**LAB - 4**

**K-NEAREST NEIGHBOUR**

**PRIYANSHU SHARMA**

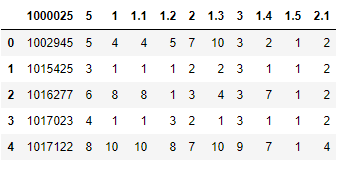
**15BCE1282**

**CODE**

import pandas as pd

data = pd.read\_csv('C:/Users/PRIYANSHU SHARMA/Desktop/PRIYANSHU/6 STUDY/6 SEMSTER/MACHINE LEARNING/LAB/breast.csv')

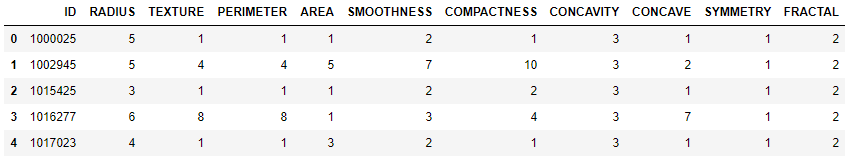
data.head()



colnames=['ID', 'RADIUS', 'TEXTURE', 'PERIMETER', 'AREA', 'SMOOTHNESS', 'COMPACTNESS', 'CONCAVITY', 'CONCAVE', 'SYMMETRY', 'FRACTAL']

data = pd.read\_csv('C:/Users/PRIYANSHU SHARMA/Desktop/PRIYANSHU/6 STUDY/6 SEMSTER/MACHINE LEARNING/LAB/breast.csv', names=colnames, header=None)

data.head()



from sklearn.cross\_validation import train\_test\_split

X = data.iloc[0:, [1,7]].values

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,data['FRACTAL'], test\_size=0.3, random\_state=0)

from matplotlib.colors import ListedColormap

def plot\_decision\_regions(X, y, classifier,test\_idx=None, resolution=0.02):

# setup marker generator and color map

markers = ('s', 'x', 'o', '^', 'v')

colors = ('green', 'red', 'lightgreen', 'gray', 'cyan')

cmap = ListedColormap(colors[:len(np.unique(y))])

# plot the decision surface

x1\_min, x1\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

x2\_min, x2\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

xx1, xx2 = np.meshgrid(np.arange(x1\_min, x1\_max, resolution),

np.arange(x2\_min, x2\_max, resolution))

Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)

Z = Z.reshape(xx1.shape)

plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)

plt.xlim(xx1.min(), xx1.max())

plt.ylim(xx2.min(), xx2.max())

# plot class samples

for idx, cl in enumerate(np.unique(y)):

plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],

alpha=0.8, c=cmap(idx),

marker=markers[idx], label='Benign' if cl == 2 else 'Malignant')

from sklearn.neighbors import KNeighborsClassifier

import numpy as np

import matplotlib.pyplot as plt

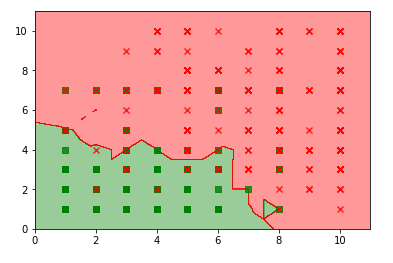
knn = KNeighborsClassifier(n\_neighbors=9, p=2, metric='minkowski')

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

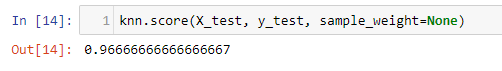
plot\_decision\_regions(X,data['FRACTAL'],

classifier=knn, test\_idx=range(105,150))

plt.show()



knn.score(X\_test, y\_test, sample\_weight=None)



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

myList = list(range(1,50))

# subsetting just the odd ones

neighbors = list(filter(lambda x: x % 2 != 0, myList))

# empty list that will hold cv scores

cv\_scores = []

# perform 10-fold cross validation

for k in neighbors:

knn = KNeighborsClassifier(n\_neighbors=k, p=2,metric='minkowski')

scores = cross\_val\_score(knn, X\_train, y\_train, cv=10, scoring='accuracy')

cv\_scores.append(scores.mean())

# changing to misclassification error

MSE = [1 - x for x in cv\_scores]

# determining best k

optimal\_k = neighbors[MSE.index(min(MSE))]

print("The optimal number of neighbors is:"+str(optimal\_k))

# plot misclassification error vs k

plt.plot(neighbors, MSE)

plt.xlabel('Number of Neighbors K')

plt.ylabel('Misclassification Error')

plt.show()

