K- nearest neighbour algorithm

```
Q1.
    1. Download the Iris data set
    2. Devide the whole data set into three data sets namely train(65% of data), cros
s valid(20% ofdata), test(15% of data) data sets by using a [0-1] random number genar
ator.
    3. Find the best 'k' in range (3-20) for which the knn will give best accuracy on
cross valid data set
    4. Report the Precision, recall and Accuracy for choosen 'k' on test data
Q2.
    1. appply the same algorithm on mnist
Note1: for the distance measure you can choose any of the Euclidean or Manhattan.
Note2: It is your choice to normalize the data.
```

In [23]:
```
import pandas as pd
import numpy as np
```

Useful functions for KNN algorithm

Devide the whole data set into three data sets namely train(65% of data), cross valid(20% ofdata), test(15% of data) data sets .

In [24]:
```
def get_train_test_crossvalid_indices(size):
    # creating index array ie., from 0 to size-1
    indices = np.arange(size)
    np.random.shuffle(indices)
    # number of rows in iris dataframe. ie.,150
    size = iris.shape[0]
    # 65% of the data as training data
    train_size = (size*65)//100
    #15% of the data as test data
    test_size = (size*15)//100
    #20% of the data as cross validation data
    crsvld_size = size - train_size - test_size

    print("\n train_size: {} \n test_size: {} \n crsvld_size: {}\n".format(trai
n_size, test_size, crsvld_size))

    #split the indices list according to sizes specified.
    return np.split(indices, [train_size, train_size+test_size])
```

Euclidian Distance

In [25]:
```
import math
def euclidean_distance(p,q):
    return math.sqrt(sum(pow((p-q), 2)))
```

Get Nearest Neighbours

In [26]:
```
from operator import itemgetter
def get_Nearest_neighbours(train_data, crsvld_data_point, k = 10):
    # print(point)
    # print()
    distance_label_pair = list()
    for i in range(len(train_data)):
        label = train_data[i][-1]
        # slicing of data point as distance_label[:-1] excludes last element ie
., label
        distance_label_pair.append([euclidean_distance(train_data[i][:-1], crsv
ld_data_point[:-1]), label])
    #sort the distance_label pairs based on distances and get top k points
    k_nearest_points = sorted(distance_label_pair, key = itemgetter(0))[:k]
    # print(k_nearest_points)
    # It will return just labels of nearest data points
    return [data_point[1] for data_point in k_nearest_points]
```

Get Majority Voted Element

In [27]:
```
from collections import Counter
def get_Majoriy_Voted_label(neighbours):
    return Counter(neighbours).most_common()[0][0]
```

KNN function

In [28]:
```
def knn(train_data, crsvld_data, k=10):
    count = 0
    for i in range(len(crsvld_data)):
        nearest_neighbours = get_Nearest_neighbours(train_data, crsvld_data[i],
k)
        predicted = get_Majoriy_Voted_label(nearest_neighbours)
#         print("{} = {} ".format(crsvld_data[i][-1], predicted))
        if predicted == crsvld_data[i][-1]:
            count = count+1
    print("Accuracy for k = {} is {}%".format(k, (count/len(crsvld_data)) * 100
))
```

Q1 . PERFORM knn on IRIS flower dataset

In [29]:
```
iris = pd.read_csv('Iris.csv')
print("Shape: {}".format(iris.shape))
print("number of Data points : {}".format(iris.shape[0]))
print(iris.columns)
```

```
Shape: (150, 6)
number of Data points : 150
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [30]:
```
del iris['Id']
```

In [31]:
```
iris.tail()
```

Out[31]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

split the actual data into train, test and crossvalid data

In [32]:
```
#get the indices lists for test, train and crossvalidate data..
train_indices, test_indices, crsvld_indices = get_train_test_crossvalid_indices
(iris.shape[0])

# create train, test and cross validation datasets..
# as_matrix() gives the dataframe as numpy array..
train_data = iris.iloc[list(train_indices)].as_matrix()
crsvld_data = iris.iloc[list(crsvld_indices)].as_matrix()
test_data = iris.iloc[list(test_indices)].as_matrix()
```

```
train_size: 97
test_size: 22
crsvld_size: 31
```

Testing KNN for different values of k and their accuracy

In [33]:
```
for k in range(3,21):
    knn(train_data, crsvld_data,k)
```

```
Accuracy for k = 3 is 93.54838709677419%
Accuracy for k = 4 is 93.54838709677419%
Accuracy for k = 5 is 90.32258064516128%
Accuracy for k = 6 is 93.54838709677419%
Accuracy for k = 7 is 93.54838709677419%
Accuracy for k = 8 is 93.54838709677419%
Accuracy for k = 9 is 96.7741935483871%
Accuracy for k = 10 is 96.7741935483871%
Accuracy for k = 11 is 96.7741935483871%
Accuracy for k = 12 is 96.7741935483871%
Accuracy for k = 13 is 96.7741935483871%
Accuracy for k = 14 is 96.7741935483871%
Accuracy for k = 15 is 90.32258064516128%
Accuracy for k = 16 is 93.54838709677419%
Accuracy for k = 17 is 93.54838709677419%
Accuracy for k = 18 is 90.32258064516128%
Accuracy for k = 19 is 90.32258064516128%
Accuracy for k = 20 is 90.32258064516128%
```