TIME SERIES MODELING & ANALYSIS

Instructor Name: Reza Jafari

HW#: 4

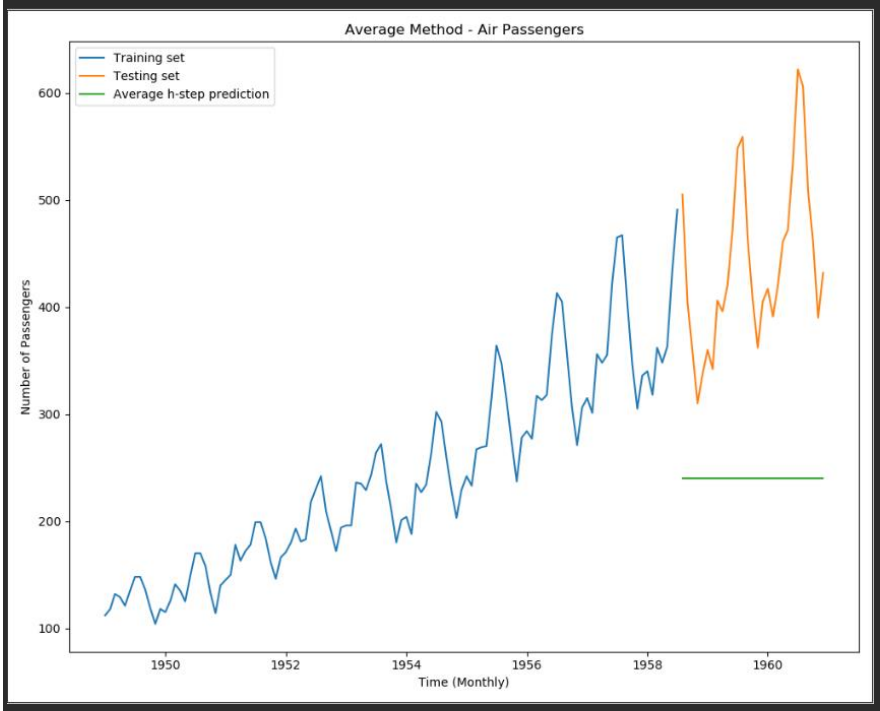
Submitted by: Dinesh Kumar Padmanabhan

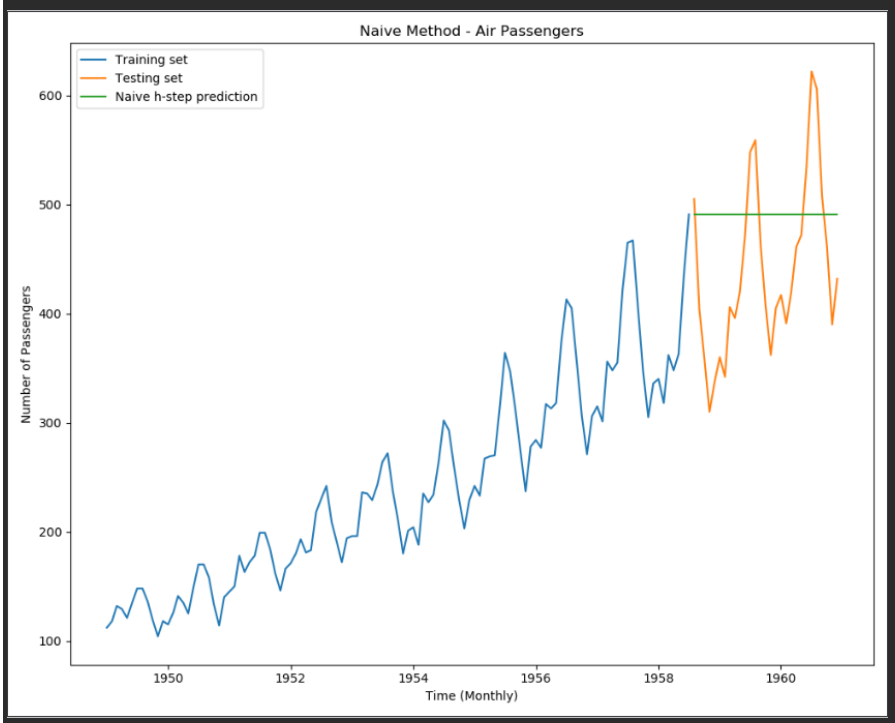
Date: 09-Oct-2020

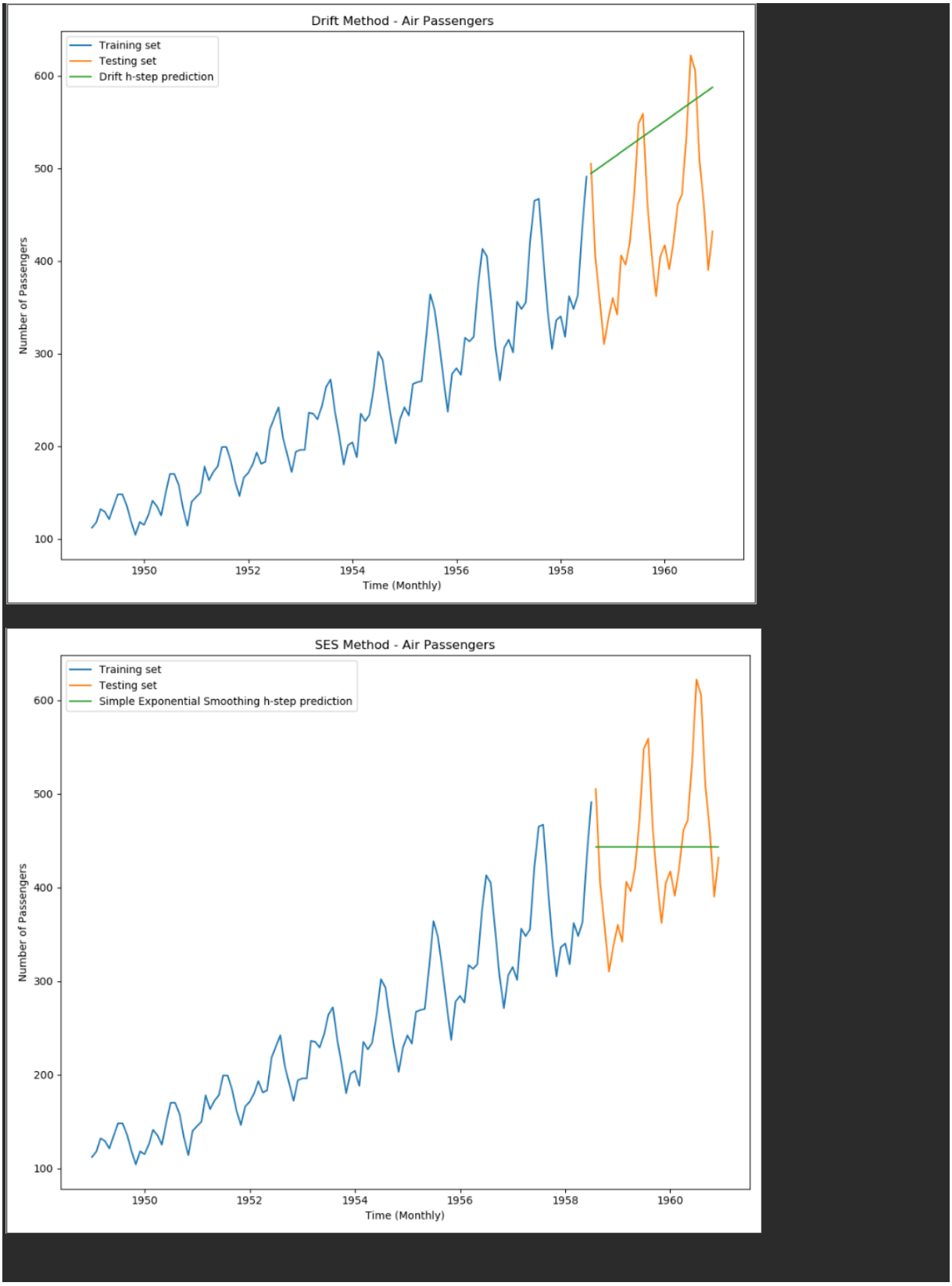
ANSWERS TO ASKED QUESTIONS

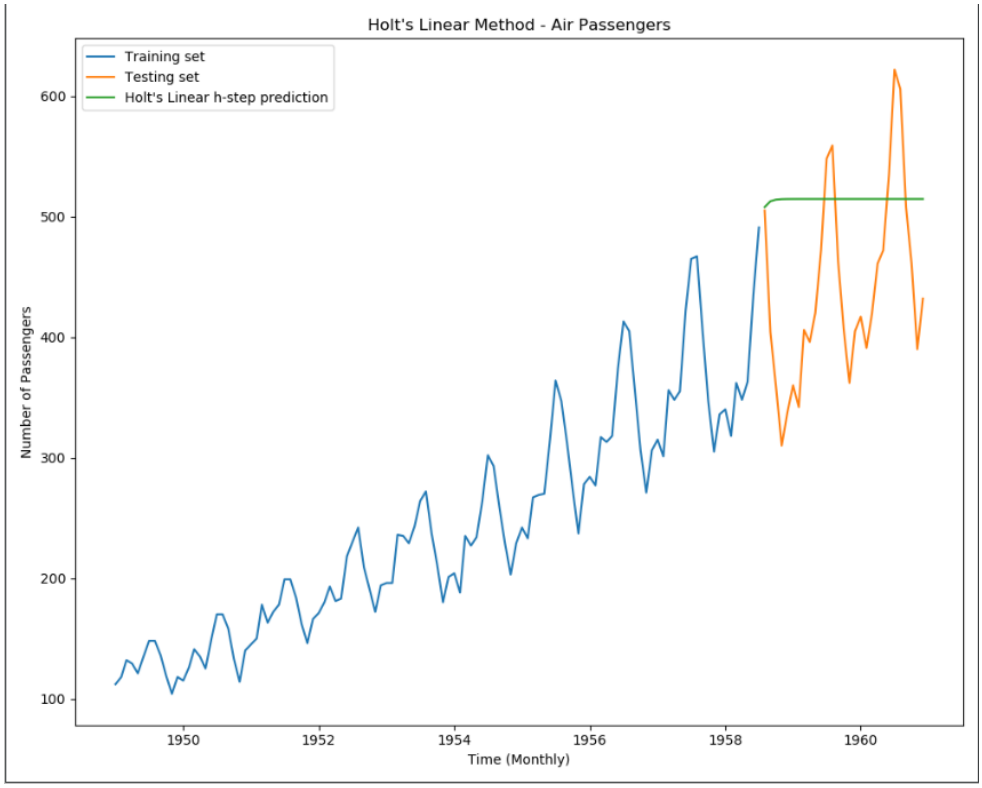
```
#%%========================================================================
#In this homework, you will use the simple and advanced forecast methods on 5
different dataset and
# compare the performance of 6 simple and advanced forecasting methods:
# 1. Average Method
# 2. Naïve Method
# 3. Drift Method
# 4. Simple Exponential Smoothing Method (alfa=0.5)
# 5. Holt's Linear Method
# 6. Holt-Winter Seasonal Method
# Develop a python program (using above libraries and the codes developed in the
previous LABs/HW) to
# perform the following tasks:
# Load the Sales values of dataset "AirPassengers.csv". Split the dataset into
training and testing sets (20%
# test). Use the following command from the 'sklearn' package:
# train_test_split(y, shuffle= False, test_size=0.2)
# %%------------------------------------------------------------------------


print('\n')
print(20 * "-" + "AIR PASSENGERS DATASET" + 20 * "-")
df = pd.read_csv('AirPassengers.csv', index_col='Month', parse_dates=True)
df.index.freq = 'MS'
y = df['#Passengers']
train, test = train_test_split(y, shuffle=False, test_size=0.2)
train.index.freq = 'MS'
test.index.freq = 'MS'
h = len(test)
```

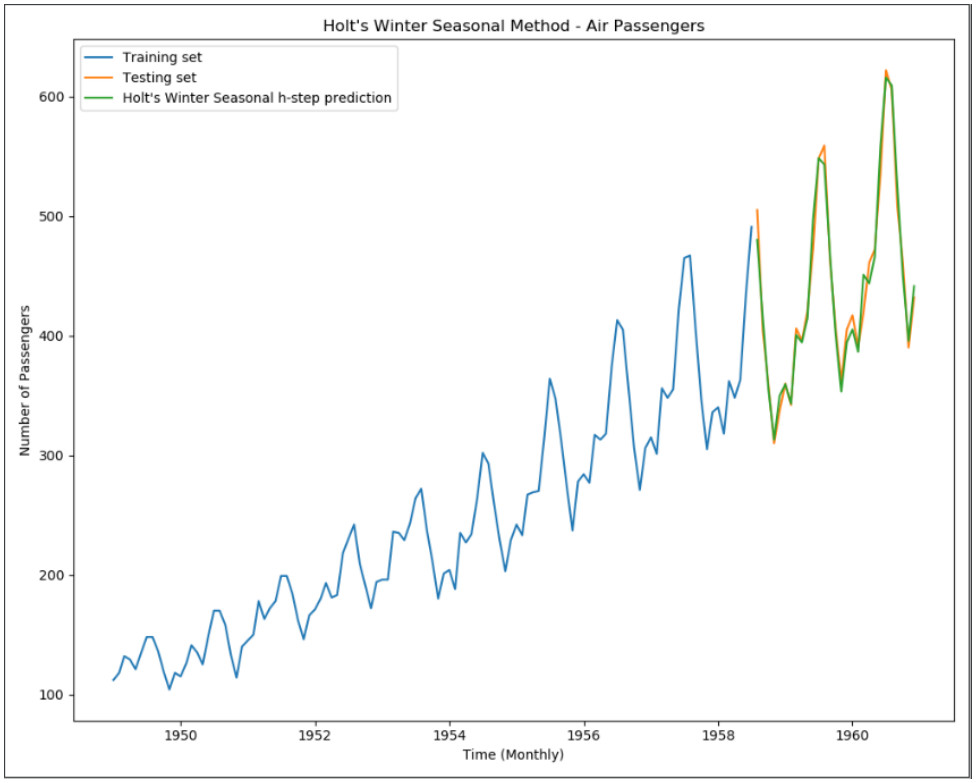
```
#%%=========================================================================================
#1.Using all 6 forecast methods above, perform h-step ahead prediction (h is the
length of test set).
#The goal is to compare the h-step ahead prediction versus the test set.

#2. Plot the training set, test set and multi-step ahead forecast in one graph.
Add appropriate legend, title, x-axis, and y-axis label.
```





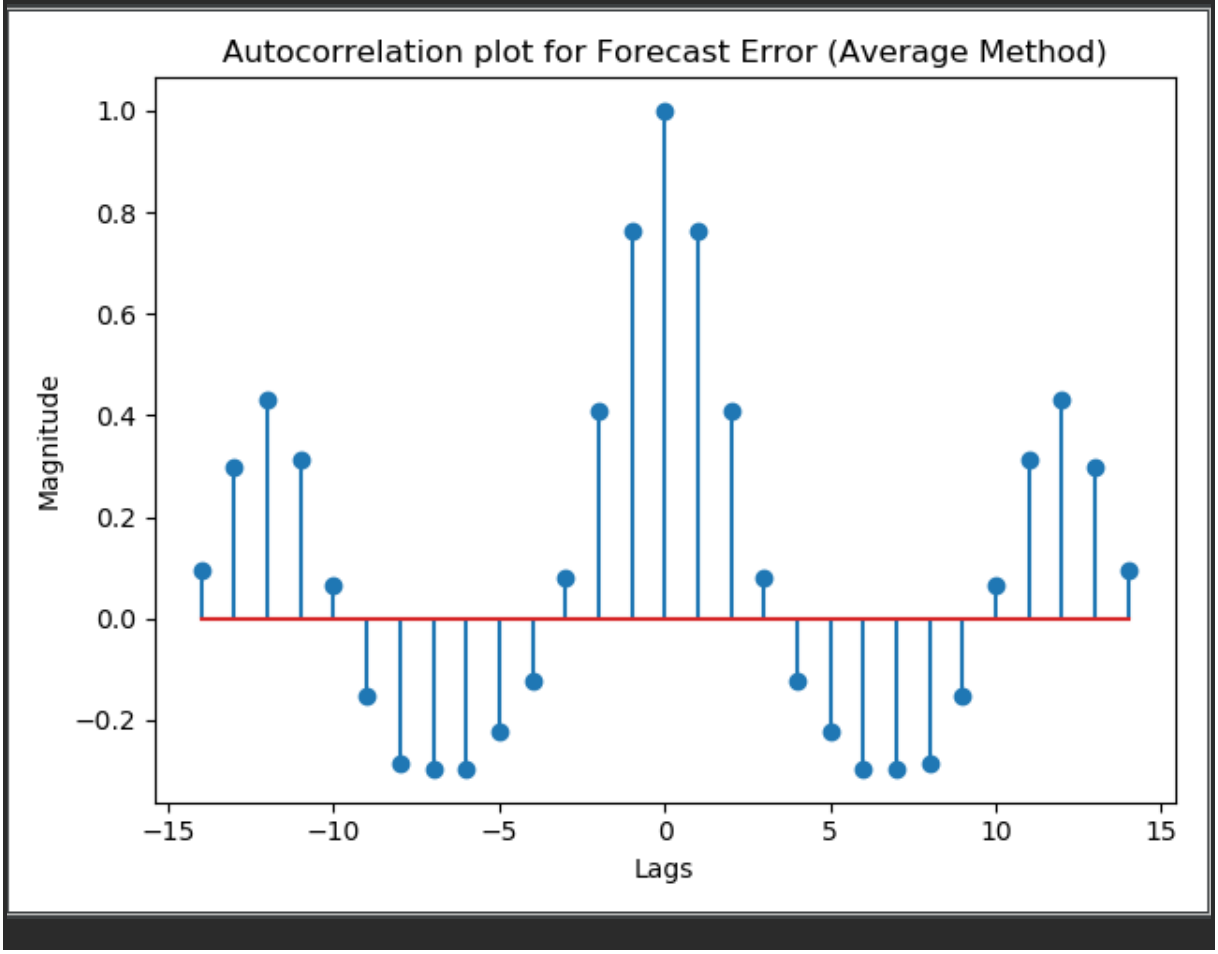

Drift Method - Air Passengers
Training set
Testing set
Drift h-step prediction
Number of Passengers
Time (Monthly)
100
200
300
400
500
600
1950
1952
1954
1956
1958
1960
SES Method - Air Passengers
Training set
Testing set
Simple Exponential Smoothing h-step prediction
Number of Passengers
Time (Monthly)
100
200
300
400
500
600
1950
1952
1954
1956
1958
1960

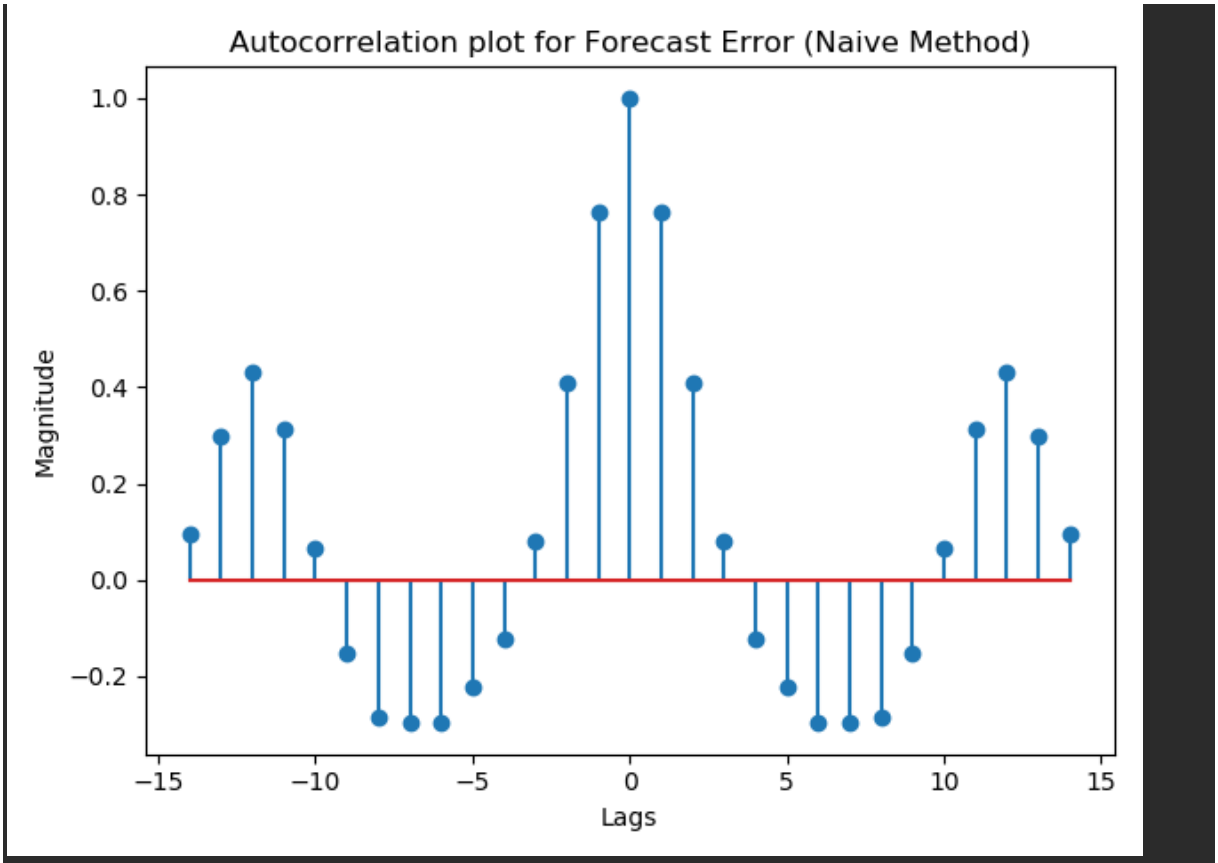

Holt's Linear Method - Air Passengers
Training set
Testing set
Holt's Linear h-step prediction
Number of Passengers
Time (Monthly)
100
200
300
400
500
600
1950
1952
1954
1956
1958
1960

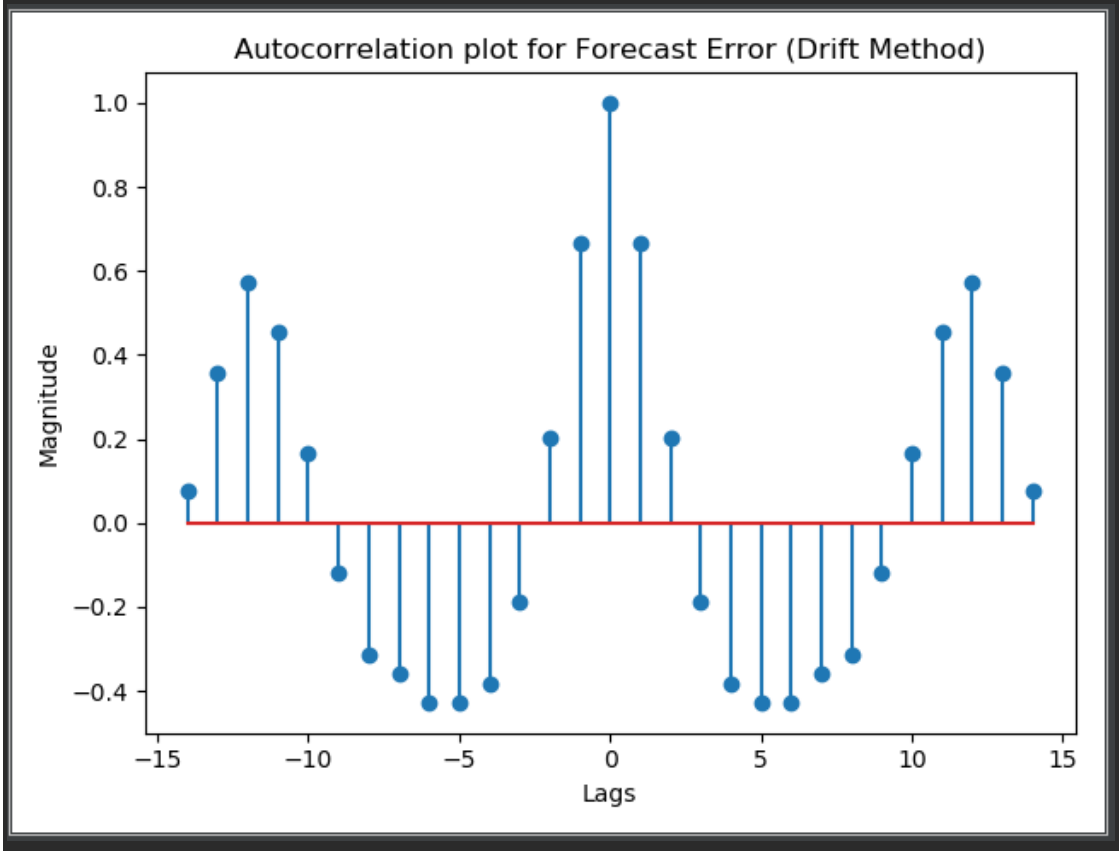

Holt's Winter Seasonal Method - Air Passengers
Training set
Testing set
Holt's Winter Seasonal h-step prediction
Number of Passengers
Time (Monthly)
100
200
300
400
500
600
1950
1952
1954
1956
1958
1960

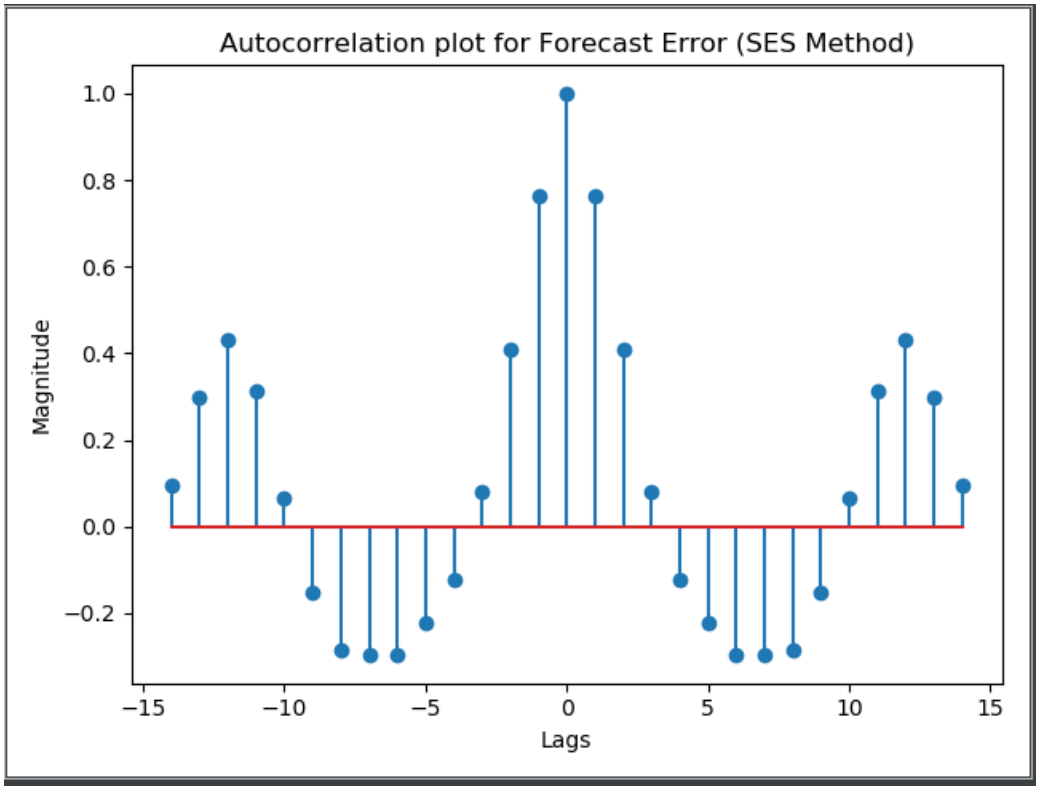
```
#%%=============================================================================
#3. Calculate mean square error (MSE) of the forecast error for all 6 forecast
methods.
#4. Display the variance of prediction error versus the forecast error on the
console for each method.
# Write down your observation.
# %%-----------------------------------------------------------------------------

--------------------FORECAST METHOD| AVERAGE--------------------
Mean Square Error of prediction errors for Average method:  8449.445394016202
Mean Square Error of forecast errors for Average method:  46249.62880907372
Mean of prediction errors for Average method:  69.75059564050484
Variance of prediction errors for Average method:  3584.299801810989
Variance of forecast errors for Average method:  6104.489892984543

--------------------FORECAST METHOD| NAIVE--------------------
Mean Square Error of prediction errors for Naive method:  724.5526315789474
Mean Square Error of forecast errors for Naive method:  8673.931034482759
Mean of prediction errors for Naive method:  3.324561403508772
Variance of prediction errors for Naive method:  713.4999230532471
Variance of forecast errors for Naive method:  6104.489892984539

--------------------FORECAST METHOD| DRIFT--------------------
Mean Square Error of prediction errors for Drift method:  727.0580499380247
Mean Square Error of forecast errors for Drift method:  15038.70328005434
Mean of prediction errors for Drift method:  1.0580363392144634
Variance of prediction errors for Drift method:  725.9386090429264
Variance of forecast errors for Drift method:  4926.776585972305

--------------------FORECAST METHOD| SIMPLE EXPONENTIAL SMOOTHING-----------------
Mean Square Error of prediction errors for SES method:  1089.417676638178
Mean Square Error of forecast errors for SES method:  6111.587940955658
Mean of prediction errors for SES method:  5.806571246133478
Variance of prediction errors for SES method:  1055.7014070017538
Variance of forecast errors for SES method:  6104.489892984539

--------------------HOLTS LINEAR TREND--------------------
Mean Square Error of prediction errors for Holt's Linear method:  660.61057016647
Mean Square Error of forecast errors for Holt's Linear method:  11601.005359574126
Mean of prediction errors for Holt's Linear method:  2.0439643336030606
Variance of prediction errors for Holt's Linear method:  656.4327799694288
Variance of forecast errors for Holt's Linear method:  6126.640445815883
--------------------HOLTS WINTER SEASONAL TREND--------------------
Mean Square Error of prediction errors for Holt's Winter Seasonal method:
83.62768650440252
Mean Square Error of forecast errors for Holt's Winter Seasonal method:
161.34788696610448
Mean of prediction errors for Holt's Winter Seasonal method:
-0.3466845219027313
Variance of prediction errors for Holt's Winter Seasonal method:
83.50749634667562
Variance of forecast errors for Holt's Winter Seasonal method:
161.29979684133863
```

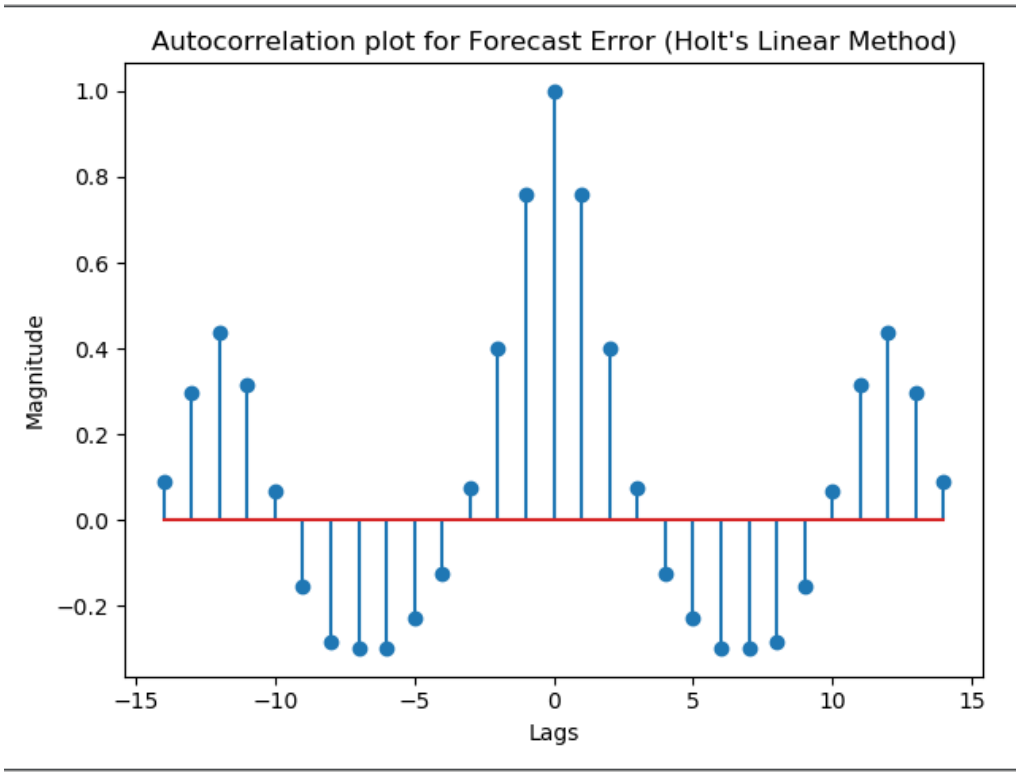
```
#%%========================================================================================
#5. # Plot the ACF of the forecast errors for each forecast. (# of lags 15)
# %%----------------------------------------------------------------------------------
```

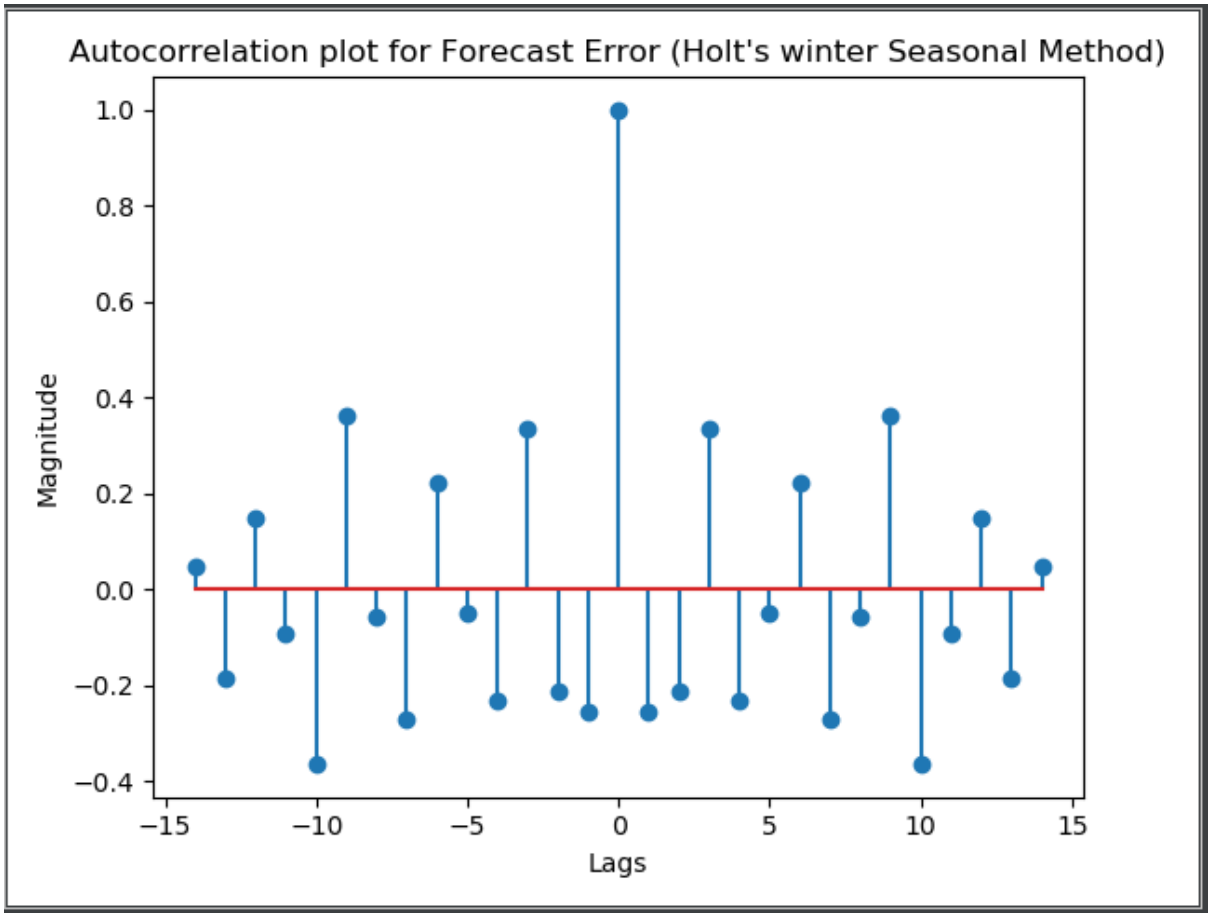



Autocorrelation plot for Forecast Error (Naive Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−15
−10
−5
0
5
10
15


Autocorrelation plot for Forecast Error (Drift Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−15
−10
−5
0
5
10
15

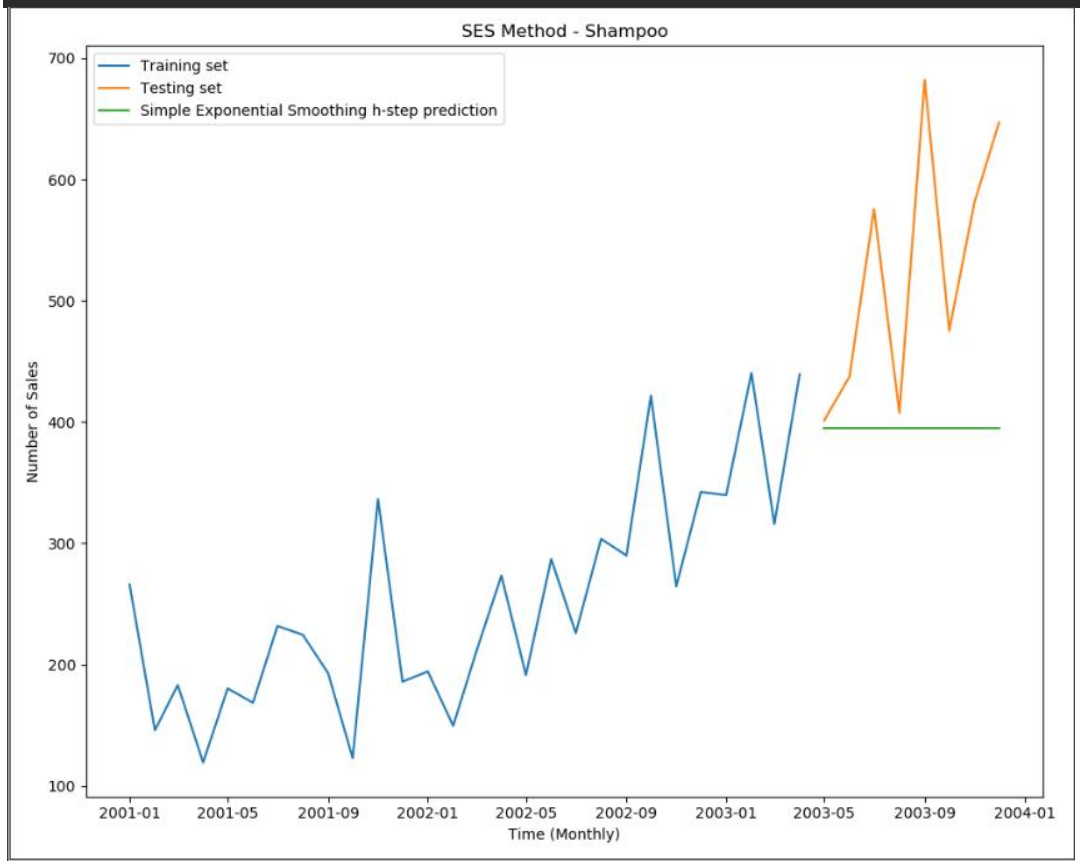

Autocorrelation plot for Forecast Error (SES Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−15
−10
−5
0
5
10
15

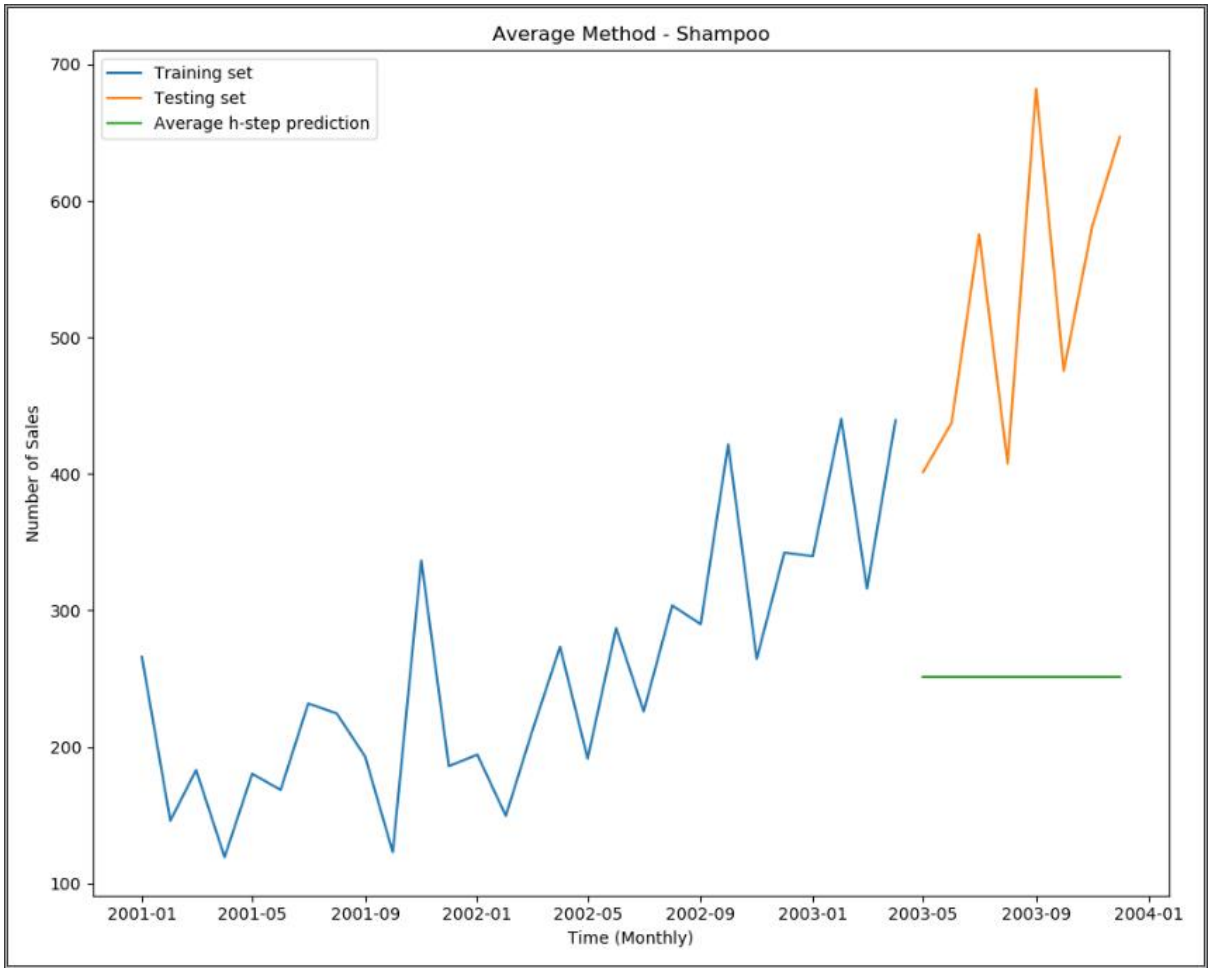

Autocorrelation plot for Forecast Error (Holt's Linear Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−15
−10
−5
0
5
10
15

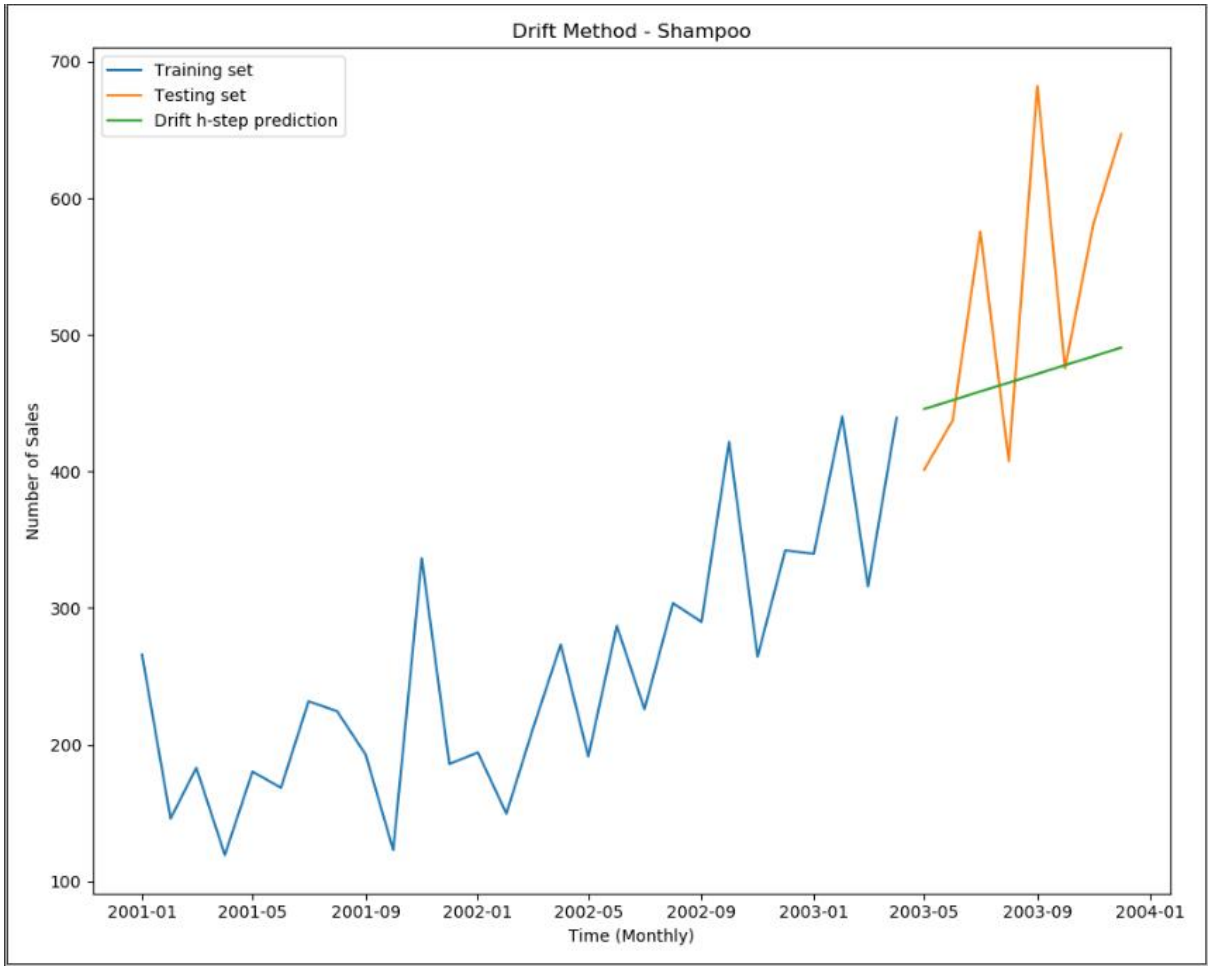

Autocorrelation plot for Forecast Error (Holt's winter Seasonal Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−15
−10
−5
0
5
10
15

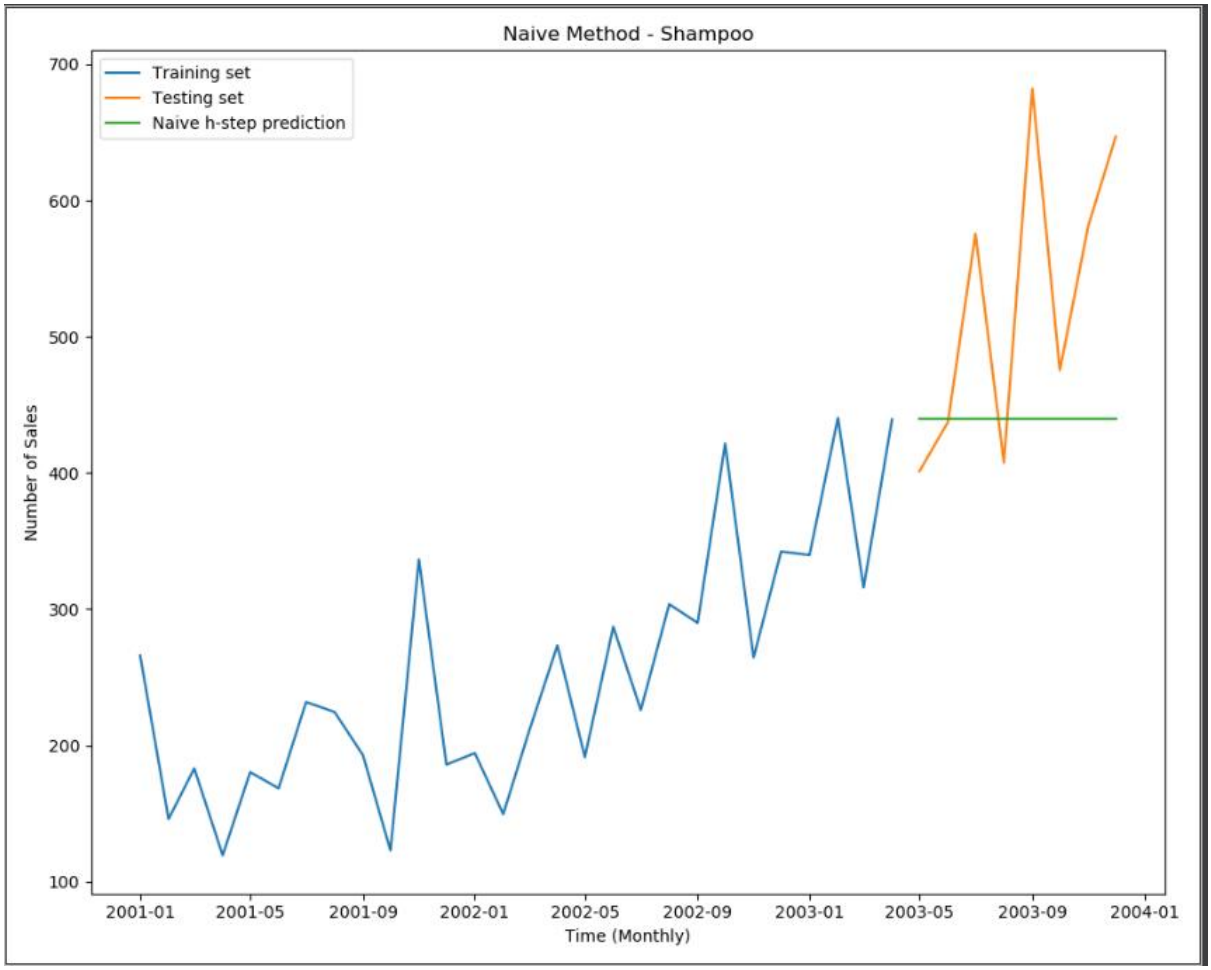
```
#%%=============================================================================
# DATA SET - SHAMPOO
# %%-----------------------------------------------------------------------------

Mean Square Error of prediction errors for Average method:  9044.974878612678
Mean Square Error of forecast errors for Average method:  85774.23349489793
Mean of prediction errors for Average method:  45.28035635512069
Variance of prediction errors for Average method:  6994.664206965959
Variance of forecast errors for Average method:  10556.273593749995

Mean Square Error of prediction errors for Naive method:  8387.768518518518
Mean Square Error of forecast errors for Naive method:  18057.998749999995
Mean of prediction errors for Naive method:  6.4185185185185185
Variance of prediction errors for Naive method:  8346.571138545954
Variance of forecast errors for Naive method:  10556.273593749997

Mean Square Error of prediction errors for Drift method:  10028.908158645114
Mean Square Error of forecast errors for Drift method:  12169.632031893001
Mean of prediction errors for Drift method:  17.032548218000688
Variance of prediction errors for Drift method:  9738.800459846596
Variance of forecast errors for Drift method:  8836.975347865226

Mean Square Error of prediction errors for SES method:  5320.696554796655
Mean Square Error of forecast errors for SES method:  27636.886353926835
Mean of prediction errors for SES method:  9.571828272662776
Variance of prediction errors for SES method:  5229.07665831531
Variance of forecast errors for SES method:  10556.273593749998
Mean Square Error of prediction errors for Holt's Linear method:
4437.3544470728475
Mean Square Error of forecast errors for Holt's Linear method:  28231.284973348847
Mean of prediction errors for Holt's Linear method:  20.700119891480274
Variance of prediction errors for Holt's Linear method:  4008.85948355119
Variance of forecast errors for Holt's Linear method:  10324.212200480055

Mean Square Error of prediction errors for Holt's Winter Seasonal method:  nan
Mean Square Error of forecast errors for Holt's Winter Seasonal method:
42619.61972440532
Mean of prediction errors for Holt's Winter Seasonal method:  nan
Variance of prediction errors for Holt's Winter Seasonal method:  nan
Variance of forecast errors for Holt's Winter Seasonal method:  9401.272624113657

Q value of forecast errors for Average method:  2.5853614772884224
Q value of forecast errors for Naive method:  2.585361477288423
Q value of forecast errors for Drift method:  3.131804536739435
Q value of forecast errors for SES method:  2.585361477288423
Q value of forecast errors for Holt's Linear method:  2.6300584160086147
Q value of forecast errors for Holt's Winter Seasonal method:  3.4038398950273283

Correlation Coefficient between Forecast Error and Test set for Average Method:
1.0
Correlation Coefficient between Forecast Error and Test set for Naive Method:
0.9999999999999998
Correlation Coefficient between Forecast Error and Test set for Drift Method:
0.9927563811603678
Correlation Coefficient between Forecast Error and Test set for SES Method:
0.9999999999999998
Correlation Coefficient between Forecast Error and Test set for Holt's Linear
Method: 0.9998938126254218
```

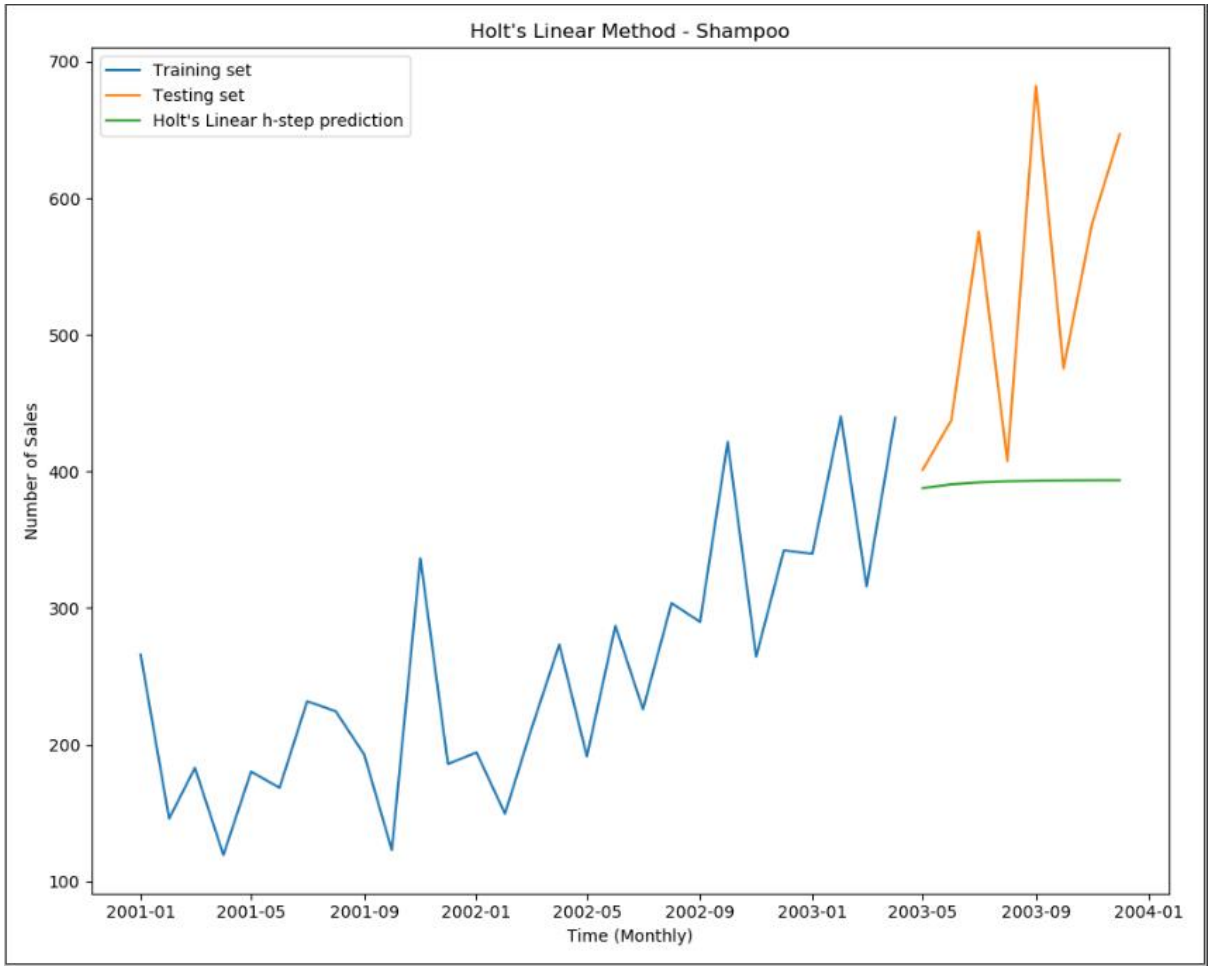
```
Correlation Coefficient between Forecast Error and Test set for Holt's winter
seasonal Method: 0.5297408498696383

         Q_val    MSE(P)    MSE(F)   var(P)    var(F)  corrcoeff
Methods
Average   2.59   9044.97  85774.23  6994.66  10556.27       1.00
Naive     2.59   8387.77  18058.00  8346.57  10556.27       1.00
Drift     3.13  10028.91  12169.63  9738.80   8836.98       0.99
SES       2.59   5320.70  27636.89  5229.08  10556.27       1.00
HoltL     2.63   4437.35  28231.28  4008.86  10324.21       1.00
HoltW     3.40       NaN  42619.62      NaN   9401.27       0.53
```

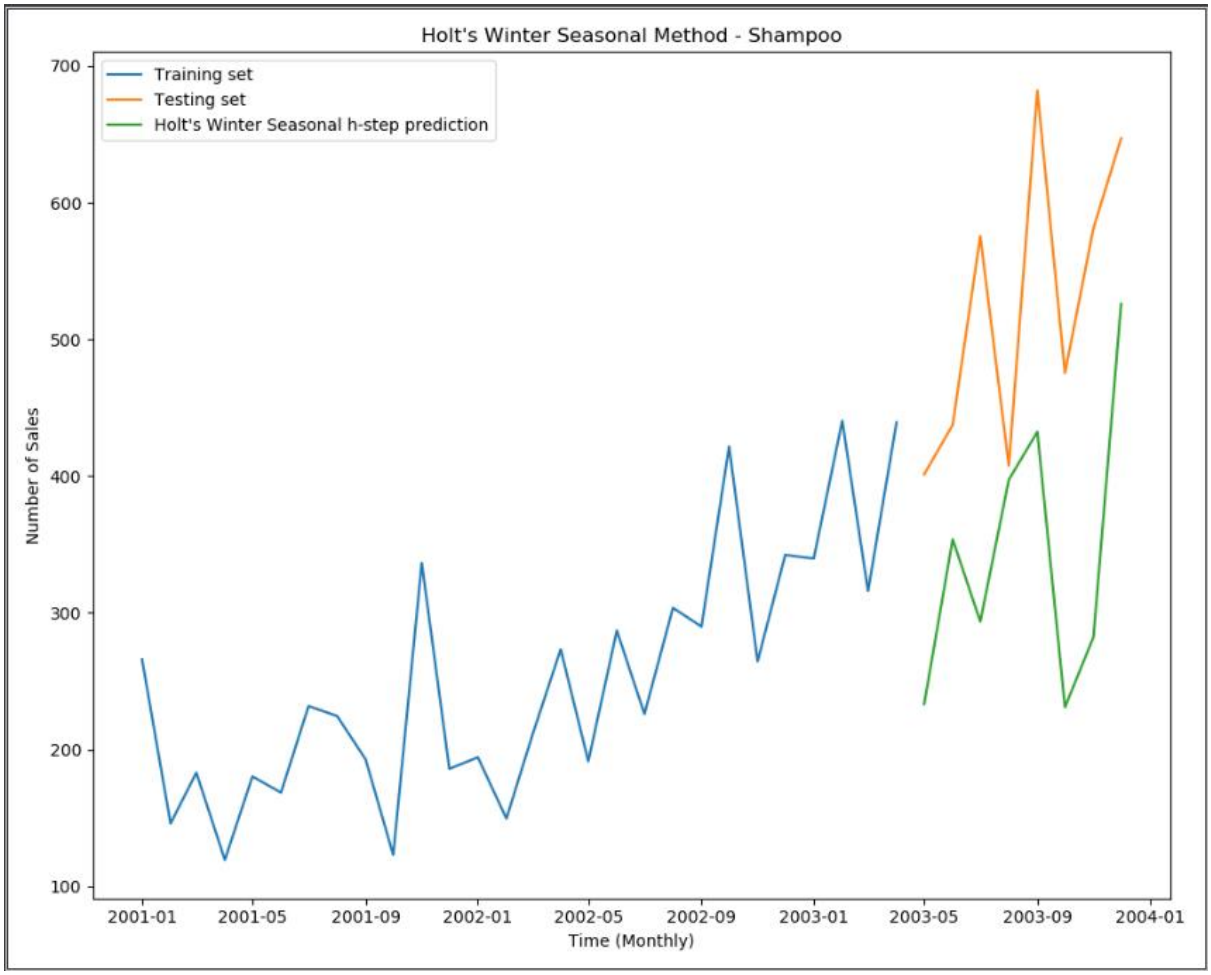


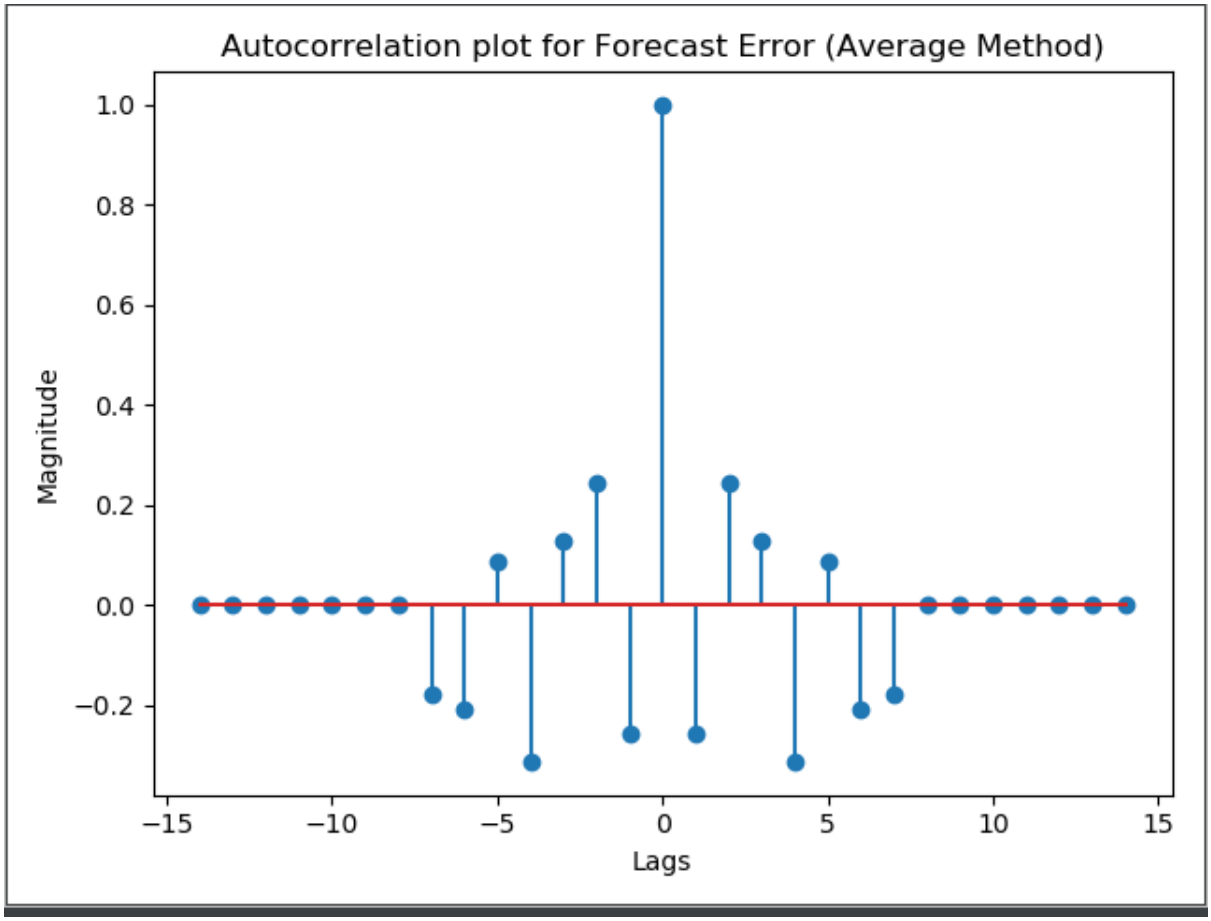

Average Method - Shampoo
Training set
Testing set
Average h-step prediction
Number of Sales
Time (Monthly)
700
600
500
400
300
200
100
2001-01
2001-05
2001-09
2002-01
2002-05
2002-09
2003-01
2003-05
2003-09
2004-01

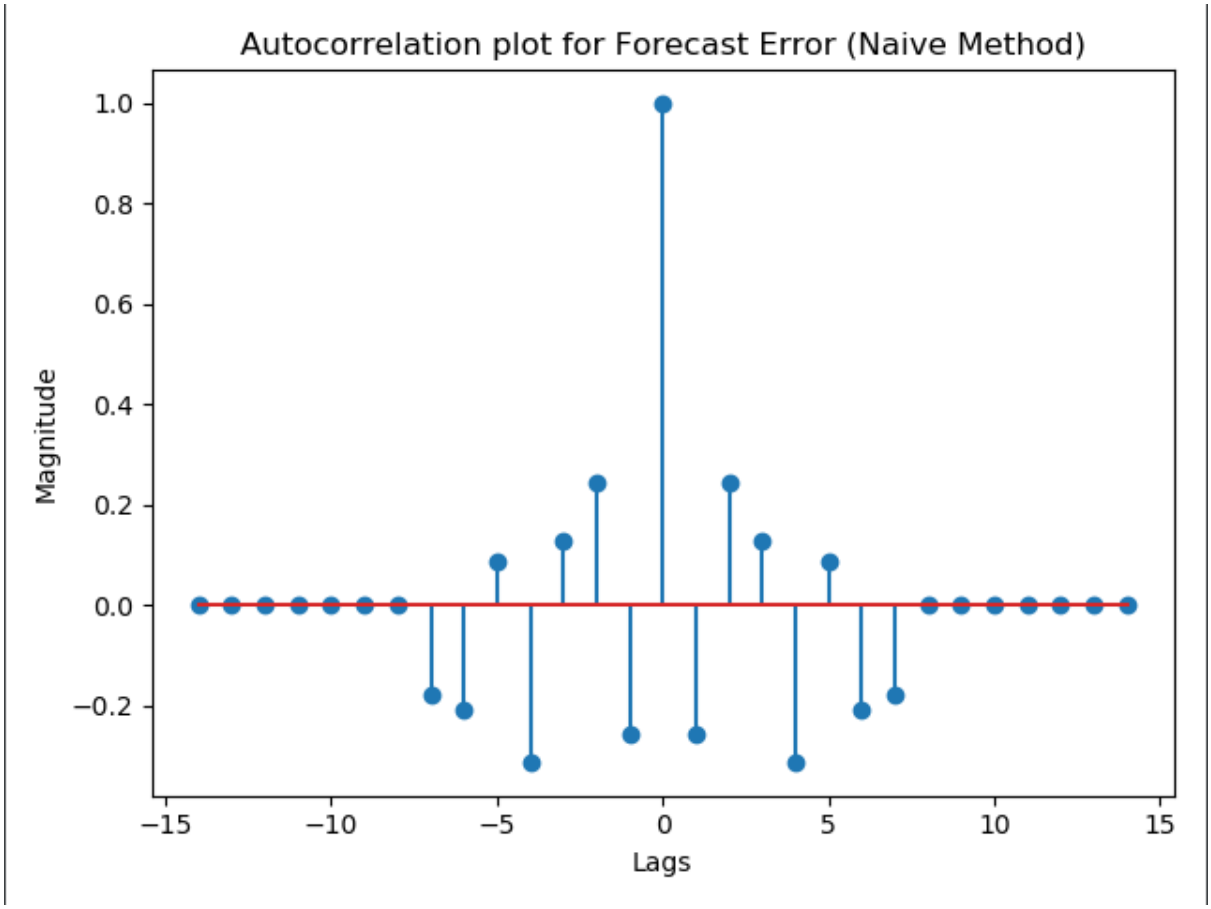

Drift Method - Shampoo
Training set
Testing set
Drift h-step prediction
Number of Sales
Time (Monthly)
700
600
500
400
300
200
100
2001-01
2001-05
2001-09
2002-01
2002-05
2002-09
2003-01
2003-05
2003-09
2004-01

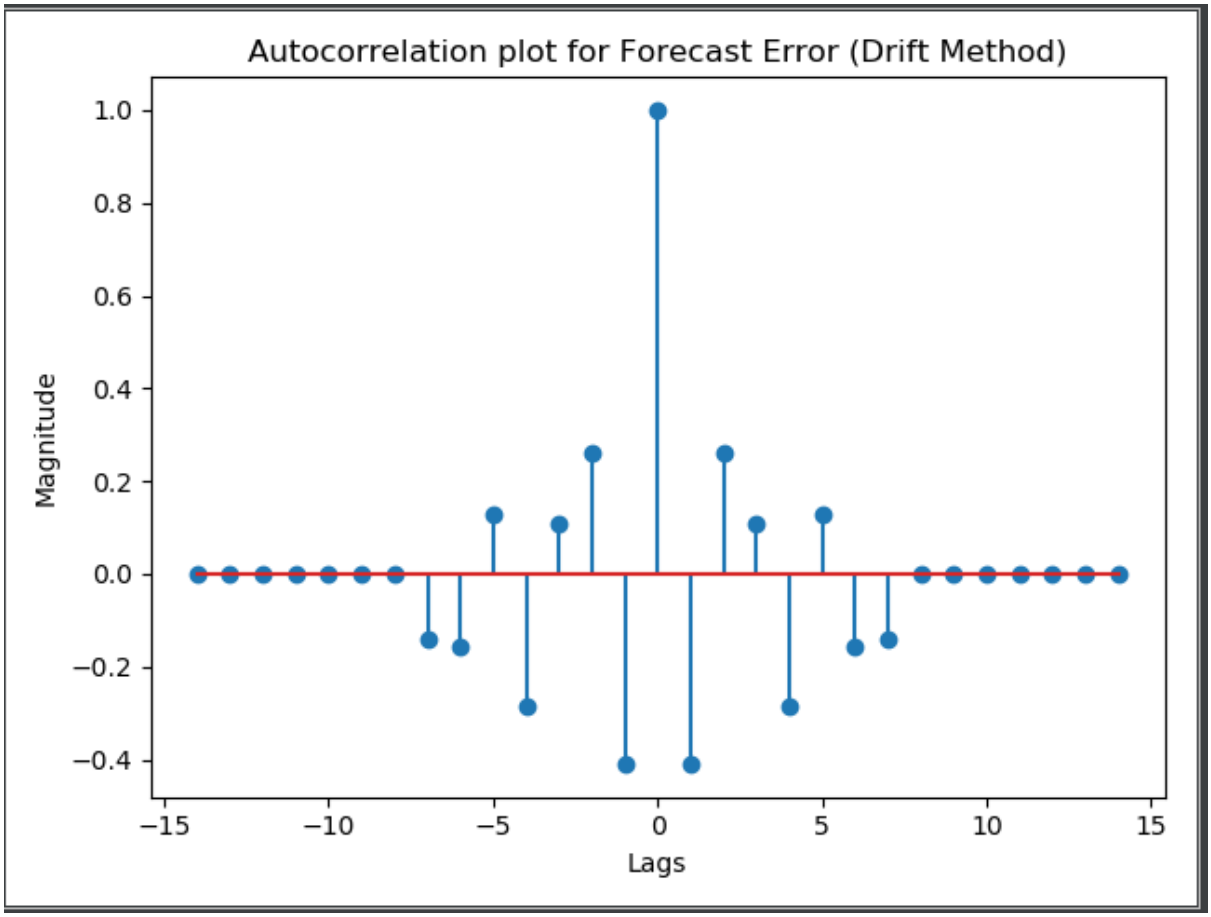

Naive Method - Shampoo
Training set
Testing set
Naive h-step prediction
Number of Sales
Time (Monthly)
700
600
500
400
300
200
100
2001-01
2001-05
2001-09
2002-01
2002-05
2002-09
2003-01
2003-05
2003-09
2004-01

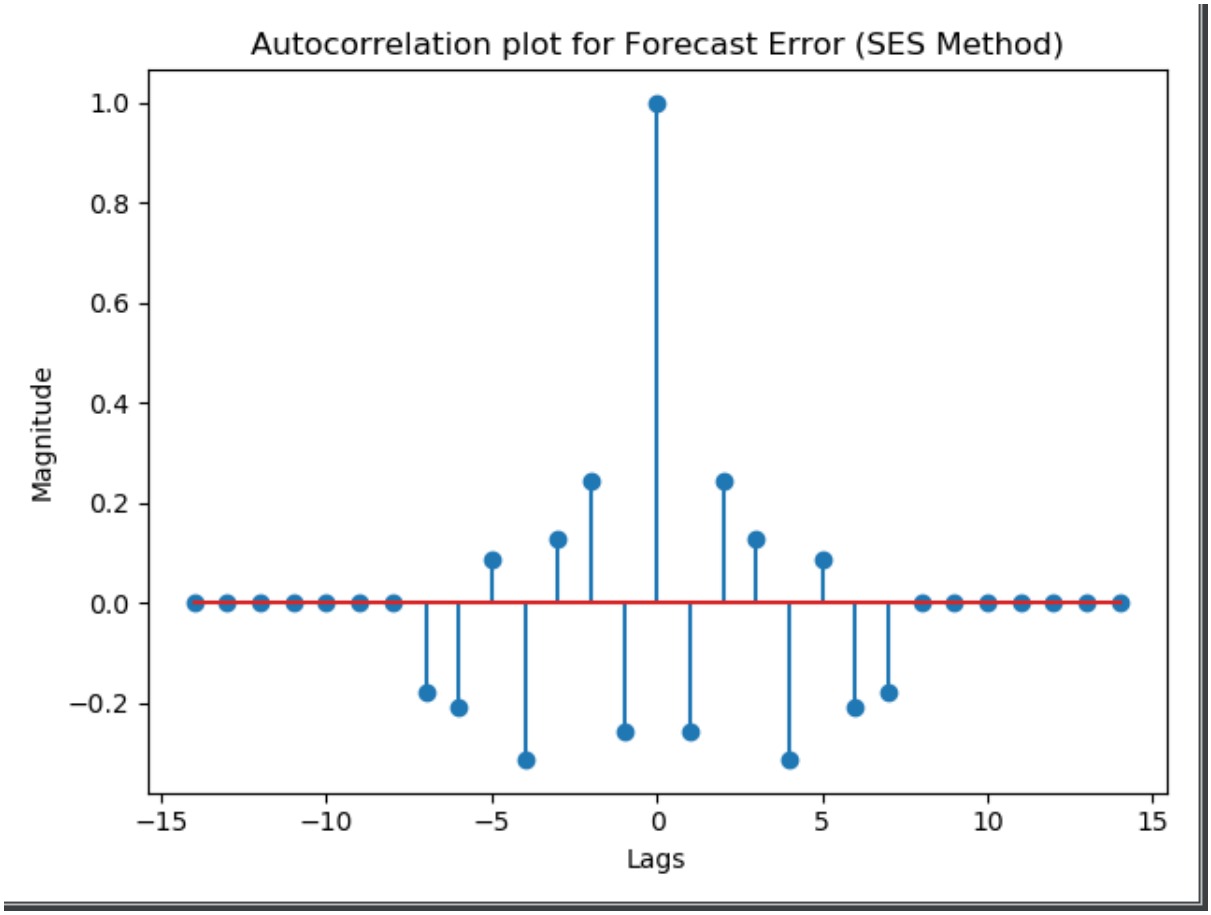

Holt's Linear Method - Shampoo
Training set
Testing set
Holt's Linear h-step prediction
Number of Sales
Time (Monthly)
700
600
500
400
300
200
100
2001-01
2001-05
2001-09
2002-01
2002-05
2002-09
2003-01
2003-05
2003-09
2004-01

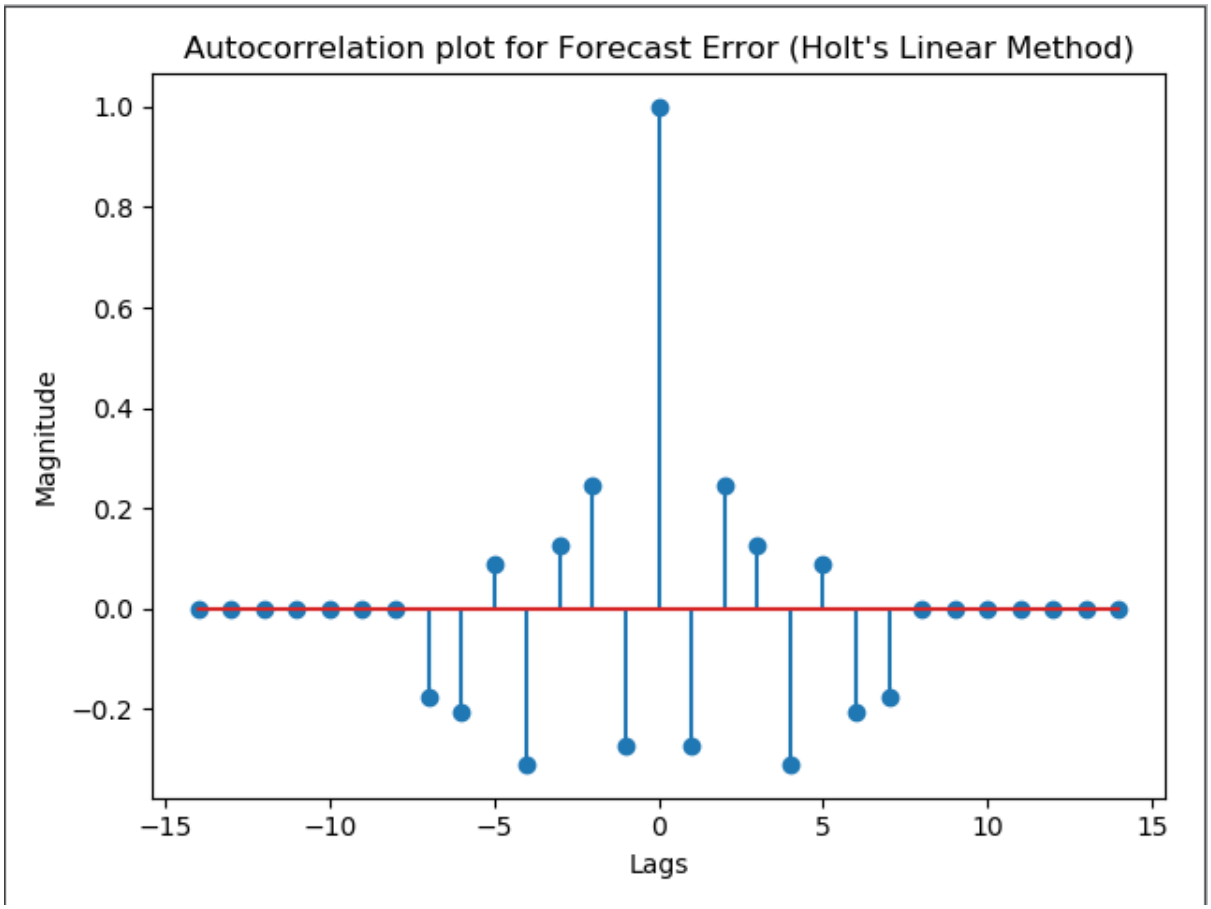

Holt's Winter Seasonal Method - Shampoo
Training set
Testing set
Holt's Winter Seasonal h-step prediction
Number of Sales
Time (Monthly)
700
600
500
400
300
200
100
2001-01
2001-05
2001-09
2002-01
2002-05
2002-09
2003-01
2003-05
2003-09
2004-01

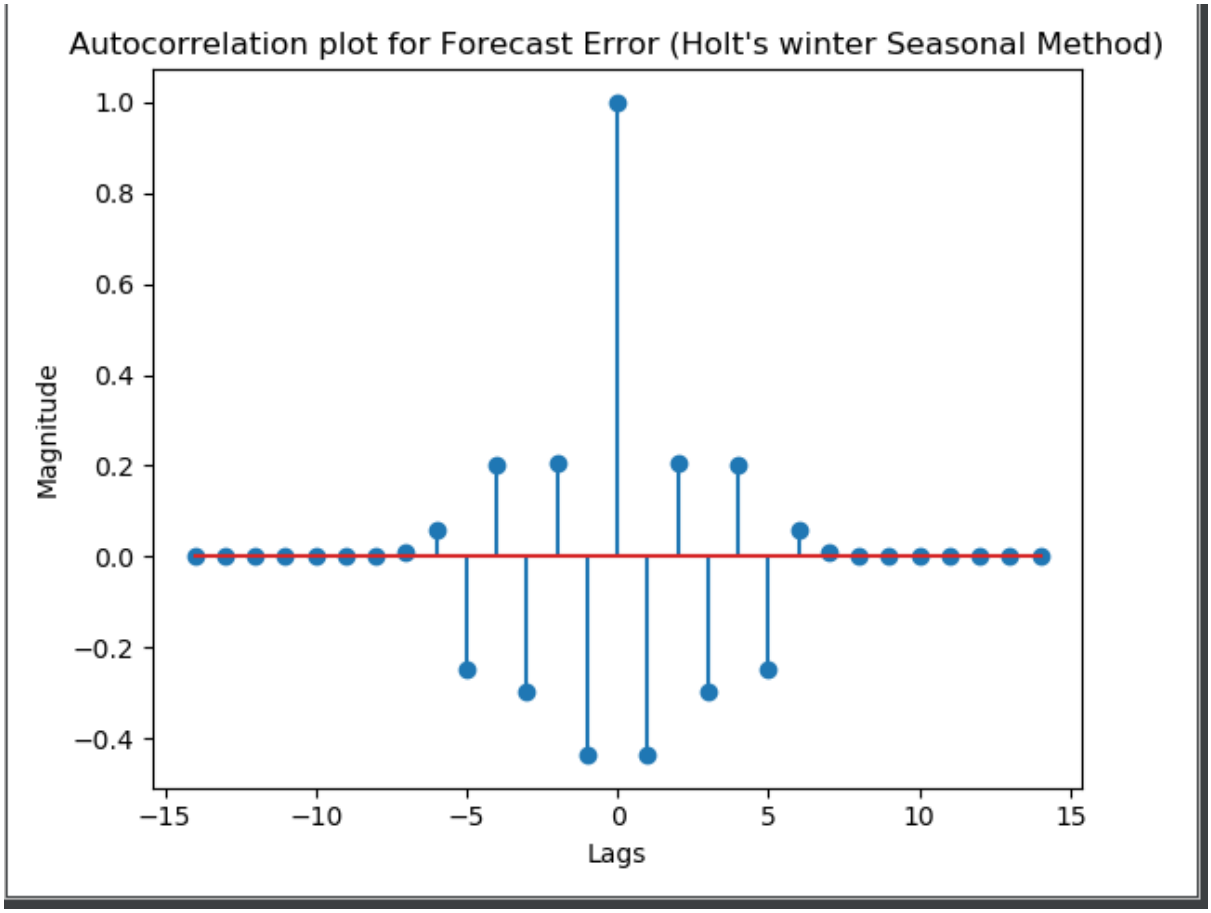

Autocorrelation plot for Forecast Error (Average Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−15
−10
−5
0
5
10
15


Autocorrelation plot for Forecast Error (Naive Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−15
−10
−5
0
5
10
15

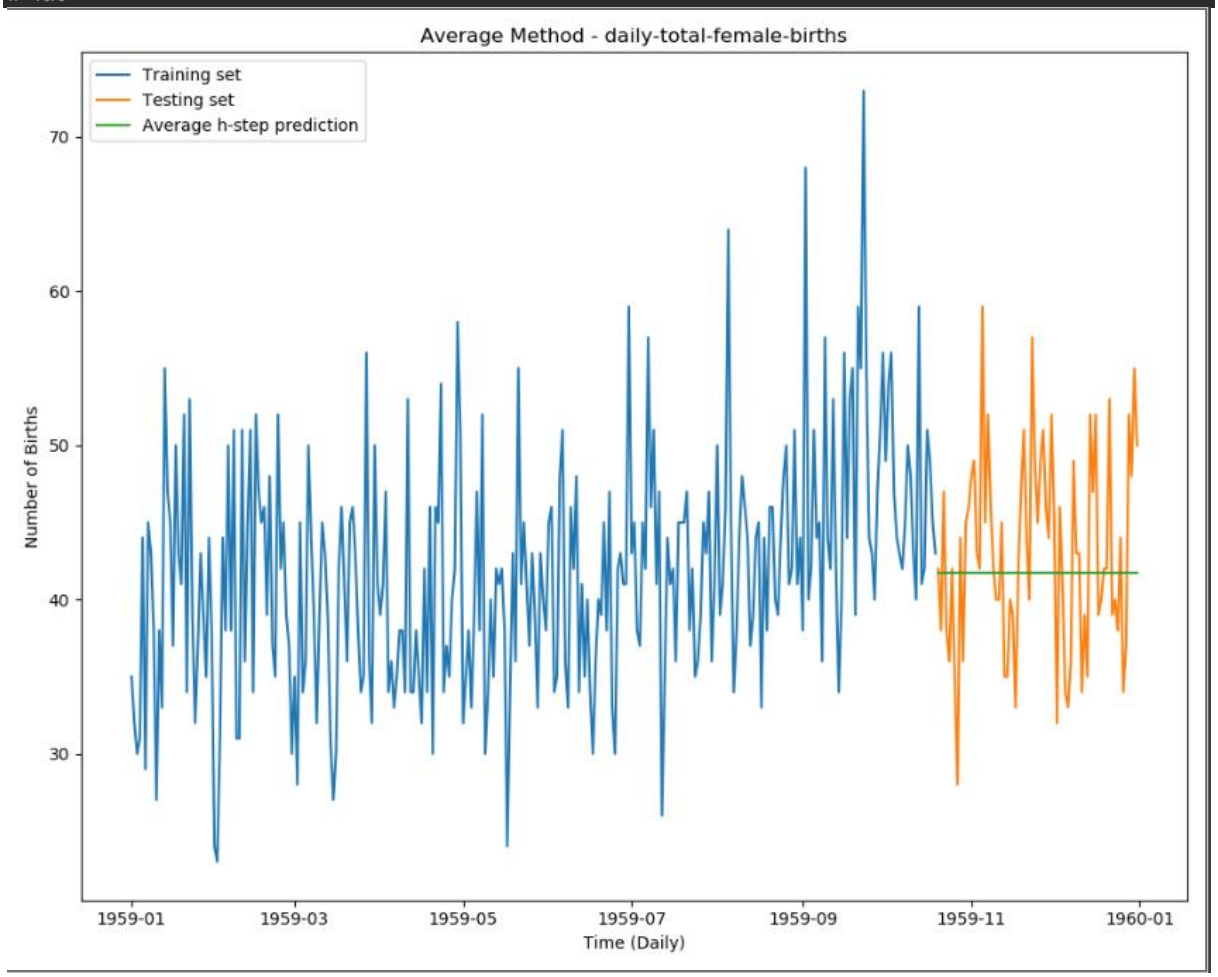

Autocorrelation plot for Forecast Error (Drift Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−15
−10
−5
0
5
10
15

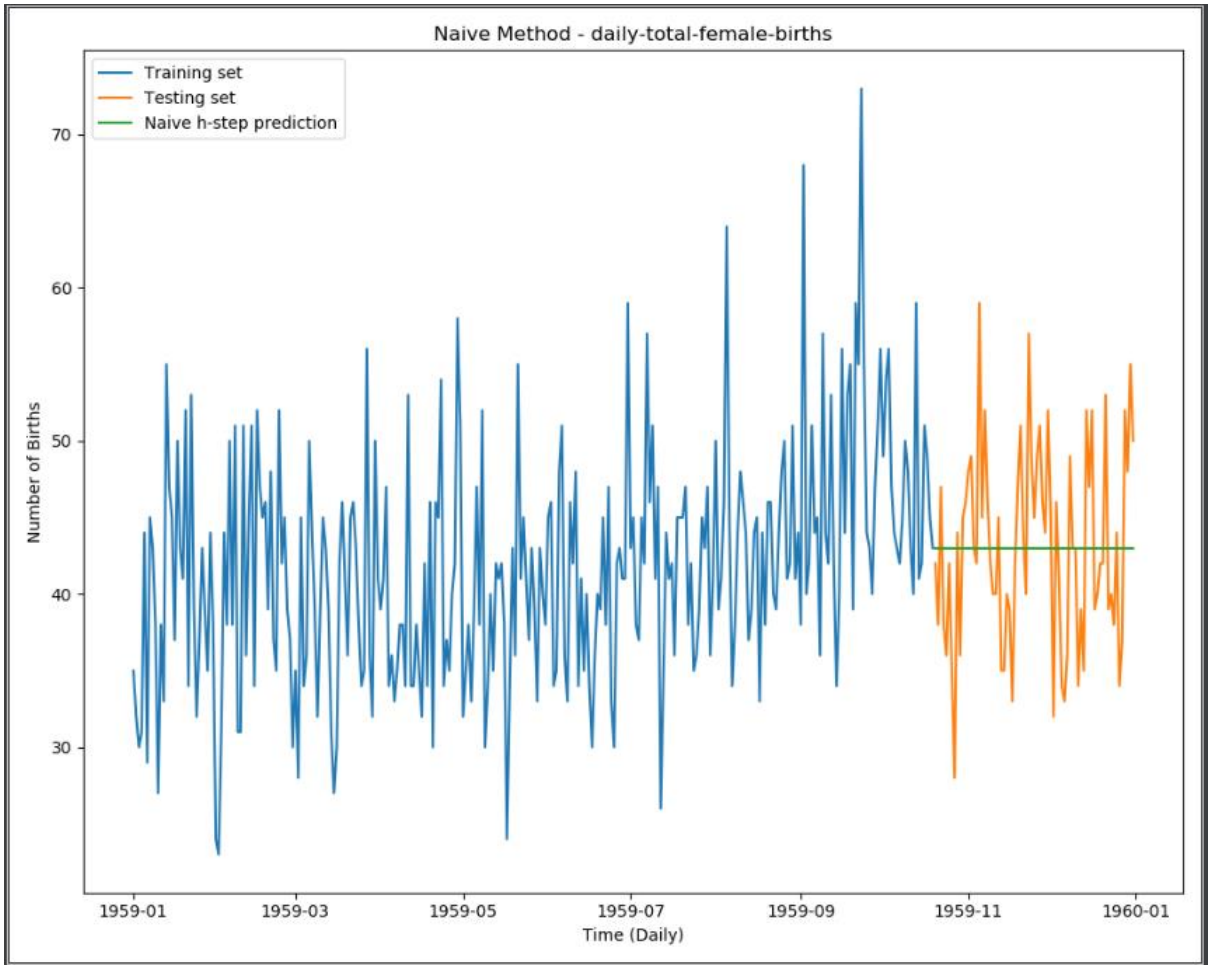

Autocorrelation plot for Forecast Error (SES Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−15
−10
−5
0
5
10
15

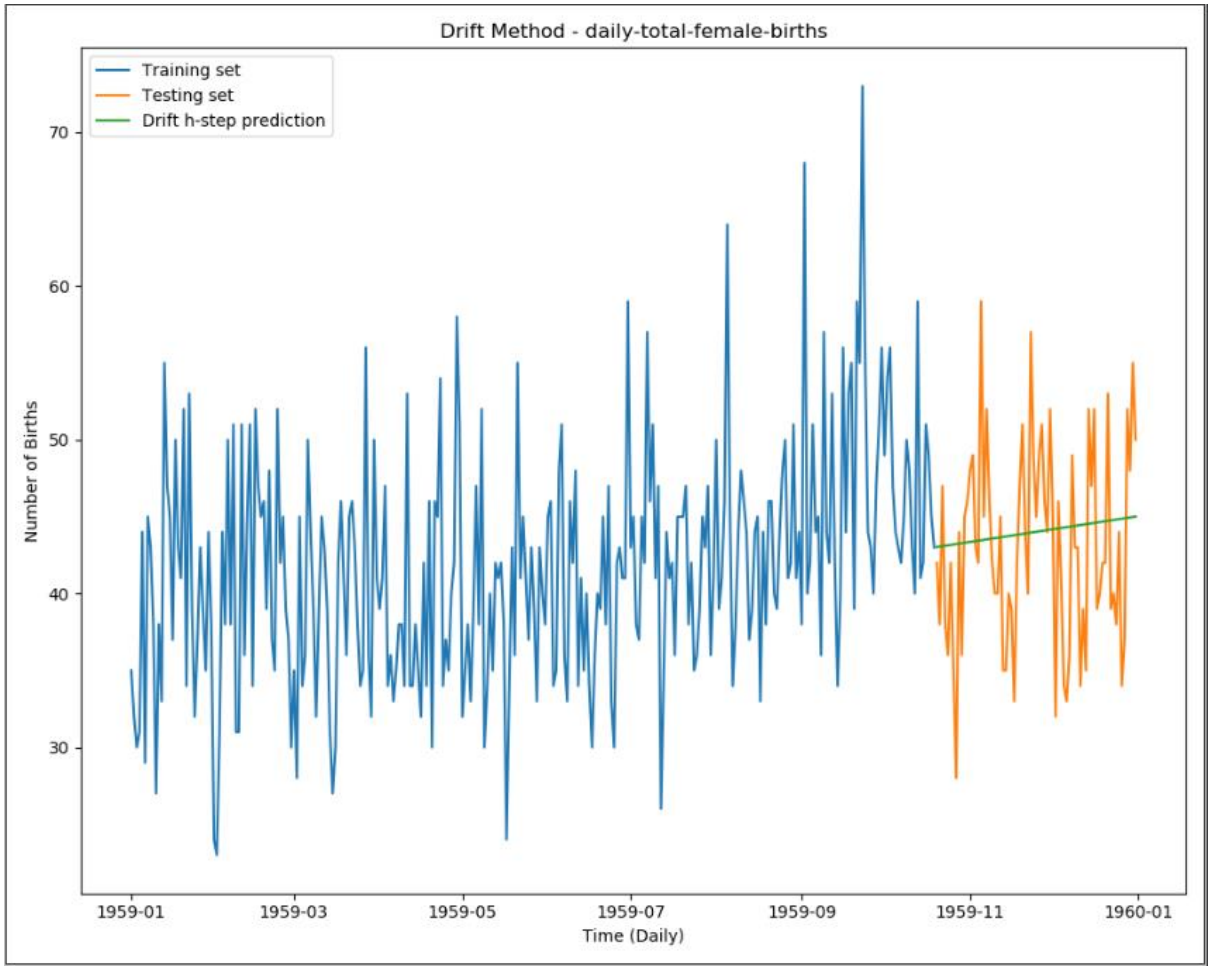

Autocorrelation plot for Forecast Error (Holt's Linear Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−15
−10
−5
0
5
10
15

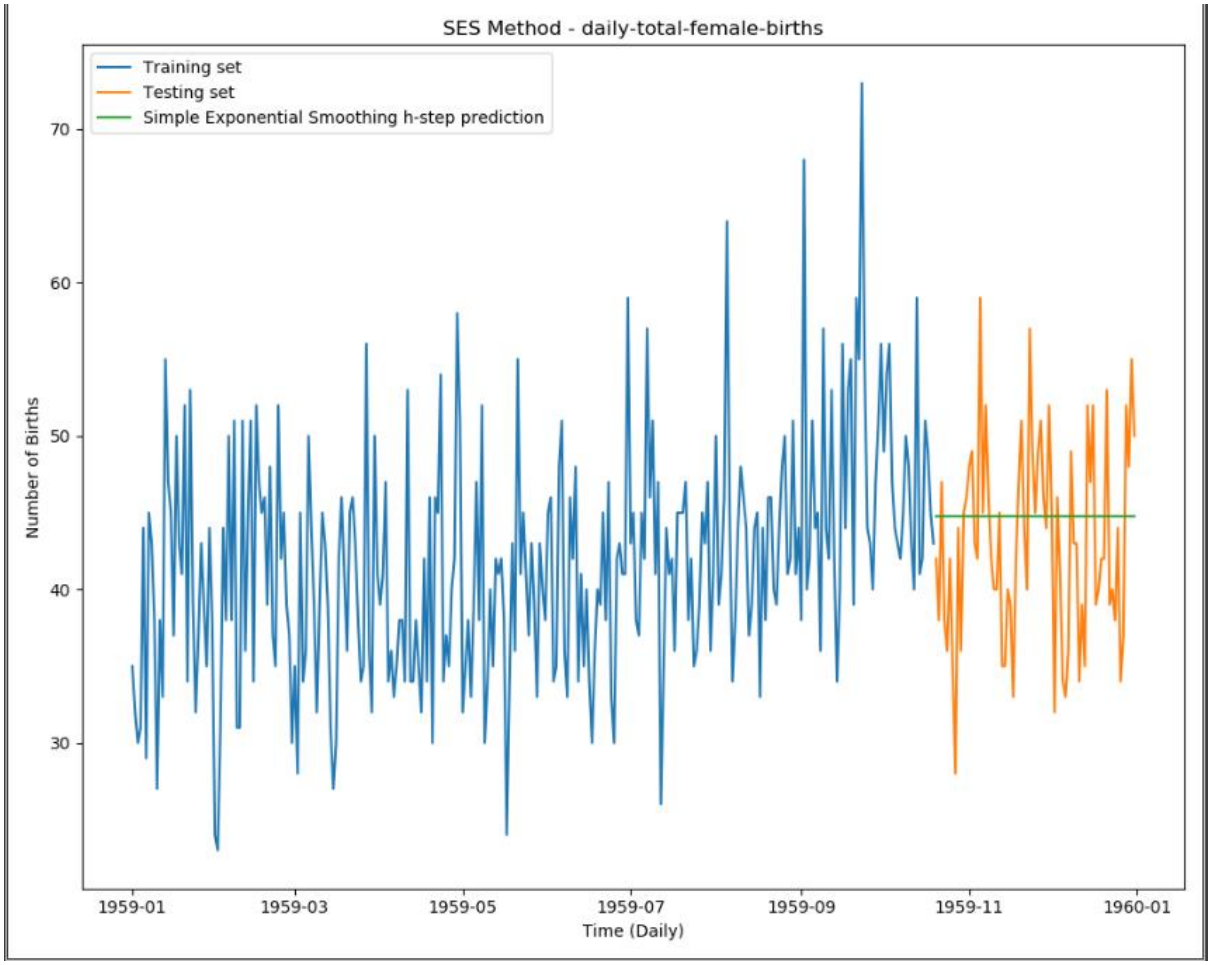

Autocorrelation plot for Forecast Error (Holt's winter Seasonal Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−15
−10
−5
0
5
10
15

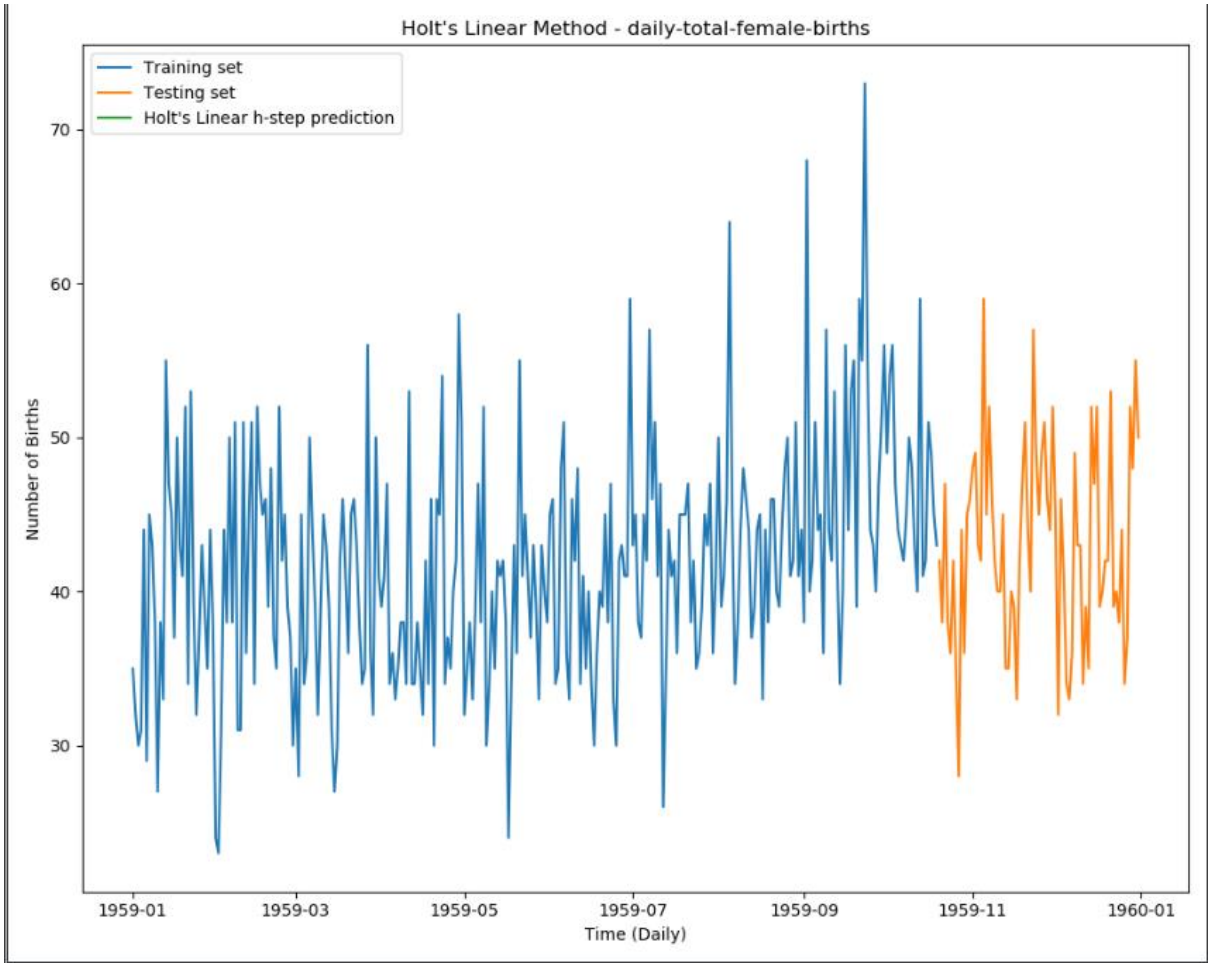
```
#%%=============================================================================
# DATA SET - daily-total-female-births Dataset results
# %%-----------------------------------------------------------------------------
```

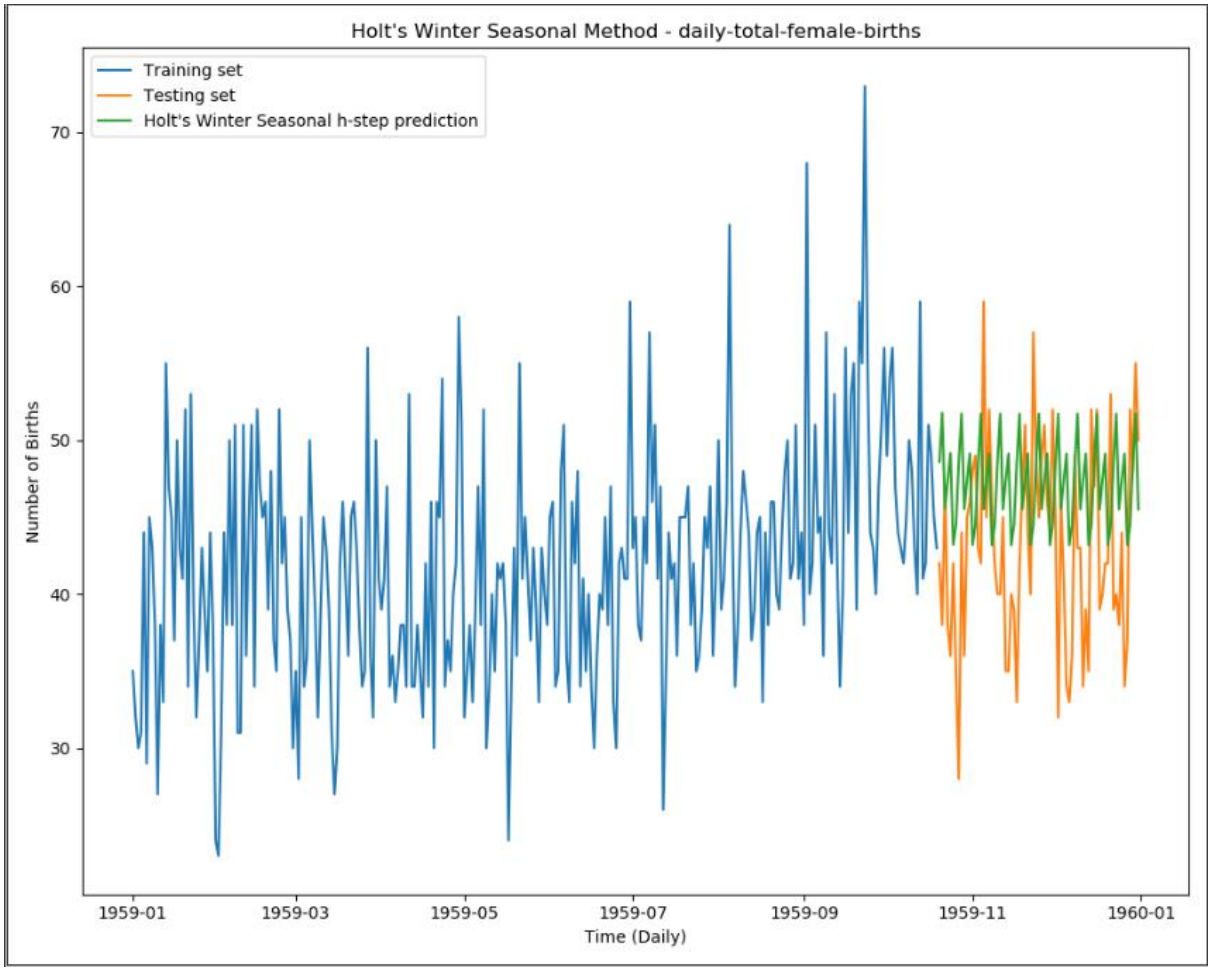


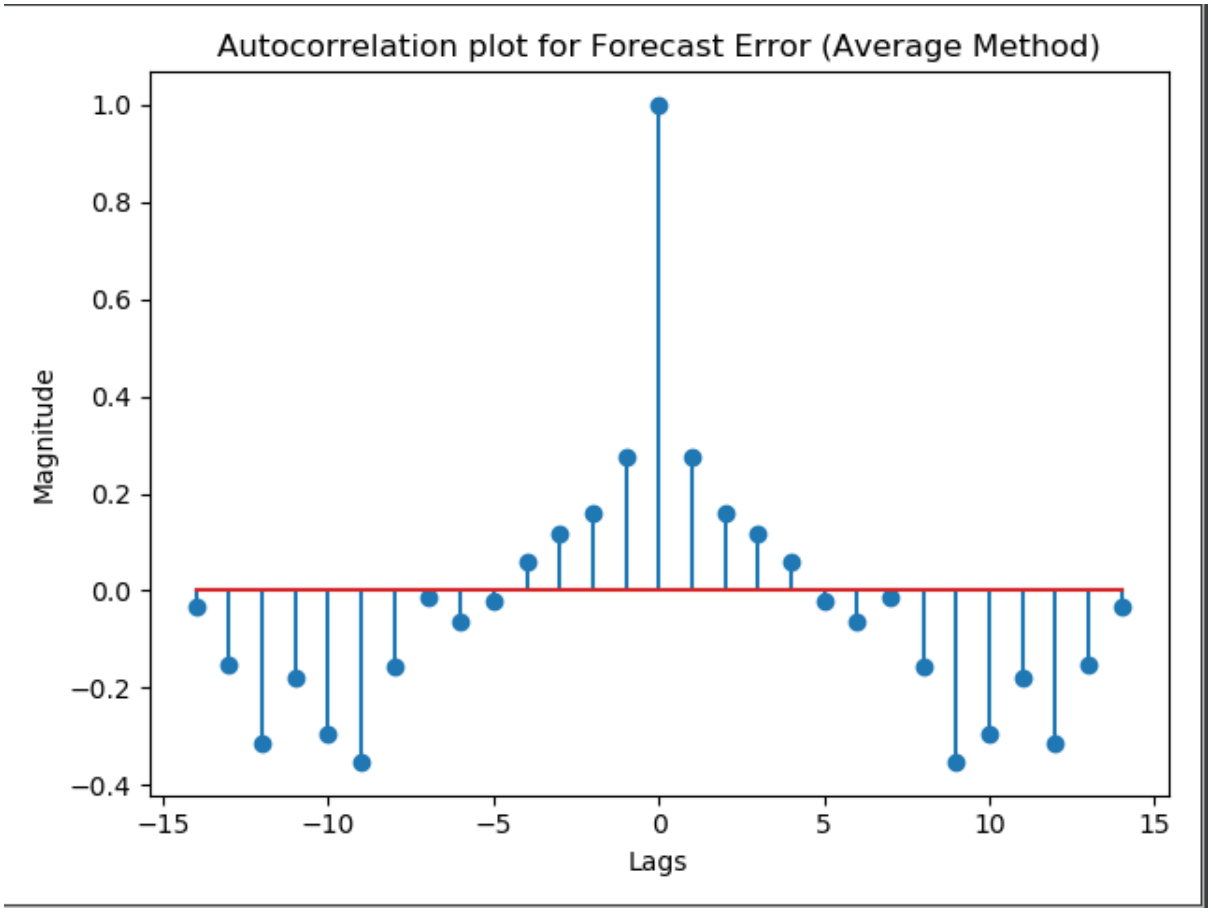

Naive Method - daily-total-female-births
Training set
Testing set
Naive h-step prediction
Number of Births
70
60
50
40
30
1959-01
1959-03
1959-05
1959-07
1959-09
1959-11
1960-01
Time (Daily)

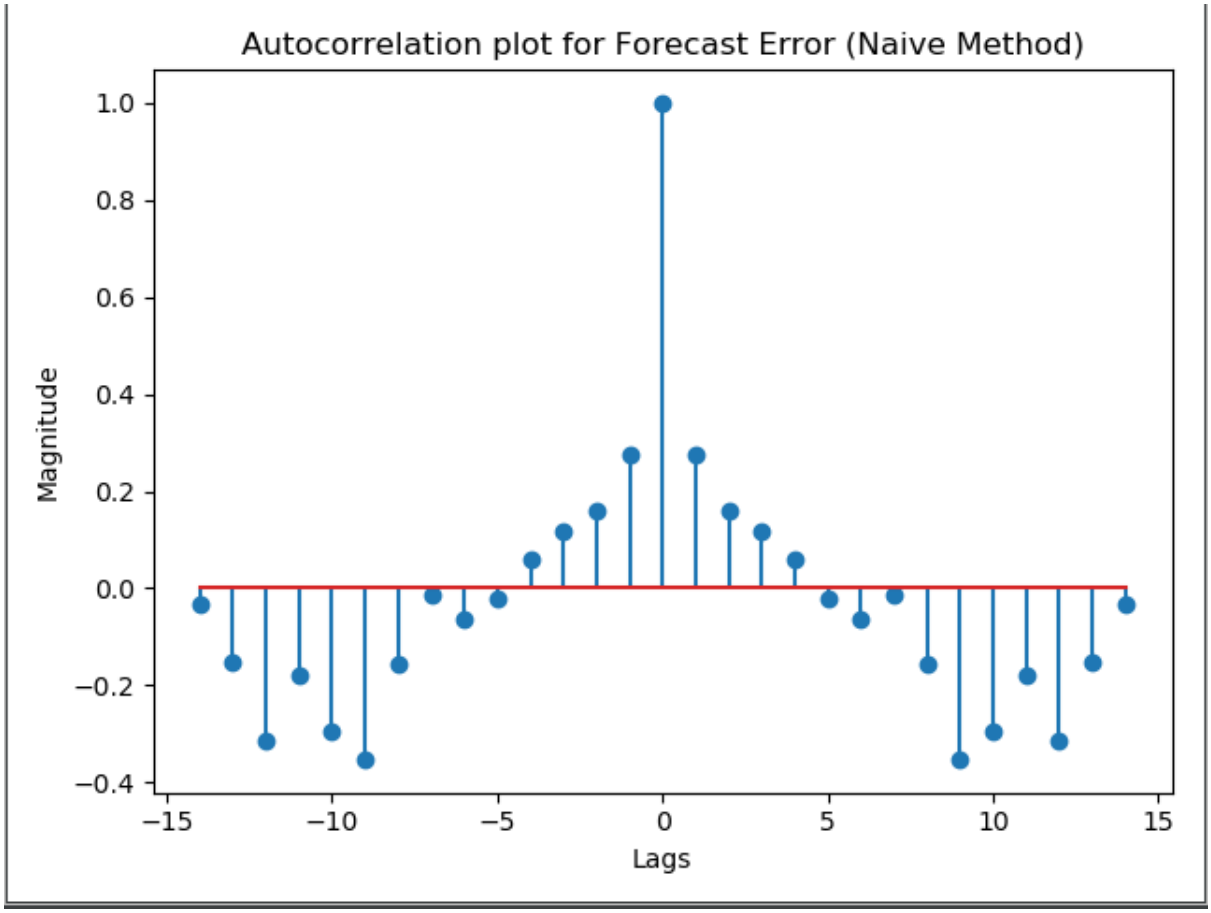

Drift Method - daily-total-female-births
Training set
Testing set
Drift h-step prediction
Number of Births
70
60
50
40
30
1959-01
1959-03
1959-05
1959-07
1959-09
1959-11
1960-01
Time (Daily)

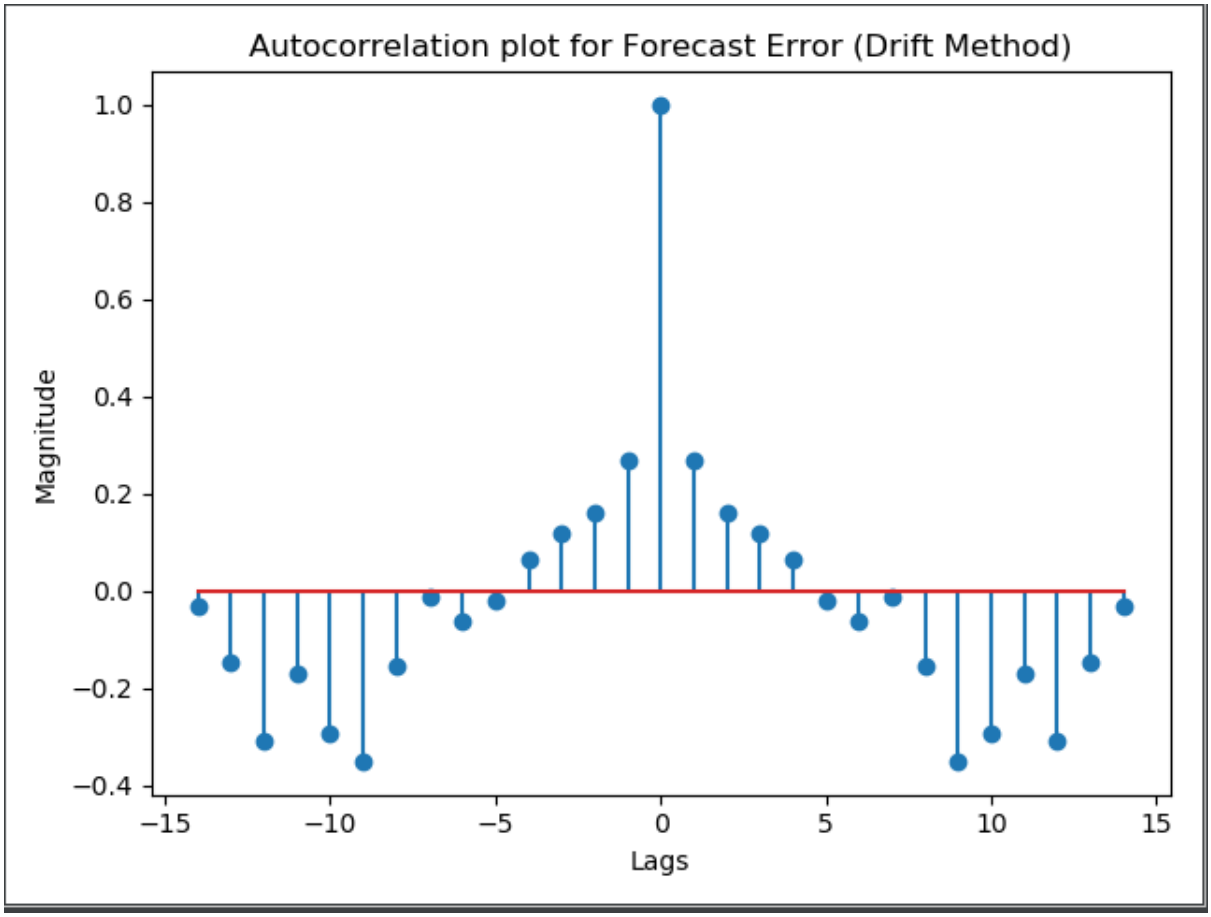

SES Method - daily-total-female-births
Training set
Testing set
Simple Exponential Smoothing h-step prediction
Number of Births
70
60
50
40
30
1959-01
1959-03
1959-05
1959-07
1959-09
1959-11
1960-01
Time (Daily)

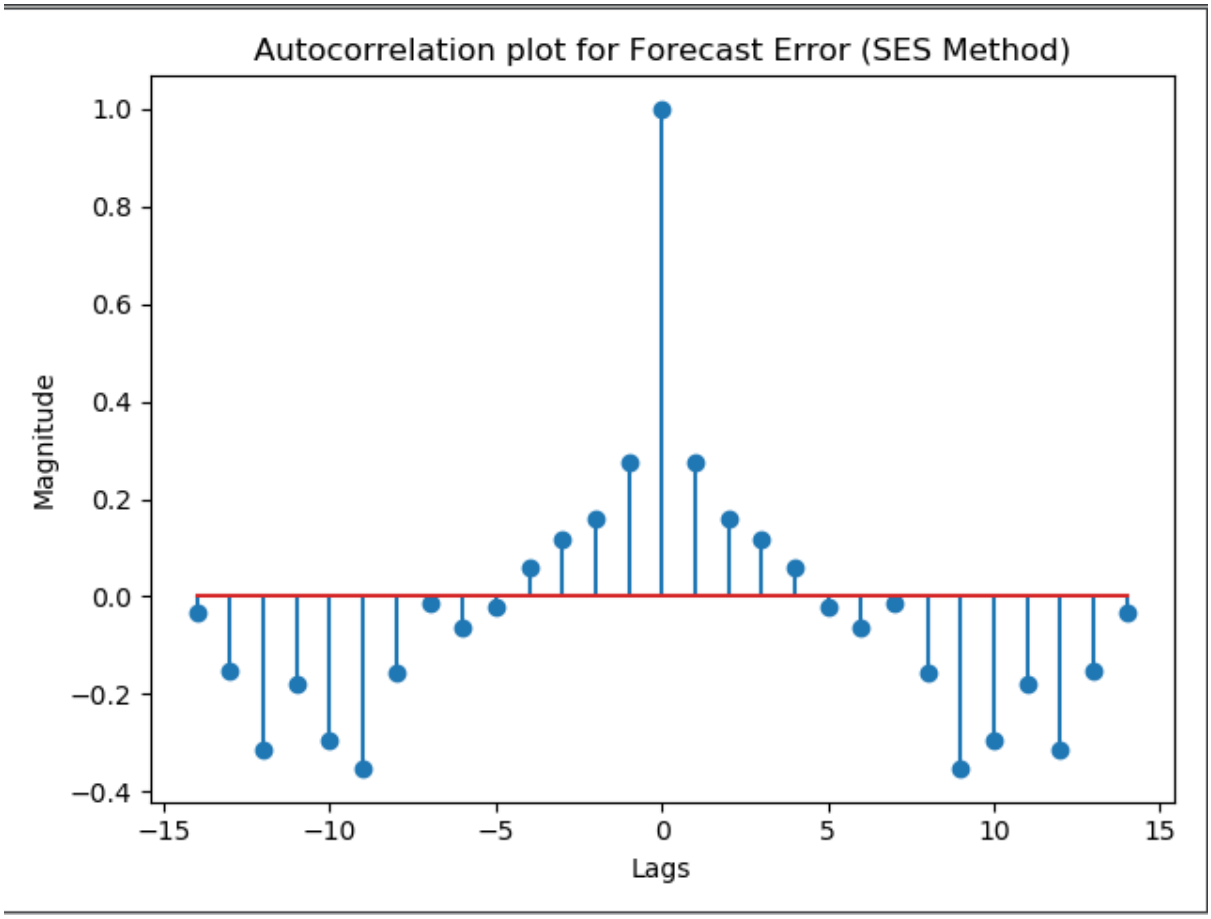

Holt's Linear Method - daily-total-female-births
Training set
Testing set
Holt's Linear h-step prediction
Number of Births
70
60
50
40
30
1959-01
1959-03
1959-05
1959-07
1959-09
1959-11
1960-01
Time (Daily)

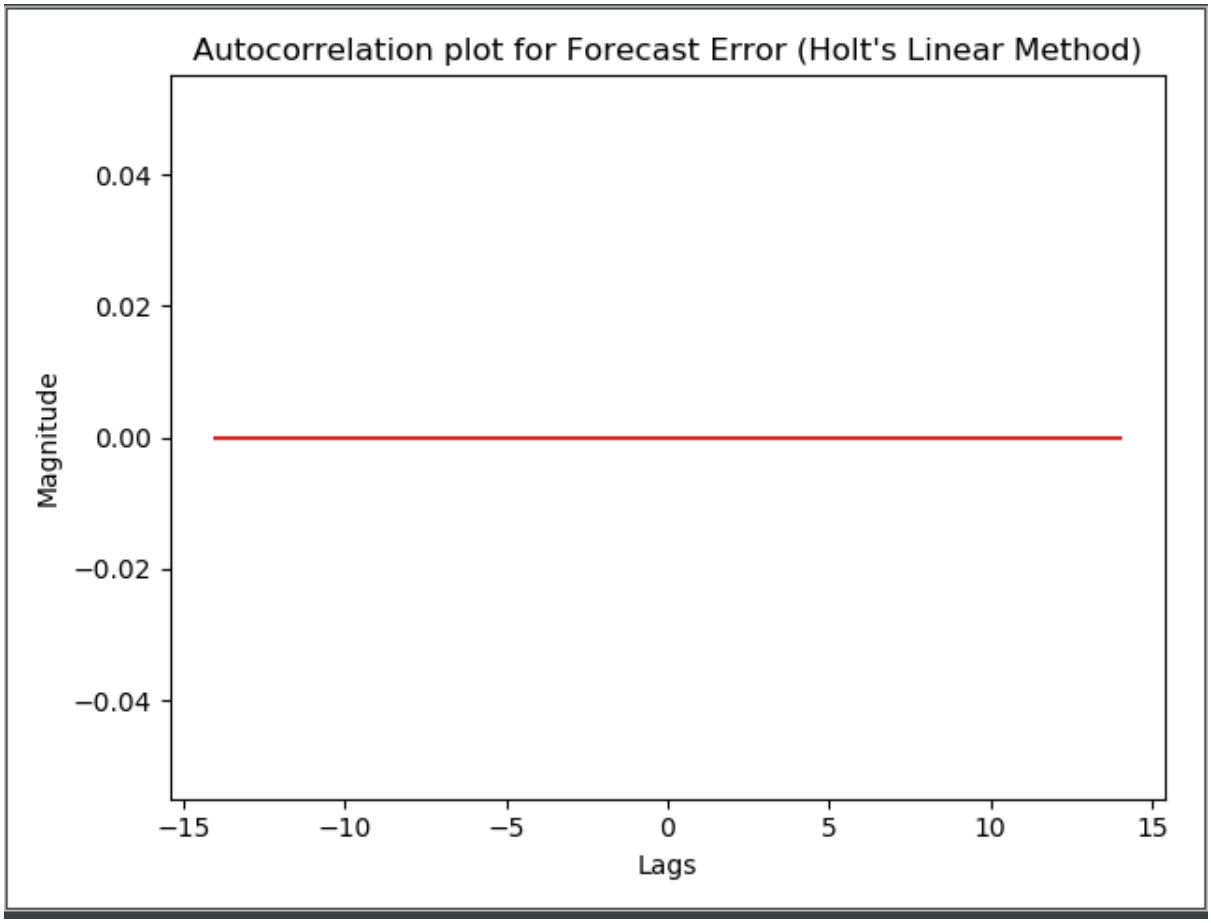

Holt's Winter Seasonal Method - daily-total-female-births
Training set
Testing set
Holt's Winter Seasonal h-step prediction
Number of Births
70
60
50
40
30
1959-01
1959-03
1959-05
1959-07
1959-09
1959-11
1960-01
Time (Daily)

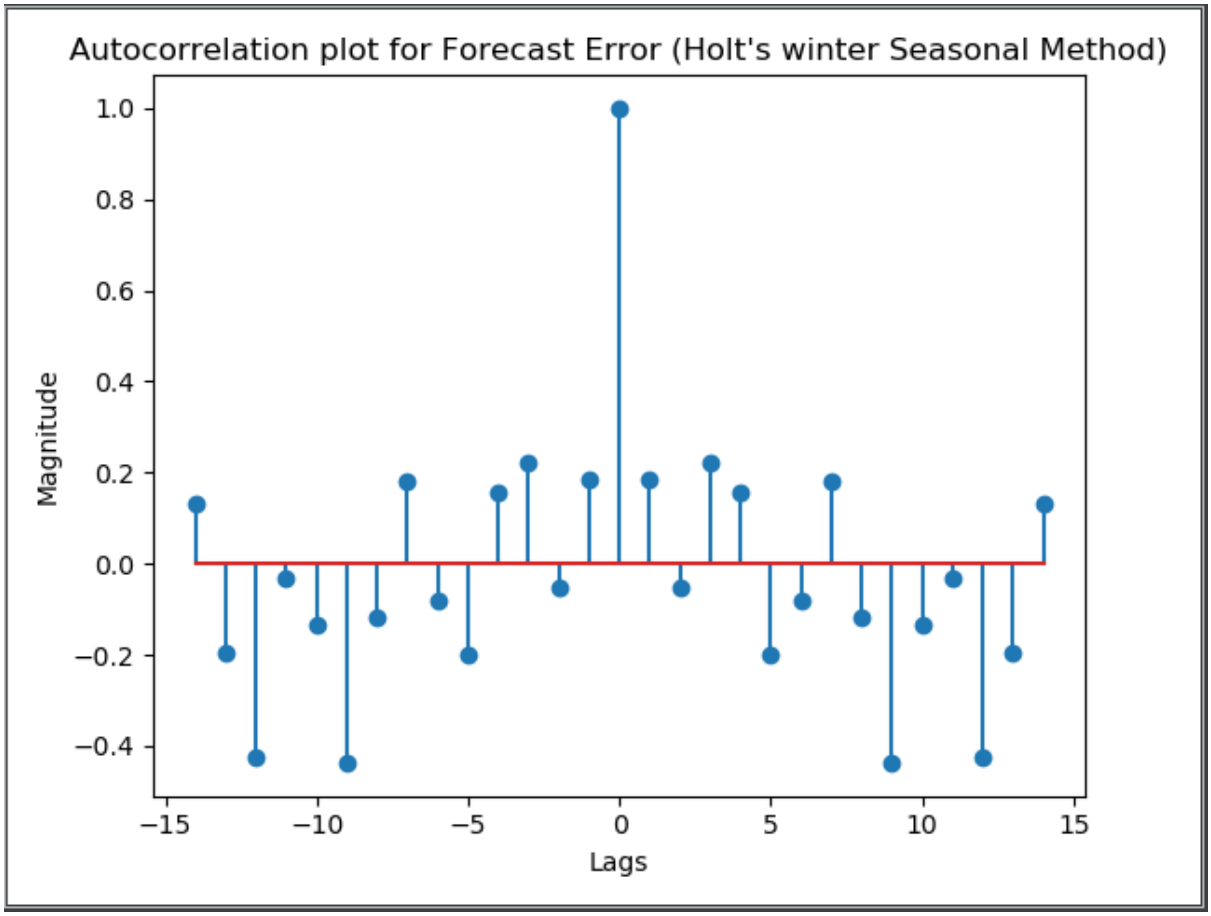

Autocorrelation plot for Forecast Error (Average Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−15
−10
−5
0
5
10
15


Autocorrelation plot for Forecast Error (Naive Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−15
−10
−5
0
5
10
15


Autocorrelation plot for Forecast Error (Drift Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−15
−10
−5
0
5
10
15

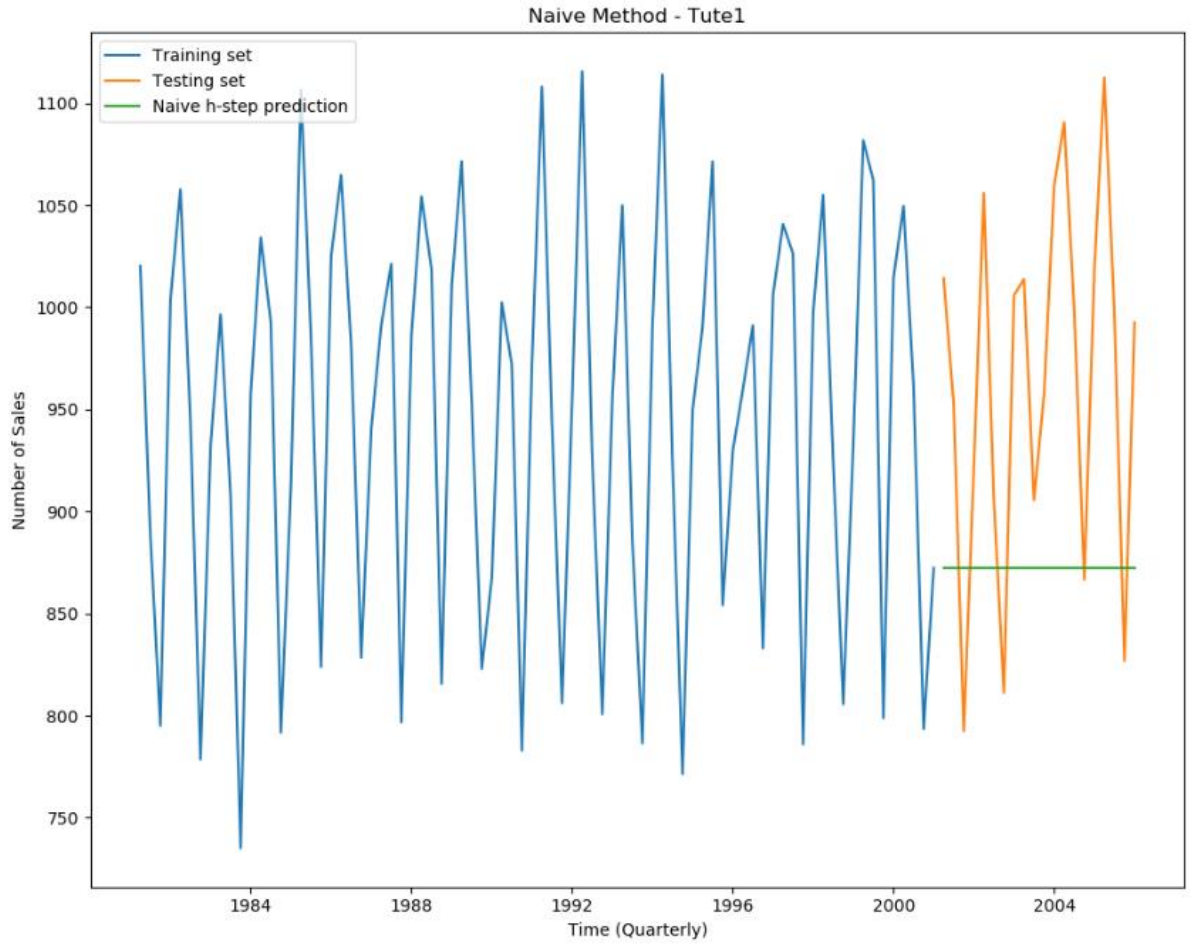

Autocorrelation plot for Forecast Error (SES Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−15
−10
−5
0
5
10
15

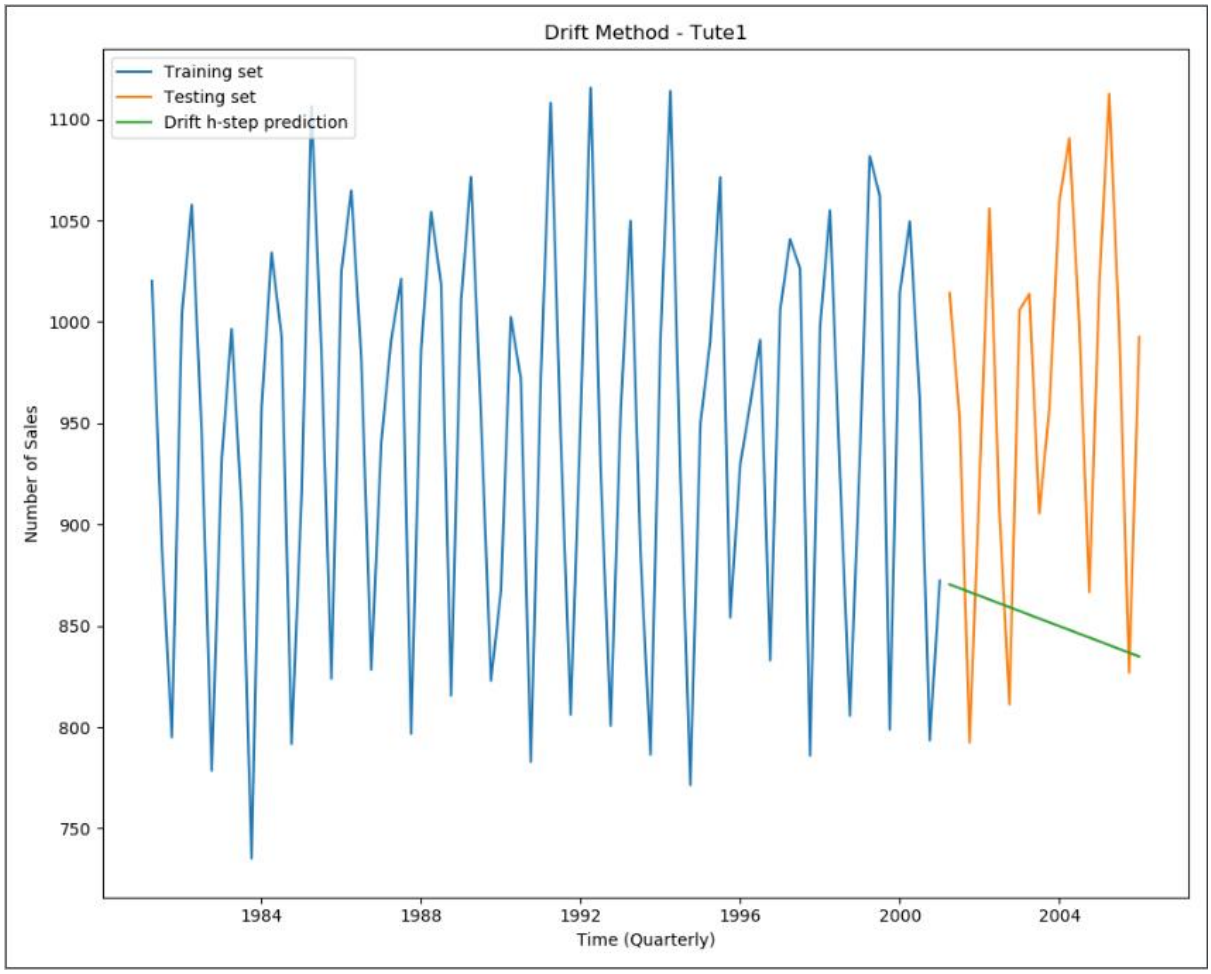

Autocorrelation plot for Forecast Error (Holt's Linear Method)
0.04
0.02
0.00
−0.02
−0.04
Magnitude
−15
−10
−5
0
5
10
15
Lags

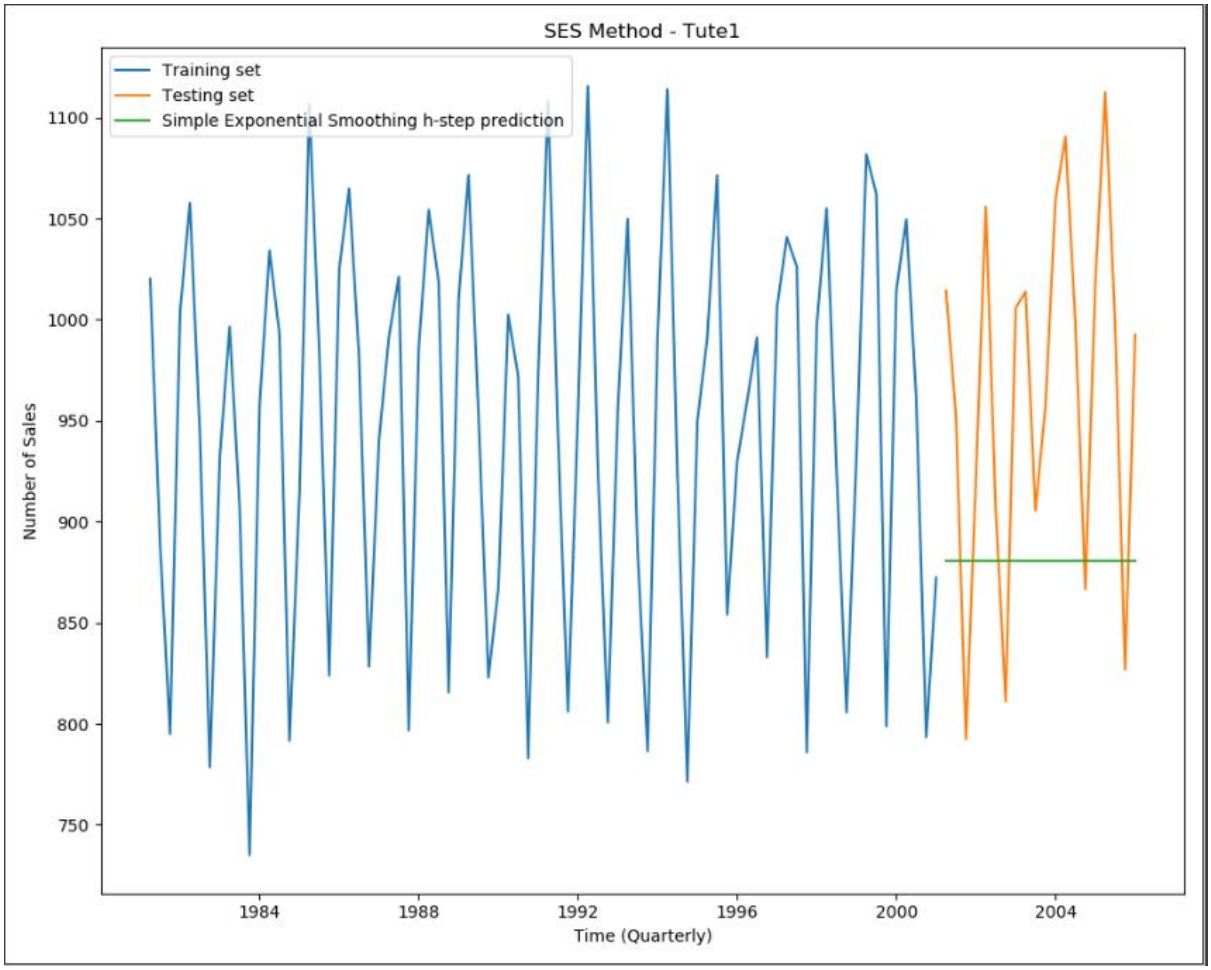

Autocorrelation plot for Forecast Error (Holt's winter Seasonal Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−15
−10
−5
0
5
10
15

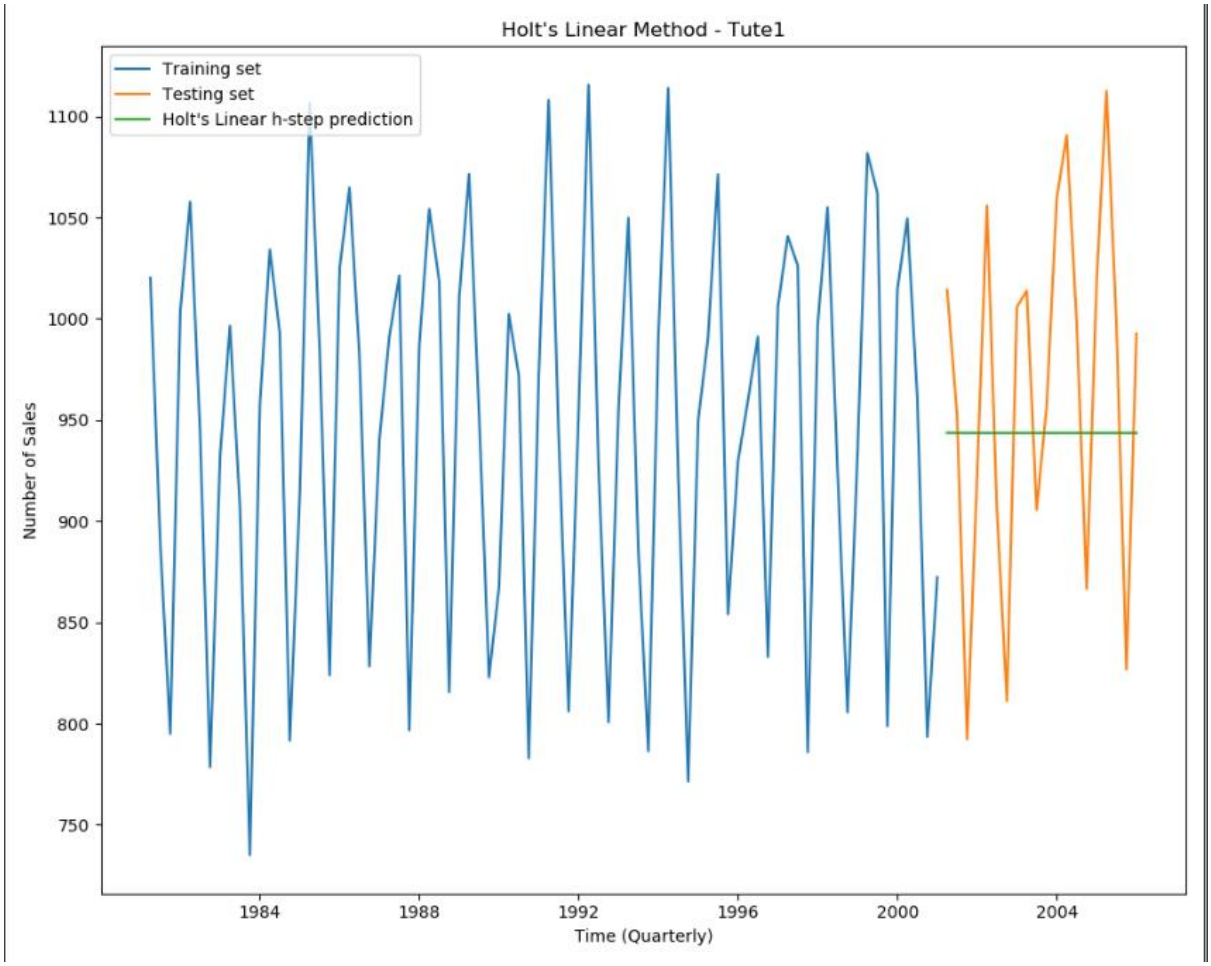
```
#%%==========================================================================
#5. # DATA SET - Tute1
# %%--------------------------------------------------------------------------

************************ Tute1 Dataset results **********************
Mean Square Error of prediction errors for Average method:  10655.610456466507
Mean Square Error of forecast errors for Average method:  8329.793699999993
Mean of prediction errors for Average method:  2.3022752350665976
Variance of prediction errors for Average method:  10650.309985208507
Variance of forecast errors for Average method:  7914.245475000001
Mean Square Error of prediction errors for Naive method:  20158.124430379754
Mean Square Error of forecast errors for Naive method:  16515.880500000007
Mean of prediction errors for Naive method:  -1.8721518987341783
Variance of prediction errors for Naive method:  20154.619477647815
Variance of forecast errors for Naive method:  7914.245475000001
Mean Square Error of prediction errors for Drift method:  21602.639345564097
Mean Square Error of forecast errors for Drift method:  21054.50801450089
Mean of prediction errors for Drift method:  4.272024907863501
Variance of prediction errors for Drift method:  21584.389148750684
Variance of forecast errors for Drift method:  8420.164665995035
Mean Square Error of prediction errors for SES method:  15457.181134379398
Mean Square Error of forecast errors for SES method:  15051.769711754296
Mean of prediction errors for SES method:  -3.5351612578184874
Variance of prediction errors for SES method:  15444.683769260619
Variance of forecast errors for SES method:  7914.245475000001
Mean Square Error of prediction errors for Holt's Linear method:
10286.710143339806
Mean Square Error of forecast errors for Holt's Linear method:  8375.41105225168
Mean of prediction errors for Holt's Linear method:  8.825486630339082
Variance of prediction errors for Holt's Linear method:  10208.820929077512
Variance of forecast errors for Holt's Linear method:  7914.088261311188
Mean Square Error of prediction errors for Holt's Winter Seasonal method:
2304.036191157657
Mean Square Error of forecast errors for Holt's Winter Seasonal method:  nan
Mean of prediction errors for Holt's Winter Seasonal method:  2.0655411823806746
Variance of prediction errors for Holt's Winter Seasonal method:
2299.7697307815465
Variance of forecast errors for Holt's Winter Seasonal method:  nan
Q value of forecast errors for Average method:  25.462830810652367
Q value of forecast errors for Naive method:  25.46283081065237
Q value of forecast errors for Drift method:  22.169687620680403
Q value of forecast errors for SES method:  25.46283081065237
Q value of forecast errors for Holt's Linear method:  25.459449082394205
Q value of forecast errors for Holt's Winter Seasonal method:  nan
Correlation Coefficient between Forecast Error and Test set for Average Method:
0.9999999999999998
Correlation Coefficient between Forecast Error and Test set for Naive Method: 1.0
Correlation Coefficient between Forecast Error and Test set for Drift Method:
0.9933419598521821
Correlation Coefficient between Forecast Error and Test set for SES Method: 1.0
Correlation Coefficient between Forecast Error and Test set for Holt's Linear
Method: 0.9999999797537357
Correlation Coefficient between Forecast Error and Test set for Holt's winter
seasonal Method: nan
         Q_val    MSE(P)    MSE(F)    var(P)   var(F)  corrcoeff
Methods
Average  25.46  10655.61   8329.79  10650.31  7914.25       1.00
Naive    25.46  20158.12  16515.88  20154.62  7914.25       1.00
Drift    22.17  21602.64  21054.51  21584.39  8420.16       0.99
```

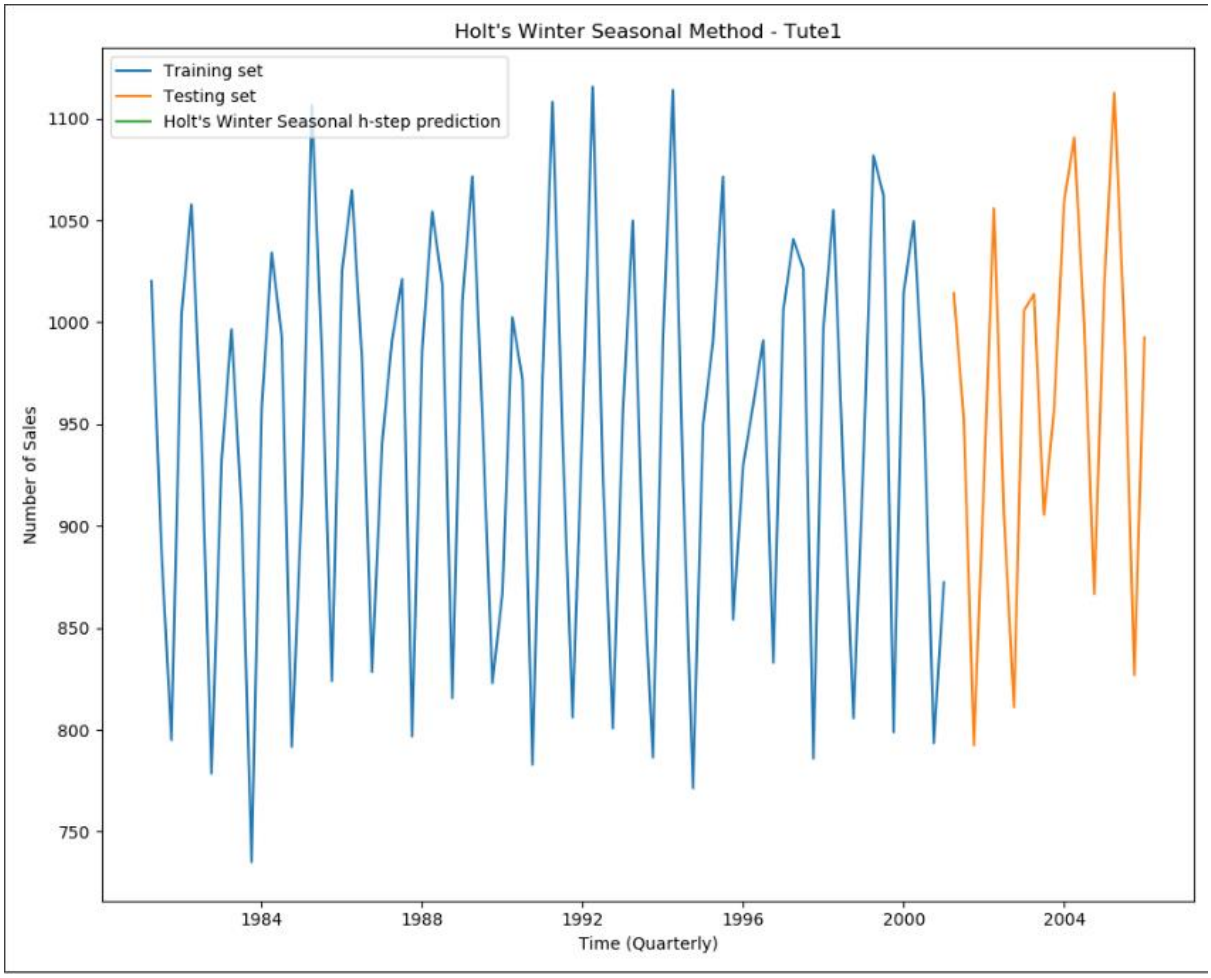
SES	25.46	15457.18	15051.77	15444.68	7914.25	1.00
HoltL	25.46	10286.71	8375.41	10208.82	7914.09	1.00
HoltW	NaN	2304.04	NaN	2299.77	NaN	NaN

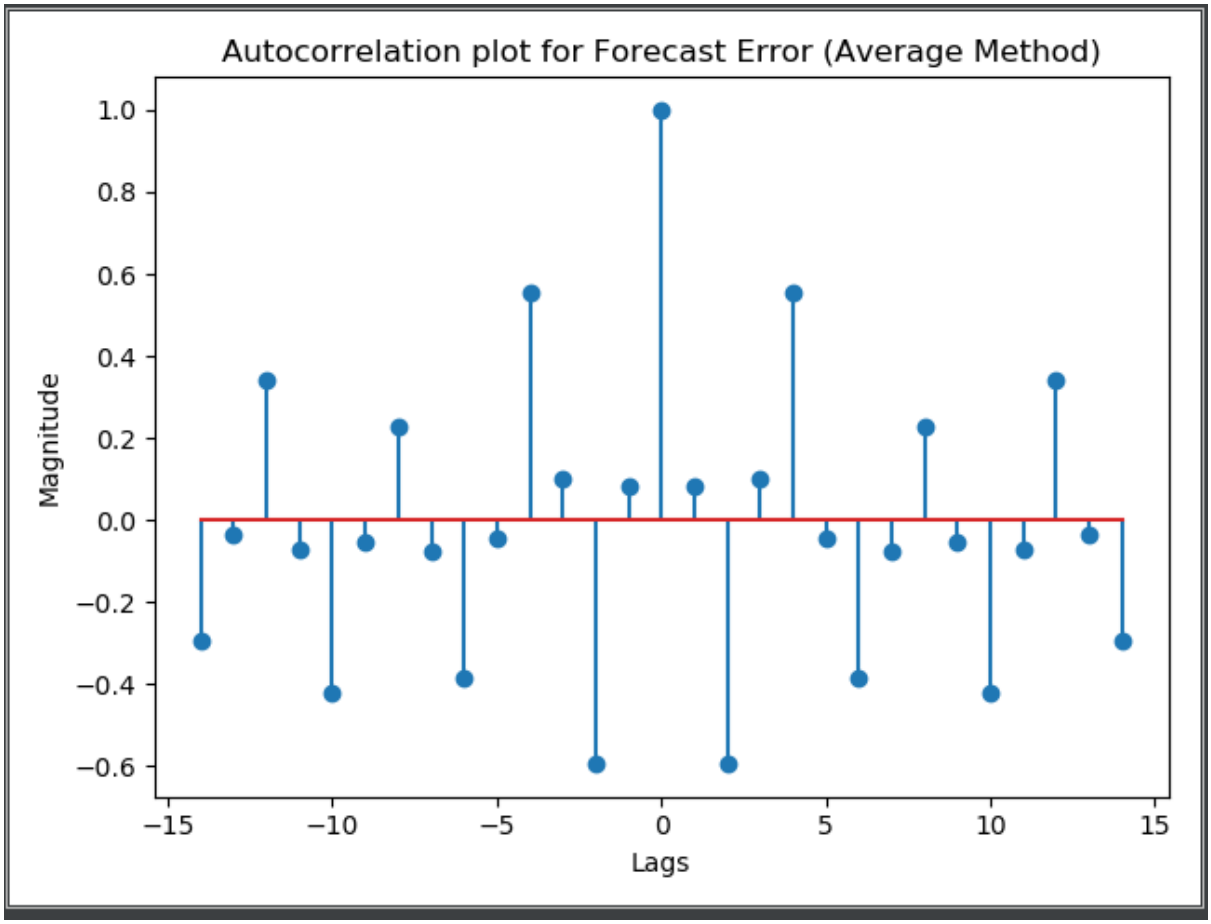


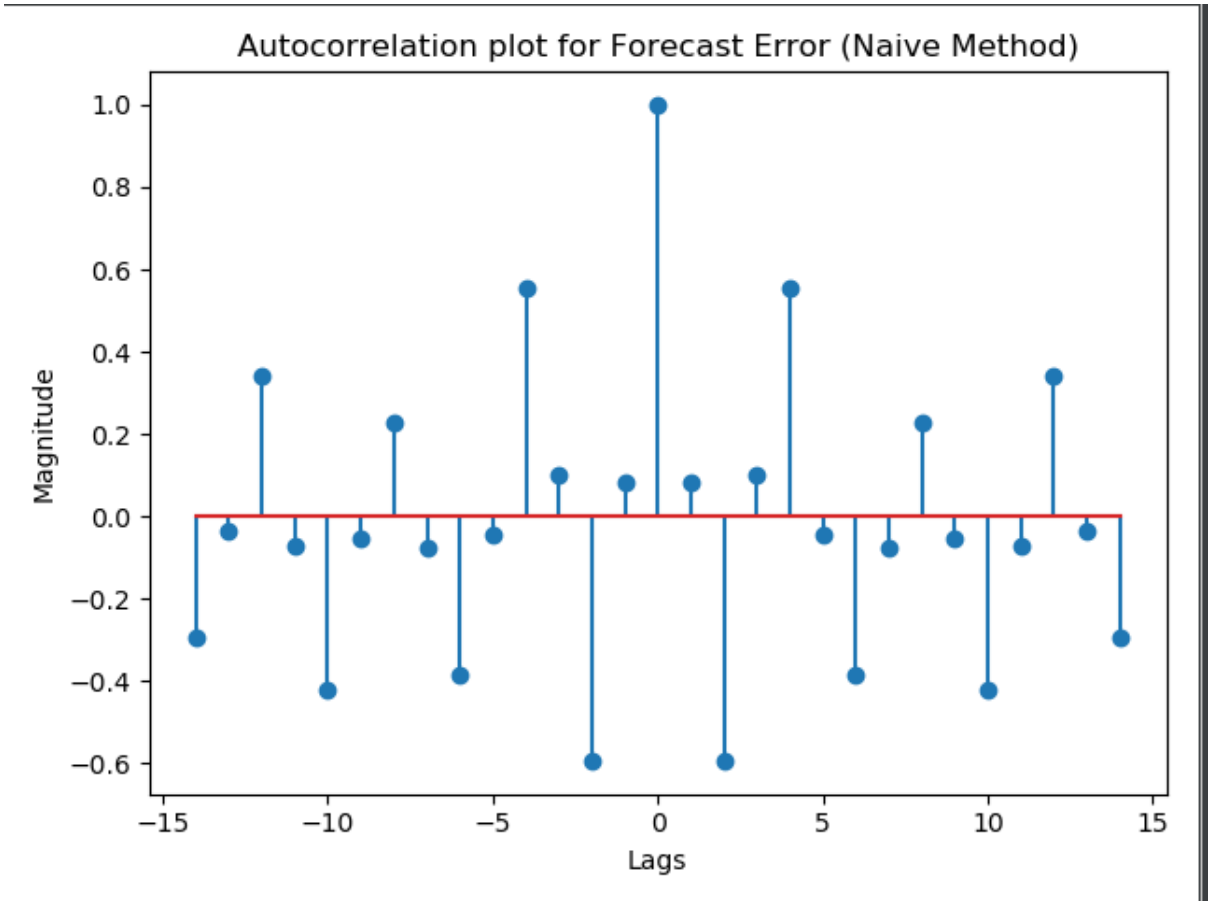

Drift Method - Tute1
Training set
Testing set
Drift h-step prediction
Number of Sales
Time (Quarterly)
1100
1050
1000
950
900
850
800
750
1984
1988
1992
1996
2000
2004

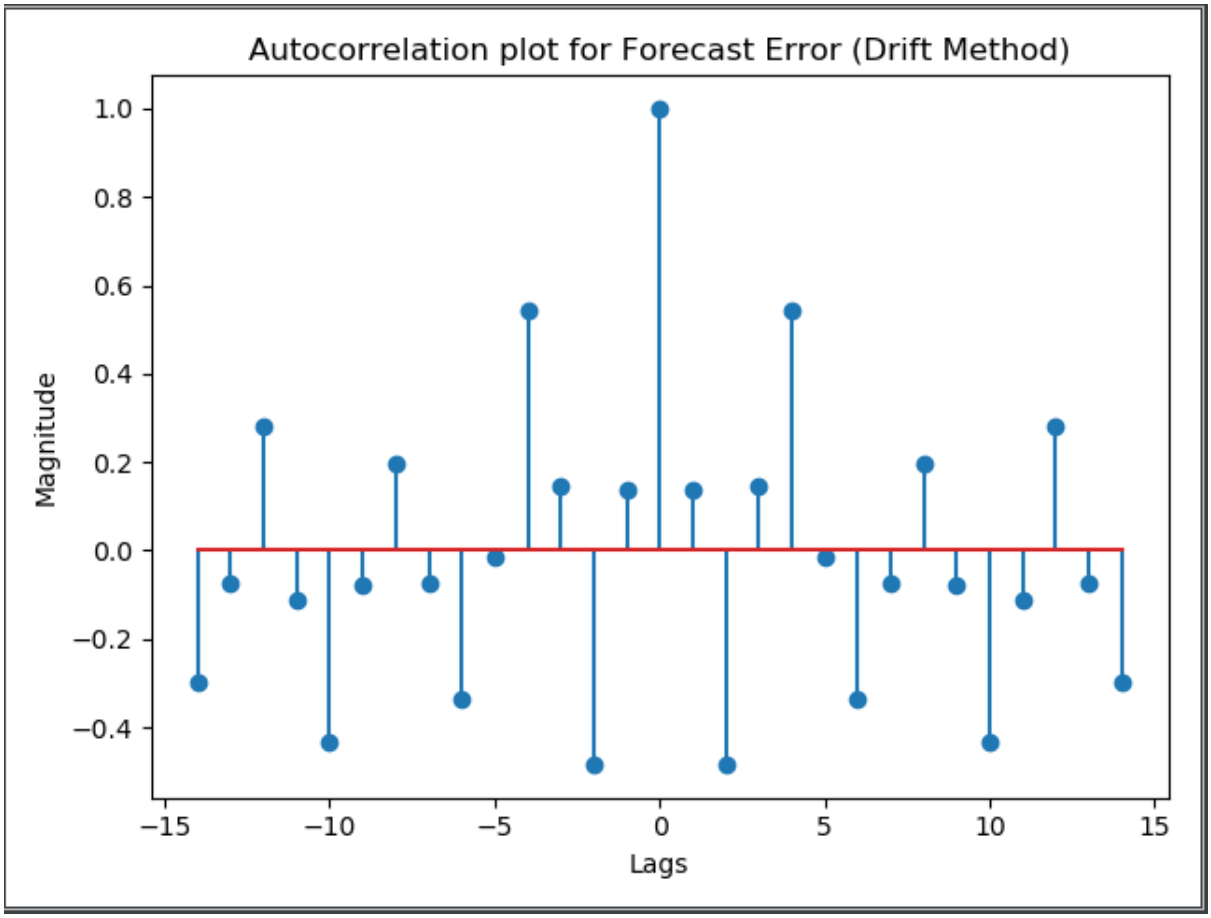

SES Method - Tute1
Training set
Testing set
Simple Exponential Smoothing h-step prediction
Number of Sales
1100
1050
1000
950
900
850
800
750
1984
1988
1992
1996
2000
2004
Time (Quarterly)

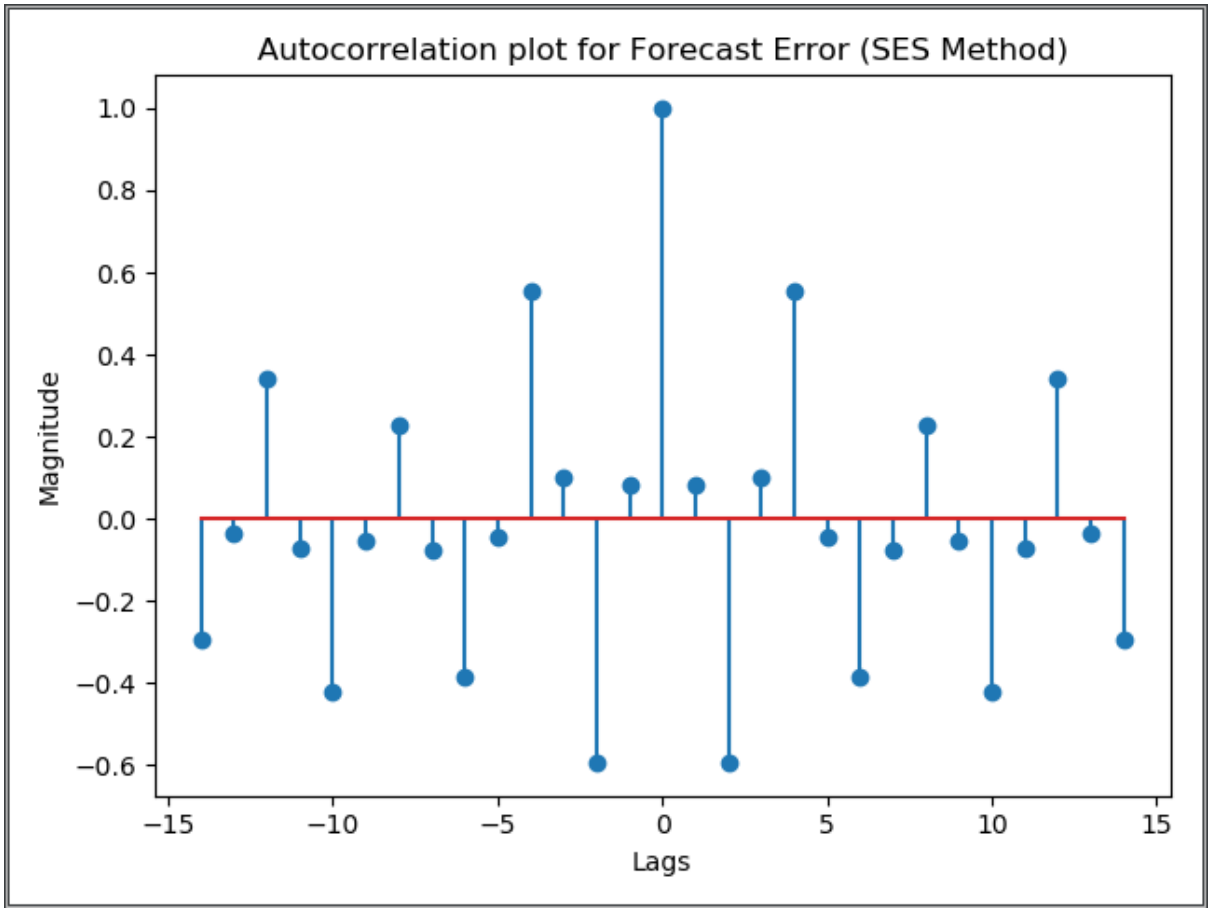

Holt's Linear Method - Tute1
Training set
Testing set
Holt's Linear h-step prediction
Number of Sales
1100
1050
1000
950
900
850
800
750
1984
1988
1992
1996
2000
2004
Time (Quarterly)

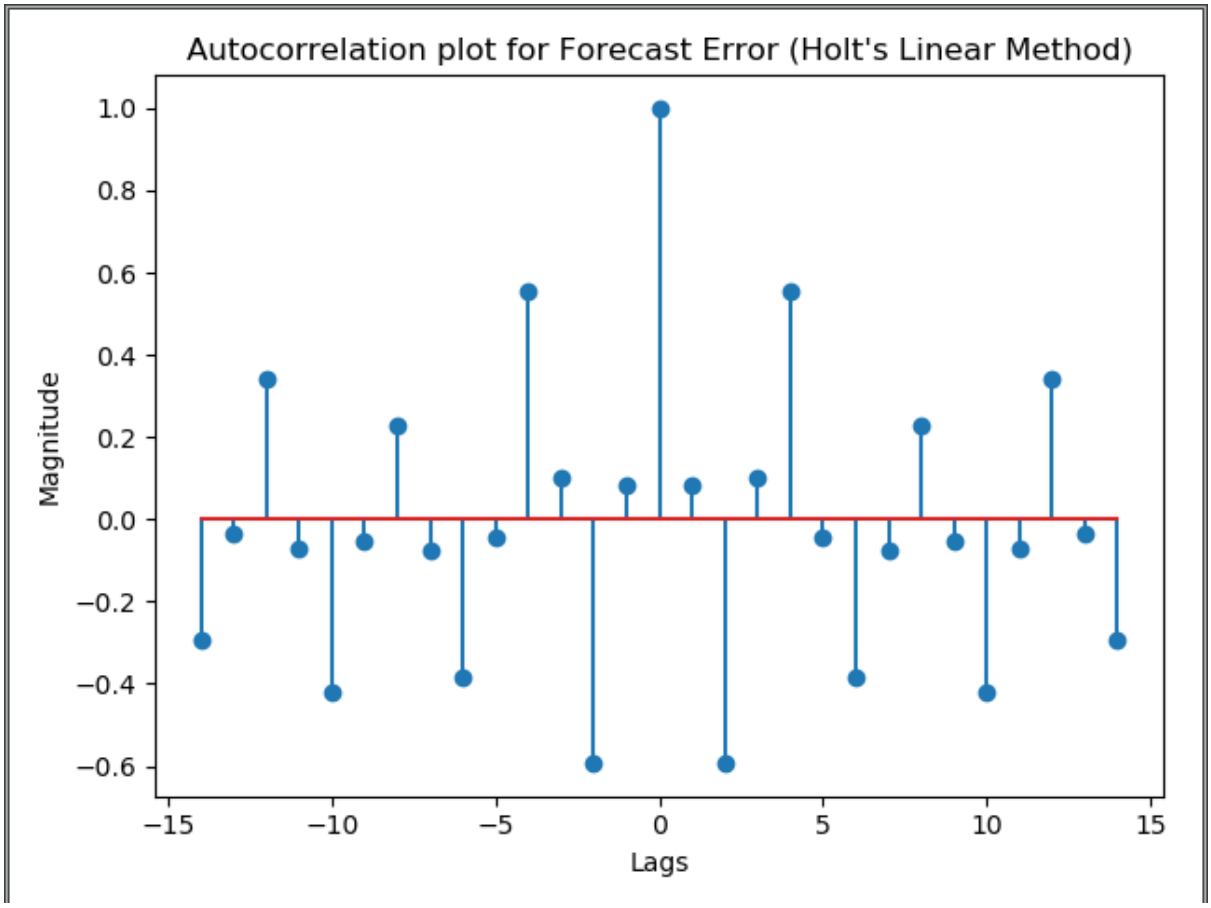

Holt's Winter Seasonal Method - Tute1
Training set
Testing set
Holt's Winter Seasonal h-step prediction
Number of Sales
1100
1050
1000
950
900
850
800
750
1984
1988
1992
1996
2000
2004
Time (Quarterly)

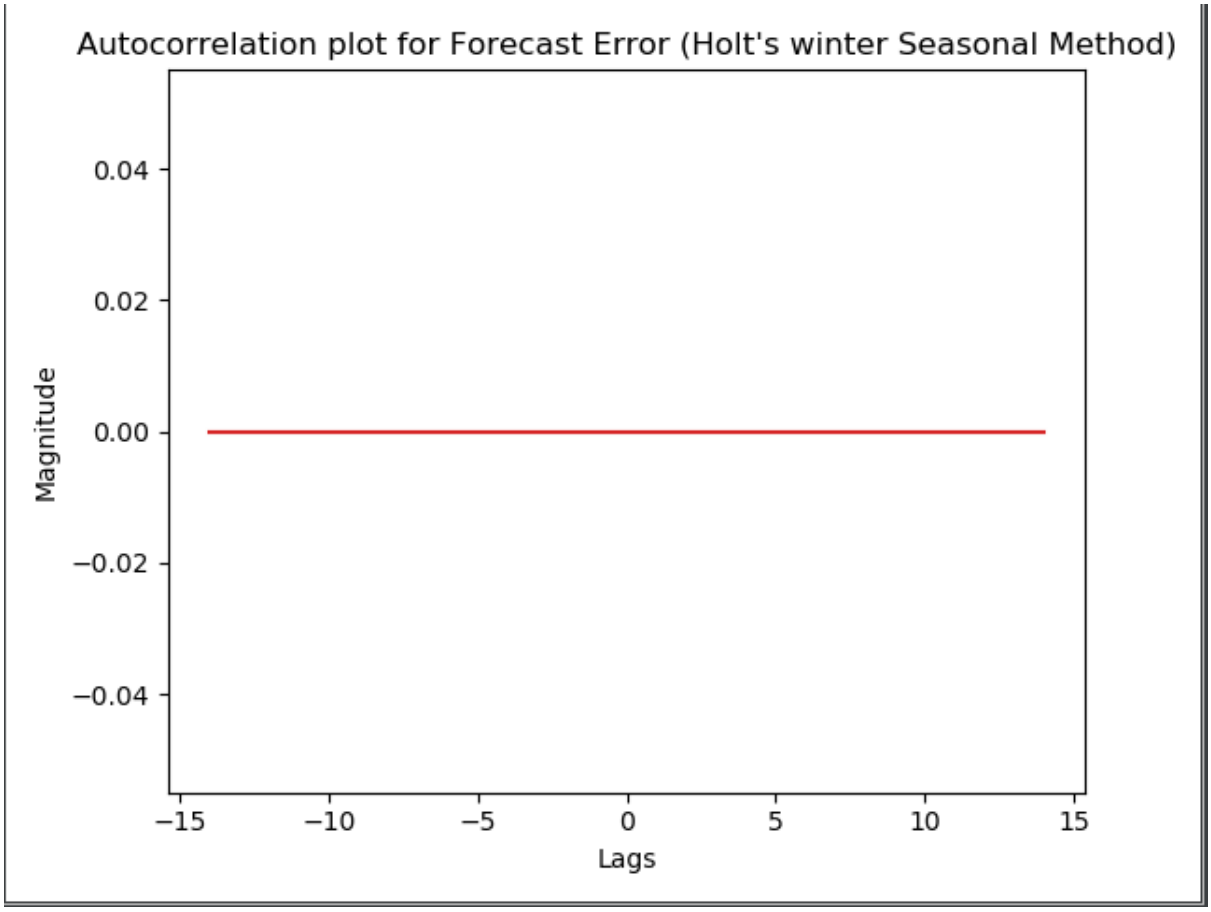

Autocorrelation plot for Forecast Error (Average Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−15
−10
−5
0
5
10
15


Autocorrelation plot for Forecast Error (Naive Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−15
−10
−5
0
5
10
15


Autocorrelation plot for Forecast Error (Drift Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−15
−10
−5
0
5
10
15

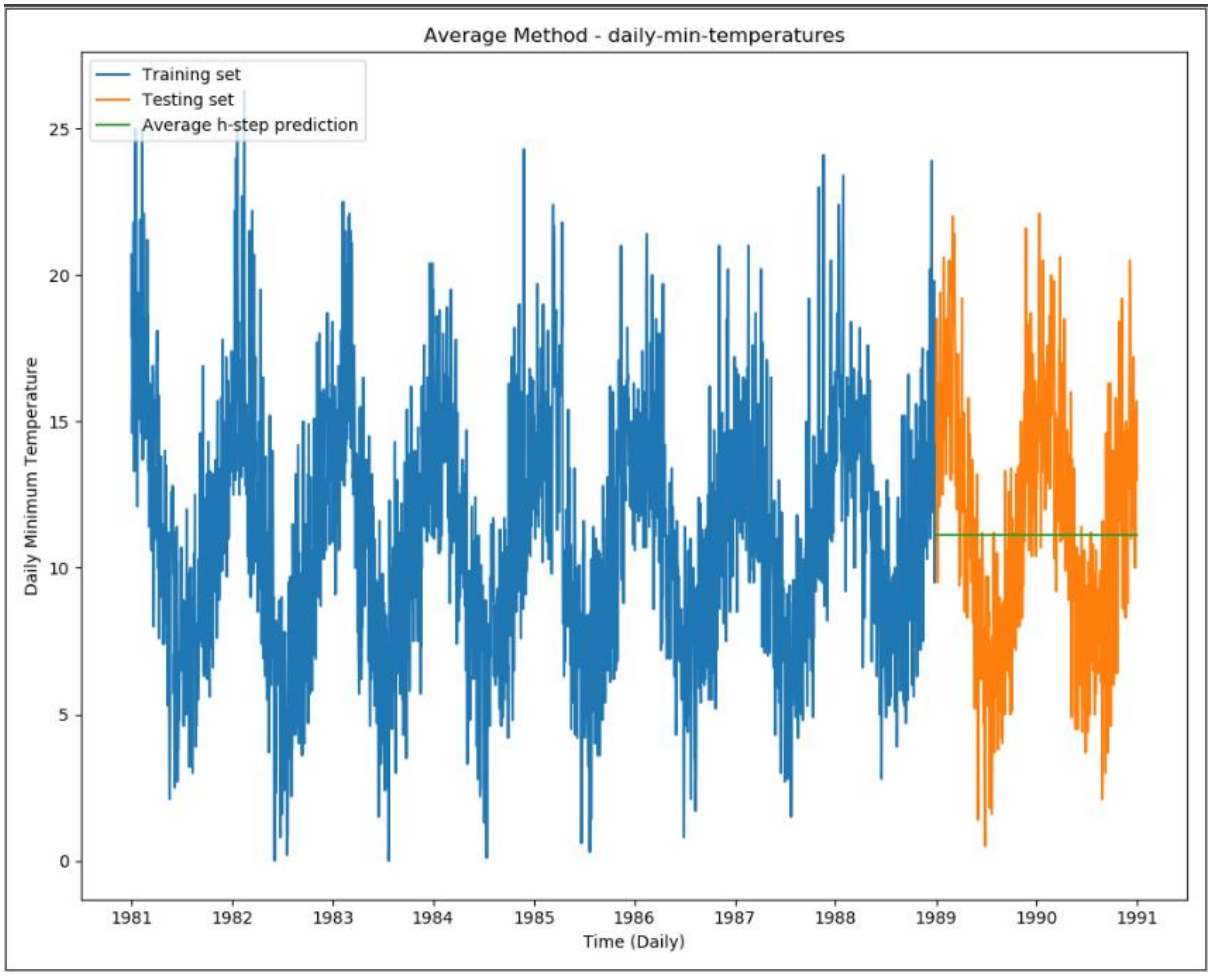

Autocorrelation plot for Forecast Error (SES Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−15
−10
−5
0
5
10
15

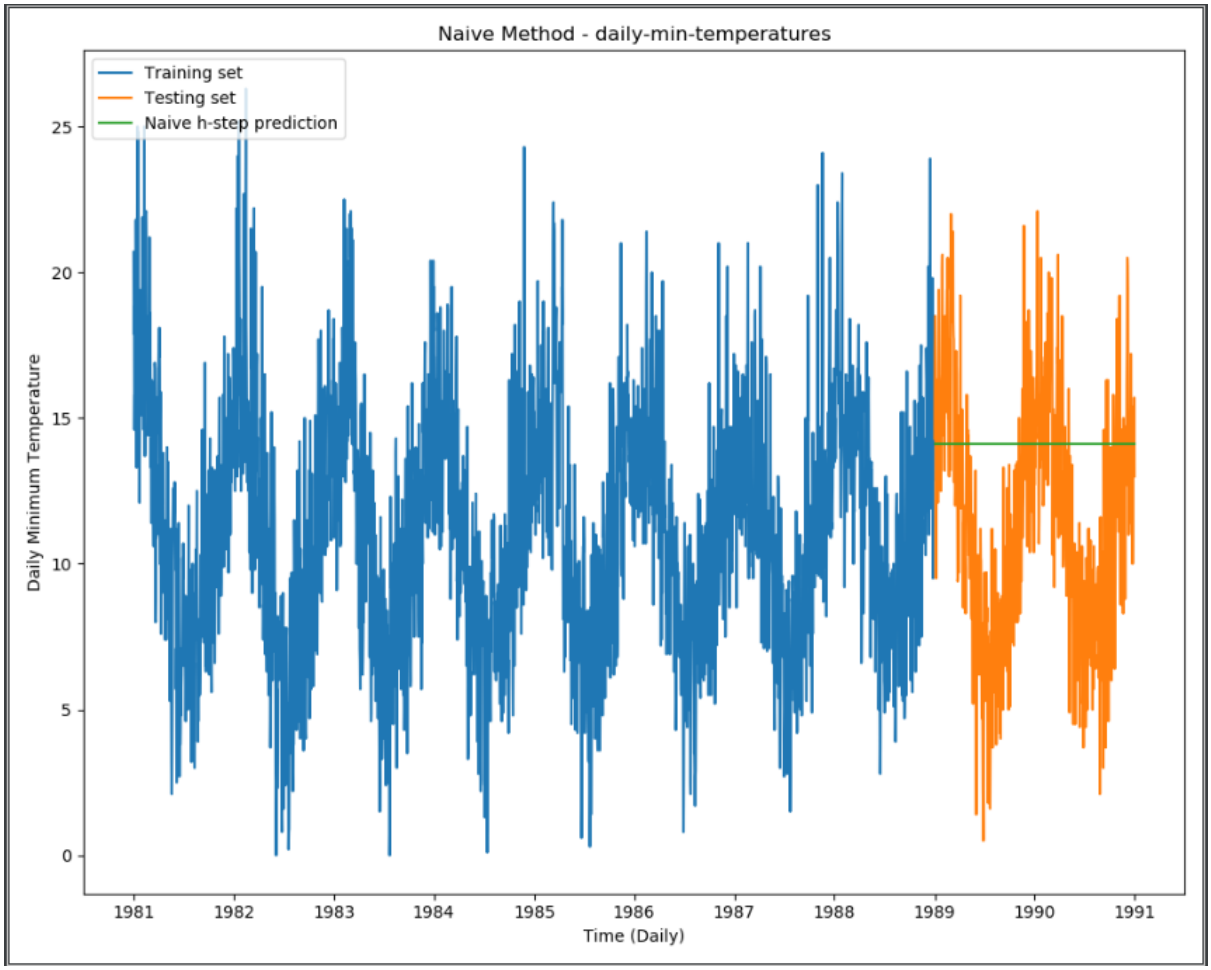

Autocorrelation plot for Forecast Error (Holt's Linear Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−15
−10
−5
0
5
10
15

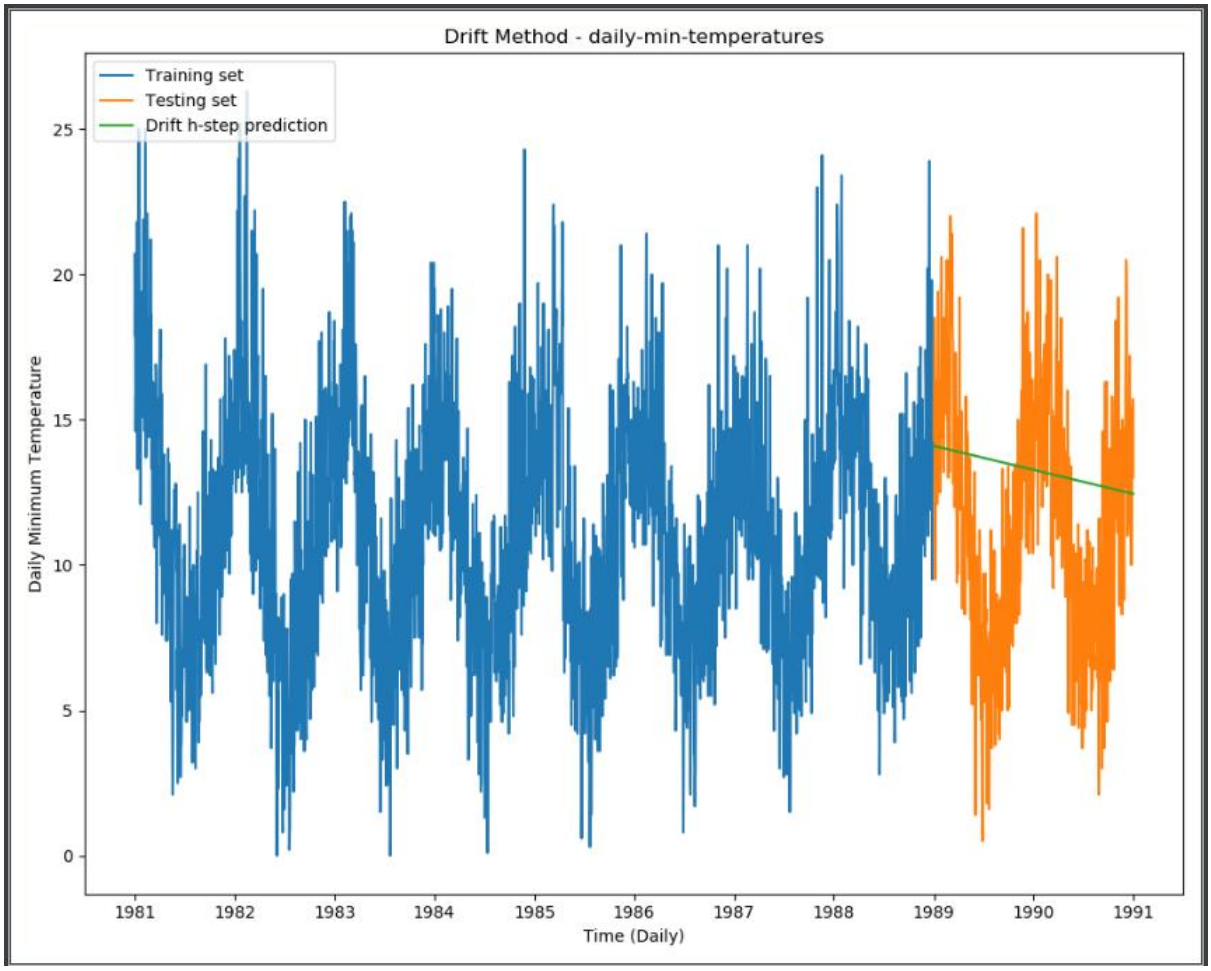

Autocorrelation plot for Forecast Error (Holt's winter Seasonal Method)
Magnitude
Lags
0.04
0.02
0.00
−0.02
−0.04
−15
−10
−5
0
5
10
15

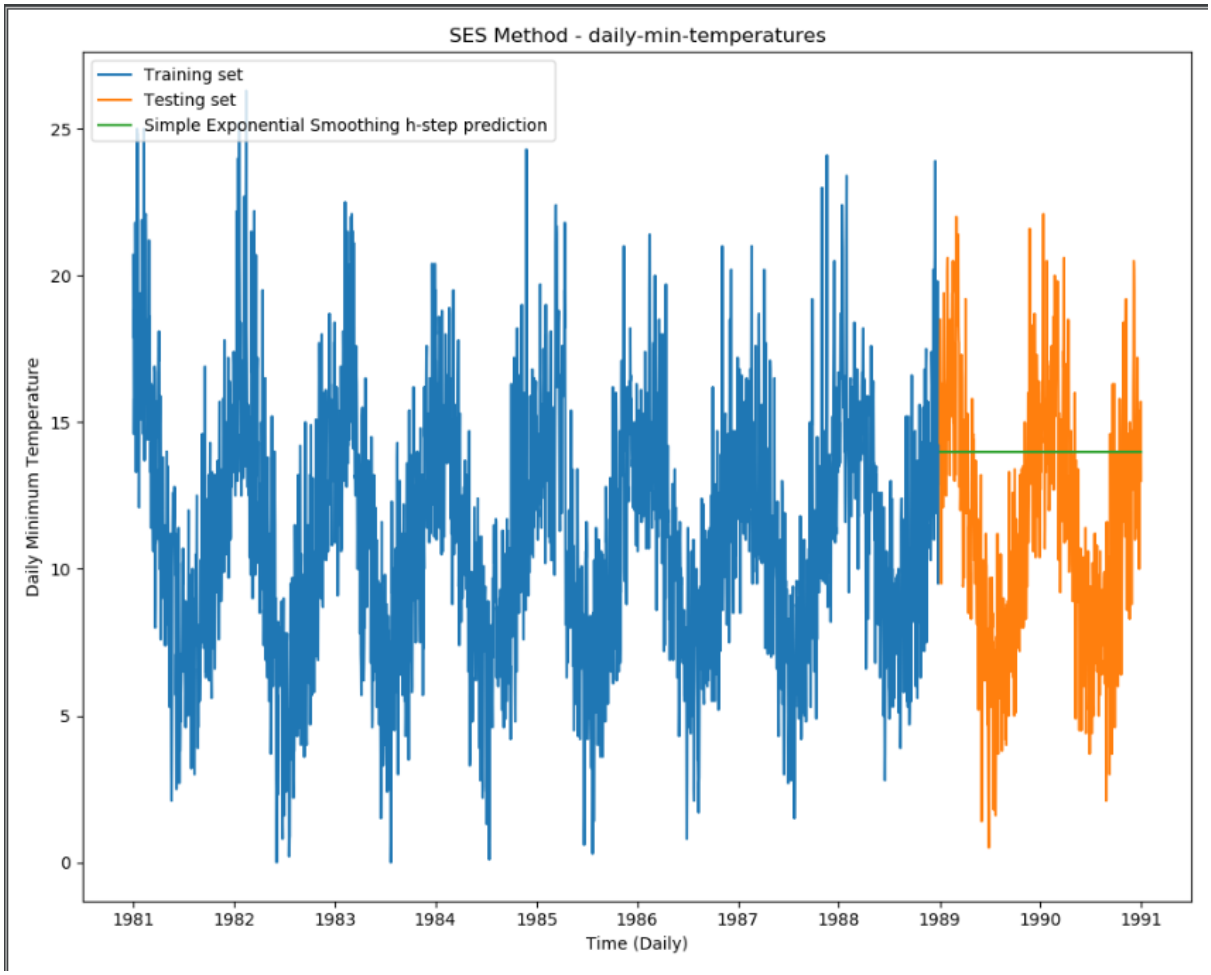
```
************************ daily-min-temperatures Dataset results
**********************
Mean Square Error of prediction errors for Average method:  16.52744634436777
Mean Square Error of forecast errors for Average method:  16.970755939200593
Mean of prediction errors for Average method:  -0.4248361485268812
Variance of prediction errors for Average method:  16.346960591272616
Variance of forecast errors for Average method:  16.841155939200615
Mean Square Error of prediction errors for Naive method:  7.78570058239123
Mean Square Error of forecast errors for Naive method:  23.780410958904117
Mean of prediction errors for Naive method:  -0.0022610483042137647
Variance of prediction errors for Naive method:  7.785695470051794
Variance of forecast errors for Naive method:  16.841155939200615
Mean Square Error of prediction errors for Drift method:  7.812360774644045
Mean Square Error of forecast errors for Drift method:  19.91198942652102
Mean of prediction errors for Drift method:  0.015711157148715328
Variance of prediction errors for Drift method:  7.812113934185093
Variance of forecast errors for Drift method:  16.64372775150135
Mean Square Error of prediction errors for SES method:  7.0929180407400425
Mean Square Error of forecast errors for SES method:  23.2170687846984
Mean of prediction errors for SES method:  -0.004596909610741578
Variance of prediction errors for SES method:  7.092896909162073
Variance of forecast errors for SES method:  16.8411559392006
Mean Square Error of prediction errors for Holt's Linear method:
7.076695158921313
Mean Square Error of forecast errors for Holt's Linear method:  23.045560850862756
Mean of prediction errors for Holt's Linear method:  -0.0039263640428656545
Variance of prediction errors for Holt's Linear method:  7.076679742586717
Variance of forecast errors for Holt's Linear method:  16.8411559392006
Mean Square Error of prediction errors for Holt's Winter Seasonal method:
6.421777873463634
Mean Square Error of forecast errors for Holt's Winter Seasonal method:
22.885478951288594
Mean of prediction errors for Holt's Winter Seasonal method:  -
0.002929325338810025
Variance of prediction errors for Holt's Winter Seasonal method:
6.421769292516694
Variance of forecast errors for Holt's Winter Seasonal method:  7.904786577001876
Q value of forecast errors for Average method:  4224.030010660942
Q value of forecast errors for Naive method:  4224.030010660942
Q value of forecast errors for Drift method:  4170.046814056614
Q value of forecast errors for SES method:  4224.030010660941
Q value of forecast errors for Holt's Linear method:  4224.030010660941
Q value of forecast errors for Holt's Winter Seasonal method:  371.664959578396
Correlation Coefficient between Forecast Error and Test set for Average Method:
0.9999999999999997
Correlation Coefficient between Forecast Error and Test set for Naive Method:
0.9999999999999997
Correlation Coefficient between Forecast Error and Test set for Drift Method:
0.9932371884975368
Correlation Coefficient between Forecast Error and Test set for SES Method:
0.9999999999999997
Correlation Coefficient between Forecast Error and Test set for Holt's Linear
Method: 0.9999999999999997
Correlation Coefficient between Forecast Error and Test set for Holt's winter
seasonal Method: 0.5189855191920892
           Q_val  MSE(P)  MSE(F)  var(P)  var(F)  corrcoeff
Methods
Average  4224.03   16.53   16.97   16.35   16.84       1.00
Naive    4224.03    7.79   23.78    7.79   16.84       1.00
```

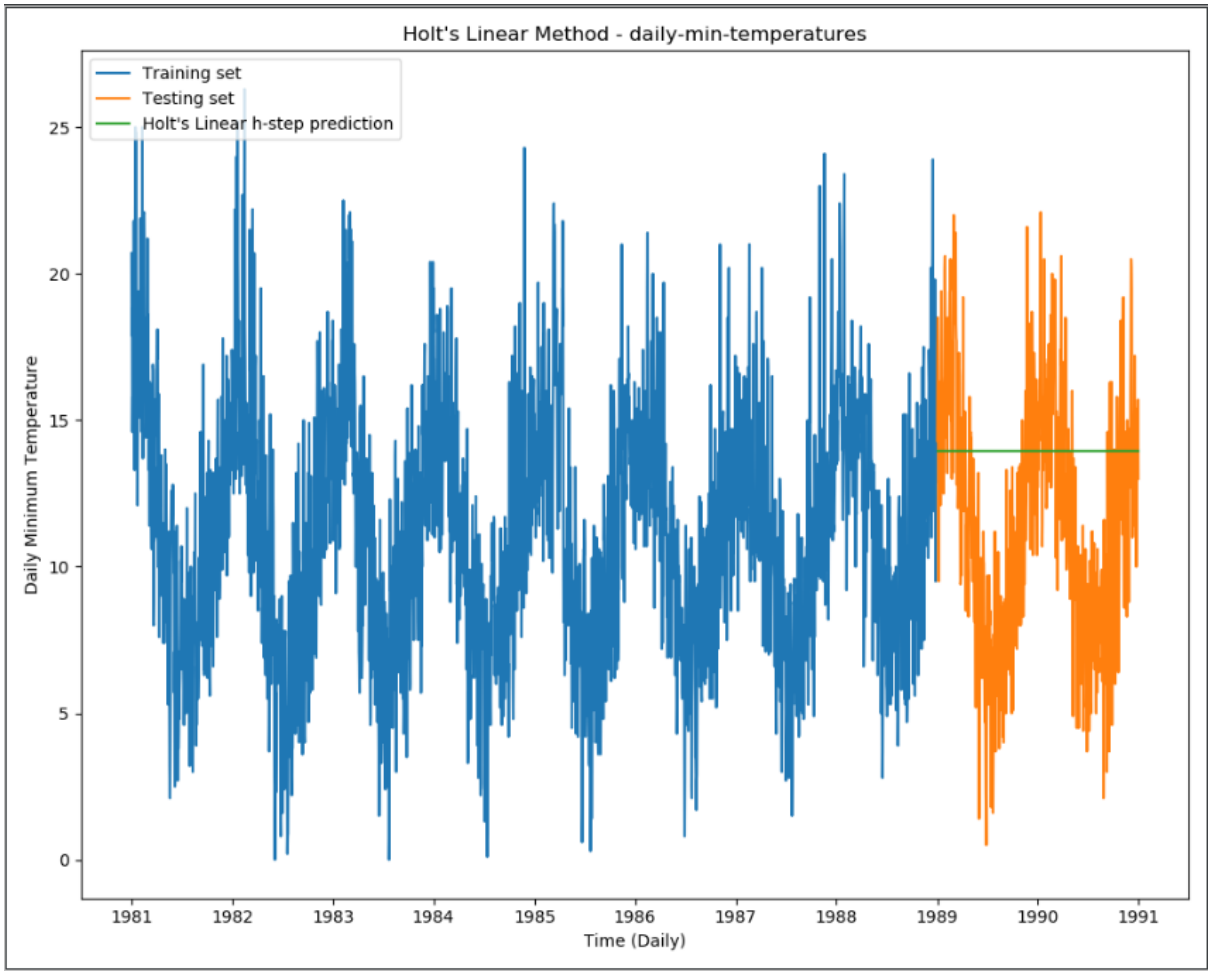
```
Drift    4170.05    7.81   19.91    7.81   16.64       0.99
SES      4224.03    7.09   23.22    7.09   16.84       1.00
HoltL    4224.03    7.08   23.05    7.08   16.84       1.00
HoltW     371.66    6.42   22.89    6.42    7.90       0.52
```

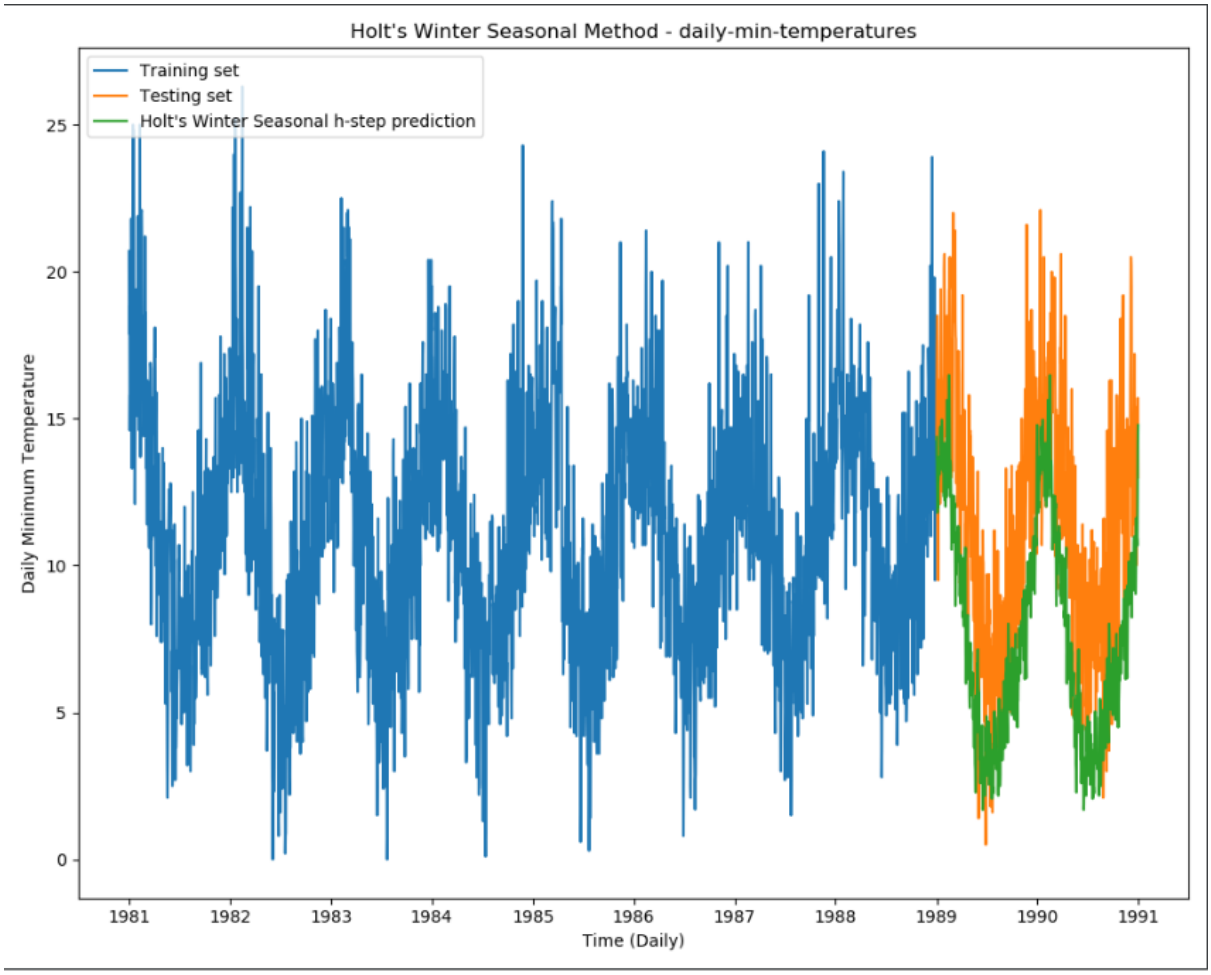


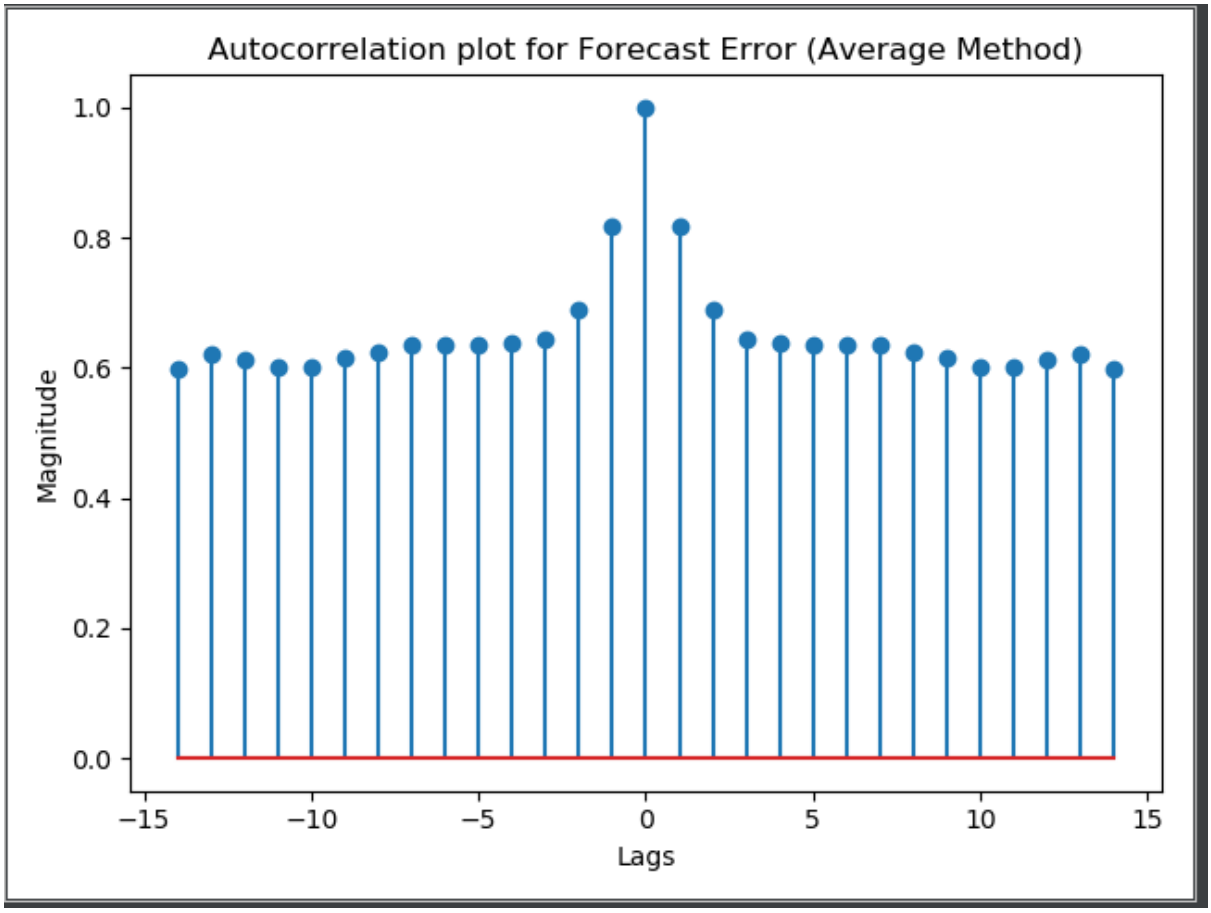

Naive Method - daily-min-temperatures
Training set
Testing set
Naive h-step prediction
Daily Minimum Temperature
Time (Daily)
0
5
10
15
20
25
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991

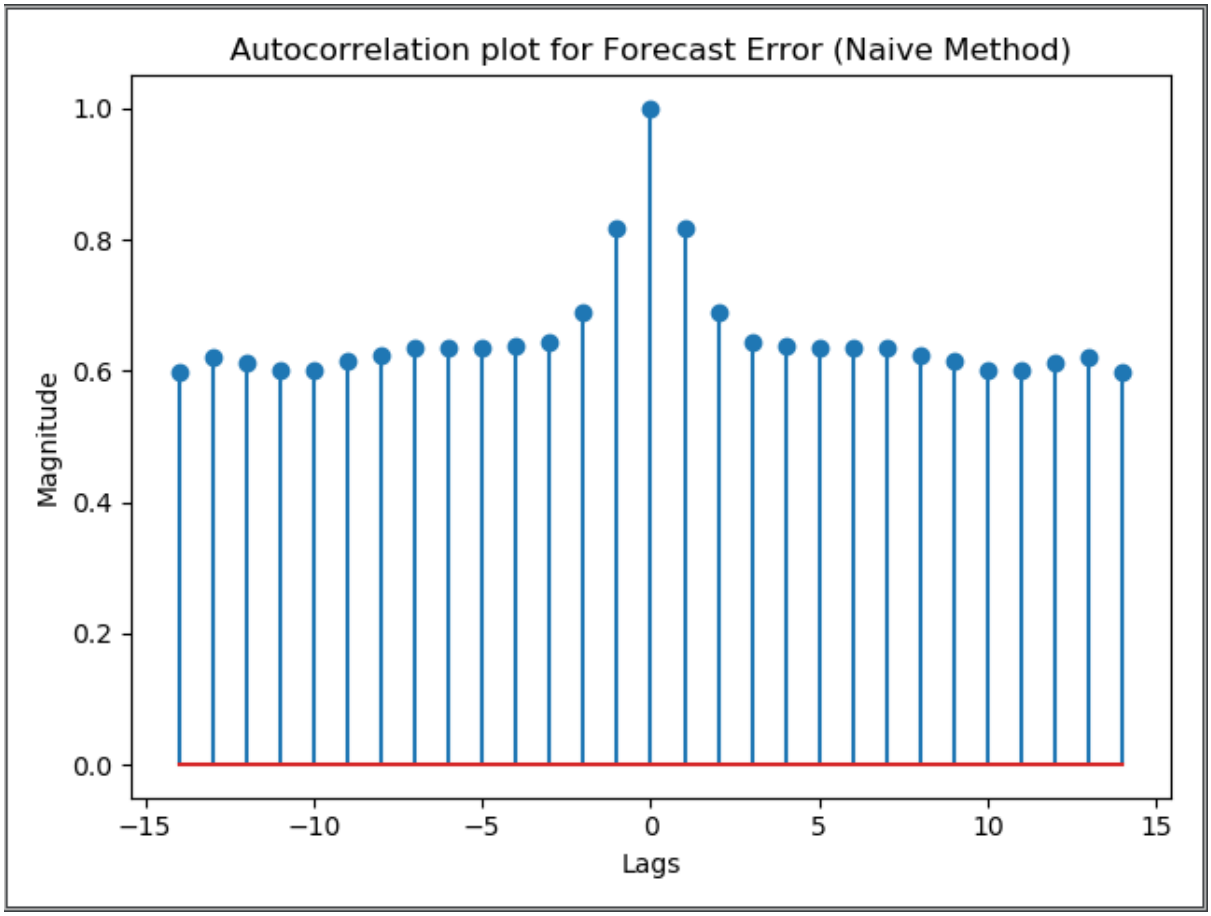

Drift Method - daily-min-temperatures
Training set
Testing set
Drift h-step prediction
Daily Minimum Temperature
Time (Daily)
0
5
10
15
20
25
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991

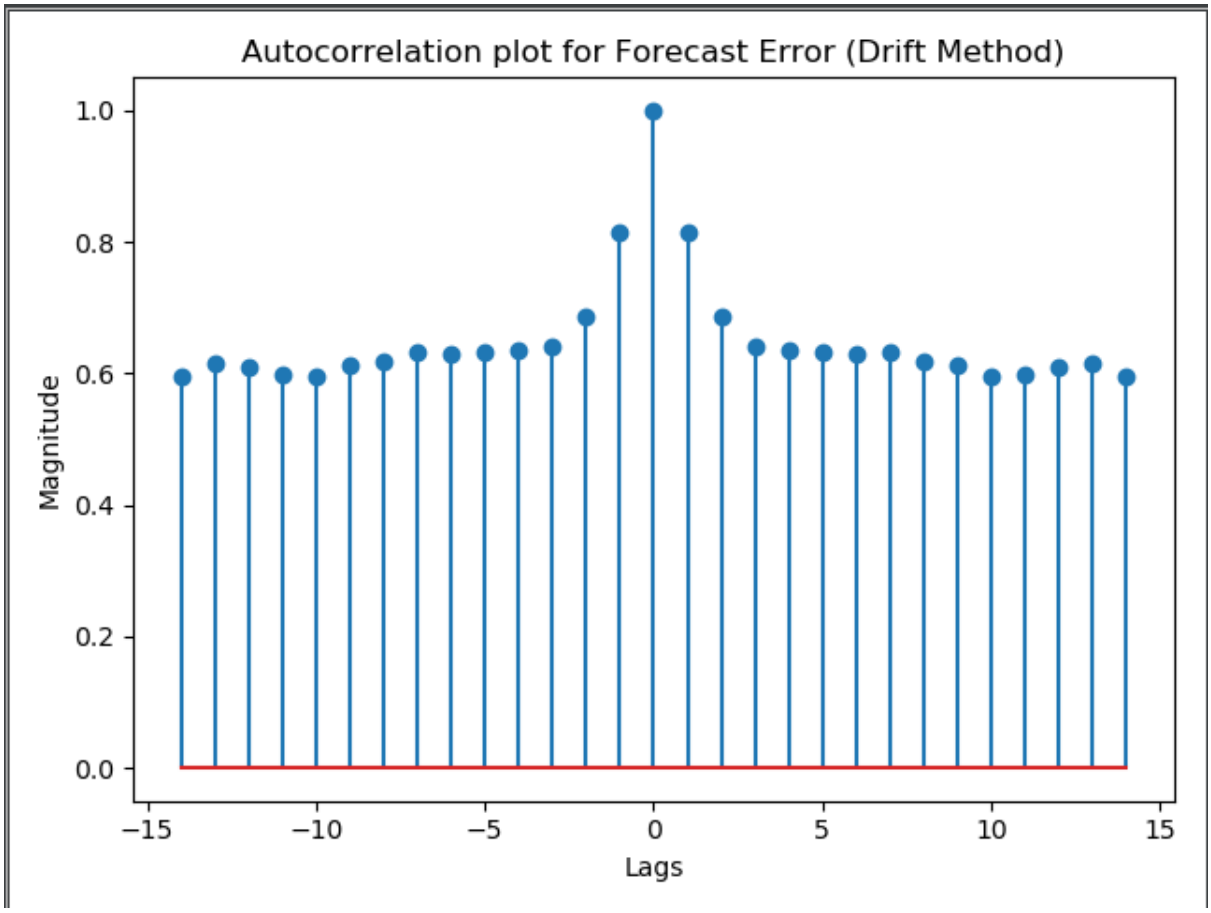

SES Method - daily-min-temperatures
Training set
Testing set
Simple Exponential Smoothing h-step prediction
Daily Minimum Temperature
Time (Daily)
0
5
10
15
20
25
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991

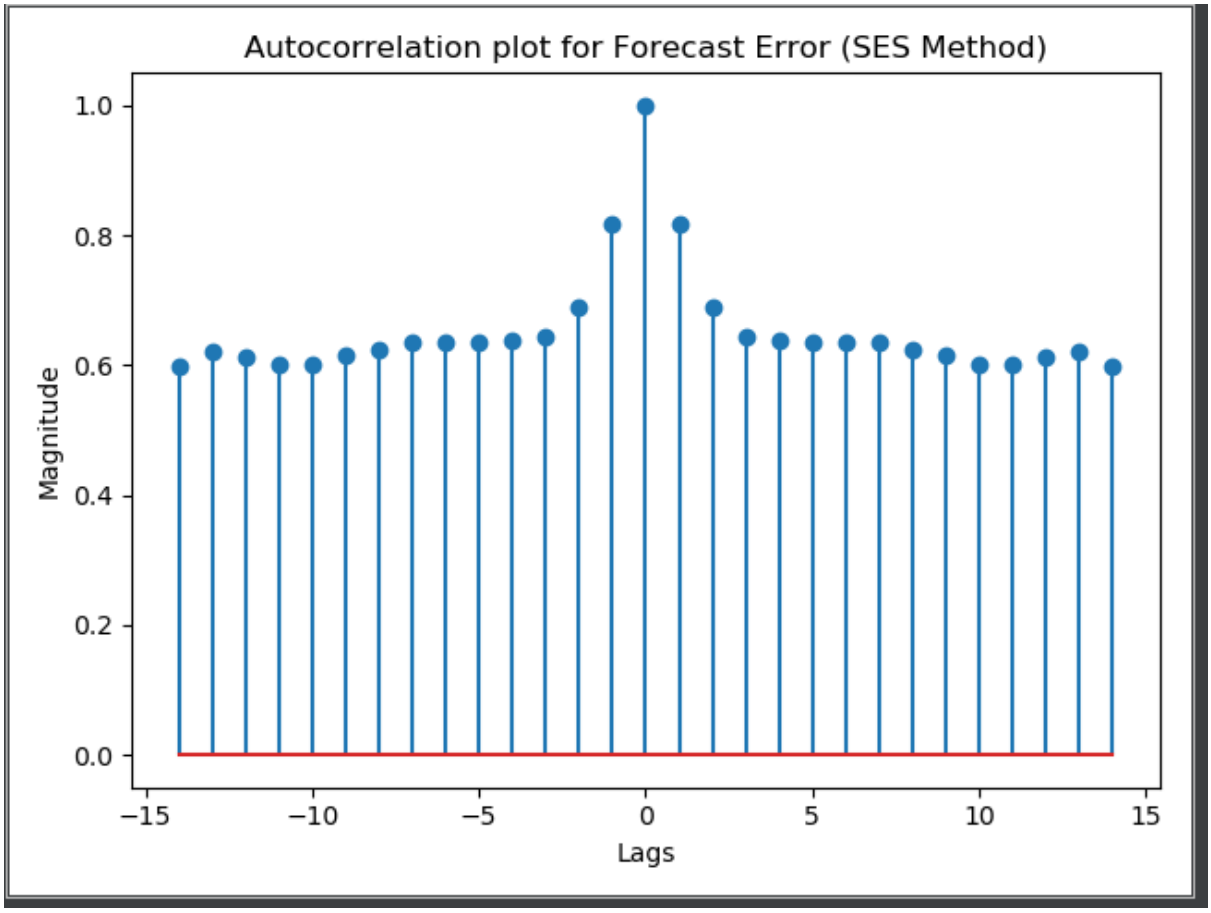

Holt's Linear Method - daily-min-temperatures
Training set
Testing set
Holt's Linear h-step prediction
Daily Minimum Temperature
Time (Daily)
0
5
10
15
20
25
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991

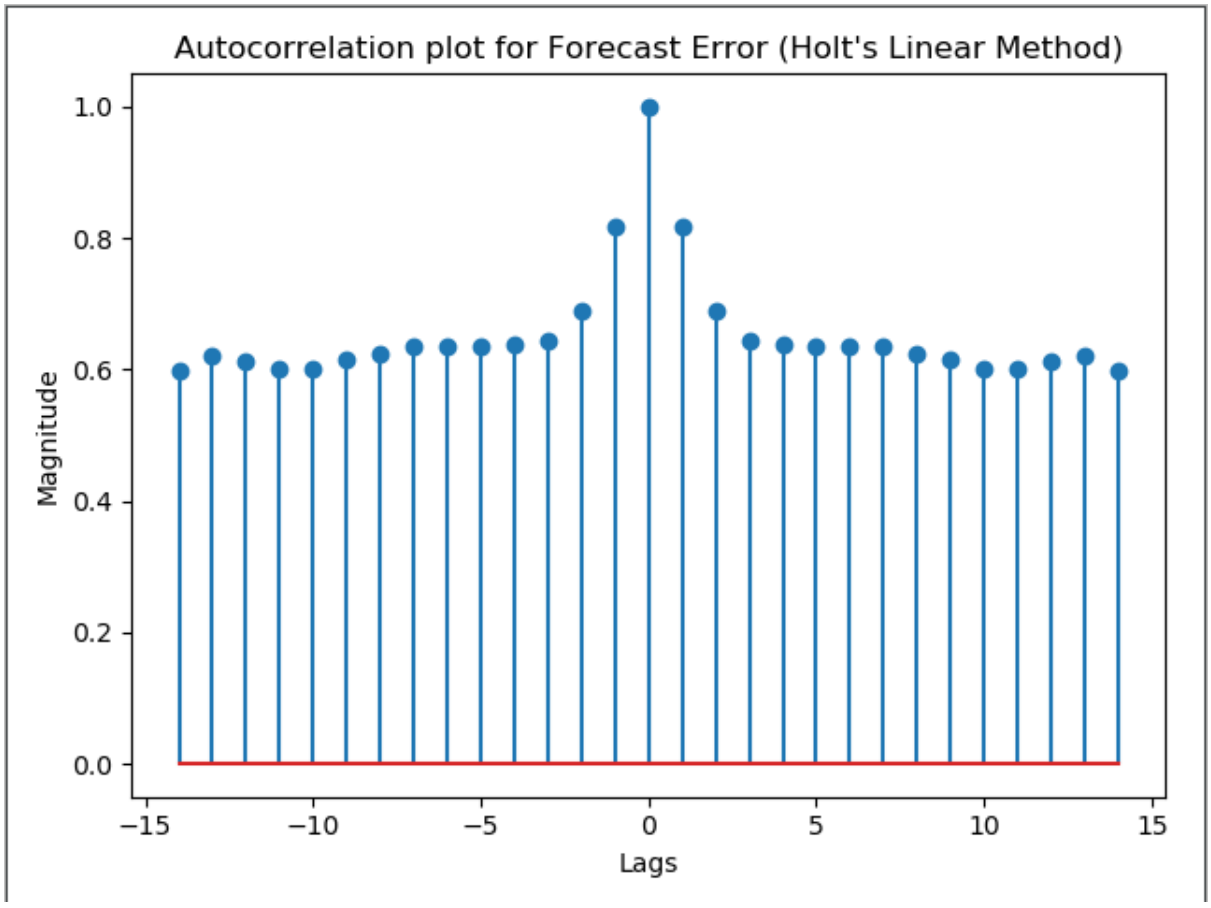

Holt's Winter Seasonal Method - daily-min-temperatures
Training set
Testing set
Holt's Winter Seasonal h-step prediction
Daily Minimum Temperature
Time (Daily)
0
5
10
15
20
25
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991

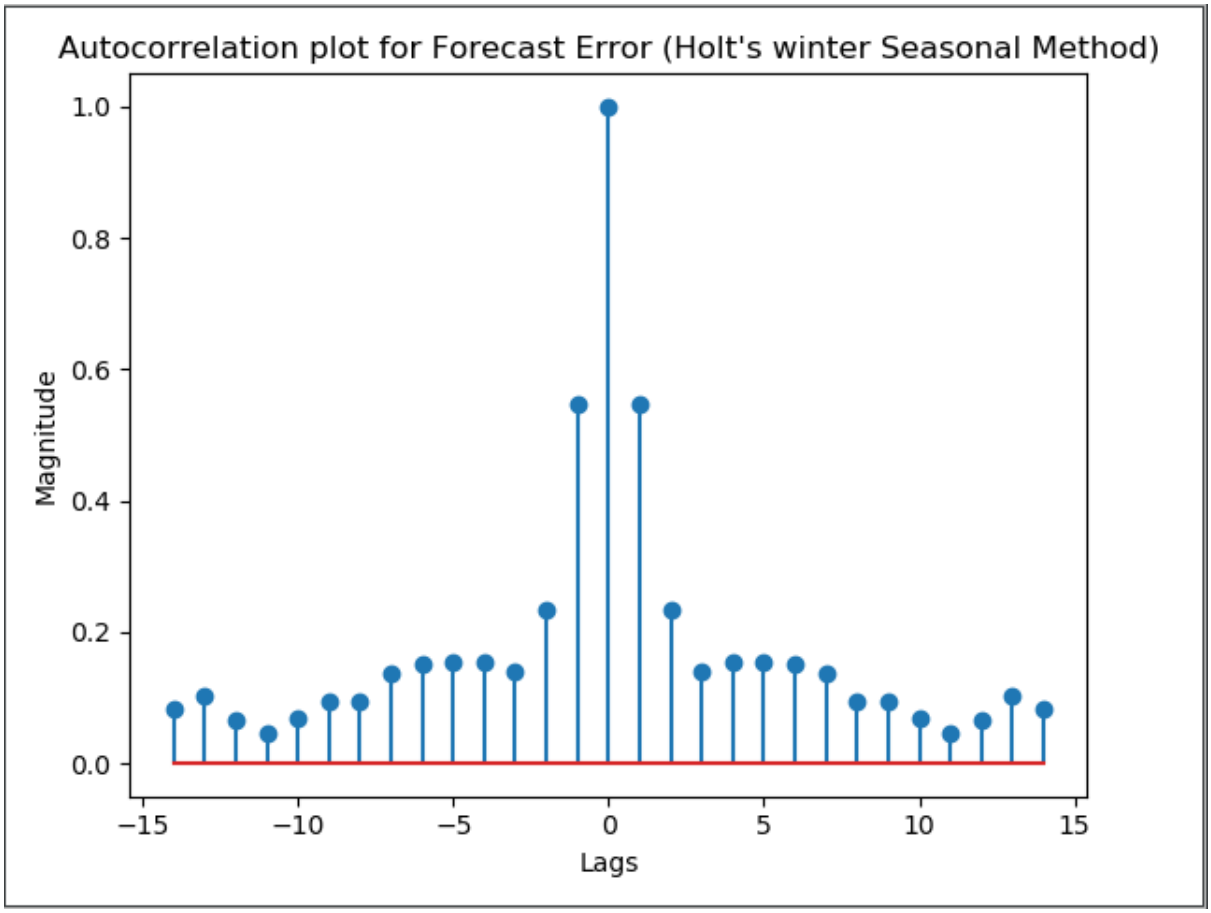

Autocorrelation plot for Forecast Error (Average Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−15
−10
−5
0
5
10
15


Autocorrelation plot for Forecast Error (Naive Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−15
−10
−5
0
5
10
15


Autocorrelation plot for Forecast Error (Drift Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−15
−10
−5
0
5
10
15


Autocorrelation plot for Forecast Error (SES Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−15
−10
−5
0
5
10
15


Autocorrelation plot for Forecast Error (Holt's Linear Method)
Magnitude
Lags
1.0
0.8
0.6
0.4
0.2
0.0
−15
−10
−5
0
5
10
15


Autocorrelation plot for Forecast Error (Holt's winter Seasonal Method)
Magnitude
1.0
0.8
0.6
0.4
0.2
0.0
−15
−10
−5
0
5
10
15
Lags

JUSTIFICATIONS

AIR PASSENGERS

Below are the AutoCorrelation (ACF) plots for the forecast errors. From the plots, we can observe that Holt's winter seasonal plot has most of the coefficients close to zero compared to the other plots. But still no model has adequately captured the information in the data.

From the below results, we can observe that Holt's winter seasonal method has low Q-value and Mean square error value for predicted and forecast errors. Also, there is a very less difference between variance of predicted and forecast compared to other methods.

SHAMPOO

Below are the autocorrelation plots for forecast errors. From the plots, we can observe that none of the methods have coefficients close to zero. That means no model has adequately captured the information in the data.

From the below results, we can observe that Q-value is less for Average, Naïve and SES methods and MSE for forecast error is less for Drift method. Also, there is a less difference between variance of predicted error and forecast error for Drift method.

Daily total female births dataset:

Below are the autocorrelation plots for forecast errors. From the plots, we can observe that none of the methods have coefficients close to zero. That means no model has adequately captured the information in the data.

From the below results, we can observe that Q-value is less for Holt's winter seasonal method and MSE Predicted error value is less for Holt's winter. Also, the difference between Variance of forecast and predicted error is less for Holt's winter method.

Tute1 Dataset:

Below are the autocorrelation plots for forecast errors. From the plots, we can observe that most of the coefficients of Holt's winter seasonal method are close to zero compared to other methods.

From the below results, we can observe that that Q-value and MSE of predicted and forecast error is less for Holt's winter seasonal. Also, the difference between the variance of forecast and predicted error is less for Holt's winter seasonal method.

Daily-min-temperatures Dataset:

Below are the autocorrelation plots for forecast errors. From the plots, we can observe that most of the coefficients of Holt's winter seasonal method are close to zero compared to other methods

From the below results, we can observe that that Q-value and MSE of predicted and forecast error is less for Holt's winter seasonal. Also, the difference between the variance of forecast and predicted error is less for Holt's winter seasonal method.

```
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.tsa.holtwinters as ets
from statsmodels.tsa.api import SimpleExpSmoothing
import numpy as np
from pandas.plotting import register_matplotlib_converters
from sklearn.model_selection import train_test_split
from Autocorrelation import cal_auto_corr
from Pearson_Correlation_Coefficient import correlation_coefficent_cal
import warnings

warnings.filterwarnings("ignore")
register_matplotlib_converters()

'-------------------------Air Passengers Dataset----------------------'
df = pd.read_csv('AirPassengers.csv', index_col='Month', parse_dates=True)
df.index.freq = 'MS'
y = df['#Passengers']
train, test = train_test_split(y, shuffle=False, test_size=0.2)
train.index.freq = 'MS'
test.index.freq = 'MS'
h = len(test)

print('************************* Air Passengers Dataset results
**********************')
#Average Method
def avg_method(train):
    y_hat_avg = np.mean(train)
    return y_hat_avg

train_pred_avg = []
for i in range(1,len(train)):
    res = avg_method(train.iloc[0:i])
    train_pred_avg.append(res)

test_forecast_avg1 = np.ones(len(test)) * avg_method(train)
test_forecast_avg = pd.DataFrame(test_forecast_avg1).set_index(test.index)
residual_error_avg = np.array(train[1:]) - np.array(train_pred_avg)
forecast_error_avg = test - test_forecast_avg1
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)

# Naive Method
def naive_method(t):
    return t

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

res = np.ones(len(test)) * train[-1]
naive_test_forecast1 = np.ones(len(test)) * res
naive_test_forecast = pd.DataFrame(naive_test_forecast1).set_index(test.index)
residual_error_naive = np.array(train[1:]) - np.array(naive_train_pred)
forecast_error_naive = test - naive_test_forecast1
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)

# Drift method
def drift_method(t, h):
    y_hat_drift = t[len(t)-1] + h*((t[len(t)-1]-t[0])/(len(t) - 1))
    return y_hat_drift

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

drift_test_forecast1=[]
for h in range(1, len(test)+1):
    res = drift_method(train, h)
    drift_test_forecast1.append(res)

drift_test_forecast = pd.DataFrame(drift_test_forecast1).set_index(test.index)
residual_error_drift = np.array(train[1:]) - np.array(drift_train_forecast)
forecast_error_drift = np.array(test) - np.array(drift_test_forecast1)
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)

#SES Method
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 = train[0]
ses_train_pred = ses(train, 0.50, l0)
ses_test_forecast1 = np.ones(len(test)) * (0.5*(train[-1]) + (1-
0.5)*(ses_train_pred[-1]))
ses_test_forecast = pd.DataFrame(ses_test_forecast1).set_index(test.index)
residual_error_ses = np.array(train[1:]) - np.array(ses_train_pred)
forecast_error_ses = np.array(test) - np.array(ses_test_forecast1)
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)

# SES Method using statsmodels for alpha=0.5
# ses_train = train.ewm(alpha=0.5, adjust=False).mean()  # Another way of doing it
ses_model1 = SimpleExpSmoothing(train)
ses_fitted_model1 = ses_model1.fit(smoothing_level=0.5, optimized=False)
ses_train_pred1 = ses_fitted_model1.fittedvalues.shift(-1)
ses_test_forecast1 = ses_fitted_model1.forecast(steps=len(test))
ses_test_forecast1 = pd.DataFrame(ses_test_forecast1).set_index(test.index)
MSE_test_SES1 = np.square(np.subtract(test.values,
np.ndarray.flatten(ses_test_forecast1.values))).mean()

# Holt's Linear Trend
holtl_fitted_model = ets.ExponentialSmoothing(train, trend='multiplicative',
damped=True, seasonal=None).fit()
holtl_train_pred = holtl_fitted_model.fittedvalues
holtl_test_forecast = holtl_fitted_model.forecast(steps=len(test))
holtl_test_forecast = pd.DataFrame(holtl_test_forecast).set_index(test.index)
residual_error_holtl = np.subtract(train.values,
np.ndarray.flatten(holtl_train_pred.values))
forecast_error_holtl = np.subtract(test.values,
np.ndarray.flatten(holtl_test_forecast.values))
MSE_train_holtl = np.mean((residual_error_holtl)**2)
MSE_test_holtl = np.mean((forecast_error_holtl)**2)
print("Mean Square Error of prediction errors for Holt's Linear method: ",
MSE_train_holtl)
print("Mean Square Error of forecast errors for Holt's Linear method: ",
MSE_test_holtl)
mean_pred_holtl = np.mean(residual_error_holtl)
var_pred_holtl = np.var(residual_error_holtl)
var_forecast_holtl = np.var(forecast_error_holtl)
```

```
print("Mean of prediction errors for Holt's Linear method: ", mean_pred_holtl)
print("Variance of prediction errors for Holt's Linear method: ", var_pred_holtl)
print("Variance of forecast errors for Holt's Linear method: ",
var_forecast_holtl)

# Holt's Winter Seasonal Trend
holtw_fitted_model = ets.ExponentialSmoothing(train, trend='mul', damped=True,
seasonal='mul', seasonal_periods=12).fit()
holtw_train_pred = holtw_fitted_model.fittedvalues
holtw_test_forecast = holtw_fitted_model.forecast(steps=len(test))
holtw_test_forecast = pd.DataFrame(holtw_test_forecast).set_index(test.index)

residual_error_holtw = np.subtract(train.values,
np.ndarray.flatten(holtw_train_pred.values))
forecast_error_holtw = np.subtract(test.values,
np.ndarray.flatten(holtw_test_forecast.values))
MSE_train_holtw = np.mean((residual_error_holtw)**2)
MSE_test_holtw = np.mean((forecast_error_holtw)**2)
print("Mean Square Error of prediction errors for Holt's Winter Seasonal method:
", MSE_train_holtw)
print("Mean Square Error of forecast errors for Holt's Winter Seasonal method: ",
MSE_test_holtw)
mean_pred_holtw = np.mean(residual_error_holtw)
var_pred_holtw = np.var(residual_error_holtw)
var_forecast_holtw = np.var(forecast_error_holtw)
print("Mean of prediction errors for Holt's Winter Seasonal method: ",
mean_pred_holtw)
print("Variance of prediction errors for Holt's Winter Seasonal method: ",
var_pred_holtw)
print("Variance of forecast errors for Holt's Winter Seasonal method: ",
var_forecast_holtw)

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(test_forecast_avg, label='Average h-step prediction')
plt.xlabel('Time (Monthly)')
plt.ylabel('Number of Passengers')
plt.title('Average Method - Air Passengers')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(naive_test_forecast, label='Naive h-step prediction')
plt.xlabel('Time (Monthly)')
plt.ylabel('Number of Passengers')
plt.title('Naive Method - Air Passengers')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(drift_test_forecast, label='Drift h-step prediction')
plt.xlabel('Time (Monthly)')
plt.ylabel('Number of Passengers')
plt.title('Drift Method - Air Passengers')
```

```
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(ses_test_forecast, label='Simple Exponential Smoothing h-step prediction')
plt.xlabel('Time (Monthly)')
plt.ylabel('Number of Passengers')
plt.title('SES Method - Air Passengers')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(holtl_test_forecast, label="Holt's Linear h-step prediction")
plt.xlabel('Time (Monthly)')
plt.ylabel('Number of Passengers')
plt.title("Holt's Linear Method - Air Passengers")
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(holtw_test_forecast, label="Holt's Winter Seasonal h-step prediction")
plt.xlabel('Time (Monthly)')
plt.ylabel('Number of Passengers')
plt.title("Holt's Winter Seasonal Method - Air Passengers")
plt.legend(loc='upper left')
plt.show()

# Auto_correlation for forecast errors and Q value for prediction errors and
forecast errors
#Average Method
k = len(test)
lags = 15
avg_forecast_acf = cal_auto_corr(forecast_error_avg, lags)
Q_forecast_avg = k * np.sum(np.array(avg_forecast_acf[lags:])**2)
print('Q value of forecast errors for Average method: ', Q_forecast_avg)
plt.figure()
plt.stem(range(-(lags-1),lags), avg_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (Average Method)')
plt.show()

# Naive method
k = len(test)
lags = 15
naive_forecast_acf = cal_auto_corr(forecast_error_naive, lags)
Q_forecast_naive = k * np.sum(np.array(naive_forecast_acf[lags:])**2)
print('Q value of forecast errors for Naive method: ', Q_forecast_naive)
plt.figure()
plt.stem(range(-(lags-1),lags), naive_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (Naive Method)')
```

```
plt.show()

# Drift Method
k = len(test)
lags = 15
drift_forecast_acf = cal_auto_corr(forecast_error_drift, lags)
Q_forecast_drift = k * np.sum(np.array(drift_forecast_acf[lags:])**2)
print('Q value of forecast errors for Drift method: ', Q_forecast_drift)
plt.figure()
plt.stem(range(-(lags-1),lags), drift_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (Drift Method)')
plt.show()

# SES method
k = len(test)
lags = 15
ses_forecast_acf = cal_auto_corr(forecast_error_ses, lags)
Q_forecast_SES = k * np.sum(np.array(ses_forecast_acf[lags:])**2)
print('Q value of forecast errors for SES method: ', Q_forecast_SES)
plt.figure()
plt.stem(range(-(lags-1), lags), ses_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (SES Method)')
plt.show()

# holt's linear method
k = len(test)
lags = 15
holtl_forecast_acf = cal_auto_corr(forecast_error_holtl, lags)
Q_forecast_holtl = k * np.sum(np.array(holtl_forecast_acf[lags:])**2)
print("Q value of forecast errors for Holt's Linear method: ", Q_forecast_holtl)
plt.figure()
plt.stem(range(-(lags-1), lags), holtl_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title("Autocorrelation plot for Forecast Error (Holt's Linear Method)")
plt.show()

# holt's Winter Seasonal method
k = len(test)
lags = 15
holtw_forecast_acf = cal_auto_corr(forecast_error_holtw, lags)
Q_forecast_holtw = k * np.sum(np.array(holtw_forecast_acf[lags:])**2)
print("Q value of forecast errors for Holt's Winter Seasonal method: ",
Q_forecast_holtw)
plt.figure()
plt.stem(range(-(lags-1), lags), holtw_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title("Autocorrelation plot for Forecast Error (Holt's winter Seasonal
Method)")
plt.show()

corr_avg = correlation_coefficent_cal(forecast_error_avg, test)
corr_naive = correlation_coefficent_cal(forecast_error_naive, test)
corr_drift = correlation_coefficent_cal(forecast_error_drift, test)
```

```
corr_ses = correlation_coefficent_cal(forecast_error_ses, test)
corr_holtl = correlation_coefficent_cal(forecast_error_holtl, test)
corr_holtw = correlation_coefficent_cal(forecast_error_holtw, test)
print("Correlation Coefficient between Forecast Error and Test set for Average
Method: {}".format(corr_avg))
print("Correlation Coefficient between Forecast Error and Test set for Naive
Method: {}".format(corr_naive))
print("Correlation Coefficient between Forecast Error and Test set for Drift
Method: {}".format(corr_drift))
print("Correlation Coefficient between Forecast Error and Test set for SES Method:
{}".format(corr_ses))
print("Correlation Coefficient between Forecast Error and Test set for Holt's
Linear Method: {}".format(corr_holtl))
print("Correlation Coefficient between Forecast Error and Test set for Holt's
winter seasonal Method: {}".format(corr_holtw))
d = {'Methods':['Average', 'Naive', 'Drift', 'SES', "HoltL", "HoltW"],
     'Q_val': [round(Q_forecast_avg, 2), round(Q_forecast_naive, 2),
round(Q_forecast_drift,2), round(Q_forecast_SES,2), round(Q_forecast_holtl,2),
round(Q_forecast_holtw,2)],
     'MSE(P)': [round(MSE_train_avg,2), round(MSE_train_naive,2),
round(MSE_train_drift,2), round(MSE_train_SES,2), round(MSE_train_holtl,2),
round(MSE_train_holtw,2)],
     'MSE(F)': [round(MSE_test_avg,2), round(MSE_test_naive,2),
round(MSE_test_drift,2), round(MSE_test_SES,2), round(MSE_test_holtl,2),
round(MSE_test_holtw,2)],
     'var(P)': [round(var_pred_avg,2), round(var_pred_naive,2),
round(var_pred_drift,2), round(var_pred_SES,2), round(var_pred_holtl,2),
round(var_pred_holtw,2)],
     'var(F)':[round(var_forecast_avg,2), round(var_forecast_naive,2),
round(var_forecast_drift,2), round(var_forecast_SES,2),
round(var_forecast_holtl,2), round(var_forecast_holtw,2)],
     'corrcoeff':[round(corr_avg,2), round(corr_naive,2), round(corr_drift,2),
round(corr_ses,2), round(corr_holtl,2), round(corr_holtw,2)]}
df = pd.DataFrame(data=d)
df = df.set_index('Methods')
pd.set_option('display.max_columns', None)
print(df)


'----------------------------------------------------shampoo Dataset------------
-----------------------------------'
df = pd.read_csv('shampoo.csv', index_col='Month', parse_dates=True)
df.index.freq = 'MS'
y = df['Sales']
y.index = pd.date_range(start='2001-01-01', end='2003-12-01', freq='MS')
train, test = train_test_split(y, shuffle=False, test_size=0.2)
train.index.freq = 'MS'
test.index.freq = 'MS'
h = len(test)
print('************************ Shampoo Dataset results ************************')
#Average Method
def avg_method(train):
    y_hat_avg = np.mean(train)
    return y_hat_avg

train_pred_avg = []
for i in range(1,len(train)):
    res = avg_method(train.iloc[0:i])
    train_pred_avg.append(res)
```

```
test_forecast_avg1 = np.ones(len(test)) * avg_method(train)
test_forecast_avg = pd.DataFrame(test_forecast_avg1).set_index(test.index)
residual_error_avg = np.array(train[1:]) - np.array(train_pred_avg)
forecast_error_avg = test - test_forecast_avg1
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)

# Naive Method
def naive_method(t):
    return t

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

res = np.ones(len(test)) * train[-1]
naive_test_forecast1 = np.ones(len(test)) * res
naive_test_forecast = pd.DataFrame(naive_test_forecast1).set_index(test.index)
residual_error_naive = np.array(train[1:]) - np.array(naive_train_pred)
forecast_error_naive = test - naive_test_forecast1
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)

# Drift method
def drift_method(t, h):
    y_hat_drift = t[len(t)-1] + h*((t[len(t)-1]-t[0])/(len(t) - 1))
    return y_hat_drift

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

drift_test_forecast1=[]
for h in range(1, len(test)+1):
```

```
    res = drift_method(train, h)
    drift_test_forecast1.append(res)

drift_test_forecast = pd.DataFrame(drift_test_forecast1).set_index(test.index)
residual_error_drift = np.array(train[1:]) - np.array(drift_train_forecast)
forecast_error_drift = np.array(test) - np.array(drift_test_forecast1)
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)

#SES Method
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 = train[0]
ses_train_pred = ses(train, 0.50, l0)
ses_test_forecast1 = np.ones(len(test)) * (0.5*(train[-1]) + (1-
0.5)*(ses_train_pred[-1]))
ses_test_forecast = pd.DataFrame(ses_test_forecast1).set_index(test.index)
residual_error_ses = np.array(train[1:]) - np.array(ses_train_pred)
forecast_error_ses = np.array(test) - np.array(ses_test_forecast1)
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)

# SES Method using statsmodels for alpha=0.5
# ses_train = train.ewm(alpha=0.5, adjust=False).mean()  # Another way of doing it
ses_model1 = SimpleExpSmoothing(train)
ses_fitted_model1 = ses_model1.fit(smoothing_level=0.5, optimized=False)
ses_train_pred1 = ses_fitted_model1.fittedvalues.shift(-1)
ses_test_forecast1 = ses_fitted_model1.forecast(steps=len(test))
ses_test_forecast1 = pd.DataFrame(ses_test_forecast1).set_index(test.index)
MSE_test_SES1 = np.square(np.subtract(test.values,
np.ndarray.flatten(ses_test_forecast1.values))).mean()

# Holt's Linear Trend
holtl_fitted_model = ets.ExponentialSmoothing(train, trend='multiplicative',
damped=True, seasonal=None).fit()
holtl_train_pred = holtl_fitted_model.fittedvalues
```

```
holtl_test_forecast = holtl_fitted_model.forecast(steps=len(test))
holtl_test_forecast = pd.DataFrame(holtl_test_forecast).set_index(test.index)
residual_error_holtl = np.subtract(train.values,
np.ndarray.flatten(holtl_train_pred.values))
forecast_error_holtl = np.subtract(test.values,
np.ndarray.flatten(holtl_test_forecast.values))
MSE_train_holtl = np.mean((residual_error_holtl)**2)
MSE_test_holtl = np.mean((forecast_error_holtl)**2)
print("Mean Square Error of prediction errors for Holt's Linear method: ",
MSE_train_holtl)
print("Mean Square Error of forecast errors for Holt's Linear method: ",
MSE_test_holtl)
mean_pred_holtl = np.mean(residual_error_holtl)
var_pred_holtl = np.var(residual_error_holtl)
var_forecast_holtl = np.var(forecast_error_holtl)
print("Mean of prediction errors for Holt's Linear method: ", mean_pred_holtl)
print("Variance of prediction errors for Holt's Linear method: ", var_pred_holtl)
print("Variance of forecast errors for Holt's Linear method: ",
var_forecast_holtl)

# Holt's Winter Seasonal Trend
holtw_fitted_model = ets.ExponentialSmoothing(train, trend='mul', damped=True,
seasonal='mul', seasonal_periods=12).fit()
holtw_train_pred = holtw_fitted_model.fittedvalues
holtw_test_forecast = holtw_fitted_model.forecast(steps=len(test))
holtw_test_forecast = pd.DataFrame(holtw_test_forecast).set_index(test.index)
residual_error_holtw = np.subtract(train.values,
np.ndarray.flatten(holtw_train_pred.values))
forecast_error_holtw = np.subtract(test.values,
np.ndarray.flatten(holtw_test_forecast.values))
MSE_train_holtw = np.mean((residual_error_holtw)**2)
MSE_test_holtw = np.mean((forecast_error_holtw)**2)
print("Mean Square Error of prediction errors for Holt's Winter Seasonal method:
", MSE_train_holtw)
print("Mean Square Error of forecast errors for Holt's Winter Seasonal method: ",
MSE_test_holtw)
mean_pred_holtw = np.mean(residual_error_holtw)
var_pred_holtw = np.var(residual_error_holtw)
var_forecast_holtw = np.var(forecast_error_holtw)
print("Mean of prediction errors for Holt's Winter Seasonal method: ",
mean_pred_holtw)
print("Variance of prediction errors for Holt's Winter Seasonal method: ",
var_pred_holtw)
print("Variance of forecast errors for Holt's Winter Seasonal method: ",
var_forecast_holtw)

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(test_forecast_avg, label='Average h-step prediction')
plt.xlabel('Time (Monthly)')
plt.ylabel('Number of Sales')
plt.title('Average Method - Shampoo')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
```

```
ax.plot(naive_test_forecast, label='Naive h-step prediction')
plt.xlabel('Time (Monthly)')
plt.ylabel('Number of Sales')
plt.title('Naive Method - Shampoo')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(drift_test_forecast, label='Drift h-step prediction')
plt.xlabel('Time (Monthly)')
plt.ylabel('Number of Sales')
plt.title('Drift Method - Shampoo')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(ses_test_forecast, label='Simple Exponential Smoothing h-step prediction')
plt.xlabel('Time (Monthly)')
plt.ylabel('Number of Sales')
plt.title('SES Method - Shampoo')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(holtl_test_forecast, label="Holt's Linear h-step prediction")
plt.xlabel('Time (Monthly)')
plt.ylabel('Number of Sales')
plt.title("Holt's Linear Method - Shampoo")
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(holtw_test_forecast, label="Holt's Winter Seasonal h-step prediction")
plt.xlabel('Time (Monthly)')
plt.ylabel('Number of Sales')
plt.title("Holt's Winter Seasonal Method - Shampoo")
plt.legend(loc='upper left')
plt.show()

# Auto_correlation for forecast errors and Q value for prediction errors and
forecast errors
#Average Method
k = len(test)
lags = 15
avg_forecast_acf = cal_auto_corr(forecast_error_avg, lags)
Q_forecast_avg = k * np.sum(np.array(avg_forecast_acf[lags:])**2)
print('Q value of forecast errors for Average method: ', Q_forecast_avg)
plt.figure()
plt.stem(range(-(lags-1),lags), avg_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
```

```
plt.title('Autocorrelation plot for Forecast Error (Average Method)')
plt.show()

# Naive method
k = len(test)
lags = 15
naive_forecast_acf = cal_auto_corr(forecast_error_naive, lags)
Q_forecast_naive = k * np.sum(np.array(naive_forecast_acf[lags:])**2)
print('Q value of forecast errors for Naive method: ', Q_forecast_naive)
plt.figure()
plt.stem(range(-(lags-1),lags), naive_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (Naive Method)')
plt.show()

# Drift Method
k = len(test)
lags = 15
drift_forecast_acf = cal_auto_corr(forecast_error_drift, lags)
Q_forecast_drift = k * np.sum(np.array(drift_forecast_acf[lags:])**2)
print('Q value of forecast errors for Drift method: ', Q_forecast_drift)
plt.figure()
plt.stem(range(-(lags-1),lags), drift_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (Drift Method)')
plt.show()

# SES method
k = len(test)
lags = 15
ses_forecast_acf = cal_auto_corr(forecast_error_ses, lags)
Q_forecast_SES = k * np.sum(np.array(ses_forecast_acf[lags:])**2)
print('Q value of forecast errors for SES method: ', Q_forecast_SES)
plt.figure()
plt.stem(range(-(lags-1), lags), ses_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (SES Method)')
plt.show()

# holt's linear method
k = len(test)
lags = 15
holtl_forecast_acf = cal_auto_corr(forecast_error_holtl, lags)
Q_forecast_holtl = k * np.sum(np.array(holtl_forecast_acf[lags:])**2)
print("Q value of forecast errors for Holt's Linear method: ", Q_forecast_holtl)
plt.figure()
plt.stem(range(-(lags-1), lags), holtl_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title("Autocorrelation plot for Forecast Error (Holt's Linear Method)")
plt.show()

# holt's Winter Seasonal method
k = len(test)
lags = 15
holtw_forecast_acf = cal_auto_corr(forecast_error_holtw, lags)
```

```
Q_forecast_holtw = k * np.sum(np.array(holtw_forecast_acf[lags:])**2)
print("Q value of forecast errors for Holt's Winter Seasonal method: ",
Q_forecast_holtw)
plt.figure()
plt.stem(range(-(lags-1), lags), holtw_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title("Autocorrelation plot for Forecast Error (Holt's winter Seasonal
Method)")
plt.show()

corr_avg = correlation_coefficent_cal(forecast_error_avg, test)
corr_naive = correlation_coefficent_cal(forecast_error_naive, test)
corr_drift = correlation_coefficent_cal(forecast_error_drift, test)
corr_ses = correlation_coefficent_cal(forecast_error_ses, test)
corr_holtl = correlation_coefficent_cal(forecast_error_holtl, test)
corr_holtw = correlation_coefficent_cal(forecast_error_holtw, test)
print("Correlation Coefficient between Forecast Error and Test set for Average
Method: {}".format(corr_avg))
print("Correlation Coefficient between Forecast Error and Test set for Naive
Method: {}".format(corr_naive))
print("Correlation Coefficient between Forecast Error and Test set for Drift
Method: {}".format(corr_drift))
print("Correlation Coefficient between Forecast Error and Test set for SES Method:
{}".format(corr_ses))
print("Correlation Coefficient between Forecast Error and Test set for Holt's
Linear Method: {}".format(corr_holtl))
print("Correlation Coefficient between Forecast Error and Test set for Holt's
winter seasonal Method: {}".format(corr_holtw))
d = {'Methods':['Average', 'Naive', 'Drift', 'SES', "HoltL", "HoltW"],
     'Q_val': [round(Q_forecast_avg, 2), round(Q_forecast_naive, 2),
round(Q_forecast_drift,2), round(Q_forecast_SES,2), round(Q_forecast_holtl,2),
round(Q_forecast_holtw,2)],
     'MSE(P)': [round(MSE_train_avg,2), round(MSE_train_naive,2),
round(MSE_train_drift,2), round(MSE_train_SES,2), round(MSE_train_holtl,2),
round(MSE_train_holtw,2)],
     'MSE(F)': [round(MSE_test_avg,2), round(MSE_test_naive,2),
round(MSE_test_drift,2), round(MSE_test_SES,2), round(MSE_test_holtl,2),
round(MSE_test_holtw,2)],
     'var(P)': [round(var_pred_avg,2), round(var_pred_naive,2),
round(var_pred_drift,2), round(var_pred_SES,2), round(var_pred_holtl,2),
round(var_pred_holtw,2)],
     'var(F)':[round(var_forecast_avg,2), round(var_forecast_naive,2),
round(var_forecast_drift,2), round(var_forecast_SES,2),
round(var_forecast_holtl,2), round(var_forecast_holtw,2)],
     'corrcoeff':[round(corr_avg,2), round(corr_naive,2), round(corr_drift,2),
round(corr_ses,2), round(corr_holtl,2), round(corr_holtw,2)]}
df = pd.DataFrame(data=d)
df = df.set_index('Methods')
pd.set_option('display.max_columns', None)
print(df)


'---------------------------------------- daily-total-female-births Dataset
-----------------------------------'
df = pd.read_csv('daily-total-female-births.csv', index_col='Date',
parse_dates=True)
df.index.freq = 'D'
y = df['Births']
```

```
y.index = pd.date_range(start='1959-01-01', end='1959-12-31', freq='D')
train, test = train_test_split(y, shuffle=False, test_size=0.2)
train.index.freq = 'D'
test.index.freq = 'D'
h = len(test)
print('************************ daily-total-female-births Dataset results
**********************')
#Average Method
def avg_method(train):
    y_hat_avg = np.mean(train)
    return y_hat_avg

train_pred_avg = []
for i in range(1,len(train)):
    res = avg_method(train.iloc[0:i])
    train_pred_avg.append(res)

test_forecast_avg1 = np.ones(len(test)) * avg_method(train)
test_forecast_avg = pd.DataFrame(test_forecast_avg1).set_index(test.index)
residual_error_avg = np.array(train[1:]) - np.array(train_pred_avg)
forecast_error_avg = test - test_forecast_avg1
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)

# Naive Method
def naive_method(t):
    return t

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

res = np.ones(len(test)) * train[-1]
naive_test_forecast1 = np.ones(len(test)) * res
naive_test_forecast = pd.DataFrame(naive_test_forecast1).set_index(test.index)
residual_error_naive = np.array(train[1:]) - np.array(naive_train_pred)
forecast_error_naive = test - naive_test_forecast1
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)
```

```
# Drift method
def drift_method(t, h):
    y_hat_drift = t[len(t)-1] + h*((t[len(t)-1]-t[0])/(len(t) - 1))
    return y_hat_drift

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

drift_test_forecast1=[]
for h in range(1, len(test)+1):
    res = drift_method(train, h)
    drift_test_forecast1.append(res)

drift_test_forecast = pd.DataFrame(drift_test_forecast1).set_index(test.index)
residual_error_drift = np.array(train[1:]) - np.array(drift_train_forecast)
forecast_error_drift = np.array(test) - np.array(drift_test_forecast1)
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)

#SES Method
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 = train[0]
ses_train_pred = ses(train, 0.50, l0)
ses_test_forecast1 = np.ones(len(test)) * (0.5*(train[-1]) + (1-
0.5)*(ses_train_pred[-1]))
ses_test_forecast = pd.DataFrame(ses_test_forecast1).set_index(test.index)
residual_error_ses = np.array(train[1:]) - np.array(ses_train_pred)
forecast_error_ses = np.array(test) - np.array(ses_test_forecast1)
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)

# SES Method using statsmodels for alpha=0.5
# ses_train = train.ewm(alpha=0.5, adjust=False).mean()  # Another way of doing it
ses_model1 = SimpleExpSmoothing(train)
ses_fitted_model1 = ses_model1.fit(smoothing_level=0.5, optimized=False)
ses_train_pred1 = ses_fitted_model1.fittedvalues.shift(-1)
ses_test_forecast1 = ses_fitted_model1.forecast(steps=len(test))
ses_test_forecast1 = pd.DataFrame(ses_test_forecast1).set_index(test.index)
MSE_test_SES1 = np.square(np.subtract(test.values,
np.ndarray.flatten(ses_test_forecast1.values))).mean()

# Holt's Linear Trend
holtl_fitted_model = ets.ExponentialSmoothing(train, trend='add', damped=True,
seasonal=None).fit()
holtl_train_pred = holtl_fitted_model.fittedvalues
holtl_test_forecast = holtl_fitted_model.forecast(steps=len(test))
holtl_test_forecast = pd.DataFrame(holtl_test_forecast).set_index(test.index)
residual_error_holtl = np.subtract(train.values,
np.ndarray.flatten(holtl_train_pred.values))
forecast_error_holtl = np.subtract(test.values,
np.ndarray.flatten(holtl_test_forecast.values))
MSE_train_holtl = np.mean((residual_error_holtl)**2)
MSE_test_holtl = np.mean((forecast_error_holtl)**2)
print("Mean Square Error of prediction errors for Holt's Linear method: ",
MSE_train_holtl)
print("Mean Square Error of forecast errors for Holt's Linear method: ",
MSE_test_holtl)
mean_pred_holtl = np.mean(residual_error_holtl)
var_pred_holtl = np.var(residual_error_holtl)
var_forecast_holtl = np.var(forecast_error_holtl)
print("Mean of prediction errors for Holt's Linear method: ", mean_pred_holtl)
print("Variance of prediction errors for Holt's Linear method: ", var_pred_holtl)
print("Variance of forecast errors for Holt's Linear method: ",
var_forecast_holtl)

# Holt's Winter Seasonal Trend
holtw_fitted_model = ets.ExponentialSmoothing(train, trend='add', damped=True,
seasonal='add').fit()
holtw_train_pred = holtw_fitted_model.fittedvalues
holtw_test_forecast = holtw_fitted_model.forecast(steps=len(test))
holtw_test_forecast = pd.DataFrame(holtw_test_forecast).set_index(test.index)
residual_error_holtw = np.subtract(train.values,
np.ndarray.flatten(holtw_train_pred.values))
forecast_error_holtw = np.subtract(test.values,
np.ndarray.flatten(holtw_test_forecast.values))
MSE_train_holtw = np.mean((residual_error_holtw)**2)
MSE_test_holtw = np.mean((forecast_error_holtw)**2)
print("Mean Square Error of prediction errors for Holt's Winter Seasonal method:
", MSE_train_holtw)
print("Mean Square Error of forecast errors for Holt's Winter Seasonal method: ",
MSE_test_holtw)
mean_pred_holtw = np.mean(residual_error_holtw)
var_pred_holtw = np.var(residual_error_holtw)
var_forecast_holtw = np.var(forecast_error_holtw)
print("Mean of prediction errors for Holt's Winter Seasonal method: ",
mean_pred_holtw)
print("Variance of prediction errors for Holt's Winter Seasonal method: ",
var_pred_holtw)
```

```
print("Variance of forecast errors for Holt's Winter Seasonal method: ",
var_forecast_holtw)

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(test_forecast_avg, label='Average h-step prediction')
plt.xlabel('Time (Daily)')
plt.ylabel('Number of Births')
plt.title('Average Method - daily-total-female-births')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(naive_test_forecast, label='Naive h-step prediction')
plt.xlabel('Time (Daily)')
plt.ylabel('Number of Births')
plt.title('Naive Method - daily-total-female-births')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(drift_test_forecast, label='Drift h-step prediction')
plt.xlabel('Time (Daily)')
plt.ylabel('Number of Births')
plt.title('Drift Method - daily-total-female-births')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(ses_test_forecast, label='Simple Exponential Smoothing h-step prediction')
plt.xlabel('Time (Daily)')
plt.ylabel('Number of Births')
plt.title('SES Method - daily-total-female-births')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(holtl_test_forecast, label="Holt's Linear h-step prediction")
plt.xlabel('Time (Daily)')
plt.ylabel('Number of Births')
plt.title("Holt's Linear Method - daily-total-female-births")
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(holtw_test_forecast, label="Holt's Winter Seasonal h-step prediction")
plt.xlabel('Time (Daily)')
plt.ylabel('Number of Births')
```

```
plt.title("Holt's Winter Seasonal Method - daily-total-female-births")
plt.legend(loc='upper left')
plt.show()

# Auto_correlation for forecast errors and Q value for prediction errors and
forecast errors
#Average Method
k = len(test)
lags = 15
avg_forecast_acf = cal_auto_corr(forecast_error_avg, lags)
Q_forecast_avg = k * np.sum(np.array(avg_forecast_acf[lags:])**2)
print('Q value of forecast errors for Average method: ', Q_forecast_avg)
plt.figure()
plt.stem(range(-(lags-1),lags), avg_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (Average Method)')
plt.show()

# Naive method
k = len(test)
lags = 15
naive_forecast_acf = cal_auto_corr(forecast_error_naive, lags)
Q_forecast_naive = k * np.sum(np.array(naive_forecast_acf[lags:])**2)
print('Q value of forecast errors for Naive method: ', Q_forecast_naive)
plt.figure()
plt.stem(range(-(lags-1),lags), naive_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (Naive Method)')
plt.show()

# Drift Method
k = len(test)
lags = 15
drift_forecast_acf = cal_auto_corr(forecast_error_drift, lags)
Q_forecast_drift = k * np.sum(np.array(drift_forecast_acf[lags:])**2)
print('Q value of forecast errors for Drift method: ', Q_forecast_drift)
plt.figure()
plt.stem(range(-(lags-1),lags), drift_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (Drift Method)')
plt.show()

# SES method
k = len(test)
lags = 15
ses_forecast_acf = cal_auto_corr(forecast_error_ses, lags)
Q_forecast_SES = k * np.sum(np.array(ses_forecast_acf[lags:])**2)
print('Q value of forecast errors for SES method: ', Q_forecast_SES)
plt.figure()
plt.stem(range(-(lags-1), lags), ses_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (SES Method)')
plt.show()

# holt's linear method
```

```
k = len(test)
lags = 15
holtl_forecast_acf = cal_auto_corr(forecast_error_holtl, lags)
Q_forecast_holtl = k * np.sum(np.array(holtl_forecast_acf[lags:])**2)
print("Q value of forecast errors for Holt's Linear method: ", Q_forecast_holtl)
plt.figure()
plt.stem(range(-(lags-1), lags), holtl_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title("Autocorrelation plot for Forecast Error (Holt's Linear Method)")
plt.show()

# holt's Winter Seasonal method
k = len(test)
lags = 15
holtw_forecast_acf = cal_auto_corr(forecast_error_holtw, lags)
Q_forecast_holtw = k * np.sum(np.array(holtw_forecast_acf[lags:])**2)
print("Q value of forecast errors for Holt's Winter Seasonal method: ",
Q_forecast_holtw)
plt.figure()
plt.stem(range(-(lags-1), lags), holtw_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title("Autocorrelation plot for Forecast Error (Holt's winter Seasonal
Method)")
plt.show()

corr_avg = correlation_coefficent_cal(forecast_error_avg, test)
corr_naive = correlation_coefficent_cal(forecast_error_naive, test)
corr_drift = correlation_coefficent_cal(forecast_error_drift, test)
corr_ses = correlation_coefficent_cal(forecast_error_ses, test)
corr_holtl = correlation_coefficent_cal(forecast_error_holtl, test)
corr_holtw = correlation_coefficent_cal(forecast_error_holtw, test)
print("Correlation Coefficient between Forecast Error and Test set for Average
Method: {}".format(corr_avg))
print("Correlation Coefficient between Forecast Error and Test set for Naive
Method: {}".format(corr_naive))
print("Correlation Coefficient between Forecast Error and Test set for Drift
Method: {}".format(corr_drift))
print("Correlation Coefficient between Forecast Error and Test set for SES Method:
{}".format(corr_ses))
print("Correlation Coefficient between Forecast Error and Test set for Holt's
Linear Method: {}".format(corr_holtl))
print("Correlation Coefficient between Forecast Error and Test set for Holt's
winter seasonal Method: {}".format(corr_holtw))
d = {'Methods':['Average', 'Naive', 'Drift', 'SES', "HoltL", "HoltW"],
     'Q_val': [round(Q_forecast_avg, 2), round(Q_forecast_naive, 2),
round(Q_forecast_drift,2), round(Q_forecast_SES,2), round(Q_forecast_holtl,2),
round(Q_forecast_holtw,2)],
     'MSE(P)': [round(MSE_train_avg,2), round(MSE_train_naive,2),
round(MSE_train_drift,2), round(MSE_train_SES,2), round(MSE_train_holtl,2),
round(MSE_train_holtw,2)],
     'MSE(F)': [round(MSE_test_avg,2), round(MSE_test_naive,2),
round(MSE_test_drift,2), round(MSE_test_SES,2), round(MSE_test_holtl,2),
round(MSE_test_holtw,2)],
     'var(P)': [round(var_pred_avg,2), round(var_pred_naive,2),
round(var_pred_drift,2), round(var_pred_SES,2), round(var_pred_holtl,2),
round(var_pred_holtw,2)],
     'var(F)':[round(var_forecast_avg,2), round(var_forecast_naive,2),
```

```
round(var_forecast_drift,2), round(var_forecast_SES,2),
round(var_forecast_holtl,2), round(var_forecast_holtw,2)],
     'corrcoeff':[round(corr_avg,2), round(corr_naive,2), round(corr_drift,2),
round(corr_ses,2), round(corr_holtl,2), round(corr_holtw,2)]}
df = pd.DataFrame(data=d)
df = df.set_index('Methods')
pd.set_option('display.max_columns', None)
print(df)


'------------------------------------------------ Tute1 Dataset ---------------
-----------------------------------'
df = pd.read_csv('tute1.csv', header=0)
df['Date'] = pd.date_range(start='1981-3-1', end='2006-3-1', freq='Q')
y = df['Sales']
y.index = df.Date
train, test = train_test_split(y, shuffle=False, test_size=0.2)
train.index.freq = 'Q'
test.index.freq = 'Q'
h = len(test)

print('************************ Tute1 Dataset results **********************')
#Average Method
def avg_method(train):
    y_hat_avg = np.mean(train)
    return y_hat_avg

train_pred_avg = []
for i in range(1,len(train)):
    res = avg_method(train.iloc[0:i])
    train_pred_avg.append(res)

test_forecast_avg1 = np.ones(len(test)) * avg_method(train)
test_forecast_avg = pd.DataFrame(test_forecast_avg1).set_index(test.index)
residual_error_avg = np.array(train[1:]) - np.array(train_pred_avg)
forecast_error_avg = test - test_forecast_avg1
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)

# Naive Method
def naive_method(t):
    return t

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

res = np.ones(len(test)) * train[-1]
naive_test_forecast1 = np.ones(len(test)) * res
```

```
naive_test_forecast = pd.DataFrame(naive_test_forecast1).set_index(test.index)
residual_error_naive = np.array(train[1:]) - np.array(naive_train_pred)
forecast_error_naive = test - naive_test_forecast1
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)

# Drift method
def drift_method(t, h):
    y_hat_drift = t[len(t)-1] + h*((t[len(t)-1]-t[0])/(len(t) - 1))
    return y_hat_drift

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

drift_test_forecast1=[]
for h in range(1, len(test)+1):
    res = drift_method(train, h)
    drift_test_forecast1.append(res)

drift_test_forecast = pd.DataFrame(drift_test_forecast1).set_index(test.index)
residual_error_drift = np.array(train[1:]) - np.array(drift_train_forecast)
forecast_error_drift = np.array(test) - np.array(drift_test_forecast1)
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)

#SES Method
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 = train[0]
```

```
ses_train_pred = ses(train, 0.50, 10)
ses_test_forecast1 = np.ones(len(test)) * (0.5*(train[-1]) + (1-
0.5)*(ses_train_pred[-1]))
ses_test_forecast = pd.DataFrame(ses_test_forecast1).set_index(test.index)
residual_error_ses = np.array(train[1:]) - np.array(ses_train_pred)
forecast_error_ses = np.array(test) - np.array(ses_test_forecast1)
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)

# SES Method using statsmodels for alpha=0.5
# ses_train = train.ewm(alpha=0.5, adjust=False).mean()  # Another way of doing it
ses_model1 = SimpleExpSmoothing(train)
ses_fitted_model1 = ses_model1.fit(smoothing_level=0.5, optimized=False)
ses_train_pred1 = ses_fitted_model1.fittedvalues.shift(-1)
ses_test_forecast1 = ses_fitted_model1.forecast(steps=len(test))
ses_test_forecast1 = pd.DataFrame(ses_test_forecast1).set_index(test.index)
MSE_test_SES1 = np.square(np.subtract(test.values,
np.ndarray.flatten(ses_test_forecast1.values))).mean()

# Holt's Linear Trend
holtl_fitted_model = ets.ExponentialSmoothing(train, trend='add', damped=True,
seasonal=None).fit()
holtl_train_pred = holtl_fitted_model.fittedvalues
holtl_test_forecast = holtl_fitted_model.forecast(steps=len(test))
holtl_test_forecast = pd.DataFrame(holtl_test_forecast).set_index(test.index)
residual_error_holtl = np.subtract(train.values,
np.ndarray.flatten(holtl_train_pred.values))
forecast_error_holtl = np.subtract(test.values,
np.ndarray.flatten(holtl_test_forecast.values))
MSE_train_holtl = np.mean((residual_error_holtl)**2)
MSE_test_holtl = np.mean((forecast_error_holtl)**2)
print("Mean Square Error of prediction errors for Holt's Linear method: ",
MSE_train_holtl)
print("Mean Square Error of forecast errors for Holt's Linear method: ",
MSE_test_holtl)
mean_pred_holtl = np.mean(residual_error_holtl)
var_pred_holtl = np.var(residual_error_holtl)
var_forecast_holtl = np.var(forecast_error_holtl)
print("Mean of prediction errors for Holt's Linear method: ", mean_pred_holtl)
print("Variance of prediction errors for Holt's Linear method: ", var_pred_holtl)
print("Variance of forecast errors for Holt's Linear method: ",
var_forecast_holtl)

# Holt's Winter Seasonal Trend
holtw_fitted_model = ets.ExponentialSmoothing(train, trend='add', damped=True,
seasonal='add', seasonal_periods=4).fit()
holtw_train_pred = holtw_fitted_model.fittedvalues
holtw_test_forecast = holtw_fitted_model.forecast(steps=len(test))
holtw_test_forecast = pd.DataFrame(holtw_test_forecast).set_index(test.index)
residual_error_holtw = np.subtract(train.values,
np.ndarray.flatten(holtw_train_pred.values))
```

```
forecast_error_holtw = np.subtract(test.values,
np.ndarray.flatten(holtw_test_forecast.values))
MSE_train_holtw = np.mean((residual_error_holtw)**2)
MSE_test_holtw = np.mean((forecast_error_holtw)**2)
print("Mean Square Error of prediction errors for Holt's Winter Seasonal method:
", MSE_train_holtw)
print("Mean Square Error of forecast errors for Holt's Winter Seasonal method: ",
MSE_test_holtw)
mean_pred_holtw = np.mean(residual_error_holtw)
var_pred_holtw = np.var(residual_error_holtw)
var_forecast_holtw = np.var(forecast_error_holtw)
print("Mean of prediction errors for Holt's Winter Seasonal method: ",
mean_pred_holtw)
print("Variance of prediction errors for Holt's Winter Seasonal method: ",
var_pred_holtw)
print("Variance of forecast errors for Holt's Winter Seasonal method: ",
var_forecast_holtw)

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(test_forecast_avg, label='Average h-step prediction')
plt.xlabel('Time (Quarterly)')
plt.ylabel('Number of Sales')
plt.title('Average Method - Tute1')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(naive_test_forecast, label='Naive h-step prediction')
plt.xlabel('Time (Quarterly)')
plt.ylabel('Number of Sales')
plt.title('Naive Method - Tute1')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(drift_test_forecast, label='Drift h-step prediction')
plt.xlabel('Time (Quarterly)')
plt.ylabel('Number of Sales')
plt.title('Drift Method - Tute1')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(ses_test_forecast, label='Simple Exponential Smoothing h-step prediction')
plt.xlabel('Time (Quarterly)')
plt.ylabel('Number of Sales')
plt.title('SES Method - Tute1')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
```

```
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(holtl_test_forecast, label="Holt's Linear h-step prediction")
plt.xlabel('Time (Quarterly)')
plt.ylabel('Number of Sales')
plt.title("Holt's Linear Method - Tute1")
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(holtw_test_forecast, label="Holt's Winter Seasonal h-step prediction")
plt.xlabel('Time (Quarterly)')
plt.ylabel('Number of Sales')
plt.title("Holt's Winter Seasonal Method - Tute1")
plt.legend(loc='upper left')
plt.show()

# Auto_correlation for forecast errors and Q value for prediction errors and
forecast errors
#Average Method
k = len(test)
lags = 15
avg_forecast_acf = cal_auto_corr(forecast_error_avg, lags)
Q_forecast_avg = k * np.sum(np.array(avg_forecast_acf[lags:])**2)
print('Q value of forecast errors for Average method: ', Q_forecast_avg)
plt.figure()
plt.stem(range(-(lags-1),lags), avg_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (Average Method)')
plt.show()

# Naive method
k = len(test)
lags = 15
naive_forecast_acf = cal_auto_corr(forecast_error_naive, lags)
Q_forecast_naive = k * np.sum(np.array(naive_forecast_acf[lags:])**2)
print('Q value of forecast errors for Naive method: ', Q_forecast_naive)
plt.figure()
plt.stem(range(-(lags-1),lags), naive_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (Naive Method)')
plt.show()

# Drift Method
k = len(test)
lags = 15
drift_forecast_acf = cal_auto_corr(forecast_error_drift, lags)
Q_forecast_drift = k * np.sum(np.array(drift_forecast_acf[lags:])**2)
print('Q value of forecast errors for Drift method: ', Q_forecast_drift)
plt.figure()
plt.stem(range(-(lags-1),lags), drift_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (Drift Method)')
plt.show()
```

```
# SES method
k = len(test)
lags = 15
ses_forecast_acf = cal_auto_corr(forecast_error_ses, lags)
Q_forecast_SES = k * np.sum(np.array(ses_forecast_acf[lags:])**2)
print('Q value of forecast errors for SES method: ', Q_forecast_SES)
plt.figure()
plt.stem(range(-(lags-1), lags), ses_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (SES Method)')
plt.show()

# holt's linear method
k = len(test)
lags = 15
holtl_forecast_acf = cal_auto_corr(forecast_error_holtl, lags)
Q_forecast_holtl = k * np.sum(np.array(holtl_forecast_acf[lags:])**2)
print("Q value of forecast errors for Holt's Linear method: ", Q_forecast_holtl)
plt.figure()
plt.stem(range(-(lags-1), lags), holtl_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title("Autocorrelation plot for Forecast Error (Holt's Linear Method)")
plt.show()

# holt's Winter Seasonal method
k = len(test)
lags = 15
holtw_forecast_acf = cal_auto_corr(forecast_error_holtw, lags)
Q_forecast_holtw = k * np.sum(np.array(holtw_forecast_acf[lags:])**2)
print("Q value of forecast errors for Holt's Winter Seasonal method: ",
Q_forecast_holtw)
plt.figure()
plt.stem(range(-(lags-1), lags), holtw_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title("Autocorrelation plot for Forecast Error (Holt's winter Seasonal
Method)")
plt.show()

corr_avg = correlation_coefficent_cal(forecast_error_avg, test)
corr_naive = correlation_coefficent_cal(forecast_error_naive, test)
corr_drift = correlation_coefficent_cal(forecast_error_drift, test)
corr_ses = correlation_coefficent_cal(forecast_error_ses, test)
corr_holtl = correlation_coefficent_cal(forecast_error_holtl, test)
corr_holtw = correlation_coefficent_cal(forecast_error_holtw, test)
print("Correlation Coefficient between Forecast Error and Test set for Average
Method: {}".format(corr_avg))
print("Correlation Coefficient between Forecast Error and Test set for Naive
Method: {}".format(corr_naive))
print("Correlation Coefficient between Forecast Error and Test set for Drift
Method: {}".format(corr_drift))
print("Correlation Coefficient between Forecast Error and Test set for SES Method:
{}".format(corr_ses))
print("Correlation Coefficient between Forecast Error and Test set for Holt's
Linear Method: {}".format(corr_holtl))
print("Correlation Coefficient between Forecast Error and Test set for Holt's
```

```
winter seasonal Method: {}".format(corr_holtw))
d = {'Methods':['Average', 'Naive', 'Drift', 'SES', "HoltL", "HoltW"],
     'Q_val': [round(Q_forecast_avg, 2), round(Q_forecast_naive, 2),
round(Q_forecast_drift,2), round(Q_forecast_SES,2), round(Q_forecast_holtl,2),
round(Q_forecast_holtw,2)],
     'MSE(P)': [round(MSE_train_avg,2), round(MSE_train_naive,2),
round(MSE_train_drift,2), round(MSE_train_SES,2), round(MSE_train_holtl,2),
round(MSE_train_holtw,2)],
     'MSE(F)': [round(MSE_test_avg,2), round(MSE_test_naive,2),
round(MSE_test_drift,2), round(MSE_test_SES,2), round(MSE_test_holtl,2),
round(MSE_test_holtw,2)],
     'var(P)': [round(var_pred_avg,2), round(var_pred_naive,2),
round(var_pred_drift,2), round(var_pred_SES,2), round(var_pred_holtl,2),
round(var_pred_holtw,2)],
     'var(F)':[round(var_forecast_avg,2), round(var_forecast_naive,2),
round(var_forecast_drift,2), round(var_forecast_SES,2),
round(var_forecast_holtl,2), round(var_forecast_holtw,2)],
     'corrcoeff':[round(corr_avg,2), round(corr_naive,2), round(corr_drift,2),
round(corr_ses,2), round(corr_holtl,2), round(corr_holtw,2)]}
df = pd.DataFrame(data=d)
df = df.set_index('Methods')
pd.set_option('display.max_columns', None)
print(df)


'-------------------------------------- daily-min-temperatures Dataset --------
-----------------------------------'
df = pd.read_csv('daily-min-temperatures.csv', index_col='Date', parse_dates=True)
y = df['Temp']
train, test = train_test_split(y, shuffle=False, test_size=0.2)
h = len(test)
print('************************ daily-min-temperatures Dataset results
**********************')
#Average Method
def avg_method(train):
    y_hat_avg = np.mean(train)
    return y_hat_avg

train_pred_avg = []
for i in range(1,len(train)):
    res = avg_method(train.iloc[0:i])
    train_pred_avg.append(res)

test_forecast_avg1 = np.ones(len(test)) * avg_method(train)
test_forecast_avg = pd.DataFrame(test_forecast_avg1).set_index(test.index)
residual_error_avg = np.array(train[1:]) - np.array(train_pred_avg)
forecast_error_avg = test - test_forecast_avg1
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)
```

```
# Naive Method
def naive_method(t):
    return t

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

res = np.ones(len(test)) * train[-1]
naive_test_forecast1 = np.ones(len(test)) * res
naive_test_forecast = pd.DataFrame(naive_test_forecast1).set_index(test.index)
#print('h-step ahead Forecast for Naive Method', naive_test_forecast)
residual_error_naive = np.array(train[1:]) - np.array(naive_train_pred)
forecast_error_naive = test - naive_test_forecast1
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)

# Drift method
def drift_method(t, h):
    y_hat_drift = t[len(t)-1] + h*((t[len(t)-1]-t[0])/(len(t) - 1))
    return y_hat_drift

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

drift_test_forecast1=[]
for h in range(1, len(test)+1):
    res = drift_method(train, h)
    drift_test_forecast1.append(res)

drift_test_forecast = pd.DataFrame(drift_test_forecast1).set_index(test.index)
residual_error_drift = np.array(train[1:]) - np.array(drift_train_forecast)
forecast_error_drift = np.array(test) - np.array(drift_test_forecast1)
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)

#SES Method
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 = train[0]
ses_train_pred = ses(train, 0.50, l0)
ses_test_forecast1 = np.ones(len(test)) * (0.5*(train[-1]) + (1-
0.5)*(ses_train_pred[-1]))
ses_test_forecast = pd.DataFrame(ses_test_forecast1).set_index(test.index)
residual_error_ses = np.array(train[1:]) - np.array(ses_train_pred)
forecast_error_ses = np.array(test) - np.array(ses_test_forecast1)
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)

# SES Method using statsmodels for alpha=0.5
# ses_train = train.ewm(alpha=0.5, adjust=False).mean()  # Another way of doing it
ses_model1 = SimpleExpSmoothing(train)
ses_fitted_model1 = ses_model1.fit(smoothing_level=0.5, optimized=False)
ses_train_pred1 = ses_fitted_model1.fittedvalues.shift(-1)
ses_test_forecast1 = ses_fitted_model1.forecast(steps=len(test))
ses_test_forecast1 = pd.DataFrame(ses_test_forecast1).set_index(test.index)
MSE_test_SES1 = np.square(np.subtract(test.values,
np.ndarray.flatten(ses_test_forecast1.values))).mean()

# Holt's Linear Trend
holtl_fitted_model = ets.ExponentialSmoothing(train, trend='add', damped=True,
seasonal=None).fit()
holtl_train_pred = holtl_fitted_model.fittedvalues
holtl_test_forecast = holtl_fitted_model.forecast(steps=len(test))
holtl_test_forecast = pd.DataFrame(holtl_test_forecast).set_index(test.index)
residual_error_holtl = np.subtract(train.values,
np.ndarray.flatten(holtl_train_pred.values))
forecast_error_holtl = np.subtract(test.values,
np.ndarray.flatten(holtl_test_forecast.values))
MSE_train_holtl = np.mean((residual_error_holtl)**2)
MSE_test_holtl = np.mean((forecast_error_holtl)**2)
print("Mean Square Error of prediction errors for Holt's Linear method: ",
MSE_train_holtl)
print("Mean Square Error of forecast errors for Holt's Linear method: ",
MSE_test_holtl)
mean_pred_holtl = np.mean(residual_error_holtl)
var_pred_holtl = np.var(residual_error_holtl)
var_forecast_holtl = np.var(forecast_error_holtl)
print("Mean of prediction errors for Holt's Linear method: ", mean_pred_holtl)
```

```
print("Variance of prediction errors for Holt's Linear method: ", var_pred_holtl)
print("Variance of forecast errors for Holt's Linear method: ",
var_forecast_holtl)

# Holt's Winter Seasonal Trend
holtw_fitted_model = ets.ExponentialSmoothing(train, trend='add', damped=True,
seasonal='add', seasonal_periods=365).fit()
holtw_train_pred = holtw_fitted_model.fittedvalues
holtw_test_forecast = holtw_fitted_model.forecast(steps=len(test))
holtw_test_forecast = pd.DataFrame(holtw_test_forecast).set_index(test.index)
residual_error_holtw = np.subtract(train.values,
np.ndarray.flatten(holtw_train_pred.values))
forecast_error_holtw = np.subtract(test.values,
np.ndarray.flatten(holtw_test_forecast.values))
MSE_train_holtw = np.mean((residual_error_holtw)**2)
MSE_test_holtw = np.mean((forecast_error_holtw)**2)
print("Mean Square Error of prediction errors for Holt's Winter Seasonal method:
", MSE_train_holtw)
print("Mean Square Error of forecast errors for Holt's Winter Seasonal method: ",
MSE_test_holtw)
mean_pred_holtw = np.mean(residual_error_holtw)
var_pred_holtw = np.var(residual_error_holtw)
var_forecast_holtw = np.var(forecast_error_holtw)
print("Mean of prediction errors for Holt's Winter Seasonal method: ",
mean_pred_holtw)
print("Variance of prediction errors for Holt's Winter Seasonal method: ",
var_pred_holtw)
print("Variance of forecast errors for Holt's Winter Seasonal method: ",
var_forecast_holtw)

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(test_forecast_avg, label='Average h-step prediction')
plt.xlabel('Time (Daily)')
plt.ylabel('Daily Minimum Temperature')
plt.title('Average Method - daily-min-temperatures')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(naive_test_forecast, label='Naive h-step prediction')
plt.xlabel('Time (Daily)')
plt.ylabel('Daily Minimum Temperature')
plt.title('Naive Method - daily-min-temperatures')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(drift_test_forecast, label='Drift h-step prediction')
plt.xlabel('Time (Daily)')
plt.ylabel('Daily Minimum Temperature')
plt.title('Drift Method - daily-min-temperatures')
plt.legend(loc='upper left')
plt.show()
```

```
fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(ses_test_forecast, label='Simple Exponential Smoothing h-step prediction')
plt.xlabel('Time (Daily)')
plt.ylabel('Daily Minimum Temperature')
plt.title('SES Method - daily-min-temperatures')
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(holtl_test_forecast, label="Holt's Linear h-step prediction")
plt.xlabel('Time (Daily)')
plt.ylabel('Daily Minimum Temperature')
plt.title("Holt's Linear Method - daily-min-temperatures")
plt.legend(loc='upper left')
plt.show()

fig, ax = plt.subplots(figsize=(10,8))
ax.plot(train, label='Training set')
ax.plot(test, label='Testing set')
ax.plot(holtw_test_forecast, label="Holt's Winter Seasonal h-step prediction")
plt.xlabel('Time (Daily)')
plt.ylabel('Daily Minimum Temperature')
plt.title("Holt's Winter Seasonal Method - daily-min-temperatures")
plt.legend(loc='upper left')
plt.show()

# Auto_correlation for forecast errors and Q value for prediction errors and
forecast errors
#Average Method
k = len(test)
lags = 15
avg_forecast_acf = cal_auto_corr(forecast_error_avg, lags)
Q_forecast_avg = k * np.sum(np.array(avg_forecast_acf[lags:])**2)
print('Q value of forecast errors for Average method: ', Q_forecast_avg)
plt.figure()
plt.stem(range(-(lags-1),lags), avg_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (Average Method)')
plt.show()

# Naive method
k = len(test)
lags = 15
naive_forecast_acf = cal_auto_corr(forecast_error_naive, lags)
Q_forecast_naive = k * np.sum(np.array(naive_forecast_acf[lags:])**2)
print('Q value of forecast errors for Naive method: ', Q_forecast_naive)
plt.figure()
plt.stem(range(-(lags-1),lags), naive_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (Naive Method)')
plt.show()
```

```
# Drift Method
k = len(test)
lags = 15
drift_forecast_acf = cal_auto_corr(forecast_error_drift, lags)
Q_forecast_drift = k * np.sum(np.array(drift_forecast_acf[lags:])**2)
print('Q value of forecast errors for Drift method: ', Q_forecast_drift)
plt.figure()
plt.stem(range(-(lags-1),lags), drift_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (Drift Method)')
plt.show()

# SES method
k = len(test)
lags = 15
ses_forecast_acf = cal_auto_corr(forecast_error_ses, lags)
Q_forecast_SES = k * np.sum(np.array(ses_forecast_acf[lags:])**2)
print('Q value of forecast errors for SES method: ', Q_forecast_SES)
plt.figure()
plt.stem(range(-(lags-1), lags), ses_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title('Autocorrelation plot for Forecast Error (SES Method)')
plt.show()

# holt's linear method
k = len(test)
lags = 15
holtl_forecast_acf = cal_auto_corr(forecast_error_holtl, lags)
Q_forecast_holtl = k * np.sum(np.array(holtl_forecast_acf[lags:])**2)
print("Q value of forecast errors for Holt's Linear method: ", Q_forecast_holtl)
plt.figure()
plt.stem(range(-(lags-1), lags), holtl_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title("Autocorrelation plot for Forecast Error (Holt's Linear Method)")
plt.show()

# holt's Winter Seasonal method
k = len(test)
lags = 15
holtw_forecast_acf = cal_auto_corr(forecast_error_holtw, lags)
Q_forecast_holtw = k * np.sum(np.array(holtw_forecast_acf[lags:])**2)
print("Q value of forecast errors for Holt's Winter Seasonal method: ",
Q_forecast_holtw)
plt.figure()
plt.stem(range(-(lags-1), lags), holtw_forecast_acf, use_line_collection=True)
plt.xlabel('Lags')
plt.ylabel('Magnitude')
plt.title("Autocorrelation plot for Forecast Error (Holt's winter Seasonal
Method)")
plt.show()

corr_avg = correlation_coefficent_cal(forecast_error_avg, test)
corr_naive = correlation_coefficent_cal(forecast_error_naive, test)
corr_drift = correlation_coefficent_cal(forecast_error_drift, test)
corr_ses = correlation_coefficent_cal(forecast_error_ses, test)
corr_holtl = correlation_coefficent_cal(forecast_error_holtl, test)
```

```
corr_holtw = correlation_coefficent_cal(forecast_error_holtw, test)
print("Correlation Coefficient between Forecast Error and Test set for Average
Method: {}".format(corr_avg))
print("Correlation Coefficient between Forecast Error and Test set for Naive
Method: {}".format(corr_naive))
print("Correlation Coefficient between Forecast Error and Test set for Drift
Method: {}".format(corr_drift))
print("Correlation Coefficient between Forecast Error and Test set for SES Method:
{}".format(corr_ses))
print("Correlation Coefficient between Forecast Error and Test set for Holt's
Linear Method: {}".format(corr_holtl))
print("Correlation Coefficient between Forecast Error and Test set for Holt's
winter seasonal Method: {}".format(corr_holtw))
d = {'Methods':['Average', 'Naive', 'Drift', 'SES', "HoltL", "HoltW"],
     'Q_val': [round(Q_forecast_avg, 2), round(Q_forecast_naive, 2),
round(Q_forecast_drift,2), round(Q_forecast_SES,2), round(Q_forecast_holtl,2),
round(Q_forecast_holtw,2)],
     'MSE(P)': [round(MSE_train_avg,2), round(MSE_train_naive,2),
round(MSE_train_drift,2), round(MSE_train_SES,2), round(MSE_train_holtl,2),
round(MSE_train_holtw,2)],
     'MSE(F)': [round(MSE_test_avg,2), round(MSE_test_naive,2),
round(MSE_test_drift,2), round(MSE_test_SES,2), round(MSE_test_holtl,2),
round(MSE_test_holtw,2)],
     'var(P)': [round(var_pred_avg,2), round(var_pred_naive,2),
round(var_pred_drift,2), round(var_pred_SES,2), round(var_pred_holtl,2),
round(var_pred_holtw,2)],
     'var(F)':[round(var_forecast_avg,2), round(var_forecast_naive,2),
round(var_forecast_drift,2), round(var_forecast_SES,2),
round(var_forecast_holtl,2), round(var_forecast_holtw,2)],
     'corrcoeff':[round(corr_avg,2), round(corr_naive,2), round(corr_drift,2),
round(corr_ses,2), round(corr_holtl,2), round(corr_holtw,2)]}
df = pd.DataFrame(data=d)
df = df.set_index('Methods')
pd.set_option('display.max_columns', None)
print(df)
```