In [101]:
```
from sklearn.datasets import load_iris
import pandas as pd
import numpy as np
```

In [102]:
```
iris = load_iris()
```

In [103]:
```
data = pd.DataFrame(iris.data,columns = iris.feature_names)
```

In [104]:
```
data.head()
```

Out[104]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [105]:
```
from sklearn.preprocessing import StandardScaler
```

In [106]:
```
1 data
```

Out[106]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [107]:
```
1 scaler = StandardScaler()
```

In [108]:
```
1 data = scaler.fit_transform(data)
```

In [109]:
```
1 data = pd.DataFrame(data,columns = iris.feature_names)
```

In [110]:
```
1 data.describe()
```

Out[110]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	1.500000e+02	1.500000e+02	1.500000e+02	1.500000e+02
mean	-1.690315e-15	-1.842970e-15	-1.698641e-15	-1.409243e-15
std	1.003350e+00	1.003350e+00	1.003350e+00	1.003350e+00
min	-1.870024e+00	-2.433947e+00	-1.567576e+00	-1.447076e+00
25%	-9.006812e-01	-5.923730e-01	-1.226552e+00	-1.183812e+00
50%	-5.250608e-02	-1.319795e-01	3.364776e-01	1.325097e-01
75%	6.745011e-01	5.586108e-01	7.627583e-01	7.906707e-01
max	2.492019e+00	3.090775e+00	1.785832e+00	1.712096e+00

In [111]:
```
1 from sklearn.decomposition import PCA
```

In [112]:
```
1 pca = PCA(n_components = 2)
```

In [113]:
```
1 principal_components = pca.fit_transform(data)
```

In [114]:
```
1 data = pd.DataFrame(principal_components,columns = ["PC1","PC2"])
```

In [115]:
```
1 data
```

Out[115]:

	PC1	PC2
0	-2.264703	0.480027
1	-2.080961	-0.674134
2	-2.364229	-0.341908
3	-2.299384	-0.597395
4	-2.389842	0.646835
...	...	...
145	1.870503	0.386966
146	1.564580	-0.896687
147	1.521170	0.269069
148	1.372788	1.011254
149	0.960656	-0.024332

150 rows × 2 columns

In [116]:
```
1 pca.explained_variance_ratio_.sum()
```

Out[116]:
```
0.9581320720000164
```

In [117]:
```
1 data['Target'] = iris.target
```

In [118]:
```
1 data
```

Out[118]:

	PC1	PC2	Target
0	-2.264703	0.480027	0
1	-2.080961	-0.674134	0
2	-2.364229	-0.341908	0
3	-2.299384	-0.597395	0
4	-2.389842	0.646835	0
...	...	...	...
145	1.870503	0.386966	2
146	1.564580	-0.896687	2
147	1.521170	0.269069	2
148	1.372788	1.011254	2
149	0.960656	-0.024332	2

150 rows × 3 columns

In [119]:
```
1 from sklearn.model_selection import train_test_split
```

In [126]:
```
1 X_train,X_test,y_train,y_test = train_test_split(data.drop(['Target'],1),data['Target'],random_state=1)
```

In [127]:
```
1 from sklearn.linear_model import LogisticRegression
```

In [128]:
```
1 model1 = LogisticRegression()
```

In [129]:
```
model1.fit(X_train,y_train)
```

Out[129]:
```
LogisticRegression()
```

In [130]:
```
from sklearn.metrics import classification_report
```

In [131]:
```
print(classification_report(model1.predict(X_test),y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       0.94      1.00      0.97        15
           2       1.00      0.90      0.95        10

    accuracy                           0.97        38
   macro avg       0.98      0.97      0.97        38
weighted avg       0.98      0.97      0.97        38
```

Without Principal Components

In [132]:
```
data = pd.DataFrame(iris.data,columns = iris.feature_names)
```

In [133]:
```
data['Target'] = iris.target
```

In [134]:
```
from sklearn.model_selection import train_test_split
```

In [135]:
```
X_train,X_test,y_train,y_test = train_test_split(data.drop('Target',1),data['Target'],random_state=1)
```

In [136]:
```
1 model1.fit(X_train,y_train)
```

```
C:\Users\yashm\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:765: ConvergenceWarning: lbfgs failed to c
onverge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.h
tml)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modu
les/linear_model.html#logistic-regression)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[136]:
```
LogisticRegression()
```

In [137]:
```
1 print(classification_report(model1.predict(X_test),y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       0.94      1.00      0.97        15
           2       1.00      0.90      0.95        10

    accuracy                           0.97        38
   macro avg       0.98      0.97      0.97        38
weighted avg       0.98      0.97      0.97        38
```

In [ ]:
```
1
```

In [ ]:
```
1
```