Assignment : Data Mining In Action

Vraj Mehta (13488642)
Spring 2019
University of Technology Sydney

Section 1: The data mining problem (inputs, output)

This dataset represents the activity of a large number of customers who are interested in buying policies from an insurance company. Each **QuoteNumber** corresponds to a **potential customer** and the **QuoteConversion_Flag** indicates whether the customer **purchased a policy** or not,'1' for customer purchased the policy while '0' for not purchased.

There are total of **3** datasets: a **training** dataset (30 columns including *QuoteConversion_Flag* with 78225 rows), a **testing** dataset (29 columns **not** including *QuoteConversion_Flag* with 260776 rows), **predicted** dataset generated from training and testing dataset (including only *Quote_ID* and *QuoteConversion_Flag*) to be submitted on Kaggle. The predicted dataset contains the **predicted** values of QuoteConversion_Flag.

The provided **features** in the training and testing dataset are ***anonymized*** and provide a rich representation of the prospective customer and policy. They include specific *coverage* information, *sales* information, *personal* information, *property* information, and *geographic* information. The task is to predict **QuoteConversion_Flag** for each *QuoteNumber* in the testing dataset.

All the **preprocessing** (*Binning, discretisation, binarisation etc)*, **transformations** (*string to number, category data etc)*, **dividing** the training data set (*partitioning, cross validation etc)*, using the **classifier** (*learning* the data and *prediction* it), **parameter optimisation,** choosing the **best features** and checking the **score** (*ROC curve, scorer* etc) is done on the **training** and **testing** dataset.

To summarise, clearly the **input** is the training data, also the testing data and **output** is the predicted data, having only 2 columns, QuoteID and QuoteConversion_Flag (Predicted).

Section 2: Data preprocessing and Transformation

There are many things to be done in data preprocessing. The main objective of data preprocessing is to identify relevant features for our model that can make better predictions.

I have used both KNIME and Python for data preprocessing.

Data preprocessing:

A) Extract Date Features

Date variables are a special type of categorical variable. While, at first glance, a date gives us nothing more than a specific point on a timeline, when pre-processing properly, they can highly enrich the dataset. Common date formats contain numbers and sometimes text as well to specify months and days. Getting dates into a friendly format and extracting features of dates into new variables can be useful preprocessing steps.

I have extracted the following features from the date:

- Day
- Month
- Day of Week
- Year
- Quarter
- Week of Year

```
In [2]: import pandas as pd
        import numpy as np
        import datetime

In [3]: cols = ['Original_Quote_Date']

In [4]: data = pd.read_csv('Assignment3_TrainingSet.csv',usecols=cols, nrows = 78226)

In [5]: data['Original_Quote_Date']= pd.to_datetime(data['Original_Quote_Date'])

In [6]: data.dtypes
Out[6]: Original_Quote_Date    datetime64[ns]
        dtype: object

In [7]: data['day'] = data['Original_Quote_Date'].dt.day

In [8]: data[['Original_Quote_Date','day']].head()
Out[8]:
```

	Original_Quote_Date	day
0	2013-09-07	7
1	2014-05-29	29
2	2015-05-13	13
3	2014-02-09	9
4	2015-09-04	4

```
In [10]: data['month'] = data['Original_Quote_Date'].dt.month

In [12]: data['dayofweek'] = data['Original_Quote_Date'].dt.dayofweek

In [17]: data['year'] = data['Original_Quote_Date'].dt.year

In [19]: data['quarter'] = data['Original_Quote_Date'].dt.quarter

In [21]: data['weekofyear'] = data['Original_Quote_Date'].dt.weekofyear
```

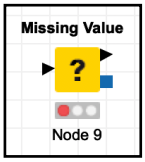
The Python code to extract the date features.

Libraries used: Pandas, Numpy and datetime.

B) Missing Values

Handling missing values is extremely important. If missing values are not handled properly, we may draw incorrect reference about the data. Also many classifiers will throw error, if we do not handle missing values.

Below is the column count with the number of missing values.



```
In [28]: data.isnull().sum()

Out[28]: Quote_ID                     0
         Original_Quote_Date          0
         QuoteConversion_Flag         0
         Field_info1                  0
         Field_info2                  0
         Field_info3                  0
         Field_info4                  0
         Coverage_info1               0
         Coverage_info2               0
         Coverage_info3               0
         Sales_info1                  0
         Sales_info2                  0
         Sales_info3                  0
         Sales_info4                  0
         Sales_info5                  0
         Personal_info1              40
         Personal_info2               0
         Personal_info3               0
         Personal_info4               0
         Personal_info5           37039
         Property_info1              24
         Property_info2               0
         Property_info3               0
         Property_info4               0
         Property_info5               0
         Geographic_info1             0
         Geographic_info2             0
         Geographic_info3             0
         Geographic_info4             0
         Geographic_info5             0
         dtype: int64
```

As we can see:
Personal_Info1, Personal_Info5 and **Property_Info1** have missing values as shown in the figure.

I have used pandas in python to check which attribute is having missing values.

There are two approaches to solve this problem:
1) Remove the row completely with missing values.
2) Impute missing values.

I have taken 2nd approach. Since Property_Info1 and Personal_Info1 both are nominal attributes ('Y' & 'N'), I have fixed it using "**Most frequent value**".
Personal_Info5 is fixed using "**Median**" as it is having numeric values (all whole numbers).

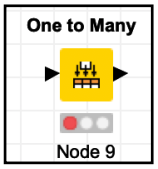
Done using **Missing Value node** in KNIME.

C) One-hot encoding (Binarisation) for categorical

	A	B	C	D	E
1	Geographic_Info5	CA	NJ	IL	TX
2	CA	1	0	0	0
3	NJ	0	1	0	0
4	NJ	0	1	0	0
5	IL	0	0	1	0
6	NJ	0	1	0	0
7	NJ	0	1	0	0
8	TX	0	0	0	1
9	CA	1	0	0	0
10	IL	0	0	1	0
11	NJ	0	1	0	0
12	IL	0	0	1	0
13	CA	1	0	0	0
14	CA	1	0	0	0
15	TX	0	0	0	1
16	CA	1	0	0	0
17	TX	0	0	0	1

The binarisation is the method of converting nominal to binary values(0's and 1's). from the given dataset, the binarisation operation is performing on Geographic_Info5 to binarise, **'CA', 'IL', 'NJ' & 'TX' .**

Used **One to Many node** in KNIME.

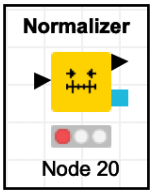


D) Remove Outliers

Removing outliers is important as it can confuse the classifier and may result in incorrect predictions. There are many techniques to remove outliers: Scatter Matrix/Plot, Variance, Box Plot etc.

E) Normalization

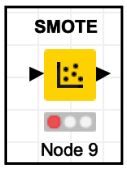
Normalization is important for numeric attributes, especially when you are giving it Support Vector Machine(SVM) or Neural Networks.

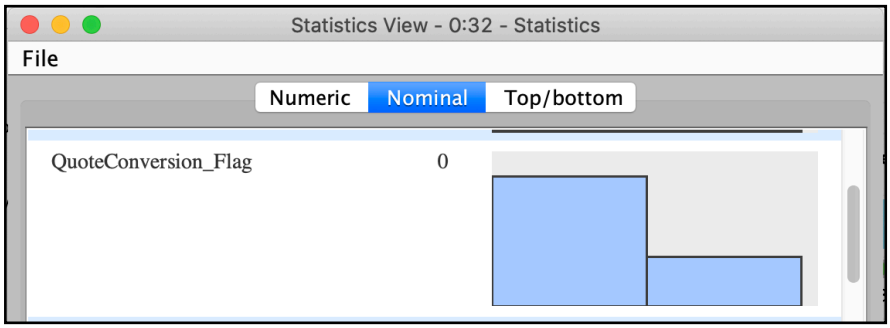


Transformations:

Transformation are essential as the classifier may need the attribute according to a particular type only. It also varies according to the classifier used. Below are the transformations I did for all the models I build.

A) Converting Unbalanced to Balanced Data





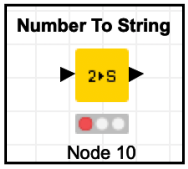
As seen in the figure for QuoteConversion_Flag, '0' has 63315 and '1' has 14910 occurrences.

Clearly, the data is **unbalanced.** I have converted it to balanced data using **SMOTE node.**

Chosen **Oversample Minority classes** (increase the number of '1's to a level it is equal to the number of '0's) and Nearest Neighbour as '5'.

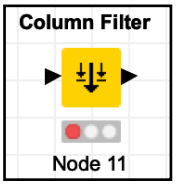
B) Number to String

In the given scenario we need to predict **QuoteConversion_Flag** which is an **Integer**. If we are using Decision Tree, Random Forests etc, the target column should always be a categorical data (String).
Number to String will convert values in QuoteConversion_Flag to categorical data.



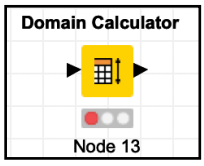
C) Column Filter

Column Filter is used to exclude or include only the particular attributes needed at various stages of building a model. We are filtering out Quote_ID and QuoteConversion_Flag (Predicted) after prediction with the testing dataset and submitting on kaggle.



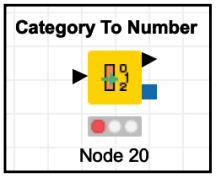
D) Domain Calculator

Domain Calculator is used to transform the Quote_Date attribute from string to date domain which can be identified by the classifier.



E) Category to Number

Category to Number used when one has check linear correlation between different attributes. As linear correlation calculates based on distance.



Section 3: Approach to Solve the Problem

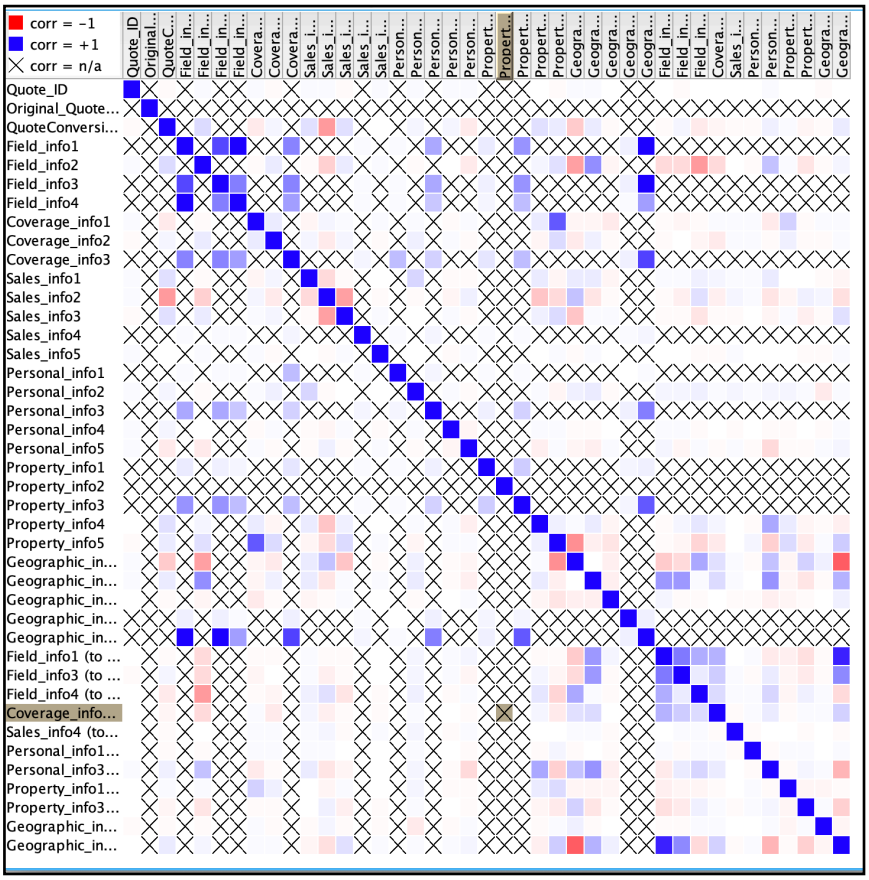
The task is to predict QuoteConversion_Flag for an unknown dataset. Basically, to predict whether a particular customer will purchase the insurance policy or not. The other 29 attributes including Field_Info, Coverage_Info, Sales_Info, Personal_Info, Property_Info and Geographic_Info describe a particular customer. These attributes can also be called **features** for this dataset. All the attributes are of different types (nominal, ordinal, interval, ratio). The classifier that we chose should take into account all the different types of attributes or else we need to convert it into other type.

The first step to solve a data mining problem is data preprocessing and transformation. I have extracted date, fixed missing values, binarisation, normalization etc. Transformation is converting some attribute from number to string, unbalanced to balanced data etc. Data preprocessing and transformation also depends on the classifier chosen.

The next thing is feature engineering. It is a major part of solving a data mining problem. As we know all machine learning algorithms use some input data to create outputs. It is very important that we prepare proper input dataset that are compatible with the machine learning algorithms, hence improves the performance.

The next step is choose different classifiers and compare them, one which receives the highest ROC for both the values ‘0’ and ‘1’ will be best classifier. Once we have got the best classifier, we have to do parameter optimisation for the classifier chosen.

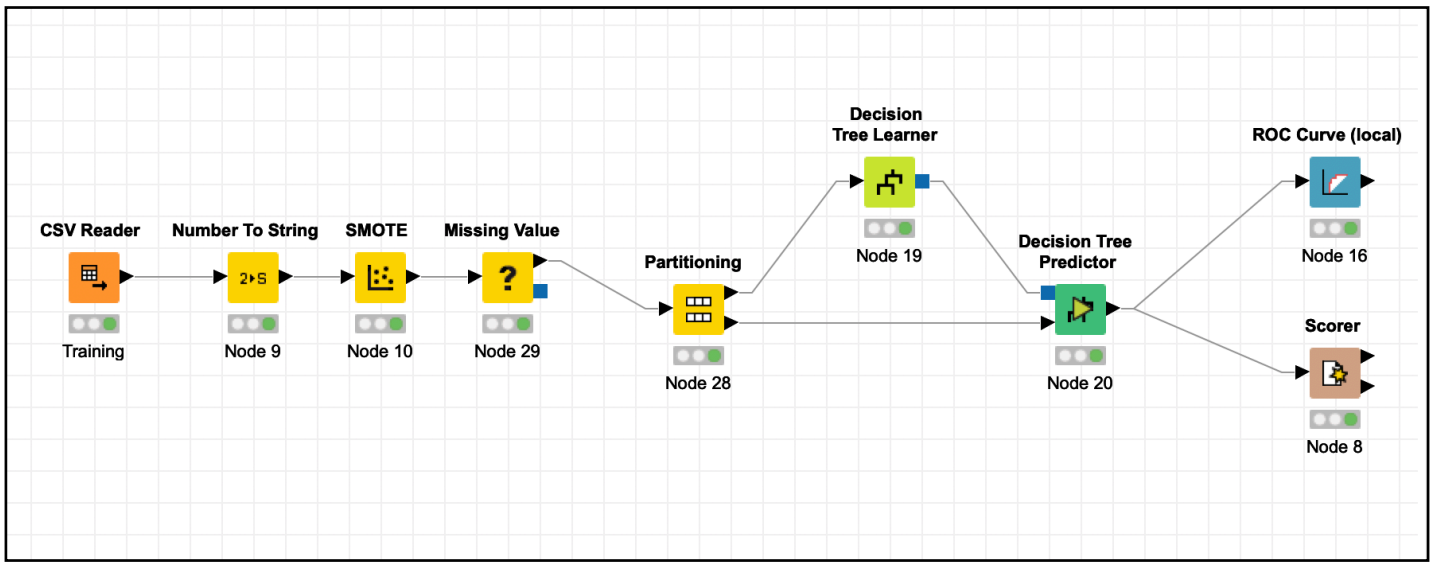
The above two steps are also dependent on each other. Once we have found out the best classifier, we can do some more feature selection or data preprocessing and try to increase the accuracy. The last step will be parameter optimization, tune the parameters that can help achieve the highest accuracy or highest Area Under Curve (AUC) possible.



Linear correlation between attributes.

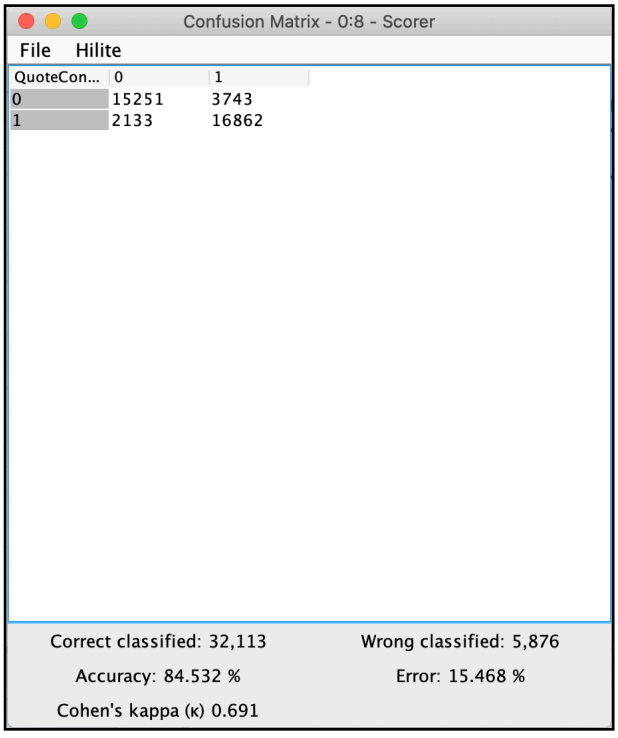
Section 4: Classification Techniques

Decision Tree Classifier



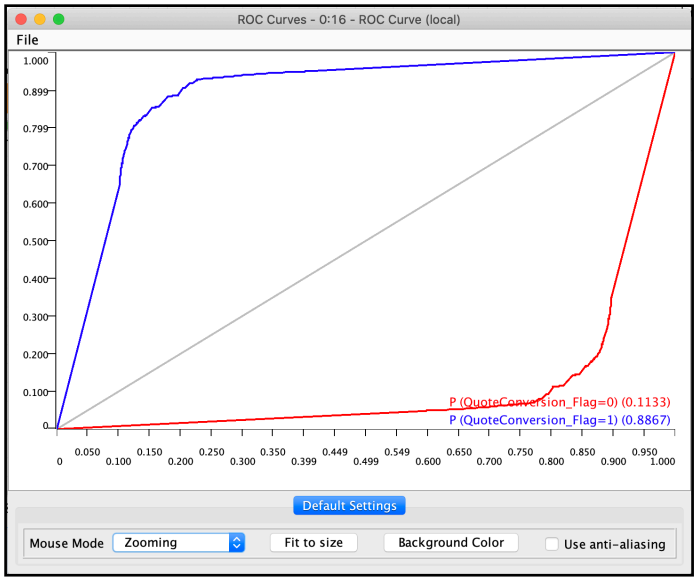
All the features has been taken into account for decision tree learner. Let us examine the steps in detail.

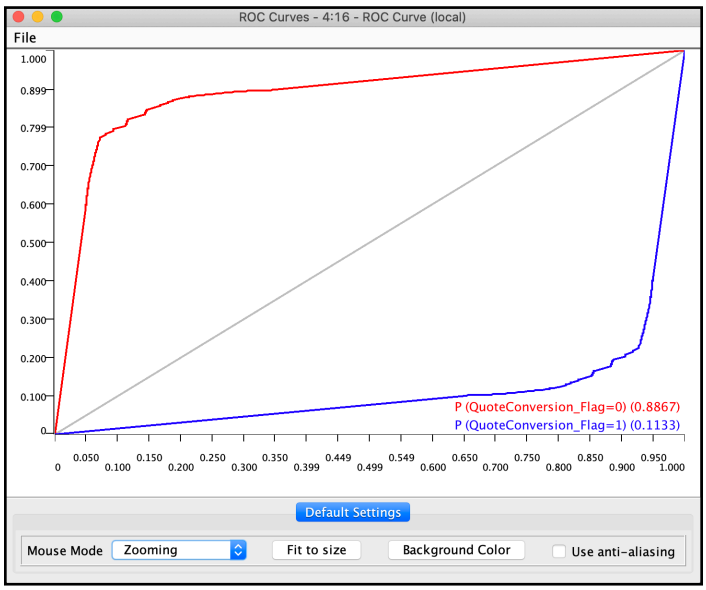
1. The csv reader node reads the Excel file which contains the dataset. The input file has many missing values which will be taken care later.
2. QuoteConversion_Flag is converted from integer to categorical data in Number to String node.
3. The data is highly unbalanced, SMOTE node is used to balance the data. Oversample minority classes with '5' as nearest neighbours.
4. For this classifier, the missing value treatment node is configured to replace nominal missing values with most frequent occurring value and the ordinal/interval missing values to be replaced with Median.
5. The partition node configuration was tweaked by giving various different percentages for partitioning. After trying out several other proportions, the partition parameter value was finalised to 70% training set and 30% test set. (Stratified Sampling)
6. Included all the features where target column is 'QuoteConversion_Flag'.



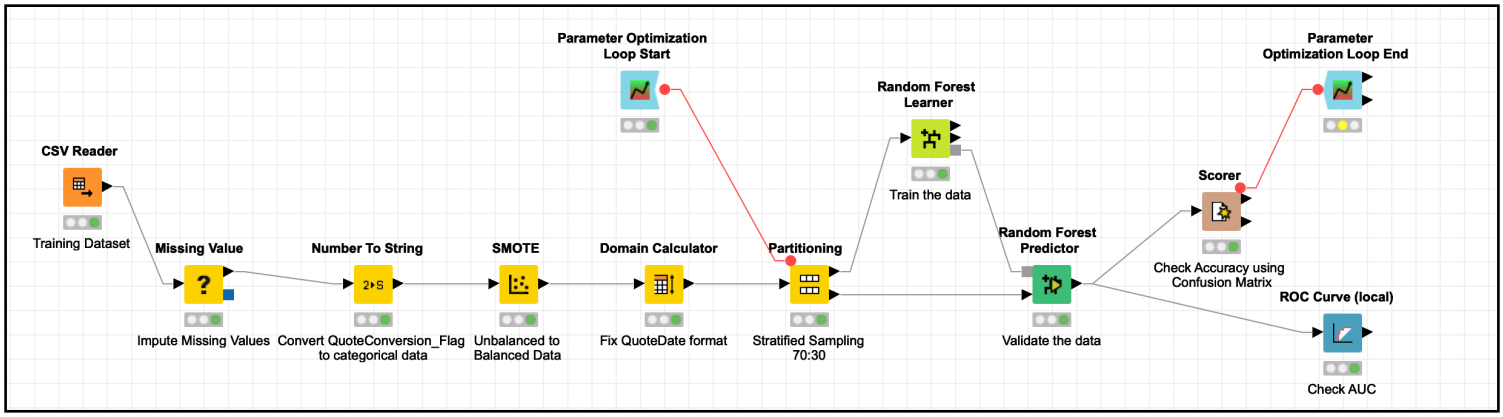
Finally **tuning the parameters** for Decision Tree Learner, the best parameters are:
Quality Measure : Gini Index
Pruning Method : No pruning
Number of threads: 8

Overall the accuracy of this model is **84.53%** and,
Area Under Curve(AUC) using ROC for both QuoteConversionFlag = 0 & 1 is **0.8867.**





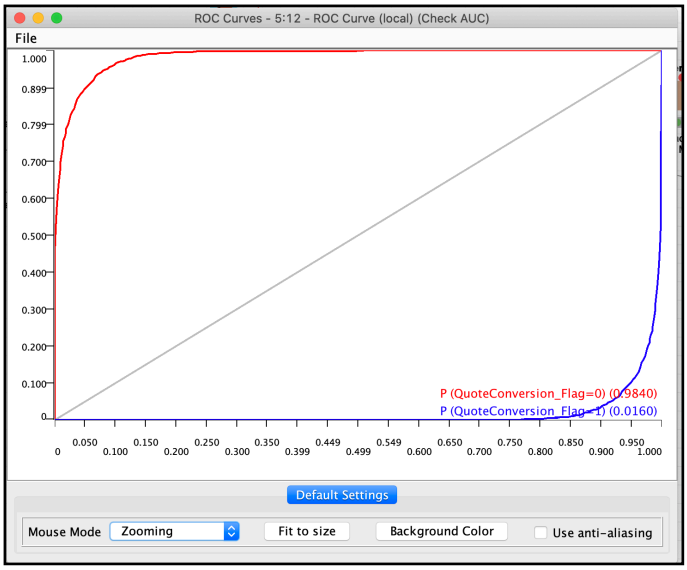
Random Forest Classifier



The initial data preprocessing steps are the same as used for decision tree model. Missing value node to fix the missing values, Number to string node to convert QuoteConversion_Flag from integer to categorical data. SMOTE node to balance the data. Domain Calculator to fix QuoteDate, so that it can be recognised by Random Forest Learner.

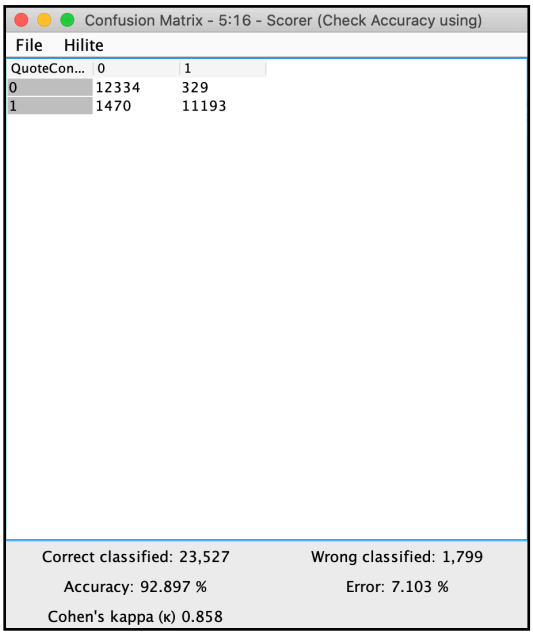
The parameters tuned for Random Forest Learner are:

Split Criterion : Gini Index
Number of Models : 100



The ROC curve and confusion matrix are shown in the figure.

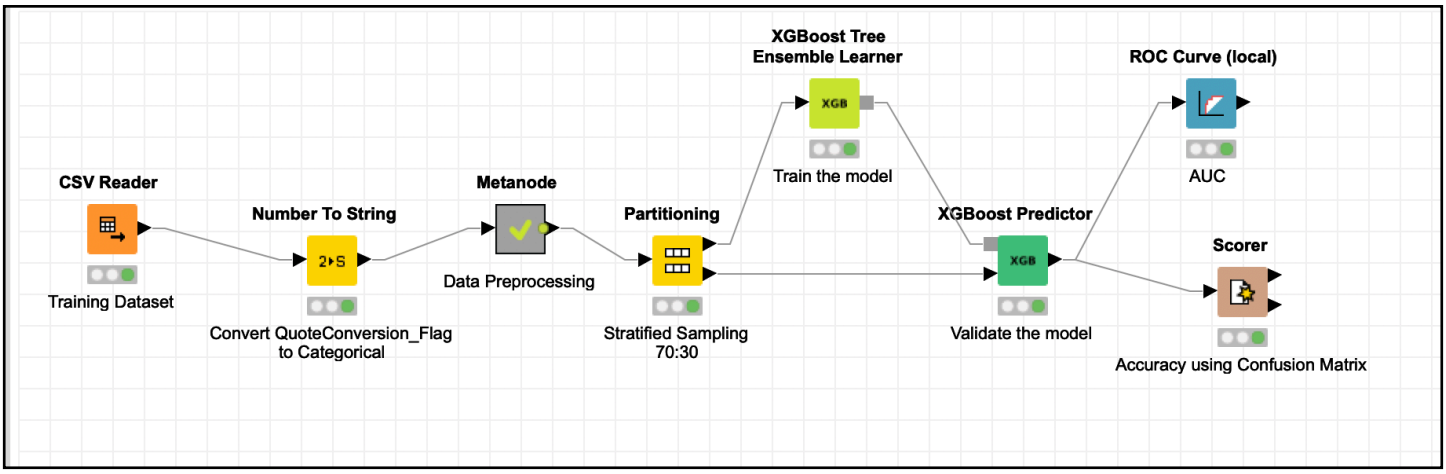
Overall the accuracy of this model is **92.89%** and,
Area Under Curve(AUC) using ROC for both QuoteConversion_Flag=0 & 1 is **0.9840.**



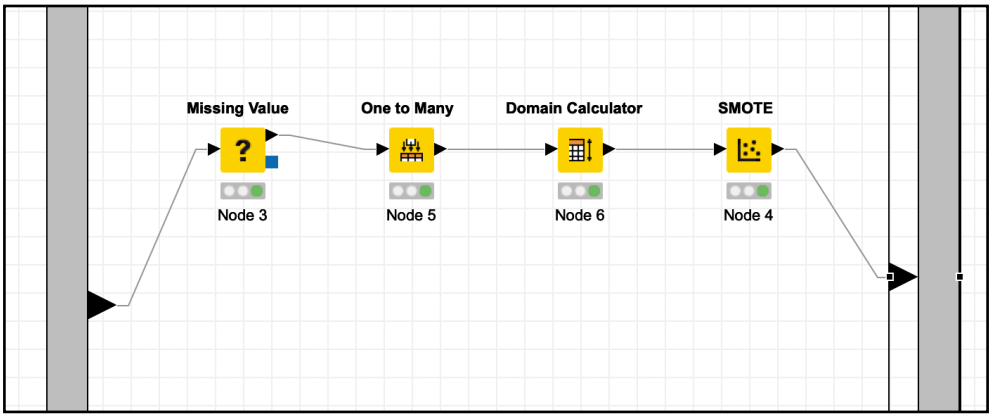
Random Forrest is able to give better results than Decision tree. It is because random forest consists of large number of individual decision trees that work as an ensemble.

Random forest is a combination of many decision trees, obviously it is going to give better result than decision trees.

XGBoost Classifier

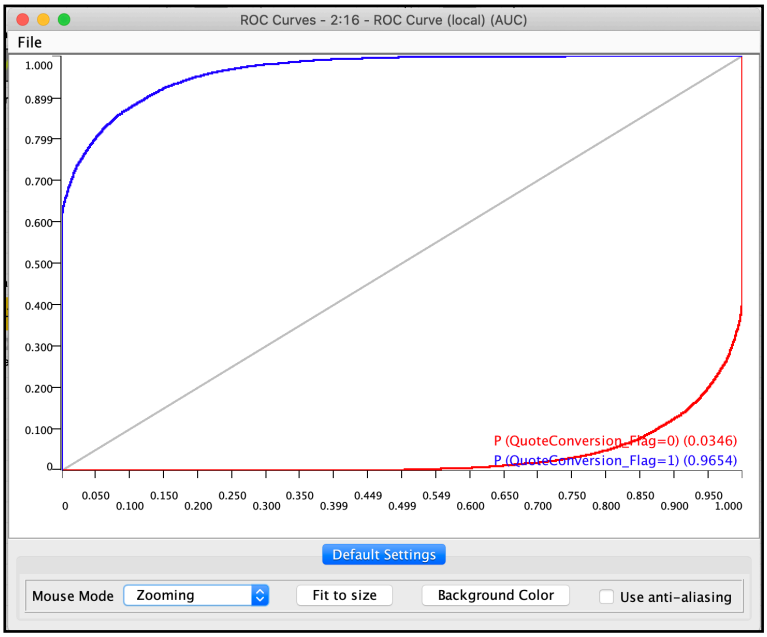


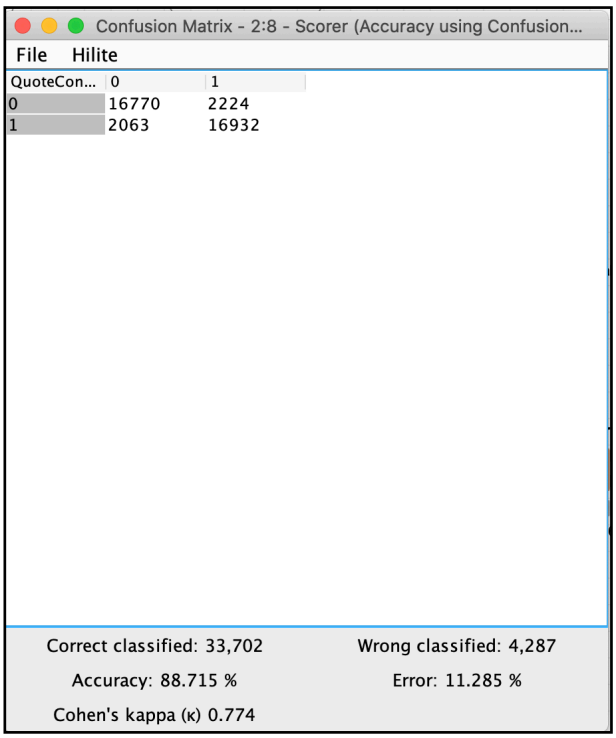
This model is different in terms of pre-processing. The input is training dataset with extracted date features (Day, Month, Day of Week, Year, Quarter, Week of Year). Then there is Number to String Node to convert QuoteConversion_Flag from integer to categorical so that it can identified as class column by the learner. The Data Preprocessing metanode has Nodes to fix missing values. It replaces it with Most frequent for nominal data and Median for interval/ordinal. Then there is One to Many node to Binarise 'Geographic_Info5'. Domain Calculator to fix QuoteDate format and finally SMOTE node to covert the unbalanced data to balanced data.



The parameters tuned for XGBoost Tree Ensemble:

Boosting Rounds: 100
Objective: Binary logistic
Booster: Tree
Max-dept: 6

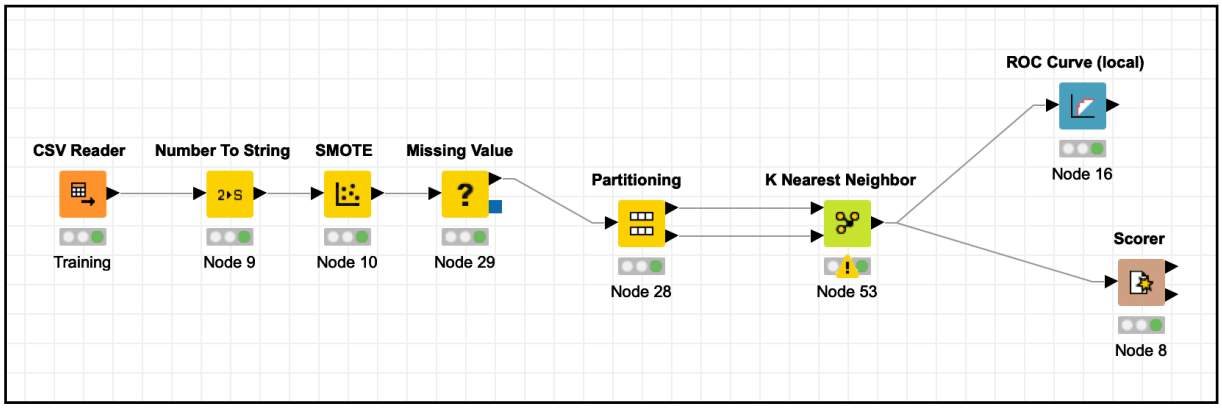




The ROC curve and confusion matrix are shown in the figure.

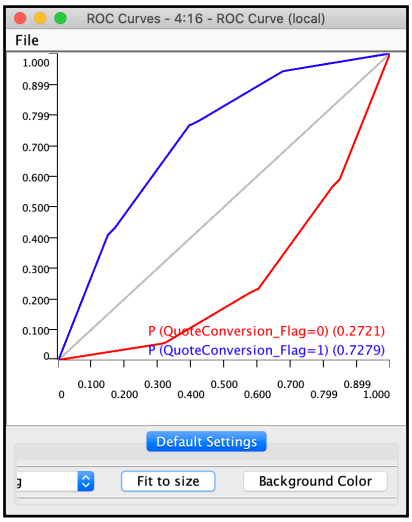
Overall the accuracy of this model is **88.71%** and, Area Under Curve(AUC) using ROC for both QuoteConversion_Flag=0 & 1 is **0.9654.**

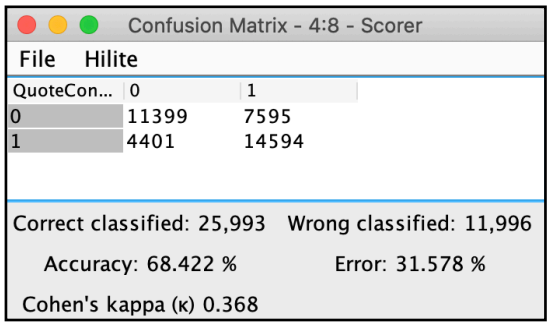
k-Nearest Neighbour



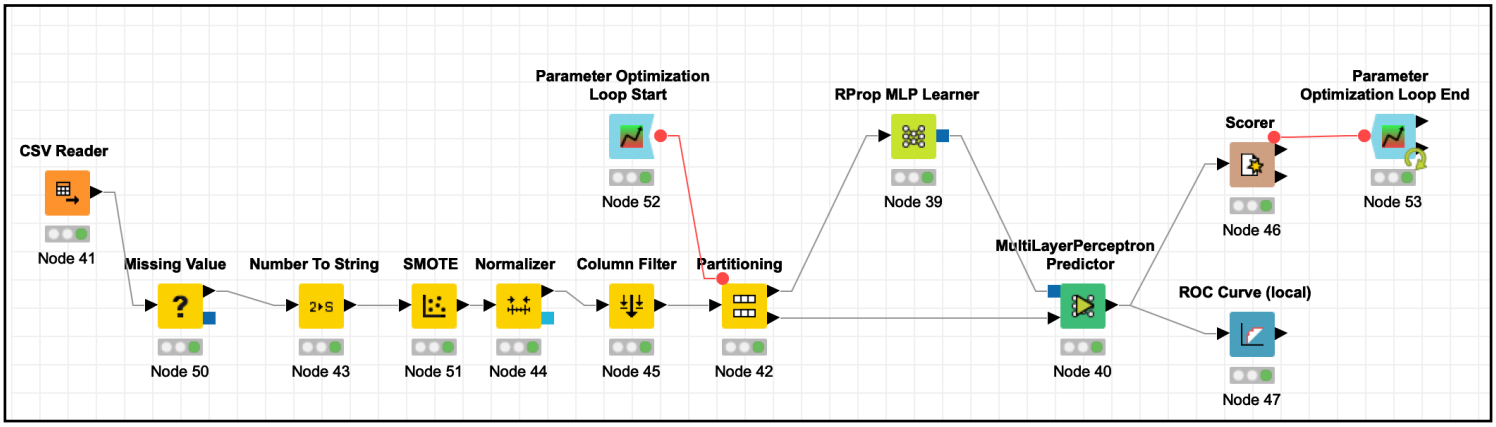
Basic preprocessing like Number to String, SMOTE and Missing vale. Used k-Nearest Neighbour node to predict QuoteConversion_Flag. '3' is the number of nearest neighbour to consider.

Overall the accuracy of this model is **68.42%** and,
Area Under Curve(AUC) using ROC for both QuoteConversion = 0 & 1 is **0.7279.**

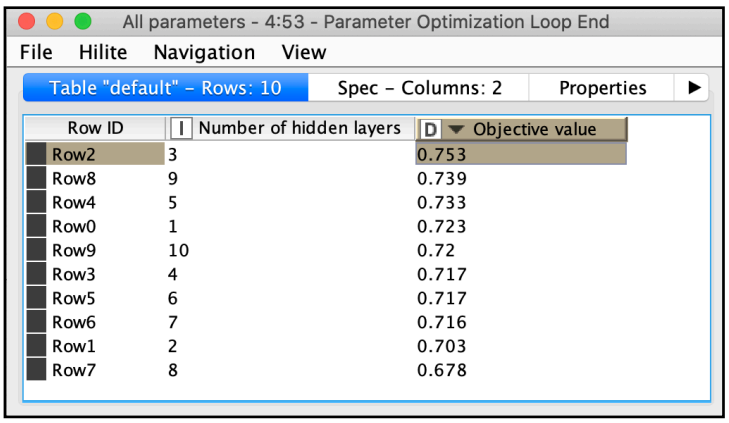




RProp MLP/Neural Network Classifier



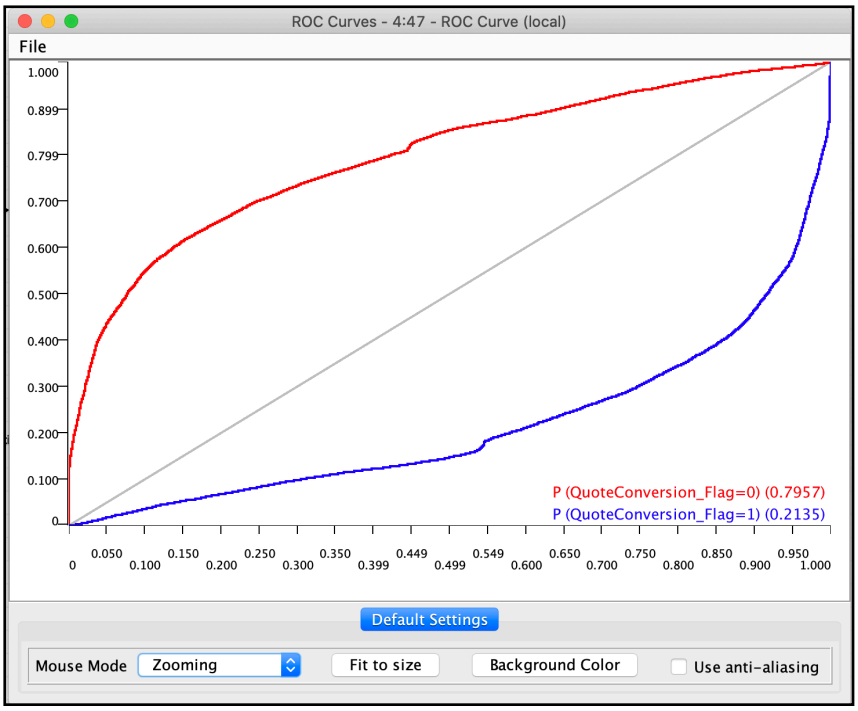
The first 3 nodes are similar to other classifiers. Missing Value node to impute the missing values in the dataset. Number to String node to convert QuoteConversion_Flag to categorical data and SMOTE node to balance the dataset.
Normalizer node is used to normalize the data between 0 and 1, as neural network will take a longer time if we don't normalize the data. Also, Neural Network only take double values as input. The Column Filter node is include only features that are of double type. Stratified Sampling is used to portioned the dataset for learning and validation.

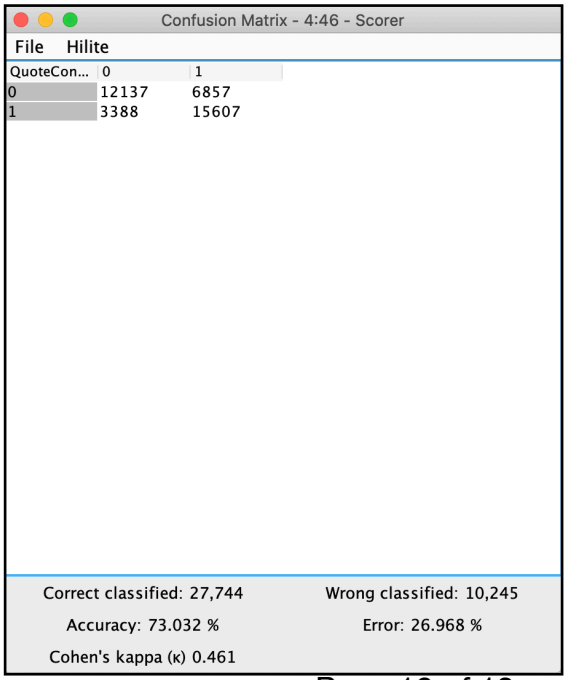


Row ID	Number of hidden layers	Objective value
Row2	3	0.753
Row8	9	0.739
Row4	5	0.733
Row0	1	0.723
Row9	10	0.72
Row3	4	0.717
Row5	6	0.717
Row6	7	0.716
Row1	2	0.703
Row7	8	0.678

The **best parameters** for RProp MLP Learner are:
Max number of iterations: 100
Number of hidden layers: 3
Number of hidden neurons per layer: 20

Using these parameters,
Accuracy: 73.03%
AUC: 0.7957 (Both for '0' & '1')





QuoteCon...	0	1
0	12137	6857
1	3388	15607

Correct classified: 27,744 Wrong classified: 10,245
Accuracy: 73.032 % Error: 26.968 %
Cohen's kappa (κ) 0.461

Section 5: Summary and Best Classifier

Comparing the above mentioned classifiers their accuracy and area under curve(AUC):

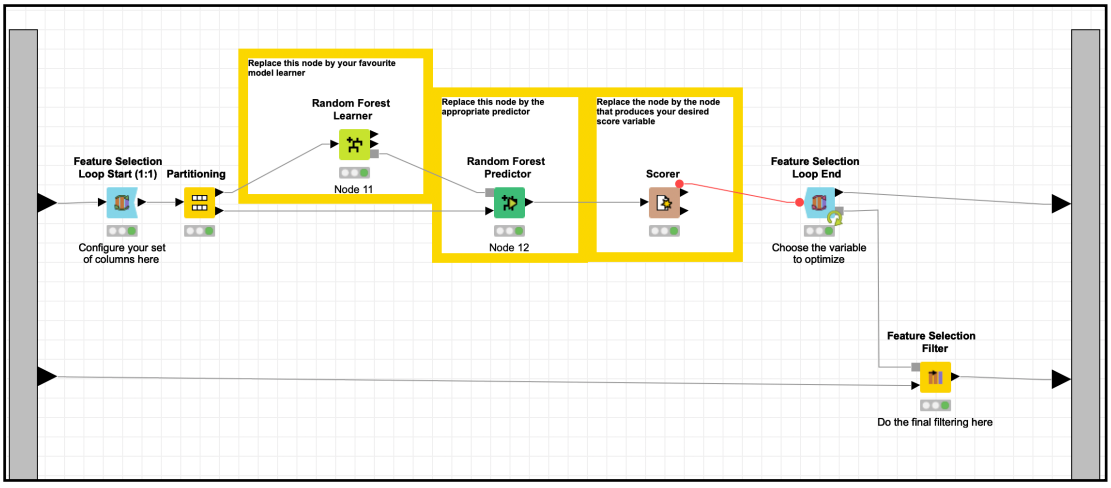
Classifier	Accuracy	Area under Curve (AUC)
Decision Tree	84.53%	0.8867
Random Forest	92.89%	0.9840
k-Nearest Neighbour	68.42%	0.7279
XGBoost	88.71%	0.9654
RProp MLP/Neural Network	73.03%	0.7957

Note: Apart from these classifiers, I also tried. *Support Vector Machine(SVM)* which does not work for this dataset. *Multilayer Perceptron* gives very low accuracy. Then there are *Tree ensemble, Gradient Tree ensemble* which gives very similar results as decision trees & random forests, and also works the same way. That is the reason, I haven't included these classifiers in the report.

We can clearly state that Baseline Random Forest is giving the best result compared to others for both Accuracy and Area under Curve(AUC). Random Forest is a great classifier, to produce a predictive model. Its default parameters already return great results and the system is great at avoiding overfitting. Moreover, it is a pretty good indicator of the importance it assigns to the features.

Given the dataset of such variety, attributes of all types(nominal, ordinal, interval), having missing data, maybe some outliers. The Random forest is able to handle all the things nicely. If we compare it to Multilayer Perceptron/Neural Networks, it can only take numeric values in consideration. Some other models may not handle missing data or outliers satisfactory. Support Vector Machine(SVM) or Neural Networks are also good classifiers but for particularly this dataset (insurance policy), Random forest outshines them. Moreover, in models that uses tree ensembles, Random Forest works the best.

We can further do some feature selection, parameter optimization and cross validation to improve the accuracy of model.

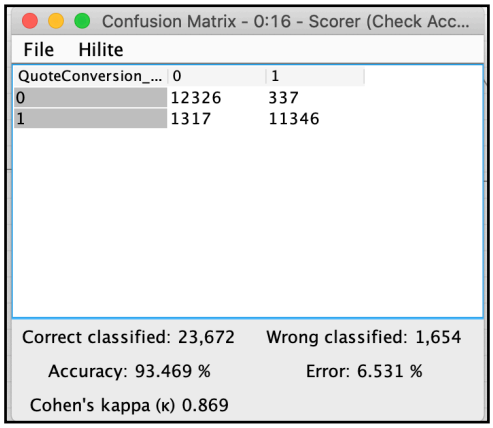


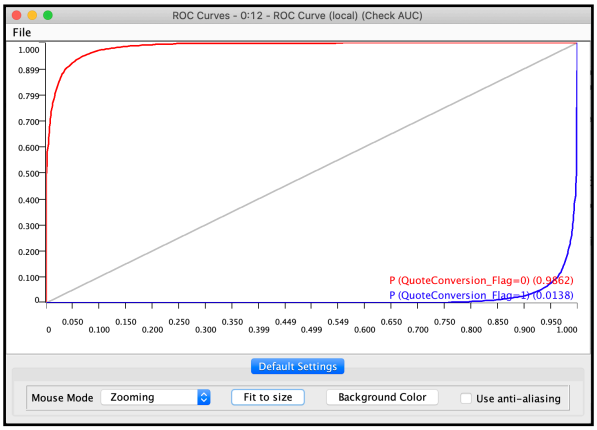
1) **Feature Selection**

Performed feature selection on the training dataset with Random forest as classifier.

Total **15 features** are **important:**

QuoteID, Original_Quote_Date, Field_Info1, Coverage_Info3,	**(Field, Coverage)**
Sales_Info2, Sales_Info4, Sales_Info5,	**(Sales)**
Personal_Info2, Personal_Info3, Personal_Info5,	**(Personal)**
Property_Info3, Property_Info4, Property_Info5,	**(Property)**
Geographic_Info4, Geographic_Info5.	**(Geographic)**

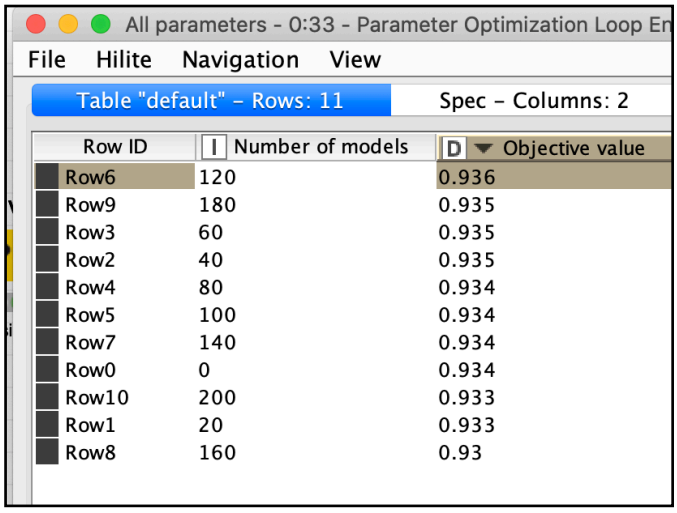




Overall the accuracy of this model improved to **93.47%** and,
Area Under Curve(AUC) using ROC for both QuoteConversion_Flag=0 & 1 is **0.9862.**

2) Paramater Optimization

Optimising parameters is also an important step once we have identified the classifier.
For Random Forest, I have optimised **'Number of Models'** using Parameter Optimization Node.



Row ID	Number of models	Objective value
Row6	120	0.936
Row9	180	0.935
Row3	60	0.935
Row2	40	0.935
Row4	80	0.934
Row5	100	0.934
Row7	140	0.934
Row0	0	0.934
Row10	200	0.933
Row1	20	0.933
Row8	160	0.93

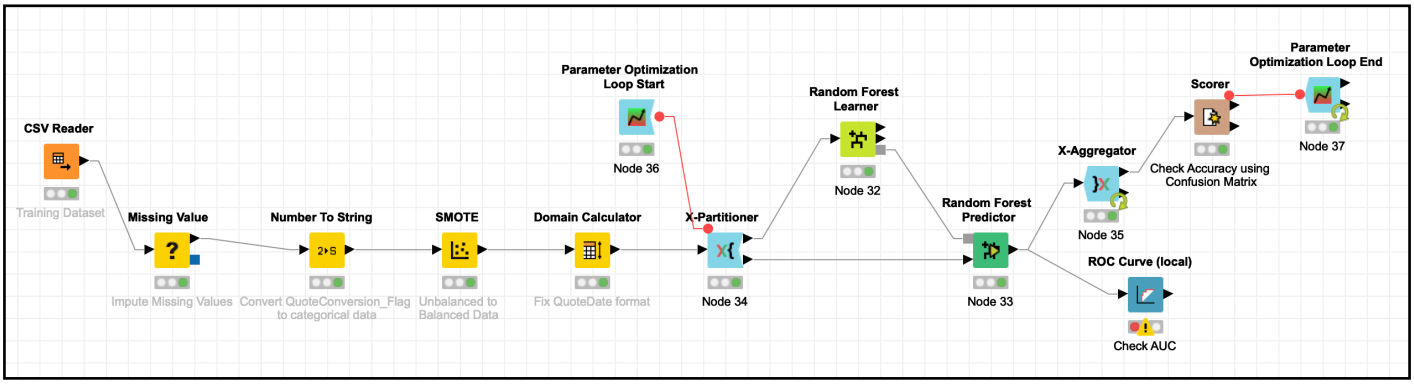
As seen in figure,
Number of Models = 120,
Is the best parameter for Random Forest for this dataset.

Split Criterion: Gini Index
Do not use Static Random Seed

These are the parameters that give best results for Random Forests.

3) k-fold validation (k = 5)

K-fold validations is important to check if the validate the dataset that is been sent to training and testing. It is done using X-Partitioner and X-Aggregator node in KNIME.



The results remain the same after 5 validations. Thus, the accuracy which is predicted is correct.

Section 6: Reflection

This report talks about understanding the data mining problem. What are inputs and outputs.
I have reflected on the following topics:

1) Data Preprocessing - extremely important to extract features, so that model can learn from it and help in better prediction
2) Transformation - sometimes the data has missing values, or it may be unbalanced. How to deal with those kind of issues.
3) Feature Engineering is more important than choosing a classifier. It is the features that help in solving the problem (The classifier can learn better).
4) Choosing a classifier highly depends on what is composed in the dataset. If it has numeric (interval) values, maybe Neural Networks are better. But if it is a mixture of all types of attributes Tree ensemble classifiers handle it better.
5) Once we have got the best classifier and the features, it is important that we optimise the parameters for that classifier as much as possible. (Parameter Optimization)

To Conclude:

Random Forest is the best classifier.
QuoteID, Original_Quote_Date, Field_Info1, Coverage_Info3, Sales_Info2, Sales_Info4, Sales_Info5, Personal_Info2, Personal_Info3, Personal_Info5, Property_Info3, Property_Info4, Property_Info5, Geographic_Info4, Geographic_Info5
are the best features.
Number of Models = 120 and **Split Criterion**: Gini Index, are the best parameters.

Using this model,
Overall the accuracy of this model is **93.47%** and,
Area Under Curve(AUC) using ROC for both QuoteConversion_Flag=0 & 1 is **0.9862.**
These results are only on training dataset. (Partitioned to training and validation)

Score on Kaggle is: **0.80090**
Note: While submitting on kaggle one has to connect Test Data directly to the Predictor Node. Test data doesn't contain QuoteConversion_Flag. Output of Predictor Node, is the one uploaded to Kaggle.