Kingdom of Saudi Arabia
Ministry of Education
University of Jeddah
College of Computer Science and Engineering
Department of Computer Science and Artificial Intelligence

المملكة العربية السعودية
وزارة التعليم
جامعة جدّة
كلية علوم وهندسة الحاسب
قسم علوم الحاسب والذكاء الاصطناعي

Lab 3
CCAI 312 Pattern Recognition
Third Trimester 2023

Student Name: Ruba Khalid Alsulami

Student ID: 2110618

		Max Score	Student Score
PLO S2 / CLO 2 / SO 2	Task 1	2	
PLO C4 / CLO 3 / SO 7	Task 2	2	
Total			

Step 13: We will now train the model using training data and iterate with different values of k=3,5,9.

```
# With k = 3
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
# instantiate learning model (k = 3)
knn_model = KNeighborsClassifier(n_neighbors = 3)
#Fitting the model
knn_model.fit(X_train, y_train)
y_pred = knn_model.predict(X_test) # predict the response
print(accuracy_score(y_test, y_pred)) # Evaluate accuracy
```

The answer is:(**0.9831315737249454**)

```
# With k = 5
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train) # Fitting the model
y_pred = knn_model.predict(X_test) # Predict the response
print(accuracy_score(y_test, y_pred)) # Evaluate accuracy
```

The answer is:(**0.9807501488390553**)

```
#With k = 9
knn_model = KNeighborsClassifier(n_neighbors=9)
knn_model.fit(X_train, y_train) # Fitting the model
y_pred = knn_model.predict(X_test) # Predict the response
print(accuracy_score(y_test, y_pred)) # Evaluate accuracy
```

The answer is:(**0.9793609843222861**)

What is the best k value? (**k=3**)

Task 1: [PLO S2 / CLO 2 / SO 2] [2 marks]

1. First import required libraries, load "**Athlete Selection**" dataset into a data frame and print the first 5 rows.

```
import pandas as pd
import numpy as np

athleteSelection= pd.read_csv('/kaggle/input/lab3athlete/AthleteSelection (1).csv')
athleteSelection.head(5)
```

[2]:

	Athlete	Speed	Agility	Selected
0	x1	2.50	6.00	0
1	x2	3.75	8.00	0
2	x3	2.25	5.50	0
3	x4	3.25	8.25	0
4	x5	2.75	7.50	0

+ Code + Markdown

2. Create a new variable "names" that contain the dataframe indexes and print it.

```
names=athleteSelection.index
print(names)
```

```
RangeIndex(start=0, stop=20, step=1)
```

+ Code + Markdown

3. Store features and labels in numpy arrays X and y and print the first feature of the first example. (Hint: Use **pop** method)

[38]:

```
X = np.array(athleteSelection.iloc[:, 1:3])
Y = np.array(athleteSelection['Selected'])

print(X[0,0])
```

```
2.5
```

```
#or
X = np.array(athleteSelection.iloc[:, 1:3])
Y = np.array(athleteSelection.iloc[:, 1])

print(X[0,0])
```

```
2.5
```

4. Fit NearestNeighbors sickit_learn model to the data with **K=2** and **radius=0.4**.
NearestNeighbors is Unsupervised learner for implementing neighbor searches

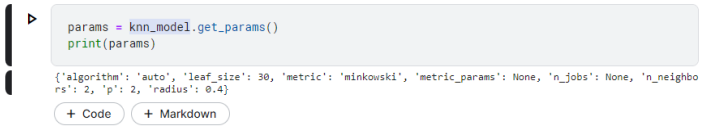
```
from sklearn.neighbors import NearestNeighbors
knn_model = NearestNeighbors(n_neighbors = 2, radius=0.4)
knn_model.fit(X)
```

[42]:

```
NearestNeighbors(n_neighbors=2, radius=0.4)
```

+ Code + Markdown

5. Get parameters of this model (also called estimator)



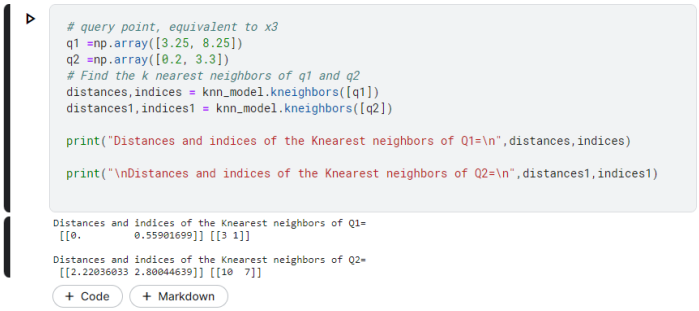
```
params = knn_model.get_params()
print(params)
```

```
{'algorithm': 'auto', 'leaf_size': 30, 'metric': 'minkowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 2, 'p': 2, 'radius': 0.4}
```

+ Code + Markdown

6. Find k nearest neighbors of a point by
returning the distances (array) and the indices(array) of the nearest k points

```
q1 = [3.25,8.25] # query point, equivalent to x3
q2 = [0.2, 3.3]
```



```
# query point, equivalent to x3
q1 =np.array([3.25, 8.25])
q2 =np.array([0.2, 3.3])
# Find the k nearest neighbors of q1 and q2
distances,indices = knn_model.kneighbors([q1])
distances1,indices1 = knn_model.kneighbors([q2])

print("Distances and indices of the Knearest neighbors of Q1=\n",distances,indices)

print("\nDistances and indices of the Knearest neighbors of Q2=\n",distances1,indices1)
```

```
Distances and indices of the Knearest neighbors of Q1=
 [[0.         0.55901699]] [[3 1]]

Distances and indices of the Knearest neighbors of Q2=
 [[2.22036033 2.80044639]] [[10  7]]
```

+ Code + Markdown

7. What does the following code do?

```
q = [5.0,7.5]
q3n = athlete_neigh.kneighbors([q], n_neighbors = 3)[1][0]
for n in q3n:
  print(names[n])
```

This code snippet is using a k-nearest neighbors (KNN) model to find the 3 nearest neighbors to the point q = 5.0, 7.5 in a dataset.
The kneighbors method of the KNN model is called with the point q and the number of neighbors to find (in this case, 3). The method returns two arrays: the distances to the nearest neighbors and the indices of the nearest neighbors in the dataset.
The code then selects the indices of the nearest neighbors from the second array using [1][0] and stores them in the variable q3n.
Finally, the code loops through the indices in q3n and prints the names of the corresponding data points in the dataset using the names array. This allows you to identify the 3 nearest neighbors to the point q in the dataset and see their names.

8. Fit KNeighborsClassifier sickit_learn model to the data with K=3. KNeighborsClassifier is classifier implementing the k-nearest neighbors vote.

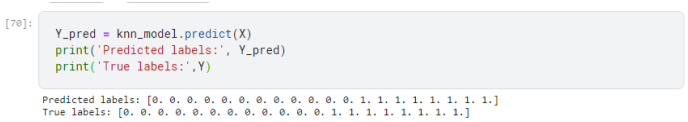
```
from sklearn.neighbors import KNeighborsClassifier

X = np.array(athleteSelection.iloc[:, 1:3])
Y = np.array(athleteSelection['Selected'])
knn_model= KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X,Y)
```

```
[82]: KNeighborsClassifier(n_neighbors=3)
```

+ Code + Markdown

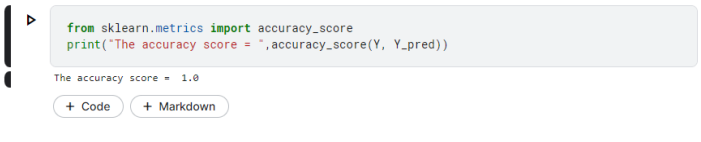
9. Evaluate the model Using training data as test set (Hint: Use model predict method)



```
[70]: Y_pred = knn_model.predict(X)
print('Predicted labels:', Y_pred)
print('True labels:',Y)
```

```
Predicted labels: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1.]
True labels: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1.]
```

10. How accurate is the model? Is the accuracy good or bad? Why do you believe the model was so accurate?



```
from sklearn.metrics import accuracy_score
print("The accuracy score = ",accuracy_score(Y, Y_pred))
```

```
The accuracy score =  1.0
```

+ Code + Markdown

Task 2: [PLO C4 / CLO 3 / SO 7] **[2 mark]**

1. Read athlete_test file and

   store features and labels in numpy arrays X_test and y_test (Hint: Use pop method)

```
[1]:
import pandas as pd
import numpy as np
athlete_test=pd.read_csv('/kaggle/input/lab3athlete/AthleteTest (1).csv')

X_test = np.array(athlete_test.iloc[:, 1:3])
y_test = np.array(athlete_test['Selected'])
```

2. Fit KNeighborsClassifier sickit_learn model to the data with K=3.

```
from sklearn.neighbors import KNeighborsClassifier
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_test, y_test)

[27]: KNeighborsClassifier(n_neighbors=3)

+ Code  + Markdown
```

3. Evaluate the model Using X_test and y_test data as test set (Hint: Use model predict method)

```
from sklearn.metrics import accuracy_score

y_pred = knn_model.predict(X_test)

print('Predicted labels:', y_pred)
print('True labels:', y_test)

print("The accuracy score = ",accuracy_score(y_test, y_pred))

Predicted labels: [0 0 0 0 0 0 1 0 0 0]
True labels: [0 0 0 1 0 0 1 0 0 1]
The accuracy score =  0.8

+ Code  + Markdown
```

4. Use StandardScaler from sklearn to map features values to unit variance and fit and evaluate a new KNeighborsClassifier model

```
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_std = scaler.fit_transform(X_test)

knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_test, y_test)

print(knn_model)
print("The accuracy score = ",accuracy_score(y_test, y_pred))

KNeighborsClassifier(n_neighbors=3)
The accuracy score =  0.8

+ Code  + Markdown
```

5. Use MinMaxScaler from sklearn to map features values to unit variance and fit and evaluate a new KNeighborsClassifier model

```
[38]:
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_std = scaler.fit_transform(X_test)

knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_test, y_test)

print(knn_model)
print("The accuracy score = ",accuracy_score(y_test, y_pred))
```

```
KNeighborsClassifier(n_neighbors=3)
The accuracy score =  0.8
```

6. Evaluate previous three models with different values of K

```
for k in range(1,7):
    knn_model = KNeighborsClassifier(n_neighbors=k)
    knn_model.fit(X_test, y_test)
    y_pred = knn_model.predict(X_test)
    print("\nThe accuracy score and K=",k ,"KNeighborsClassifier=",accuracy_score(y_test, y_p

    scaler = StandardScaler()
    X_std = scaler.fit_transform(X_test)
    knn_model1 = KNeighborsClassifier(n_neighbors=k)
    knn_model1.fit(X_test, y_test)
    y_pred = knn_model1.predict(X_test)
    print("The accuracy score with StandardScaler= ",accuracy_score(y_test, y_pred))

    scaler1 = MinMaxScaler()
    X_std = scaler1.fit_transform(X_test)
    knn_model2 = KNeighborsClassifier(n_neighbors=k)
    knn_model2.fit(X_test, y_test)
    y_pred = knn_model2.predict(X_test)
    print("The accuracy score with MinMaxScaler= ",accuracy_score(y_test, y_pred))
```

```
The accuracy score and K= 1 KNeighborsClassifier= 0.9
The accuracy score with StandardScaler=  0.9
The accuracy score with MinMaxScaler=  0.9

The accuracy score and K= 2 KNeighborsClassifier= 0.7
The accuracy score with StandardScaler=  0.7
The accuracy score with MinMaxScaler=  0.7

The accuracy score and K= 3 KNeighborsClassifier= 0.8
The accuracy score with StandardScaler=  0.8
The accuracy score with MinMaxScaler=  0.8

The accuracy score and K= 4 KNeighborsClassifier= 0.7
The accuracy score with StandardScaler=  0.7
The accuracy score with MinMaxScaler=  0.7

The accuracy score and K= 5 KNeighborsClassifier= 0.7
The accuracy score with StandardScaler=  0.7
The accuracy score with MinMaxScaler=  0.7

The accuracy score and K= 6 KNeighborsClassifier= 0.7
The accuracy score with StandardScaler=  0.7
The accuracy score with MinMaxScaler=  0.7
```

+ Code + Markdown

7. Which of the models is most accurate?

The most accurate model that have the highest accuracy on the test data. which is (k = 1)