# K-Nearest Neighbors

**Instructions:**

Please share your answers filled in-line in the word document. Submit code separately wherever applicable.

Please ensure you update all the details:

**Name: Anandakrishnan k v ;;;;;;;; Batch ID:**19042021

**Topic: K-Nearest Neighbors**

1. **Business Problem**

A glass manufacturing plant uses different earth elements to design new glass materials based on customer requirements. For that, they would like to automate the process of classification as it’s a tedious job to manually classify them. Help the company achieve its objective by correctly classifying the glass type based on the other features using KNN algorithm.

1. **Model Building**
   1. **Build the model on the scaled data (try multiple options).**
   2. **Perform KNN and use cross validation techniques to get optimum K value.**
   3. **Train and test the model and perform cross validation techniques. Compare accuracies, precision and recall and explain them in the documentation.**
   4. **Briefly explain the model output in the documentation.**

**Python Code:-**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import seaborn as sn

# load the data set

glass = pd.read\_csv("C://Users//user//Downloads//knn//glass.csv")

glass.shape

glass1 = glass.copy(deep=True)

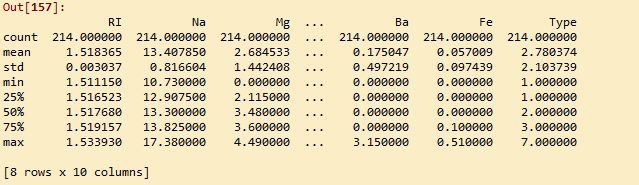
###### Null value Treatment ########

glass1.isna().sum() ## no null values

###### Summary of the data set ####

glass1.columns

glass1.describe()



####### Outlier Treatment ########

# Boxplots

glass\_bx\_input = glass1.iloc[:, 0:9] ## except output column

for i in glass\_bx\_input.columns:

sns.boxplot(glass\_bx\_input[i].dropna())

plt.show() ## combined boxplot not showig any outliers

############ Zero variance analysis #############

glass1.shape

## importing ###

from sklearn.feature\_selection import VarianceThreshold

# Feature selector that removes all low-variance features that meets the variance threshold limit

var\_thres = VarianceThreshold(threshold=0) # Threshold is subjective.

var\_thres.fit(glass1) ### fit the var\_thres to data set

# Generally we remove the columns with zero variance, but i took thresold value 0 (Near Zero Variance)

var\_thres.get\_support() ### it giving an array out, where zero variant column treat as False value. we already fit var\_thres to data set. so it gives corresponding information on data set

glass1.columns[var\_thres.get\_support()] ## non-zero variant column names

constant\_columns = [column for column in glass1.columns if column not in glass1.columns[var\_thres.get\_support()]]

print(len(constant\_columns)) ### number of zero variant variables

# since number of zero variant columns = 0 ==> none of the variables or column having zero variant property ; so directy going for further analysis without doing any zero variant treatment

# Normalization function

def norm\_func(i):

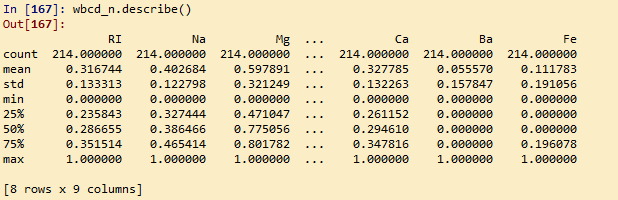
x = (i-i.min()) / (i.max()-i.min())

return (x)

# Normalized data frame (except output column)

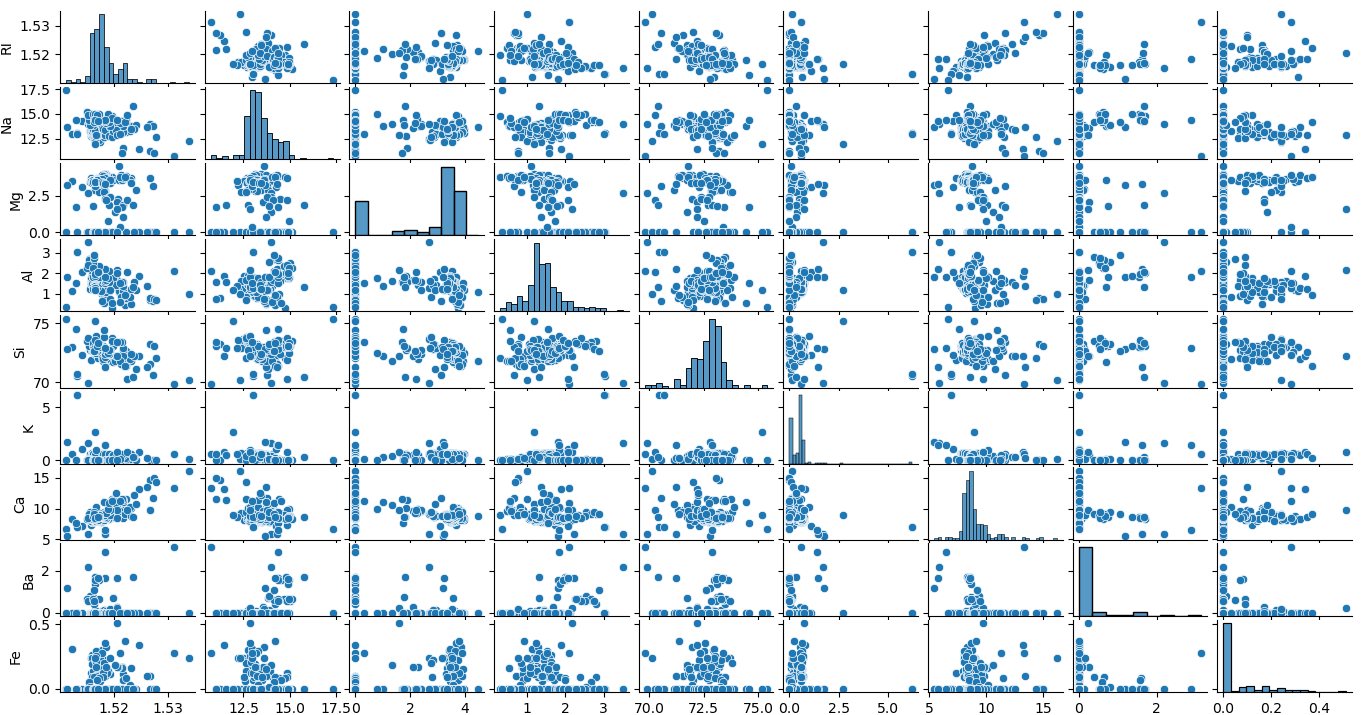
wbcd\_n = norm\_func(glass1.iloc[:, 0:9])

wbcd\_n.describe()



# univariate & byvariate plot after normalisation

sn.pairplot(glass1.iloc[:, 0:9])



X = np.array(wbcd\_n.iloc[:,:]) # Predictors

Y = np.array(glass1['Type']) # Target

# train & test split

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

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y,stratify= Y, test\_size = 0.2) ## proportion of output labels should be same

# KNeighborsClassifier

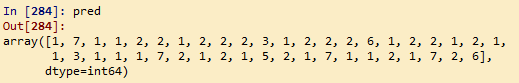
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors = 3) #consider 3 nearest neighbors

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

pred = knn.predict(X\_test)

pred

****

# Evaluate the model

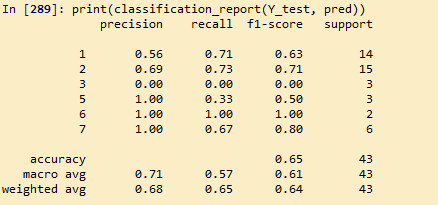
from sklearn.metrics import accuracy\_score

print(accuracy\_score(Y\_test, pred))

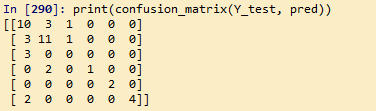
C:\Users\user\Downloads\knn\paint.png

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

print(classification\_report(Y\_test, pred))



print(confusion\_matrix(Y\_test, pred))

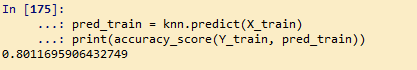


Predictions on Type ‘1’ glass falls more in other Types( 2,3 & 7)

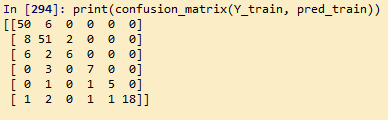
# evaluation on train data

pred\_train = knn.predict(X\_train)

print(accuracy\_score(Y\_train, pred\_train))



print(confusion\_matrix(Y\_train, pred\_train))



Compare to test prediction in train prediction happened more mis\_classification. Especially for type “2” glasses

# creating empty list variable

acc = []

train\_ac\_value = []

test\_ac\_value = []

k\_value = []

# running KNN algorithm for 3 to 50 nearest neighbours(odd numbers) and

# storing the accuracy values

for i in range(3,50,2):

neigh = KNeighborsClassifier(n\_neighbors=i)

neigh.fit(X\_train, Y\_train)

train\_acc = np.mean(neigh.predict(X\_train) == Y\_train)

test\_acc = np.mean(neigh.predict(X\_test) == Y\_test)

acc.append([train\_acc, test\_acc])

train\_ac\_value.append(train\_acc)

test\_ac\_value.append(test\_acc)

k\_value.append(i)

# accuracy table for different k value

accuracy\_table = pd.DataFrame(columns=["k\_value","train\_score", 'test\_score',"difference\_square"])

difference=[]

error=[]

l = len(acc)

for i in range(l):

difference = train\_ac\_value[i]- test\_ac\_value[i]

error.append(difference\*difference) ## to get the minimum value of error in magnitude by eliminating -ve sign

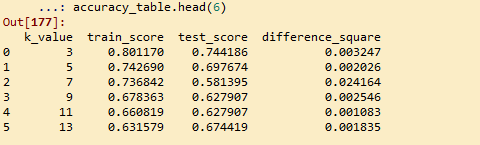
accuracy\_table.k\_value = pd.Series(k\_value)

accuracy\_table.train\_score = pd.Series(train\_ac\_value)

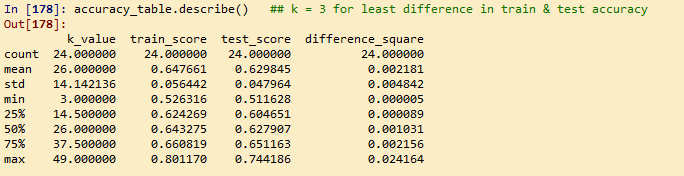
accuracy\_table.test\_score = pd.Series(test\_ac\_value)

accuracy\_table.difference\_square = pd.Series(error)

accuracy\_table.head(6)



accuracy\_table.describe() ## k = 3 for least difference in train & test accuracy



# graphical representaion of train & test accuracy with different k value

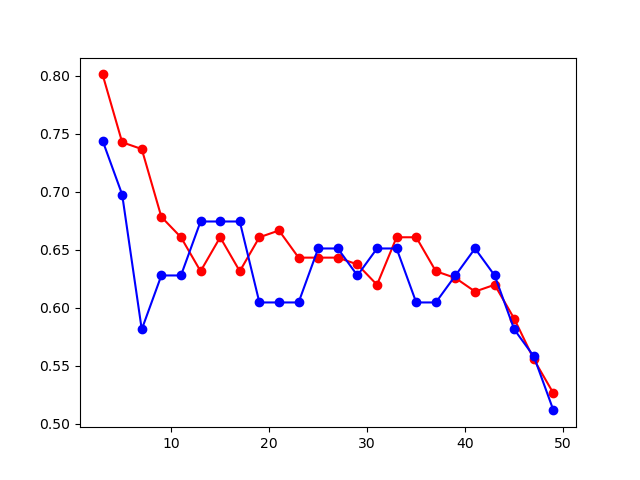
import matplotlib.pyplot as plt # library to do visualizations

# train accuracy plot

plt.plot(np.arange(3,50,2),[i[0] for i in acc],"ro-")

# test accuracy plot

plt.plot(np.arange(3,50,2),[i[1] for i in acc],"bo-")



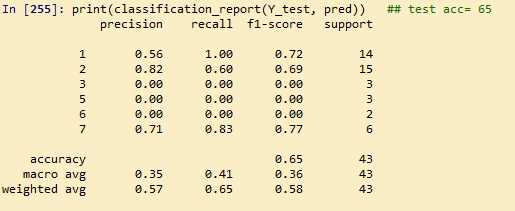
# model for k = 39

knn = KNeighborsClassifier(n\_neighbors = 39) #consider 39 nearest neighbors

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

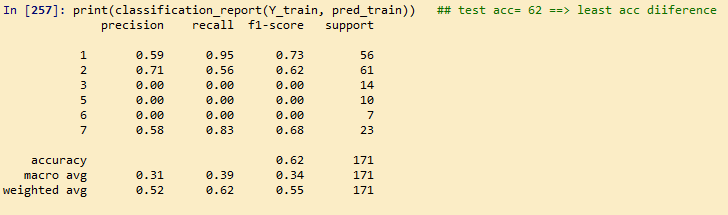
pred = knn.predict(X\_test)

print(classification\_report(Y\_test, pred)) ## test acc= 65



pred\_train = knn.predict(X\_train) # evaluation on train data

print(classification\_report(Y\_train, pred\_train)) ## test acc= 62 ==> least acc diiference



### checking accuracy while applying cross validation

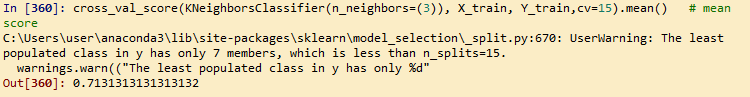
from sklearn.model\_selection import cross\_val\_score

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, stratify= Y, test\_size = 0.2) ## proportion of output labels would be sames for each split

cross\_val\_score(KNeighborsClassifier(n\_neighbors=(3)), X\_train, Y\_train, cv=4)

C:\Users\user\Downloads\knn\paint.png

cross\_val\_score(KNeighborsClassifier(n\_neighbors=(3)), X\_train, Y\_train,cv=4).mean() # mean score



Q2)

**Business Problem**

A National Park in India is dealing with the problem of segregation of its species based on the different attributes they have. Build a KNN model to automatically classify new species. Explain any inferences you draw in the documentation.

**Python Code:-**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

# load the data set

zoo = pd.read\_csv("C://Users//user//Downloads//knn//Zoo.csv")

zoo.shape

zoo1 = zoo.copy(deep=True)

###### Null value Treatment ########

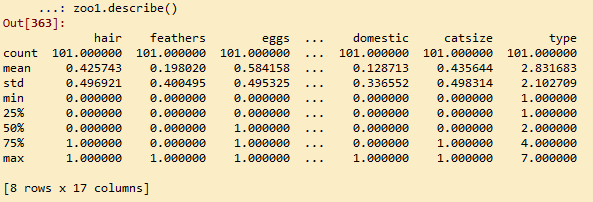
zoo1.isna().sum() ## no null values

zoo1 = zoo1.iloc[:, 1:] # Excluding animal name column ==> less informative

###### Summary of the data set ####

zoo1.columns

zoo1.describe()



####### Outlier Treatment ########

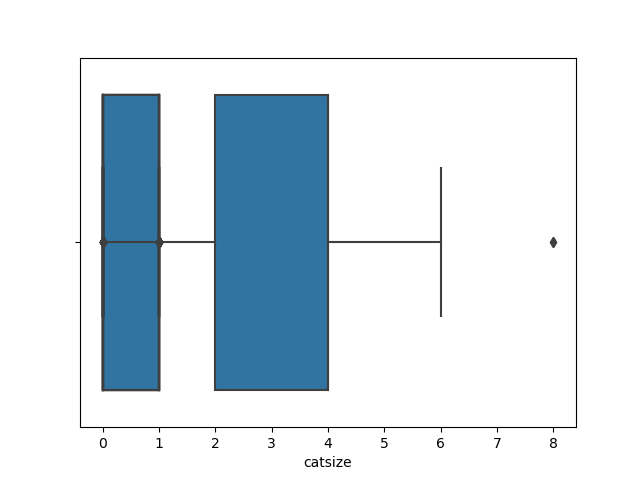
# Boxplots

zoo\_bx\_input = zoo1.iloc[:, 0:16] ## except output column

for i in zoo\_bx\_input.columns:

sns.boxplot(zoo\_bx\_input[i].dropna())

plt.show() ## combined boxplot not showig any outliers



############ Zero variance analysis #############

zoo1.shape

## importing ###

from sklearn.feature\_selection import VarianceThreshold

# Feature selector that removes all low-variance features that meets the variance threshold limit

var\_thres = VarianceThreshold(threshold=0) # Threshold is subjective.

var\_thres.fit(zoo1) ### fit the var\_thres to data set

# Generally we remove the columns with zero variance, but i took thresold value 0 (Near Zero Variance)

var\_thres.get\_support() ### it giving an array out, where zero variant column treat as False value. we already fit var\_thres to data set. so it gives corresponding information on data set

zoo1.columns[var\_thres.get\_support()] ## non-zero variant column names

constant\_columns = [column for column in zoo1.columns if column not in zoo1.columns[var\_thres.get\_support()]]

print(len(constant\_columns)) ### number of zero variant variables

# since number of zero variant columns = 0 ==> none of the variables or column having zero variant property ; so directy going for further analysis without doing any zero variant treatment

# Normalization function

def norm\_func(i):

x = (i-i.min()) / (i.max()-i.min())

return (x)

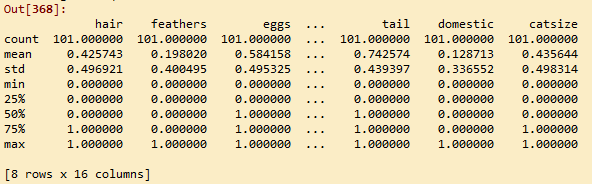
# Normalized data frame (except output)

wbcd\_n = norm\_func(zoo1.iloc[:, 0:16])

wbcd\_n.describe()

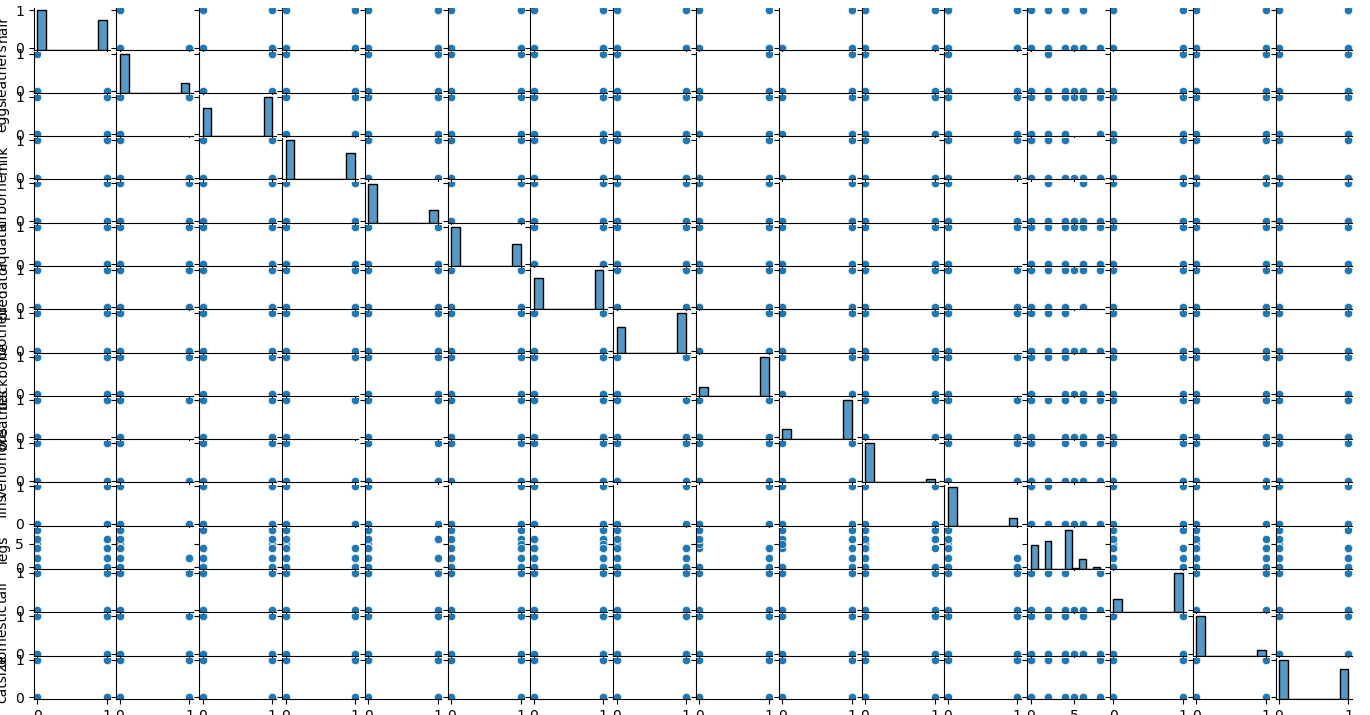
# univariate & byvariate plot after normalisation

sn.pairplot(zoo1.iloc[:, 0:16])



# univariate & byvariate plot after normalisation

sn.pairplot(zoo1.iloc[:, 0:16])



# Differentiate predictors (input variables) & targeters(output variable)

X = np.array(wbcd\_n.iloc[:,:]) # Predictors

Y = np.array(zoo1['type']) # Target

# train & test split

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

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.2)

# KNeighborsClassifier

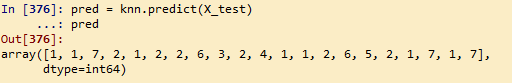
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors = 5) #consider 5 nearest neighbors

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

pred = knn.predict(X\_test)

pred

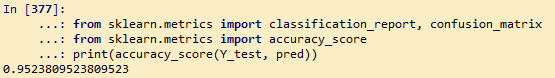


# Evaluate the model

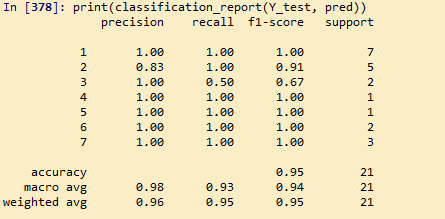
from sklearn.metrics import classification\_report, confusion\_matrix

from sklearn.metrics import accuracy\_score

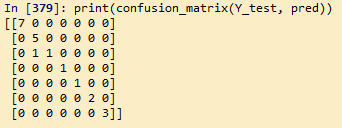
print(accuracy\_score(Y\_test, pred))



print(classification\_report(Y\_test, pred))



print(confusion\_matrix(Y\_test, pred))

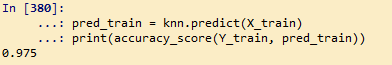


Misclassification on test data prediction happeneds only for Type “2” species, that’s even only one

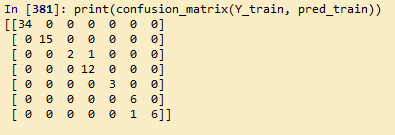
# error on train data

pred\_train = knn.predict(X\_train)

print(accuracy\_score(Y\_train, pred\_train))



print(confusion\_matrix(Y\_train, pred\_train))



# creating empty list variable

acc = []

train\_ac\_value = []

test\_ac\_value = []

k\_value = []

# running KNN algorithm for 3 to 50 nearest neighbours(odd numbers) and

# storing the accuracy values

for i in range(3,50,2):

neigh = KNeighborsClassifier(n\_neighbors=i)

neigh.fit(X\_train, Y\_train)

train\_acc = np.mean(neigh.predict(X\_train) == Y\_train)

test\_acc = np.mean(neigh.predict(X\_test) == Y\_test)

acc.append([train\_acc, test\_acc])

train\_ac\_value.append(train\_acc)

test\_ac\_value.append(test\_acc)

k\_value.append(i)

# accuracy table for different k value

accuracy\_table = pd.DataFrame(columns=["k\_value","train\_score", 'test\_score',"difference\_square"])

difference=[]

error=[]

l = len(acc)

for i in range(l):

difference = train\_ac\_value[i]- test\_ac\_value[i]

error.append(difference\*difference) ## to get the minimum value of error in magnitude by eliminating -ve sign

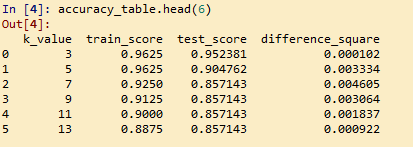
accuracy\_table.k\_value = pd.Series(k\_value)

accuracy\_table.train\_score = pd.Series(train\_ac\_value)

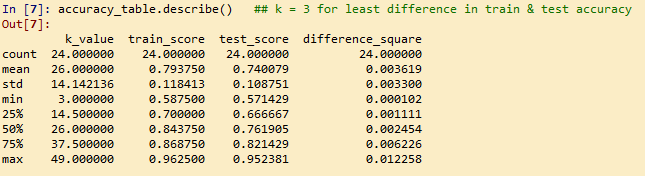
accuracy\_table.test\_score = pd.Series(test\_ac\_value)

accuracy\_table.difference\_square = pd.Series(error)

accuracy\_table.head(6)



accuracy\_table.describe() ## k = 39 for least difference in train & test accuracy



# graphical representaion of train & test accuracy with different k value

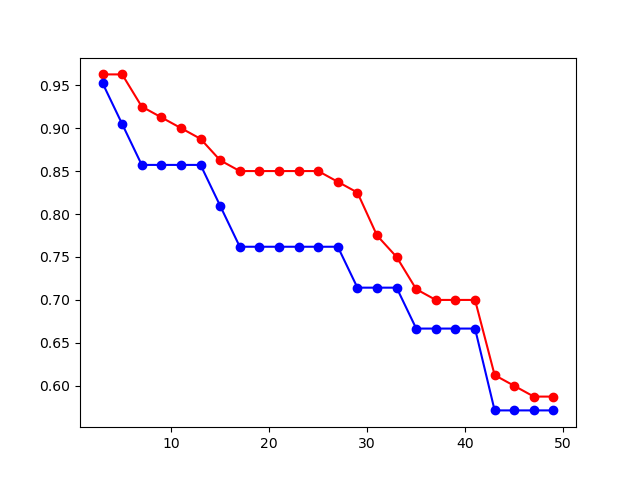
import matplotlib.pyplot as plt # library to do visualizations

# train accuracy plot

plt.plot(np.arange(3,50,2),[i[0] for i in acc],"ro-")

# test accuracy plot

plt.plot(np.arange(3,50,2),[i[1] for i in acc],"bo-")



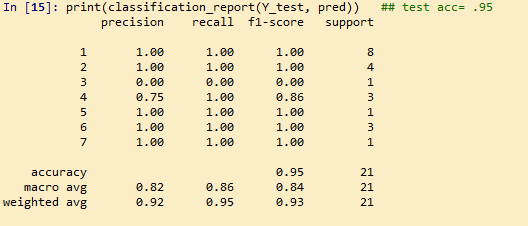
# model for k = 3

knn = KNeighborsClassifier(n\_neighbors = 3) #consider 3 nearest neighbors

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

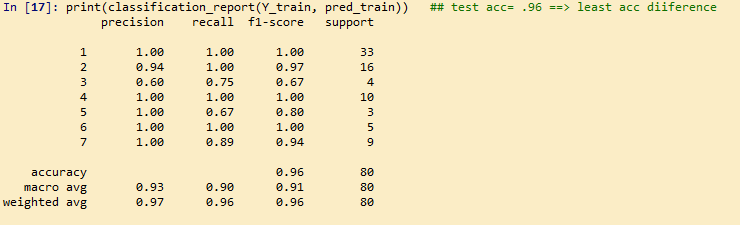
pred = knn.predict(X\_test)

print(classification\_report(Y\_test, pred)) ## test acc= .95



pred\_train = knn.predict(X\_train) # evaluation on train data

print(classification\_report(Y\_train, pred\_train)) ## test acc= .96 ==> least acc diiference

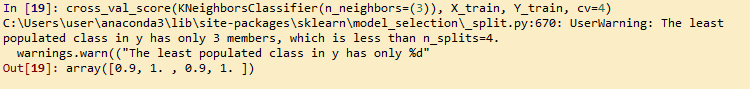


### checking accuracy while applying cross validation

from sklearn.model\_selection import cross\_val\_score

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, stratify= Y, test\_size = 0.2) ## proportion of output labels would be sames for each split

cross\_val\_score(KNeighborsClassifier(n\_neighbors=(3)), X\_train, Y\_train, cv=4)



cross\_val\_score(KNeighborsClassifier(n\_neighbors=(3)), X\_train, Y\_train,cv=15).mean() # mean score

