Assignment 1

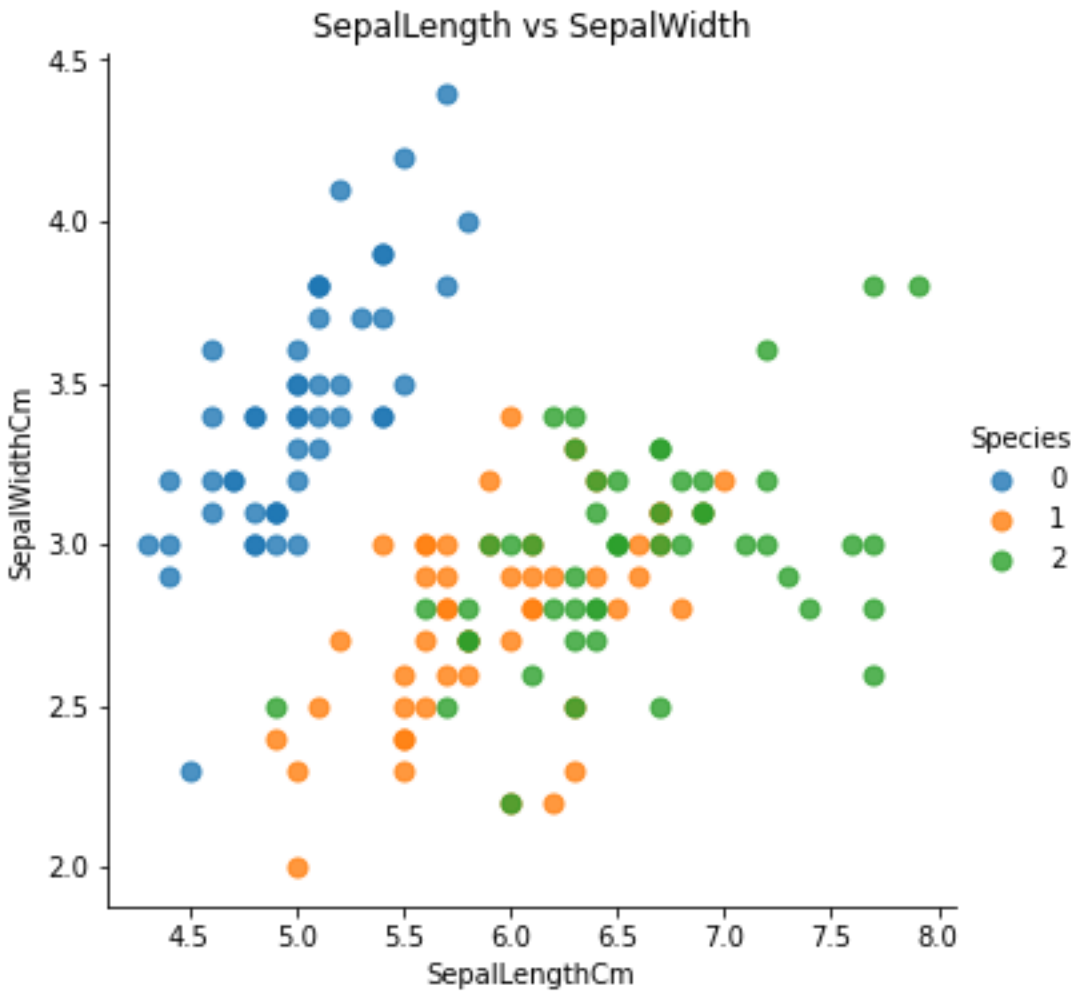
Subject: Artificial Intelligence

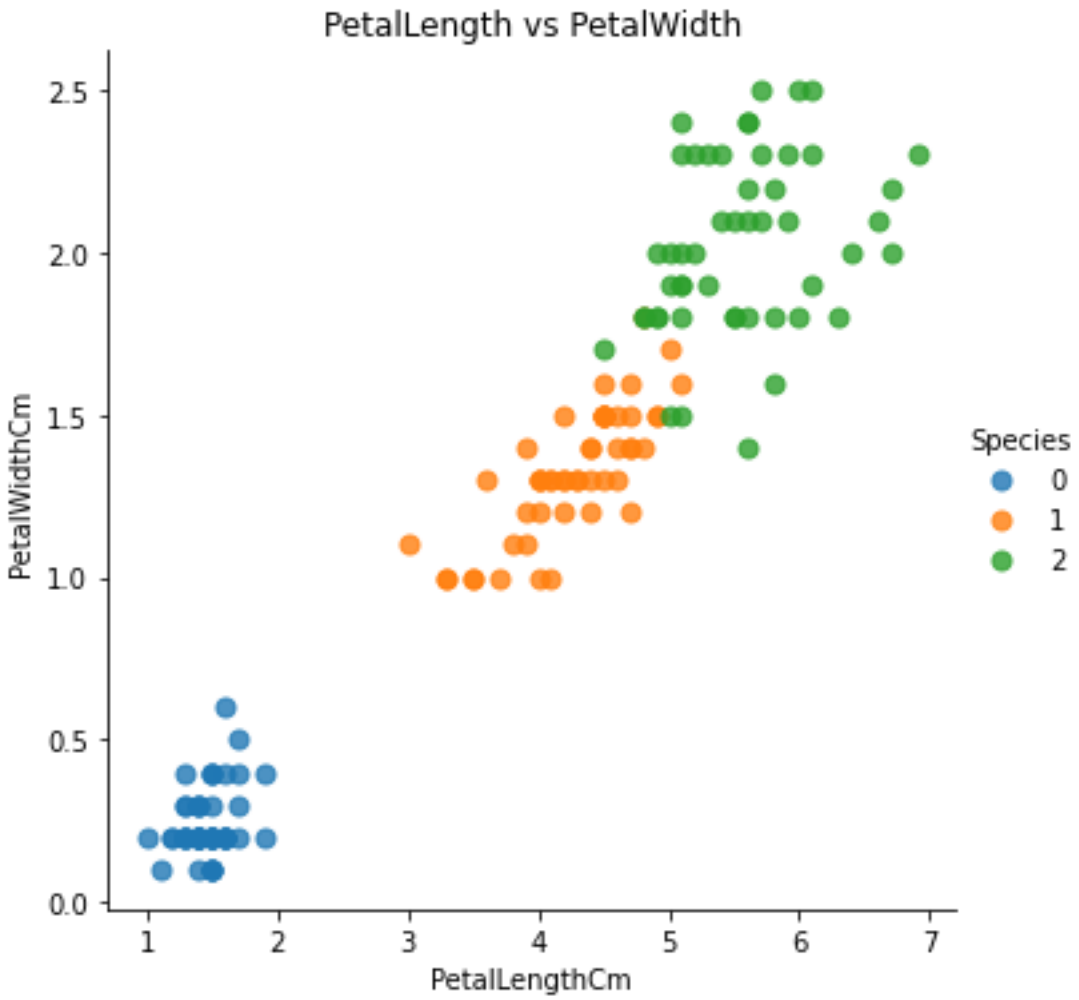
Name: Ishani Bhadoria

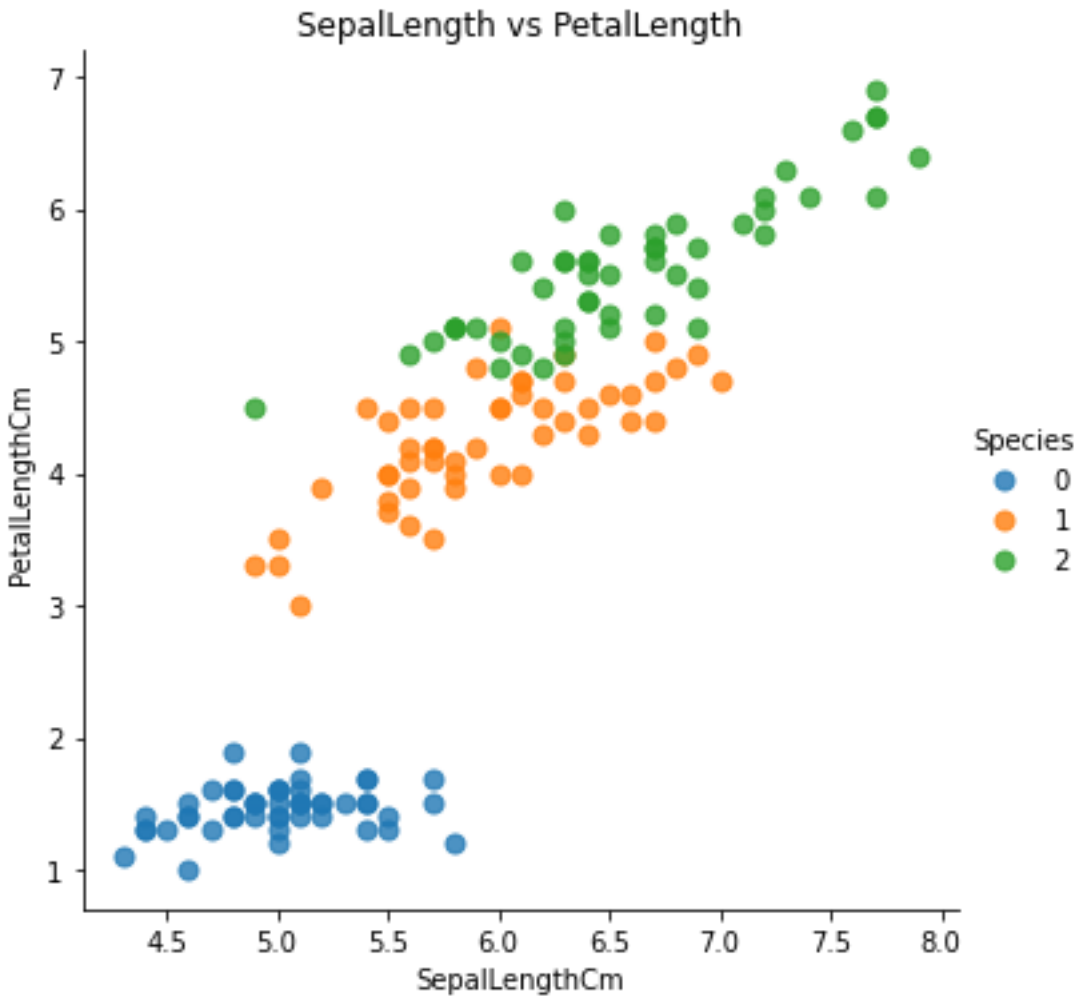
Student ID: 1113227

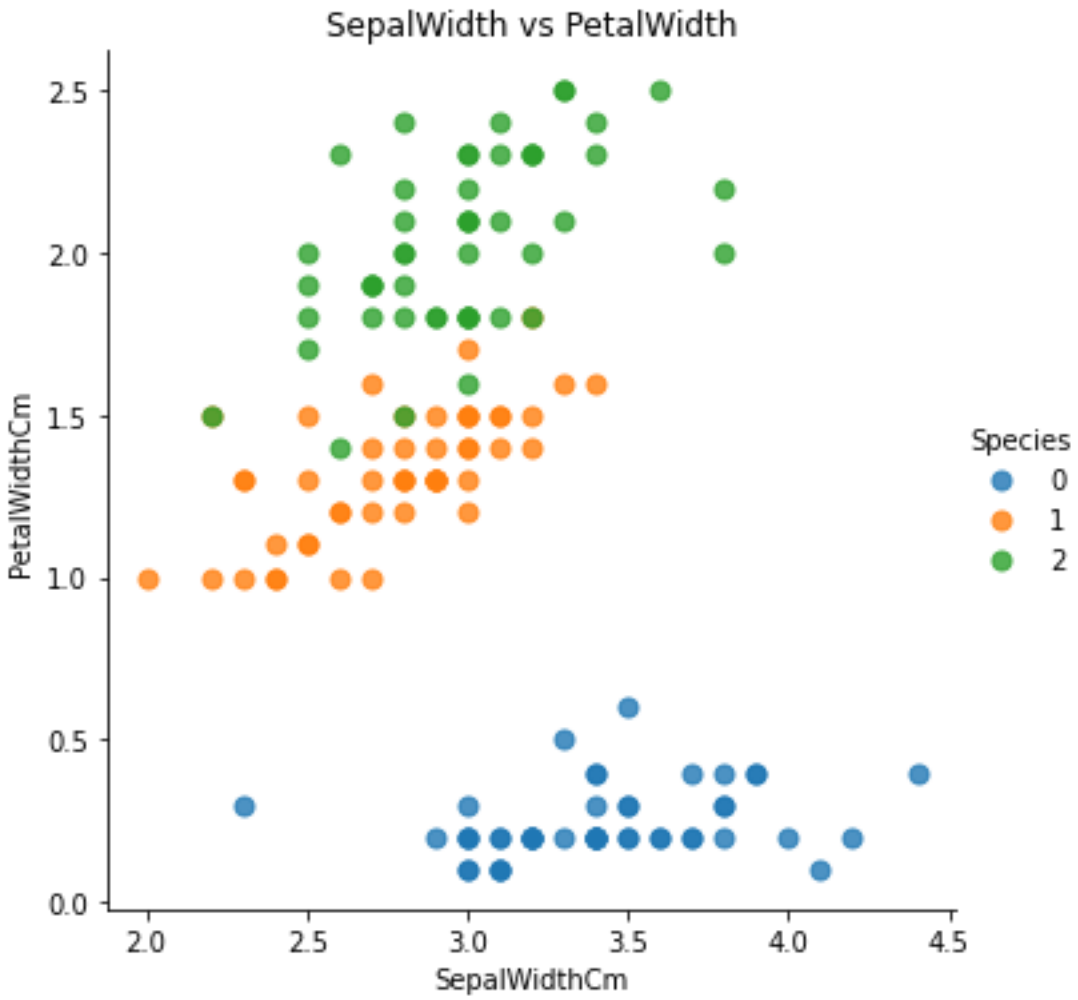
Code Explanation and screenshot:

1. The code is written in python language. The code is written in a jupyter notebook.
2. A neural network is a supervised learning algorithm, meaning that the input data containing the independent variables and the output data containing the dependent variable is given to it.
3. The nodes in the input layer are linked via three weight parameters to the output layer. In the output layer, the values are multiplied by their corresponding weights in the input nodes and are applied together. The bias word is eventually applied to the sum. The b refers to the bias term in the above figure.
4. Here, the bias word is very important. Suppose if we have a person who is not smoking, is not obese, and does not exercise, the sum of the input node and weight products would be zero. In that event, no matter how much we train the algorithms, the output will always be zero.
5. Therefore, even if we do not have any non-zero data about the guy, we need a bias term in order to be able to make predictions. To make a robust neural network, the bias term is important.
6. However, we have values in the shape of 0 to 2. in our production. We want our performance to be in the same format. We need an activation function to do so, which squashes input values from 2 to 0. The Sigmoid function is one such activation function.
7. There are some other ways of finding the loss, but we will use the function of the mean squared error cost. A function of cost is simply the function that finds the cost of the predictions given.
8. Graphs for the different values in species:


SepalLength vs SepalWidth
SepalWidthCm
SepalLengthCm
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Species
0
1
2


PetalLength vs PetalWidth
PetalWidthCm
PetalLengthCm
Species
0
1
2
0.0
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7


SepalLength vs PetalLength
PetalLengthCm
SepalLengthCm
Species
0
1
2
1
2
3
4
5
6
7
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

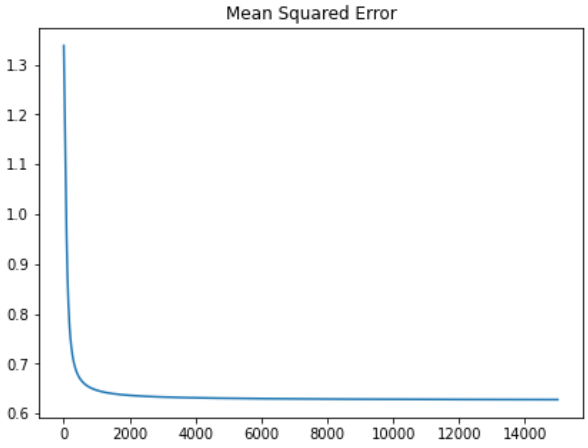


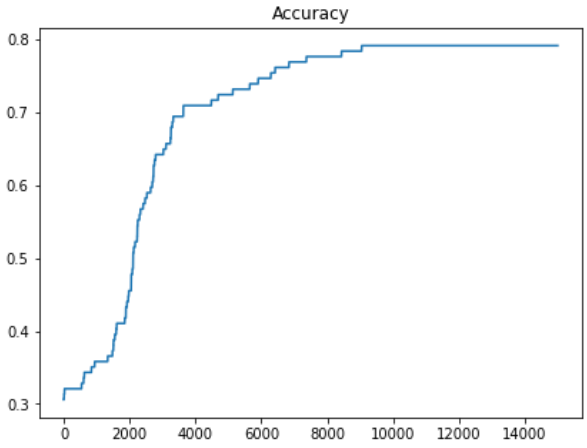
9. Output of the program gives accuracy:

```
accu = accuracy(output_layer_outputs, y_test) * 100
print("Accuracy: {}".format(accu))
```

```
Accuracy: 73.33333333333333
```

10. Graph for accuracy and mean square root:


Mean Squared Error
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0
2000
4000
6000
8000
10000
12000
14000


Accuracy
0.8
0.7
0.6
0.5
0.4
0.3
0
2000
4000
6000
8000
10000
12000
14000