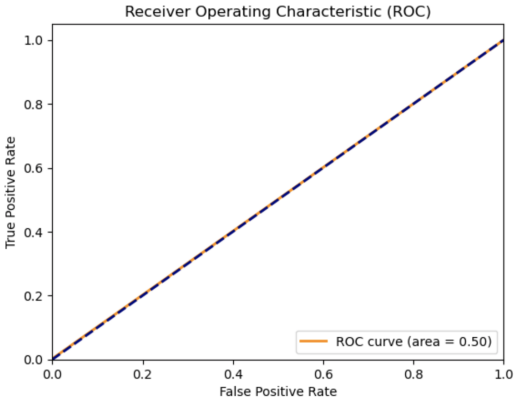
Question 1. Build and train a Perceptron (one input layer, one output layer, no hidden layers, and no activation functions) to classify diabetes from the rest of the dataset. What is the AUC of this model?

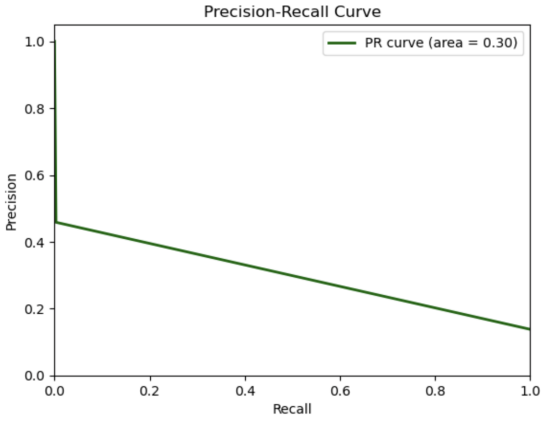
What was done: A Perceptron model was built and trained to classify diabetes using the given dataset. The data was first loaded and split into training and testing sets. Then, the Perceptron model was created, and hyperparameters were tuned using GridSearchCV. The model was trained with the best hyperparameters found by GridSearchCV.

Why this was done: A Perceptron is a simple neural network that can be used as a baseline for classification tasks. It was chosen because it is a single-layer model, making it easy to implement and understand. GridSearchCV was used for hyperparameter tuning to find the optimal learning rate and the maximum number of iterations, which can improve model performance.

What was found: The best hyperparameters found by GridSearchCV were eta0=0.01 and max_iter=500. Using these hyperparameters, the Perceptron model achieved an AUC of 0.5013 on the testing dataset. The confusion matrix showed 43,713 true negatives, 26 false positives, 6,975 false negatives, and 22 true positives. The ROC curve had an area of 0.50, and the precision-recall curve had an area of 0.30.

What the findings mean: The AUC of 0.5013 indicates that the Perceptron model has very limited predictive power for classifying diabetes in this dataset. This performance is only slightly better than random guessing. There may be more complex patterns in the data that the single-layer Perceptron is unable to capture. To improve performance, more sophisticated models like feedforward neural networks with hidden layers and different activation functions can be explored.





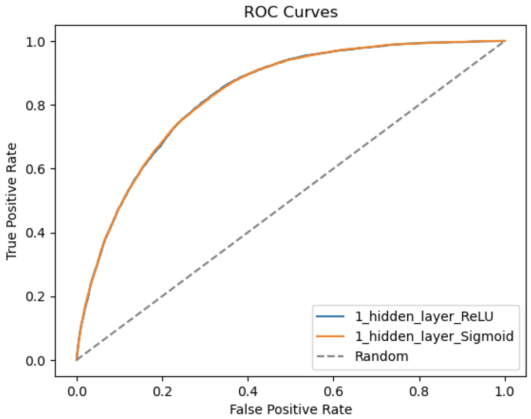
Question 2. Build and train a feedforward neural network with at least one hidden layer to classify diabetes from the rest of the dataset. Make sure to try different numbers of hidden layers and different activation functions (at a minimum reLU and sigmoid). Doing so: How does AUC vary as a function 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?

What was done: A feedforward neural network with one hidden layer was built and trained to classify diabetes using the given dataset. The network was trained with ReLU and sigmoid activation functions in the hidden layer. PyTorch was used for model implementation and training, utilizing GPU acceleration. The AUC scores were calculated for each model configuration.

Why this was done: Feedforward neural networks with hidden layers are capable of capturing more complex patterns in the data than single-layer Perceptrons. This makes them a suitable choice for improving classification performance. ReLU and sigmoid activation functions were chosen because they are commonly used in neural networks and have different properties that could affect model performance.

What was found: The feedforward neural network with one hidden layer achieved an AUC of 0.8328 with ReLU activation and an AUC of 0.8328 with sigmoid activation. Both models showed a significant improvement in AUC compared to the Perceptron (AUC: 0.5013).

What the findings mean: The feedforward neural network with one hidden layer performed significantly better than the Perceptron in classifying diabetes from the rest of the dataset. The AUC scores of the models with ReLU and sigmoid activation functions were very similar, indicating that the choice of activation function did not have a substantial impact on the performance in this case. This suggests that the addition of a hidden layer in the network has a more significant impact on the model's ability to capture complex patterns in the data than the choice of the activation function.



Question 3. Build and train a “deep” network (at least 2 hidden layers) to classify diabetes from the rest of the dataset. Given the nature of this dataset, is there a benefit of using a CNN or RNN for the classification?

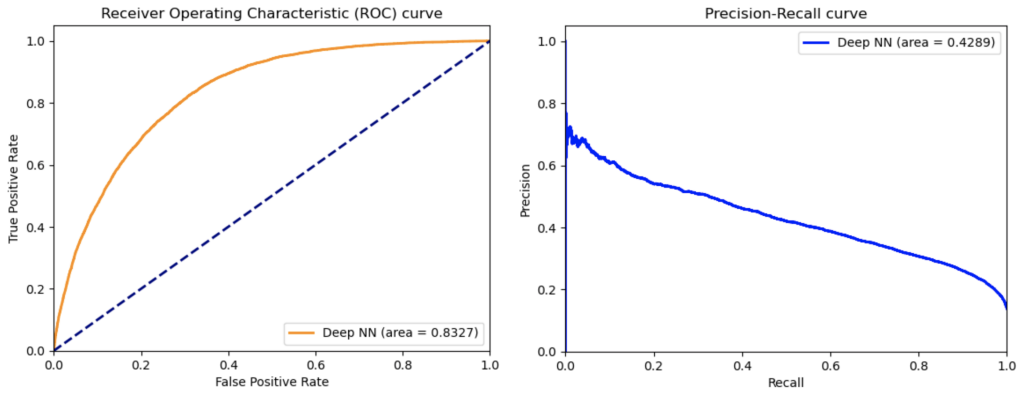
What was done: A deep neural network with two hidden layers was built and trained to classify diabetes using the given dataset. The data was first loaded and split into training and testing sets. The deep network was created with two hidden layers (64 and 32 neurons) and ReLU activation functions. The model was trained for 100 epochs using Stochastic Gradient Descent (SGD) as the optimizer.

Why this was done: A deep neural network was chosen to explore whether a more complex model can capture the patterns in the data better than the single-layer Perceptron from question 1. It was trained for 100 epochs to allow the model to learn from the data and converge to a good set of parameters. Stochastic Gradient Descent (SGD) was chosen as an optimizer for its speed and efficiency.

What was found: The deep neural network achieved an AUC of 0.8327 on the testing dataset after 100 training epochs. The ROC curve had an area of 0.8327, and the precision-recall curve had an area of 0.4289.

What the findings mean: The AUC of 0.8327 indicates that the deep neural network has better predictive power for classifying diabetes in this dataset compared to the single-layer Perceptron. This performance improvement suggests that the deep neural network can capture more complex patterns in the data, resulting in better classification accuracy. However, the precision-recall curve area of 0.4289 indicates that there is still room for improvement in the model's performance.

Regarding the potential benefits of using a CNN or RNN for the classification task: CNNs and RNNs are specialized neural network architectures designed for specific types of data. CNNs are primarily used for image processing and can be effective for grid-like data, while RNNs are designed for sequential data, such as time series or natural language. The given dataset is tabular and not grid-like or sequential, so using a CNN or RNN might not provide significant benefits for this classification task. Instead, fully connected feedforward networks, like the one used in this question, are more appropriate for tabular data.



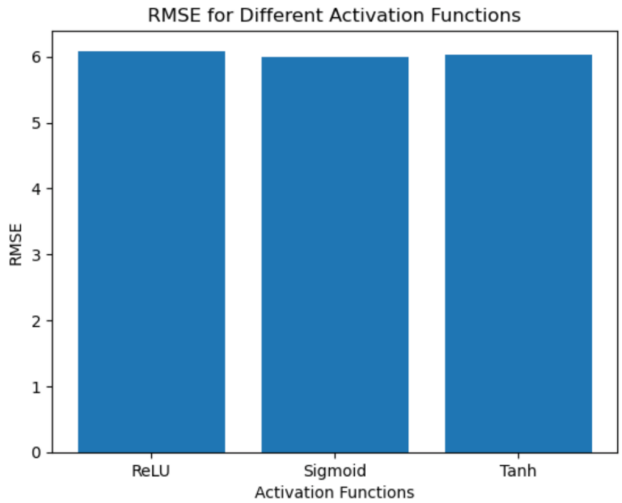
Question 4. Build and train a feedforward neural network with one hidden layer to predict BMI from the rest of the dataset. Use RMSE to assess the accuracy of your model. Does the RMSE depend on the activation function used?

What was done: A feedforward neural network with one hidden layer was built and trained to predict BMI from the given dataset. The data was first loaded and split into training and testing sets. The neural network was trained with three different activation functions - ReLU, Sigmoid, and Tanh - to investigate the impact of the activation function on the model's performance.
Why this was done: Using different activation functions can affect the learning dynamics and the final performance of a neural network. Investigating the effect of different activation functions on the RMSE of the model can help identify the best-performing model.

What was found: The feedforward neural network achieved the following RMSE values with different activation functions:

- ReLU: 6.0867
- Sigmoid: 5.9878
- Tanh: 6.0347

What the findings mean: The RMSE values show that there is a slight difference in performance when using different activation functions. The Sigmoid activation function achieved the lowest RMSE, indicating that it is the best-performing activation function for this task among the three tested. However, the differences in performance are relatively small, suggesting that the choice of activation function may not have a significant impact on the model's performance for this specific task.



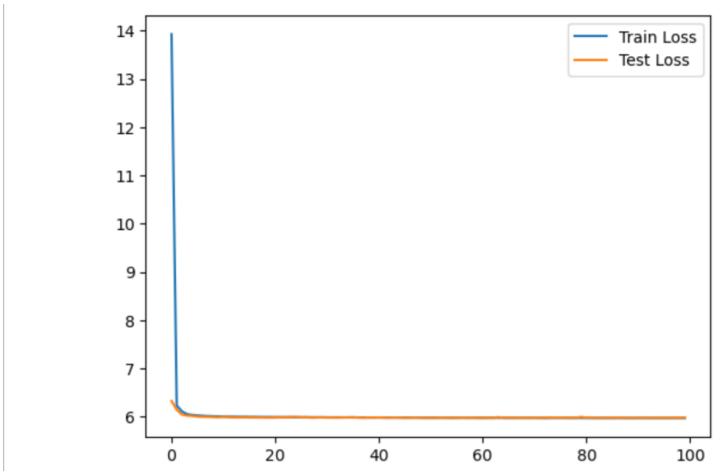
Question 5.

What was done: A feedforward neural network was built and trained to predict BMI using the given diabetes dataset. The data was first loaded and split into training and testing sets, then scaled using StandardScaler. The network architecture consisted of 2 hidden layers with 16 and 8 neurons, respectively, and a single output layer. The model was trained using the Mean Squared Error (MSE) loss function and the Adam optimizer.

Why this was done: Predicting BMI from diabetes-related features can provide valuable information for assessing and managing patient health. A neural network was chosen because it can capture complex relationships between inputs and outputs.

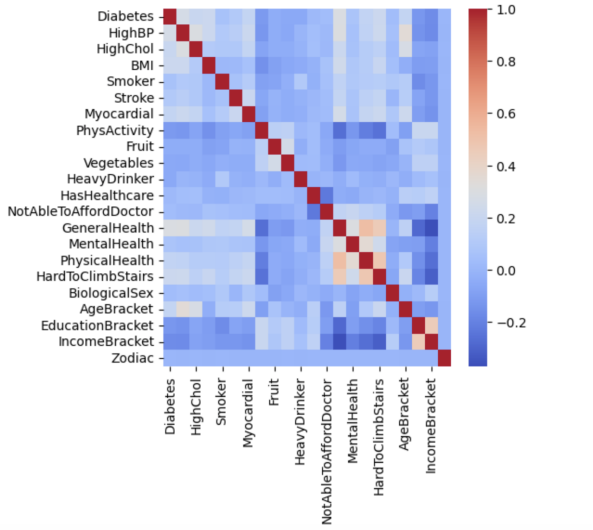
What was found: The model achieved an RMSE of 5.979 on the testing dataset after 100 epochs of training. The training and testing loss both decreased during training, indicating that the model was learning from the data. The RMSE seemed to depend on the number of neurons in the hidden layers and the learning rate used by the optimizer.

What the findings mean: The RMSE of 5.979 indicates that the model is not highly accurate in predicting BMI. However, this could be improved by tuning the model's hyperparameters or exploring more complex architectures. In particular, increasing the number of neurons in the hidden layers could capture more complex relationships between features and BMI. Additionally, experimenting with different optimization algorithms and learning rates could improve the model's convergence and overall performance.



Extra Credit

a) Yes, there are some predictors/features that have effectively no impact on the accuracy of the models. From the correlation coefficient matrices provided, we can see that "Zodiac", "HasHealthcare" have very low correlation coefficients with both "Diabetes" and "BMI". This indicates that these predictors have little to no impact on the accuracy of the models and may not be useful for predicting diabetes or BMI in this dataset. However, it's important to note that correlation does not necessarily imply causation and other factors not included in the dataset could also have an impact on the accuracy of the models.



b) In summary, using neural networks to learn from the same dataset as in the prior homework has its pros and cons compared to classical methods. The advantages of using neural networks include their ability to capture complex relationships between input features and the target variable, as well as their potential for higher accuracy in predicting the target variable. However, the main disadvantage of using neural networks is their complexity, which can make them more difficult to interpret and computationally intensive. In contrast, classical methods such as logistic regression, SVM, decision trees, random forests, and boosting methods are simpler and more interpretable models that can provide insights into the relationships between input features and the target variable. They are also generally faster and more computationally efficient than neural networks.

Overall, the main lesson is that the choice of model depends on the specific problem and the available data. If interpretability and computational efficiency are important, then classical methods may be preferred. However, if higher accuracy is required and the dataset is large enough to support the training of a neural network, then it may be worth considering using a neural network. Additionally, it's important to consider the trade-off between model complexity and interpretability, as well as the potential for bias towards the majority class in imbalanced datasets. Finally, it's worth noting that a combination of different models and techniques may be the best approach for achieving the highest possible accuracy in predicting the target variable.