Assessment 1:

Predict diabetes using Perceptron

Introduction

machine learning classification is an significant field of artificial intelligence that focuses on classifying content based on its characteristics. It forms the foundation for numerous applications by allowing machines to learn from data, identify patterns and render decisions or forecasts. An example of machine learning is the algorithm of Perceptron introduced by Frank Rosenblatt in 1957. Renowned for its straightforwardness, the algorithm of Perceptron serves as the foundation of many neural network models and holds a significant role in the evolution of deep learning.

The significance of binary classification by perceptrons is particularly noticeable in diagnosing medical conditions. Binary classification helps doctors distinguish between the presence and absence of treatment, facilitating timely and accurate treatment. It is a tool for making decisions for making diagnoses, forecasting the outcome for patients, and tailoring medical treatment, underscoring its vital role in enhancing healthcare.

Past problems

Diabetes represents a worldwide health problem caused by high blood sugar that causes serious damage to the heart, blood vessels, eyes, kidneys and nerves. Correct diagnosis of diabetes is important because early diagnosis and treatment can prevent or delay negative health consequences. Therefore, using machine learning algorithms such as detectors to predict diabetes is very important to solve this global health problem.

For the purpose of this study, we used a specific diabetes case study to implement and analyze the detector algorithm. These data include blood sugar, blood pressure, insulin, BMI, etc. with different targets that indicate the presence or absence of diabetes. It includes many medical parameters such as. By applying a algorithm of Perceptron to this data, this study aims to investigate the algorithm's effectiveness in classifying people according to diabetes risk and demonstrate the potential of machine learning to solve health problems.

Algorithm of Perceptron

The algorithm of Perceptron, developed by Frank Rosenblatt in 1957, is one of the simplest types of artificial neural network and a foundation for more advanced neural network models. It serves as a binary classifier, making it a fundamental tool in machine learning.

Working Principle:

The working principle of the algorithm of Perceptron is straightforward. It takes multiple binary inputs, processes them using learned weights, and produces one binary output. The algorithm is modeled on a single-layer neural network with the following steps:

Initialization: The weights of the input features and the bias are initialized, usually set to zero.

Weighted Sum: Each input xi is multiplied by its corresponding weight wi, and the results are summed up along with the bias.

$z=w1x1+w2x2+...+wnxn+b$

Activation Function: The weighted sum z is passed through an activation function, typically a step function, which returns 1 if z is greater than a certain threshold, and 0 otherwise.

Learning: The algorithm learns by updating the weights and bias based on the error between the predicted output and the actual label. The learning rate determines the size of the weight and bias adjustments.

Binary Classification:

The Perceptron is inherently a binary classifier; it categorizes the input data into one of two classes, labeled as 0 or 1. When applied to a problem like diabetes prediction, the Perceptron helps in determining whether an individual has diabetes (1) or not (0) based on various medical features.

Implementation Details

Python and scikit-learn:

The implementation of the algorithm of Perceptron for this study was carried out using Python, a versatile and widely-used programming language in the field of data science and machine learning. The scikit-learn library, a powerful tool in Python for machine learning and statistical modeling, was employed to access the Perceptron model and various utilities for preprocessing and model evaluation.

Steps for Implementation:

Data Loading and Preprocessing: The diabetes dataset was loaded using pandas, and preprocessing steps such as handling missing values and feature scaling were applied.

Data Splitting: The dataset was divided into training and testing sets using train_test_split from scikit-learn.

Model Initialization and Training: The Perceptron model was initialized and trained on the training data using the fit method.

Model Evaluation: The model's performance was evaluated on the testing set using metrics like accuracy, precision, recall, and F1 score.

Input:

In [5]:
```
import os
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Perceptron, LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
from sklearn.preprocessing import StandardScaler

# Load the Diabetes Dataset
# Get the current working directory
cwd = os.getcwd()

# Construct a file path
file_path = os.path.join(cwd, 'diabetes.csv')
diabetes_df = pd.read_csv(file_path)

# Preprocessing
# Replace zero values with NaN for specific columns
columns_with_zeros = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
diabetes_df[columns_with_zeros] = diabetes_df[columns_with_zeros].replace(0, np.nan)

# Fill NaN values with the mean of the column
diabetes_df.fillna(diabetes_df.mean(), inplace=True)

# Feature Scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(diabetes_df.drop('Outcome', axis=1))
y = diabetes_df['Outcome']

# Split the Data into Training and Testing Sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Initialize and Train the Perceptron Model
perceptron_model = Perceptron(random_state=42)
perceptron_model.fit(X_train, y_train)

# Make Predictions and Evaluate the Perceptron Model
y_pred = perceptron_model.predict(X_test)
print("Perceptron Model Metrics:")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1 Score:", f1_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Initialize and Train a Benchmark Model (Logistic Regression)
benchmark_model = LogisticRegression(random_state=42)
benchmark_model.fit(X_train, y_train)

# Make Predictions and Evaluate the Benchmark Model
benchmark_y_pred = benchmark_model.predict(X_test)
print("\nBenchmark Model (Logistic Regression) Metrics:")
print("Accuracy:", accuracy_score(y_test, benchmark_y_pred))
print("Precision:", precision_score(y_test, benchmark_y_pred))
print("Recall:", recall_score(y_test, benchmark_y_pred))
print("F1 Score:", f1_score(y_test, benchmark_y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, benchmark_y_pred))
```

Output:

```
Perceptron Model Metrics:
Accuracy: 0.6558441558441559
Precision: 0.5217391304347826
Recall: 0.43636363636363634
F1 Score: 0.4752475247524752
Confusion Matrix:
 [[77 22]
 [31 24]]

Benchmark Model (Logistic Regression) Metrics:
Accuracy: 0.7532467532467533
Precision: 0.6666666666666666
Recall: 0.6181818181818182
F1 Score: 0.6415094339622642
Confusion Matrix:
 [[82 17]
 [21 34]]
```

By leveraging Python and scikit-learn, the study successfully implemented the algorithm of Perceptron on the diabetes dataset, laying the groundwork for assessing its classification capabilities and real-world applicability.

Data Preprocessing

Before moving on to model training, an important step involved in the model process is data preprocessing. Diabetes data should be monitored to check for missing results and measure appropriate features:

Remove missing results:

Some features in diabetes data (such as blood sugar, blood pressure, skin thickness, insulin and BMI) are biologically unacceptable which has the value zero. To ensure that the data is more robust and the training model is not affected by uncertainties, these are treated as missing values and replaced by the mean of the corresponding lines.

Feature Scaling:

Given the diversity of features in the dataset, feature scaling is important to standardize the input variables. Features are scaled using a scaling model; this ensures that no particular feature dominates the model due to its size. This step is necessary for algorithms such as perceptrons that are sensitive to the scale of the input.

Model training

Split the dataset:

The basic step in model training is to split the dataset into training and testing sets. In this study, 80% of the data was used for training the model, and the remaining 20% served as the testing set to evaluate the model's performance on unseen data.

Initializing and Training the Perceptron Model:

The Perceptron model was initialized with default parameters, and the training process involved learning the optimal weights and bias that minimize the classification error on the training data. The model iteratively adjusted the weights and bias during training, using the input features to learn and make predictions about the diabetes outcome.

Challenges and Learnings:

Throughout the implementation of the algorithm of Perceptron on the diabetes dataset, several challenges emerged that provided valuable learning opportunities.

Challenges:

Data Quality: The presence of missing or implausible values in the dataset required thoughtful handling to maintain data integrity.

Feature Scaling: The diversity in the range of feature values necessitated meticulous feature scaling to ensure model stability and performance.

Model Tuning: Achieving a balance between model complexity and performance posed challenges, requiring iterative tuning and testing.

Learnings:

Addressing these challenges led to several insights:

Data Preprocessing: The importance of meticulous data preprocessing was underscored, as it directly impacts model reliability.

Algorithm Sensitivity: A deeper understanding of the sensitivity of the algorithm of Perceptron to input scales and parameter tuning was developed.

Performance Metrics: The study reinforced the importance of multiple evaluation metrics to obtain a holistic view of model performance.

Results Assessment:

The Perceptron model demonstrated a commendable ability to classify instances in the diabetes dataset, with an accuracy of 65.58%. However, a critical assessment reveals areas for improvement. The model's precision and recall indicate potential misclassifications, and the F1 Score suggests a need for a better balance between false positives and false negatives. These insights point towards the model's limitations in handling class imbalances and complex feature interactions in the dataset.

Future Work:

To enhance the performance of the Perceptron model, several improvements and modifications are suggested:

Hyperparameter Tuning: Exploration of different learning rates and iterations could optimize the model’s learning process.

Feature Engineering: Developing new features or refining existing ones may improve the model’s ability to capture underlying patterns.

Advanced Algorithms: Investigating more sophisticated algorithms or ensemble methods might yield better classification results.

Conclusion:

This study embarked on implementing and analyzing the algorithm of Perceptron for diabetes prediction, navigating through challenges, and gaining insights into machine learning classification. The results, while promising, highlighted areas for improvement, paving the way for future exploration and optimization. In conclusion, the algorithm of Perceptron shows potential in diagnosis due to its simplicity and importance in neural networks, but its results can be further improved by improving the maintenance and adaptation to the complexity of medical records.

References:

1. Rosenblatt, F. (1958). Perceptron: An example of information storage and organization in the brain. Psychological Review, 65(6), 386-408.

2. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... and Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

3. James, G., Witten, D., Hastie, T., and Tibshirani, R. (2017). Introduction to Education: Applications in R. Springer.

4. Dua, D. and Graf, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California School of Informatics, Computer Science.

5. World Health Organization. (2021). diabetes. Retrieved from https://www.who.int/news-room/fact-sheets/detail/diabetes

6. Fundamentals of Python software. Python Language Reference, version 3.9. Available at http://www.python.org

7. McKinney, W. (2010). Data structure for computation in Python. Proceedings of the 9th Python Science Conference, 51-56.

Closing Statement:

This methodology combines the data presented and the results obtained, organizing the research into a coherent and complete report. Exploring the application of perceptron algorithms to predict diabetes poses the challenge of competition, learning and development time, revealing the potential and diversity of machine learning in solving world health problems.

Github Link: https://github.com/atharva-gangal/Deep_Learning_Assignment_1