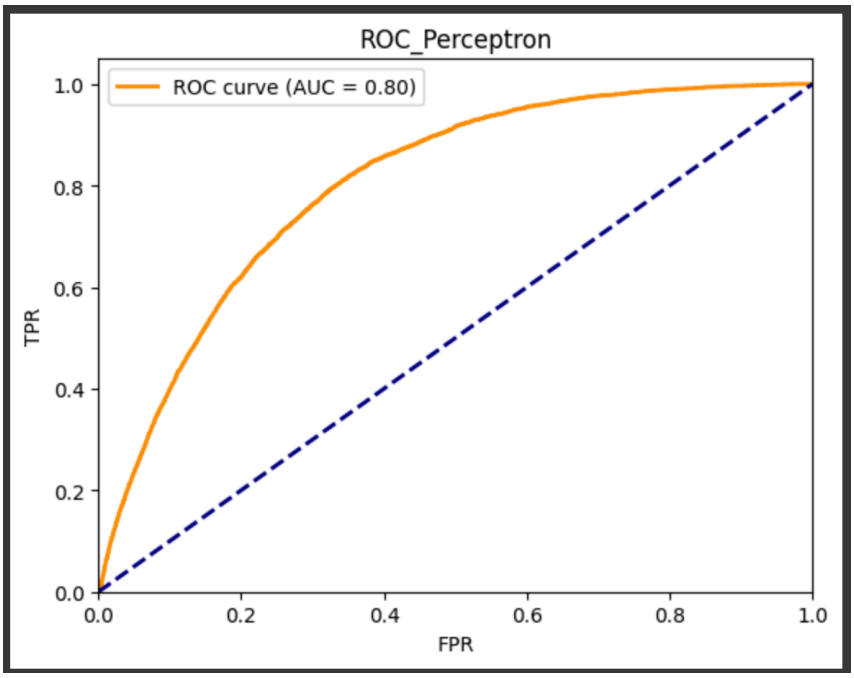
Data Preprocessing

It is stated that the data is carefully curated which indicates that there should not be too much missing data. Conducting exploratory analysis, it is suggested that the dataset is imbalanced as the number of individuals that have diabetes is very much lower than the number of individuals that do not have diabetes. This indicates that I should be including a class_weight parameter when running the models. I drop the Diabetes column as the outcome variable and use the rest of the dataframe as the predictor variables. Then I do a train_test_split with a 20% ratio on test data.

Question 1

Using the data after the train_test_split, I build a perceptron model (one input layer, one output layer, no hidden layers and no activation functions) with the Perceptron function on parameter: class_weight = balanced. Due to the reason of imbalanced distribution of dataset on diabetic individuals and non-diabetic individuals, setting class_weight = balanced enable the model to adjust the weights of the training samples so that each class is given equal weight during training. After fitting the model with X_train and y_train, I ran the performance metrics including the AUC score, F1-score, accuracy score, precision and recall. It indicated that the AUC score of the perceptron model is 0.80177, F1-score is 0.18918, Accuracy is 0.22150, Precision is 0.87737 and Recall is 0.22150. Below is the ROC plot for the perceptron model:

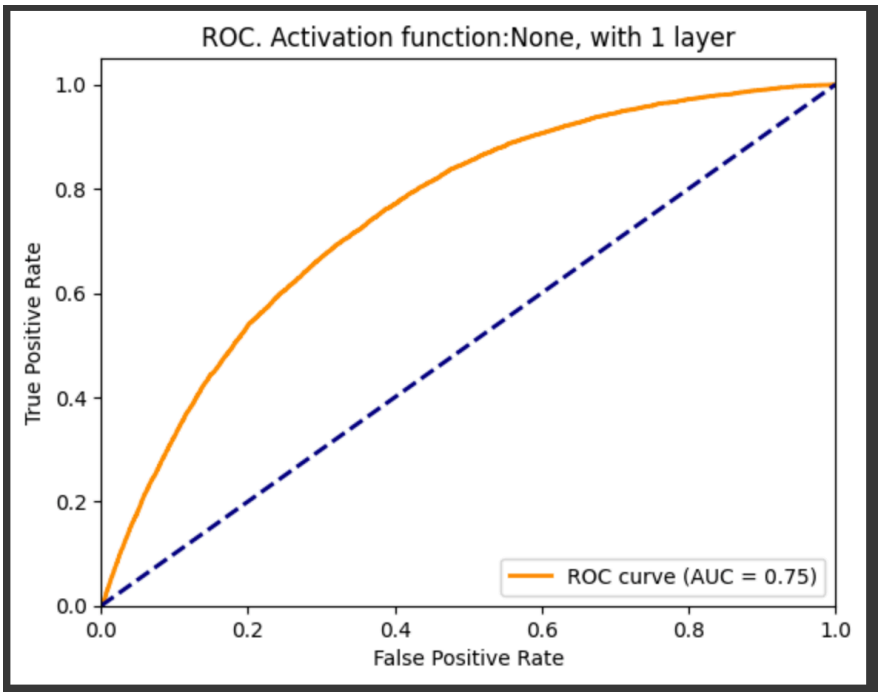


Question 2

To approach this question, it is essential to normalize the X_train and X_test data using standardization technique. The neural network is implemented using PyTorch, the self-defined classifier class defined the architecture with an input layer, number of hidden layer of choice, and an output layer. Activation function can be chosen from ReLU, Sigmoid, tanh or none. The train_model function trained the neural network using binary cross-entropy loss and Adam optimizer whereas the evaluate_model function evaluates the trained neural network using the roc_auc_score function to obtain the AUC score of the overall performance of the model. The question indicated that we should build and train a feedforward neural network with one and above hidden layer, hence the choice of number of hidden layers is an increment of 1 (1, 2, 3 and 4 layers). Then the performance is compared with the perceptron model's performance. The result from the different number of hidden layers and different activation functions is as below:

```
Activation: relu, Hidden Layers: 1, AUC: 0.6211388674990173
Activation: relu, Hidden Layers: 2, AUC: 0.647080797460914
Activation: relu, Hidden Layers: 3, AUC: 0.6153957137937602
Activation: relu, Hidden Layers: 4, AUC: 0.564960613564325
Activation: sigmoid, Hidden Layers: 1, AUC: 0.5827985536863769
Activation: sigmoid, Hidden Layers: 2, AUC: 0.6715565142293005
Activation: sigmoid, Hidden Layers: 3, AUC: 0.6674104087937561
Activation: sigmoid, Hidden Layers: 4, AUC: 0.6927891721896027
Activation: tanh, Hidden Layers: 1, AUC: 0.6314446089866101
Activation: tanh, Hidden Layers: 2, AUC: 0.6063982838578613
Activation: tanh, Hidden Layers: 3, AUC: 0.7007176124705822
Activation: tanh, Hidden Layers: 4, AUC: 0.4817236916306946
Activation: None, Hidden Layers: 1, AUC: 0.7892238345115118
Activation: None, Hidden Layers: 2, AUC: 0.7747187105494023
Activation: None, Hidden Layers: 3, AUC: 0.7721485778299757
Activation: None, Hidden Layers: 4, AUC: 0.7784884327379573
```

When the activation function is ReLU, AUC score increased from 0.62114 (1 hidden layer) to 0.64708 (2 hidden layers) but decreased from having 2 hidden layers to 4 hidden layers: 0.61540 (3 hidden layers) and 0.56496 (4 hidden layers). When the activation function is sigmoid, AUC score increased from 0.58280 (1 hidden layer) to 0.67156 (2 hidden layers) but decreased to 0.66741 (3 hidden layers) and increased to 0.69279 (4 hidden layers). When the activation function is tanh, AUC score decreased from 0.63144 (1 hidden layer) to 0.60640 (2 hidden layers) but increased to 0.70072 (3 hidden layers) and decreased to 0.48172 (4 hidden layers). When there are no activation function taken into account, AUC score decreased from 0.78922 (1 hidden layer) to 0.77472 (2 hidden layers) to 0.77215 (3 hidden layers) then increased to 0.77849 (4 hidden layers). The best model among all combinations is the feedforward network with 1 hidden layer and no activation function having the highest AUC score of 0.78922. Below is the ROC plot for the model:



Comparing the performances of the feedforward neural network and the perceptron model, it is to conclude that the perceptron model have a better performance. This is possibly due to the simplicity of the perceptron that are less complex than neural networks with multiple hidden layers.

Question 3

Using the normalized X_train and X_test alongside with the self_defined classifier class and the functions from Question 2, I build a deep neural network with two and above hidden layers to classify diabetes from the rest of the dataset. Activation function can be chosen from ReLU, Sigmoid, tanh or none. The train_model function trained the neural network using binary cross-entropy loss and Adam optimizer whereas the evaluate_model function evaluates the trained neural network using the roc_auc_score function to obtain the AUC score of the overall performance of the model. The differences compared to Question 2 is the choice of number of hidden layers is an increment of 1 starting from 2 (2, 3, 4 and 5 layers). The result from the different number of hidden layers and different activation functions is as below:

```
Activation: relu, Hidden Layers: 2, AUC: 0.6488944762813644
Activation: relu, Hidden Layers: 3, AUC: 0.679442515860653
Activation: relu, Hidden Layers: 4, AUC: 0.7125610279822479
Activation: relu, Hidden Layers: 5, AUC: 0.6279003739825944
Activation: sigmoid, Hidden Layers: 2, AUC: 0.6642504889601953
Activation: sigmoid, Hidden Layers: 3, AUC: 0.6508742958146992
Activation: sigmoid, Hidden Layers: 4, AUC: 0.691677990583397
Activation: sigmoid, Hidden Layers: 5, AUC: 0.6680754176628227
Activation: tanh, Hidden Layers: 2, AUC: 0.6474619627346766
Activation: tanh, Hidden Layers: 3, AUC: 0.4858229047763716
Activation: tanh, Hidden Layers: 4, AUC: 0.5226778805559371
Activation: tanh, Hidden Layers: 5, AUC: 0.5833645907755021
Activation: None, Hidden Layers: 2, AUC: 0.7792615673657868
Activation: None, Hidden Layers: 3, AUC: 0.7723593807450795
Activation: None, Hidden Layers: 4, AUC: 0.7722936152152793
Activation: None, Hidden Layers: 5, AUC: 0.7785574559928636
```

Comparing for 2 to 5 hidden layers model, the highest AUC scores are observed for the models without activation functions and slightly behind no activation function is using sigmoid as the activation function. The best combination will be to have 2 hidden layers with no activation function. Given the nature of the dataset which is consist of many tabular data such as age, BMI and blood pressure, indicating there are no benefit of using CNN or RNN for classification. This is due to the reason that CNN are typically used for image classifications tasks whereas the RNN are typically used for sequence classification tasks making both not a suitable replacement for the deep neural network trained in the question.

Question 4

Dropping the BMI column as the outcome variable and the rest of the dataframe as predictor variables, I do a new train_test_split with a 20% ratio of the test data. Then I normalize the X_trainBMI and X_testBMI by standardizing them. The neural network is implemented using PyTorch, the self-defined BMI class defined the architecture with an input layer, number of hidden layer of choice, and an output layer. Activation function can be chosen from ReLU, Sigmoid, tanh or none. The train_model function trained the neural network using mean squared error loss and Adam optimizer whereas the evaluate_model function evaluates the trained neural network using the mean_squared_error function to obtain the RMSE of the overall performance of the model. The question require me to build and train a feedforward neural network with one hidden layer to predict BMI, hence the comparison be dependent of the type of activation functions. The result is as below:

```
Activation: relu, Hidden Layers: 1, RMSE: 13.4679777713718
Activation: sigmoid, Hidden Layers: 1, RMSE: 7.535490896915587
Activation: tanh, Hidden Layers: 1, RMSE: 7.486776592693549
Activation: None, Hidden Layers: 1, RMSE: 13.37983844207177
```

Comparing between different activation functions, the model with the tanh activation function has a better performance with rmse of 7.48678. Since RMSE is a measure of the difference between the predicted values of a model and the actual values. Lower RMSE indicates that the model is more accurate in predicting BMI.

Question 5

Using X_trainBMI and X_test BMI alongside with the self-defined BMI class and functions from Question 4, I build a neural network with one and above hidden layers to classify diabetes from the rest of the dataset. This enables me to determine which number of hidden layer and activation function combination will produce the best prediction model.

```
Activation: relu, Hidden Layers: 1, RMSE: 13.16200640692193
Activation: relu, Hidden Layers: 2, RMSE: 7.35417443051858
Activation: relu, Hidden Layers: 3, RMSE: 7.821253526896446
Activation: relu, Hidden Layers: 4, RMSE: 7.141162030948596
Activation: sigmoid, Hidden Layers: 1, RMSE: 7.247512474264877
Activation: sigmoid, Hidden Layers: 2, RMSE: 7.554297784919039
Activation: sigmoid, Hidden Layers: 3, RMSE: 7.878619098800868
Activation: sigmoid, Hidden Layers: 4, RMSE: 7.374132986919758
Activation: tanh, Hidden Layers: 1, RMSE: 26.439795427096897
Activation: tanh, Hidden Layers: 2, RMSE: 7.248416882472646
Activation: tanh, Hidden Layers: 3, RMSE: 6.764501157344723
Activation: tanh, Hidden Layers: 4, RMSE: 6.881741698704751
Activation: None, Hidden Layers: 1, RMSE: 15.009102433690394
Activation: None, Hidden Layers: 2, RMSE: 14.236713922462695
Activation: None, Hidden Layers: 3, RMSE: 11.023021654029108
Activation: None, Hidden Layers: 4, RMSE: 11.080919956365568
```

Observing from the results obtained, the best combination would be a deep neural network with 3 hidden layers and activation function of tanh. This is due to the reason that RMSE is a measure of the difference between the predicted values of a model and the actual values. Lower RMSE indicates that the model is more accurate in predicting BMI and the combination of 3 hidden layers and activation function of tanh is giving the lowest RMSE score among the other combination which is 6.76450. Optimal RMSE value for a given dataset may depend on several factors, including the complexity of the problem, the quality of the data, and the specific task at hand.

Extra Credit 1

To determine if there are any predictor variables that have effectively no impact on the accuracy on these models, I run the permutation_importance in the perceptron model to determine the unimportant features from all predictor variables on predicting diabetes using permutation_importance. Instead of running permutation_importance on every model, I run it only on the perceptron model to reduce the computational time. Using permutation_importance allow me to get a more robust, unbiased and model agnostic method for identifying the most important features in a model. The results obtained indicates that based on F1-score, accuracy and precision score, “BiologicalSex”, “IncomeBracket”, “Zodiac”, “Smoker”, “MentalHealth” and “PhysicalHealth” are the most unimportant feature in terms of diabetes prediction. This may be due to the reason that “BiologicalSex”, “IncomeBracket”, “Smoker”, “MentalHealth” and “PhysicalHealth” are all features that have limited effect on health condition whereas “Zodiac” is highly uncorrelated with any of the health conditions.

Extra Credit 2

The overall pros of using neural network to learn from the same dataset as in the prior homework relative to using classical methods like logistic regression and SVM is that neural network has better ability to learn complex, non-linear relationships between the predictor variables and the outcome variable while also having the ability to automatically extract relevant features from the dataset. However, the overall cons of using neural network to learn from the same dataset is that neural networks are computationally expensive and require large amount of training data to avoid overfitting. Neural network can also be difficult to interpret and provide limited insight into how they end up in their predictions (Black Box).