**Ahsanullah University of Science and Technology**



Department of Computer Science and Engineering

Program: Bachelor of Science in Computer Science and Engineering

Course No: CSE 4108

Course Title: Artificial Intelligence Lab

Assignment No: 03

Date of Submission: 08/08/2022

Submitted to:

Mr. Md. Siam Ansary

Lecturer, Department of CSE, AUST.

Ms. Tamanna Tabassum

Lecturer, Department of CSE, AUST.

Submitted by,

Name: S. M. Tasnimul Hasan

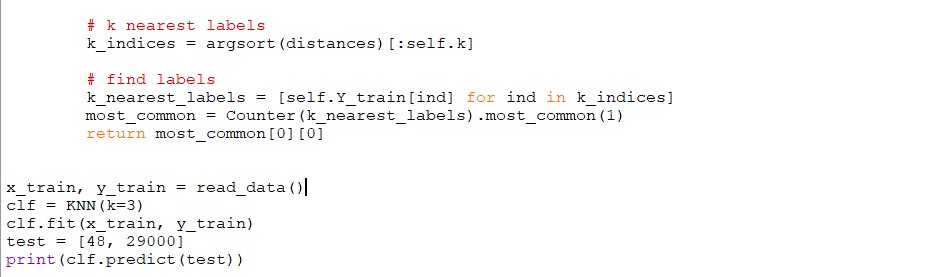
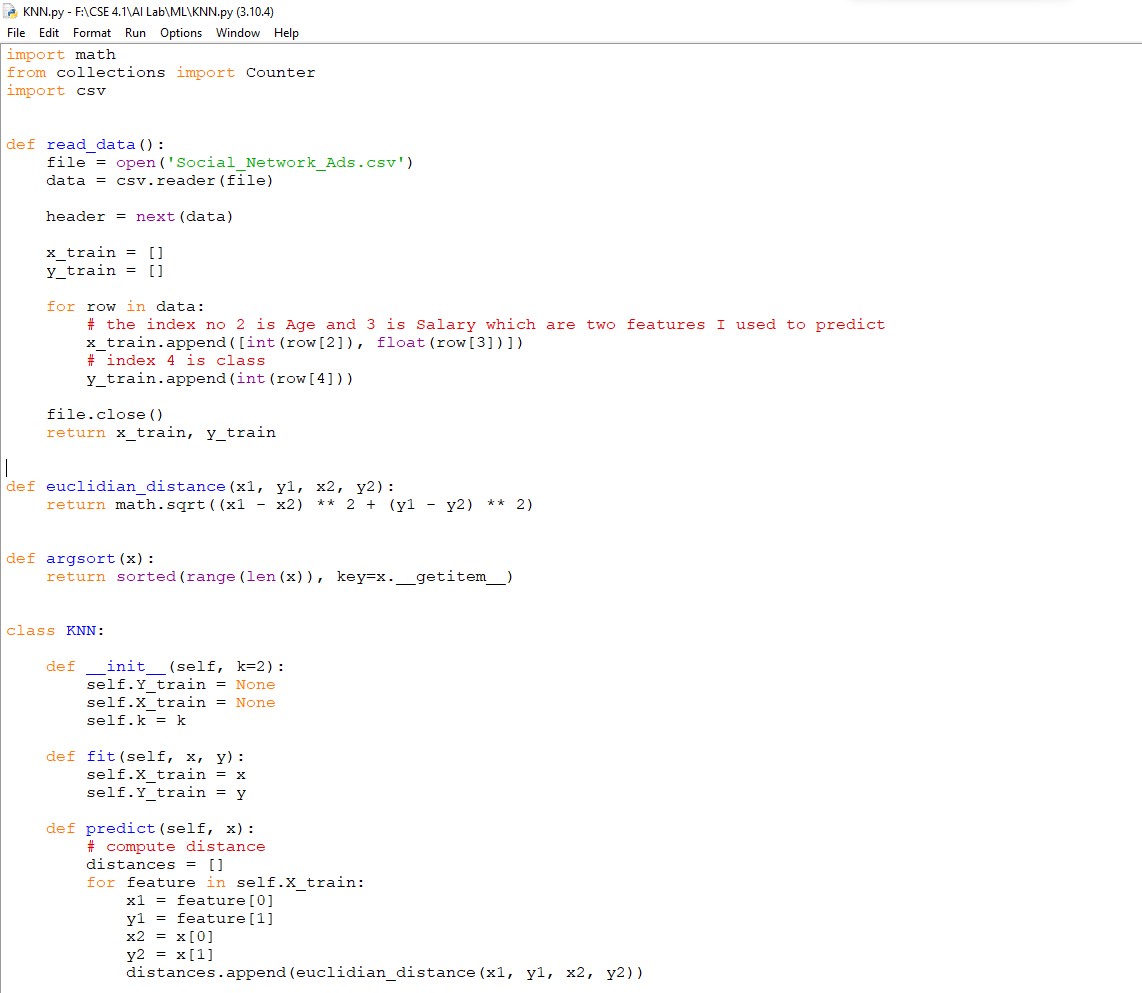
Student ID: 180204142

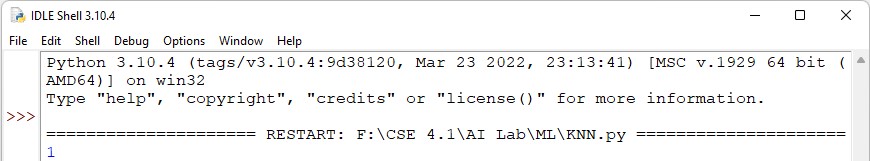
**Question 1**: Implement K Nearest Neighbor classifier in Python.

**Solution:**

Python Code:

**import** math  
**from** collections **import** Counter  
**import** csv  
  
**def** read\_data():  
file = open(**'Social\_Network\_Ads.csv'**)  
data = csv.reader(file)  
header = next(data)  
x\_train = []  
y\_train = []  
**for** row **in** data:  
*# the index no 2 is Age and 3 is Salary which are two features I*used to predict  
x\_train.append([int(row[2]), float(row[3])])  
*# index 4 is class*y\_train.append(int(row[4]))  
file.close()  
**return** x\_train, y\_train  
  
**def** euclidian\_distance(x1, y1, x2, y2):  
**return** math.sqrt((x1 - x2) \*\* 2 + (y1 - y2) \*\* 2)  
  
**def** argsort(x):  
**return** sorted(range(len(x)), key=x.\_\_getitem\_\_)  
  
**class** KNN:  
**def** \_\_init\_\_(self, k=2):  
self.Y\_train = **None**self.X\_train = **None**self.k = k  
**def** fit(self, x, y):  
self.X\_train = x  
self.Y\_train = y  
**def** predict(self, x):  
*# compute distance*distances = []  
**for** feature **in** self.X\_train:  
x1 = feature[0]  
y1 = feature[1]  
x2 = x[0]  
y2 = x[1]  
distances.append(euclidian\_distance(x1, y1, x2, y2))  
*# k nearest labels*k\_indices = argsort(distances)[:self.k]  
*# find labels*k\_nearest\_labels = [self.Y\_train[ind] **for** ind **in** k\_indices]  
most\_common = Counter(k\_nearest\_labels).most\_common(1)  
**return** most\_common[0][0]  
  
x\_train, y\_train = read\_data()  
clf = KNN(k=3)  
clf.fit(x\_train, y\_train)  
test = [48, 29000]  
print(clf.predict(test))





**Question 2**: Implement K Means Clustering algorithm in Python.

**Solution:**

Python Code:

**import** csv  
**import** random  
**import** math  
**import** matplotlib.pyplot **as** plt  
  
**def** euclidian\_distance(x1, y1, x2, y2):  
**return** math.sqrt((x1 - x2) \*\* 2 + (y1 - y2) \*\* 2)  
  
**def** calc\_mean(data\_points):  
sumx = 0  
sumy = 0  
n = len(data\_points)  
**for** x **in** data\_points:  
sumx = sumx + x[0]  
sumy = sumy + x[1]  
  
**return** [sumx / n, sumy / n]  
  
**class** Kmean:  
**def** \_\_init\_\_(self, k=1, max\_itr=1000):  
self.k = k  
self.centroid = **None**self.max\_itr = max\_itr  
  
**def** fit(self, x):  
rand\_ind = random.sample(range(0, len(x)), self.k)  
self.centroid = [x[ind] **for** ind **in** rand\_ind]  
  
**for** i **in** range(0, self.max\_itr):  
clusters = self.assign\_cluster(x)  
prev\_centroid = self.centroid  
self.centroid = self.move\_centroid(x, clusters)  
**if** self.centroid == prev\_centroid:  
**break  
return** clusters  
  
**def** assign\_cluster(self, x):  
distance = []  
clusters = []  
**for** row **in** x:  
**for** centroid **in** self.centroid:  
x1 = row[0]  
y1 = row[1]  
  
x2 = centroid[0]  
y2 = centroid[1]  
distance.append(euclidian\_distance(x1, y1, x2, y2))  
min\_distance = min(distance)  
min\_ind = distance.index(min\_distance)  
clusters.append(min\_ind)  
distance.clear()  
**return** clusters  
  
**def** move\_centroid(self, x, clusters):  
unique\_clusters = list(set(clusters))  
new\_centroid = []  
**for** cluster **in** unique\_clusters:  
temp = []  
**for** ind **in** range(0, len(clusters)):  
**if** cluster == clusters[ind]:  
temp.append(x[ind])  
new\_centroid.append(calc\_mean(temp))  
**return** new\_centroid  
  
**def** read\_csv():  
file = open(**'custering.csv'**)  
data = csv.reader(file)  
  
header = next(data)  
x\_train = []  
**for** row **in** data:  
  
x\_train.append([float(row[0]), float(row[1])])  
  
file.close()  
**return** x\_train  
  
x\_train = read\_csv()  
kmean = Kmean(k=3)  
y\_mean = kmean.fit(x\_train)  
*# print(y\_mean)*t1 = []  
t2 = []  
t3 = []  
  
**for** x **in** range(0, 2):  
**for** ind **in** range(0, len(x\_train)):  
**if** y\_mean[ind] == x:  
**if** x == 0:  
t1.append(x\_train[ind])  
**elif** x == 1:  
t2.append(x\_train[ind])  
**else**:  
t3.append(x\_train[ind])  
  
*# plt.scatter([p[0] for p in t1], [p[1] for p in t1], color='green')  
# plt.scatter([p[0] for p in t2], [p[1] for p in t2], color='red')  
# plt.scatter([p[0] for p in t3], [p[1] for p in t3], color='blue')  
  
# plt.show()*print(**"Cluster: 0"**)  
print(t1)  
print(**"Cluster: 1"**)  
print(t2)  
print(**"Cluster: 2"**);  
print(t3)

