University of Milano-Bicocca
School of Science
Department of Informatics, System and Communication
Master degree in Data Science

Diabetes prediction

Report for Machine Learning project and report for KNIME challenge

Abstract
This article discuss about predicting diabetes with a set of easy to get features (does not require medical visit) through the KNIME platform. We compare various models and, after chosing MLP, we optimize it's parameters and make feature selection. We also create two data apps that can be deployed to a KNIME server: an interactive dashboard for data exploration and an app to predict diabetes with the chosen trained model.

Andrea Muscio [889901], Samir Doghmi [897358], Edoardo Fava [851665]

17 February 2023 - academic year 2022-2023

Contents

1 Introduction

Diabetes is a chronic disease that affects millions of people worldwide. It is characterized by high levels of glucose in the blood due to defects in insulin production, insulin action, or both, which can lead to serious health complications if not properly managed. One of the biggest challenges in managing diabetes is the early detection and diagnosis of the disease.

This report is one of the deliverables for a KNIME challenge with fixed goals for all participants.

1.1 Goals

The goals of this project are defined by the KNIME challenge committee.

- Design and development of an interactive dashboard (a KNIME data app) for univariate and multivariate data visualization
- Train and deploy a supervised classification model to detect diabetes

The model performance will be evaluated by the committee with log-loss (see section 1.4).

In this article, we propose a machine learning approach using KNIME to classify early stage diabetes cases. We will use a variety of algorithms, including Decision Trees, SVM (Support Vector Machine), MLP (Multi-Layer Perceptrons), and Naïve Bayes, to classify patients as either diabetic or non-diabetic. We will use log-loss as the primary metric for comparing the performance of these models.
The data used in this study is provided by the KNIME challenge committee, which contains a large dataset of anonymized patients information. This data will be used to train and test the machine learning models.
The goal of this study is to demonstrate that machine learning algorithms can be used to classify early stage diabetes cases with high performance, and to compare the log-loss of different algorithms to find the best model for this task. We believe that this research has the potential to improve the early detection and diagnosis of diabetes based on features that does not require medical exams, which could lead to better outcomes for patients.

1.2 Dataset

Description of the variables
Our dataset is made of 40108 records each with the following 18 features:

- `Diabetes`: 0 = no diabetes, 1 = diabetes (target variable);
- `Age`: value representing the range of years
  - `1`: 18-24
  - `2`: 25-29
  - `3`: 30-34
  - `4`: 35-39
  - `5`: 40-44
  - `6`: 45-49
  - `7`: 50-54
  - `8`: 55-59
  - `9`: 60-64
  - `10`: 65-69
  - `11`: 70-74
  - `12`: 75-79
  - `13`: 80+
- `Sex`: 0 = female, 1 = male;
- cHighChol—: 0 = no high cholesterol, 1 = high cholesterol;
- `CholCheck`: 0 = no cholesterol check in 5 years, 1 = cholesterol check in 5 years;
- `BMI`: Body Mass Index (BMI);
- `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 time per day 0 = no, 1 = yes;
- `Veggies`: Consume Vegetables 1 ore more time per days 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.

1.3 Workflows

We organized the project in different KNIME workflows, some of them are also deliverables for the KNIME challenge. The following are all the workflows with their responsibilities. Every workflow will be explained better in the next sections of the report.

- data visualization workflow (deliverable): data app that allows data exploration
- model comparison workflow: the workflow we used to compare different models and models parameters
- exported workflow (deliverable): data app that predicts the diabetes
- Diabetes workflow (deliverable): trains the best model and generates the exported workflow with the correct inducer saved inside
- log-loss test workflow: a workflow to compare different log-loss computation approaches

1.4 Log-loss

Let $y_i \in \{0, 1\}$ the class attribute value of a record $i \in \{1, ..., \#\text{ of records}\}$ and let $\hat{y}_i \in [0, 1]$ the model output probability of y_i being 1. We define the local log-loss, the log-loss for one record, as in equation 1, and we define the overall log-loss as in equation 2.

$$\text{logloss}_i = -(y_i \ln \hat{y}_i + (1 - y_i) \ln (1 - \hat{y}_i)) \tag{1}$$

$$\text{logloss} = \frac{1}{n}\sum_{i=1}^{n}\text{logloss}_i \tag{2}$$

In order to calculate the log-loss in KNIME, we tried different approaches: the math formula, an equivalent formula using an `if` switching between the two log and both the previous with lower and upper bounds applied to the probability. Our final choice is the math formula with the fix because it's the standard formula and because the fix is a standard fix necessary to avoid computing $\log 0$. The final formula we used to calculate log-loss is equation 4, where s stands for sensibility and it's the minimum safe value of double numbers stored in the computer memory, typically 10^{-15}.

$$\text{fix}(x) = \max\left(s, \min\left(1-s, x\right)\right) \tag{3}$$

$$\text{logloss} = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \ln \text{fix}(\hat{y}_i) + (1-y_i)\ln \text{fix}(1-\hat{y}_i)\right) \tag{4}$$

To go in the details of the various computing options and the computing of the sensibility value s, refer to the log-loss test workflow along with the internal comments and documentation.

2 Data visualization

Data visualization is an important part of the project because it allows humans to understand the data and the correlations between them, additionally it's one of the three main points of the Knime workflows to be delivered.
This section is divided into the following parts that corresponds to subsections.

- Introduction: we introduce to the Diabetes problem with some history and a timeline;
- Single values: one simple graph per feature;
- Correlation: first page that visualizes relations between features;
- Exploration dashboard: truly interactive dashboard for data and correlations exploration;
- Data Explorer: "Data Explorer" block visualization.

For all the graphs we disabled view controls because we want the data app to be simple, this way the user can be focused on the graph and selection without distractions about the presentation of the data. We chose a good set of view settings in a way that live settings were not necessary to the user. We also disabled selection where it was not necessary, to improve performance in cases like the single values page. In the single values page the selection was not necessary because interactivity is not needed in a page where every graph is independent from the others.

3 Preprocessing

The provided data is very clean, so a minimal preprocessing is needed. In particular, there are no missing values.

To partition the data, we arbitrarily decided to use a stratified sampling on the class attribute Diabetes. Since we evaluate the models on 10-fold validation, we train the model on 10% of the dataset. This ensures that the predicted log-loss is consistent with the inducer log-loss and avoid overfitting.

After the model testing we need the log-loss value; in order to compute it we use a mathematical formula in the Math Formula block. Unfortunately, this block can only handle numerical values, while we need the `Diabetes` value in the formula whose type is `String`. The solution we propose is to add a column called `Diabetes (int)` that contains the exact values of the `Diabetes` column with a different attribute type. We decided to add this data transformation at the beginning of the workflow, because it's more performing than transforming the `Diabetes` values from `String` to `Integer` for every model in our model comparison workflow. On the downside, it requires to filter `Diabetes (int)` before any model training since there is a 100% correspondence between the two columns. This choice also allows better readability of the workflow when calculating log-loss, because the log-loss calculation metanode would have had more blocks. So in all our workflows regarding training there will be two columns related to the class variable in the main data table: `Diabetes` and `Diabetes (int)`.

4 Choice of the model

4.1 Models committed

Different Machine Learning models have been picked to forecast the value of diabetes. The following is the list of KNIME nodes employed:

- Heuristic Models: Heuristics are problem-solving or decision-making techniques that use a minimum of significant information, past results, and background to produce a feasible and practical solution to a given problem in a reasonable amount of time. Those strategies focus on providing quick results with an acceptable range of accuracy. In This paper, the emphasis was placed on the implementation of Random Forest (RF), the Decision Tree (DT), and J48
- Regression Based Models: Logistic regression is a popular method to model binary data. It's used to predict dichotomous dependent variable based on one or more continuous or nominal independent variables. We tried the Logistic and SimpleLogistic algorithms.
- Probabilistic Models: They exploit Bayes formula and compute posterior probability to classify records. Among the most widely used methods are those based on probabilistic assumptions and we chose to use the Naive Bayes, NBTree and more in general BayesNet, where all features are considered as attributes and are independent given the target class;
- Artificial Neural Network: Among the classifiers belonging to this category of algorithms has been used the Multilayer Perceptron, which uses the technique of backward propagation error to classify the instances;
- Support Vector Machine Models: The main idea is that based on the training data, the algorithm tries to find the optimal hyperplane which can be used to classify new data points. SPegasos, and few different kernel caching techniques for SVMs, have been explored for predicting the target variable.

4.2 Performance measures

It is often difficult to state which metrics are the most suitable to compare different models, in other words the capability of the classification model to give reliable predictions on new record, since each has specific features that measure various aspects of the evaluated data.

The efficiency of any classification model is determined using a Confusion Matrix, which partitions the data as True Positive (TP), False Positive (FP), True Negative (TN), and False negative (FN). These four different types of errors are used in calculating many evaluation metrics for classifiers such as Accuracy, Recall, Precision, F1-measure and Roc Curve. Below we briefly introduce them.

- accuracy $= \dfrac{TP + TN}{TP + TN + FP + FN}$
- recall $= \dfrac{TP}{TP + FN}$
- precision $= \dfrac{TP}{TP + FP}$
- F1-measure $= \dfrac{2 * precision * recall}{precision * recall}$

Another method of evaluating the model is to plot the ROC curve, which relates the percentage of false positives against the percentage of true positives at different threshold levels. Once the curve is obtained, the area under the graph (AUC) can be computed, that value can denote how well the model is able to distinguish between classes.

4.3 Cross-Validation

In k-fold cross-validation, each data point is entered into a test set exactly once and into a training set $k - 1$ times. The disadvantage of this method is that the training algorithm must be rerun from zero k times, which is computationally expansive. For SMO(SVM-puk), it was decided to take 50% of the data-set as a sample for performance reasons.

The outcomes of a cross-validation test k-fold are often summarized with the average of the performance measure taken to evaluate the model. The value assigned to the variable k for this analysis is set equal to 10. In addition, to ensure that the class variable is not over/under-represented in the sets, it was implemented the stratified sampling with regard to the class attribute `Diabetes`. Also, to provide the same seed and the same number of iterations in the cross validation nodes, we dynamically connected three different flow variables to the x-aggregator node: number of folds, use random seed, random seed.

4.4 First results

We discuss how to interpret the results obtained in our experiments with the support of the tables and graphs below.

Classifier	Recall	Precision	F-measure	Accuracy	AUC
RF	0.705	0.71	0.707	0.702	0.768
DT	0.698	0.717	0.708	0.705	0.768
J48	0.728	0.73	0.728	0.729	0.759
Logistic	0.778	0.741	0.759	0.747	0.822
SimpleLogistic	0.777	0.74	0.758	0.747	0.822
SPegasos	0.929	0.636	0.755	0.693	0.808
SMO (SVM-poly)	0.79	0.702	0.762	0.747	0.746
SMO (SVM-puk)	0.768	0.717	0.742	0.727	0.483
BayesNet (K2)	0.734	0.749	0.742	0.739	0.815
NBTress	0.782	0.736	0.758	0.745	0.815
BayesNet (BNC)	0.734	0.749	0.742	0.739	0.815
NaiveBayes	0.679	0.752	0.714	0.721	0.8
MLP (1h.layer)	0.795	0.736	0.764	0.749	0.824
MLP (2h.layers)	0.801	0.731	0.764	0.748	0.822
MLP (4h.layers)	0.801	0.733	0.765	0.749	0.822

Table 1: Comparing classifiers

As discussed in the purpose section, our aim is to identify the model with the highest performance. An initial analysis does not reveal a model that performs better in classifying and predicting diabetes. In general, classifiers provided an accuracy between 0.693 and 0.749. Performance measures such as Precison, F-measure and AUC showed values with not significant differences among the classifiers. It's worthwide to mention that SPegasos achieved a very high value in recall (0.929). This metric is also regarded as being among the most important for medical studies, since it is desired to miss as few positive instances as possible.
The multilayer perceptrons with 1, 2 and 4 hidden layers performed better in distinguishing between all the positive and negative class points, and showed minimal differences in results.

In figure 1 the ROC curves for the probability of diabetes being equal to 1 are displayed. However, we have shown the four models considered to be the most meaningful. There are crossings between the curves of logistic and MLP models, but these are minimal and does not allow to hypothesize significant trade-offs between the two.

After these assessments, was computed the log-loss on each fold of all the

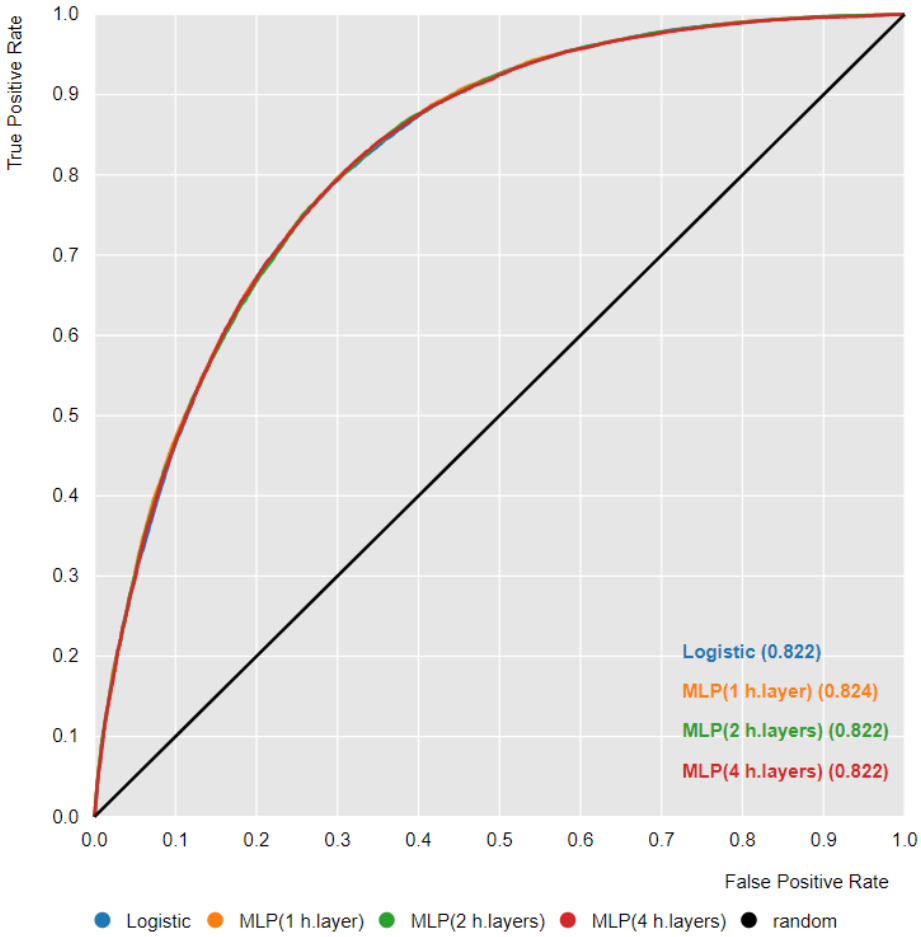

Figure 1: ROC curve of four selected models

models. The scorer of the previous ML methods can be visualized through the box plots of figure 2.

These results reinforce the idea that no model is clearly better at predicting diabetes. The variability of the records for the Logistic and MLP classifiers is very low. Taking a closer look, The lowest score was achieved by MLP 1h.layer, while MLP 4 hidden layers showed the widest interquartile range. Also, for some classification models, it was not possible to determine the percentage probability of the target variable. In fact, the SMO (SVM-poly) model reported a binary probability , which yielded after computation an estimated log-loss of 8.729.

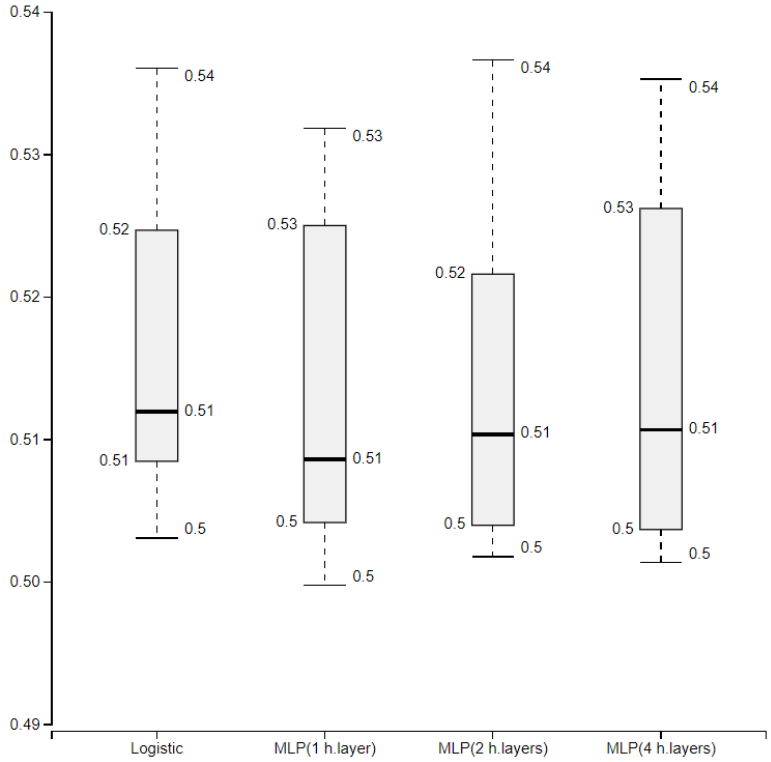

Figure 2: Log-loss of the four selected models

4.5 Feature selection

By taking in account the previously selected four classifiers showed in the box plot, we made the decision to apply a feature selection procedure to discover which attributes were redundant or irrelevant. The major benefit to perform this activity is the reduction in the dataset's dimensionality.
In order to carry out such task, the AttirbuteSelectedClassifier node was employed with the Wrapped approach. Further, in the set-ups, Multilayer perceptron was used as the base classifier and to estimate the accuracy of subsets, after making different attempts. The search was performed with a GreedyStepwise strategy and was left up the node to arbitrarily select the number of input attributes to retain in the output.
The data-set was reduced to 9 dimensions: `Age`, `HighChol`, `CholCheck`, `BMI`, `Veggie`, `HvyAlcholConsump`, `GenHlth`, `Stroke`, and `Hypertension`.

Figure 3 summarizes the log-loss computations corresponding to this approach. In general, we got low-values for almost all performances measure,

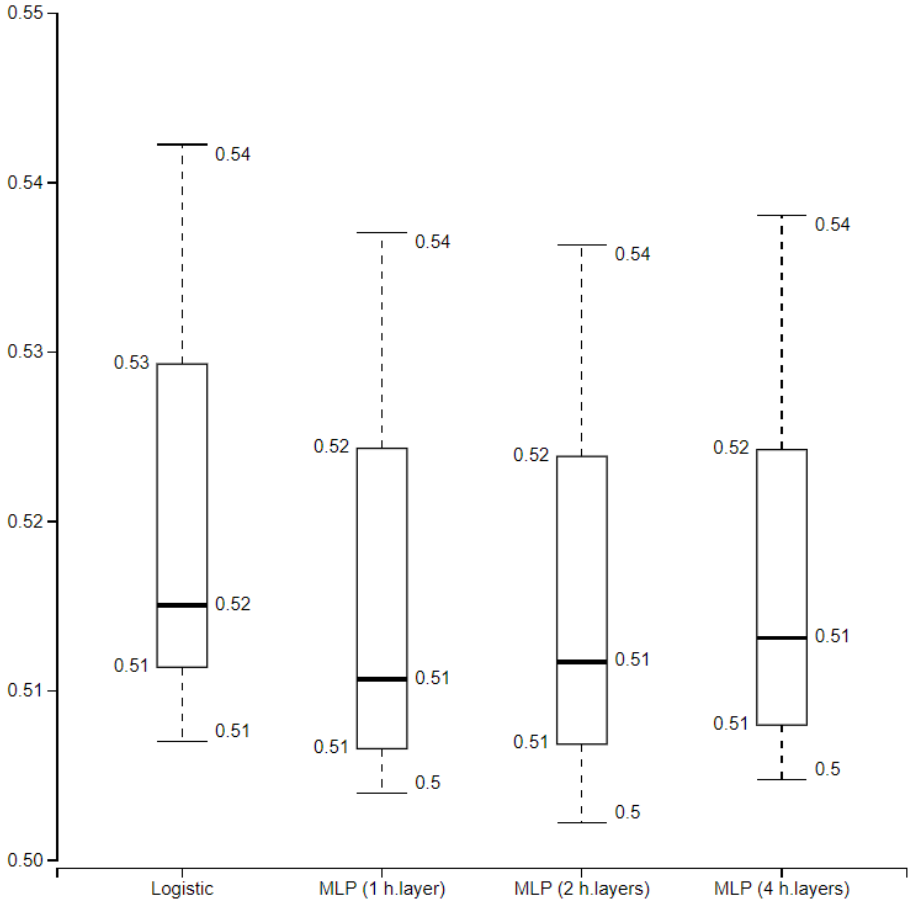

Figure 3: Log-loss computation after feature selection

with a slightly worsening for Logistic node. Minimal adjustment were reported for the MLP's nodes, where the average logloss increased by one hundredth for MLP 2 (0.515) and MLP 4 (0.516) with h.layers.

4.6 Feature selection on Log-loss

Next, it was choose to deploy the "feature selection loop" node interated thorugh four fold cross-validation, to select the "best" attributes and decrease dimensionality. Thus, starting from the set of all variables, the algorithm at each iteration filters an attribute with a Forward Feature Selection strategy. This is a technique that leads to sub-optimal feature selection based on the performance metrics of various clusters. In first step, we tried to maximise accuracy, but we had overall improvements of one hundredth. The same methodology was applied to minimize the log-loss value.

Classifier	Recall	Precision	F-measure	Accuracy	AUC	Log-loss
Logistic	0.778	0.741	0.759	0.747	0.822	0.516
MLP (1h.layer)	0.796	0.735	0.764	0.749	0.824	0.509
MLP (2h.layers)	0.8	0.733	0.765	0.749	0.824	0.513
MLP (4h.layers)	0.8	0.733	0.765	0.749	0.824	0.512

Table 2: Comparing classifiers

Table 3 showed that the MLP 1 hidden layer performs better than the other classifiers by not selecting `smoker`, `veggie`,and `mentHlth` as explanatory attributes.
The value achieved is 0.509, and given such results, we selected this algorithm for the further steps of the challenge.

4.7 Parameter optimization of MLP

Once identified the MLP as best model, we decided to perform parameter optimization. Using the Weka alternative to the default MLP allows greater flexibility, so we decided to change node, confident to find parameters that will increase the performance respect to the original MLP. We applied parameter optimization to the following attributes: hidden layers (also available in original MLP), learning rate (not available in original MLP) and momentum (not available in original MLP).

We used a brute force approach starting from the Weka default parameters for that attributes. In the first iteration we covered many values of the parameters with a big step. Then we perform again the brute force approach with smaller values windows focusing on the more performant values, and a smaller step.
The criteria used to narrow down the parameters window is the log-loss. In particular, we compared the values with box plots and making a rank based on log-loss mean value for each triplet of parameter values. Taking into account the best 10 models, we empirically decides to focus on the most frequent values of the three parameters. We also tried to use a correlation matrix, but that wasn't much useful. We stopped the process when we couldn't notice any improvent. For every iteration, the data was first sampled 50% and then a cross validation stratified on diabetes with 5 iterations was applied.

- 1st iteration
  - hidden layers: 1 to 6, step 1

  - learning rate 0.01 to 0.09 step 0.02
  - momentum 0.01 to 0.09 step 0.02
- 2nd iteration
  - hidden layers: 1 to 5, step 1
  - learning rate 0.0005 to 0.0015 step 0.0002
  - momentum 0.05 to 0.08 step 0.01
- 3rd iteration
  - hidden layers: 1 to 5, step 1
  - learning rate 0.0005 to 0.0009 step 0.0001
  - momentum 0.06 to 0.08 step 0.005
- 4th iteration
  - hidden layers: 2 to 6, step 1
  - learning rate 0.0005 to 0.0008 step 0.0001
  - momentum 0.06 to 0.07 step 0.005

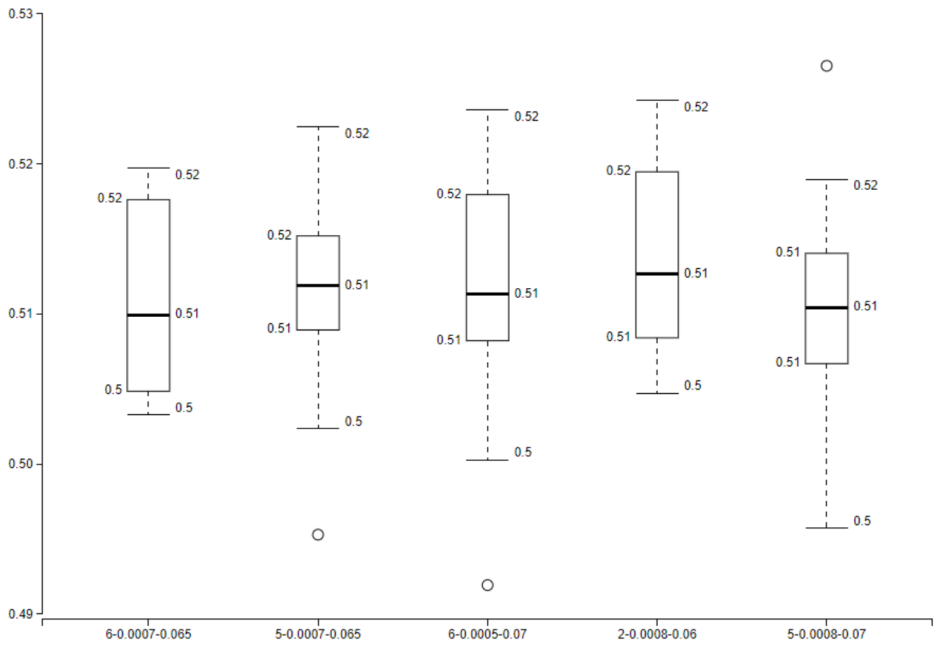

Figure 4: Log-loss of top 5 models with optimized parameters

At the end, the results are not significantly better than the default MLP. Then, we arbitrarily identify the best model as MLP 1 hidden layer.

5 Deployment

The objective is to deploy a data app that can predict the diabetes value with an input having all the training columns but diabetes. In order to train the model one time and then deploy and run undefined times, we decided to create a workflow that uses an already trained model; to do so, we used the KNIME integrated deployment extension. The main idea is to train a model that is saved in the workflow of the data app, without the need to read the model from an external source. The extension allows us to capture part of the workflow and export it as a standalone workflow: we use this feature to create the data app workflow with the trained model saved inside.
We call the workflow responsible for training and exporting the Diabetes workflow, and we call the data app the exported workflow. The Diabetes workflow, for those reasons, is composed by

- a data input part, where training data is read and preprocessed
- a training part where we use the best model with the best settings we found in the model comparison workflow
- a generated workflow part where the exported workflow is run and captured
- an export part where the exported workflow is saved to file and deployed on a server

We later added a scoring part to visualize the performance of the trained model with the chosen method.
The choice of generating the exported workflow has the drawback that changing the model requires changing the whole workflow, but we kept our choice because (1) the generation is a rare event during the lifetime of the data app, (2) the model generation is automatic, (3) the model deployment to a server is also automatic. (4) In the case of exported workflow modification without model change (like a UI change), the deployment performance is the same as exporting a data app detached from the model, if the Diabetes workflow contains the completed workflow data (all the blocks saved with green status).

The generated workflow will be surrounded by the Capture Workflow Start and Capture Workflow End blocks that enable the selection of a given part

of the workflow to save it and eventually run or export it, as in our case.
The internal structure of the data app is simple: based on the minimum necessities, we identified 4 parts for the normal execution that are the following

- data input page: where the user should upload the data
- data reading: parsing of the uploaded data
- prediction: if there is no error in parsing, predict the diabetes value
- results page: allow download and navigation of the results, and visualize the prediction data in a quick aggregated form.

In the case the data reading fails, if the error can be catched by try/catch blocks, we skip the prediction part and, instead of the results page, we show an error page. Clicking "next" when in the error page always makes the workflow fail.
We know that in the case of execution fail KNIME shows all the necessary information, but we taught of separating general errors that could be caused by something or someone different from the user, and parsing errors that are caused by the user input. For this reason we created an informative page for input errors only that shows the user in a different way what is wrong with the uploaded file.

6 Conclusions

We are happy about our results and how we obtained them. We trained many models, and after picking the best we also applied feature selection and optimization, obtaining what we consider a good inducer. We also taught about non-technical aspects, meaning we cared about user experience in the data visualization, that is complete and well organized, and the diabetes prediction workflows, that is simple and effective. All this is automatized and reproducible thanks to KNIME workflows.
We are also happy for our experience, not just the above mentioned results: we had the opportunity to deepen the KNIME software and some models, and of course make into practice our studies.

Possible future developments are the testing of the models based on other measures different from log-loss, and, of course, more model testing. Additionally, a usability test can be made on the data apps user interface in order to improve and explore new possibilities also in that field.

Glossary

accuracy is calculated as the ratio between the number of correct predictions to the total number of predictions and it measures the capability of the classification model to give reliable predictions on new records. it's a valid choice with well-balanced class. 7

data visualization workflow The workflow we used for data visualization and exploration, it's a data app and it's one of the deliverables. 3

Diabetes workflow The workflow that trains the best model and creates the exported workflow. 3, 14, 16

exported workflow A workflow generated by the Diabetes workflow that predicts the diabetes, it's a data app and it's one of the deliverables. 3, 14, 16

F1-measure is the harmonic mean of both recall and precision, meaning that it penalizes extreme values of either. 7

Forward Feature Selection strategy Select the features that improve most the model at each iteration. 11

GreedyStepwise Performs a greedy forward or backward search through the space of attribute subsets. 10

log-loss the log-loss metric as defined in sub sub section 1.4 in a mathematical form. 1, 3–5, 8, 10–12, 15, 16

log-loss test workflow The workflow we used to test different log-loss computation approaches. 3, 4

model comparison workflow The workflow we used to compare models, it's very big and computationally expensive. 3, 5, 14

precision it indicates how many predictions are actually positive out of the total positive predicted. A high Precision value results in a lower number of false positives. 7

recall is defined as the ratio of the total number of correctly classified positive classes divide by the total number of positive classes. This metric

is also regarded as being among the most important for medical studies, since it is desired to miss as few positive instances as possible, which results in high recall. 7

Wrapped approach follows a greedy search approach by evaluating all possible combinations of features against the evaluation criteria. 10

Acronyms

BMI Body Mass Index. 2

CHD coronary heart disease. 2

MI myocardial infarction. 2