Predicting Diabetes with multilayer Perceptrons

Diabetes is a chronic medical condition that is associated with elevated blood sugar levels in the body.

Diabetes can be divided into two subtypes: type 1 and type 2.

Type 1 diabetes results from the body's inability to produce sufficient insulin and is rare compared to Type 2 diabetes.

Type 2 diabetes results from the body's gradual resistance to insulin and can be prevented if diagnosed early.

If we have labeled dataset that contains some vital measurements of patients (blood insulin level, age, etc), as well as a true label indicating the onset of diabetes in the patient sometime after the measurements were taken, then we can train a neural network on this data and use it in order to make predictions on new patients.

The dataset that we are going to use is Prima Indian Diabetes that hosted at Kaggle.

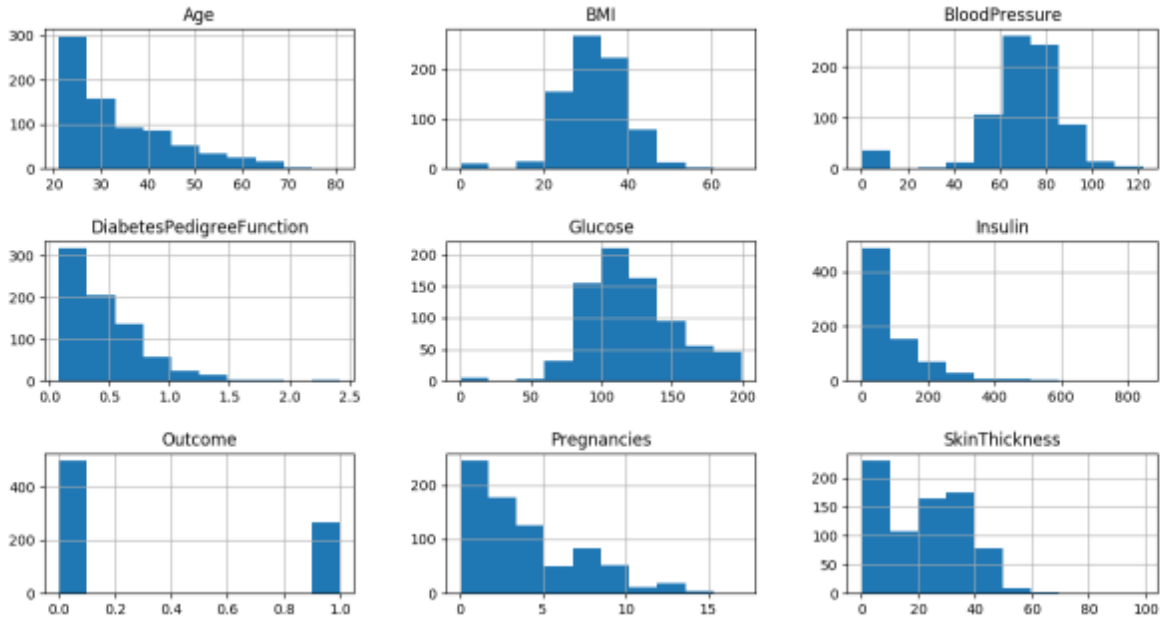
This specific tribe carries a gene that is able to store at their bodies whater glucose and carbonhydrates they may eat, which is beneficial in an enviroment were famines are common.

Due to modern's society diet, the incidence of Type 2 diabetes among Pima Indians is the highest in the world.

The dataset consists of 9 categories:

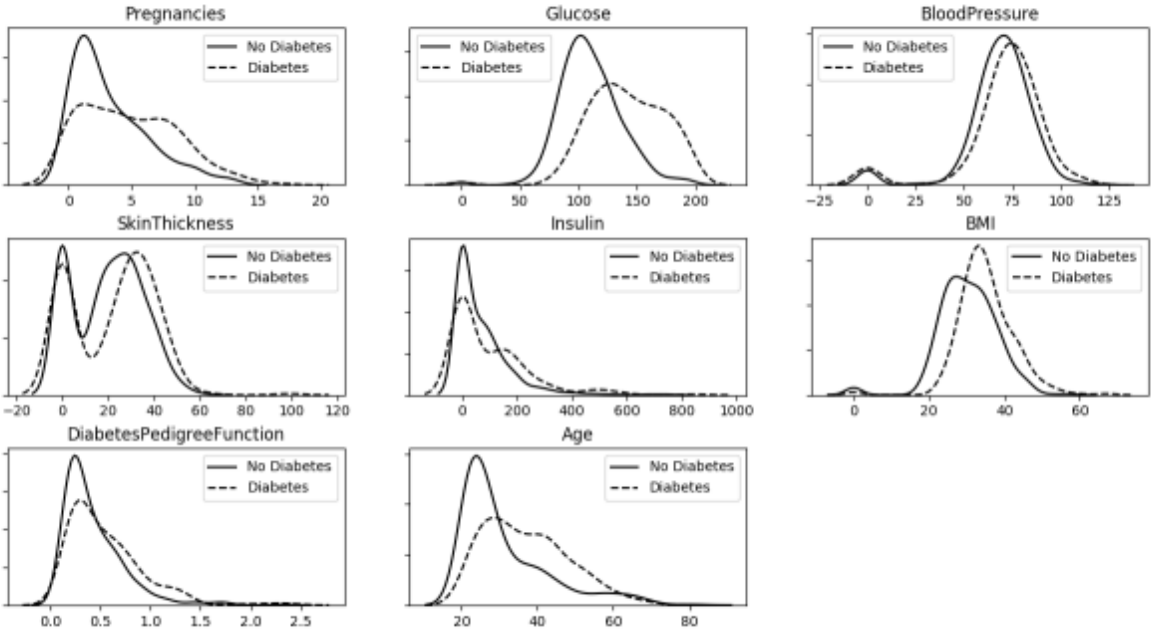
- **Pregnacies**, **Glucose**, **BloodPressure**, **SkinThickness**, **Insulin**, **BMI**,**Age** that are metrics of each person.
- **DiabetesPedigreeFunction**, which is a summarized score that indicates the genetic predisposition of the patient for diabetes.
- **Outcome** --> 1 for patients that developped diabetes within five years and 0 otherwise.

To see an accurate representation of the distribution of our numerical data, we display the histograms for each of our variable:



By plotting histogram of each variable, we can conclude that there are non valid data. **Glucose** level is almost 200, **BloodPressure** ,**BMI** and **Glucose** variables have 0 values and **Pregnancies** are over 10. Also, we note that the variables are on different scales.

For further investigation, by plotting distributions for each variable for people with diabetes and those they don't have, we deduce that **Glucose** ,**BMI** and **Age** are strong predictors for diabetes since their distributions differ one each other, in response to **BloodPressure** and **SkinThickness** that are poorer predictors.



After we search for **NaN** values in our data, in order to "correct" our data, we will replace invalid values, such as 0 at **Glycose** and **BloodPressure**, with the mean of each variable.

Data standardization

The goal of data standarization is to transform the numeric variables so each variance has zero mean and unit variance.

In neural networks, it is important to standardize the data in order to ensure that the backpropagation algorithm works as intended. Another positive effect of data standardization is that it shrinks the magnitude of variables, tranforming them to a scale that is more proportional.

It is known that the variable with the greatest scale tends to dominate when training the neural network, causing the neural network to emphasize on the variables with a greater scale.

The last step in data preprocessing is to split the data into training, testing and validation sets:

- **Training set:** The neural network will be trained on this subset of data.
- **Validation set:** This set allow us to perform hyperparameter tuning (that is tuning the number off hidden layers) using an unbiased source of data.
- **Testing set:** The final evaluationof the neural network will be based on this subset of this data.

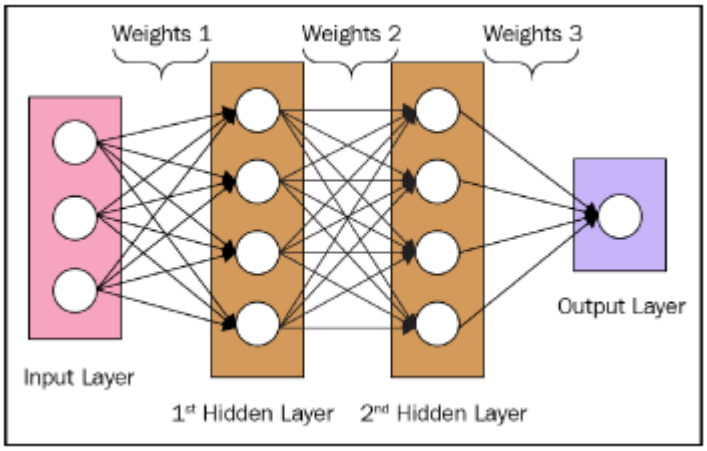
The purpose of splitting the data into those sets is to avoid overfitting and to provide an unbiased source of data for evaluating model performance.

We should split the original data into 80% training and 20% testing, and then split again the training data into 80% training and 20% validation. It is important that the spliting of data must be random.

Network Model

We will use **Multi Layer Perceptron (MLP)**, a class of feedforward neural network that has at least one hidden layer that is activated by a non-linear activation function. This multilayer architecture and non-linear activation allows MLP to produce non-linear decision boundaries.

The model will have 2 hidden layers as seen in picture below:



There should be 8 inputs at **input layer**, 32 nodes at first hidden layer, 16 nodes at second one with 1 node in **output layer**.

As for activation functions for our **hidden layers**, we will use the ***relu*** function (f(x) = max(0,x)) and for the **output layer**, since we need a binary classification, we will use the **sigmoid** function (f(x) = 1/(1+exp(-x))).

We will use ***adam*** optimizer to update our weights and since we have a binary classification problem, our loss function will be ***binary crossentropy*** **.

The model will be trained under 200 epochs.

Result Analysis

After we evaluate our network for **training** and **test** data, our accuracy receivings are **90%** and **87%** respectively.

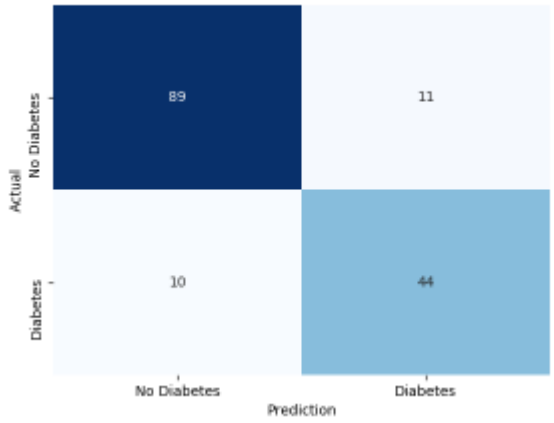
Another tool for our analysis is **Confusion matrix**. This method provides analysis on true negative, false negative, true positive and false positive predictions that are made by our model.

The definition of above predictions is as follows:

- **True Negative (TN)**: Actual class is negative (no diabetes), and the model predicted negative (no diabetes)
- **False Negative (FN)**: Actual class is positive (diabetes), but the model predicted negative (no diabetes)
- **True Positive (TP)**: Actual class is positive (diabetes), and the model predicted positive (diabetes)
- **False Positive (FP)**: Actual class is negative (no diabetes), but the model predicted positive (diabetes)

We want our false positive and false negative numbers to be as low as possible, and for the true negative and true positive numbers to be as high as possible.

The results of confusion matrix are seen below:

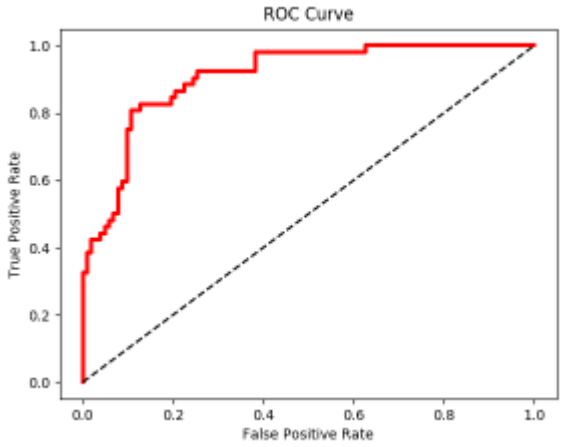


Finally, our last analysis tool will be the **ROC** curve. The ROC curve is a plot with **True Positive Rate (TPR)** on *y* axis and the **False Positice Rate (FPR)** on *x* axis.

Above measures are defined as TPR = TP/(TP+FN) and FPR = FP/(TN+FP).

When we analyze the ROC curve, we mainly focus at the **area under the curve (AUR)** to evaluate the performance of the model. A large AUC indicates that the model is able to differenciate the classes with high accuracy, while a low AUC indicates that the model makes poor predictions. If the ROC curve is similar to the diagonal, the model does random predictions.

The ROC curve of our model is seen below:



This shows that our model is able to make good predictions.