PAF- KARACHI INSTITUTE OF ECONOMICS & TECHNOLOGY
College of Engineering

(Software Engineering)

Artificial Intelligence

Semester: Spring 2022 **Date of Experiment:** _____

Student name: Rehan Abu Hashir **Faculty Signature:** _____

Student ID: 10673

Lab08	**Machine Learning Algorithm (K Nearest Neighbor)**		
PLOs	PLO1 – Engineering Knowledge	Bloom's Taxonomy	C1 – Recall
	PLO3 – Design & Development		C3 - Apply
	PLO8 – Ethics		P2 – Set

LAB TASK PERFORMANCE

CLO's	Aspects of Assessments	Excellent (75-100%)	Average (50-75%)	Poor (<50%)	Marks
CLO1 10%	**Recall** The associated concepts of Programming Language.	Complete understanding of Programming / actively participate during lecture.	Complete understanding of Programming / less actively participate during lecture.	Student lacks clear understanding of concepts of Programming / Unable to read and interpret it.	
CLO4 80%	**Design & Develop ML** Design/Develop solutions for K nearest Neighbor algorithm with python programming language.	Accurately implement the K nearest Neighbor algorithm obtain the correct output as per requirement/ given tasks.	Implement the K nearest Neighbor with minor errors that will lead to a slightly different output as per given in a task.	Not able to implement the K nearest Neighbor algorithm and don't understand how required output and task is achieved.	
CLO7 10%	**Lab Safety** Properly handle lab infrastructure/safety precautions	Properly handle lab equipment & obey safety measures.	Moderate level lab handling and safety measurements	Minor or no safety measurements has been considered.	
				Total Marks: 10	

Lab 9- Machine Learning Algorithm (K Nearest Neighbor)

Learning Outcome:

Implement the Machine Learning Algorithm **K Nearest Neighbor** in Python Programming Language.

Theory:

What is KNN Algorithm?

K nearest neighbors or **KNN** Algorithm is a simple algorithm which uses the entire dataset in its training phase. Whenever a prediction is required for an unseen data instance, it searches through the entire training dataset for k-most similar instances and the data with the most similar instance is finally returned as the prediction.

KNN is often used in search applications where you are looking for similar items, like find items similar to this one.

Algorithm suggests that if you're similar to your neighbors, then you are one of them. For example, if apple looks more similar to peach, pear, and cherry (fruits) than monkey, cat or a rat (animals), then most likely apple is a fruit.

How does a KNN Algorithm work?

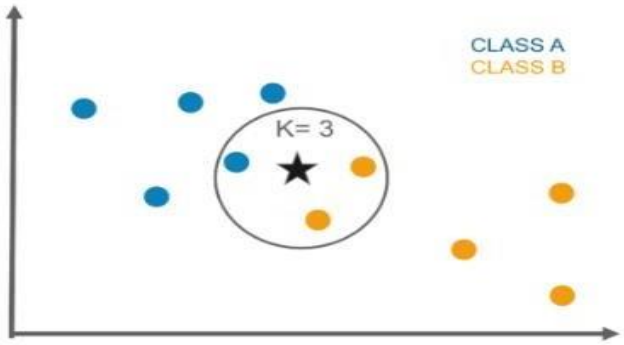
The k-nearest neighbor's algorithm uses a very simple approach to perform classification. When tested with a new example, it looks through the training data and finds the k training examples that are closest to the new example. It then assigns the most common class label (among those k-training examples) to the test example.

What does 'k' in KNN Algorithm represent?

K in KNN algorithm represents the number of nearest neighbor points which are voting for the new test data's class.

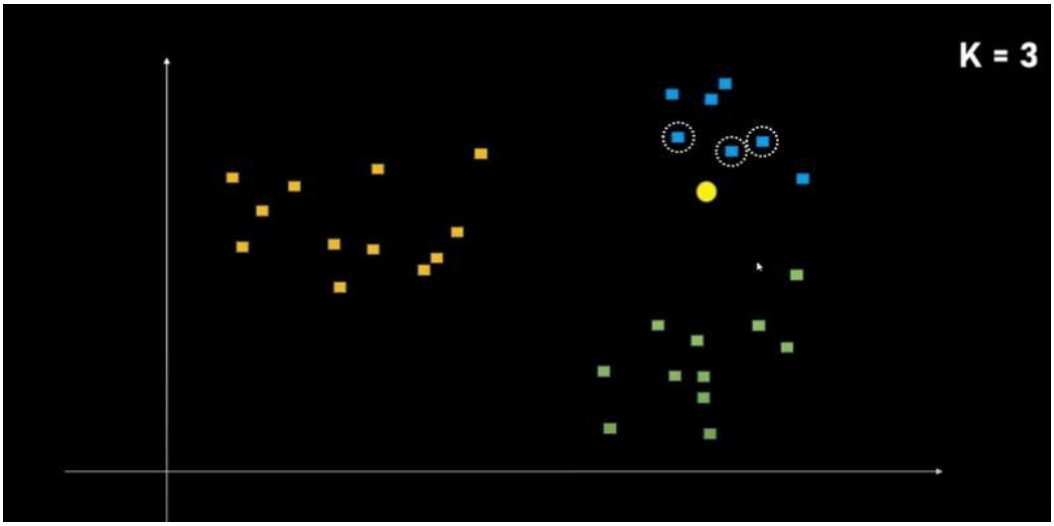
If **k=1,** then test examples are given the same label as the closest example in the training set.
If **k=3,** the labels of the three closest classes are checked and the most common (i.e., occurring at least twice) label is assigned, and so on for larger ks.
Calculate the distance between the points by using Euclidean Distance,

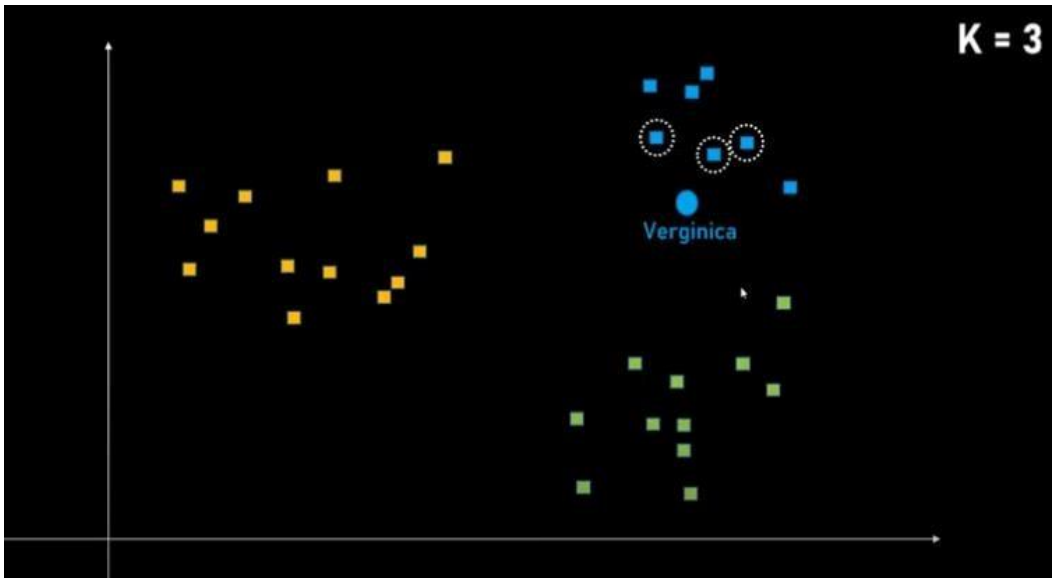
$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$



How to predict?

When we set the value of K is 3 then it finds 3 nearest points and all the three points are Verginica That's why it predict its Verginica.





KNN Algorithm Implementation:

Load the Iris Dataset.

```
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()
```

Dataset features:

```
iris.feature_names
```

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

Data Target Names:

```
iris.target_names
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

Create a Data frame:

```
df = pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Here,
Target 0 = Setosa flower

```
df['target'] = iris.target
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

Target 1 = Versicolor flower

```
df[df.target==1].head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
50	7.0	3.2	4.7	1.4	1
51	6.4	3.2	4.5	1.5	1
52	6.9	3.1	4.9	1.5	1
53	5.5	2.3	4.0	1.3	1
54	6.5	2.8	4.6	1.5	1

Target 2 = Virginica flower

```
df[df.target==2].head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
100	6.3	3.3	6.0	2.5	2
101	5.8	2.7	5.1	1.9	2
102	7.1	3.0	5.9	2.1	2
103	6.3	2.9	5.6	1.8	2
104	6.5	3.0	5.8	2.2	2

Target with flower names:

```
df['flower_name'] =df.target.apply(lambda x: iris.target_names[x])
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

```
df[45:55]
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
45	4.8	3.0	1.4	0.3	0	setosa
46	5.1	3.8	1.6	0.2	0	setosa
47	4.6	3.2	1.4	0.2	0	setosa
48	5.3	3.7	1.5	0.2	0	setosa
49	5.0	3.3	1.4	0.2	0	setosa
50	7.0	3.2	4.7	1.4	1	versicolor
51	6.4	3.2	4.5	1.5	1	versicolor

Declare Variable for all Target Names:

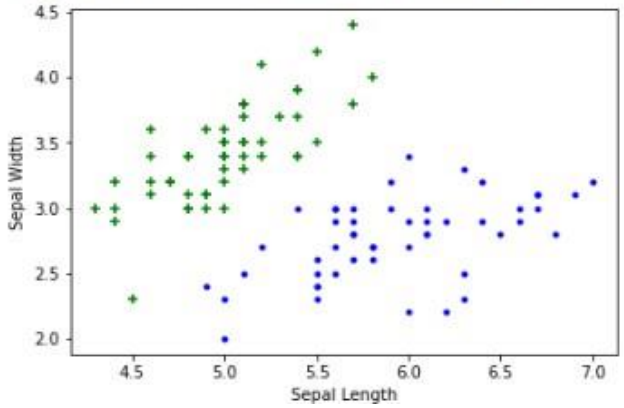
```
df0 = df[:50]
df1 = df[50:100]
df2 = df[100:]
```

Plot the Sepal Length and Width:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="green",marker='+')
plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="blue",marker='.')
```

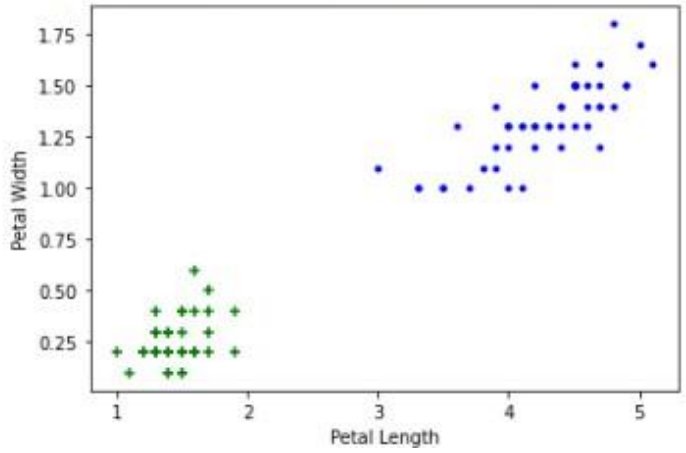
```
<matplotlib.collections.PathCollection at 0x7fbc5ad138d0>
```



Plot the Petal Length and Width:

```
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.scatter(df0['petal length (cm)'], df0['petal width (cm)'],color="green",marker='+')
plt.scatter(df1['petal length (cm)'], df1['petal width (cm)'],color="blue",marker='.')
```

```
<matplotlib.collections.PathCollection at 0x7fbc5a80d150>
```



Train and Test the Model:

```
from sklearn.model_selection import train_test_split
```

```
X = df.drop(['target','flower_name'], axis='columns')
y = df.target
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

```
len(X_train)
```

```
120
```

```
len(X_test)
```

```
30
```

Import the Model with Neighbor = 3:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
```

Fit and Predict the Model:

```
knn.fit(X_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
knn.predict([[4.8,3.0,1.5,0.3]])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: Use
  "X does not have valid feature names, but"
array([0])
```

It is predict flower become Setosa, if the values of sepal and petal length and width are 4.8, 3.0, 1.5, 0.3.

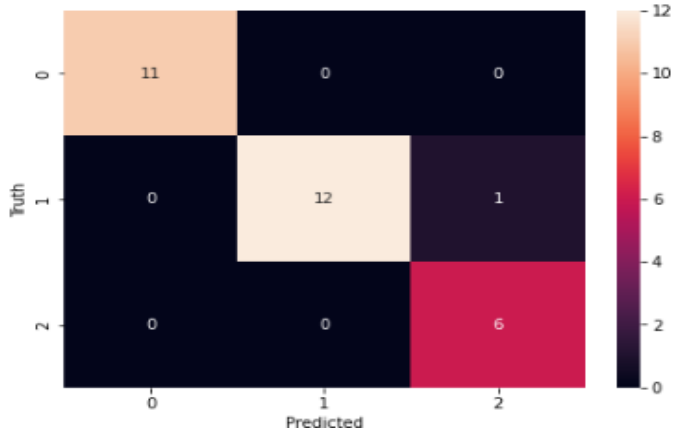
Confusion Matrix:

```
from sklearn.metrics import confusion_matrix
y_pred = knn.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[11,  0,  0],
       [ 0, 12,  1],
       [ 0,  0,  6]])
```

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(42.0, 0.5, 'Truth')
```



Lab Task:

Task # 01: Implement the Example Code and predict at least 10 samples by changing the sepal and petal sizes.

```
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()
```

```
iris.feature_names
```

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

```
[ ] iris.target_names
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
[ ] df = pd.DataFrame(iris.data, columns= iris.feature_names)
    df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
df['target'] = iris.target
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
[ ] df[df.target==1].head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
50	7.0	3.2	4.7	1.4	1
51	6.4	3.2	4.5	1.5	1
52	6.9	3.1	4.9	1.5	1
53	5.5	2.3	4.0	1.3	1
54	6.5	2.8	4.6	1.5	1

```
[ ] df[df.target==2].head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
100	6.3	3.3	6.0	2.5	2
101	5.8	2.7	5.1	1.9	2
102	7.1	3.0	5.9	2.1	2
103	6.3	2.9	5.6	1.8	2
104	6.5	3.0	5.8	2.2	2

```
[ ] df['flower_name'] = df.target.apply(lambda x: iris.target_names[x])
    df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

```
[ ] df[45:55]
```

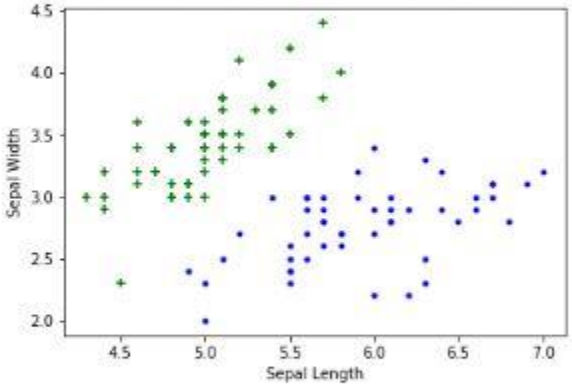
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
45	4.8	3.0	1.4	0.3	0	setosa
46	5.1	3.8	1.6	0.2	0	setosa
47	4.6	3.2	1.4	0.2	0	setosa
48	5.3	3.7	1.5	0.2	0	setosa
49	5.0	3.3	1.4	0.2	0	setosa
50	7.0	3.2	4.7	1.4	1	versicolor
51	6.4	3.2	4.5	1.5	1	versicolor
52	6.9	3.1	4.9	1.5	1	versicolor
53	5.5	2.3	4.0	1.3	1	versicolor
54	6.5	2.8	4.6	1.5	1	versicolor

```
[ ] df0 = df[:50]
    df1 = df[50:100]
    df2 = df[100:]
```

```
[ ] import matplotlib.pyplot as plt
    %matplotlib inline
```

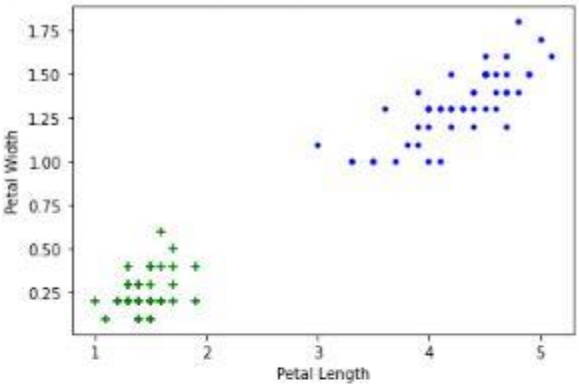
```
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="green",marker='+')
plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="blue",marker='.')
```

```
<matplotlib.collections.PathCollection at 0x7febd6d6d550>
```




```
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.scatter(df0['petal length (cm)'], df0['petal width (cm)'],color="green",marker='+')
plt.scatter(df1['petal length (cm)'], df1['petal width (cm)'],color="blue",marker='.')
```

```
<matplotlib.collections.PathCollection at 0x7febd6d6d590>
```




```
from sklearn.model_selection import train_test_split
```

```
x = df.drop(['target','flower_name'], axis='columns')
y = df.target
```

```
x_train, x_test, y_train, y_test = train_test_split (x, y, test_size=0.2, random_state=1)
```

```
len(x_train)
```

```
120
```

```
len(x_test)
```

```
30
```

```
[ ] from sklearn.neighbors import KNeighborsClassifier
    knn = KNeighborsClassifier(n_neighbors=3)
```

```
[ ] knn.fit(x_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
[ ] knn.predict([[4.8,3.0,1.5,0.3]])
```

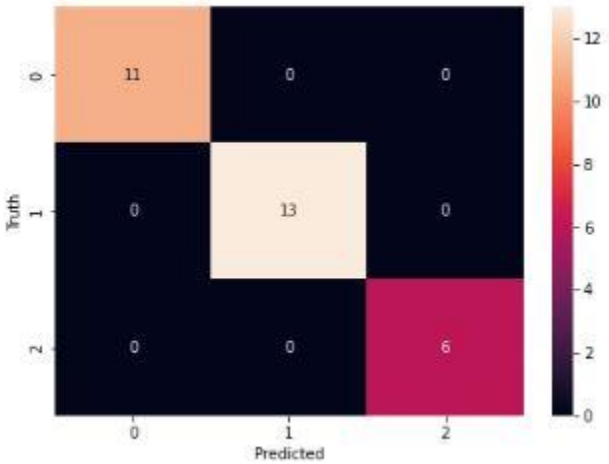
```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but KNeighbo
  "X does not have valid feature names, but"
array([0])
```

```
[ ] from sklearn.metrics import confusion_matrix
    y_pred = knn.predict (x_test)
    cm = confusion_matrix(y_test, y_pred)
    cm
```

```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  0,  6]])
```

```
[ ] %matplotlib inline
    import matplotlib.pyplot as plt
    import seaborn as sn
    plt.figure(figsize=(7,5))
    sn.heatmap(cm, annot=True)
    plt.xlabel('Predicted')
    plt.ylabel('Truth')
```

```
Text(42.0, 0.5, 'Truth')
```




```
[28] knn.predict([[7.0,3.2,4.7,1.4]])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but K
  "X does not have valid feature names, but"
array([1])
```

```
knn.predict([[6.9,3.1,4.9,1.5]])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but KNeighborsCl
  "X does not have valid feature names, but"
array([1])
```

```
[31] knn.predict([[4.6,3.1,1.5,0.2]])

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have vali
  "X does not have valid feature names, but"
array([0])
```

```
knn.predict([[6.3,2.9,5.6,1.8]])

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not h
  "X does not have valid feature names, but"
array([2])
```

Task # 02: Write a program in Python to implement the K nearest Neighbor algorithm. Suppose we have height and weight and its corresponding T-shirt size of several customers. Your task is to predict the T-shirt size of the following cases,
Ahmed **height 159cm** and **weight is 58kg**.
Yasir **height 164cm** and **weight is 64kg**.
Rahim **height 171cm** and **weight is 61kg**.

Height in (cms)	Weight in (kgs)	T-Shirt (Size)
158	59	M
158	59	M
160	63	M
160	59	M
163	60	M
163	60	M
160	61	L
163	64	L
165	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	L

```
import pandas as pd
df=pd.read_csv(r"t-shirt.csv")
df
```

	Height	Weight	Size	Target
0	158	59	M	1
1	158	59	M	1
2	160	63	M	1
3	160	59	M	1
4	163	60	M	1
5	163	60	M	1
6	160	61	L	0
7	163	64	L	0
8	165	64	L	0
9	165	61	L	0
10	165	62	L	0
11	165	65	L	0
12	168	62	L	0
13	168	63	L	0
14	168	66	L	0
15	170	63	L	0
16	170	64	L	0

[1]

16	170	64	L	0
17	170	68	L	0

[2]
```
from sklearn.model_selection import train_test_split
X=df.drop(['Target','Size'],axis='columns')
y=df. Target
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1)
len(X_train)
```

```
14
```

[3]
```
len(X_test)
```

```
4
```

```
from sklearn.neighbors import  KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

[5]
```
knn.predict([[159,58]])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not h
  "X does not have valid feature names, but"
array([1])
```

```
[6] knn.predict([[164,64]])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid featu
  "X does not have valid feature names, but"
array([0])
```

```
knn.predict([[171,61]])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid featu
  "X does not have valid feature names, but"
array([0])
```