Banco de Portugal classification about companies being late in reporting tasks

Introduction/ problem description

Banco de Portugal has been observing that some companies often lag behind when submitting their information reports. Thus, the purpose of this project is to find whether it is possible to classify the companies as late in the reporting tasks or not, based on financial and economical aspects concerning those companies, and also to acknowledge the attributes that contribute to this situation.

For this, two datasets were made available: a train set and a test set. Both comprise 55 variables:

- an identifying variable, *i*;
- forty continuous variables, *x1 to x40*;
- twelve nominal categorical variables, *c1 to c12*;
- two ordinal categorical variables, *o1 and o2*.

The train set also contains information about the target variable, *y*, which is binary (taking the value 0 if the company is on time in the submission of the report or 1 if the company is late in the submission).

Whilst the train set has 5832 observations, the test set includes 5843 observations.

The softwares chosen to work on this project were R and Knime.

Preprocessing methods

First, it was found that the target variable is quite unbalanced, having approximately 64% of its values equal to 0 and 36% equal to 1. This means that most companies deliver their reports on time, making it harder to predict when they are going to submit them late.

The analysis of the variables was started by studying the missing values of the data sets.

In the train set, 22 of the 55 variables mentioned above present missing values. It is important to note that the variables that present the most missing values are *x13* to *x18* (more than 1000 missing values in each of these variables), being these variables responsible for approximately 84% of the total of missing values in the train set. The same happens in the test set.

The first preprocessing method tried was to remove the variables *x13* to *x18*, but it didn't result in a better score in the Kaggle platform, when compared to using no preprocessing at all.

The second method applied was the MICE (Multivariate Imputation by Chained Equations) package of R. It was used in both the train and test sets. MICE imputes missing values with plausible data values and each variable has its own imputation model. The random forest method was chosen to draw these plausible values. Even though it was expected that this method would improve the results in the Kaggle platform, it didn't.

After this, an assessment of the correlation among the continuous variables and the target variable within the train set was made, as seen on Fig. 1. Several pairs of highly correlated variables were found, which suggested great redundancy among the train dataset and the need to proceed to feature selection.

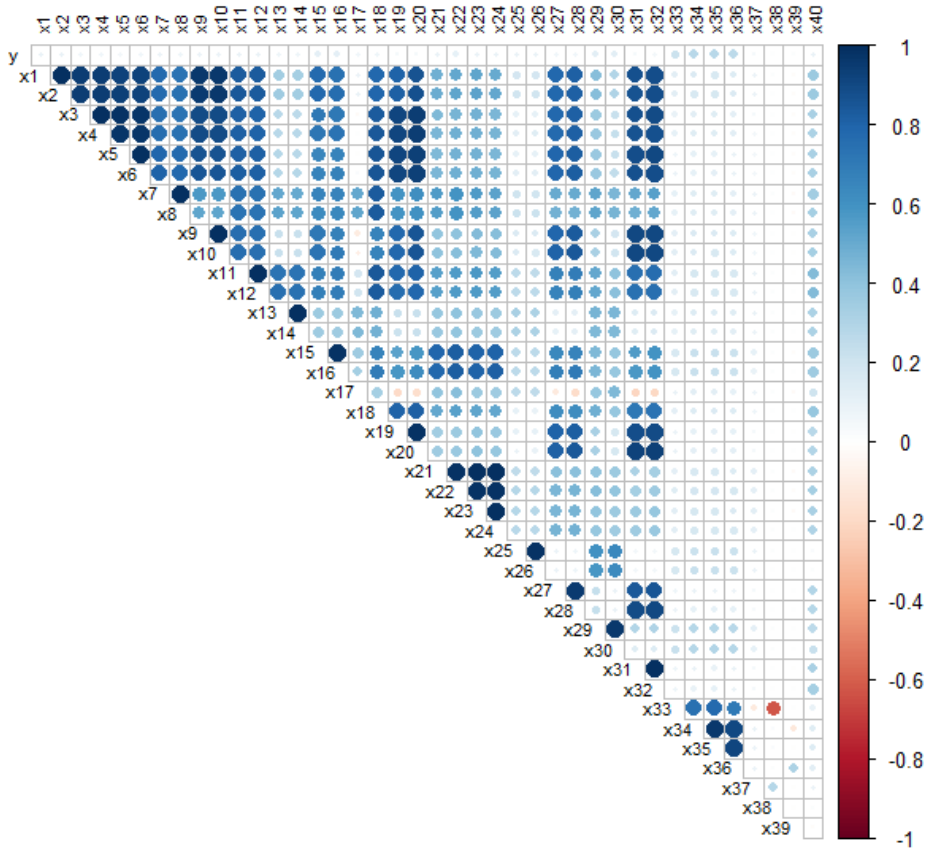


Fig. 1. Correlation matrix for continuous and target variables from the training data set.

The first approach to feature selection was made using the DALEX package in R, to find the most important features in one of the initial models that were developed. The function *model_part* disclosed the root mean square error loss after permutations within each variable. The variables c12, c7, x35, x6, c4, x22 emerged from the plot with greatest RMSE loss, and were selected to rerun the same predictive model, removing all the remaining. However, this preprocessing step didn't lead to better results in the Kaggle leaderboard and it was discarded[1].

Taking into consideration the correlations between the numerical variables, another preprocessing method was tried. From the pairs of variables that presented a correlation of 0,7 or more one of the variables was removed. The obtained training dataset consisted in the following variables: the ordinal and categorical variables, x3, x7, x11, x13, x15, x17, x21, x25, x26, x29, x33, x34, x36, x37, x39 and x40, After the conducted experiments it was possible to see that this method did not improve the Kaggle score.

Next, we tried to do the feature selection on knime to find the set of features that could give us the best accuracy for the predictions.

[1] The group acknowledged, on a later stage of the work, that RMSE loss is appropriate as a metric for continuous variables importance assessment, but not for binary variables. However, a decision was made to not repeat this preprocessing with another metric, given that this predictive model was far from being our best submission in the competition.

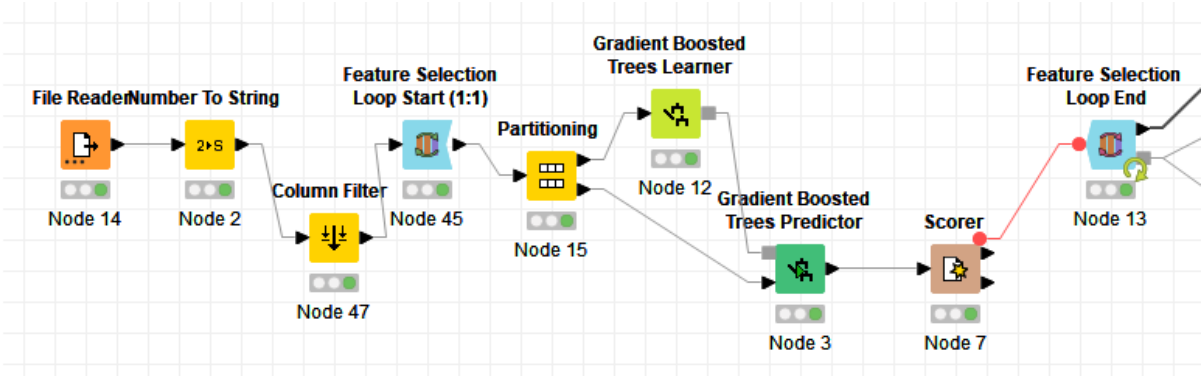


Fig. 2. knime workflow for the feature selection

We ran the feature selection loop on a Gradient Boosted Trees model using the Genetic Algorithm as the feature selection strategy. After the tuning of the parameters of the genetic algorithm that allowed us to get the best accuracy we set the advanced settings as it is below.

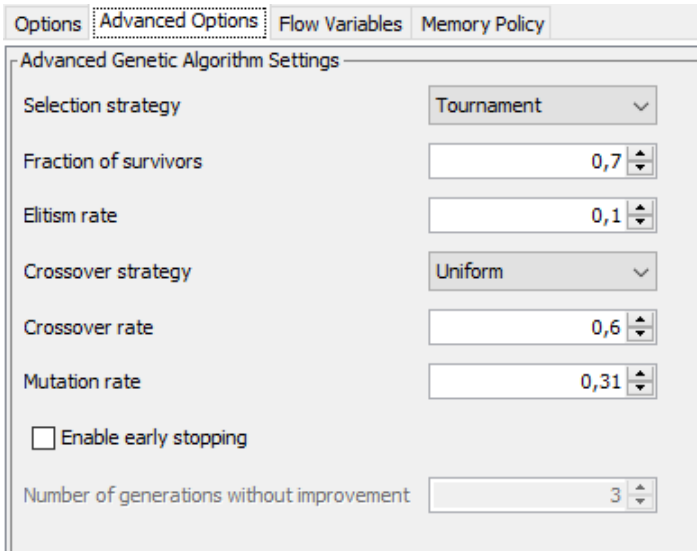


Fig. 3. Advanced options for the Genetic Algorithm

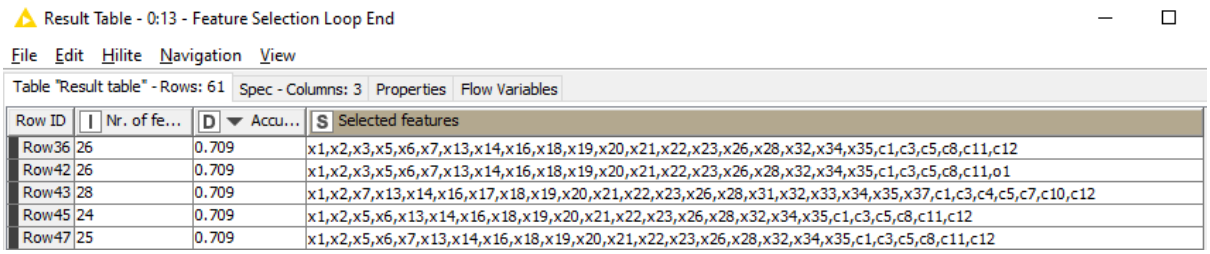
After running this loop and analyzing the scorer node we got an accuracy of 0.692, the confusion matrix and other metrics for the model used are the ones below.

Table "default" - Rows: 3 | Spec - Columns: 11 | Properties | Flow Variables

Row ID	TruePo...	FalsePo...	TrueNe...	FalseN...	Recall	Precision	Sensitivity	Specificity	F-meas...	Accuracy	Cohen'...
0	731	321	76	39	0.949	0.695	0.949	0.191	0.802	?	?
1	76	39	731	321	0.191	0.661	0.191	0.949	0.297	?	?
Overall	?	?	?	?	?	?	?	?	?	0.692	0.17

Fig. 4. Accuracy statistics

Analyzing the results given by the feature selection loop we got 61 different results with the number of features going from 18 on the smallest set to 35 on the larger. The accuracy went from 0.653 to 0.709 on the most accurate set of features and was the one with the best accuracy that was chosen to proceed with the predictions for the test dataset.
And with these results we could conclude that making the predictions with feature selection was the way to go to get the best score possible.
The best set of features is the one below with 26 features, 20 from the continuous variables, 6 from the categorical variables and none from the ordinal variables.


Result Table - 0:13 - Feature Selection Loop End

File Edit Hilite Navigation View

Table "Result table" - Rows: 61 | Spec - Columns: 3 | Properties | Flow Variables

Row ID	Nr. of fe...	Accu...	Selected features
Row36	26	0.709	x1,x2,x3,x5,x6,x7,x13,x14,x16,x18,x19,x20,x21,x22,x23,x26,x28,x32,x34,x35,c1,c3,c5,c8,c11,c12
Row42	26	0.709	x1,x2,x3,x5,x6,x7,x13,x14,x16,x18,x19,x20,x21,x22,x23,x26,x28,x32,x34,x35,c1,c3,c5,c8,c11,o1
Row43	28	0.709	x1,x2,x7,x13,x14,x16,x17,x18,x19,x20,x21,x22,x23,x26,x28,x31,x32,x33,x34,x35,x37,c1,c3,c4,c5,c7,c10,c12
Row45	24	0.709	x1,x2,x5,x6,x13,x14,x16,x18,x19,x20,x21,x22,x23,x26,x28,x32,x34,x35,c1,c3,c5,c8,c11,c12
Row47	25	0.709	x1,x2,x5,x6,x7,x13,x14,x16,x18,x19,x20,x21,x22,x23,x26,x28,x32,x34,x35,c1,c3,c5,c8,c11,c12

Fig. 5. Top five set of features with the best accuracy

Predictive models

The first predictive model tried was a decision tree from the Rpart package of R, with the default parameters and the method equal to “class”.

This model was first trained with the train set without preprocessing, resulting in a Kaggle score of 67,94%. This was the starting point of the work proposed.

This model was also trained with the train set without the variables *x13* to *x18* (the ones responsible for most of the missing values) and with the train set with the missing values imputed by MICE. In both these experiences, the Kaggle score did not improve. It was concluded that the decision tree model from Rpart is not suited for this problem.

The second predictive model tried was a decision tree from the C5.0 package of R, with noGlobalPruning = TRUE (in order to simplify the tree) and minCases = 10 (minCases = 50 was also tried but it produced worse results in predictions).

This model was trained with the train without preprocessing, resulting in a Kaggle score of 68,356%. That is, the predictions were better than the ones produced by the Rpart package. The model was also trained with the train set with the missing values imputed by MICE, but it resulted in worse predictions (even worse than the ones from Rpart).

The C5.0 model was also trained with only 6 variables (x6, x22, x35, c4, c7 and c12). These variables were the ones considered to be the most important by the DALEX library, which was referred above in the preprocessing methods part. This resulted in worse predictions than the ones produced by the same model trained with the train set without any preprocessing.

After these experiments, several models were trained using autoML in KNIME. It is important to note that the autoML node also automates the preprocessing of the data. Models such as Decision Trees, Random Forests, Logistic Regression, Neural Networks, XGBoost Trees and Gradient Boosted Trees were considered. During these experiments, the previously referred preprocessing methods (except for feature selection) were also tried. Despite this, it was possible to see that the model that produced the better results (considering the score presented in Kaggle) was the Gradient Boosted Trees model without any preprocessing.

The previous conclusion led to a more focused experimentation of the gradient boosted trees model and to the application of a different preprocessing method - feature selection.

Making the feature selection on knime explained before we proceeded for the prediction using only the set of features with the best accuracy.

Filtering the test and train datasets we were able to learn the model on the filtered train dataset and predict with the filtered test dataset and after submitting the results of the predictions we got our best score on the competition of 0,69104.

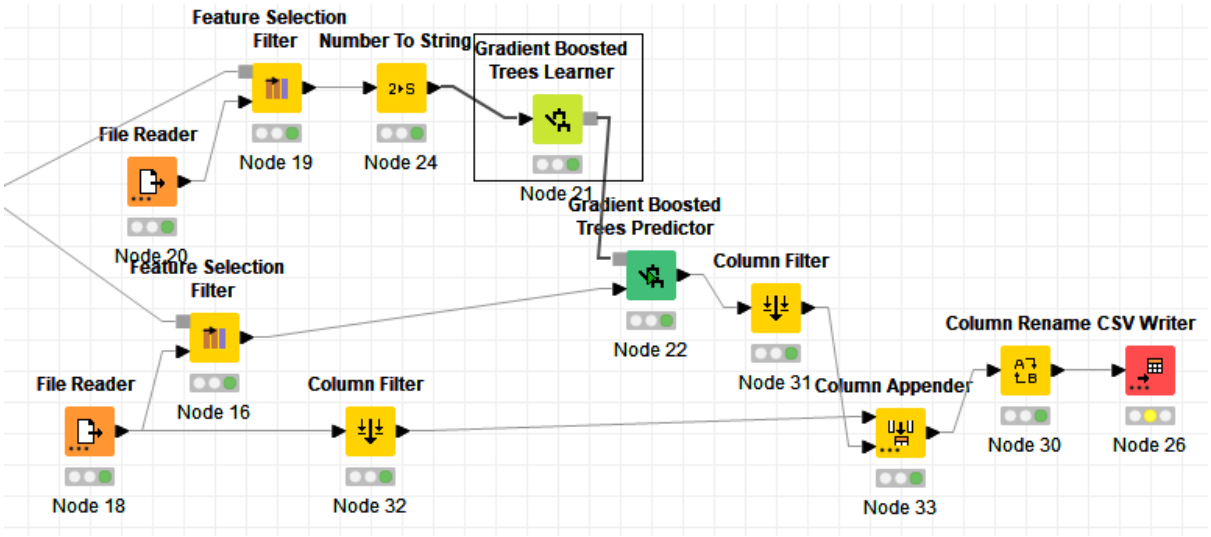


Fig. 6. knime workflow to make the predictions with the set of features with the best score

Accuracy measures

In this part of the report, the accuracy measures for the two best experimented models will be shown and analysed. Consider as model 1 the gradient boosted trees with feature selection and as model 2 the gradient boosted trees model trained with autoML.

These measures were computed considering a partitioning of the training data into 80/20.

Note that the positive class refers to the cases where y=1, when the company is late in the submission of the report. In the context of this problem, it is safe to assume that obtaining false negatives is worse than obtaining false positives. When a company is predicted to be late in their submission and happens to deliver on time is a better situation than when a company is unexpectedly late in their submission. Taking this into consideration, along with the accuracy measure, the sensitivity will also be analyzed since it is assumed the model must be reliable when predicting that a company will not be late (what happens when this last measure is high).

	Accuracy	**Sensitivity**
Model 1	0,688	0,247
Model 2	0,688	0,726

Note that the two analyzed models present the same accuracy but when considering the sensitivity measure it is possible to see that the second model presents a considerably higher value. Taking that into consideration and making the assumptions that were explained above it is possible to say that in the context of the problem it would be more advantageous to use the second model to predict which companies are going to be late in their report submission.

Explainability

In this part of the report, the explainability of the best model experiment (regarding the score presented in Kaggle's public leaderboard) will be explored. The gradient boosted trees algorithm can be considered as a black-box model.

It is also important to note that the Dalex library, previously referred to, constitutes a form of explainable machine learning, but in this specific problem it was primarily used as a form of feature selection.

In this section, different explainability techniques will be applied, such as, feature relevance (SHAP), explanations by simplification (LIME) and visual explanations (Partial Dependence Plots).

Firstly, consider the first observation of the test set. The model predicted that the class assigned to this observation is the positive class (y=1). With the use of the XAI view node in knime it was possible to obtain a local explanation for this specific observation.

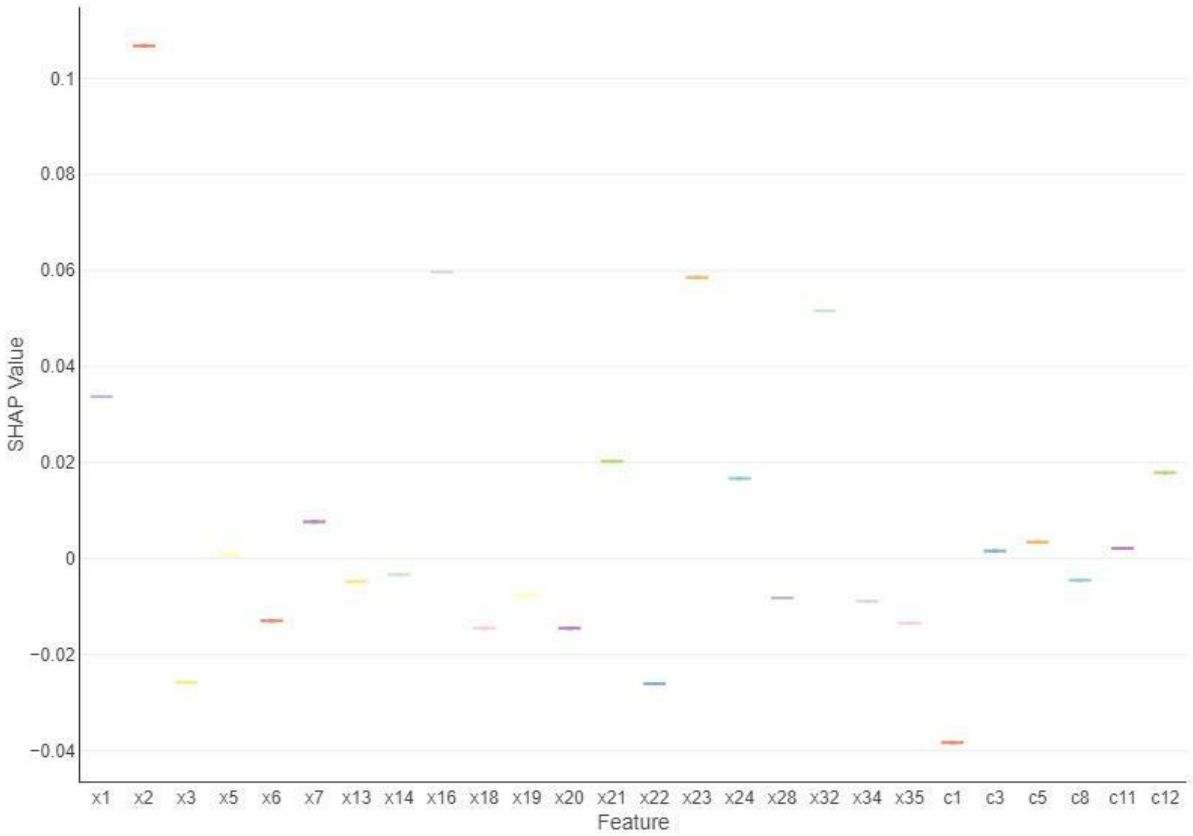


Fig. 2. Shap values for the first test instance by feature.

Analyzing the image it is possible to see that the features that most contributed positively for the classification of this instance were x2, x16, x23, x32; whereas feature c1 was the one that contributed negatively for the classification of the instance.

Considering now the first 100 observations of the test set (maximum number to be handled by the node used). Note that it was assumed that these 100 observations can represent the globality of the observations present in the test set.

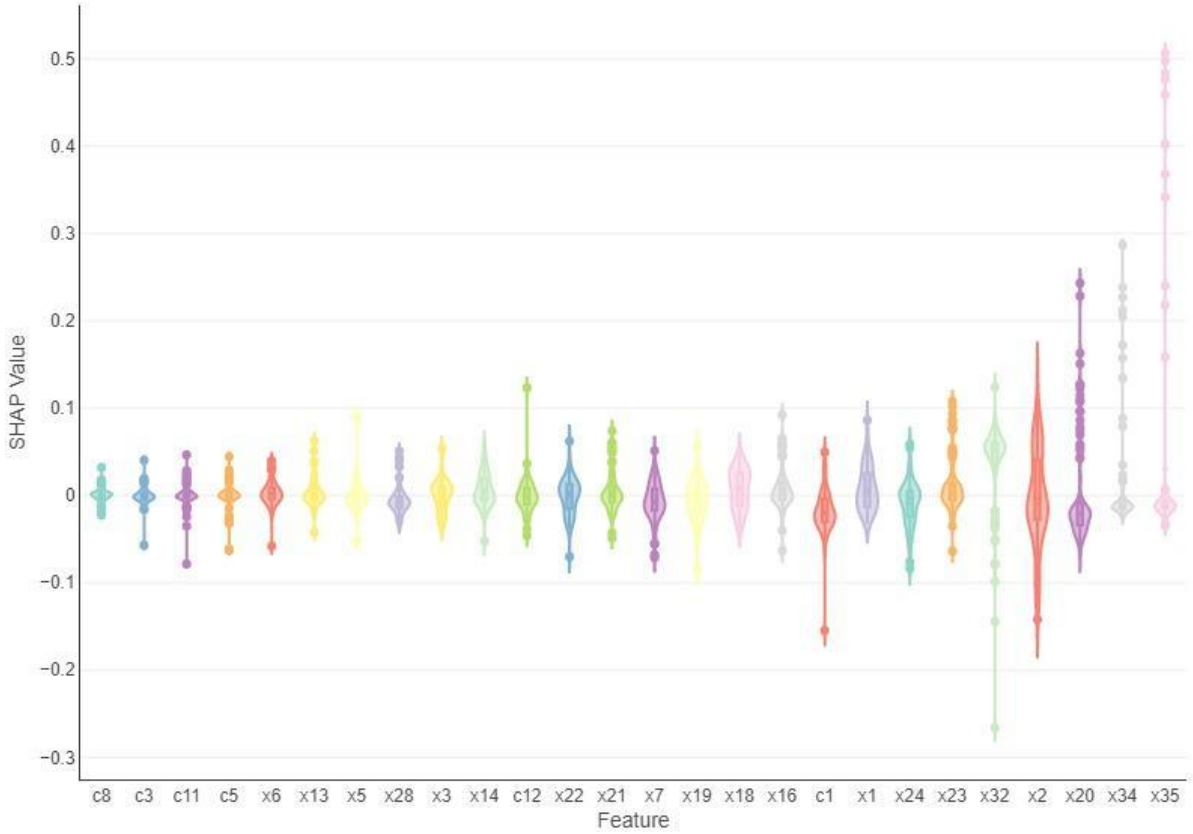


Fig. 3. Shap values for the first 100 test instances by feature.

The image above presents the explanations feature violins. When the shap value is the same for different instances, the wider the violin will be for that SHAP value. Analyzing the image above will provide a global explanation.

It is possible to see that when considering the most dense part of the violins (larger number of instances considered) the SHAP values are not very different from zero. Despite this, note that in some features this wider part of the violin is clearly located either in the positive or negative part of the graphic, which indicates that the value of that feature contributed respectively, positively or negatively for the classification of the test instance as the positive class (y=1).

It is important to note that with this graph it is also possible to see the appearance of some extreme SHAP values; in those specific instances the value of that had a very large predictive power in the model, thus had a very large contribution in the classification of the test instance.