In [2]:
```
#Preprocessing steps and Machine learning algorithms
```

In [1]:
```
import pandas as pd
name=pd.Series(["John", "Sarah", "Rajesh","Sarah", "Rajesh","Maria", "Amit
name
```

Out[1]:
```
0         John
1        Sarah
2       Rajesh
3        Sarah
4       Rajesh
        ...
395       Ritu
396      James
397       Lila
398    Amitabh
399      Lucas
Length: 400, dtype: object
```

In [2]:
```
age=pd.Series([67, 12, 31, 58, 42, 19, 75, 5, 64, 27, 50, 7, 36, 72, 14, 4
age
```

Out[2]:
```
0      67
1      12
2      31
3      58
4      42
       ..
395    51
396    43
397    20
398    61
399    39
Length: 400, dtype: int64
```

In [3]:
```

bp=pd.Series(["120/80", "130/85", "115/70", "140/90", "110/75", "125/82",
bp
```

Out[3]:
```
0      120/80
1      130/85
2      115/70
3      140/90
4      110/75
        ...
395    112/72
396    132/86
397    116/75
398    138/88
399    113/71
Length: 400, dtype: object
```

In [8]:
```
dia=pd.Series([143, 234, 176, 321, 267, 189, 355, 289, 156, 398, 249, 134,
dia
```

Out[8]:
```
0      143
1      234
2      176
3      321
4      267
      ...
395    271
396    199
397    352
398    232
399    169
Length: 400, dtype: int64
```

In [6]:
```
result=pd.Series([ "Yes", "No", "Yes", "No", "Yes",  "No", "Yes", "No", "Y
result
```

Out[6]:
```
0      Yes
1       No
2      Yes
3       No
4      Yes
      ...
395    Yes
396     No
397     No
398    Yes
399    Yes
Length: 400, dtype: object
```

In [9]:
```
report=pd.DataFrame({"Name":name,"Age":age,"BP":bp,"Diabities":dia,"Result
report
```

Out[9]:

	Name	Age	BP	Diabities	Result
0	John	67	120/80	143	Yes
1	Sarah	12	130/85	234	No
2	Rajesh	31	115/70	176	Yes
3	Sarah	58	140/90	321	No
4	Rajesh	42	110/75	267	Yes
...	...	...	...	...	...
395	Ritu	51	112/72	271	Yes
396	James	43	132/86	199	No
397	Lila	20	116/75	352	No
398	Amitabh	61	138/88	232	Yes
399	Lucas	39	113/71	169	Yes

400 rows × 5 columns

In [16]:
```
report.to_csv('report.csv',index=False)
report.to_csv('data.csv',index=False)
```

In [ ]:
```

```

In [19]:
```
c=pd.Series([228.8,237.6,219.3,248.5,230.2,200.7,205.9,189.4,231.5,194.6,2
c
```

Out[19]:
```
0      228.8
1      237.6
2      219.3
3      248.5
4      230.2
       ...
395    196.2
396    232.1
397    190.6
398    221.7
399    238.0
Length: 400, dtype: float64
```

In [20]:
```
report=pd.DataFrame({"Name":name,"Age":age,"BP":bp,"Cholesterol":c,"Diabit
report
```

Out[20]:

	Name	Age	BP	Cholesterol	Diabities	Result
0	John	67	120/80	228.8	143	Yes
1	Sarah	12	130/85	237.6	234	No
2	Rajesh	31	115/70	219.3	176	Yes
3	Sarah	58	140/90	248.5	321	No
4	Rajesh	42	110/75	230.2	267	Yes
...	...	...	...	...	...	...
395	Ritu	51	112/72	196.2	271	Yes
396	James	43	132/86	232.1	199	No
397	Lila	20	116/75	190.6	352	No
398	Amitabh	61	138/88	221.7	232	Yes
399	Lucas	39	113/71	238.0	169	Yes

400 rows × 6 columns

In [22]:
```
report.to_csv('report.csv',index=False)
report.to_csv('data.csv',index=False)
```

In [3]:
```
import pandas as pd
report=pd.read_csv("data.csv")
```

In [4]:
```
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
label_encoder=LabelEncoder()
report['BP']=label_encoder.fit_transform(report['BP'])
report['Diabities']=label_encoder.fit_transform(report['Diabities'])
report['Cholesterol']=label_encoder.fit_transform(report['Cholesterol'])
```

In [33]:
```
x=report.drop(['Name','Age','BP','Cholesterol','Diabities','Result'],axis=

```

In [25]:
```
from sklearn.preprocessing import StandardScaler
scalar=StandardScaler()
x=scalar.fit_transform(x)
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
df=pd.read_csv("report.csv")
```

In [5]:
```
1 df.head()
```

Out[5]:

	Name	Age	BP	Cholesterol	Diabities	Result
0	John	67	120/80	228.8	143	Yes
1	Sarah	12	130/85	237.6	234	No
2	Rajesh	31	115/70	219.3	176	Yes
3	Sarah	58	140/90	248.5	321	No
4	Rajesh	42	110/75	230.2	267	Yes

In [ ]:
```
1
```

In [7]:
```
1 import pandas as pd
2 data=pd.read_csv("data.csv")
3 data
```

Out[7]:

	Name	Age	BP	Cholesterol	Diabities	Result
0	John	67	120/80	228.8	143	Yes
1	Sarah	12	130/85	237.6	234	No
2	Rajesh	31	115/70	219.3	176	Yes
3	Sarah	58	140/90	248.5	321	No
4	Rajesh	42	110/75	230.2	267	Yes
...	...	...	...	...	...	...
395	Ritu	51	112/72	196.2	271	Yes
396	James	43	132/86	232.1	199	No
397	Lila	20	116/75	190.6	352	No
398	Amitabh	61	138/88	221.7	232	Yes
399	Lucas	39	113/71	238.0	169	Yes

400 rows × 6 columns

In [8]:
```
1 data.columns
```

Out[8]:
```
Index(['Name', 'Age', 'BP', 'Cholesterol', 'Diabities', 'Result'], dtype='obj
ect')
```

In [9]:
```
1 from sklearn.preprocessing import LabelEncoder
2 label_encoder=LabelEncoder()
3 data['Result']=label_encoder.fit_transform(data['Result'])
4 data['BP']=label_encoder.fit_transform(data['BP'])
5 x=data.drop(['Name','Age'],axis=1)
6 y=data['Result']
```

In [10]:
```
1 x.head()
```

Out[10]:

	BP	Cholesterol	Diabities	Result
0	30	228.8	143	1
1	53	237.6	234	0
2	18	219.3	176	1
3	72	248.5	321	0
4	8	230.2	267	1

In [11]:
```
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 [12]:
```
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=100,random_state=40)
5 classifier.fit(x_train,y_train)
```

Out[12]:
```
RandomForestClassifier(random_state=40)
```
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 [ ]:
```
1
```

In [13]:
```
#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:
 [[49  0]
 [ 0 51]]
Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        49
           1       1.00      1.00      1.00        51

    accuracy                           1.00       100
   macro avg       1.00      1.00      1.00       100
weighted avg       1.00      1.00      1.00       100
```

In [14]:
```
#Naive bayes
from sklearn.naive_bayes import GaussianNB
classifier1=GaussianNB()
classifier1.fit(x_train,y_train)
preds=classifier1.predict(x_test)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,preds)
print("Confusion Matrix:",cm)
```

```
Confusion Matrix: [[49  0]
 [ 0 51]]
```

In [31]:
```
#K neighbours classifier knn
from sklearn.neighbors import KNeighborsClassifier
data1=KNeighborsClassifier(n_neighbors=7)
data1.fit(x_train,y_train)
print("Predictions: \n",data1.predict(x_test))
print("Accuracy Score: \n",data1.score(x_test,y_test))
```

```
Predictions:
 [0 1 1 0 0 0 0 1 0 1 0 1 1 0 0 0 0 1 0 1 1 0 1 1 1 1 0 1 1 0 0 0 0 0 1 0 1
 0 1 1 1 1 0 0 1 0 1 1 0 1 0 0 1 1 0 1 0 0 0 0 0 0 0 1 0 1 1 1 0 1 1 1 1 1
 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 0 1 0 1 1 0 1 1 0 1 1]
Accuracy Score:
 1.0
```

In [33]:
```
#SVM support vector machine
cm=confusion_matrix(y_test,preds)
print("Accuracy: ",accuracy_score(y_test,preds))
print("Confusion Matrix: ",cm)
```

```
Accuracy:  1.0
Confusion Matrix:  [[49  0]
 [ 0 51]]
```

In [4]:
```
import pandas as pd
report=pd.read_csv("report.csv")
```

In [19]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder=LabelEncoder()
report['Result']=label_encoder.fit_transform(report['Result'])
report['BP']=label_encoder.fit_transform(report['BP'])
x=report.drop(['Name','Age'],axis=1)
y=report['Result']
```

In [12]:
```

```

In [20]:
```
print(x)
print(y)
```

```
     BP  Cholesterol  Diabities  Result
0    30           31          8       1
1    53           36         38       0
2    18           24         21       1
3    72           43         61       0
4     8           32         44       1
..   ..          ...        ...     ...
395  12           10         46       1
396  58           35         31       0
397  22            4         67       0
398  67           26         37       1
399  13           38         19       1

[400 rows x 4 columns]
0      1
1      0
2      1
3      0
4      1
      ..
395    1
396    0
397    0
398    1
399    1
Name: Result, Length: 400, dtype: int32
```

In [21]:
```
#Standard Scaler
from sklearn.preprocessing import StandardScaler
scalar=StandardScaler()
x_scaled=scalar.fit_transform(x)
print("Original data",x[:10])
print("Scaled data:",x_scaled[:10])
```

```
Original data    BP  Cholesterol  Diabities  Result
0  30           31           8       1
1  53           36          38       0
2  18           24          21       1
3  72           43          61       0
4   8           32          44       1
5  41           14          27       0
6  62           17          69       1
7  25            3          52       0
8  49           34          14       1
9  33            8          81       0
Scaled data: [[-0.49366865  0.58700788 -1.40982649  0.95118973]
 [ 0.42163799  0.99621519 -0.17602268 -1.05131497]
 [-0.97121994  0.01411765 -0.87517817  0.95118973]
 [ 1.17776087  1.56910543  0.76989358 -1.05131497]
 [-1.36917935  0.66884935  0.07073809  0.95118973]
 [-0.0559133  -0.80429697 -0.62841741 -1.05131497]
 [ 0.77980146 -0.55877258  1.09890793  0.95118973]
 [-0.69264835 -1.70455305  0.39975244 -1.05131497]
 [ 0.26245423  0.83253227 -1.16306573  0.95118973]
 [-0.37428083 -1.29534574  1.59242946 -1.05131497]]
```

In [23]:
```
#Min Max SCaler
from sklearn.preprocessing import MinMaxScaler
scalar=MinMaxScaler()
x_scaled=scalar.fit_transform(x)
print("Original data",x[:10])
print("Scaled data:",x_scaled[:10])
```

```
Original data    BP  Cholesterol  Diabities  Result
0  30           31           8       1
1  53           36          38       0
2  18           24          21       1
3  72           43          61       0
4   8           32          44       1
5  41           14          27       0
6  62           17          69       1
7  25            3          52       0
8  49           34          14       1
9  33            8          81       0
Scaled data: [[0.34883721 0.70454545 0.09756098 1.        ]
 [0.61627907 0.81818182 0.46341463 0.        ]
 [0.20930233 0.54545455 0.25609756 1.        ]
 [0.8372093  0.97727273 0.74390244 0.        ]
 [0.09302326 0.72727273 0.53658537 1.        ]
 [0.47674419 0.31818182 0.32926829 0.        ]
 [0.72093023 0.38636364 0.84146341 1.        ]
 [0.29069767 0.06818182 0.63414634 0.        ]
 [0.56976744 0.77272727 0.17073171 1.        ]
 [0.38372093 0.18181818 0.98780488 0.        ]]
```

In [24]:
```
#Normalizer
from sklearn.preprocessing import Normalizer
scalar=Normalizer()
x_Normalized=scalar.fit_transform(x)
print("Original data",x[:10])
print("Scaled data:",x_Normalized[:10])
```

```
Original data    BP  Cholesterol  Diabities  Result
0  30           31           8          1
1  53           36          38          0
2  18           24          21          1
3  72           43          61          0
4   8           32          44          1
5  41           14          27          0
6  62           17          69          1
7  25            3          52          0
8  49           34          14          1
9  33            8          81          0
Scaled data: [[0.68358593 0.70637212 0.18228958 0.0227862 ]
 [0.71148952 0.4832759  0.51012456 0.        ]
 [0.49135598 0.65514131 0.57324864 0.02729755]
 [0.6943005  0.41465169 0.58822682 0.        ]
 [0.14545455 0.58181818 0.8        0.01818182]
 [0.80314998 0.27424633 0.52890364 0.        ]
 [0.65738337 0.18025028 0.73160407 0.01060296]
 [0.43270991 0.05192519 0.90003661 0.        ]
 [0.79974023 0.5549218  0.22849721 0.01632123]
 [0.37572849 0.09108569 0.92224265 0.        ]]
```

In [27]:

```
#One hot encoder
from sklearn.preprocessing import OneHotEncoder
scalar=OneHotEncoder()
x_Encoded=scalar.fit_transform(x)
print("Original data: ",x)
print(" \nScaled data: ",x_Encoded)
```

```
Original data:       BP  Cholesterol  Diabities  Result
0    30           31           8       1
1    53           36          38       0
2    18           24          21       1
3    72           43          61       0
4     8           32          44       1
..   ..          ...         ...     ...
395  12           10          46       1
396  58           35          31       0
397  22            4          67       0
398  67           26          37       1
399  13           38          19       1

[400 rows x 4 columns]

Scaled data:    (0, 30) 1.0
  (0, 118)      1.0
  (0, 140)      1.0
  (0, 216)      1.0
  (1, 53)       1.0
  (1, 123)      1.0
  (1, 170)      1.0
  (1, 215)      1.0
  (2, 18)       1.0
  (2, 111)      1.0
  (2, 153)      1.0
  (2, 216)      1.0
  (3, 72)       1.0
  (3, 130)      1.0
  (3, 193)      1.0
  (3, 215)      1.0
  (4, 8)        1.0
  (4, 119)      1.0
  (4, 176)      1.0
  (4, 216)      1.0
  (5, 41)       1.0
  (5, 101)      1.0
  (5, 159)      1.0
  (5, 215)      1.0
  (6, 62)       1.0
  :     :
  (393, 216)    1.0
  (394, 74)     1.0
  (394, 115)    1.0
  (394, 203)    1.0
  (394, 216)    1.0
  (395, 12)     1.0
  (395, 97)     1.0
  (395, 178)    1.0
  (395, 216)    1.0
  (396, 58)     1.0
  (396, 122)    1.0
  (396, 163)    1.0
  (396, 215)    1.0
  (397, 22)     1.0
  (397, 91)     1.0
  (397, 199)    1.0
```

```
(397, 215)    1.0
(398, 67)     1.0
(398, 113)    1.0
(398, 169)    1.0
(398, 216)    1.0
(399, 13)     1.0
(399, 125)    1.0
(399, 151)    1.0
(399, 216)    1.0
```

In [29]:
```
1
```

```
Original data:       BP  Cholesterol  Diabities  Result
0    30           31           8       1
1    53           36          38       0
2    18           24          21       1
3    72           43          61       0
4     8           32          44       1
..   ..          ...         ...     ...
395  12           10          46       1
396  58           35          31       0
397  22            4          67       0
398  67           26          37       1
399  13           38          19       1

[400 rows x 4 columns]
Scaled data:  [1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 0 1 1 0 1 0 1 0 0 1 1 0 1 1 0 1
0 0 1 0 1 0
 1 1 0 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0 1
 0 1 0 1 0 0 1 1 1 0 1 1 0 1 0 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1 0 0 1 1 0 1
 0 1 0 0 1 1 0 1 1 0 1 0 0 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0
 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0 1 0 1 0 1 0 0 1 1 1 0 1 1 0 1 0 0 1 0 1 0
 1 1 0 1 0 1 0 1 1 0 1 0 0 1 1 0 1 0 1 0 0 1 1 0 1 1 0 1 0 0 1 0 1 0 1 1 0
 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0 1 0 1 0
 1 0 0 1 1 1 0 1 1 0 1 0 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1 0 0 1 1 0 1 0 1 0
 0 1 1 0 1 1 0 1 0 0 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0 0 1 1
 0 1 0 1 0 1 0 0 1 1 0 1 0 1 0 1 0 1 0 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0 1 0
 1 0 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0 1 0 1 0 0 1 1 1 0 0 1 1]
```

In [33]:
```
#PCA Principe Component Analysis
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
x_pca=pca.fit_transform(x)
print("Original data: ",x[:10])
print("Scaled data: ",x_pca[:10])
```

```
Original data:     BP  Cholesterol  Diabities  Result
0  30           31           8       1
1  53           36          38       0
2  18           24          21       1
3  72           43          61       0
4   8           32          44       1
5  41           14          27       0
6  62           17          69       1
7  25            3          52       0
8  49           34          14       1
9  33            8          81       0
Scaled data:  [[ 21.54749725  30.04963349]
 [ -8.56607425   8.12748309]
 [ 29.27567399  13.66393932]
 [-32.8492746   -8.0117183 ]
 [ 32.9077315  -10.36569876]
 [  5.15673321  13.33407608]
 [-26.38277453 -20.80260723]
 [ 13.30666942 -16.00326334]
 [  1.75601836  29.81246273]
 [ -2.11379678 -41.09550132]]
```

In [35]:
```
#Select k best
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
selector=SelectKBest(score_func=chi2,k=3)
x_selected=selector.fit_transform(x,y)
print("Original data: ",x[:10])
print("Scaled data: ",x_selected[:10])
```

```
Original data:     BP  Cholesterol  Diabities  Result
0  30           31           8       1
1  53           36          38       0
2  18           24          21       1
3  72           43          61       0
4   8           32          44       1
5  41           14          27       0
6  62           17          69       1
7  25            3          52       0
8  49           34          14       1
9  33            8          81       0
Scaled data:  [[31  8  1]
 [36 38  0]
 [24 21  1]
 [43 61  0]
 [32 44  1]
 [14 27  0]
 [17 69  1]
 [ 3 52  0]
 [34 14  1]
 [ 8 81  0]]
```

In [43]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
text='''Version Control system allows multiple devleopers, designers and t
vectorizer=TfidfVectorizer()
x_tfidf=vectorizer.fit_transform([text])
print("Original data: ",text)
print("Scaled data: ",x_tfidf)
```

```
Original data:  Version Control system allows multiple devleopers, designers
and team members to work together on the same project. It helps them work sma
rter  and faster! a version control system is critical to ensure everyone has
the access to the latestbcpde and modification are tracked
Scaled data:    (0, 29) 0.12126781251816648
  (0, 3)        0.12126781251816648
  (0, 17)       0.12126781251816648
  (0, 15)       0.12126781251816648
  (0, 0)        0.12126781251816648
  (0, 11)       0.12126781251816648
  (0, 9)        0.12126781251816648
  (0, 8)        0.12126781251816648
  (0, 5)        0.12126781251816648
  (0, 13)       0.12126781251816648
  (0, 10)       0.12126781251816648
  (0, 22)       0.12126781251816648
  (0, 26)       0.12126781251816648
  (0, 12)       0.12126781251816648
  (0, 14)       0.12126781251816648
  (0, 20)       0.12126781251816648
  (0, 21)       0.12126781251816648
  (0, 25)       0.36380343755449945
  (0, 19)       0.12126781251816648
  (0, 28)       0.12126781251816648
  (0, 31)       0.24253562503633297
  (0, 27)       0.36380343755449945
  (0, 16)       0.12126781251816648
  (0, 24)       0.12126781251816648
  (0, 2)        0.36380343755449945
  (0, 6)        0.12126781251816648
  (0, 7)        0.12126781251816648
  (0, 18)       0.12126781251816648
  (0, 1)        0.12126781251816648
  (0, 23)       0.24253562503633297
  (0, 4)        0.24253562503633297
  (0, 30)       0.24253562503633297
```

In [7]:
```
#Label encoder
import pandas as pd
report=pd.read_csv("data.csv")
from sklearn.preprocessing import LabelEncoder
label_encoder=LabelEncoder()
report['Result']=label_encoder.fit_transform(report['Result'])
report['BP']=label_encoder.fit_transform(report['BP'])
x=report.drop(['Name','Age'],axis=1)
y=report['Result']
print(x)
print(y)
```

```
     BP  Cholesterol  Diabities  Result
0    30        228.8        143       1
1    53        237.6        234       0
2    18        219.3        176       1
3    72        248.5        321       0
4     8        230.2        267       1
..   ..          ...        ...     ...
395  12        196.2        271       1
396  58        232.1        199       0
397  22        190.6        352       0
398  67        221.7        232       1
399  13        238.0        169       1

[400 rows x 4 columns]
0      1
1      0
2      1
3      0
4      1
      ..
395    1
396    0
397    0
398    1
399    1
Name: Result, Length: 400, dtype: int32
```

In [ ]:
```
1
```