MLPClassifier is a supervised learning algorithm that models complex relationships in data through a network of interconnected hidden layers. It consists of an input layer, hidden layers where each neuron weights sum of inputs followed by non-linear activation function and an output layer. It is part of the feedforward neural network and considered to be effective and produces promising results in health data classification1. Bidirectional Long Short-Term Memory (BiLSTM) model is another popular learning algorithm in disease prediction. It is a type of recurrent neural network (RNN) that uses two LSTM layers running in opposite directions — one processing the input sequence forward, and the other backward — allowing the model to learn information from both the past and the future at each time step. It is primarily used for contextual information or time-series data prediction2,3.

MLP and BiLSTM are chosen for comparison in training for BRFSS 2015 diabetes dataset, which is in tabular format. Using MLP to train classification data seems to be obvious for its fast and accurate prediction in flat datasets, However, the feedforward characteristics prevent it from looking back into the dataset and observe correlations between seemingly random factors. Diseases like diabetes are known for its difficulty of being detected through simple correlations. The imperfection of this algorithm becomes the strength of the less popular BiLSTM. BiLSTM is adept at uncovering relationships between distant features that MLP might ignore. It also has proved its potentials in disease classification by previous research studies2. Therefore, comparing BiLSTM with MLP might be helpful to understand the long-range dependencies between feature dimensions that might be missed by feedforward models.

The implementation of MLP utilized Scikit-learn’s MLPClassifier, combined with RandomizedSearchCV for hyperparameter tuning. Scikit-learn offers a clean and user-friendly interface which enables efficient model development and rapid experimentation. The use of RandomizedSearchCV allowed for a strategic exploration of the hyperparameter space by randomly sampling combinations, leading to faster convergence toward optimal configurations.

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| 0.0 | 0.79 | 0.70 | 0.74 | 7090 |
| 1.0 | 0.73 | 0.81 | 0.77 | 7049 |
|  |  |  |  |  |
| **Accuracy** |  |  | **0.76** | **14139** |
| **Macro Avg** | 0.76 | 0.76 | 0.75 | 14139 |
| **Weighted Avg** | 0.76 | 0.76 | 0.75 | 14139 |
| **(Table1)** |  |  |  |  |

After conducting more than 10 rounds of training and parameter exploration, the model achieved an accuracy of approximately 75%, with a notable improvement in precision and recall for patients diagnosed with diabetes. This indicates a stronger ability to correctly identify positive cases which is important for medical prediction tasks. Table 1 presents the classification results from one representative round. Across all training rounds, the variation in performance remained minimal, with differences ranging between only by 1-2%, suggesting a consistent model behavior. The hyperparameters used to generate the results in Table 1 were as follows:

**{'solver': 'adam', 'n\_iter\_no\_change': 40, 'max\_iter': 20000, 'learning\_rate': 'constant', 'hidden\_layer\_sizes': (128, 64, 32), 'early\_stopping': True, 'alpha': 0.01, 'activation': 'logistic'}.**

This configuration, particularly the use of the logistic sigmoid activation function combined with adam optimizer, consistently reappeared as one of the most effective combinations across multiple training rounds. The logistic activation proved well-suited for binary classification1 in this context while the Adam delivered stable and efficient convergence. With the two parameters combined along with other fine tunings, the model’s performance achieves a comparably good performance rate compared with other similar approaches4,5 with a promising potential.

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| 0.0 | 0.76 | 0.65 | 0.70 | 7090 |
| 1.0 | 0.69 | 0.79 | 0.74 | 7049 |
|  |  |  |  |  |
| **Accuracy** |  |  | **0.72** | **14139** |
| **Macro Avg** | 0.73 | 0.72 | 0.72 | 14139 |
| **Weighted Avg** | 0.73 | 0.72 | 0.72 | 14139 |
| **(Table2)** |  |  |  |  |

The training with BiLSTM uses two stacked, two-directional hidden layers and a Sigmoid function at the output, which converts the final layer’s output into a probability between 0 and 1 for predicting the likelihood of diabetes. It is expected that the overall performance of BiLSTM was lower compared to MLPClassifier. Table 2 presents the classification results from one of the representative rounds.

However, it is worth noting that with only 2,000 epochs (significantly fewer than the 20,000 epochs used in MLP training), the BiLSTM model was able to achieve a comparable recall rate. This suggests the potential of BiLSTM in situations where recall is prioritized to minimize false negatives and dataset scale is small. This provides an insightful direction for future optimization especially in medical applications where early detection and recall are critical.

To summarize, both MLPClassifier and BiLSTM demonstrated reasonably strong predictive performance and offer promising potential for further improvement. Existing research studies using Pima Indian Diabetes Dataset have reported achieving up to 95% of accuracy rate6 by leveraging a combination of classical machine learning and deep neural networks.

After multiple rounds of hyperparameter tuning, it became evident that the current performance may be limited not only by model configuration, but also by underlying data challenges such as noise, class overlap, or missing information. This suggests that additional improvements could be achieved through more advanced tuning, data preprocessing, or hybrid modeling strategies that combine MLP or BiLSTM with ensemble methods.

After applying both BiLSTM and Feedforward Neural Networks (FNN) to our diabetes prediction task, I arrived at three key conclusions:

1. **BiLSTM is not suitable for tabular data**  
   While BiLSTM is powerful for sequential or contextual data (e.g., text, time series), it is excessive—if not inappropriate—for tabular datasets where each row is independent. FNN or tree-based models are more efficient and effective for this structure.
2. **Data quality limits predictive performance**  
   Although the dataset used is rated 10.00/10.00 for usability on Kaggle, it may still contain imperfections such as noise, missing values, edge cases, or class overlap. These issues introduce challenges for model learning and generalization, highlighting a fundamental limitation: even advanced models cannot compensate for weak or ambiguous data.
3. **Hyperparameter tuning improves targeted metrics but cannot overcome data limitations**  
   Hyperparameter search—particularly when focused on improving **recall**—did enhance performance for identifying positive diabetes cases. However, it did not significantly boost overall accuracy. This reinforces the idea that **data quality is central** to model performance and cannot be replaced by algorithmic optimization alone.

**Why I Prioritized Recall**

* Diabetes is a chronic condition with high healthcare and personal costs if left undiagnosed. By prioritizing **recall**, we intentionally increase the model’s sensitivity to positive cases, minimizing **false negatives**. While this may lead to more **false positives** (healthy individuals incorrectly flagged), the tradeoff is justifiable in a healthcare setting where early intervention can save lives. My final model achieved a **10% improvement in recall for positive diabetes cases**, suggesting that this system could serve as a clinical **decision support tool** to assist physicians in screening for potential diabetes risk.