SINGLE AND MULTI LAYER PERCEPTRON (ASSESSMENT - 3)

CSE4020(MACHINE LEARNING)LAB:L49-L50

MARCH 9, 2022

ANIRUDH VADERA
20BCE2940

QUESTION:

1. Perceptron:

Construct a perceptron to train AND and OR logic gates.

Expected Output

----

1. **Initial Weight**
2. **Actual and Predicted Value of training samples**
3. **Final Weight**
4. **Error**

2. Multi-Layer Perceptron Network

Train a MLP using back propagation network to classify cars based on its specification. Perform the result analysis on test samples.

Expected Output

----

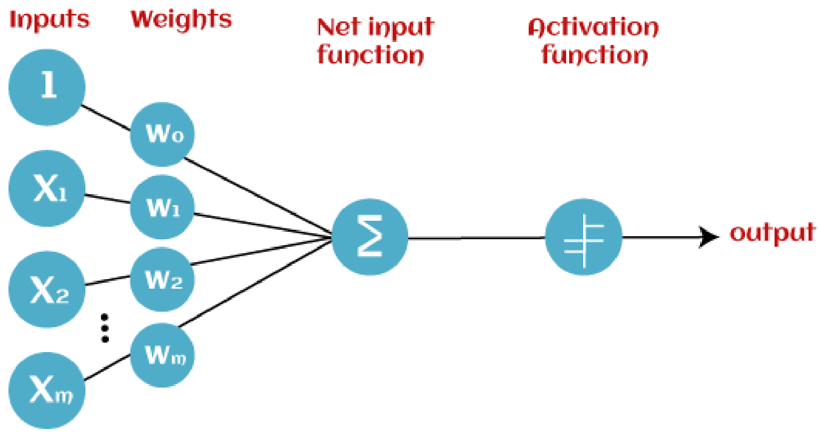
1) **Initial Weight**
2) **Final Weights**
3) **Actual and Predicted Value of training samples**
4) **Precision, recall, error and Accuracy**

Description:

Perceptron:

Perceptron is Machine Learning algorithm for supervised learning of various binary classification tasks. Further, ***Perceptron is also understood as an Artificial Neuron or neural network unit that helps to detect certain input data computations in business intelligence***.

Perceptron model is also treated as one of the best and simplest types of Artificial Neural networks. However, it is a supervised learning algorithm of binary classifiers. Hence, we can consider it as a single-layer neural network with four main parameters, i.e., **input values, weights and Bias, net sum, and an activation function.**



- **Input Nodes or Input Layer:**

This is the primary component of Perceptron which accepts the initial data into the system for further processing. Each input node contains a real numerical value.

- **Wight and Bias:**

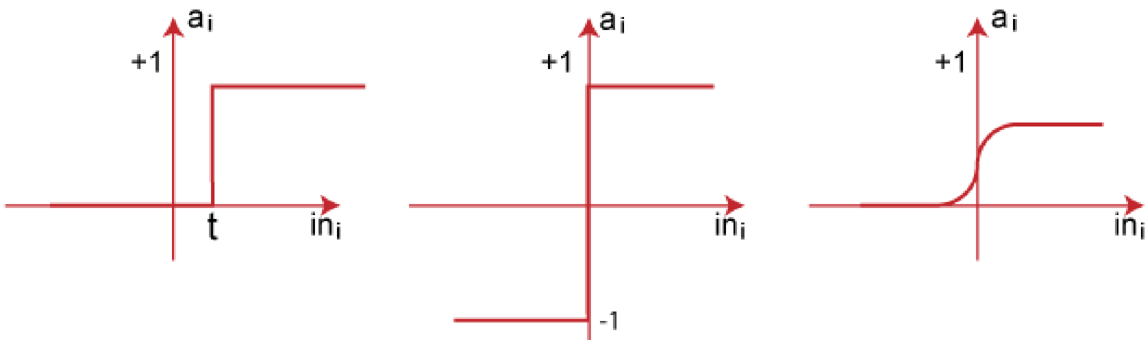
Weight parameter represents the strength of the connection between units. This is another most important parameter of Perceptron components. Weight is directly proportional to the strength of the associated input neuron in deciding the output. Further, Bias can be considered as the line of intercept in a linear equation.

- **Activation Function:**

These are the final and important components that help to determine whether the neuron will fire or not. Activation Function can be considered primarily as a step function.

Types of Activation functions:

- Sign function
- Step function, and
- Sigmoid function



Step Function Sign Function Sigmoid Function

Types of Perceptron Models

Based on the layers, Perceptron models are divided into two types. These are as follows:

1. Single-layer Perceptron Model
2. Multi-layer Perceptron model

Single Layer Perceptron Model:

This is one of the easiest Artificial neural networks (ANN) types. A single-layered perceptron model consists feed-forward network and also includes a threshold transfer function inside the model. The main objective of the single-layer perceptron model is to analyse the linearly separable objects with binary outcomes.

In a single layer perceptron model, its algorithms do not contain recorded data, so it begins with inconstantly allocated input for weight parameters. Further, it sums up all inputs (weight). After adding all inputs, if the total sum of all inputs is more than a pre-determined value, the model gets activated and shows the output value as +1.

If the outcome is same as pre-determined or threshold value, then the performance of this model is stated as satisfied, and weight demand does not change. However, this model consists of a few discrepancies triggered when multiple weight inputs values are fed into the model. Hence, to find desired output and minimize errors, some changes should be necessary for the weights input.

Multi-Layered Perceptron Model:

Like a single-layer perceptron model, a multi-layer perceptron model also has the same model structure but has a greater number of hidden layers.

The multi-layer perceptron model is also known as the Backpropagation algorithm, which executes in two stages as follows:

- **Forward Stage:** Activation functions start from the input layer in the forward stage and terminate on the output layer.
- **Backward Stage:** In the backward stage, weight and bias values are modified as per the model's requirement. In this stage, the error between actual output and demanded originated backward on the output layer and ended on the input layer.

Hence, a multi-layered perceptron model has considered as multiple artificial neural networks having various layers in which activation function does not remain linear, similar to a single layer perceptron model. Instead of linear, activation function can be executed as sigmoid, TanH, ReLU, etc., for deployment.

A multi-layer perceptron model has greater processing power and can process linear and non-linear patterns. Further, it can also implement logic gates such as AND, OR, XOR, NAND, NOT, XNOR, NOR.

Formula used:

Weighted sum: ∑wi*xi = x1*w1 + x2*w2 +…wn*xn

Add a special term called **bias 'b'** to this weighted sum to improve the model's performance.

∑wi*xi + b

An activation function is applied with the above-mentioned weighted sum, which gives us output either in binary form or a continuous value as follows:

Y = f(∑wi*xi + b)

Correcting errors:

wi = wi + learning_rate*(outi-output(Error)) * Xij

Bias = bias + learning_rate*(output)

Question 1:

AND GATE:

Code:

```
import numpy as np
from sklearn.metrics import accuracy_score, precision_score,
recall_score,classification_report


x = [ [-1,-1], [-1,1], [1,-1], [1,1]]
x_array = np.asarray(x)
# expected outputs (AND port is the product of each entry
out = np.asarray([-1,-1,-1,1])


m = 2
n = 4


w=[]
for i in range(m):
   w.append(0.5)
theta = 0
bias = -1
learning_rate = 1


print("Initial Weights : " , w)


sum = 0
epochs = 10
e = 0
error = [0,0,0,0]
E = 100
```

```
pred = []

while E!=0 and e < epochs:
   pred = []
   E = 0
   for i in range(n):
      sum = 0
      for j in range(m):
         sum += x[i][j]*w[j]
      sum = sum + bias
      if(sum > theta):
         output = 1
      else:
         output = -1
      pred.append(output)
      error[i] = out[i] - output
      for j in range(m):
         if(out[i] != output):
            w[j] = w[j] + learning_rate*(out[i]-output)*x[i][j]
            bias = bias + learning_rate*out[i]
   for i in error:
      E = E + abs(i)
   e = e + 1
   print("After " , e , "Epochs : ")
   print("Prediction : " , pred)
   print("Actual Values : " , out)
   print("Error : " , error)
```

```
print("The Final Weights are : ", w)

print("The Final Bias is : " , bias)

print("The Final Error is " , "Error Array : ",error,"Error : ",E)]

# Checking the accuracy of our model

print('Accuracy: ',accuracy_score(out,pred))

print('Precision: %.3f' % precision_score(out, pred,average='micro'))

print('Recall: %.3f' % recall_score(out, pred,average='micro'))


# Our Model Report

print('*************** Evaluation on Our Model ***************')

print('Accuracy Score: ', accuracy_score(out,pred))

# Look at classification report to evaluate the model

print(classification_report(out, pred))

print('--------------------------------------------------------')

print("")
```

Code Snippets:

```python

sum = 0
epochs = 10
e = 0
error = [0,0,0,0]
E = 100
pred = []

while E!=0 and e < epochs:
    pred = []
    E = 0
    for i in range(n):
        sum = 0
        for j in range(m):
            sum += x[i][j]*w[j]
        sum = sum + bias
        if(sum > theta):
            output = 1
        else:
            output = -1
        pred.append(output)
        error[i] = out[i] - output
        for j in range(m):
            if(out[i] != output):
                w[j] = w[j] + learning_rate*(out[i]-output)*x[i][j]
                bias = bias + learning_rate*out[i]
    for i in error:
        E = E + abs(i)
    e = e + 1
    print("After " , e , "Epochs : ")
    print("Prediction : " , pred)
    print("Actual Values : " , out)
    print("Error : " , error)

```

Output and Results:

Required Result:

```
   X  Y  OUT(AND)
0 -1 -1        -1
1 -1  1        -1
2  1 -1        -1
3  1  1         1
```

After several Epochs:

```
In [1]: runfile('C:/Users/Anirudh/OneDrive/Desktop/python/perce
Initial Weights :  [0.5, 0.5]
After  1 Epochs :
Prediction :  [-1, -1, -1, -1]
Actual Values :  [-1 -1 -1  1]
Error :  [0, 0, 0, 2]
After  2 Epochs :
Prediction :  [-1, 1, 1, 1]
Actual Values :  [-1 -1 -1  1]
Error :  [0, -2, -2, 0]
After  3 Epochs :
Prediction :  [-1, -1, -1, 1]
Actual Values :  [-1 -1 -1  1]
Error :  [0, 0, 0, 0]
The Final Weights are :  [2.5, 2.5]
The Final Bias is :  -3
The Final Error is  Error Array :  [0, 0, 0, 0] Error :  0
Accuracy:  1.0
Precision: 1.000
Recall: 1.000
*************** Evaluation on Our Model ***************
Accuracy Score:  1.0
              precision    recall  f1-score   support

          -1       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         1

    accuracy                           1.00         4
   macro avg       1.00      1.00      1.00         4
weighted avg       1.00      1.00      1.00         4


--------------------------------------------------------
```

Initial Weights:

```
In [1]: runfile('C:/Users/Anirudh/
Initial Weights :  [0.5, 0.5]
```

Actual and Predicted Values of the Training Samples:

After every epochs Predicted and actual values are shown finally they both converge to become same.

```
After  1 Epochs :
Prediction :  [-1, -1, -1, -1]
Actual Values :  [-1 -1 -1  1]
Error :  [0, 0, 0, 2]
After  2 Epochs :
Prediction :  [-1, 1, 1, 1]
Actual Values :  [-1 -1 -1  1]
Error :  [0, -2, -2, 0]
After  3 Epochs :
Prediction :  [-1, -1, -1, 1]
Actual Values :  [-1 -1 -1  1]
Error :  [0, 0, 0, 0]
```

Final Weights:

```
The Final Weights are :  [2.5, 2.5]
The Final Bias is :  -3
The Final Error is  Error Array :  [0, 0, 0, 0] Error :  0
```

Accuracy Analysis(Errors):

Error after every epoch:

```
After  1 Epochs :
Prediction :  [-1, -1, -1, -1]
Actual Values :  [-1 -1 -1  1]
Error :  [0, 0, 0, 2]
After  2 Epochs :
Prediction :  [-1, 1, 1, 1]
Actual Values :  [-1 -1 -1  1]
Error :  [0, -2, -2, 0]
After  3 Epochs :
Prediction :  [-1, -1, -1, 1]
Actual Values :  [-1 -1 -1  1]
Error :  [0, 0, 0, 0]
```

Final error:

```
The Final Error is  Error Array :  [0, 0, 0, 0] Error :  0
```

Classification Report(Accuracy,Recall and Precision):

```
Accuracy:  1.0
Precision: 1.000
Recall: 1.000
*************** Evaluation on Our Model ***************
Accuracy Score:  1.0
              precision    recall  f1-score   support

          -1       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         1

    accuracy                           1.00         4
   macro avg       1.00      1.00      1.00         4
weighted avg       1.00      1.00      1.00         4

-------------------------------------------------------
```

OR GATE:

Code:

```
import numpy as np

from sklearn.metrics import accuracy_score, precision_score,
recall_score,classification_report

x = [ [-1,-1], [-1,1], [1,-1], [1,1]]

x_array = np.asarray(x)

# expected outputs (AND port is the product of each entry

out = np.asarray([-1,1,1,1])

m = 2

n = 4

w=[]

for i in range(m):

    w.append(0.5)

theta = 0

bias = -1

learning_rate = 1

print("Initial Weights : " , w)

sum = 0

epochs = 10

e = 0

error = [0,0,0,0]

E = 100

pred = []


while E!=0 and e < epochs:

    pred = []

    E = 0

    for i in range(n):
```

```
        sum = 0
        for j in range(m):
            sum += x[i][j]*w[j]
        sum = sum + bias
        if(sum > theta):
            output = 1
        else:
            output = -1
        pred.append(output)
        error[i] = out[i] - output
        for j in range(m):
            if(out[i] != output):
                w[j] = w[j] + learning_rate*(out[i]-output)*x[i][j]
                bias = bias + learning_rate*out[i]
    for i in error:
        E = E + abs(i)
    e = e + 1
    print("After " , e , "Epochs : ")
    print("Prediction : " , pred)
    print("Actual Values : " , out)
    print("Error : " , error)



print("The Final Weights are : ", w)
print("The Final Bias is : " , bias)
print("The Final Error is : " , "Error Array : ",error,"Error : ",E)
# Checking the accuracy of our model
print('Accuracy: ',accuracy_score(out,pred))
print('Precision: %.3f' % precision_score(out, pred,average='micro'))
```

print('Recall: %.3f' % recall_score(out, pred,average='micro'))

Our Model Report

print('************** Evaluation on Our Model **************')

print('Accuracy Score: ', accuracy_score(out,pred))

Look at classification report to evaluate the model

print(classification_report(out, pred))

print('---')

print("")

Code Snippets:

```
sum = 0
epochs = 10
e = 0
error = [0,0,0,0]
E = 100
pred = []

while E!=0 and e < epochs:
    pred = []
    E = 0
    for i in range(n):
        sum = 0
        for j in range(m):
            sum += x[i][j]*w[j]
        sum = sum + bias
        if(sum > theta):
            output = 1
        else:
            output = -1
        pred.append(output)
        error[i] = out[i] - output
        for j in range(m):
            if(out[i] != output):
                w[j] = w[j] + learning_rate*(out[i]-output)*x[i][j]
                bias = bias + learning_rate*out[i]
    for i in error:
        E = E + abs(i)
    e = e + 1
    print("After " , e , "Epochs : ")
    print("Prediction : " , pred)
    print("Actual Values : " , out)
    print("Error : " , error)
```

Output and Results:

Required Result:

	X	Y	OUT(OR)
0	-1	-1	-1
1	-1	1	1
2	1	-1	1
3	1	1	1

After several Epochs:

```
In [2]: runfile('C:/Users/Anirudh/OneDrive/Desktop/python/percept
Initial Weights :  [0.5, 0.5]
After  1 Epochs :
Prediction :  [-1, -1, -1, 1]
Actual Values :  [-1  1  1  1]
Error :  [0, 2, 2, 0]
After  2 Epochs :
Prediction :  [1, 1, 1, 1]
Actual Values :  [-1  1  1  1]
Error :  [-2, 0, 0, 0]
After  3 Epochs :
Prediction :  [-1, 1, 1, 1]
Actual Values :  [-1  1  1  1]
Error :  [0, 0, 0, 0]
The Final Weights are :  [2.5, 2.5]
The Final Bias is :  1
The Final Error is :  Error Array :  [0, 0, 0, 0] Error :  0
Accuracy:  1.0
Precision: 1.000
Recall: 1.000
*************** Evaluation on Our Model ***************
Accuracy Score:  1.0
              precision    recall  f1-score   support

          -1       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         3

    accuracy                           1.00         4
   macro avg       1.00      1.00      1.00         4
weighted avg       1.00      1.00      1.00         4

--------------------------------------------------------
```

Initial Weights:

```
In [1]: runfile('C:/Users/Anirudh/
Initial Weights :  [0.5, 0.5]
```

Actual and Predicted Values of the Training Samples:

After every epochs Predicted and actual values are shown finally they both converge to become same.

```
After  1 Epochs :
Prediction :  [-1, -1, -1, 1]
Actual Values :  [-1  1  1  1]
Error :  [0, 2, 2, 0]
After  2 Epochs :
Prediction :  [1, 1, 1, 1]
Actual Values :  [-1  1  1  1]
Error :  [-2, 0, 0, 0]
After  3 Epochs :
Prediction :  [-1, 1, 1, 1]
Actual Values :  [-1  1  1  1]
Error :  [0, 0, 0, 0]
```

Final Weights:

```
The Final Weights are :  [2.5, 2.5]
The Final Bias is :  1
The Final Error is :  Error Array :  [0, 0, 0, 0] Error :  0
```

Accuracy Analysis(Errors):

Error after every epoch:

```
After  1 Epochs :
Prediction :  [-1, -1, -1, 1]
Actual Values :  [-1  1  1  1]
Error :  [0, 2, 2, 0]
After  2 Epochs :
Prediction :  [1, 1, 1, 1]
Actual Values :  [-1  1  1  1]
Error :  [-2, 0, 0, 0]
After  3 Epochs :
Prediction :  [-1, 1, 1, 1]
Actual Values :  [-1  1  1  1]
Error :  [0, 0, 0, 0]
```

Final error:

```
The Final Error is  Error Array :  [0, 0, 0, 0] Error :  0
```

Classification Report(Accuracy,Recall and Precision):

```
Accuracy:  1.0
Precision: 1.000
Recall: 1.000
*************** Evaluation on Our Model ***************
Accuracy Score:  1.0
              precision    recall  f1-score   support

          -1       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         3

    accuracy                           1.00         4
   macro avg       1.00      1.00      1.00         4
weighted avg       1.00      1.00      1.00         4

--------------------------------------------------------
```

Inference:

- **It took 3 epochs for our model to reach the correct results we wanted which means it converged nor too quickly or slowly so the value of learning rate was perfect**
- **After getting the final values of the weights and bias the manual calculations for both AND and OR GATES comes out to be correct with theta as 0.**
- **The accuracy of our model is 100 percent whereas precision is 100 percent and recall is also 100 percent.**

Question 2:

Code:

```
import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

from sklearn.metrics import plot_confusion_matrix, confusion_matrix, accuracy_score, precision_score, recall_score,classification_report

import seaborn as sns

from sklearn.neural_network import MLPClassifier


# Importing the dfset

df=pd.read_csv("C:/Users/Anirudh/OneDrive/Desktop/car_evaluation.csv")

print(df)

print("\n")


# Check for missing values

print("Checking for missing values :")

print(df.isnull().sum())

print("\n")


# Printing the header of the dfset

print("dfset Header : ")

print(df.head())

print("\n")


# Information regarding the columns
```

```
print("Information regarding the columns : ")
print(df.info())
print("\n")

# Information related to the dfset
print("dfset Details : ")
print(df.describe())
print("\n")

# Convert categories into integers for each column.
df.Buying=df.Buying.replace({'low':0, 'med':1, 'high':2, 'vhigh':3})
df.Maint=df.Maint.replace({'low':0, 'med':1, 'high':2, 'vhigh':3})
df.Doors=df.Doors.replace({'2':0, '3':1, '4':2, '5more':3})
df.Persons=df.Persons.replace({'2':0, '4':1, 'more':2})
df.Lug_Boot=df.Lug_Boot.replace({'small':0, 'med':1, 'big':2})
df.Safety=df.Safety.replace({'low':0, 'med':1, 'high':2})
df.Decision=df.Decision.replace({'unacc':0, 'acc':1, 'good':2, 'vgood':3})

# Now let's see the head of our dfframe.
print("After Trimming and correcting the dfset looks like follows : ")
print(df.head())

# Extracting Independent and dependent Variable
X = df[df.columns[:-1]]
Y = df['Decision']

# Splitting the dfset into training and testing set
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size= 0.1, random_state=10)
```

```
# Initialize a Multi-layer Perceptron classifier.

mlp = MLPClassifier(solver='lbfgs', alpha=1e-5,random_state=10, max_iter=1000,
shuffle=True, verbose=False)

# Train the classifier.
mlp.fit(X_train, Y_train)

# Make predictions.
Y_pred = mlp.predict(X_test)


class_names = ["Unacc","Acc","Good","VGood"]
# Plot confusion matrix for MLP.
mlp_matrix = confusion_matrix(Y_test,Y_pred)
plt.figure(figsize=(8,8))
sns.set(font_scale=1.4)
sns.heatmap(mlp_matrix,annot=True,square=True,
cbar=False,linewidth=0.5,fmt="d",cmap="Blues",xticklabels=class_names,
yticklabels=class_names)
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix for Analysis');


# Actual and Predicted Values
print("Actual Values")
print(Y_test)

print("Predicted Values")
```

```
print(Y_pred)

print("Prediction of first ten elements : ",list(Y_pred)[0:10])
print("Actual Value of first ten elements: ",list(Y_test)[0:10])

# Checking the accuracy of our model
print('Accuracy: ',accuracy_score(Y_test,Y_pred))
print('Precision: %.3f' % precision_score(Y_test, Y_pred,average='micro'))
print('Recall: %.3f' % recall_score(Y_test, Y_pred,average='micro'))

# Our Model Report
print('*************** Evaluation on Our Model ***************')
print('Accuracy Score: ', accuracy_score(Y_test,Y_pred))
# Look at classification report to evaluate the model
print(classification_report(Y_test, Y_pred))
print('--------------------------------------------------------')
print("")


print('*************** Weights and Bias ***************')

print("weights between input and first hidden layer:")
print(mlp.coefs_[0])
print("\nweights between first hidden and second hidden layer:")
print(mlp.coefs_[1])


print("We can generalize the above to access a neuron in the following way:")
print(mlp.coefs_[0][:,0])
```

print("w0 = ", mlp.coefs_[0][0][0])

print("w1 = ", mlp.coefs_[0][1][0])

print("w2 = ", mlp.coefs_[0][2][0])

print("w3 = ", mlp.coefs_[0][3][0])

print("w4 = ", mlp.coefs_[0][4][0])

print("w5 = ", mlp.coefs_[0][5][0])

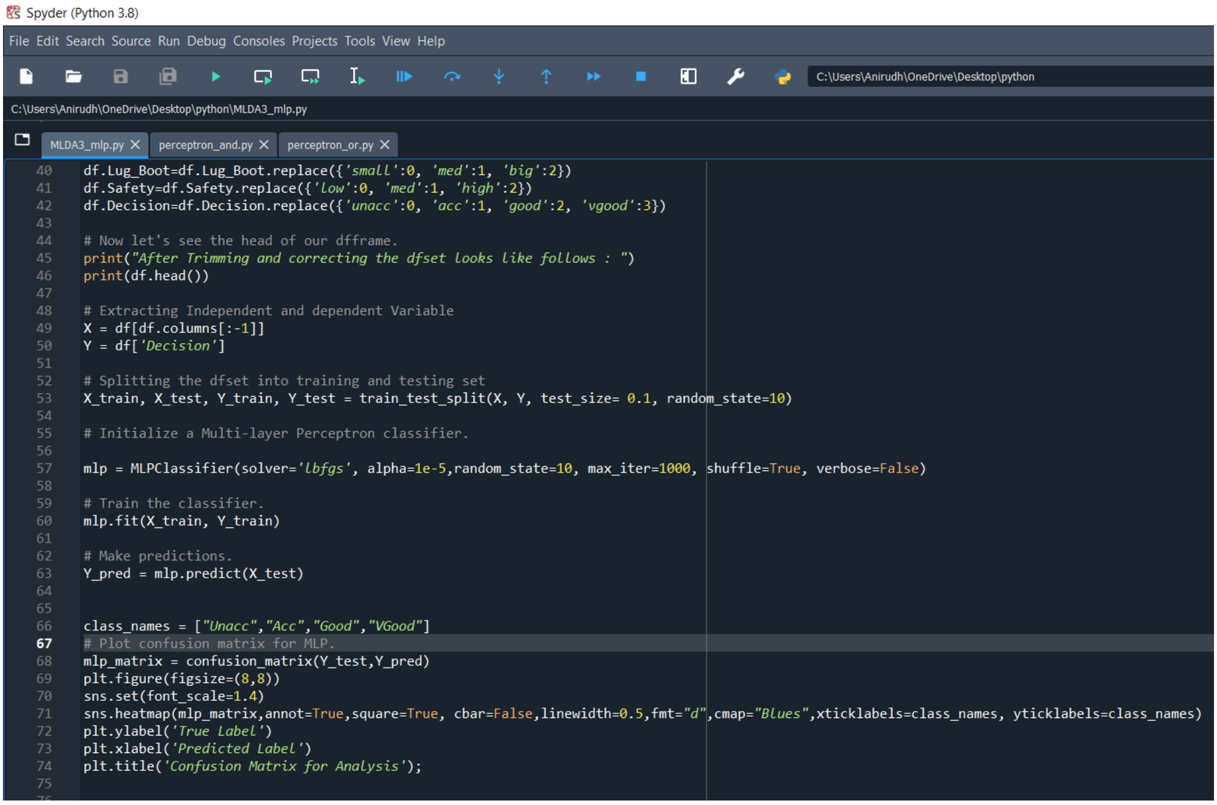
print("Bias values for first hidden layer:")

print(mlp.intercepts_[0])

print("\nBias values for second hidden layer:")

print(mlp.intercepts_[1])

Code Snippets:



Output and Results:

Dataset Details:

Dataset:

```
In [16]: runfile('C:/Users/Anirudh/OneDrive/Desktop/python/MLDA3_ml
     Buying  Maint  Doors Persons Lug_Boot Safety Decision
0     vhigh  vhigh      2       2    small    low    unacc
1     vhigh  vhigh      2       2    small    med    unacc
2     vhigh  vhigh      2       2    small   high    unacc
3     vhigh  vhigh      2       2      med    low    unacc
4     vhigh  vhigh      2       2      med    med    unacc
...     ...    ...    ...     ...      ...    ...      ...
1723    low    low  5more    more      med    med     good
1724    low    low  5more    more      med   high    vgood
1725    low    low  5more    more      big    low    unacc
1726    low    low  5more    more      big    med     good
1727    low    low  5more    more      big   high    vgood

[1728 rows x 7 columns]


Checking for missing values :
Buying      0
Maint       0
Doors       0
Persons     0
Lug_Boot    0
Safety      0
Decision    0
dtype: int64
```

As the missing values is none we can proceed further:

Dataset Details:

```
dfset Header :
  Buying  Maint Doors Persons Lug_Boot Safety Decision
0  vhigh  vhigh     2       2    small    low    unacc
1  vhigh  vhigh     2       2    small    med    unacc
2  vhigh  vhigh     2       2    small   high    unacc
3  vhigh  vhigh     2       2      med    low    unacc
4  vhigh  vhigh     2       2      med    med    unacc
```

```
Information regarding the columns :
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Buying    1728 non-null   object
 1   Maint     1728 non-null   object
 2   Doors     1728 non-null   object
 3   Persons   1728 non-null   object
 4   Lug_Boot  1728 non-null   object
 5   Safety    1728 non-null   object
 6   Decision  1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
None


dfset Details :
       Buying  Maint Doors Persons Lug_Boot Safety Decision
count    1728   1728  1728    1728     1728   1728     1728
unique      4      4     4       3        3      3        4
top     vhigh  vhigh     2       2    small    low    unacc
freq      432    432   432     576      576    576     1210
```

We convert all the ordinal data into numerical data to perform calculations:

```
After Trimming and correcting the dfset looks like follows :
   Buying  Maint  Doors  Persons  Lug_Boot  Safety  Decision
0       3      3      0        0         0       0         0
1       3      3      0        0         0       1         0
2       3      3      0        0         0       2         0
3       3      3      0        0         1       0         0
4       3      3      0        0         1       1         0
```

Weights and Bias:

***************** Weights and Bias *****************

weights between input and first hidden layer:

[[-5.89006527e+00 -8.61129856e-01 -4.89471304e-01 8.35068418e+00

  2.72825035e+00 -3.73319076e-01 -8.84089616e-01 -1.42160270e+00

 -2.95354067e+00 -8.58595201e-01 -1.45173328e+00 6.83208485e+00

-1.22462762e+00 -2.90366885e+00 -2.60664498e+00 -5.31222146e-01
-9.32662808e-02 -1.07911801e+00 -1.19994080e+00 -1.22174174e+00
-1.55857552e+00 -3.26966971e+00 -1.14971897e+00 2.60870944e+00
-8.10980153e-01 -1.65791369e+00 1.22767567e+00 3.40736435e+00
2.19969133e+00 -3.09467927e+00 1.17702550e+00 2.69999663e+00
6.61572257e+00 -3.33107663e-01 1.69062724e+00 3.94890963e+00
-1.83709035e+00 -3.93981864e+00 -2.15588687e-01 6.75848382e+00
-4.85867973e-01 2.16596969e+00 -7.03313685e-01 1.49178217e+01
-2.10091881e+00 4.73246344e+00 -2.98778700e-01 -4.86515831e-01
-5.59666378e-01 -3.24067661e-01 1.21802120e+00 -3.24244316e+00
-5.67292801e-01 -1.30565270e-01 -9.96559894e+00 -5.19453135e-01
9.01043636e+00 -3.18694963e+00 4.51513399e+00 -6.89970215e-01
8.07163988e-02 -7.06365644e-02 3.45969453e-01 1.91867691e+00
-1.73178081e+00 -8.40437987e-01 -6.31722605e-02 -2.56769567e+00
1.14630832e+01 -2.40074167e+00 9.18829046e-01 2.56421526e-01
2.98802216e+00 -3.85339653e-01 -1.21542443e+00 -7.77992835e-02
3.20765101e+00 -1.70009988e+00 1.10210625e+00 -3.92495055e-02
-5.40627220e-01 2.65513459e+00 -2.04298859e+00 2.68992657e+00
3.62359494e+00 -1.73403142e+00 -1.47628832e+00 -2.14151149e-01
-1.02277714e+00 -5.38585218e-01 1.79985894e-02 -6.31266838e-01
-2.07331226e-01 5.77079829e+00 -2.88504804e-01 -2.11229018e+00
9.03794468e+00 -1.54942271e-01 -8.64665836e-02 9.41949134e-03]
[4.34501512e+00 -7.90194133e-01 -2.07197555e-01 8.76576256e+00
2.15966832e+00 -4.45404052e-01 9.35702635e-01 -3.59729212e-01
-4.48937278e+00 -9.17580819e-01 -3.13060459e-01 8.79853708e-01
-1.78492893e+00 -5.53037256e-01 -1.62047236e+00 -2.89281437e-01
-1.23298517e+00 -1.22232185e+00 1.16366880e+01 -5.64467875e-01
-4.32652530e+00 5.80309384e+00 -1.02599964e+00 2.67777464e+00
-2.56264238e-01 -6.22117845e-01 -2.15787496e+00 3.73975603e+00

1.57463833e+00 -2.12690952e+00 3.41011908e+00 2.49605983e+00
5.09424026e+00 -8.94729240e-01 -1.30346943e+00 -1.11919329e+00
-2.95136158e+00 6.68741112e+00 -2.30831695e-01 7.46967165e+00
-5.46885562e-01 4.52734338e+00 -1.38689622e+00 1.37136385e+01
4.42072594e-01 -5.05495017e-01 -1.87206278e-01 -2.92630778e-01
-8.55061542e-01 -2.40824393e-01 5.85434365e+00 -2.54602702e+00
-2.45953592e-01 -2.15107517e+00 -4.79072840e+00 -2.51953476e-01
7.49807403e+00 -1.64662814e+00 4.72538967e+00 -8.78557305e-01
-9.71098710e-02 -2.39947956e-01 -4.85296592e+00 1.54003788e+00
-3.34077575e+00 -1.12486062e+00 -1.86734096e+00 -3.71192685e+00
8.67179536e+00 -1.60770944e+00 -1.32664642e-01 -8.34752488e-01
2.82482873e+00 -5.27770742e-01 -1.82540740e+00 3.78933889e-01
2.08198395e+00 -6.06832921e-01 1.47206551e+00 -4.83776015e-02
-4.85260435e-01 4.37364617e+00 -3.66340235e-01 2.72479181e+00
9.98527746e+00 4.67101405e+00 -1.19887436e+00 -3.16775779e-01
-2.00393887e-01 -3.07021360e-01 -1.94452421e-01 -1.50089085e+00
-2.01224590e-01 3.16366714e+00 -4.83466269e-01 -1.67600405e+00
5.87043679e+00 2.00708544e+00 -7.07932576e-02 -1.93442824e-01]
[-5.58786554e-02 -6.60908308e-01 -1.83823868e+00 -1.51031988e+00
4.33250987e+00 -5.76530189e-01 2.17728216e+00 -1.49131270e+00
9.71979058e-01 -1.14224976e+00 -3.11768106e-01 -8.96592163e-01
-2.66144116e+00 6.37912219e-01 -3.66731875e+00 -2.59339270e+00
-7.02502842e-01 -1.01260987e+00 3.63826114e-01 -3.39290379e-01
3.25026172e+00 -1.67235213e+00 7.84494454e-01 -5.21714435e-01
5.39920285e-01 5.83279189e-03 -5.03518400e-01 2.02871182e+00
2.00063856e+00 2.51088400e+00 -2.49610439e-01 8.66839837e-01
3.98445897e+00 -1.34378964e+00 -8.87172805e-03 7.77489409e-01
-3.39011255e+00 -1.57498079e+00 5.33312229e-03 -4.52450961e+00
-9.71295886e-02 -4.12063402e+00 -1.81466420e+00 2.15770403e-01

6.21023960e-01 1.37957840e+00 3.73090759e-03 -2.91614640e-01
-1.67843662e+00 -3.62534471e-01 4.55305643e+00 4.84620953e-01
-5.82294813e-01 -7.10289490e-01 -4.52851358e+00 -1.34159225e-01
7.90063042e-01 -9.85566049e-01 5.01205141e+00 5.84862699e-01
-1.12573795e-01 -3.28824623e-01 5.44220781e-01 7.74635985e-01
1.25555061e+00 -1.16621079e+00 -1.24196724e+01 -5.78524485e-02
1.89612793e+00 3.29225586e+00 -1.02841738e+00 -2.63327922e+00
1.05044966e+00 -2.00883391e-01 1.34650919e+00 -1.97686211e+00
1.07571602e+00 -8.74898183e-01 4.02198884e-01 -2.33514718e-01
-2.80576520e-01 -1.77343169e+00 2.24254367e+00 3.34746314e-01
2.70437558e-01 7.26552552e-01 -5.88096110e+00 -2.42612666e-01
-4.85125590e-01 -3.96362563e-01 -1.97232822e-01 -1.38477401e+00
-9.29848412e-02 2.59781597e+00 -1.01906822e+00 -1.79811571e+00
-2.29485360e+00 2.11953876e+00 -2.10958215e-01 -9.63579409e-02]
[-2.44585010e+00 -2.92828700e-01 -1.48896691e+00 3.86274566e-01
3.39025605e+00 3.28024313e-01 -1.30338179e+01 -8.28995161e-01
3.74196101e+00 -1.42343366e+00 -5.11774348e+00 -2.34575822e+00
-1.84236693e+00 -5.05677016e+00 -2.89655522e+00 -2.10450754e+00
-8.02807071e-01 6.13630463e-01 1.42701522e+00 -3.28552164e-01
-6.01519684e-01 1.11115815e+00 -8.80891264e-01 6.79839049e+00
-1.63917573e+01 -6.95846911e+00 -6.37851373e-01 -1.87514981e-01
-7.22829388e+00 2.76457627e+00 5.46892280e+00 -1.65692110e+01
-2.58248848e+00 5.74695788e-01 2.14384929e+00 6.49474561e+00
-3.13051700e+00 -8.56080272e-01 5.49564059e-02 1.44406889e+00
-3.04185461e-01 5.91789159e+00 -8.16221959e-01 9.84688167e-01
-2.35061889e-01 1.35641975e+00 -2.36045955e-01 -2.94848476e-01
-8.39276376e-01 -1.22854176e+00 -3.07792826e-01 -4.08123173e-01
-1.38221156e-01 3.98369598e+00 -7.69510271e+00 -5.51513909e-01
-5.82641494e+00 -1.92571163e+00 3.88480074e+00 -3.81071925e-01

-1.96272586e-01 -7.92980610e-01 -4.86370719e-01 -1.32956422e-01
4.80051942e-01 -4.98573772e+00 7.38462588e+00 -4.64307250e+00
5.06340439e-01 2.42928759e+00 -2.97329797e+00 -3.33223419e+00
2.62584948e+00 -2.93406077e-01 8.66603519e-01 -1.56416761e+00
-7.47011115e+00 -2.99284765e-01 1.58741551e+00 -5.14668786e-02
-1.36752488e-01 -1.29341233e+00 8.32907439e-01 -1.00685098e+00
-1.55332026e+00 -3.49417915e+00 -1.07526483e+01 -2.17467367e-01
-8.69864535e-01 1.83703483e-01 -9.09814550e-02 -4.68055576e-01
-2.28696553e-01 -2.53304007e+00 -3.34416153e-01 -1.27144350e+00
-3.26095063e+00 1.03008087e+00 -4.17578199e-02 -1.17438428e-01]
[-1.08624662e+00 4.80251148e-01 2.24445239e-01 -3.51807119e+00
2.33839288e+00 -1.08656257e-01 9.57661479e-01 -7.18677546e-01
5.77320780e+00 -8.51552862e-01 2.97249330e-01 6.94927548e+00
-8.46368257e-01 -1.59549071e+00 -4.82458648e-01 -9.01835202e-01
-2.22643630e-01 -2.12977660e+00 -8.91462819e-01 -4.29206684e-01
-2.36330496e+00 -6.43711254e+00 -1.79184218e-01 -2.84839650e+00
9.02382389e-01 -8.32481574e-01 6.11763633e-01 2.44046456e+00
3.95297335e+00 6.95212396e+00 -4.38659486e-01 2.72884973e+00
3.05480900e+00 6.15326105e+00 1.93981541e+00 -1.74967773e+00
-1.58730773e+00 -6.34446992e+00 -1.00801247e-01 -7.94442259e-01
-1.88293409e-01 -1.34757959e+00 -4.40980772e-01 -2.11532529e+00
1.40225181e-01 2.65246455e+00 -1.86460252e-01 -2.84403545e-01
-4.28918441e-01 -6.44065649e-01 3.94792232e+00 -8.35720761e-01
-2.38226308e-01 5.44020554e-02 -2.29689069e+00 -6.02566034e-02
-2.88852880e+00 2.97993904e-01 3.84188872e+00 -2.87823346e+00
-1.05110119e-01 6.19368063e-02 -7.50816448e-01 2.59091796e+00
1.09162174e+00 -2.79626447e+00 -5.47912984e+00 5.41805393e+00
-9.62254787e-01 -3.82806023e-01 -3.73039201e-01 -2.35122886e+00
5.89566707e-01 -2.04453983e-01 4.89811038e-01 -1.31923773e+00

2.28523234e+00 -4.81137452e-01 2.75609114e-01 -6.95699400e-02
-1.46148145e-01 -1.33599619e+00 -3.48429924e+00 3.95306930e-01
-2.51082260e+00 2.55969917e+00 2.01569956e+00 -1.69060644e-01
-6.29943769e-01 -3.63028230e-01 -8.85958331e-02 -4.76014836e-01
-2.28513369e-01 3.71742247e+00 -2.93777963e-01 -1.10244611e+00
-8.41190787e+00 5.24510094e+00 4.44181958e-03 -9.73081676e-03]
[-3.73983716e+00 -1.61182678e-01 2.76635194e-01 -8.47468746e+00
2.21184938e+00 -5.34748843e-01 1.27314859e+00 -5.06809994e-02
-1.35764517e+01 1.53768664e+00 1.27974254e+00 -3.06004453e+00
1.10915224e+00 6.89939355e-01 -1.36662336e+00 -2.05093923e+00
-1.65859404e+00 -2.08078723e+00 1.84243613e+00 -4.34212164e-01
-1.96298735e+00 4.36236177e+00 -2.43803909e-01 1.39609251e+00
6.60704130e+00 1.92733546e+00 -5.29137090e-01 2.37910630e+00
3.03332568e+00 -1.36539335e+01 -2.57936300e+00 3.81901683e-01
-7.40618982e+00 6.85441078e+00 -1.58372573e-01 -3.00819136e+00
-4.48230642e+00 -8.42750688e+00 -8.43048476e-02 -4.55661896e+00
-1.45337746e-01 7.19720157e-01 1.03438283e-01 -7.94043496e+00
-1.50664842e+00 -5.46385861e+00 2.63728921e-01 -2.94121099e-01
-3.88683236e-01 -2.16442525e-01 4.43489744e+00 6.82038048e+00
-2.21962895e-01 -3.19482911e+00 4.04722467e+00 -8.04316229e-01
-4.38033148e-01 -6.48300612e-01 -4.40750910e-01 -2.07978865e+00
-1.84015828e-01 -5.49211665e-01 -2.73838706e-01 -1.40515786e+01
3.95168977e-01 9.95802167e+00 -1.50620329e+00 -7.23504069e+00
-1.14474001e+01 -6.23496121e+00 -7.90386647e-01 -3.85226522e+00
8.37332844e-01 -2.39598045e-01 -2.04315377e+00 1.69919679e-01
-3.34685436e+00 -7.66153265e-01 1.47906383e+00 -2.99170570e-02
1.91397192e-01 9.44899846e+00 -1.63406182e+00 3.01810168e+00
-4.80467056e+00 -2.52572298e+00 1.18428331e+00 -4.15051987e-01
3.86622199e-02 -4.00099802e-01 -1.06970046e-01 -3.75086628e-01

-1.71394097e-01 -6.18355840e+00 -4.23291234e-01 -1.93006495e+00
-4.55970816e+00 3.63937137e+00 -7.28936827e-03 6.62987604e-02]]

weights between first hidden and second hidden layer:

[[4.49453259e+00 -3.18422025e+00 -7.53019330e-01 -4.65013127e-01]
[4.10884538e+00 -2.01719477e+00 -9.34572077e-01 -1.23737974e+00]
[4.29884457e-01 1.17134055e+00 9.05116475e-01 -2.17040859e+00]
[5.23662256e+00 -1.41638639e+00 -2.83146276e-01 -3.55197977e+00]
[-5.10245580e+00 2.12745244e+00 -2.64576250e-01 3.17733695e+00]
[5.24229729e-01 -2.32521171e-01 -3.41170376e-01 -2.12368507e-01]
[8.27365423e+00 -1.68839752e+00 -2.96167311e+00 -3.54315112e+00]
[6.94844622e-01 -4.00548347e-01 -1.10840157e-01 -1.41208666e-01]
[9.33451201e+00 -7.53440211e+00 5.93143987e+00 -7.89285879e+00]
[1.68702933e+00 -1.44335120e+00 2.32668128e-01 -4.61058756e-01]
[3.66388703e+00 -2.08702295e+00 -1.32073181e+00 -2.80527583e-01]
[-1.67915199e+00 6.57279444e+00 -9.10772232e-01 -3.93672287e+00]
[2.16211963e+00 -1.12079026e+00 -2.37891086e-01 -6.49533928e-01]
[5.73008520e+00 -3.91909826e+00 -1.40877754e+00 -1.16665371e-01]
[-7.07990781e+00 3.03538574e+00 2.17472314e+00 1.50813455e+00]
[5.72945598e-01 1.60195733e-01 1.26990275e-01 -3.88540280e-01]
[1.61433115e+00 -1.16802409e+00 -5.45032811e-02 -1.30951368e-01]
[1.01770561e+00 -8.42436542e-01 -1.09071117e-01 -8.47245775e-02]
[6.11902235e+00 -3.40924523e+00 -3.07558021e-01 -2.21003266e+00]
[2.25918403e-01 2.80572680e-02 9.94398095e-02 -1.67772625e-01]
[4.53310318e+00 8.27066165e-01 -2.01759511e+00 -2.97448446e+00]
[-3.55069762e-01 5.41066175e+00 1.01551155e+00 -5.86668828e+00]
[1.01396046e+00 -9.28138089e-01 9.28464699e-01 -1.37946647e+00]
[5.62583910e+00 -4.54164739e+00 -1.54457782e+00 6.68143790e-01]
[1.48814795e+01 -9.67202605e+00 -2.25333992e+00 -2.58899240e+00]

```
[ 4.87012626e+00 -2.74375065e+00 -1.05129052e+00 -8.31496236e-01]
[-3.05029729e-01  2.09799156e+00  8.26969044e-02 -1.36032361e+00]
[-4.56916203e+00  1.13826753e-01  1.42472577e+00  2.78636632e+00]
[ 4.72772681e+00 -1.04003115e+00 -3.43843160e+00 -3.53077804e-01]
[ 7.25419317e+00 -1.66066180e+00  4.26736133e+00 -9.36194397e+00]
[ 1.78317563e+00 -3.98488552e-01  4.11224088e-01 -1.45794881e+00]
[ 8.45622499e+00 -5.15589951e+00 -1.36520067e+00 -2.07474470e+00]
[ 8.28214368e+00 -4.31253458e+00  1.24124983e+00 -5.48144116e+00]
[ 3.79058836e+00 -3.32151232e+00 -2.89614128e+00  2.79775482e+00]
[ 3.98565255e-01  3.41829112e+00 -8.22255317e-01 -3.45401881e+00]
[ 4.28406082e+00  1.09449010e+00 -8.98876970e-01 -4.81119174e+00]
[-6.83147970e-01  9.10061985e-02  3.47874569e-01  1.59902274e-01]
[ 8.29083230e+00  2.46851069e+00 -4.00684877e+00 -6.69123629e+00]
[-2.13365635e-01 -2.46069819e-02  1.92539630e-01  1.95642547e-01]
[-7.86641563e+00  6.49356749e+00 -7.53792346e-01  2.23689780e+00]
[-4.18369714e-02  1.86360620e-01  1.79952817e-01  2.15880954e-01]
[ 6.14896272e+00 -5.72780205e+00 -1.24009882e+00  3.25727920e-01]
[ 4.72996672e-01 -1.51835520e-01 -8.39351250e-02 -3.75424822e-02]
[ 1.27987894e+00  1.12118005e+01 -8.05099083e+00 -4.69617356e+00]
[ 2.18488585e+00 -9.86417498e-01 -3.10511383e-01 -1.15938727e+00]
[ 2.49281837e+00  8.97840624e-01  3.22817337e+00 -6.07132812e+00]
[ 2.48088974e-01  4.98965638e-02  3.51355647e-01 -2.52032278e-01]
[ 3.40233280e-01 -3.81082424e-01 -1.15845747e-01 -1.58700173e-01]
[ 2.20251738e+00 -1.29431140e+00 -3.37801678e-01 -4.07657591e-01]
[-2.90927062e-02  7.62909411e-02  8.40346900e-02 -4.21899027e-02]
[-5.30587962e+00 -1.35457019e+00 -1.26183992e+00  7.95755705e+00]
[ 4.56641043e-01 -7.58293790e+00  4.21015730e+00  2.46311689e+00]
[ 1.55213623e-02  4.05688798e-02  1.43447253e-01  1.41892472e-01]
[ 9.49611286e-02  2.76500989e+00 -9.87439822e-01 -1.52189170e+00]
```

[6.39823462e+00 -7.22582143e+00 -3.59153909e-01 1.48115689e+00]

[4.51938480e-01 -4.42325439e-01 -3.53862379e-02 1.06583806e-02]

[9.36387874e+00 -4.91554539e+00 -4.47285715e+00 -1.27840064e-01]

[-9.35515187e-01 -4.46699223e-01 -2.10213915e-01 1.59979927e+00]

[-1.05477265e+01 2.93489159e+00 3.50011225e+00 4.39095992e+00]

[2.46553657e+00 -4.19230990e-01 -8.53181342e-01 -9.75330944e-01]

[1.37230531e-01 -6.99543118e-02 -6.35254638e-02 9.56995904e-02]

[4.52379758e-01 8.33361496e-01 -1.14402764e-01 -1.15966805e+00]

[2.99353893e+00 -6.28959698e-01 -3.32519157e-01 -2.23029000e+00]

[7.00596276e+00 -5.25036144e+00 -5.21777011e-01 -1.46400675e+00]

[2.70362889e+00 1.30709167e+00 -1.71985059e+00 -2.24363325e+00]

[-5.22305041e+00 6.48491539e+00 6.22708252e+00 -7.80663119e+00]

[9.32013647e+00 -1.49554866e-01 -2.92240028e+00 -6.70739604e+00]

[5.98054590e+00 -2.53256377e+00 1.60695798e+00 -4.99169181e+00]

[5.89651082e+00 -2.13259196e+00 4.31248732e-01 -4.35795512e+00]

[4.48373397e+00 -1.85664860e+00 -6.93926483e-01 -2.28330510e+00]

[8.77381983e-01 1.31142787e+00 -1.08729459e+00 -1.60822058e+00]

[-8.48136612e-02 1.82187144e+00 -9.68834146e-01 -2.46990713e-01]

[-2.88630471e+00 -2.71112142e-01 8.79318823e-01 1.72898413e+00]

[-2.36725308e-02 -2.77146927e-01 -7.10330121e-02 -2.50949767e-02]

[1.83122793e+00 -8.79050085e-01 -1.57132298e-01 -1.19835228e+00]

[2.17716819e+00 -5.25435230e+00 1.74845833e+00 1.44547874e+00]

[-3.89740090e+00 5.10413751e+00 -5.71575336e-01 -8.07997754e-01]

[-4.14088137e-01 3.52182990e-01 2.40663307e-01 4.23206755e-03]

[4.17039431e-01 -1.27375975e+00 1.50939186e-01 7.73212294e-01]

[-8.33235196e-02 6.96715128e-02 1.55532496e-01 -3.21944965e-02]

[2.57217352e-01 -2.42523993e-01 -1.50292345e-01 -2.46406393e-01]

[7.25407182e+00 -3.89758139e+00 4.23432890e-01 -3.90015981e+00]

[4.48606871e+00 -2.42396143e+00 -1.09530475e-01 -2.28399900e+00]

[-2.23490237e+00 -8.17199600e-01 1.14856663e+00 1.89103863e+00]

[5.94559928e-01 6.31125899e+00 -3.92826489e+00 -2.84486272e+00]

[-1.85053552e+00 5.21100690e+00 -1.28448950e+00 -2.21359631e+00]

[7.54110888e+00 -2.60560654e+00 -1.58862658e+00 -3.72142524e+00]

[1.12186599e-01 1.25788583e-01 9.75183738e-02 1.08715057e-01]

[-3.67980293e-02 5.36760601e-02 -1.80217052e-01 1.52656604e-01]

[5.03723404e-01 -5.74762529e-01 -2.44460001e-01 5.64267312e-02]

[2.14710169e-01 9.77128087e-02 -1.35314456e-01 -1.34367382e-01]

[5.58404275e-01 -2.95607865e-01 1.37401825e-01 -2.99232911e-01]

[8.03259455e-03 -6.91541931e-02 1.41567242e-01 1.86105754e-01]

[8.54234037e+00 -1.53326009e+00 1.55077983e+00 -8.12500356e+00]

[3.15961945e-01 -4.46652826e-01 -3.97361185e-02 -2.27681481e-01]

[-2.04135583e+00 9.15927784e-01 1.21031175e+00 6.04713813e-01]

[-7.89855834e-01 7.97009484e+00 -3.57422136e+00 -3.59507306e+00]

[-2.12968534e+00 2.00929139e+00 -1.52731439e+00 1.40411765e+00]

[-1.78157637e-01 7.96556295e-03 2.33945550e-01 2.36149996e-01]

[-1.64390529e-01 -4.09546354e-02 8.95844405e-02 -6.88735539e-02]]

```
We can generalize the above to access a neuron in the following way:
[-5.89006527  4.34501512 -0.05587866 -2.4458501  -1.08624662 -3.73983716]
w0 =  -5.890065268157985
w1 =  4.345015156978585
w2 =  -0.055878655429227805
w3 =  -2.4458501014744747
w4 =  -1.0862466217312448
w5 =  -3.7398371613520327
Bias values for first hidden layer:
[-0.77841513  0.0149091  -0.34669671  2.13094579  1.69044835 -0.63393133
  1.50250092 -1.07291586  3.44964608 -1.44615686 -0.48598281 -4.53724444
 -1.96950564 -1.27129491 -3.33380552 -0.89571383  0.41923804 -1.5192849
 -7.11644368 -0.56678411  1.25470954  6.62762869  0.17816472 -0.62168611
  1.56874741  1.43486026 -0.61743552  0.67434656 -2.34323451  4.27840062
  1.13511362 -0.38935192  3.66494159 -6.12433497  3.0243849   2.34065497
  0.03387637  1.34140653 -0.20678213 -0.73353102 -0.33998485 -1.17104084
 -0.94524946 -3.95374824  0.8835631   1.48373113 -0.200298   -0.39073234
 -0.84988981 -0.9161537  -8.10730784 -5.46237547 -0.47988245  2.57048299
 -5.70945347 -0.50888198  0.10894943 -2.8755956   5.48494954 -0.07802834
 -0.0648897  -0.48268812  2.65205645  2.84799458  2.00996351  2.02948839
 -1.9183676   3.03530231  4.11720721  1.89281495 -0.10194413 -0.11029074
  0.34206237 -0.46411032  0.28416077 -2.59404363 -0.71510259 -0.3293978
  0.87625783 -0.22946792 -0.32500329  0.74622628  0.96579572 -3.34691003
  1.25939244  1.44725032  1.60452894 -0.33482216 -0.83766892 -0.58529995
  0.26083856 -0.77060792 -0.0670628   3.1071647  -0.52948427 -1.14333139
  0.56714474 -2.47479718 -0.02705274 -0.22568655]

Bias values for second hidden layer:
[  9.58959949   2.86165903   1.95981096 -14.61840811]
```

Initial Weights and Bias:

Initial Weights:

weights between input and first hidden layer:

[[-5.89006527e+00 -8.61129856e-01 -4.89471304e-01 8.35068418e+00
2.72825035e+00 -3.73319076e-01 -8.84089616e-01 -1.42160270e+00
-2.95354067e+00 -8.58595201e-01 -1.45173328e+00 6.83208485e+00
-1.22462762e+00 -2.90366885e+00 -2.60664498e+00 -5.31222146e-01
-9.32662808e-02 -1.07911801e+00 -1.19994080e+00 -1.22174174e+00
-1.55857552e+00 -3.26966971e+00 -1.14971897e+00 2.60870944e+00
-8.10980153e-01 -1.65791369e+00 1.22767567e+00 3.40736435e+00
2.19969133e+00 -3.09467927e+00 1.17702550e+00 2.69999663e+00
6.61572257e+00 -3.33107663e-01 1.69062724e+00 3.94890963e+00
-1.83709035e+00 -3.93981864e+00 -2.15588687e-01 6.75848382e+00
-4.85867973e-01 2.16596969e+00 -7.03313685e-01 1.49178217e+01
-2.10091881e+00 4.73246344e+00 -2.98778700e-01 -4.86515831e-01
-5.59666378e-01 -3.24067661e-01 1.21802120e+00 -3.24244316e+00
-5.67292801e-01 -1.30565270e-01 -9.96559894e+00 -5.19453135e-01
9.01043636e+00 -3.18694963e+00 4.51513399e+00 -6.89970215e-01
8.07163988e-02 -7.06365644e-02 3.45969453e-01 1.91867691e+00
-1.73178081e+00 -8.40437987e-01 -6.31722605e-02 -2.56769567e+00
1.14630832e+01 -2.40074167e+00 9.18829046e-01 2.56421526e-01
2.98802216e+00 -3.85339653e-01 -1.21542443e+00 -7.77992835e-02
3.20765101e+00 -1.70009988e+00 1.10210625e+00 -3.92495055e-02
-5.40627220e-01 2.65513459e+00 -2.04298859e+00 2.68992657e+00
3.62359494e+00 -1.73403142e+00 -1.47628832e+00 -2.14151149e-01
-1.02277714e+00 -5.38585218e-01 1.79985894e-02 -6.31266838e-01
-2.07331226e-01 5.77079829e+00 -2.88504804e-01 -2.11229018e+00
9.03794468e+00 -1.54942271e-01 -8.64665836e-02 9.41949134e-03]
[4.34501512e+00 -7.90194133e-01 -2.07197555e-01 8.76576256e+00

2.15966832e+00 -4.45404052e-01 9.35702635e-01 -3.59729212e-01
-4.48937278e+00 -9.17580819e-01 -3.13060459e-01 8.79853708e-01
-1.78492893e+00 -5.53037256e-01 -1.62047236e+00 -2.89281437e-01
-1.23298517e+00 -1.22232185e+00 1.16366880e+01 -5.64467875e-01
-4.32652530e+00 5.80309384e+00 -1.02599964e+00 2.67777464e+00
-2.56264238e-01 -6.22117845e-01 -2.15787496e+00 3.73975603e+00
1.57463833e+00 -2.12690952e+00 3.41011908e+00 2.49605983e+00
5.09424026e+00 -8.94729240e-01 -1.30346943e+00 -1.11919329e+00
-2.95136158e+00 6.68741112e+00 -2.30831695e-01 7.46967165e+00
-5.46885562e-01 4.52734338e+00 -1.38689622e+00 1.37136385e+01
4.42072594e-01 -5.05495017e-01 -1.87206278e-01 -2.92630778e-01
-8.55061542e-01 -2.40824393e-01 5.85434365e+00 -2.54602702e+00
-2.45953592e-01 -2.15107517e+00 -4.79072840e+00 -2.51953476e-01
7.49807403e+00 -1.64662814e+00 4.72538967e+00 -8.78557305e-01
-9.71098710e-02 -2.39947956e-01 -4.85296592e+00 1.54003788e+00
-3.34077575e+00 -1.12486062e+00 -1.86734096e+00 -3.71192685e+00
8.67179536e+00 -1.60770944e+00 -1.32664642e-01 -8.34752488e-01
2.82482873e+00 -5.27770742e-01 -1.82540740e+00 3.78933889e-01
2.08198395e+00 -6.06832921e-01 1.47206551e+00 -4.83776015e-02
-4.85260435e-01 4.37364617e+00 -3.66340235e-01 2.72479181e+00
9.98527746e+00 4.67101405e+00 -1.19887436e+00 -3.16775779e-01
-2.00393887e-01 -3.07021360e-01 -1.94452421e-01 -1.50089085e+00
-2.01224590e-01 3.16366714e+00 -4.83466269e-01 -1.67600405e+00
5.87043679e+00 2.00708544e+00 -7.07932576e-02 -1.93442824e-01]
[-5.58786554e-02 -6.60908308e-01 -1.83823868e+00 -1.51031988e+00
4.33250987e+00 -5.76530189e-01 2.17728216e+00 -1.49131270e+00
9.71979058e-01 -1.14224976e+00 -3.11768106e-01 -8.96592163e-01
-2.66144116e+00 6.37912219e-01 -3.66731875e+00 -2.59339270e+00
-7.02502842e-01 -1.01260987e+00 3.63826114e-01 -3.39290379e-01

3.25026172e+00 -1.67235213e+00 7.84494454e-01 -5.21714435e-01
5.39920285e-01 5.83279189e-03 -5.03518400e-01 2.02871182e+00
2.00063856e+00 2.51088400e+00 -2.49610439e-01 8.66839837e-01
3.98445897e+00 -1.34378964e+00 -8.87172805e-03 7.77489409e-01
-3.39011255e+00 -1.57498079e+00 5.33312229e-03 -4.52450961e+00
-9.71295886e-02 -4.12063402e+00 -1.81466420e+00 2.15770403e-01
6.21023960e-01 1.37957840e+00 3.73090759e-03 -2.91614640e-01
-1.67843662e+00 -3.62534471e-01 4.55305643e+00 4.84620953e-01
-5.82294813e-01 -7.10289490e-01 -4.52851358e+00 -1.34159225e-01
7.90063042e-01 -9.85566049e-01 5.01205141e+00 5.84862699e-01
-1.12573795e-01 -3.28824623e-01 5.44220781e-01 7.74635985e-01
1.25555061e+00 -1.16621079e+00 -1.24196724e+01 -5.78524485e-02
1.89612793e+00 3.29225586e+00 -1.02841738e+00 -2.63327922e+00
1.05044966e+00 -2.00883391e-01 1.34650919e+00 -1.97686211e+00
1.07571602e+00 -8.74898183e-01 4.02198884e-01 -2.33514718e-01
-2.80576520e-01 -1.77343169e+00 2.24254367e+00 3.34746314e-01
2.70437558e-01 7.26552552e-01 -5.88096110e+00 -2.42612666e-01
-4.85125590e-01 -3.96362563e-01 -1.97232822e-01 -1.38477401e+00
-9.29848412e-02 2.59781597e+00 -1.01906822e+00 -1.79811571e+00
-2.29485360e+00 2.11953876e+00 -2.10958215e-01 -9.63579409e-02]
[-2.44585010e+00 -2.92828700e-01 -1.48896691e+00 3.86274566e-01
3.39025605e+00 3.28024313e-01 -1.30338179e+01 -8.28995161e-01
3.74196101e+00 -1.42343366e+00 -5.11774348e+00 -2.34575822e+00
-1.84236693e+00 -5.05677016e+00 -2.89655522e+00 -2.10450754e+00
-8.02807071e-01 6.13630463e-01 1.42701522e+00 -3.28552164e-01
-6.01519684e-01 1.11115815e+00 -8.80891264e-01 6.79839049e+00
-1.63917573e+01 -6.95846911e+00 -6.37851373e-01 -1.87514981e-01
-7.22829388e+00 2.76457627e+00 5.46892280e+00 -1.65692110e+01
-2.58248848e+00 5.74695788e-01 2.14384929e+00 6.49474561e+00

-3.13051700e+00 -8.56080272e-01 5.49564059e-02 1.44406889e+00
-3.04185461e-01 5.91789159e+00 -8.16221959e-01 9.84688167e-01
-2.35061889e-01 1.35641975e+00 -2.36045955e-01 -2.94848476e-01
-8.39276376e-01 -1.22854176e+00 -3.07792826e-01 -4.08123173e-01
-1.38221156e-01 3.98369598e+00 -7.69510271e+00 -5.51513909e-01
-5.82641494e+00 -1.92571163e+00 3.88480074e+00 -3.81071925e-01
-1.96272586e-01 -7.92980610e-01 -4.86370719e-01 -1.32956422e-01
 4.80051942e-01 -4.98573772e+00 7.38462588e+00 -4.64307250e+00
 5.06340439e-01 2.42928759e+00 -2.97329797e+00 -3.33223419e+00
 2.62584948e+00 -2.93406077e-01 8.66603519e-01 -1.56416761e+00
-7.47011115e+00 -2.99284765e-01 1.58741551e+00 -5.14668786e-02
-1.36752488e-01 -1.29341233e+00 8.32907439e-01 -1.00685098e+00
-1.55332026e+00 -3.49417915e+00 -1.07526483e+01 -2.17467367e-01
-8.69864535e-01 1.83703483e-01 -9.09814550e-02 -4.68055576e-01
-2.28696553e-01 -2.53304007e+00 -3.34416153e-01 -1.27144350e+00
-3.26095063e+00 1.03008087e+00 -4.17578199e-02 -1.17438428e-01]
[-1.08624662e+00 4.80251148e-01 2.24445239e-01 -3.51807119e+00
 2.33839288e+00 -1.08656257e-01 9.57661479e-01 -7.18677546e-01
 5.77320780e+00 -8.51552862e-01 2.97249330e-01 6.94927548e+00
-8.46368257e-01 -1.59549071e+00 -4.82458648e-01 -9.01835202e-01
-2.22643630e-01 -2.12977660e+00 -8.91462819e-01 -4.29206684e-01
-2.36330496e+00 -6.43711254e+00 -1.79184218e-01 -2.84839650e+00
 9.02382389e-01 -8.32481574e-01 6.11763633e-01 2.44046456e+00
 3.95297335e+00 6.95212396e+00 -4.38659486e-01 2.72884973e+00
 3.05480900e+00 6.15326105e+00 1.93981541e+00 -1.74967773e+00
-1.58730773e+00 -6.34446992e+00 -1.00801247e-01 -7.94442259e-01
-1.88293409e-01 -1.34757959e+00 -4.40980772e-01 -2.11532529e+00
 1.40225181e-01 2.65246455e+00 -1.86460252e-01 -2.84403545e-01
-4.28918441e-01 -6.44065649e-01 3.94792232e+00 -8.35720761e-01

-2.38226308e-01 5.44020554e-02 -2.29689069e+00 -6.02566034e-02
-2.88852880e+00 2.97993904e-01 3.84188872e+00 -2.87823346e+00
-1.05110119e-01 6.19368063e-02 -7.50816448e-01 2.59091796e+00
1.09162174e+00 -2.79626447e+00 -5.47912984e+00 5.41805393e+00
-9.62254787e-01 -3.82806023e-01 -3.73039201e-01 -2.35122886e+00
5.89566707e-01 -2.04453983e-01 4.89811038e-01 -1.31923773e+00
2.28523234e+00 -4.81137452e-01 2.75609114e-01 -6.95699400e-02
-1.46148145e-01 -1.33599619e+00 -3.48429924e+00 3.95306930e-01
-2.51082260e+00 2.55969917e+00 2.01569956e+00 -1.69060644e-01
-6.29943769e-01 -3.63028230e-01 -8.85958331e-02 -4.76014836e-01
-2.28513369e-01 3.71742247e+00 -2.93777963e-01 -1.10244611e+00
-8.41190787e+00 5.24510094e+00 4.44181958e-03 -9.73081676e-03]
[-3.73983716e+00 -1.61182678e-01 2.76635194e-01 -8.47468746e+00
2.21184938e+00 -5.34748843e-01 1.27314859e+00 -5.06809994e-02
-1.35764517e+01 1.53768664e+00 1.27974254e+00 -3.06004453e+00
1.10915224e+00 6.89939355e-01 -1.36662336e+00 -2.05093923e+00
-1.65859404e+00 -2.08078723e+00 1.84243613e+00 -4.34212164e-01
-1.96298735e+00 4.36236177e+00 -2.43803909e-01 1.39609251e+00
6.60704130e+00 1.92733546e+00 -5.29137090e-01 2.37910630e+00
3.03332568e+00 -1.36539335e+01 -2.57936300e+00 3.81901683e-01
-7.40618982e+00 6.85441078e+00 -1.58372573e-01 -3.00819136e+00
-4.48230642e+00 -8.42750688e+00 -8.43048476e-02 -4.55661896e+00
-1.45337746e-01 7.19720157e-01 1.03438283e-01 -7.94043496e+00
-1.50664842e+00 -5.46385861e+00 2.63728921e-01 -2.94121099e-01
-3.88683236e-01 -2.16442525e-01 4.43489744e+00 6.82038048e+00
-2.21962895e-01 -3.19482911e+00 4.04722467e+00 -8.04316229e-01
-4.38033148e-01 -6.48300612e-01 -4.40750910e-01 -2.07978865e+00
-1.84015828e-01 -5.49211665e-01 -2.73838706e-01 -1.40515786e+01
3.95168977e-01 9.95802167e+00 -1.50620329e+00 -7.23504069e+00

-1.14474001e+01 -6.23496121e+00 -7.90386647e-01 -3.85226522e+00

8.37332844e-01 -2.39598045e-01 -2.04315377e+00 1.69919679e-01

-3.34685436e+00 -7.66153265e-01 1.47906383e+00 -2.99170570e-02

1.91397192e-01 9.44899846e+00 -1.63406182e+00 3.01810168e+00

-4.80467056e+00 -2.52572298e+00 1.18428331e+00 -4.15051987e-01

3.86622199e-02 -4.00099802e-01 -1.06970046e-01 -3.75086628e-01

-1.71394097e-01 -6.18355840e+00 -4.23291234e-01 -1.93006495e+00

-4.55970816e+00 3.63937137e+00 -7.28936827e-03 6.62987604e-02]]

Initial Bias:

```
[-0.77841513  0.0149091  -0.34669671  2.13094579  1.69044835 -0.63393133
  1.50250092 -1.07291586  3.44964608 -1.44615686 -0.48598281 -4.53724444
 -1.96950564 -1.27129491 -3.33380552 -0.89571383  0.41923804 -1.5192849
 -7.11644368 -0.56678411  1.25470954  6.62762869  0.17816472 -0.62168611
  1.56874741  1.43486026 -0.61743552  0.67434656 -2.34323451  4.27840062
  1.13511362 -0.38935192  3.66494159 -6.12433497  3.0243849   2.34065497
  0.03387637  1.34140653 -0.20678213 -0.73353102 -0.33998485 -1.17104084
 -0.94524946 -3.95374824  0.8835631   1.48373113 -0.200298   -0.39073234
 -0.84988981 -0.9161537  -8.10730784 -5.46237547 -0.47988245  2.57048299
 -5.70945347 -0.50888198  0.10894943 -2.8755956   5.48494954 -0.07802834
 -0.0648897  -0.48268812  2.65205645  2.84799458  2.00996351  2.02948839
 -1.9183676   3.03530231  4.11720721  1.89281495 -0.10194413 -0.11029074
  0.34206237 -0.46411032  0.28416077 -2.59404363 -0.71510259 -0.3293978
  0.87625783 -0.22946792 -0.32500329  0.74622628  0.96579572 -3.34691003
  1.25939244  1.44725032  1.60452894 -0.33482216 -0.83766892 -0.58529995
  0.26083856 -0.77060792 -0.0670628   3.1071647  -0.52948427 -1.14333139
  0.56714474 -2.47479718 -0.02705274 -0.22568655]
```

Final Weights and Bias:

Final Weights:

Final Bias:

[[4.49453259e+00 -3.18422025e+00 -7.53019330e-01 -4.65013127e-01]

[4.10884538e+00 -2.01719477e+00 -9.34572077e-01 -1.23737974e+00]

[4.29884457e-01 1.17134055e+00 9.05116475e-01 -2.17040859e+00]

[5.23662256e+00 -1.41638639e+00 -2.83146276e-01 -3.55197977e+00]

[-5.10245580e+00 2.12745244e+00 -2.64576250e-01 3.17733695e+00]

[5.24229729e-01 -2.32521171e-01 -3.41170376e-01 -2.12368507e-01]

```
[ 8.27365423e+00 -1.68839752e+00 -2.96167311e+00 -3.54315112e+00]
[ 6.94844622e-01 -4.00548347e-01 -1.10840157e-01 -1.41208666e-01]
[ 9.33451201e+00 -7.53440211e+00  5.93143987e+00 -7.89285879e+00]
[ 1.68702933e+00 -1.44335120e+00  2.32668128e-01 -4.61058756e-01]
[ 3.66388703e+00 -2.08702295e+00 -1.32073181e+00 -2.80527583e-01]
[-1.67915199e+00  6.57279444e+00 -9.10772232e-01 -3.93672287e+00]
[ 2.16211963e+00 -1.12079026e+00 -2.37891086e-01 -6.49533928e-01]
[ 5.73008520e+00 -3.91909826e+00 -1.40877754e+00 -1.16665371e-01]
[-7.07990781e+00  3.03538574e+00  2.17472314e+00  1.50813455e+00]
[ 5.72945598e-01  1.60195733e-01  1.26990275e-01 -3.88540280e-01]
[ 1.61433115e+00 -1.16802409e+00 -5.45032811e-02 -1.30951368e-01]
[ 1.01770561e+00 -8.42436542e-01 -1.09071117e-01 -8.47245775e-02]
[ 6.11902235e+00 -3.40924523e+00 -3.07558021e-01 -2.21003266e+00]
[ 2.25918403e-01  2.80572680e-02  9.94398095e-02 -1.67772625e-01]
[ 4.53310318e+00  8.27066165e-01 -2.01759511e+00 -2.97448446e+00]
[-3.55069762e-01  5.41066175e+00  1.01551155e+00 -5.86668828e+00]
[ 1.01396046e+00 -9.28138089e-01  9.28464699e-01 -1.37946647e+00]
[ 5.62583910e+00 -4.54164739e+00 -1.54457782e+00  6.68143790e-01]
[ 1.48814795e+01 -9.67202605e+00 -2.25333992e+00 -2.58899240e+00]
[ 4.87012626e+00 -2.74375065e+00 -1.05129052e+00 -8.31496236e-01]
[-3.05029729e-01  2.09799156e+00  8.26969044e-02 -1.36032361e+00]
[-4.56916203e+00  1.13826753e-01  1.42472577e+00  2.78636632e+00]
[ 4.72772681e+00 -1.04003115e+00 -3.43843160e+00 -3.53077804e-01]
[ 7.25419317e+00 -1.66066180e+00  4.26736133e+00 -9.36194397e+00]
[ 1.78317563e+00 -3.98488552e-01  4.11224088e-01 -1.45794881e+00]
[ 8.45622499e+00 -5.15589951e+00 -1.36520067e+00 -2.07474470e+00]
[ 8.28214368e+00 -4.31253458e+00  1.24124983e+00 -5.48144116e+00]
[ 3.79058836e+00 -3.32151232e+00 -2.89614128e+00  2.79775482e+00]
[ 3.98565255e-01  3.41829112e+00 -8.22255317e-01 -3.45401881e+00]
```

[4.28406082e+00 1.09449010e+00 -8.98876970e-01 -4.81119174e+00]

[-6.83147970e-01 9.10061985e-02 3.47874569e-01 1.59902274e-01]

[8.29083230e+00 2.46851069e+00 -4.00684877e+00 -6.69123629e+00]

[-2.13365635e-01 -2.46069819e-02 1.92539630e-01 1.95642547e-01]

[-7.86641563e+00 6.49356749e+00 -7.53792346e-01 2.23689780e+00]

[-4.18369714e-02 1.86360620e-01 1.79952817e-01 2.15880954e-01]

[6.14896272e+00 -5.72780205e+00 -1.24009882e+00 3.25727920e-01]

[4.72996672e-01 -1.51835520e-01 -8.39351250e-02 -3.75424822e-02]

[1.27987894e+00 1.12118005e+01 -8.05099083e+00 -4.69617356e+00]

[2.18488585e+00 -9.86417498e-01 -3.10511383e-01 -1.15938727e+00]

[2.49281837e+00 8.97840624e-01 3.22817337e+00 -6.07132812e+00]

[2.48088974e-01 4.98965638e-02 3.51355647e-01 -2.52032278e-01]

[3.40233280e-01 -3.81082424e-01 -1.15845747e-01 -1.58700173e-01]

[2.20251738e+00 -1.29431140e+00 -3.37801678e-01 -4.07657591e-01]

[-2.90927062e-02 7.62909411e-02 8.40346900e-02 -4.21899027e-02]

[-5.30587962e+00 -1.35457019e+00 -1.26183992e+00 7.95755705e+00]

[4.56641043e-01 -7.58293790e+00 4.21015730e+00 2.46311689e+00]

[1.55213623e-02 4.05688798e-02 1.43447253e-01 1.41892472e-01]

[9.49611286e-02 2.76500989e+00 -9.87439822e-01 -1.52189170e+00]

[6.39823462e+00 -7.22582143e+00 -3.59153909e-01 1.48115689e+00]

[4.51938480e-01 -4.42325439e-01 -3.53862379e-02 1.06583806e-02]

[9.36387874e+00 -4.91554539e+00 -4.47285715e+00 -1.27840064e-01]

[-9.35515187e-01 -4.46699223e-01 -2.10213915e-01 1.59979927e+00]

[-1.05477265e+01 2.93489159e+00 3.50011225e+00 4.39095992e+00]

[2.46553657e+00 -4.19230990e-01 -8.53181342e-01 -9.75330944e-01]

[1.37230531e-01 -6.99543118e-02 -6.35254638e-02 9.56995904e-02]

[4.52379758e-01 8.33361496e-01 -1.14402764e-01 -1.15966805e+00]

[2.99353893e+00 -6.28959698e-01 -3.32519157e-01 -2.23029000e+00]

[7.00596276e+00 -5.25036144e+00 -5.21777011e-01 -1.46400675e+00]

```
[ 2.70362889e+00  1.30709167e+00 -1.71985059e+00 -2.24363325e+00]
[-5.22305041e+00  6.48491539e+00  6.22708252e+00 -7.80663119e+00]
[ 9.32013647e+00 -1.49554866e-01 -2.92240028e+00 -6.70739604e+00]
[ 5.98054590e+00 -2.53256377e+00  1.60695798e+00 -4.99169181e+00]
[ 5.89651082e+00 -2.13259196e+00  4.31248732e-01 -4.35795512e+00]
[ 4.48373397e+00 -1.85664860e+00 -6.93926483e-01 -2.28330510e+00]
[ 8.77381983e-01  1.31142787e+00 -1.08729459e+00 -1.60822058e+00]
[-8.48136612e-02  1.82187144e+00 -9.68834146e-01 -2.46990713e-01]
[-2.88630471e+00 -2.71112142e-01  8.79318823e-01  1.72898413e+00]
[-2.36725308e-02 -2.77146927e-01 -7.10330121e-02 -2.50949767e-02]
[ 1.83122793e+00 -8.79050085e-01 -1.57132298e-01 -1.19835228e+00]
[ 2.17716819e+00 -5.25435230e+00  1.74845833e+00  1.44547874e+00]
[-3.89740090e+00  5.10413751e+00 -5.71575336e-01 -8.07997754e-01]
[-4.14088137e-01  3.52182990e-01  2.40663307e-01  4.23206755e-03]
[ 4.17039431e-01 -1.27375975e+00  1.50939186e-01  7.73212294e-01]
[-8.33235196e-02  6.96715128e-02  1.55532496e-01 -3.21944965e-02]
[ 2.57217352e-01 -2.42523993e-01 -1.50292345e-01 -2.46406393e-01]
[ 7.25407182e+00 -3.89758139e+00  4.23432890e-01 -3.90015981e+00]
[ 4.48606871e+00 -2.42396143e+00 -1.09530475e-01 -2.28399900e+00]
[-2.23490237e+00 -8.17199600e-01  1.14856663e+00  1.89103863e+00]
[ 5.94559928e-01  6.31125899e+00 -3.92826489e+00 -2.84486272e+00]
[-1.85053552e+00  5.21100690e+00 -1.28448950e+00 -2.21359631e+00]
[ 7.54110888e+00 -2.60560654e+00 -1.58862658e+00 -3.72142524e+00]
[ 1.12186599e-01  1.25788583e-01  9.75183738e-02  1.08715057e-01]
[-3.67980293e-02  5.36760601e-02 -1.80217052e-01  1.52656604e-01]
[ 5.03723404e-01 -5.74762529e-01 -2.44460001e-01  5.64267312e-02]
[ 2.14710169e-01  9.77128087e-02 -1.35314456e-01 -1.34367382e-01]
[ 5.58404275e-01 -2.95607865e-01  1.37401825e-01 -2.99232911e-01]
[ 8.03259455e-03 -6.91541931e-02  1.41567242e-01  1.86105754e-01]
```

[8.54234037e+00 -1.53326009e+00 1.55077983e+00 -8.12500356e+00]

[3.15961945e-01 -4.46652826e-01 -3.97361185e-02 -2.27681481e-01]

[-2.04135583e+00 9.15927784e-01 1.21031175e+00 6.04713813e-01]

[-7.89855834e-01 7.97009484e+00 -3.57422136e+00 -3.59507306e+00]

[-2.12968534e+00 2.00929139e+00 -1.52731439e+00 1.40411765e+00]

[-1.78157637e-01 7.96556295e-03 2.33945550e-01 2.36149996e-01]

[-1.64390529e-01 -4.09546354e-02 8.95844405e-02 -6.88735539e-02]]

Final Weights:

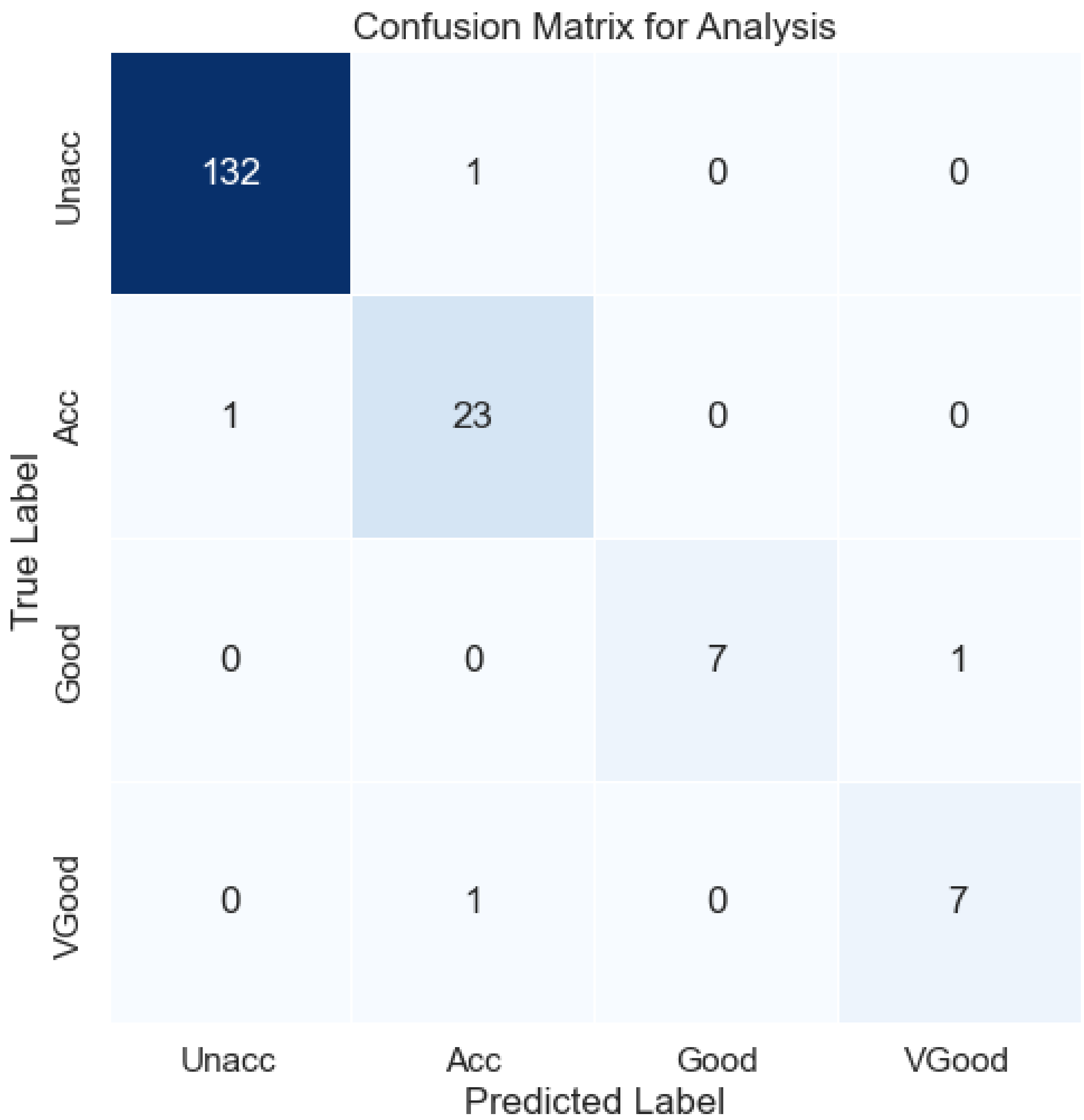
```
[  9.58959949   2.86165903   1.95981096 -14.61840811]
```

Actual and Predicted Values of Training Samples:

```
Actual Values
954     0
115     0
1422    0
35      0
1579    2
       ..
951     0
981     0
191     0
1636    2
778     0
Name: Decision, Length: 173, dtype: int64
Predicted Values
[0 0 0 0 2 0 1 1 0 1 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0
 0 0 1 0 0 0 3 0 1 0 0 0 0 0 1 2 3 2 0 1 0 1 0 0 0 0 3 0 0 1 1 0 0 3 0 0 0
 2 0 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
 0 0 0 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 3 0 0 1 0 1 3 0 0 0 0
 0 0 0 0 3 0 0 0 0 0 3 2 0 1 0 0 0 0 0 0 0 0 0 0 2 0]
```

```
Prediction of first ten elements :  [0, 0, 0, 0, 2, 0, 1, 1, 0, 1]
Actual Value of first ten elements:  [0, 0, 0, 0, 2, 0, 1, 1, 0, 1]
```

Confusion Matrix:



It shows us 4 were wrong predictions and rest all predictions were correct.

1 prediction was stated as Acc while it was Unacc

1 prediction was stated as Unacc while it was Acc

1 prediction was stated as Vgood while it was Acc

1 prediction was stated as Acc while it was VGood

Accuracy Analysis (Precision, Accuracy, Recall, Error):

```
Accuracy:  0.976878612716763
Precision: 0.977
Recall: 0.977
*************** Evaluation on Our Model ***************
Accuracy Score:  0.976878612716763
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       133
           1       0.92      0.96      0.94        24
           2       1.00      0.88      0.93         8
           3       0.88      0.88      0.88         8

    accuracy                           0.98       173
   macro avg       0.95      0.93      0.93       173
weighted avg       0.98      0.98      0.98       173

--------------------------------------------------------
```

Inference:

The confusion matrix tells us

- **4 were wrong predictions and rest all predictions were correct.**
- **1 prediction was stated as Acc while it was Unacc**
- **1 prediction was stated as Unacc while it was Acc**
- **1 prediction was stated as Vgood while it was Acc**
- **1 prediction was stated as Acc while it was VGood**

- **The accuracy of our model is 97.68 percent whereas precision is 97.7 percent and recall is also 97.7 percent.**