Saif Ali Rahman

Ub number: 18010207

Logistic regression approach for diabetes detection

**Abstract:**

This research report will consist of an analysis of the Pima diabetes data set along with an Ai demonstration using the logistic regression approach. This report will provide a system that can help diagnose people with diabetes according to various parameters and do so accurately. The proposed system will be using a logistic regression approach and trained using the Pima diabetes dataset provided by the UCI Machine Learning Data Set Repository.

29th November 2021

1. Introduction

Diabetes is a disease that occurs when your blood glucose levels (also known as blood sugar levels) is too high. Glucose is the main source of energy in the body that is provided by the food we eat. Insulin is a hormone made in the pancreas which helps glucose from food enter your cells to be converted into energy. If your body does not produce enough or any insulin, glucose stays in your blood leading to high blood glucose/sugar levels.

This research report will demonstrate a Python solution, developed to determine whether an individual is diabetic or not based on testing done to the patient, resulting in 8 different attributes that the solution will use to predict diabetes. If the model works, diabetes can be predicted in future for patients which could mean a decrease in hospital costs on unnecessary appointments.

The attributes are as follows:

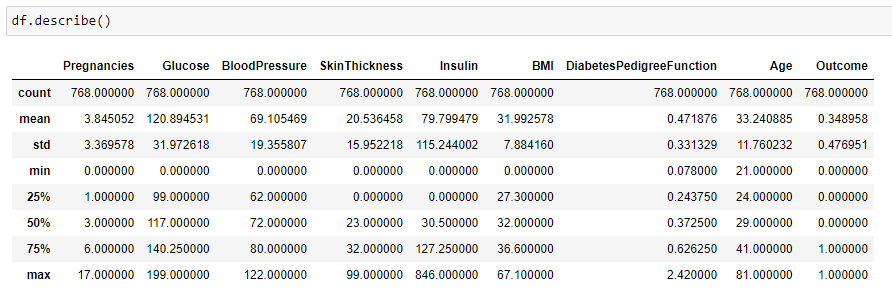
Number of pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes pedigree function and Age.

The solution is trained using the Pima Diabetes dataset and is using a logistic regression approach. The aim is to evaluate and determine the accuracy of the solution with the goal of attaining a high accuracy model.

1. Background

The provided dataset in predicting the onset of diabetes based on diagnostic measures includes 768 instances and 8 Attributes (9 including the Outcome). Attribute 1 is the number of times pregnant, 2 is the Glucose levels, 3 is the Blood Pressure, 4 is the Thickness of the skin under the triceps, 5 is the Insulin levels, 6 is the BMI, 7 is the diabetes Pedigree Function and 8 is the Age of the individual. Each one of the instances have 2 possible Outcomes, ‘1’ and ‘0’. ‘1’ meaning the individual is diabetic and ‘0’ meaning the opposite. In the dataset there was a total of 268 cases of diabetes out of 768 instances, 500 were classed as not diabetic.

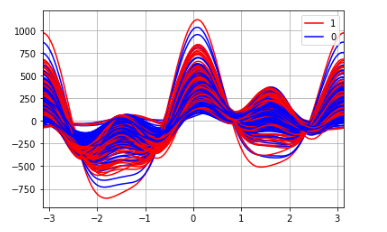
Some of the attributes in the diabetes data set were missing and denoted by a ‘0’. For example, having ‘0’ Glucose levels would be impossible and is therefore not useable data for an accurate prediction model. The dataset is described in figure 1.

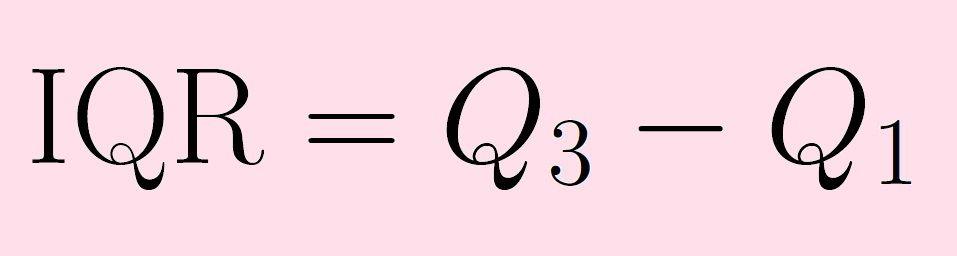
Figure 1 describing the dataset

1. Methodology
   1. preprocessing of the data

Before attempting to implement the solution into the dataset, the dataset was first observed and visualized to detect any outliers or nonsensical data that could potentially affect the outcome. The data was first visualized as an Andrews plot, being the only viable method that would be able to display data with as many attributes as the data set had to display, the Andrews plot was used to visualize the entire dataset as shown in figure 2.

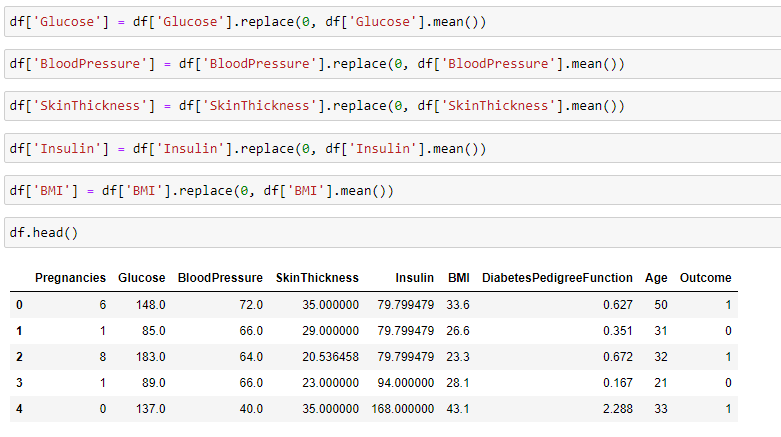
Figure 2 showing an Andrews plot of the entire dataset



The Andrews plot presents some outliers in the data however for the most part the data does seem to follow a trend. To further process the data, I ran the command df.describe(). By running the command df.describe(), the count, mean, std, min, 25%, 50%, 75% and max of the dataset are shown. Looking at this information, 5 of the 8 attributes have a ‘0’ value that doesn’t make sense. It is important to check datasets in this way to understand the data and to be able to clean the data so that it is in a useable state. To resolve this issue, I took each attribute that has nonsensical ‘0’ values and replaced them with the mean value of the attributes as shown in figure 3. Having done this the data was then cleaned of outliers by looking for the interquartile range using the formula.

By implementing the formula into the solution, outliers would be removed from the dataset.

However, by removing nonsensical data and removing outliers, I was unable to determine a change to the data as the outcome remained the same regardless of the outlier’s presence and by extension the inclusion of nonsensical data.

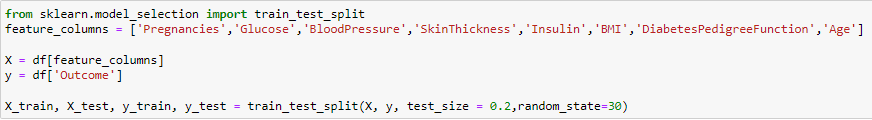
Figure 3 showing the removal of nonsensical data.

* 1. Developing the solution

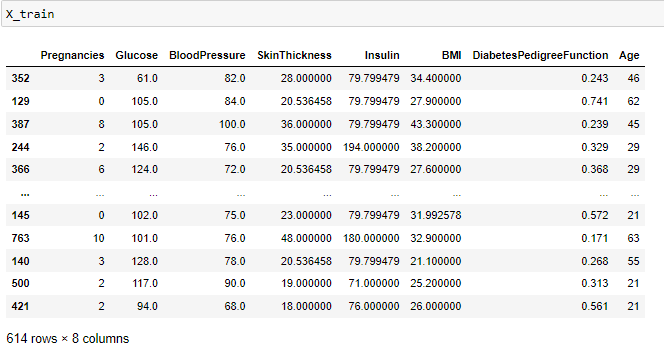
Before implementing the solution, I had to first determine what are the inputs and outputs of the system. In this case the diabetes dataset contained numerous attributes to describe the patients’ condition and would then result in the output of 2 possible variables, ‘1’ and ‘0’. ‘1’ meaning the individual was diabetic and ‘0’ meaning the latter. The system takes 8 attributes and uses them to come to this conclusion. This helped in my choosing of the solution which would be a logistic regression approach. Logistic regression is perfect for a prediction model and in fact makes use of a classification method where the output is binary(1 or 0), making this solution a perfect candidate for this scenario. The goal is to be able to predict accurately based on the 8 attributes if an individual would be diabetic.

* 1. Creating and training the solution

To train against the data set I used the train-test-split method. This method splits arrays or matrices into random train and test subsets. This method is used to split the data into 2 parts, in this case the attributes were trained separately from the outcome. To use this method, I had to create an array of the attributes and give it a name in this case I called it ‘feature \_columns’. I labelled this as ‘X’ and the ‘Outcome’ as ‘y’. From here the data is then split into 2 parts, attributes, outcomes which is shown in figure 4.

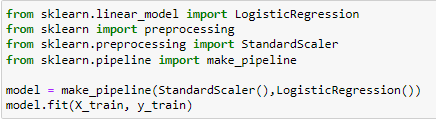
Figure 4 showing the Train-test-split method.

The data was split into a test size of 20% and a training size of 80% with a random state of 30. Looking at the code the attributes are clearly separated from the outcome by ‘X’ and ‘y’. So, if there was an attempt to display X\_train the following as shown in figure 5 would be seen.

Figure 5 showing the output of X\_train to demonstrate the splitting of data.

I could then use the X\_train and y\_train that is produced by this method and input them into the logistic regression model as shown in figure 6.

Figure 6 showing how the logistic Regression model was created.



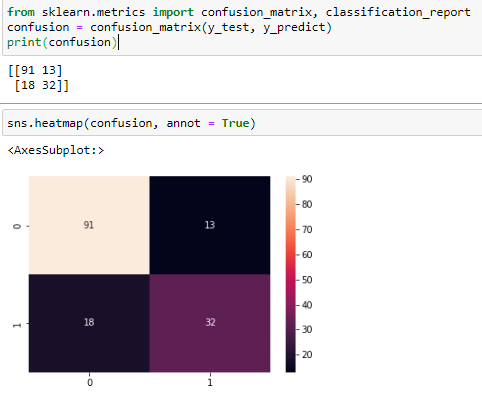
When creating the logistic regression model, I had to standardize the data using the ‘StandardScaler’ method which standardizes the dataset. Standardizing the data is a requirement in many machine learning

estimators (sklearn.preprocessing.StandardScaler, 2021). Once this was done, I had to simply fit the model which produced our prediction model to predict diabetes based off a set of parameters.

1. Analysis
   1. Testing

To test the system, the data was split into 2 separate parts as mentioned before. One part training which was 80% of the data, and one part testing which was 20% of the data. This way the 20% of data is purely used for testing whereas the other 80% will be trained and implemented in the logistic regression model. The test data can be used to create a y\_predict variable which will be used to compare the y\_test against it. I took this information and created a confusion matrix as shown in figure 7.

Figure 7 showing the confusion matrix.

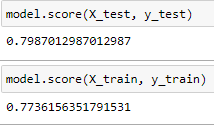


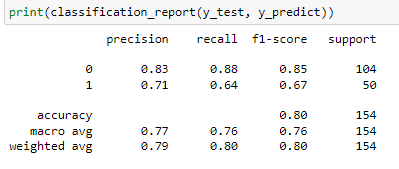
The confusion matrix shows TruePositive, FalsePositive, TrueNegative and FalseNegative values.

* True Positive shows the case where the output of the model is ‘0’ and the expected outcome is ‘0’
* True Negative shows the case where the output of the model is ‘1’ and the expected outcome is ‘1’
* False Positive shows the case where the output of the model is ‘0’ and the expected outcome is ‘1’
* False Negative shows the case where the output of the model is ‘1’ and the expected outcome is ‘0’

I also used model.score which shows the accuracy of the model for both the test which is the smaller 20% and the training set which was 80% of the data set. Figure 8 shows the score. I also created a classification report to show the accuracy of my model in more detail shown in figure 9.

Figure 8 showing the accuracy of both the test set and the training set.



Figure 9 shows the classification\_report.

* 1. Results

Overall, the developed model shows that the test score and the train score are relatively close to each other, being within 2% of each other. Looking at the confusion matrix the model is shown to be accurate and does its intended job of predicting diabetes to a high degree of accuracy as it displays few false positives and negatives and more true positives and false positives. There is of course room for improvement to become even more accurate by perhaps changing testing ratios.

1. Conclusions

In conclusion, the development of the model using a logistic regression approach has shown that an artificial network can predict the onset of diabetes based upon pretested conditions accurately. The model shows that by splitting the data into different percentages for training and testing, the results are still accurate and in this case within 2% accurate of each other. This method was trained after removing outliers and correcting nonsensical data and yielded an accurate model.

* 1. Future work

In terms of improvements in future, further research into different training methods is required to yield an even higher accuracy percentage. Another area to improve would be to include further testing of test ratios to see if there is any effect to the data and what the best testing ratio would be for this scenario. I could potentially look at how other algorithms could benefit the accuracy of my model. The data set required more preprocessing than I had given it, perhaps I could have implemented the min-max algorithm to normalize the data initially before any thing else was done. This could have yielded a higher accuracy model and is something that should be investigated in my further work. Finally timing how long the Ai takes to be trained and implemented is also good practice that should be implemented into future work as this helps with creating an efficient model, especially considering this is a medical model where there is an emphasis on being able to know the status of a patient with urgency and certainty.

1. Appendices





1. Bibliography

Kaggle.com. 2016. *Pima Indians Diabetes Database*. [online] Available at: <https://www.kaggle.com/uciml/pima-indians-diabetes-database> [Accessed 29 November 2021].

National Institute of Diabetes and Digestive and Kidney Diseases. 2016. *What is Diabetes? | NIDDK*. [online] Available at: <https://www.niddk.nih.gov/health-information/diabetes/overview/what-is-diabetes> [Accessed 29 November 2021].

scikit-learn. n.d. *sklearn.preprocessing.StandardScaler*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html> [Accessed 29 November 2021].

scikit-learn. n.d. *sklearn.linear\_model.LogisticRegression*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LogisticRegression.html> [Accessed 29 November 2021].

Buskirk, D., 2021. *» Using a Pandas Andrews Curve Plot for Multidimensional Data*. [online] Blog.learningtree.com. Available at: <https://blog.learningtree.com/using-a-pandas-andrews-curve-plot-for-multidimensional-data/> [Accessed 29 November 2021].

Brownlee, J., 2020. *Train-Test Split for Evaluating Machine Learning Algorithms*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/> [Accessed 29 November 2021].

scikit-learn. n.d. *sklearn.model\_selection.train\_test\_split*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.train\_test\_split.html> [Accessed 29 November 2021].