Predict delivery time

October 16, 2021

1 To Predict delivery time using sorting time

1.0.1 Importing the Libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
from sklearn.model_selection import train_test_split
### Splitting the dataset into the Training set and test set
```

[3]:
```
from sklearn.linear_model import LinearRegression
## Training the simple linear Regression Model
```

[4]:
```
import statsmodels.api as sm
```

1.0.2 Importing the dataset

[5]:
```
data_sets = pd.read_csv("delivery_time.csv")
```

[6]:
```
x = data_sets.iloc[:,:-1].values
y = data_sets.iloc[:, -1].values
```

[7]:
```
print(data_sets)
```

```
    Delivery_Time  Sorting_Time
0           21.00            10
1           13.50             4
2           19.75             6
3           24.00             9
4           29.00            10
5           15.35             6
6           19.00             7
7            9.50             3
8           17.90            10
9           18.75             9
10          19.83             8
11          10.75             4
12          16.68             7
```

```
13          11.50             3
14          12.03             3
15          14.88             4
16          13.75             6
17          18.11             7
18           8.00             2
19          17.83             7
20          21.50             5
```

[8]:
```
print(x)
```

```
[[21.  ]
 [13.5 ]
 [19.75]
 [24.  ]
 [29.  ]
 [15.35]
 [19.  ]
 [ 9.5 ]
 [17.9 ]
 [18.75]
 [19.83]
 [10.75]
 [16.68]
 [11.5 ]
 [12.03]
 [14.88]
 [13.75]
 [18.11]
 [ 8.  ]
 [17.83]
 [21.5 ]]
```

[9]:
```
print(y)
```

```
[10  4  6  9 10  6  7  3 10  9  8  4  7  3  3  4  6  7  2  7  5]
```

1.0.3 Splitting the dataset into the Training set and test set

[10]:
```
data_sets.describe()
```

[10]:

	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
```

[11]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,␣
↪random_state= 0)
```

1.0.4 Training the simple linear Regression Model on the Training set

[12]:
```
regressor = LinearRegression()
regressor.fit(x_train,y_train)
```

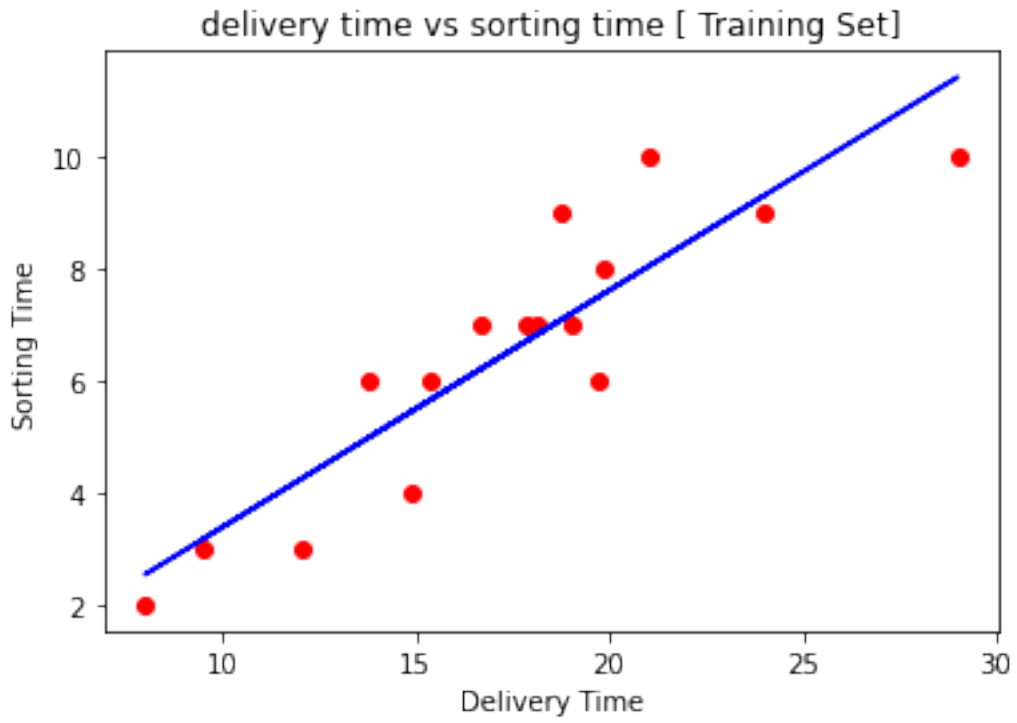
[12]:
```
LinearRegression()
```

1.0.5 Predicting the Test set result

[13]:
```
y_predict = regressor.predict(x_test)
```

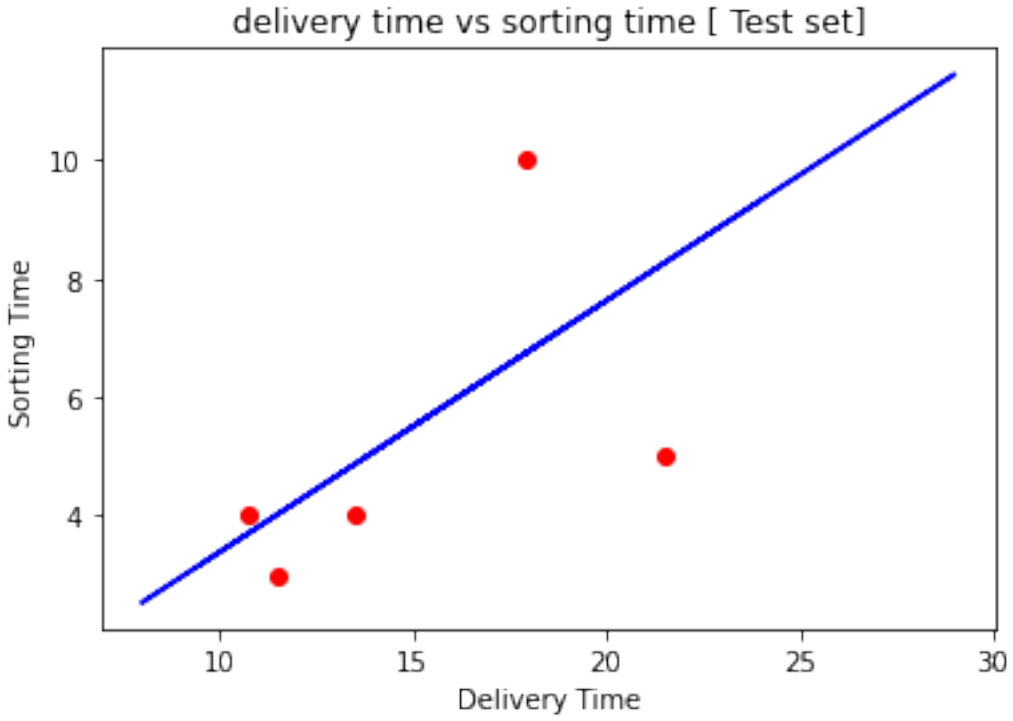
1.0.6 Visualising the Training set results

[14]:
```
plt.scatter(x_train, y_train, color = 'red')
plt.plot(x_train, regressor.predict(x_train), color ="blue")
plt.title("delivery time vs sorting time [ Training Set]")
plt.xlabel("Delivery Time")
plt.ylabel("Sorting Time")
plt.show()
```



1.0.7 Visualising the test set result

[15]:
```
plt.scatter(x_test, y_test, color = 'red')
plt.plot(x_train, regressor.predict(x_train), color ="blue")
plt.title("delivery time vs sorting time [ Test set]")
plt.xlabel("Delivery Time")
plt.ylabel("Sorting Time")
plt.show()
```



1.0.8 Regression Itself

[16]:
```
x_stats = sm.add_constant(x)
results = sm.OLS(y,x_stats).fit()
results.summary()
```

[16]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.682
```

```
Model:                            OLS   Adj. R-squared:                  0.666
Method:                 Least Squares   F-statistic:                     40.80
Date:                Sat, 16 Oct 2021   Prob (F-statistic):           3.98e-06
Time:                        15:35:31   Log-Likelihood:                -36.839
No. Observations:                  21   AIC:                             77.68
Df Residuals:                      19   BIC:                             79.77
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.7567      1.134     -0.667      0.513      -3.130       1.617
x1             0.4137      0.065      6.387      0.000       0.278       0.549
==============================================================================
Omnibus:                        1.409   Durbin-Watson:                   1.346
Prob(Omnibus):                  0.494   Jarque-Bera (JB):                0.371
Skew:                           0.255   Prob(JB):                        0.831
Kurtosis:                       3.405   Cond. No.                         62.1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```