**Delivery Time and Sorting Time**

**Delivery\_time -> Predict delivery time using sorting time**

Do the necessary transformations for input variables for getting better R^2 value

**Inferences from the Data Set:**

Data Set talks about the Delivery Time with respect to Sorting Time with 21 Observations

**Columns:**

Delivery Time

Sorting Time

**Data Set Size:** 21

Data give is found to be a continuous data for which a simple linear regression can be performed getting deeper into the data analysis and its behavior

**Delivery Time :**

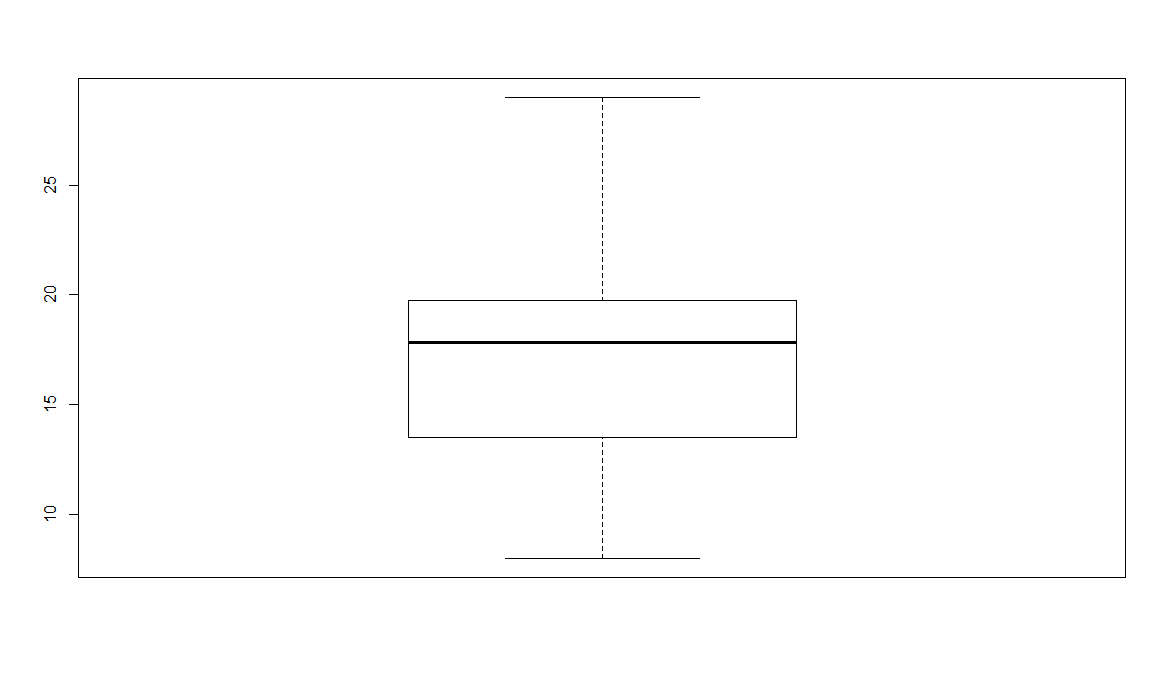
Ranges between 8 - 29

For this Delivery Time the mean is 16.79, it is just the average of the Delivery Time data

The median for the given data is 17.83, it speaks about the center of data

A comparison between mean and median tell us that data is skewed (median=17.83>mean=16.79), if data was not skewed, we would have considered mean but hear it is skewed so we take Median to talk about data.

The Data is Right Skewed, Skewness= 0.32



**Sorting Time :**

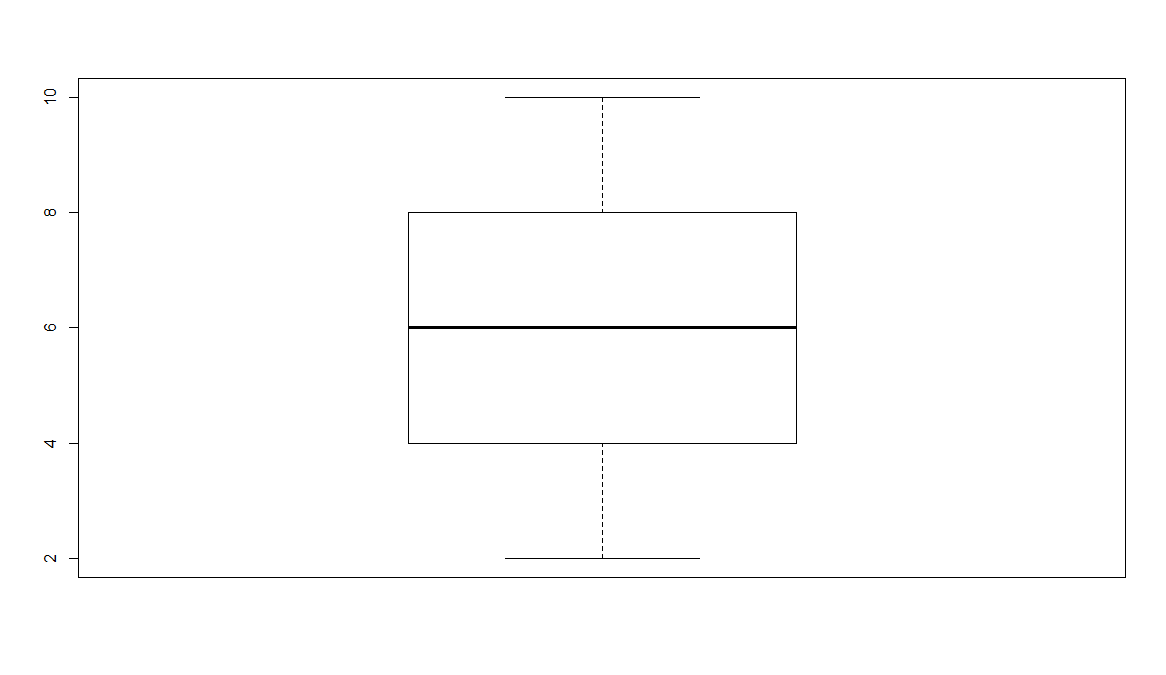
Ranges between 2 - 10

For this Sorting Time the mean is 6.19 , it is just the average of the Sorting Time data

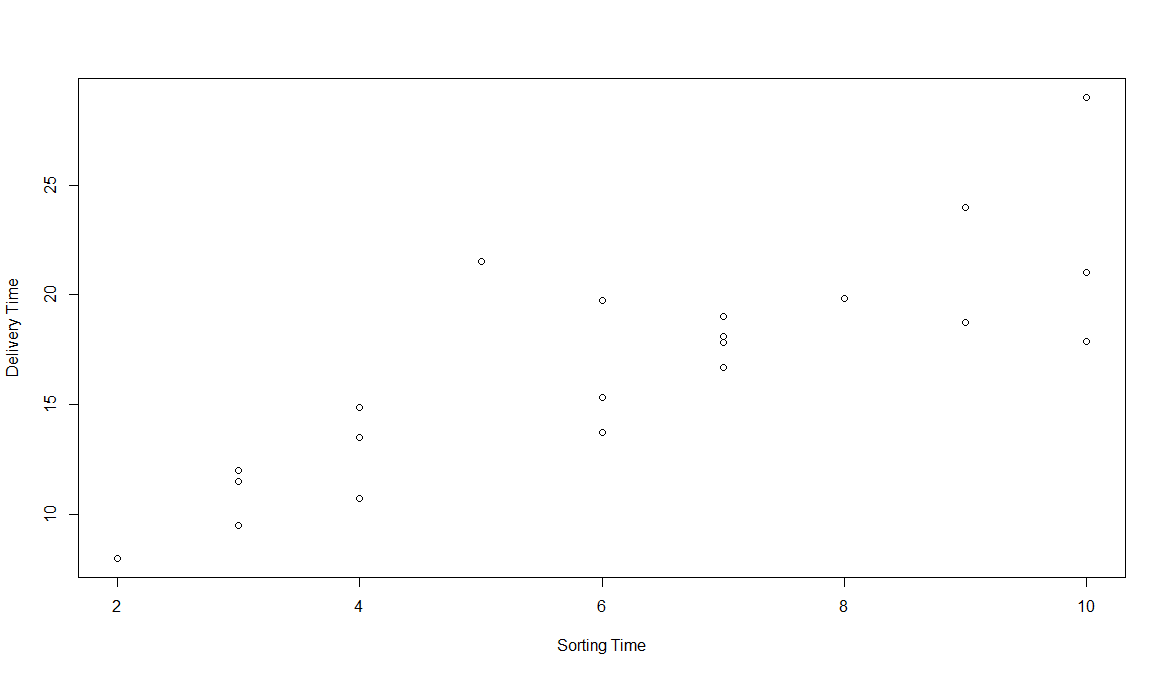
The median for the given data is 6, it speaks about the center of data

A comparison between mean and median tell us that data is skewed (median=6<mean=6.19), if data was not skewed, we would have considered mean but hear it is skewed so we take Median to talk about data.

Skewness =0.04



**Plot for Delivery Time vs Sorting Time**



The above scatter diagram infer that the Delivery Time and Sorting Time are moderately positive correlated.

**Correlation Coefficient:**

**> cor(`Sorting Time`,`Delivery Time`)**

**[1] 0.8259973**

Based on the correlation value obtained which is 0.82(approx.) also tells that it is Positive correlation

We use **lm() function from Base Package in R-Studio** to estimate the Years of experience using the other variable Salary whereas in **python LinearRegression() is used from the sklearn package**

> reg <- lm(`Delivery Time`~`Sorting Time`,data = DD\_ST)

> summary(reg)

Call:

lm(formula = `Delivery Time` ~ `Sorting Time`, data = DD\_ST)

Residuals:

Min 1Q Median 3Q Max

-5.1729 -2.0298 -0.0298 0.8741 6.6722

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 6.5827 1.7217 3.823 0.00115 \*\*

`Sorting Time` 1.6490 0.2582 6.387 3.98e-06 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.935 on 19 degrees of freedom

Multiple R-squared: 0.6823, Adjusted R-squared: 0.6655

F-statistic: 40.8 on 1 and 19 DF, p-value: 3.983e-06

**P-values:**

coefficient p-values are used to determine which terms to keep in the regression model

Look at the r-squared values are 0.68

Lets apply some transformation on the data to get a better transformation, there are different types of transformation techniques like log transformation, exponential transformation, Quadratic model..

Lets also look into the plots how they are behaving

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **output** | **input** | **cor** | **R^2** | **RMSE** | **Model** | **Plot** |
| DT | ST | 0.82 | 0.68 | 2.79 | SLR | C:\Users\RAVI\Desktop\d1 |
| DT | log(ST) | 0.83 | 0.69 | 2.73 | LT | C:\Users\RAVI\Desktop\d2 |
| log(DT) | ST | 0.84 | 0.71 | 2.94 | ET |  |
| DT | ST\*ST |  | 0.69 | 2.74 | QM-2D | C:\Users\RAVI\Desktop\d4 |
| DT | ST\*ST\*ST |  | 0.7 | 2.69 | QM-3D | C:\Users\RAVI\Desktop\d5 |
| log(DT) | log(ST) | 0.87 | 0.77 | 2.74 | LT |  |

DT= Delivery Time ST= Sorting Time ET= Exponential Transformation

QM2D=Quadratic model 2Degree QM3D=Quadratic model 3Degree

After transformation techniques obtained R-squared values are very less which is 0.77 so R^2 value should >0.8 we tell as this is not strong model

**Packages**

**R Studio**

* readr
* ggplot2
* moments

**Python**

* import pandas as pd
* import numpy as np
* import matplotlib.pyplot as plt
* from sklearn.linear\_model import LinearRegression
* import statsmodels.api as sm
* import statsmodels.formula.api as smf
* from sklearn import metrics

**CODES:**

**R code:**

**# Simple Linear Regression Assignment #**

**# 2) Delivery\_time -> Predict delivery time using sorting time**

**# Do the necessary transformations for input variables for getting better R^2 value for the model prepared.**

library(readr)

library(ggplot2)

library(moments)

DD\_ST <- read\_csv("C:/RAVI/Data science/Assignments/Module 6 Simple linear regression/DataSets/delivery\_time.csv")

View(DD\_ST)

attach(DD\_ST)

summary(DD\_ST)

range(DD\_ST$`Delivery Time`)

range(DD\_ST$`Sorting Time`)

skewness(`Delivery Time`)

skewness(`Sorting Time`)

**#Exploratory Data Analysis**

boxplot(DD\_ST$`Delivery Time`)

boxplot(DD\_ST$`Sorting Time`)

**#scatter plot for Caloriesconsumed vs Weightgained (Plot x,y)**

plot(`Sorting Time`,`Delivery Time`)

**#calculate correlation coefficient**

cor(`Sorting Time`,`Delivery Time`)

**#Simple Regression model**

reg <- lm(`Delivery Time`~`Sorting Time`,data = DD\_ST)

summary(reg)

**#values prediction**

**#Confidence interval Calculation**

confint(reg,level = 0.95)

pred <- predict(reg,interval = "predict")

**#predict function gives fit value and its lower and upeer values as a range**

pred <- as.data.frame(pred)

pred

**#####Plot Graph for both Actual values and also the predicted linear Graph(Actual:Red,Predicted:Blue)#########**

ggplot() + geom\_point(aes(x =`Sorting Time` , y =`Delivery Time` ),colour='red') +

geom\_line(aes(x = `Sorting Time`, y = predict(reg, newdata=DD\_ST)),colour='blue') +

ggtitle('Sorting Time vs Delivery Time') +xlab('Sorting Time') +ylab('Delivery Time')

cor(pred$fit,`Delivery Time`)

**#Calculate Residuals "Errors"**

reg$residuals

reg$residuals^2

mean(reg$residuals^2)

rmse <- sqrt(mean(reg$residuals^2))

rmse

**############ Applying transformations##############**

**############ lOGORITHMIC MODEL x = log(Sorting Time); y = Delivery Time ############**

plot(log(`Sorting Time`),`Delivery Time`)

cor(log(`Sorting Time`),`Delivery Time`)

log\_reg <- lm(`Delivery Time` ~ log(`Sorting Time`),data = DD\_ST)

summary(log\_reg)

**#values prediction**

**#Confidence interval Calculation**

confint(log\_reg,level = 0.95)

pred\_log <- predict(log\_reg,interval ="predict")

**#predict function gives fit value and its lower and upeer values as a range**

pred\_log <- as.data.frame(pred\_log)

pred\_log

rmse\_log <- sqrt(mean(log\_reg$residuals^2))

rmse\_log

**##########Plot Graph for both Actual values and also the predicted linear Graph(Actual:Red,Predicted:Blue)#########**

ggplot() + geom\_point(aes(x =`Sorting Time` , y =`Delivery Time` ), colour='red') +

geom\_line(aes(x =`Sorting Time`, y = predict(log\_reg, newdata=DD\_ST)), colour='blue') +

ggtitle('Sorting Time vs Delivery Time') +xlab('Sorting Time') + ylab('Delivery Time')

**############ EXPONENTIAL MODEL x = Sorting Time; y = log(Delivery Time) ############**

plot(`Sorting Time`,log(`Delivery Time`))

cor(`Sorting Time`,log(`Delivery Time`))

log\_reg2 <- lm(log(`Delivery Time`) ~ `Sorting Time`,data = DD\_ST)

summary(log\_reg2)

**#values prediction**

**#Confidence interval Calculation**

confint(log\_reg2,level = 0.95)

pred\_log2 <- predict(log\_reg2,interval ="predict")

**#predict function gives fit value and its lower and upeer values as a range**

pred\_log2 <- as.data.frame(pred\_log2)

log\_reg2$residuals #output is log(AT) so we are getting less values apply antilog

pred<- exp(pred\_log2) #anti-log=exponential

pred

cor(pred\_log2$fit,`Delivery Time`)

res\_log2=`Delivery Time`-pred$fit

rmse2 <- sqrt(mean(res\_log2^2))

rmse2

**##########Plot Graph for both Actual values and also the predicted linear Graph(Actual:Red,Predicted:Blue)#########**

ggplot() +geom\_point(aes(x =`Sorting Time` , y =`Delivery Time` ),colour='red') +

geom\_line(aes(x = `Sorting Time`, y = predict(log\_reg2, newdata=DD\_ST)),colour='blue') +

ggtitle('Sorting Time vs Delivery Time') + xlab('Sorting Time') +ylab('Delivery Time')

**############Polynomial model with 2 degree (quadratic model) ;x =Sorting Time^2 ; y = Delivery Time ############**

**#### input=x & X^2 (2-degree); output=y ####**

reg\_quad2<- lm(`Delivery Time` ~ `Sorting Time`+I(`Sorting Time`\*`Sorting Time`),data =DD\_ST)

summary(reg\_quad2)

**#prediction**

**#Confidence interval Calculation**

confint(reg\_quad2,level = 0.95)

pred\_quad2<-predict(reg\_quad2,interval = "predict")

pred\_quad2 <- as.data.frame(pred\_quad2)

pred\_quad2

resq=`Delivery Time`-pred\_quad2$fit

rmse\_quad<-sqrt(mean(resq^2))

rmse\_quad

**##########Plot Graph for both Actual values and also the predicted linear Graph(Actual:Red,Predicted:Blue)#########**

ggplot() + geom\_point(aes(x =`Sorting Time` , y =`Delivery Time` ),colour='red') +

geom\_line(aes(x = `Sorting Time`, y = predict(reg\_quad2, newdata=DD\_ST)),colour='blue') +

ggtitle('Sorting Time vs Delivery Time') + xlab('Sorting Time') +ylab('Delivery Time')

**############Polynomial model with 3 degree (quadratic model) ;x = Sorting Time^3; y = Delivery Time ############**

**#### input=x & X^2 & x^3 (3-degree); output=y ####**

reg\_quad3<- lm(`Delivery Time` ~ `Sorting Time`+I(`Sorting Time`\*`Sorting Time`)+I(`Sorting Time`\*`Sorting Time`\*`Sorting Time`),data =DD\_ST)

summary(reg\_quad3)

**#prediction**

**#Confidence interval Calculation**

confint(reg\_quad3,level = 0.95)

pred\_quad3<-predict(reg\_quad3,interval = "predict")

pred\_quad3 <- as.data.frame(pred\_quad3)

pred\_quad3

resq3=`Delivery Time`-pred\_quad3$fit

rmse\_quad3<-sqrt(mean(resq3^2))

rmse\_quad3

**##########Plot Graph for both Actual values and also the predicted linear Graph(Actual:Red,Predicted:Blue)#########**

ggplot() + geom\_point(aes(x =`Sorting Time` , y =`Delivery Time` ),colour='red') +

geom\_line(aes(x = `Sorting Time`, y = predict(reg\_quad3, newdata=DD\_ST)),colour='blue') +

ggtitle('Sorting Time vs Delivery Time') + xlab('Sorting Time') +ylab('Delivery Time')

**############ log transformation x = log(Sorting Time); y = log(Delivery Time) ############**

plot(log(`Sorting Time`),log(`Delivery Time`))

cor(log(`Sorting Time`),log(`Delivery Time`))

log\_log\_reg2 <- lm(log(`Delivery Time`) ~ log(`Sorting Time`),data = DD\_ST)

summary(log\_log\_reg2)

**#values prediction**

**#Confidence interval Calculation**

confint(log\_log\_reg2,level = 0.95)

pred\_log\_log2 <- predict(log\_log\_reg2,interval ="predict")

**#predict function gives fit value and its lower and upeer values as a range**

pred\_log\_log2 <- as.data.frame(pred\_log\_log2)

log\_log\_reg2$residuals #output is log(AT) so we are getting less values apply antilog

pred<- exp(pred\_log\_log2) #anti-log=exponential

pred

cor(pred\_log\_log2$fit,`Delivery Time`)

res\_log\_log2=`Delivery Time`-pred$fit

rmse\_log2 <- sqrt(mean(res\_log\_log2^2))

rmse\_log2

**##########Plot Graph for both Actual values and also the predicted linear Graph(Actual:Red,Predicted:Blue)#########**

ggplot() + geom\_point(aes(x =`Sorting Time` , y =`Delivery Time` ), colour='red') +

geom\_line(aes(x = `Sorting Time`, y = predict(log\_log\_reg2, newdata=DD\_ST)), colour='blue') +

ggtitle('Sorting Time vs Delivery Time') + xlab('Sorting Time') + ylab('Delivery Time')

**PYTHON:**

**# For reading data set**

**# importing necessary libraries**

import pandas as pd **# deals with data frame**

import numpy as np **# deals with numerical values**

DD\_ST = pd.read\_csv("C:/RAVI/Data science/Assignments/Module 6 Simple linear regression/DataSets/delivery\_time.csv")

DD\_ST.columns="DeliveryTime","SortingTime"

import matplotlib.pylab as plt **#for different types of plots**

plt.scatter(x=DD\_ST['SortingTime'], y=DD\_ST['DeliveryTime'],color='green'**)# Scatter plot**

np.corrcoef(DD\_ST.SortingTime,DD\_ST.DeliveryTime) **#correlation**

help(np.corrcoef)

import statsmodels.formula.api as smf

plt.hist(DD\_ST["DeliveryTime"])

model = smf.ols('DeliveryTime ~ SortingTime', data=DD\_ST).fit()

model.summary()

**#values prediction**

**#Confidence interval Calculation**

pred1 = model.predict(pd.DataFrame(DD\_ST['SortingTime']))

pred1

print (model.conf\_int(0.95)) # 95% confidence interval

res = DD\_ST.DeliveryTime - pred1

sqres = res\*res

mse = np.mean(sqres)

rmse = np.sqrt(mse)

**######### Model building on Transformed Data#############**

**# Log Transformation**

**# x = log(SortingTime); y = DeliveryTime**

plt.scatter(x=np.log(DD\_ST['SortingTime']),y=DD\_ST['DeliveryTime'],color='brown')

np.corrcoef(np.log(DD\_ST.SortingTime), DD\_ST.DeliveryTime) #correlation

model2 = smf.ols('DeliveryTime ~ np.log(SortingTime)',data=DD\_ST).fit()

model2.summary()

pred2 = model2.predict(pd.DataFrame(DD\_ST['SortingTime']))

pred2

print(model2.conf\_int(0.95)) # 95% confidence level

res2 = DD\_ST.DeliveryTime - pred2

sqres2 = res2\*res2

mse2 = np.mean(sqres2)

rmse2 = np.sqrt(mse2)

**# Exponential transformation**

plt.scatter(x=DD\_ST['SortingTime'], y=np.log(DD\_ST['DeliveryTime']),color='orange')

np.corrcoef(DD\_ST.SortingTime, np.log(DD\_ST.DeliveryTime)) **#correlation**

model3 = smf.ols('np.log(DeliveryTime) ~ SortingTime',data=DD\_ST).fit()

model3.summary()

pred\_log = model3.predict(pd.DataFrame(DD\_ST['SortingTime']))

pred\_log

pred3 = np.exp(pred\_log)

pred3

print(model3.conf\_int(0.95)) **# 95% confidence level**

res3 = DD\_ST.DeliveryTime - pred3

sqres3 = res3\*res3

mse3 = np.mean(sqres3)

rmse3 = np.sqrt(mse3)

**############Polynomial model with 2 degree (quadratic model) ;x = SortingTime\*SortingTime; y = DeliveryTime############**

**#### input=x & X^2 (2-degree); output=y ####**

model4 = smf.ols('DeliveryTime ~ SortingTime+I(SortingTime\*SortingTime)', data=DD\_ST).fit()

model4.summary()

pred\_p2 = model4.predict(pd.DataFrame(DD\_ST['SortingTime']))

pred\_p2

print(model3.conf\_int(0.95)) **# 95% confidence level**

res4 = DD\_ST.DeliveryTime - pred\_p2

sqres4 = res4\*res4

mse4 = np.mean(sqres4)

rmse4 = np.sqrt(mse4)

**###########Polynomial model with 3 degree (quadratic model) ;x = SortingTime\*SortingTime\*SortingTime; y = DeliveryTime############**

**#### input=x & X^2 (2-degree); output=y ####**

model5 = smf.ols('DeliveryTime ~ SortingTime+I(SortingTime\*SortingTime)+I(SortingTime\*SortingTime\*SortingTime)', data=DD\_ST).fit()

model5.summary()

pred\_p3 = model5.predict(pd.DataFrame(DD\_ST['SortingTime']))

pred\_p3

print(model5.conf\_int(0.95)) **# 95% confidence level**

res5 = DD\_ST.DeliveryTime - pred\_p3

sqres5 = res5\*res5

mse5 = np.mean(sqres5)

rmse5 = np.sqrt(mse5)

**# Log Transformation**

**# x = log(SortingTime); y = log(DeliveryTime)**

plt.scatter(x=np.log(DD\_ST['SortingTime']),y=np.log(DD\_ST['DeliveryTime']),color='brown')

np.corrcoef(np.log(DD\_ST.SortingTime), np.log(DD\_ST.DeliveryTime)) #correlation

model6 = smf.ols('np.log(DeliveryTime) ~ np.log(SortingTime)',data=DD\_ST).fit()

model6.summary()

pred\_log6 = model6.predict(pd.DataFrame(DD\_ST['SortingTime']))

pred\_log6

pred6 = np.exp(pred\_log6)

pred6

print(model6.conf\_int(0.95)) **# 95% confidence level**

res6 = DD\_ST.DeliveryTime - pred6

sqres6 = res6\*res6

mse6 = np.mean(sqres6)

rmse6 = np.sqrt(mse6)