In [1]:

```
import pandas as pd
import numpy as np
from sklearn import datasets
from collections import Counter
```

In [2]:

```
iris = datasets.load_iris()
Species = iris.target
data = pd.DataFrame(np.c_[iris.data,Species.reshape((Species.shape[0],1))],columns=iris.
feature_names+['Species'])
data.head()
```

Out[2]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | Species |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | 0.0 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | 0.0 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | 0.0 |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | 0.0 |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | 0.0 |

In [3]:

```
data['Species'].value_counts()
```

Out[3]:

```
2.0    50
1.0    50
0.0    50
Name: Species, dtype: int64
```

In [4]:

```
from sklearn.model_selection import train_test_split
train,test=train_test_split(data,test_size=0.2,random_state=123)
```

In [5]:

```
class knn():
    def __init__(self,X,Y,k_neighbors):
        self.k_neighbors=k_neighbors
        self.X_train=X
        self.Y_train=Y
        self.target=set(Y)

    def euclidean_distance(self,row1,row2):
        distance=0.0
        for i in range(len(row1)):
            distance+=(row1[i]-row2[i])**2
            return np.sqrt(distance)

    def sort_distance(self,r):
        return r[2]

    def get_neighbors(self,row):
        dist=[]
        for row_index in range(len(self.X_train)):
            d=self.euclidean_distance(self.X_train.iloc[row_index,:],row)
            dist.append((self.X_train.iloc[row_index,:],self.Y_train.iloc[row_index],d))
        dist.sort(key=self.sort_distance)
        neighbors=[]
        for i in range(self.k_neighbors):
            neighbors.append(dist[i][1])
        return neighbors
    def predict(self,row):
        neigh=self.get_neighbors(row)
        neighbors=Counter(neigh)
        count=0
        pred=""
        for i in self.target:
            if neighbors[i]>count:
                count=neighbors[i]
                pred=i
                return pred
```

In [6]:

```
Y=train['Species']
X=train.drop('Species',axis=1)
clf=knn(X,Y,5)
X.loc[0,:]
```

Out[6]:

```
sepal length (cm)    5.1
sepal width (cm)     3.5
petal length (cm)    1.4
petal width (cm)     0.2
Name: 0, dtype: float64
```

In [7]:

```
predictions=[]
Y_test=test['Species']
X_test=test.drop('Species',axis=1)
for row in range(len(X_test)):
    pred=clf.predict(X_test.iloc[row,:])
    predictions.append(pred)
```

In [8]:

```
from sklearn.metrics import accuracy_score
accuracy_score(Y_test,predictions)
```

Out[8]:

```
0.6333333333333333
```

In [9]:

```
from sklearn.neighbors import KNeighborsClassifier
neigh=KNeighborsClassifier(n_neighbors=3)
neigh.fit(X,Y)
pred1=neigh.predict(X_test)
accuracy_score(Y_test,pred1)
```

Out[9]:

```
0.9666666666666667
```

In [ ]: