Multivariate Modeling
DATS 6450-15
Lab # 5
Taisha Ferguson
February 24, 2020 (TF)

Abstract

The primary purpose of this lab was to use Linear Regression and the Least Square Estimate methods to estimate model the relationship between the bill amount and the tip amount in a given dataset. The tip amount was the dependent variable and the bill amount was the independent variable. After creating the linear regression model via the least square estimate model the model had and Adjusted R-Square of 99% meaning that 99% of the variation in observed tip amounts can be explained by the relationship between tip amount and bill amount.

Introduction

In this lab the relationship between tip amount and bill amount from a given dataset was modeled by Linear Regression using Least Square Estimate method. Linear Regression is a statistical model that estimates the linear relationship between the dependent variable and independent variable by creating estimated coefficients for a linear equation. The coefficients are estimated by the Least Square method which minimizes the function Sum of Square Errors.

Methods, theory, and procedures

In order the estimate the model the linear relationship between bill and tip amounts the following equations were used:

Linear Regression Model

$$y_t = \beta_0 + \beta_1 x_{1,t} + \varepsilon_t$$

Least Square Estimate (Matrix Form)

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

For the Linear Regression equation, y_t is the predicted tip amount, $x_{1,t}$ is the given bill amount, ε_t is the error not accounted for in the model, and β_0 β_1 are coefficients that are learned through the Least Square Estimation method. The Least Square formula is the matrix form of the calculation where **X** and **Y** are the matrix presentation of the x and y observations.

The calculations for these methods were implemented in python first semi-manually using python libraries Pandas and Numpy and then using the OLS estimator from the Statsmodel library. Both estimates produced the same results.

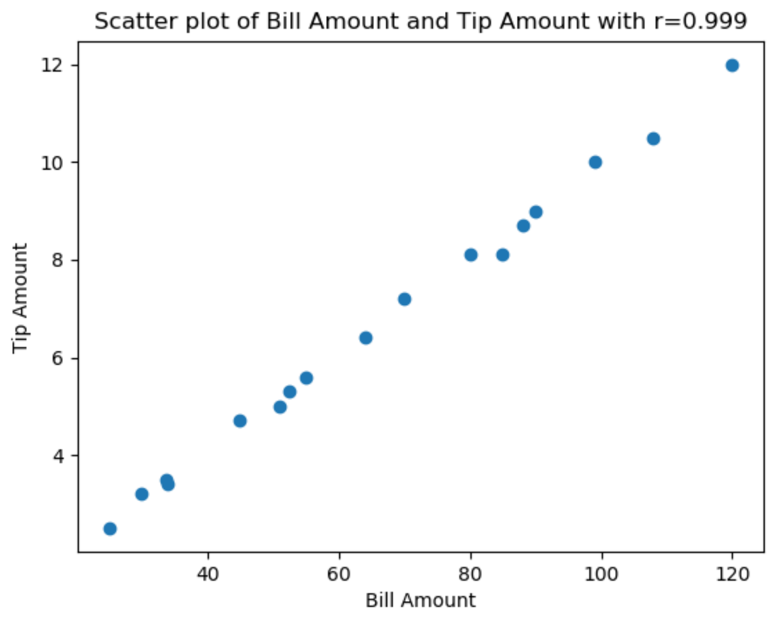
After computed the coefficients, they were then used to make predictions. The predictions were analyzed using a several statistical test such as the Root Mean Squared Error (RMSE), the

Box Pierce test, the R-squared, and the Adjusted R squared. All of these test confirmed the the model was successful at predicted the tip amount based on the relationship between tip amounts and bill amounts.

Answers to Asked Questions

Q1: Plot the scatter plot of the bill versus tip and calculate the correlation coefficient between them. Display the correlation coefficient on the title as it was done in previous labs. Do you think that the linear regression model could be a good estimator for this dataset? Justify your answer.

A1: Yes, I believe that linear regression could be a good estimator for this data because there is a high linear correlation between the tip amount and the bill amount is this dataset as evidenced in the correlation coefficient .99 and the scatter plot below. Correlation coefficients with values close to one represent a strong linear relationship.



Q2: Calculate the mean of the forecast errors and display as a message. Is this a bias estimator or an unbiased estimator? Justify your answer.

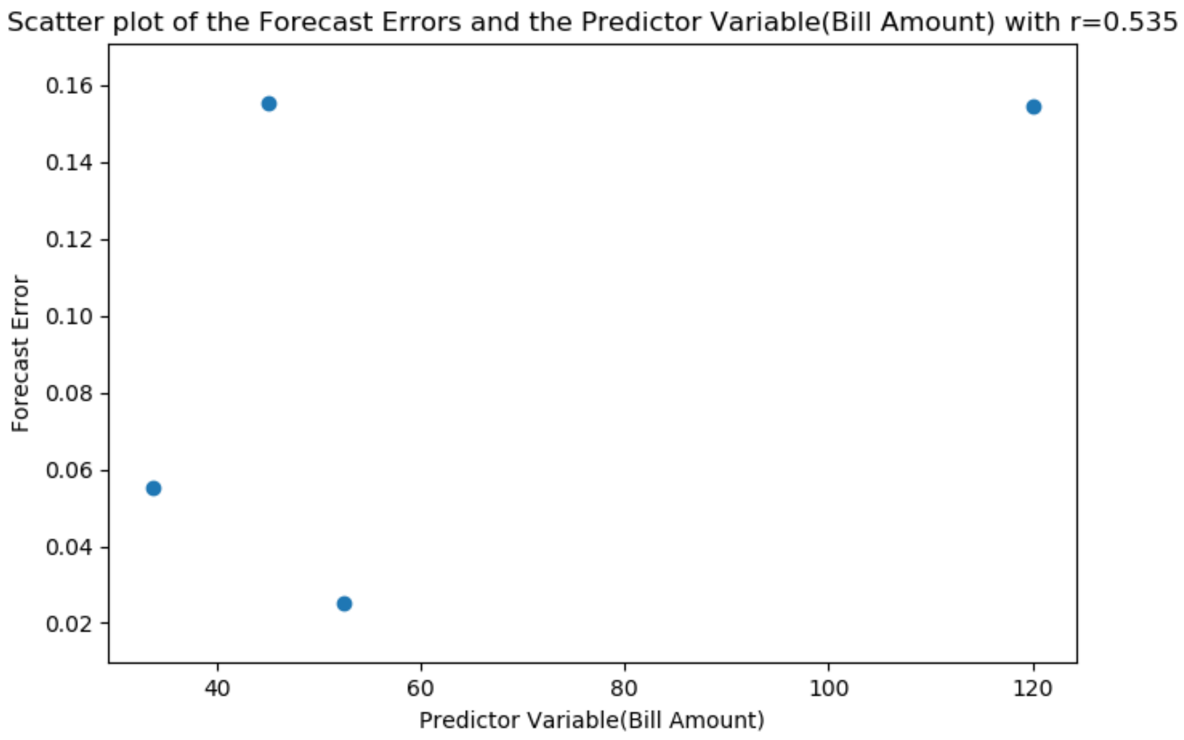
A2: The mean of the forecast error is 0.098. This estimator is not biased because the mean is close to 0

Q3: Estimate the standard error for this predictor using the following equation and display the value as a message. What is your observation about the standard error?

A3: The standard error of the for this predictions is 0.161. This standard error is a little higher than the highest error term.

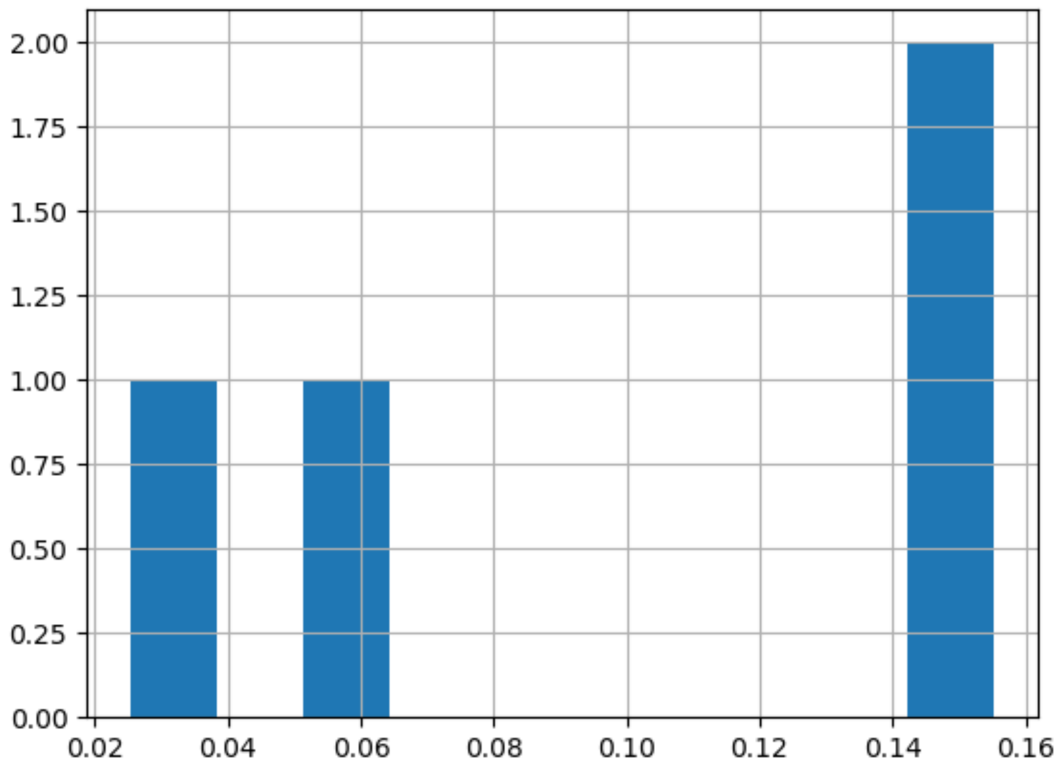
Q4: Plot the scatter plot between forecast errors and predictor variable and display the correlation coefficient between them on the title. Are they related? Justify the accuracy of this predictor by observing the correlation coefficient between them.

A4: There appears to be a moderate correlation between the forecast errors and the predicted values. The correlation is that as the Bill amount increases the forecast error increases. This correlation makes sense because there is a larger range for errors as the predictor increases.



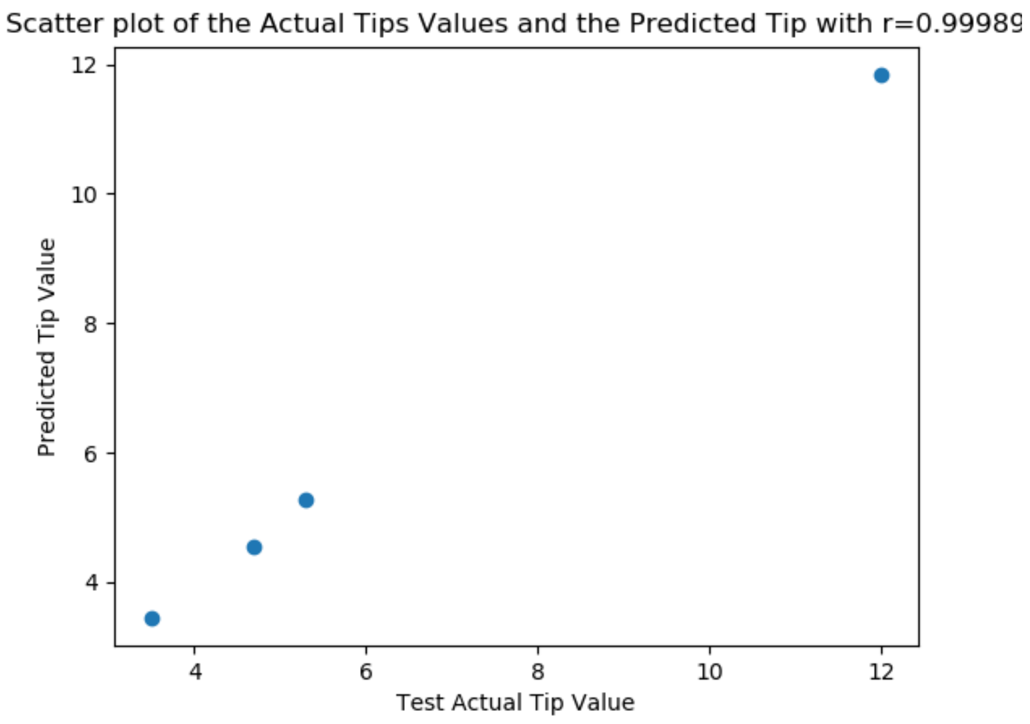
Q5: Plot the histogram for the forecast errors. Is it a normal distribution? Justify your answer.

A5: I believe that the forecast errors a close to be being normally distributed because there is an equal amount of values on the each side of the mean.



Q6: Plot the scatter plot between y-test and $\hat{y}t$and display the correlation coefficient between them on the title. Justify the accuracy of this predictor by observing the correlation coefficient between y-test and $\hat{y}t$.

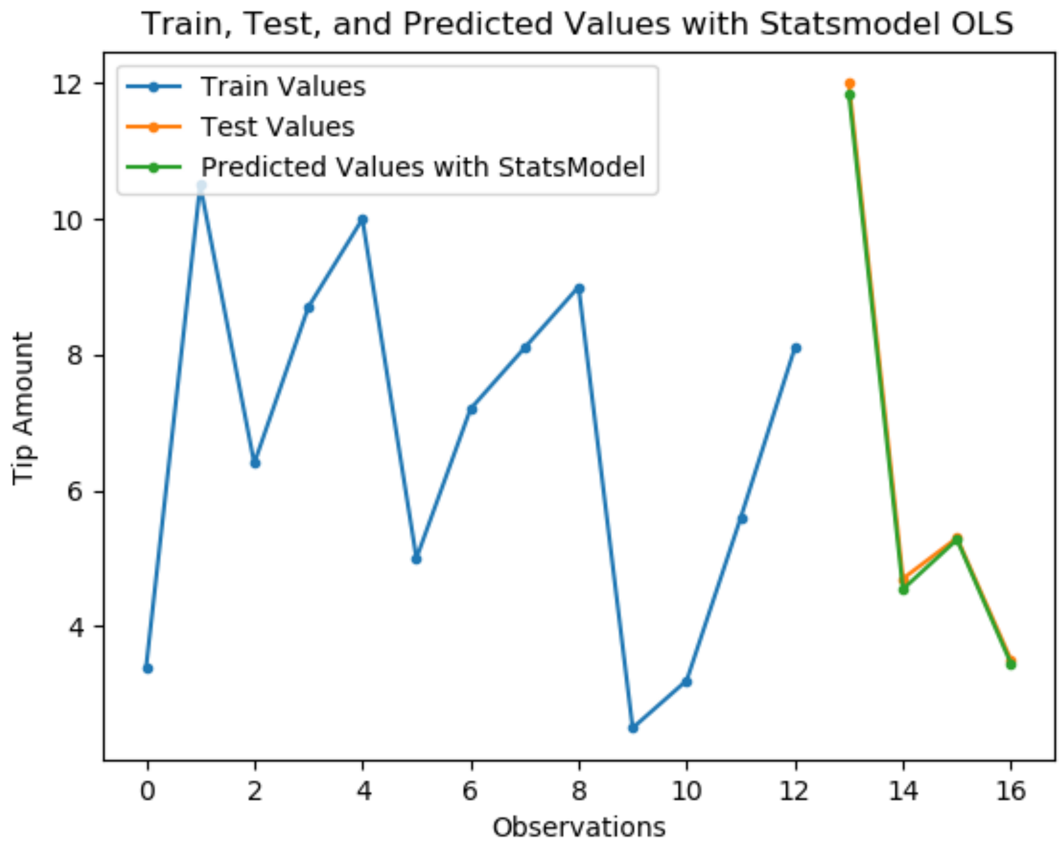
A6: The predictions show strong accuracy because the correlation is .99 which is very close to one. This mean that they have an almost perfectly positive linear relationship.



Q7: Using the Python library “statsmodels”and OLS function, find the regression model of above data set. Compare the coefficients of the derive model with one calculated in step 4. Display the summery of the regression model and interpret the t-value of the regression coefficient. Which parameters in the model can be removed? Compare the R-squared and adjusted R-squared in the table with the values calculated in step10. Perform a prediction using OLS function and plot the train, test and predicted values in one graph.

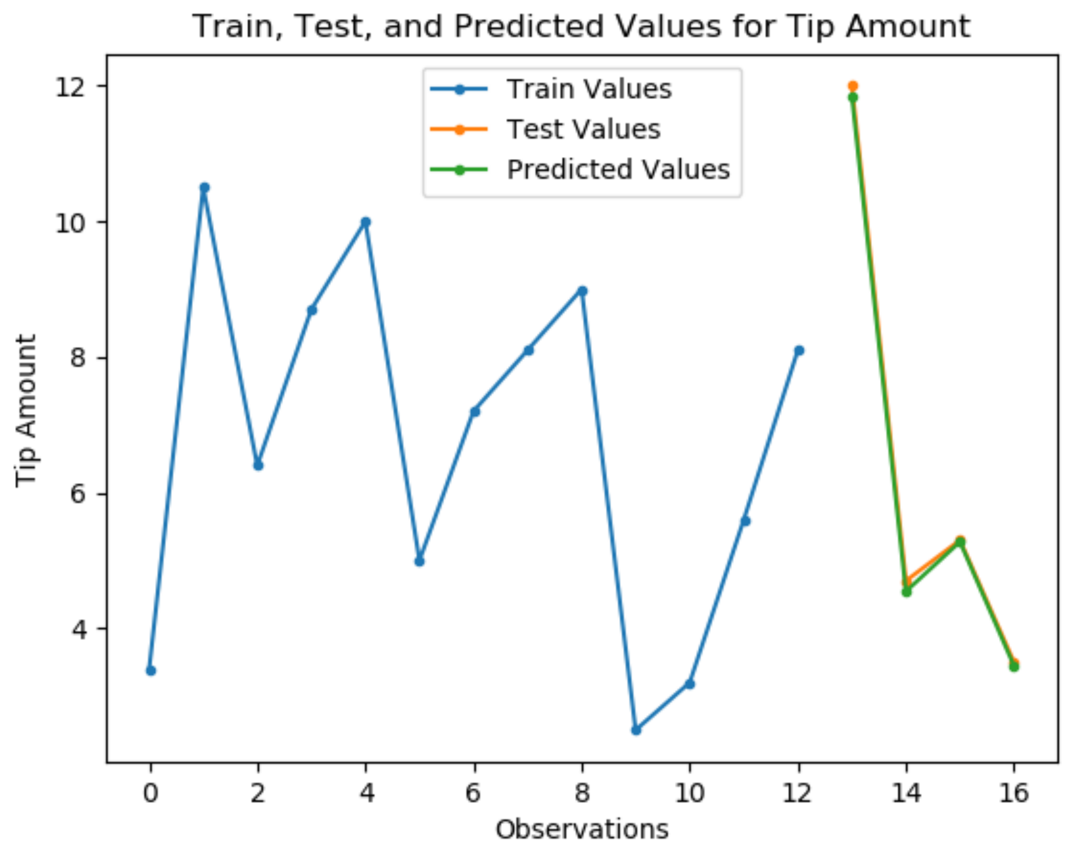
A7: The coefficients from the statsmodel ‘s OLS function and my manual calculations were exactly the same. The t values and the corresponding p-values indicate that the one parameter, Bill Amount, is statistically significant and should not be removed. The R-squared and adjusted R-square values are almost exactly the same.

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                      y   R-squared:                       0.996
Model:                            OLS   Adj. R-squared:                  0.996
Method:                 Least Squares   F-statistic:                     2914.
Date:                Mon, 24 Feb 2020   Prob (F-statistic):           1.09e-14
Time:                        17:16:52   Log-Likelihood:                 5.7379
No. Observations:                  13   AIC:                            -7.476
Df Residuals:                      11   BIC:                            -6.346
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1642      0.131      1.257      0.235      -0.123       0.452
x1             0.0973      0.002     53.985      0.000       0.093       0.101
==============================================================================
Omnibus:                        0.851   Durbin-Watson:                   2.838
Prob(Omnibus):                  0.653   Jarque-Bera (JB):                0.616
Skew:                          -0.476   Prob(JB):                        0.735
Kurtosis:                       2.521   Cond. No.                         202.
==============================================================================
```

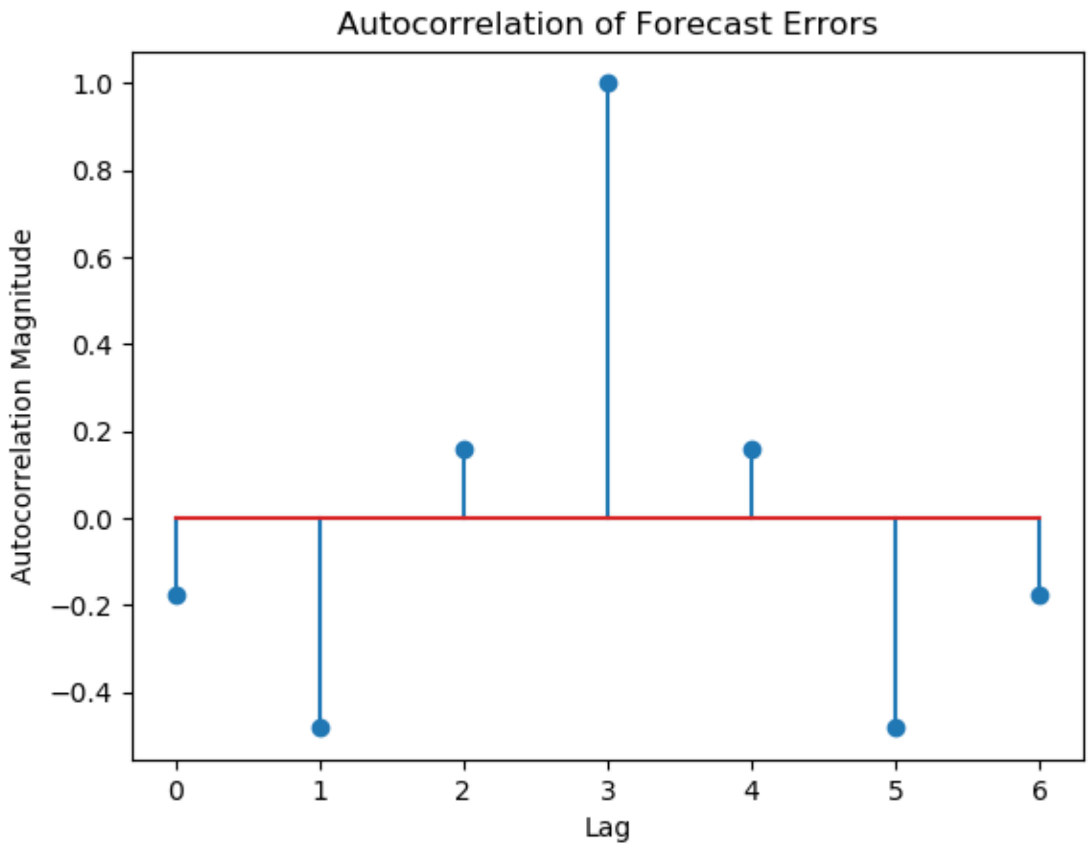


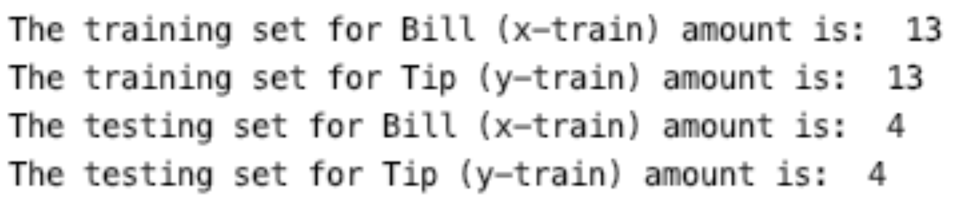
Results from Manual Linear Regression Calculation

```
The intercept for this linear regression model is:  0.164
The slope for this linear regression model is:  0.097
The R2 for this prediction is:  0.99978
The adjusted R2 for this prediction is:  0.99967
```

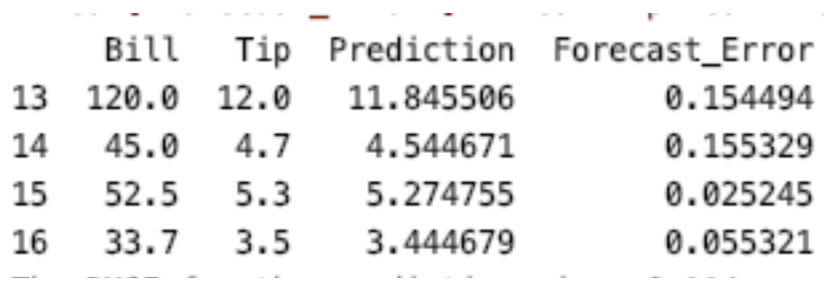


Other Graphs and Outputs From Code





```
The training set for Bill (x-train) amount is:  13
The training set for Tip (y-train) amount is:  13
The testing set for Bill (x-train) amount is:  4
The testing set for Tip (y-train) amount is:  4
```



```
     Bill   Tip  Prediction  Forecast_Error
13  120.0  12.0   11.845506        0.154494
14   45.0   4.7    4.544671        0.155329
15   52.5   5.3    5.274755        0.025245
16   33.7   3.5    3.444679        0.055321
```

Conclusion

The primary purpose of this lab was to use Linear Regression and the Least Square Estimate methods to estimate model the relationship between the bill amount and the tip amount in a given dataset. After creating the linear regression model via the least square estimate model the model had and Adjusted R-Square of 99% meaning that 99% of the variation in observed tip amounts can be explained by the relationship between tip amount and bill amount.

Appendix – Python Code

```
# Import Libraries
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
from numpy.linalg import inv
import statsmodels.api as sm

# 1 -  Using Pandas library load the time series data called Bill-Tip from the BB.
df = pd.read_csv("Bill-Tip.csv")

print(df.head())
print(df.info())

# 2 - Split the dataset into training set and test set.
```

```python
# Use 80% for training and 20% for testing. Display the training set and testing set
array as follow:
train, test = train_test_split(df, test_size=0.2, shuffle=False)
print(train.head())
print(train.info())
print(test.head())
print(test.info())

print("The training set for Bill (x-train) amount is: ", len(train['Bill']))
print("The training set for Tip (y-train) amount is: ", len(train['Tip']))
print("The testing set for Bill (x-train) amount is: ", len(test['Bill']))
print("The testing set for Tip (y-train) amount is: ", len(test['Tip']))

# 3 - Plot the scatter plot of the bill versus tip and calculate the correlation
coefficient between them.
# Display the correlation coefficient on the title as it was done in previous labs.
# Do you think that the linear regression model could be a good estimator for this
dataset? Justify your answer.

def correlation_coefficent_cal(x,y):
    n= len(x)
    x_bar = np.sum(x) / len(x)
    y_bar = np.sum(y) / len(y)
    numerator = 0
    denominator1 = 0
    denominator2 = 0
    i = 0
    while i < n:
        numerator = numerator + ((x[i] - x_bar) * (y[i] - y_bar))
        denominator1 = denominator1 + (x[i] - x_bar)*(x[i] - x_bar)
        denominator2 = denominator2 + (y[i] - y_bar)*(y[i] - y_bar)
        i=i+1
    r = numerator /(np.sqrt(denominator1*denominator2))
    return r

r_xy = np.round(correlation_coefficent_cal(df.Bill, df.Tip), decimals=3)
plt.scatter(df.Bill, df.Tip)
plt.xlabel("Bill Amount")
plt.ylabel("Tip Amount")
plt.title("Scatter plot of Bill Amount and Tip Amount with r={}".format(r_xy))
plt.show()

# 4 - Construct matrix X and Y using x-train and y-train dataset and estimate the
regression model coefficients β0,
# β1using LSE method (above equation).
# β0is called the intercept and β1is the slope of the linear regression model. Display
the message as:

X1s = np.ones(len(train.Bill))
X_transpose = np.array([X1s, np.array(train.Bill)])
X = X_transpose.transpose()
Y = np.array(train.Tip).transpose()
B_hat = inv(X_transpose.dot(X))
B_hat = B_hat.dot(X.transpose())
B_hat = B_hat.dot(Y)
print(B_hat)

print("The intercept for this linear regression model is: ", np.round(B_hat[0],
decimals=3))
print("The slope for this linear regression model is: ",np.round(B_hat[1],
decimals=3))
```

```
# 5 - Perform a prediction for the length of test-set and plot the train, test and
predicted values in one graph.
# Add appropriate x-label, y-label, title and legend to your graph.

test['Prediction'] = test.Bill*B_hat[1]+B_hat[0]
print(test.head())
fig, ax = plt.subplots()
ax.plot(train.Tip, label="Train Values", marker='.')
ax.plot(test.Tip, label="Test Values", marker='.')
ax.plot(test.Prediction, label="Predicted Values", marker='.')
ax.set_xlabel("Observations")
ax.set_ylabel("Tip Amount")
ax.set_title("Train, Test, and Predicted Values for Tip Amount")
ax.legend()
plt.show()

# 6 - Calculate the forecast errors for this prediction by comparing the test-set
values versus the predicted values.
test['Forecast_Error'] = test.Tip-test.Prediction
print(test)

# 7 - Calculate the root mean square error (RMSE) of forecast errors
# (calculated in the previous step) and display the message: RMSE =
np.sqrt(np.mean(e**2))

RSME = np.sqrt(np.mean(np.square(test.Forecast_Error)))
print("The RMSE for the predictions is: ", np.round(RSME,decimals=3))


# 8 - Calculate the mean of the forecast errors and display as a message.
# Is this a bias estimator or an unbiased estimator? Justify your answer.

Mean_Forecast_Errors = np.mean(test.Forecast_Error)
print("The Mean of the forecast errors for the predictions is: ",
np.round(Mean_Forecast_Errors,decimals=3))

# 9 - Estimate the standard error for this predictor using the following equation and
display the value as a
# message. What is your observation about the standard error?

standard_error= np.sqrt(np.sum(np.square(test.Forecast_Error))*.5)
print("The standard error of the for this predictions is: ",
np.round(standard_error,decimals=3))

# 10 - Calculate the R2 and adjusted R2 using the equation in lecture note.
# Write down your observation about R2and adjusted R2. Justify the accuracy of the
predictor model.
print(correlation_coefficent_cal(np.array(test.Tip), np.array(test.Prediction)))
R2= np.square(correlation_coefficent_cal(np.array(test.Tip),
np.array(test.Prediction)))
R2_adj = 1-((3/2)*(1-R2))

print("The R2 for this prediction is: ", np.round(R2,decimals=5))
print("The adjusted R2 for this prediction is: ", np.round(R2_adj,decimals=5))

# 11 - Graph the ACF of the forecast errors and justify the predictor accuracy using
AFC.

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
    reverselist = acf_list[::-1]
    fulllist = reverselist+acf_list[1:]
    rangey = np.arange((-len(y)+1),(len(y)))
    df=pd.DataFrame({'ACFValue':fulllist}, index=rangey)
    print(df)
    return df
ACF_Forecast_Errors = autocorrelation(np.array(test.Forecast_Error))
plt.stem(ACF_Forecast_Errors)
plt.title("Autocorrelation of Forecast Errors")
plt.ylabel("Autocorrelation Magnitude")
plt.xlabel("Lag")
plt.show()

# 12 - Plot the scatter plot between forecast errors and predictor variable and
# display the correlation coefficient between them on the title.
# Are they related? Justify the accuracy of this predictor by observing the
correlation coefficient between them.
r_xy = np.round(correlation_coefficent_cal(np.array(test.Forecast_Error),
np.array(test.Bill)), decimals=3)
plt.scatter(test.Bill, test.Forecast_Error)
plt.xlabel("Predictor Variable(Bill Amount)")
plt.ylabel("Forecast Error")
plt.title("Scatter plot of the Forecast Errors and the Predictor Variable(Bill Amount)
with r={}".format(r_xy))
plt.show()

# 13 - Plot the histogram for the forecast errors. Is it a normal distribution?
Justify your answer.
test['Forecast_Error'].hist()
plt.show()


# 14 - Plot the scatter plot between y-test and ŷt and display the correlation
coefficient between them on the title.
# Justify the accuracy of this predictor by observing the correlation coefficient
between y-test and ŷt
r_xy = np.round(correlation_coefficent_cal(np.array(test.Tip),
np.array(test.Prediction)), decimals=5)
plt.scatter(test.Tip, test.Prediction)
plt.xlabel("Test Actual Tip Value")
plt.ylabel("Predicted Tip Value")
plt.title("Scatter plot of the Actual Tips Values and the Predicted Tip with
r={}".format(r_xy))
plt.show()

# 15 - Find a 95% prediction interval for this predictor using the following equation
```

```
X_star1s = np.ones(len(test.Bill))
x_star= np.array([X_star1s, np.array(test.Bill)]).transpose()
print('x_star: ', x_star)
print(x_star.shape)

X1s = np.ones(len(train.Bill))
X_transpose = np.array([X1s, np.array(train.Bill)])
X = X_transpose.transpose()
z = inv(X_transpose.dot(X))
print(z.shape)

#test['X_star'] = x_star
print(x_star[0])
x1=x_star[0].reshape(1,2)
print(x1.shape)
print(x1)
print(test.head())

y_hat1 = np.sqrt(1+(x1.dot(z).dot(x1.transpose())))
print(y_hat1[0][0])

x2=x_star[1].reshape(1,2)
y_hat2 = standard_error*np.sqrt(1+(x2.dot(z).dot(x2.transpose())))

x3=x_star[2].reshape(1,2)
y_hat3 = standard_error*np.sqrt(1+(x3.dot(z).dot(x3.transpose())))

x4=x_star[3].reshape(1,2)
y_hat4 = standard_error*np.sqrt(1+(x4.dot(z).dot(x4.transpose())))

interval_cal = [y_hat1[0][0], y_hat2[0][0], y_hat3[0][0], y_hat4[0][0]]

test['Interval_Calculation'] = interval_cal

test["Lower 95% Interval"] = test.Prediction -
1.96*standard_error*test.Interval_Calculation
test["Upper 95% Interval"] = test.Prediction +
1.96*standard_error*test.Interval_Calculation

print(test.head())

# 16 - Calculate the Q value using the following equation:

Q_value = len(test)*np.sum(np.square(ACF_Forecast_Errors.loc[1:,'ACFValue']))
print("The Q value for this method is = ", Q_value)

# 17 -  Using the Python library “statsmodels” and OLS function, find the regression
model of above data set.
# Compare the coefficients of the derive model with one calculated in step 4.
# Display the summery of the regression model and interpret the t-value of the
regression coefficient.
# Which parameters in the model can be removed? Compare the R-squared and adjusted R-
squared in the table
# with the values calculated in step10.
# Perform a prediction using OLS function and plot the train, test and predicted
values in one graph.

#train.Bill = sm.add_constant(train.Bill)
X=np.array(train.Bill)
Y=np.array(train.Tip)
X = sm.add_constant(X)
```

```
model = sm.OLS(Y,X)
results = model.fit()
print(results.summary())

X_pred = np.array(test.Bill)
X_pred = sm.add_constant(X_pred)
pred = results.predict(X_pred)
print(pred)

test['Predict_OLS'] = pred

fig, ax = plt.subplots()
ax.plot(train.Tip, label="Train Values", marker='.')
ax.plot(test.Tip, label="Test Values", marker='.')
ax.plot(test.Predict_OLS, label="Predicted Values with StatsModel", marker='.')
ax.set_xlabel("Observations")
ax.set_ylabel("Tip Amount")
ax.set_title("Train, Test, and Predicted Values with Statsmodel OLS")
ax.legend()
plt.show()
```