Multivariate Modeling
DATS 6450-15
Lab # 4
Taisha Ferguson
February 17, 2020 (TF)

Abstract

The primary purpose of this lab was to compare different simple forecasting models on the same data set in order to find the model with the best performance. Different statistical test such as Q values, Mean Square Error, Mean of Prediction Errors, and Variance of Prediction Errors were used to test the performance on each model.

Introduction

In this lab the Average, Naïve, Drift, and Simple Exponential Smoothing (SES) forecasting methods were tested on the same dataset in order to compare their performance. These methods are all known as simple forecasting and are used as benchmarks for more complex forecasting models. The dataset was sample dataset with 5 values. Using the first few values one-step ahead prediction was used to calculate the next and then the next predicted value. After computing the predicted values, different statistical tests were used to compare the errors of each model.

Methods, theory, and procedures

In order to calculate the predicted values for the different lags hand calculations and then Python code were used based on the following formulas:

Average Method

$$\hat{y}_{T+h|T} = \bar{y} = (y_1 + \cdots + y_T)/T.$$

Naïve Method

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

Drift Method

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

Simple Exponential Smoothing

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

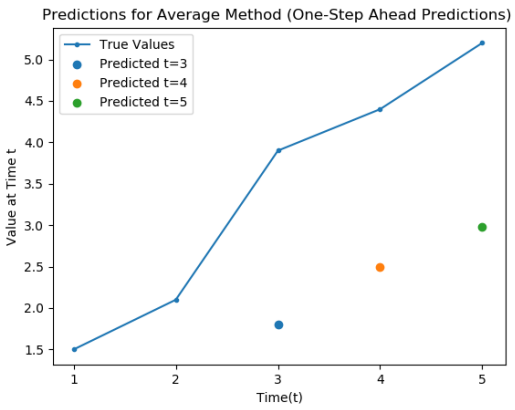
For each of the above methods, one-step ahead predictions were used. For one-step ahead calculations, all of the previous values are used in the computation of the next prediction.

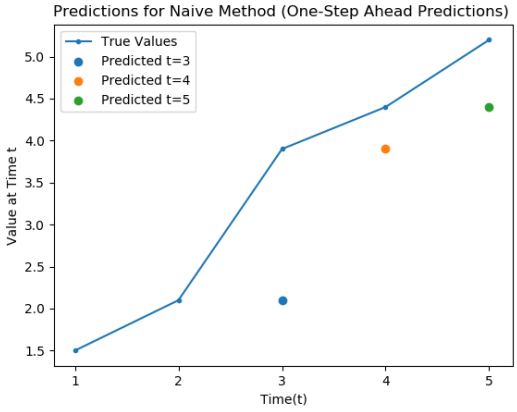
After computing the predicted values up to t=5, which is the last true value that was given, the errors were calculated by subtracting the true values form the predicted values. Using the predicted errors, the mean squared error(MSE), the mean of the predicted errors, the variance of the predicted errors, the ACF of the predicted errors, and the Q values were calculated. Each of these values are different metrics for judging the performance of the different models. For MSE lower values usually indicate a better model. If the mean of the prediction errors are close to zero that means there is likely no bias in the model. Typically for good models, you want a mostly constant variance in the prediction errors. Q values is an overall metric for autocorrelation amongst prediction errors. Models with lower Q values reflect lower amount of autocorrelation between predicted errors.

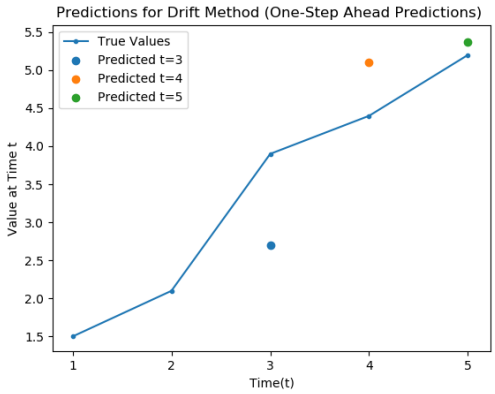
Answers to Asked Questions

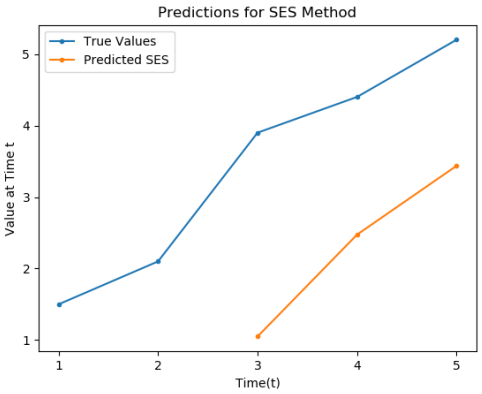
Q: Compare the above 4 methods by looking at Q values, ACF, MSE, mean and variance of prediction errors and pick the best estimator. You need to justify your answer why the picked estimator is better than the others. Please keep in mind that by using the drift method as an estimator, you will lose the first two data samples. This means that if your original dataset is T, then you will have T-2 estimates.

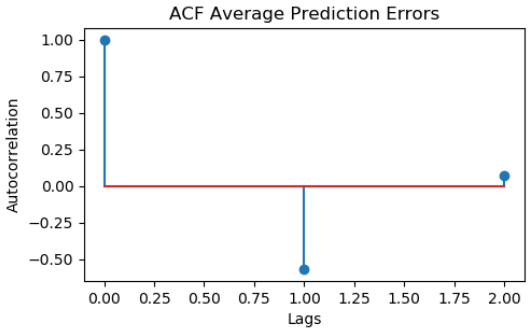
A: Based on the forecasting metrics Q value, ACF, MSE, and mean and variance of prediction errors I believe that the Drift method is the best estimate for this data set. The Drift method had the lowest MSE and the second best Q value. The one drawback of the Drift method was it's high prediction error variance. The second best model in my opinion was the Naïve method. It had the second lowest MSE and the lowest Q value. It also had the second lowest mean of prediction error and the lowest variance of the prediction errors.

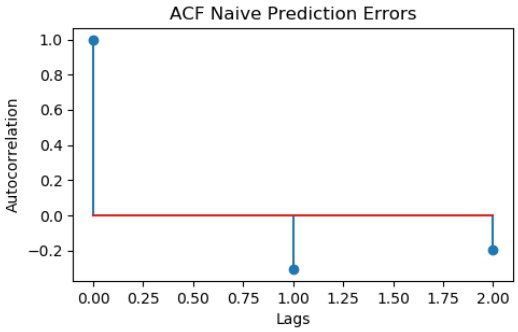


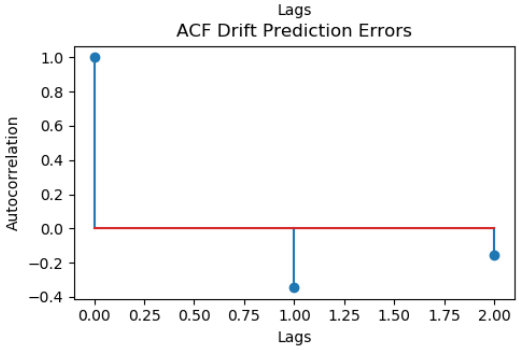


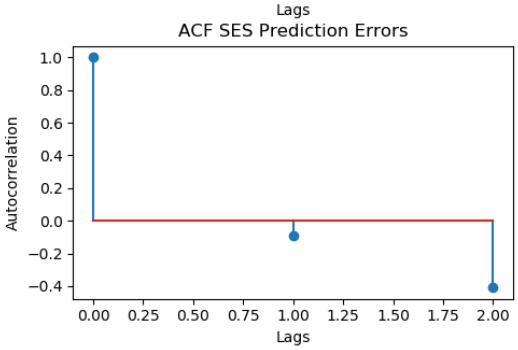












Forecasting Methods	Q Values	MSE	Prediction Error Mean	Prediction Error Variance
Average	1.318009735	4.323541667	2.075	0.017916667
Naive	0.525950463	1.376666667	1.033333333	0.308888889
Drift	0.568486629	0.652592593	0.111111111	0.640246914

SES	0.695024875	4.978177083	2.179166667	0.229409722

Conclusion

The main objective of this lab was to test the Average, Naïve, Drift, and Simple Exponential Smoothing forecasting methods on the same dataset and evaluate their performance. For this dataset, the Drift method had the best performance. This was concluded due to it's low Q value and MSE score in relation to the other models. The Naïve method also did well in comparison to other models with the second lowest MSE and the lowest Q value.

Appendix A – Handwritten Computations

① Average Forecast Method (One-Step Ahead Prediction)

$$\hat{y}_{t+n|T} = \frac{y_1 + y_2 + \cdots y_T}{T}$$

t	y_t	$\hat{y}_{t|T}$	error = $y_t - \hat{y}_t$	error²
1	1.5			
2	2.1			
3	3.9	1.8	2.1	4.41
4	4.4	2.5	1.9	3.61
5	5.2	2.975	2.225	4.95

$$MSE = \frac{4.41 + 3.61 + 4.95}{3}$$

$$\boxed{MSE = 4.32}$$

② Naive Method (One-Step Ahead Prediction)

t	y_t	$\hat{y}_t$	error	error²
1	1.5			
2	2.1	1.5	0.6	.36
3	3.9	2.1	1.8	3.24
4	4.4	3.9	0.5	.25
5	5.2	~~4.4~~ 4.2	0.8	0.64

$$MSE = \frac{.36 + 3.24 + .25 + 0.64}{4}$$

~~= 4.49~~ $\boxed{MSE = 1.225}$

(8) Drift Forecast Method (One step Ahead Prediction)

$$\hat{Y}_t = Y_T + h\left(\frac{Y_T - Y_1}{T-1}\right)$$

$h=1 \Rightarrow$ one-step ahead

t	y_t	$\hat{y}_t$	error	error^2
1	1.5			
2	2.1			
3	3.9	2.7	1.2	1.44
4	4.4	5.1	-0.7	.49
5	5.2	5.367	-0.167	.027889

$$\hat{Y}_3 = 2.1 + \left(\frac{2.1-1.5}{1}\right)$$

$$= 2.7$$

$$\hat{Y}_4 = 3.9 + \left(\frac{3.9-1.5}{2}\right)$$

$$= 5.1$$

$$\hat{Y}_5 = 4.4 + \left(\frac{4.4-1.5}{3}\right)$$

$$= 5.367$$

$$MSE = \frac{1.44 + .49 + .027889}{3}$$

$$\boxed{MSE = .65}$$

(9) Simple Exponential Smoothing Method

$$\hat{Y}_{t+1|t} = \alpha Y_t + (1-\alpha)\hat{Y}_{t|t-1}$$

$\alpha = 0.5 \quad l_0 = 0$

t	Y_t	$\hat{Y}_t$	error	error^2
1	1.5			
2	2.1	.75	1.35	1.8225
3	3.9	1.425	2.475	6.1256
4	4.4	2.66	1.74	3.0276
5	5.2	3.53	1.67	2.7889

$\hat{Y}_2 = (.5)(1.5) = .75$

$\hat{Y}_3 = (.5)(2.1) + (.5)(.75) = 1.425$

$\hat{Y}_4 = (.5)(3.9) + (.5)(1.425) = 2.66$

$\hat{Y}_5 = (.5)(4.4) + (.5)(2.66) = 3.53$

$$MSE = \frac{1.8225 + 6.1256 + 3.0276 + 2.7889}{4}$$

$$\boxed{MSE = 3.44}$$

Appendix B – Python Code

```
# Import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


# ---------------------- AVERAGE METHOD ------------------------------------------
----------
# 2 - Write a python code that perform the task in step 1.
# Plot the True values versus Predicted values in one graph with different marker.
# You need to plot 3 graphs (because 3 models were developed in step 1) for this
section.
# Add an appropriate title, legend, x-label, y-label to each graph
yt = np.array([1.5, 2.1,3.9, 4.4, 5.2])
y_hat = []
sum=0
for i in range(0,4):
    sum += yt[i]
    average = sum/(i+1)
    y_hat.append(average)
y_hat = np.array(y_hat[1:])
print("average predictions:", y_hat)
error = yt[2:] - y_hat
#print(error)
MSE1 = np.mean(np.square(error))
print("average MSE: ", MSE1)
df= pd.DataFrame(index=[1,2,3,4,5], columns= ['yt', 'y_hat', 'error'])
df['yt'] = np.array([1.5, 2.1,3.9, 4.4, 5.2])
df.iloc[2:,1] = y_hat
df.iloc[2:,2] = error
fig, ax = plt.subplots()
ax.plot(df['yt'], label="True Values", marker='.')
ax.scatter(x=3, y=df['y_hat'][3], label="Predicted t=3")
ax.scatter(x=4, y=df['y_hat'][4], label="Predicted t=4")
ax.scatter(x=5, y=df['y_hat'][5], label="Predicted t=5")
ax.set_xticks([1,2,3,4,5])
ax.set_xlabel("Time(t)")
ax.set_ylabel("Value at Time t")
ax.set_title("Predictions for Average Method (One-Step Ahead Predictions) ")
ax.legend()
plt.show()


# 3 - Using python, calculate the forecast errors(3 errors for 3 models)
# which is the difference between predicted values and true values and display it as.
print(df.head())

# 4 - Using python, calculate the mean and variance of prediction errors.
mean1 = np.mean(df['error'])
var1 = np.var(df['error'])
print("The mean of forecast errors for the Average Method is = ",
np.mean(df['error']))
print("The variance of forecast for the Average Method is = ", np.var(df['error']))

# 5 - Calculate the sum square of the prediction errors (SSE) and display the
following message on the console.
print("The sum square error for the Average Method is: ",
```

```
np.sum(np.square(df['error'])))

# 6 - Calculate the Q value for this estimateand display the following message(h=3 to
be consistent)
def autocorrelation(y):
    y_bar = np.sum(y) / len(y)
    acf_list = []
    numerator = 0
    denomintator = 0
    n = len(y)
    for i in range(len(y)):
        denomintator += np.square(y[i] - y_bar)
    p = 0
    while p < n:
        r = n - p
        for j, z in zip(range(p, n), range(0, r)):
            numerator += ((y[j] - y_bar) * (y[z] - y_bar))
        quotient = numerator / denomintator
        acf_list.append(quotient)
        numerator = 0
        p = p + 1
    return acf_list
errorlist = list(df['error'][2:])
print(errorlist)
ACFerrors1 = autocorrelation(error)
print(ACFerrors1)
Q1 = 4*np.sum(np.square(ACFerrors1[1:]))
print("The Q value for the Averaging Method is = ", Q1)

# ---------------------- NAIVE METHOD ----------------------------------------------
--------
# 7 - Repeat step 1 through 6 with the Naive Method
# Calculate one step Ahead Prediction
yt = np.array([1.5, 2.1,3.9, 4.4, 5.2])
y_hat_naive = yt[1:4]
error = yt[2:] - y_hat_naive
print("Naive predictions: ", y_hat_naive)
#print(error)
MSE2 = np.mean(np.square(error))
print("Naive MSE:", MSE2)

# Plot True Values versus Predicted Values
df= pd.DataFrame(index=[1,2,3,4,5], columns= ['yt', 'y_hat', 'error'])
df['yt'] = np.array([1.5, 2.1,3.9, 4.4, 5.2])
df.iloc[2:,1] = y_hat_naive
df.iloc[2:,2] = error
fig, ax = plt.subplots()
ax.plot(df['yt'], label="True Values", marker='.')
ax.scatter(x=3, y=df['y_hat'][3], label="Predicted t=3")
ax.scatter(x=4, y=df['y_hat'][4], label="Predicted t=4")
ax.scatter(x=5, y=df['y_hat'][5], label="Predicted t=5")
ax.set_xticks([1,2,3,4,5])
ax.set_xlabel("Time(t)")
ax.set_ylabel("Value at Time t")
ax.set_title("Predictions for Naive Method (One-Step Ahead Predictions) ")
ax.legend()
plt.show()

# 3 - Using python, calculate the forecast errors(3 errors for 3 models)
# which is the difference between predicted values and true values and display it as.
print(df.head())
```

```
# 4 - Using python, calculate the mean and variance of prediction errors.
mean2 = np.mean(df['error'])
var2 = np.var(df['error'])
print("The mean of forecast errors for the Naive Method is = ", np.mean(df['error']))
print("The variance of forecast for the Naive Method is = ", np.var(df['error']))

# 5 - Calculate the sum square of the prediction errors (SSE) and display the
following message on the console.
print("The sum square error for the Naive Method is: ",
np.sum(np.square(df['error'])))

# 6 - Calculate the Q value for this estimateand display the following message(h=3 to
be consistent)
errorlist = list(df['error'][2:])
print(errorlist)
ACFerrors2 = autocorrelation(error)
print(ACFerrors2)
Q2 = 4*np.sum(np.square(ACFerrors2[1:]))
print("The Q value for the Naive Method is = ", Q2)


#---------------------- DRIFT METHOD ------------------------------------------------
-------
# 8 Repeat step 1 through 6 with the drift method
yt = np.array([1.5, 2.1,3.9, 4.4, 5.2])
y_hat_drift = []
for i in range(1,4):
        x = yt[i] + ((yt[i]-yt[0])/i)
        y_hat_drift.append(x)
y_hat_drift = np.array(y_hat_drift)
print(y_hat_drift)
error = yt[2:] - y_hat_drift
print("Drift predictions: ", y_hat_drift)
#print(error)
MSE3 = np.mean(np.square(error))
print("Drift MSE: ",MSE3)
df= pd.DataFrame(index=[1,2,3,4,5], columns= ['yt', 'y_hat', 'error'])
df['yt'] = np.array([1.5, 2.1,3.9, 4.4, 5.2])
df.iloc[2:,1] = y_hat_drift
df.iloc[2:,2] = error
fig, ax = plt.subplots()
ax.plot(df['yt'], label="True Values", marker='.')
ax.scatter(x=3, y=df['y_hat'][3], label="Predicted t=3")
ax.scatter(x=4, y=df['y_hat'][4], label="Predicted t=4")
ax.scatter(x=5, y=df['y_hat'][5], label="Predicted t=5")
ax.set_xticks([1,2,3,4,5])
ax.set_xlabel("Time(t)")
ax.set_ylabel("Value at Time t")
ax.set_title("Predictions for Drift Method (One-Step Ahead Predictions) ")
ax.legend()
plt.show()


# 3 - Using python, calculate the forecast errors(3 errors for 3 models)
# which is the difference between predicted values and true values and display it as.
print(df.head())

# 4 - Using python, calculate the mean and variance of prediction errors.
mean3 = np.mean(df['error'])
var3 = np.var(df['error'])
```

```
print("The mean of forecast errors for the Drift Method is = ", np.mean(df['error']))
print("The variance of forecast for the Drift Method is = ", np.var(df['error']))

# 5 - Calculate the sum square of the prediction errors (SSE) and display the
following message on the console.
print("The sum square error for the Drift Method is: ",
np.sum(np.square(df['error'])))

# 6 - Calculate the Q value for this estimateand display the following message(h=3 to
be consistent)
errorlist = list(df['error'][2:])
print(errorlist)
ACFerrors3 = autocorrelation(error)
print(ACFerrors3)
Q3 = 4*np.sum(np.square(ACFerrors3[1:]))
print("The Q value for the Drift Method is = ", Q3)


#---------------------- Simple Exponential Smoothing (SES) METHOD --------------------
-------------------------------------
yt = np.array([1.5, 2.1,3.9, 4.4, 5.2])
y_hat_SES = [0,0,0]

for i in range(1,4):
    if i == 1:
        x = 0.5*yt[i]
        y_hat_SES[i-1] = x

    else:
        x = 0.5 * yt[i] + 0.5*y_hat_SES[i-2]
        y_hat_SES[i-1] =x


print(y_hat_SES)

y_hat_SES = np.array(y_hat_SES)
print(y_hat_SES)
error = yt[2:] - y_hat_SES
#print(error)
print("SES predictions: ", y_hat_SES)
MSE4 = np.mean(np.square(error))
print("SES MSE: ", MSE4)
# Plot the True Values versus the Predicted Values

df= pd.DataFrame(index=[1,2,3,4,5], columns= ['yt', 'yh_SES'])
df['yt'] = np.array([1.5, 2.1,3.9, 4.4, 5.2])
df.iloc[2:,1] = y_hat_SES
fig, ax = plt.subplots()
ax.plot(df['yt'], label="True Values", marker='.')
ax.plot(df['yh_SES'], label="Predicted SES", marker = ".")
ax.set_xticks([1,2,3,4,5])
ax.set_xlabel("Time(t)")
ax.set_ylabel("Value at Time t")
ax.set_title("Predictions for SES Method")
ax.legend()
plt.show()

# Calculate Forecast errors

df['SES_errors'] = df['yt']-df['yh_SES']
print(df.head())
```

```
# Calculate Mean and Variance of Prediction Errors
mean4 = np.mean(df['SES_errors'])
var4 = np.var(df['SES_errors'])
print("The mean of forecast errors for the SES Method is  = ",
np.mean(df['SES_errors']))
print("The variance of forecast errors for the SES Method is= ",
np.var(df['SES_errors']))

# Calculate the SSE
print("The sum square error for the SES is: ", np.sum(np.square(df['SES_errors'])))


# Calculate the Q value

T2errors = list(df['SES_errors'][2:])
print(T2errors)
ACFerrors4 = autocorrelation(T2errors)
print(ACFerrors4)

Q4 = 4*np.sum(np.square(ACFerrors4[1:]))
print("The Q value for the SES method is = ", Q4)

# 10 - Create a table to compare the 4 forecast methods above by displaying Q
values(h=3), MSE, Mean of prediction errors,
# variance of prediction errors


df= pd.DataFrame(columns= ['Forecasting Methods', 'Q Values', 'MSE', 'Prediction Error
Mean','Prediction Error Variance'])
df['Forecasting Methods'] = ["Average", "Naive", "Drift", "SES"]
df['Q Values'] = [Q1,Q2,Q3,Q4]
df["MSE"] = [MSE1, MSE2, MSE3, MSE4]
df['Prediction Error Mean'] = [mean1,mean2, mean3, mean4]
df['Prediction Error Variance'] = [var1, var2, var3, var4]

print(df.head())
df.to_csv("modeltable1.csv")

# 11 - Using the Python program developed in the previous LAB, plot the ACF of
predicted errors

fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
ax1.stem(ACFerrors1)
ax1.set_title("ACF Average Prediction Errors")
ax1.set_ylabel('Autocorrelation')
ax1.set_xlabel('Lags')
ax2.stem(ACFerrors2)
ax2.set_title("ACF Naive Prediction Errors")
ax2.set_ylabel('Autocorrelation')
ax2.set_xlabel('Lags')
ax3.stem(ACFerrors3)
ax3.set_title("ACF Drift Prediction Errors")
ax3.set_ylabel('Autocorrelation')
ax3.set_xlabel('Lags')
ax4.stem(ACFerrors4)
ax4.set_title("ACF SES Prediction Errors")
ax4.set_ylabel('Autocorrelation')
ax4.set_xlabel('Lags')
plt.show()
```