Assignment-04-Simple Linear Regression-1

1) Delivery_time -> Predict delivery time using sorting time

Build a simple linear regression model by performing EDA and do necessary transformations and select the best model using R or Python.

In [1]:

```
# import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import statsmodels.formula.api as smf
```

In [2]:

```
dataset=pd.read_csv("C:/Users/LENOVO/Documents/Custom Office Templates/delivery_time.csv")
dataset
```

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

EDA and Data Visualization

In [3]:

```
dataset.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
```

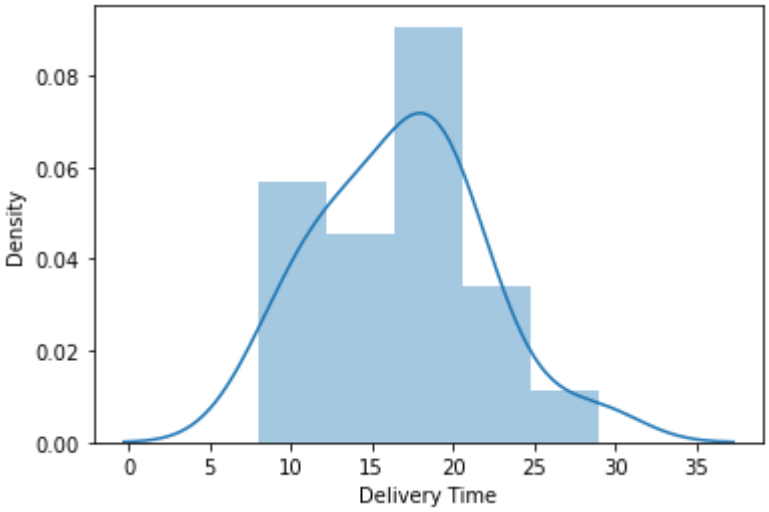
In [4]:

```
sns.distplot(dataset['Delivery Time'])
```

```
C:\Users\LENOVO\anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a f
uture version. Please adapt your code to use either `displot` (a figure-leve
l function with similar flexibility) or `histplot` (an axes-level function f
or histograms).
  warnings.warn(msg, FutureWarning)
```

Out[4]:

```
<AxesSubplot:xlabel='Delivery Time', ylabel='Density'>
```



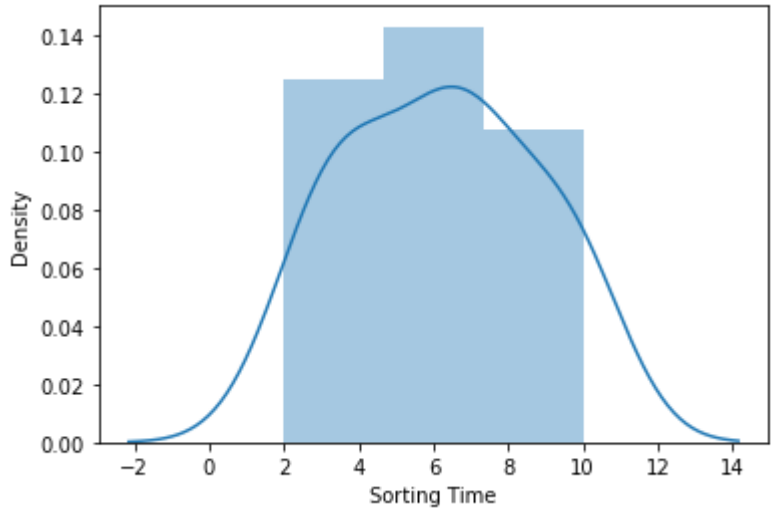
In [5]:

```
sns.distplot(dataset['Sorting Time'])
```

```
C:\Users\LENOVO\anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a f
uture version. Please adapt your code to use either `displot` (a figure-leve
l function with similar flexibility) or `histplot` (an axes-level function f
or histograms).
  warnings.warn(msg, FutureWarning)
```

Out[5]:

```
<AxesSubplot:xlabel='Sorting Time', ylabel='Density'>
```



Feature Engineering

In [9]:

```
# Renaming Coluimns
dataset=dataset.rename({'Delivery Time':'delivery_time','Sorting Time':'sorting_time'},axis
dataset
```

Out[9]:

	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

Correlation Analysis

In [10]:

```
dataset.corr()
```

Out[10]:

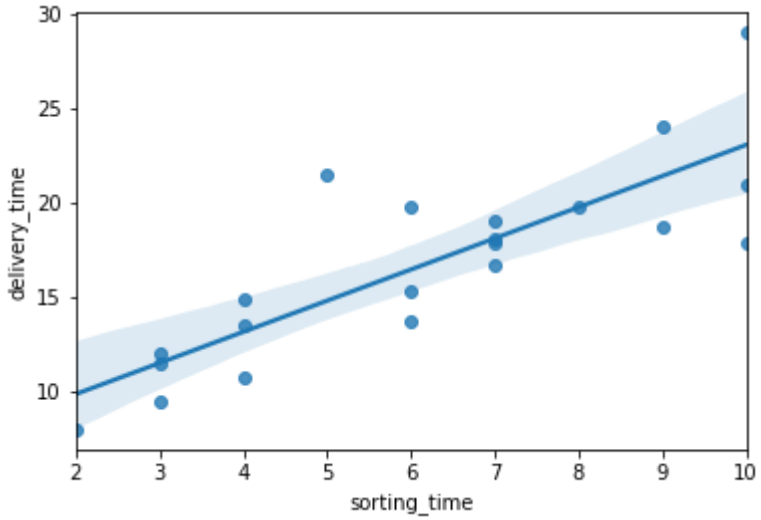
	delivery_time	sorting_time
delivery_time	1.000000	0.825997
sorting_time	0.825997	1.000000

In [11]:

```
sns.regplot(x=dataset['sorting_time'],y=dataset['delivery_time'])
```

Out[11]:

```
<AxesSubplot:xlabel='sorting_time', ylabel='delivery_time'>
```



Model Building

In [12]:

```
model=smf.ols("delivery_time~sorting_time",data=dataset).fit()
```

Model Testing

In [13]:

```
# Finding the coefficient parameters
model.params
```

Out[13]:

```
Intercept       6.582734
sorting_time    1.649020
dtype: float64
```

In [14]:

```
# Finding the tvalues and pvalues
model.tvalues,model.pvalues
```

Out[14]:

```
(Intercept       3.823349
 sorting_time    6.387447
 dtype: float64,
 Intercept       0.001147
 sorting_time    0.000004
 dtype: float64)
```

In [15]:

```
# Finding Rsquared Values
model.rsquared, model.rsquared_adj
```

Out[15]:

```
(0.6822714748417231, 0.6655489208860244)
```

Model Predictions

In [16]:

```
# Manual prediction for say sorting time 5
delivery_time=(6.582734)+(1.649020)*(5)
delivery_time
```

Out[16]:

```
14.827834
```

In [17]:

```
# Automatic Prediction for say sorting_time 5,8
new_data=pd.Series([5,8])
new_data
```

Out[17]:

```
0    5
1    8
dtype: int64
```

In [19]:

```
data_pred=pd.DataFrame(new_data,columns=['sorting_time'])
data_pred
```

Out[19]:

	sorting_time
0	5
1	8

In [20]:

```
model.predict(data_pred)
```

Out[20]:

```
0    14.827833
1    19.774893
dtype: float64
```

In [ ]:

```

```

Assignment-04-Simple Linear Regression-2

2) Salary_hike -> Build a prediction model for Salary_hike

In [21]:

```
dataset1=pd.read_csv("C:/Users/LENOVO/Documents/Custom Office Templates/Salary_Data.csv")
dataset1
```

Out[21]:

	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

EDA and Visualization

In [22]:

```
dataset1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

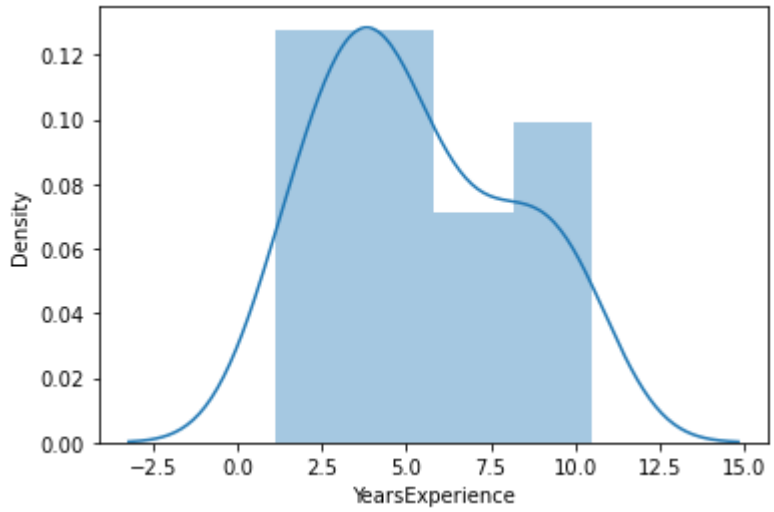
In [23]:

```
sns.distplot(dataset1['YearsExperience'])
```

```
C:\Users\LENOVO\anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a f
uture version. Please adapt your code to use either `displot` (a figure-leve
l function with similar flexibility) or `histplot` (an axes-level function f
or histograms).
  warnings.warn(msg, FutureWarning)
```

Out[23]:

```
<AxesSubplot:xlabel='YearsExperience', ylabel='Density'>
```



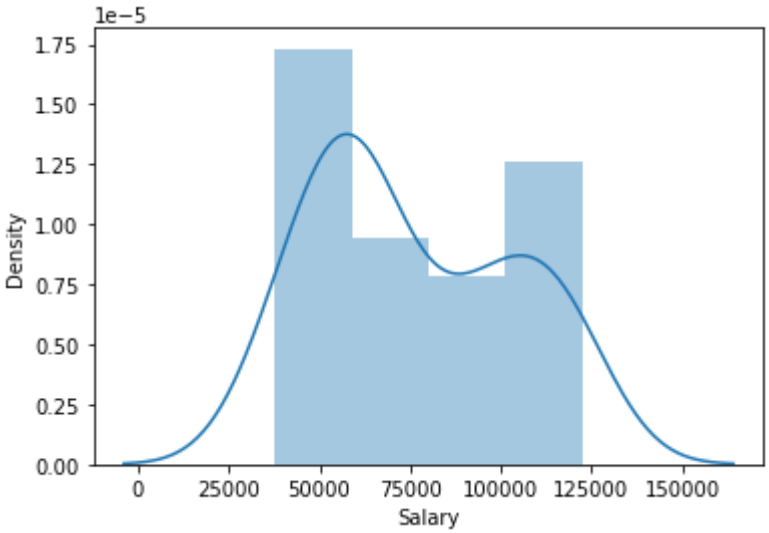
In [24]:

```
sns.distplot(dataset1['Salary'])
```

```
C:\Users\LENOVO\anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a f
uture version. Please adapt your code to use either `displot` (a figure-leve
l function with similar flexibility) or `histplot` (an axes-level function f
or histograms).
  warnings.warn(msg, FutureWarning)
```

Out[24]:

```
<AxesSubplot:xlabel='Salary', ylabel='Density'>
```



Correlation Analysis

In [25]:

```
dataset1.corr()
```

Out[25]:

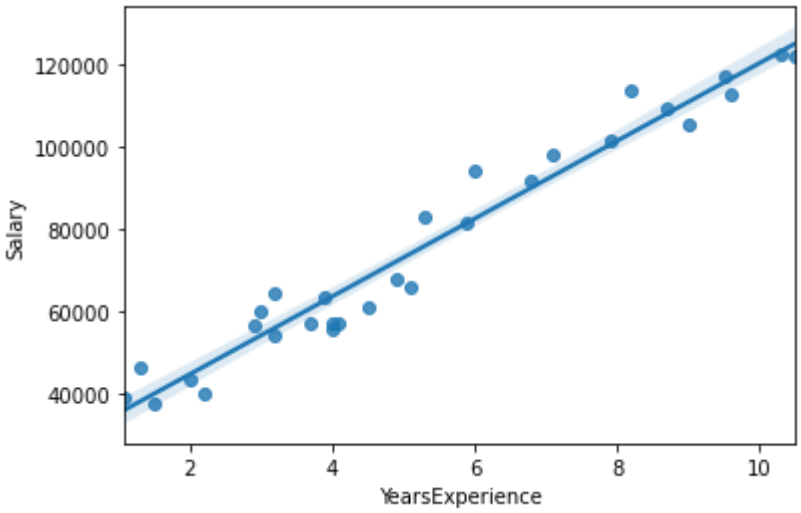
	YearsExperience	Salary
YearsExperience	1.000000	0.978242
Salary	0.978242	1.000000

In [27]:

```
sns.regplot(x=dataset1['YearsExperience'],y=dataset1['Salary'])
```

Out[27]:

```
<AxesSubplot:xlabel='YearsExperience', ylabel='Salary'>
```



Model Building

In [29]:

```
model=smf.ols("Salary~YearsExperience",data=dataset1).fit()
```

Model Testing

In [30]:

```
# Finding Coefficient parameters
model.params
```

Out[30]:

```
Intercept          25792.200199
YearsExperience     9449.962321
dtype: float64
```

In [31]:

```
# Finding tvalues and pvalues
model.tvalues, model.pvalues
```

Out[31]:

```
(Intercept          11.346940
 YearsExperience    24.950094
 dtype: float64,
 Intercept          5.511950e-12
 YearsExperience    1.143068e-20
 dtype: float64)
```

In [32]:

```
# Finding Rsquared values
model.rsquared, model.rsquared_adj
```

Out[32]:

```
(0.9569566641435086, 0.9554194021486339)
```

Model Predictions

In [34]:

```
# Manual prediction for say 3 years Experience
Salary=(25792.200199)+(9449.962321)*(3)
Salary
```

Out[34]:

```
54142.087162
```

In [35]:

```
# Automatic Prediction for say 3 & 5 Years Experience
```

In [36]:

```
new_data=pd.Series([3,5])
new_data
```

Out[36]:

```
0    3
1    5
dtype: int64
```

In [38]:

```
data_pred=pd.DataFrame(new_data,columns=['YearsExperience'])
data_pred
```

Out[38]:

	YearsExperience
0	3
1	5

In [39]:

```
model.predict(data_pred)
```

Out[39]:

```
0    54142.087163
1    73042.011806
dtype: float64
```

In [ ]:

```

```