



Data Classification with KNIME

Assessment 3

Contents

Contents	2
Introduction	1
Data mining Problem:	1
Input:	1
Output:	1
Pre-processing	2
Decision tree	3
Parameter optimisation	3
Results	4
K Nearest Neighbour	5
Random Forest Classification	6
Feature Selection	6
Parameter optimisation	7
SVM Classification	8
Parameter optimisation	8
Neural Network Classification	11
Parameter Optimisation	11
C: Summary	13
Kaggle submission and score	13
References	15
Appendix	16

Introduction

Accurately predicting customer subscription behaviour can greatly enhance marketing strategies and optimise resource allocation. This project focuses on predicting the 'subscribed' attribute from a marketing dataset using a range of classification techniques. By utilising multiple methods including Decision Trees, K-Nearest Neighbours, Random Forests, Support Vector Machines (SVM) and Neural Networks, we aim to identify the most efficient approach to deliver high prediction accuracy.

Each technique has a pre-processing stage that handles missing values, encodes categorical data, normalises predictors when needed, and partitions the data into training and testing subsets. The clear visualisation of target values and consistent random seed input further strengthen the reproducibility of our experiments enhancing decision making for targeted marketing strategies by forecasting customer subscription likelihood with precision.

Data mining Problem:

The primary objective of this project is to predict the subscription status of a customer based on historical data containing various attributes. In other words, we need to determine whether a customer will subscribe, using the available features in our dataset. This prediction problem is critical for informing marketing strategies and optimising customer targeting, as an accurate prediction can really assist in decision making.

Our approach will be to use the data analytic tools of KNIME to explore multiple classification techniques including Decision Trees, K-Nearest Neighbours, Random Forests, Support Vector Machines, and Neural Networks to identify the most accurate predictive model. But before this, a set of pre-processing techniques will be applied to ensure data quality and address any issues within the data set that may affect our outcome such as missing values or imbalanced classes.

By comparing the performance of these classifiers and partaking in parameter optimisation our goal is to create a strong process through data training and validation

Input:

The unbalanced 'Marketing dataset' includes historical customer records including demographic details such as job, marital status, education, financial information default status, housing, loan, and other factors including contact method, month. Each record includes the target variable, 'subscribed', indicating whether the customer subscribed to the service. After some pre-processing techniques, this data will be used to train the models create as well as be partitioned to validate our models.

Output:

The analysis will produce a binary prediction (1/0) for the 'subscribed' attribute of each customer record. The final predictions are compiled for Kaggle submission, along with performance insights.

Pre-processing

Beginning with the Marketing Training dataset, I carried out essential pre-processing steps before diving into the classification tasks. First, I utilised two Column Auto Type Cast nodes to identify any values labelled “?” or “unknown” and marked these entries as missing values. Next, a Missing Value node was applied to impute these gaps using the most frequent value for string-based attributes and the median for numeric attributes. Finally, a Number to String node was used to convert the 'subscribed' attribute to string format, ensuring compatibility with the classification models.

Before training the model, I incorporated a Colour Manager node to clearly visualise the target values. In this setup, red represents a value of 0 and green represents a value of 1, making it easier to assess the distribution and verify the target attribute. Lastly, I used a Partitioning node to split the dataset into a 70/30 ratio—70% for training and 30% for testing. To ensure that the results are replicable across all classification experiments, I maintained the default random seed (1745823728740).

These were the fundamental pre-processing techniques carried out during the training and validation phase for all classification models. However, specific models required additional pre-processing steps such as applying Category to Number nodes, normalisers, and other transformations to refine the data further, as you will see later in the report.

Balancing the dataset

As shown in the following pie chart below, the “subscribed” attribute was showing to be highly unbalanced with almost 89% of the outcomes belonging to “0”. To combat this, I used a “SMOTE” node to turn the data into a balanced dataset in hopes of improving the accuracy of the following training models to come. The results after balancing the dataset are clear through the pie charts below.

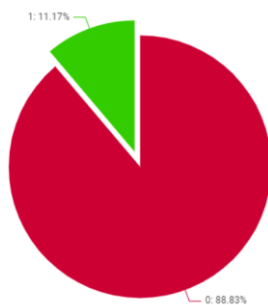


Figure 2: Pie Chart of Subscribed Attribute Before SMOTE Node



Figure 1: Pie Chart of Subscribed Attribute After SMOTE Node

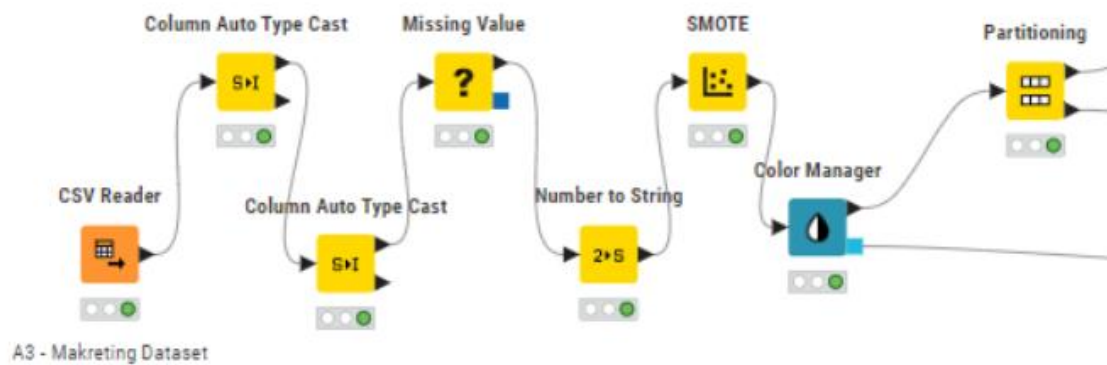


Figure 3: KNIME Workflow: Pre-processing Fundamentals

Decision tree

After the pre-processing techniques were applied, I applied a decision tree learner and decision tree predictor node followed by a scorer node and ROC curve node to test the accuracy of the training model with the default settings of the decision tree learner node.

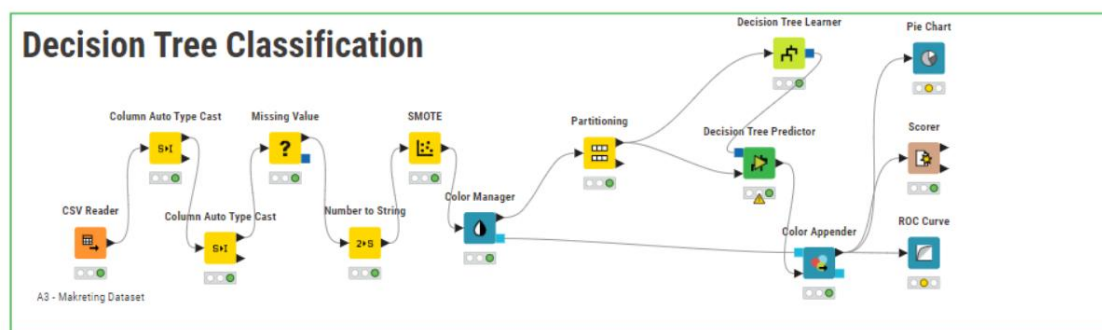


Figure 4: Decision Tree Classification Workflow

Parameter optimisation

Without changing anything, an accuracy rate of 89% was reached. Although this is moderately high for an initial accuracy rate, I decided to apply some parameter optimisation to see if I can increase the accuracy. The quality method as well as the pruning method were changed to understand how the results might be affected. The results of various combinations are highlighted in the table below.

Quality Method	Pruning Method	Accuracy
Gini Index	MDL	92%
Gini Index	No Pruning	92%
Gain Ratio	MDL	91%
Gain Ratio	No Pruning	89%

Since The Gini index method showed for a slightly better accuracy rate, I decided to stick with this method when analysing the ROC curve as well as the final table results.

ROC Curve

Appending the columns within the tree predictor node allowed for an accuracy test using the ROC Curve analysis. Testing for the positive value within the subscribed attribute, we were left with a strong AUC of 0.947

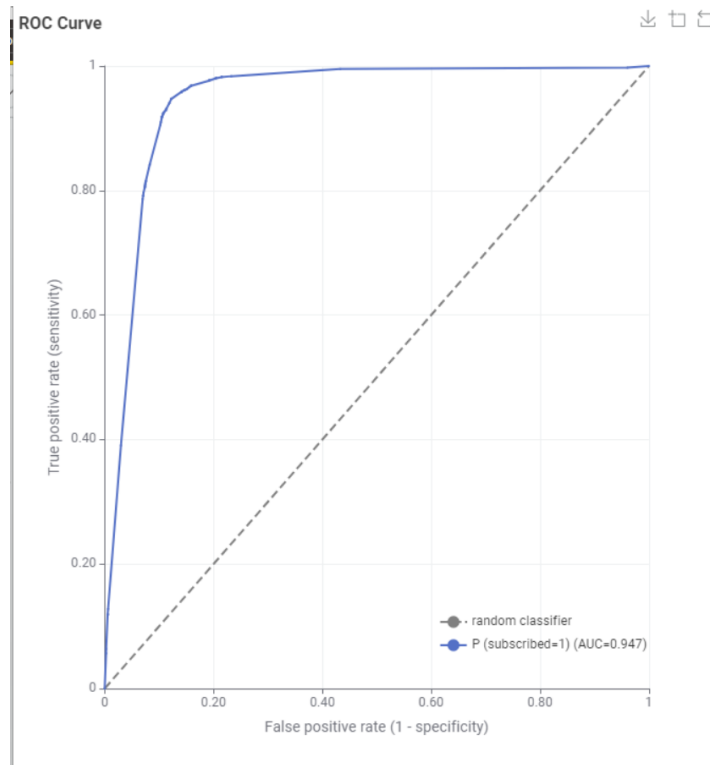


Figure 5: ROC Curve of Decision Tree Classification

Results

The final task is to evaluate the results after essential parameter optimisation was performed and compare the training results to the validation results which is done in a table below.

Rows: 3 Columns: 11										
#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegativ... Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)
1	0	14486	587	15713	1968	0.88	0.961	0.88	0.964	0.919
2	1	15713	1968	14486	587	0.964	0.889	0.964	0.88	0.925
3	Overall									0.922

Figure 6: Decision Tree Training Result Table

Rows: 3 Columns: 11										
#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegativ... Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)
1	0	6092	375	6721	850	0.878	0.942	0.878	0.947	0.909
2	1	6721	850	6092	375	0.947	0.888	0.947	0.878	0.916
3	Overall									0.913

Figure 7: Decision Tree Validation Result Table

	Accuracy	Recall	Precision	F1 Score	Sensitivity	Specificity	AUC
Training	92%	96%	89%	93%	96%	88%	95%
Validation	92%	94%	89%	92%	95%	88%	95%

K Nearest Neighbour

Using the same pre-processing techniques, including the same random seed used in the previous partitioning node, the dataset was run through a K nearest neighbour classification technique.

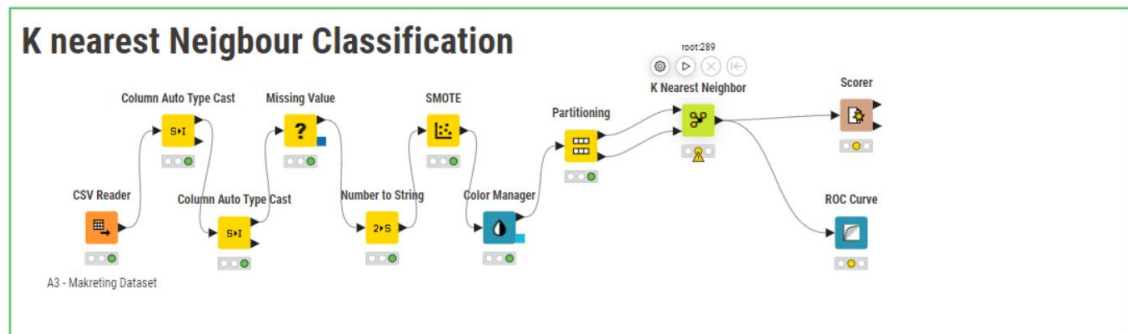


Figure 8: K Nearest Classification Workflow

With the default settings, we get an accuracy rate of approximately 92%. I decided to adjust the “Number of neighbours to consider” setting within the K nearest Neighbour node in attempts to increase the accuracy. Below is a table of the results

No. of neighbours	Accuracy rate
2	91.9%
3	91.6%
5	90.9%
7	90.5%

It seems that the smaller number of neighbours included in the node, the accuracy rate increased incrementally in the scorer node. We will keep this value at 2 and evaluate the final accuracy values for both the training and validating dataset as well as the ROC curve created.

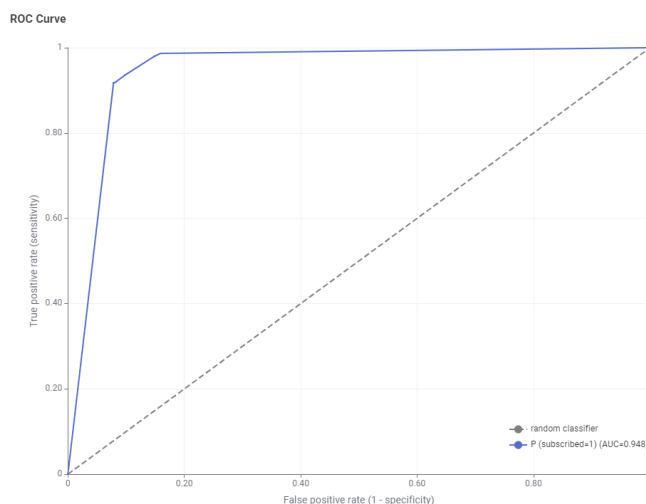


Figure 9: ROC Curve for K Nearest Neighbour

#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegatives Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's kappa Number (double)
1	0	6264	457	6639	678	0.902	0.932	0.902	0.936	0.917	0	0
2	1	6639	678	6264	457	0.936	0.907	0.936	0.902	0.921	0	0
3	Overall	0	0	0	0	0	0	0	0	0	0.919	0.838

Figure 10: K Nearest Neighbour Training Result Table

#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegatives Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's kappa Number (double)
1	0	15964	490	15810	490	0.97	0.97	0.97	0.97	0.97	0	0
2	1	15810	490	15964	490	0.97	0.97	0.97	0.97	0.97	0	0
3	Overall	0	0	0	0	0	0	0	0	0	0.97	0.94

Figure 11: K Nearest Neighbour Validation Results Table

	Accuracy	Recall	Precision	F1 Score	Sensitivity	Specificity	AUC
Training	97%	97%	97%	97%	97%	97%	99.8%
Validation	92%	94%	90%	92%	94%	90%	95%

Random Forest Classification

Continuing with the same pre-processing method and assigning them to a random forest classification process which includes a Random Forest Learner node, followed by the Random Forest Predictor node, the Scorer node shows an accuracy rate of 96% with the default settings.

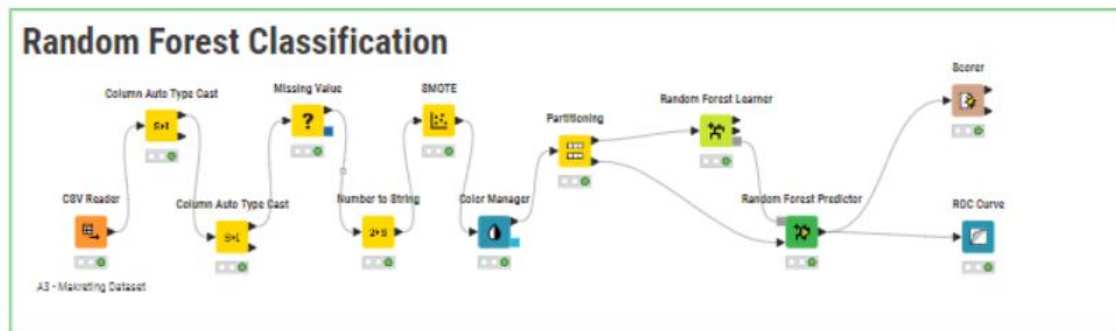


Figure 12: Random Forest Workflow

Feature Selection

Attempting to increase this accuracy rate, the relevant columns used to create this prediction are analysed in the "Attribute statistics" section within the Random Forest Learner node.

Rows: 21 | Columns: 6

Table Statistics

#	RowID	#splits (level 0) Number (integer)	#splits (level 1) Number (integer)	#splits (level 2) Number (integer)	#candidates (level 0) Number (integer)	#candidates (level 1) Number (integer)	#candidates (level 2) Number (integer)
<input type="checkbox"/>	1	age 1	9	11	15	48	74
<input type="checkbox"/>	2	job 0	5	14	16	49	66
<input type="checkbox"/>	3	marital 0	1	2	13	39	76
<input type="checkbox"/>	4	educat 0	6	10	17	35	79
<input type="checkbox"/>	5	default 1	2	5	23	34	81
<input type="checkbox"/>	6	housin 0	0	0	16	34	68
<input type="checkbox"/>	7	loan 0	0	0	25	37	68
<input type="checkbox"/>	8	contac 1	1	3	15	37	72
<input type="checkbox"/>	9	month 8	24	37	23	41	79
<input type="checkbox"/>	10	day_of 0	1	3	16	33	80
<input type="checkbox"/>	11	duratio 16	36	48	16	43	64
<input type="checkbox"/>	12	campa 0	2	13	29	33	79
<input type="checkbox"/>	13	pdays 17	15	22	19	31	76
<input type="checkbox"/>	14	previos 7	11	14	28	39	83
<input type="checkbox"/>	15	poutco 17	18	20	21	45	90
<input type="checkbox"/>	16	emp.vi 11	15	27	23	40	75
<input type="checkbox"/>	17	cons.p 3	5	25	14	35	81
<input type="checkbox"/>	18	cons.c 5	17	37	21	37	72
<input type="checkbox"/>	19	euribor 6	9	27	14	31	68
<input type="checkbox"/>	20	nr.emp 7	22	42	17	40	88

Figure 13: Random Forest Learner Attribute Statistics

Parameter optimisation

The final parameter optimisation was done by applying different split correlations in the Random Forest Learner. Between the three options: information gain, information gain ratio and lastly, Gini index, the results changed less than 1%. Although, since Gini Index was slightly higher, it was chosen for the final split correlation.

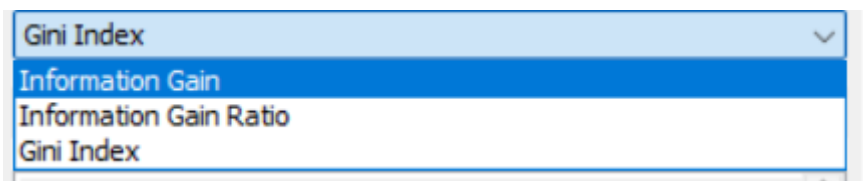


Figure 14: Random Forest Learner options

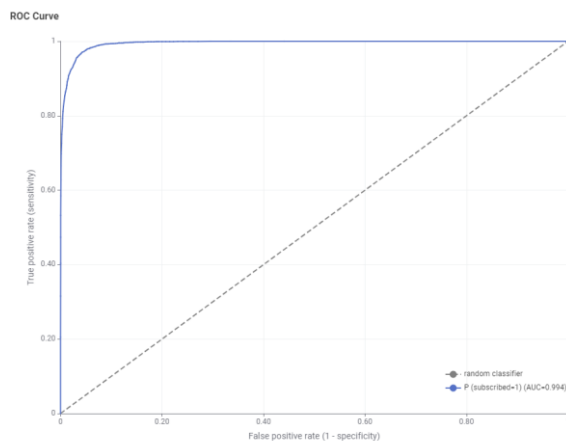


Figure 15: ROC Curve for Random Forest

Rows: 3 Columns: 11													Table Statistics		
<input type="checkbox"/>	#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegatives Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's kappa Number (double)		
<input type="checkbox"/>	1	0	6452	104	6992	490	0.929	0.984	0.929	0.985	0.956				
<input type="checkbox"/>	2	1	6992	490	6452	104	0.985	0.935	0.985	0.929	0.959				

Figure 16: Random Forest Training Results Table

#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegatives Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's kappa Number (double)
1	0	16406	21	16279	48	0.997	0.999	0.997	0.999	0.998	0.998	0.996
2	1	16279	48	16406	21	0.999	0.997	0.999	0.997	0.998	0.998	0.996
3	Overall										0.998	0.996

Figure 17: Random Forest Validation Results Table

	Accuracy	Recall	Precision	F1 Score	Sensitivity	Specificity	AUC
Training	99.8%	99.9%	99.9%	99.8%	99.7%	99.9%	100%
Validation	96%	99%	94%	96%	99%	93%	99%

SVM Classification

Before using the SVM Learner and SVM Predictor nodes on KNIME, it is important that we convert all the categorical columns used for input into numeric values. The subscribed column however as it is our target column can stay a string. We will also change the partitioning percentage to 5% for training and incorporate normalisation before executing the training model.

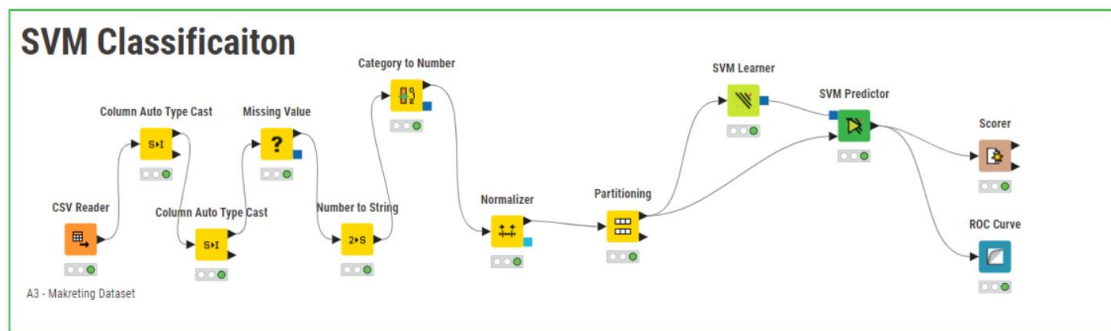


Figure 18: KNIME Workflow for SVM Classification

Parameter optimisation

With the default parameters, we achieved an accuracy rate of 92%. We will attempt to achieve a higher accuracy rate by altering the configuration of the SVM learner node. Beginning by testing which kernel works best. We tested for the default settings for Polynomial, Hyper Tangent and lastly RBF. We then altered the values of each kernel.

Polynomial

Bias	Power	Gamma	Accuracy rate
1	1	1	92%
1	2	1	95%
2	2	2	95%
2	1	2	92%
2	1	1	92%
1	1	2	92%

Hyper Tangent

Kappa	Delta	Accuracy rate
0.1	0.5	89%
0.1	1	88%
0.1	2	88%
0.1	1	9%

RBF

Sigma	Accuracy rate
0.1	100%
0.3	100%
0.5	100%
0.8	100%
1	100%

The training data shows for an accuracy rate of 100% when using the RBF kernel of this node no matter the parameter values. I wanted to test if this would hold up when validating the dataset, so I ran the test of different values again, this time with the validation model instead of the training model and using parameter optimisation loop nodes to allow the dataset to test itself against 10 different sigma values in 0.1 intervals

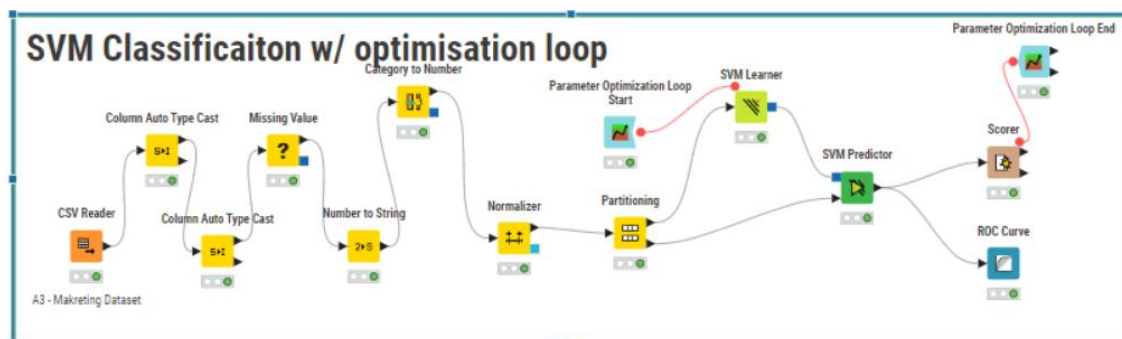


Figure 19: KNIME workflow of SVM Classification with optimisation loop

► 1: Best parameters ► 2: All parameters ⚙ Flow Variables

#	RowID	Sigma	Objective value
		Number (double)	Number (double)
<input type="checkbox"/>	1	Row0 0.1	0.887
<input type="checkbox"/>	2	Row1 0.2	0.887
<input type="checkbox"/>	3	Row2 0.3	0.887
<input type="checkbox"/>	4	Row3 0.4	0.887
<input type="checkbox"/>	5	Row4 0.5	0.887
<input type="checkbox"/>	6	Row5 0.6	0.887
<input type="checkbox"/>	7	Row6 0.7	0.887
<input type="checkbox"/>	8	Row7 0.8	0.887
<input type="checkbox"/>	9	Row8 0.9	0.887
<input type="checkbox"/>	10	Row9 1	0.887

Figure 20: Results of SVM after utilising parameter loop nodes

Unfortunately, the accuracy rate wasn't retained but also remained unchanged when using different Sigma values. Using this method moving forward, we look at the ROC Curve and results table.

ROC Curve

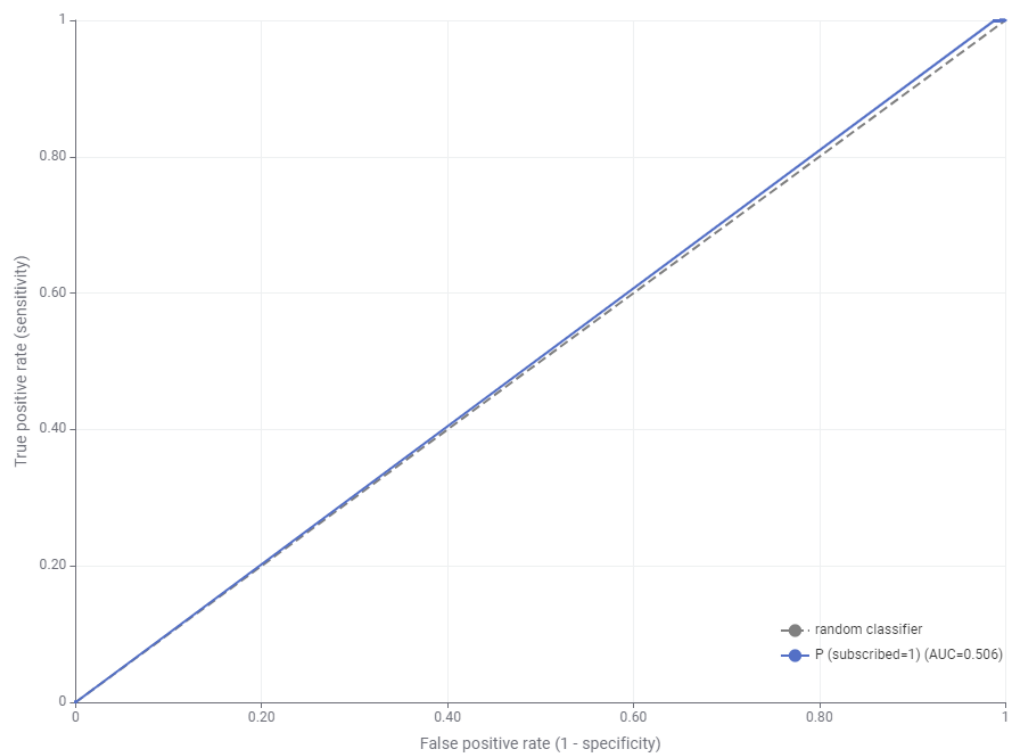


Figure 21: ROC Curve for SVM

Rows: 3 | Columns: 11

#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegativ... Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's kappa Number (double)
1	0	1180	0	138	0	1	1	1	1	1	1	1
2	1	138	0	1180	0	1	1	1	1	1	1	1
3	Overall										1	1

Figure 22: Training for SVM Classification Table Results

Rows: 3 | Columns: 11

#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegativ... Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's kappa Number (double)
1	0	22216	2826	0	0	1	0.887	1	0	0.94		
2	1	0	0	22216	2826	0		0	1			
3	Overall										0.887	0

Figure 23: Validation for SVM Classification Table Results

	Accuracy	Recall	Precision	F1 Score	Sensitivity	Specificity	AUC
Training	100%	100%	100%	100%	100%	100%	100%
Validation	89%	0%	89%	94%	0%	100%	50%

Neural Network Classification

Using the category to number node in this workflow again, we now train and test the data through neural network classification but utilising the MLP Learner and Predictor nodes.

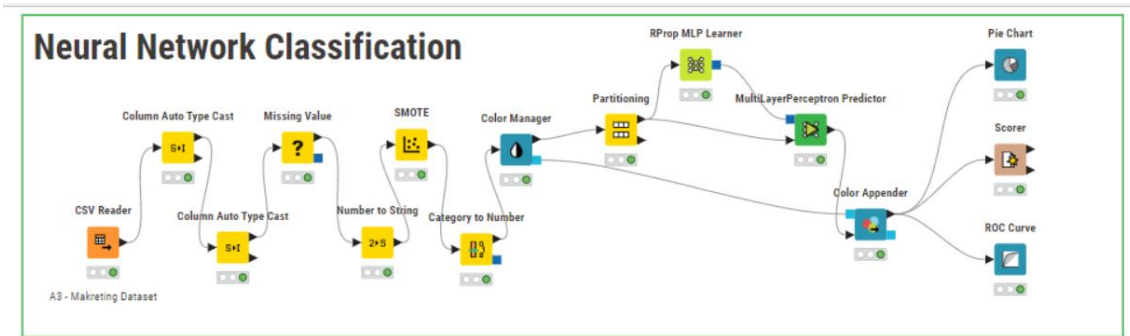


Figure 24: KNIME Workflow of Neural Network Classification

With the default settings we reached a low accuracy rate of approximately 50% as shown by the scorer node. This is significantly lower than all the other models used so I wanted to experiment with a view different parameters to see if we could at least get it close to the accuracy rate of the other models before counting this model out.

Parameter Optimisation

In attempts to increase this accuracy, we shall optimise the parameters beginning with adjusting the number of hidden layers and number of hidden neurons per layer withing the settings of the learner node to understand how this effects the outcome.

No. of hidden layers	No. hidden neurons per layer	Accuracy rate
1	5	50%
1	10	50%
1	15	50%
1	20	86%
1	25	50%
2	20	87%
3	20	50%
5	20	82%
10	20	50%

As you can see by the highlighted value, the best result was found with 2 hidden layers and 20 hidden neurons per layer. It's now time to explore this value in the ROC Curve and complete the validation process

ROC Curve

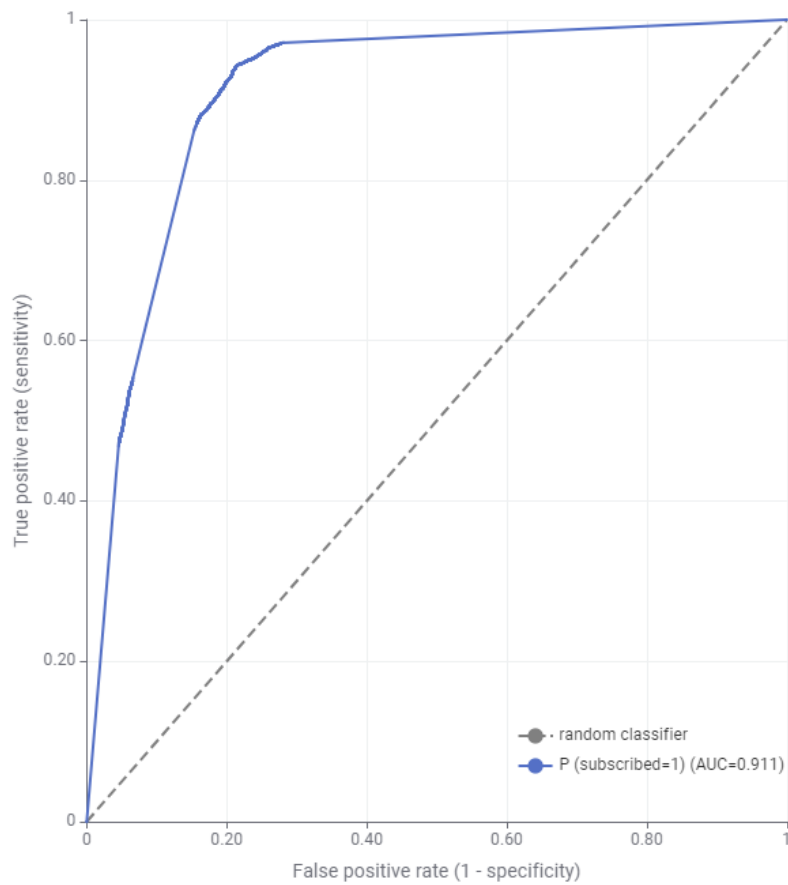


Figure 25: ROC Curve of Neural Network Classification

Rows: 3 Columns: 11										
#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegatives Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)
1	0	12920	934	15366	3534	0.785	0.933	0.785	0.943	0.853
2	1	15366	3534	12920	934	0.943	0.813	0.943	0.785	0.873
3	Overall									0.864

Figure 26: Neural Network Training Results Table

Rows: 3 Columns: 11										
#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegatives Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)
1	0	5427	406	6690	1515	0.782	0.93	0.782	0.943	0.85
2	1	6690	1515	5427	406	0.943	0.815	0.943	0.782	0.874
3	Overall									0.863

Figure 27: Neural Network Training Results Table

	Accuracy	Recall	Precision	F1 Score	Sensitivity	Specificity	AUC
Training	86%	94%	81%	87%	94%	79%	91%
Validation	86%	94%	81%	87%	94%	78%	91%

C: Summary

Model	Accuracy	Recall	Precision	F1 Score	Sensitivity	Specificity	AUC
Decision Tree	92%	94%	89%	92%	95%	88%	95%
K Nearest Neighbour	92%	94%	90%	92%	94%	90%	95%
Random Forest	96%	99%	94%	96%	99%	93%	99%
SVM	89%	0%	89%	94%	0%	100%	50%
Neural Network	86%	94%	81%	87%	94%	78%	91%

The comparative analysis of multiple classifiers shows that the Random Forest model created the highest accuracy rate of 96% while the Decision Tree, and K Nearest Neighbour each achieved an accuracy of 92%, while the Neural Network model reached a lower rate of 87% accuracy.

From a business perspective, the consistently high performance of the Random Forest models supports their use for targeted marketing strategies, ensuring reliable predictions of customer subscription behaviour. This not only aids in efficient resource allocation but also enhances decision-making processes. Future fine-tuning, especially for the SVM, may provide even deeper insights into the data patterns, further bolstering our predictive capabilities.

Kaggle submission & score

According the Kaggle, the most accurate model giving the highest submission score of 0.62658 is the Random Forest Classification. Close behind is the Decision Tree method with 0.61213








	SVM.csv Complete · 1h ago	0.00000	<input type="checkbox"/>
	Neural_Network.csv Complete · 4h ago	0.51111	<input type="checkbox"/>
	random forest.csv Complete · 13d ago	0.62658	<input type="checkbox"/>
	random forest.csv Complete · 13d ago	0.62211	<input type="checkbox"/>
	k nearest neighbour.csv Complete · 13d ago	0.54511	<input type="checkbox"/>
	k nearest neighbour.csv Error · 13d ago		
	Decision Tree.csv Complete · 14d ago · Decision Tree Classification	0.61213	<input type="checkbox"/>

Figure 28: Kaggle Results

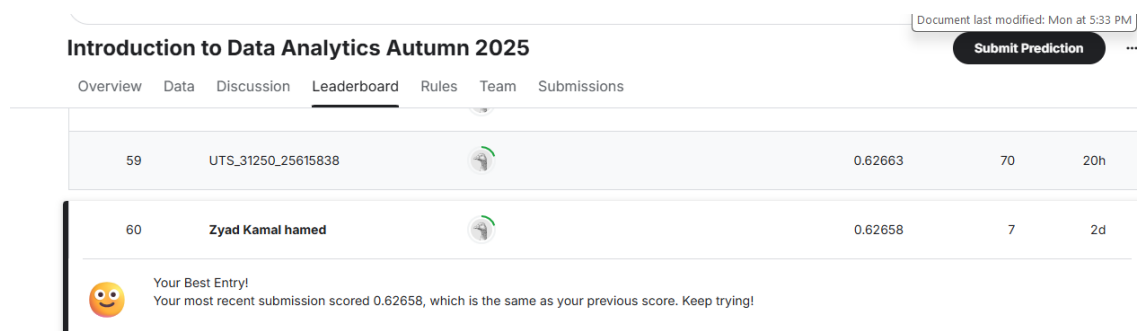


Figure 29: Kaggle Position

References

Kaggle Competition Leaderboard: Kaggle. (n.d.). *Introduction to Data Analytics Autumn 2025* / Kaggle. Retrieved May 14, 2025, from <https://www.kaggle.com/competitions/introduction-to-data-analytics-autumn-2025/leaderboard#>

Appendix

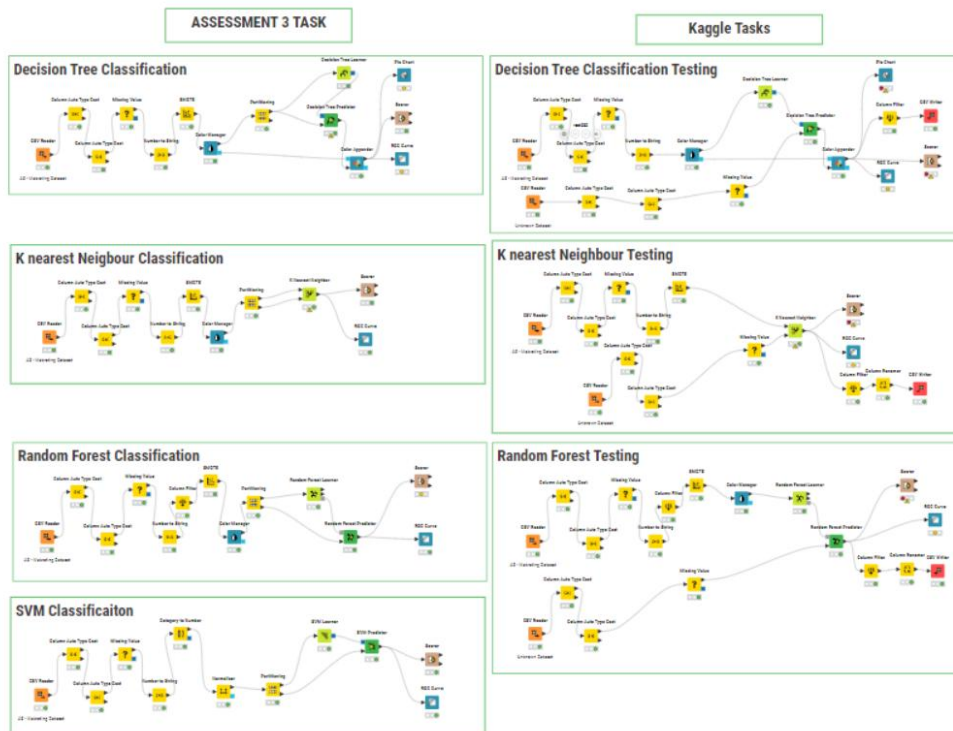


Figure 30: KNIME Workspace Part 1

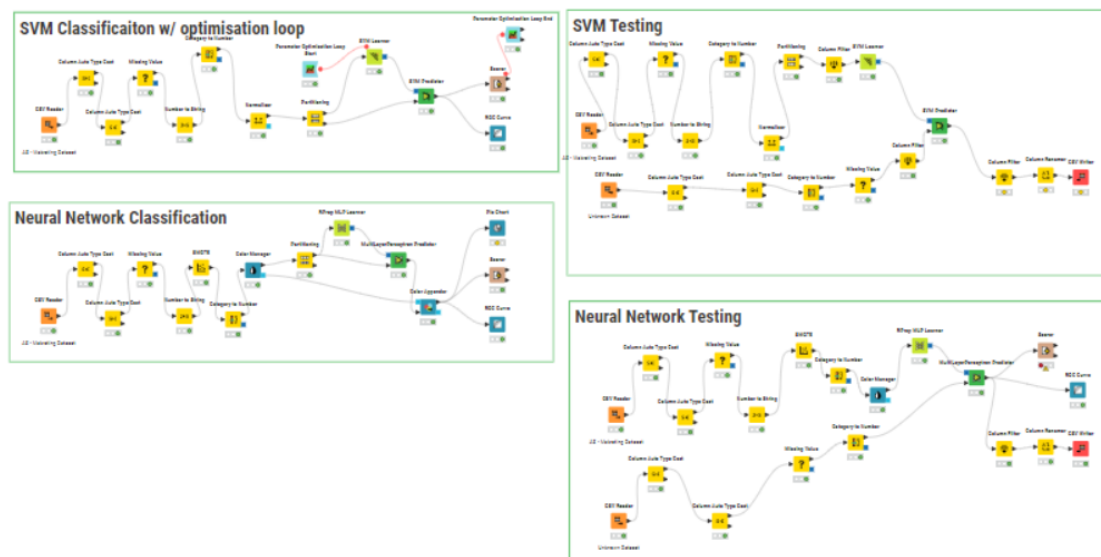


Figure 31: KNIME Workspace Part 2