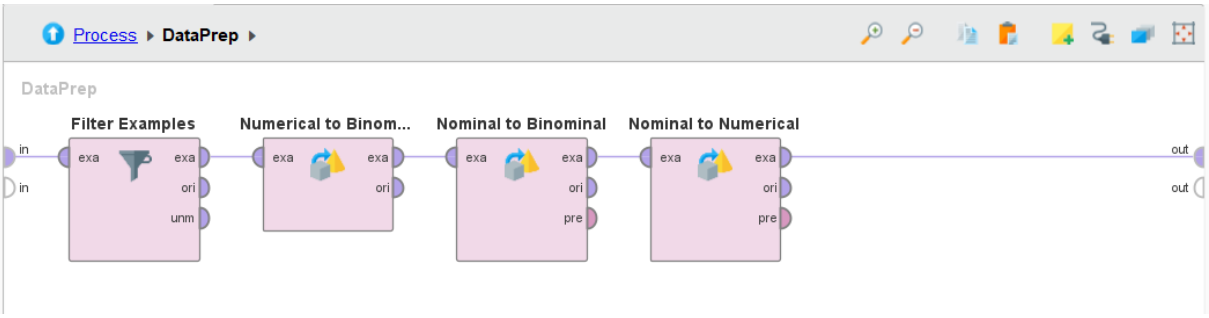
Student: Neda Jevtić

Homework 1: Customer Default Prediction (report)

Firstly, when I have imported the Data Set, I had to change the type of some variables (into binominal, polynominal etc.) because the RapidMiner tried to guess the data types but it does not do it correctly every time. It is important to double check since some of the algorithms/techniques will treat different columns of different data types in different ways. If we hadn´t changed the data type, we could have come across some issues in the following steps. More specifically, I used a *SubProcess* (named *DataPrep*) shown in the picture below. Within it, I included several *Operators* that change the data types of original columns in the following way:



1. Default payment next month (from *integer/numerical*)→***binominal*** type because it contains only two values True=1 (meaning that customer will Default on payment) and False=0 (meaning that customer will not default on payment/they will manage to repay)
2. Gender (from *polynominal*)→**binominal**
3. Education and Marital status (from *polynominal*)→***numerical*** (since many machine learning models cannot operate with variables of polynominal type and also used *unique integers* as coding type)
4. Filter examples→ in the original dataset, columns *Education* and *Marital Status* had some of the values which were equal to 0 (those zeros were probably some errors or outliers). I excluded them by filtering them out. Additionally, I came across some outliers (extremely positive and negative values) in columns such as *Pay_amount_1, Pay_amount_2, Pay_amount_3, Bill_amount_1*....up to *Bill_amount_6*. Such values were also filtered out by using *Filter Examples* operator.

Now, I decided to do some exploratory data analysis because we all know that we should have a good machine learning model, which includes exploring the data, preparing the data and experimenting with data. Additionally, another thing that needs to be done at the end is model evaluation.

After the data preparation, I used an operator Set Roles in order to set the target role for the attribute *„default payment next month"* which is **label**. By using *SMOTE upsampling* operator, the second/minority class (True or in other words, Default on Payment) will be balanced out with the majority class (False or No default on payment). If this has not been done, the model is very likely to predict the majority class every time.

The following step is using an operator *Split Data*, which split the data into 2 parts. One is a training set (70%) and the other one is a test set (30%). The parameter I chose for this operator is *shuffled sampling type* instead of linear or automatic one, because I believed that in this scenario such type of sampling would give more realistic results. This is one of the most important steps because we should never use the training set for evaluating the model.

Then, I applied several machine learning models which were trained and tested by using the operator *Apply Model*. Lastly, in order not to forget to evaluate the performance of the model, we should also use the operator *Performance (Binominal Classification*). Moreover, I also sent out attribute *Weights* to the process result for each model I used in order to see which variables had the biggest influence on predictions.

In this section, I will elaborate a bit more on which machine learning models I used, their results in confusion matrix and which classification method/algorithm I found the most appropriate in our example (Customer Default Prediction).

Decision Tree

As explained during our lectures/seminars, decision tree represents a non-parametric supervised learning algorithm, which is most often used for classification and regression tasks. In the picture below, you can see a table which contains attribute names and corresponding weights delivered through this port.

attribute	weight
Education	0.117
Credit_Given	0.121
Maritial_Status	0.119
Pay_Amount_1	0.106
Pay_Amount_2	0.118
Pay_Amount_5	0.113
Past_pay_1	0.196
Pay_Amount_4	0.109

Withing decision tree algorithm, I have used pruning techniques which can remove parts from the tree which do not have significant power to classify instances. The performance accuracy with applied pruning and pre-pruning does not vary as much compared to non-pruning model. The results are shown in table below.

	Decision Tree (performance accuracy)
w/o pruning & pre-pruning	64.08%
with pruning	64.01%
with pruning & pre-pruning	64.01%

Random forest

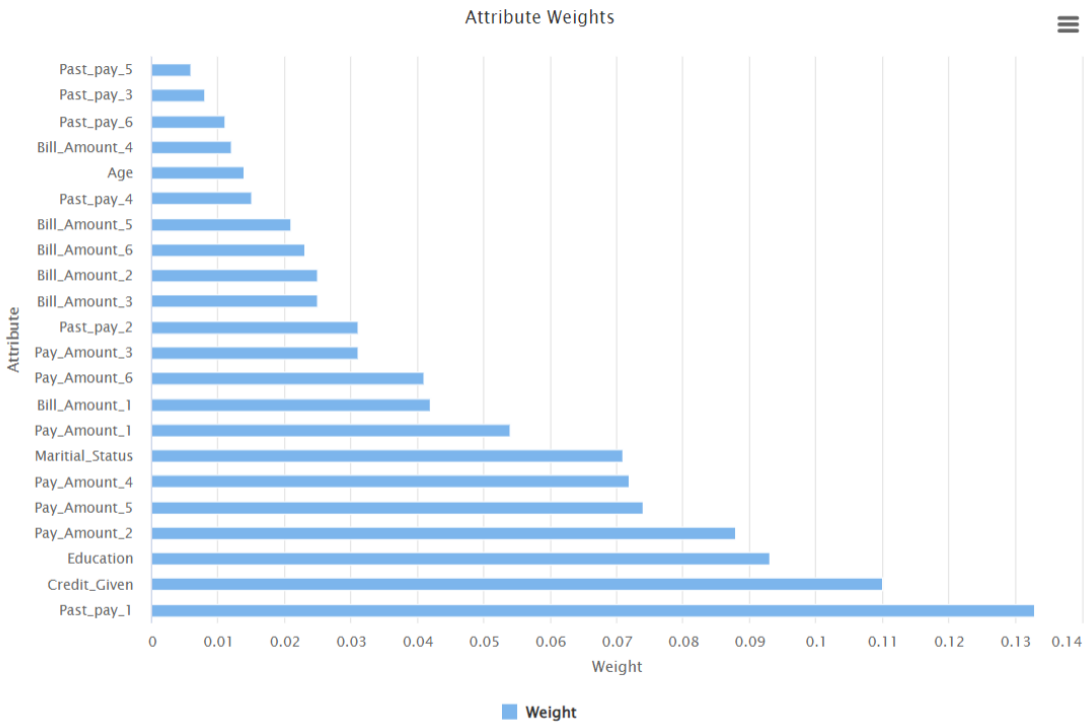
Similarly as Decision Tree, Random forest can handle both classification and regression problems, which is I chose to test it out. The table below represents confusion matrix (with parameters applied pruning and applied pre-pruning).

accuracy: 69.11%

	true false	true true	class precision
pred. false	6403	3695	63.41%
pred. true	620	3250	83.98%
class recall	91.17%	46.80%	

As we can see from the matrix, this algorithm was very good at predicting customers who are capable to repay the debt (in other words, very good at predicting True Falses = 91.17%). On the other hand, what is more important for us in this example is to correctly predict patterns and behaviour that lead to future inability to repay debts, so the algorithm is less successful in predicting True Trues/customers who will likely Default on payment (46.80%).

Student: Neda Jevtić



Logistic regression

Then I decided to test out Logistic Regression model because it helps us predict events that have a binary outcome (such as *Default on payment* or *No default on payment*). It is Supervised Learning technique, used for predicting, as previously mentioned, categorical dependent variable by using a set of independent variables. Confusion matrix is shown below.

accuracy: 67.23%

	true false	true true	class precision
pred. false	4388	1942	69.32%
pred. true	2635	5003	65.50%
class recall	62.48%	72.04%	

As we can see, the overall accuracy is 67.23% (lower than the Random Forest). However, this model is much better at predicting True defaults on payment, which is more important for us in our problem scenario.

k-NN (K-Nearest Neighbour)

It is a supervised Machine Learning algorithm and it can also be used for both regression and classification problem statements. It can be described as being reliant on the idea that similar data points tend to have similar values or labels. If the parameter is set to k=3, confusion matrix looks like this:

accuracy: 85.07%

	true false	true true	class precision
pred. false	6176	1239	83.29%
pred. true	847	5706	87.07%
class recall	87.94%	82.16%	

The overall accuracy is 85.07% which is the highest one so far, compared to above mentioned algorithms. Additionally, it is also very good at predicting both types of customers who are likely to default on payment (82.16%) and the ones who are not likely to default (87.94%).

Gradient boosted trees

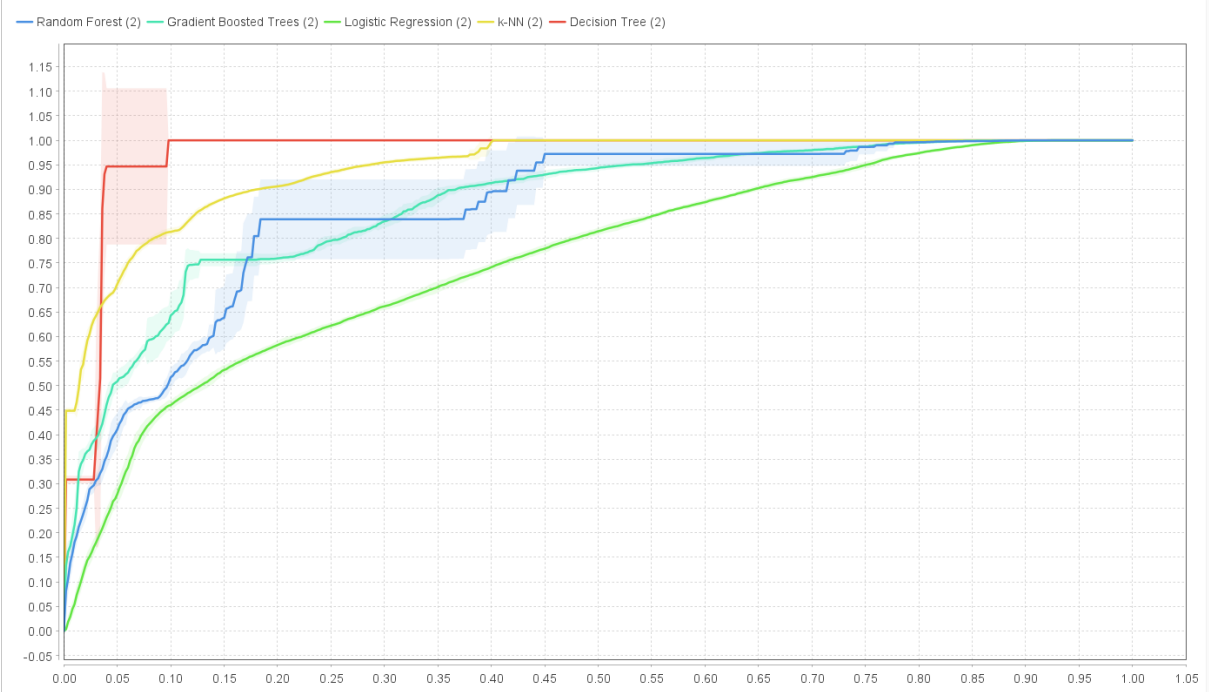
Lastly, I decided to test out this algorithm because, according to theory, is more accurate than some other algorithms and can handle larger datasets. The confusion matrix is shown below.

accuracy: 77.28%

	true false	true true	class precision
pred. false	4522	672	87.06%
pred. true	2501	6273	71.50%
class recall	64.39%	90.32%	

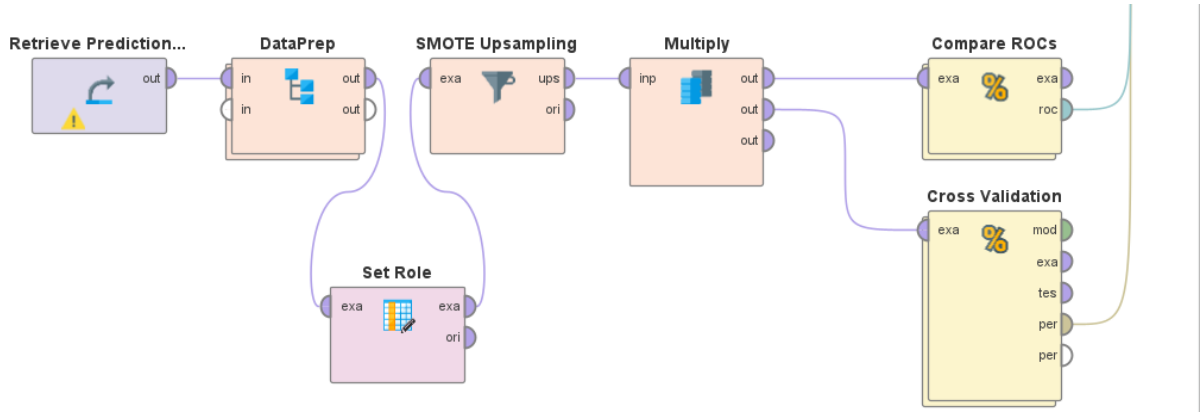
Even though the overall accuracy is lower (77.28%) when compared to the k-NN model (85.07%), this algorithm better at predicting customers who will most likely default on their debt repayment (90.32%) and this percentage is the highest among all of the algorithms I have tested. This percentage can be interpreted as `*We are currently managing to predict 90 out of 100 clients who will have true default on payment´*. On the other hand, it is less successful on predicting customers who won´t default on their payment (64.39%).

ROC graph



Legend: Dark Blue → Random Forest;
Turquoise → Gradient Boostes Trees;
Green→ Logistic Regression;
Yellow →k-NN;
Red → Decision Tree

After testing all the above mentioned models, Gradient Boosted Trees and k-NN had the highest accuracies (77.28% and 85.07% respectively). However, I also wanted to check what would be the best model for our case by using an operator **Compare ROCs.** I set up this subprocess by Drag-and-Drop-ing my pre-selected Machine Learning algorithms (*Decision Tree, Random Forest, k-NN, Logistic Regression, Gradient Boosted Trees*). I got a *ROC Graph* as an output (showed on the picture above). In this graph, the True Positive rate is plotted on the y-axis and False Positive rate is plotted on the x-axis. In order to find out where do these curves come from, I decided to use another operator/subprocess (*Cross Validation*) which has Decision Tree inside. The process is shown below.



How did I get from the ROC point characteristic, which was calculated from Confusion Matrix, to a curve? To do that we have to extract the confidence for each prediction made by our classification algorithms. The confidence is similar to the probability that the predicted class will be actually correct. Thus, it tells us how sure the model is for its prediction. In Rapid Miner we can retrieve those confidence values from *Performance operators*.

Summary

After testing five above-explained Machine Learning Algorithms, I have come to conclusion that **Gradient Boosted Trees** and **k-NN** gave the best performance and accuracies for the problem set at hand. After using the operator *Compare ROCs,* I also got the results which suggested that Gradient Boosted Tress and k-NN would be the one of the most appropriate models. However, the final model selection is dependent on preference and purpose of our analysis. Since our problem case is to determine typical features and patterns of behavior that lead to a future inability to make debt repayments, I wanted to focus on the model which successfully predicted potential Default on payment clients (the ones who are not able to repay the debt), while not neglecting the clients and the patterns that lead to successful payout of the debt. Thus, I would go for **k-NN model** as the most appropriate classification algorithm.