MACHINE LEARNING PROJECT REPORT KNIME CHALLENGE: DIABETES PREDICTION COMPETITION

Team 02: Eleonora Brambatti (858098), Alessandro Castagna (898057), Daniel Colombaro (897062)

17th February 2023

Abstract

Is it possible to classify whether a person is likely or unlikely to have Diabetes, looking into a dataset? This is the question that Kaggle Community Competitions hosted by ML GDEs or TFUGs, sponsored by Google Developers has asked in their "Diabetes Prediction Competition".
The aim of this analysis is to find the best performing Machine Learning model that can give an answer to the previous question, in other words, the goal of the work is to find the best predictive model that can say if a person can suffer from diabetes or not, taking into account different factors like: diseases, wealth, diet and family history.
According to the competition rules, the selection of the best model refers to the value of the Logloss function of the model, that is one of the values that measures the goodness of the model.

Contents

1 Introduction

Diabetes is among the most prevalent chronic diseases in the world, impacting millions of people each year and exerting a significant financial burden on the economy. Diabetes is a serious chronic disease in which individuals lose the ability to effectively regulate levels of glucose in the blood, and can lead to reduced quality of life and life expectancy. When the sugars are introduced in the body they are released into the bloodstream. This is the moment when insulin is released by the pancreas. People who suffer from diabetes are characterized by their body not making enough insulin or not being able to use the insulin as it is needed.
Doctors say that, for people suffering from diabetes it is possible to associate some factors to high levels of sugar remaining in the bloodstream, for example: being overweight, obesity, physical inactivity, insulin resistance, genes and family history, genetic mutations, hormonal diseases, heart diseases, vision loss, lower-limb amputation and kidney diseases. Unfortunately, there is no cure for diabetes, but strategies like losing weight, eating healthily, being active and receiving medical treatments can mitigate the harms of this disease in many cases.
For this reason, predictive models for diabetes risk are important tools for public health officials and what we are going to do in this analysis is to find the best performing model that can predict the probability of people that suffer or not from diabetes and to find which are the factors that can be associated to the diabetes.

2 Dataset description

The dataset used in the analisys has been synthetically cured from the orginial data source using CTGAN, it is formed by 40108 rows (the records) and 18 columns, which represent the attributes taken into account in the analysis. In particular the dataset contains 17 feature variables and one target variable, which are:

- ***Age*** (numeric): 3-level age category (_AGEG5YR see codebook). 1 = 18-24, 9 = 60-64, 13 = 80 or older;

- ***Sex*** (binary): 0 = female, 1 = male;
- ***HighCol*** (binary): 0 = no high cholesterol, 1 = high cholesterol;
- ***ColCheck*** (binary): 0 = no cholesterol check in 5 years, 1 = yes cholesterol check in 5 years;
- ***BMI*** (numeric): Body Mass Index;
- ***Smoker*** (binary): Have you smoked al least 100 cigarettes in your entire life? [Note: 5 packs = 100 cigarettes] 0 = no, 1 = yes;
- ***HeartDiseaseorAttack*** (binary): Coronary heart disease (CHD) or myocardial infarction (MI). 0 = no, 1 = yes;
- ***PhysActivity*** (binary): Physical activity in past 30 days - not including job. 0 = no, 1 = yes;
- ***Fruits*** (binary): Consume fruits one or more times per day. 0 = no, 1 = yes;
- ***Veggies*** (binary): Consume vegetables one or more times per day. 0 = no, 1 = yes;
- ***HvyAlcoholConsump*** (binary): Adult male: more than 14 drinks per week. Adult female: more than 7 drinks per week. 0 = no, 1 = yes;
- ***GenHlth*** (categorical): Would you say that in general your health is: (scale 1-5) 1 = excellent, 2 = very good, 3 = good, 4 = fair, 5 = poor;
- ***MentHlth*** (numeric): Days of poor mental scale 1-30 days;
- ***PhysHlth*** (numeric): Physical illness or injury days in past 30 days scale 1-30;
- ***DiffWalk*** (binary): Do you have serious difficultly walking or climbing stairs? 0 = no, 1 = yes;
- ***Hypertension*** (binary): 0 = no hypertension, 1 = hypertension;
- ***Stroke*** (binary): 0 = no, 1 = yes;
- ***Diabetes*** (binary): 0 = no diabetes, 1 = diabetes (Target variable).

3 Development of the work

As we anticipated in the "Introduction" part, the aim of the work is to find the best predictive model that can give us the best result in terms of classification of people with diabetes or not. So, we create different performing models and we compare them calculating for each one the Logloss function value.
To do that we used the open source software "KNIME Analytic Platform". This platform allows us to build step by step different models connecting different nodes, where each node starts with an input and, with the appropriate configuration, ends with a specific output that can be used as the input of the following node.
As we said before, the evaluation metric for this competition is the Logloss, in fact, this value is an indicator of how close the prediction probability is to the corresponding actual value that in this case can be 0 or 1. Once we calculated all the predictive models Logloss values, we compared the results and we chose the best performing model.
In the light of this result, we built two different workflows, a training one and a deployment one, the first contains the nodes relative to our best model and it is used to show our result and the second one contains also the nodes for the testing part, in fact it will be used to obtain the value of the Logloss using as the input dataset the test set.
As the last part of the work we create a Data App that is an interactive dashboard through which we can show univariate and multivariate data visualization.

4 Realization of the workflow

To create our models, we did different steps. Once we gave the dataset as the initial input to the first node, we started the preprocessing part in which the raw data of the dataset are prepared for further analysis through a series of actions, in this way the efficiency of the model is improved. Then, with the new data we started the classification part choosing the models we wanted to test, the techniques we wanted to apply to train the models and calculating the value of the performance (Logloss value). We concatenated all the Logloss obtained in order to compare them. For the choice of the best performing model we took into account the best result in terms of classification method used and of the value of the Logloss.

4.1 Preprocessing

First of all, we started by resolving the missing value problem. If there is an absence of data in the dataset the KNIME "Missing Value" node operates replacing the missing values for categorical attributes with the most frequent value in the distribution of the variable and for numeric attributes with the median of the values of the variable. We tried four different methods of preprocessing, which are:

1. We only eliminated the missing values of the dataset and we didn't process them in any other way.
2. We calculated the correlation value between attributes and we removed the variables that were high correlated, i.e. if the correlation coefficient value was higher than 0,5.
3. We grouped the continuous variables into bins; for the "MentHlth" and "PhysHlth" variables, which represent the days of mental and physical illness recorded in a month, we chose from 1 to 15 days and from 16 to 30, in order to divided in two parts these attributes. For the "BMI" variable we divided the values in 4 different bins:
   - "Underweight" when $x \leq 18,5$ (where x is the value of the "BMI" variable);
   - "Normal weight" when $18,5 < x \leq 24$;

- "Overweight" when $24 < x \leq 30$;
- "Obese" when $x > 30$;

This division is conventionally used for the "BMI" values. Even in this case, after these discretizations we removed the variables that were high correlated (correlation > 0,5).

4. We did the same things done at point 3 above, but this time the four categories that we create for the "BMI" variable are transformed into binary variables, in fact, a binary column is created for each category with the value equal to 1 for the correspondent category and 0 for the other ones. Also in this case we filtered the dataset removing the attributes if the correlation value between two variables was higher than 0,5.

We tested each of these 4 new dataset with different classification models, in order to have a large choice of results among which we could select the best performing model.

4.2 Models

As we anticipated before, we implemented many different models using the appropriate KNIME node connected to "Weka Predictor" node. The machine learning algorithms used are:

- Logistic Regression: it is often used in binary classification problems and it is an algorithm that works by estimating the probability of an event occurring based on the input features;
- J48 (Decision Tree): it divides the input space into different regions associated with a class label and it predicts by traversing the tree based on the values of the input features;
- Random Forest: this is a collection of decision trees which are trained on different subsets of the input data and then combining their predictions in order to have a more accurate result;
- XGBoost (linear and tree): it is based on the gradient boosting framework and is designed to improve the performance of standard gradient boosting algorithms by using a set of advanced features.;
- NBTree (Decision Tree): it classifies the data more efficiently and effectively than existing other decision trees, the algorithm sorts the example to a leaf and then assigns a class label by applying a Naive Bayes on that leaf;
- Naive Bayes: based on the Bayes theorem of probability, it calculates the probability of each class label given the input features (that are independent of all other features), and then selecting the class with the highest probability as the prediction;
- Multilayer perceptron: It consists of multiple layers of nodes (also called neurons) that are connected in a feedforward manner. The input layer receives the input data, which is then processed through one or more hidden layers before producing the final output through the output layer;
- Support Vector Machine: it finds an hyperplane that maximizes the margin between the different classes. This makes it especially effective in cases where the classes are not easily separable.

4.3 Classification methods

Then we implemented different classification techniques for evaluating the performance of machine learning models described in the paragraph above. In particular we trained all the listed models above with the holdout and cross validation techniques.

4.3.1 Holdout

This technique evaluates the performance of a trained model splitting the initial dataset into two subsets called training set and test or validation set. The model is trained on the training set and then evaluated on the validation set to estimate its performance on new, unseen data. We chose to split the data in the $\frac{2}{3}$ of the dataset for the training set and the remaining $\frac{1}{3}$ of the dataset for the test set.
The negative part of using the holdout method is that this technique can be sensitive to the specific random sampling used to split the data, and may not always provide an accurate estimate of the model performance. We can have both case in which the partition of the dataset gives very good or very bad results in terms of performance evaluation of the model, so it is possible to have the overfitting problem.

4.3.2 Cross Validation

This is the technique that is considered the best practice for evaluating and selecting models, in particular we used the k-folds cross validation. With this method the dataset is divided into k equally sized folds, and the model is trained and tested k times. In each iteration, one of the folds is used for testing, while the remaining k-1 folds are used for training.
The performance of the model is then evaluated by computing the average of the evaluation metric over the k iterations. Due to the fact that this method is not only based on using a single train-test split, the results obtained through the cross validation are more accurate and so the estimation of the performance of the model is better. In our specific case, we used 10 folds.

4.4 Feature Selection

We created also a feature selection metanode in order to see if the results obtained with this process were better than the ones obtained without it. We put this metanode between the preprocessing part and the output of the feature selection goes through both the holdout process and the cross validation process. In the feature selection process, the most relevant variables are selected while the irrelevant, redundant, or noisy variables are removed. In this way it is possible to improve the model performance because the dimensionality of the data is reduced. In particular we developed three different path, in the first one we used the filter method, in the second one the backward feature selection and the last one is

the backward feature selection.
The advantages of feature selection are improved model performance, reduced model complexity, faster training times, and increased interpretability of the model. However, selecting the right set of features can be not easy and it can give unreliable results.

4.4.1 Filter Method

For this process we used the KNIME node "Attribute Selected Classifier". It combines feature selection and classification into a single step. It involves selecting a subset of the most relevant features from a dataset using a filter method, and then applying a classification algorithm on the selected features.

4.4.2 Backward Feature Selection

In this method for feature selection the process begins with all the variables included in the dataset and iteratively removes variables until the optimal subset of features is identified. At the beginning is trained a model using all the initial variables, then the attributes with the most irrelevant impact of the model are removed. This process is iterated till the subset of variables that give the best performance is identified. The limitation of this method is that is possible that the best subset of features is not identified.

4.4.3 Forward Feature Selection

The forward feature selection follows the opposite process of the backward feature selection. This process begins with an empty set of variables and iteratively adds the variables that have the highest impact on the model's performance. Even in this case the process is repeated until the best subset of variables is identified. As well as for the backward feature selection, the limitations of this method is that the best subset of variables can not be identified.

4.5 PCA

We also tried to execute a Principal Component Analysis (PCA), a model used to reduce the dimensionality of the dataset (the number of variables). It is different from a feature selection because in this case new variables which can maximize the level of information are created. The first principal component captures the maximum amount of variance in the data, the second principal component captures the second maximum amount of variance, and so on, in this way the new features capture most of the variability in the original dataset.

4.6 Logloss results

At this point, having developed all the models we want to train, the last step is to add the nodes relative to the Logloss function. In the "Math Formula" KNIME node we inserted the Logloss function formula, that is the following one:

$$\frac{1}{N}\sum_{n=1}^{N} H(p_n, q_n) = -\sum_{n=1}^{N}[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)]$$

where y_n is the vector which represents the true class lable and $\hat{y}_n$ is the predicted probability of the n^{th} class. This function, also known as the cross-entropy loss, measures the difference between the predicted probability distribution of the model and the true probability distribution of the target class. We interpreted the Logloss as follows: the lower the value of the Logloss and the better is the performance of the model.

In the sub-paraghaphs below we showed the top three value results of the Logloss for each classification method but the PCA. This operation has produced only high values of the Logloss, so we do not consider it for the choice of the best model.

4.6.1 Holdout

In Table 1 we can find the results relative to the top three Logloss (the lowest values) with the holdout method. In the "Preprocessing" column are stored which of the four types of preprocessing (reference to the "Preprocessing" paragraph above) is the one of the model, in the "Model" column we find the classificator of the model and in the "Logloss" column we find the value which measure the performance of the model.

Preprocessing	Model	Logloss
1, 2, 3, 4	Logistic Regression	0.224
1, 2, 3, 4	XGboost linear	0.227
1, 2, 3, 4	XGboost tree	0.228

Table 1: Best three Logloss with the holdout classification

We can see that the model with the best result of the Logloss is the one of the logistic regression (0.224). This model has the same result for each one of the preprocessing type used.

4.6.2 Cross Validation

In Table 2 there are the best three results of the performance evaluation of the model using the cross validation technique. The structure of the table is the same as Table 1.

Preprocessing	Model	Logloss
1, 2, 3, 4	Logistic Regression	0.225
3, 4	XGboost tree	0.226
1, 2	XGboost tree	0.227

Table 2: Best three Logloss with the cross validation classification

Even in this case, the best performing model is the Logistic Regression one (with no distinction between different preprocessing). In this case the value is higher than the one found with the holdout.

4.6.3 Feature Selection and Holdout

In Table 3 are reported the best three results of the Logloss obtained with the feature selection and the holdout classification. In the "Preprocessing-Type" column are recorded the type of preprocessing (a number from 1 to 4 as used in Table 1 and 2) and the type of feature selection correspondent to the model (filter method: FM, backward feature selection: BW and forward feature selection: FW). The other two column "Model" and "Logloss" are the same as the ones in Table 1 and Table 2.

Preprocessing-Type	Model	Logloss
4-BW, 4-FW	Logistic Regression	0.223
4-BW	Multilayer perceptron	0.225
1-BW, 1-FW, 2-BW, 2-FW	Logistic Regression	0.226

Table 3: Best three Logloss with the feature selection and holdout classification

The best model is the Logistic Regression, but this time the only type of preprocessing is the one with the discretization of the numeric variables and the splitment of the "BMI" column into 4 binary columns, each one crated for each bin. The type of feature selection correspondent to the lowest value of the Logloss are the backward and the forward feature selection.

4.6.4 Feature Selection and Cross Validation

In Table 4 we find the best three results of the Logloss for the feature selection and cross validation methods. The structure of the table is the same as Table 3.

Preprocessing-Type	Model	Logloss
1-BW, 1-FW	Logistic Regression	0.223
3-BW, 3-FW, 4-FW	Logistic Regression	0.226
4-BW	Logistic Regression	0.227

Table 4: Best three Logloss with the feature selection and holdout classification

We can see that the lowest value of the Logloss corresponds to the Logistic Regression with no initial preprocessing and using the backward or forward feature selection.

5 Best predictive model

We can say that all the Logloss calculated with the different methods are very similar to each other. Our considerations for the choice of the best predictive model are the following:

1. Considering the Logloss values, as we see through the tables above, the lowest are the ones relative to the feature selection applied to the holdout and to the cross validation (0.223), then we have the value obtained with the holdout classification (0.224) and then there is the one obtained with the cross validation (0.225).
   Despite the fact that the lowest is the Logloss value and the best is the performance, in this case, we chose to consider the value obtained with the cross validation method. We explain in the previous paragraphs that there are many limitations linked to the used of the holdout method and the feature selection, for this reason and also because the cross validation method, in general, is a more reliable method.
   In addition to this, this three Logloss value are very close and are very close to the second and third value in Table 2 found with a different model, this let us think that select the one relative to the cross validation can give the best prediction.

2. We can see that, about the model, the Logistic Regression is the best classificator in any case. This is a very efficient classificator and in particular, is very good for binary classification problem, like the one in our case.

3. As it is shown in Table 2, the preprocessing part does not influence the value of the Logloss obtained with cross validation applied to the Logistic Regression model but for the best model we chose the process with fourth one, because we think that doing the discretization of numeric variable could have a better effect on the model performance. In support of our decision we can see from Tables 1, 3 and 4 the fourth type of preprocessing is quite always associated with the lowest values of the Logloss. For this reason we select the model with this type of preprocessing.

5.1 KNIME training workflow

In our KNIME training workflow we inserted all the nodes required to build our best predictive model. As we said in the previous paragraph, the best model we select among the ones tested, starts (after receiving the input dataset) with resolving the missing values dataset problem and then with the fourth type of preprocessing in which numeric columns are transformed into bins columns and then into binary column. With this process the number of columns change from 18 to 21. Then there is the correlation filter which remove the "Obese" column because it was high correlated ($> 0,5$) with other variables.
Then we created the Cross Validation metanode in which we find the partitioner node, the classificator (Logistic Regression) node, the weka predictor node and the aggregator node. In this metanode the output of the predictor is connected also with the "accuracy" metanode, in which is calculated the value of the accuracy of the model that in this case is equal to 0.754.
The output of the "Cross Validation" metanode is the input of the "Logloss" metanode in which through the "Math formula" and other nodes, the performance Evaluation of the model is calculated and shown in a table. As we know, this value is equal to 0.225.

5.2 KNIME deployment workflow

The KNIME deployment workflow contains two node for two input dataset, the training and the test set. For this reason the classification method we have inserted in the deployment workflow contains, instead of the metanode of the "Cross Validation", the one of the "Holdout", in fact, the holdout method provides a partition of the

dataset.
The training dataset goes through the preprocessing with the missing value replacement process and the correlation filter process such as in the training workflow. Then it pass through the "Holdout" metanode (which contains the metanode of the accuracy) and then it ends with the "Logloss" metanode.
The test set passes through another "Preprocessing" metanode that is the same as the one used for the training dataset. Then it enters in the second input port of the "Holdout metanode" and it is connected to the "Weka Predictor" node, in this way the prediction are made on the test set.
The final step is the "Logloss" metanode whose output is the Logloss value obtained training the model on the test set.

6 Realization of the Data App

We created two dashboard for our Data App, in the first one we show some graphs relative to the explanatory variables and in the second one we concentrated on the classification model. The dashboard are interactive, it is possible to select through filters different options of view.

6.1 Visualization of the explanatory variables

In the dashboard of the explanatory we started showing the correlation matrix (Figure 1) of the dataset variables.

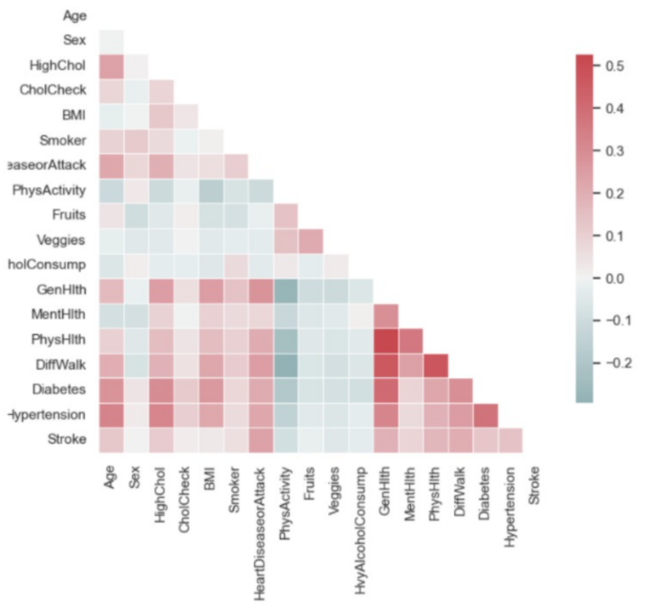

Figure 1: Correlation matrix of the dataset variables.

In Figure 1 it is possible to see the value of the correlation for each couple of variables. Colours close to red indicate a high positive correlation, while colours close to blue indicate a low negative correlation.

The second visualization (Figure 2) is the bar chart of the diabetes variable (target variable). It show the total of records that suffer from diabetes (Diabetes = 1) and that do not suffer from diabetes (Diabetes = 0).

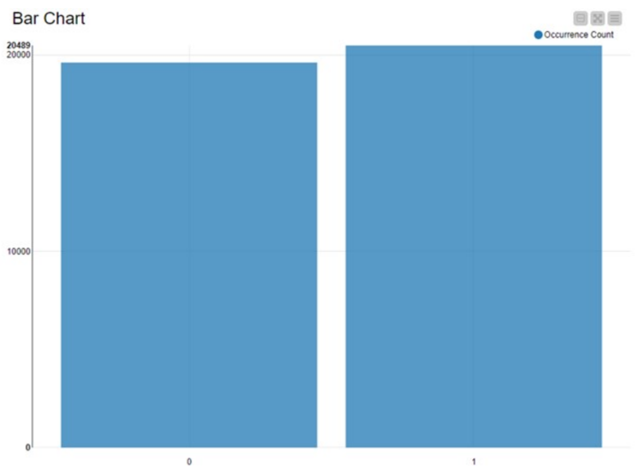

Figure 2: Barchart of the diabetes variable.

We can see from Figure 2 that there is a balance between the obeservations, in fact, the diabetes records equal to 0 and to 1 are almost the same. This is an important factor that makes reliable our analysis.

Then we focused on the numeric variables "Age", "BMI", "MentHlth" and "PhysHlth". We used a bar chart (Figure 3) to show the mean values of these variables. It is possible to choose to visualize only the records when the diabetes values is equal to 0 or 1 or both.

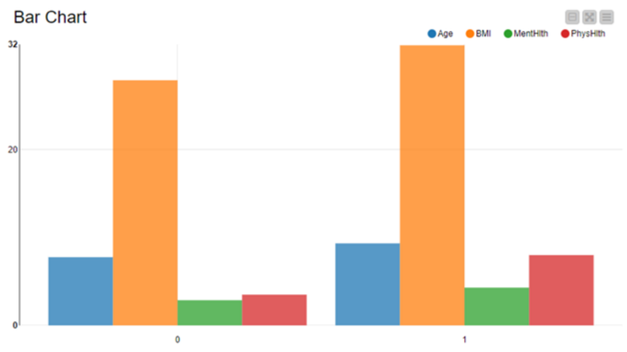

Figure 3: Barchart of the "Age", "BMI", "MentHlth" and "PhysHlth" variables when Diabetes = 0 and Diabetes = 1.

From Figure 3 when can say that the means of the variables are lower when Diabetes value is equal to 0 and higher when it is equal to 1. This can indicate that these variables have an influence on diabetes.

Then we used four violin plots and four density plots, each one for each variable which shown the distributions of the variables when the diabetes value is equal to 0 and to 1. In the violin plot (we report the violin of the "Age" variable, Figure 4 when diabetes = 1) it is possible to see the values of the max, the min, the mean, the median and the first and third quartile.

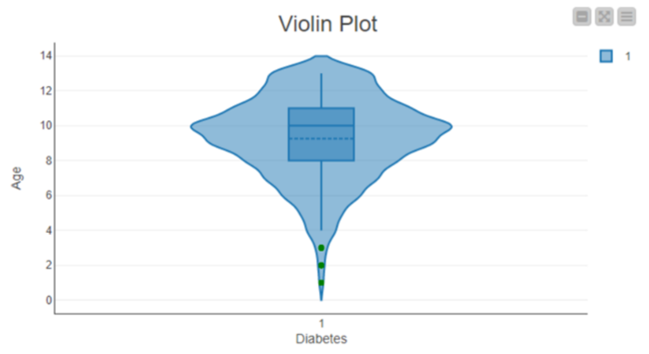

Figure 4: Violin plot of the "Age" variable distribution

In Figure 5 is shown the density plot of the "Age" variable when diabetes value = 0 (blue line) and when diabetes value = 1.

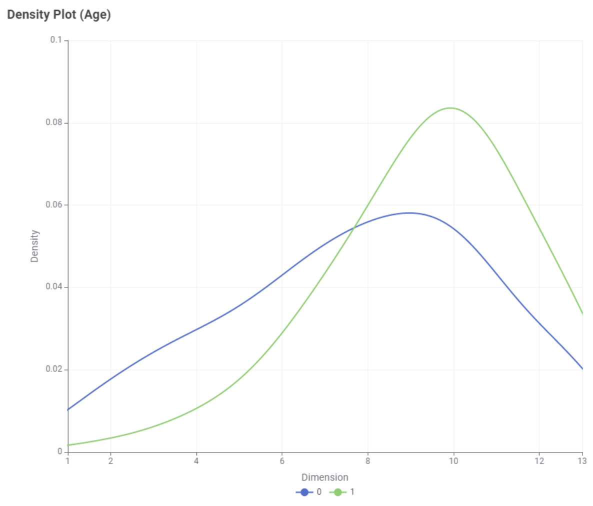

Figure 5: Density plot of the "Age" variable distribution

From Figure 4 and 5 we can say that, due to the fact that the distributions change if diabetes value is 0 or 1, diabetes can depend on the age of a person. And we can say the same thing about the other three variables because they have the same behaviour.

The last part of the first dashboard shows a table which contains in the rows all the attributes and in the columns the values of extreme values, standard deviation, variance, mean and others and clicking on each attribute it is possible to see the histogram of the variable.

6.2 Visualization of the classification model

The second dashboard has two sections. In the first one there is an interactive tool with which is it possible to choose the value of the treshold (prediction probability). This filter is connected to different charts.

The first one is a bar chart which shows the values of the statistics of the model like:

- AUC: "Area Under the ROC Curve", it is a common metric used to evaluate the performance of a binary classification model;
- Accuracy: it measures the proportion of correct predictions among all predictions made by the model;
- Error rate: it measures the proportion of incorrect predictions made by the model ;
- False Negative Rate: it measures the proportion of actual positive examples that are incorrectly predicted as negative by the model;
- Precision: measures the proportion of true positive predictions made by the model out of all positive predictions made;
- Specificity: it measures the proportion of true negative predictions made by the model out of all negative predictions made;
- F-Measure: combines precision and recall into a single score that reflects the balance between them;
- Younden's J Index: it is the difference between the true positive rate (sensitivity) and the false positive rate (1 - specificity) at each possible threshold value.

These values (but the AUC) changes with the change of the trashold.

The second graph (Figure 6) represents the ROC curve (Receiver Operating Characteristic) which is a plot of the true positive rate (TPR) against the false positive rate (FPR) at various classification thresholds. The area under the curve is the AUC and in this case this value is equal to 0.8205.

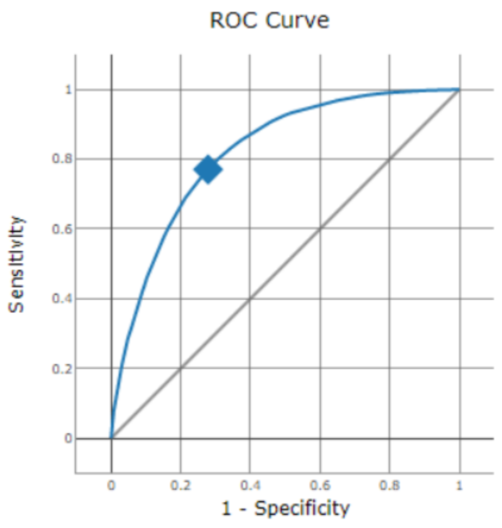

Figure 6: ROC curve

When the threshold value changes, the blue square in Figure 6 moves.

Then we showed the confusion matrix (Figure 7) that is a table used to evaluate the performance of a classification model by comparing the predicted class labels with the actual class labels. It summarizes the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions made by the model.

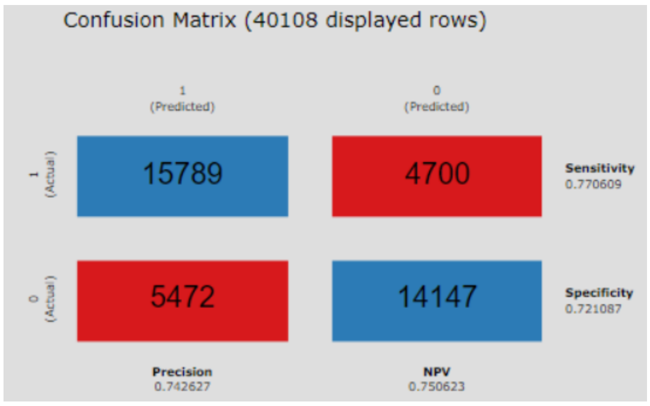

Figure 7: Confusion Matrix

The last graph that changes with the threshold filter is the graph that shows the distribution of class probability (Figure 8).

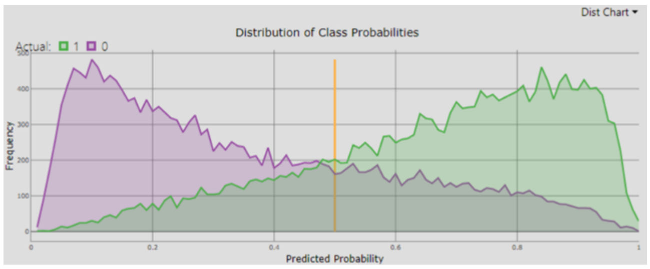

Figure 8: Distribution of class probability

On the x-axis there are the Predicted Probability and on the y-axis there are the number of frequencies. From Figure 8 we can see that the two distribution of the prediction of diabetes class 0 and 1 are quite the opposite because the "0 distribution" is pointed close to value around 0, while the "1 distribution" has the highest frequencies values correspondent to the highest value of the probability (around 1).

Then we showed the cumulative gain chart (Figure 9) that is a graphical representation of the performance of the model which plots the cumulative number of records with diabetes value = 1 (positive outcomes) against the percentage of the total population targeted by the model.

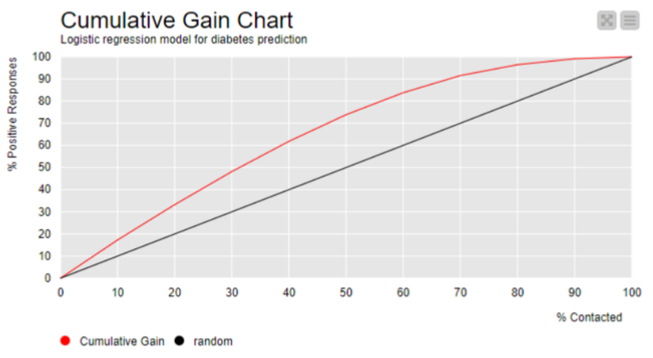

Figure 9: Cumulative Gain Chart

It compares the performance of the model to a random selection of the population. The x-axis of the chart represents the percentage of the population that is targeted, while the y-axis represents the percentage of records with the diabetes value = 1 that are obtained by targeting that percentage of the population.

As we can see in Figure 9, in the cumulative gain chart, the line representing the performance of the model is compared to a diagonal line representing random selection. If the model outperforms random selection, the line for the model will be above the diagonal line, and the area between the two lines represents the improvement in performance that is obtained by using the model.

The cumulative gain chart is connected to another graphs that is the lift chart, which is the last graph we presented in the dashboard.
The the lift is the ratio of the model's performance to the random selection. The x-axis of the chart represents the percentage of the population that is targeted, while the y-axis represents the lift (Figure 10).

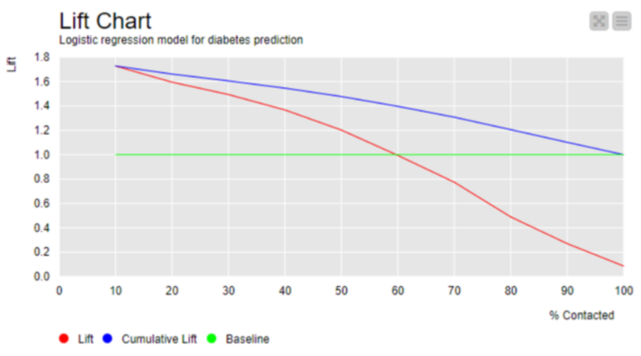

Figure 10: Lift chart

The green line in Figure 10 refers to the model of a random selection of people, while the red line refers to the cumulative gain chart.

The last part of the dashboard reports a serie of tables which sum up the statistics of the model seen with the graphs, in particular there are three tables:

1. confusion matrix of the model;
2. class statistics which reports the values of recall, precision, sensivity, specificity and F-measure;
3. overall statistics which include the values of overall accuracy, overall error, cohen's kappa and the number of records classified correctly and incorrectly.