Predict diabetes using Perceptron

Irfan F Boelhasrin
University of Adelaide
Adelaide, South Australia
irfan.boelhasrin@student.adelaide.edu.au

Abstract

Diabetes is a growing global health issue, particularly affecting populations in low- and middle-income countries. Early diagnosis is crucial to prevent severe complications and reduce treatment costs. In this study, we developed a machine learning model using a Single Layer Perceptron (SLP) to classify diabetes risk based on the Pima Indian Diabetes dataset, consisting of 768 samples with 8 medical predictor variables. We trained the perceptron model using PyTorch, evaluating its performance with metrics such as accuracy, confusion matrix, and ROC AUC. Our perceptron model demonstrated competitive performance, highlighting its potential in binary classification tasks for early diabetes detection with the final model achieved an accuracy of 79.69% and an ROC AUC of 0.84 after hyperparameter tuning.

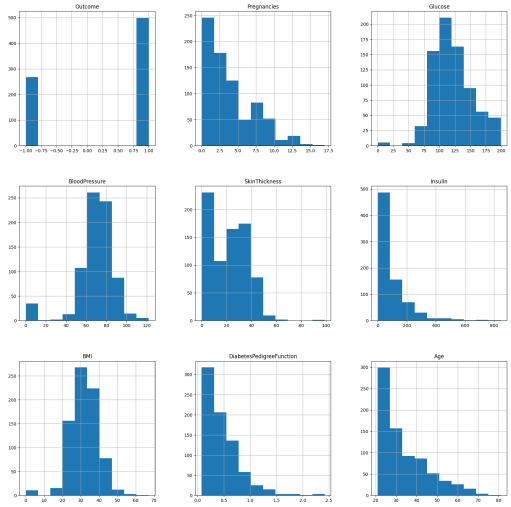


Figure 1. Histogram of dataset variables

1. Introduction

Diabetes is a chronic disease that occurs when the body cannot produce or properly use insulin, leading to elevated blood sugar levels. It is a major global health issue and one of the four priority noncommunicable diseases. Since 1980, the prevalence of diabetes has nearly doubled, with 422 million adults affected worldwide in 2014. This rise is linked to risk factors such as obesity and has been more rapid in low- and middle-income countries. In 2012, diabetes contributed to 3.7 million deaths, with a significant proportion occurring before the age of 70. Most cases are type 2 diabetes, which is increasingly affecting younger populations [5].

A key challenge in addressing noncommunicable diseases like diabetes is the high number of undiagnosed cases due to minimal early symptoms. Early diagnosis is crucial for preventing disease progression and costly treatments, especially in underserved areas with limited healthcare providers [1]. Machine learning models can assist in early detection, but their lack of transparency, often referred to as the "black-box" effect, can hinder adoption in healthcare. Building trust through model explainability is essential, particularly where errors could have serious consequences [1].

In this paper, we develop a machine learning model using a Perceptron. The Pima Indian dataset was employed, consisting of women of Pima Indian heritage over the age of 20, residing in Phoenix, Arizona, USA, at the time of the survey. The dataset, originally from the National Institute of Diabetes and Digestive and Kidney Diseases, is a widely used benchmark, historically employed to train early neural network models like ADAP [6].

2. Data and method description

2.1. Dataset

The Pima Indians Diabetes dataset was used for training and testing. It was sourced from Kaggle by University of California Irvine (UCI), with a pre-processed version available in LIBSVM format in National Taiwan University (NTU) [4, 7]. The dataset consists of 768 samples, with 8 medical predictor variables. The target variable is binary, where 1 represents a positive diabetes diagnosis, and -1 rep-

resents a negative diagnosis. Fig. 1 shows the distribution of the variables.

The 8 predictor variables are described as follows:

- **Pregnancies**: Number of times pregnant.
- **Glucose**: Plasma glucose concentration a 2 hours in an oral glucose tolerance test.
- **BloodPressure**: Diastolic blood pressure (mm Hg).
- **SkinThickness**: Triceps skin fold thickness (mm).
- **Insulin**: 2-Hour serum insulin (mu U/ml).
- **BMI**: Body mass index (weight in kg/(height in m)$\hat{2}$).
- **DiabetesPedigree**: Diabetes pedigree function.
- **Age**: Age (years).

2.2. Perceptron

A Single Layer Perceptron (SLP) is the simplest and most fundamental architecture of an Artificial Neural Network (ANN), based on an artificial neuron called a Threshold Logic Unit (TLU). The TLU receives input signals $x_1, x_2, \ldots, x_n$, where the importance of each input is represented by the weights of the connections between neurons $w_1, w_2, \ldots, w_n$. The TLU then computes a weighted sum of its inputs:

$$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \mathbf{x}^\top \mathbf{w},$$

and applies a step function to that sum, outputting the result:

$$h_{\mathbf{w}}(\mathbf{x}) = \text{step}(z), \quad \text{where} \quad z = \mathbf{x}^\top \mathbf{w}.$$

Alternatively, when introducing bias b and using an activation function ϕ, the perceptron computes:

$$h_{\mathbf{W},b}(\mathbf{x}) = \phi(\mathbf{x}^\top \mathbf{W} + b).$$

A single-layer perceptron consists of a single layer of TLUs, with each TLU connected to all the inputs [2].

In this paper, the experiment will be conducted using Facebook's PyTorch library, one of the most popular Deep Learning library which grew popularity in 2018 for it's simplicity [2].

3. Implementation

3.1. Dataset preparation

To ensure proper model training, the input features were standardised using `StandardScaler` to have a mean of 0 and a standard deviation of 1. Additionally, the target variable was converted to the scale of 0 and 1.

Hyperparameters	Value
Optimiser	SGD
Learning rate	0.01
Batch size	32
Number of epochs	50

Table 1. Baseline model hyperparameters

The dataset was then split into training and testing sets, with a 3:1 ratio for training to testing. The training set was further divided into training and validation sets with a 5:1 ratio. To meet PyTorch requirements, all sets were converted to the `torch.Tensor` data type.

3.2. Model implementation

The perceptron was implemented using the PyTorch framework, structured as a single-layer model with a fully connected layer followed by a sigmoid activation function. The model takes an input vector and applies a linear transformation using a fully connected layer, followed by the sigmoid activation to output probabilities. Algorithm 1 shows the algorithm of our Single Layer Perceptron model.

Algorithm 1 Single Layer Perceptron Model

1: **Input:** Feature vector x_{in} with dimension input_dim
2: Initialize fully connected layer fc1 with weights W and bias b
3: Initialize sigmoid activation function σ
4: Compute linear transformation: $z = W \cdot x_{\text{in}} + b$
5: Apply sigmoid activation: $\hat{y} = \sigma(z)$
6: **return** $\hat{y}$

The training process of our perceptron model used the training set over a specified number of epochs and calculated both the training and validation loss, as well as accuracy, during each epoch. Accuracy was calculated based on the predicted outputs, and will be used as our model's evaluation metric. The model's parameters were updated using an optimizer, and the loss was computed using a specified loss function. At the end of each epoch's training phase, the model was validated using the validation set.

After the training phase was completed, we used the testing set to evaluate our model. The evaluation phase computed the loss and accuracy, and generated the confusion matrix and ROC curve values as additional visual performance metrics, based on the comparison between the model's predicted outputs and the actual output values.

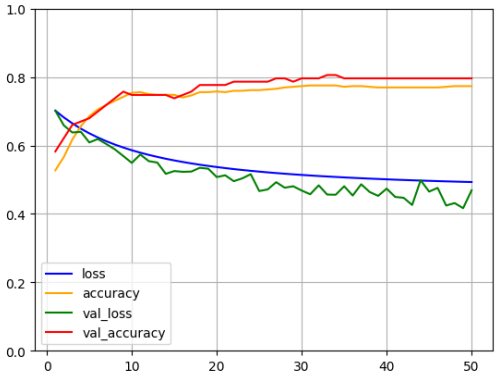


Figure 2. Learning curves of the baseline model

4. Experiments

4.1. Baseline model result

Our initial Perceptron model uses mostly default hyperparameters which are typically used in Perceptron models, as shown on Tab. 1. A simple Perceptron learning algorithm resembles the Stochastic Gradient Descent (SGD) classifier, therefore we use SGD as our optimiser with a default learning rate of 0.01. The default batch size of 32 was also applied, while we set the number of epochs to 50, a sufficiently larger number compared to the default value of 1, which would certainly be insufficient for training [2]. In all experiments, we used the Binary Cross-Entropy loss function for backpropagation, along with a sigmoid activation function to ensure compatibility with our binary classification task. Fig. 2 shows the learning curves over 50 epochs, showing the mean training and validation accuracy and loss at each epoch.

Model training and evaluation resulted in a test accuracy of 78.52%, loss of 0.49, and ROC AUC of 0.84. Fig. 3 shows the confusion matrix heatmap and ROC curve for the baseline model.

4.2. Data imputations

We detected inconsistent values in the variables, where Glucose, BloodPressure, SkinThickness, Insulin, and BMI should not have zero values [1]. Although there may be reasons for these inconsistencies, and we do not have sufficient information to conclude that the values are incorrect, we will treat them as missing values to assess whether this approach improves our model. Therefore, in our first experiment, we trained the model using the dataset processed with median imputation. Fig. 4 shows the histogram of the variables after imputation.

However, the results of the data imputation experiment showed slightly lower values for both accuracy and loss, with 75.39% accuracy and a loss of 0.48, indicating that imputation did not necessarily improve the model's performance.

4.3. Hyperparameter tuning

Another experiment we conducted was hyperparameter tuning. Several hyperparameters were adjusted to optimise the model's performance:

- **Learning rate**: Tested values were 0.01, 0.001, and 0.0001.
- **Batch size**: Values of 16, 32, and 64 were tested.
- **Number of epochs**: We trained the model for both 50 and 100 epochs.
- **Optimiser type**: Adam and SGD optimisers were tested.

The top ten hyperparameter combinations are displayed in Tab. 2, indicating that the Batch Size hyperparameter was insignificant on our model's performance. The best results were achieved with a learning rate of 0.01, a batch size of 16, 100 epochs, and the SGD optimiser, yielding an accuracy of 79.69% and a loss of 0.47. The ROC AUC for the baseline model and the tuned model were very similar, as shown in Fig. 5

4.4. Model comparison

The final experiment we conducted was to compare the performance of our Perceptron model with another high-performing non-neural network machine learning classification method. A Linear Kernel Support Vector Machine (SVM) with a C value of 1 achieved an accuracy of 89%. However, this result was obtained in R environment instead of Python, with slightly different preprocessing techniques, including feature engineering and k-NN imputation [3].

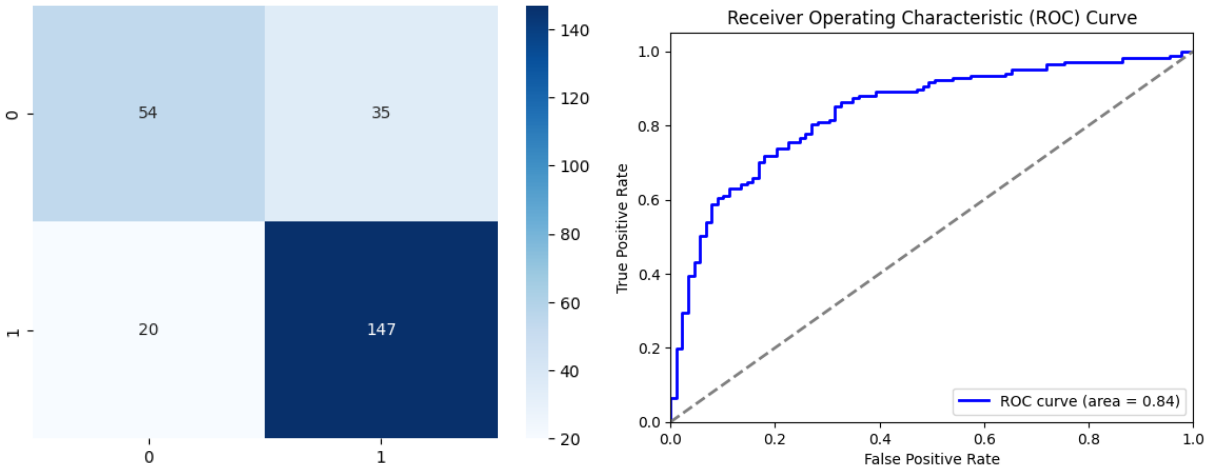
Training the SVM model on our data resulted in an accuracy of 78.12%, which, while lower than the performance of our best-performing Perceptron model, was still comparable. Fig. 6 shows the ROC AUC curves comparing our best-performing model and the SVM.

4.5. Code

The complete code for this project, including data preprocessing, model implementation and experiments, and hyperparameter tuning, is available on GitHub:

GitHub Repository: COMP SCI 7318 Deep Learning Fundamentals Assignment 1: Predict diabetes using Perceptron (https://github.com/ifboelhasrin/DLF_A1)

The repository is publicly available, so no access permissions are required.



(a) Confusion matrix of the baseline model

(b) ROC curve of the baseline model

Figure 3. Performance analysis plots of the baseline model

Learning Rate	Batch Size	Num Epochs	Optimiser	Test Accuracy	Test Loss
0.0100	16	100	sgd	79.69%	0.4741
0.0100	32	100	sgd	79.69%	0.4741
0.0100	64	100	sgd	79.69%	0.4741
0.0100	64	100	adam	79.30%	0.4713
0.0100	16	100	adam	79.30%	0.4713
0.0100	32	100	adam	79.30%	0.4713
0.0100	16	50	adam	78.91%	0.4720
0.0010	64	100	adam	78.91%	0.4785
0.0010	16	100	adam	78.91%	0.4785
0.0010	32	100	adam	78.91%	0.4785

Table 2. Hyperparameter tuning results with varying learning rates, batch sizes, and optimisers

5. Conclusion

In this study, we developed a machine learning model using a Single Layer Perceptron (SLP) to classify diabetes based on the Pima Indian dataset. After conducting several experiments, we observed that our SLP model performed well with an accuracy of 79.69% after tuning the hyperparameters.

The data imputation experiment did not lead to a significant improvement in performance, but it was a valuable step in addressing inconsistencies within the dataset. Additionally, while the SVM model achieved a higher accuracy of 89% under different conditions, our SLP model's performance remained competitive when evaluated on the same dataset and metrics. The result of these experiments showed the potential of SLP model for binary classification tasks such as diabetes prediction, especially when properly tuned.

Future work could explore more complex architectures, such as Multi-Layer Perceptrons, and implement additional preprocessing techniques to address the dataset's imbalanced data.

References

[1] V Chang, J Bailey, QA Xu, and Z Sun. Pima indians diabetes mellitus classification based on machine learning (ml) algorithms. *Neural computing & applications*, 35(22):16157–16173, 2023. 1, 3

[2] A Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Incorporated, 2019. 2, 3

[3] H Kaur and V Kumari. Predictive modelling and analytics for diabetes using a machine learning approach. *Applied computing & informatics*, 18(1-2):90–100, 2022. 3

[4] UCI Machine Learning. Pima indians diabetes database, 2016. Accessed: 2024-09-17. 1

[5] World Health Organization. Global report on diabetes. Technical report, World Health Organization, 2016. 1

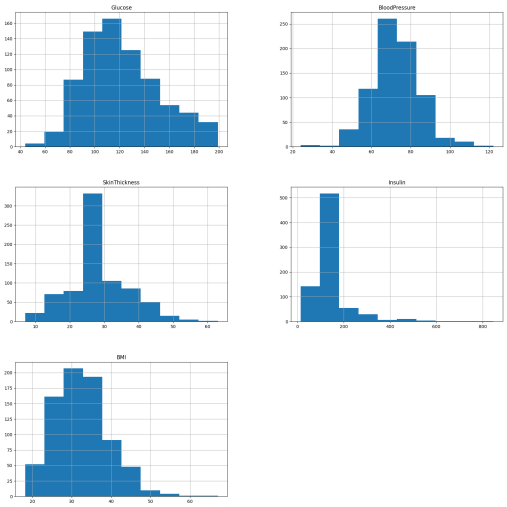


Figure 4. The distribution of the imputed variables

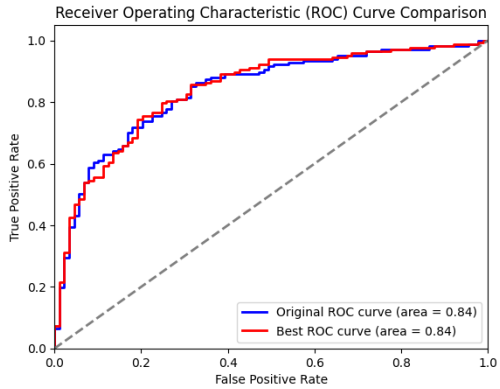


Figure 5. The ROC AUC curves comparison between the original and the best performing model

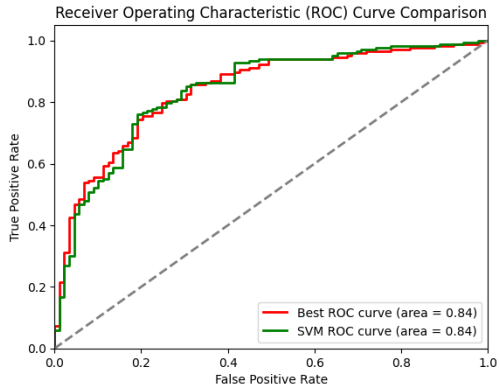


Figure 6. The ROC AUC curves comparison between the best performing model and SVM

[6] JW Smith, JE Everhart, WC Dickson, WC Knowler, and RS Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings - Symposium on Computer Application in Medical Care*, 1988. 1

[7] National Taiwan University. Libsvm data: Classification (binary class), 2016. Accessed: 2024-09-17. 1