Diabetes prediction with Gradient Boosted Decision Tree

KNIME Challenge 2023 - Università degli Studi Milano Bicocca

Stella Cervini Daniel Montalbano Giuseppe Sabino

***Abstract*—Artificial intelligence is developing impressively in all areas of the modern world. AI-enabled tools can extract relevant insights from large amounts of data and generate actionable insights that could be applied to many situations, in fact they are used to surfacing data insights.**
They are important for:

- **Eliminate disturbing elements: AI can help make sense of the massive amount of clinical data, medical literature and population data and use it to make informed decisions.**
- **Provide contextual relevance: AI can help healthcare providers extend the view by rapidly interpreting billions of data points to identify contextually relevant information for individual patients.**
- **Reduce errors caused by fatigue: human error is costly and fatigue can lead to errors. AI algorithms do not suffer from fatigue or distractions, and they can process massive amounts of data with incredible speed and accuracy [1].**

The use of artificial intelligence does not aim to eliminate the figure of the doctor, but rather to support it to give greater support in view of these phases.
With this paper, we will try to explain the phases of a machine learning process using a Gradient Boosted Decision Tree (GBDT) Model with a view to predicting the probability that a person has diabetes or not given a set of data. Then this model will be used to make a data app useful for a general practitioner for a screening phase of all the patient and to have a first impact on the patient's health.

I. Introduction

Diabetes is a chronic medical condition characterized by high blood sugar levels. It occurs when the body cannot produce enough insulin, which is a hormone that regulates blood sugar levels, or cannot effectively use the insulin it produces. This can lead to a variety of complications over time, including damage to the heart, blood vessels, eyes, kidneys, and nerves. Diabetes is a chronic disease that affects millions of people worldwide and is a major cause of disability and death. Accurate forecasting of diabetes can help healthcare providers to better manage the disease and reduce the risk of complications. Machine learning has emerged as a powerful tool for diabetes forecasting, allowing researchers to analyze large datasets and identify complex patterns that may be difficult to detect using traditional statistical methods.

A. Gradient Boost Decision Tree

To ensure maximum performance, several Machine learning and Deep learning models have been tested. In order to have predictions that were as close as possible to reality, the model with a lower logloss was chosen: a Gradient boost decision tree. A decision tree is a highly reliable algorithm [2] which poses several questions, each of which narrows down the possible values until you are confident enough to make a prediction. For each answer, there are separate branches. Regardless of the answers to the questions, at the end a prediction is reached (leaf node). Gradient Boosted Decision Tree use an ensemble of decision trees to predict a target label. Ensemble Learning [3] refers to the process of using multiple models at a time or one model several times keeping in mind the average of the results of each model, to build different decision trees in random data points.
Gradient boosting combines weak "learners" into a single strong learner in an iterative fashion.

1) Calculate the average of the target label
2) Calculate the residuals
3) Construct a decision tree
4) Predict the target label using all the trees within the ensemble
5) Compute the new residuals
6) Repeat steps 3 to 5 until the number of iterations matches the number specified by the hyperparameter
7) Once trained, use all the trees in the ensemble to make a final prediction as to the value of the target variable

For this paper, the output of the model corresponds to the probability P that the patient has diabetes.

II. Data preprocessing

In this phase, we performed every tasks that allowed us to improve the quality of the dataset provided to us.

A. Dataset

The dataset contains about 40000 rows and the columns are relative to lifestyle factor, general information about the person and clinical factor:

- Age: 3-level age category 1 = 18-24, 9 = 60-64, 13 = 80 or older
- Sex: 0 = female, 1 = male
- HighChol: 0 = no high cholesterol, 1 = high cholesterol
- CholCheck: 0 = no cholesterol check in 5 years, 1 = yes cholesterol check in 5 years
- BMI: Body Mass Index
- Smoker: Have you smoked at least 100 cigarettes in your entire life? 0 = no, 1 = yes

- HeartDiseaseorAttack: Coronary heart disease (CHD) or myocardial infarction (MI) 0 = no, 1 = yes
- PhysActivity: Physical activity in past 30 days - not including job 0 = no, 1 = yes
- Fruits: Consume Fruit one or more times per day 0 = no, 1 = yes
- Veggies: Consume Vegetables 1 or more times per day 0 = no, 1 = yes
- HvyAlcoholConsump: Adult male: more than 14 drinks per week. Adult female: more than 7 drinks per week. 0 = no, 1 = yes
- GenHlth: Would you say that in general your health is: (scale 1-5) 1 = excellent, 2 = very good, 3 = good, 4 = fair, 5 = poor
- MentHlth: Days of poor mental health scale 1-30 days
- PhysHlth: Physical illness or injury days in past 30 days scale 1-30
- DiffWalk: Do you have serious difficulty walking or climbing stairs? 0 = no, 1 = yes
- Hypertension: 0 = no hypertension, 1 = hypertension
- Stroke: 0 = no, 1 = yes
- Diabetes: 0 = no diabetes, 1 = diabetes (Target variable)

B. Duplicate rows

First of all, duplicate data have been eliminated, to avoid training or testing the same observation twice, and to avoid data leakage, the presence of test set data within the training set.

C. Number to string

While analyzing every dataset's attribute, we noticed that the target variable Diabetes's type was Integer, and we need it as a String, in order to process it into the model.
Simply, we converted its type through the use of a appropriate node.

D. Partitioning

A partitioning operation with stratified sampling was then performed, dividing the total dataset into training set (80%) and test set (the remaining 20%). This last part will then be used to test the model, and the subsequent operations will be implemented only through apply nodes, in order to avoid data leakage also in this case.

E. Missing values

The presence of missing values was checked.
In the dataset provided to pursue the training of the model, there were no NaN, therefore no further operations were necessary. However, looking forward to the deployment where the user has the possibility to upload his own file, the workflow was designed to verify this point, in case the new dataset contains null values. In this case, the NaN values will be replaced with the most present value of the same column.

F. Outliers

We wanted to analyze the outliers values of every variable, in order to exclude rows that could bring noise into our model. The Box plot shows the presence of some outliers in the following variables:

- MentlHlth
- PhysHlth
- BMI

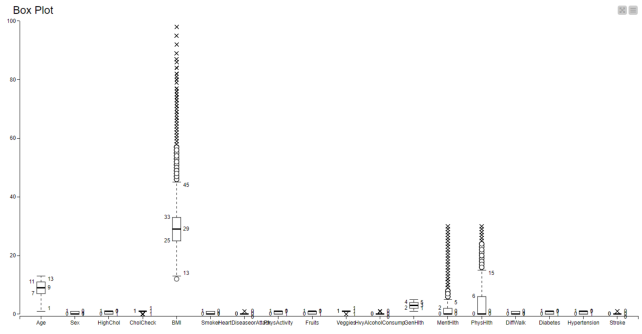

Fig. 1. Outliers boxplot.

Regarding MentHlth and PhysHlth, the observations were within the scale provided by the specification, so the best decision was to not remove the values because we considered relevant to analyze all the people, regardless of their health issues. However, the outliers in the BMI variable have been removed, because according to the definition of the Body Mass Index, the value can be between 12 and 45 [4].
Therefore, values less than 12 and greater than 45 were excluded as these are possible input errors. In this way, we tried to obtain a cleaner set of data to give in input to the model.

G. Column merger

In order to decrease the dimensionality of the dataset, we decided to merge Fruits and Veggies variables, both binary variables, creating a new column called healthy_food, which is the OR combination of Fruits and Veggies.
The fruit and vegetable columns were then removed.

H. Normalization

The large difference in the scale of numbers could cause problems when trying to combine the values as features during modeling. Normalization avoids these problems by creating new values that maintain the overall distribution and proportions in the source data, keeping the values within an applied scale in all numeric columns used in the model. we normalized using a min-max scale.

I. Class imbalance

To avoid Class imbalance of the diabetes class an oversampling was carried out through the SMOTE node which creates artificial data to avoid unbalanced data.

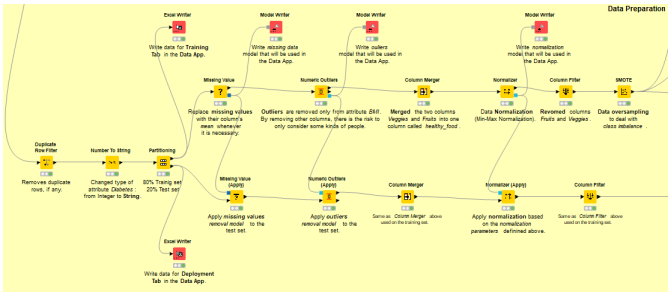

Fig. 2. Preprocessing

J. Feature Selection

Feature Selection is the process of detecting relevant features and removing irrelevant, redundant, or noisy data. This process speeds up data mining algorithms,improves predictive accuracy, and increases comprehensibility. Irrelevant features are those that provide no usefulinformation, and redundant features provide no more information than the currently selected features [5].

A forward feature selection was used, that is an iterative method that starts with no feature and adds the feature that improves the model the most at each iteration

The learner and the predictor of the model are present inside the loop of the feature, the data that allows the training of the model is entered as input to the learner, and the data that allows it to be tested as input to the predictor. These two sets are not fixed, in fact cross validation with 3 folds is used, in order to generalize the learning of the model. This method uses a subdivision of the total dataset into k parts of equal number and at each step the kth part of the dataset becomes a validation set, while the others constitute the training set. In this way the model is trained on k different parts as to avoid overfittings problems. LogLoss was used for the evaluation, it is indicative of how close the prediction probability is to the corresponding actual/true value. The more the predicted probability diverges from the actual value, the higher is the log-loss value. [6]

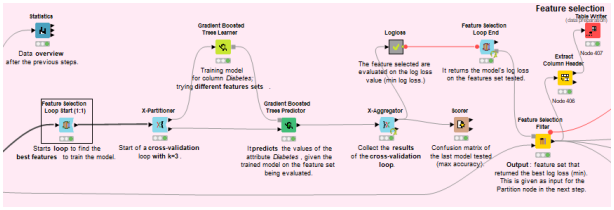

Fig. 3. Feature Selection

III. Model Optimization

Before proceeding with the testing phase, it is useful to optimize the model, by performing a validation on its parameters in order to select those that carry a minor error.

As far as the GBDT is concerned, the parameters on which it may be necessary to validate are:

- Limit number of levels (tree depth): number of tree levels to be learned. For a Gradient Boosted Trees, usually a depth less than 10 is sufficient. Larger trees will quickly lead to overfitting.
- Number of models: the number of decision trees to learn. A ”reasonable” value can range from very few (say 10) to many thousands for small data sets with few target category values.

Similarly to what was shown a little while ago, there is a loop which varies the parameters entered into the model at each iteration, in our case the depth (“depth”) can vary from 1 to 10 and the number of models (n_models) from 1 to 500. A search strategy called Hillclimbing is used :A random start combination is created and direct neighbors are evaluated. The best combination of neighbors is the starting point for the next iteration. If no neighbor improves the objective function, the cycle ends. As before, cross validation was used to avoid overfitting, and the parameters that generate a lower logloss are selected.

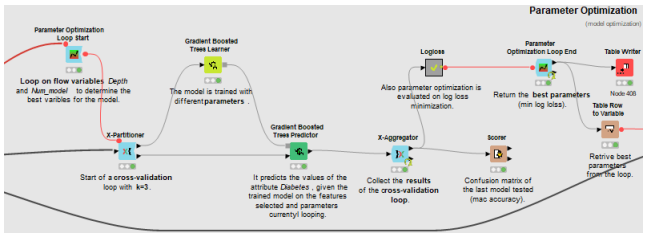

Fig. 4. Parameter optimization

IV. Model Testing

Now that the suitable parameters and features have been selected, it is possible to test the model before the deployment.

The data used for validation is entered into the learner’s input, while the test data is entered into the predictor’s input, partitioned at the beginning of the workflow, so as to have ”hidden” data that can be used to simulate a prediction. As done so far, model evaluation is done via logloss. Logloss measures the error between the model’s predictions and the actual classes of test data, expressing the error as the negative logarithm of the predicted probability for the correct class. This formula penalizes incorrect predictions most with a predicted probability close to 0 or 1, while it penalizes least predictions with a predicted probability close to 0.5.

The final result leads to a log loss of 0.531 and to an accuracy of 73.6%.

Row ID	I TruePositives	I FalsePositives	I TrueNegatives	I FalseNegatives	D Recall	D Precision	D Sensitivity	D Specificity	D F-measure
0	2414	831	2939	1089	0.689	0.744	0.689	0.78	0.715
1	2939	1089	2414	831	0.78	0.73	0.78	0.689	0.754

Fig. 5. Accuracy statistics

Finally, the ROC curve was plotted. The Receiver Operating Characteristic Curve (ROC) is a curve showing the performance of a classification model at different probability thresholds.

The ROC plot is therefore obtained by plotting FPR and TPR on the Cartesian frame of reference, where FPR (False Positive Rate) is plotted on the x-axis and TPR (True Positive Rate) is plotted on the y-axis for different probability threshold values including between 0.0 and 1.0.
In general, the model is more accurate the closer its ROC Curve is to the upper left corner of the graph. This goodness value is called Area Under the Curve (AUC), takes values in [0,1] and represents a measure of accuracy: the higher the value, the higher the accuracy [7].

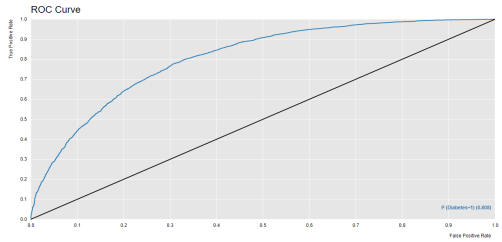

Fig. 6. ROC Curve.

V. Deployment

The data app is available online in order to allow users to upload their data in a xlsx file and test the model.
To make it possibile, we used KNIME's Workflow Integration extension, in particular the trained model on the previous dataset. Inside this app, it is possible to upload a file and visualize the dashboard created after running the trained model and interact with some charts in order to have a first explorative analysis.
The explorative analysis section includes:

- Statistics table with overall of dataset
- Bar Chart
- Box Plot

So, after uploading the dataset, the model will process it and it will show the relative interactive section:

- ROC curve
- Scorer with all the results of the model
- Log loss value

It is also available a button to allow user to download all the computed metrics given as output by the model.

VI. Conclusion

In conclusion, from the point of view of accuracy, the model predicts correctly the condition in about 3 out of 4 people. Another important factor is related to the confusion matrix. In fact, there is a smaller number of FN than FP, and this means that fewer errors were made in predicting an unidentified diabetic patient, compared to identifying a non-sick patient as diabetic.
A log loss value of 0.51 indicates that the model's predictions are reasonably close to the true values, meaning that it is making accurate predictions for the majority of cases.
Therefore, the threshold for what is considered a "good" log loss value may depend on the specific requirements also on the costs associated with different types of errors(false negatives may be more costly than false positives in this context).
In any case, a log loss value of 0.51 is a positive sign and indicates that the model is performing better than a random guess and may be useful for diabetes prediction.

VII. Bibliography

References

[1] https://www.ibm.com/it-it/watson-health/learn/artificial-intelligence-medicine#: :text=Cos'
[2] S. Ravikumar and P. Saraf, "Prediction of Stock Prices using Machine Learning (Regression, Classification) Algorithms,",2020
[3] Martin Sewell "Ensemble learning", 2011
[4] https://patient.info/doctor/bmi-calculator-calculator
[5] Vipin Kumar and Sonajharia Minz,Feature Selection: A literature Review,2014
[6] https://towardsdatascience.com/intuition-behind-log-loss-score-4e0c9979680a
[7] https://www.med4.care/curva-roc-receiver-operating-characteristic-introduzione-e-applicazione-ai-test-diagnostici/