**The KNN Classifier**

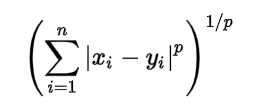
1. Load the data

2. Initialize K to your chosen number of neighbors

3. For each example in the data

3.1 Calculate the distance between the query example and the current example from the data.

3.2 Add the distance and the index of the example to an ordered collection

  
As per the equation, we have to select the p-value also.

p = 1 , Manhattan Distance

p = 2 , Euclidean Distance

p = infinity , Cheybchev Distance

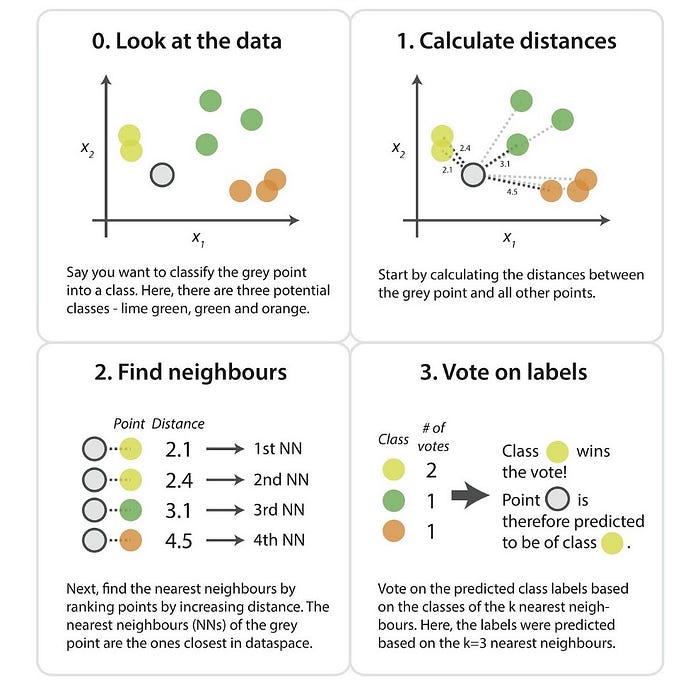
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances

5. Pick the first K entries from the sorted collection

6. Get the labels of the selected K entries

7. If regression, return the mean of the K labels

8. If classification, return the mode of the K labels



***Importing Libraries***

*import matplotlib.pyplot as plt*

*import pandas as pd*

*from sklearn.model\_selection import train\_test\_split*

*from sklearn.preprocessing import StandardScaler*

*from sklearn.neighbors import KNeighborsClassifier*

*from sklearn.metrics import classification\_report,confusion\_matrix*

#To import the dataset and load it into our pandas dataframe, execute the following code:

*url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"*

***# Assign colum names to the dataset***

*names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']*

***# Read dataset to pandas dataframe***

*dataset = pd.read\_csv(url, names=names)*

**#**from sklearn.datasets import load\_iris

*#*dataset = load\_iris()

**To see what the dataset actually looks like, execute the following command:**

*dataset.head()*

**The next step is to split our dataset into its attributes and labels. To do so, use the following code:**

*X = dataset.iloc[:, :-1].values*

*y = dataset.iloc[:, -1].values*

**To create training and test splits, execute the following script:**

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)*

**The following script performs feature scaling:**

*scaler = StandardScaler()*

*X\_train = scaler.fit\_transform(X\_train)*

*X\_test = scaler.fit\_transform(X\_test)*

**Training and Predictions:**

*classifier = KNeighborsClassifier(n\_neighbors=5)*

*classifier.fit(X\_train, y\_train)*

*y\_pred = classifier.predict(X\_test)*

**Evaluating the Algorithm:**

*print(confusion\_matrix(y\_test, y\_pred))*

*print(classification\_report(y\_test, y\_pred))*

*y\_pred = classifier.predict([[4,3.5,3,6.5]])*

### **Comparing Error Rate with the K Value:**

To do so, let's first calculate the mean of error for all the predicted values where K ranges from 1 and 40. Execute the following script:

*error = []*

**# Calculating error for K values between 1 and 40**

for i in range(1, 51):

knn = KNeighborsClassifier(n\_neighbors=i)

knn.fit(X\_train, y\_train)

pred\_i = knn.predict(X\_test)

error.append(np.mean(pred\_i != y\_test))

**The next step is to plot the error values against K values. Execute the following script to create the plot:**

*plt.figure(figsize=(10,6))*

*plt.plot(range(1, 51), error, color='red', linestyle='dashed', marker='o', markerfacecolor='blue', markersize=10)*

*plt.title("Error rate vs k value",fontsize=20)*

*plt.xlabel("k- values")*

*plt.ylabel("error rate")*

*plt.xticks(range(1,50))*

*plt.show()*