BT-3172: Special Topics in Bioinformatics: Practical computing for bioinformatics
Lab 6: Machine learning in Bioinformatics.

W.M. Ayesha Sanahari | s13722 | 2017s16470

In this practical, you will learn how to use Python scikit-learn package to implement K-means and K-nearest neighbors (KNN) algorithms to learn from the popular iris data set.

After using PyCharm to write your scripts, **copy the codes to the appropriate space below the questions**. Also, submit the Python files separately so we can test them. Use the following format to name each script: YourIndexNo_PrimaryQuestion.py (submit two programs for the two questions)

1) K-means algorithm.
   In this problem, you will learn how to implement the K-means clustering algorithm on the famous iris data set (Fisher 1936). The iris data is available in the "iris.csv" file.
   I. Import the necessary packages and read the iris data file into a Pandas DataFrame.

   II. Implement the K-means algorithm using the scikit-learn package on **sepal length and sepal width** data. Pick the K value for the algorithm and explain the reason for the K value selection below. Print and write the sepal length and width values for the centroids below.

   K value is taken as 3 because it has mentioned three different varieties.

   kmeans cluster_centers: [[6.81276596 3.07446809]
   [5.77358491 2.69245283]
   [5.006 3.428]]

   III. A student has measured the sepal length, sepal width, petal length, and petal width of two species-unknown iris plants (measured in centimeters), which is given below in the above specified order.
   Plant1: 4.6, 3.0, 1.5, 0.2
   Plant 2: 6.2, 3.0,4.1,1.2
   He suspects that these plants fall into either one of the three iris species: *Iris setosa*, *Iris versicolour*, and *Iris virginica*. Use the K-means model you trained in the question (II) to predict the species of the two plants **using only the sepal length and width variables**. Write down the predicted labels below.

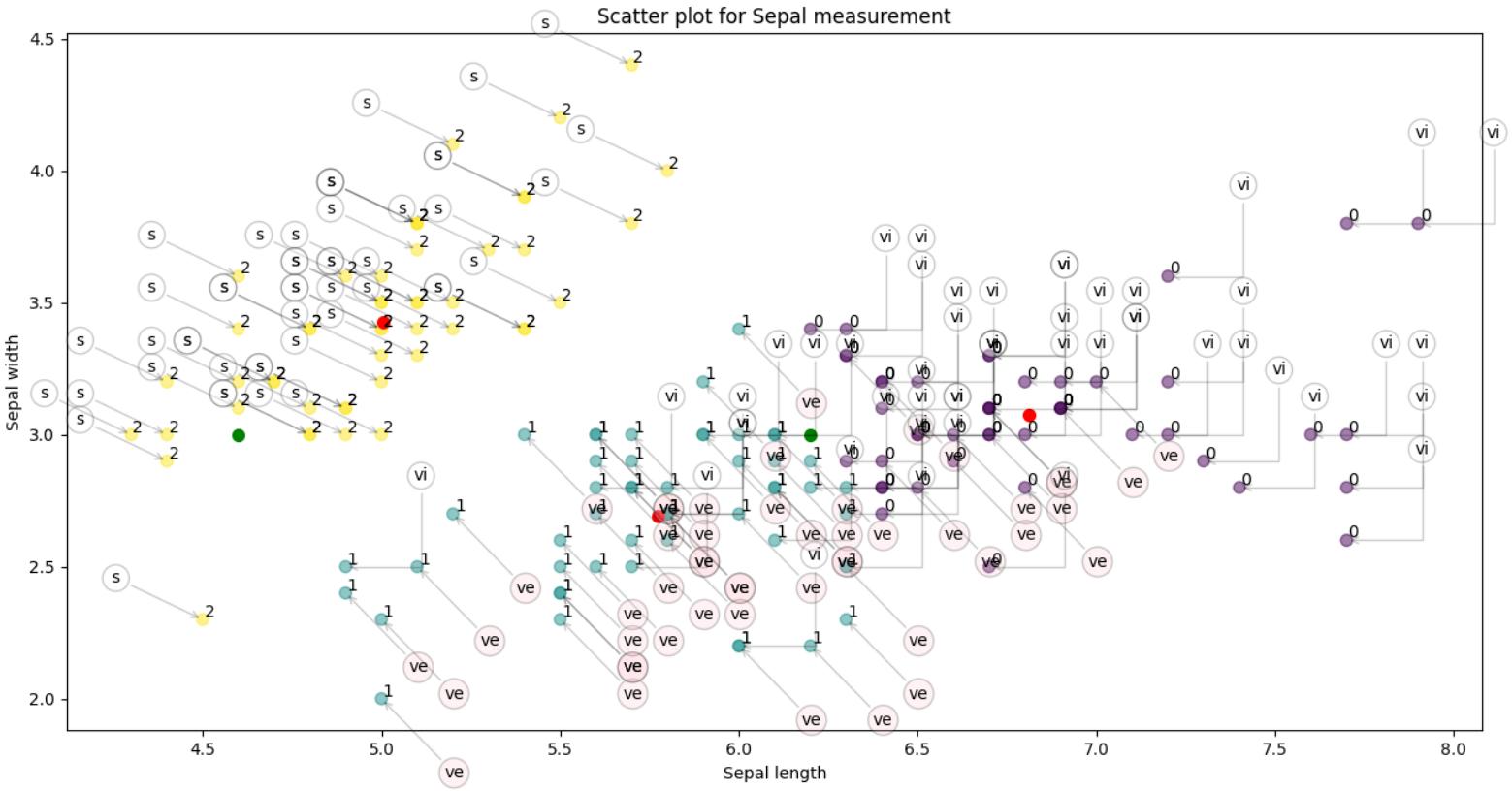
   predicted lables for sepals data: [2 1]

   plant 1 = setosa
   plant 2 = versicolour

   IV. Draw a scatter plot for the sepal measurement data. Represent each cluster in different colors (except red). Show the centroids of each cluster in red color and the predicted two new observations in another color. Show the cluster label for each data point except the centroids.
   Hint: use plt.annotate() function in matplotlib to label data points.

Show the original species labels in the scatter plot using an arrow to point at each data point. Use the v_short column to obtain the shortened labels. The species names and cluster labels should not overlap. Using these labels identify the species of the above predicted plant labels and write them for each plant in the space below.

according to scatter plot,
plant 1 - *Iris setosa*
plant 2 - *Iris versicolour*

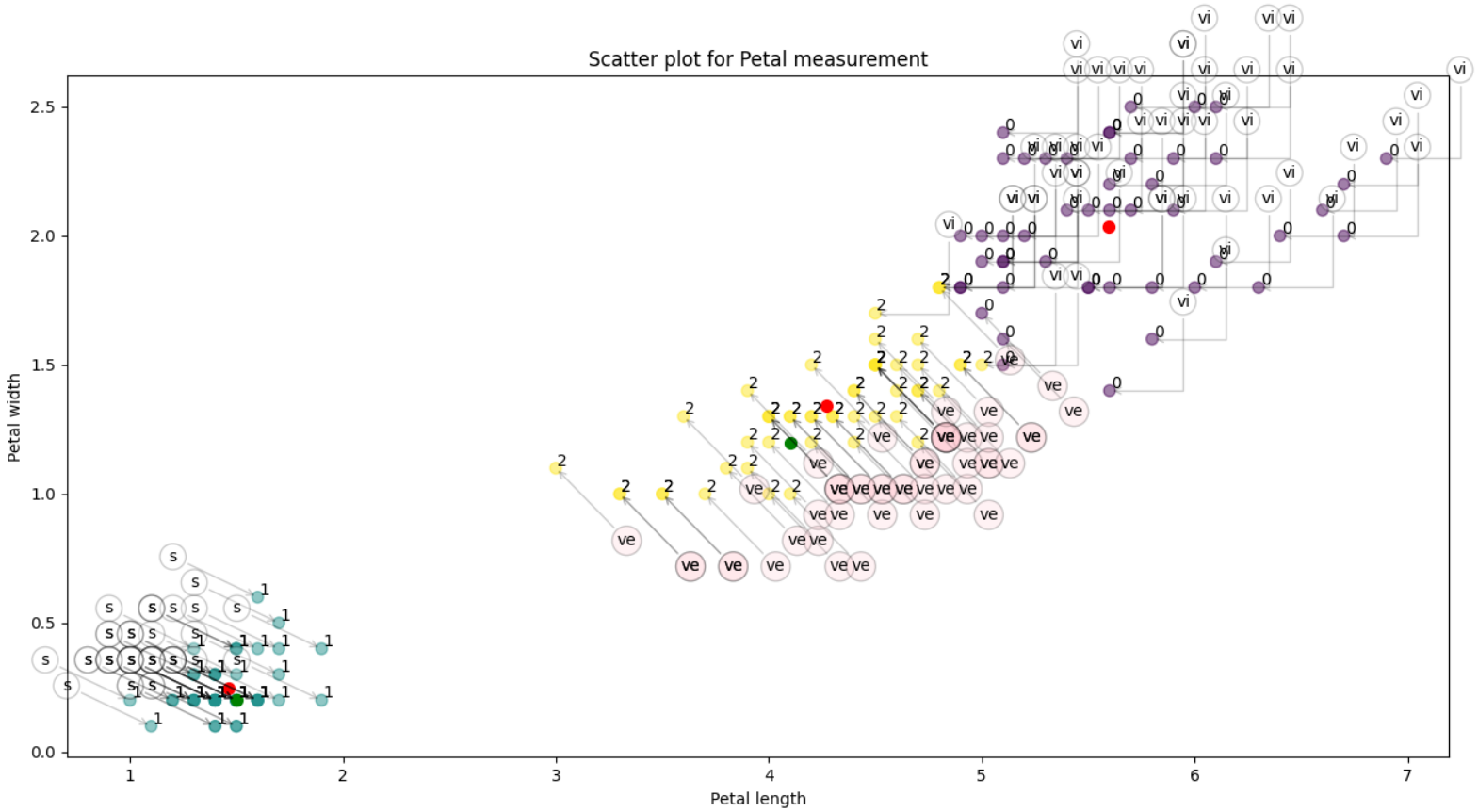


V. Now, implement the K-means algorithm using the scikit-learn package on petal length and petal width data. Use this trained model to predict the species of the two plants (question III) **using only the petal length and width variables**. Write down the predicted labels below.

predicted lables for petals data: [1 2]

VI. Draw a scatter plot for the petal measurement data. Represent each cluster in different colors (except red). Show the centroids of each cluster in red color and the predicted two new observations in another color. Show the cluster label for each data point except the centroids.
Hint: use plt.annotate() function in Matplotlib to label data points.
Show the original species labels in the scatter plot using an arrow to point at each data point. Use the v_short column to obtain the shortened labels. The species names and cluster labels should not overlap. Using these labels identify the species of the above predicted plant labels and write them for each plant in the space below.

according to scatter plot,
plant 1 - *Iris setosa*
plant 2 - *Iris versicolour*




```
"""
 Author : Ayesha Sanahari
 Date : 22/Jan/2021
 use Python scikit-learn package to implement K-means algorithms to learn from the popular iris data set.
 Input: iris.csv data file
 Output: scatter plot of iris data with predictions
"""
import numpy as np
from sklearn.cluster import KMeans
import pandas as pd
import matplotlib.pyplot as plt

#reading data from pandas
iris=pd.read_csv('iris.csv')
#define label column
df2=iris['v_short']

# Add sepal data columns to Dataframe
df=pd.DataFrame(iris,columns=['sepal.length','sepal.width'])
df3=pd.DataFrame(iris,columns=['v_short'])
```

```
kmeans= KMeans(n_clusters=3).fit(df)
# getting centroids
centroids=kmeans.cluster_centers_
print("kmeans cluster_centers:", centroids)

# specise unknown data sample
plant=np.array([[4.6, 3.0, 1.5, 0.2],[6.2, 3.0, 4.1, 1.2]])
X = plant[:, [0, 1]]

# Predicting the lables of new inserted data
predicted=kmeans.predict(X)
print("predicted lables for sepals data:", predicted)
labels=kmeans.labels_
print("labels:", labels)

#scatter plot of sepal length ,sepal width ,unknown saple data with cetroids
plt.scatter(df['sepal.length'],df['sepal.width'],c=kmeans.labels_.astype(float),s=50,alpha=
0.5)
plt.scatter(X[:,0],X[:,1],c='green',s=50)
plt.scatter(centroids[:,0],centroids[:,1],c='red',s=50)
plt.title("Scatter plot for Sepal measurement")
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')


#annotation
for df2, x, y,z in zip(df2, df.iloc[:, 0], df.iloc[:, 1],df3.iloc[:,0]):
    if z=='s':
        plt.annotate(
            df2,
            xy=(x, y), xytext=(-50, 20),
            textcoords='offset points', ha='right', va='bottom',
            bbox=dict(boxstyle='circle,pad=0.3',fc="white", alpha=0.2),
            arrowprops=dict(arrowstyle = '->', connectionstyle="arc,rad=40",alpha=0.2))
    if z == 've':
        plt.annotate(
            df2,
            xy=(x, y), xytext=(50, -50),
            textcoords='offset points', ha='right', va='bottom',
            bbox=dict(boxstyle='circle,pad=0.3', fc="pink", alpha=0.2),
            arrowprops=dict(arrowstyle='->', connectionstyle="arc,rad=40", alpha=0.2))
    if z == 'vi':
        plt.annotate(
            df2,
            xy=(x, y), xytext=(50, 50),
            textcoords='offset points', ha='right', va='bottom',
            bbox=dict(boxstyle='circle,pad=0.3', fc="white", alpha=0.2),
            arrowprops=dict(arrowstyle='->', connectionstyle="angle,angleA=-
90,angleB=180,rad=0", alpha=0.2))
```

```
#label annotation
for label, x, y in zip(labels, df.iloc[:, 0], df.iloc[:, 1]):
  plt.annotate(
      label,
      xy=(x, y), xytext=(1, 1.5),
      textcoords='offset points')
plt.show()

print("for the petals data")
#define label collumns
df2=iris['v_short']

# Add petal data columns to Dataframe
df4=pd.DataFrame(iris,columns=['petal.length','petal.width'])

kmeans= KMeans(n_clusters=3).fit(df4)
# getting centroids
centroids=kmeans.cluster_centers_
print("kmeans cluster_centers:", centroids)

# specise unknown data sample
plant=np.array([[4.6, 3.0, 1.5, 0.2],[6.2, 3.0, 4.1, 1.2]])
X = plant[:, [2, 3]]

# Predicting the lables of new inserted data
predicted=kmeans.predict(X)
print("predicted lables for petals data:", predicted)
labels=kmeans.labels_

#scatter plot of sepal length ,sepal width ,unknown saple data with cetroids
plt.scatter(df4['petal.length'],df4['petal.width'],c=kmeans.labels_.astype(float),s=50,alph
a=0.5)
plt.scatter(X[:,0],X[:,1],c='green',s=50)
plt.scatter(centroids[:,0],centroids[:,1],c='red',s=50)
plt.title("Scatter plot for Petal measurement")
plt.xlabel('Petal length')
plt.ylabel('Petal width')


#annotation
for df2, x, y,z in zip(df2, df4.iloc[:, 0], df4.iloc[:, 1],df3.iloc[:,0]):
   if z=='s':
      plt.annotate(
         df2,
         xy=(x, y), xytext=(-50, 20),
         textcoords='offset points', ha='right', va='bottom',
         bbox=dict(boxstyle='circle,pad=0.3',fc="white", alpha=0.2),
         arrowprops=dict(arrowstyle = '->', connectionstyle="arc,rad=40",alpha=0.2))
   if z == 've':
```

```
        plt.annotate(
            df2,
            xy=(x, y), xytext=(50, -50),
            textcoords='offset points', ha='right', va='bottom',
            bbox=dict(boxstyle='circle,pad=0.3', fc="pink", alpha=0.2),
            arrowprops=dict(arrowstyle='->', connectionstyle="arc,rad=40", alpha=0.2))
    if z == 'vi':
        plt.annotate(
            df2,
            xy=(x, y), xytext=(50, 50),
            textcoords='offset points', ha='right', va='bottom',
            bbox=dict(boxstyle='circle,pad=0.3', fc="white", alpha=0.2),
            arrowprops=dict(arrowstyle='->', connectionstyle="angle,angleA=-
90,angleB=180,rad=0", alpha=0.2))

    #label annotation
for label, x, y in zip(labels, df4.iloc[:, 0], df4.iloc[:, 1]):
    plt.annotate(
        label,
        xy=(x, y), xytext=(1, 1.5),
        textcoords='offset points')
plt.show()
```

2) K-nearest neighbors (KNN) algorithm.

I. Now, train the iris data set you used for the question 1 using the KNN algorithm. However, for this instance, do not use the CSV file to read the data, instead use the bundled iris data set in the scikit-learn package by importing the datasets sub module. Load the iris data and extract the labels. Then, standardize the data using the StandardScaler class in the preprocessing sub module. Also, read the Plant 1 data record in question 1 (III) as a test data record and standardize it using the same transform model used for standardizing the iris data set. Furthermore, create a 2-d array for the two data records (Plant 1 and Plant 2) and standardize the array as before.

II. First, train the KNN model using **only the sepal measurements**.

i. Extract the standardized sepal data and train the KNN model to find the two nearest neighbors of the test data record (question I). Find the two-nearest neighbors and print the indices, data values, labels, and species of the two-nearest neighbors.

```
Nearest Neightbours:
index: 3      coordinates: [-1.50652052  0.09821729]     label: 0    species: setosa
index: 38     coordinates: [-1.74885626 -0.13197948]     label: 0    species: setosa
```

ii. Then, train the model again for 5-nearest neighbors for standardized sepal data and predict the probability of each plant categorizing into each species and write them below. Further, predict the class labels for the two plants and write the labels and the species predicted from the KNN algorithm.

```
Predictions:
Plant1 -    0   setosa        (probability: [1. 0. 0.])
Plant2 -    2   virginica     (probability: [0.  0.4 0.6])
```

III. Train the KNN model using **only the petal measurements**.

i. Extract the standardized petal data and train the KNN model to find the two nearest neighbors of the test data record (question I). Find the two-nearest neighbors and print the indices, data values, labels, and species of the two-nearest neighbors.

```
Nearest Neightbours:
index: 27       coordinates: [-1.2833891 -1.3154443]       label: 0     species: setosa
index: 3        coordinates: [-1.2833891 -1.3154443]       label: 0     species: setosa
```

ii. Then, train the model again for 5-nearest neighbors for standardized petal data and predict the probability of each plant categorizing into each species and write them below. Further, predict the class labels for the two plants and write the labels and the species predicted from the KNN algorithm using only the petal measurements.

```
Predictions:
Plant1 -    0    setosa       (probability: [1. 0. 0.])
Plant2 -    1    versicolor   (probability: [0. 1. 0.])
```

IV. Train the KNN model using **both the sepal and petal measurements**.

i. Train the KNN model for all the standardized measurement data to find the two nearest neighbors of the test data record (question I). Find the two-nearest neighbors and print the indices, data values, labels, and species of the two-nearest neighbors.

```
Nearest Neightbours:
index: 3    coordinates: [-1.50652052  0.09821729 -1.2833891  -1.3154443 ]  label: 0   species: setosa
index: 38   coordinates: [-1.74885626 -0.13197948 -1.39706395 -1.3154443 ]  label: 0   species: setosa
```

ii. Then, train the model again for 5-nearest neighbors for all measurement data and predict the probability of each plant categorizing into each species and write them below. Further, predict the class labels for the two plants and write the labels and the species predicted from the KNN algorithm.

```
Predictions:
Plant1 -    0    setosa       (probability: [1. 0. 0.])
Plant2 -    1    versicolor   (probability: [0. 1. 0.])
```

```
"""
 Author : Ayesha Sanahari
 Date : 22/Jan/2021
 Use Python scikit-learn package to implement K-nearest neighbors(KNN) algorithm to
learn from the popular iris data set.

"""
import numpy as np
from sklearn.datasets import load_iris
from sklearn import preprocessing
```

```
from sklearn.neighbors import KNeighborsClassifier

# Loading dataset into variables
iris = load_iris()
irisData = iris['data']
irisTargets = iris['target']
irisSpp = iris['target_names']

#scaling iris data
scalar = preprocessing.StandardScaler().fit(irisData)
scaledIris = scalar.transform(irisData)

#plant 1 data and scaling
plant1=np.array([[4.6, 3.0, 1.5, 0.2]])
scaledpalnt1= scalar.transform(plant1)

#both plants data and scaling
plants = np.array([[4.6, 3.0, 1.5, 0.2], [6.2, 3.0, 4.1, 1.2]])
scaledpalnts= scalar.transform(plants)

#selecting sepal collumn
sepal=scaledIris[:, :2]
testDataplant1 = scaledpalnt1[:, :2]
testDataPlants = scaledpalnts[:, :2]


#for sepal data
#KNeighborsClassifier for 2 neighbors
classifier = KNeighborsClassifier(n_neighbors=2).fit(sepal, irisTargets)

#print stats
for test in testDataplant1:
   print("Predicted species:")
   for i in classifier.predict(testDataplant1):
      print(irisSpp[i], end='\n')
   print("Nearest Neightbours:")
   for n in classifier.kneighbors([test], 2)[1][0]:
      print("index:", n, end='\t')
      print("coordinates:", sepal[n], end='\t')
      tmp =  classifier.predict([sepal[n]])[0]
      print('label:', tmp, end='\t')
      print('species:', irisSpp[tmp])

##KNeighborsClassifier for 5 neighbors
print("for 5 neighbours")
classifier = KNeighborsClassifier(n_neighbors=5).fit(sepal, irisTargets)
for test in testDataPlants:
   print("Predicted species:")
   for i in classifier.predict(testDataPlants):
      print(irisSpp[i], end='\n')
```

```
    print("Nearest Neightbours:")
    for n in classifier.kneighbors([test], 5)[1][0]:
        print("index:", n, end='\t')
        print("coordinates:", sepal[n], end='\t')
        tmp =  classifier.predict([sepal[n]])[0]
        print('label:', tmp, end='\t')
        print('species:', irisSpp[tmp])

#predictions
print("Predictions:")
for pred_species, pred_probability in zip(classifier.predict(testDataPlants),
                                          classifier.predict_proba(testDataPlants)):
    print(pred_species, end='\t')
    print(irisSpp[pred_species], end='\t')
    print(f'(probability: {pred_probability})')


#for petal data
print("\n\n for petal data")
#select petal data column
testDataplant1 = scaledpalnt1[:, 2:]
testDataPlants = scaledpalnts[:, 2:]
petal = scaledIris[:, 2:]
print("with 2 n_neighbors ")

#KNeighborsClassifier for 2 neighbors
classifier = KNeighborsClassifier(n_neighbors=2).fit(petal, irisTargets)

#print result
for test in testDataplant1:
    print("Predicted species:")
    for i in classifier.predict(testDataplant1):
        print(irisSpp[i], end='\n')
    print("Nearest Neightbours:")
    for n in classifier.kneighbors([test], 2)[1][0]:
        print("index:", n, end='\t')
        print("coordinates:", petal[n], end='\t')
        tmp =  classifier.predict([petal[n]])[0]
        print('label:', tmp, end='\t')
        print('species:', irisSpp[tmp])

print("for 5 neighbours")
#KNeighborsClassifier for 2 neighbors and both plants
classifier = KNeighborsClassifier(n_neighbors=5).fit(petal, irisTargets)

#print result
for test in testDataPlants:
    print("Predicted species:")
    for i in classifier.predict(testDataPlants):
        print(irisSpp[i], end='\n')
```

```
    print("Nearest Neightbours:")
    for n in classifier.kneighbors([test], 5)[1][0]:
        print("index:", n, end='\t')
        print("coordinates:", petal[n], end='\t')
        tmp =  classifier.predict([petal[n]])[0]
        print('label:', tmp, end='\t')
        print('species:', irisSpp[tmp])

#predictions
print("Predictions:")
testDataplant1 = scaledpalnt1[:, 2:]
for pred_species, pred_probability in zip(classifier.predict(testDataPlants),
                                          classifier.predict_proba(testDataPlants)):
    print(pred_species, end='\t')
    print(irisSpp[pred_species], end='\t')
    print(f'(probability: {pred_probability})')

#using both petal and sepal data
print("\n\n for sepal and petal data")
testDataPlants = scaledpalnts[:, :]
data = scaledIris[:, :]
testDataplant1 = scaledpalnt1[:, :]
print("with 2 n_neighbors ")

#KNeighborsClassifier for all data
classifier = KNeighborsClassifier(n_neighbors=2).fit(data, irisTargets)

#print result
for test in testDataplant1:
    print("Predicted species:")
    for i in classifier.predict(testDataplant1):
        print(irisSpp[i], end='\n')
    print("Nearest Neightbours:")
    for n in classifier.kneighbors([test], 2)[1][0]:
        print("index:", n, end='\t')
        print("coordinates:", data[n], end='\t')
        tmp =  classifier.predict([data[n]])[0]
        print('label:', tmp, end='\t')
        print('species:', irisSpp[tmp])

#KNeighborsClassifier for all data with 5 n_neighbors
print("for 5 neighbours")
classifier = KNeighborsClassifier(n_neighbors=5).fit(data, irisTargets)
for test in testDataPlants:
    print("Predicted species:")
    for i in classifier.predict(testDataPlants):
        print(irisSpp[i], end='\n')
    print("Nearest Neightbours:")
    for n in classifier.kneighbors([test], 5)[1][0]:
        print("index:", n, end='\t')
```

```
        print("coordinates:", data[n], end='\t')
        tmp =  classifier.predict([data[n]])[0]
        print('label:', tmp, end='\t')
        print('species:', irisSpp[tmp])


print("Predictions:")
for pred_species, pred_probability in zip(classifier.predict(testDataPlants),
                                          classifier.predict_proba(testDataPlants)):
    print(pred_species, end='\t')
    print(irisSpp[pred_species], end='\t')
    print(f'(probability: {pred_probability})')
```

References

- Fisher, Ronald A. "The use of multiple measurements in taxonomic problems." *Annals of eugenics* 7.2 (1936): 179-188.