Churn Prediction

Workflows and data are available in the EXAMPLES server under *50_Applications/_18_Churn_Prediction*

The Churn Prediction Problem

Typical information that is available about customers concerns demographics, behavioral data, revenue information. At the time of renewing contracts, some customers do and some do not: they churn. It would be extremely useful to know in advance which customers are at risk of churning, as to prevent it - especially in the case of high revenue customers.

This is a prediction problem. Starting with a small training set, where we can see who has churned and who has not in the past, we want to predict which customer will churn (churn = 1) and which customer will not (churn = 0).

attr 1, attr 2, …, attr n => churn (0/1)

This Example

This example uses the same data as the Churn Analysis example. However, here the data set has been split into contract related data (telco plan, fees, etc…) and telco operational data, such as call times in different time zones throughout the day and corresponding paid amounts. The original data set is available as a free download from Ian Pardoe via www.iainpardoe.com/teaching/dsc433/data/Churn.xls.

The contract data contains, among various attributes, a churn field: churn=0 indicates a renewed contract; churn =1 indicates a closed contract.

Pre-processing

After rejoining the two parts of the data, contractual and operational, converting the churn attribute to a string for future machine learning algorithms, and coloring data rows in red (churn=1) or blue (churn=0) for purely esthetical purposes, we now want to train a machine learning model to predict churn as 0 or 1 depending on all other customer attributes.

Training

As usual, we are spoiled for choice when it comes to choosing a machine learning algorithm for training. For use cases, we often deploy a **decision tree** because of its nice tree visualization and highlighting property. However, be aware that you can use any other available machine learning algorithm as long as it produces nominal class-like predictions. For example, **Random Forests**, aka **Ensemble Trees**, are currently the most frequently adopted machine learning algorithms.

Whatever machine learning algorithm you choose, you always need to train it and evaluate it. For this reason, the Partitioning node is required to partition most of the data (80%) for training and the small remaining amount (20%) for evaluation.

To train a decision tree (Decision Tree Learner node), you need to specify the column with the class values to learn (Churn), an information (quality) measure, a pruning strategy (if any), the depth of the tree through the number of records per node (higher number – shorter tree), and the split strategies for nominal and numerical values. At the end of the training phase, the “View” option in the node context menu shows the decision path through the tree to reach leaves with churning and not churning customers. After that the PMML model is saved to a file.

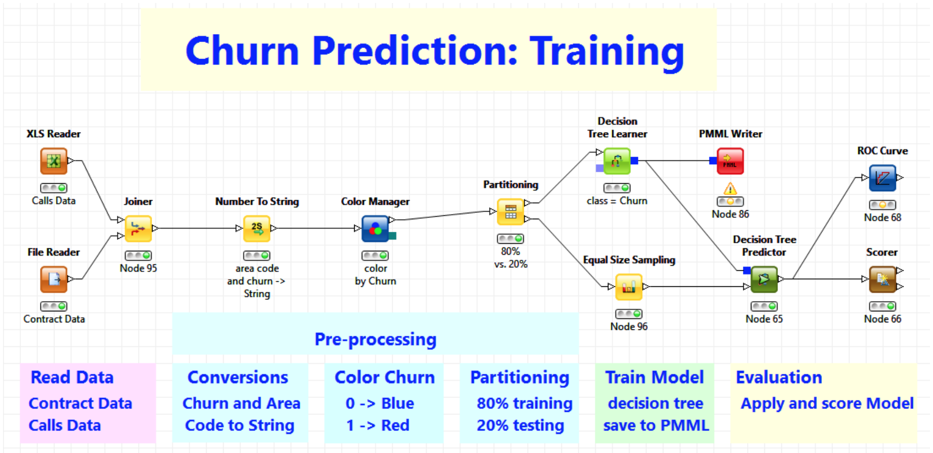
Evaluation

So, we trained a model. But what if the model has not learned anything useful? We need to evaluate it before running it for real on real data. For the evaluation, we use that 20% of data we have kept aside and not used in the training phase, to feed a Decision Tree Predictor node. This node applies the model to all data rows one by one and produces the likelihood that that customer has of churning given his/her contract and operational data (P(Churn=0/1)). Depending on the value of such probability, a predicted class will be assigned to the data row (Prediction (Churn) =0/1).

The number of times that the predicted class coincides with the original churn class is the basis for any measure for the model quality as it is calculated by the Scorer node.

Notice that the customers with churn=0 are, hopefully, many more than the customers with Churn=1. If you want to take this fact into account and give more weight to the error made on the class Churn=1, then you can introduce an Equal Size Sampling node on the test set to under-sample the more numerous class Churn=0.

Notice also that the Scorer node - or any other scoring node - allows you to evaluate and compare different models. A subsequent Sorter node would allow you to select and retain only the best performing model.



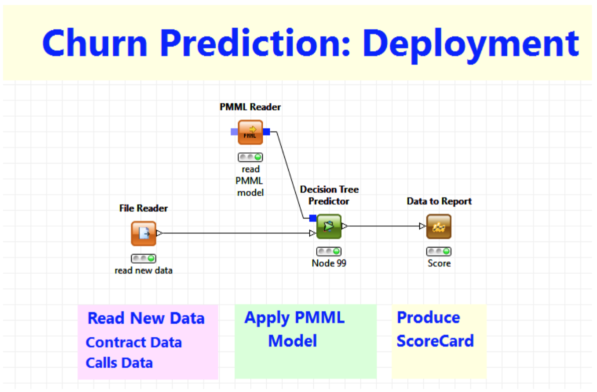
Deployment

When we are satisfied with our model performance, we can move it into production for deployment on real data. Here we need only read the stream of real-life data coming in through a file or database or whatever other data source and the generated model. We then apply a Decision Tree Predictor, a PMML Predictor or a JPMML Classifier to run the model on the real-life input data. The output data will contain a few additional columns with the prediction class and the probability distributions for both classes churn=0 and churn=1, if so specified in the predictor configuration settings.

Please note that a PMML Predictor node or a JPMML Classifier node will make you independent of the selected machine learning model!

If your model needs some data preprocessing, this can also be added to the PMML model using a PMML-compatible data manipulation node in the training workflow.

Also notice that, if the data preparation part has been incorporated into the PMML model, the only really necessary node in this deployment productive workflow, besides the reader nodes, is the predictor node.


Churn Prediction: Deployment
PMML Reader
read PMML model
File Reader
read new data
Decision Tree Predictor
Node 99
Data to Report
Score
Read New Data
Contract Data
Calls Data
Apply PMML Model
Produce ScoreCard