

Cover sheet for submission of work for assessment



UNIT DETAILS

Unit name	Data Science Principles			Class day/time	Thursday	Office use only
Unit code	COS10022	Assignment no.	2	Due date	26/03/2023	
Name of lecturer	Dr. James Jackson					
Tutor/marker's name	Dr. James Jackson					Faculty or school date stamp

STUDENT(S)

Family Name	Given Name	Student ID Number
Luu	Tuan Hoang	104180391

DECLARATION AND STATEMENT OF AUTHORSHIP

1. I/we have not impersonated, or allowed myself/ourselves to be impersonated by any person for the purposes of this assessment.
2. This assessment is my/our original work and no part of it has been copied from any other source except where due acknowledgement is made.
3. No part of this assessment has been written for me/us by any other person except where such collaboration has been authorised by the lecturer/teacher concerned.
4. I/we have not previously submitted this work for this or any other course/unit.
5. I/we give permission for my/our assessment response to be reproduced, communicated, compared and archived for plagiarism detection, benchmarking or educational purposes.

I/we understand that:

6. Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offence that may lead to exclusion from the University. Plagiarised material can be drawn from, and presented in, written, graphic and visual form, including electronic data and oral presentations. Plagiarism occurs when the origin of the material used is not appropriately cited.

Student signature/s

I/we declare that I/we have read and understood the declaration and statement of authorship.

A handwritten signature in black ink, appearing to be "Luan", written on a light gray background.

COS10022 Data Science Principles

Assignment 2 - HE Semester (Jan), 2023

I. Overview

This assignment focuses on:

- Describe the main ideas, methods, and equipment used to clean the data and build prediction models.
- Choose features and models while doing a small amount of prebuilt tool discovery and implementation in a data science project.

II. Abstract

100,000 tuples from three different financial credit score classes make up the dataset. The original data has 24 attributes in total. The first objective of this project is to clean and prepare the data for use later, and the second objective is to create two predictive models to forecast the "Credit_Score" class.

III. Data preparation (70%)

1. Task 1

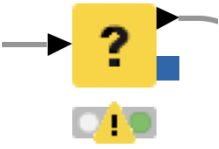
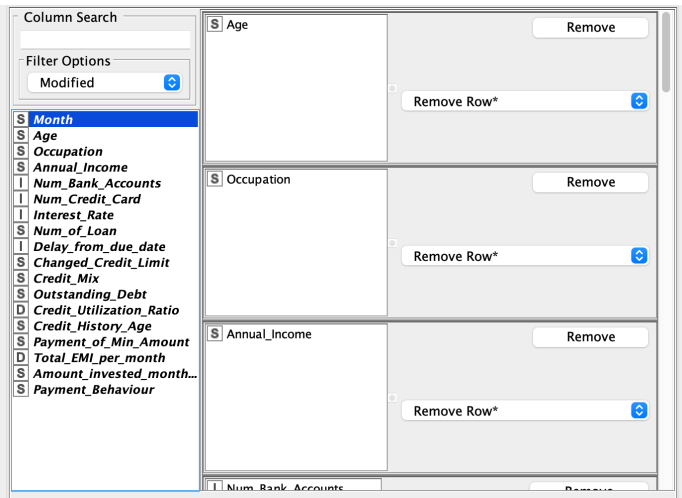

The removed attribute is "Name". In the context of predicting credit score, it is generally recommended to remove the name or any other personally identifiable information from the dataset for model training.

Only data that is relevant to the borrower's creditworthiness, such as their income, credit history, and employment status, should be used in the credit score prediction model.

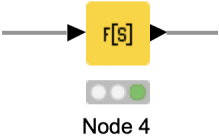
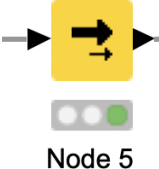
Also it is worth knowing that there are a total of 10137 unique names, which cannot be used for later model training as the learner will automatically ignore the "Name" attribute due to too many unique nominal values in the column, and we will also limit number of unique nominal value per attribute to 600 in Naïve Bayes learner.

Because of aforementioned reasons, "Name" attribute is excluded from the dataset.

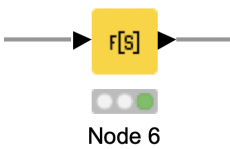
2. Task 2

Sequence	Node	Configuration
1	Missing Value  Node 2	<p>The window below is the configuration for the node. Filter options set to “Modified” to show every column that will be changed if there is any missing value. Missing value handle selection all set to “Remove Row*”.</p> 
2	Rule-based Row Filter  Node 3	<p>Expression (EXCLUDE TRUE MATCHES):</p> <pre> \$Monthly_Inhand_Salary\$ < 0 => TRUE \$Num_Bank_Accounts\$ < 0 => TRUE \$Num_Credit_Card\$ < 0 => TRUE \$Changed_Credit_Limit\$ MATCHES "[^0-9.-]" => TRUE </pre> <p>Infeasible values will be considered as TRUE and any row that contains these values will be excluded in the output.</p>

3. Task 3

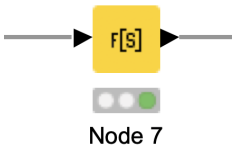
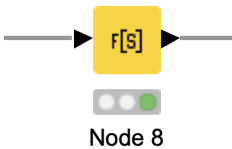
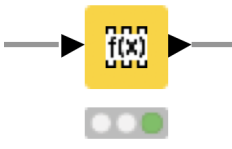
Sequence	Node	Configuration
1	String Manipulation  Node 4	<p>Expression:</p> <pre>toInt(regexReplace(\$Age\$, "[^0-9.-]", ""))</pre> <p>We use regular expressions to form a search pattern for non-numerical data. Pattern brackets with caret are used to find a range of characters that is NOT in range. Any character that is not from the range 0 to 9, is not comma (double format) and is not minus sign (negative value) will be replaced with an empty string, which can also be understood as being removed from the string. At the end we convert the whole column to integer format with toInt() function.</p>
2	Rule-based Row Filter  Node 5	<p>Expression (INCLUDE TRUE MATCHES):</p> <pre>\$Age\$ > 0 AND \$Age\$ <= 120 => TRUE</pre> <p>We identify values in range (0;120] as TRUE using AND logical gate and remove any value outside of that range, which is considered as FALSE (e.g: -500).</p>

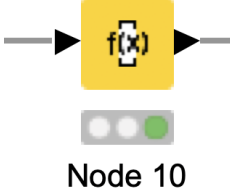
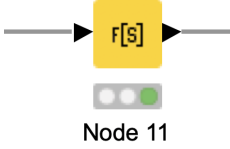
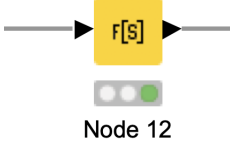
4. Task 4

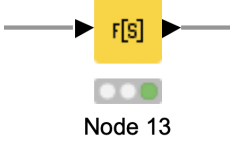
Sequence	Node	Configuration
1	String Manipulation  Node 6	<p>Expression</p> <pre>toDouble(regexReplace(\$Annual_Income\$, "[^0-9.-]", ""))</pre> <p>We use regular expressions to form a search pattern for non-numerical data. Pattern brackets with caret are used to find a range of characters that is NOT in range. Any character that is not from the range 0 to 9, is not comma (double format) and is not</p>

		minus sign (negative value) will be replaced with an empty string, which can also be understood as being removed from the string. At the end we convert the whole column to double format with toDouble() function.
--	--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

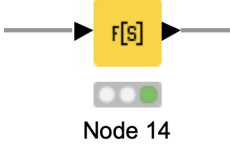
5. Task 5

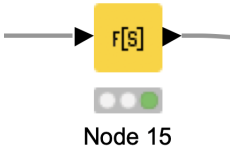
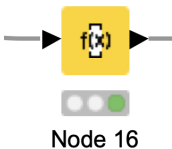
Sequence	Node	Configuration
1	String Manipulation  Node 7	<p>Expression:</p> <pre>replace(\$Occupation\$, "_____" , null)</pre> <p>This expression replaces all occurrences of "_____" into Null (literal value).</p>
2	String Manipulation  Node 8	<p>Expression:</p> <pre>toInt(regexReplace(\$Num_of_Loan\$, "[^0-9.-]", ""))</pre> <p>We use regular expressions to form a search pattern for non-numerical data. Pattern brackets with caret are used to find a range of characters that is NOT in range. Any character that is not from the range 0 to 9, is not comma (double format) and is not minus sign (negative value) will be replaced with an empty string, which can also be understood as being removed from the string. At the end we convert the whole column to integer format with toInt() function.</p>
3	Math Formula (Multi Column)  Node 9	<p>Expression:</p> <pre>abs(\$\$CURRENT_COLUMN\$\$)</pre> <p>Math Formula (Multi Column) executes the same mathematical expression based on the values in a row for a set of selected columns. It manipulates multiple columns with the same expression, allowing us to use only a single node that would otherwise require multiple nodes. abs() function is used to return the absolute value of</p>

		selected columns: "Num_Bank_Accounts" (Number (integer)) and "Num_Credit_Card" (Number (integer)).
4	Math Formula  Node 10	<p>Expression:</p> <pre>if(\$Num_of_Loan\$ < 0, 0, \$Num_of_Loan\$)</pre> <p>This expression manipulates the column based on mathematical conditions. If "Num_of_Loan" is lower than 0, it will be identified as TRUE, and the value will be changed to 0 which is the true interval. Vice versa, the false interval is "Num_of_loan", meaning if "Num_of_Loan" is greater than or equal to 0, the original value remains unmodified.</p>
5	String Manipulation  Node 11	<p>Expression:</p> <pre>toInt(regexReplace(\$Num_of_Delayed_Payment\$, "[^0-9.-]", ""))</pre> <p>We use regular expressions to form a search pattern for non-numerical data. Pattern brackets with caret are used to find a range of characters that is NOT in range. Any character that is not from the range 0 to 9, is not comma (double format) and is not minus sign (negative value) will be replaced with an empty string, which can also be understood as being removed from the string. At the end we convert the whole column to integer format with toInt() function.</p>
6	String Manipulation  Node 12	<p>Expression:</p> <pre>replace(\$Credit_Mix\$, "_", "Unknown")</pre> <p>This expression replaces all occurrences of "_" into "Unknown".</p>

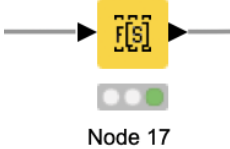
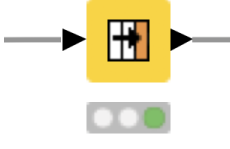
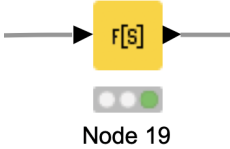
7	String Manipulation  Node 13	<p>Expression:</p> <pre>toDouble(regexReplace(\$Outstanding_Debt\$, "[^0-9.-]", ""))</pre> <p>We use regular expressions to form a search pattern for non-numerical data. Pattern brackets with caret are used to find a range of characters that is NOT in range. Any character that is not from the range 0 to 9, is not comma (double format) and is not minus sign (negative value) will be replaced with an empty string, which can also be understood as being removed from the string. At the end we convert the whole column to double format with toDouble() function.</p>
---	----------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

6. Task 6

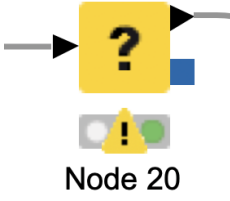
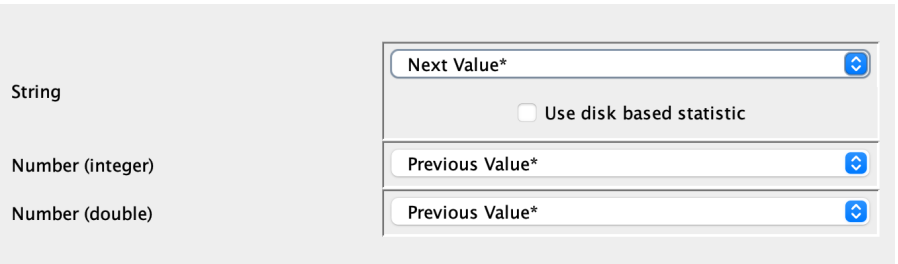
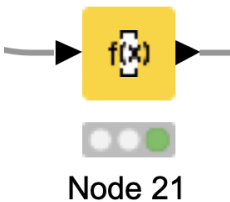
Sequence	Node	Configuration
1	String Manipulation  Node 14	<p>Expression:</p> <pre>toInt(strip(substr(\$Credit_History_Age\$, 0, 2)))</pre> <p>3 functions are used to extract the number of years into integer format. Firstly, substr() substrings starting from the first character (index 0) and outputs 2 characters. E.g: "22 Years and 1 Months" returns "22" after subtraction. Secondly, a strip function is used to remove any remaining space (E.g: "2 Years and 1 Months" returns "2 " after subtraction). Lastly we convert the result to integer format for later calculation and we will store the output by appending a new column with the name "Years".</p>

2	<p>String Manipulation</p>  <p>Node 15</p>	<p>Expression:</p> <pre>toInt(strip(substr(\$Credit_History_Age\$, indexOf(\$Credit_History_Age\$, "d") + 1, 3)))</pre> <p>The idea for this expression is essentially the same as the previous node, but there is a slight difference. The starting index is the index of “d” plus one equal to the index of the space after the “d” letter which only appears once in the string, and outputs 3 characters starting from the space after “d” letter (E.g: 22 Years and 1 Months” returns “ 1 ” after subtraction). Similar to the previous node, we strip out any space in the string and convert the column into integer format and store the output by appending a new column with the name “Months”.</p>
3	<p>Math Formula</p>  <p>Node 16</p>	<p>Expression:</p> <pre>\$Years\$ * 12 + \$Months\$</pre> <p>This expression is simply an equation to calculate the number of months using values from temporarily created columns (“Years” and “Months”), and the output will be stored in a newly appended “Total_CHA” column as integer format.</p>

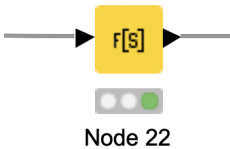
7. Task 7


Sequence	Node	Configuration
1	String Manipulation (Multi Column)  Node 17	<p>Expression:</p> <pre>toDouble(regexReplace(\$\$CURRENTCOLUMN\$\$,"[^0-9.-]",""))</pre> <p>We use “String Manipulation (Multi Column)” to apply the same expression to multiple columns, which is to remove non-numeric characters using regular expression and convert selected columns into integer format.</p>
2	String Replacer  Node 18	<p>We use the “String Replacer” node in order to replace any string in the “Payment_Behaviour” column that contains a wildcard pattern “!@” with “Unknown”. Asterisk (*) symbol is used to identify if the pattern matches an arbitrary number of characters.</p> <div> <p>Target column Payment_Behaviour</p> <p>Pattern type <input checked="" type="radio"/> Wildcard pattern <input type="radio"/> Regular expression</p> <p>Pattern !@*</p> <p>Replacement text Unknown</p> <p>Replace ... <input checked="" type="radio"/> ... whole string</p> </div>
3	String Manipulation  Node 19	<p>Expression:</p> <pre>toDouble(\$Changed_Credit_Limit\$)</pre> <p>This expression converts the column into double format.</p>

8. Task 8


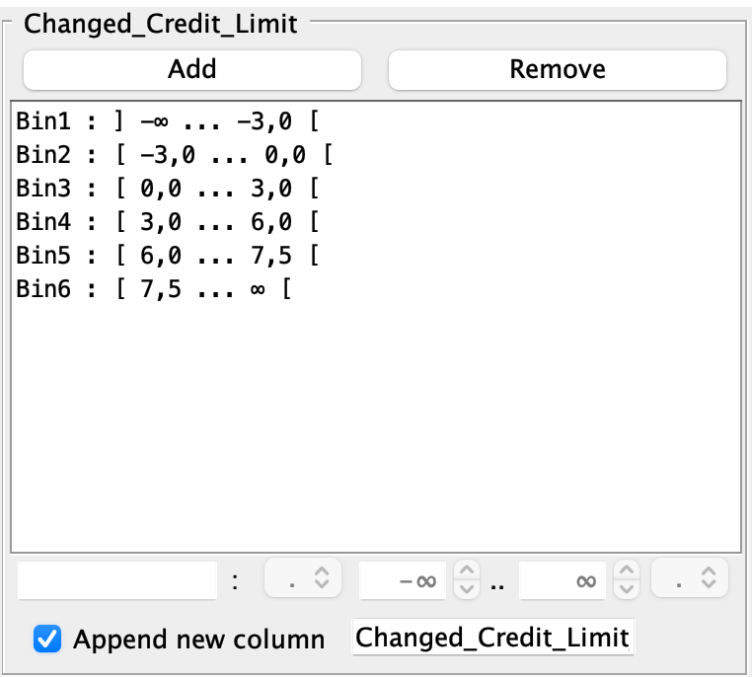
Sequence	Node	Configuration
1	Missing Value  Node 20	<p>This node modifies every missing value based on the data type. The following picture is the configuration of the node.</p> 
2	Math Formula  Node 21	<p>Expression:</p> <pre>if(\$Monthly_Balance\$ < 0, 0, \$Monthly_Balance\$)</pre> <p>This expression manipulates the column based on mathematical conditions. If “Monthly_Balance” is lower than 0, it will be identified as TRUE, and the value will be changed to 0 which is the true interval. Vice versa, the false interval is “Monthly_Balance”, meaning if “Monthly_Balance” is greater than or equal to 0, the original value remains unmodified.</p>

9. Task 9

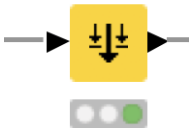
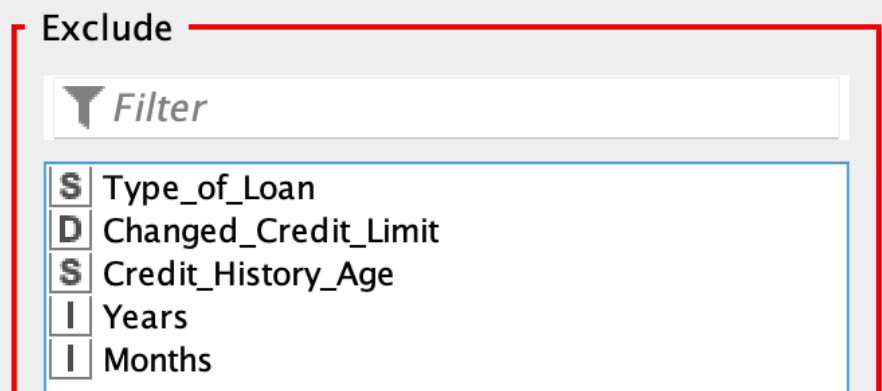
Sequence	Node	Configuration
1	String Manipulation  Node 22	<p>Expression:</p> <pre>substr(\$Type_of_Loan\$, 0, indexOfChars(\$Type_of_Loan\$, ","))</pre> <p>To simplify the data in “Type_of_Loan”, this expression substrings from the first index of the string to the index of the first comma in</p>

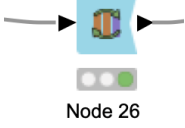
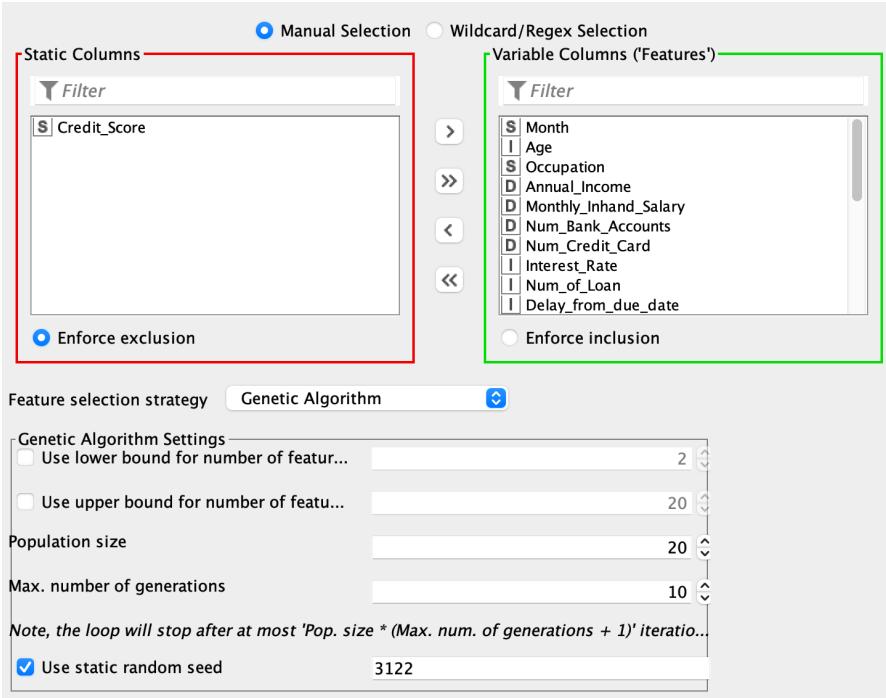
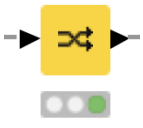
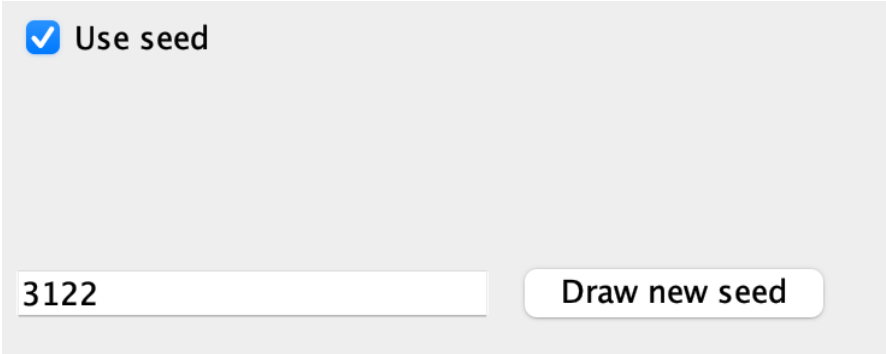
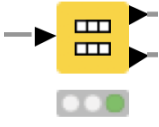
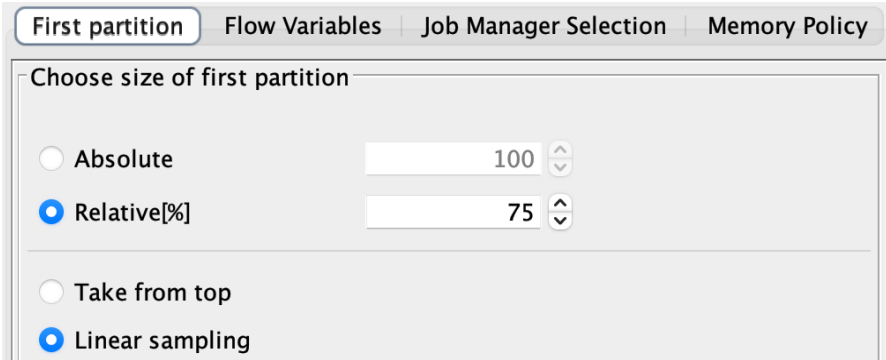
		<p>the string. The node outputs two types of output base on different string pattern:</p> <ul style="list-style-type: none"> - "Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan" will be "Auto Loan" after subtraction - "Auto Loan" will be "" (empty string) after subtraction as there is no comma in the string, resulting in 0 as end index so the node does not return any value <p>Output will be stored in a newly appended "Type_of_Loan_new" column.</p>
2	<p>Rule Engine</p>  <p>Node 23</p>	<p>Expression:</p> <pre>\$Type_of_Loan_new\$ LIKE "" => \$Type_of_Loan\$ TRUE => \$Type_of_Loan_new\$</pre> <p>If the occurrence is an empty string, it will be replaced with the original value with respectively the same row ID in the "Type_of_Loan" column. If there is a string, the occurrence is considered as TRUE and remains the default value.</p>

10. Task 10

Sequence	Node	Configuration
1	Numeric Binner  Node 24	<p>We use a “Numeric Binner” node to bin a number of intervals (bins). Each bin gets a unique name and a defined range. By choosing the desired bracket, the left and right values of each bin can be included or excluded from the bin: "[" or "]".</p> 

11. Task 11

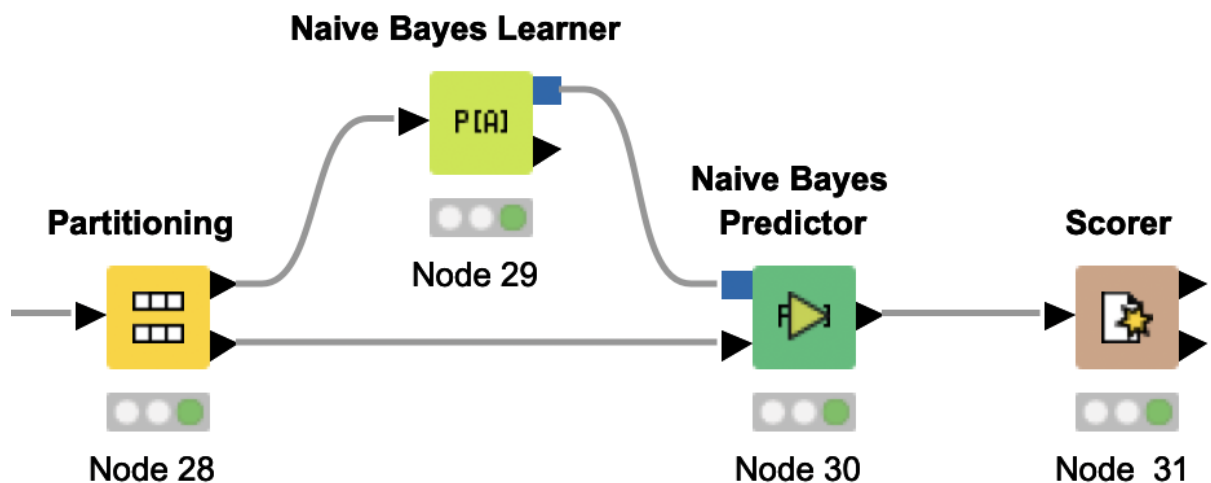
Sequence	Node	Configuration
1	Column Filter  Node 25	

2	<p>Feature Selection Loop Start (1:1)</p>  <p>Node 26</p>	<p>Total selected attributes (Excluding “Credit_Score” class label): 12 attributes.</p> 
3	<p>Shuffle</p>  <p>Node 27</p>	
4	<p>Partitioning</p>  <p>Node 28</p>	

		Output: - First partition (Training set): 68196 tuples - Second partition (Test set): 22732 tuples
--	--	----------------------------------------------------------------------------------------------------------

IV. Naïve Bayes Model

1. Screenshot of Naïve Bayes model in KNIME



2. Naïve Bayes Learner Configuration

Classification Column:

Default probability:

Minimum standard deviation

Threshold standard deviation

Maximum number of unique nominal values per attribute:

☐ Ignore missing values
☐ Create PMML 4.2 compatible model

3. Naïve Bayes Model Results

Confusion Matrix

Row ID	I Good	I Stand...	I Poor
Good	3376	620	107
Standard	3069	6651	2363
Poor	1034	1636	3876

Accuracy Statistics

Row ID	I TrueP...	I FalseP...	I TrueN...	I False...	D Recall	D Precisi...	D Sensiti...	D Specifi...	D F-me...	D Accur...	D Cohen...
Good	3376	4103	14526	727	0.823	0.451	0.823	0.78	0.583	?	?
Standard	6651	2256	8393	5432	0.55	0.747	0.55	0.788	0.634	?	?
Poor	3876	2470	13716	2670	0.592	0.611	0.592	0.847	0.601	?	?
Overall	?	?	?	?	?	?	?	?	?	0.612	0.404

Based on the result, we can clearly see that the Naïve Bayes model performs poorly as the precision score of the "Good" class is really low with the score of 0.451.

- The best metric to evaluate this would be the precision of the "Good" class, also known as positive predictive value (PPV). Out of all the positive predictions the model makes, precision is the percentage of correct positive predictions.

In this case, the bank wishes to reduce the risk associated with providing money to customers, which entails avoiding lending to borrowers who might default on their payments. As a result, it's critical for the bank to precisely identify clients who are "Good" and have a lower risk of default.

The bank can reduce the risk of default by concentrating on precision for the "Good" class and making sure that the customers who are identified as "Good" are indeed good borrowers.

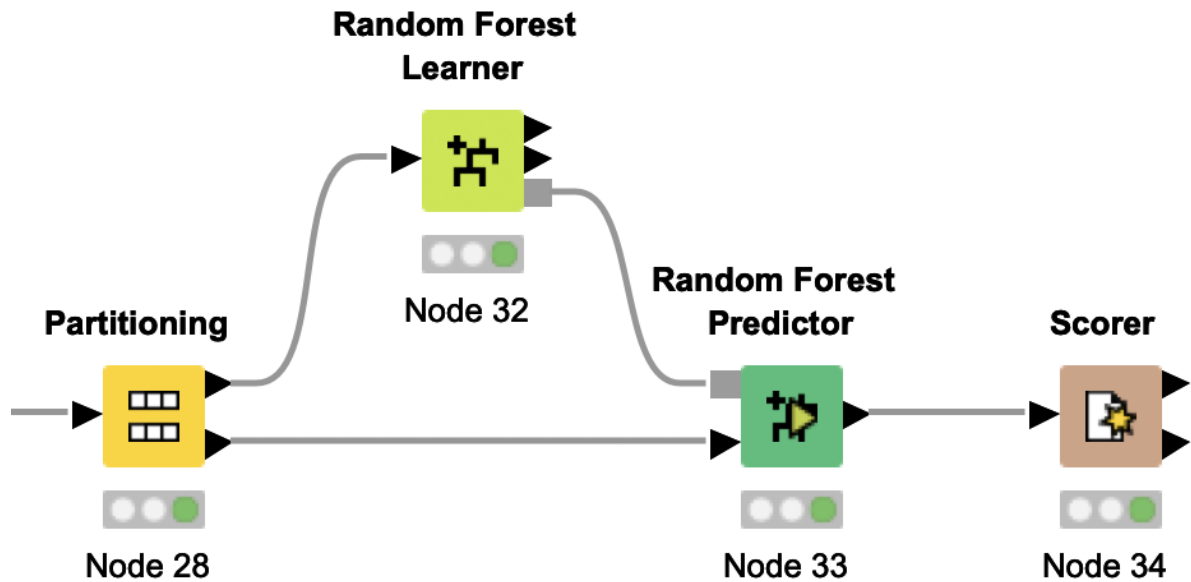
The formula for precision: $Precision (PPV) = \frac{TP}{TP + FP}$

Where TP (True Positives) are the number of "Good" customers who are correctly identified, and FP (False Positives) are the number of either "Standard" or "Poor" customers who are incorrectly identified as "Good".

Recall is also an alternative metric to take into account, but it might not be the best one to assess the bank's goal of reducing the risk of lending money to the "Good" class. This is due to the fact that recall does not account for the number of "Standard" or "Poor" credit score consumers who the model wrongly identifies as "Good".

V. Random Forest Model

1. Screenshot of Random Forest Model in KNIME



2. Random Forest Model Results

Confusion Matrix

Row ID	Good	Stand...	Poor
Good	2850	1171	82
Standard	1056	9476	1551
Poor	223	1562	4761

Accuracy statistics

Row ID	TrueP...	FalseP...	TrueN...	False...	Recall	Precisi...	Sensiti...	Specifi...	F-me...	Accur...	Cohen...
Good	2850	1279	17350	1253	0.695	0.69	0.695	0.931	0.692	?	?
Standard	9476	2733	7916	2607	0.784	0.776	0.784	0.743	0.78	?	?
Poor	4761	1633	14553	1785	0.727	0.745	0.727	0.899	0.736	?	?
Overall	?	?	?	?	?	?	?	?	?	0.752	0.587

3. Comparison between Naïve Bayes model and Random Forest model

Naïve Bayes model vs. Random Forest model

	Naïve Bayes model	Random Forest model
Accuracy	0.612	0.752
Precision (Good)	0.451	0.69

By comparing the precision score of the “Good” class, we can clearly see that the Random Forest model presents more suitable results to the given context of the bank than the Naïve Bayes model due to the fact that Random Forest tends to perform better than Naive Bayes when dealing with complex relationships between features and when there are a large number of features in the dataset in general. As mentioned before, we use precision of the “Good” class to compare two models for this specific case as the bank can reduce the risk of default by concentrating on precision for the "Good" class and making sure that the customers who are identified as "Good" are indeed good borrowers.

4. We can look at a variety of metrics, such as precision, recall, F1 score, and accuracy, to determine which class the random forest model performs the best on. The precision, recall, and F1 score can be calculated using the confusion matrix, which shows the number of true positives, false positives, true negatives, and false negatives for each class. Precision measures how often the model correctly predicts a specific class out of all predictions made for that class. Recall measures how often the model correctly predicts a specific class out of all instances of that class in the dataset. The F1 score is a combination of precision and recall and provides a measure of the overall effectiveness of the model.

Random Forest Model Score

	Recall	Precision	F-measure
Good	0.695	0.69	0.692
Standard	0.784	0.776	0.78
Poor	0.727	0.745	0.736

In the context of minimizing the risk of lending money to customers, the most suitable metric to evaluate the model would be precision, because the cost of false positives is

higher than the cost of false negatives. The choice of metric depends on the specific objectives associated with false positives and false negatives. If the bank wants to minimize the risk of lending money to customers, then "Good" in "Credit_Score" should be the major target. Therefore, we would want to evaluate the precision for each class to determine which class the model performs the best on, as in this case, precision is more important than recall because the cost of false positives (e.g: Lending money to customer whose credit is bad) is higher than the cost of false negatives (e.g: Miss the chance to lend money to customer who would have paid back the loan). Therefore, precision is a more suitable metric for class evaluation. We can clearly see that "Standard" is the class in which Random Forest model performs predictions the best with not only highest precision score but also with highest recall score and F-measure score.