31250 Introduction to Data Analytics

AUTUMN 2021

Assignment 3: Data Mining in Action

Ahmed Khursheed - 13477878

Table of Contents

List of Figures

1 Introduction

This report is in continuation of the previous report: Data Exploration and Preparation in which the insurance company's dataset was explored, understanding of attributes was carried out, relations amongst attributes were established and the dataset was prepared to be worked upon.

In continuation to that, this report will perform some data analysis, mining on the given training dataset and develop multiple classifiers to predict the whether the customer will buy insurance from the company or not by predicting the Quote_Flag attribute. Multiple classifiers will be explored with different parameter settings to achieve the best result on the unknown dataset.

This task will follow the CRISM-DM which stands for Cross-Industry Standard Process for Data Mining in which the task is divided into different stages which are in a loop until the best results are achieved in predicting if the customer would buy the insurance or not. The diagram below explains the different stages and an overall view of the methodology. For the task at hand, KNIME is used.

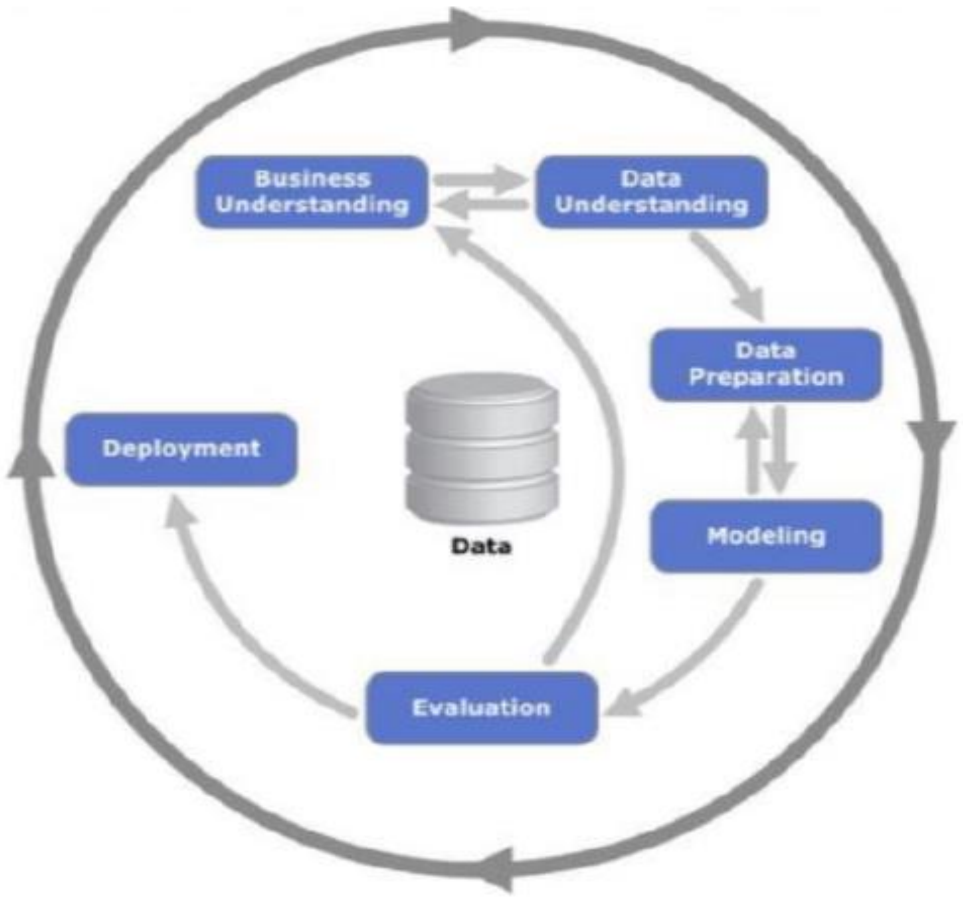
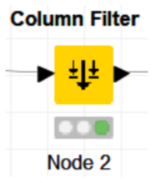
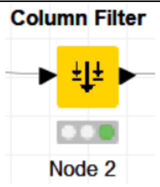
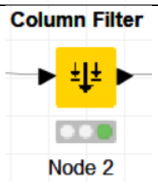
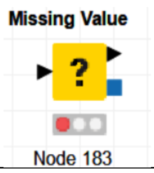
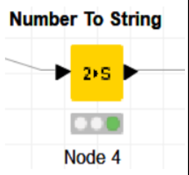
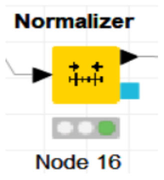
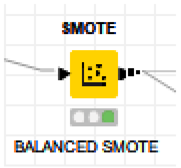
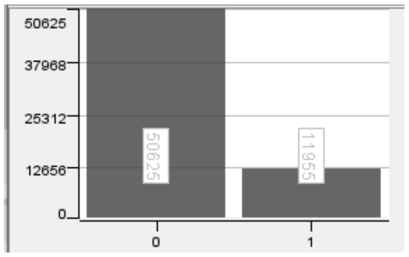
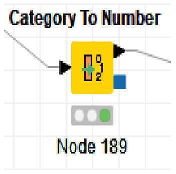


Figure 1 CRISP-DM methodology

2 Data Pre-Processing and Preparation

In the previous report we explored each attribute and derived a basic understanding and established some meaningful relations amongst them. For this task, we have a similar dataset on which analysis needs to be performed. Therefore, some insight into each attribute which we can carry forward can be found here in the previous report. To achieve meaningful results from the analysis and supply the classification models with valid data as per their requirements, data pre-processing is required. The following pre-processing was carried out on the training dataset:

Attribute(s)	Transformation	Reason/Purpose
Quote_ID & Qoute_Date	Filtered out 	Quote_ID and Quote_Date do not have any role in the determination of the Quote_Flag attribute hence it is not required and filtered out using the Column Filter Node.
Personal_info1 & Property_info2	Filtered out 	Most records for these attributes are N and 0 respectively, which do not provide any meaningful input, hence they are filtered out using the Column Filter Node.
Personal_info5	Filtered out 	The attribute had 29788 missing values which is a significant amount and its unmeaningful. Therefore, this was filtered out using the Column Filter Node.
Property_info1 & Geographic_info4	Missing entries filtered out 	In some rows, these two columns were missing entries, the specific rows were filtered out using the Missing Value Node.
Quote_Flag	Transformed to String 	In order for classifiers to use Quote_Flag to train and use the attribute as their prediction attribute, the following attribute needed to be converted to a string. It was done using a Number to String Node.
Field_info2, Coverage_info1, Coverage_info2, Sales_info2, Personal_info2, Property_info5, Geographic_info1 and Geographic_info2	Normalised (Min-Max 0-1) 	The following attributes were normalised in order to achieve better results using the Normalizer Node. Min-max normalization was done between 0-1 for this task.

All	Multiple rows of data generated in order to balance data.  SMOTE stands for: Synthetic Minority Over-Sampling Technique.	The data was balanced using the SMOTE node in order to achieve better results. As seen below, the dataset was unbalanced and might have produced biased results. 
All essential attributes	Transformed to Double 	In order to work with Nodes like Support Vector Machine (SVM) and Neural Network (NN) classifiers, attributes needed to be converted to double/integer as the nodes require numeric values and do not work well with string values. This was done using a Category to Number Node.

After performing the following essential pre-processing methods described above, we had a filtered and processed dataset on which classification models could be applied. Furthermore, the settings of each node described in the pre-processing stage can be found in Appendix – A.

3 Data Mining Method

The goal of the task is to achieve a highly balanced prediction accuracy in terms of **f**-measure (the individual accuracy of predicting 1 and 0 of the classification model) for both our prediction outcomes, 0 and 1 and for that, data pre-processing must be done. It is essential that we filter out columns, rows and entries that are outliers, incorrect, missing values and maintain consistency so that analysis can be performed on the dataset. For the problem at hand however, which is to predict whether the quote was purchased by the customer or not, only the best classifier must be optimized.

In order to begin training classification models, we must first perform the necessary pre-processing steps. To train and test the data, the data is split using a Partition Node in a 70% to 30% ratio between training and testing data, respectively (except for SVM). To train the classification model, 70% of the dataset is used and in order to verify and cross validate the prediction ability of the model 30% of test set. The model is trained and then applied to the unknown dataset. Many accuracy metrics are generated in order to check and compare the accuracy like ROC-Curve and **f**-measure which can give an accurate measure of how good the classification model is.

3.1 Support Vector Machine (SVM)

SVM is a classifier which works best on numeric values, it is not compatible with string values. Therefore, a String to Number Node was used as discussed previously. Initially to train SVM, an attempt was made to get the learner running on the raw dataset, but due to the high computing time (>24 hours) of the SVM Learner as seen in fig. 2 it had to be aborted.

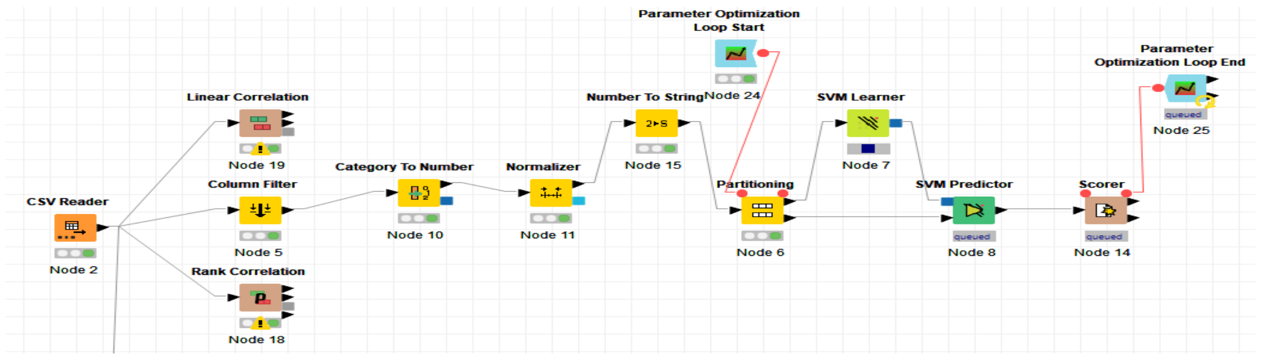


Figure 2 SVM Learner

To achieve some results from the SVM Learner, the advice from head analyst Mr. Mukesh's was followed which was to use only 5% of data and split that into 70-30 ratio and train the model. As seen in fig. 6, parameter optimization nodes were used to obtain best results from our computation for the **σ** value. The following settings were used:

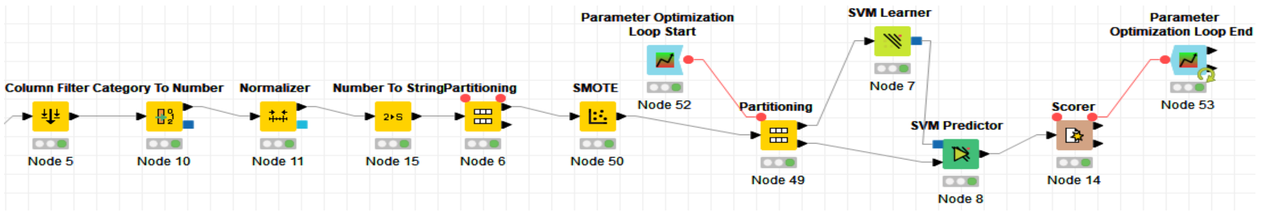
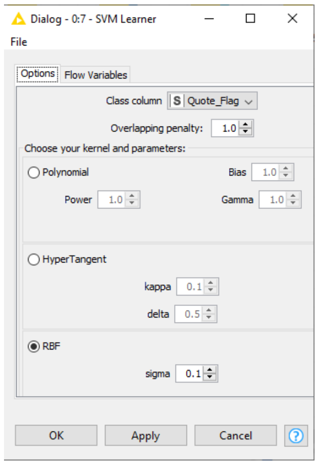


Figure 6 KNIME workflow showing SVM Learner with SMOTE and double partition used..



Dialog - 0:7 - SVM Learner

File

Options | Flow Variables

Class column: S Quote_Flag

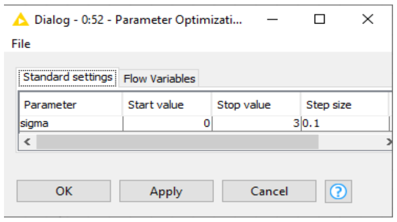
Overlapping penalty: 1.0

Choose your kernel and parameters:

- Polynomial — Bias 1.0, Power 1.0, Gamma 1.0
- HyperTangent — kappa 0.1, delta 0.5
- [✓ RBF] — sigma 0.1

OK | Apply | Cancel

Figure 5 SVM Learner Node Settings



Dialog - 0:52 - Parameter Optimizati...

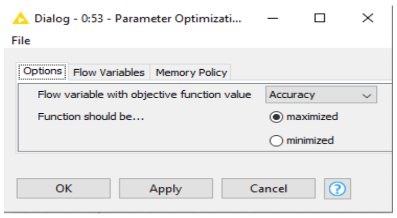
File

Standard settings | Flow Variables

Parameter	Start value	Stop value	Step size
sigma	0	3	0.1

OK | Apply | Cancel

Figure 7 Settings for Parameter Optimization Start and End



Dialog - 0:53 - Parameter Optimizati...

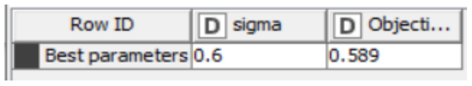
File

Options | Flow Variables | Memory Policy

Flow variable with objective function value: Accuracy

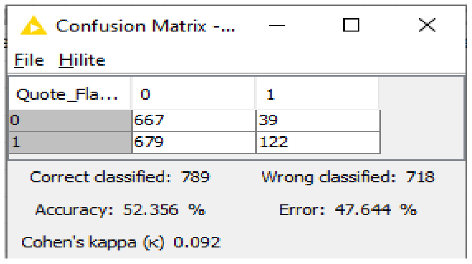
Function should be... [✓ maximized] minimized

OK | Apply | Cancel



Row ID	D sigma	D Objecti...
Best parameters	0.6	0.589

Figure 4 Best parameters obtained from parameter optimization loop end: σ=0.6 accuracy=58.9%



Confusion Matrix -...

File Hilite

Quote_Fla...	0	1
0	667	39
1	679	122

Correct classified: 789 Wrong classified: 718

Accuracy: 52.356 % Error: 47.644 %

Cohen's kappa (κ) 0.092

Figure 3 Confusion Matrix obtained from 5% data SVM

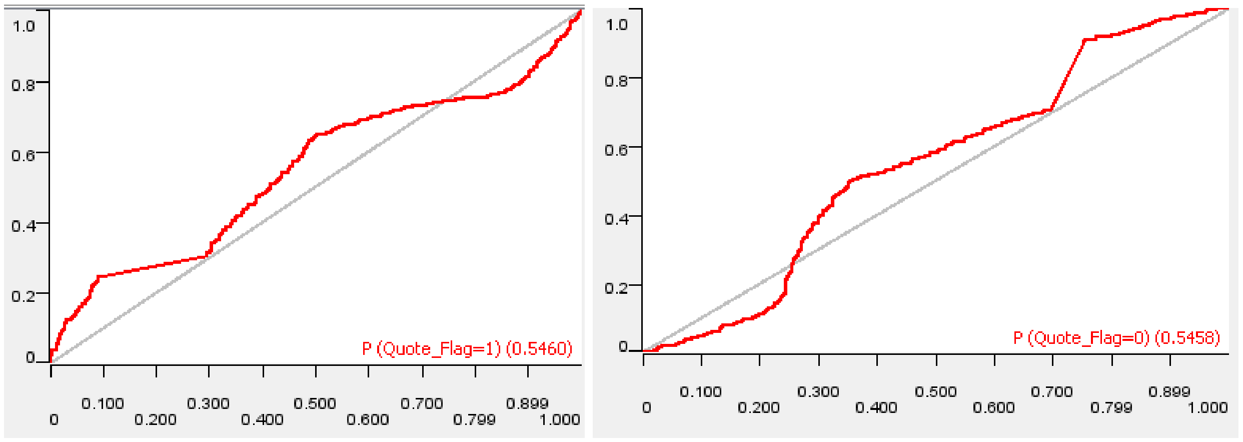


Figure 8 ROC Curves for SVM Learner

An accuracy of 58.9% was obtained from the parameter loop optimization nodes according to their settings as seen in fig. 7. However, when the SVM was run again with the same 0.6 sigma value which was supposed to be the best, a lower accuracy was achieved. This is due to the model being only trained of 70% of 5% of the data which is a small fraction and does not capture the different occurrences of each attribute in the dataset as seen in the ROC Curves in fig. 8 as well. Due to the low scores, this is not the optimal classifier.

3.2 Decision Tree Learner (DTL)

Here is the workflow for the Decision Tree Learner Classification used:

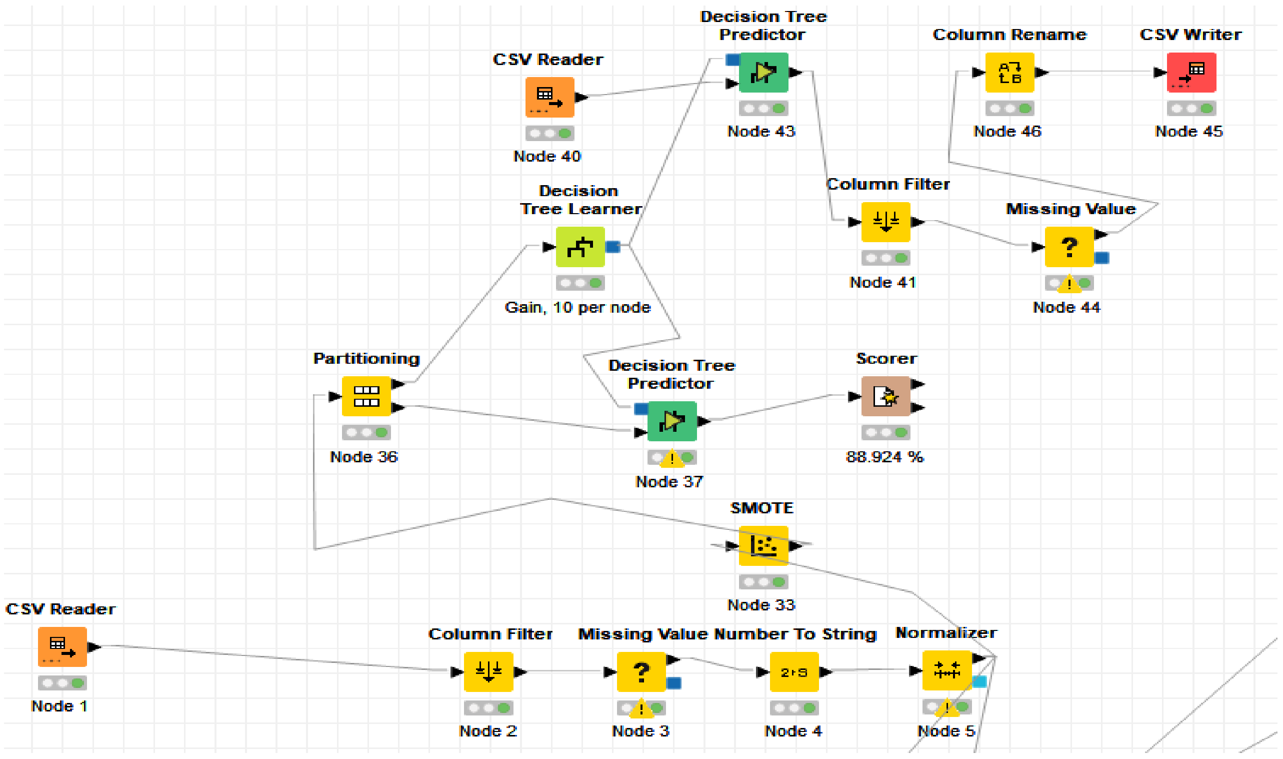
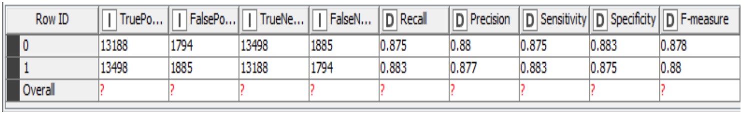


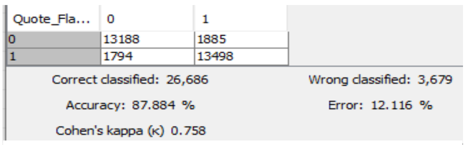
Figure 9 Decision tree learner with SMOTE

A singular decision tree is needed to get a grasp of all other tree classifiers and learners. Decision Tree Predictor is one of the most accurate and strong classifiers as it can handle numerical and string values as seen below in the results.



Row ID	I TruePo...	I FalsePo...	I TrueNe...	I FalseN...	D Recall	D Precision	D Sensitivity	D Specificity	D F-measure
0	13188	1794	13498	1885	0.875	0.88	0.875	0.883	0.878
1	13498	1885	13188	1794	0.883	0.877	0.883	0.875	0.88
Overall	?	?	?	?	?	?	?	?	?

Figure 11 F-measure for Decision tree learner model with highest score with SMOTE



Quote_Fla...	0	1
0	13188	1885
1	1794	13498

Correct classified: 26,686 Wrong classified: 3,679

Accuracy: 87.884 % Error: 12.116 %

Cohen's kappa (κ) 0.758

Figure 10 Confusion Matrix for Decision Tree Learner

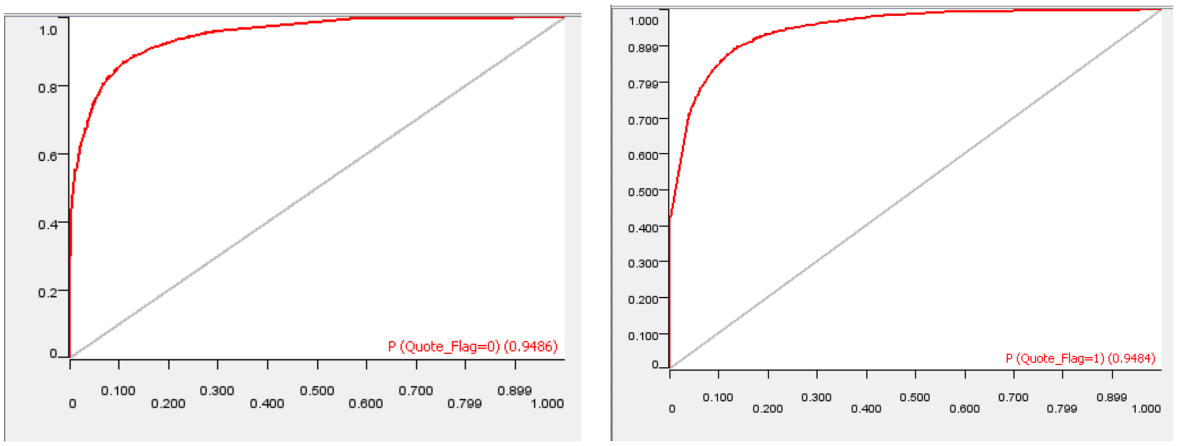
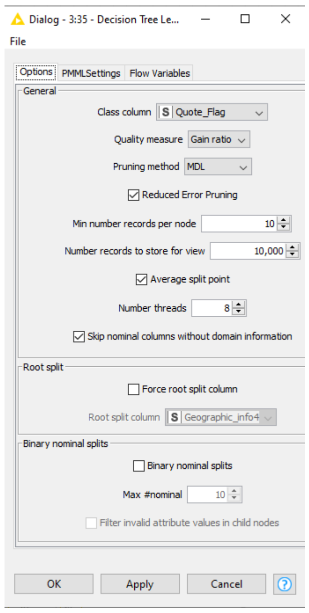


Figure 12 ROC Curves for Decision Tree Learner

Decision tree learner was used with the different settings and the following settings as seen in fig. 13 produced the best results. The quality measure was set to Gain ratio and MDL pruning was also performed to achieve better results. As seen in figure 11, the **f-**measure being approximately 8.7 for both the outcomes 0 and 1 is quite a good result. Furthermore, the ROC-Curves as seen in figure 12 are really showing promising results for the prediction of Quote_Flag. Also, the confusion matrix has a 87.884% accuracy. This classifier can be a potential ideal classifier until a better one is found which can further be worked on.



General

Class column: S Quote_Flag

Quality measure: Gain ratio

Pruning method: MDL

[✓] Reduced Error Pruning

Min number records per node: 10

Number records to store for view: 10,000

[✓] Average split point

Number threads: 8

[✓] Skip nominal columns without domain information

Root split

[ ] Force root split column

Root split column: S Geographic_info4

Binary nominal splits

[ ] Binary nominal splits

Max #nominal: 10

[ ] Filter invalid attribute values in child nodes

Figure 13 Settings for Decision Tree Learner Node

3.3 K-Nearest Neighbour (KNN)

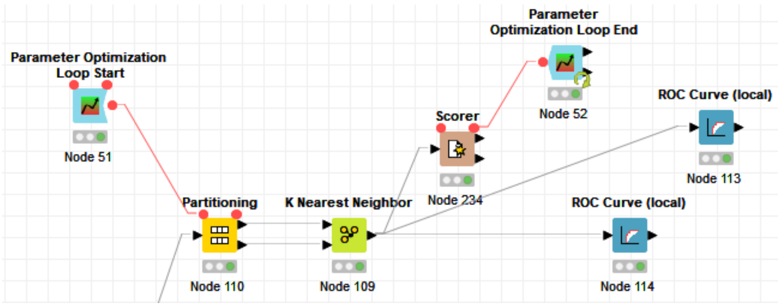
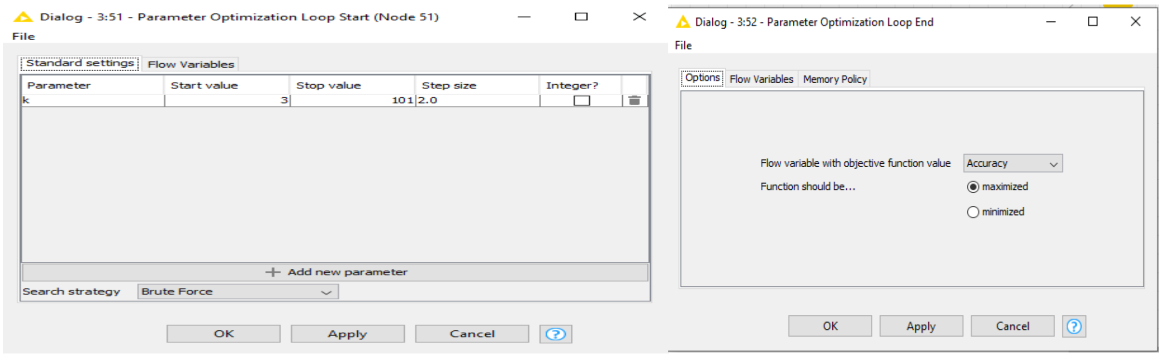


Figure 14 KNIME Workflow for KNN Classifier

KNN is one of the most used classifier methods. A parameter optimization loop was used in order to achieve a best value for k as seen in fig. 15. The best k value achieved from this loop was 86 as seen in fig. 17. Settings for KNN can be found in [Appendix – B](#).


Dialog - 3:51 - Parameter Optimization Loop Start (Node 51)

Standard settings | Flow Variables

Parameter	Start value	Stop value	Step size	Integer?
k	3	101	2.0	☐

Search strategy: Brute Force

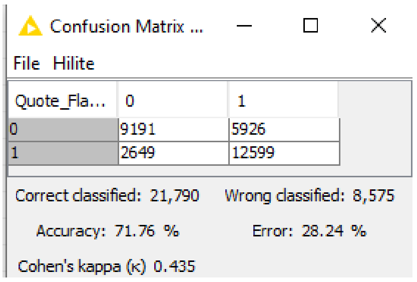
Dialog - 3:52 - Parameter Optimization Loop End

Options | Flow Variables | Memory Policy

Flow variable with objective function value: Accuracy

Function should be... [✓ maximized] minimized

Figure 15 Parameter optimization loop settings for KNN


Confusion Matrix ...

Quote_Fla...	0	1
0	9191	5926
1	2649	12599

Correct classified: 21,790 Wrong classified: 8,575

Accuracy: 71.76 % Error: 28.24 %

Cohen's kappa (κ) 0.435

Figure 16 Confusion matrix for KNN

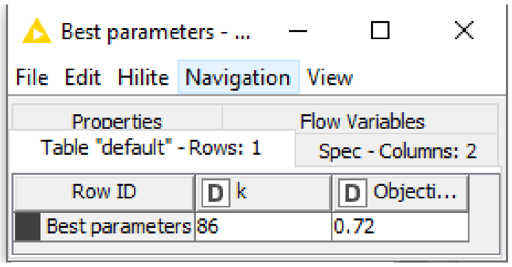
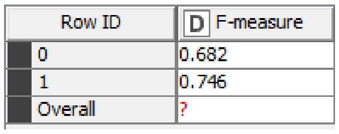

Best parameters - ...

Table "default" - Rows: 1 Spec - Columns: 2

Row ID	k	Objecti...
Best parameters	86	0.72

Figure 17 Best Parameter for k



Row ID	F-measure
0	0.682
1	0.746
Overall	?

Figure 18 **f**-measure for KNN

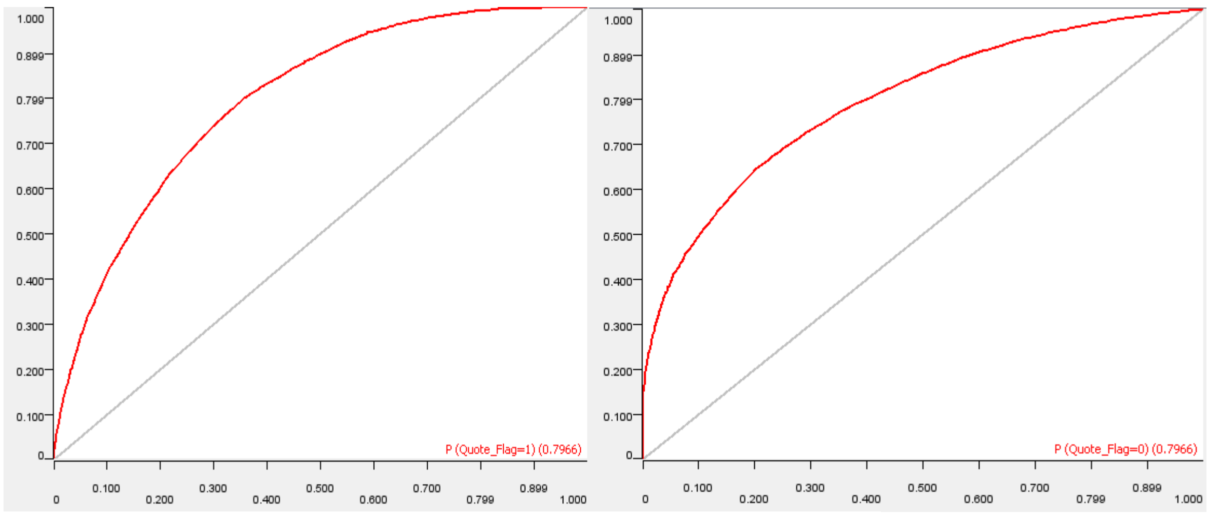


Figure 19 ROC-Curves for KNN

KNN however did not produce the best results, as seen above in fig. 16, the accuracy was only 71.76% although many k values and different settings were tested out. Also, the **f**-measure as seen in fig. 18 for KNN was not high like Decision Tree Classifier discussed above. The areas under the ROC-Curve in fig. 19 reflect the same. All facts considered; this classifier will be discarded.

3.4 Random Forest Learner

Random forest learner produced one of the best results out of all the classifiers. It is better than the rest as it can handle both numeric and categorical values. Also, another factor is that it is one of the best classifiers when it comes to considering high number of variables which our given data set had. The workflow can be seen below.

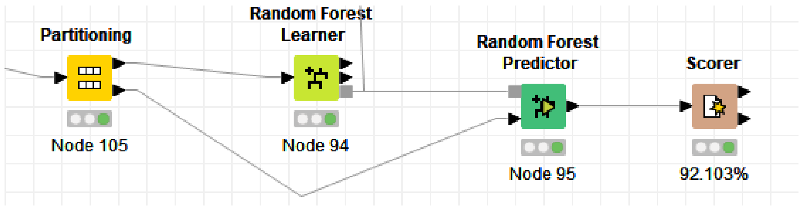
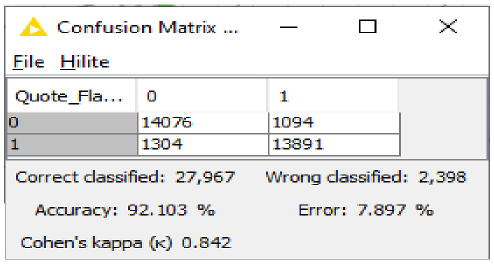


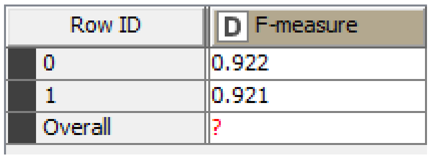
Figure 20 KNIME Workflow for Random Forest Learner



Quote_Fla...	0	1
0	14076	1094
1	1304	13891

Correct classified: 27,967 Wrong classified: 2,398
Accuracy: 92.103 % Error: 7.897 %
Cohen's kappa (κ) 0.842

Figure 21 Confusion Matrix for Random Forest Learner



Row ID	D F-measure
0	0.922
1	0.921
Overall	?

Figure 22 f-score for Random Forest Learner

The settings for the Random Forest Learner Node can be found in Appendix – C.

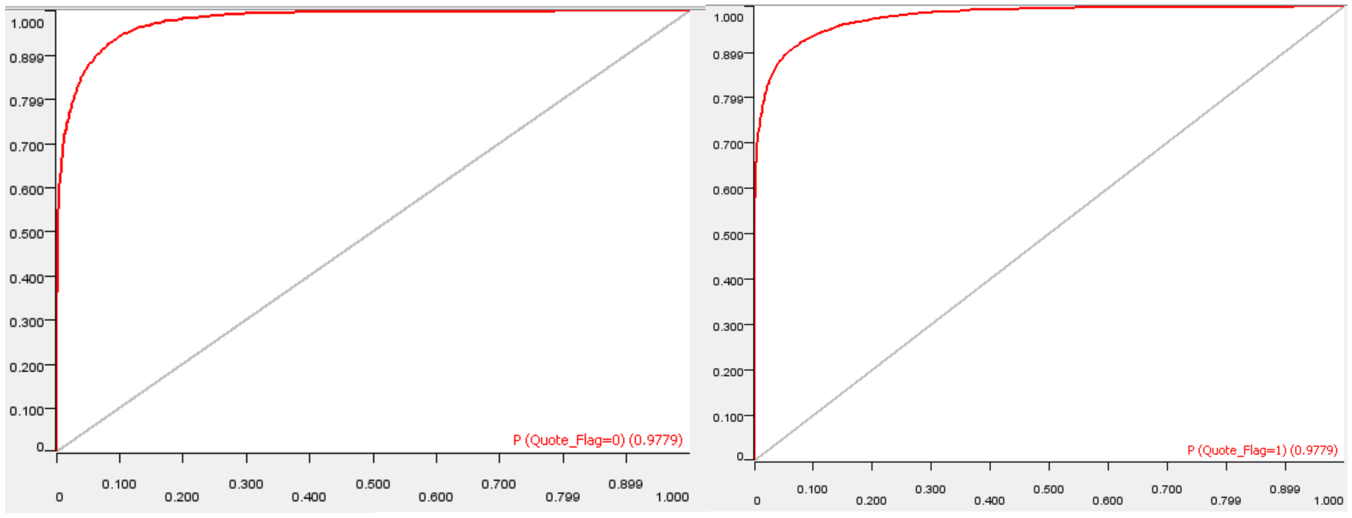


Figure 23 ROC-Curves for Random Forest Learner

As seen above, the results for the Random Forest Learner are really accurate as seen in the confusion matrix in fig. 21 which is 92.103%. The ROC-Curve reflects the same as we can see that it is almost touching one on the top left corner. Furthermore, we see an ideal f-measure score for both the variables as well. This classifier can be a potential ideal classifier to predict Quote_Flag until a better one is found which can further be worked on.

3.5 Tree Ensemble Learner

Tree Ensemble Learner, another classifier like Decision Tree Learner which can handle a lot of variables, numeric and categorical both types of values. The following classifier was made for the given dataset. The workflow can be seen below in fig. 26.

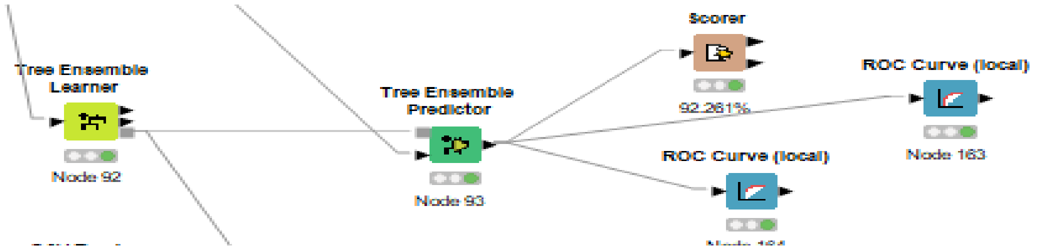


Figure 28 KNIME Workflow for Tree Ensemble Learner

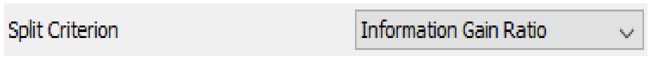
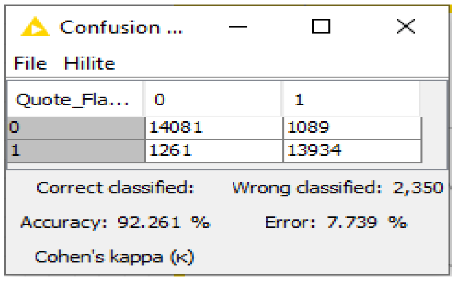

Split Criterion: Information Gain Ratio

Figure 24 Setting for Tree Ensemble Learner



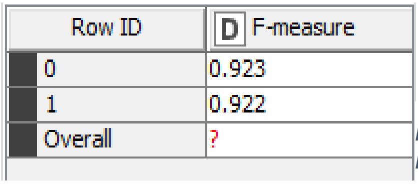
Quote_Fla...	0	1
0	14081	1089
1	1261	13934

Correct classified: Wrong classified: 2,350

Accuracy: 92.261 % Error: 7.739 %

Cohen's kappa (κ)

Figure 26 Confusion Matrix for Tree Ensemble Learner



Row ID	D F-measure
0	0.923
1	0.922
Overall	?

Figure 27 f-score for Tree Ensemble Learner

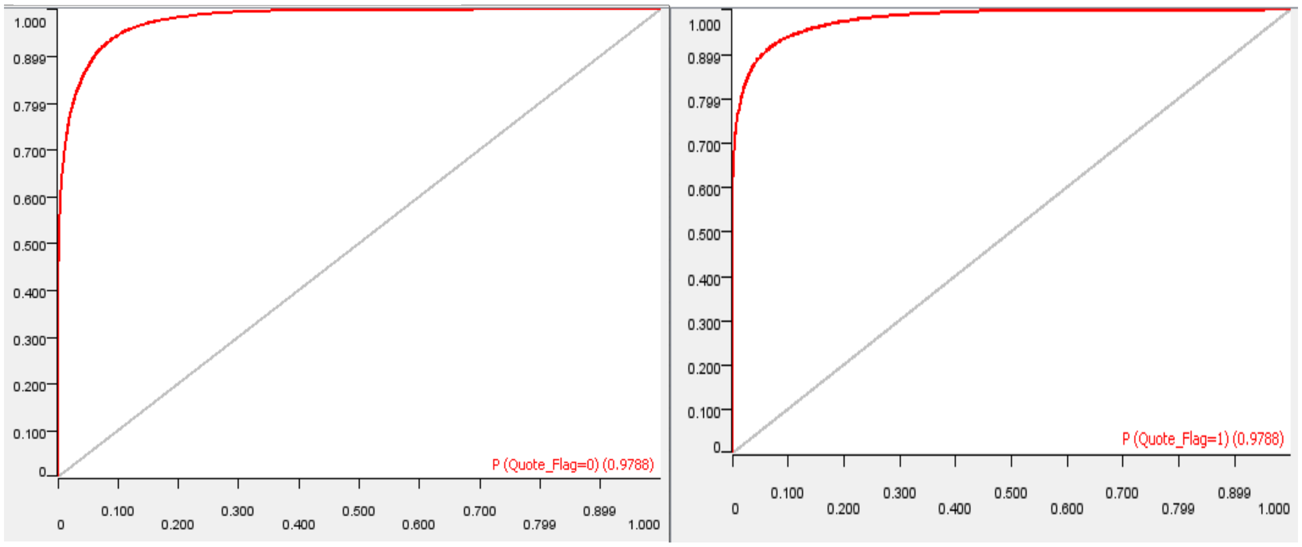


Figure 25 ROC-Curves for Tree Ensemble Learner

As seen above, the results for the Tree Ensemble Learner are really accurate as seen in the confusion matrix in fig. 26 which are 92.261%. The ROC-Curve reflects the same as we can see that it is almost touching one on the top left corner in fig. 24. Furthermore, we see an ideal f-measure score for both the variables as well. This classifier can be a potential ideal classifier to predict Quote_Flag until a better one is found which can further be worked on.

3.6 Neural Network (RPropMLP Learner)

Neural Networks are one of the classifiers which make a spider like network like a nervous system much like the human brain to predict an outcome. The following Neural Network RPropMLP Learner was used. The workflow can be seen below:

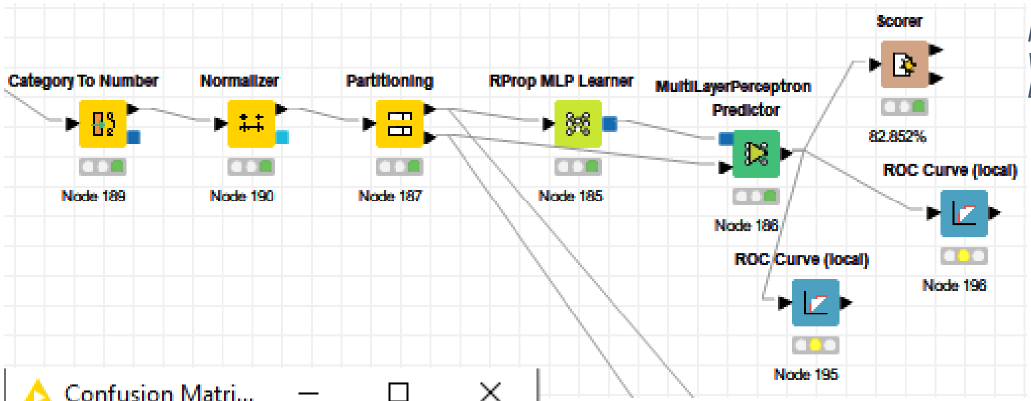
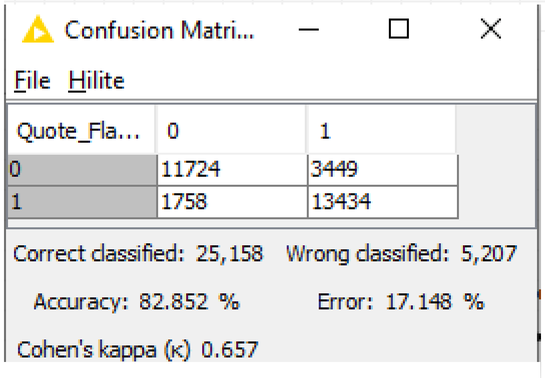


Figure 29 KNIME Workflow for RPropMLP Learner


Confusion Matri...

File Hilite

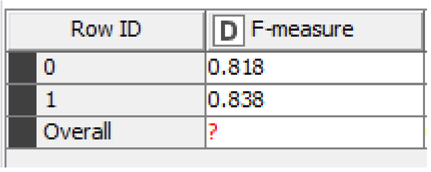
Quote_Fla...	0	1
0	11724	3449
1	1758	13434

Correct classified: 25,158 Wrong classified: 5,207

Accuracy: 82.852 % Error: 17.148 %

Cohen's kappa (κ) 0.657

Figure 30 Confusion Matrix for RPropMLP Learner



Row ID	D F-measure
0	0.818
1	0.838
Overall	?

Figure 31 f-measure for RPropMLP Learner

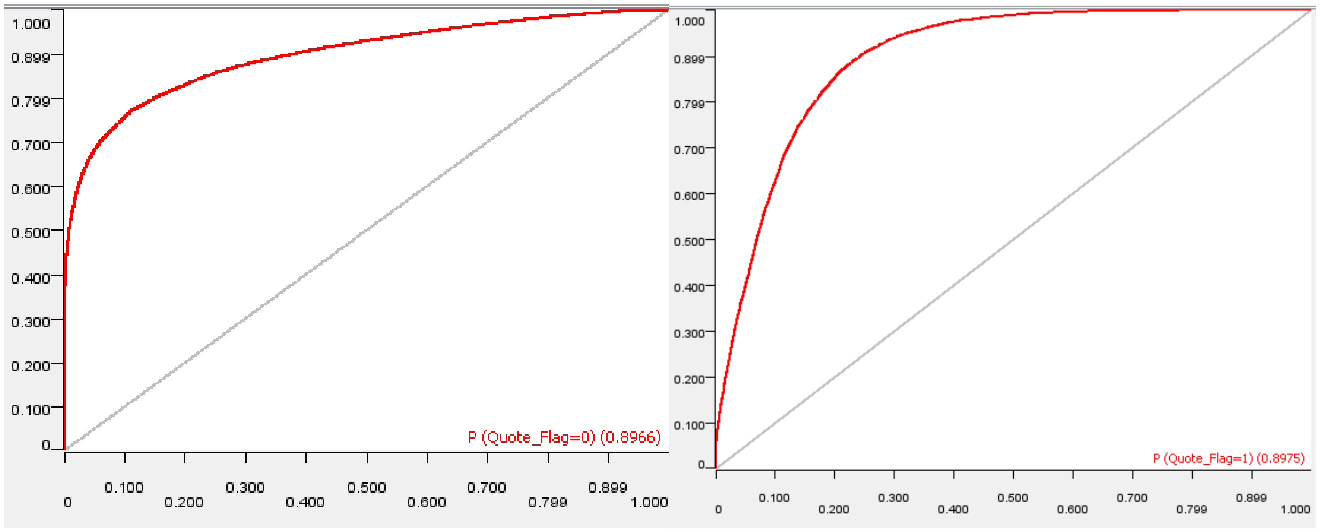
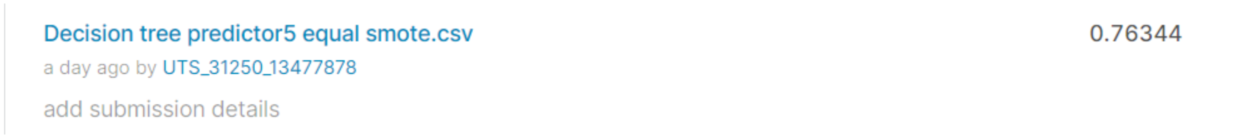


Figure 32 ROC-Curves for RPropMLP Learner

As seen above, the results for the RPropMLP Learner are accurate as seen in the confusion matrix in fig. 30 which are 82.852%. The ROC-Curve reflects the same as we can see that it is not as good as the previous classifiers as seen in fig. 32. Furthermore, we do not see an f-measure score for both the variables as well. This classifier can't be a potential ideal classifier to predict Quote_Flag hence it will be discarded.

4 Chosen Classifier and Conclusion

The chosen classifier must be that which performs best on the unknown dataset. Therefore, when all these models were applied on the unknown dataset, the Decision Tree Learner produced the best results for us to work with as verified on Kaggle.com where the produced results can be verified.



Decision tree predictor5 equal smote.csv 0.76344

a day ago by UTS_31250_13477878

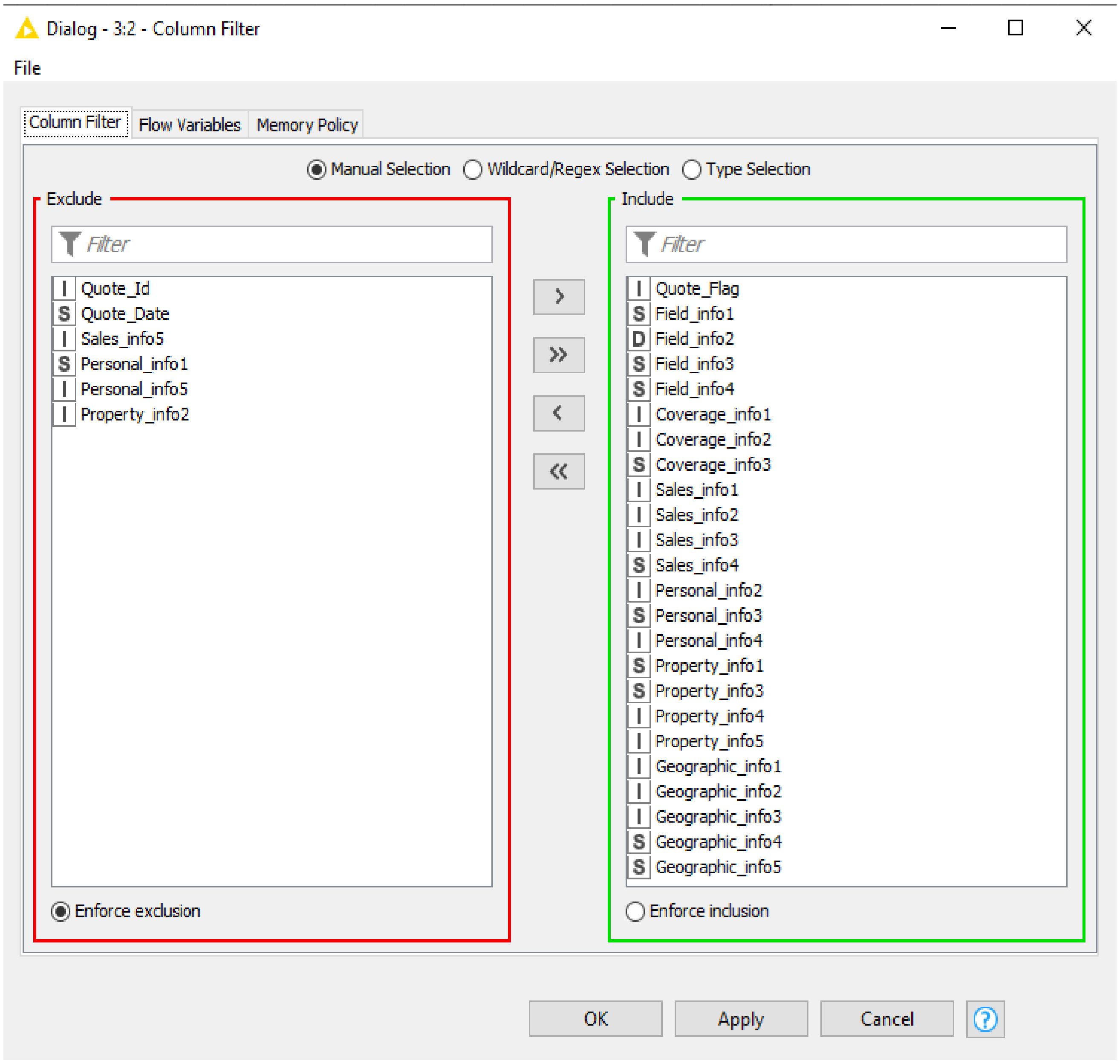
add submission details

Among all the models, SVM was the worst as it needed a lot of pre-processing and computation time to run and produced the worst results. Also, the Decision Tree Learner is works well with the amount of attributes we have and the mix of numerical and categorical attributes.

5 Appendix

5.1 Appendix – A: Settings of nodes used in Pre-processing.

5.1.1 Column Filter Node



Dialog - 3:2 - Column Filter

File

Column Filter | Flow Variables | Memory Policy

[✓ Manual Selection] Wildcard/Regex Selection Type Selection

Exclude

Filter

I	Quote_Id
S	Quote_Date
I	Sales_info5
S	Personal_info1
I	Personal_info5
I	Property_info2

[✓ Enforce exclusion]

Include

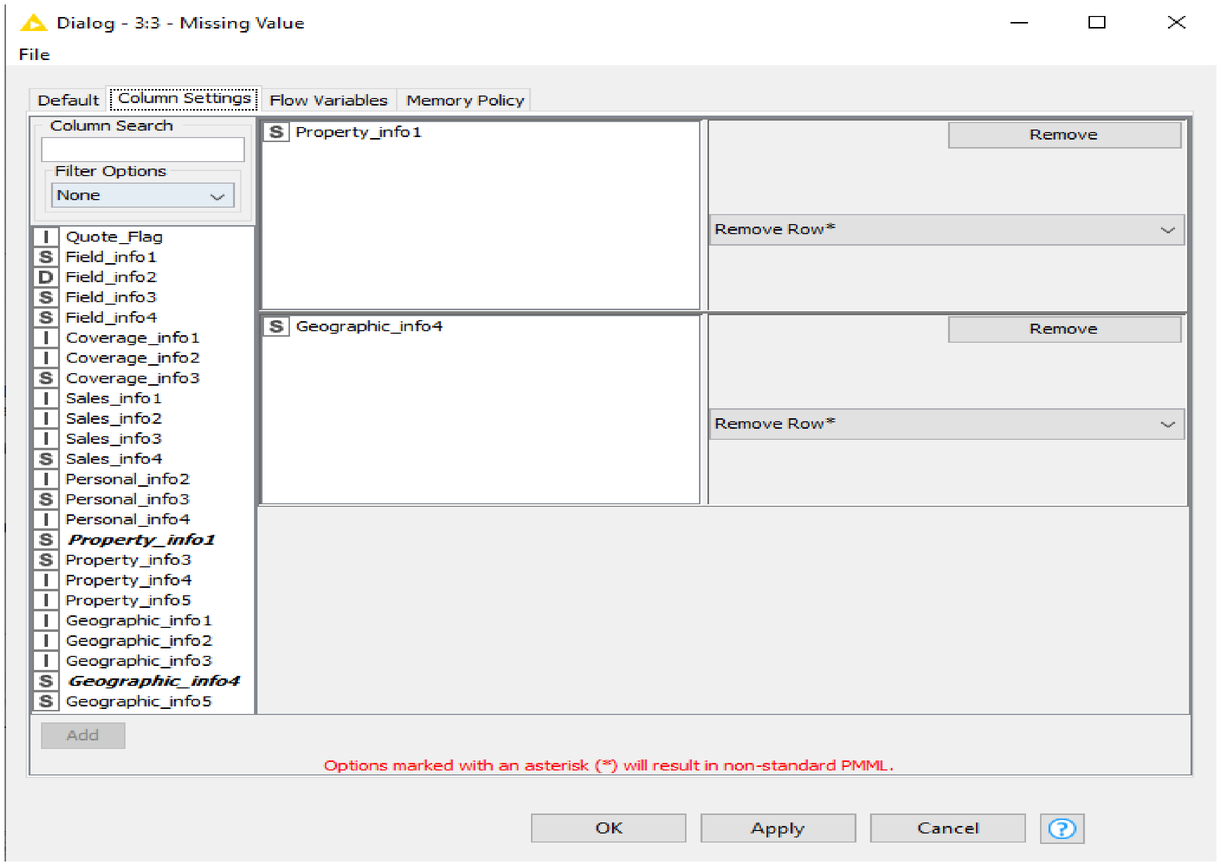
Filter

I	Quote_Flag
S	Field_info1
D	Field_info2
S	Field_info3
S	Field_info4
I	Coverage_info1
I	Coverage_info2
S	Coverage_info3
I	Sales_info1
I	Sales_info2
I	Sales_info3
S	Sales_info4
I	Personal_info2
S	Personal_info3
I	Personal_info4
S	Property_info1
S	Property_info3
I	Property_info4
I	Property_info5
I	Geographic_info1
I	Geographic_info2
I	Geographic_info3
S	Geographic_info4
S	Geographic_info5

Enforce inclusion

OK Apply Cancel

5.1.2 Missing Value Node


Dialog - 3:3 - Missing Value

File

Default | Column Settings | Flow Variables | Memory Policy

Column Search

Filter Options: None

- I Quote_Flag
- S Field_info1
- D Field_info2
- S Field_info3
- S Field_info4
- I Coverage_info1
- I Coverage_info2
- S Coverage_info3
- I Sales_info1
- I Sales_info2
- I Sales_info3
- S Sales_info4
- I Personal_info2
- S Personal_info3
- I Personal_info4
- S ***Property_info1***
- S Property_info3
- I Property_info4
- I Property_info5
- I Geographic_info1
- I Geographic_info2
- I Geographic_info3
- S ***Geographic_info4***
- S Geographic_info5

S Property_info1 — Remove — Remove Row*

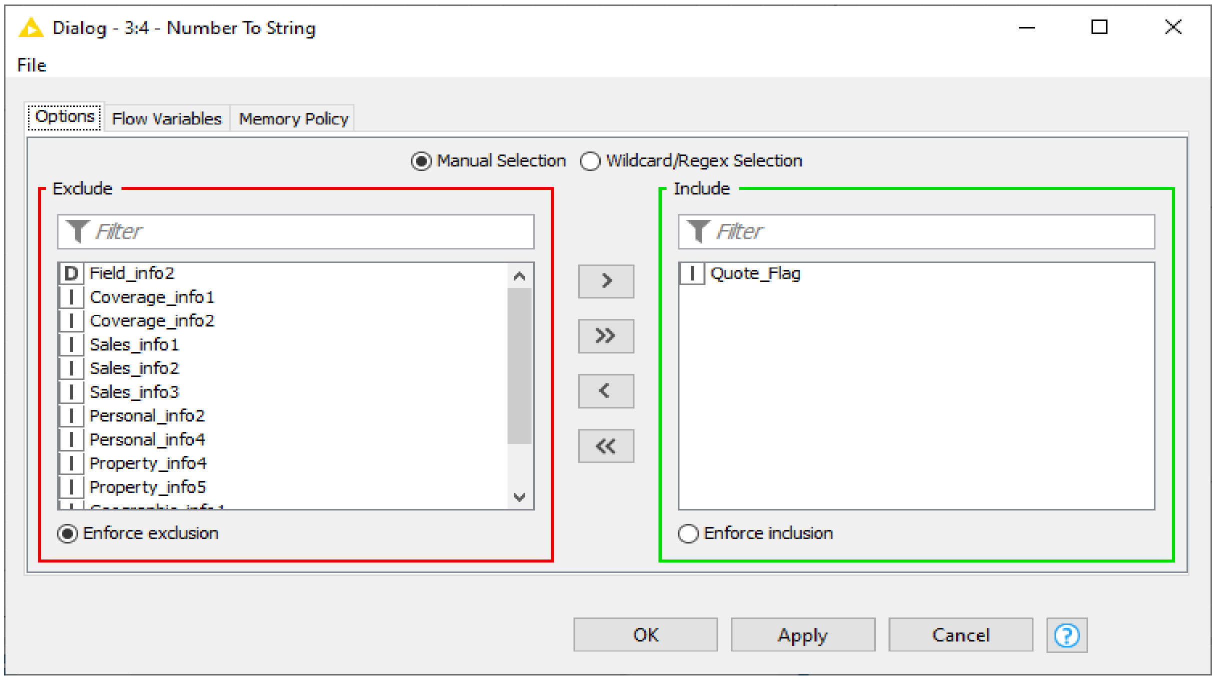
S Geographic_info4 — Remove — Remove Row*

Add

Options marked with an asterisk (*) will result in non-standard PMML.

OK | Apply | Cancel

5.1.3 Number to String Node


Dialog - 3:4 - Number To String

File

Options | Flow Variables | Memory Policy

[● Manual Selection] ○ Wildcard/Regex Selection

Exclude

- D Field_info2
- I Coverage_info1
- I Coverage_info2
- I Sales_info1
- I Sales_info2
- I Sales_info3
- I Personal_info2
- I Personal_info4
- I Property_info4
- I Property_info5

[● Enforce exclusion]

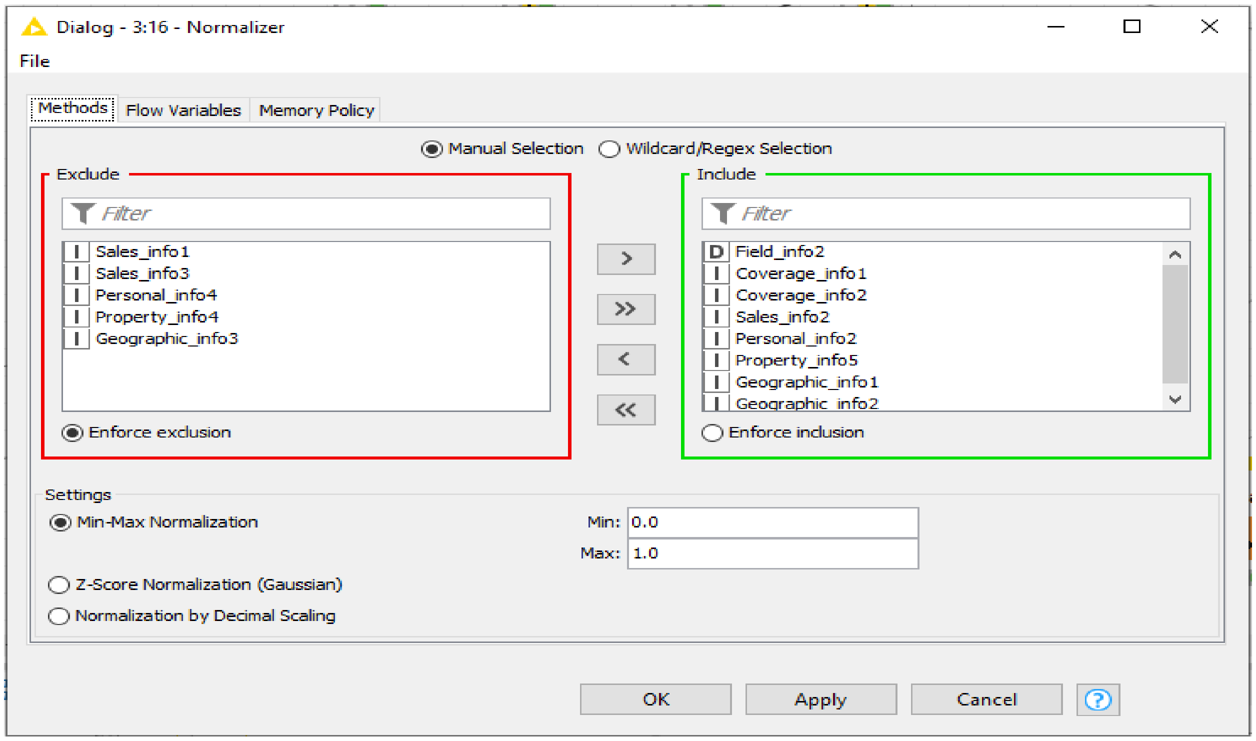
Include

- I Quote_Flag

○ Enforce inclusion

OK | Apply | Cancel

5.1.4 Normalizer Node



Dialog - 3:16 - Normalizer

File

Methods | Flow Variables | Memory Policy

[✓ Manual Selection] Wildcard/Regex Selection

Exclude

Sales_info1
Sales_info3
Personal_info4
Property_info4
Geographic_info3

[✓ Enforce exclusion]

Include

Field_info2
Coverage_info1
Coverage_info2
Sales_info2
Personal_info2
Property_info5
Geographic_info1
Geographic_info2

Enforce inclusion

Settings

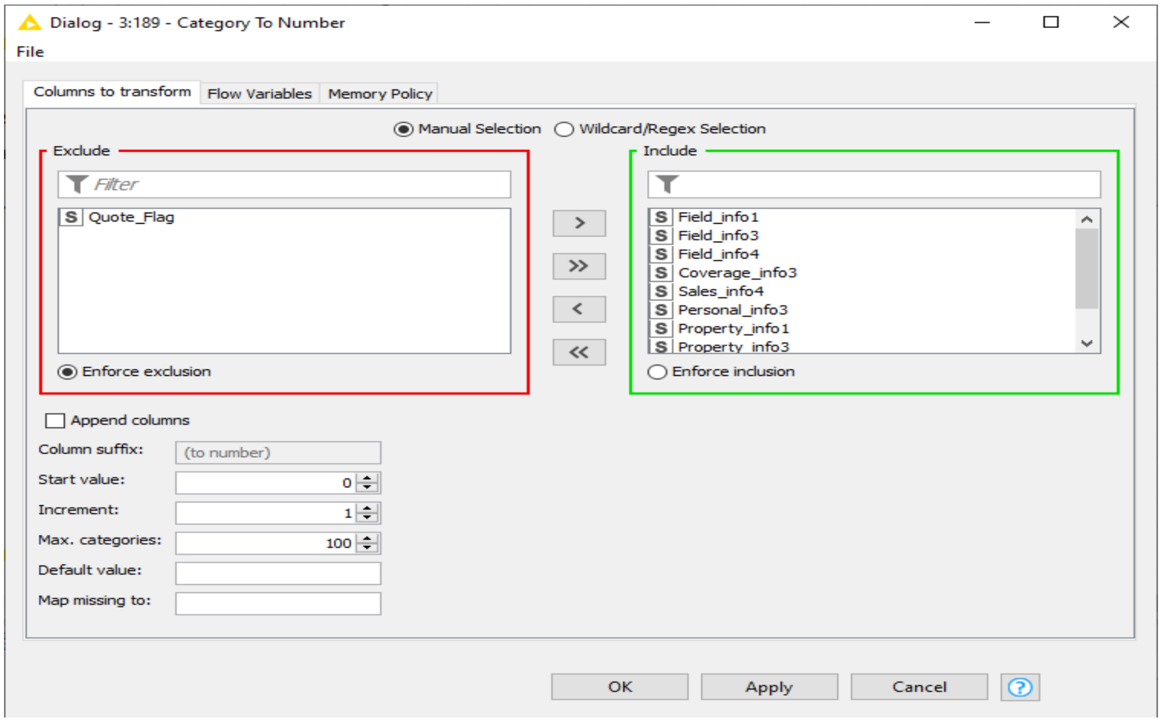
[✓ Min-Max Normalization] Min: 0.0 Max: 1.0

Z-Score Normalization (Gaussian)

Normalization by Decimal Scaling

OK Apply Cancel

5.1.5 Category to Number Node



Dialog - 3:189 - Category To Number

File

Columns to transform | Flow Variables | Memory Policy

[✓ Manual Selection] Wildcard/Regex Selection

Exclude

Quote_Flag

[✓ Enforce exclusion]

Include

Field_info1
Field_info3
Field_info4
Coverage_info3
Sales_info4
Personal_info3
Property_info1
Property_info3

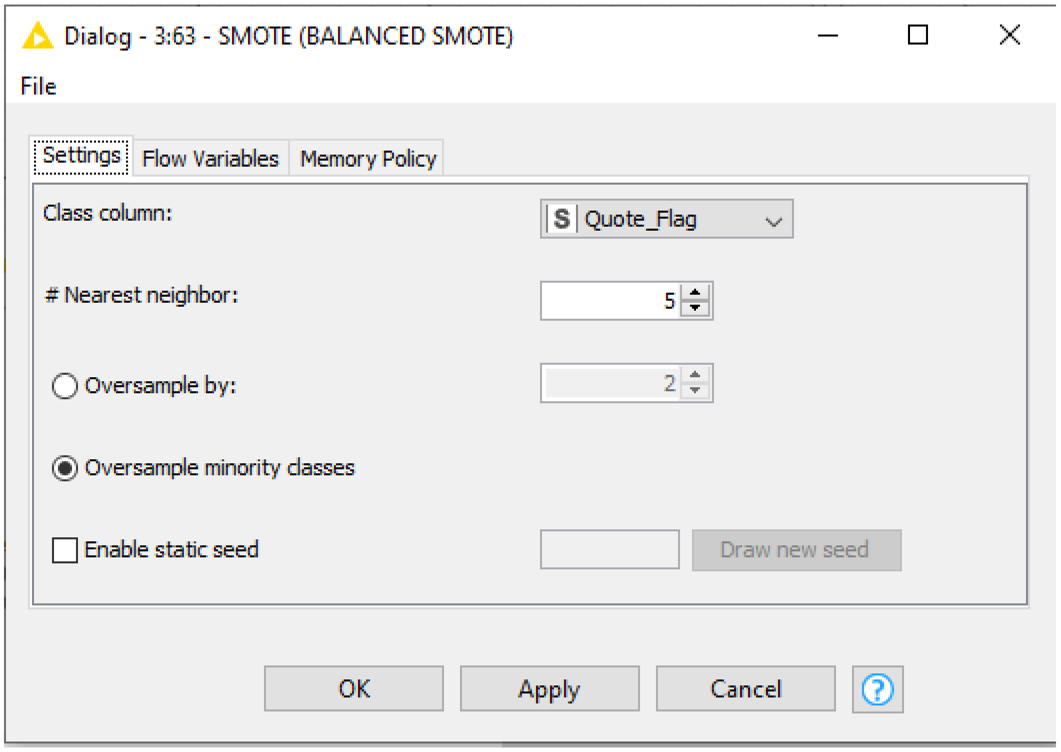
Enforce inclusion

Append columns

Column suffix:	(to number)
Start value:	0
Increment:	1
Max. categories:	100
Default value:	
Map missing to:	

OK Apply Cancel

5.1.6 Smote Node


Dialog - 3:63 - SMOTE (BALANCED SMOTE)

File

Settings | Flow Variables | Memory Policy

Class column: S Quote_Flag

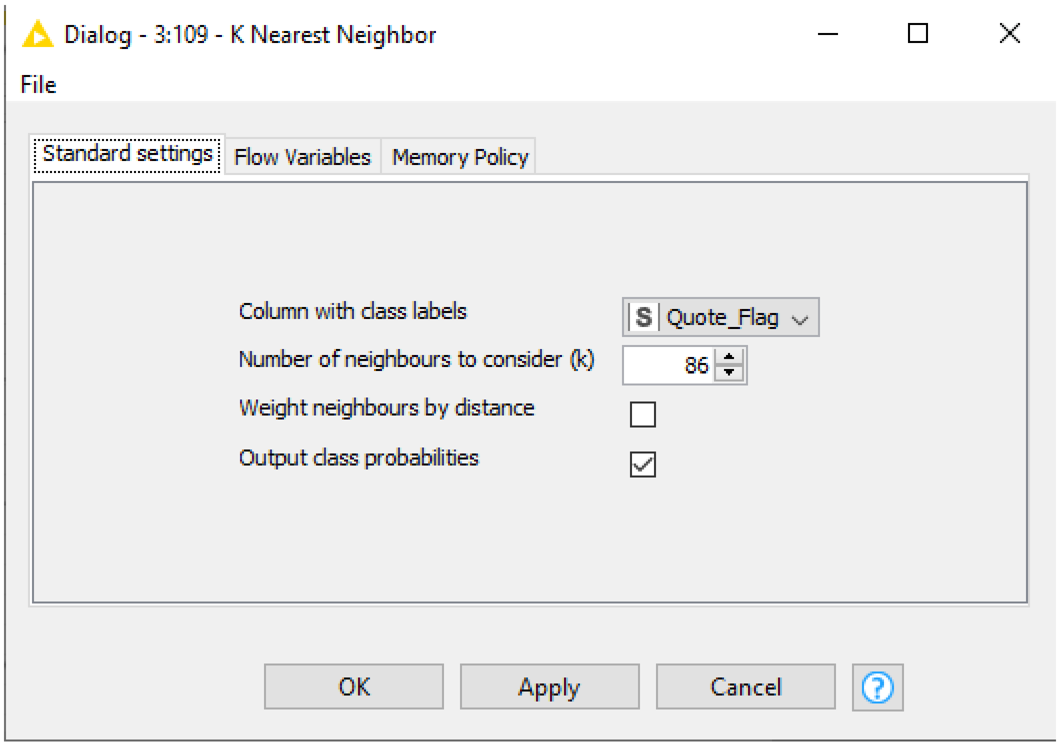
Nearest neighbor: 5

( ) Oversample by: 2

(●) Oversample minority classes

☐ Enable static seed

5.2 Appendix – B: KNN Configuration

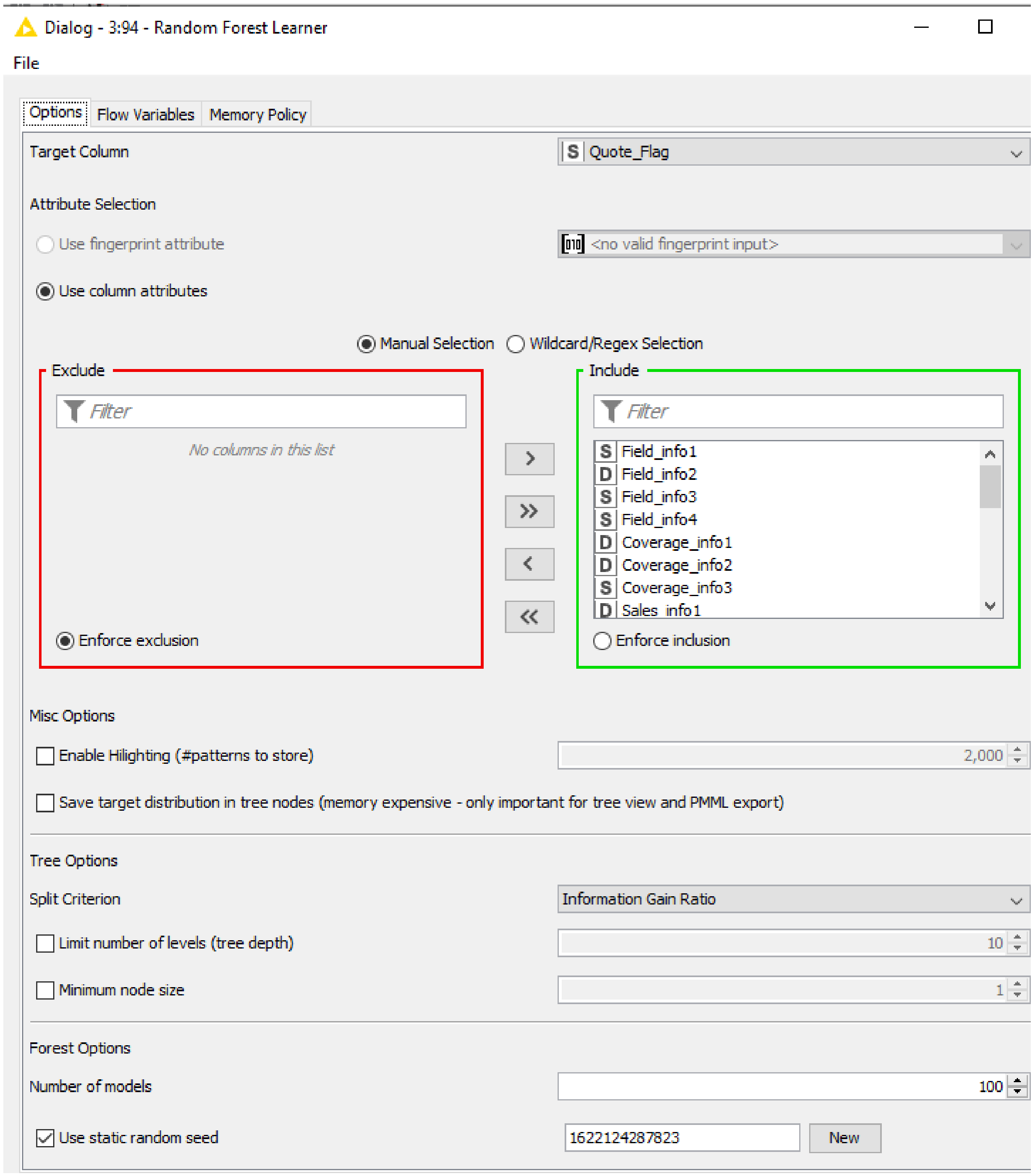

Dialog - 3:109 - K Nearest Neighbor

File

Standard settings | Flow Variables | Memory Policy

Column with class labels	S Quote_Flag
Number of neighbours to consider (k)	86
Weight neighbours by distance	☐
Output class probabilities	✓

5.3 Appendix – C: Random Forest Learner Configuration


Dialog - 3:94 - Random Forest Learner

File

Options | Flow Variables | Memory Policy

Target Column: S Quote_Flag

Attribute Selection

Use fingerprint attribute: <no valid fingerprint input>

[✓ Use column attributes]

[✓ Manual Selection] Wildcard/Regex Selection

Exclude

Filter

No columns in this list

[✓ Enforce exclusion]

Include

Filter

S	Field_info1
D	Field_info2
S	Field_info3
S	Field_info4
D	Coverage_info1
D	Coverage_info2
S	Coverage_info3
D	Sales_info1

Enforce inclusion

Misc Options

☐ Enable Hilighting (#patterns to store) 2,000

☐ Save target distribution in tree nodes (memory expensive - only important for tree view and PMML export)

Tree Options

Split Criterion: Information Gain Ratio

☐ Limit number of levels (tree depth) 10

☐ Minimum node size 1

Forest Options

Number of models: 100

☑ Use static random seed 1622124287823 New