Utilizing Machine Learning Techniques in Diabetes Detection

Zeynep Bezeklioglu *†

October 8, 2024

Abstract

Machine learning, a significant field within computer science, offers tools for developing models that can make accurate predictions and assist in decision-making processes. This study investigates the application of machine learning to predict diabetes in patients based on health metrics such as glucose levels, blood pressure, and body mass index (BMI). A logistic regression model, a statistical method suitable for binary classification tasks, was implemented to determine the likelihood of diabetes. The model was trained and tested using patient health data, and its performance was evaluated in predicting diabetic versus non-diabetic cases, achieving a high level of accuracy. The findings demonstrate the effectiveness of machine learning in improving diagnostic precision and supporting healthcare decision-making. This approach has potential benefits for early diabetes detection and personalized treatment, illustrating how machine learning could be applied effectively in healthcare.

1 Introduction

Diabetes is a widespread and significant medical condition affecting millions of individuals globally. Characterized by abnormal glucose levels in the blood, diabetes necessitates timely and accurate diagnosis to mitigate potential complications. Traditionally, the diagnosis of diabetes involves a combination of blood tests and clinical assessments. In recent years, there has been an increasing integration of machine learning into healthcare, encompassing applications such as predictive analytics and personalized medicine. By analyzing large volumes of patient data, machine learning algorithms can identify complex patterns and risk factors that traditional diagnostic methods may overlook. These algorithms enhance early detection by predicting an individual's likelihood of developing diabetes based on specific risk factors. Machine learning enables systems to learn autonomously from data, reducing reliance on explicit programming.

*Student at Bahcesehir College Nakkastepe Science and Technology High School, Istanbul
†Advised by: Guillermo Goldsztein, Georgia Institute of Technology

This research employs machine learning techniques, particularly supervised learning, to construct a computational model aimed at diagnosing diabetes. Supervised learning involves training models on labeled data, where each instance includes an expected outcome. Utilizing a dataset obtained from Kaggle—a platform recognized for its diverse array of datasets—a computational framework specifically designed for diabetes diagnosis has been developed.

This article is organized as follows: First, the dataset utilized in this study is described, detailing its features and labels. Next, an overview of supervised learning is provided, emphasizing its relevance within the broader context of machine learning applicable to this research. Subsequently, logistic regression models are examined, focusing on their utility in predicting the presence of diabetes. Additionally, the binary cross-entropy loss function, which is critical for assessing model performance, is addressed. Following this, the coding process used for model implementation and training is detailed, including the procedure for splitting the dataset into training and validation sets to ensure effective model training and accurate evaluation. Finally, potential future directions for this research are outlined.

2 Overview of the Diabetes Dataset

The features of this dataset consist of the number of pregnancies, glucose, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function, and age. The label is the conclusion: does the patient have diabetes or not. 0 indicates that the patient does not have diabetes while 1 indicates that the patient does have diabetes. Because there are two different outcomes for the label (0 and 1), this problem is known as a binary classification problem. The table represents the first five rows of the dataset for diabetes patients. The dataset contains 768 patients.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Figure 1: First five rows of the dataset for diabetes patients

3 Supervised Learning

This research uses a supervised learning approach, where models are trained on datasets that include both input features and their corresponding labels. In

the context of this study, the dataset comprises patient information, with each patient serving as an individual example.

Supervised learning problems are characterized by their dual data types: the label or target variable, which represents the outcome to be predicted, and the features, which are the descriptive attributes of each example. In this study, the label is the presence or absence of diabetes, while the features include various health metrics.

The main aim is to create a computational model capable of predicting the likelihood of diabetes in patients who are not part of the existing dataset. To accomplish this, the model will leverage the features of patients to estimate their diabetes risk.

4 Materials and Methods

4.1 Binary Classification Problem

In the context of diabetes prediction, each patient is classified as either having diabetes or not. This means that the label associated with each patient takes one of two values: 1 if the patient has diabetes and 0 if the patient does not. This type of problem, where the label takes one of two possible values, is known as a binary classification problem. Each example, or patient in this case, belongs to one of two categories based on their label.

A model designed for binary classification problems operates by taking the features of an example as input and outputting a number between 0 and 1. This number, which we denote as $\hat{y}$, represents the model's prediction of the example's label. Importantly, $\hat{y}$ is a function of the example's features.

4.2 Logistic Regression

In this study, logistic regression was employed as the primary method for classification tasks. Logistic regression was preferred due to the requirement for a larger number of examples in neural networks to mitigate the risk of over-fitting and achieve accurate predictions. Given the available data, logistic regression offered a more effective approach for this research.

Logistic regression is a statistical model utilized for binary classification, where the objective is to predict a binary outcome (e.g., yes/no, true/false). The model predicts the probability that a given input belongs to a specific class.

It is defined by the following sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Figure 2 displays the graph of the sigmoid function:

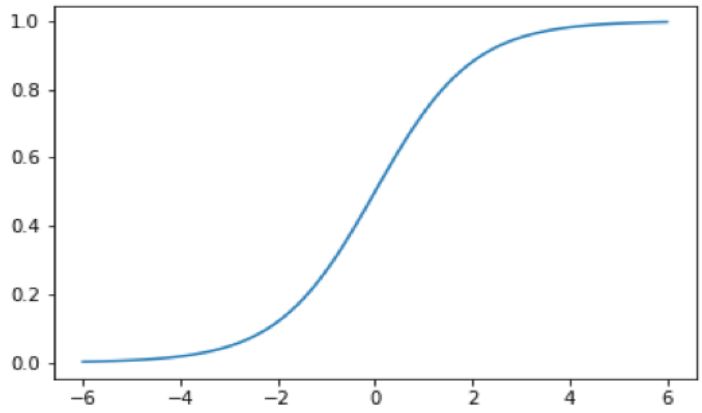

Figure 2: A graph of the sigmoid function

Key properties of the logistic regression equation are as follows:

- It converts any real-valued input x into a probability between 0 and 1.
- The output is always between 0 and 1. As x approaches positive infinity, $\sigma(x)$ approaches 1. As x approaches negative infinity, $\sigma(x)$ approaches 0.
- At x=0, the function output is exactly 0.5, marking the midpoint of the curve.
- The sigmoid function is an increasing function, meaning that as x increases, the output also increases, but it never decreases.

When using logistic regression, our input to the model is a linear combination of the features. Specifically, the prediction $\hat{y}$ is calculated as follows:

$$\hat{y} = \sigma(w_1x_1 + w_2x_2 + \ldots + w_{12}x_{12} + b)$$

where:

- $x_1, x_2, \ldots, x_{12}$ are the 12 features of each example.
- $w_1, w_2, \ldots, w_{12}$ and b are the parameters of the model.
- σ denotes the sigmoid function, which maps the linear combination of features to a probability between 0 and 1.

As we delve deeper into the mechanics of logistic regression, it is essential to understand the role of the parameters in the model. The parameters $w_1, w_2, \ldots, w_{12}$ and b determine the model's predictions. These parameters are not predefined; instead, they are learned from the training data. The goal of training a logistic regression model is to find the set of parameters that minimizes the binary cross-entropy error on the training set. This error measure evaluates how well the model's predictions match the actual class labels.

4.3 Binary Cross Entropy Error

Binary cross-entropy (BCE) is a measure used to evaluate the performance of a binary classification model. It quantifies how well the model's predicted probabilities match the actual binary labels. Here's how it works:

Consider an example with features $x_1, x_2, \ldots, x_{12}$. Suppose the true label for this example is y, which can either be 0 or 1. The model predicts the probability $\hat{y}$ that the label is 1, where $0 < \hat{y} < 1$. If the predicted probability ($\hat{y}$) is close to 0, it means the model predicts a low likelihood of having diabetes, suggesting that the individual is unlikely to have the condition; conversely, if $\hat{y}$ is close to 1, it indicates a high likelihood of having diabetes, suggesting that the individual is more likely to have the condition.

The binary cross-entropy error for this prediction is calculated using the formula:

$$BCE(y, \hat{y}) = -(y \, log \, (\hat{y}) + (1 - y) \, log \, (1 - \hat{y}))$$

The properties of Binary Cross-Entropy Error are as follows:

1. Non-negative:
   - The binary cross-entropy error is always greater than or equal to 0: $\text{BCE}(y, \hat{y}) \geq 0$.
2. Zero Error for Perfect Prediction:
   - If the predicted probability $\hat{y}$ exactly matches the actual label y (i.e., $\hat{y} = y$), then the BCE error is 0: BCE (y, $\hat{y}$) = 0.
3. Error Decreases as Prediction Approaches Actual Label:
   - The closer the predicted probability $\hat{y}$ is to the actual label y, the smaller the binary cross-entropy error. In other words, the model's performance improves as its predictions more closely align with the true labels.

4.4 Code Overview

The Python programming language was used for this project. To implement and test the algorithms, Google Colab was employed as the development environment.

4.4.1 Importing Libraries

The process started with importing necessary libraries. Pandas manages and manipulates the dataset, while NumPy handles mathematical operations on arrays. Sklearn scales the data and prepares it for analysis. TensorFlow and Keras run the core algorithms to find the best predictors. Matplotlib's Pyplot and Seaborn were used for visualizing the results.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.losses import BinaryCrossentropy
from sklearn.metrics import classification_report
```

Figure 3: Import statements for accessing various models

4.4.2 Training and Validation Set

To develop the logistic regression model, the dataset was divided into two distinct subsets: the training set and the validation set. The training set consists of examples used for building and training the model, including both the features and the corresponding labels. The validation set, on the other hand, serves as a separate subset for evaluating the model's performance, allowing for the assessment of its generalization to new, unseen data.

In this project, 75% of the dataset was allocated for training, while the remaining 25% was reserved for validation. This split was performed randomly to ensure that both subsets are representative of the overall dataset. Each example had a 75% probability of being assigned to the training set and a 25% probability of being included in the validation set. Both the training and validation sets consist of features and labels, ensuring that the model is adequately trained and tested.

```
[ ] X = df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness' , 'Insulin' ,'BMI', 'DiabetesPedigreeFunction', 'Age']].to_numpy()
    Y = df['Outcome'].to_numpy()
    X_train, X_val, Y_train, Y_val = train_test_split(X,Y,test_size=0.25,random_state=42)
    scaler = preprocessing.StandardScaler()
    scaler.fit(X_train)
    X_train_scaled = scaler.transform(X_train)
    X_val_scaled = scaler.transform(X_val)
```

Figure 4: Code used for dividing the dataset into training and validation sets

In this process, the X values, which represent the features, and the Y values, which represent the outcomes, are taken from the data frame and converted into NumPy arrays. The "train_test_split" function is used to split the data into training and validation sets, with 25% of the data allocated for validation (`test_size=0.25`) and a random seed of 42 (`random_state=42`) to ensure reproducibility.

```
[ ]  model = Sequential()

     model.add(Dense(7, activation='relu'))
     model.add(Dense(1, activation='sigmoid'))

     model.compile(loss='binary_crossentropy', metrics=['accuracy'])
     history = model.fit(X_train_scaled, Y_train, epochs=100, verbose=0, validation_data=(X_val_scaled, Y_val))

     J_list = history.history['loss']
     plt.plot(J_list)
```

Figure 5: Code for activating and training the model

The model is then trained using both the training and validation datasets. It includes a hidden layer with a ReLU activation function and an output layer with a sigmoid activation function.

The ReLU function in the hidden layer allows the model to learn complex patterns, while the sigmoid function in the output layer provides probability estimates for classification. Training the model adjusts its weights to improve accuracy.

	accuracy	loss	val_accuracy	val_loss
0	0.619792	0.750813	0.625000	0.720419
1	0.645833	0.727485	0.645833	0.700196
2	0.661458	0.708239	0.661458	0.681552
3	0.678819	0.690445	0.682292	0.664435
4	0.682292	0.673535	0.682292	0.647733
..	...	...	...	...
195	0.796875	0.413218	0.760417	0.539182
196	0.788194	0.413117	0.760417	0.539500
197	0.788194	0.412890	0.760417	0.539854
198	0.791667	0.413053	0.760417	0.540086
199	0.789931	0.412865	0.760417	0.540187

Figure 6: Model performance metrics: loss and accuracy

The model's performance is measured by accuracy, which shows how often it makes correct predictions. Accuracy is found by dividing the number of correct predictions by the total number of predictions, giving a number between 0 and 1. A higher number means the model is better, while a lower number means it is not as good. The model had an accuracy of 0.789931 on the training set and 0.760417 on the validation set after 200 epochs.

5 Discussion

The results of this research show that supervised learning and logistic regression can effectively diagnose diabetes, achieving a model accuracy of 76%. This is

a significant improvement over random guessing, which would only yield 50% accuracy. These findings indicate that machine learning can be a valuable tool in healthcare.

This study supports existing research on the importance of data-driven methods in medicine. It confirms that using artificial intelligence, such as machine learning, can improve diagnostic processes and aligns with the broader trend of applying these technologies to healthcare challenges.

To further improve the model's performance, several steps can be taken. First, exploring more advanced techniques like neural networks could help uncover more complex patterns in the data, potentially boosting accuracy. Second, incorporating additional features, such as genetic markers or lifestyle information, could lead to a more comprehensive and accurate model. These enhancements could ultimately result in earlier and more precise diabetes diagnoses, improving patient outcomes.

References

[HJLS13] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant. *Applied Logistic Regression.* John Wiley & Sons, 2013.

[KS17] R. Kumari and S. K. Srivastava. Machine learning: A review on binary classification. *International Journal of Computer Applications*, 160(7), 2017.

[KZP07] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas. Supervised machine learning: A review of classification techniques. *Emerging Artificial Intelligence Applications in Computer Engineering*, 160(1):3–24, 2007.

[MG16] A. C. Müller and S. Guido. *Introduction to machine learning with Python: a guide for data scientists.* O'Reilly Media, Inc., 2016.

[Mur12] K. P. Murphy. *Machine learning: a probabilistic perspective.* MIT Press, 2012.

[SV20] M. Soni and S. Varma. Diabetes prediction using machine learning techniques. *International Journal of Engineering Research & Technology (IJERT)*, 9(09):2278–0181, 2020.