# Predicting Diabetes with a Neural Network

## 1. Introduction

Diabetes is a life-changing disease related to how the body processes insulin. The disease itself is split between Type 1 and Type 2 [12]. In Type 1 diabetes, the body is unable to produce due to an autoimmune response in the pancreas, which produces insulin []. In Type 2 diabetes, the pancreas is unable to make enough insulin, and the body becomes resistant to insulin. The body becomes unable to use insulin or requires more insulin to function [].

Managing blood sugar and ketone levels becomes a life-or-death matter for those who are diagnosed with diabetes. Injectable insulin is used to help manage the condition. Affected individuals must vigilantly monitor what they choose to eat/drink with what their blood sugar levels are. If someone is at risk of becoming diabetic or begins to show signs of the disorder, it's incredibly important to monitor them and get them a timely diagnosis.

## 2. Data and Preparation

### 2.1 Challenges Involving Data Selection

I encountered some difficulties, initially, after choosing the data I would use to train and test a model. The first dataset was a 23-field 50,000-row dataset on Kaggle that had admitted it was randomly generated [2]. After preparing the initial data set using different scalers, principal component analysis, and even cross-validated recursive feature elimination (not all at the same time), the trained model continuously produced results that were less than 40% accurate. These were horrendous results for a machine learning model of any kind. Eventually, I suspected the data itself was the problem, so I went back to Kaggle to check the data again and the data was no longer available. I believe the dataset was made private/retracted due to poor quality.

### 2.2 About the Selected Data

After the debacle involving the initial training data, I found another dataset that contained 9 fields and 100,000 rows and was confirmed to be sourced from a multitude of anonymized medical records [3]. Each row is an individual person who either has been diagnosed with diabetes or is at risk of developing diabetes. The fields of data included demographic information (gender and age), smoking status, diagnoses (hypertension, heart disease, and diabetes), and other medical data (BMI, HbA1c level, blood glucose level). This data also appears to be more focused than the original dataset. There is also an added benefit of 50,000 additional rows of data. The more concise nature of the data also means that it would be easier to acquire the data needed to run it through the model in a real-world setting. Across the full 100,000-row dataset, only 8,500 diabetes diagnoses are in the data, 8.5% of the data. So BrimNet will need to make sure its weights and biases can handle the class imbalance.

### 2.3 Preparing Data for Modeling

The first step taken was dummying the data since the smoking status field had multiple values. Afterwards, a scaling method needed to be chosen. Since the data was a mix of categorical (diagnosis fields, gender, and smoking status) and numerical (age and medical fields), there were a couple of options available. Using a min-max scaler would ensure that all inputs are between 0 and 1, thereby keeping the Boolean nature of the categorical fields after dummying but may give outliers in the numerical fields more weight than expected. Converting to z-scores using a standardizer would bring everything into units of standard deviation but the Boolean fields no longer be 0's or 1's. I decided to leave the Boolean fields alone and manually convert the numerical fields into z-scores using a function that looped through the columns and only converted the specified (numerical) fields.

After scaling the data, it was split into training and testing sets of features (everything except the diabetes diagnosis) and targets (diabetes diagnosis) using Scikit-Learn's train_test_split function [11]. The testing data was split into 15% of the total data, 15,000 rows. The feature data sets came to 15 total fields. After that, these objects were converted into tensors to be useable with PyTorch.

## 3. Model Creation

### 3.1 About Neural Network Models

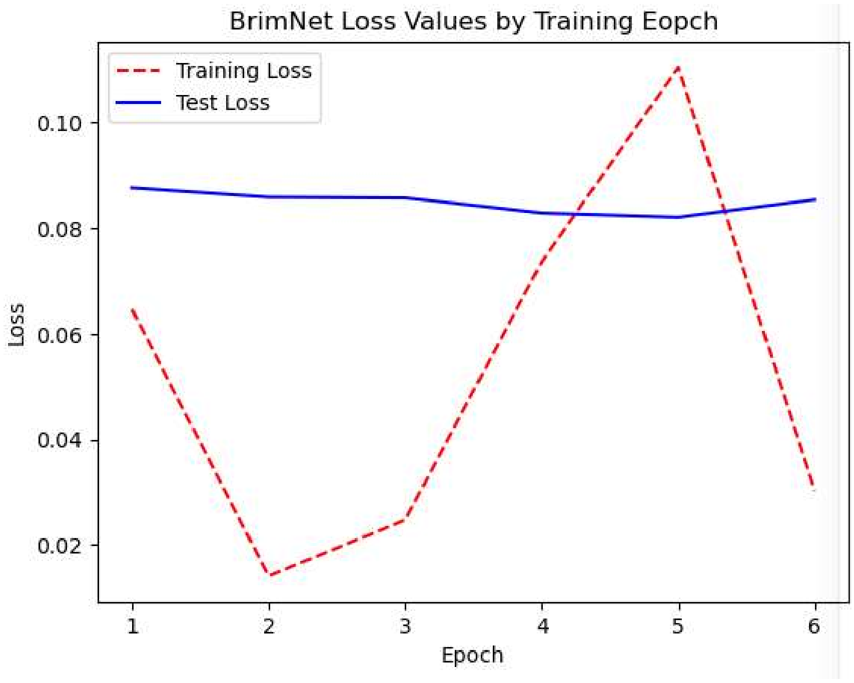
The purpose of this project was to create a neural network that could predict if a patient was diabetic based on the information provided. These models use a series of layers to transform data through matrix multiplication and activation functions on small batches of data. After each batch, the initially generated weights and biases in the model are modified with the hopes of improving model performance. After all batches are run (an epoch), the process begins again and this repeats until the desired number of epochs are run through.

### 3.2 Creating the Model

To create this model, I am using PyTorch [6]. I have used it in the past, which was a significant factor in choosing it. Selecting the correct transformation layers, activation functions, total number of layers, batch sizes, and epoch numbers is an art that can start through trial and error while being informed by the task type. The model object, BrimNet, was constructed as a child class of PyTorch's nn.Module class. Since this task is for classification, the last activation function in BrimNet is a sigmoid function, ensuring the output is the probability of the patient being diabetic [9]. For the transformational layers, I used linear layers, which put in place simple matrix multiplication using the inputs with the batch's current weights and biases [5]. For the activation layers that were not the sigmoid at the end, I tested ReLU and hyperbolic tangent [7, 10]. Both activation functions tested well but ReLU ended up with loss values of 0.01-0.02 less than hyperbolic tangent. The loss function used to evaluate the model was binary cross entropy (BCE Loss) due to the task being classification [4]. The optimization function used was PyTorch's RMSProp (root mean squared propagation) [8].

The structure of BrimNet begins with 15 inputs and scales to 24 nodes with the first linear layer to follow the 60% increase rule. These outputs are then fed into a ReLU activation function. This pair of layers is done three times, maintaining a width of 24 nodes in the hidden layers. The final pair of layers is a linear layer with 24 inputs and one output feeding into the sigmoid function.

The batch size that was chosen was 50 samples. Smaller sizes were experimented with, but I found them to perform slightly worse with respect to overfitting. The last thing to determine was the number of epochs to use with BrimNet. To test that, I stored the loss values at the end of each epoch for the training and testing set into lists to graphically compare them. We can see from the graph that the testing data consistently performed with a loss value of between 0.08 and 0.1 while the training data performed best after 2 epochs. After 2 epochs, the training data loss values swing wildly, indicating overfitting. The lowest testing loss value came after 5 epochs, but model evaluation metrics were worse than the two-epoch model. So the

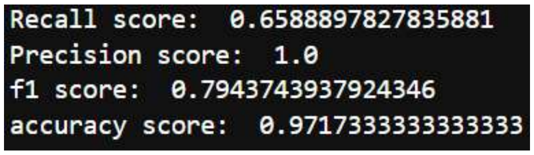


two-epoch model was chosen and trained.

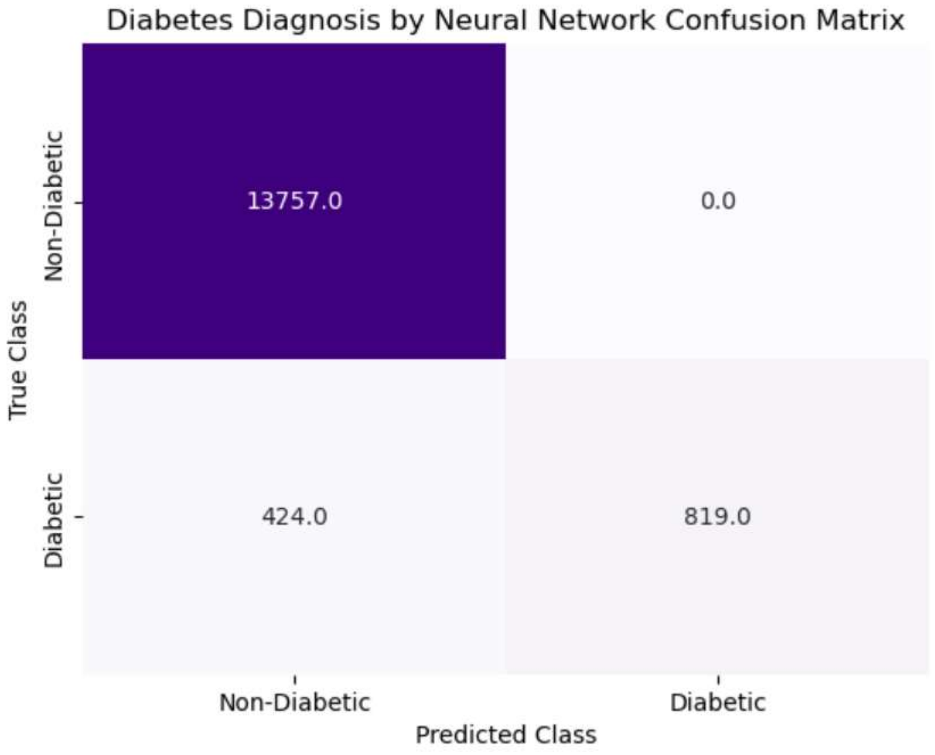
## 4. Model Evaluation and Analysis

### 4.1 Classification Metrics Used to Score Model

Since BrimNet's task was to predict a patient's diabetes diagnosis, standard classification metrics were used including accuracy, recall, precision, and F1 scores along with a confusion matrix. With the imbalance in the dataset, extra emphasis will need to be placed on the recall score. While BrimNet was around 97.17% accurate overall, it was only around 66% accurate when considering only the positive diabetic cases in the testing data (recall score). The confusion matrix below displays the overall performance of the model with the testing set.



```
Recall score:  0.6588897827835881
Precision score:  1.0
f1 score:  0.7943743937924346
accuracy score:  0.9717333333333333
```



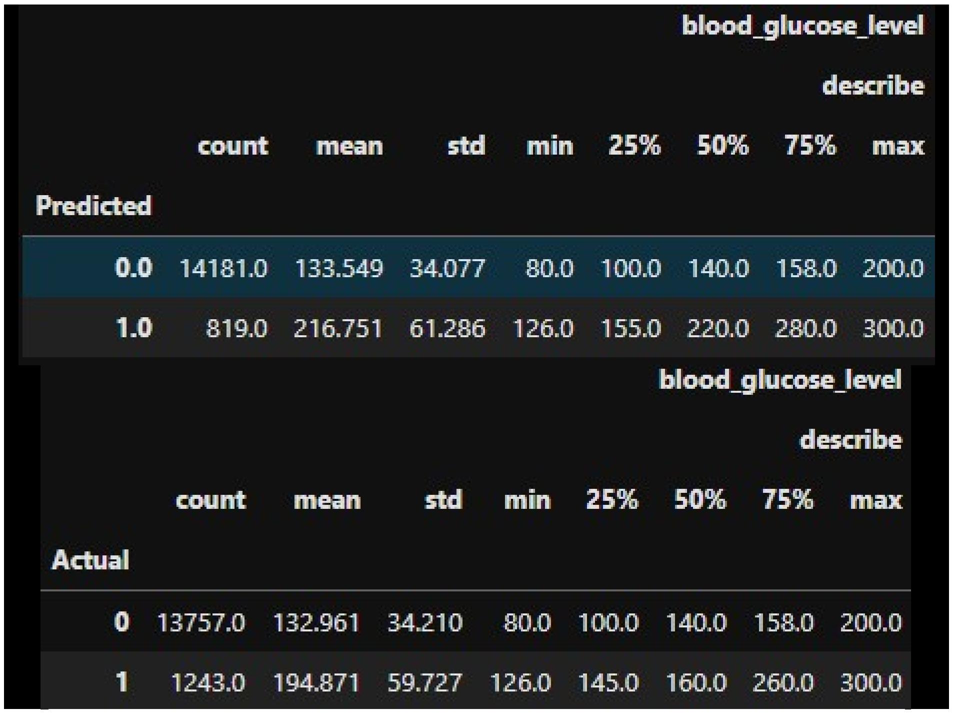
We can see that BrimNet performed incredibly well with non-diabetic patients, perfectly identifying all true negative values. However, as illustrated by the recall score, the actual diabetic patients were correctly identified around 66% of the time. This would be an area to attempt to improve the model.

### 4.2 Analyzing the False Negative Rate

BrimNet scored well for patients who were at risk and did not have a formal diabetes diagnosis. However, it also had a large population of false positives, a full third of all diabetic patients in the testing set. Looking into this could provide additional insights for how to approach the optimizing the model in the future.

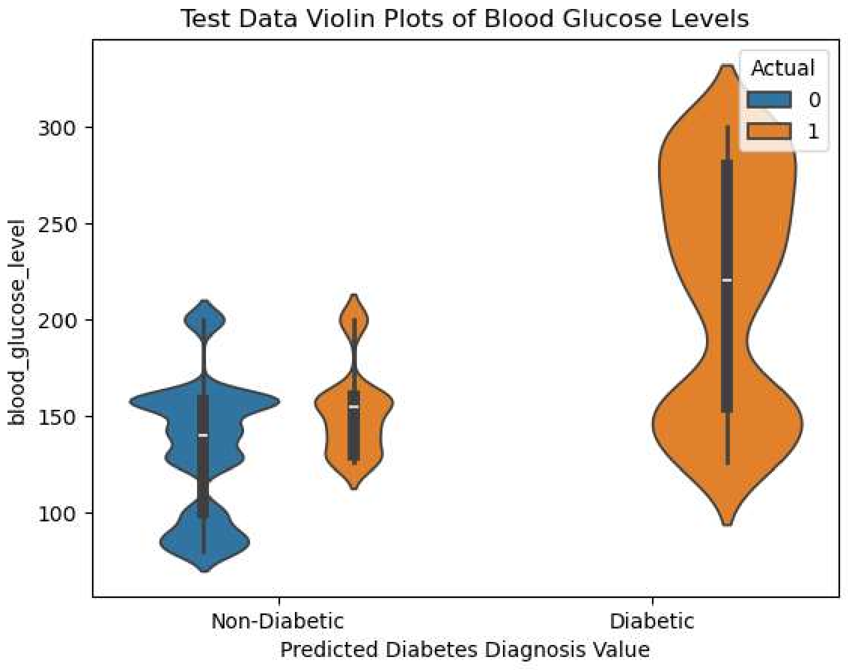
To the right, we have blood glucose summary statistic data for patients the model predicted to be diabetic and for the diagnosed diabetics. The values for non-diabetics (predicted and actual) were similar to each other while the predicted diabetics for the most part had higher blood glucose levels in each metric/quartile listed.

The violin plot shows the range and quartile information graphically while also adding frequency context akin to a histogram. For display, we can see how the data was broken down with predicted values displayed across the x-axis and actual values utilizing color. We can see that the false negative population's blood sugar levels are, on average, higher than the actual non-diabetics but they do occupy the same range as the upper end of the non-diabetic patients. The range for the true positives is higher and larger for blood glucose and its average is higher as well. The false negative population would easily fit within the lower end of the true positives. These patients could be diabetics who have managed their blood sugar levels well aside from other factors that led the model to misclassify them.

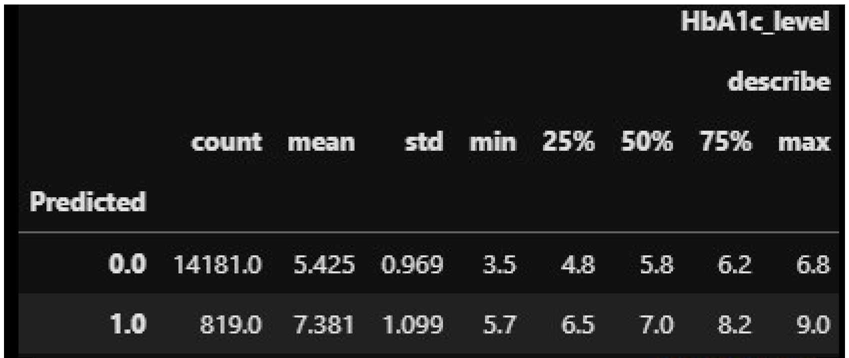


| | blood_glucose_level | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | describe | | | | | | | |
| | count | mean | std | min | 25% | 50% | 75% | max |
| Predicted | | | | | | | | |
| 0.0 | 14181.0 | 133.549 | 34.077 | 80.0 | 100.0 | 140.0 | 158.0 | 200.0 |
| 1.0 | 819.0 | 216.751 | 61.286 | 126.0 | 155.0 | 220.0 | 280.0 | 300.0 |

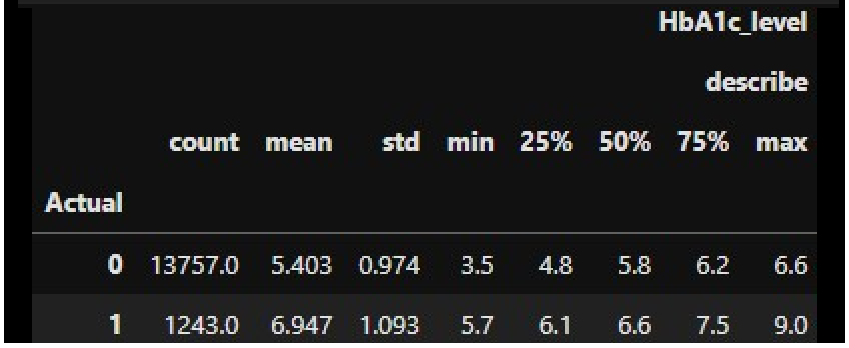
| | blood_glucose_level | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | describe | | | | | | | |
| | count | mean | std | min | 25% | 50% | 75% | max |
| Actual | | | | | | | | |
| 0 | 13757.0 | 132.961 | 34.210 | 80.0 | 100.0 | 140.0 | 158.0 | 200.0 |
| 1 | 1243.0 | 194.871 | 59.727 | 126.0 | 145.0 | 160.0 | 260.0 | 300.0 |



A1C values are another metric commonly used to evaluate a patient's diabetes status. The A1C test measures glycated hemoglobin, which reflects average blood sugars [1]. A1C values below 5.7 are considered normal, 5.7 to 6.5 are considered pre-diabetic, and those above 6.5 are considered diabetic.



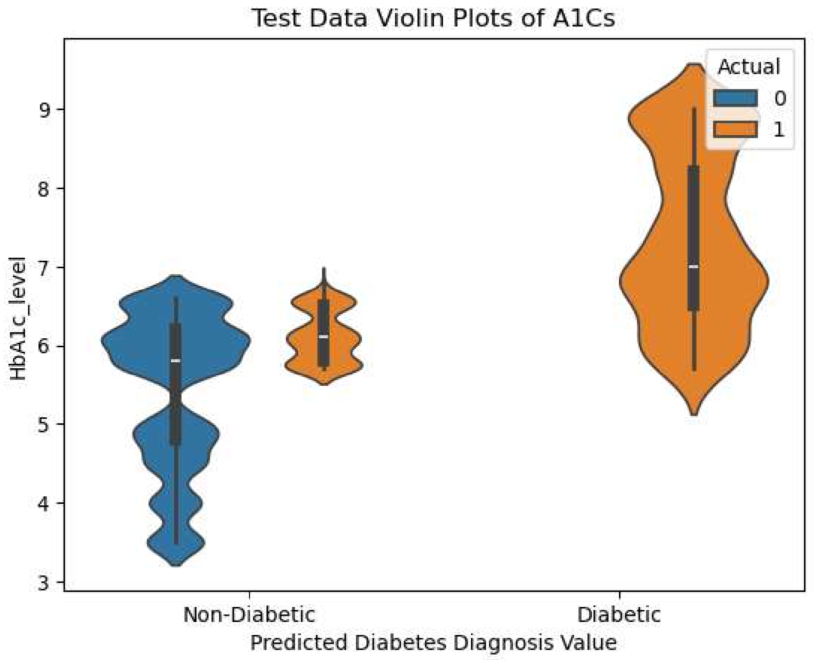
| | HbA1c_level | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | describe | | | | | | | |
| | count | mean | std | min | 25% | 50% | 75% | max |
| Predicted | | | | | | | | |
| 0.0 | 14181.0 | 5.425 | 0.969 | 3.5 | 4.8 | 5.8 | 6.2 | 6.8 |
| 1.0 | 819.0 | 7.381 | 1.099 | 5.7 | 6.5 | 7.0 | 8.2 | 9.0 |



| | HbA1c_level | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | describe | | | | | | | |
| | count | mean | std | min | 25% | 50% | 75% | max |
| Actual | | | | | | | | |
| 0 | 13757.0 | 5.403 | 0.974 | 3.5 | 4.8 | 5.8 | 6.2 | 6.6 |
| 1 | 1243.0 | 6.947 | 1.093 | 5.7 | 6.1 | 6.6 | 7.5 | 9.0 |

Comparing summary statistics for predicted and actual populations, we see, once again, that values for those predicted to be non-diabetic and patients who are actually non-diabetic are similar and within fractions of a standard deviation of each other.

The population of predicted diabetics appeared to have a higher mean and quartile A1C values than the set of actual diabetics in the testing data.

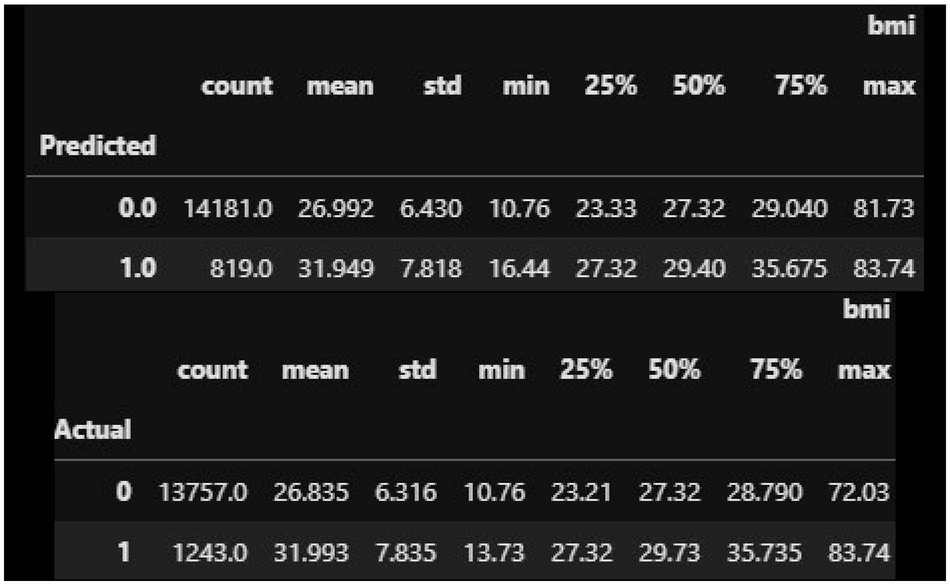


Once again, we see that the false negatives from the testing data occupy the upper range of data in the predicted non-diabetic class and the lower range of the predicted diabetic class. Their mean and quartile values are higher than their true negative counterparts and lower than their true positive counterparts. Noting the ranges listed above, we see that the false negative population fits neatly within the pre-diabetic range of A1C values. The dataset, unfortunately, does not clarify if "pre-diabetic" was included as diabetic or non-diabetic. It cannot be concluded either way but it may be likely that "pre-diabetics" are included in the diabetic population.

BMI is the other numeric metric that was used as part of BrimNet's classification process. This metric brings height and weight together to quantify an individual's build.

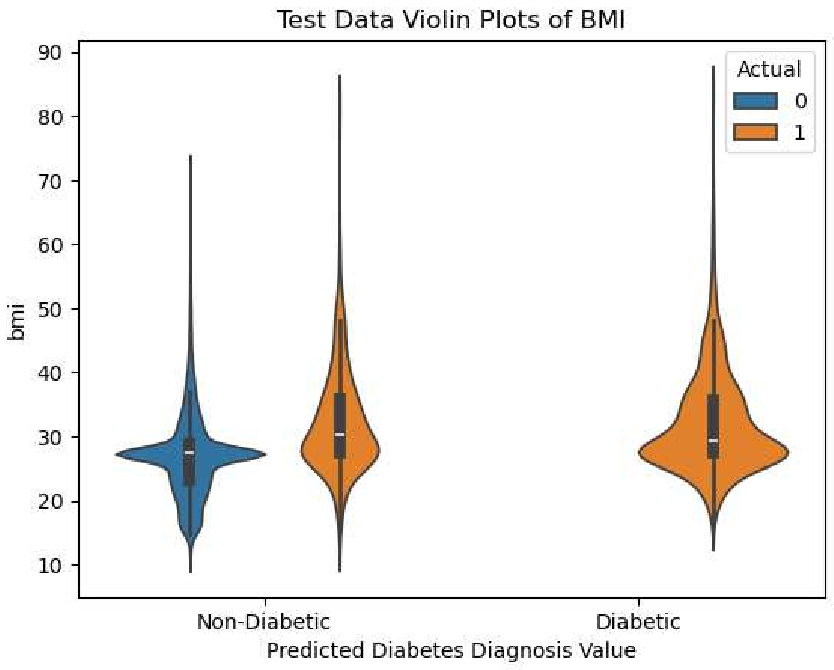
The summary statistics for BMI across the predicted and actual populations change somewhat for BMI. Here, we can see that the predicted non-diabetic populations have slightly higher values for almost all metrics. At the same time, the predicted diabetic and actual diabetic values are mostly similar outside of their minimum values.

The violin plot shows that the false negative population occupies a similar range as the true positive population from the testing data. Their means are similar as well. Given that the false negative and true positive populations appear similar with respect to BMI, it may be possible that BMI was not a strong factor with respect to predicting if a patient was diabetic or not.



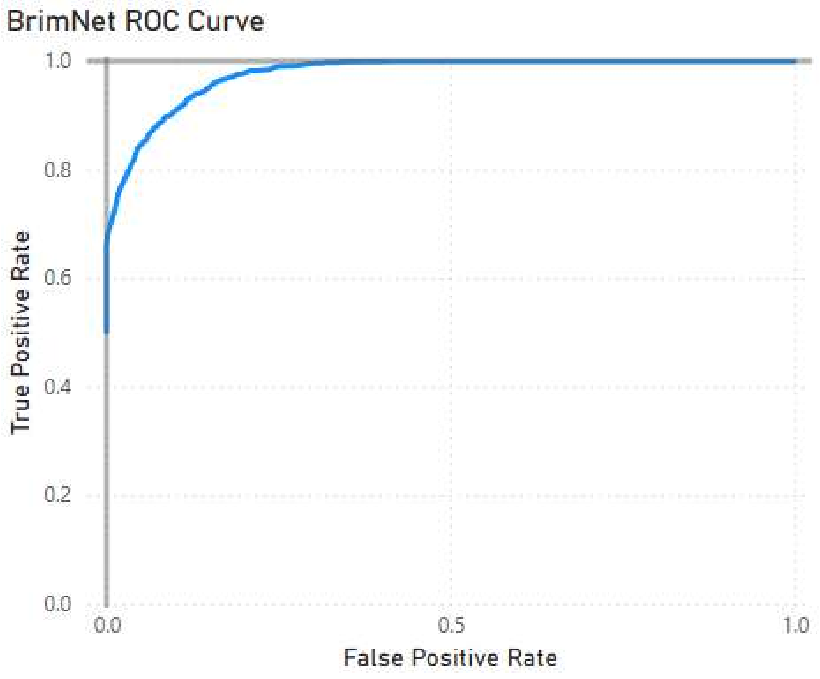
| | bmi | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | count | mean | std | min | 25% | 50% | 75% | max |
| **Predicted** | | | | | | | | |
| **0.0** | 14181.0 | 26.992 | 6.430 | 10.76 | 23.33 | 27.32 | 29.040 | 81.73 |
| **1.0** | 819.0 | 31.949 | 7.818 | 16.44 | 27.32 | 29.40 | 35.675 | 83.74 |

| | bmi | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | count | mean | std | min | 25% | 50% | 75% | max |
| **Actual** | | | | | | | | |
| **0** | 13757.0 | 26.835 | 6.316 | 10.76 | 23.21 | 27.32 | 28.790 | 72.03 |
| **1** | 1243.0 | 31.993 | 7.835 | 13.73 | 27.32 | 29.73 | 35.735 | 83.74 |



**4.3 BrimNet ROC Curve**

The ROC Curve is another standard method to assess the quality of a classification model. BrimNet's ROC curve quickly moves towards the upper-left corner of the plot, indicating that the model predicts well at higher threshold values for the probability of a diabetes diagnosis.



## 5. Model Uses

BrimNet would be a great aid in attempting to diagnose diabetes in new patients. It *would not* and should not be a deciding factor for diagnoses. A doctor would be able to conduct the final tests necessary for a formal diabetes diagnosis.

## 6. Conclusion

Using machine learning to help in the diagnosis of diseases and disorders would be incredibly helpful for doctors, PAs, and medical professionals. While established tests do exist to diagnose patients with diabetes, additional information can help with the diagnostic process. The information era will also allow medical professionals to more efficiently analyze patient data. BrimNet and models like it would be another tool for doctors and PAs to use to help them help their patients. We can see from the data that diabetes diagnostics using machine learning is not always clear cut, depending on the information within the dataset. Pre-diabetics could be classified either way using BrimNet but the specific data values could also be used to help them with respect to treatment and guiding them away from becoming diabetic due to insulin resistance.

## 7. References

[1] American Diabetes Association. (2025). *What is the A1C Test?* Retrieved from American Diabetes Association: https://diabetes.org/about-diabetes/a1c

[2] Choudhary, A. (2025). *Diabetes Prediction in America Dataset*. Retrieved from Kaggle: https://www.kaggle.com/datasets/ashaychoudhary/diabetes-prediction-in-america-dataset

[3] Mustafa, M. (2023). *Diabetes prediction dataset*. Retrieved from Kaggle: https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset

[4] PyTorch. (2025). *BCELoss*. Retrieved from PyTorch: https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html#torch.nn.BCELoss

[5] PyTorch. (2025). *Linear*. Retrieved from PyTorch: https://pytorch.org/docs/stable/generated/torch.nn.Linear.html#torch.nn.Linear

[6] PyTorch. (2025). *PyTorch documentation*. Retrieved from PyTorch: https://pytorch.org/docs/stable/index.html

[7] PyTorch. (2025). *ReLU*. Retrieved from PyTorch: https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html#torch.nn.ReLU

[8] PyTorch. (2025). *RMSProp*. Retrieved from PyTorch: https://pytorch.org/docs/stable/generated/torch.optim.RMSprop.html#torch.optim.RMSprop

[9] PyTorch. (2025). *Sigmoid*. Retrieved from PyTorch: https://pytorch.org/docs/stable/generated/torch.nn.Sigmoid.html#torch.nn.Sigmoid

[10] PyTorch. (2025). *Tanh*. Retrieved from PyTorch: https://pytorch.org/docs/stable/generated/torch.nn.Tanh.html#torch.nn.Tanh

[11] Scikit-Learn Developers. (2024). *train_test_split*. Retrieved from Scikit-Learn: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html#sklearn.model_selection.train_test_split

[12] UVA Health. (2025). *Type 1 vs Type 2 Diabetes*. Retrieved from UVA Health: https://uvahealth.com/services/diabetes-care/types