In [28]:

```
# QUESTION - 1 - iris_01.csv
import pandas as pd

df = pd.read_csv("E:/5th sem/Elective 1/lab/iris_ass6.csv")
print(df)

df1 = df.drop("Unnamed: 0", axis=1)#column 1 just 1,2,3,4....unnamed bcs column header is missing
print(df1)

df1.isnull().sum()#total count of null values got

#mean
mean_sl = df1['sepal length'].mean()
mean_sl = round(mean_sl, 1)#round of by 1 decimal place
df1['sepal length'].fillna(value=mean_sl, inplace=True)

mean_sw = df1['sepal width (cm)'].mean()
mean_sw = round(mean_sw, 1)
df1['sepal width (cm)'].fillna(value=mean_sw, inplace=True)

mean_pl = df1['petal length (cm)'].mean()
mean_pl = round(mean_pl, 1)
df1['petal length (cm)'].fillna(value=mean_pl, inplace=True)

mean_pw = df1['petal width (cm)'].mean()
mean_pw = round(mean_pw, 1)
df1['petal width (cm)'].fillna(value=mean_pw, inplace=True)

print("Mean sepal length: ", mean_sl)
print("Mean sepal width: ", mean_sw)
print("Mean petal length: ", mean_pl)
print("Mean petal width: ", mean_pw)

print(df1)


#median
df2 = df.drop("Unnamed: 0", axis=1)

med_sl = df2['sepal length'].median()
med_sl = round(med_sl, 1)
```

```
df2['sepal length (cm)'].fillna(value=med_sl, inplace=True)

med_sw = df2['sepal width (cm)'].median()
med_sw = round(med_sw, 1)
df2['sepal width (cm)'].fillna(value=med_sw, inplace=True)

med_pl = df2['petal length (cm)'].median()
med_pl = round(med_pl, 1)
df2['petal length (cm)'].fillna(value=med_pl, inplace=True)

med_pw = df2['petal width (cm)'].median()
med_pw = round(med_pw, 1)
df2['petal width (cm)'].fillna(value=med_pw, inplace=True)

print("Median sepal length: ", med_sl)
print("Median sepal width: ", med_sw)
print("Median petal length: ", med_pl)
print("Median petal width: ", med_pw)

print(df2)


#mode
df3 = df.drop("Unnamed: 0", axis=1)

#Similar to iloc, in that both provide integer-based lookups.
#Use iat if you only need to get or set a single value in a DataFrame or Series.
#sirf mode nahi chalta

mode_sl = df3['sepal length'].mode().iat[0]
mode_sl = round(mode_sl, 1)
df3['sepal length (cm)'].fillna(value=mode_sl, inplace=True)

mode_sw = df3['sepal width (cm)'].mode().iat[0]
mode_sw = round(mode_sw, 1)
df3['sepal width (cm)'].fillna(value=mode_sw, inplace=True)

mode_pl = df3['petal length (cm)'].mode().iat[0]
mode_pl = round(mode_pl, 1)
df3['petal length (cm)'].fillna(value=mode_pl, inplace=True)

mode_pw = df3['petal width (cm)'].mode().iat[0]
mode_pw = round(mode_pw, 1)
df3['petal width (cm)'].fillna(value=mode_pw, inplace=True)

print("Mode sepal length: ", mode_sl)
print("Mode sepal width: ", mode_sw)
print("Mode petal length: ", mode_pl)
print("Mode petal width: ", mode_pw)

print(df3)

#zero value
df4 = df.drop("Unnamed: 0", axis=1)

df4['sepal length'].fillna(value=0, inplace=True)
df4['sepal width (cm)'].fillna(value=0, inplace=True)
df4['petal length (cm)'].fillna(value=0, inplace=True)
df4['petal width (cm)'].fillna(value=0, inplace=True)

print(df4)

#replace with maximum value
df5 = df.drop("Unnamed: 0", axis=1)

max_sl = df5['sepal length'].max()
df5['sepal length (cm)'].fillna(value=max_sl, inplace=True)

max_sw = df5['sepal width (cm)'].max()
df5['sepal width (cm)'].fillna(value=max_sw, inplace=True)

max_pl = df5['petal length (cm)'].max()
df5['petal length (cm)'].fillna(value=max_pl, inplace=True)

max_pw = df5['petal width (cm)'].max()
df5['petal width (cm)'].fillna(value=max_pw, inplace=True)

print("Maximum sepal length: ", max_sl)
print("Maximum sepal width: ", max_sw)
print("Maximum petal length: ", max_pl)
print("Maximum petal width: ", max_pw)

print(df5)
```

```
145            6.8             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               0.0               5.4               2.3
149           5.9               3.0               5.1               1.8

     Class  Predicted_class
0        0                0
1        0                0
2        0                0
3        0                0
4        0                0
..     ...              ...
145      2                2
146      2                2
147      2                2
148      2                0
149      2                2

[150 rows x 6 columns]
```

In [7]:

```
# QUESTION - 2
import pandas as pd
import numpy as np

df5 = pd.read_csv("E:/5th sem/Elective 1/lab/iris_ass6.csv")
print(df5)

iris_y = pd.DataFrame(df5, columns=['Class', 'Predicted_class'])
print(iris_y)

from sklearn.metrics import confusion_matrix
y_true = iris_y['Class']
y_pred = iris_y['Predicted_class']

cm = confusion_matrix(y_true, y_pred)
print("Confusion Matrix => ")
print(cm)

TP = np.diag(cm)
print("TRUE POSITIVE => ",TP)

FP = cm.sum(axis=0) - np.diag(cm)
print("FALSE POSITIVE => ",FP)

FN = cm.sum(axis=1) - np.diag(cm)
print("FALSE NEGATIVE => ",FN)

TN = cm.sum() - (FP + FN + TP)
print("TRUE NEGATIVE => ",TN)

TPR = TP/(TP+FN)
print("TRUE POSITIVE RATE => ",TPR)

TNR = TN/(TN+FP)
print("TRUE NEGATIVE RATE => ",TNR)

FPR = FP/(FP+TN)
print("FALSE POSITIVE RATE => ",FPR)

FNR = FN/(FN+TP)
```

```
print("FALSE NEGATIVE RATE => ",FNR)

#accuracy
ACC = (TP+TN)/(TP+FP+FN+TN)
print("ACCURACY => ",ACC)

#F1 and F beta score by code
precision = TP/TP+FP
recall = TP/TP+FN
print("precision: ",precision)
print("recall: ",recall)

F1 = (2*precision*recall)/(precision+recall)
print("By code F1 score value is: ",F1)

beta = 0.5
F_beta = ((1+(beta*beta))*(precision*recall))/((beta*beta*precision) + recall)
print("By code F beta score value is: ",F_beta)
```

```
     Unnamed: 0  sepal length  sepal width (cm)  petal length (cm)  \
0             0           5.1               3.5                1.4   
1             1           4.9               3.0                1.4   
2             2           4.7               3.2                1.3   
3             3           4.6               3.1                1.5   
4             4           5.0               3.6                1.4   
..          ...           ...               ...                ...   
145         145           0.0               3.0                5.2   
146         146           6.3               2.5                5.0   
147         147           6.5               3.0                5.2   
148         148           6.2               0.0                5.4   
149         149           5.9               3.0                5.1   

     petal width (cm)  Class  Predicted_class  
0                 0.2      0                0  
1                 0.2      0                0  
2                 0.2      0                0  
3                 0.2      0                0  
4                 0.2      0                0  
..                ...    ...              ...  
145               2.3      2                2  
```

```
146               1.9      2                2
147               2.0      2                2
148               2.3      2                0
149               1.8      2                2

[150 rows x 7 columns]
     Class  Predicted_class
0        0                0
1        0                0
2        0                0
3        0                0
4        0                0
..     ...              ...
145      2                2
146      2                2
147      2                2
148      2                0
149      2                2

[150 rows x 2 columns]
Confusion Matrix =>
[[46  2  2]
 [ 0 45  5]
 [ 2  4 44]]
TRUE POSITIVE =>  [46 45 44]
FALSE POSITIVE =>  [2 6 7]
FALSE NEGATIVE =>  [4 5 6]
TRUE NEGATIVE =>  [98 94 93]
TRUE POSITIVE RATE =>  [0.92 0.9  0.88]
TRUE NEGATIVE RATE =>  [0.98 0.94 0.93]
FALSE POSITIVE RATE =>  [0.02 0.06 0.07]
FALSE NEGATIVE RATE =>  [0.08 0.1  0.12]
ACCURACY =>  [0.96       0.92666667 0.91333333]
inbuilt F1 score:  0.9003839159426148
inbuilt F beta score for beta = 0.5:  0.9007939090258347
precision:  [3. 7. 8.]
recall:  [5. 6. 7.]
By code F1 score value is:  [3.75       6.46153846 7.46666667]
By code F beta score value is:  [3.26086957 6.77419355 7.77777778]
```

In [31]:

```

# QUESTION - 3

import pandas as pd
import numpy as np

df5 = pd.read_csv("E:/5th sem/Elective 1/lab/iris_ass6.csv")
print(df5)

#creating model

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(iris_x, iris_true, test_size=0.2, random_state=3)

from sklearn.linear_model import LogisticRegression
model = LogisticRegression(multi_class='ovr') #one-vs-rest use krke model banayae
model.fit(x_train, y_train)
#model created


from sklearn.metrics import confusion_matrix

y_true = y_test
y_pred = model.predict(x_test)

cm2 = confusion_matrix(y_true, y_pred)
print("Confusion Matrix => ")
print(cm2)

print("y testing was: ")
print(y_true)
print("y predcited is: ")
print(y_pred)

TP = np.diag(cm2)
FP = cm2.sum(axis=0) - np.diag(cm2)
FN = cm2.sum(axis=1) - np.diag(cm2)
TN = cm2.sum() - (FP + FN + TP)
print("TRUE POSITIVE => ", TP)
print("FALSE POSTITVE => ", FP)
print("FALSE NEGATIVE => ", FN)
print("TRUE NEGATIVE => ", TN)

TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("TRUE POSITIVE RATE => ", TPR)
print("TRUE NEGATIVE RATE => ", TNR)
print("FALSE POSITIVE RATE => ", FPR)
print("FALSE NEGATIVE RATE => ", FNR)

ACC = (TP+TN)/(TP+FP+FN+TN)
print("ACCURACY => ",ACC)

#F1 and F beta score by code
precision = TP/TP+FP
recall = TP/TP+FN
print("precision: ",precision)
print("recall: ",recall)

F1 = (2*precision*recall)/(precision+recall)
print("By code F1 score value is: ",F1)

beta = 0.5
F_beta = ((1+(beta*beta))*(precision*recall))/((beta*beta*precision) + recall)
print("By code F beta score value is: ",F_beta)


#same way for ques 6
```

```
     Unnamed: 0  sepal length  sepal width (cm)  petal length (cm)  \
0             0           5.1               3.5                1.4
1             1           4.9               3.0                1.4
2             2           4.7               3.2                1.3
3             3           4.6               3.1                1.5
4             4           5.0               3.6                1.4
..          ...           ...               ...                ...
145         145           0.0               3.0                5.2
146         146           6.3               2.5                5.0
147         147           6.5               3.0                5.2
148         148           6.2               0.0                5.4
```

```
149         149           5.9                3.0                5.1

     petal width (cm)  Class  Predicted_class
0                 0.2      0                0
1                 0.2      0                0
2                 0.2      0                0
3                 0.2      0                0
4                 0.2      0                0
..                ...    ...              ...
145               2.3      2                2
146               1.9      2                2
147               2.0      2                2
148               2.3      2                0
149               1.8      2                2

[150 rows x 7 columns]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-31-b1a4a79ae885> in <module>
     10 
     11 from sklearn.model_selection import train_test_split
---> 12 x_train, x_test, y_train, y_test = train_test_split(iris_x, iris_true, test_size=0.2, random_state=3)
     13 
     14 from sklearn.linear_model import LogisticRegression

NameError: name 'iris_x' is not defined
```

In [30]:

```
#Ques 4

new_feature=[]
new_feature_1=[]

import pandas as pd
empty = pd.DataFrame()

df = pd.read_csv("E:/5th sem/Elective 1/lab/iris_ass6.csv")
shape=df.shape
rows=shape[0]

feature=df["sepal length"]
max_val=feature.max()
min_val=feature.min()

feature_1=df["sepal width (cm)"]
max_val_1=feature_1.max()
min_val_1=feature_1.min()

for i in range(rows):
    normal=(feature[i]-min_val)/(max_val-min_val)
    new_feature.append(normal)

empty["feature_1"]=new_feature

for i in range(rows):
    normal=(feature_1[i]-min_val_1)/(max_val_1-min_val_1)
    new_feature_1.append(normal)

empty["feature_2"]=new_feature_1

empty.to_csv("hi.csv") #will be created at desktop
```

In [19]:

```
# QUESTION - 5

import pandas as pd
import numpy as np

df = pd.read_csv("E:/5th sem/Elective 1/lab/iris_ass6.csv")
print(df)

X=df[['sepal length','sepal width (cm)','petal length (cm)','petal width (cm)']]
Y=df[['Class']]

from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_scaled=scaler.fit_transform(X)
print(X_scaled)
Transposed=X_scaled.T
Covariance_matrix=np.cov(Transposed)
print(Covariance_matrix)
value,vector=np.linalg.eig(Covariance_matrix)
print(value)
print(vector)
percentage_values=[]
for i in range(len(value)):
    percentage_values.append(value[i]/np.sum(value))
print(percentage_values)
projected_1=X_scaled.dot(vector.T[0])
projected_2=X_scaled.dot(vector.T[1])
res=pd.DataFrame(projected_1,columns=['PC1'])
res['PC2']=projected_2
res["Species"]=Y
print(res)
```

```
     Unnamed: 0  sepal length  sepal width (cm)  petal length (cm)  \
0             0           5.1               3.5                1.4
1             1           4.9               3.0                1.4
2             2           4.7               3.2                1.3
3             3           4.6               3.1                1.5
4             4           5.0               3.6                1.4
..          ...           ...               ...                ...
145         145           0.0               3.0                5.2
146         146           6.3               2.5                5.0
```

```
147         147           6.5                 3.0                5.2
148         148           6.2                 0.0                5.4
149         149           5.9                 3.0                5.1

     petal width (cm)  Class  Predicted_class
0                 0.2      0                0
1                 0.2      0                0
2                 0.2      0                0
3                 0.2      0                0
4                 0.2      0                0
```

In [ ]: