Delivery_time -> Predict delivery time using sorting time

1. Import libraries

In [1]:

```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

2. Import Data

In [2]:

```
delivery_time = pd.read_csv('C:/Users/Ravi Kiran/Simple Linear Regression/delivery_time
delivery_time
```

Out[2]:

	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

3. Data Understanding

Perform Initial Analysis

In [3]:

```
delivery_time.shape
```

Out[3]:

```
(21, 2)
```

In [4]:

```
delivery_time.isna().sum()
```

Out[4]:

```
Delivery Time    0
Sorting Time     0
dtype: int64
```

In [5]:

```
delivery_time.dtypes
```

Out[5]:

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

In [6]:

```
delivery_time.describe()
```

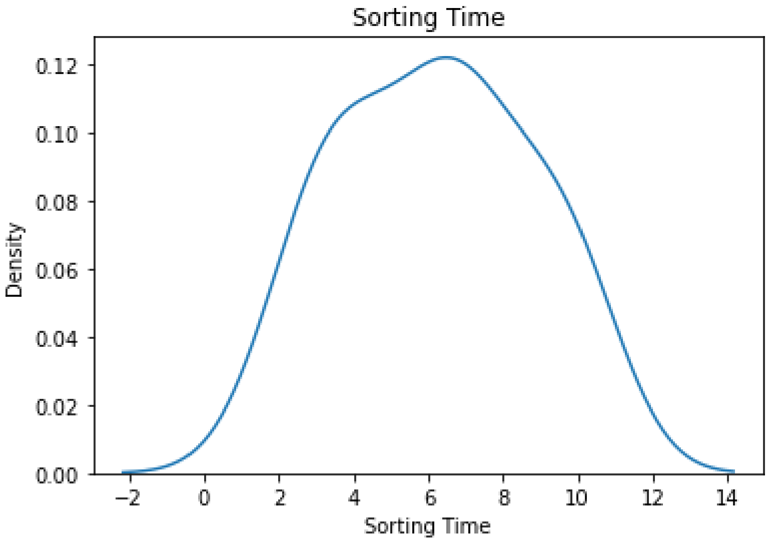
Out[6]:

	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

Normality Test

In [7]:

```
sns.distplot(a=delivery_time['Sorting Time'],hist=False)
plt.title('Sorting Time')
plt.show()
```



In [8]:

```
delivery_time['Sorting Time'].skew()
```

Out[8]:

```
0.047115474210530174
```

In [9]:

```
delivery_time['Sorting Time'].kurtosis()
```

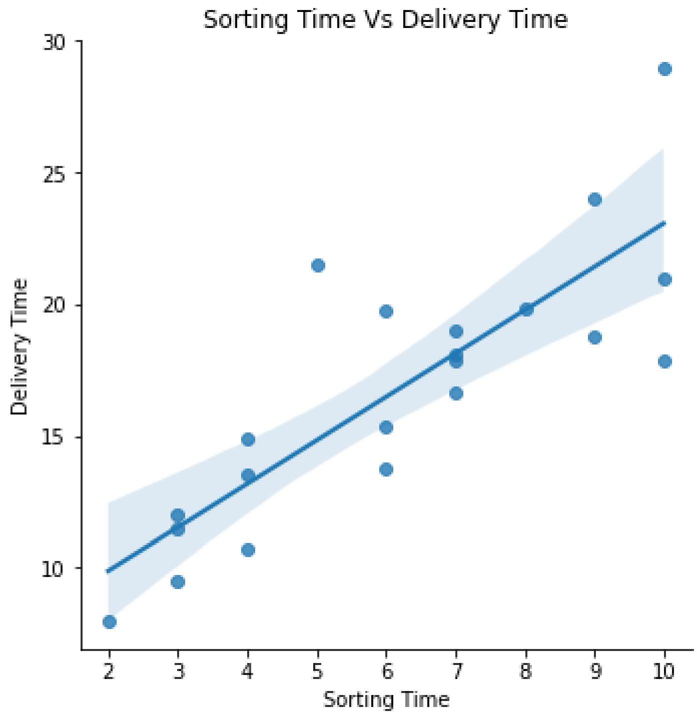
Out[9]:

```
-1.14845514534878
```

Linearity Test

In [10]:

```
sns.lmplot(x='Sorting Time',y='Delivery Time',data=delivery_time)
plt.title('Sorting Time Vs Delivery Time')
plt.show()
```



In [11]:

```
delivery_time.corr()
```

Out[11]:

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

4.Data Preparation

In [12]:

```
x = delivery_time[['Sorting Time']]
y = delivery_time[['Delivery Time']]
```

In [13]:

```
x
```

Out[13]:

	Sorting Time
0	10
1	4
2	6
3	9
4	10
5	6
6	7
7	3
8	10
9	9
10	8
11	4
12	7
13	3
14	3
15	4
16	6
17	7
18	2
19	7
20	5

In [14]:

```
y
```

Out[14]:

	Delivery Time
0	21.00
1	13.50
2	19.75
3	24.00
4	29.00
5	15.35
6	19.00
7	9.50
8	17.90
9	18.75
10	19.83
11	10.75
12	16.68
13	11.50
14	12.03
15	14.88
16	13.75
17	18.11
18	8.00
19	17.83
20	21.50

5.Model Building

In [15]:

```
from sklearn.linear_model import LinearRegression
```

In [16]:

```
time_model = LinearRegression()
```

In [17]:

```
time_model.fit(x,y)
```

Out[17]:

```
LinearRegression()
```

6.Model Prediction

In [18]:

```
y_pred = time_model.predict(x)
y_pred
```

Out[18]:

```
array([[23.07293294],
       [13.17881356],
       [16.47685335],
       [21.42391304],
       [23.07293294],
       [16.47685335],
       [18.12587325],
       [11.52979366],
       [23.07293294],
       [21.42391304],
       [19.77489315],
       [13.17881356],
       [18.12587325],
       [11.52979366],
       [11.52979366],
       [13.17881356],
       [16.47685335],
       [18.12587325],
       [ 9.88077377],
       [18.12587325],
       [14.82783346]])
```

In [20]:

```
error = y - y_pred
error
```

Out[20]:

	Delivery Time
0	-2.072933
1	0.321186
2	3.273147
3	2.576087
4	5.927067
5	-1.126853
6	0.874127
7	-2.029794
8	-5.172933
9	-2.673913
10	0.055107
11	-2.428814
12	-1.445873
13	-0.029794
14	0.500206
15	1.701186
16	-2.726853
17	-0.015873
18	-1.880774
19	-0.295873
20	6.672167

7. Model Evaluation

In [23]:

```
test_data = pd.DataFrame(data={'Sort time':[5,11,13]})
test_data
```

Out[23]:

	Sort time
0	5
1	11
2	13

In [22]:

```
time_model.predict(test_data)
```

Out[22]:

```
array([[14.82783346],
       [24.72195284],
       [28.01999263]])
```

==

Salary_hike -> Build a prediction model for Salary_hike

Import data

In [48]:

```
Salary_data = pd.read_csv('C:/Users/Ravi Kiran/Simple Linear Regression/Salary_Data.csv
Salary_data
```

Out[48]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

Data Understanding

Perform Initial Analysis

In [27]:

```
Salary_data.dtypes
```

Out[27]:

```
YearsExperience    float64
Salary             float64
dtype: object
```

In [28]:

```
Salary_data.shape
```

Out[28]:

```
(30, 2)
```

In [29]:

```
Salary_data.describe()
```

Out[29]:

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

In [7]:

```
Salary_data.isna().sum()
```

Out[7]:

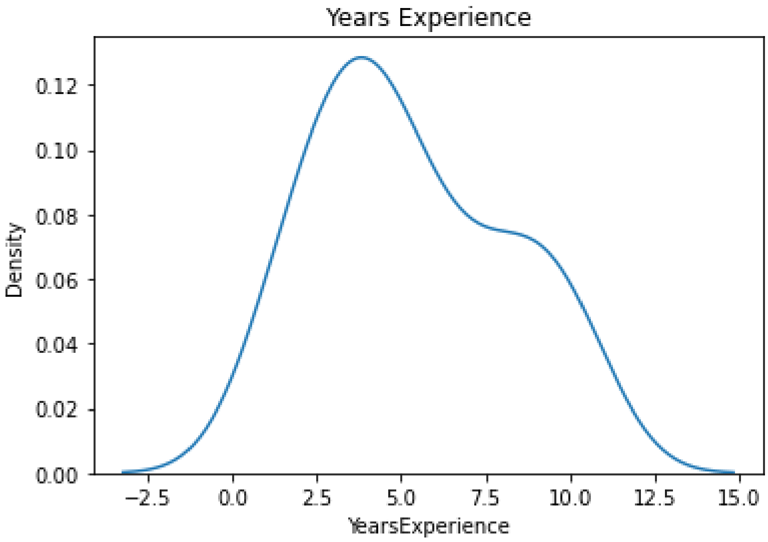
```
YearsExperience    0
Salary             0
dtype: int64
```

Assumptions Check

Normality Test

In [30]:

```
sns.distplot(a=Salary_data['YearsExperience'],hist=False)
plt.title('Years Experience')
plt.show()
```



In [32]:

```
Salary_data['YearsExperience'].skew()
```

Out[32]:

```
0.37956024064804106
```

In [33]:

```
Salary_data['YearsExperience'].kurtosis()
```

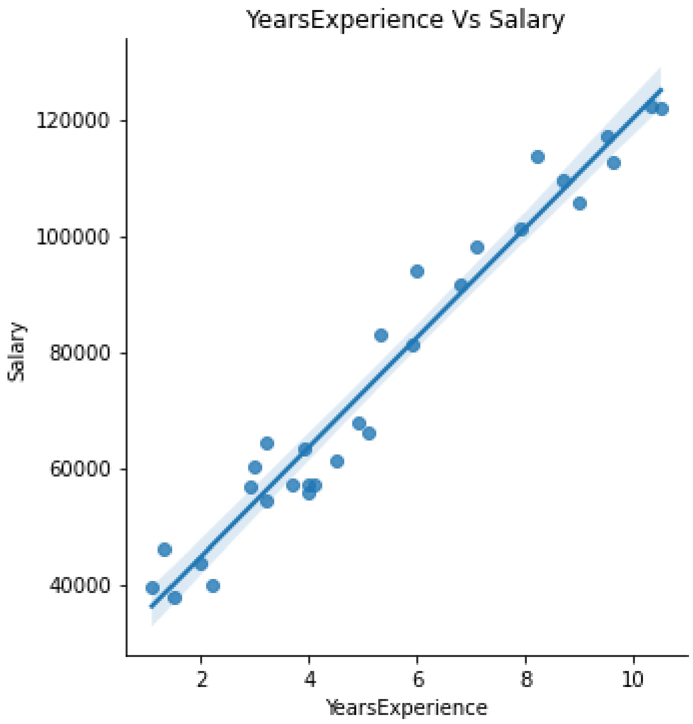
Out[33]:

```
-1.0122119403325072
```

Linearity Test

In [34]:

```
sns.lmplot(x='YearsExperience',y='Salary',data=Salary_data)
plt.title('YearsExperience Vs Salary')
plt.show()
```



In [35]:

```
Salary_data.corr()
```

Out[35]:

	YearsExperience	Salary
YearsExperience	1.000000	0.978242
Salary	0.978242	1.000000

Data Preparation

In [36]:

```
x = Salary_data[['YearsExperience']]
y = Salary_data[['Salary']]
```

In [37]:

```
x
```

Out[37]:

	YearsExperience
0	1.1
1	1.3
2	1.5
3	2.0
4	2.2
5	2.9
6	3.0
7	3.2
8	3.2
9	3.7
10	3.9
11	4.0
12	4.0
13	4.1
14	4.5
15	4.9
16	5.1
17	5.3
18	5.9
19	6.0
20	6.8
21	7.1
22	7.9
23	8.2
24	8.7
25	9.0
26	9.5
27	9.6
28	10.3
29	10.5

In [38]:

```
y
```

Out[38]:

	Salary
0	39343.0
1	46205.0
2	37731.0
3	43525.0
4	39891.0
5	56642.0
6	60150.0
7	54445.0
8	64445.0
9	57189.0
10	63218.0
11	55794.0
12	56957.0
13	57081.0
14	61111.0
15	67938.0
16	66029.0
17	83088.0
18	81363.0
19	93940.0
20	91738.0
21	98273.0
22	101302.0
23	113812.0
24	109431.0
25	105582.0
26	116969.0
27	112635.0
28	122391.0
29	121872.0

Model Building

In [39]:

```
Salary_model = LinearRegression()
```

In [40]:

```
Salary_model.fit(x,y)
```

Out[40]:

```
LinearRegression()
```

Model Prediction

In [44]:

```
y_pred = Salary_model.predict(x)
y_pred
```

Out[44]:

```
array([[ 36187.15875227],
       [ 38077.15121656],
       [ 39967.14368085],
       [ 44692.12484158],
       [ 46582.11730587],
       [ 53197.09093089],
       [ 54142.08716303],
       [ 56032.07962732],
       [ 56032.07962732],
       [ 60757.06078805],
       [ 62647.05325234],
       [ 63592.04948449],
       [ 63592.04948449],
       [ 64537.04571663],
       [ 68317.03064522],
       [ 72097.0155738 ],
       [ 73987.00803809],
       [ 75877.00050238],
       [ 81546.97789525],
       [ 82491.9741274 ],
       [ 90051.94398456],
       [ 92886.932681  ],
       [100446.90253816],
       [103281.8912346 ],
       [108006.87239533],
       [110841.86109176],
       [115566.84225249],
       [116511.83848464],
       [123126.81210966],
       [125016.80457395]])
```

In [45]:

```
error = y - y_pred
error
```

Out[45]:

	Salary
0	3155.841248
1	8127.848783
2	-2236.143681
3	-1167.124842
4	-6691.117306
5	3444.909069
6	6007.912837
7	-1587.079627
8	8412.920373
9	-3568.060788
10	570.946748
11	-7798.049484
12	-6635.049484
13	-7456.045717
14	-7206.030645
15	-4159.015574
16	-7958.008038
17	7210.999498
18	-183.977895
19	11448.025873
20	1686.056015
21	5386.067319
22	855.097462
23	10530.108765
24	1424.127605
25	-5259.861092
26	1402.157748
27	-3876.838485
28	-735.812110
29	-3144.804574

Model Evaluation

In [46]:

```
test_data = pd.DataFrame(data = {'Years' : [5,1,0]})
test_data
```

Out[46]:

	Years
0	5
1	1
2	0

In [47]:

```
Salary_model.predict(test_data)
```

Out[47]:

```
array([[73042.01180594],
       [35242.16252012],
       [25792.20019867]])
```

In [ ]:

```

```