

In [2]: 1 *#Apple dataset*

In [1]: 1 **import** pandas **as** pd
2 t=pd.Series([10.5,11,11.5,12,12.5,13,13.5,14,14.5,15,15.5,16,16.5,17,17.5,
3 t

Out[1]: 0 10.5
1 11.0
2 11.5
3 12.0
4 12.5
...
295 28.0
296 28.5
297 29.0
298 29.5
299 30.0
Length: 300, dtype: float64

In [13]: 1 h=pd.Series([66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,5
2 h

Out[13]: 0 66
1 67
2 68
3 69
4 70
...
295 96
296 97
297 98
298 99
299 100
Length: 300, dtype: int64

In [14]: 1 r=pd.Series([205,210,215,220,225,230,235,240,245,250,255,260,265,270,275,2
2 r

Out[14]: 0 205
1 210
2 215
3 220
4 225
...
295 375
296 380
297 390
298 395
299 400
Length: 300, dtype: int64

```
In [15]: 1 apple=pd.Series([21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,
2         apple
```

```
Out[15]: 0      21
1      22
2      23
3      24
4      25
...
295    51
296    52
297    53
298    54
299    55
Length: 300, dtype: int64
```

```
In [16]: 1 apple=pd.DataFrame({"Temperature":t,"Humidity":h,"Rainfall":r,"Apple":appl
2         apple
```

```
Out[16]:
```

	Temperature	Humidity	Rainfall	Apple
0	10.5	66	205	21
1	11.0	67	210	22
2	11.5	68	215	23
3	12.0	69	220	24
4	12.5	70	225	25
...
295	28.0	96	375	51
296	28.5	97	380	52
297	29.0	98	390	53
298	29.5	99	395	54
299	30.0	100	400	55

300 rows × 4 columns

```
In [19]: 1 apple.to_csv("apple.csv",index=False)
2         apple=pd.read_csv("apple.csv")
```

```
In [20]: 1 x=apple.drop(["Apple"],axis=1)
2         y=apple['Apple']
```

In [43]:

```

1 print(x)
2 print(y)

```

	Temperature	Humidity	Rainfall
0	10.5	66	205
1	11.0	67	210
2	11.5	68	215
3	12.0	69	220
4	12.5	70	225
..
295	28.0	96	375
296	28.5	97	380
297	29.0	98	390
298	29.5	99	395
299	30.0	100	400

[300 rows x 3 columns]

```

0    21
1    22
2    23
3    24
4    25
..
295  16
296  17
297  18
298  19
299  20

```

Name: Orange, Length: 300, dtype: int64

In [22]:

```

1 import pandas as pd
2 import numpy as np
3 from sklearn.preprocessing import StandardScaler
4 from sklearn.neural_network import BernoulliRBM
5 from sklearn.model_selection import train_test_split
6 from sklearn.ensemble import RandomForestClassifier
7 from sklearn.preprocessing import LabelEncoder
8 from sklearn import linear_model

```

In [23]:

```

1 scaler=StandardScaler()
2 x=scaler.fit_transform(x)
3 x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_s
4 classifier=RandomForestClassifier(n_estimators=1,random_state=5)
5 classifier.fit(x_train,y_train)

```

Out[23]: RandomForestClassifier(n_estimators=1, random_state=5)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [24]: 1 #Checking accuracy with random forest
2 from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
3 preds=classifier.predict(x_test)
4 cm=confusion_matrix(y_test,preds)
5 report=classification_report(y_test,preds)
6 print("Accuracy: ",accuracy_score(y_test,preds))
7 print("Confusion Matrix: \n",cm)
8 print("Report: \n",report)
```

Accuracy: 1.0

Confusion Matrix:

```
[[2 0 0 ... 0 0 0]
 [0 1 0 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 ...
 [0 0 0 ... 2 0 0]
 [0 0 0 ... 0 3 0]
 [0 0 0 ... 0 0 1]]
```

Report:

	precision	recall	f1-score	support
6	1.00	1.00	1.00	2
7	1.00	1.00	1.00	1
8	1.00	1.00	1.00	1
9	1.00	1.00	1.00	1
10	1.00	1.00	1.00	1
11	1.00	1.00	1.00	1
12	1.00	1.00	1.00	3
13	1.00	1.00	1.00	1
14	1.00	1.00	1.00	1
15	1.00	1.00	1.00	1
16	1.00	1.00	1.00	4
17	1.00	1.00	1.00	4
18	1.00	1.00	1.00	2
19	1.00	1.00	1.00	1
20	1.00	1.00	1.00	2
21	1.00	1.00	1.00	1
22	1.00	1.00	1.00	1
23	1.00	1.00	1.00	3
26	1.00	1.00	1.00	1
27	1.00	1.00	1.00	3
30	1.00	1.00	1.00	1
31	1.00	1.00	1.00	1
32	1.00	1.00	1.00	3
33	1.00	1.00	1.00	1
34	1.00	1.00	1.00	1
35	1.00	1.00	1.00	4
36	1.00	1.00	1.00	2
38	1.00	1.00	1.00	1
40	1.00	1.00	1.00	2
41	1.00	1.00	1.00	1
42	1.00	1.00	1.00	2
43	1.00	1.00	1.00	2
44	1.00	1.00	1.00	3
45	1.00	1.00	1.00	2
47	1.00	1.00	1.00	1
48	1.00	1.00	1.00	1
49	1.00	1.00	1.00	3
50	1.00	1.00	1.00	1
51	1.00	1.00	1.00	2
53	1.00	1.00	1.00	2
54	1.00	1.00	1.00	3
55	1.00	1.00	1.00	1
accuracy			1.00	75
macro avg	1.00	1.00	1.00	75

weighted avg	1.00	1.00	1.00	75
--------------	------	------	------	----

```
In [44]: 1 #Linear regression for apple dataset
2 import pandas
3 from sklearn import linear_model
4
5 df=pandas.read_csv("apple.csv")
6
7 x=df[['Temperature','Humidity','Rainfall']]
8 y=df['Apple']
9
10 regr=linear_model.LinearRegression()
11 regr.fit(x,y)
12
13 predictedapple=regr.predict([[21,60,300]])
14
15 print(predictedapple)
16
17 print(regr.coef_)
18
19
```

[15.]

[-5.10373812e-16 1.00000000e+00 1.11022302e-16]

C:\Users\Nikhil\anaconda3\anaconda\envs\bb\Lib\site-packages\sklearn\base.py:
464: UserWarning: X does not have valid feature names, but LinearRegression w
as fitted with feature names
warnings.warn(

```
In [45]: 1 train=apple[['Temperature','Humidity','Rainfall']]
2 train.info()
3
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Temperature  300 non-null    float64
1   Humidity     300 non-null    int64
2   Rainfall     300 non-null    int64
dtypes: float64(1), int64(2)
memory usage: 7.2 KB
```

```
In [46]: 1 #SVR fpr apple dataset
2 from sklearn.model_selection import train_test_split
3 data=pd.read_csv('apple.csv')
4 y=data['Apple']
5 x_train,x_test,y_train,y_test=train_test_split(train,y,test_size=0.2,random_state=42)
6
7 from sklearn.svm import SVR
8 regressor=SVR(kernel='rbf')
9 regressor.fit(x_train,y_train)
10 from sklearn.metrics import r2_score
11 preds=regressor.predict(x_test)
12 print(r2_score(y_test,preds))
```

0.9929220109440607

```
In [47]: 1 #Root mean square error rmse
2 import math
3 import numpy as np
4 MSE=np.square(np.subtract(y_test,preds)).mean()
5 rsme=math.sqrt(MSE)
6 print("Root Mean Sqaure Error: \n")
7 print(rsme)
```

Root Mean Sqaure Error:

1.270733866232302

```
In [ ]: 1 #Orange dataset
```

```
In [27]: 1 import pandas as pd
2 t1=pd.Series([10.5,11,11.5,12,12.5,13,13.5,14,14.5,15,15.5,16,16.5,17,17.5,18,18.5,19,19.5,20,20.5,21,21.5,22,22.5,23,23.5,24,24.5,25,25.5,26,26.5,27,27.5,28,28.5,29,29.5,30,30.5,31,31.5,32,32.5,33,33.5,34,34.5,35,35.5,36,36.5,37,37.5,38,38.5,39,39.5,40,40.5,41,41.5,42,42.5,43,43.5,44,44.5,45,45.5,46,46.5,47,47.5,48,48.5,49,49.5,50,50.5,51,51.5,52,52.5,53,53.5,54,54.5,55,55.5,56,56.5,57,57.5,58,58.5,59,59.5,60,60.5,61,61.5,62,62.5,63,63.5,64,64.5,65,65.5,66,66.5,67,67.5,68,68.5,69,69.5,70,70.5,71,71.5,72,72.5,73,73.5,74,74.5,75,75.5,76,76.5,77,77.5,78,78.5,79,79.5,80,80.5,81,81.5,82,82.5,83,83.5,84,84.5,85,85.5,86,86.5,87,87.5,88,88.5,89,89.5,90,90.5,91,91.5,92,92.5,93,93.5,94,94.5,95,95.5,96,96.5,97,97.5,98,98.5,99,99.5,100,100.5,101,101.5,102,102.5,103,103.5,104,104.5,105,105.5,106,106.5,107,107.5,108,108.5,109,109.5,110,110.5,111,111.5,112,112.5,113,113.5,114,114.5,115,115.5,116,116.5,117,117.5,118,118.5,119,119.5,120,120.5,121,121.5,122,122.5,123,123.5,124,124.5,125,125.5,126,126.5,127,127.5,128,128.5,129,129.5,130,130.5,131,131.5,132,132.5,133,133.5,134,134.5,135,135.5,136,136.5,137,137.5,138,138.5,139,139.5,140,140.5,141,141.5,142,142.5,143,143.5,144,144.5,145,145.5,146,146.5,147,147.5,148,148.5,149,149.5,150,150.5,151,151.5,152,152.5,153,153.5,154,154.5,155,155.5,156,156.5,157,157.5,158,158.5,159,159.5,160,160.5,161,161.5,162,162.5,163,163.5,164,164.5,165,165.5,166,166.5,167,167.5,168,168.5,169,169.5,170,170.5,171,171.5,172,172.5,173,173.5,174,174.5,175,175.5,176,176.5,177,177.5,178,178.5,179,179.5,180,180.5,181,181.5,182,182.5,183,183.5,184,184.5,185,185.5,186,186.5,187,187.5,188,188.5,189,189.5,190,190.5,191,191.5,192,192.5,193,193.5,194,194.5,195,195.5,196,196.5,197,197.5,198,198.5,199,199.5,200,200.5,201,201.5,202,202.5,203,203.5,204,204.5,205,205.5,206,206.5,207,207.5,208,208.5,209,209.5,210,210.5,211,211.5,212,212.5,213,213.5,214,214.5,215,215.5,216,216.5,217,217.5,218,218.5,219,219.5,220,220.5,221,221.5,222,222.5,223,223.5,224,224.5,225,225.5,226,226.5,227,227.5,228,228.5,229,229.5,230,230.5,231,231.5,232,232.5,233,233.5,234,234.5,235,235.5,236,236.5,237,237.5,238,238.5,239,239.5,240,240.5,241,241.5,242,242.5,243,243.5,244,244.5,245,245.5,246,246.5,247,247.5,248,248.5,249,249.5,250,250.5,251,251.5,252,252.5,253,253.5,254,254.5,255,255.5,256,256.5,257,257.5,258,258.5,259,259.5,260,260.5,261,261.5,262,262.5,263,263.5,264,264.5,265,265.5,266,266.5,267,267.5,268,268.5,269,269.5,270,270.5,271,271.5,272,272.5,273,273.5,274,274.5,275,275.5,276,276.5,277,277.5,278,278.5,279,279.5,280,280.5,281,281.5,282,282.5,283,283.5,284,284.5,285,285.5,286,286.5,287,287.5,288,288.5,289,289.5,290,290.5,291,291.5,292,292.5,293,293.5,294,294.5,295,295.5,296,296.5,297,297.5,298,298.5,299,299.5,300,300.5,301,301.5,302,302.5,303,303.5,304,304.5,305,305.5,306,306.5,307,307.5,308,308.5,309,309.5,310,310.5,311,311.5,312,312.5,313,313.5,314,314.5,315,315.5,316,316.5,317,317.5,318,318.5,319,319.5,320,320.5,321,321.5,322,322.5,323,323.5,324,324.5,325,325.5,326,326.5,327,327.5,328,328.5,329,329.5,330,330.5,331,331.5,332,332.5,333,333.5,334,334.5,335,335.5,336,336.5,337,337.5,338,338.5,339,339.5,340,340.5,341,341.5,342,342.5,343,343.5,344,344.5,345,345.5,346,346.5,347,347.5,348,348.5,349,349.5,350,350.5,351,351.5,352,352.5,353,353.5,354,354.5,355,355.5,356,356.5,357,357.5,358,358.5,359,359.5,360,360.5,361,361.5,362,362.5,363,363.5,364,364.5,365,365.5,366,366.5,367,367.5,368,368.5,369,369.5,370,370.5,371,371.5,372,372.5,373,373.5,374,374.5,375,375.5,376,376.5,377,377.5,378,378.5,379,379.5,380,380.5,381,381.5,382,382.5,383,383.5,384,384.5,385,385.5,386,386.5,387,387.5,388,388.5,389,389.5,390,390.5,391,391.5,392,392.5,393,393.5,394,394.5,395,395.5,396,396.5,397,397.5,398,398.5,399,399.5,400,400.5,401,401.5,402,402.5,403,403.5,404,404.5,405,405.5,406,406.5,407,407.5,408,408.5,409,409.5,410,410.5,411,411.5,412,412.5,413,413.5,414,414.5,415,415.5,416,416.5,417,417.5,418,418.5,419,419.5,420,420.5,421,421.5,422,422.5,423,423.5,424,424.5,425,425.5,426,426.5,427,427.5,428,428.5,429,429.5,430,430.5,431,431.5,432,432.5,433,433.5,434,434.5,435,435.5,436,436.5,437,437.5,438,438.5,439,439.5,440,440.5,441,441.5,442,442.5,443,443.5,444,444.5,445,445.5,446,446.5,447,447.5,448,448.5,449,449.5,450,450.5,451,451.5,452,452.5,453,453.5,454,454.5,455,455.5,456,456.5,457,457.5,458,458.5,459,459.5,460,460.5,461,461.5,462,462.5,463,463.5,464,464.5,465,465.5,466,466.5,467,467.5,468,468.5,469,469.5,470,470.5,471,471.5,472,472.5,473,473.5,474,474.5,475,475.5,476,476.5,477,477.5,478,478.5,479,479.5,480,480.5,481,481.5,482,482.5,483,483.5,484,484.5,485,485.5,486,486.5,487,487.5,488,488.5,489,489.5,490,490.5,491,491.5,492,492.5,493,493.5,494,494.5,495,495.5,496,496.5,497,497.5,498,498.5,499,499.5,500,500.5,501,501.5,502,502.5,503,503.5,504,504.5,505,505.5,506,506.5,507,507.5,508,508.5,509,509.5,510,510.5,511,511.5,512,512.5,513,513.5,514,514.5,515,515.5,516,516.5,517,517.5,518,518.5,519,519.5,520,520.5,521,521.5,522,522.5,523,523.5,524,524.5,525,525.5,526,526.5,527,527.5,528,528.5,529,529.5,530,530.5,531,531.5,532,532.5,533,533.5,534,534.5,535,535.5,536,536.5,537,537.5,538,538.5,539,539.5,540,540.5,541,541.5,542,542.5,543,543.5,544,544.5,545,545.5,546,546.5,547,547.5,548,548.5,549,549.5,550,550.5,551,551.5,552,552.5,553,553.5,554,554.5,555,555.5,556,556.5,557,557.5,558,558.5,559,559.5,560,560.5,561,561.5,562,562.5,563,563.5,564,564.5,565,565.5,566,566.5,567,567.5,568,568.5,569,569.5,570,570.5,571,571.5,572,572.5,573,573.5,574,574.5,575,575.5,576,576.5,577,577.5,578,578.5,579,579.5,580,580.5,581,581.5,582,582.5,583,583.5,584,584.5,585,585.5,586,586.5,587,587.5,588,588.5,589,589.5,590,590.5,591,591.5,592,592.5,593,593.5,594,594.5,595,595.5,596,596.5,597,597.5,598,598.5,599,599.5,600,600.5,601,601.5,602,602.5,603,603.5,604,604.5,605,605.5,606,606.5,607,607.5,608,608.5,609,609.5,610,610.5,611,611.5,612,612.5,613,613.5,614,614.5,615,615.5,616,616.5,617,617.5,618,618.5,619,619.5,620,620.5,621,621.5,622,622.5,623,623.5,624,624.5,625,625.5,626,626.5,627,627.5,628,628.5,629,629.5,630,630.5,631,631.5,632,632.5,633,633.5,634,634.5,635,635.5,636,636.5,637,637.5,638,638.5,639,639.5,640,640.5,641,641.5,642,642.5,643,643.5,644,644.5,645,645.5,646,646.5,647,647.5,648,648.5,649,649.5,650,650.5,651,651.5,652,652.5,653,653.5,654,654.5,655,655.5,656,656.5,657,657.5,658,658.5,659,659.5,660,660.5,661,661.5,662,662.5,663,663.5,664,664.5,665,665.5,666,666.5,667,667.5,668,668.5,669,669.5,670,670.5,671,671.5,672,672.5,673,673.5,674,674.5,675,675.5,676,676.5,677,677.5,678,678.5,679,679.5,680,680.5,681,681.5,682,682.5,683,683.5,684,684.5,685,685.5,686,686.5,687,687.5,688,688.5,689,689.5,690,690.5,691,691.5,692,692.5,693,693.5,694,694.5,695,695.5,696,696.5,697,697.5,698,698.5,699,699.5,700,700.5,701,701.5,702,702.5,703,703.5,704,704.5,705,705.5,706,706.5,707,707.5,708,708.5,709,709.5,710,710.5,711,711.5,712,712.5,713,713.5,714,714.5,715,715.5,716,716.5,717,717.5,718,718.5,719,719.5,720,720.5,721,721.5,722,722.5,723,723.5,724,724.5,725,725.5,726,726.5,727,727.5,728,728.5,729,729.5,730,730.5,731,731.5,732,732.5,733,733.5,734,734.5,735,735.5,736,736.5,737,737.5,738,738.5,739,739.5,740,740.5,741,741.5,742,742.5,743,743.5,744,744.5,745,745.5,746,746.5,747,747.5,748,748.5,749,749.5,750,750.5,751,751.5,752,752.5,753,753.5,754,754.5,755,755.5,756,756.5,757,757.5,758,758.5,759,759.5,760,760.5,761,761.5,762,762.5,763,763.5,764,764.5,765,765.5,766,766.5,767,767.5,768,768.5,769,769.5,770,770.5,771,771.5,772,772.5,773,773.5,774,774.5,775,775.5,776,776.5,777,777.5,778,778.5,779,779.5,780,780.5,781,781.5,782,782.5,783,783.5,784,784.5,785,785.5,786,786.5,787,787.5,788,788.5,789,789.5,790,790.5,791,791.5,792,792.5,793,793.5,794,794.5,795,795.5,796,796.5,797,797.5,798,798.5,799,799.5,800,800.5,801,801.5,802,802.5,803,803.5,804,804.5,805,805.5,806,806.5,807,807.5,808,808.5,809,809.5,810,810.5,811,811.5,812,812.5,813,813.5,814,814.5,815,815.5,816,816.5,817,817.5,818,818.5,819,819.5,820,820.5,821,821.5,822,822.5,823,823.5,824,824.5,825,825.5,826,826.5,827,827.5,828,828.5,829,829.5,830,830.5,831,831.5,832,832.5,833,833.5,834,834.5,835,835.5,836,836.5,837,837.5,838,838.5,839,839.5,840,840.5,841,841.5,842,842.5,843,843.5,844,844.5,845,845.5,846,846.5,847,847.5,848,848.5,849,849.5,850,850.5,851,851.5,852,852.5,853,853.5,854,854.5,855,855.5,856,856.5,857,857.5,858,858.5,859,859.5,860,860.5,861,861.5,862,862.5,863,863.5,864,864.5,865,865.5,866,866.5,867,867.5,868,868.5,869,869.5,870,870.5,871,871.5,872,872.5,873,873.5,874,874.5,875,875.5,876,876.5,877,877.5,878,878.5,879,879.5,880,880.5,881,881.5,882,882.5,883,883.5,884,884.5,885,885.5,886,886.5,887,887.5,888,888.5,889,889.5,890,890.5,891,891.5,892,892.5,893,893.5,894,894.5,895,895.5,896,896.5,897,897.5,898,898.5,899,899.5,900,900.5,901,901.5,902,902.5,903,903.5,904,904.5,905,905.5,906,906.5,907,907.5,908,908.5,909,909.5,910,910.5,911,911.5,912,912.5,913,913.5,914,914.5,915,915.5,916,916.5,917,917.5,918,918.5,919,919.5,920,920.5,921,921.5,922,922.5,923,923.5,924,924.5,925,925.5,926,926.5,927,927.5,928,928.5,929,929.5,930,930.5,931,931.5,932,932.5,933,933.5,934,934.5,935,935.5,936,936.5,937,937.5,938,938.5,939,939.5,940,940.5,941,941.5,942,942.5,943,943.5,944,944.5,945,945.5,946,946.5,947,947.5,948,948.5,949,949.5,950,950.5,951,951.5,952,952.5,953,953.5,954,954.5,955,955.5,956,956.5,957,957.5,958,958.5,959,959.5,960,960.5,961,961.5,962,962.5,963,963.5,964,964.5,965,965.5,966,966.5,967,967.5,968,968.5,969,969.5,970,970.5,971,971.5,972,972.5,973,973.5,974,974.5,975,975.5,976,976.5,977,977.5,978,978.5,979,979.5,980,980.5,981,981.5,982,982.5,983,983.5,984,984.5,985,985.5,986,986.5,987,987.5,988,988.5,989,989.5,990,990.5,991,991.5,992,992.5,993,993.5,994,994.5,995,995.5,996,996.5,997,997.5,998,998.5,999,999.5,1000,1000.5,1001,1001.5,1002,1002.5,1003,1003.5,1004,1004.5,1005,1005.5,1006,1006.5,1007,1007.5,1008,1008.5,1009,1009.5,1010,1010.5,1011,1011.5,1012,1012.5,1013,1013.5,1014,1014.5,1015,1015.5,1016,1016.5,1017,1017.5,1018,1018.5,1019,1019.5,1020,1020.5,1021,1021.5,1022,1022.5,1023,1023.5,1024,1024.5,1025,1025.5,1026,1026.5,1027,1027.5,1028,1028.5,1029,1029.5,1030,1030.5,1031,1031.5,1032,1032.5,1033,1033.5,1034,1034.5,1035,1035.5,1036,1036.5,1037,1037.5,1038,1038.5,1039,1039.5,1040,1040.5,1041,1041.5,1042,1042.5,1043,1043.5,1044,1044.5,1045,1045.5,1046,1046.5,1047,1047.5,1048,1048.5,1049,1049.5,1050,1050.5,1051,1051.5,1052,1052.5,1053,1053.5,1054,1054.5,1055,1055.5,1056,1056.5,1057,1057.5,1058,1058.5,1059,1059.5,1060,1060.5,1061,1061.5,1062,1062.5,1063,1063.5,1064,1064.5,1065,1065.5,1066,1066.5,1067,1067.5,1068,1068.5,1069,1069.5,1070,1070.5,1071,1071.5,1072,1072.5,1073,1073.5,1074,1074.5,1075,1075.5,1076,1076.5,1077,1077.5,1078,1078.5,1079,1079.5,1080,1080.5,1081,1081.5,1082,1082.5,1083,1083.5,1084,1084.5,1085,1085.5,1086,1086.5,1087,1087.5,1088,1088.5,1089,1089.5,1090,1090.5,1091,1091.5,1092,1092.5,1093,1093.5,1094,1094.5,1095,1095.5,1096,1096.5,1097,1097.5,1098,1098.5,1099,1099.5,1100,1100.5,1101,1101.5,1102,1102.5,1103,1103.5,1104,1104.5,1105,1105.5,1106,1106.5,1107,1107.5,1108,1108.5,1109,1109.5,1110,1110.5,1111,1111.5,1112,1112.5,1113,1113.5,1114,1114.5,1115,1115.5,1116,1116.5,1117,1117.5,1118,1118.5,1119,1119.5,1120,1120.5,1121,1121.5,1122,1122.5,1123,1123.5,1124,1124.5,1125,1125.5,1126,1126.5,1127,1127.5,1128,1128.5,1129,1129.5,1130,1130.5,1131,1131.5,1132,1132.5,1133,1133.5,1134,1134.5,1135,1135.5,1136,1136.5,1137,1137.5,1138,1138.5,1139,1139.5,1140,1140.5,1141,1141.5,1142,1142.5,1143,1143.5,1144,1144.5,1145,1145.5,1146,1146.5,1147,1147.5,1148,1148.5,1149,1149.5,1150,1150.5,1151,1151.5,1152,1152.5,1153,1153.5,1154,1154.5,1155,1155.5,1156,1156.5,1157,1157.5,1158,1158.5,1159,1159.5,1160,1160.5,1161,1161.5,1162,1162.5,1163,1163.5,1164,1164.5,1165,1165.5,1166,1166.5,1167,1167.5,1168,1168.5,1169,1169.5,1170,1170.5,1171,1171.5,1172,1172.
```

```
In [28]: 1 h1=pd.Series([66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,  
2 h1
```

```
Out[28]: 0      66  
1      67  
2      68  
3      69  
4      70  
...  
295    96  
296    97  
297    98  
298    99  
299   100  
Length: 300, dtype: int64
```

```
In [29]: 1 r1=pd.Series([205,210,215,220,225,230,235,240,245,250,255,260,265,270,275,  
2 r1
```

```
Out[29]: 0      205  
1      210  
2      215  
3      220  
4      225  
...  
295    375  
296    380  
297    390  
298    395  
299    400  
Length: 300, dtype: int64
```

```
In [30]: 1 orange=pd.Series([21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39  
2 orange
```

```
Out[30]: 0      21  
1      22  
2      23  
3      24  
4      25  
..  
295    16  
296    17  
297    18  
298    19  
299    20  
Length: 300, dtype: int64
```



```
In [31]: 1 orange=pd.DataFrame({"Temperature":t1,"Humidity":h1,"Rainfall":r1,"Orange"  
2 orange
```

```
Out[31]:
```

	Temperature	Humidity	Rainfall	Orange
0	10.5	66	205	21
1	11.0	67	210	22
2	11.5	68	215	23
3	12.0	69	220	24
4	12.5	70	225	25
...
295	28.0	96	375	16
296	28.5	97	380	17
297	29.0	98	390	18
298	29.5	99	395	19
299	30.0	100	400	20

300 rows × 4 columns

In [32]:

```

1 orange.to_csv("orange.csv",index=False)
2 x=orange.drop(["Orange"],axis=1)
3 y=orange['Orange']
4 print(x)
5 print(y)

```

	Temperature	Humidity	Rainfall
0	10.5	66	205
1	11.0	67	210
2	11.5	68	215
3	12.0	69	220
4	12.5	70	225
..
295	28.0	96	375
296	28.5	97	380
297	29.0	98	390
298	29.5	99	395
299	30.0	100	400

[300 rows x 3 columns]

0	21
1	22
2	23
3	24
4	25
..	
295	16
296	17
297	18
298	19
299	20

Name: Orange, Length: 300, dtype: int64

```
In [38]: 1 #Linear regression for orange dataset
2 import pandas
3 from sklearn import linear_model
4
5 df=pandas.read_csv("orange.csv")
6
7 x=df[['Temperature','Humidity','Rainfall']]
8 y=df['Orange']
9
10 regr=linear_model.LinearRegression()
11 regr.fit(x,y)
12
13 predictedorange=regr.predict([[10,65,200]])
14
15 print(predictedorange)
16
17 print(regr.coef_)
18
```

```
[20.10462802]
[-17.4356313    2.24138777    1.37229122]
```

C:\Users\Nikhil\anaconda3\anaconda\envs\bb\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
warnings.warn(

```
In [35]: 1 train=orange[['Temperature','Humidity','Rainfall']]
2 train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Temperature  300 non-null    float64
1   Humidity     300 non-null    int64
2   Rainfall     300 non-null    int64
dtypes: float64(1), int64(2)
memory usage: 7.2 KB
```

```
In [36]: 1 #SVR for apple dataset
2 from sklearn.model_selection import train_test_split
3 data=pd.read_csv('orange.csv')
4 y=data['Orange']
5 x_train,x_test,y_train,y_test=train_test_split(train,y,test_size=0.2,random_state=42)
6
7 from sklearn.svm import SVR
8 regressor=SVR(kernel='rbf')
9 regressor.fit(x_train,y_train)
10 from sklearn.metrics import r2_score
11 preds=regressor.predict(x_test)
12 print(r2_score(y_test,preds))
```

```
0.38661139794101884
```

```
In [39]: 1 #Root mean square error
          2 import math
          3 import numpy as np
          4 MSE=np.square(np.subtract(y_test,preds)).mean()
          5 rsme=math.sqrt(MSE)
          6 print("Root Mean Sqaure Error: \n")
          7 print(rsme)
```

Root Mean Sqaure Error:

7.322303416292769

In []:

1

In []:

1