**S13674**

**D.P.Bhagya Hendalage**

**BT-3172: Special Topics in Bioinformatics: Practical computing for bioinformatics**

**Lab 6: Machine learning in Bioinformatics.**

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.

* 1. Import the necessary packages and read the iris data file into a Pandas DataFrame.
  2. 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 3, there are three different species present in the dataset

Sepal length and width values for the centroids

[[6.81276596 3.07446809]

[5.006 3.428 ]

[5.77358491 2.69245283]]

* 1. 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

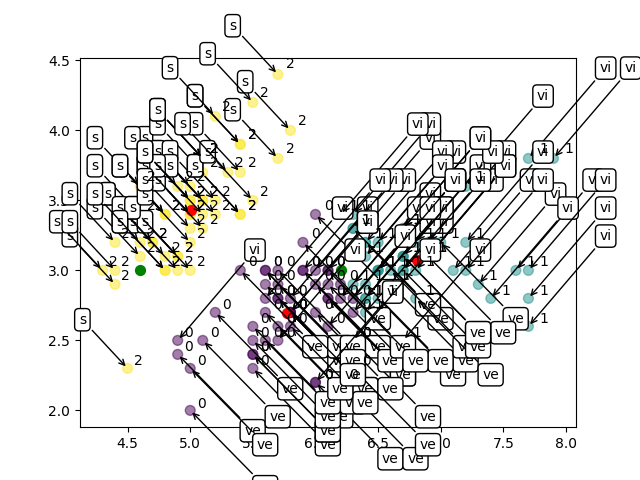
He suspects that these plants fall into either one of the three iris species: *Iris setosa*, *Iris versicolour*, and *Iris virginica*. Use the K-means model you trained in the question (II) to predict the species of the two plants **using only the sepal length and width variables**. Write down the predicted labels below.

[0 2] Plant1 *Iris setosa* Plant 2 *Iris versicolour*

* 1. 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.

**

Plant1: *Iris setosa*

Plant2 : *Iris versicolour*

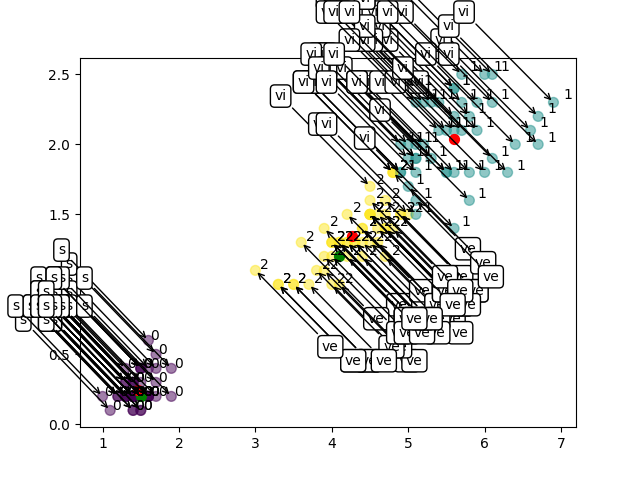
* 1. Now, implement the K-means algorithm using the scikit-learn package on petal length and petal width data. Use this trained model to predict the species of the two plants (question III) **using only the petal length and width variables**. Write down the predicted labels below.

[1 1] plant 1 and 2 are *Iris setosa*

* 1. 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.



plant 1 = *Iris setosa*

plant 2 = *Iris versicolour*

**import** pandas **as** pd  
**from** sklearn.cluster **import** KMeans  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
*# read iris dataset*iris\_data = pd.read\_csv(**'iris.csv'**)  
*#print(iris\_data)  
  
# get sepal length and width*sep\_len\_wid = pd.DataFrame(iris\_data, columns=[**'sepal.length'**, **'sepal.width'**])  
v\_label=pd.DataFrame(iris\_data,columns=[**'v\_short'**])  
print(v\_label)  
*#print(sep\_len\_wid)  
# labels of dataset*labels = iris\_data[**'v\_short'**]  
*#print(labels)  
#Kmeans test*kmeans=KMeans(n\_clusters=3).fit(sep\_len\_wid)  
*#getting cetroids*centrodis=kmeans.cluster\_centers\_  
*#print(centrodis)*lable=kmeans.labels\_  
print(lable)  
  
*# unknown data sample*unknown=np.array([[4.6, 3.0, 1.5, 0.2],[6.2, 3.0,4.1,1.2]])  
new\_x = unknown[:, [0, 1]]  
predicted=kmeans.predict(new\_x)  
print(predicted)  
*#scatter plot of sepal length ,sepal width ,unknown sapal data with cetroids*plt.scatter(sep\_len\_wid[**'sepal.length'**],sep\_len\_wid[**'sepal.width'**],c=kmeans.labels\_.astype(float),s=50,alpha=0.5)  
*#plt.show()  
# unknown data*plt.scatter(unknown[:,0],unknown[:,1],c=**'green'**,s=50)  
plt.scatter(centrodis[:,0],centrodis[:,1],c=**'red'**,s=50)  
*#plt.show()  
#annotation of v\_short labels***for** labels, x, y,z **in** zip(labels, sep\_len\_wid.iloc[:, 0], sep\_len\_wid.iloc[:, 1],v\_label.iloc[:,0]):  
 **if** z==**'s'**:  
 plt.annotate(  
 labels,  
 xy=(x, y),  
 xytext=(-30, 30), textcoords=**"offset points"**,  
 va=**"bottom"**, ha=**"right"**,  
 bbox=dict(boxstyle=**"round"**, fc=**"w"**),  
 arrowprops=dict(arrowstyle=**"->"**))  
 **elif** z == **'ve'**:  
 plt.annotate(  
 labels,  
 xy=(x, y),  
 xytext=(60, -60), textcoords=**"offset points"**,  
 va=**"bottom"**, ha=**"right"**,  
 bbox=dict(boxstyle=**"round"**, fc=**"w"**),  
 arrowprops=dict(arrowstyle=**"->"**))  
  
 **else**:  
 plt.annotate(  
 labels,  
 xy=(x, y),  
 xytext=(60, 60),  
 textcoords=**"offset points"**,  
 va=**"bottom"**, ha=**"right"**,  
 bbox=dict(boxstyle=**"round"**, fc=**"w"**),  
 arrowprops=dict(arrowstyle=**"->"**))  
  
*#label annotation***for** label, x, y **in** zip(lable, sep\_len\_wid.iloc[:, 0], sep\_len\_wid.iloc[:, 1]):  
 plt.annotate(  
 label,  
 xy=(x, y),  
 textcoords=**'offset points'** )  
plt.show()  
print(**"for the petals data"**)  
*#define label columns*labels=iris\_data[**'v\_short'**]  
  
*##add petal data to data frame*pet\_len\_wid=pd.DataFrame(iris\_data,columns=[**'petal.length'**,**'petal.width'**])  
v\_label=pd.DataFrame(iris\_data,columns=[**'v\_short'**])  
  
*#kmeans test for petal*kmeans = KMeans(n\_clusters=3, random\_state=0).fit(pet\_len\_wid)  
  
*#getting cetroids*centrodis=kmeans.cluster\_centers\_  
predicted=kmeans.predict(new\_x)  
print(predicted)  
  
*#define labels*label2=kmeans.labels\_  
print(labels)  
  
*#unknown sample petal data*new\_x = unknown[:, [2, 3]]  
  
*#scatter plot of petal length ,petal width ,unknown sapae data with cetroids*plt.scatter(pet\_len\_wid[**'petal.length'**],pet\_len\_wid[**'petal.width'**],c=kmeans.labels\_.astype(float),s=50,alpha=0.5)  
plt.scatter(new\_x[:,0],new\_x[:,1],c=**'green'**,s=50)  
plt.scatter(centrodis[:,0],centrodis[:,1],c=**'red'**,s=50)  
*#plt.show()  
#annotation from v\_short***for** labels, x, y,z **in** zip(labels, pet\_len\_wid.iloc[:, 0], pet\_len\_wid.iloc[:,1],v\_label.iloc[:,0]):  
 **if** z==**'s'**:  
 plt.annotate(  
 labels,  
 xy=(x, y), xytext=(-60, 60),  
 textcoords=**"offset points"**,  
 va=**"bottom"**, ha=**"right"**,  
 bbox=dict(boxstyle=**"round"**, fc=**"w"**),  
 arrowprops=dict(arrowstyle=**"->"**))  
 **elif** z == **'ve'**:  
 plt.annotate(  
 labels,  
 xy=(x, y), xytext=(60, -60),  
 textcoords=**"offset points"**,  
 va=**"bottom"**, ha=**"right"**,  
 bbox=dict(boxstyle=**"round"**, fc=**"w"**),  
 arrowprops=dict(arrowstyle=**"->"**))  
 **else**:  
 plt.annotate(  
 labels,  
 xy=(x, y), xytext=(-60, 60),  
 textcoords=**"offset points"**,  
 va=**"bottom"**, ha=**"right"**,  
 bbox=dict(boxstyle=**"round"**, fc=**"w"**),  
 arrowprops=dict(arrowstyle=**"->"**))  
*#annotation of labels***for** label, x, y **in** zip(label2, pet\_len\_wid.iloc[:, 0], pet\_len\_wid.iloc[:, 1]):  
 plt.annotate(  
 label,  
 xy=(x, y),  
 textcoords=**'offset points'**)  
  
plt.show()

1. K-nearest neighbors (KNN) algorithm.
   1. 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.
   2. First, train the KNN model using **only the sepal measurements**.
      1. 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.

for sepal data

for 2 neighbors

species prediction:

setosa

2-Nearest Neightbours:

index: 3 data value: [-1.50652052 0.09821729] label: 0 species: setosa

index: 38 data value: [-1.74885626 -0.13197948] label: 0 species: setosa

* + 1. 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 for sepal data:

label 0 species setosa (probability: [1. 0. 0.])

label 2 species virginica (probability: [0. 0.4 0.6])

for 5 neighbours

species prediction:

setosa

virginica

5-Nearest Neightbours:

index: 3 data value: [-1.50652052 0.09821729] label: 0 species: setosa

index: 38 data value: [-1.74885626 -0.13197948] label: 0 species: setosa

index: 12 data value: [-1.26418478 -0.13197948] label: 0 species: setosa

index: 45 data value: [-1.26418478 -0.13197948] label: 0 species: setosa

index: 8 data value: [-1.74885626 -0.36217625] label: 0 species: setosa

species prediction:

setosa

virginica

5-Nearest Neightbours:

index: 91 data value: [ 0.31099753 -0.13197948] label: 2 species: virginica

index: 127 data value: [ 0.31099753 -0.13197948] label: 2 species: virginica

index: 97 data value: [ 0.4321654 -0.36217625] label: 1 species: versicolor

index: 138 data value: [ 0.18982966 -0.13197948] label: 2 species: virginica

index: 103 data value: [ 0.55333328 -0.36217625] label: 1 species: versicolor

* 1. Train the KNN model using **only the petal measurements**.
     1. 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.

for petal data

for 2 neighbors

species prediction:

setosa

2-Nearest Neightbours:

index: 43 data value: [-1.22655167 -0.78891556] label: 0 species: setosa

index: 23 data value: [-1.16971425 -0.92054774] label: 0 species: setosa

* + 1. 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 for petal data:

label 0 species setosa (probability: [1. 0. 0.])

label 1 species versicolor (probability: [0. 1. 0.])

for 5 neighbours

species prediction:

setosa

versicolor

5-Nearest Neightbours:

index: 34 data value: [-1.2833891 -1.3154443] label: 0 species: setosa

index: 39 data value: [-1.2833891 -1.3154443] label: 0 species: setosa

index: 3 data value: [-1.2833891 -1.3154443] label: 0 species: setosa

index: 27 data value: [-1.2833891 -1.3154443] label: 0 species: setosa

index: 7 data value: [-1.2833891 -1.3154443] label: 0 species: setosa

species prediction:

setosa

versicolor

5-Nearest Neightbours:

index: 92 data value: [0.13754657 0.00087755] label: 1 species: versicolor

index: 95 data value: [0.25122143 0.00087755] label: 1 species: versicolor

index: 82 data value: [0.08070915 0.00087755] label: 1 species: versicolor

index: 88 data value: [0.194384 0.13250973] label: 1 species: versicolor

index: 99 data value: [0.194384 0.13250973] label: 1 species: versicolor

* 1. Train the KNN model using **both the sepal and petal measurements**.
     1. 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.

for sepal and petal data

for 2 neighbors

species prediction:

setosa

2-Nearest Neightbours:

index: 3 data value: [-1.50652052 0.09821729 -1.2833891 -1.3154443 ] label: 0 species: setosa

index: 38 data value: [-1.74885626 -0.13197948 -1.39706395 -1.3154443 ] label: 0 species: setosa

* + 1. 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 for sepal and petal data:

label: 0 species: setosa (probability: [1. 0. 0.])

label: 1 species: versicolor (probability: [0. 1. 0.])

for 5 neighbours

species prediction:

setosa

versicolor

5-Nearest Neightbours:

index: 3 data value: [-1.50652052 0.09821729 -1.2833891 -1.3154443 ] label: 0 species: setosa

index: 38 data value: [-1.74885626 -0.13197948 -1.39706395 -1.3154443 ] label: 0 species: setosa

index: 12 data value: [-1.26418478 -0.13197948 -1.34022653 -1.44707648] label: 0 species: setosa

index: 45 data value: [-1.26418478 -0.13197948 -1.34022653 -1.18381211] label: 0 species: setosa

index: 8 data value: [-1.74885626 -0.36217625 -1.34022653 -1.3154443 ] label: 0 species: setosa

species prediction:

setosa

versicolor

5-Nearest Neightbours:

index: 97 data value: [ 0.4321654 -0.36217625 0.30805885 0.13250973] label: 1 species: versicolor

index: 74 data value: [ 0.67450115 -0.36217625 0.30805885 0.13250973] label: 1 species: versicolor

index: 91 data value: [ 0.31099753 -0.13197948 0.47857113 0.26414192] label: 1 species: versicolor

index: 71 data value: [ 0.31099753 -0.59237301 0.13754657 0.13250973] label: 1 species: versicolor

index: 63 data value: [ 0.31099753 -0.36217625 0.53540856 0.26414192] label: 1 species: versicolor

CODE:

*#import packages***import** numpy **as** np  
**from** sklearn.datasets **import** load\_iris  
**from** sklearn.preprocessing **import** StandardScaler  
**from** sklearn.neighbors **import** KNeighborsClassifier  
  
*# Loading dataset*iris = load\_iris()  
*#print(iris)  
# extract only data*iris\_data = iris[**'data'**]  
*#print(iris\_data)  
# labels of data*iris\_targets = iris[**'target'**]  
*#print(iris\_targets)  
#target names*iris\_spp = iris[**'target\_names'**]  
*#print(iris\_spp)  
  
# standarize the data*scalar=StandardScaler().fit(iris\_data)  
iris\_standards=scalar.transform(iris\_data)  
*#print(iris\_standards)  
  
# create arrays and standalize the data*plant1=np.array([[4.6, 3.0, 1.5, 0.2]])  
plants = np.array([[4.6, 3.0, 1.5, 0.2], [6.2, 3.0, 4.1, 1.2]])  
plant1\_standards= scalar.transform(plant1)  
plants\_standards= scalar.transform(plants)  
  
*#selecting sepal column*sepal=iris\_standards[:, :2]  
test\_plant1 = plant1\_standards[:, :2]  
test\_plants = plants\_standards[:, :2]  
*# for sepal data training and predicting for 2 neighbors*print(**'for sepal data'**,end=**'\n'**)  
print(**"for 2 neighbors"**)  
classifier = KNeighborsClassifier(n\_neighbors=2).fit(sepal, iris\_targets)  
  
**for** test **in** test\_plant1:  
 print(**"species prediction:"**)  
 **for** i **in** classifier.predict(test\_plant1):  
 print(iris\_spp[i], end=**'\n'**)  
 print(**" 2-Nearest Neightbours:"**)  
 **for** n **in** classifier.kneighbors([test], 2)[1][0]:  
 print(**"index:"**, n, end=**'\t'**)  
 print(**"data value:"**, sepal[n], end=**'\t'**)  
 pred = classifier.predict([sepal[n]])[0]  
 print(**'label:'**, pred, end=**'\t'**)  
 print(**'species:'**, iris\_spp[pred])  
  
*# for sepal data training and predicting for 5 neighbors*print(**"for 5 neighbours"**)  
classifier = KNeighborsClassifier(n\_neighbors=5).fit(sepal, iris\_targets)  
**for** test **in** test\_plants:  
 print(**"species prediction:"**)  
 **for** i **in** classifier.predict(test\_plants):  
 print(iris\_spp[i], end=**'\n'**)  
 print(**"5-Nearest Neightbours:"**)  
 **for** n **in** classifier.kneighbors([test], 5)[1][0]:  
 print(**"index:"**, n, end=**'\t'**)  
 print(**"data value:"**, sepal[n], end=**'\t'**)  
 pred = classifier.predict([sepal[n]])[0]  
 print(**'label:'**, pred, end=**'\t'**)  
 print(**'species:'**, iris\_spp[pred])  
*#predictions*print(**"Predictions for sepal data:"**)  
**for** pred\_species, pred\_probability **in** zip(classifier.predict(test\_plants),  
 classifier.predict\_proba(test\_plants)):  
 print(**'label'**,pred\_species, end=**'\t'**)  
 print(**'species'**,iris\_spp[pred\_species], end=**'\t'**)  
 print(**f'(probability: {**pred\_probability**})'**)  
  
*#for petal data*print(**"for petal data"**,end=**'\n'**)  
*#select petal data column*test\_plants = plants\_standards[:, 2:]  
petal = iris\_standards[:, 2:]  
print(**"for 2 neighbors "**)  
  
*# for petal data training and predicting for 2 neighbors*classifier = KNeighborsClassifier(n\_neighbors=2).fit(petal, iris\_targets)  
  
*#print result***for** test **in** test\_plant1:  
 print(**"species prediction:"**)  
 **for** i **in** classifier.predict(test\_plant1):  
 print(iris\_spp[i], end=**'\n'**)  
 print(**"2-Nearest Neightbours:"**)  
 **for** n **in** classifier.kneighbors([test], 2)[1][0]:  
 print(**"index:"**, n, end=**'\t'**)  
 print(**"data value:"**, petal[n], end=**'\t'**)  
 pred = classifier.predict([petal[n]])[0]  
 print(**'label:'**, pred, end=**'\t'**)  
 print(**'species:'**, iris\_spp[pred])  
  
print(**"for 5 neighbours"**)  
*# for petal data training and predicting for 5 neighbors*classifier = KNeighborsClassifier(n\_neighbors=5).fit(petal, iris\_targets)  
  
*#print result***for** test **in** test\_plants:  
 print(**"species prediction:"**)  
 **for** i **in** classifier.predict(test\_plants):  
 print(iris\_spp[i], end=**'\n'**)  
 print(**"5-Nearest Neightbours:"**)  
 **for** n **in** classifier.kneighbors([test], 5)[1][0]:  
 print(**"index:"**, n, end=**'\t'**)  
 print(**"data value:"**, petal[n], end=**'\t'**)  
 pred = classifier.predict([petal[n]])[0]  
 print(**'label:'**, pred, end=**'\t'**)  
 print(**'species:'**, iris\_spp[pred])  
*#predictions*print(**"Predictions for petal data:"**,end=**'\n'**)  
test\_plant1 = plant1\_standards[:, 2:]  
**for** pred\_species, pred\_probability **in** zip(classifier.predict(test\_plants),  
 classifier.predict\_proba(test\_plants)):  
 print(**'label'**,pred\_species, end=**'\t'**)  
 print(**'species'**,iris\_spp[pred\_species], end=**'\t'**)  
 print(**f'(probability: {**pred\_probability**})'**)  
  
*#using both petal and sepal data*print(**"for sepal and petal data"**)  
test\_plants = plants\_standards[:, :]  
data = iris\_standards[:, :]  
test\_plant1 = plant1\_standards[:, :]  
print(**"for 2 neighbors "**)  
  
*# for sepal and petal data training and predicting for 2 neighbors*classifier = KNeighborsClassifier(n\_neighbors=2).fit(data, iris\_targets)  
  
*#print result***for** test **in** test\_plant1:  
 print(**"species prediction:"**)  
 **for** i **in** classifier.predict(test\_plant1):  
 print(iris\_spp[i], end=**'\n'**)  
 print(**"2-Nearest Neightbours:"**)  
 **for** n **in** classifier.kneighbors([test], 2)[1][0]:  
 print(**"index:"**, n, end=**'\t'**)  
 print(**"data value:"**, data[n], end=**'\t'**)  
 pred = classifier.predict([data[n]])[0]  
 print(**'label:'**, pred, end=**'\t'**)  
 print(**'species:'**, iris\_spp[pred])  
  
*# for sepal and petal data training and predicting for 5 neighbors*print(**"for 5 neighbours"**)  
classifier = KNeighborsClassifier(n\_neighbors=5).fit(data, iris\_targets)  
**for** test **in** test\_plants:  
 print(**"species prediction:"**)  
 **for** i **in** classifier.predict(test\_plants):  
 print(iris\_spp[i], end=**'\n'**)  
 print(**"5-Nearest Neightbours:"**)  
 **for** n **in** classifier.kneighbors([test], 5)[1][0]:  
 print(**"index:"**, n, end=**'\t'**)  
 print(**"data value:"**, data[n], end=**'\t'**)  
 pred = classifier.predict([data[n]])[0]  
 print(**'label:'**, pred, end=**'\t'**)  
 print(**'species:'**, iris\_spp[pred])  
*# prediction*print(**"Predictions for sepal and petal data:"**)  
**for** pred\_species, pred\_probability **in** zip(classifier.predict(test\_plants),  
 classifier.predict\_proba(test\_plants)):  
 print(**'label:'**,pred\_species, end=**'\t'**)  
 print(**'species:'**,iris\_spp[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.