

ML Assignment 2

Diabetic Prediction Model using Logistic Regression



IT19043456 – KULATILAKE T. T

IT18118896 – EISHAN DINUKA W. H. A.

IT19022734 – LIYANAGE P. L. R. S.

IT19043524 – DEEMUD G. H. K

May 27, 2022

Table of Contents

[Problem Definition 2](#_Toc104213620)

[Dataset Description 3](#_Toc104213621)

[Methodology 4](#_Toc104213622)

[Results and Discussion 5](#_Toc104213623)

[Individual Contribution 6](#_Toc104213624)

[References 7](#_Toc104213625)

[Appendix 8](#_Toc104213626)

# Problem Definition

Diabetes mellitus is a collection of disorders that impact your body's ability to use blood sugar (glucose). Because glucose is a significant source of energy for the cells that make up your muscles and tissues, it is essential to your health. It's also the primary source of energy for your brain. The underlying cause of diabetes differs depending on the kind. However, regardless of the type of diabetes you have, it can cause an excess of sugar in your blood. Blood sugar levels that are too high can cause major health concerns [1].

In this mini project, we will be predicting whether an individual is a diabetic patient or not based on their Glucose level, Blood pressure, BMI and Age. As the machine learning model, we will be using a supervised machine learning algorithm known as logistic regression model.

This statistical model (also known as the logit model) is frequently used in classification and predictive analytics. Based on a collection of independent variables, logistic regression calculates the likelihood of an event occurring, such as in this case, has diabetic or not. The dependent variable is confined between 0 and 1 because the outcome is a probability. A logit transformation is performed to the odds in logistic regression, which is the probability of success divided by the probability of failure. This logistic function is represented by the following formulas, which are also known as log odds or the natural logarithm of odds [2].

There are three types of logistic regression models.

1. Binary Logistic Regression

2. Multinomial Logistic Regression

3. Ordinal Logistic Regression

# Dataset Description

As the dataset, we used a publicly available dataset from Kaggle which represents the features that can be used to predict a diabetic patient and non-diabetic person. The dataset is available in Kaggle as “Pima Indians Diabetes Database [3]”.

The National Institute of Diabetes and Digestive and Kidney Diseases provided this data. The dataset's goal is to diagnose whether a patient has diabetes using diagnostic metrics included in the collection. The selection of these cases from a wider database was subjected to several limitations. All of the patients at this clinic are Pima Indian women who are at least 21 years old [3].

There are various medical predictor factors in the dataset, as well as one goal variable, Outcome. The number of pregnancies the patient has had, their BMI, insulin level, and age are all predictor variables [3]. Dataset contains 779 records.

# Methodology

Our mini project consists of 7 parts which are as follows.

1. Importing libraries and observing the dataset.

2. Analyzing the data.

3. Data preprocessing.

4. Building the model.

5. Testing the performance of the model.

6. Improving the model.

## Importing libraries and observing the dataset.

As the first step, we must import the necessary libraries and the dataset in order to prepare the data for processing. In our project, Outcome 0 stands for a non-diabetic patient and Outcome 1 stands for a diabetic patient.

Here, we will be importing pandas, numpy, seaborn and pylot from matplotlib python libraries. The reason for importing seaborn library is it provides high-level interface for drawing attractive and informative statistical graphics. It is based on matplotlib library. Similarly, we use pylot which is from matplotlib library for creating a plotting area to plot some lines and labels.

After that, we will be importing the dataset which we have obtained from Kaggle. If we output the length of the dataset, it will show as 778 by executing len(diabetes\_data), and also we check the index range in the dataset by diabetes\_data.index.

Then we list the column names in the dataset by executing diabetes\_data.columns and we will obtain an output listing all the available columns in the dataset. After that, we are going to explain what each column describes by executing diabetes\_data.info() method. It will describe what type of data each column contains in the dataset. Also we can list the datatypes of the column separately by executing diabetes\_data.dtypes.

After that, we describe the entire dataset to see what information it contains. For that, we should execute diabetes\_data.describe() method.

## Analyzing the Data.

As the next step, we will begin the analysis part of the data in the dataset. First, we will plot the data of how many people have diabetes or not based on the existing data in the dataset. For that, we will use the countplot method in “seaborn” library which we imported earlier. We can execute sns.countplot(x=’Outcome’, data=diabetes\_data). Here, for ‘X’, we will give what information should be displayed in the X axis of the graph. It should be one of the column names of the dataset.

According to the output, we can see that most of the people are in the “non-diabetic” category (images of the results are included in the Appendix section of this document).

Then we want to find out the diabetic and non-diabetic people according to their Age. In that case, we use jointplot method in the same “seaborn” library. We executes sns.jointplot(x=’Age’,y=’Outcome’, data=diabetes\_data) and it will show another graph which contains the X axis as the Age of the people and Y axis as the Outcome. According to the results, we can see that younger and middle age people are mostly getting into diabetes.

Then by following the first approach, we will again use the countplot method from “seaborn” library to plot a graph to find out how many diabetic and non-diabetic people based on their Gender. For that, we will execute sns.countplot(x=’Outcome’, data=diabetes\_data, hue=’Sex’) code. According to the results, we can see that among the non-diabetic people, majority are male and also among diabetic people, majority are male. That concludes the analysis part of the dataset.

## Data Preprocessing.

Next is a crucial step of this mini project which is the Data Preprocessing part. The reason for being a crucial step is because, neglecting data preprocessing might cause so many errors in the output and also it adversely affects the accuracy of the model.

First, we check if there are any null values in the dataset. For that, we will use diabetes\_data.isna() command. After executing the command, we witnessed some null values. Then we listed out how many null values can be seen based on each column. For that we executed diabetes\_data.isna().sum() command and according to the output, we can see that there are 10 null values present under “Age” column and no null values present in other columns.

Then we visualize the null values by using heatmap method from “seaborn” library by executing sns.heatmap(diabetes\_data.isna()) command. Then we can barely see the null values in the “Age” column.

After that, we plotted a distribution for the age column to see whether the younger people or older people are prominent in the dataset. For that, we used displot in the same “seaborn” library. We executed sns.displot(x=’Age’, data=diabetes\_data) command. According to the output, we can see that most of the records are younger people with an age around 20 years.

Then in order to get rid of the null values, we will fill those null values with the mean value of the Age column. For that, we executed diabetes\_data[‘Age’].fillna(diabetes\_data[‘Age’].mean(), inplace=True) command. After that, we again listed out how many null values present after the alteration using the same command used previously and we received an output as 0 which means there are no null values present.

Then we again check if there are any empty values in the dataset by executing diabetes\_data.tail() command and we didn’t witnessed any empty values which means we removed null values successfully from the dataset.

After that, we checked whether there are any non-numeric data available in the dataset. By analyzing, we found out that, “Name” and “Sex” columns contains non-numeric values. Since "Name" column is a non-numeric type, and it is not useful for this Machine learning model, so we removed it. But since "Sex" column is a non-numeric, but it can be useful to train this machine learning model, we will convert it into numeric. In order to convert it into numeric, we executed gender=pd.get\_dummies(diabetes\_data[‘Sex’], drop\_first=True) command and then created a new column called ‘Gender’ with those values. 1 represents Male and 0 represents Female.

Then we removed the ‘name’ column by executing diabetes\_data.drop([‘Name’], axis=1, inplace=True) command and also dropped the ‘Sex’ column since we have a new column called ‘Gender’ with numeric values.

As the final step of the data preprocessing, we extracted only the necessary columns and assigned them into a new “diabetes\_data” variable

## Building the Model.

Now the interesting part begins, which is building and training the model. As the first step, we separated the columns in the dataset into two arrays containing dependent variables (Y axis) and independent variables (X axis).

As the dependent variable, “Outcome” column will be taken and as independent variables (X-axis), all other remaining variables will be taken.

Then we imported train\_test\_spliit package from sklearn.model\_selection library in-order to split the dataset into training and testing datasets. Size of the training dataset will be 70% and size of the testing dataset will be 30%.

Then we finally imported our Logistic Regression model from sklearn.linear\_model library. After that we started to train out model with the training data by executing model.fit(x\_train, y\_train) command.

After the training is completed, we generated an accuracy score for the test dataset by executing model.score(x\_test, y\_test) command and we received an accuracy score of 0.7665369649805448 which is roughly 77% accuracy which is good.

# Results and Discussion

# Individual Contribution

# References

# Appendix