Predict Diabetes Using Perceptron

Anonymous CVPR submission

Paper ID

Abstract

The paper is about the implementation of the Perceptron Model to predict the diabetes using a raw medical dataset. This is done on the Pima Indians diabetes dataset which is a binary classification dataset. The perceptron algorithm is a building block for more advance Neural Network. It is designed to classify data into two distinct classes and can be seen as a linear binary classifier. The data is standardized and divided into training and test sets. In hyper-parameters tuning GridSearchCV is applied to tune the model and enhancing its performance.The model is evaluated based on the matrix that is accuracy, precision, recall, F1 score and confusion matrix. The results say that once the perception model is optimized the model is consistent in predicting diabetes. To compare the perceptron model i have used SVM model as it is used for non-linear separable data. The shows that SVM model is more consistent in predicting diabetes.

1. Introduction

Diabetes is a medical condition where millions of individual are effected worldwide. Detecting diabetes earlier was difficult as there were no proper classification model and was making it complicated, making a proper classification model is essential in medical field. Machine learning model have become virtual tool for identifying pattern in medical data that may not immediately identify my professional doctors. These model allow various health factor such as Glucose level and BMI to predict diabetes.

This paper use two machine learning models that is perceptron and SVM for binary classification task to predict diabetes. A perceptron is a linear binary classifier making it efficient in training and evaluating and is best for baseline model. On the other hand SVM can detect non-linear relationship in the data using kernel function giving consistent accuracy where in some cases linear model may under perform.

I compare the performance of the two models focusing on there accuracy, precision, F1 score, recall and confusion matrix. By comparing computational simplicity of perceptron with the SVM capability to capture more complex and non-learn data pattern by doing we can examine model simplicity and predict performance using medical dataset.

2. Methodology

In this assignment, I implemented and compared two supervised machine learning algorithms that is perceptron and SVM for diabetes dataset using Python's Scikit-learn lirary, which provides optimized tools for machine learning. The goal is to see how perceptron model handles the binary classification problem, where the target variable indicates whether or not the individual have diabetes and then compare with SVM model and see how it is better then perceptron model. The steps included in the methodology are data preprocessing, model training, hyper-parameter tuning and evaluating the model performance. This is a visualization of the dataset.

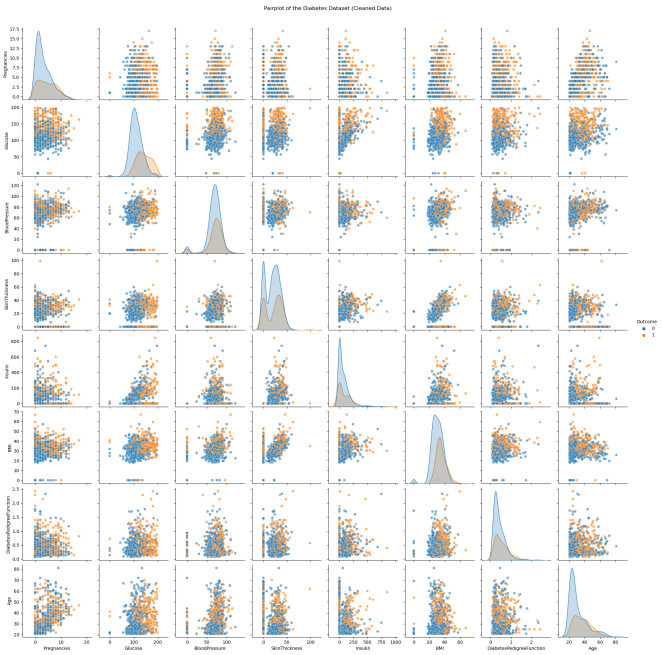

Figure 1. Pairplot of the diabetes dataset

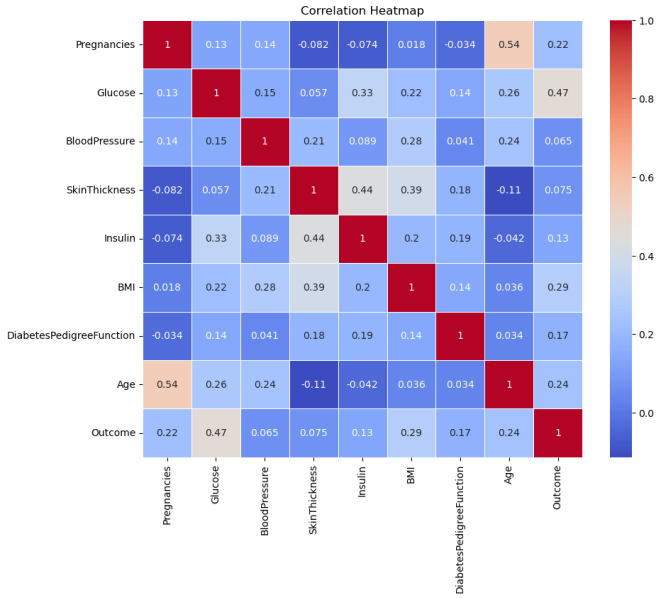

Figure 2. Correlation Heatmap of features

2.1. Data Preprocessing

The dataset consists of several features related to an individual's health, that is glucose levels, BMI and age. The features in the data has a varying scales so standardization is important to make sure each feature contributes eaqually to the learning process. I have use StandardScaler class from Scikit-learn to standardize the input features in the dataset as it ensure all features are equally contributed to the model training as for perceptron it is important because it is sensitive to feature scales. The dataset is split into training and testing sets using 80:20 ratio with the help of train-test-split method to ensure clear separation between the data using for model fitting and evaluation.

2.2. Handling outliers

The outliers in the dataset can affect the performance of machine learning model as it uneven the data. Instead of removing the outliers as it would mean to losing the valuable data I choose to replace it with the median values of the respected column. This is because the median is not effect by extreme values as it make more suitable choice to ensure the data remains balanced. This method helps the model to learn from data which gives better overall patterns and improves its performance.

2.3. Model Implementation

2.3.1 Perceptron Model

It is a building blocks for more advanced Neural Network. It is designed to classify the data into two distinct classes and can be seen as a linear binary classifiers. I have used Sklearn

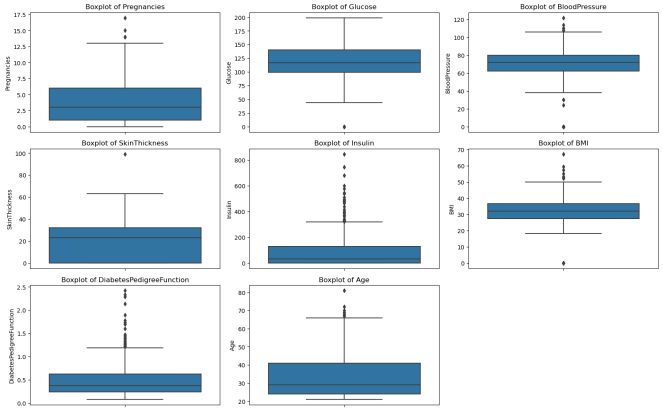

Figure 3. Boxplots of key features(before outlier replaced)

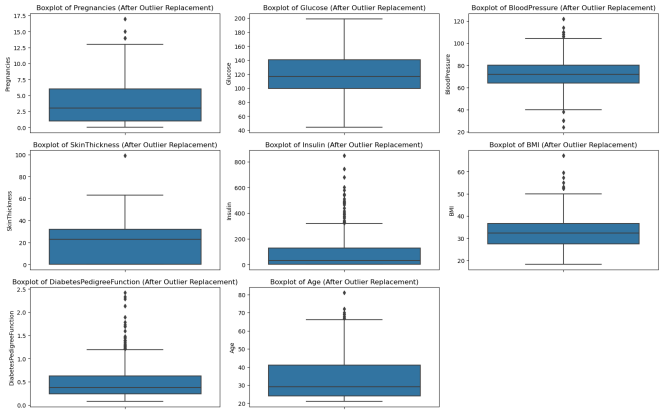

Figure 4. Boxplots of key features(after outlier replaced)

as the perceptron class implements the perceptron algorithm for supervised classification tasks. It maps the input features to an output using an linear decision boundary. It takes a weighted sum of input features and applies it step activation function and predicts either a positive class (1) or negative class (0). The algorithm works by adjusting weights applied to input features based on how far its predictions are from the actual labels. It is done by using a process called error correlation.

2.3.2 Perceptron Algorithm

First, weight initialization is that model start by assigning random initial weights to the input features. Then weighted sum is like for each sample the model computes a weighted sum of that input features.

$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b \qquad (1)$$

where, w - weights, x - features values and b - bias terms. After the weighted sum next comes activation functions where the output of the perceptron is passed through step

function. If the weighted sum (Z) is greater than or equal to 0 the output is positive class(1) or it is negative class(0). After the activation function it come error calculation and weight update where the perceptron calculates the error (difference between predicted and actual lable). If the prediction is incorrect the weights are updated using the following rules

$$w_i = w_i + \eta \cdot (y - \hat{y}) \cdot x_i \tag{2}$$

where, η - learning rate, y - actual label, $\hat{y}$ - predicted label, x_i - input features
This process is repeated for all training samples and weights are updated iteratively until a stopping function (converges or max iteration) is met.

2.3.3 SVM Model

To compare performance SVM was implemented using svc class from Scikit-learn. SVM is a powerful algorithm that works by finding the optimal hyperplane that separates data points of different classes. It supports both linear and non-linear classification using kernel functions. I have applied both the linear and non-linear function(RBF) kernels in this study.

2.3.4 Hyperparameter Tuning

I have used GridSearchCV to perform hyperparameter tuning on both the models. The grid search see multiple combinations of parameter to find the configuration for the best performance. In perceptron we tuned the learning rate and number of iterations.For SVM we tuned the kernel type and C-parameter.

Model	Parameters Tuned	Best Value(s)
Perceptron	Learning rate, Epochs	l2, 0.01, 0.01, 1000, 1e-4
SVM	Kernel (linear, RBF), C-parameter	RBF, 1.0

Table 1. Best Hyperparameters for Perceptron and SVM Models

The following hyperparameters were tuned using a grid search for perceptron:
* The penalty parameter was tested with three regularization techniques L2(Ridge), L1(Lasso), and ElasticNet to control overfitting.
* The alpha parameter which controls the regularization strength was tested with values ranging from 0.00001 to 0.1.
* The eta0 parameter which is set the initial learning rate for weight updates was varied between 0.001 to 1.0.
* The max-iter parameter was fixed to 1000 specifying the maximum number of iterations for model training.
* The tolerance(tol) parameter which is determines the stopping function was tested with the values of 1e-3, 1e-4 and 1e-5 to control convergence sensitivity.
The following hyperparameter were tuned using grid search for SVM:
* c to regularization parameter, tested with values of 0.1, 1, 10 ans 100 to control the trade-off between maximizing the margin and minimizing classification errors.
* Gamma is the kernel coefficient with the options like scale, auto , 0.01 and 0.001 to control the influence of individual data points on the decision boundary.
* Different kernel types were tested that is linear, RBF, polynomil and sigmoid to determine the best fit for the data.
*Degree for the polynomial kernel was 2,3 and 4 to test different levels of complexity.
* Tolerance(tol) is a stoping function that was adjusted to 1e-3 and 1e-4 to control the model convergence.
I can say that during hyperparameter tunning adding regularization significantly improved the performance of perceptron model.

2.4. Model Performance

The model is evaluated on metrics for both the models based on accuracy, precision, recall and F1-score. The tuning perceptron model achieved accuracy of 73% and SVM model achieved high accuracy of 86%. The perceptron remains less intensive and can perform well when the data is mostly lineraly separable.
Confusion matrix for both models(Figure 5 and Figure 6) show how each model is performed based on the true positives, true negative, false positives and false negative as it is important in medical where false negatives should be minimized.

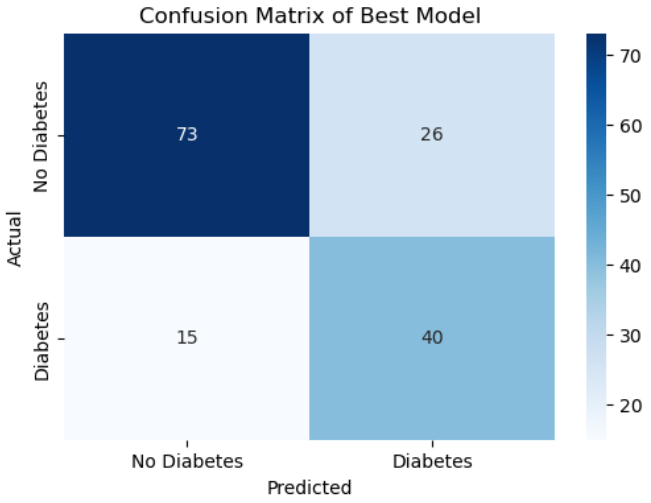

Figure 5. Confusion matrix of perceptron model

From these result SVM has more accuracy compare to perceptron and still perceptron achieves strong results in linear.

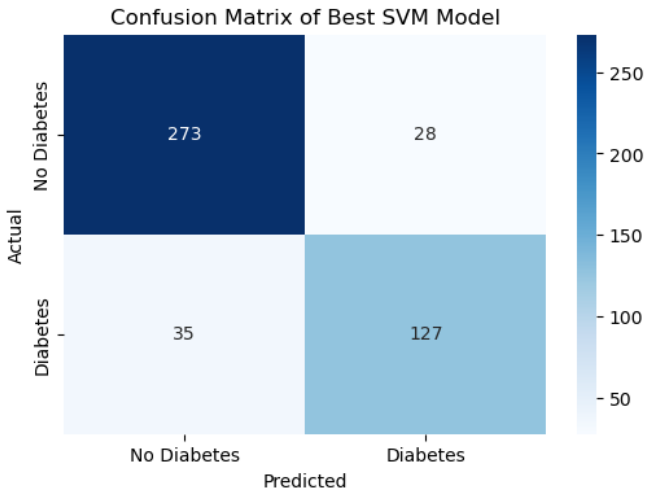

Figure 6. Comfusion matrix of SVM Model

Table 2. Perceptron Model After Hyperparameter Tuning

Parameter	Value
Alpha	0.1
Eta0 (Learning Rate)	0.1
Max Iterations (max_iter)	1000
Penalty	Elasticnet
Tolerance (tol)	0.001
Best Cross-Validation Score	0.7443
Test Accuracy	0.7338
Precision	0.6061
Recall	0.7273
F1 Score	0.6612

Table 3. SVM Model After Hyperparameter Tuning

Parameter	Value
C	100
Degree	2
Gamma	Auto
Kernel	RBF
Tolerance (tol)	0.001
Best Cross-Validation Score	85.62%
Test Set Accuracy	86.39%

SVM higher recall value makes it more suitable for medical application but false negative have to be reduced.

2.5. Conclusion of Results

The perceptron performs well in linear classification. The SVM performs well in non-linear decision boundaries as it gives more accuracy and recall then perceptron. I can say that perceptron is useful for linear separable data and SVM is preferred in more complex cases where non-linear is present.

2.6. Reflection

The goal of this project is to implement perceptron algorithm for predicting diabetes and compare with another model. There were several decision made like dataset, preprocessing techniques and model selection.

2.7. Lessons learned

Throughout the project I learned simpler model like perceptron are useful for basic problems and sophisticated algorithms like SVM achieve more in non-linear dataset. the the process of hyperparameter tuning has more impact on model training as experimenting with different values to choose the best one.

2.8. Future work

The project is on basic linear model classification and it is compared with SVM there are more to explore as:-
* To use more advance techniques like creating new features or using dimention reduction method like PCS will improve the model performance.
* Experimenting with multiple layer perceptron or other neural network architecture could give better prediction performance.
* While the dataset we used is balanced class distribution in future we can use SMOTE techniques to handle imbalanced dataset.

References

[1] Rathod, A.S., Pandey, S., Bharti, K. and Upadhyay, S., 2022. A Diabetes Prediction Model Using Hybrid Machine Learning Algorithm. *International Journal of Intelligent Systems*, [online] Available at: https://www.iieta.org/journals/isi/paper/10.18280/isi.270312.

[2] Heydari, M., Teimouri, M., Heshmati, Z. and Alavinia, S.M., 2016. Comparison of various classification algorithms in the diagnosis of type 2 diabetes in Iran. *International Journal of Diabetes in Developing Countries*, 36(2), pp.167-173.

[3] Gupta, M., 2024. Perceptron Algorithm. *L3 Supervised Learning Part 3* [PDF] Available at: https://myuni.adelaide.edu.au/courses/94658/pages/module-2-online-learning#Breakdown-machine-learning

2.9. Link To github

https://github.com/Anusha0113/Deep-Learning-Fundamentals