================================================
```

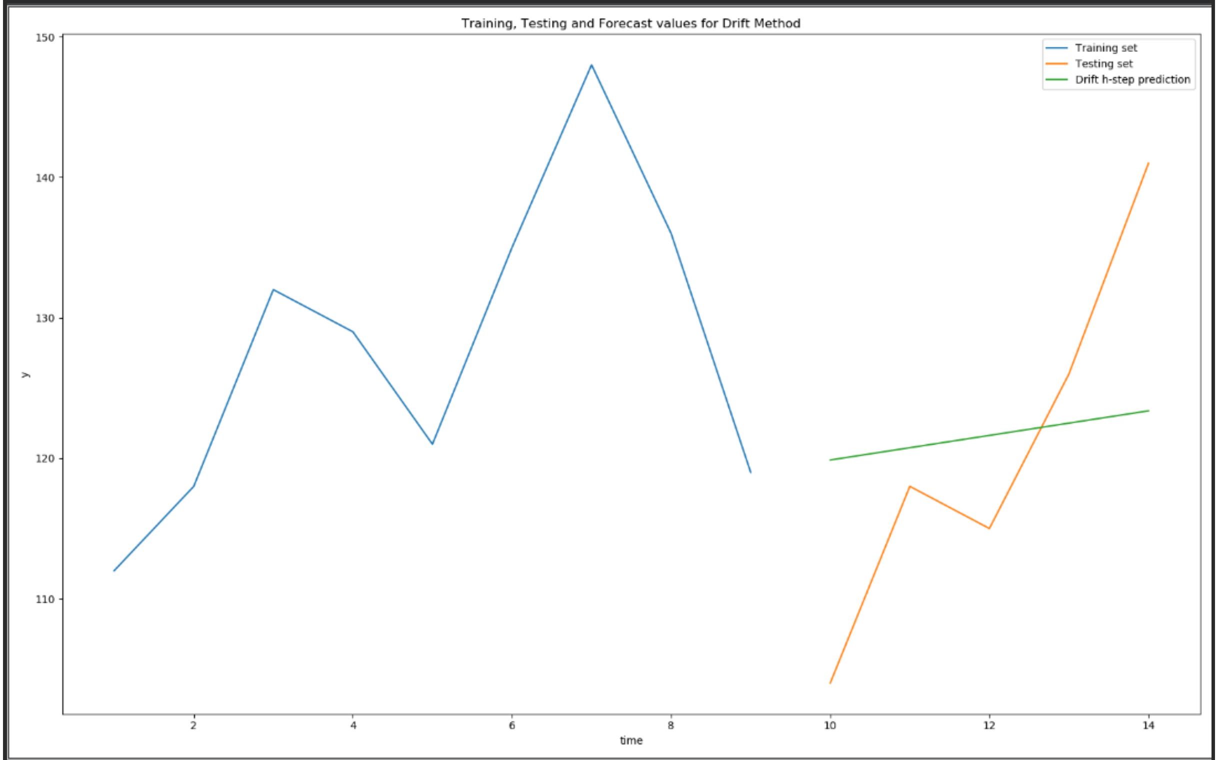



```
#%%=========================================================================
# 3: Using python, calculate the MSE of prediction errors and the forecast errors.
# Display the result on the console.
# %%-----------------------------------------------------------------------

Average method 1-step prediction:
[112.0, 115.0, 120.66666666666667, 122.75, 122.4, 124.5, 127.85714285714286,
128.875]
Average method h-step prediction:
[127.77777778 127.77777778 127.77777778 127.77777778 127.77777778]

Mean Square Error of prediction errors for Average method:  159.04233648667798
Mean Square Error of forecast errors for Average method:  200.44938271604934


#%%=========================================================================
# 4: Using python, calculate the variance of prediction error and the variance of
forecast error. Display the result.
# %%-----------------------------------------------------------------------

Mean of prediction errors for Average method:  7.993898809523808
Variance of prediction errors for Average method:  95.13991830977182
Variance of forecast errors for Average method:  151.76
```

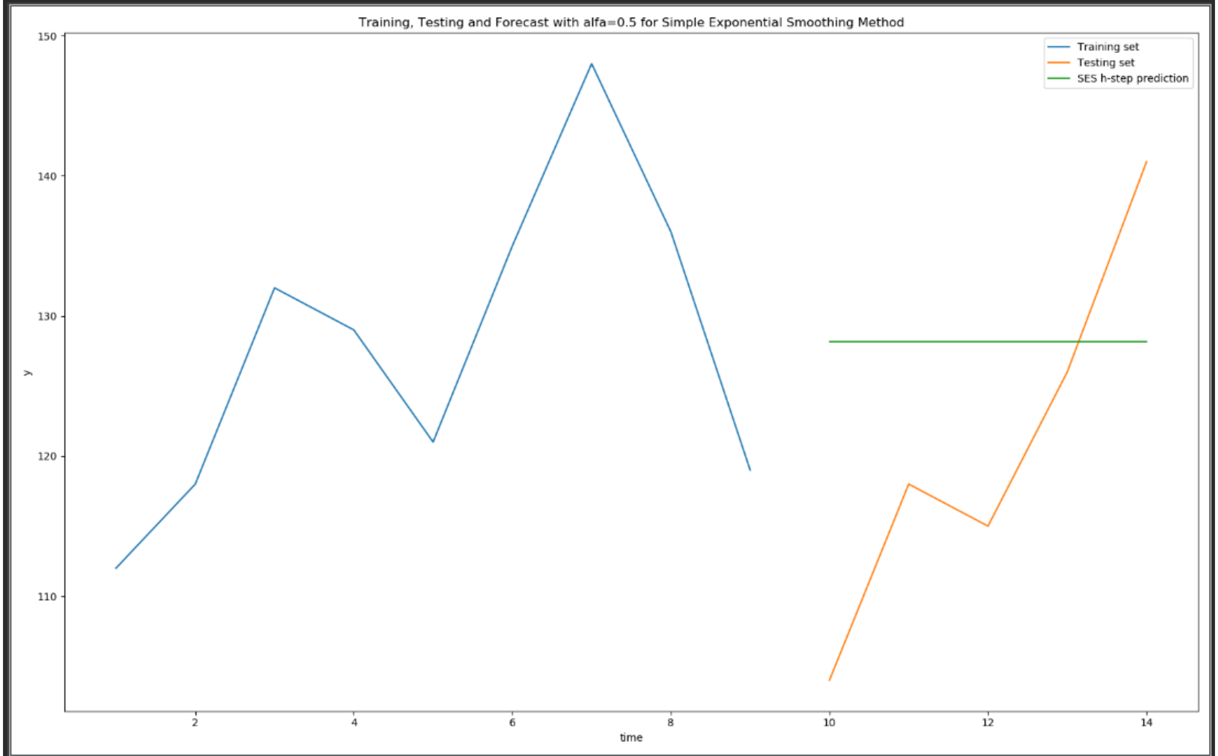
```
Average Autocorrelation Function:
[0.046811024090599146, -0.21182764494224582, -0.04682758913462302,
0.4002377834160776, -0.023101138293326388, -0.6757791485049354,
0.010486713368453861, 1.0, 0.010486713368453861, -0.6757791485049354, -
0.023101138293326388, 0.4002377834160776, -0.04682758913462302, -
0.21182764494224582, 0.046811024090599146]


#%%===========================================================================
# 5: Calculate the Q value for this estimate and display the Q-value on the
console. (# of lags = 8)
# %%--------------------------------------------------------------------------

Q value for Average method:  6.000897787361459

#%%===========================================================================
# 6: Repeat step 1 through 5 with the Naïve method.
# %%--------------------------------------------------------------------------

Naive method 1-step prediction:  [112, 118, 132, 129, 121, 135, 148, 136]
Naive method h-step prediction:  [119. 119. 119. 119. 119.]

Mean Square Error of prediction errors for Naive method:  137.875
Mean Square Error of forecast errors for Naive method:  155.0
Mean of prediction errors for Naive method:  0.875

Variance of prediction errors for Naive method:  137.109375
Variance of forecast errors for Naive method:  151.76

Q value for Naive method:  5.85083686901892
```



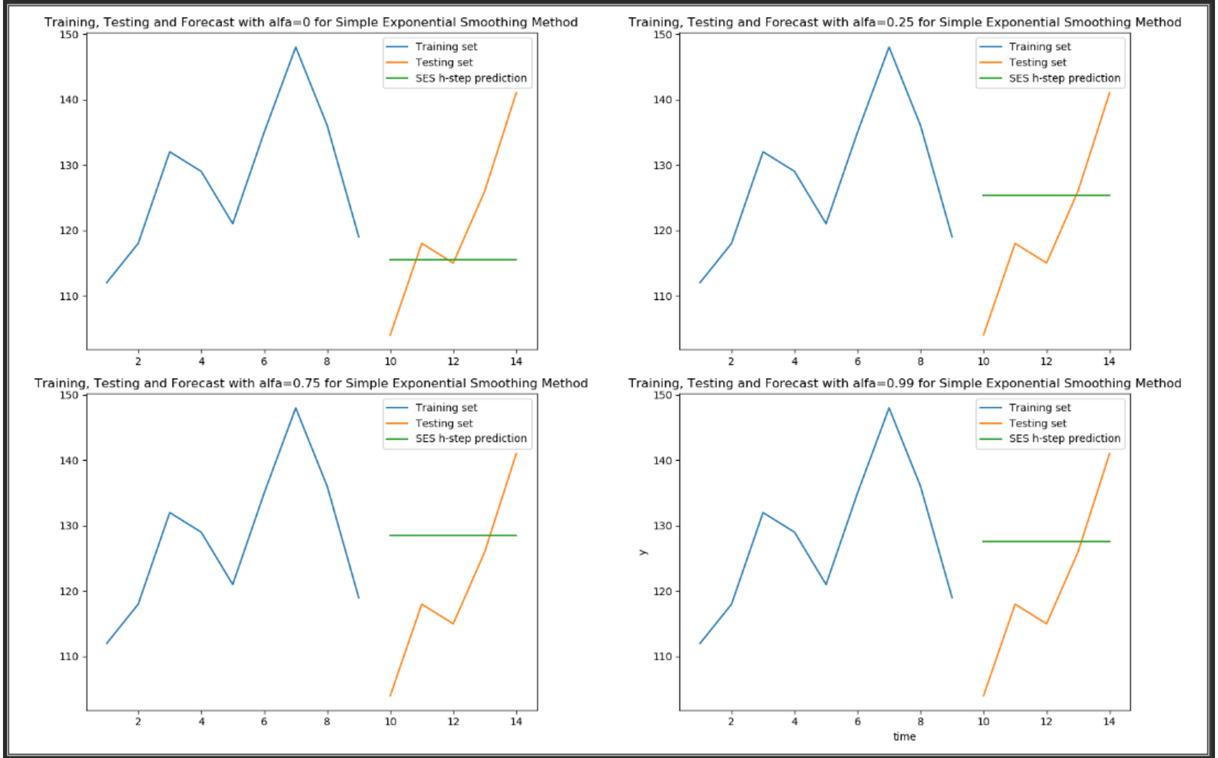
```
#%%===========================================================================
# 7: Repeat step 1 through 5 with the drift method.
# %%--------------------------------------------------------------------------

Drift method 1-step prediction:
[112, 124.0, 142.0, 134.66666666666666, 123.25, 139.6, 154.0, 139.42857142857142]
Drift method h-step prediction:
[119.875, 120.75, 121.625, 122.5, 123.375]

Mean Square Error of prediction errors for Drift method:  175.71585104875277
Mean Square Error of forecast errors for Drift method:  125.271875
Mean of prediction errors for Drift method:  -3.8681547619047585

Variance of prediction errors for Drift method:  160.7532297867063
Variance of forecast errors for Drift method:  124.59124999999999

Q value for Drift method:  7.385119910860086
```



```
#%%=================================================================================
#8: Repeat step 1 through 5 with the simple exponential method.
# Consider alfa = 0.5 and the initial condition to be the first sample in the
training set.
# %%--------------------------------------------------------------------------------

SES method 1-step prediction with damping factor 0.5:
[112, 115.0, 123.5, 126.25, 123.625, 129.3125, 138.65625, 137.328125]
SES method h-step prediction with damping factor 0.5:
[128.1640625 128.1640625 128.1640625 128.1640625 128.1640625]

Mean Square Error of prediction errors for SES method:  150.55020141601562
Mean Square Error of forecast errors for SES method:  205.98941650390626
Mean of prediction errors for SES method:  4.041015625

Variance of prediction errors for SES method:  134.22039413452148
Variance of forecast errors for SES method:  151.76

Q value for SES method:  5.172049247582443
```



```
#%%=================================================================================
#9: Using SES method, plot the test set, training set and the h-step forecast in
one graph for alfa = 0, 0.25, 0.75 and 0.99. You can use a subplot 2x2.
# Add an appropriate title, legend, x-label, y-label to each graph.
# No need to include the 1-step prediction in this graph
# %%---------------------------------------------------------------------------------
```



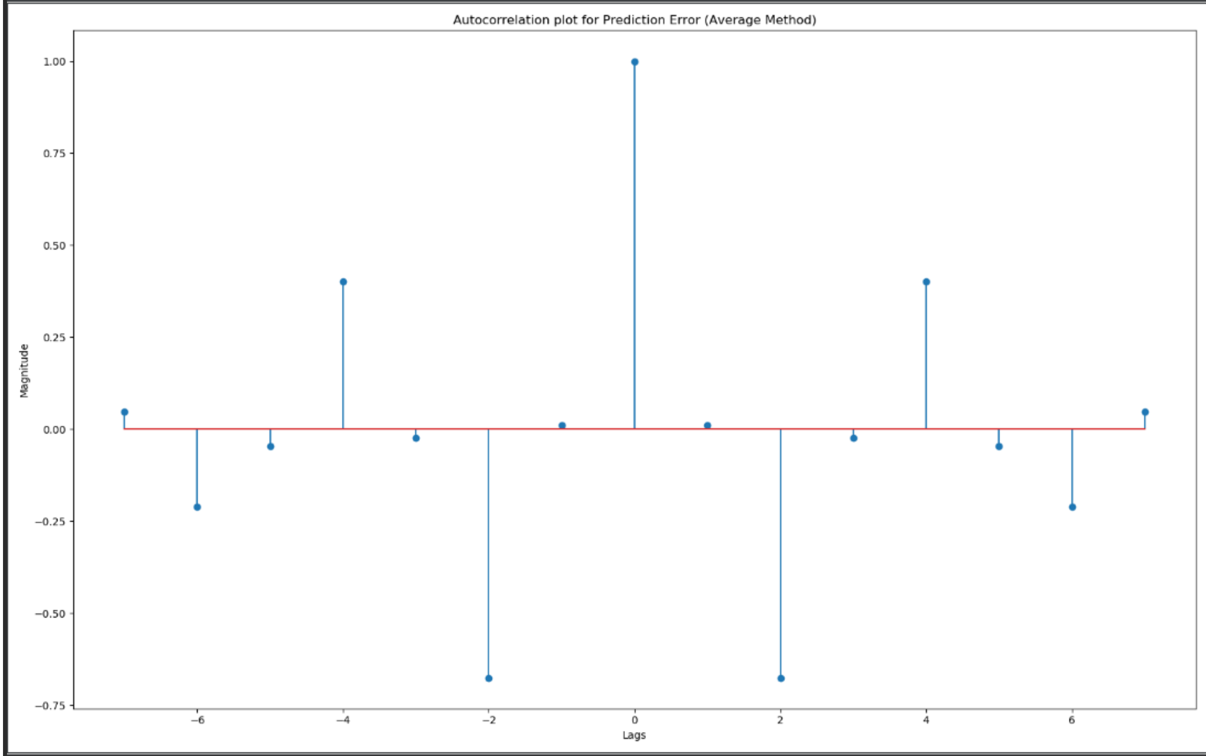

```
#%%=================================================================================
#10: Create a table and compare the four forecast method above by displaying, Q
values, MSE, mean of prediction errors, variance of prediction errors.
# %%---------------------------------------------------------------------------------
```

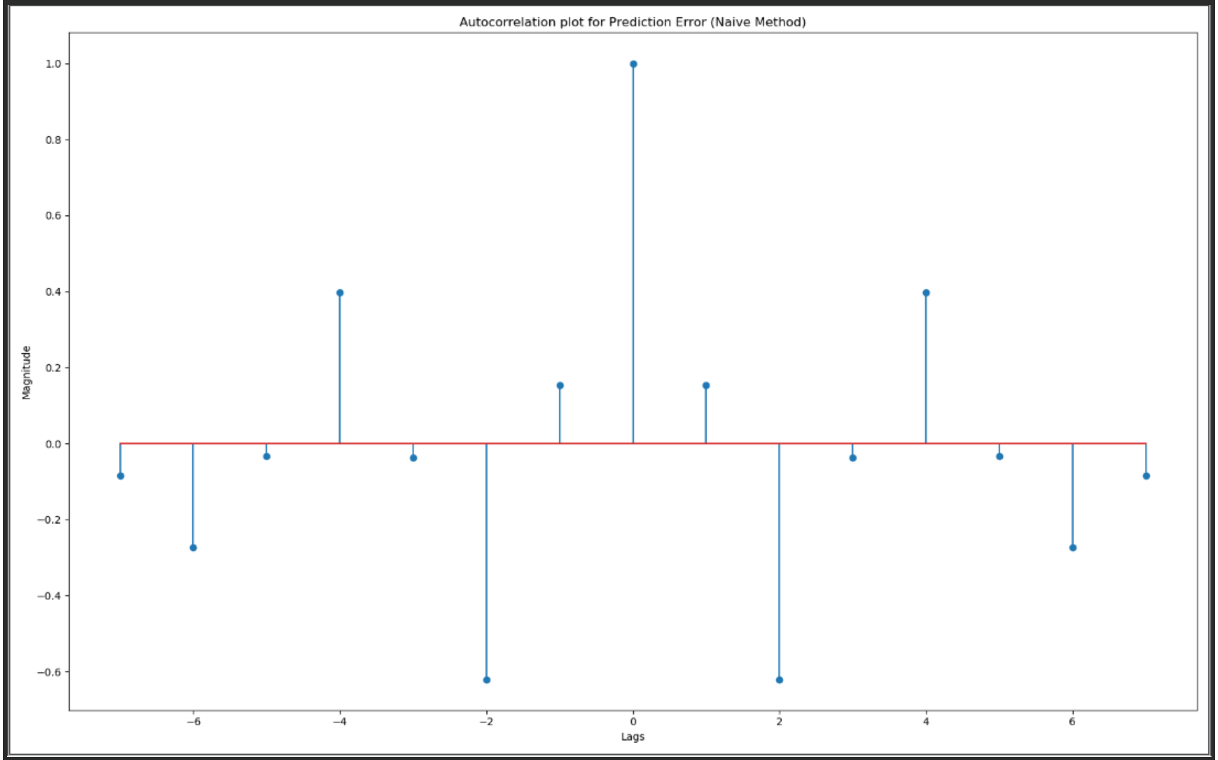
-------------------COMPARISION OF FORECAST METHODS-------------------

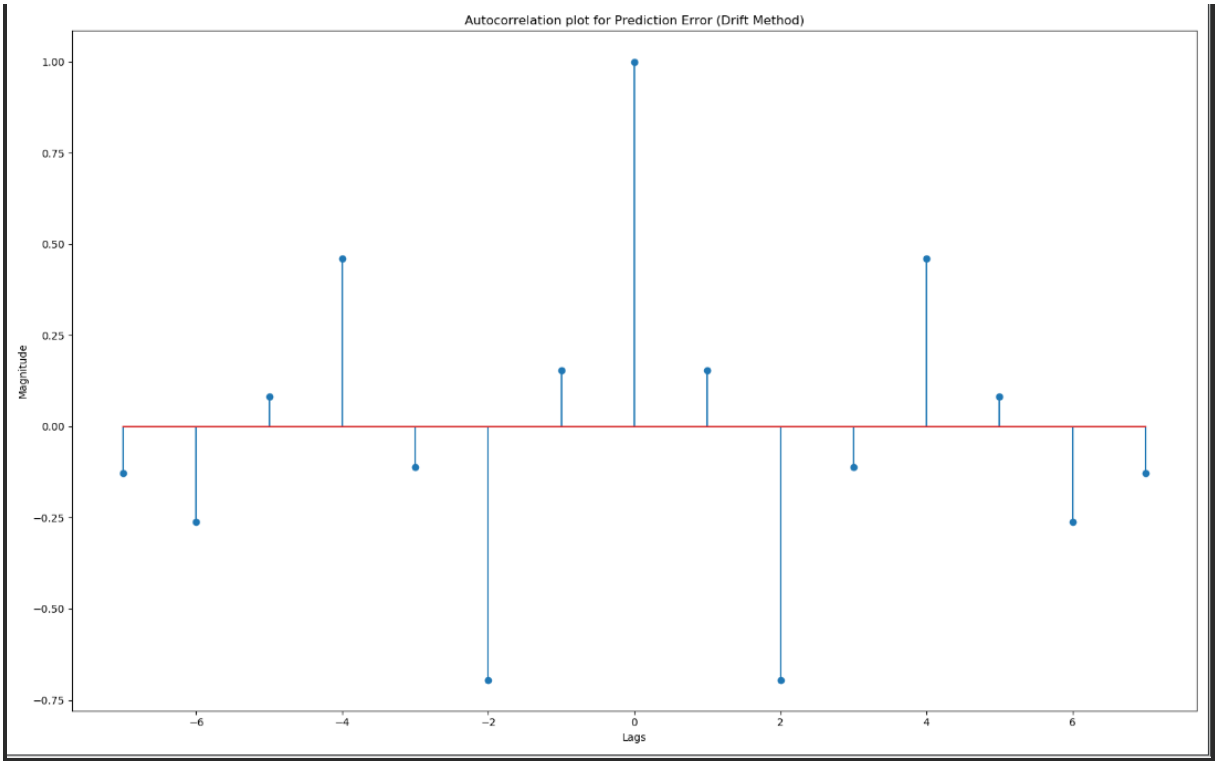
Method	Q_val	MSE_pred	MSE_forecast	Mean_pred
Average	6.000898	159.042336	200.449383	7.993899
Naive	5.850837	137.875000	155.000000	0.875000
Drift	7.385120	175.715851	125.271875	-3.868155
Simple Exponential Smoothing	5.172049	150.550201	205.989417	4.041016

Method	variance_pred	variance_forecast
Average	95.139918	151.76000
Naive	137.109375	151.76000
Drift	160.753230	124.59125
Simple Exponential Smoothing	134.220394	151.76000

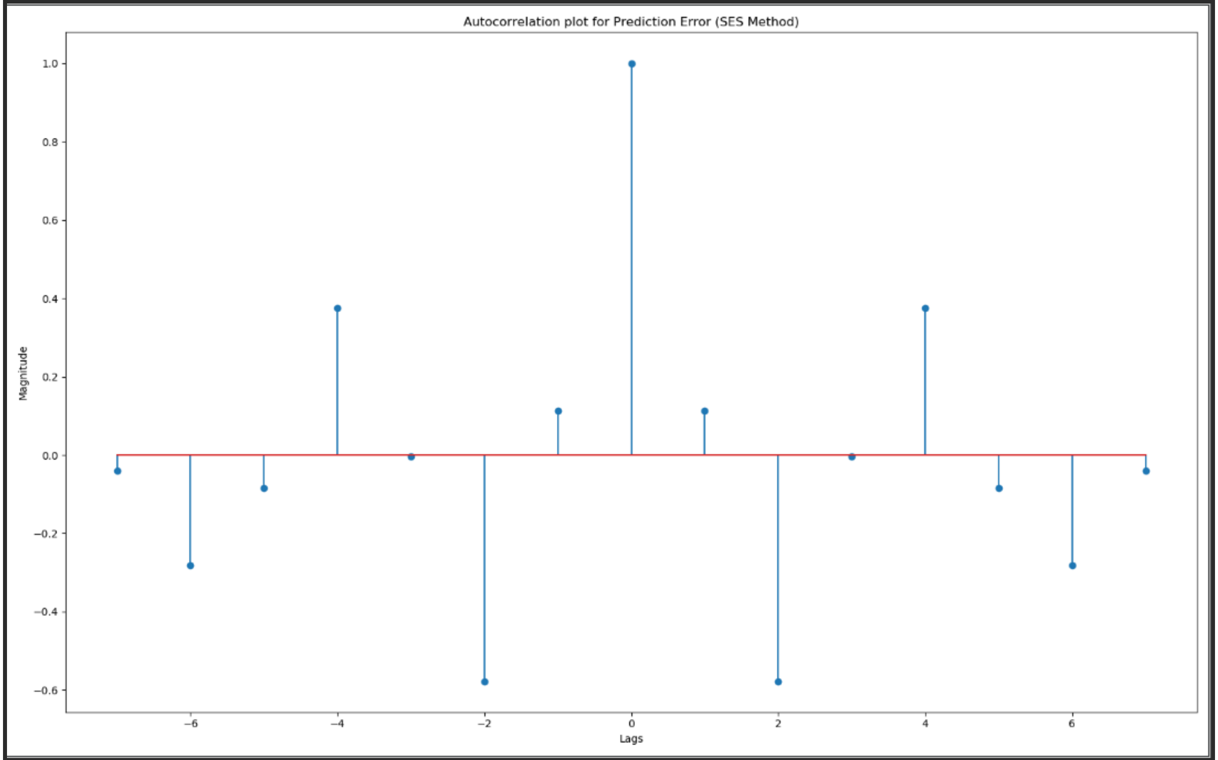
```
#%%=====================================================================
#11: Using the python program developed in the previous LAB, plot the ACF of
prediction errors.
# %%--------------------------------------------------------------------
```







Autocorrelation plot for Prediction Error (Drift Method)
Magnitude
Lags
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−6
−4
−2
0
2
4
6



Autocorrelation plot for Prediction Error (SES Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−6
−4
−2
0
2
4
6

```
#%%==============================================================================#
#12: Compare the above 4 methods by looking at the variance of prediction error
versus the variance of forecast error and pick the best estimator. Justify your
answer.
# %%------------------------------------------------------------------------------
```

Method	variance_pred	variance_forecast
Average	95.139918	151.76000
Naive	137.109375	151.76000
Drift	160.753230	124.59125
Simple Exponential Smoothing	134.220394	151.76000

Here the best estimator would be Naïve method since the difference between the variance of prediction error and the variance of forecast error is minimal.

CONCLUSION

For the data set provided different forecast methods such as Average , Naïve, Drift and simple exponential smoothing residuals was calculated along with forecast accuracy. we performed one-step-ahead prediction and h-step prediction. With the help of simple datasets, we plotted various time series plots and made a comparison with respect to forecast accuracy. Below table summarizes the different forecast methods and their forecast accuracy. From the table we can infer that the best estimator would be Naïve method since the difference between the variance of prediction error and the variance of forecast error is minimal.

Method	Q-Value	MSE-Prediction	MSE-Forecast	Mean-Preditcion	Variance-Prediction	Variance-Forecast
Average	6.000898	159.042336	200.449383	7.993899	95.139918	151.76
Naive	5.850837	137.875	155	0.875	137.109375	151.76
Drift	7.38512	175.715851	125.271875	-3.868155	160.75323	124.59125
Simple Exponential Smoothing	5.172049	150.550201	205.989417	4.041016	134.220394	151.76

CHALLENGE

Calculations was little tricky to understand in the beginning, after lot of clarifications it provided clarity.

APPENDIX

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from Autocorrelation import *

yt = [112, 118, 132, 129, 121, 135, 148, 136, 119]
yf = [104, 118, 115, 126, 141]


print('\n')
print(20 * "-" + "FORECAST METHOD| AVERAGE" + 20 * "-")
def avg_method(yt):
    return np.mean(yt)

yhat1 = []
for i in range(1,len(yt)):
    res = avg_method(yt[0:i])
    yhat1.append(res)
print("Average method 1-step prediction: ", yhat1)

y_fr = np.ones(len(yf)) * (np.mean(yt))
print("Average method h-step prediction: ", y_fr)

plt.figure(figsize=(16,10))
plt.plot(range(1, len(yt)+1), yt, label='Training set')
plt.plot(range(len(yt)+1, (len(yt)+1) + len(yf)), yf, label='Testing set')
plt.plot(range(len(yt)+1, (len(yt)+1) + len(y_fr)), y_fr, label='Average h-step
prediction')
plt.xlabel('TIME')
plt.ylabel('Y')
plt.title('AVERAGE METHOD: Training, Testing and Forecast values')
plt.legend()
plt.show()

residual_error_avg = np.array(yt[1:]) - np.array(yhat1)
forecast_error_avg = yf - y_fr
MSE_train_avg = np.mean((residual_error_avg)**2)
MSE_test_avg = np.mean((forecast_error_avg)**2)

print('Mean Square Error of prediction errors for Average method: ',
MSE_train_avg)
print('Mean Square Error of forecast errors for Average method: ', MSE_test_avg)

mean_pred_avg = np.mean(residual_error_avg)
var_pred_avg = np.var(residual_error_avg)
var_forecast_avg = np.var(forecast_error_avg)
print('Mean of prediction errors for Average method: ', mean_pred_avg)
print('Variance of prediction errors for Average method: ', var_pred_avg)
print('Variance of forecast errors for Average method: ', var_forecast_avg)


k = len(yt)
lags = len(residual_error_avg)
avg_acf = cal_auto_corr(residual_error_avg, lags)
print(avg_acf)
```

```
Q_avg = k * np.sum(np.array(avg_acf[8:])**2)
print('Q value for Average method: ', Q_avg)
print('\n')

print(20 * "-" + "FORECAST METHOD| NAIVE" + 20 * "-")
def naive_method(yt):
    return yt

yhat2 = []
for i in range(0, len(yt)-1):
    res = naive_method(yt[i])
    yhat2.append(res)

print("Naive method 1-step prediction: ", yhat2)
y_fr = np.ones(len(yf)) * yt[-1]
print("Naive method h-step prediction: ", y_fr)


residual_error_naive = np.array(yt[1:]) - np.array(yhat2)
forecast_error_naive = yf - y_fr
MSE_train_naive = np.mean((residual_error_naive)**2)
MSE_test_naive = np.mean((forecast_error_naive)**2)

print('Mean Square Error of prediction errors for Naive method: ',
MSE_train_naive)
print('Mean Square Error of forecast errors for Naive method: ', MSE_test_naive)


mean_pred_naive = np.mean(residual_error_naive)
var_pred_naive = np.var(residual_error_naive)
var_forecast_naive = np.var(forecast_error_naive)
print('Mean of prediction errors for Naive method: ', mean_pred_naive)
print('Variance of prediction errors for Naive method: ', var_pred_naive)
print('Variance of forecast errors for Naive method: ', var_forecast_naive)

plt.figure(figsize=(16,10))
plt.plot(range(1, len(yt)+1), yt, label='Training set')
plt.plot(range(len(yt)+1, (len(yt)+1) + len(yf)), yf, label='Testing set')
plt.plot(range(len(yt)+1, (len(yt)+1) + len(y_fr)), y_fr, label='Naive h-step
prediction')
plt.xlabel('time')
plt.ylabel('y')
plt.title('Training, Testing and Forecast values for Naive Method')
plt.legend()
plt.show()

k = len(yt)
lags = len(residual_error_naive)
naive_acf = cal_auto_corr(residual_error_naive, lags)
Q_naive = k * np.sum(np.array(naive_acf[8:])**2)
print('Q value for Naive method: ', Q_naive)
print('\n')

print(20 * "-" + "FORECAST METHOD| DRIFT" + 20 * "-")
yhat3 = []
def drift_method(t, h):
    res = t[len(t)-1] + h*((t[len(t)-1]-t[0])/(len(t) - 1))
    return res
```

```
for i in range(1, len(yt)):
    if i==1:
        yhat3.append(yt[0])
    else:
        h = 1
        res = drift_method(yt[0:i], h)
        yhat3.append(res)

print("Drift method 1-step prediction: ", yhat3)

y_fr = []
for h in range(1, len(yf)+1):
    res = drift_method(yt,h)
    y_fr.append(res)
print("Drift method h-step prediction: ", y_fr)
residual_error_drift = np.array(yt[1:]) - np.array(yhat3)
forecast_error_drift = np.array(yf) - np.array(y_fr)
MSE_train_drift = np.mean((residual_error_drift)**2)
MSE_test_drift = np.mean((forecast_error_drift)**2)

print('Mean Square Error of prediction errors for Drift method: ',
MSE_train_drift)
print('Mean Square Error of forecast errors for Drift method: ', MSE_test_drift)

mean_pred_drift = np.mean(residual_error_drift)
var_pred_drift = np.var(residual_error_drift)
var_forecast_drift = np.var(forecast_error_drift)
print('Mean of prediction errors for Drift method: ', mean_pred_drift)
print('Variance of prediction errors for Drift method: ', var_pred_drift)
print('Variance of forecast errors for Drift method: ', var_forecast_drift)

plt.figure(figsize=(16,10))
plt.plot(range(1, len(yt)+1), yt, label='Training set')
plt.plot(range(len(yt)+1, (len(yt)+1) + len(yf)), yf, label='Testing set')
plt.plot(range(len(yt)+1, (len(yt)+1) + len(y_fr)), y_fr, label='Drift h-step
prediction')
plt.xlabel('time')
plt.ylabel('y')
plt.title('Training, Testing and Forecast values for Drift Method')
plt.legend()
plt.show()

k = len(yt)
lags = len(residual_error_drift)
drift_acf = cal_auto_corr(residual_error_drift, lags)
Q_drift = k * np.sum(np.array(drift_acf[8:])**2)
print('Q value for Drift method: ', Q_drift)
print('\n')

print(20 * "-" + "FORECAST METHOD| SIMPLE EXPONENTIAL METHOD" + 20 * "-")

def ses(t, damping_factor, l0):
    yhat4 = []
    yhat4.append(l0)
    for i in range(1, len(t)-1):
        res = damping_factor*(t[i]) + (1-damping_factor)*(yhat4[i-1])
        yhat4.append(res)
```

```
    return yhat4

l0 = yt[0]
ses_0 = ses(yt, 0, l0)
ses_25 = ses(yt, 0.25, l0)
ses_50 = ses(yt, 0.50, l0)
ses_75 = ses(yt, 0.75, l0)
ses_99 = ses(yt, 0.99, l0)

print("SES method 1-step prediction with damping factor 0.5: ", ses_50)
ses_fr_50 = np.ones(len(yf)) * (0.5*(yt[-1]) + (1-0.5)*(ses_50[-1]))
print("SES method h-step prediction with damping factor 0.5: ", ses_fr_50)

residual_error_ses = np.array(yt[1:]) - np.array(ses_50)
forecast_error_ses = np.array(yf) - np.array(ses_fr_50)
MSE_train_SES = np.mean((residual_error_ses)**2)
MSE_test_SES = np.mean((forecast_error_ses)**2)

print('Mean Square Error of prediction errors for SES method: ', MSE_train_SES)
print('Mean Square Error of forecast errors for SES method: ', MSE_test_SES)

mean_pred_SES = np.mean(residual_error_ses)
var_pred_SES = np.var(residual_error_ses)
var_forecast_SES = np.var(forecast_error_ses)
print('Mean of prediction errors for SES method: ', mean_pred_SES)
print('Variance of prediction errors for SES method: ', var_pred_SES)
print('Variance of forecast errors for SES method: ', var_forecast_SES)

plt.figure(figsize=(16,10))
plt.plot(range(1, len(yt)+1), yt, label='Training set')
plt.plot(range(len(yt)+1, (len(yt)+1) + len(yf)), yf, label='Testing set')
plt.plot(range(len(yt)+1, (len(yt)+1) + len(y_fr)), ses_fr_50, label='SES h-step
prediction')
plt.xlabel('time')
plt.ylabel('y')
plt.title('Training, Testing and Forecast with alfa=0.5 for Simple Exponential
Smoothing Method')
plt.legend()
plt.show()

k = len(yt)
lags = len(residual_error_ses)
ses_acf = cal_auto_corr(residual_error_ses, lags)
Q_SES = k * np.sum(np.array(ses_acf[8:])**2)
print('Q value for SES method: ', Q_SES)
print('\n')

ses_fr_0 = np.ones(len(yf)) * (0.5*(yt[-1]) + (1-0.5)*(ses_0[-1]))
ses_fr_25 = np.ones(len(yf)) * (0.5*(yt[-1]) + (1-0.5)*(ses_25[-1]))
ses_fr_75 = np.ones(len(yf)) * (0.5*(yt[-1]) + (1-0.5)*(ses_75[-1]))
ses_fr_99 = np.ones(len(yf)) * (0.5*(yt[-1]) + (1-0.5)*(ses_99[-1]))

fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(16, 10))
ax[0,0].plot(range(1, len(yt)+1), yt, label='Training set')
ax[0,0].plot(range(len(yt)+1, (len(yt)+1) + len(yf)), yf, label='Testing set')
ax[0,0].plot(range(len(yt)+1, (len(yt)+1) + len(y_fr)), ses_fr_0, label='SES h-
step prediction')
ax[0,0].set_title('Training, Testing and Forecast with alfa=0 for Simple
```

```
Exponential Smoothing Method')
ax[0,1].plot(range(1, len(yt)+1), yt, label='Training set')
ax[0,1].plot(range(len(yt)+1, (len(yt)+1) + len(yf)), yf, label='Testing set')
ax[0,1].plot(range(len(yt)+1, (len(yt)+1) + len(y_fr)), ses_fr_25, label='SES h-
step prediction')
ax[0,1].set_title('Training, Testing and Forecast with alfa=0.25 for Simple
Exponential Smoothing Method')
ax[1,0].plot(range(1, len(yt)+1), yt, label='Training set')
ax[1,0].plot(range(len(yt)+1, (len(yt)+1) + len(yf)), yf, label='Testing set')
ax[1,0].plot(range(len(yt)+1, (len(yt)+1) + len(y_fr)), ses_fr_75, label='SES h-
step prediction')
ax[1,0].set_title('Training, Testing and Forecast with alfa=0.75 for Simple
Exponential Smoothing Method')
ax[1,1].plot(range(1, len(yt)+1), yt, label='Training set')
ax[1,1].plot(range(len(yt)+1, (len(yt)+1) + len(yf)), yf, label='Testing set')
ax[1,1].plot(range(len(yt)+1, (len(yt)+1) + len(y_fr)), ses_fr_99, label='SES h-
step prediction')
ax[1,1].set_title('Training, Testing and Forecast with alfa=0.99 for Simple
Exponential Smoothing Method')
plt.xlabel('time')
plt.ylabel('y')
ax[0,0].legend(loc="upper right")
ax[0,1].legend(loc="upper right")
ax[1,0].legend(loc="upper right")
ax[1,1].legend(loc="upper right")
plt.show()


print(20 * "-" + "COMPARISION OF FORECAST METHODS" + 20 * "-")
d = {'Method':['Average', 'Naive', 'Drift', 'Simple Exponential Smoothing'],
     'Q_val': [Q_avg, Q_naive, Q_drift, Q_SES],
     'MSE_pred': [MSE_train_avg, MSE_train_naive, MSE_train_drift, MSE_train_SES],
     'MSE_forecast': [MSE_test_avg, MSE_test_naive, MSE_test_drift, MSE_test_SES],
     'Mean_pred': [mean_pred_avg, mean_pred_naive, mean_pred_drift,
mean_pred_SES],
     'variance_pred': [var_pred_avg, var_pred_naive, var_pred_drift,
var_pred_SES],
     'variance_forecast':[var_forecast_avg, var_forecast_naive,
var_forecast_drift, var_forecast_SES]}
df = pd.DataFrame(data=d)
df = df.set_index('Method')
pd.set_option('display.max_columns', None)
print(df.head())


plt.figure(figsize=(16,10))
plt.stem(range(-(lags-1),lags), avg_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Prediction Error (Average Method)')
plt.show()

plt.figure(figsize=(16,10))
plt.stem(range(-(lags-1),lags), naive_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Prediction Error (Naive Method)')
plt.show()
```

```
plt.figure(figsize=(16,10))
plt.stem(range(-(lags-1),lags), drift_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Prediction Error (Drift Method)')
plt.show()

plt.figure(figsize=(16,10))
plt.stem(range(-(lags-1),lags), ses_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Prediction Error (SES Method)')
plt.show()


#Autocorrelation Function

import numpy as np

def auto_corr(y, k):
    T = len(y)
    y_mean = np.mean(y)
    res_num = 0
    res_den = 0
    for t in range(k, T):
        res_num += (y[t] - y_mean) * (y[t-k] - y_mean)

    for t in range(0, T):
        res_den += (y[t] - y_mean)**2

    result = res_num/res_den
    return result

def cal_auto_corr(y, k):
    res = []
    res1 = []
    for t in range(0, k):
        result = auto_corr(y, t)
        res.append(result)
    for t in range(k-1, 0, -1):
        res1.append(res[t])
    res1.extend(res)
    return res1
```

REFERENCES

https://otexts.com/fpp2/#