**Machine Learning Assignment**

**K-Nearest Neigh-bours**

**19103058**

**L.A. LALIT VIYOGI**

**-----------------------------------------------------------------------------------------------**

**About the Data:**

Here, We have used IRIS datasets provided in assignment for implementation of KNN algorithms.

Dataset contain the information about sepal length, sepal width, petal length, petal width, and label based upon the category in which it falls. There are basic three category

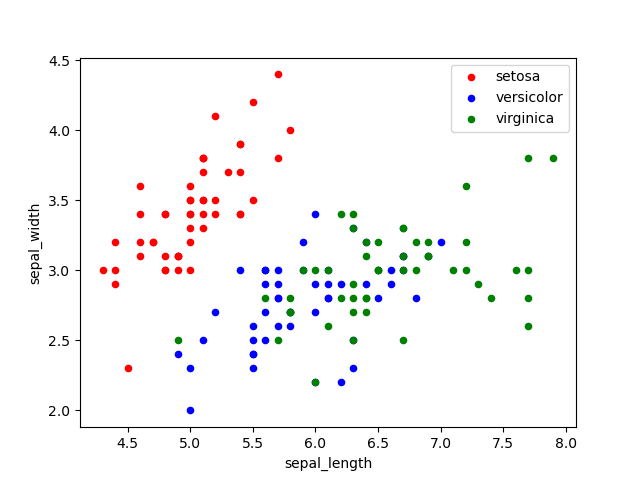
1. Iris-Setosa,
2. Iris- Versicolor,
3. Iris- Virginica

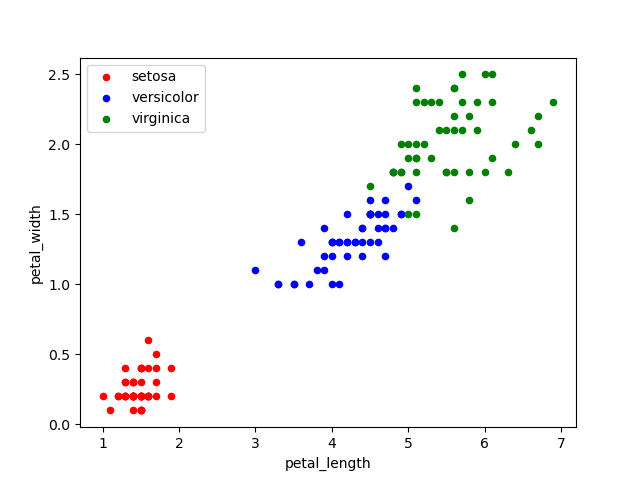
There are about 150 example in given dataset in which about 80% are used for training of model and rest of them are for testing of model.

**-----------------------------------------------------------------------------------------------**

**Insights from Data:**

As mentioned above, Data is divided in three category so we need to draw graph to understand the basic property of each class in terms length and width of sepal and petal.





From these chart visualizations, we get some rough insights as mentioned below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Petal length** | **Petal width** | **Sepal length** | **Sepal width** |
| **Iris-Setosa** | 1 -- 2 | 0.1 -- 0.6 | 2.8 – 4.2 | 4.4 -- 5.6 |
| **Iris- Versicolor** | 3 -- 5 | 1.0 -- 1.7 | 5.0 -- 6.8 | 2.2 – 3.2 |
| **Iris- Virginica** | 5 -- 7 | 1.5 -- 2.5 | 5.5 -- 7.8 | 2.5 – 3.8 |

**-----------------------------------------------------------------------------------------------**

**Code Used for graph Plotting of IRIS Data**

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv("/content/IRIS.csv")

#=======================================================

setosa=data[data['species']=='Iris-setosa']

versicolor=data[data['species']=='Iris-versicolor'

virginica =data[data['species']=='Iris-virginica']

plt.figure()

fig,ax=plt.subplots(1,1) ## to plot all three labels in ax(same plot)

setosa.plot(x="sepal\_length", y="sepal\_width", kind="scatter",ax=ax,label='setosa',color='r')

versicolor.plot(x="sepal\_length",y="sepal\_width",kind="scatter",ax=ax,label='versicolor',color='b')

virginica.plot(x="sepal\_length", y="sepal\_width", kind="scatter", ax=ax, label='virginica', color='g')

plt.show()

fig,ax=plt.subplots(1,1) ## Search how plt.subplots works for more understanding

setosa.plot(x="petal\_length", y="petal\_width", kind="scatter",ax=ax,label='setosa',color='r')

versicolor.plot(x="petal\_length",y="petal\_width",kind="scatter",ax=ax,label='versicolor',color='b')

virginica.plot(x="petal\_length", y="petal\_width", kind="scatter", ax=ax, label='virginica', color='g')

plt.show()

**-----------------------------------------------------------------------------------------------**

**Model preparation// KNN Implementation**

import pandas as pd

import numpy as np

from collections import Counter ## for extracting majority label#class

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

import matplotlib.pyplot as plt

def eucli\_dist(x,y):

 return np.sqrt(np.sum((x-y)\*\*2))

class KNN:

 def \_\_init\_\_(self,Kneigh=3):

  self.k=Kneigh

 def fit(self,xdata,ylabel):

  self.X\_train=xdata

  self.Y\_train=ylabel

 def predict(self,Xdata):

  y\_predict=[self.predicttool(x) for x in Xdata]

  return np.array(y\_predict)

 def predicttool(self,val):

  distances=[eucli\_dist(val,x\_train) for x\_train in self.X\_train] ### computing euclidean distances

  kindex=np.argsort(distances)[:self.k] ### sorting the diatnces and taking the index the of first k neighbours

  k\_neigh\_labels=[self.Y\_train[i] for i in kindex] # storing labels first k neighbour training samples

  majority= Counter(k\_neigh\_labels).most\_common(1)

  return majority[0][0]

def accuracy(y\_true, y\_pred):

 accuracy = np.sum(y\_true == y\_pred) / len(y\_true)

 return accuracy

if \_\_name\_\_=="\_\_main\_\_":

 print("Machine Learning -- K Nearest Neighbours")

 full\_data=pd.read\_csv('/content/IRIS.csv')

 print(full\_data.head(10))

 Xdata=full\_data.to\_numpy()

 Xdata = Xdata[:,1:4]

 Ydata=full\_data.to\_numpy()

 Ydata = Ydata[:,4]

 X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(Xdata, Ydata, test\_size=0.2, random\_state=60)

 ### creating classifiers

 krange= range(3,50,2)

 score=[]

 for i in krange:

   clf=KNN(Kneigh=i)

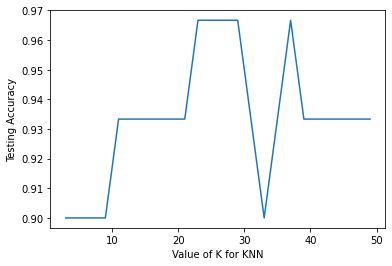
   clf.fit(X\_train,Y\_train)

   predictions = clf.predict(X\_test)

   score.append(accuracy(Y\_test, predictions))

**-----------------------------------------------------------------------------------------------**

**Graph plotting of Predictions**



***The Highest Accuracy during Testing is 96.67%.***

**Note-** All Supportive files are submitted with this document for cross verification.

**-------------------------------------------------------------------------------------------------------------**

**-- The End**

**BY-- L.A. LALIT VIYOGI**

**-------------------------------------------------------------------------------------------------------------**