Cover sheet for submission of work for assessment

SWINBURNE UNIVERSITY OF TECHNOLOGY

UNIT DETAILS

Unit name	Data Science Principles			Class day/time	Wed, 8 – 12am	Office use only
Unit code	COS10022	Assignment no.	02	Due date	26/03/2023	
Name of lecturer/teacher	Dr. Pham Thi Kim Dung					
Tutor/marker's name	Dr. Pham Thi Kim Dung					Faculty or school date stamp

STUDENT(S)

Family Name	Given Name	Student ID Number
Hau	Linh Chi	104177160

DECLARATION AND STATEMENT OF AUTHORSHIP

1. For the sake of this evaluation, I have not impersonated anyone or let anyone else to impersonate me.
2. This evaluation is all original work from myself, with the exception of the places where proper credit has been given.
3. Except where such collaboration has been approved by the lecturer or instructor in question, no portion of this evaluation has been prepared for me by anyone else.
4. I have not previously submitted this work for this or any other course/unit.
5. I accept that my assessment response may be duplicated, shared, compared, stored, and used for benchmarking, plagiarism detection, or educational purposes.

I understand that:

6. Plagiarism is known as the act of presenting another person's work, idea, or creativity as your own. It is a sort of academic fraud that might get you kicked out of the university and is considered cheating. Written, graphic, and visual works, computer data, oral presentations, and other forms of presentation can all be used to source and deliver plagiarized content. When the source of the copied material is not properly cited, plagiarism occurs.

Student signature/s

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

Further information relating to the penalties for plagiarism, which range from a formal caution to expulsion from the University is contained on the Current Students website at **www.swin.edu.au/student/**

Copies of this form can be downloaded from the Student Forms web page at **www.swinburne.edu.au/studentforms/**

COS10022 – Data Science Principles – Assignment 1

DATA CLEANING AND ANALYTICS

I. ASSIGNMENT SUMMARY

This assignment provides an overview of data cleaning and prediction model construction, including the key concepts, procedures, and tools involved.
The focus is on:

- Selecting appropriate features and models while gaining some knowledge about prebuilt tools and applying them in a data science project.
- Preparing the dataset using KNIME analytical platform for cleaning and two classification models, Naïve Bayes and Random Forest, are developed.
- Selecting relevant attributes, cleaning the dataset, partitioning the data into training and test sets, developing an effective prediction model, and explaining the results.

The goal of the assignment is to gain hands-on experience with data cleaning and model construction in a real-world setting.

II. INTRODUCTION

The focus of this report is a dataset that was gathered from the real world, consisting of 100,000 tuples that are categorised into three different financial credit score classes. The original data includes 24 attributes in total. The goals of this assignment are: firstly, to carry out the necessary data cleaning and preparation for future use, and secondly, to develop two predictive models that can be used to project the "Credit_Score" classification.
My works are described in detail in this report and include:

- Preparing the raw data for future use by cleaning and organising it.
- Constructing two models (Naïve Bayes and Random Forest classifier models) to forecast the value and categorise financial credit scores.
- Choosing and implementing suitable features and models for this data project.
- Identifying relevant attributes, splitting the dataset into training and testing sets, developing predictive models, and explaining the results.

III. DATA CLEANING

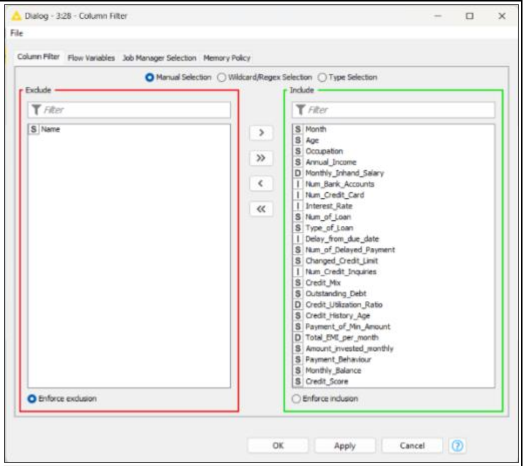
Question 1.1

Node	Configuration

Column Filter

The excluded attribute is *"Name"*. The reason for removing it are:

- Credit scores are determined by a variety of factors such as payment history, credit utilisation, length of credit history, types of credit used, new credit inquiries, etc. Therefore, a person's name does not affect their credit score.
- In the Naïve Bayes learner process, the maximum number of unique nominal values per attribute is set to 600 to limited by the amount of available data, the complexity of the problem, and the performance requirements of the application. However, the number of names in the dataset exceeds the limit number (10137 different names). So, the *"Name"* attribute is filtered out.



Dialog - 3:28 - Column Filter

File

Column Filter | Flow Variables | Job Manager Selection | Memory Policy

● Manual Selection ○ Wildcard/Regex Selection ○ Type Selection

Exclude

Filter

S Name

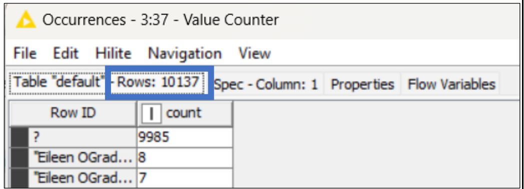
● Enforce exclusion

Include

Filter

S Month
S Age
S Occupation
S Annual_Income
D Monthly_Inhand_Salary
I Num_Bank_Accounts
I Num_Credit_Card
I Interest_Rate
S Num_of_Loan
S Type_of_Loan
I Delay_from_due_date
S Num_of_Delayed_Payment
S Changed_Credit_Limit
I Num_Credit_Inquiries
S Credit_Mix
S Outstanding_Debt
D Credit_Utilization_Ratio
S Credit_History_Age
S Payment_of_Min_Amount
D Total_EMI_per_month
S Amount_invested_monthly
S Payment_Behaviour
S Monthly_Balance
S Credit_Score

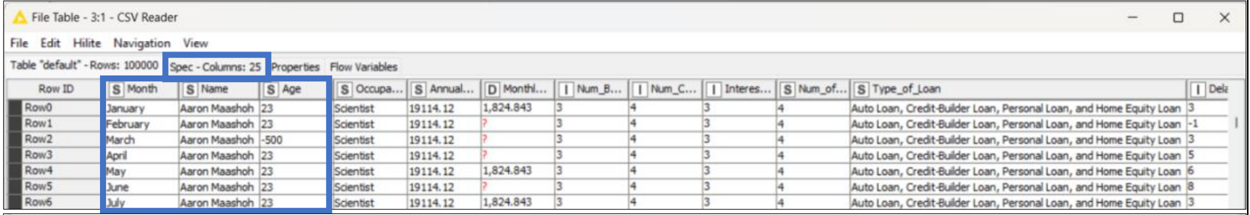
○ Enforce inclusion



Occurrences - 3:37 - Value Counter

Table "default" - Rows: 10137 | Spec - Column: 1 | Properties | Flow Variables

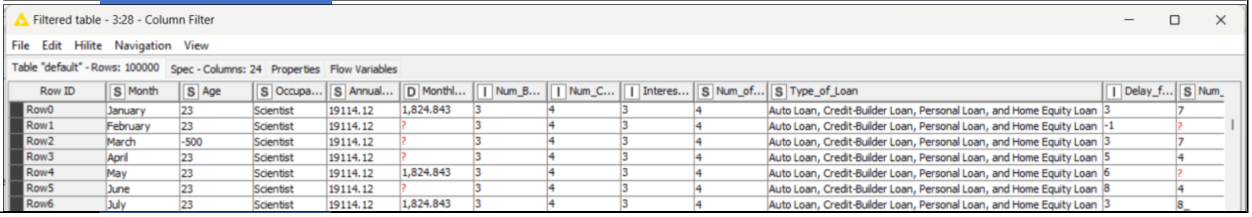
Row ID	I count
?	9985
"Eileen OGrad…	8
"Eileen OGrad…	7



File Table - 3:1 - CSV Reader

Table "default" - Rows: 100000 | Spec - Columns: 25 | Properties | Flow Variables

Row ID	S Month	S Name	S Age	S Occupa…	S Annual…	D Monthl…	I Num_B…	I Num_C…	I Interes…	S Num_of…	S Type_of_Loan	I Dela
Row0	January	Aaron Maashoh	23	Scientist	19114.12	1,824.843	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan	3
Row1	February	Aaron Maashoh	23	Scientist	19114.12	?	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan	-1
Row2	March	Aaron Maashoh	-500	Scientist	19114.12	?	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan	3
Row3	April	Aaron Maashoh	23	Scientist	19114.12	?	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan	5
Row4	May	Aaron Maashoh	23	Scientist	19114.12	1,824.843	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan	6
Row5	June	Aaron Maashoh	23	Scientist	19114.12	?	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan	8
Row6	July	Aaron Maashoh	23	Scientist	19114.12	1,824.843	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan	3

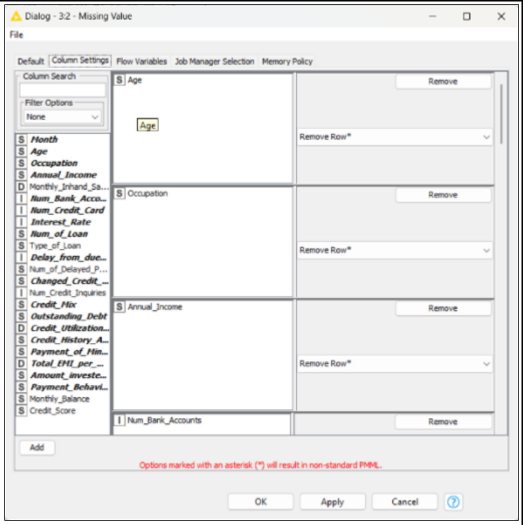


Filtered table - 3:28 - Column Filter

Table "default" - Rows: 100000 | Spec - Columns: 24 | Properties | Flow Variables

Row ID	S Month	S Age	S Occupa…	S Annual…	D Monthl…	I Num_B…	I Num_C…	I Interes…	S Num_of…	S Type_of_Loan	I Delay_f…	S Num_
Row0	January	23	Scientist	19114.12	1,824.843	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan	3	7
Row1	February	23	Scientist	19114.12	?	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan	-1	?
Row2	March	-500	Scientist	19114.12	?	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan	3	7
Row3	April	23	Scientist	19114.12	?	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan	5	4
Row4	May	23	Scientist	19114.12	1,824.843	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan	6	?
Row5	June	23	Scientist	19114.12	?	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan	8	4
Row6	July	23	Scientist	19114.12	1,824.843	3	4	3	4	Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan	3	8_

Question 1.2

S	Node	Configuration
1	Missing Value	This node is used to remove tuples with missing values from these 18 attributes *"Month", "Age", "Occupation", "Annual_Income", "Num_Bank_Accounts", "Num_Credit_Card", "Interest_Rate", "Num_of_Loan", "Delay_from_due_date", "Changed_Credit_Limit", "Credit_Mix", "Outstanding_debt", "Credit_Utilization_Ratio", "Credit_History_Age", "Payment_of_Min_Amount", "Total_EMI_per_month", "Amount_invested_monthly" and "Payment_Behaviour".* One example output of the *"Amount_invested_monthly" column*: 

Dialog - 3:2 - Missing Value

File

Default | Column Settings | Flow Variables | Job Manager Selection | Memory Policy

Column Search

Filter Options: None

S Month
S Age
S Occupation
S Annual_Income
D Monthly_Inhand_Sa…
I Num_Bank_Acco…
I Num_Credit_Card
I Interest_Rate
S Num_of_Loan
S Type_of_Loan
I Delay_from_due…
S Num_of_Delayed_P…
S Changed_Credit_…
I Num_Credit_Inquiries
S Credit_Mix
S Outstanding_Debt
D Credit_Utilization…
S Credit_History_A…
S Payment_of_Min…
D Total_EMI_per_…
S Amount_investe…
S Payment_Behavi…
S Monthly_Balance
S Credit_Score

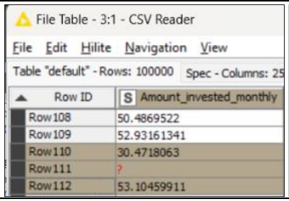
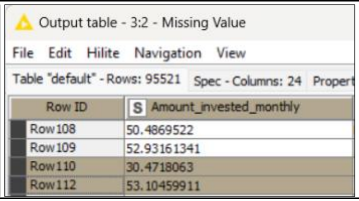
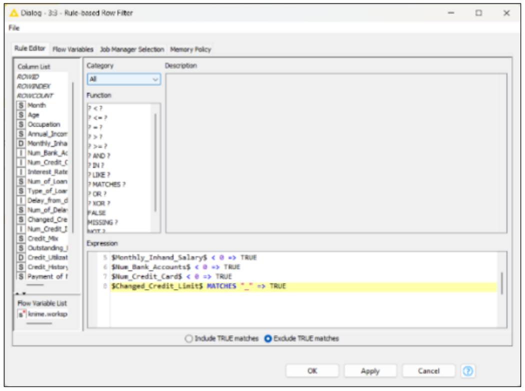
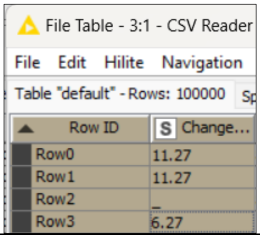
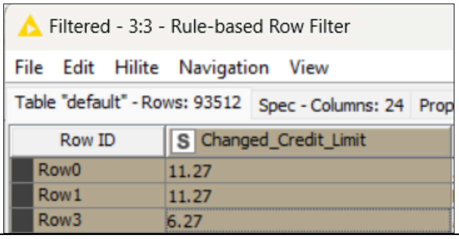
S Age: Remove Row*

S Occupation: Remove Row*

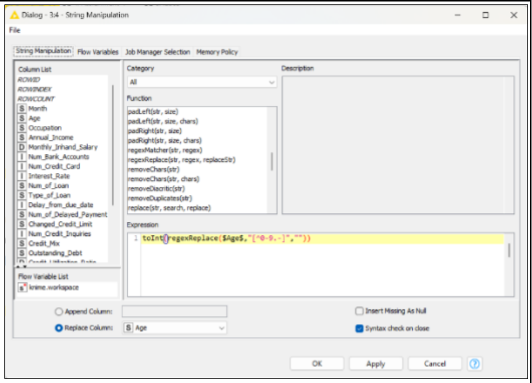
S Annual_Income: Remove Row*

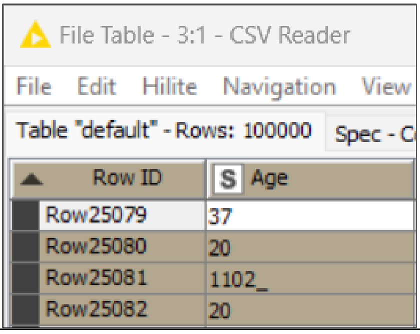
I Num_Bank_Accounts

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

		 File Table - 3:1 - CSV Reader Table "default" - Rows: 100000 Spec - Columns: 25 Row ID · S Amount_invested_monthly Row108 · 50.4869522 Row109 · 52.93161341 Row110 · 30.4718063 Row111 · ? Row112 · 53.10459911  Output table - 3:2 - Missing Value Table "default" - Rows: 95521 Spec - Columns: 24 Propert Row ID · S Amount_invested_monthly Row108 · 50.4869522 Row109 · 52.93161341 Row110 · 30.4718063 Row112 · 53.10459911
2	**Rule-based Row Filter**	Some tuples containing infeasible values, such as: • *"Monthly_Inhand_Salary"* < 0 • *"Num_Bank_Accounts"* < 0 • *"Num_Credit_Card"* < 0 • *"Changed_Credit_Limit"* contains "_" are removed by using this node with the commands (exclude TRUE matches) as follows: $Monthly_Inhand_Salary$ < 0 => TRUE $Num_Bank_Accounts$ < 0 => TRUE Num_Credit_Card < 0 => TRUE $Changed_Credit_Limit$ MATCHES "_" => TRUE  Dialog - 3:3 - Rule-based Row Filter Expression: $Monthly_Inhand_Salary$ < 0 => TRUE $Num_Bank_Accounts$ < 0 => TRUE Num_Credit_Card < 0 => TRUE $Changed_Credit_Limit$ MATCHES "_" => TRUE ○ Include TRUE matches ● Exclude TRUE matches Example output of the "C*hanged_Credit_Limit" column:*  File Table - 3:1 - CSV Reader Table "default" - Rows: 100000 Row ID · S Change... Row0 · 11.27 Row1 · 11.27 Row2 · _ Row3 · 6.27  Filtered - 3:3 - Rule-based Row Filter Table "default" - Rows: 93512 Spec - Columns: 24 Prop Row ID · S Changed_Credit_Limit Row0 · 11.27 Row1 · 11.27 Row3 · 6.27

Question 1.3

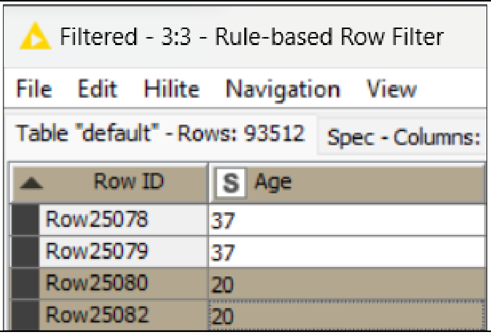
S	Node	Configuration
1	**String Manipulation**	Symbols that are not numbers are eliminated from the *"Age"* attribute and the data of this attribute is converted into the usual number format (to integer) and replace the new output to the original *"Age"* column. The "." and "-" are kept because they represent decimal number and negative number (some of them must be remove, so erasing the "-" can lead to wrong data cleaning, respectively. Expression: toInt(regexReplace(Age,"[^0-9.-]",""))  Dialog - 3:4 - String Manipulation Expression: toInt(regexReplace(Age,"[^0-9.-]","")) ○ Append Column: ● Replace Column: S Age □ Insert Missing As Null ☑ Syntax check on close Example output:

<table>
<tr>
<td></td>
<td></td>
<td>


File Table - 3:1 - CSV Reader

Table "default" - Rows: 100000 Spec - C

Row ID	S Age
Row25079	37
Row25080	20
Row25081	1102_
Row25082	20

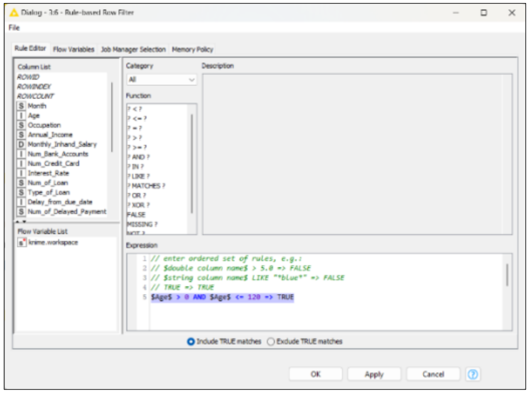


Filtered - 3:3 - Rule-based Row Filter

Table "default" - Rows: 93512 Spec - Columns:

Row ID	S Age
Row25078	37
Row25079	37
Row25080	20
Row25082	20
</td>
</tr>
<tr>
<td>2</td>
<td>Rule-based Row Filter</td>
<td>
The tuples having *"Age"* value lower than or equal to 0 or greater than 120 are dropped using the following command (include TRUE matches):

Age > 0 AND Age <= 120 => TRUE

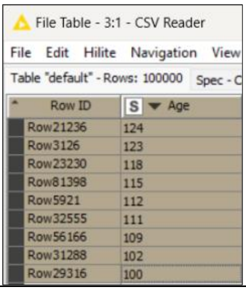


Dialog - 3:6 - Rule-based Row Filter

```
// enter ordered set of rules, e.g.:
// $double column name$ > 5.0 => FALSE
// $string column name$ LIKE "*blue*" => FALSE
// TRUE => TRUE
$Age$ > 0 AND $Age$ <= 120 => TRUE
```

● Include TRUE matches ○ Exclude TRUE matches

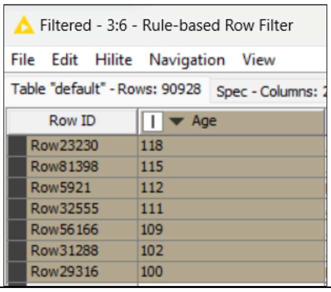
Example output:



File Table - 3:1 - CSV Reader

Table "default" - Rows: 100000 Spec - C

Row ID	S Age
Row21236	124
Row3126	123
Row23230	118
Row81398	115
Row5921	112
Row32555	111
Row56166	109
Row31288	102
Row29316	100



Filtered - 3:6 - Rule-based Row Filter

Table "default" - Rows: 90928 Spec - Columns: 2

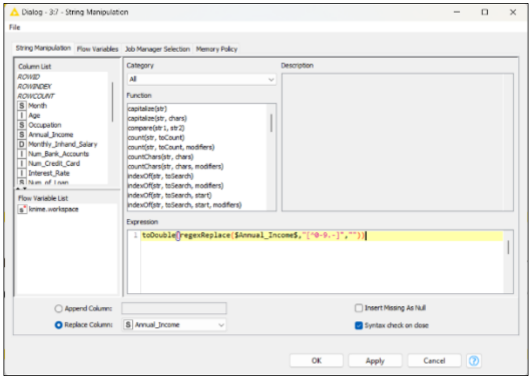
Row ID	I Age
Row23230	118
Row81398	115
Row5921	112
Row32555	111
Row56166	109
Row31288	102
Row29316	100
</td>
</tr>
</table>

Question 1.4

<table>
<tr>
<th>Node</th>
<th>Configuration</th>
</tr>
<tr>
<td>String Manipulation</td>
<td>
Non-numerical symbols in the *"Annual_Income"* column are eliminated and the remaining data are converted to the double format. The "." and "-" are kept because they represent decimal number and negative number, respectively. The new output is replaced to the original *"Annual_Income"* column.

Expression:

toDouble(regexReplace($Annual_Income$,"[^0-9.-]",""))

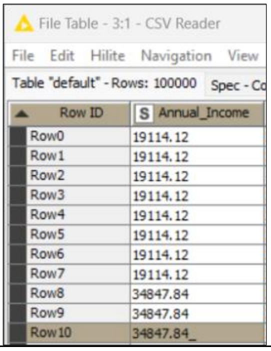


Dialog - 3:7 - String Manipulation

```
toDouble(regexReplace($Annual_Income$,"[^0-9.-]",""))
```

○ Append Column:
● Replace Column: S Annual_Income
☐ Insert Missing As Null
☑ Syntax check on close

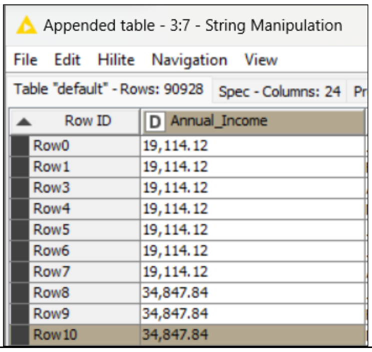
Example output:



File Table - 3:1 - CSV Reader

Table "default" - Rows: 100000 Spec - Co

Row ID	S Annual_Income
Row0	19114.12
Row1	19114.12
Row2	19114.12
Row3	19114.12
Row4	19114.12
Row5	19114.12
Row6	19114.12
Row7	19114.12
Row8	34847.84
Row9	34847.84
Row10	34847.84_

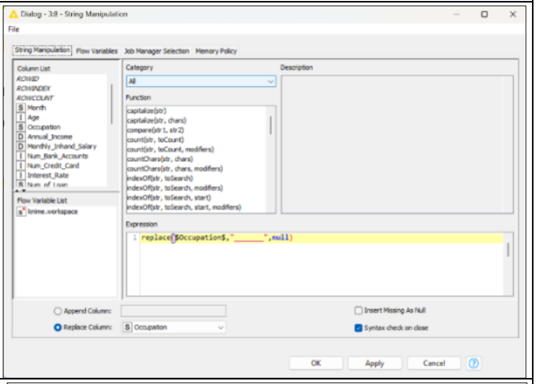
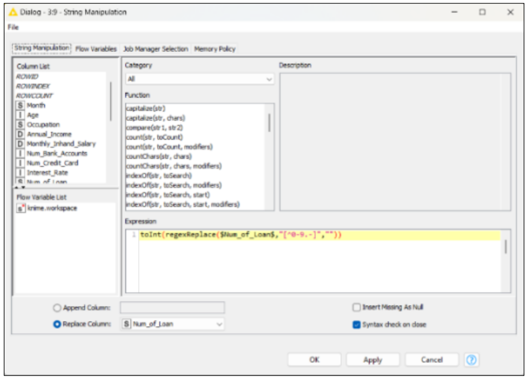
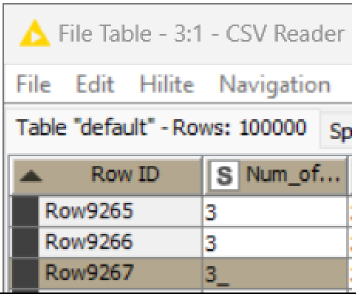
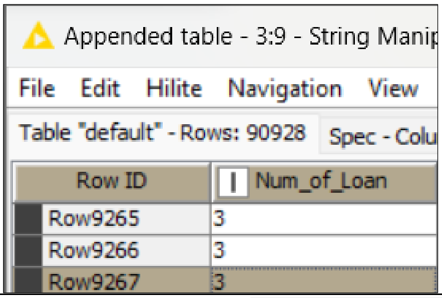
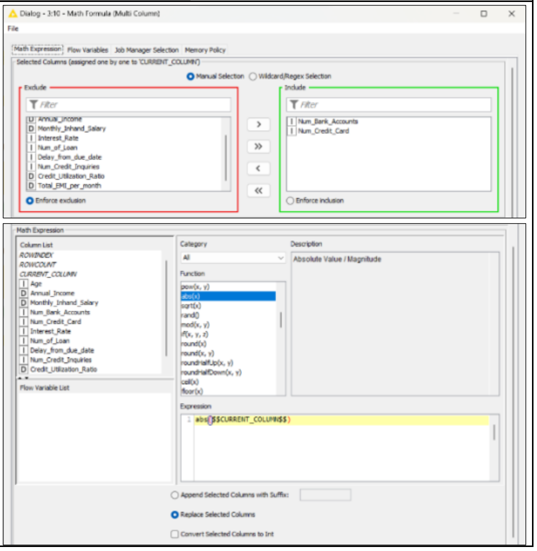


Appended table - 3:7 - String Manipulation

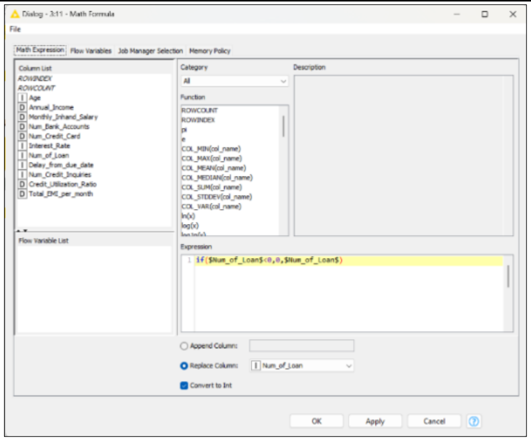
Table "default" - Rows: 90928 Spec - Columns: 24 Pr

Row ID	D Annual_Income
Row0	19,114.12
Row1	19,114.12
Row3	19,114.12
Row4	19,114.12
Row5	19,114.12
Row6	19,114.12
Row7	19,114.12
Row8	34,847.84
Row9	34,847.84
Row10	34,847.84
</td>
</tr>
</table>

Question 1.5

S	Node	Configuration
1	String Manipulation r[s]	“______” values from the *“Occupation”* attribute are changed to null. The original *“Occupation”* column is replaced with the new data. Expression: replace($Occupation$,"______", null)  Dialog - 3:8 - String Manipulation Column List: ROWID, ROWINDEX, ROWCOUNT, S Month, I Age, S Occupation, D Annual_Income, D Monthly_Inhand_Salary, I Num_Bank_Accounts, I Num_Credit_Card, I Interest_Rate, S Num_of_Loan Flow Variable List: knime.workspace Category: All Expression: replace($Occupation$,"______",null) ○ Append Column: ● Replace Column: S Occupation □ Insert Missing As Null ☑ Syntax check on close File String Manipulation Flow Variables Job Manager Selection Memory Policy Function capitalize(str) capitalize(str, chars) compare(str1, str2) count(str, toCount) count(str, toCount, modifiers) countChars(str, chars) countChars(str, chars, modifiers) indexOf(str, toSearch) indexOf(str, toSearch, modifiers) indexOf(str, toSearch, start) indexOf(str, toSearch, start, modifiers) Description OK Apply Cancel
2	String Manipulation r[s]	Non-numerical symbols in *“Num_of_Loan”* are removed and the data is converted to integer data type. The “.” and “-” are kept because they represent decimal number and negative number, respectively. The new output is replaced to the original *“Num_of_Loan”* column. Expression: toInt(regexReplace(Num_of_Loan,"[^0-9.-]",""))  Dialog - 3:9 - String Manipulation Column List: ROWID, ROWINDEX, ROWCOUNT, S Month, I Age, S Occupation, D Annual_Income, D Monthly_Inhand_Salary, I Num_Bank_Accounts, I Num_Credit_Card, I Interest_Rate, S Num_of_Loan Flow Variable List: knime.workspace Category: All Expression: toInt(regexReplace(Num_of_Loan,"[^0-9.-]","")) ○ Append Column: ● Replace Column: S Num_of_Loan □ Insert Missing As Null ☑ Syntax check on close File String Manipulation Flow Variables Job Manager Selection Memory Policy Function capitalize(str) capitalize(str, chars) compare(str1, str2) count(str, toCount) count(str, toCount, modifiers) countChars(str, chars) countChars(str, chars, modifiers) indexOf(str, toSearch) indexOf(str, toSearch, modifiers) indexOf(str, toSearch, start) indexOf(str, toSearch, start, modifiers) Description OK Apply Cancel Example output:  File Table - 3:1 - CSV Reader Table "default" - Rows: 100000 Sp Row ID | S Num_of... Row9265 | 3 Row9266 | 3 Row9267 | 3_  Appended table - 3:9 - String Manip Table "default" - Rows: 90928 Spec - Colu Row ID | I Num_of_Loan Row9265 | 3 Row9266 | 3 Row9267 | 3
3	Math Formula (Multi Column) f(x)	This node is utilised to take the absolute values of attributes *“Num_Bank_Accounts”* and *“Num_Credit_Card”* using the command as follows: abs($$CURRENT_COLUMN$$) The original *“Num_Bank_Accounts”* and *“Num_Credit_Card”* columns are replaced with the new data. (negative values of those two attributes are removed by the Rule-based Row Filter node in the question 2)  Dialog - 3:10 - Math Formula (Multi Column) Selected Columns (assigned one by one to 'CURRENT_COLUMN') ● Manual Selection ○ Wildcard/Regex Selection Exclude: D Annual_Income, D Monthly_Inhand_Salary, I Interest_Rate, I Num_of_Loan, I Delay_from_due_date, I Num_Credit_Inquiries, D Credit_Utilization_Ratio, D Total_EMI_per_month ● Enforce exclusion Include: I Num_Bank_Accounts, I Num_Credit_Card ○ Enforce inclusion Math Expression Column List: ROWINDEX, ROWCOUNT, CURRENT_COLUMN, I Age, D Annual_Income, D Monthly_Inhand_Salary, I Num_Bank_Accounts, I Num_Credit_Card, I Interest_Rate, I Num_of_Loan, I Delay_from_due_date, I Num_Credit_Inquiries, D Credit_Utilization_Ratio Category: All Function: abs(x) Description: Absolute Value / Magnitude Expression: abs($$CURRENT_COLUMN$$) ○ Append Selected Columns with Suffix: ● Replace Selected Columns □ Convert Selected Columns to Int File Math Expression Flow Variables Job Manager Selection Memory Policy Filter Flow Variable List pow(x, y) abs(x) sqrt(x) rand() mod(x, y) if(x, y, z) round(x) round(x, y) roundHalfUp(x, y) roundHalfDown(x, y) ceil(x) floor(x)

<table>
<tr>
<td>4</td>
<td>Math Formula
f(x)</td>
<td>
For the <i>“Num_of_Loan”</i> attribute, if the original values are negative, they are set to 0 and the new data is replaced to the original <i>“Num_of_Loan”</i> column.
Expression:
if($Num_of_Loan$<0,0, $Num_of_Loan$)



Category: All
Expression:
1 if($Num_of_Loan$<0,0,$Num_of_Loan$)
○ Append Column:
● Replace Column: I Num_of_Loan
☑ Convert to Int

Example output:

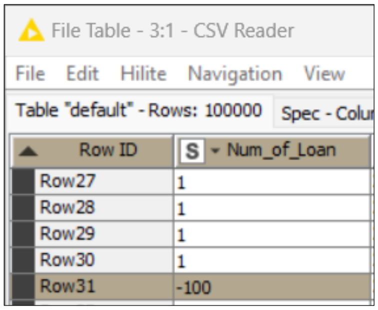


Table "default" - Rows: 100000 Spec - Colu
<table>
<tr><th>Row ID</th><th>S Num_of_Loan</th></tr>
<tr><td>Row27</td><td>1</td></tr>
<tr><td>Row28</td><td>1</td></tr>
<tr><td>Row29</td><td>1</td></tr>
<tr><td>Row30</td><td>1</td></tr>
<tr><td>Row31</td><td>-100</td></tr>
</table>

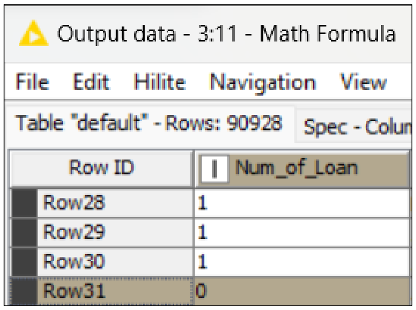
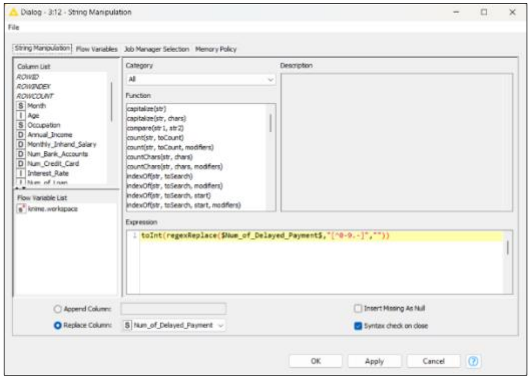


Table "default" - Rows: 90928 Spec - Colun
<table>
<tr><th>Row ID</th><th>I Num_of_Loan</th></tr>
<tr><td>Row28</td><td>1</td></tr>
<tr><td>Row29</td><td>1</td></tr>
<tr><td>Row30</td><td>1</td></tr>
<tr><td>Row31</td><td>0</td></tr>
</table>
</td>
</tr>
<tr>
<td>5</td>
<td>String Manipulation
r[s]</td>
<td>
Non-numerical symbols in <i>“Num_of_Delayed_payment”</i> are taken out and the data is converted to integer. The “.” and “-” are kept because they represent decimal number and negative number, respectively. The new output is replaced to the original <i>“Num_of_Delayed_payment”</i> column.
Expression:
toInt(regexReplace Num_of_Delayed_Payment$,"[^0-9.-]",""))



Category: All
Expression:
1 toInt(regexReplace($Num_of_Delayed_Payment$,"[^0-9.-]",""))
○ Append Column:
● Replace Column: S Num_of_Delayed_Payment
□ Insert Missing As Null
☑ Syntax check on close

Example output:

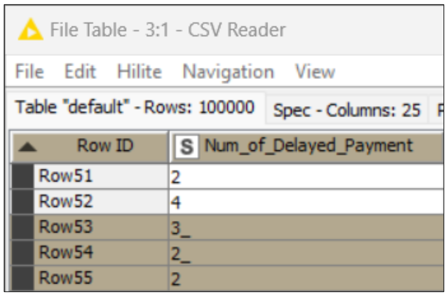


Table "default" - Rows: 100000 Spec - Columns: 25 P
<table>
<tr><th>Row ID</th><th>S Num_of_Delayed_Payment</th></tr>
<tr><td>Row51</td><td>2</td></tr>
<tr><td>Row52</td><td>4</td></tr>
<tr><td>Row53</td><td>3_</td></tr>
<tr><td>Row54</td><td>2_</td></tr>
<tr><td>Row55</td><td>2</td></tr>
</table>

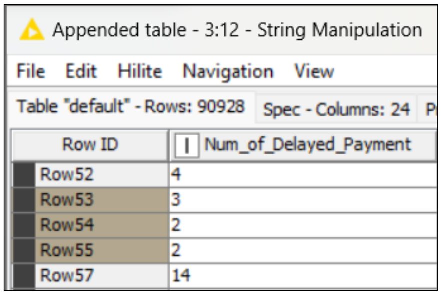
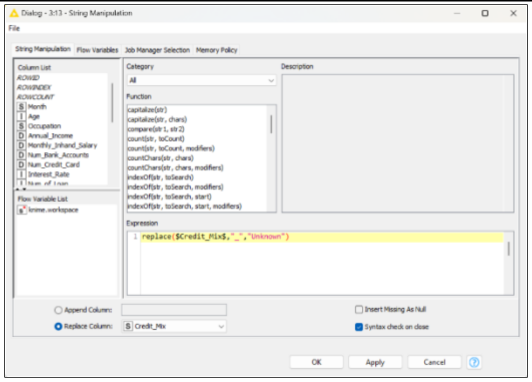
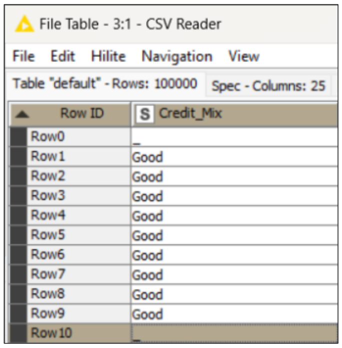
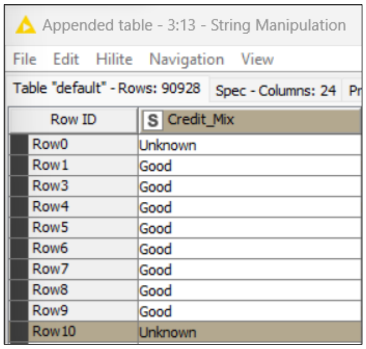
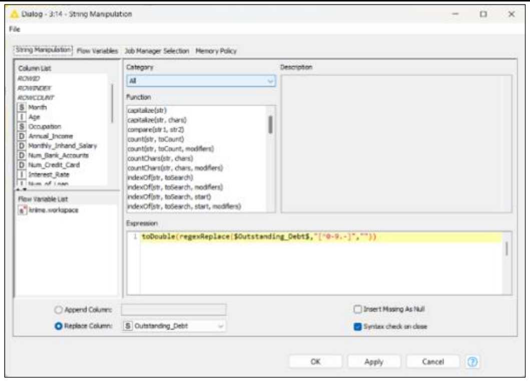


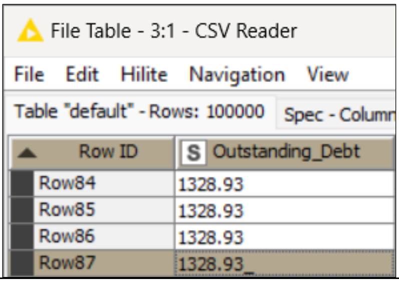
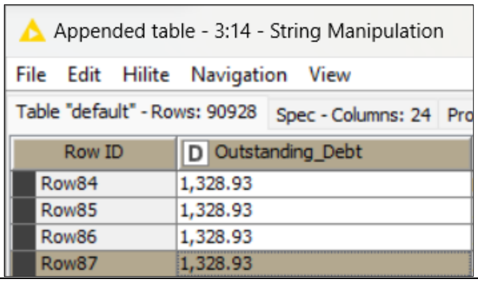
Table "default" - Rows: 90928 Spec - Columns: 24 P
<table>
<tr><th>Row ID</th><th>I Num_of_Delayed_Payment</th></tr>
<tr><td>Row52</td><td>4</td></tr>
<tr><td>Row53</td><td>3</td></tr>
<tr><td>Row54</td><td>2</td></tr>
<tr><td>Row55</td><td>2</td></tr>
<tr><td>Row57</td><td>14</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td>6</td>
<td>String Manipulation</td>
<td>
The <em>"Credit_Mix"</em> values are set to "Unknown" if the original value is "_" using the following expression:
replace($Credit_Mix$, "_","Unknown")
The new output is replaced to the original <em>"Credit_Mix"</em> column.


Category: All
Expression:
replace($Credit_Mix$,"_","Unknown")
○ Append Column:
● Replace Column: S Credit_Mix
□ Insert Missing As Null
☑ Syntax check on close

Example output:

File Table - 3:1 - CSV Reader
File Edit Hilite Navigation View
Table "default" - Rows: 100000 Spec - Columns: 25
<table>
<tr><th>Row ID</th><th>S Credit_Mix</th></tr>
<tr><td>Row0</td><td>_</td></tr>
<tr><td>Row1</td><td>Good</td></tr>
<tr><td>Row2</td><td>Good</td></tr>
<tr><td>Row3</td><td>Good</td></tr>
<tr><td>Row4</td><td>Good</td></tr>
<tr><td>Row5</td><td>Good</td></tr>
<tr><td>Row6</td><td>Good</td></tr>
<tr><td>Row7</td><td>Good</td></tr>
<tr><td>Row8</td><td>Good</td></tr>
<tr><td>Row9</td><td>Good</td></tr>
<tr><td>Row10</td><td>_</td></tr>
</table>

Appended table - 3:13 - String Manipulation
File Edit Hilite Navigation View
Table "default" - Rows: 90928 Spec - Columns: 24 Pr
<table>
<tr><th>Row ID</th><th>S Credit_Mix</th></tr>
<tr><td>Row0</td><td>Unknown</td></tr>
<tr><td>Row1</td><td>Good</td></tr>
<tr><td>Row3</td><td>Good</td></tr>
<tr><td>Row4</td><td>Good</td></tr>
<tr><td>Row5</td><td>Good</td></tr>
<tr><td>Row6</td><td>Good</td></tr>
<tr><td>Row7</td><td>Good</td></tr>
<tr><td>Row8</td><td>Good</td></tr>
<tr><td>Row9</td><td>Good</td></tr>
<tr><td>Row10</td><td>Unknown</td></tr>
</table>
</td>
</tr>
<tr>
<td>7</td>
<td>String Manipulation</td>
<td>
The non-numerical symbols in <em>"Outstanding_Debt"</em> attribute are erased and the data is converted to the double format. The "." and "-" are kept because they represent decimal number and negative number, respectively. The new output is replaced to the original <em>"Outstanding_Debt"</em> column.
Expression:
toDouble(regexReplace Outstanding_Debt$,"[^0-9.-]",""))


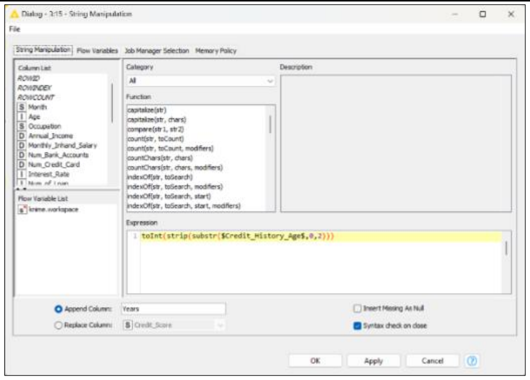
Category: All
Expression:
toDouble(regexReplace($Outstanding_Debt$,"[^0-9.-]",""))
○ Append Column:
● Replace Column: S Outstanding_Debt
□ Insert Missing As Null
☑ Syntax check on close

Example output:

File Table - 3:1 - CSV Reader
File Edit Hilite Navigation View
Table "default" - Rows: 100000 Spec - Column
<table>
<tr><th>Row ID</th><th>S Outstanding_Debt</th></tr>
<tr><td>Row84</td><td>1328.93</td></tr>
<tr><td>Row85</td><td>1328.93</td></tr>
<tr><td>Row86</td><td>1328.93</td></tr>
<tr><td>Row87</td><td>1328.93_</td></tr>
</table>

Appended table - 3:14 - String Manipulation
File Edit Hilite Navigation View
Table "default" - Rows: 90928 Spec - Columns: 24 Pro
<table>
<tr><th>Row ID</th><th>D Outstanding_Debt</th></tr>
<tr><td>Row84</td><td>1,328.93</td></tr>
<tr><td>Row85</td><td>1,328.93</td></tr>
<tr><td>Row86</td><td>1,328.93</td></tr>
<tr><td>Row87</td><td>1,328.93</td></tr>
</table>
</td>
</tr>
</table>

Question 1.6

S	Node	Configuration

String Manipulation


Dialog - 3:15 - String Manipulation

Expression:

```
toInt(strip(substr($Credit_History_Age$,0,2)))
```

● Append Column: Years
○ Replace Column: Credit_Score
□ Insert Missing As Null
☑ Syntax check on close

This node is applied to separate the number of year from the *"Credit_History_Age"* string value, erase white spaces and convert it to the number format.
The separated output is appended to the new column named *"Years"*.
Expression:
toInt(strip(substr ($Credit_History_Age$,0,2)))
Two characters counts from the first character (index 0) of a *"Credit_History_Age"* value are taken by using the expression substr(). For example, "9 Years and 6 Months" returns "9 " after substruction. The strip() expression is utilised to remove white spaces. For example, "9 " returns "9" after this function.

Example output:

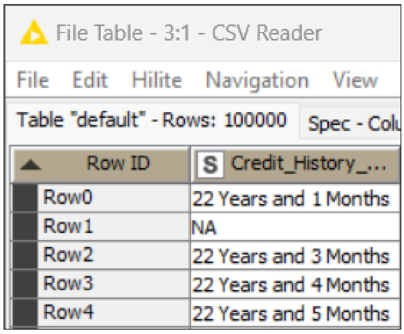

File Table - 3:1 - CSV Reader

Table "default" - Rows: 100000 Spec - Colu

Row ID	S Credit_History_...
Row0	22 Years and 1 Months
Row1	NA
Row2	22 Years and 3 Months
Row3	22 Years and 4 Months
Row4	22 Years and 5 Months

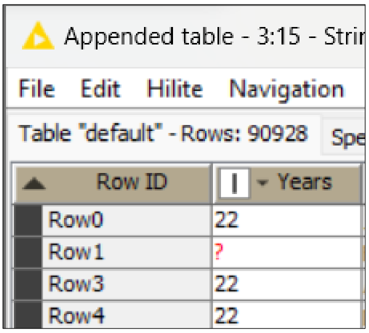
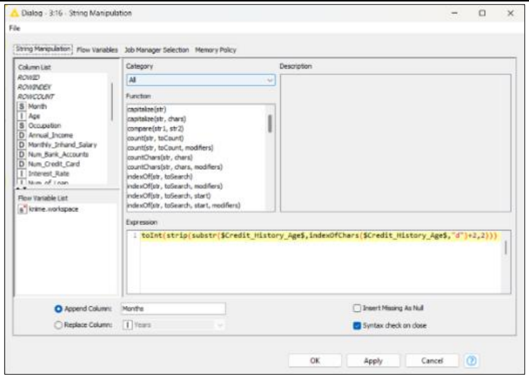

Appended table - 3:15 - Strir

Table "default" - Rows: 90928 Spe

Row ID	I Years
Row0	22
Row1	?
Row3	22
Row4	22

String Manipulation


Dialog - 3:16 - String Manipulation

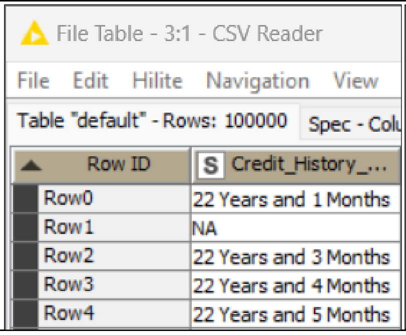
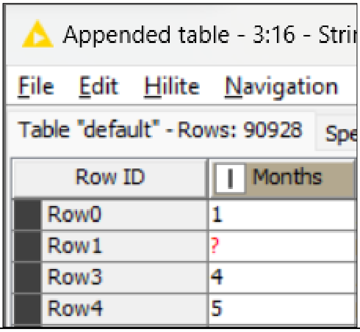
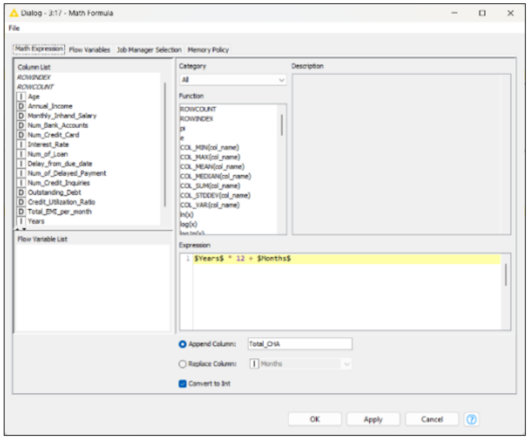
Expression:

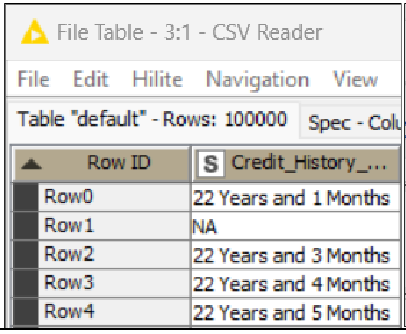
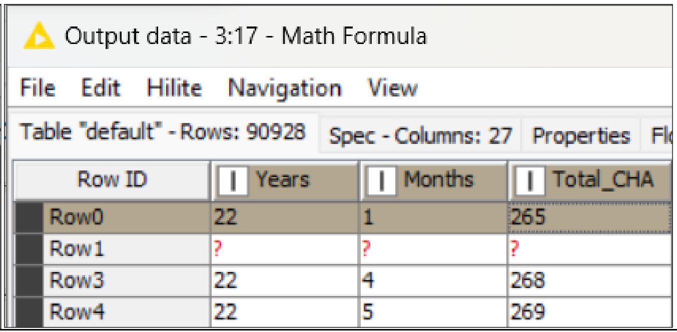
```
toInt(strip(substr($Credit_History_Age$,indexOfChars($Credit_History_Age$,"d")+2,2)))
```

● Append Column: Months
○ Replace Column: Years
□ Insert Missing As Null
☑ Syntax check on close

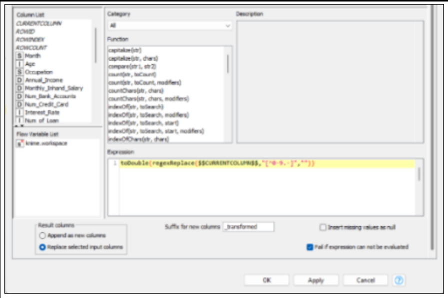
This node is applied to separate the number of month from the *"Credit_History_Age"* string value, erase white spaces and convert it to the number format.
The separated output is appended to the new column named *"Months"*.
Expression:
toInt(strip(substr Credit_History_Age$,indexOfChars ($Credit_History_Age$,"d")+2,2)))
Two characters counts from the first character starting after two characters from the letter "d" (indexOfChars($Credit_History_Age$,"d")+2) of a *"Credit_History_Age"* value are taken by using the expression substr(). For example, "9 Years and 6 Months" returns "6 " after substruction. The strip() expression is utilised to remove white spaces. For example, "6 " returns "6" after this function.

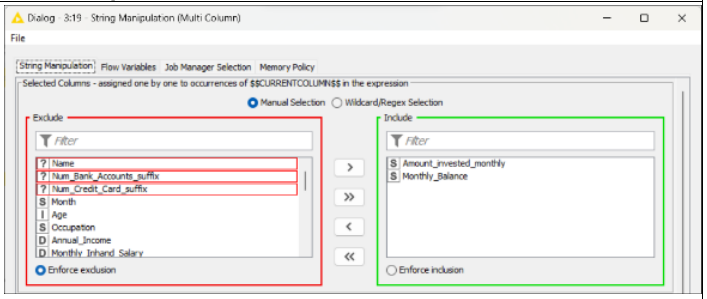
Example output:

<table>
<tr>
<td></td>
<td></td>
<td>
File Table - 3:1 - CSV Reader
Table "default" - Rows: 100000
<table>
<tr><th>Row ID</th><th>S Credit_History_...</th></tr>
<tr><td>Row0</td><td>22 Years and 1 Months</td></tr>
<tr><td>Row1</td><td>NA</td></tr>
<tr><td>Row2</td><td>22 Years and 3 Months</td></tr>
<tr><td>Row3</td><td>22 Years and 4 Months</td></tr>
<tr><td>Row4</td><td>22 Years and 5 Months</td></tr>
</table>

Appended table - 3:16 - Stri
Table "default" - Rows: 90928
<table>
<tr><th>Row ID</th><th>I Months</th></tr>
<tr><td>Row0</td><td>1</td></tr>
<tr><td>Row1</td><td>?</td></tr>
<tr><td>Row3</td><td>4</td></tr>
<tr><td>Row4</td><td>5</td></tr>
</table>
</td>
</tr>
<tr>
<td></td>
<td><b>Math Formula</b></td>
<td>The <i>“Credit_History_Age”</i> values are converted to the count of months and store it in the integer format using the following formula:
$Years$ * 12 + $Months$
The converted output is stored in a new column called <i>“Total_CHA”.</i>

Dialog - 3:17 - Math Formula
Expression: $Years$ * 12 + $Months$
● Append Column: Total_CHA
○ Replace Column: Months
☑ Convert to Int

Example output:

File Table - 3:1 - CSV Reader
Table "default" - Rows: 100000
<table>
<tr><th>Row ID</th><th>S Credit_History_...</th></tr>
<tr><td>Row0</td><td>22 Years and 1 Months</td></tr>
<tr><td>Row1</td><td>NA</td></tr>
<tr><td>Row2</td><td>22 Years and 3 Months</td></tr>
<tr><td>Row3</td><td>22 Years and 4 Months</td></tr>
<tr><td>Row4</td><td>22 Years and 5 Months</td></tr>
</table>

Output data - 3:17 - Math Formula
Table "default" - Rows: 90928 Spec - Columns: 27 Properties
<table>
<tr><th>Row ID</th><th>I Years</th><th>I Months</th><th>I Total_CHA</th></tr>
<tr><td>Row0</td><td>22</td><td>1</td><td>265</td></tr>
<tr><td>Row1</td><td>?</td><td>?</td><td>?</td></tr>
<tr><td>Row3</td><td>22</td><td>4</td><td>268</td></tr>
<tr><td>Row4</td><td>22</td><td>5</td><td>269</td></tr>
</table>
</td>
</tr>
</table>

Question 1.7

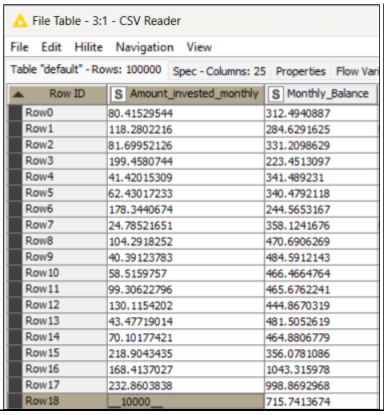
<table>
<tr><th>S</th><th>Node</th><th>Configuration</th></tr>
<tr>
<td>1</td>
<td><b>String Manipulation (Multi Column)</b></td>
<td>
Expression: toDouble(regexReplace($$CURRENTCOLUMN$$,"[^0-9.-]",""))
Result columns
○ Append as new columns
● Replace selected input columns
Suffix for new columns: _transformed
□ Insert missing values as null
☑ Fail if expression can not be evaluated


Dialog - 3:19 - String Manipulation (Multi Column)
Selected Columns - assigned one by one to occurrences of $$CURRENTCOLUMN$$ in the expression
● Manual Selection ○ Wildcard/Regex Selection
Exclude: ? Name, ? Num_Bank_Accounts_suffix, ? Num_Credit_Card_suffix, S Month, I Age, S Occupation, D Annual_Income, D Monthly_Inhand_Salary
● Enforce exclusion
Include: S Amount_invested_monthly, S Monthly_Balance
○ Enforce inclusion

This node is used to remove the non-numerical symbol in two columns: <i>“Amount_invested_monthly”</i> and <i>“Monthly_Balance”</i> and convert it to the double format.
Expression:
toDouble(regexReplace(
$$CURRENTCOLUMN$$,"[^0-9.-]",""))
This function is used to apply regex to string and replaces string if regex matches.</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td>
The “.” and “-” are kept because they represent decimal number and negative number, respectively. The new output is replaced to their two original columns.

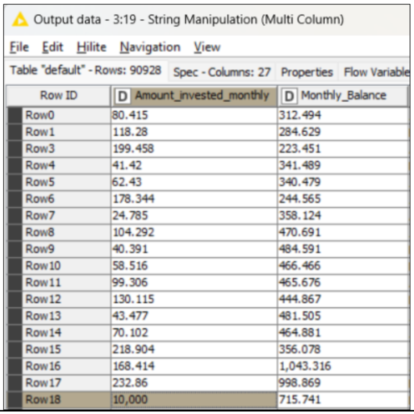
Example output:



File Table - 3:1 - CSV Reader

Table "default" - Rows: 100000 | Spec - Columns: 25

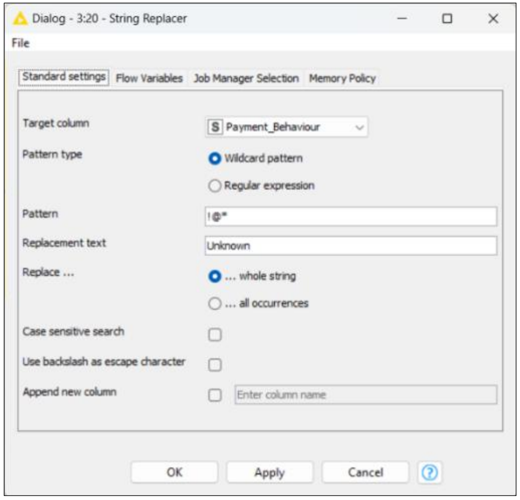
<table>
<tr><th>Row ID</th><th>S Amount_invested_monthly</th><th>S Monthly_Balance</th></tr>
<tr><td>Row0</td><td>80.41529544</td><td>312.4940887</td></tr>
<tr><td>Row1</td><td>118.2802216</td><td>284.6291625</td></tr>
<tr><td>Row2</td><td>81.69952126</td><td>331.2098629</td></tr>
<tr><td>Row3</td><td>199.4580744</td><td>223.4513097</td></tr>
<tr><td>Row4</td><td>41.42015309</td><td>341.489231</td></tr>
<tr><td>Row5</td><td>62.43017233</td><td>340.4792118</td></tr>
<tr><td>Row6</td><td>178.3440674</td><td>244.5653167</td></tr>
<tr><td>Row7</td><td>24.78521651</td><td>358.1241676</td></tr>
<tr><td>Row8</td><td>104.2918252</td><td>470.6906269</td></tr>
<tr><td>Row9</td><td>40.39123783</td><td>484.5912143</td></tr>
<tr><td>Row10</td><td>58.5159757</td><td>466.4664764</td></tr>
<tr><td>Row11</td><td>99.30622796</td><td>465.6762241</td></tr>
<tr><td>Row12</td><td>130.1154202</td><td>444.8670319</td></tr>
<tr><td>Row13</td><td>43.47719014</td><td>481.5052619</td></tr>
<tr><td>Row14</td><td>70.10177421</td><td>464.8806779</td></tr>
<tr><td>Row15</td><td>218.9043435</td><td>356.0781086</td></tr>
<tr><td>Row16</td><td>168.4137027</td><td>1043.315978</td></tr>
<tr><td>Row17</td><td>232.8603838</td><td>998.8692968</td></tr>
<tr><td>Row18</td><td>__10000__</td><td>715.7413674</td></tr>
</table>



Output data - 3:19 - String Manipulation (Multi Column)

Table "default" - Rows: 90928 | Spec - Columns: 27

<table>
<tr><th>Row ID</th><th>D Amount_invested_monthly</th><th>D Monthly_Balance</th></tr>
<tr><td>Row0</td><td>80.415</td><td>312.494</td></tr>
<tr><td>Row1</td><td>118.28</td><td>284.629</td></tr>
<tr><td>Row3</td><td>199.458</td><td>223.451</td></tr>
<tr><td>Row4</td><td>41.42</td><td>341.489</td></tr>
<tr><td>Row5</td><td>62.43</td><td>340.479</td></tr>
<tr><td>Row6</td><td>178.344</td><td>244.565</td></tr>
<tr><td>Row7</td><td>24.785</td><td>358.124</td></tr>
<tr><td>Row8</td><td>104.292</td><td>470.691</td></tr>
<tr><td>Row9</td><td>40.391</td><td>484.591</td></tr>
<tr><td>Row10</td><td>58.516</td><td>466.466</td></tr>
<tr><td>Row11</td><td>99.306</td><td>465.676</td></tr>
<tr><td>Row12</td><td>130.115</td><td>444.867</td></tr>
<tr><td>Row13</td><td>43.477</td><td>481.505</td></tr>
<tr><td>Row14</td><td>70.102</td><td>464.881</td></tr>
<tr><td>Row15</td><td>218.904</td><td>356.078</td></tr>
<tr><td>Row16</td><td>168.414</td><td>1,043.316</td></tr>
<tr><td>Row17</td><td>232.86</td><td>998.869</td></tr>
<tr><td>Row18</td><td>10,000</td><td>715.741</td></tr>
</table>
</td>
</tr>
<tr>
<td>2</td>
<td>String Replacer</td>
<td>
This node is used to set the value to “Unknown” if the original value in *“Payment_Behaviour”* column starts with “!@”. The “*” is applied to find the wildcard pattern in the right order.



Dialog - 3:20 - String Replacer

Standard settings | Flow Variables | Job Manager Selection | Memory Policy

Target column: S Payment_Behaviour

Pattern type: ● Wildcard pattern ○ Regular expression

Pattern: !@*

Replacement text: Unknown

Replace ...: ● ... whole string ○ ... all occurrences

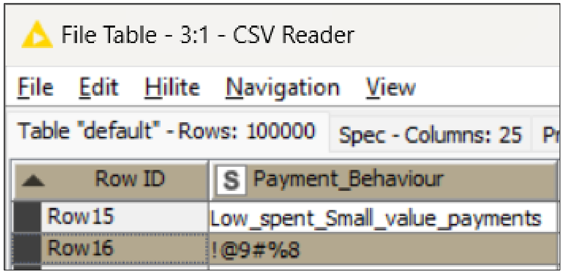
Case sensitive search □

Use backslash as escape character □

Append new column □ Enter column name

OK | Apply | Cancel

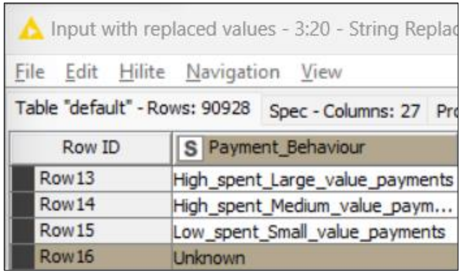
Example output:



File Table - 3:1 - CSV Reader

Table "default" - Rows: 100000 | Spec - Columns: 25

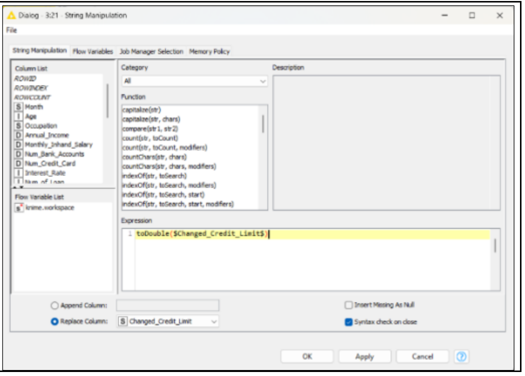
<table>
<tr><th>Row ID</th><th>S Payment_Behaviour</th></tr>
<tr><td>Row15</td><td>Low_spent_Small_value_payments</td></tr>
<tr><td>Row16</td><td>!@9#%8</td></tr>
</table>



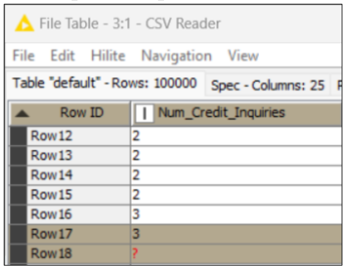
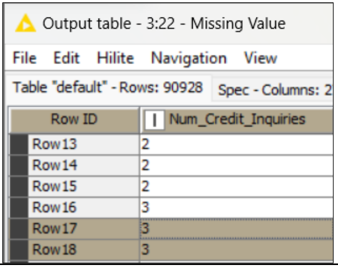
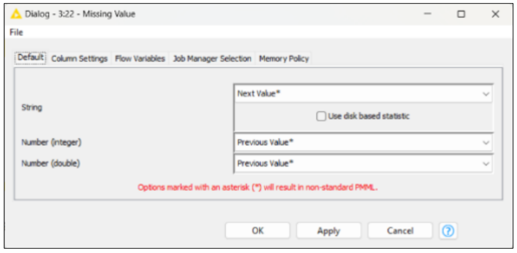
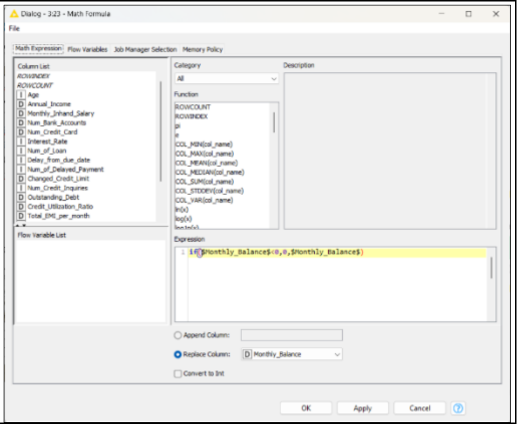
Input with replaced values - 3:20 - String Replac

Table "default" - Rows: 90928 | Spec - Columns: 27

<table>
<tr><th>Row ID</th><th>S Payment_Behaviour</th></tr>
<tr><td>Row13</td><td>High_spent_Large_value_payments</td></tr>
<tr><td>Row14</td><td>High_spent_Medium_value_paym...</td></tr>
<tr><td>Row15</td><td>Low_spent_Small_value_payments</td></tr>
<tr><td>Row16</td><td>Unknown</td></tr>
</table>
</td>
</tr>
</table>

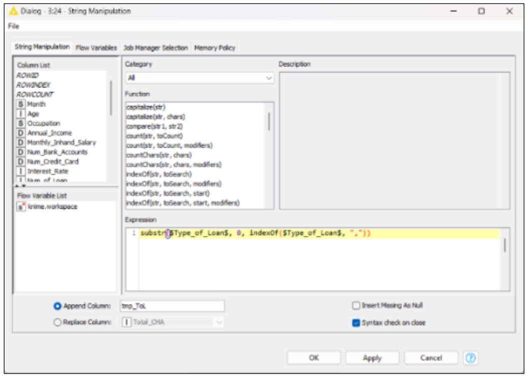
3	String Manipulation	*“Changed_Credit_Limit”* colum is converted to the double format using the expression as follows: toDouble($Changed_Credit_Limit$) The new output is replaced to the original *“Changed_Credit_Limit”* column.	Dialog - 3:21 - String Manipulation Expression: `toDouble($Changed_Credit_Limit$)` ○ Append Column: ● Replace Column: S Changed_Credit_Limit ☐ Insert Missing As Null ☑ Syntax check on close

Question 1.8

S	Node	Configuration	
1	Missing Value	This node is used to replace the missing values in all string type attributes to the “Next Value*” and all numeric format to the “Previous Value*”. Example output: File Table - 3:1 - CSV Reader Table "default" - Rows: 100000 Spec - Columns: 25 Row ID, Num_Credit_Inquiries Row12 2 Row13 2 Row14 2 Row15 2 Row16 3 Row17 3 Row18 ? Output table - 3:22 - Missing Value Table "default" - Rows: 90928 Spec - Columns: 2 Row ID, Num_Credit_Inquiries Row13 2 Row14 2 Row15 2 Row16 3 Row17 3 Row18 3	Dialog - 3:22 - Missing Value String: Next Value* ☐ Use disk based statistic Number (integer): Previous Value* Number (double): Previous Value* Options marked with an asterisk (*) will result in non-standard PMML.
2	Math Formula	The *“Monthly_Balance”* value is replaced with 0 if it is negative using the if condition formula if(condition, true value, false value): if($Monthly_Balance$<0, 0,$Num_of_Loan$) The new output is replaced to the original *“Monthly_Balance”* columns.	Dialog - 3:23 - Math Formula Expression: `if($Monthly_Balance$<0,0,$Monthly_Balance$)` ○ Append Column: ● Replace Column: D Monthly_Balance ☐ Convert to Int

Question 1.9

<table>
<tr><th>S</th><th>Node</th><th>Configuration</th></tr>
<tr>
<td>1</td>
<td>String Manipulation
 </td>
<td>
The <i>“Type_of_Loan”</i> column is simplified using this node. If the original content has more than one type separated by a comma, keep only the first part. Otherwise, keep the full description if there is no comma included. The output result is appended to a new column called <i>“tmp_ToL”</i>.
Expression:
substr($Type_of_Loan$, 0, indexOf($Type_of_Loan$, ","))
This function is applied to substract the string from <i>“Type_of_Loan”</i> column starting from the first character (index 0) to the first comma. For example, “Auto Loan, Auto Loan, and Not Specified” returns “Auto Loan” after substraction and “Credit-Builder Loan” returns missing value because there are not any comma in the value.



<pre>substr($Type_of_Loan$, 0, indexOf($Type_of_Loan$, ","))</pre>
● Append Column: tmp_ToL
○ Replace Column: Total_CHA
□ Insert Missing As Null
☑ Syntax check on close

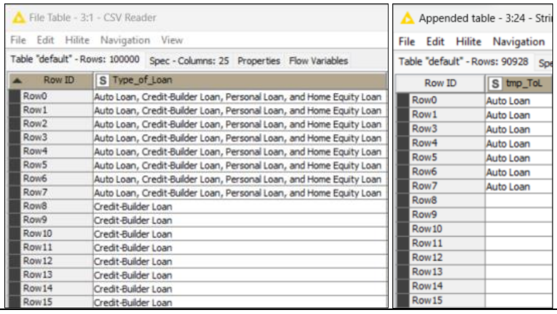
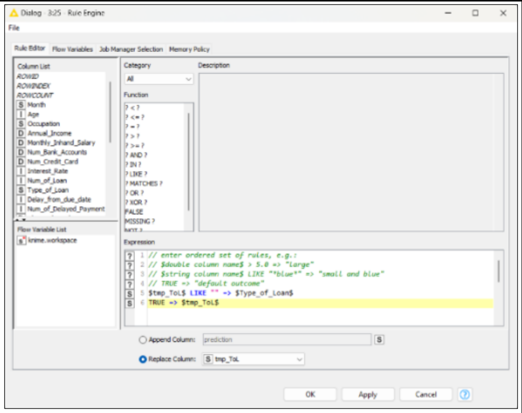
Example output:


Table "default" - Rows: 100000 Spec - Columns: 25
<table>
<tr><th>Row ID</th><th>S Type_of_Loan</th></tr>
<tr><td>Row0</td><td>Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan</td></tr>
<tr><td>Row1</td><td>Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan</td></tr>
<tr><td>Row2</td><td>Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan</td></tr>
<tr><td>Row3</td><td>Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan</td></tr>
<tr><td>Row4</td><td>Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan</td></tr>
<tr><td>Row5</td><td>Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan</td></tr>
<tr><td>Row6</td><td>Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan</td></tr>
<tr><td>Row7</td><td>Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan</td></tr>
<tr><td>Row8</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row9</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row10</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row11</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row12</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row13</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row14</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row15</td><td>Credit-Builder Loan</td></tr>
</table>
Table "default" - Rows: 90928 Spe
<table>
<tr><th>Row ID</th><th>S tmp_ToL</th></tr>
<tr><td>Row0</td><td>Auto Loan</td></tr>
<tr><td>Row1</td><td>Auto Loan</td></tr>
<tr><td>Row3</td><td>Auto Loan</td></tr>
<tr><td>Row4</td><td>Auto Loan</td></tr>
<tr><td>Row5</td><td>Auto Loan</td></tr>
<tr><td>Row6</td><td>Auto Loan</td></tr>
<tr><td>Row7</td><td>Auto Loan</td></tr>
<tr><td>Row8</td><td></td></tr>
<tr><td>Row9</td><td></td></tr>
<tr><td>Row10</td><td></td></tr>
<tr><td>Row11</td><td></td></tr>
<tr><td>Row12</td><td></td></tr>
<tr><td>Row13</td><td></td></tr>
<tr><td>Row14</td><td></td></tr>
<tr><td>Row15</td><td></td></tr>
</table>
</td>
</tr>
<tr>
<td>2</td>
<td>Rule Engine
 </td>
<td>
This node is used to recover the missing value from the previous node to its original value from the <i>“Type_of_Loan”</i> column and keep the value subtracted in the previous node. The new output is replaced to the <i>“tmp_ToL”</i> column.
Expression:
$tmp_ToL$ LIKE "" =&gt; $Type_of_Loan$ (the missing values are recovered from the original <i>“Type_of_Loan”</i> attribute)
TRUE =&gt; $tmp_ToL$ (the default outcome remains the same)



<pre>// enter ordered set of rules, e.g.:
// $double column name$ > 5.0 => "Large"
// $string column name$ LIKE "*blue*" => "small and blue"
// TRUE => "default outcome"
$tmp_ToL$ LIKE "" => $Type_of_Loan$
TRUE => $tmp_ToL$</pre>
○ Append Column: prediction
● Replace Column: S tmp_ToL

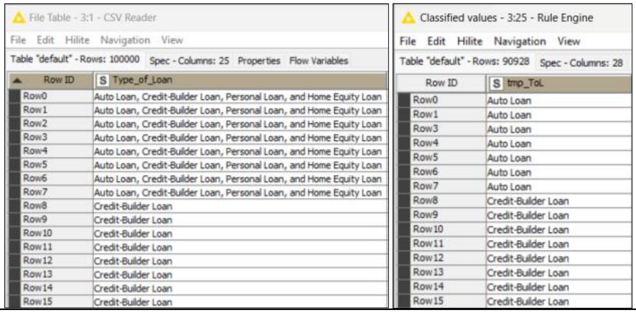
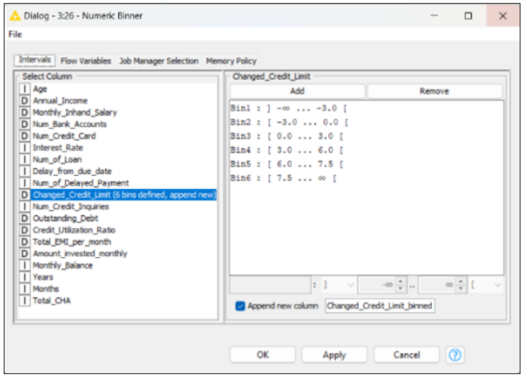
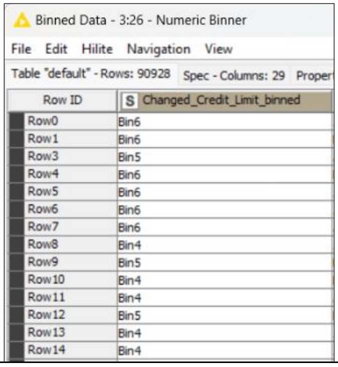
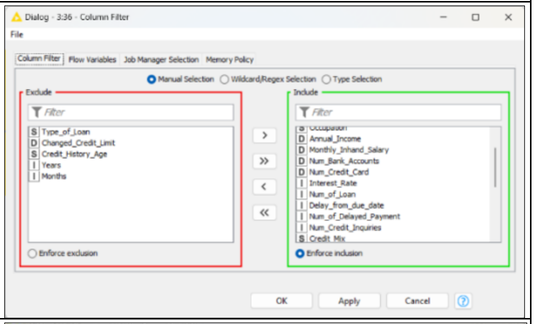
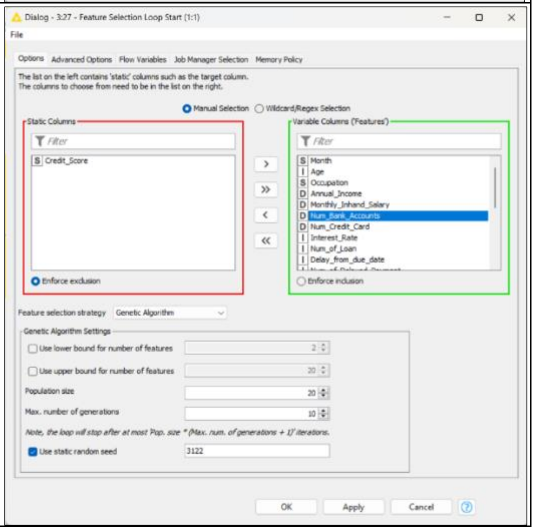
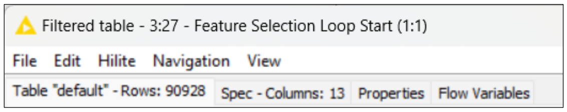
Example output:


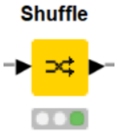
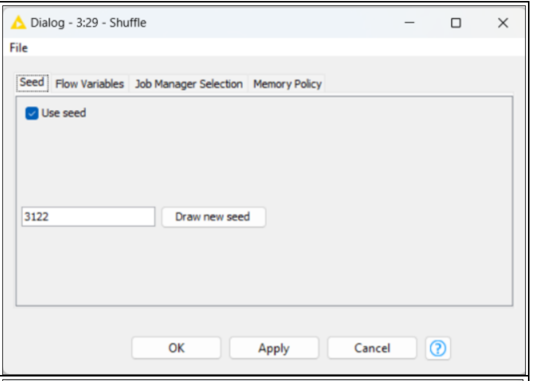
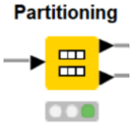
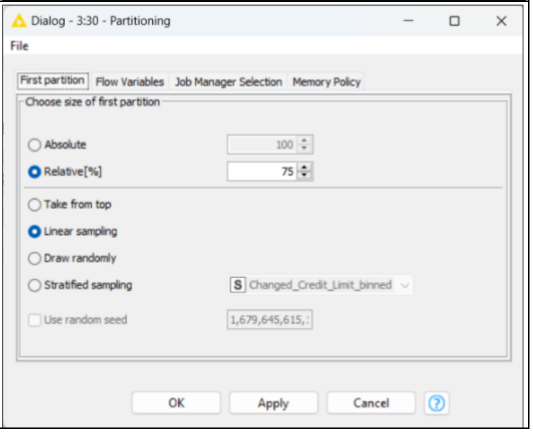
Table "default" - Rows: 100000 Spec - Columns: 25
<table>
<tr><th>Row ID</th><th>S Type_of_Loan</th></tr>
<tr><td>Row0</td><td>Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan</td></tr>
<tr><td>Row1</td><td>Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan</td></tr>
<tr><td>Row2</td><td>Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan</td></tr>
<tr><td>Row3</td><td>Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan</td></tr>
<tr><td>Row4</td><td>Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan</td></tr>
<tr><td>Row5</td><td>Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan</td></tr>
<tr><td>Row6</td><td>Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan</td></tr>
<tr><td>Row7</td><td>Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan</td></tr>
<tr><td>Row8</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row9</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row10</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row11</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row12</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row13</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row14</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row15</td><td>Credit-Builder Loan</td></tr>
</table>
Table "default" - Rows: 90928 Spec - Columns: 28
<table>
<tr><th>Row ID</th><th>S tmp_ToL</th></tr>
<tr><td>Row0</td><td>Auto Loan</td></tr>
<tr><td>Row1</td><td>Auto Loan</td></tr>
<tr><td>Row3</td><td>Auto Loan</td></tr>
<tr><td>Row4</td><td>Auto Loan</td></tr>
<tr><td>Row5</td><td>Auto Loan</td></tr>
<tr><td>Row6</td><td>Auto Loan</td></tr>
<tr><td>Row7</td><td>Auto Loan</td></tr>
<tr><td>Row8</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row9</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row10</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row11</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row12</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row13</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row14</td><td>Credit-Builder Loan</td></tr>
<tr><td>Row15</td><td>Credit-Builder Loan</td></tr>
</table>
</td>
</tr>
</table>

Question 1.10

Node	Configuration
Numeric Binner	This node is utilised to bin the *"Changed_Credit_Limit"* attribute with six bins of ranges: [−∞,−3.0), [−3.0,0), [0,3.0), [3.0,6.0), [6.0,7.5), and [7.5,∞) and put the result into a new attribute called *"Changed_Credit_Limit_binned".*  Dialog - 3:26 - Numeric Binner Intervals | Flow Variables | Job Manager Selection | Memory Policy Select Column: I Age, D Annual_Income, D Monthly_Inhand_Salary, D Num_Bank_Accounts, D Num_Credit_Card, I Interest_Rate, I Num_of_Loan, I Delay_from_due_date, I Num_of_Delayed_Payment, D Changed_Credit_Limit (6 bins defined, append new), I Num_Credit_Inquiries, D Outstanding_Debt, D Credit_Utilization_Ratio, D Total_EMI_per_month, D Amount_invested_monthly, I Monthly_Balance, I Years, I Months, I Total_CHA Changed_Credit_Limit Bin1 :] -∞ ... -3.0 [Bin2 : [-3.0 ... 0.0 [Bin3 : [0.0 ... 3.0 [Bin4 : [3.0 ... 6.0 [Bin5 : [6.0 ... 7.5 [Bin6 : [7.5 ... ∞ [☑ Append new column: Changed_Credit_Limit_binned Example output:  Binned Data - 3:26 - Numeric Binner File Edit Hilite Navigation View Table "default" - Rows: 90928 | Spec - Columns: 29 | Proper Row ID | S Changed_Credit_Limit_binned Row0 | Bin6 Row1 | Bin6 Row3 | Bin5 Row4 | Bin6 Row5 | Bin6 Row6 | Bin6 Row7 | Bin6 Row8 | Bin4 Row9 | Bin5 Row10 | Bin4 Row11 | Bin4 Row12 | Bin5 Row13 | Bin4 Row14 | Bin4

Question 1.11

S	Node	Configuration
1	**Column Filter**	The Column Fiter node is used to filter out the original columns that have their new data being appended to new columns (*"Type_of_Loan", "Changed_Credit_Limit", "Credit_History_Age"*) and attributes that are the output of the number of year and month substraction being used for converting the *"Credit_History_Age"* to the count of months (*"Years", "Months"*).  Dialog - 3:36 - Column Filter Column Filter | Flow Variables | Job Manager Selection | Memory Policy ● Manual Selection ○ Wildcard/Regex Selection ○ Type Selection Exclude: S Type_of_Loan, D Changed_Credit_Limit, S Credit_History_Age, I Years, I Months ○ Enforce exclusion Include: S Occupation, D Annual_Income, D Monthly_Inhand_Salary, D Num_Bank_Accounts, D Num_Credit_Card, I Interest_Rate, I Num_of_Loan, I Delay_from_due_date, I Num_of_Delayed_Payment, I Num_Credit_Inquiries, S Credit_Mix ● Enforce inclusion
2	**Feature Selection Loop Start (1:1)**	This node is applied to let the algorithm select the attributes using the Genetic Algorithm feature selection strategy with default population size and the maximum number of generations. The class label *"Credit_Score"* is excluded and the static random seed of 3122 is applied.  Dialog - 3:27 - Feature Selection Loop Start (1:1) Options | Advanced Options | Flow Variables | Job Manager Selection | Memory Policy The list on the left contains 'static' columns such as the target column. The columns to choose from need to be in the list on the right. ● Manual Selection ○ Wildcard/Regex Selection Static Columns: S Credit_Score ● Enforce exclusion Variable Columns ('Features'): S Month, I Age, S Occupation, D Annual_Income, D Monthly_Inhand_Salary, D Num_Bank_Accounts, D Num_Credit_Card, I Interest_Rate, I Num_of_Loan, I Delay_from_due_date ○ Enforce inclusion Feature selection strategy: Genetic Algorithm Genetic Algorithm Settings ☐ Use lower bound for number of features: 2 ☐ Use upper bound for number of features: 20 Population size: 20 Max. number of generations: 10 *Note, the loop will stop after at most Pop. size * (Max. num. of generations + 1) iterations.* ☑ Use static random seed: 3122 Example output:  Filtered table - 3:27 - Feature Selection Loop Start (1:1) File Edit Hilite Navigation View Table "default" - Rows: 90928 | Spec - Columns: 13 | Properties | Flow Variables

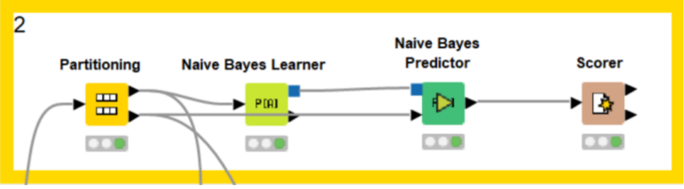
3		The dataset is shuffled with the random seed being set to 3122.	Dialog - 3:29 - Shuffle File Seed | Flow Variables | Job Manager Selection | Memory Policy ☑ Use seed 3122 Draw new seed OK Apply Cancel ?
4		The dataset is partitioned by Linear sampling and a 75:25 ratio, with 75% belongs to the training set and the remaining 25% for the test set.	Dialog - 3:30 - Partitioning File First partition | Flow Variables | Job Manager Selection | Memory Policy Choose size of first partition ○ Absolute 100 ● Relative[%] 75 ○ Take from top ● Linear sampling ○ Draw randomly ○ Stratified sampling S Changed_Credit_Limit_binned □ Use random seed 1,679,645,615, OK Apply Cancel ?

IV. DATA ANALYTICS

1. Naïve Bayes Classifier

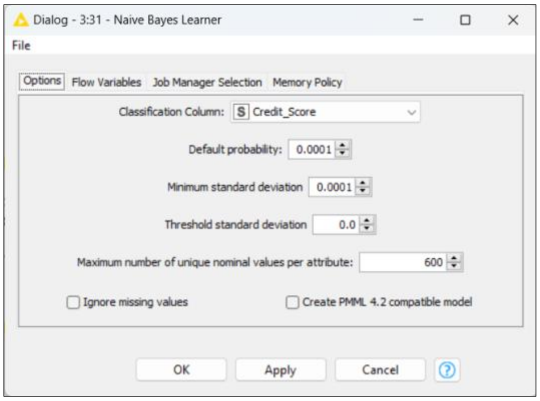
Question 2.1

A screenshot of the Naïve Bayes classifier in the KNIME workflow:



Question 2.2

The default probability value and the minimum standard deviation is set to 0.0001. The threshold standard deviation is 0 and the maximum number of unique nominal values per attribute of 600 is applied to the classifier leaner.



Dialog - 3:31 - Naive Bayes Learner

File

Options | Flow Variables | Job Manager Selection | Memory Policy

Classification Column: S Credit_Score

Default probability: 0.0001

Minimum standard deviation 0.0001

Threshold standard deviation 0.0

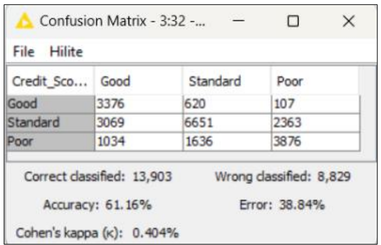
Maximum number of unique nominal values per attribute: 600

□ Ignore missing values □ Create PMML 4.2 compatible model

OK Apply Cancel ?

Question 2.3

Screenshots of the Confusion Matrix and the Accuracy statistics of the test result:



Confusion Matrix - 3:32 -...

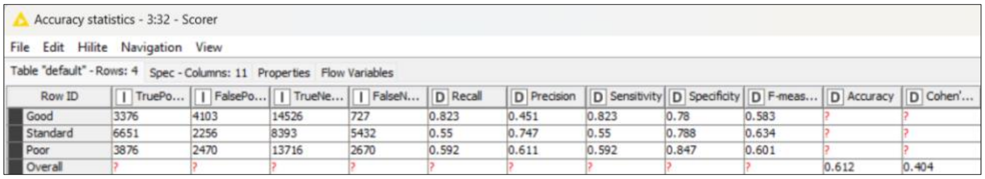
File Hilite

Credit_Sco...	Good	Standard	Poor
Good	3376	620	107
Standard	3069	6651	2363
Poor	1034	1636	3876

Correct classified: 13,903 Wrong classified: 8,829

Accuracy: 61.16% Error: 38.84%

Cohen's kappa (κ): 0.404%



Accuracy statistics - 3:32 - Scorer

File Edit Hilite Navigation View

Table "default" - Rows: 4 Spec - Columns: 11 Properties Flow Variables

Row ID	I TruePo...	I FalsePo...	I TrueNe...	I FalseN...	D Recall	D Precision	D Sensitivity	D Specificity	D F-meas...	D Accuracy	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

This Naïve Bayes classifier produces a low precision result of 0.451 which indicates that this classifier performs unsatisfactorily.

Question 2.4

The measurement being looked at to interpret the conclusion in this case is: Precision.

If the bank wants to minimise the risk of lending money to customers, the "Good" in *"Credit_Score"* should be the major target. So, the statistic represents the value being predicted as "Good" are actually "Good" is taken into consideration.

In a confusion matrix, precision is defined as the ratio of true positives (TP: The number of "Good" customers is correctly classified) to the sum of true positives and false positives (FP: The number of "Poor"/"Standard" customers is incorrectly identified as "Good" ones):

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

Precision measures the proportion of positive predictions (which means the instances that the model classifies as positive) that are actually correct.

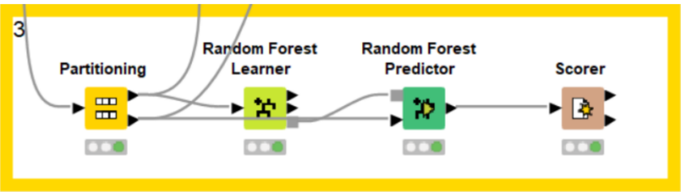
A high precision value indicates that the model is making very few false positive predictions (a few number of "Poor"/"Standard" customer incorrectly classified as "Good" ones) and vice versa.

This Naïve Bayes classifier produces a low precision result of 0.451 which indicates that this classifier perform unsatisfactorily.

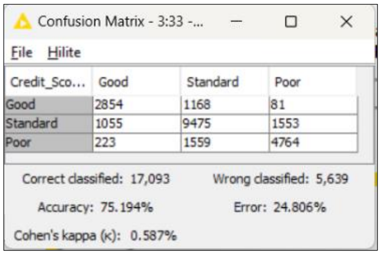
2. Random Forest Classifier

Question 3.1

A screenshot of the Random Forest classifier in the KNIME workflow:



Question 3.2



Confusion Matrix - 3:33 -...

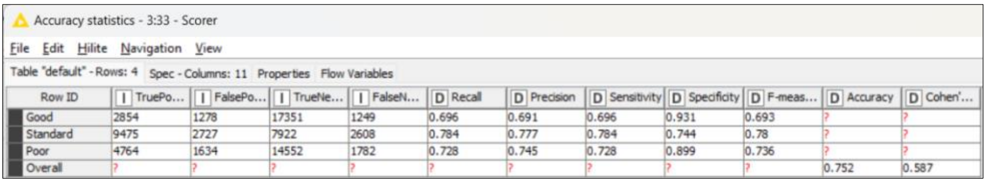
File Hilite

Credit_Sco...	Good	Standard	Poor
Good	2854	1168	81
Standard	1055	9475	1553
Poor	223	1559	4764

Correct classified: 17,093 Wrong classified: 5,639

Accuracy: 75.194% Error: 24.806%

Cohen's kappa (κ): 0.587%



Accuracy statistics - 3:33 - Scorer

File Edit Hilite Navigation View

Table "default" - Rows: 4 Spec - Columns: 11 Properties Flow Variables

Row ID	I TruePo...	I FalsePo...	I TrueNe...	I FalseN...	D Recall	D Precision	D Sensitivity	D Specificity	D F-meas...	D Accuracy	D Cohen'...
Good	2854	1278	17351	1249	0.696	0.691	0.696	0.931	0.693	?	?
Standard	9475	2727	7922	2608	0.784	0.777	0.784	0.744	0.78	?	?
Poor	4764	1634	14552	1782	0.728	0.745	0.728	0.899	0.736	?	?
Overall	?	?	?	?	?	?	?	?	?	0.752	0.587

Question 3.3

The measurement being looked at to interpret the conclusion in this case is: Precision.

If the bank wants to minimise the risk of lending money to customers, the "Good" in *"Credit_Score"* should be the major target. So, the statistic represents the value being predicted as "Good" are actually "Good" is taken into consideration.

In a confusion matrix, precision is defined as the ratio of true positives (TP: The number of "Good" customers is correctly classified) to the sum of true positives and false positives (FP: The number of "Poor"/"Standard" customers is incorrectly identified as "Good" ones):

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

Precision measures the proportion of positive predictions (which means the instances that the model classifies as positive) that are actually correct.

A high precision value indicates that the model is making very few false positive predictions (a few number of "Poor"/"Standard" customer incorrectly classified as "Good" ones) and vice versa.

Random Forest results and Naïve Bayes results Precision comparison:

	Random Forest	Naïve Bayes
Precision (Good)	0.691	0.451
Accuracy (Overall)	0.752	0.612

The Naïve Bayes classifier produces a lower precision (and accuracy) result than the Random Forest one, which indicates that Random Forest model presents a more suitable result compared to Naïve Bayes one.

Question 3.4

Some measurements that should be looked at to find out which class performs the best by the model:

	Precision	Recall	F-measure
Good	0.691	0.696	0.693
Standard	0.777	0.784	0.780
Poor	0.745	0.728	0.736

Precision, Recall and F-measure are common metrics used to evaluate the performance of classification models, including the Random Forest model:

- Precision is used to evaluate how well a model can correctly identify instances of a particular class. It measures the proportion of true positives (instances of the target class that were correctly identified by the model) out of all instances that the model classified as positive for that class. A higher precision means that the model makes fewer false positive errors, which means fewer instances that do not belong to the target class are mistakenly identified as belonging to it.
- Recall measures is used to evaluate how well a model can identify all relevant instances of a particular class. In other words, recall measures the proportion of true positives (instances of the target class that were correctly identified by the model) out of all actual positives (all instances of the target class in the dataset). In this classification problem where all classes are equally important, a higher recall for a particular class would indicate that the model is better at identifying that class.
- F-measure is a metric that combines precision and recall into a single score, which gives equal weight to both measures. It is calculated as the harmonic mean of precision and recall, with values ranging from 0 to 1, where a higher score indicates better performance. Therefore, a higher F-measure for a particular class means that the model is better at identifying instances of that class while also minimising false positives.

Looking at the results, it is obviously that class “Standard” performs the best results.

V. CONCLUSION

In conclusion, this assignment aimed to clean and prepare raw data for future use, and to create two predictive models capable of projecting *"Credit_Score"* classifications. The tasks involved in this project have been thoroughly described in this report, including preparing the data by organising and cleaning it, constructing two models (Naïve Bayes and Random Forest classifier models) to predict and categorise financial credit scores, selecting appropriate features and models for the project, identifying relevant attributes, dividing the dataset into training and testing sets, building predictive models, and providing an explanation of the results. Overall, the report comprehensively showcases and explains the data cleaning and model constructing process.