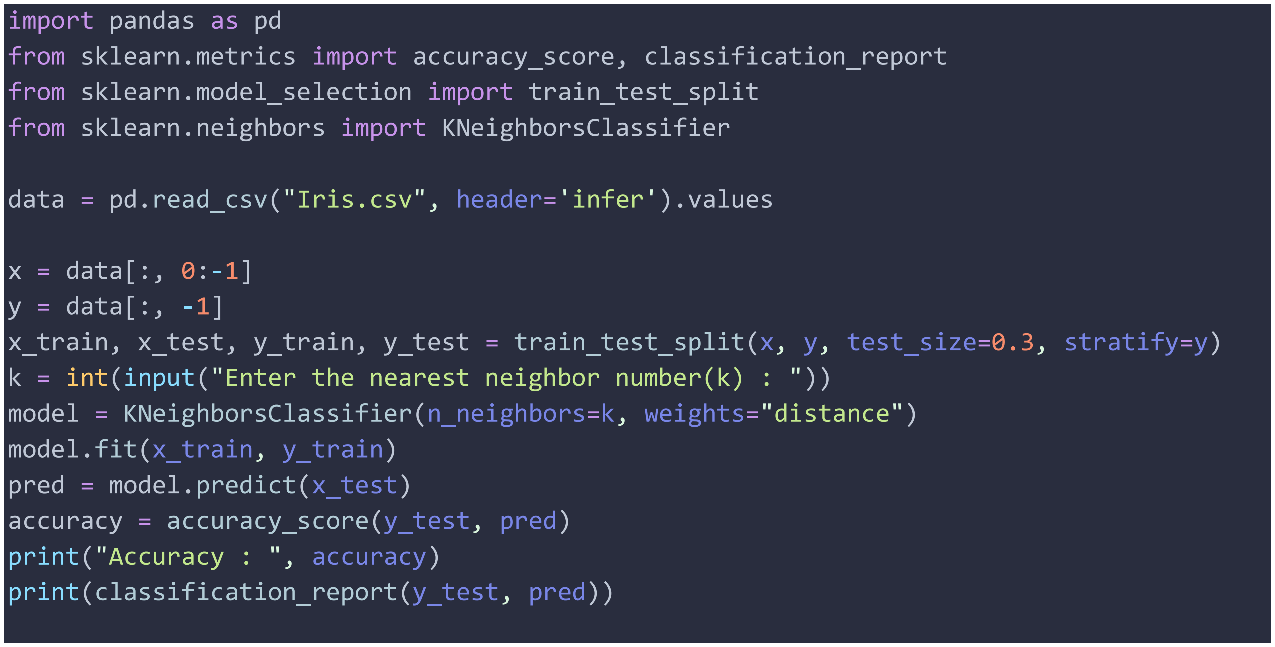
Name: Jaydeep Solanki

Roll: 22BEC059

Practical 6: Write a program for classifying iris images using a KNN classifier. Impement accuracy, precision, recall and f1-measure.

1. Using SkLearn Library.

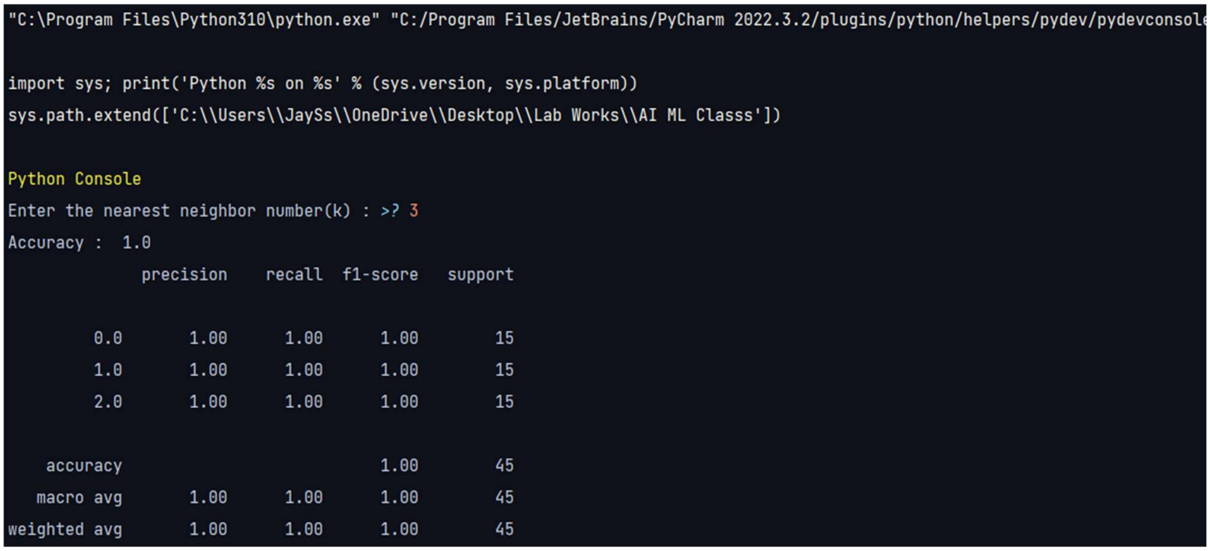


```
import pandas as pd
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

data = pd.read_csv("Iris.csv", header='infer').values

x = data[:, 0:-1]
y = data[:, -1]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, stratify=y)
k = int(input("Enter the nearest neighbor number(k) : "))
model = KNeighborsClassifier(n_neighbors=k, weights="distance")
model.fit(x_train, y_train)
pred = model.predict(x_test)
accuracy = accuracy_score(y_test, pred)
print("Accuracy : ", accuracy)
print(classification_report(y_test, pred))
```

Output:



```
"C:\Program Files\Python310\python.exe" "C:/Program Files/JetBrains/PyCharm 2022.3.2/plugins/python/helpers/pydev/pydevconsole

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['C:\\Users\\JaySs\\OneDrive\\Desktop\\Lab Works\\AI ML Classs'])

Python Console
Enter the nearest neighbor number(k) : >? 3
Accuracy :  1.0
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        15
         1.0       1.00      1.00      1.00        15
         2.0       1.00      1.00      1.00        15

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

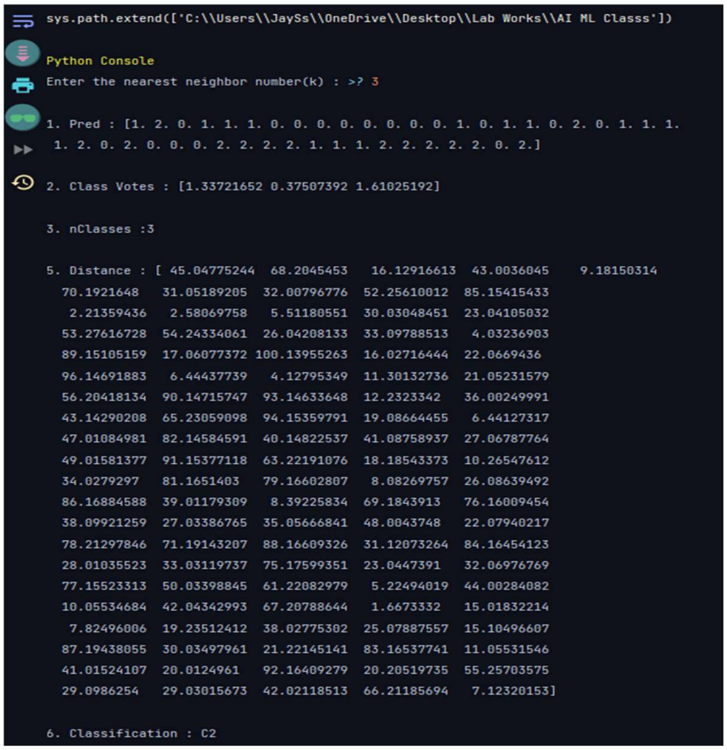
2. Without using SkLearn Library

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

data = pd.read_csv("Iris.csv", header='infer').values

x = data[:, 0:-1]
y = data[:, -1]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, stratify=y)
nClasses = np.unique(y_train).shape[0]
distance = np.zeros(shape=x_train.shape[0])
pred = np.zeros(shape=x_test.shape[0])
classVotes = np.zeros(shape=nClasses)
k = int(input("Enter the nearest neighbor number(k) : "))
for i in range(x_test.shape[0]):
    distance = np.sqrt(np.sum((x_train - x_test[i]) ** 2, axis=1))
    kMinIndex = np.argpartition(distance, k)[0:k]
    invDist = 1 / (distance + 10e-20)
    Denom = sum(invDist[kMinIndex])
    for j in range(k):
        classVotes[int(y_train[kMinIndex[j]])] += invDist[kMinIndex[j]]
    classVotes /= Denom
    pred[i] = np.argmax(classVotes)
print(f"""
1. Pred : {pred}\n
2. Class Votes : {classVotes}\n
3. nClasses :{nClasses}\n
5. Distance : {distance}\n
6. Classification : C{classVotes.tolist().index(max(classVotes.tolist()))}
""")
```

Output:



```
sys.path.extend(['C:\\Users\\JaySs\\OneDrive\\Desktop\\Lab Works\\AI ML Classs'])

Python Console
Enter the nearest neighbor number(k) : >? 3

1. Pred : [1. 2. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 1. 0. 2. 0. 1. 1. 1.
 1. 2. 0. 2. 0. 0. 0. 2. 2. 2. 2. 1. 1. 1. 2. 2. 2. 2. 2. 0. 2.]

2. Class Votes : [1.33721652 0.37507392 1.61025192]

3. nClasses :3

5. Distance : [ 45.04775244  68.2045453   16.12916613  43.0036045    9.18150314
  70.1921648   31.05189205  32.00796776  52.25610012  85.15415433
   2.21359436   2.58069758   5.51180551  30.03048451  23.04105032
  53.27616728  54.24334061  26.04208133  33.09788513   4.03236903
  89.15105159  17.06077372 100.13955263  16.02716444  22.0669436
  96.14691883   6.44437739   4.12795349  11.30132736  21.05231579
  56.20418134  90.14715747  93.14633648  12.2323342   36.00249991
  43.14290208  65.23059098  94.15359791  19.08664455   6.44127317
  47.01084981  82.14584591  40.14822537  41.08758937  27.06787764
  49.01581377  91.15377118  63.22191076  18.18543373  10.26547612
  34.0279297   81.1651403   79.16602807   8.08269757  26.08639492
  86.16884588  39.01179309   8.39225834  69.1843913   76.16009454
  38.09921259  27.03386765  35.05666841  48.0043748   22.07940217
  78.21297846  71.19143207  88.16609326  31.12073264  84.16454123
  28.01035523  33.03119737  75.17599351  23.0447391   32.06976769
  77.15523313  50.03398845  61.22082979   5.22494019  44.00284082
  10.05534684  42.04342993  67.20788644   1.6673332   15.01832214
   7.82496006  19.23512412  38.02775302  25.07887557  15.10496607
  87.19438055  30.03497961  21.22145141  83.16537741  11.05531546
  41.01524107  20.0124961   92.16409279  20.20519735  55.25703575
  29.0986254   29.03015673  42.02118513  66.21185694   7.12320153]

6. Classification : C2
```