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

Table of Figures

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.

Testing the performance of the model.

After successfully training the model, now its time to test the performance of the model. For this we used the confusion metrics provided by *"sklearn"* library. First of all, we imported the confusion from *sklearn.metrics.* Then we used the *"pandas"* library to check the confusion metrics and we executed *pd.crosstab(y_test,predict,rownames=["Actual Label"], colnames=["Predicted Label"])* command.

After that, we imported the classification report which can also be retrieved from the *"sklearn"* library. Then we retrieved a table of information containing precision, recall, f1-score and support. The ratio of accurately anticipated positive observations to the total predicted positive observations is known as precision. The ratio of accurately predicted positive observations to all observations in the actual class is known as recall and the weighted average of Precision and Recall is the F1 Score

Improving the Model.

As the final steps of this mini project, we took a step to improve this model by tuning some hyperparameters of this logistic regression model. For that, we used *GridSearchCV* provided by the *"sklearn"* library.

After doing some tuning, we received a new improved accuracy score as 0.780 which is **78%** which is a slight improvement.

Results and Discussion

In the end according to the testing results, we got a prediction accuracy level of 77% for this model, which is good to improve the model further we need to provide more data into the dataset, and we also can tryout this logistic regression analysis by using the other machine learning models available in the sci-kit-learn library too.

In the classification report, we can see good amounts in the precision which is 0.79 for predicting a patient as not a diabetic patient accurately and 0.71 for predicting a patient as a diabetic patient accurately.

Finally, we improved the hyperparameters of the logistic regression model and it improved the model a furthermore and gave an accuracy level of **78%** over the previous value, 77%.

Individual Contribution

Student No.	Name	Contribution
IT19043456	Kulatilake T. T.	Importing libraries and observing the dataset. **(Code + Report)**
IT18118896	Eishan Dinuka W. H. A.	Testing the performance of the model and improving the model. **(Code + Report)**
IT19022734	Liyanage P. L. R. S.	Data preprocessing and building the model. **(Code + Report)**
IT19043524	Deemud G. H. K.	Analysis of the data. **(Code + Report)**

References

[1] M. C. staff, "Diabetes," [Online]. Available: https://www.mayoclinic.org/diseases-conditions/diabetes/symptoms-causes/syc-20371444#:~:text=Diabetes%20dramatically%20increases%20the%20risk,Nerve%20damage%20(neuropathy)..

[2] I. staff, "What is logistic regression," [Online]. Available: https://www.ibm.com/topics/logistic-regression.

[3] kaggle.com, "Pima Indians Diabetes Database," [Online]. Available: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database.

Appendix

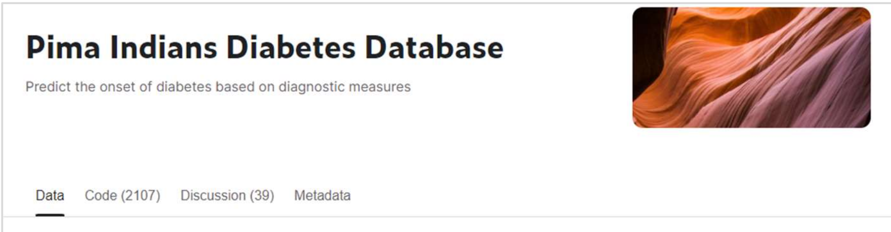

Figure 0.1: Pima Indians Database

	A	B	C	D	E	F	G	H	I	J	K
1	Name	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Sex	Outcome
2	Braund, Mr. Owen Harris	6	148	72	35	0	33.6	0.627	50	male	1
3	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	1	85	66	29	0	26.6	0.351	31	female	0
4	Heikkinen, Miss. Laina	8	183	64	0	0	23.3	0.672	32	female	1
5	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	89	66	23	94	28.1	0.167	21	female	0
6	Allen, Mr. William Henry	0	137	40	35	168	43.1	2.288	33	male	1
7	Moran, Mr. James	5	116	74	0	0	25.6	0.201	30	male	0
8	McCarthy, Mr. Timothy J	3	78	50	32	88	31	0.248	26	male	1
9	Palsson, Master. Gosta Leonard	10	115	0	0	0	35.3	0.134	29	male	0
10	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	2	197	70	45	543	30.5	0.158	53	female	1
11	Nasser, Mrs. Nicholas (Adele Achem)	8	125	96	0	0	0	0.232	54	female	1
12	Sandstrom, Miss. Marguerite Rut	4	110	92	0	0	37.6	0.191	30	female	0
13	Bonnell, Miss. Elizabeth	10	168	74	0	0	38	0.537	34	female	1
14	Saundercock, Mr. William Henry	10	139	80	0	0	27.1	1.441	57	male	0
15	Andersson, Mr. Anders Johan	1	189	60	23	846	30.1	0.398	59	male	1
16	Vestrom, Miss. Hulda Amanda Adolfina	5	166	72	19	175	25.8	0.587	51	female	1
17	Hewlett, Mrs. (Mary D Kingcome)	7	100	0	0	0	30	0.484	32	female	1
18	Rice, Master. Eugene	0	118	84	47	230	45.8	0.551	31	male	1
19	Williams, Mr. Charles Eugene	7	107	74	0	0	29.6	0.254	31	male	1
20	Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)	1	103	30	38	83	43.3	0.183	33	female	0
21	Masselmani, Mrs. Fatima	1	115	70	30	96	34.6	0.529	32	female	1
22	Fynney, Mr. Joseph J	3	126	88	41	235	39.3	0.704	27	male	0
23	Beesley, Mr. Lawrence	8	99	84	0	0	35.4	0.388	50	male	0
24	McGowan, Miss. Anna "Annie"	7	196	90	0	0	39.8	0.451	41	female	1
25	Sloper, Mr. William Thompson	9	119	80	35	0	29	0.263	29	male	1
26	Palsson, Miss. Torborg Danira	11	143	94	33	146	36.6	0.254	51	female	1
27	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia Johansson)	10	125	70	26	115	31.1	0.205	41	female	1

Figure 0.2: Dataset Structure

	Name	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Sex	Outcome
0	Braund, Mr. Owen Harris	6	148	72	35	0	33.6	0.627	50.0	male	1
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	85	66	29	0	26.6	0.351	31.0	female	0
2	Heikkinen, Miss. Laina	8	183	64	0	0	23.3	0.672	32.0	female	1
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	89	66	23	94	28.1	0.167	21.0	female	0
4	Allen, Mr. William Henry	0	137	40	35	168	43.1	2.288	33.0	male	1
...	...	...	...	...	...	...	...	...	...	...	...
773	Elias, Mr. Dibo	10	101	76	48	180	32.9	0.171	NaN	male	0
774	Hocking, Mrs. Elizabeth (Eliza Needs)	2	122	70	27	0	36.8	0.340	NaN	female	0
775	Myhrman, Mr. Pehr Fabian Oliver Malkolm	5	121	72	23	112	26.2	0.245	NaN	male	0
776	Tobin, Mr. Roger	1	126	60	0	0	30.1	0.349	NaN	male	1
777	Emanuel, Miss. Virginia Ethel	1	93	70	31	0	30.4	0.315	NaN	female	0

778 rows × 11 columns

Figure 0.3: Dataset describe

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 778 entries, 0 to 777
Data columns (total 11 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Name                      778 non-null    object 
 1   Pregnancies               778 non-null    int64  
 2   Glucose                   778 non-null    int64  
 3   BloodPressure             778 non-null    int64  
 4   SkinThickness             778 non-null    int64  
 5   Insulin                   778 non-null    int64  
 6   BMI                       778 non-null    float64
 7   DiabetesPedigreeFunction  778 non-null    float64
 8   Age                       768 non-null    float64
 9   Sex                       778 non-null    object 
 10  Outcome                   778 non-null    int64  
dtypes: float64(3), int64(6), object(2)
memory usage: 67.0+ KB
```

Figure 0.4: Explaining each column

```
Name                         object
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                         float64
Sex                          object
Outcome                       int64
dtype: object
```

Figure 0.5: data types

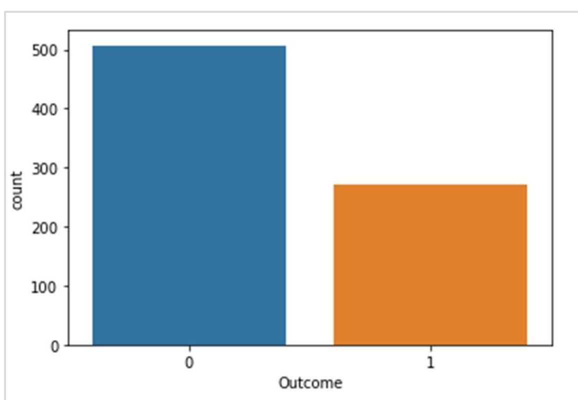

Figure 0.6: countplot of having diabities or not

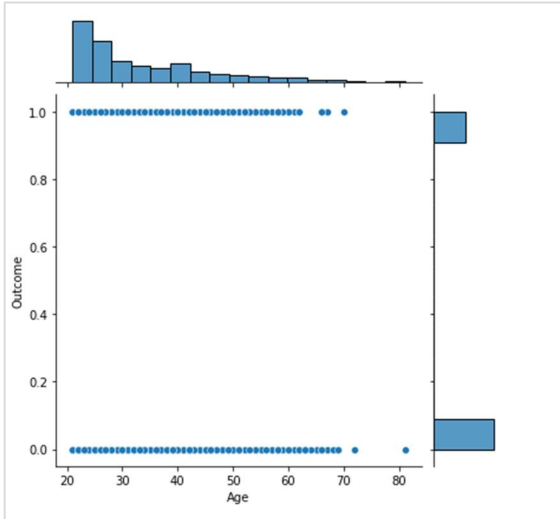

Figure 0.7: Jointplot which shows diabeties status vs Age

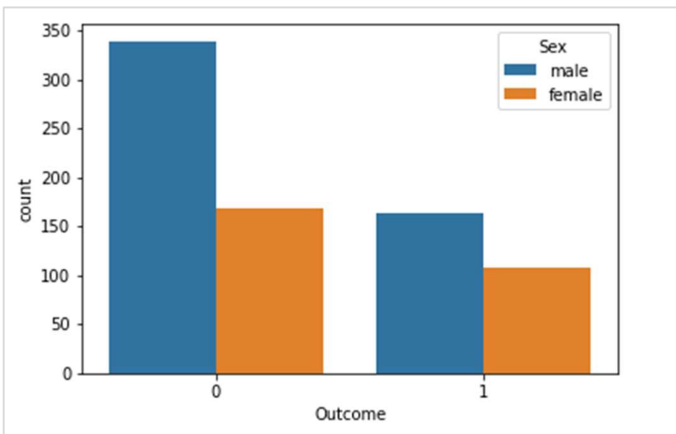

Figure 0.8: Male vs Female has diabetes

```
Name                        0
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                        10
Sex                         0
Outcome                     0
dtype: int64
```

Figure 0.9: 10 null values in Age column

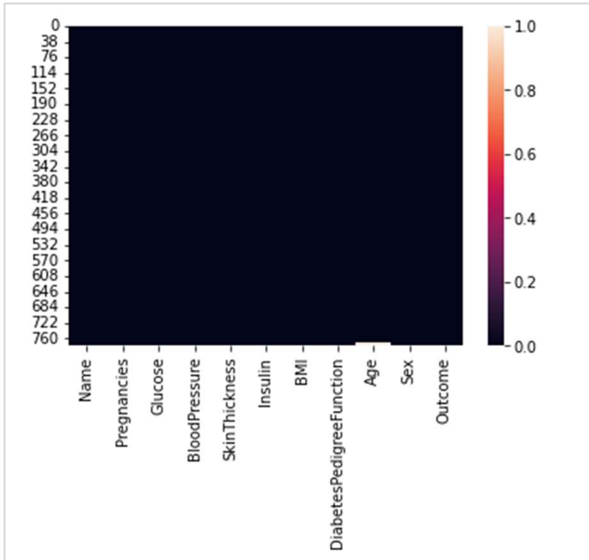

Figure 0.10: visualize null values

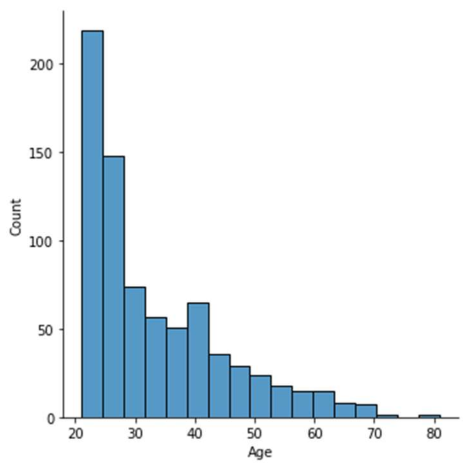

Figure 0.11: The distribution for the age column