1. Write a python program for linear regression?

In [15]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics
%matplotlib inline
```

In [3]:

```
df= pd.read_csv('student_scores.csv')
df.head()
```

Out[3]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

In [4]:

```
df.describe()
```

Out[4]:

	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

In [5]:

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

Splitting the data into train and test

In [8]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

In [10]:

```
from sklearn.linear_model import LinearRegression
linear_regressor = LinearRegression()
```

```
linear_regressor.fit(X_train, y_train)
```

Out[10]:

```
LinearRegression()
```

In [11]:

```
print(linear_regressor.intercept_)
print(linear_regressor.coef_)
```

```
2.0181600414346974
[9.91065648]
```

In [12]:

```
y_pred = linear_regressor.predict(X_test)
```

In [13]:

```
final=pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
final
```

Out[13]:

	Actual	Predicted
0	20	16.884145
1	27	33.732261
2	69	75.357018
3	30	26.794801
4	62	60.491033

In [16]:

```
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Root Mean Squared Error: 4.647447612100367
```

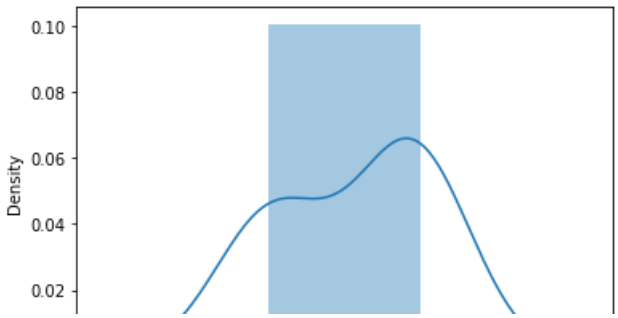
In [17]:

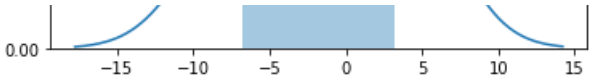
```
import seaborn as sns
sns.distplot(y_test-y_pred)
```

```
C:\Users\gulla\spyder_and_jupyter\lib\site-packages\seaborn\distributions.py:2551: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adapt your cod
e to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axe
s-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[17]:

```
<AxesSubplot:ylabel='Density'>
```





2.Write a python code for implementing K-NN algorithm?

Here I would likely load the iris dataset that is already present as a data

In [23]:

```
from sklearn.datasets import load_iris
data=pd.read_csv('iris.csv')

data.head()
```

Out[23]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [25]:

```
from sklearn.neighbors import KNeighborsClassifier
x=data.iloc[:,:5] #all parameters
y=data["Species"] #class labels
```

In [30]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

In [31]:

```
neigh=KNeighborsClassifier(n_neighbors=4)
neigh.fit(X_train,y_train)
```

Out[31]:

```
KNeighborsClassifier(n_neighbors=4)
```

In [32]:

```
y_pred = neigh.predict(X_test)
```

In [34]:

```
final=pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
final.head()
```

Out[34]:

	Actual	Predicted
114	Iris-virginica	Iris-virginica
62	Iris-versicolor	Iris-versicolor
33	Iris-setosa	Iris-setosa
107	Iris-virginica	Iris-virginica

	Actual	Predicted
7	Iris-setosa	Iris-setosa

In [ ]:

```

```

In [ ]:

```

```