In [2]:
```
#Apple dataset
```

In [1]:
```
import pandas as pd
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,
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]:
```
h=pd.Series([66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,5
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]:
```
r=pd.Series([205,210,215,220,225,230,235,240,245,250,255,260,265,270,275,2
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]:
```
apple=pd.Series([21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,
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]:
```
apple=pd.DataFrame({"Temperature":t,"Humidity":h,"Rainfall":r,"Apple":appl
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]:
```
apple.to_csv("apple.csv",index=False)
apple=pd.read_csv("apple.csv")
```

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

In [43]:
```
print(x)
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]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import BernoulliRBM
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn import linear_model
```

In [23]:
```
scaler=StandardScaler()
x=scaler.fit_transform(x)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_s
classifier=RandomForestClassifier(n_estimators=1,random_state=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]:

```
#Checking accuracy with random forest
from sklearn.metrics import confusion_matrix, accuracy_score, classificati
preds=classifier.predict(x_test)
cm=confusion_matrix(y_test,preds)
report=classification_report(y_test,preds)
print("Accuracy: ",accuracy_score(y_test,preds))
print("Confusion Matrix: \n",cm)
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]:

```
#linear regression for apple dataset
import pandas
from sklearn import linear_model

df=pandas.read_csv("apple.csv")

x=df[['Temperature','Humidity','Rainfall']]
y=df['Apple']

regr=linear_model.LinearRegression()
regr.fit(x,y)

predictedapple=regr.predict([[21,60,300]])

print(predictedapple)

print(regr.coef_)


```

```
[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]:

```
train=apple[['Temperature','Humidity','Rainfall']]
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 [46]:
```
#SVR fpr apple dataset
from sklearn.model_selection import train_test_split
data=pd.read_csv('apple.csv')
y=data['Apple']
x_train,x_test,y_train,y_test=train_test_split(train,y,test_size=0.2,rando

from sklearn.svm import SVR
regressor=SVR(kernel='rbf')
regressor.fit(x_train,y_train)
from sklearn.metrics import r2_score
preds=regressor.predict(x_test)
print(r2_score(y_test,preds))
```

```
0.9929220109440607
```

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

```
Root Mean Sqaure Error:

1.270733866232302
```

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

In [27]:
```
import pandas as pd
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
t1
```

Out[27]:
```
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 [28]:
```
h1=pd.Series([66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,
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]:
```
r1=pd.Series([205,210,215,220,225,230,235,240,245,250,255,260,265,270,275,
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]:
```
orange=pd.Series([21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39
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]:
```
orange=pd.DataFrame({"Temperature":t1,"Humidity":h1,"Rainfall":r1,"Orange"
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]:
```
orange.to_csv("orange.csv",index=False)
x=orange.drop(["Orange"],axis=1)
y=orange['Orange']
print(x)
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]:
```
#Linear regression for orange dataset
import pandas
from sklearn import linear_model

df=pandas.read_csv("orange.csv")

x=df[['Temperature','Humidity','Rainfall']]
y=df['Orange']

regr=linear_model.LinearRegression()
regr.fit(x,y)

predictedorange=regr.predict([[10,65,200]])

print(predictedorange)

print(regr.coef_)

```

```
[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 w
as fitted with feature names
  warnings.warn(
```

In [35]:
```
train=orange[['Temperature','Humidity','Rainfall']]
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]:
```
#SVR for apple dataset
from sklearn.model_selection import train_test_split
data=pd.read_csv('orange.csv')
y=data['Orange']
x_train,x_test,y_train,y_test=train_test_split(train,y,test_size=0.2,rando

from sklearn.svm import SVR
regressor=SVR(kernel='rbf')
regressor.fit(x_train,y_train)
from sklearn.metrics import r2_score
preds=regressor.predict(x_test)
print(r2_score(y_test,preds))
```

```
0.38661139794101884
```

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

```
Root Mean Sqaure Error:

7.322303416292769
```

In [ ]:
```

```

In [ ]:
```

```