תרגיל בית 5 דווח

קודם רצינו להגיד שהקורס היה מדהים נהנינו מאוד מהעבודה היה ממש אולי הקורס הכי מעניין ויפה בתואר קצת עמוס אבל מעניין ממש למדנו הרבה שי תמיד עונה על השאלות ומתייחס לסטודנטים ונותן הרבה הארכות אחלה מרצה , וגם שובל אחלה בודקת יכולים לראות ההתייחסות שלך הרבה בבדיקת תרגילי בית תודה מאוד על הקורס ❤❤

סעיף א+ב:

כדי לקראו את הדאטה עשינו מחלקה הבאה שבא לקחנו את הpath של הקובץ קראנו אותו ואז שמנו לכל סוג ערך קוד בין 0 ל 5 ואז הדפסנו את ההתפלגות של הערכים שקיבלנו :

```
class CSV:
    def __init__(self,path):
        self.path=path
        self.data,self.laber=self.read_data()

    def read_data(self):
        file = pandas.read_csv("data\\" + self.path, header=None)
        file = file.values
        data, label = file[:, 1:-1], file[:, -1]

        label = LabelEncoder().fit_transform(label)
        print(f'data: {data}')
        print(f'label: {label}')
        array=[0,0,0,0,0,0]
        for i in label:
            #print (i)
            array[i]+=1
        print("Distribution is : ",array)
        #print(len(label))
        return data, label
```

דוגמא לתוצאה:

```
C:\Users\win10\AppData\Local\Programs\Python\Python310\python.exe C:/Users/win10/PycharmProjects/LAB-5-B/mai
data: [[ 1.52101 13.64     4.49    ...  8.75     0.       0.     ]
 [ 1.51761 13.89     3.6     ...  7.83     0.       0.     ]
 [ 1.51618 13.53     3.55    ...  7.78     0.       0.     ]
 ...
 [ 1.52065 14.36     0.      ...  8.44     1.64     0.     ]
 [ 1.51651 14.38     0.      ...  8.48     1.57     0.     ]
 [ 1.51711 14.23     0.      ...  8.62     1.67     0.     ]]
label: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4
 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5]
Distribution is :  [70, 76, 17, 13, 9, 29]
```

סעיף ג+ד+ה:

כדי לנרמל את הדאטה השתמשנו בספרייה MINMAXSCALLER וכדי לחלק ל קבוצת אימון ובדיקה השתמשנו ב train_test_split באופן הבא

```
data, label = mycsv.read_data()
normalized = DataFrame(MinMaxScaler().fit_transform(data))

train, test, train_vec, test_vec = train_test_split(normalized, label, stratify=label, test_size=0.2,
                                                    random_state=1)
```

ואז כדי לעשות רשת MLP השתמשנו בספריית MLP CLASSIFIER באופן הבא

```
mlp = MLPClassifier(random_state=1, max_iter=max_iter).fit(train, train_vec)

predict = mlp.predict_proba(test)
```

```
[[2.08651798e-01 7.70514954e-01 2.08009670e-02 1.33331727e-07
  2.88028384e-05 3.34440594e-06]
 [2.85385403e-01 5.33155767e-01 5.97557774e-03 2.67288550e-03
  7.95513604e-02 9.32590058e-02]
 [1.36360647e-02 9.79066050e-01 7.29629125e-03 1.56899461e-06
  1.86752335e-09 2.29355051e-08]
 [1.84000702e-01 3.89635232e-01 4.25804916e-01 3.88922040e-06
  5.10143885e-04 4.51167369e-05]
 [1.78637464e-12 6.12286646e-04 2.25456937e-13 1.51736854e-04
  1.93928926e-03 9.97296687e-01]
 [9.35644693e-01 2.26088382e-03 6.20918223e-02 7.50992245e-08
```

Run TODO Problems Terminal Python Packages Python Console

סעיף ו+ז:

```
def calc_acc(self,train,train_vec,test,test_vec):
    mlp = MLPClassifier(random_state=1, max_iter=8000000).fit(train, train_vec)
    #print(mlp)
    predict = mlp.predict_proba(test)
    #print(predict)
    predicted_labels=self.calc_softmax_pred(predict)
    number_of_answer,number_of_correct =self.calc_micro(predicted_labels,test_vec)
    self.calc_macro(number_of_answer,number_of_correct)
```

```python
    def calc_micro(self,predicted_labels,test_vec):
        answers = [0, 0, 0, 0, 0, 0]
        results = [0, 0, 0, 0, 0, 0]
        TP = 0
        for predection, test in zip(predicted_labels, test_vec):
            answers[predection] += 1
            if predection == test:
                results[predection] += 1
                TP += 1
        print(f'Micro Result is: {TP / 43}')
        return answers,results

    def calc_macro(self,number_of_answer,number_of_correct):
        final_result = 0
        for answer, result in zip(number_of_answer, number_of_correct):
            final_result += (result/answer)

        print(f'Macro result is : {final_result / 6}')
```

```python
def calc_softmax_pred(self,predictions):
    predicted_labels = []
    counter=0
    for x in predictions:
        index = 0
        max_index = 0
        max = -1
        max_array = np.exp(x) / np.sum(np.exp(x), axis=0)

        print(f'{counter}:   {max_array}')
        for j in max_array:
            max,max_index=self.update_max(max,j,index,max_index)
            index += 1

        predicted_labels.append(max_index)
        counter += 1
    print(f'softmax result is::   {predicted_labels}')
    return predicted_labels
```

ממשנו את מה שנדרש באמצעות הפונקציות האלה ואז דוגמא לתוצאה:

```
34:   [0.2988995  0.1626316  0.1354552  0.13432171 0.13433495 0.13435705]
35:   [0.31809808 0.14964678 0.13377268 0.13281958 0.13282937 0.13283352]
36:   [0.14895995 0.31999104 0.13306951 0.13264943 0.13265527 0.13267481]
37:   [0.33142786 0.14062765 0.13302672 0.1316369  0.13164037 0.13164049]
38:   [0.19215037 0.21206759 0.17851337 0.13896277 0.13932721 0.13897868]
39:   [0.13273049 0.34207214 0.133246   0.13060754 0.13063763 0.13070621]
40:   [0.12959166 0.12961016 0.12959166 0.12959538 0.35191063 0.12970051]
41:   [0.33034836 0.13185453 0.14260814 0.1317229  0.13174309 0.13172297]
42:   [0.3521523  0.12957914 0.12956978 0.12956629 0.12956627 0.12956623]
softmax result is::   [1, 1, 1, 2, 5, 0, 1, 3, 0, 0, 0, 0, 1, 0, 0, 3, 1, 1, 3, 0, 5, 1, 5, 5, 1, 1, 0, 5, 1, 0, 5, 0, 1, 0, 0, 0, 1, 0, 1, 1, 4, 0, 0]
Micro Result is: 0.7209302325581395
Macro result is : 0.7189542483660131
```

שאר הסעיפים:

עשינו מחלקה לאלגוריתם הגנתי

```
class GA:
    def __init__(self, popSize, maxIter, X_train, Y_train, X_test, Y_test):
        self.population = []
        self.buffer = []
        self.popSize = popSize
        self.eliteRate = 0.1
        self.maxIter = maxIter
        self.X_train = X_train
        self.Y_train = Y_train
        self.X_test = X_test
        self.Y_test = Y_test

        self.initPopulation()

    def initPopulation(self):
        for _ in range(self.popSize):
            self.population.append(Citizen())
            self.buffer.append(Citizen())
```

כך שכל אחד באוכלוסייה יש לו פיטנס וגם יש לו הרשת שלו

הפיטנס שלנו היה חישוב של accuracy שנעשה באופן הבא:

```
def calcFitness(self):
    for i in range(self.popSize):
        cls = MLPClassifier(hidden_layer_sizes=self.population[i].network.hidden, max_iter=
                            activation=self.population[i].network.activateFunction, solver=
        cls.fit(self.X_train, self.Y_train)


        sum = confusion_matrix(cls.predict(self.X_test), self.Y_test).sum()
        trace_sum =confusion_matrix(cls.predict(self.X_test), self.Y_test).trace()
        self.population[i].fitness = trace_sum / sum
```

ואז בגלל לעשות mateהשתמשנו ב single point cross over

```
def mate(self):
    for i in range(self.popSize):
        i1 = random.randint(0, self.popSize - 1)
        i2 = random.randint(0, self.popSize - 1)
        while i2 == i1:
            i2 = random.randint(0, self.popSize - 1)
        spos = random.randint(0, min(self.population[i1].network.depth, self.population[i2].net
        self.buffer[i].network.hidden = self.population[i1].network.hidden[0: spos] + self.popu
                                                                                          i2].n
        self.buffer[i].network.depth = len(self.buffer[i].network.hidden)
        if random.random() < 0.25* random.random():
            self.mutate(self.buffer[i])

def mutate(self,member):
    ipos = random.randint(0, member.network.depth)
    delta = random.randint(2, 200)
    member.network.hidden[ipos] = delta
```

ואז האלגוריתם רץ בפונקציה הזאת:

```
def run(self):
    graph_arr=[]
    self.calcFitness()
    self.sortByFitness()
    member= Citizen()
    member.network=self.population[0].network
    member.reg=self.population[0].reg
    member.fitness =self.population[0].fitness
    best = member
    for i in range(self.maxIter):
        self.calcFitness()
        self.sortByFitness()
        best=self.updateBest(best)
        self.printBest(best)
        self.mate()
        self.swap()
        graph_arr.append(best.fitness)
    print()
    print("Best Train accuracy:")
    print(best.fitness)
    self.print_results(best)
```

ואז פונקציית חישוב ה Regression: חישבנו אותו לכל אחד באוכלוסייה לפי הנוסחה הנתונה במעבדה ועשינו עדכון כנדרש

```
def regression(self):
    for i in range(self.popSize):
        cls = MLPClassifier(hidden_layer_sizes=self.population[i].network.hidden, max_iter=69000,
                            activation=self.population[i].network.activateFunction, solver='adam', random_stat
        sum=0
        for j in range(len(cls.coefs_)):
            for k in range(len(cls.coefs_[j])):
                sum+=cls.coefs_[j][k]* cls.coefs_[j][k]

        c= self.cacl_creg(self.population[i])
        self.population[i].reg=sum*(1/c)*(len(self.X_train))
```

ואז דוגמת ריצה :

```
BEST CITIZEN :
FITNESS =  0.7441860465116279

BEST CITIZEN :
FITNESS =  0.7441860465116279

BEST CITIZEN :
FITNESS =  0.813953488372093

BEST CITIZEN :
FITNESS =  0.813953488372093

BEST CITIZEN :
FITNESS =  0.813953488372093

BEST CITIZEN :
FITNESS =  0.813953488372093

BEST CITIZEN :
FITNESS =  0.813953488372093

BEST CITIZEN :
FITNESS =  0.813953488372093

Best Train accuracy:
```

ואז עשינו קטע וד הזה כדי להדפיס את classification report :

```python
def print_results(self,best):
    print("Classifier Report: ")
    cls = MLPClassifier(hidden_layer_sizes=best.network.hidden, max_iter=69000,
                        activation=best.network.activateFunction, solver='adam', random_state=1)
    cls.fit(self.X_train, self.Y_train)
    predict = cls.predict(self.X_test)
    print(classification_report(self.Y_test, predict, zero_division=0))
```

```
Best Train accuracy:
0.7906976744186046
The best Test accuracy:
0.7674418604651163
Classifier Report:
```

```
Classifier Report:
              precision    recall  f1-score   support

           0       0.65      0.79      0.71        14
           1       0.75      0.60      0.67        15
           2       0.33      0.33      0.33         3
           3       1.00      1.00      1.00         3
           4       1.00      1.00      1.00         2
           5       1.00      1.00      1.00         6

    accuracy                           0.74        43
   macro avg       0.79      0.79      0.78        43
weighted avg       0.75      0.74      0.74        43
```

מחלקות נוספות שהשתמשנו בהן :

```python
class Citizen:
    def __init__(self):
        citizen_array = []
        depth = random.randint(1, 10)
        for i in range(depth):
            tmp = random.randint(2, 200)
            citizen_array.append(tmp)

        if random.randint(0, 2) == 0:
            activate = RELU
        else:
            activate = TANH

        self.network = NeuralNetwork(depth, citizen_array, activate)
        self.fitness = -1
        self.reg = 0
```

תשובות לשאלות:

כן ליעל על ידי מקבליות זה רעיון לא רע כי אז האלגוריתם ירוץ בזמן יותר קצר כי חישובים שונים יתבצעו באותו זמן .וגם אם נשתמש במספרים שלמים אז חישובים יהיו יותר יעילים וזה יקטין זמן ריצת האלגוריתם. וגם אתחול חכם כי באתחול רנדומאלי יתכן שנקבל ערכים מאוד מאוד רחוקים מהמטרה ואז זה ישפיע על החישובים ואז אם יש איזושהי יוריסטיקה לאתחול חכם יכולים לאתחל עם תוצאות הגיוניים ואז מכן להגיע לתשובות יותר טובות בזמן קצר יותר כי לא התחלנו מנקודה רחוקה