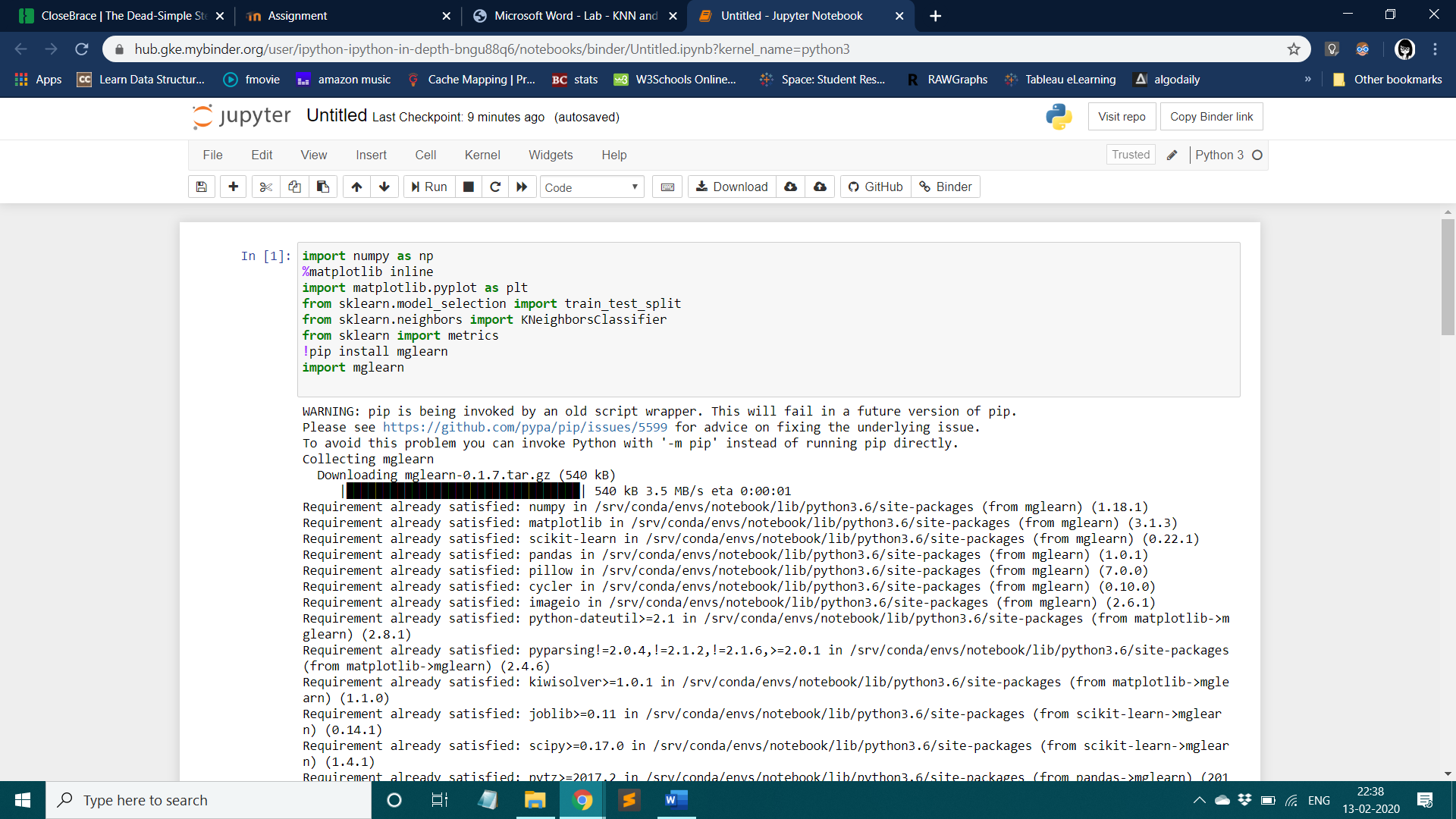
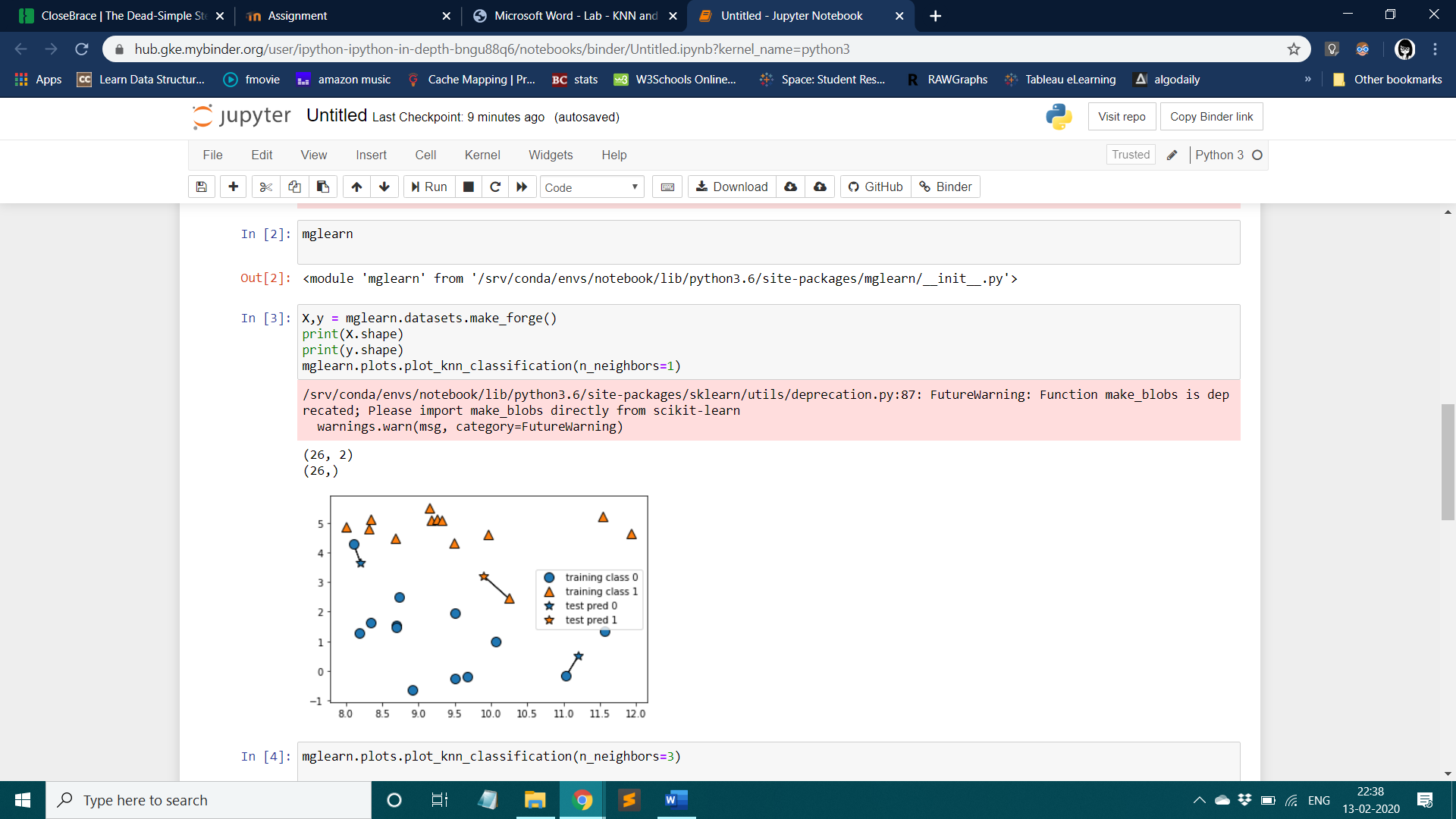
**Lab 6**

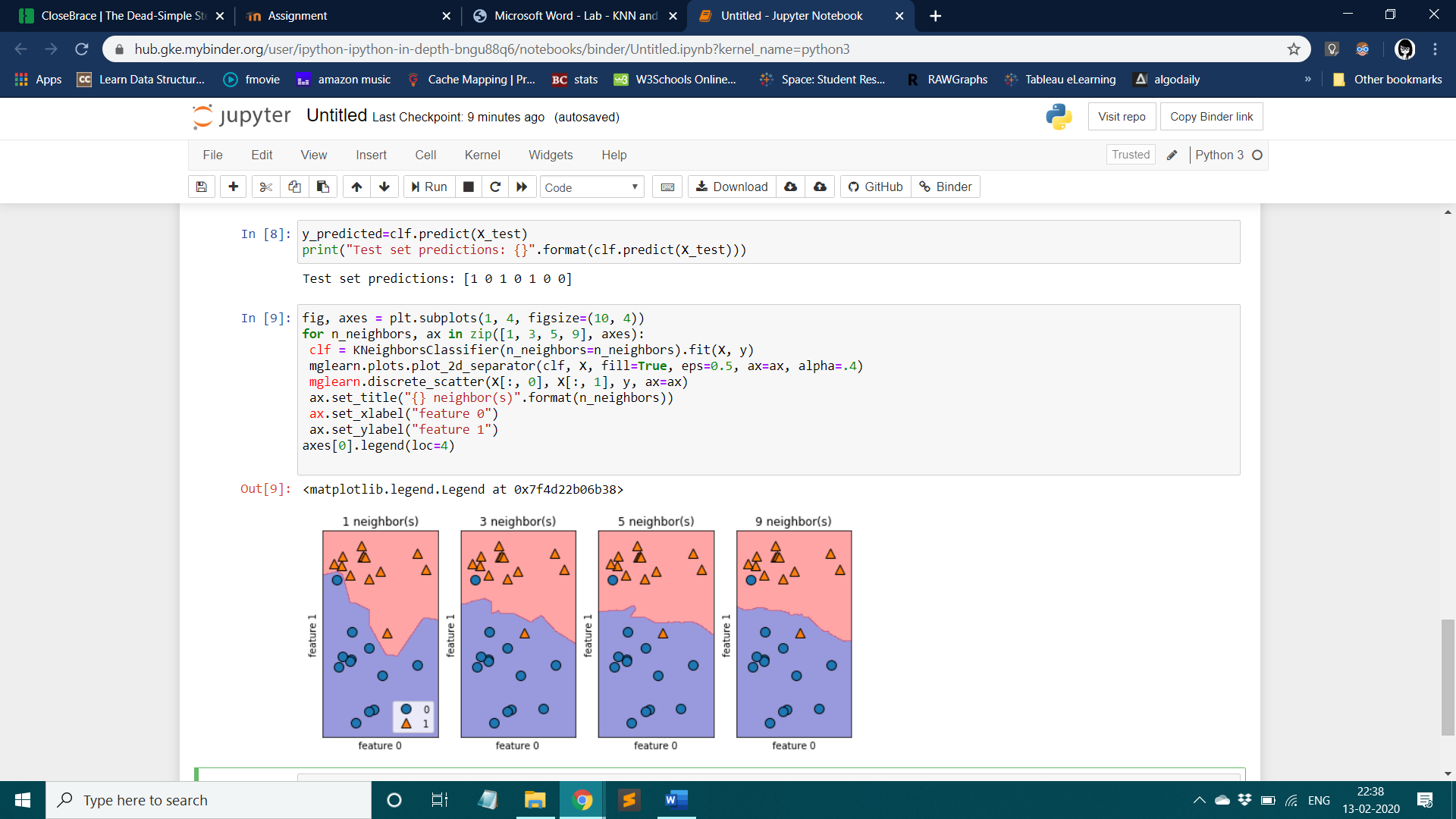
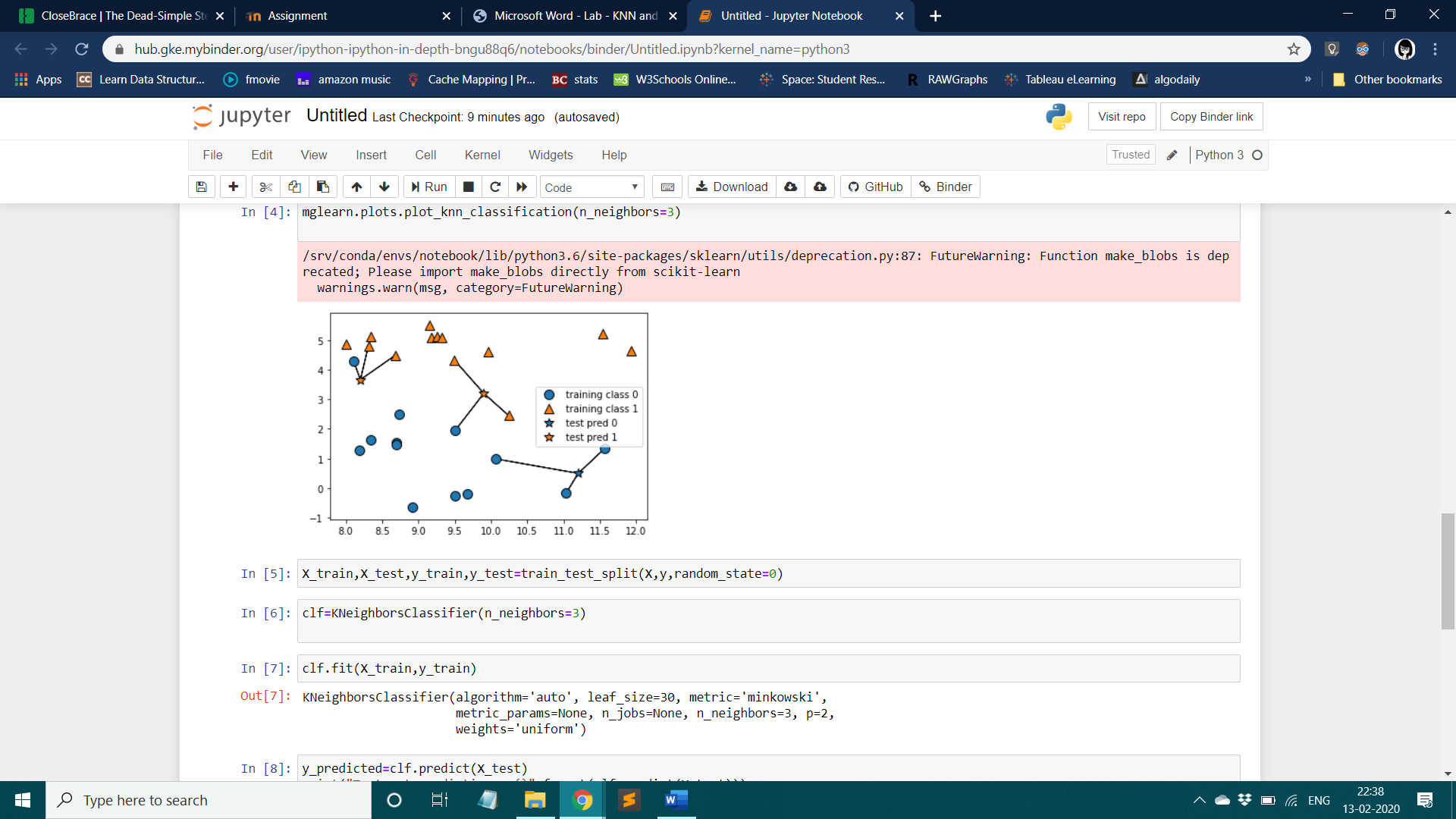
**R Harini**

**18BCE1010**

**KNN Classifier**







df=pd.read\_csv('cancer.csv')

df

df= df.drop(columns=['id','Unnamed: 32'])

df.head()

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

array=df.values

df.head()

X=array[:,1:33]

y=array[:,[0]]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

X\_train.shape,X\_test.shape

clf=KNeighborsClassifier(n\_neighbors=3)

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

acc\_train = clf.score(X\_train, y\_train)

print('Train set accuracy: ', acc\_train)

acc\_train = clf.score(X\_test, y\_test)

print('Train set accuracy: ', acc\_train)

training\_accuracy =[]

testing\_accuracy = []

neighbors\_settings = range(1, 10)

for n\_neighbors in neighbors\_settings:

clf = KNeighborsClassifier(n\_neighbors=n\_neighbors)

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

training\_accuracy.append(clf.score(X\_train, y\_train))

testing\_accuracy.append(clf.score(X\_test, y\_test))

plt.plot(neighbors\_settings, training\_accuracy, label="training accuracy")

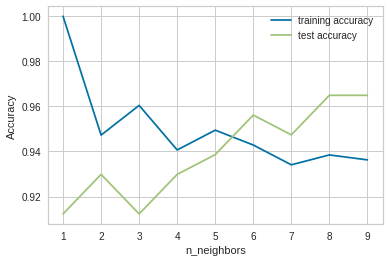
plt.plot(neighbors\_settings, testing\_accuracy, label="test accuracy")

plt.ylabel("Accuracy")

plt.xlabel("n\_neighbors")

plt.legend()

plt.show()



**KMeans Clustering**  
  
  
from sklearn.cluster import KMeans

import numpy as np

df=pd.read\_csv('protein.csv')

df

X = df.iloc[:,1:].values

kmeans = KMeans(n\_clusters=4)

y\_kmeans = kmeans.fit\_predict(X)

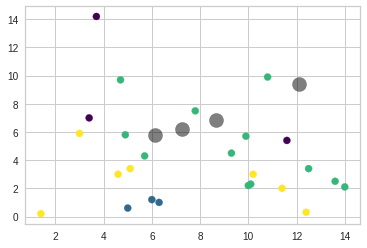
print(y\_kmeans5)

kmeans.cluster\_centers\_

plt.scatter(X[:, 1], X[:,4], c=y\_kmeans, s=50, cmap='viridis')

centers = kmeans.cluster\_centers\_

plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5);



from sklearn.metrics import pairwise\_distances\_argmin

def find\_clusters(X, n\_clusters, rseed=2):

# 1. Randomly choose clusters

rng = np.random.RandomState(rseed)

i = rng.permutation(X.shape[0])[:n\_clusters]

centers = X[i]

while True:

# 2a. Assign labels based on closest center

labels = pairwise\_distances\_argmin(X, centers)

# 2b. Find new centers from means of points

new\_centers = np.array([X[labels == i].mean(0)

for i in range(n\_clusters)])

# 2c. Check for convergence

if np.all(centers == new\_centers):

break

centers = new\_centers

return centers, labels

Error =[]

for i in range(1, 11):

kmeans = KMeans(n\_clusters = i).fit(X)

kmeans.fit(X)

Error.append(kmeans.inertia\_)

import matplotlib.pyplot as plt

plt.plot(range(1, 11), Error)

plt.title('Elbow method')

plt.xlabel('No of clusters')

plt.ylabel('Error')

plt.show()

