IMPORTING DEPENDENCIES

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

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

CREATING DATAFRAME

In [3]:
```
#Putting data into a pandas data frame
iris = datasets.load_iris()
df = pd.DataFrame(data = iris['data'], columns = iris['feature_names'])
df['class'] = iris.target
X = df
X
```

Out[3]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

ORGANIZING DATA FOR TRAINING AND TESTING

In [4]:
```
data = iris['data']
```

In [5]:
```
y = iris['target']
```

In [6]:
```
zipped_data = list(zip(data,y))
```

In [7]:
```
#Splitting Data Set into Training and Testing Data Sets
X_train, X_test, y_train, y_test = train_test_split(data, y, test_size =
.50, random_state = 1)
```

CREATING FUNCTION TO CALCULATE DISTANCE BETWEEN 2 POINTS/VECTORS

In [8]:
```
#Calculate eucledian distance between 2 data points
def eucledian(data, new_pt):
    #distance = []
    #for i in range(len(data)):
        #distance.append(np.sqrt(np.sum((data[i] - new_pt)**2)))
    distance = np.sqrt(np.sum((data - new_pt)**2))
    return distance
```

CREATING AND K NEAREST NEIGHBOR(KNN) ALGORITHM

In [20]:
```
def knn(data, y, new_pt, k):
    predict_classes = {0:'Iris-Virginica', 1:'Iris-Setosa', 2:'Iris-Vers
icolor'}
    distance = []
    flower_type = []
    flower_class = []
    #creating sorted tuples from closets to furthest of the iris data po
ints
    for i in range(len(data)):
        dist = round(eucledian(data[i], new_pt), 2)
        distance.append(dist)
        flower_type.append(y[i])
    zip_tup = list(zip(distance, flower_type))
    zip_tup.sort()

    #getting the k closets values nearest neighbors from sorted data poi
nts
    for j in range(len(zip_tup)):
        near_neighbor = zip_tup[0:k]

    #creating a list of the classes of the k closets values nearest neig
hbors from sorted data points
    for k in range(len(near_neighbor)):
        flower_class.append(near_neighbor[k][1])

    #returning the most frequent class in the k closets nearest neighbor
s
    frequent_flower_class = max(set(flower_class), key=flower_class.coun
t)
    #print(f'The flower type is {frequent_flower_class}, {predict_classe
s[frequent_flower_class]}')

    return frequent_flower_class
```

In [17]:
```
new_pt = [2.3, 0.2, 3.0, 1.3]
knn(data, y, new_pt, 5)
```

```
The flower type is 1, Iris-Setosa
```

Out[17]: 1

CALCULATING KNN ACCURACY

In [21]:
```
def knn_accuracy(X_train, y_train, X_test, y_test, k):
    correct = 0
    for i in range(len(X_test)):
        predict = knn(X_train, y_train, X_test[i], k)
        actual = y_test
        if predict == actual[i]:
            correct += 1
    accuracy = round(correct/len(X_test) * 100, 2)
    return accuracy
```

In [22]:
```
knn_accuracy(X_train, y_train, X_test, y_test, 10)
```

Out[22]: 97.33