**Logo

Description automatically generated**

**San Francisco Bay University**

**CS483 - Fundamentals of Artificial Intelligence**

**Homework Assignment #2**

**Due day: 6/26/2022**

**Instruction:**

1. **Push the source code to Github**
2. **Overdue homework submission could not be accepted.**
3. **Take academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)**

Table

Description automatically generated**Solution:**

Using the normal equation

= , , = : = = \* = , =

|  |
| --- |
| # Code for number 1  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from matplotlib import colors  #initila data  x\_init = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]  y\_init = [3, 3, 7, 7, 11, 11, 15, 15, 19, 19]  # Using Normal Equations  y = np.array([[3 ],[3] ,[7] ,[7] , [11], [11], [15], [15], [19], [19]])  x =np.array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],[1, 3, 5,  7, 9, 11, 13, 15, 17, 19]])  X = x.transpose()  X\_trans = X.transpose()  [t\_0, t\_1]= np.linalg.inv(X\_trans.dot(X)).dot(X\_trans).dot(y)  print(" Theta\_0 = ", t\_0, " \n Theta\_1 = ", t\_1)  # predicted values  y\_pred = t\_0 + t\_1\*x\_init  # compare results  plt.scatter(x\_init,y\_init, color = 'b')  plt.plot(x\_init, y\_pred, color = 'r')  plt.xlabel('X Label')  plt.ylabel('Y Label')  plt.show() |
| # result |

2. Create hypothesis function/loss function/cost function for binary classification and train your module by python program based on gradient descent algorithm from the following breast cancer dataset. After that, predict which class of last two records (highlighted in red color) is

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Clump Thickness  (Range:  1~10)  X1 | Uniformity of Cell Size  (Range:  1~10)  X2 | Uniformity of Cell Shape  (Range:  1~10)  X3 | Marginal Adhesion  (Range:  1~10)  X4 | Single Epithelial Cell Size  (Range:  1~10)  X5 | Bare Nuclei  (Range:  1~10)  X6 | Bland Chromatin  (Range:  1~10)  X7 | Normal Nucleoli  (Range:  1~10)  X8 | Mitoses  (Range:  1~10)  X9 | Class:  (2: benign,  4: malignant)  Y |
| 8 | **10** | **10** | **8** | **7** | **10** | **9** | **7** | **1** | **4** |
| 5 | **3** | **3** | **3** | **2** | **3** | **4** | **4** | **1** | **4** |
| 1 | **1** | **1** | **1** | **2** | **3** | **3** | **1** | **1** | **2** |
| 8 | **7** | **5** | **10** | **7** | **9** | **5** | **5** | **4** | **4** |
| 7 | **4** | **6** | **4** | **6** | **1** | **4** | **3** | **1** | **4** |
| 4 | **1** | **1** | **1** | **2** | **1** | **2** | **1** | **1** | **2** |
| 4 | **1** | **1** | **1** | **2** | **1** | **3** | **1** | **1** | **2** |
| 10 | **7** | **7** | **6** | **4** | **10** | **4** | **1** | **2** | **4** |
| 6 | **1** | **1** | **1** | **2** | **1** | **3** | **1** | **1** | **2** |
| 7 | **3** | **2** | **10** | **5** | **10** | **5** | **4** | **4** | **4** |
| 10 | **5** | **5** | **3** | **6** | **7** | **7** | **10** | **1** | **4** |
| 3 | **1** | **1** | **1** | **2** | **1** | **2** | **1** | **1** | **2** |
| 8 | **4** | **5** | **1** | **2** | **5** | **7** | **3** | **1** | **4** |
| 1 | **1** | **1** | **1** | **2** | **1** | **3** | **1** | **1** | **2** |
| 5 | **2** | **3** | **4** | **2** | **7** | **3** | **6** | **1** | **4** |
| 3 | **2** | **1** | **1** | **1** | **1** | **2** | **1** | **1** | **2** |
| 5 | **1** | **1** | **1** | **2** | **1** | **2** | **1** | **1** | **2** |
| 2 | **1** | **1** | **1** | **2** | **1** | **2** | **1** | **1** | **2** |
| 1 | **1** | **3** | **1** | **2** | **1** | **1** | **1** | **1** | **2** |
| 3 | **1** | **1** | **1** | **1** | **1** | **2** | **1** | **1** | **2** |
| 2 | **1** | **1** | **1** | **2** | **1** | **3** | **1** | **1** | **2** |
| 10 | **7** | **7** | **3** | **8** | **5** | **7** | **4** | **3** | **4** |
| 2 | **1** | **1** | **2** | **2** | **1** | **3** | **1** | **1** | **2** |
| 3 | **1** | **2** | **1** | **2** | **1** | **2** | **1** | **1** | **2** |
| 2 | **1** | **1** | **1** | **2** | **1** | **2** | **1** | **1** | **2** |
| 10 | **10** | **10** | **8** | **6** | **1** | **8** | **9** | **1** | **4** |
| 6 | **2** | **1** | **1** | **1** | **1** | **7** | **1** | **1** | **2** |
| 5 | **4** | **4** | **9** | **2** | **10** | **5** | **6** | **1** | **4** |
| 2 | **5** | **3** | **3** | **6** | **7** | **7** | **5** | **1** | **4** |
| 10 | **4** | **3** | **1** | **3** | **3** | **6** | **5** | **2** | **4** |
| 6 | **10** | **10** | **2** | **8** | **10** | **7** | **3** | **3** | **4** |
| 5 | **6** | **5** | **6** | **10** | **1** | **3** | **1** | **1** | **4** |
| 10 | **10** | **10** | **4** | **8** | **1** | **8** | **10** | **1** | **?** |
| 6 | **6** | **6** | **9** | **6** | **2** | **7** | **8** | **1** | **?** |

Solution:

Let = [

**Hypothesis function:**

**Loss function**:

=

**Cost function:**

=

**Partial derivative function:**

= \* , where element of

**Then we can use the gradient decent algorithm:**

, is element of

|  |
| --- |
| #Q2  # import Libraries  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  import seaborn as sns  #load data  df = pd.read\_csv("q2.csv")  df.head()  df.shape  # count the number of empty raws for each column  df.isna().sum()  # Replace missing values with average Value  zero\_not\_accepted = ['X1', 'X2', 'X3','X4','X5', 'X6', 'X7', 'X8','X9']  for column in zero\_not\_accepted:      df[column] = df[column].replace(0,np.NaN)      mean = int(df[column].mean(skipna = True))      df[column] = df[column].replace(np.NaN,mean)  # divid the data into train and test sets  from sklearn.model\_selection import train\_test\_split  X = df.iloc[:32,0:8]  y = df.iloc[:32,8]  X\_train, X\_test,y\_train, y\_test = train\_test\_split(X,y,test\_size=0.20, random\_state = 0)  # feature scaling  from sklearn.preprocessing import StandardScaler  sc\_X = StandardScaler()  X\_train = sc\_X.fit\_transform(X\_train)  X\_test = sc\_X.transform(X\_test)  X\_train  # create a function for the models  # logistic Regression  from sklearn.linear\_model import LogisticRegression  log = LogisticRegression(random\_state=0)  log.fit(X\_train,y\_train)  # Models Accuracy  print("Logistic Regression Training accuracy: ", log.score(X\_train,y\_train))  # Test model accuracy on test data using confusion matrix  ( for logistic Regression)  from sklearn.metrics import confusion\_matrix  cm = confusion\_matrix(y\_test, log.predict(X\_test))  print(cm)  # accuracy report for Logistic Regression  from sklearn.metrics import classification\_report  from sklearn.metrics import accuracy\_score  # print( classification\_report(y\_test, log.predict(X\_test)))  # print(accuracy\_score(y\_test, log.predict(X\_test)))  # predict the unknown y values  X\_toBeStudied = StandardScaler().fit\_transform(df.iloc[32:34,0:8])  y\_pred = log.predict(X\_toBeStudied)  print("The Predicted Values of Y are :", (y\_pred\*2+2))  # decoding y\_pred by using (y\_pred\*2+2)to get either 4 or 2 |

Result:

|  |
| --- |
|  |

Hence, the table can be filled as follows:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Clump Thickness  (Range:  1~10)  X1 | Uniformity of Cell Size  (Range:  1~10)  X2 | Uniformity of Cell Shape  (Range:  1~10)  X3 | Marginal Adhesion  (Range:  1~10)  X4 | Single Epithelial Cell Size  (Range:  1~10)  X5 | Bare Nuclei  (Range:  1~10)  X6 | Bland Chromatin  (Range:  1~10)  X7 | Normal Nucleoli  (Range:  1~10)  X8 | Mitoses  (Range:  1~10)  X9 | Class:  (2: benign,  4: malignant)  Y |
| 8 | **10** | **10** | **8** | **7** | **10** | **9** | **7** | **1** | **4** |
| 5 | **3** | **3** | **3** | **2** | **3** | **4** | **4** | **1** | **4** |
| 1 | **1** | **1** | **1** | **2** | **3** | **3** | **1** | **1** | **2** |
| 8 | **7** | **5** | **10** | **7** | **9** | **5** | **5** | **4** | **4** |
| 7 | **4** | **6** | **4** | **6** | **1** | **4** | **3** | **1** | **4** |
| 4 | **1** | **1** | **1** | **2** | **1** | **2** | **1** | **1** | **2** |
| 4 | **1** | **1** | **1** | **2** | **1** | **3** | **1** | **1** | **2** |
| 10 | **7** | **7** | **6** | **4** | **10** | **4** | **1** | **2** | **4** |
| 6 | **1** | **1** | **1** | **2** | **1** | **3** | **1** | **1** | **2** |
| 7 | **3** | **2** | **10** | **5** | **10** | **5** | **4** | **4** | **4** |
| 10 | **5** | **5** | **3** | **6** | **7** | **7** | **10** | **1** | **4** |
| 3 | **1** | **1** | **1** | **2** | **1** | **2** | **1** | **1** | **2** |
| 8 | **4** | **5** | **1** | **2** | **5** | **7** | **3** | **1** | **4** |
| 1 | **1** | **1** | **1** | **2** | **1** | **3** | **1** | **1** | **2** |
| 5 | **2** | **3** | **4** | **2** | **7** | **3** | **6** | **1** | **4** |
| 3 | **2** | **1** | **1** | **1** | **1** | **2** | **1** | **1** | **2** |
| 5 | **1** | **1** | **1** | **2** | **1** | **2** | **1** | **1** | **2** |
| 2 | **1** | **1** | **1** | **2** | **1** | **2** | **1** | **1** | **2** |
| 1 | **1** | **3** | **1** | **2** | **1** | **1** | **1** | **1** | **2** |
| 3 | **1** | **1** | **1** | **1** | **1** | **2** | **1** | **1** | **2** |
| 2 | **1** | **1** | **1** | **2** | **1** | **3** | **1** | **1** | **2** |
| 10 | **7** | **7** | **3** | **8** | **5** | **7** | **4** | **3** | **4** |
| 2 | **1** | **1** | **2** | **2** | **1** | **3** | **1** | **1** | **2** |
| 3 | **1** | **2** | **1** | **2** | **1** | **2** | **1** | **1** | **2** |
| 2 | **1** | **1** | **1** | **2** | **1** | **2** | **1** | **1** | **2** |
| 10 | **10** | **10** | **8** | **6** | **1** | **8** | **9** | **1** | **4** |
| 6 | **2** | **1** | **1** | **1** | **1** | **7** | **1** | **1** | **2** |
| 5 | **4** | **4** | **9** | **2** | **10** | **5** | **6** | **1** | **4** |
| 2 | **5** | **3** | **3** | **6** | **7** | **7** | **5** | **1** | **4** |
| 10 | **4** | **3** | **1** | **3** | **3** | **6** | **5** | **2** | **4** |
| 6 | **10** | **10** | **2** | **8** | **10** | **7** | **3** | **3** | **4** |
| 5 | **6** | **5** | **6** | **10** | **1** | **3** | **1** | **1** | **4** |
| 10 | **10** | **10** | **4** | **8** | **1** | **8** | **10** | **1** | **4** |
| 6 | **6** | **6** | **9** | **6** | **2** | **7** | **8** | **1** | **4** |

3. Given the following dataset, design python function as **binary** classifier for the following two classes.

1. Plot all points in two different classes first in Excel or python matplotlib functions
2. Observe what boundary decision **function** is good to separate two classes
3. Build up hypothesis function/loss function/cost function based on your selected decision function
4. Write python program to train your model
5. After model training, plot decision boundary function in Excel or python matplotlib functions

|  |  |  |
| --- | --- | --- |
| X1 | X2 | Y |
| -3.98 | **-0.12** | **1** |
| -3.464 | **-2.11** | **1** |
| -3.461 | **1.89** | **1** |
| -2.22 | **-3.474** | **1** |
| -2.02 | **0.03** | **0** |
| -2.01 | **3.459** | **1** |
| -1.42 | **-1.409** | **0** |
| -1.416 | **1.419** | **0** |
| -1.09 | **0.08** | **0** |
| -0.19 | **-4.13** | **1** |
| 0.01 | **1.02** | **0** |
| 0.03 | **-2.12** | **0** |
| 0.04 | **2.06** | **0** |
| 0.06 | **3.97** | **1** |
| 0.07 | **0.1** | **0** |
| 0.12 | **-1.12** | **0** |
| 1.11 | **0.09** | **0** |
| 1.411 | **1.419** | **0** |
| 1.414 | **-1.415** | **0** |
| 1.86 | **3.47** | **1** |
| 1.96 | **-0.12** | **0** |
| 2.11 | **-3.472** | **1** |
| 3.461 | **-1.87** | **1** |
| 3.464 | **2.07** | **1** |
| 4.12 | **0.09** | **1** |

Solution:

1. Plot all points in two different classes first in Excel or python matplotlib functions

Code:

|  |
| --- |
| #Q3  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from matplotlib import colors  import seaborn   as sns  #load data  df = pd.read\_csv("Q3.csv")  df.head()  # plot data  plt.scatter(df.X1, df.X2, c= ['r' if y == 1 else 'g' for y in df.Y])  plt.xlabel('x1')  plt.ylabel('x2')  plt.legend()  plt.show() |

Result:

|  |
| --- |
|  |

1. Observe what boundary decision **function** is good to separate two classes

A **circle** **equation** whose **center = (a,b)** and **radius = r ,** can be used as a boundary decision function to separate the two classes

Hence, **Decision\_Fun** =

*Boundary Function = ( =( -2\*a\* + ) + (( -2\*b\* + )*

*= + (-2\*a\* + +*

*Let , ,*

*,*

*Hence, the boundary function can be written in terms of theta values as follows*

*Boundary Function = + + +*

1. Build up hypothesis function/loss function/cost function based on your selected decision function

**Hypothesis function:**

**Loss function**:

=

Where

**Cost function:**

=

**Partial derivative function:**

=

= \*

= \*

= \*

but we know , similarly

= \*

but we know

**Then we can use the gradient decent algorithm:**

# Again, we know

# and the same way

**Note**: In this problem, however, we know the values of the center (a, b) and the radius (r) from the graph, which are (0,0) and r = 3 can separate the data into the required binary y labels

1. Write python program to train your model

|  |
| --- |
| # import Libraries  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  #load data  df = pd.read\_csv("Q3.csv")  df.head()  # divid the data into train and test sets  from sklearn.model\_selection import train\_test\_split  X = df.iloc[:,0:2]  y = df.iloc[:,2]  X\_train, X\_test,y\_train, y\_test = train\_test\_split(X,y,test\_size=0.20, random\_state = 0)  # feature scaling  from sklearn.preprocessing import StandardScaler  sc\_X = StandardScaler()  X\_train = sc\_X.fit\_transform(X\_train)  X\_test = sc\_X.transform(X\_test)  X\_train  # train model logistic Regression  from sklearn.linear\_model import LogisticRegression  log\_reg= LogisticRegression()  log\_reg.fit(X\_train,y\_train)  #predict y values for the given x values  print(log\_reg.predict(X\_train))  print(y\_train)  # Models Accuracy  print("Logistic Regression Training accuracy: ", log\_reg.score(X\_train,y\_train))  cf = log\_reg.coef\_  intercept = log\_reg.intercept\_  print("intercept: ",intercept ,"\n Coefficient: ",intercept) |

1. After model training, plot decision boundary function in Excel or python matplotlib functions

In this problem, we know the values of the center (a, b) and the radius (r), which are (0,0) and r = 3 from the scatter plot in part a, hence we obtain the boundary decision function as follows

**Hence:**  *Boundary Function = ( =( -2\*0\* + ) + (( -2\*0\* + ) = 3^2*

*= +*

*Which means, the values of = 1,*

*And for ; which gives the following result*

|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  #load data  df = pd.read\_csv("Q3.csv")  df.head()  # plot data  plt.scatter(df.X1, df.X2, c= ['r' if y == 1 else 'g' for y in df.Y])  circle = plt.Circle((0, 0), 3,fill=False)  ax = plt.gca()  ax.add\_patch(circle)  plt.xlabel('x1')  plt.ylabel('x2')  plt.legend()  plt.show() |

*A picture containing chart

Description automatically generated*

4. Train your hypothesis **functions** for multiclass classification from the following given dataset by python program. And then predict what Y’s value is in the last 4 samples (highlighted in red color). Verify your classifier model by plotting all points and decision lines

|  |  |  |
| --- | --- | --- |
| X1 | X2 | Y |
| 3.25 | **7.956** | **2** |
| 3.3 | **2.2** | **0** |
| 3.32 | **3.41** | **0** |
| 3.35 | **10.272** | **2** |
| 4.01 | **1.65** | **0** |
| 4.03 | **2.51** | **0** |
| 4.05 | **4.21** | **0** |
| 4.05 | **7.38** | **2** |
| 4.06 | **11.412** | **2** |
| 4.07 | **9.198** | **2** |
| 5.22 | **2.15** | **0** |
| 5.24 | **3.41** | **0** |
| 5.25 | **7.866** | **2** |
| 5.28 | **10.008** | **2** |
| 8.15 | **6.3** | **1** |
| 8.23 | **7.95** | **1** |
| 9.38 | **7.34** | **1** |
| 9.4 | **8.21** | **1** |
| 10.2 | **6.52** | **1** |
| 10.8 | **7.72** | **1** |
| 4.01 | **3.02** | **?** |
| 9.1 | **6.5** | **?** |
| 3.50 | **9.50** | **?** |
| 6.01 | **6.01** | **?** |

Solution:

1. Scatter plot for data visualization:

|  |
| --- |
| #Q4  # import Libraries  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  #load data  df = pd.read\_csv("Q4.csv")  df.head()  df.shape  # count the number of empty raws for each column  df.isna().sum()  # Replace missing values with average Value  zero\_not\_accepted = ['X1', 'X2']  for column in zero\_not\_accepted:      df[column] = df[column].replace(0,np.NaN)      mean = int(df[column].mean(skipna = True))      df[column] = df[column].replace(np.NaN,mean)  # divid the data into train and test sets  from sklearn.model\_selection import train\_test\_split  X = df.iloc[:19,0:2]  y = df.iloc[:19,2]  X\_train, X\_test,y\_train, y\_test = train\_test\_split(X,y,test\_size=0.20, random\_state = 0)  # plot data  colors = {"2":"r", "1": "g", '0':'b'}  plt.scatter(df.X1[0:19], df.X2[0:19],c= y.map(colors))  plt.xlabel('x1')  plt.ylabel('x2')  plt.legend()  plt.show() |

Data scatter plot:

|  |
| --- |
| Chart, scatter chart  Description automatically generated  Where "2":"r", "1": "g", '0':'b' |

1. Train model and predict unknown y-values

|  |
| --- |
| # import Libraries  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  #load data  df = pd.read\_csv("Q4.csv")  df.head()  df.shape  # count the number of empty raws for each column  df.isna().sum()  # Replace missing values with average Value  zero\_not\_accepted = ['X1', 'X2']  for column in zero\_not\_accepted:      df[column] = df[column].replace(0,np.NaN)      mean = int(df[column].mean(skipna = True))      df[column] = df[column].replace(np.NaN,mean)  # divid the data into train and test sets  from sklearn.model\_selection import train\_test\_split  X = df.iloc[:20,0:2]  y = df.iloc[:20,2]  X\_train, X\_test,y\_train, y\_test = train\_test\_split(X,y,test\_size=0.20, random\_state = 0)  # feature scaling  from sklearn.preprocessing import StandardScaler  sc\_X = StandardScaler()  X\_train = sc\_X.fit\_transform(X\_train)  X\_test = sc\_X.transform(X\_test)  X\_train  # logistic Regression  from sklearn.linear\_model import LogisticRegression  log = LogisticRegression(random\_state=0)  log.fit(X\_train,y\_train)  # Models Accuracy  print("Logistic Regression Training accuracy: ", log.score(X\_train,y\_train))  # Test model accuracy on test data using confusion matrix  ( for logistic Regression)  from sklearn.metrics import confusion\_matrix  cm = confusion\_matrix(y\_test, log.predict(X\_test))  print(cm)  # accuracy report for Logistic Regression  from sklearn.metrics import classification\_report  from sklearn.metrics import accuracy\_score  # print( classification\_report(y\_test, log.predict(X\_test)))  # print(accuracy\_score(y\_test, log.predict(X\_test)))  # predict the unknown y values  X\_toBeStudied = StandardScaler().fit\_transform(df.iloc[20:24,0:2])  y\_pred = log.predict(X\_toBeStudied)  print("The Predicted Values of Y are :", y\_pred)  # decoding y\_pred by using (y\_pred\*2+2)  # plot data  df\_pred = pd.read\_csv("Q4\_pred.csv")  plt.scatter(df\_pred.X1, df\_pred.X2, c= df\_pred.Y)  plt.xlabel('x1')  plt.ylabel('x2')  plt.show() |

Result

|  |
| --- |
| Chart, scatter chart  Description automatically generated |

Hence the data with the predicted y-values are as follows :

|  |  |  |
| --- | --- | --- |
| X1 | X2 | Y |
| 3.25 | **7.956** | **2** |
| 3.3 | **2.2** | **0** |
| 3.32 | **3.41** | **0** |
| 3.35 | **10.272** | **2** |
| 4.01 | **1.65** | **0** |
| 4.03 | **2.51** | **0** |
| 4.05 | **4.21** | **0** |
| 4.05 | **7.38** | **2** |
| 4.06 | **11.412** | **2** |
| 4.07 | **9.198** | **2** |
| 5.22 | **2.15** | **0** |
| 5.24 | **3.41** | **0** |
| 5.25 | **7.866** | **2** |
| 5.28 | **10.008** | **2** |
| 8.15 | **6.3** | **1** |
| 8.23 | **7.95** | **1** |
| 9.38 | **7.34** | **1** |
| 9.4 | **8.21** | **1** |
| 10.2 | **6.52** | **1** |
| 10.8 | **7.72** | **1** |
| 4.01 | **3.02** | **0** |
| 9.1 | **6.5** | **1** |
| 3.50 | **9.50** | **2** |
| 6.01 | **6.01** | **0** |