Implementing the Perceptron Algorithm to Predict Diabetes

Tanvi Krishna Murthy Jois
a1942914
The University of Adelaide
Adelaide, South Australia
a1942914@adelaide.edu.au

Abstract

Machine learning is one of the most vital tools in predictive analysis enabling easier decision making across various disciplines like healthcare, finance, transportation etc. It helps uncover insights from large datasets fast. The 'Perceptron' is one of the basic and earliest machine learning models that has served the foundation of modern-day neural networks. Majorly, the perceptron is a linear classifier used to classify data with two distinct outcomes (binary). In this project, the Pima Indians diabetes dataset is used to predict the occurrence of diabetes in a person against a few features like BMI, Blood Pressure, Age, Pregnancies, Glucose levels etc.

The dataset was pre-processed by, replacing the zero values with NaNs and replacing the NaNs with the medians, balancing the dataset using oversampling and standardizing the dataset to implement the perceptron model to it. Major statistics of the data like accuracy, precision, recall and f1-scores are achieved.

The implementation of the perceptron to the dataset yielded an overall accuracy of 76.5%.

1. Introduction

Machine learning has revolutionized the modern world by achieving predictive analysis of data which can diagnose risks and assess potential shortcomings which in-turn provides room for improvement and prevention of disasters. Namely in the healthcare industry, machine learning can prove to be very useful as early detection of diseases can help avoid severe complications which can hopefully lead to successful management of the disease or permanent cure.

Diabetes is one such disease which if left untreated can cause multiple organ failure and eventually lead to death. Prediction of diabetes due to a few characteristic features or detection of diabetes in early stages can help manage the symptoms effectively. In this paper, the Pima Indians diabetes dataset is used to apply a basic linear classifier model called 'Perceptron'.[1] The perceptron model works best for data with a binary outcome. It finds a hyperplane that separates the number of people with and without diabetes and gives the analysis accordingly. A number of factors are taken into account in training the model with the said dataset. These factors include, the number of pregnancies, Glucose levels, Blood Pressure, Skin Thickness, Insulin levels, BMI, Family History and Age. The model is trained with regards to these factors and whether the patients have diabetes which is the outcome.

The implementation of single layered perceptron is preceded by the cleaning and balancing of the dataset for better results. The implementation is later carried out and is consecutively improved by tuning the hyperparameters like number of iterations and the learning rate.

Evaluation metrics like accuracy, f1-score, recall and precision give insights of the data providing a foundation to predict diabetes in a person.

2. Methodologies

The dataset obtained from kaggle.com is first subjected to cleaning. The process is carried out by first checking for null values in the data set followed by checking for duplicated values. 'Output' column in the dataset is the predicted output of whether a person has diabetes. It is used as a label and all the other columns are considered to be the features. The zero values in the dataset are counted and a few columns like Glucose levels, Blood Pressure, Skin thickness, Insulin and BMI cannot have '0' values as it is medically impossible. These are converted into NaNs.

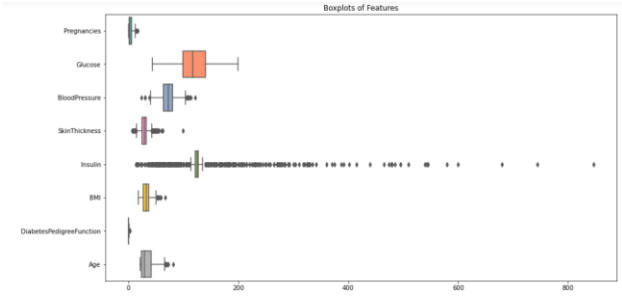

Figure 1: A boxplot showing all the features of the dataset

The NaNs are then replaced using the medians to achieve a clean dataset.

The outcome values are then counted to check the balance of the dataset which can be a major issue if it is

imbalanced. An imbalanced dataset can cause biased predictions.[2]

```
0    500
1    268
Name: Outcome, dtype: int64
```

Figure 2: Dataset before balancing

The balancing of the dataset is done by oversampling the data. Oversampling duplicates the minority class data so that it matches the number of the majority class.[3]

```
0    500
1    500
Name: Outcome, dtype: int64
```

Figure 3: Dataset after balancing

After balancing the dataset, the correlation between features is checked to make sure the features are not highly correlated. High correlation causes redundancy in data which does not align well with the predictions.

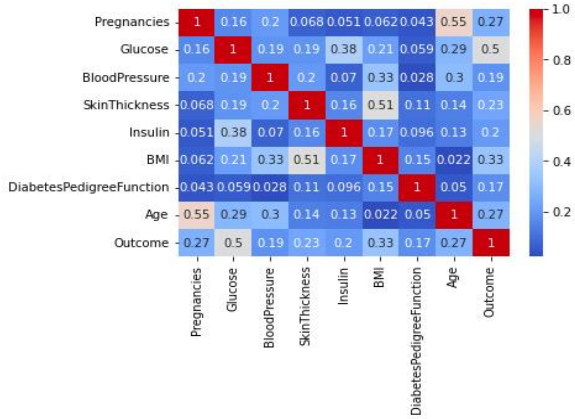

Figure 4: correlation matrix of the dataset

The outliers in a medical dataset is not supposed to be imputed as it can cause the loss of important data.

The balancing of the dataset is followed by the standardization of the data. Standardization transforms the features to have a mean of 0 and a standard deviation of 1. For each feature, standardization is done using the equation,

$$z = \frac{x_i - \delta}{\sigma}$$

Equation 1: Formula for standardization of inputs

Where,

- z is the standardization
- x_i is the value of the feature
- δ is the mean of the feature values
- σ is the standard deviation of the feature values

Below are the scatterplots of the features based on the outcome (positive or negative). The orange plots represent the people positive for diabetes and the blue represents the negative of the same. The diagonal of the scatterplot matrix is the "kernel density estimate" which is a non-parametric way to visualize the feature distribution individually.

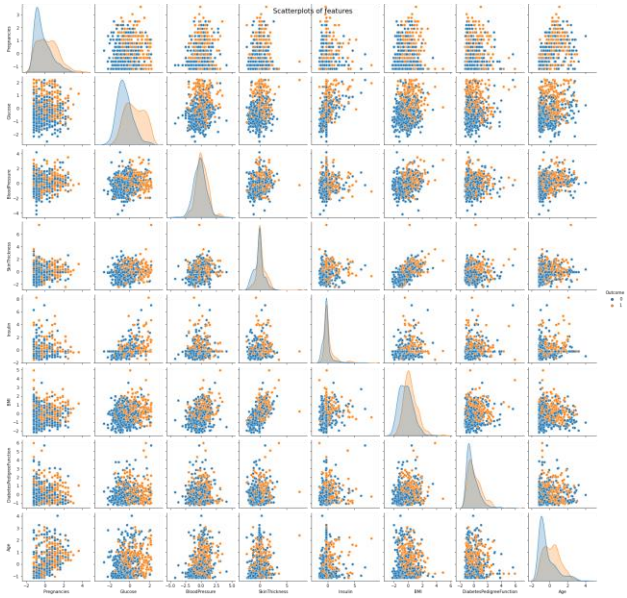

Figure 5: Pairplot between the different features of the dataset

2.1. Implementation of Perceptron

The dataset is split into test data (20%) and train data (80%) in-order to implement the perceptron algorithm. The perceptron algorithm is initiated and implemented based the activation function (step function) because it is a binary output.

$$a = \begin{cases} 0, & if\ x < 0 \\ 1, & if\ x \geq 0 \end{cases}$$

Equation 2: activation function

After initializing the weights and bias the perceptron computes the linear combination of the features and weights plus the bias.

$$z = \sum_{i=1}^{n} (w_i * x_i) + b$$

Equation 3: linear combination of features and weight plus the bias

Where,

- z is the linear output

- w is the weight vector
- x is the feature vector (input)
- b is the bias

This gives the linear output 0 or 1 based on the calculations where, 1 is positive for diabetes and 0 is negative. If the output is between 0 and 1, output greater than or equal 0.5 is considered positive and lesser than 0.5 is considered negative.

Following this, the weights and bias are updated each time the model makes an incorrect prediction. The model learns its mistakes by adjusting these parameters to compare between the predicted and actual labels. This process is iterated several times such that the model gradually learns to adjust the performance on its training data. The update equation for weights is given by, [4]

$$\boldsymbol{w_i = w_i + \mu * (\gamma_{actual} - \gamma_{predicted}) * x_i}[1]$$

Equation 4: weights update function

The update equation for bias is given by,

$$b = b + \mu * (\gamma_{actual} - \gamma_{predicted})$$

Equation 5: bias update function

Where,

- w_i is the weight associated with the feature
- μ is the learning rate
- γ_{actual} is the true label
- $\gamma_{predicted}$ is the predicted label
- x_i is the feature value

The model makes predictions with learning rate set at 0.01 for 1000 iterations.

2.2. Hyper tuning the parameters

Hyper tuning of the parameters refers to the process of finding the best combination of the learning rate and the number of iterations that helps maximize the accuracy of the model.

The model is similar to the basic perceptron algorithm explained in the 2.1 section of this paper. Additionally, a parameter grid consisting different learning rates and iterations is added to the algorithm. Following this, the algorithm searches in loop for the best combination of the learning rates and the iterations to improve accuracy of the model. Each learning rate is run for each of the iterations to find the best combination.

```
parameter_grid = {
    'learning_rate': [0.001, 0.01, 0.1, 1],
    'n_iters': [500, 1000, 2000, 5000]
}
```

Figure 6: parameter grid for hyperparameter tuning

3. Experimental Results

3.1. Analysis before hyperparameter tuning

The perceptron model implemented in this paper yielded the result of 69% using the evaluation metrics such as accuracy, a classification report consisting of

- Precision: The proportion of positive predictions that are actually correct
- Recall: The proportion of actual positive that were correctly predicted
- f1-score: The harmonic mean of precision and recall.

```
Test Accuracy: 69.00%

Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.72      0.70       102
           1       0.69      0.66      0.68        98

    accuracy                           0.69       200
   macro avg       0.69      0.69      0.69       200
weighted avg       0.69      0.69      0.69       200
```

Figure 7: classification report

Confusion matrix consisting of True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN) is depicted below[5].

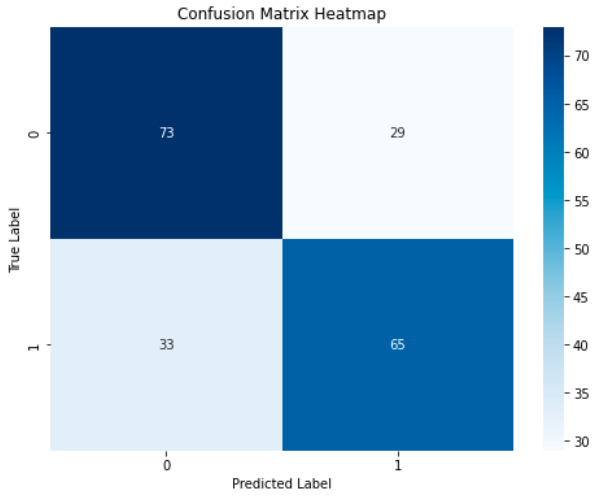

Figure 8: Confusion matrix of the model

From the classification report and the confusion matrix, we can infer that the model's overall accuracy is 69%.

- Precision of 69% is seen on both the classes (diabetic and non-diabetic).
- The balanced precision shows the model is good to predict both diabetic and non-diabetic cases equally.
- The recall suggests that 72% of non-diabetic cases were accurately identified and 66% of the actual diabetic patients were identified by the model.
- The confusion matrix suggests that 65 diabetic patients were correctly classified. 73 non-diabetic patients were correctly classified
- 33 non-diabetic patients were incorrectly predicted as diabetic
- 29 diabetic patients were incorrectly predicted as non-diabetic.

3.2. Analysis after hyperparameter tuning

After subjecting the model to different learning rates with various number of iterations, the best accuracy of the model was seen in the learning rate of 0.001 with 2000 iterations.
This gave the accuracy of 76.5% for the model showing significant improvement.
The confusion matrix showed that

- 77 non-diabetic patients were properly classified
- 76 diabetic patients were correctly classified
- 21 non-diabetic patients were incorrectly classified as diabetic
- 26 diabetic patients were incorrectly classified as non-diabetic

```
Best Hyperparameters: Learning Rate = 0.001, Iterations = 2000
Best Accuracy: 76.5
Confusion Matrix for Best Hyperparameters:
 [[76 26]
 [21 77]]
```

Figure 9: Accuracy of model after tuning hyperparameters

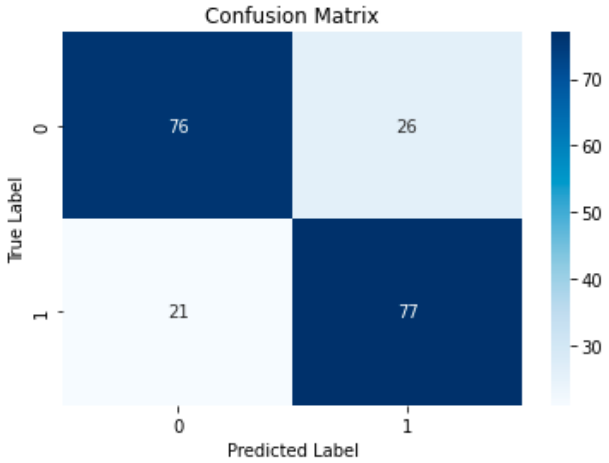

Figure 10: Confusion matrix after hyperparameter tuning

3.3. Inference of both the results

The model after hyperparameter tuning is clearly more reliable. Prediction is a very important aspect of medical field as any wrong answer can lead to late diagnosis or delayed treatment. Hyperparameter tuning clearly improved the model for better.

3.4. Room for improvement

Although the single layer perceptron algorithm in this paper is giving an accuracy of 76.5%, there is still room for improvement. The model can be enhanced using other techniques like regularization etc. Also, we can use a multi-layered perceptron[6] for better results. In the modern world there are better models with greater predictive abilities like Logistic regression, decision tree models, Random Forest classifiers, Support vector machines, K-nearest neighbors, Neural networks etc. which can yield better results.

3.5. Shortcomings

There are several shortcomings for the single layer perceptron model used in our paper.

- The perceptron being a linear classifier assumes all data is linearly separable.
- The perceptron algorithm is very sensitive to outliers. Because this is a medical data, removal of outliers is not possible
- If the data is not linearly separable, the perceptron algorithm oscillates and does not converge on a solution
- The perceptron algorithm is limited to binary classification and cannot handle a multi-class classification problem

3.6. Code

A jupyter notebook (.ipynb file) is used to code this project. The code for this paper is available in this link, https://github.com/tanvijois/Deep_Learning

4. Conclusion

Upon successfully implementing the perceptron algorithm to the Pima Indians diabetes dataset, it yielded an overall accuracy of 76.5%. Reducing false negatives in the medical field is the most critical task and the perceptron model has achieved it in an overall perspective. Although the most basic form of the perceptron algorithm, a simple linear classifier[7] is applied in this paper, it was proven proper pre-processing techniques and fine tuning can improve the model to give better results.

The usage of advanced models like Multi-layered Perceptron, Decision trees, Logistic Regression, Neural Networks, Support Vector Machines, K-Nearest Neighbors algorithms etc. can help classify this dataset more accurately.

References

[1] Singh Jaswinder and Banerjee Rajdeep, *A Study on Single and Multi-layer Perceptron Neural Network*. IEEE, 2019.

[2] N. V Chawla, N. Japkowicz, and A. Ko, "Editorial: Special Issue on Learning from Imbalanced Data Sets." [Online]. Available: http://purl.org/peter.turney/bibliographies/cost-

[3] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results," in *2020 11th International Conference on Information and Communication Systems, ICICS 2020*, Institute of Electrical and Electronics Engineers Inc., Apr. 2020, pp. 243–248. doi: 10.1109/ICICS49469.2020.239556.

[4] Brownlee Jason, "How To Implement The Perceptron Algorithm From Scratch In Python." Accessed: Oct. 06, 2024. [Online]. Available: https://machinelearningmastery.com/implement-perceptron-algorithm-scratch-python/.

[5] E. Beauxis-Aussalet and L. Hardman, "Visualization of Confusion Matrix for Non-Expert Users."

[6] M.-C. Popescu and V. E. Balas, "Multilayer Perceptron and Neural Networks."

[7] "Perceptron." Accessed: Oct. 06, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Perceptron.