Note: Graphs are included at the end

Machine Learning Hw 4

jl11895

April 2024

1 Question 1

Build and train a Perceptron (one input layer, one output layer, no hidden layers and no activation functions) to classify diabetes from the rest of the dataset. What is the AUC of this model?

1.1 Data Pre-processing

In this question, the diabetes is selected as the output classification and the rest of the features are selected as input. To improve data training speed and model performance, the non-dummy variables are normalized.

1.2 Model building and training

In my code, I built a function called Perceptron_model, which takes feature samples, target variable, random_seed and test_size as input. Within this function, the dataset is first splitted into training and testing set and then the model is built by Perceptron() function with tol = 0.001 and max_iteration 100. Then I fit the model with training dataset and compute the predicted value and the subsequent AUC score by using the test set. This function ultimately returns auc score of the model, false positive rate and true positive rate for the purpose of drawing ROC_AUC curve. Last but not least, I write a for loop to see how does different test size and random_seed change the auc score I achieved.

1.3 Why answer in this way

It is intuitive why I build a perceptron with no hidden layer and no activation function.It is simply asked by the question. The reason why I set the tol to 0.001 is that it could help the model not to waste much time on training. In other words, when the amount of change in weights of the model across the iteration is less than 0.001, I think it is fine to say that the result converges so it will not need to each the max_ite. I want to see the changes to the auc score caused by test size and random seed is because this dataset is actually small enough so that specific random_seed and test_size when splitting the dataset may matter.

1.4 Result, Graph and explanations

The AUC score for test size 0.1 and seed 42 is 0.46321601408786617
The AUC score for test size 0.1 and seed 100 is 0.7257852831537498
The AUC score for test size 0.1 and seed 500 is 0.7496376964411858
The AUC score for test size 0.1 and seed 1000 is 0.7505942798202493
The AUC score for test size 0.2 and seed 42 is 0.7454763946398784
The AUC score for test size 0.2 and seed 100 is 0.647096037116941
The AUC score for test size 0.2 and seed 500 is 0.6521600376321759
The AUC score for test size 0.2 and seed 1000 is 0.7395785654446878
The AUC score for test size 0.3 and seed 42 is 0.7752554131690021
The AUC score for test size 0.3 and seed 100 is 0.7547185260221259
The AUC score for test size 0.3 and seed 500 is 0.7997377404333548
The AUC score for test size 0.3 and seed 1000 is 0.7521528887504894
The AUC score for test size 0.4 and seed 42 is 0.6928643611206261
The AUC score for test size 0.4 and seed 100 is 0.761659106364154
The AUC score for test size 0.4 and seed 500 is 0.7753292459142587
The AUC score for test size 0.4 and seed 1000 is 0.6683063743504726
The AUC score for test size 0.5 and seed 42 is 0.47668695806851696
The AUC score for test size 0.5 and seed 100 is 0.7652929391122285
The AUC score for test size 0.5 and seed 500 is 0.6725239606153637
The AUC score for test size 0.5 and seed 1000 is 0.7649461284077217
As you can see, my concern is correct. The auc score of the perceptron model varies a lot when the test size and random seed are modified. The highest AUC score I can conclude is when test size is 0.3 and random seed 500. The corresponding graph is listed below. The result tells us that indeed when the data set is small, large test set will result in a smaller training set so that the prediction by the model is overly optimistic. Also, in a smaller dataset, random seed matters maybe because the training set may include or exclude some outliers or key examples. Once again, we could see the importance of k-fold validation.

2 Question 2

Build and train a feedforward neural network with at least one hidden layer to classify diabetes from the rest of the dataset. Make sure to try different numbers of hidden layers and different activation functions (at a minimum reLU and sigmoid). Doing so: How does AUC vary as a function of the number of hidden layers and is it dependent on the kind of activation function used (make sure to include "no activation function" in your comparison). How does this network perform relative to the Perceptron?

2.1 Data pre-processing

In this question, we are doing binary classification problem with fully connected neural network. I balanced the class in the data set by random under sampler, split the dataset into train and test and then assign the train and test loader. Also. I used pytorch and cuda to compute the result

2.2 Model building and training

The model is written in a function FCNN(nn.Module), and the traing and testing process are written in functions trian and test.
In the function FCNN, the parameters are input_size(number of features), hidden_size(an array of number of neurons in each hidden layer) , output_size and activation function. In the training process, I used BCEWithLogitsLoss() as the criterion, backpropogaiton to learn better parameters and return the loss
In the test process, the loss function is still BCEWithLogitsLoss and the average loss over the total number of samples and accuracy is returned.

2.3 Why answer in this way

The dataset is under-sampled because the class is imbalanced. I choose BCEWIthlogicloss over cross entropy because we are dealing with only binary classification and cross entropy loss is more suitable for multi-class classification problem. The optimizer I applied is SGD with learning rate 0.01 and l2 regularization to reduce the correlation between the features.

2.4 Data, Graph and Explanation

Now in this question, we are required to try different number of hidden layers and different activation functions. Here are the results:
The auc_score for fully connected neural network with hidden layer dimension [10] and none activation is 0.8182021790767063
The auc_score for fully connected neural network with hidden layer dimension [10, 10, 10] and none activation is 0.8186903885141429
The auc_score for fully connected neural network with hidden layer dimension [10, 10, 10, 10, 10] and none activation is 0.8198754797302787
The au_score for fully connected neural network with hidden layer dimension [10] and relu activation is 0.8214169157121418
The auc_score for fully connected neural network with hidden layers dimension [10, 10, 10] and relu activation is 0.8280197239934353
The auc_score for fully connected neural network with hidden layer dimension [10, 10, 10, 10, 10] and relu activation is 0.8283543681162989
The auc_score for fully connected neural network with hidden layer dimension [10] and sigmoid activation is 0.825830050089766
The auc_score for fully connected neural network with hidden layer dimension [10, 10, 10] and sigmoid activation is 0.8253236517317553
The auc_score for fully connected neural network with hidden layer dimension [10, 10, 10, 10, 10] and sigmoid activation is 0.6466288026299618
First of all, I set the number of neurons in the hidden layer to be 10 because it is the average of the input size(21 because we have 21 features) and the

output size (1). Increasing the number of neurons has the effect of improving the model but it may lead to overfitting; The over pattern is that for nearly all different activation functions, the auc improves, even slightly, as the number of hidden layers increase. However, based on my comparison, I have to say that maybe for certain activation functions like sigmoid, it is not the more hidden layers the better. The decrease in AUC when using sigmoid function maybe due to overfiting. So here is my conclusion, the number of hidden layers will improve the complexity of the model and the performance of the model but it may lead to problem of overfitting. I also notice that the auc score of both relu and sigmoid activation function is better than that of no activation function. I think this suggests that auc may be dependent on the present of activation function. However, I think to conclude that AUC is dependent on different types of activation function, there should be more strong differences in AUC score calculated by different activation. In my comparison, the difference is nearly negligible.
Last but not least, due to the innate property of random_state, it is hard to generate the exact answer every time. My answers keep oscillating between 0.82 to 0.83. I have to emphasize that the difference between the answers each time is nuanced;thus, even though my model is well-structured, the conclusion may not be affirmative.

3 Question 3

This question can be solved based on the structure I used in the second question. In this question, I built a "deeper" feedforward neural network with relu activation because from the previous question, I find that relu activation has a better performance when the hidden layer number increases. I increase the number of hidden layers to eight.
The auc_score for fully connected neural network with hidden layer dimension [10, 10, 10, 10, 10, 10, 10, 10] and relu activation is 0.8237637566968343
It is noticeable here that when the number of hidden layers increases, the auc score of this model is actually lower, I think this can be interpreted as the effect of over-fitting. More hidden layers does make the model in the training set perform better but it leads to overfitting when tested. Again, due to the nature of random_state and the small size of this dataset, there are a few times when the auc score turns out to be close to or higher than those found in Question 2. In this case, I think it is not a better choice to use CNN because there is no spatial feature in the input.

4 Question 4

4. Build and train a feedforward neural network with one hidden layer to predict BMI from the rest of the dataset. Use RMSE to assess the accuracy of your model. Does the RMSE depend on the activation function used?

4.1 Data pre-processing

In this question, the target is now BMI. I normalized the BMI by standardscaler and all other non-dummy variables. Then I split the into training and testing set and also use Dataloader to load them as train and test loader.

4.2 Model Building and training

The model is built in a function called FCBMI which has only one hidden layer. Also it is a feed forward neural network The training blueprint is written in trainBMI, the criterion is set to be MSEloss() and apply backpropagation to find better parameter. The testing blueprint is written in testBMI where rmse is also calculated here.

4.3 Why answer in this way

Instead of dealing with binary classification, we are dealing with regression prediction here where rmse is used to assess the accuracy of the model. Thus it is necessary to change the loss function to MSELoss(). Also, since this is a regression prediction, I added weight-decay, l-2 regularization, in the SGD optimizer to prevent the negative effect of correlated features.

4.4 Data, graph and Explanation

Here I tried to calculate the rmse using models of different activation function: relu, sigmoid, none.
The rmse of none with one hidden layer: 0.9266
The rmse of relu with one hidden layer: 0.9060
The rmse of sigmoid with one hidden layer: 0.9086
The rmse of relu activation function is the best. From the data I have, I think it is fair to say that the rmse is not dependent on the activation function used because the differences I can discern is too slight to conclude a dependent relation

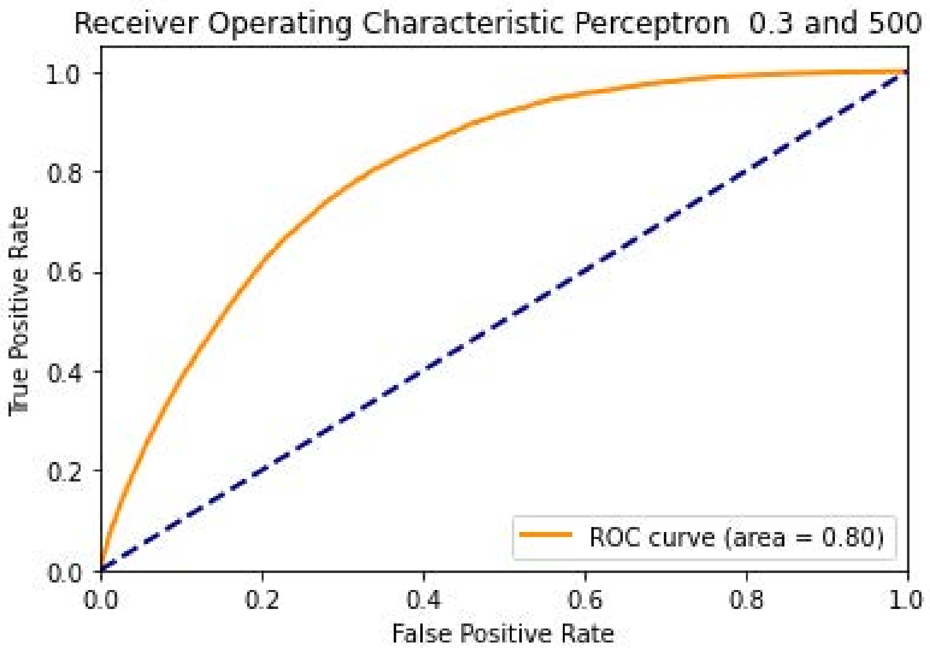
5 Question 5

The structure of the precious problem can be used here but the only difference is that I build the model in a new function called FCBModified. Based on what I had in question four, I find out that relu activation has the best performance, even slightly. So in this question, I choose relu activation to build my model and added more hidden layers to improve the performance. After doing so, I adjusted the learning rate and l2 lambda, which is used to prevent the impact of correlated features to the model, to decrease the rmse. Here is the lowest I can get. when learning rate is 0.01 and weight_decay is 0.001 and activation function is relu, I have the lowest rmse 0.9044.

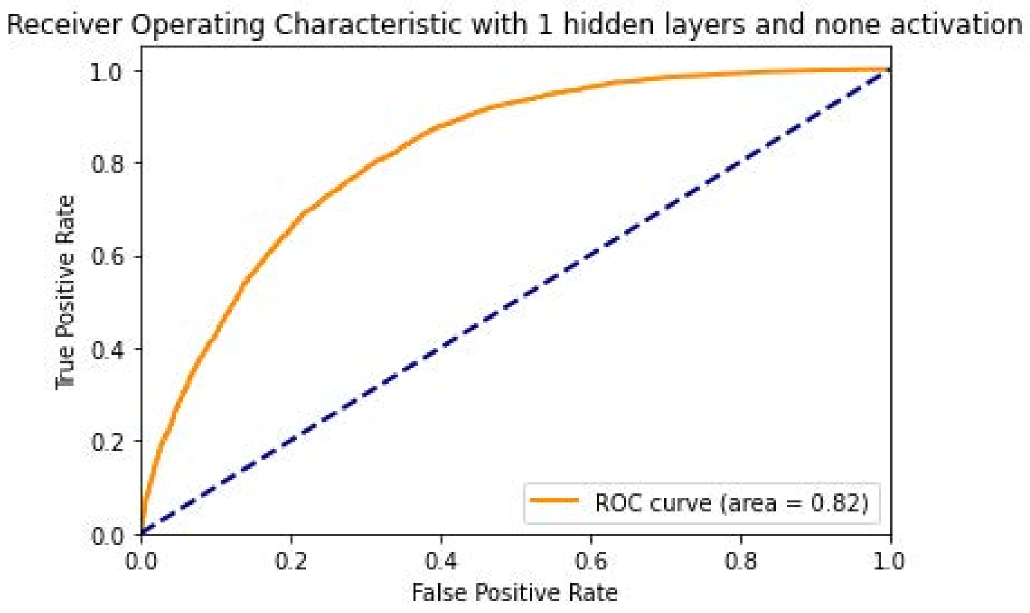
6 Extra Credit b

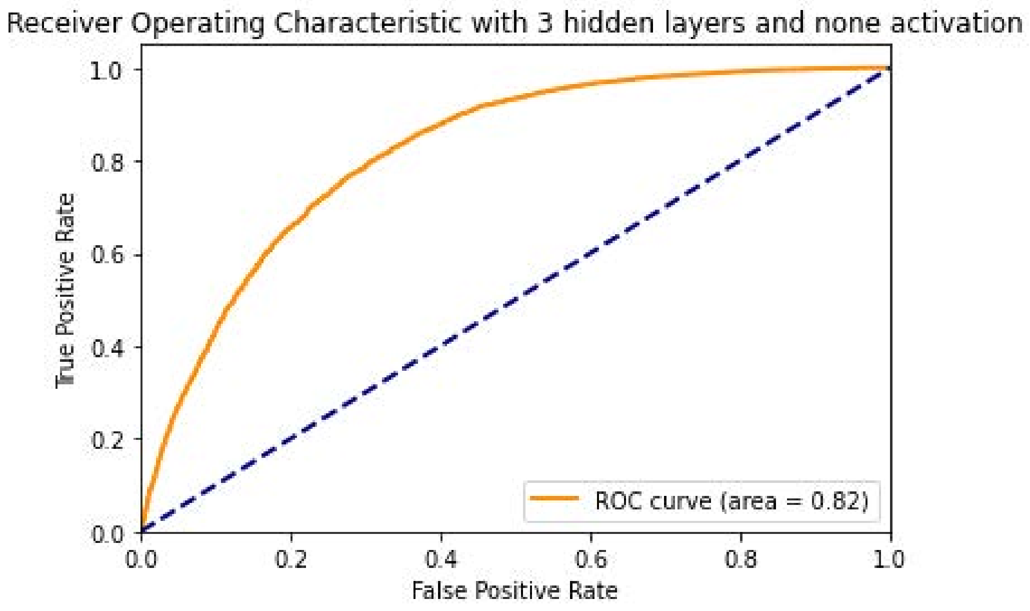
Pros:
The time needed to build a neural network is shorter compared to other methods, at least shorter than what we had in previous homeworks.
While most of the methods can only be used to focus on one purpose: either regression or classification, the neural network can be used to serve both of the purposes.
By adjusting the number of neuron and the number of hidden layers, it is a lot easier for neural network to have a more complex and comprehensive model.
Neural network can spot preserve the features in the input data.
Cons:
Using the neural network has a risk of overfiting when the dataset provided is small.
It is essentially a black box where the meaning between weights or neurons may not be interpreable. Also, it is often hard to decide to the number of neurons should be included in the hidden layers
The neural network is more computational expensive, which requires a GPU.
Overall: when choosing the model, it is important to examine the dataset, purpose or goal of the question and the computational ability.

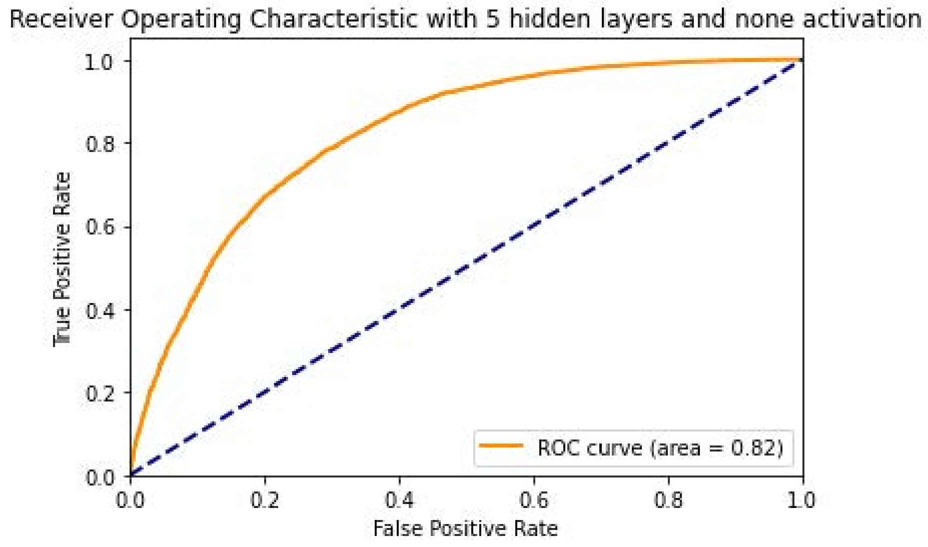
Question 1 Graph:

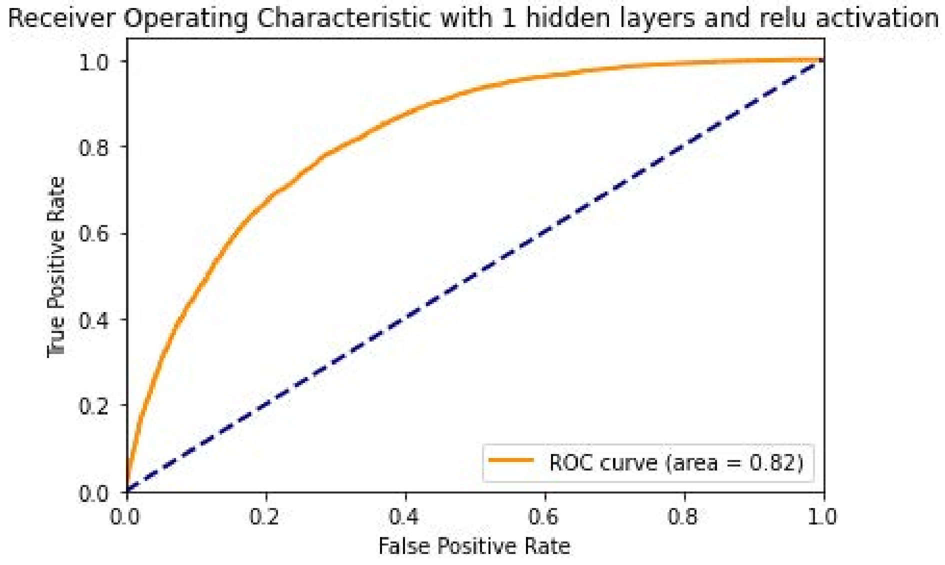


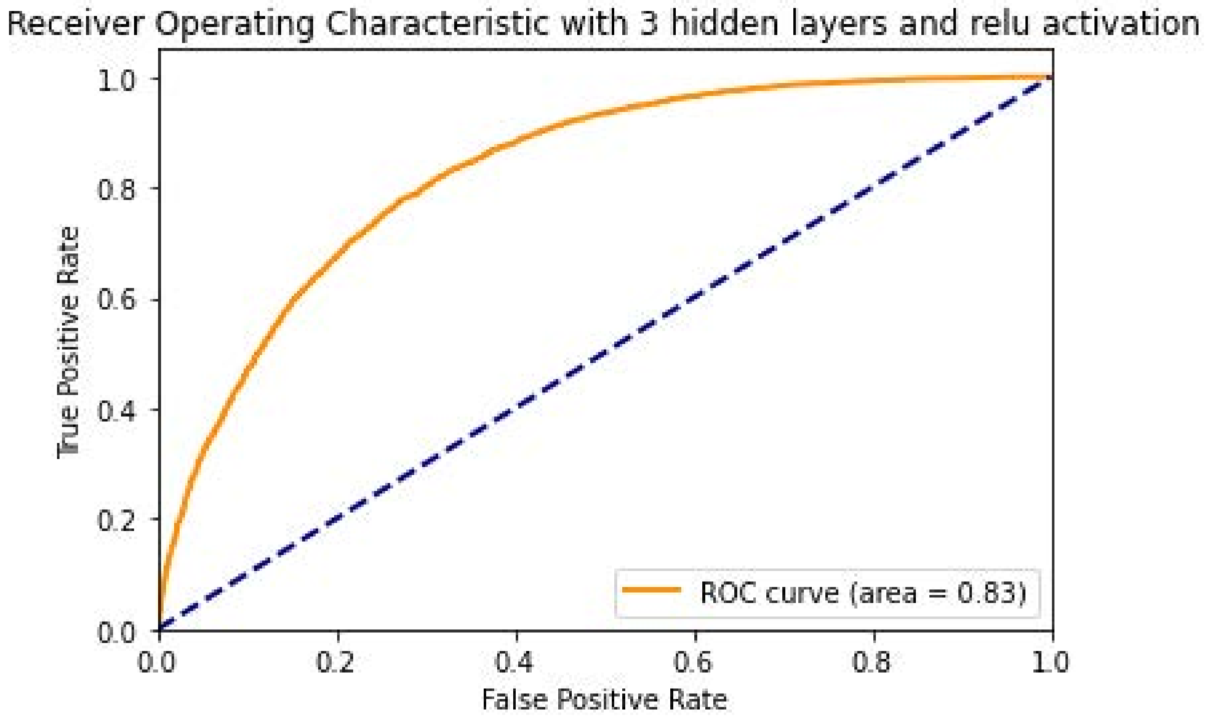
Question 2 Graphs:



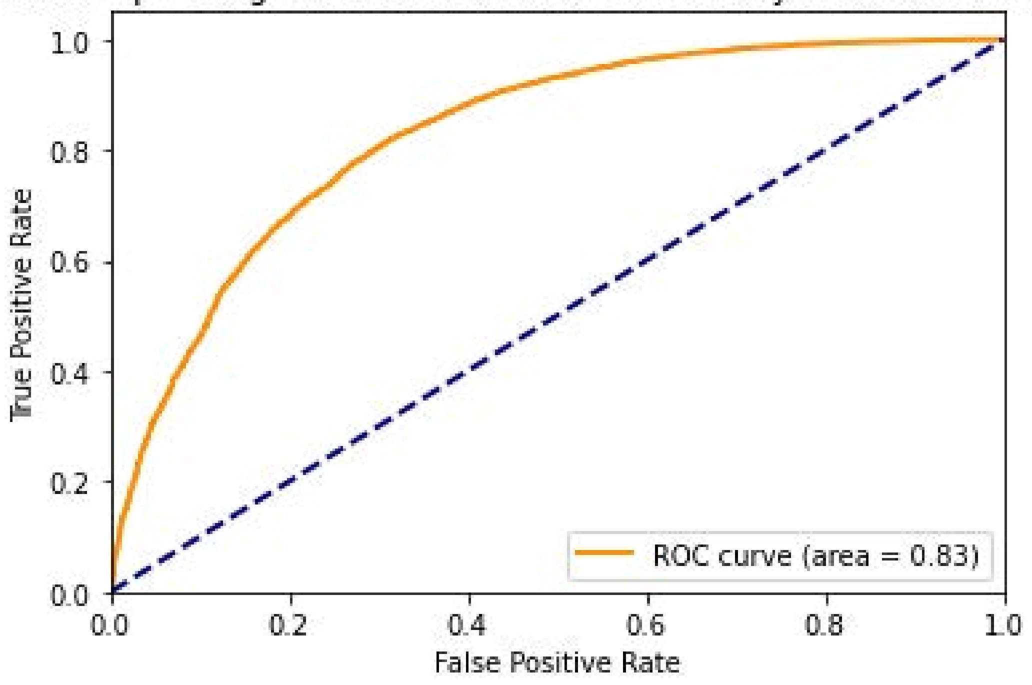



Receiver Operating Characteristic with 5 hidden layers and none activation
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.82)

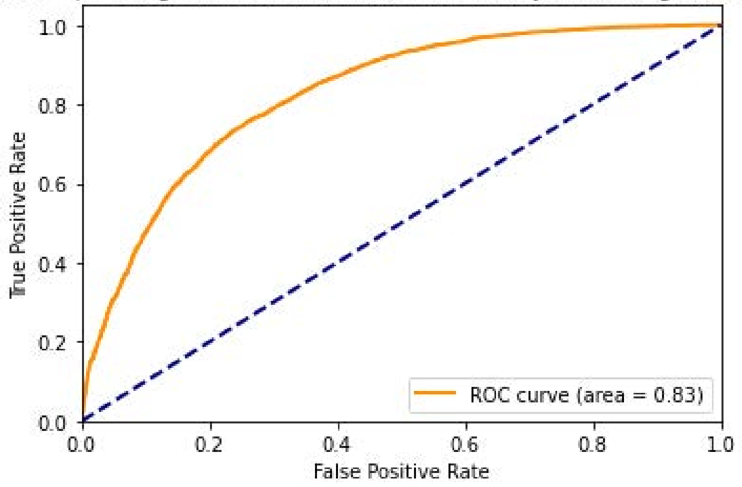

Receiver Operating Characteristic with 1 hidden layers and relu activation
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.82)


Receiver Operating Characteristic with 3 hidden layers and relu activation
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.83)

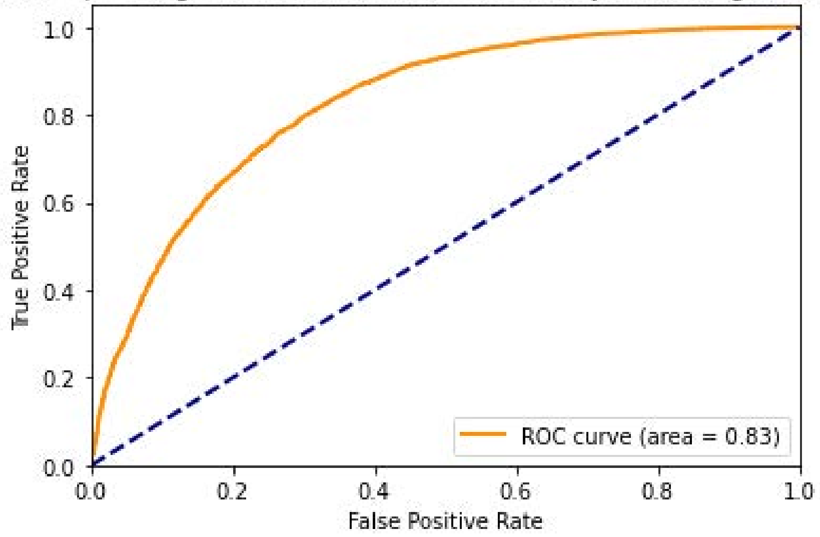
Receiver Operating Characteristic with 5 hidden layers and relu activation

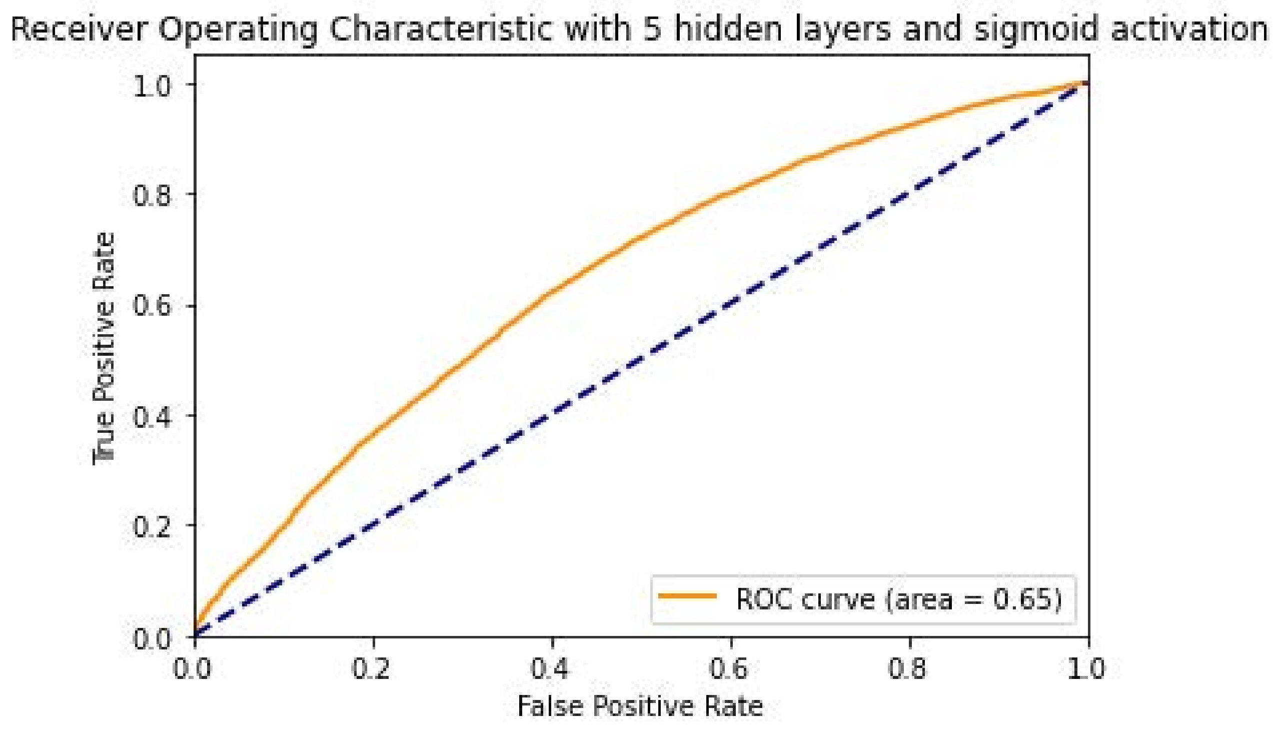


Receiver Operating Characteristic with 1 hidden layers and sigmoid activation

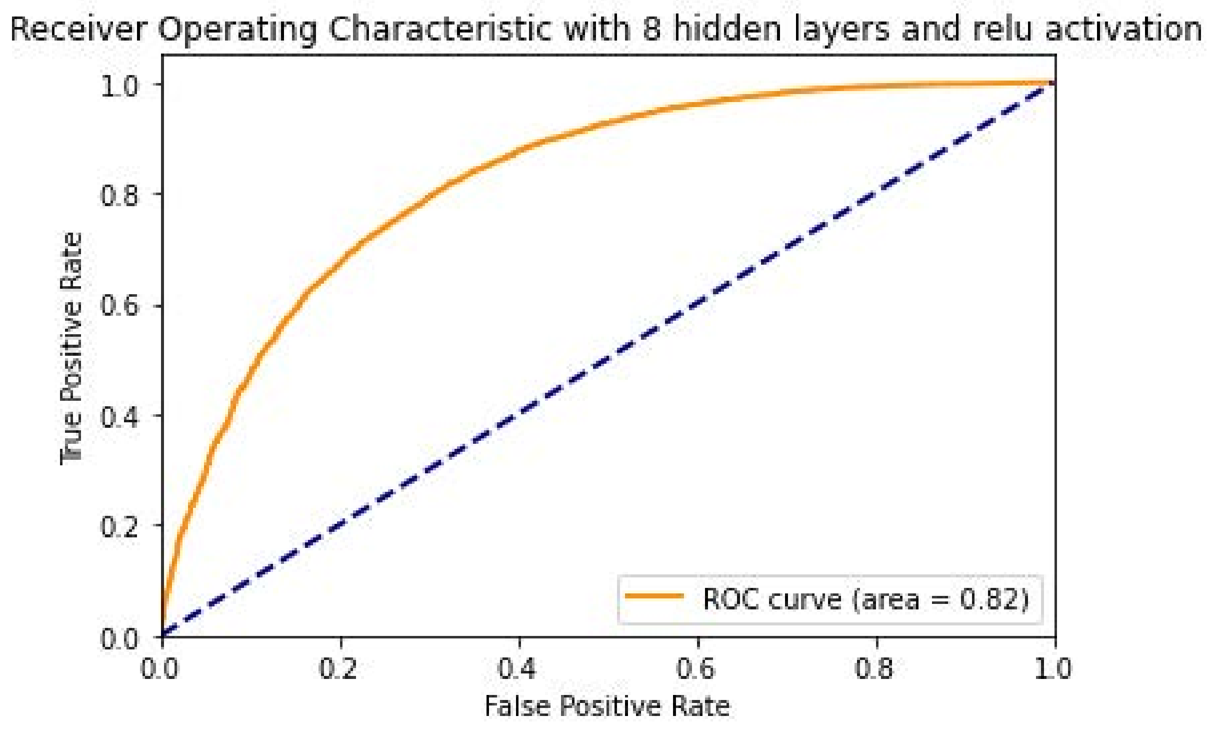


Receiver Operating Characteristic with 3 hidden layers and sigmoid activation

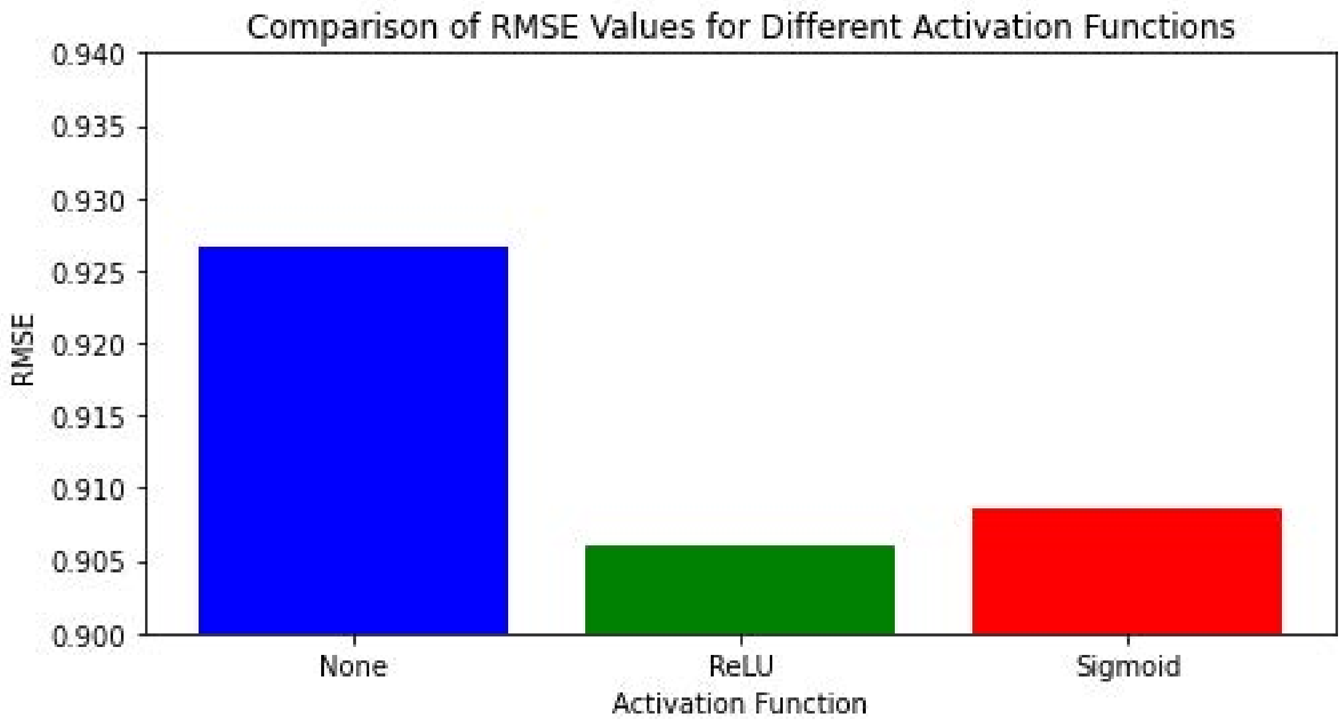




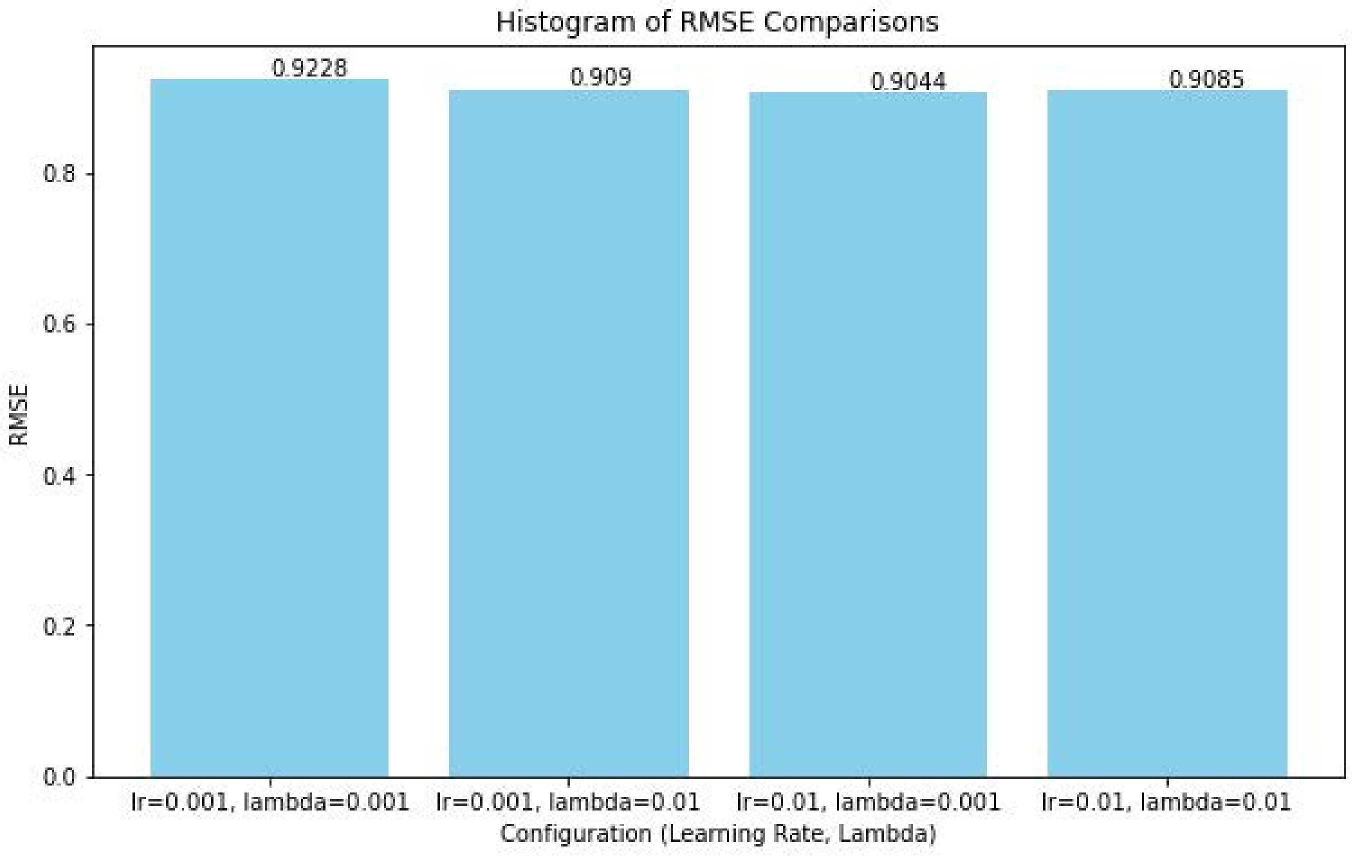
Question 3 graph



Question 4 graph



Question 5 graph


Histogram of RMSE Comparisons
RMSE
lr=0.001, lambda=0.001
lr=0.001, lambda=0.01
lr=0.01, lambda=0.001
lr=0.01, lambda=0.01
Configuration (Learning Rate, Lambda)