Predict delivery time using sorting time

```
import pandas as pd
import seaborn as sns
import statsmodels.formula.api as smf

df=pd.read_csv('C:/Users/RIG1/Desktop/DS ASSIGNMENTS/QUESTIONS -all 
assignments/ASS 4/delivery_time.csv')
df.head()
```

```
   Delivery Time  Sorting Time
0          21.00            10
1          13.50             4
2          19.75             6
3          24.00             9
4          29.00            10
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Delivery Time  21 non-null     float64
 1   Sorting Time   21 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 464.0 bytes
```

```
df.describe()
```

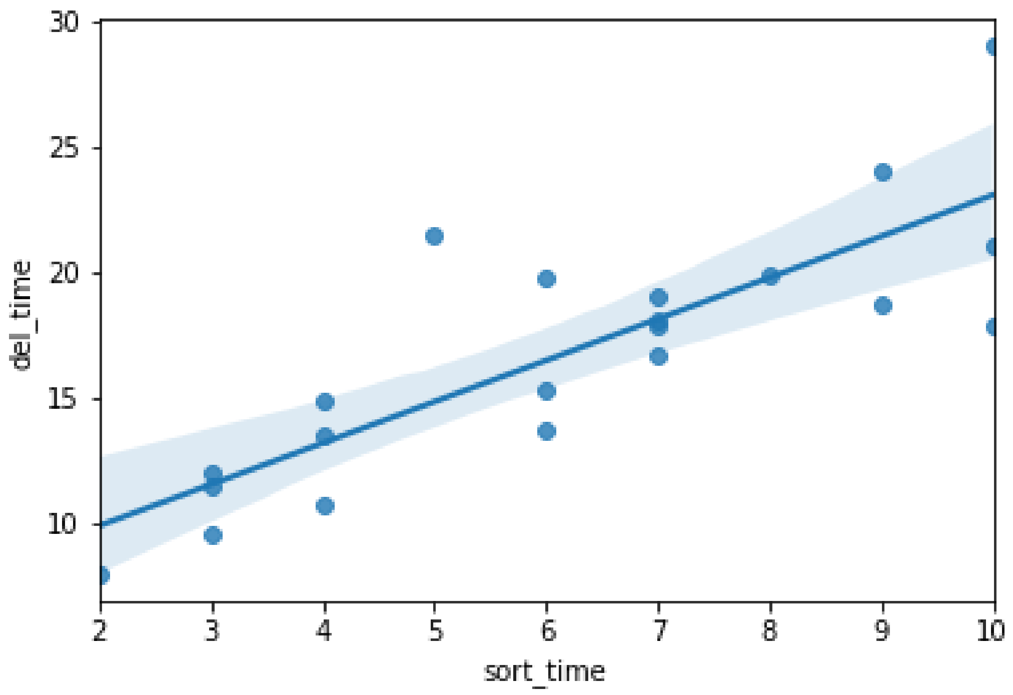
```
       Delivery Time  Sorting Time
count      21.000000     21.000000
mean       16.790952      6.190476
std         5.074901      2.542028
min         8.000000      2.000000
25%        13.500000      4.000000
50%        17.830000      6.000000
75%        19.750000      8.000000
max        29.000000     10.000000
```

```
df.corr()
```

```
               Delivery Time  Sorting Time
Delivery Time       1.000000      0.825997
Sorting Time        0.825997      1.000000
```

```
df.dtypes
```

```
Delivery Time    float64
Sorting Time       int64
dtype: object
```

rename both the column name into small one
```
df=df.rename({'Delivery Time': 'del_time' , 'Sorting
Time' :'sort_time'},axis=1)
df.head()
```
```
   del_time  sort_time
0     21.00         10
1     13.50          4
2     19.75          6
3     24.00          9
4     29.00         10
```
Plotting regression plot
```
sns.regplot(x='sort_time',y='del_time',data=df)
```
```
<AxesSubplot:xlabel='sort_time', ylabel='del_time'>
```


now in order to create our regression equation/model we will import our liberary statsmodels.formula.api.

Y=B0+B1*X

del_time=B0+B1*sort_time
```
model=smf.ols('del_time~sort_time',data=df).fit()
```
```
###ESTIMATED VALUES
model.params
```

```
Intercept    6.582734
sort_time    1.649020
dtype: float64
```

```
#Intercept    6.582734----------B0
#sort_time    1.649020----------B1 -------(slope)
```

```
# PVALUE AND TVALUE
print(model.pvalues,'\n',model.tvalues)
```

```
Intercept    0.001147
sort_time    0.000004
dtype: float64
 Intercept    3.823349
sort_time    6.387447
dtype: float64
```

```
#B1= 1.649020(slope)
#B0= 6.582734(intercept)
```

If the above p value or B0 or sort_time--------0.000004 <=alpha(0.05)============(then,reject H0)=======(which means y depends on x(del_time depends on sort_time))

Then it means, the sorting-time is important for us to predict the delivery-time

if p value>0.05=====it means sort_time is not important for del_time parameters========then,reject H1======which means y do not depends on x

```
# now compute rsquare for validation purpose to check whether the
equation we got is good or bad
```

```
model.rsquared
```

```
0.6822714748417231
```

now its time to predict the delivery time

here is the Automatic Prediction if lets say sorting time is 12, 8

```
delivery_time=pd.Series([12,8])
delivery_time
```

```
0    12
1     8
dtype: int64
```

```
prediction=pd.DataFrame(delivery_time,columns=['sort_time'])
```

```
model.predict(prediction)
```

```
0    26.370973
1    19.774893
dtype: float64
```

Manual prediction for same values- sorting time 12

```
delivery_time = (6.582734) + (1.649020)*(12)
delivery_time
```

```
26.370973999999997
```