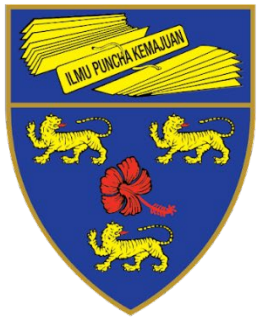


UNIVERSITI MALAYA

Faculty of Computer Science & Information Technology

Master of Data Science
Semester I Session 2023/2024

WQD7005 Data Mining

Alternative Assessment 1 (Case Study)
Title: E-Commerce Customer Behaviour Analysis

Github link: https://github.com/Dennis-1121/WQD7005_17205824_AA1

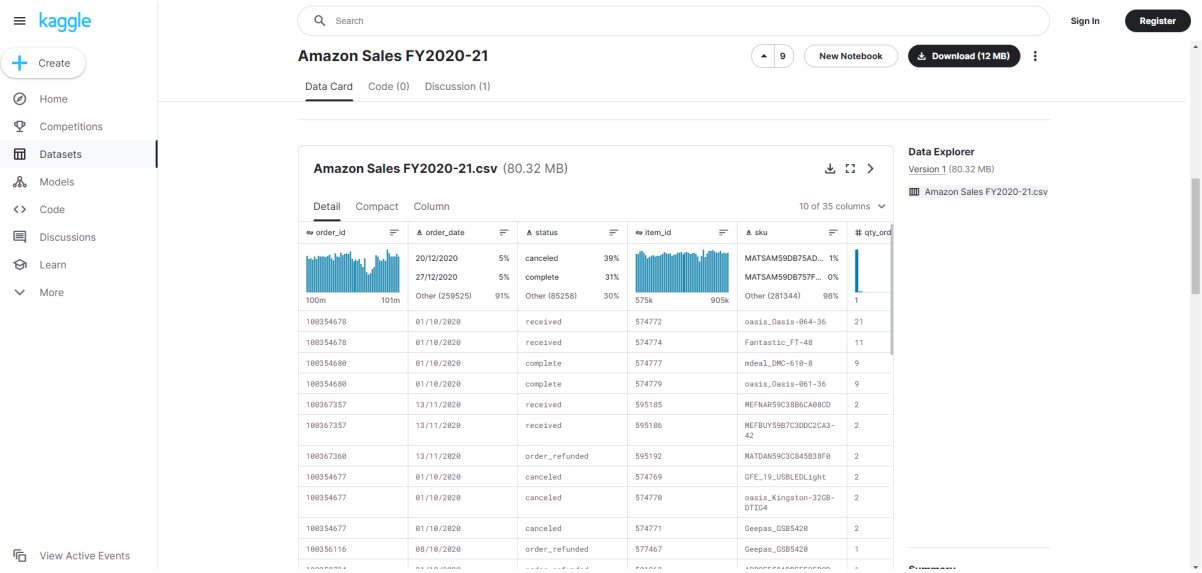
Prepared by
Ang Qi Kang (17205824)

Lecturer: Professor Teh Ying Wah

Introduction

The e-commerce sector is blooming with the advancement of technology today. In order to grab the opportunities, sellers should make the right decision in when to have promotions and what items they should sell based on the geographic location.

The e-commerce dataset is obtained from the Kaggle. The datasets consist records from year 2020 to year 2021. From the datasets, I selected the records for 1000 unique customer IDs. The datasets features are 'order_id', 'order_date', 'status', 'item_id', 'sku', 'qty_ordered', 'price', 'value', 'discount_amount', 'total', 'category', 'payment_method', 'bi_st', 'cust_id', 'year', 'month', 'ref_num', 'Name Prefix', 'First Name', 'Middle Initial', 'Last Name', 'Gender', 'age', 'full_name', 'E Mail', 'Sign in date', 'Phone No. ', 'Place Name', 'County', 'City', 'State', 'Zip', 'Region', 'User Name', 'Discount_Percent'.



To obtain a similar dataset which has ‘cust_id’, ‘age’, ‘gender’, ‘location’, ‘MembershipLevel’, ‘TotalSpent’, ‘TotalPurchases’, ‘FavoriteCategory’, ‘LastPurchaseDate’, and ‘Churn’, to analyse the customer behaviour, we aggregate the datasets we obtained. ‘TotalPurchases’ is obtained through the number of transactions made by the customer. ‘MembershipLevel’ is determine based on the customer ‘TotalPurchase’. If it is more than 80% of the maximum TotalPurchase in the whole datasets, then it will be classified as Platinum. While it is more than 60%, it is classified as Gold. If it is more than 40%, it is classified as Silver. If it is more than 20%, it is classified as Bronze. ‘TotalSpent’ is the total amount spent by a customer in after adding up all the transactions. ‘FavoriteCategory’ is the category of item the customer bought the most, ‘LastPurchaseDate’ will be the last transaction date made by the customer and ‘Churn’ is to determine whether the customer is still active. This is obtained by comparing the last date in the datasets with the last date of the customer transaction. If the difference is more than 1 year, hence, it is defined as churned else he or she is still active.

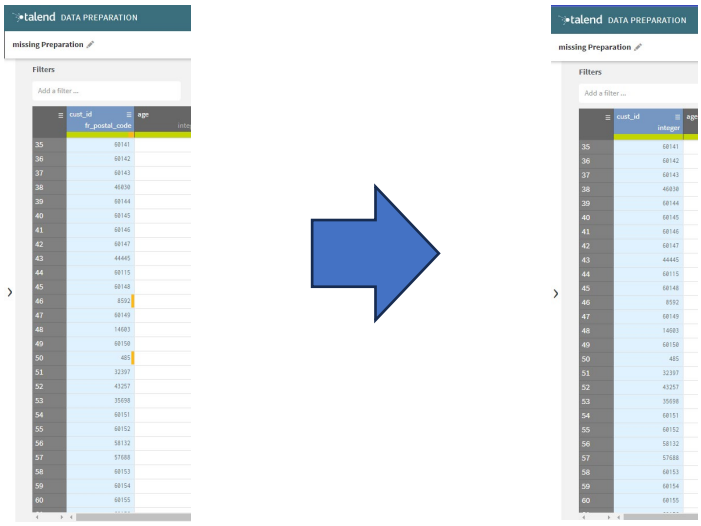
After aggregating the datasets, we obtained a dataset with 1021 rows with 12 features. Each row will represent the customer while the features are ‘cust_id’, ‘age’, ‘gender’, ‘location’, ‘MembershipLevel’, ‘TotalSpent’, ‘TotalPurchases’, ‘TotalDiscount’, ‘FavoriteCategory’, ‘FavoritePaymentMethod’, ‘LastPurchaseDate’, and ‘Churn’

cust_id	age	Gender	City	Location	Members	TotalPurcl	TotalSpen	TotalDisc	FavoriteC	FavoriteP	LastPurchaseDate	Churn
60124	43	F	Vinson	South	Bronze	7	4013.8	0	Men's Fas	cod	11/13/2020	1
42485	28	M	Graham	South	Silver	21	4718.14	855.96	Appliance	Easypay	12/28/2020	
53620	65	M	Grand For	Midwest	Bronze	5	4494.5	0	Appliance	cod	10/25/2020	1
56836	33	M	Laupahoe	West	Bronze	11	13084.1	0	Appliance	Easypay	10/24/2020	1
60125	73	F	Glendo	West	Bronze	4	1020	0	Men's Fas	Payaxis	2/10/2020	1
51286	64	F	Farmingtc	South	Bronze	1	3139.2	0	Mobiles &	Payaxis	1/10/2020	1
60126	75	F	Nashville	South	Bronze	3	780.3	0	Computin	cod	10/19/2020	1
60127	52	M	Warwick	Northeast	Bronze	2	3900	0	Home & Li	Payaxis	9/10/2020	1
60128	31	M	Sarasota	South	Bronze	1	35	0	Mobiles &	cod	1/10/2020	1
56449	71	F	Brownsto	Midwest	Gold	49	16175.2	0	Men's Fas	Payaxis	9/19/2021	0
60129	43	M	Mill Sprin	South	Bronze	1	29.9	0	Men's Fas	cod	1/10/2020	1
60130	66	F	Brighton	Midwest	Bronze	1	64.5	0	Mobiles &	cod	1/10/2020	1
31655	38	F	Phoenix	West	Gold	52	2516.23	97.97	Men's Fas	jazzwallet	12/4/2021	0
60131	71	M	Fort Smitl	South	Bronze	1		0	Women's	cod	1/10/2020	1
15106	64	M	Drew	South	Silver	26	11572	592	Women's	easypay_v	5/21/2021	0
37011	38	M	Snyder	Midwest	Silver	19	650.7	20	Soghaat	cod	6/29/2021	0
52163	60	M	Riverton	Northeast	Bronze	1	103.5	0	Appliance	cod	1/10/2020	1
60132	75	M	Huntingto	South	Bronze	4	474	0	Women's	Payaxis	1/10/2020	1
44511	18	M	Benton	South	Bronze	1	35	0	Men's Fas	cod	1/10/2020	1

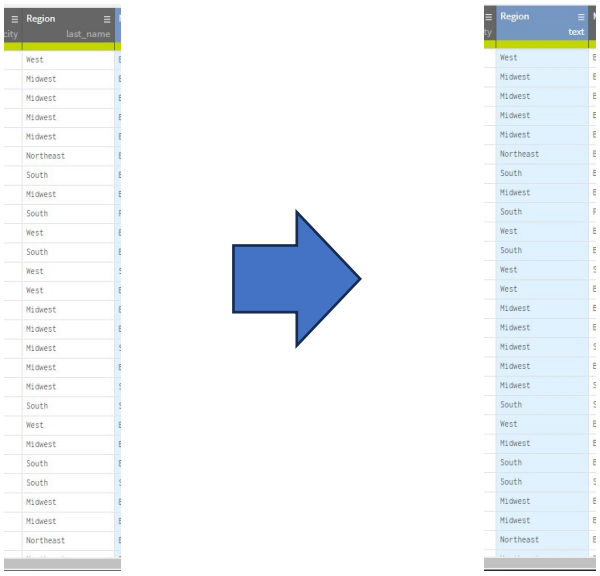
Data Import and Preprocessing

The datasets are imported into Talend Data Preparation for preprocessing.

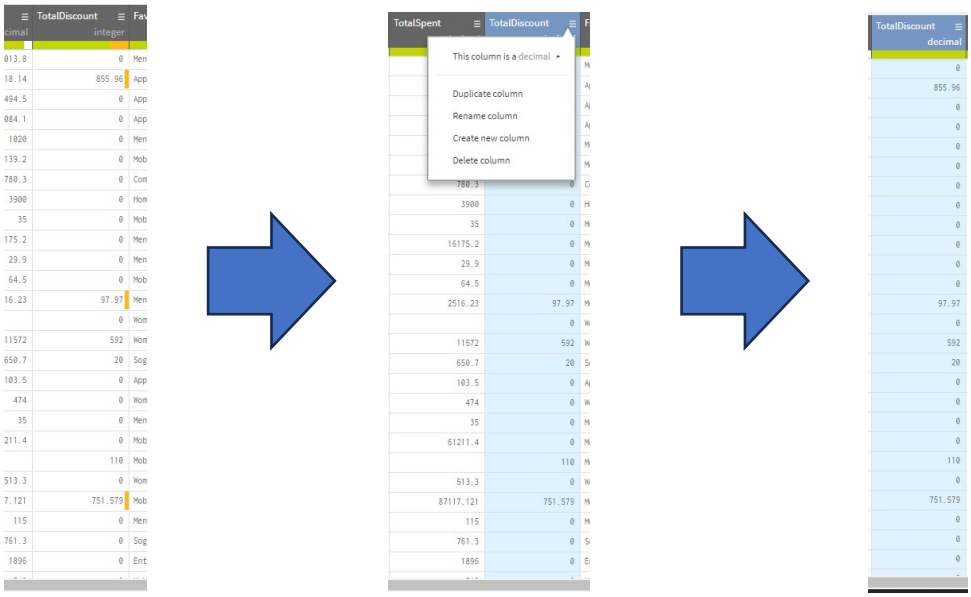
1. The cust_id shows it has 87 **invalid values in the column**. However, after inspection, all the are correct which is in integer and the only difference is that the invalid values are either in 3 digits or 4 digits number. This is due to the incorrect datatype set to the cust_id. The Talend Data Preparation detects the cust_id as postal code as majority of the values having 5 digits. Hence, we change the datatype from postal code to integer. The figure below shows the changes of the data type. In order to change the data type, we select the hamburger icon for more actions, hover over to 'This column is a fr_postal_code' and choose the data type wanted.



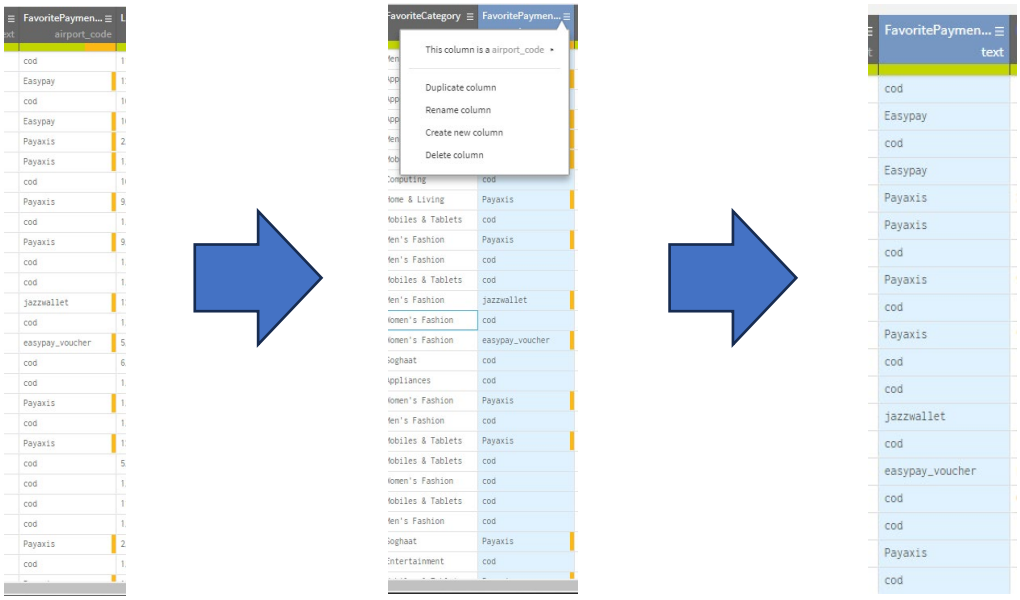
2. The **data type of region is detected as last_name**. Hence, we changed from the data type for region column from last_name to text. The figure below shows the changes of the data type. In order to change the data type, we select the hamburger icon for more actions, hover over to 'This column is a last_name' and choose the data type wanted.



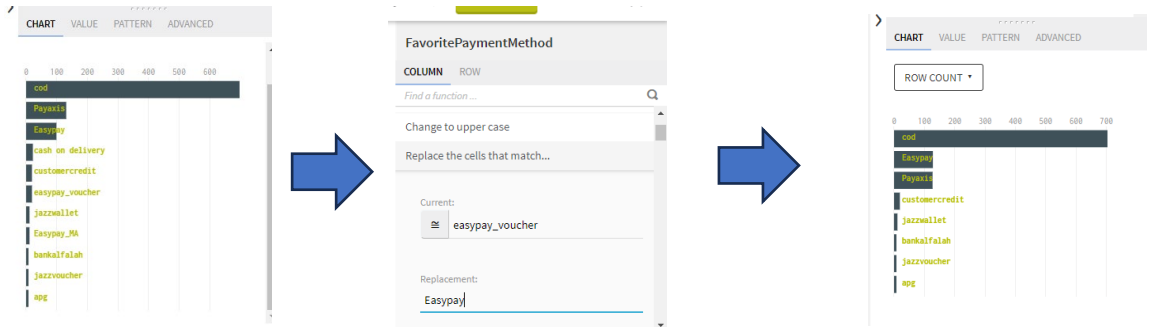
3. The TotalDiscount column also **shows invalid values in the column**. Inspecting through the column, as there are some of the values having decimal, hence, we change the data type of TotalDiscount from integer to decimal. The figure below shows the changes of the data type. In order to change the data type, we select the hamburger icon for more actions, hover over to 'This column is a decimal' and choose the data type wanted.



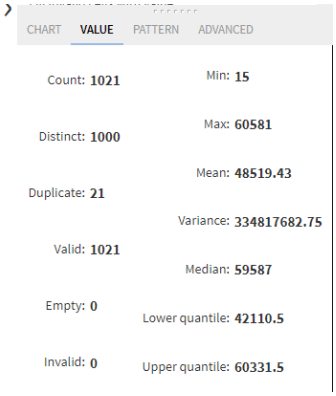
4. The FavoritePaymentMethod column **also having invalid values** as shown in the figure below. The data type detected for the column is airport_code. However, the value should be in text. Hence, we changed the data type from airport_code to text. The figure below shows the changes of the data type. In order to change the data type, we select the hamburger icon for more actions, hover over to 'This column is a airport_code' and choose the data type wanted.



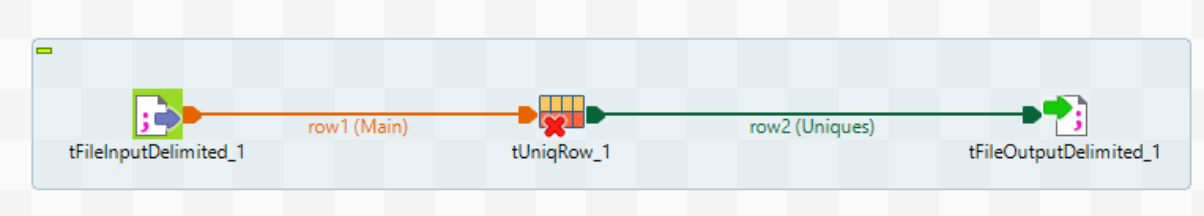
5. When inspecting through the 'FavoritePaymentMethod' column, we found that there is **inconsistency when filling the data**. This can be seen in the figure below where there are values for 'cod' and 'cash on delivery' which give the same meaning. 'Easypay_MA' and 'easypay_voucher' also belongs to the 'Easypay' category. Hence, we changed the value both the cases mentioned above using 'Replace the cell that match function' by inputting the value that we would like to change followed by the value we would change to. The figure below shows the steps in changing the value.



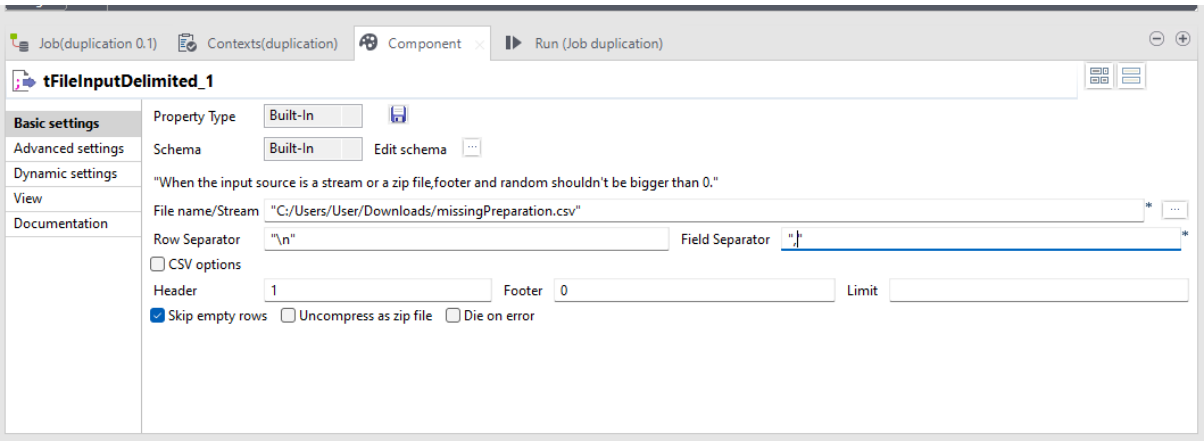
6. Inspecting over the cust_id which represents each customer in the datasets, we found that it has duplicate records. As one customer represents one row, hence, there should not be more than one row having the same cust_id value. The figure below shows the number of duplicates found in cust_id.



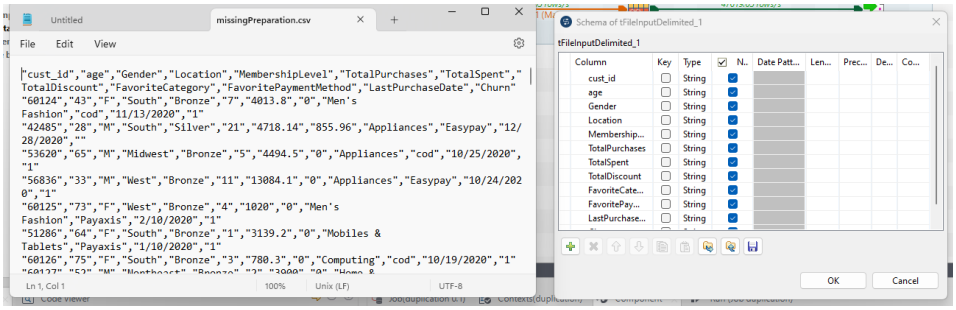
After the duplicates was found, we should **remove the duplicates** else it may result in overfitting when conducting the analysis. First, we tried to remove the duplicates in the Talend Data Preparation Free Desktop using the 'deduplicate rows with identical values function', however, this function is only available in subscription version of Talend Data Preparation 7.3 or 8.0. Hence, we exported the csv file to the Talend Open Studio for Data Integration. In Talend Open Studio for Data Integration, we first create a new project for this operation named AA1. Then we proceed to create a job named duplication by right-clicking on the Job Designs and hover over to create a job. Then it will prompt us for job name, purpose and description. Although providing the purpose and description are optional, but it is a good practice to provide as it helps us to understand easily what is the job created for. After creating the job, we proceed to develop the workflow by dragging the nodes needed. In this operation, the nodes needed are 'tFileInputDelimiter', 'tUniqRow' and 'tFileOutputDelimiter'. The workflow shows as the figure below.



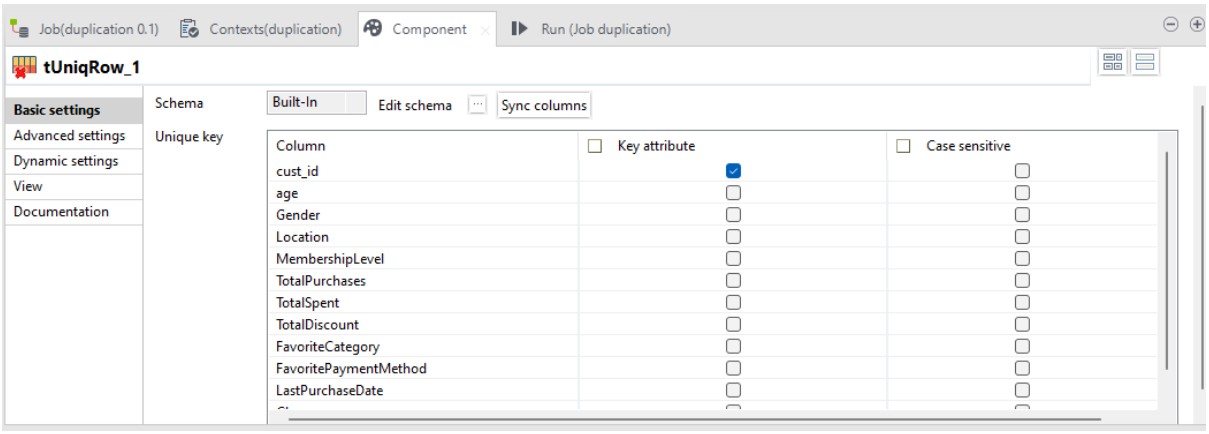
After connecting the nodes together, then we proceed to configure the nodes. In the tFileInputDelimiter, we have to configure the file path to the csv file, the field separator used, the row separator used, the header and lastly to add the schema. The figure below shows the configuration for the node.



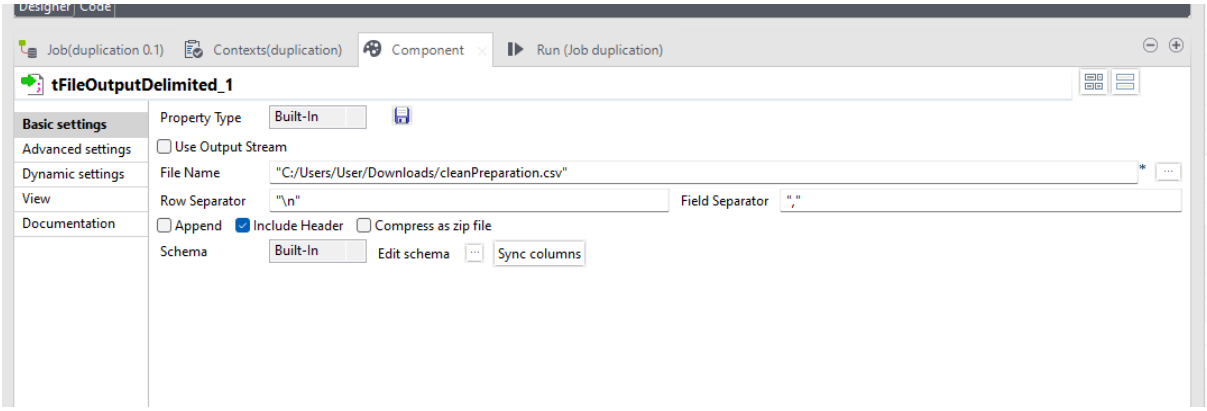
Based on the figure below, we can observe from the csv file exported from the Data Preparation for Desktop, it forces all the data type into string. Hence, when editing the schema, we insert all the columns according to the sequence and also set the data type to String else, the Talend Open Studio would not be able to run as the data unable to parse into the workflow.



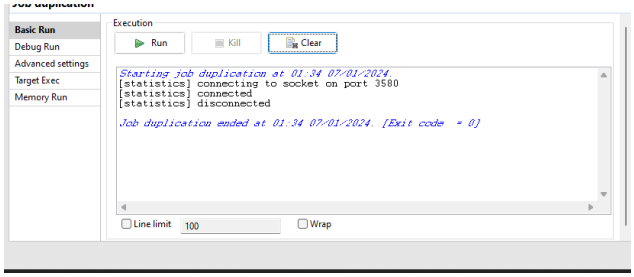
Next, we will be configuring the tUniqRow node, clicking the sync columns will allow the schema to be sync from the first node. For the unique key, we will be selecting only the cust_id, this is because each customer can only has one customer ID but customers can be having same gender, age, location, MembershipLevel, TotalPurchases, TotalSpent, TotalDiscount, FavoriteCategory, FavoritePaymentMethod, LastPurchaseDate and Churn.

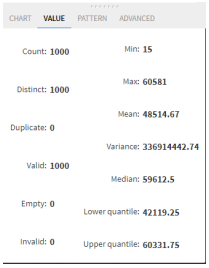


Proceed to the next node which is the tFileOutputDelimiter which is used to write the output to the csv file. In this node, we will be configuring the file name, row separator, field separator and check on the include Header. Before executing it, check the schema to match the schema before. The configurations can be seen in the figure below.

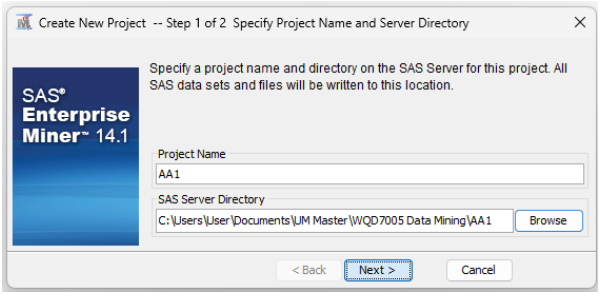


The job ended successfully as shown in the figure below. The missing value has been removed successfully.

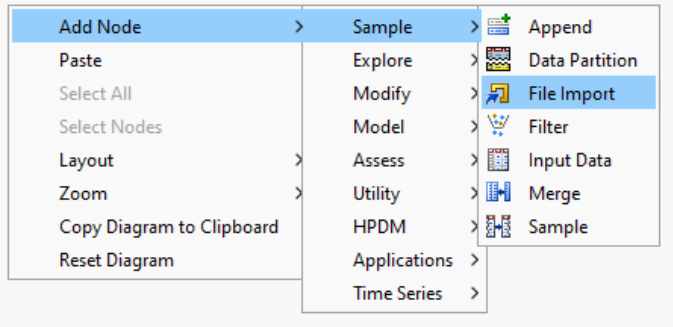




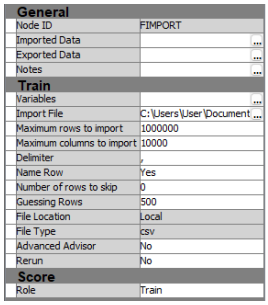
7. **Imputing missing values in SAS Enterprise Miner**. First, we created a new project for AA1, then proceed to create a diagram by right-clicking on the Diagrams on the upper left and hover over to Create Diagram.

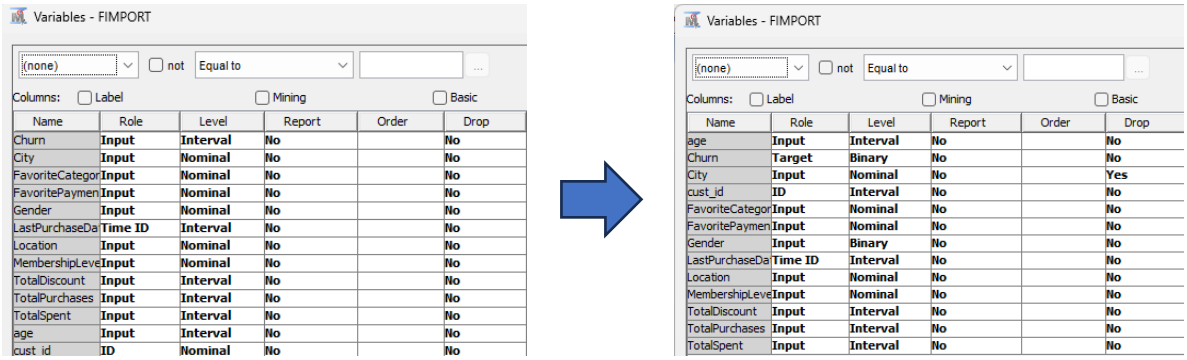


After done creating a diagram, in the diagram, we added the file import node by right-clicking the diagram space and hover to add node and choose from the sample to look for File Import node to import the csv file into SAS Enterprise Miner. The steps are show in the figure below.

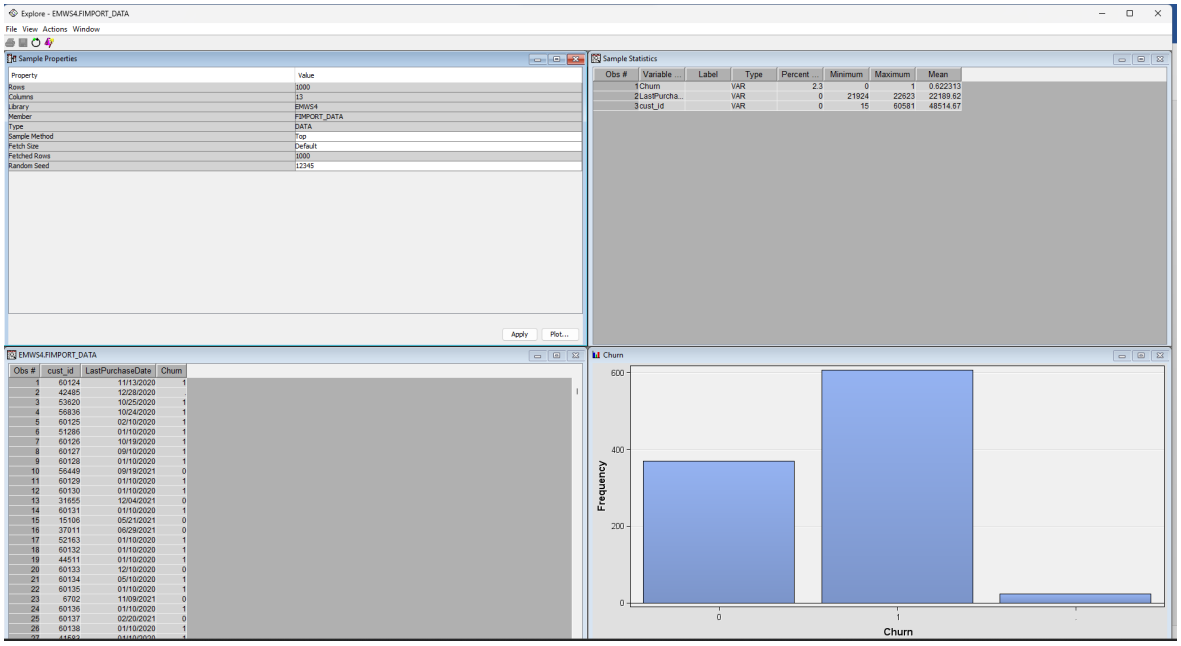


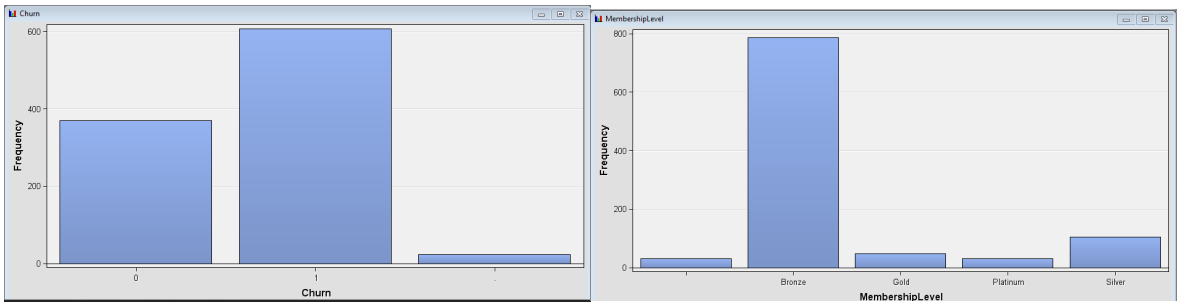
After adding the node, we will configure the settings before importing the datasets as shown in figure below. We inserted the path to the csv file in the Import File and ensure the file type to be the csv, check the delimiter to follow the delimiter that set previously and put yes for the Name row for reading the header. Then we proceed to edit the variables' role and level before executing the node.

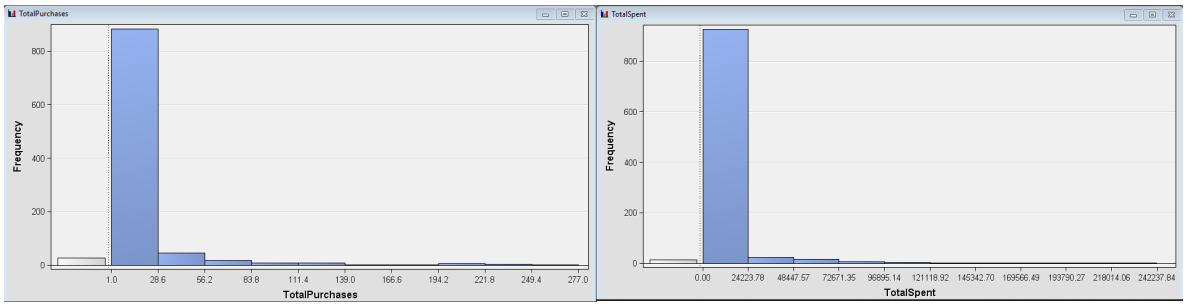




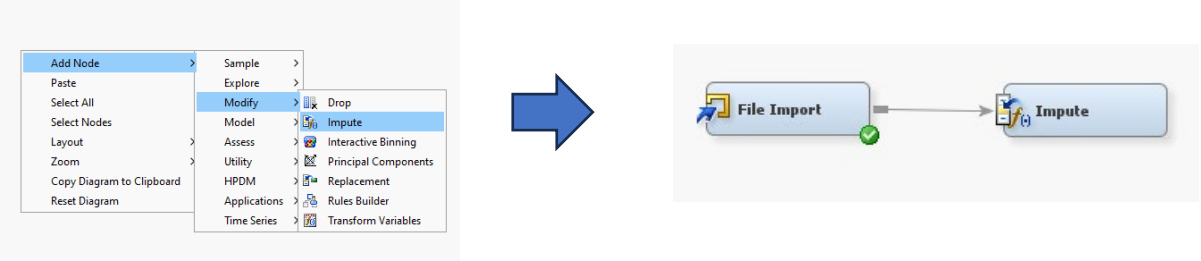
Exploring the variable statistics one by one, missing values in the variable are identified as shown in figures below. There are missing values in 4 features which are 'Churn', 'MembershipLevel', 'TotalPurchases' and 'TotalSpent' which can be seen in the bar chart of each statistics report. For nominal and binary features, the missing values are represented by the bar without the label. While for the interval features, the missing values are represented by white column in the bar chart.



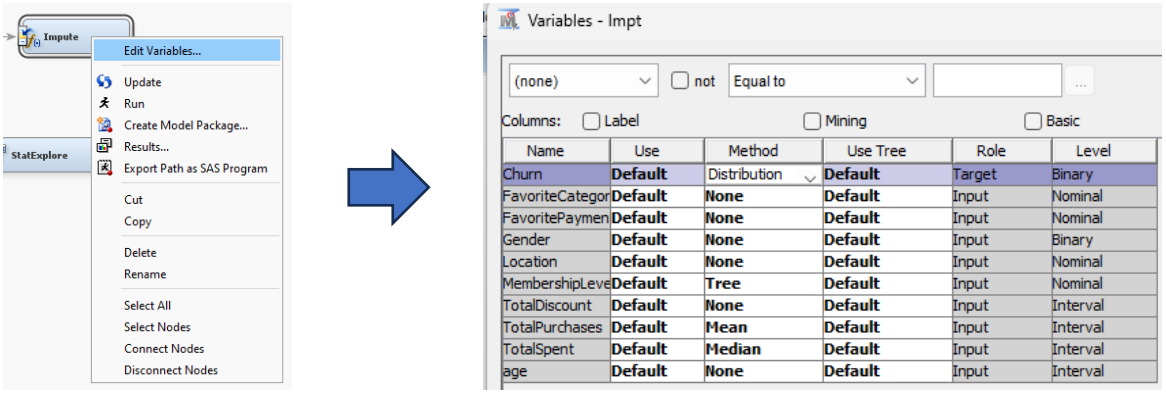




To impute the missing values for the 4 features mentioned, we added thee impute node by right-clicking in empty space of diagram and hover over to add node followed by modify and look for the impute. After adding the node, we will connect the impute node to the file import node to allow the node to access the data.



Before imputing for the missing values, we have to inspect and study the statistics reports of the features that contain missing values. The reports can serve as an aiding tool in assisting in making decision in selecting the imputation method of each feature. For nominal or binary target feature, there are 3 different methods in imputing the missing values which are count, default constant value and distributions. The count method is to impute the missing value by giving it the most frequently occurrence target variable value. For the default constant value, this is to replace the missing value using the default constant value that we set while lastly the distribution method will take consideration of the probability distribution of non-missing observations. For the target feature which is the 'Churn' feature that is created by comparing the difference of the last purchase date with the last transactions date in the datasets, we should use the distribution method to impute the value. This is because replacing the value using count or default constant value may face the bias results during the analysis. Imputing based on the non-missing observations would reflect more towards the customer personal behaviour. Hence, the target feature will be imputed using distributions. The steps are shown in the figures below where we right-click impute node and select edit variables. Then at the method of Churn, we select Distribution.



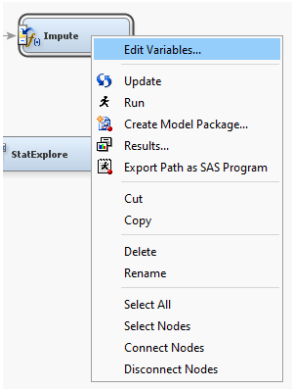
Name	Use	Method	Use Tree	Role	Level
Churn	Default	Distribution	Default	Target	Binary
FavoriteCategor	Default	None	Default	Input	Nominal
FavoritePaymen	Default	None	Default	Input	Nominal
Gender	Default	None	Default	Input	Binary
Location	Default	None	Default	Input	Nominal
MembershipLeve	Default	Tree	Default	Input	Nominal
TotalDiscount	Default	None	Default	Input	Interval
TotalPurchases	Default	Mean	Default	Input	Interval
TotalSpent	Default	Median	Default	Input	Interval
age	Default	None	Default	Input	Interval

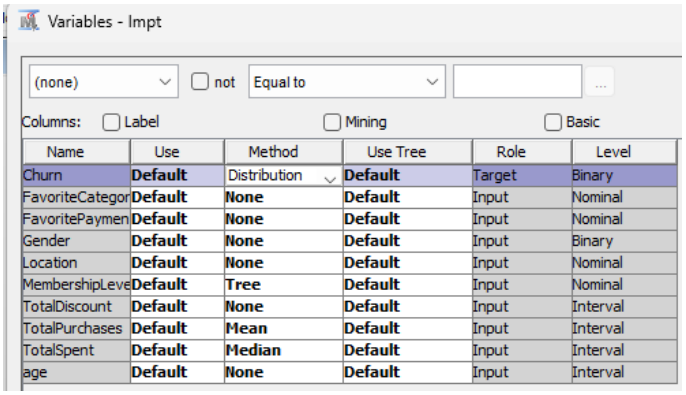
For input class variables like MembershipLevel, there are several ways to impute the missing values which are count, default constant value, distribution, tree and tree surrogate. For the count, default constant value and distribution, they work just like mentioned in previous part. While for the tree, it uses the remaining input and rejected variables as predictors to estimate the replacement value. For the tree surrogate, it uses the same algorithm with the Tree imputation. The only difference is that it has an addition of splitting rule in which the rule is used as a back-up to the main splitting rule. When the main rule is prevented, the next surrogate is invoked. From the figure below, it shows the mode of MembershipLevel is Bronze. Replacing the missing values with the count does not truly reflect the real situation as the membership was given when the customer achieved certain level of spending. For imputing with default constant value, we did not take in consideration because it may raise bias situation. For the distribution method, it depends on the probability distribution of the non-missing observation while compared with the tree imputation, they estimate the value using remaining input and rejected

variables as the predictors. The tree inherently captures interactions between variables. If there are interactions or combination of features that influence the 'MembershipLevel', the tree method might be more effective. The steps are shown in the figures below where we select edit variables, and at the row for MembershipLevel, we select Tree for the method column.

Sample Statistics

Obs #	Variable ...	Label	Type	Percent ...	Minimum	Maximum	Mean	Number o...	Mode Per...	Mode
1	Membershi...		CLASS	3.1	.	.	.5		78.6	BRONZE
2	Churn		VAR	2.3	0	1	0.622313	.		
3	cust_id		VAR	0	15	60581	48514.67	.		





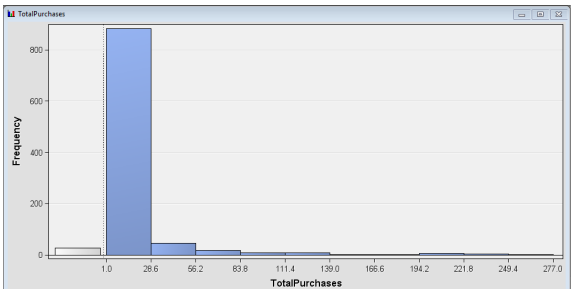
Name	Use	Method	Use Tree	Role	Level
Churn	Default	Distribution	Default	Target	Binary
FavoriteCategor	Default	None	Default	Input	Nominal
FavoritePaymen	Default	None	Default	Input	Nominal
Gender	Default	None	Default	Input	Binary
Location	Default	None	Default	Input	Nominal
MembershipLeve	Default	Tree	Default	Input	Nominal
TotalDiscount	Default	None	Default	Input	Interval
TotalPurchases	Default	Mean	Default	Input	Interval
TotalSpent	Default	Median	Default	Input	Interval
age	Default	None	Default	Input	Interval

For imputing the interval variables, the imputation method that are taking into considerations are mean and median. Before deciding which to be selected to use, we explore the statistics report of the variables. Below the figures shows the min, max, mean, distributions of data, and boxplot for the TotalPurchases and TotalSpent. Both the mean of the variables fell in the highest frequency distribution group. Then we further explore the boxplot for both the variables. Due to the outliers in both features, median is a better choice for imputation as mean will be affected by the outliers. From the boxplot, we are able to observed that if we replace the missing value with mean, it will fall outside of the boxplot. The distribution for both TotalPurchases and TotalSpent skewed to the right.

TotalPurchases:

Sample Statistics

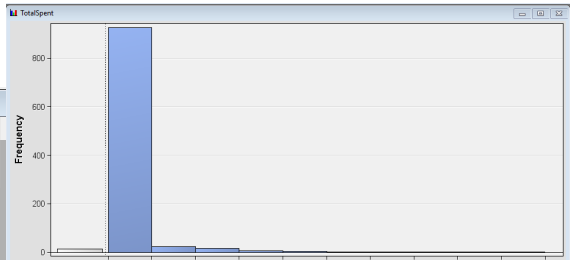
Obs #	Variable ...	Label	Type	Percent ...	Minimum	Maximum	Mean
1	Churn		VAR	2.3	0	1	0.622313
2	TotalPurch...		VAR	2.6	1	277	11.923
3	cust_id		VAR	0	15	60581	48514.67



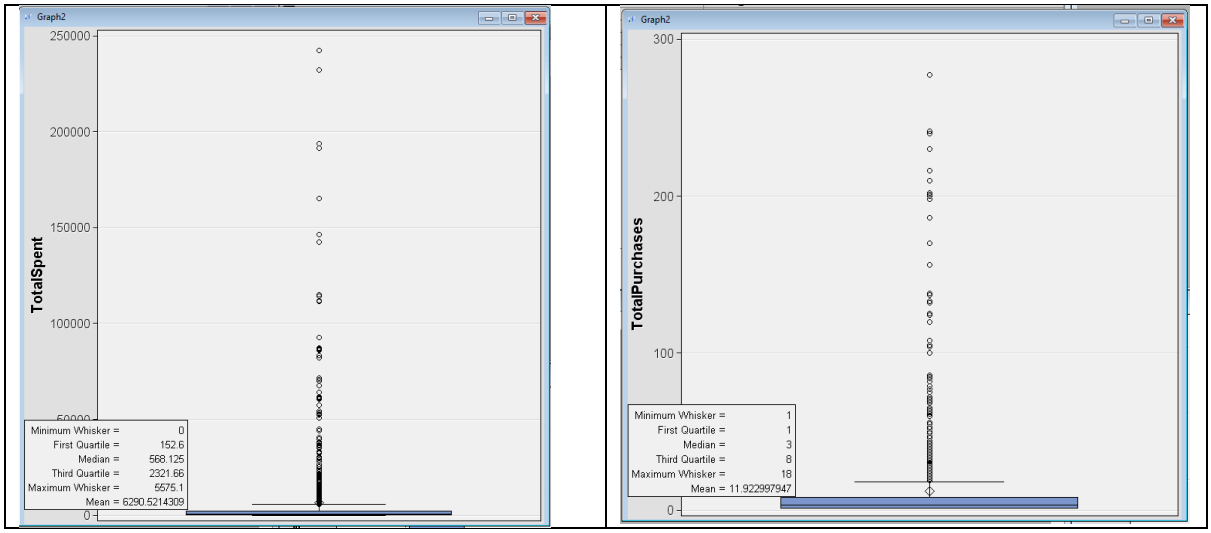
TotalSpent:

Sample Statistics

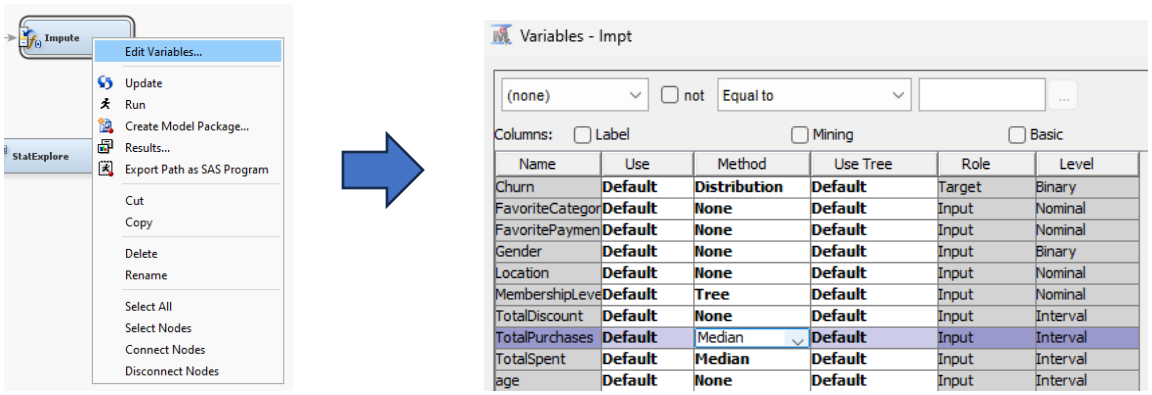
Obs #	Variable ...	Label	Type	Percent ...	Minimum	Maximum	Mean
1	Churn		VAR	2.3	0	1	0.622313
2	TotalSpent		VAR	1.3	0	242237.8	6290.521
3	cust_id		VAR	0	15	60581	48514.67



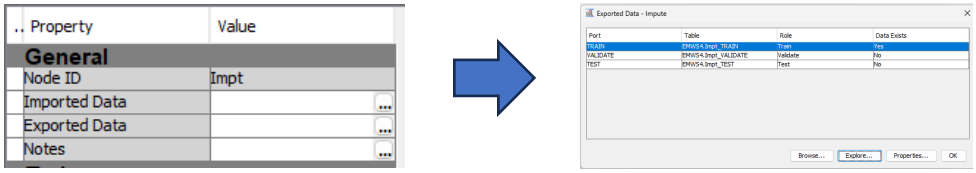
TotalPurchases Boxplot	TotalSpent Boxplot



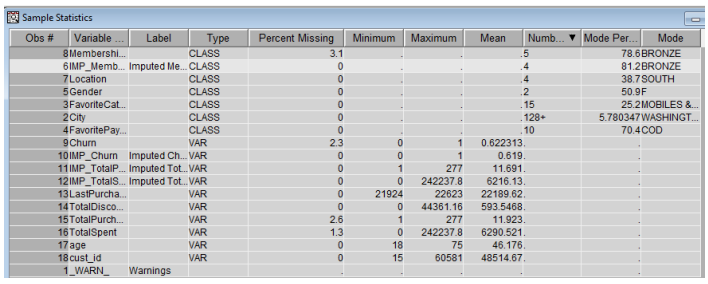
The steps to select median for both TotalPurchases and TotalSpent can be seen in the figures below. For both the variables, the imputation methods are set to median.



After done configuring the imputation method, we confirm our choices and proceed to run the node. When successfully executed the node, we are able to view the results as shown in the figure below. We selected the exported data on the left, and it will pop a window prompt us to choose between train, validate and test. In this case, we select train to explore. It creates 4 new columns which are started with 'IMP_'. From the 4 columns starting from the 'IMP_', we are able to observe that the percent missing is 0. We have successfully imputed for the missing values.



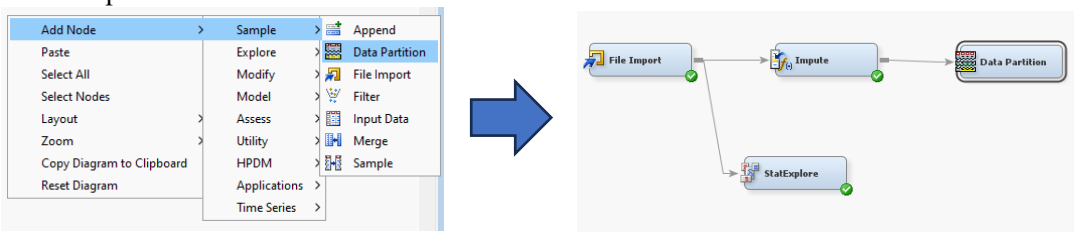
Results:



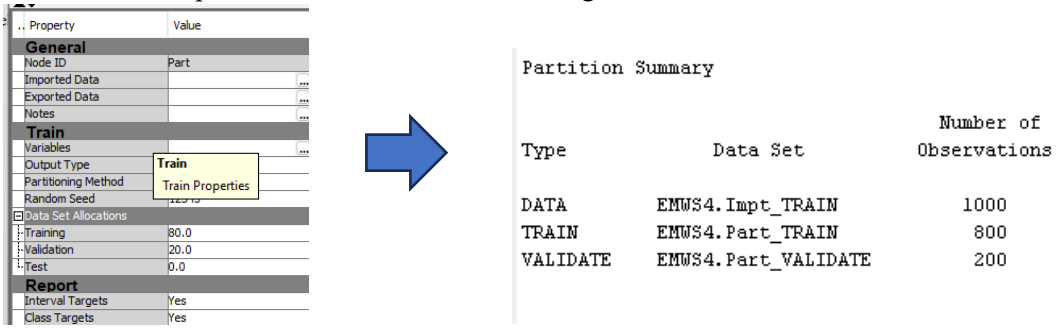
Decision Tree Analysis

1. Data Partition

Before conducting decision tree analysis, we split the data into 80% training sets and 20% validation sets. This is achieved by using the data partition node. The node can be added by right-clicking and hover to add node followed by sample and select the data partition node. The steps are shown in the figures below. After adding the node, we connect the data partition node to the impute node.

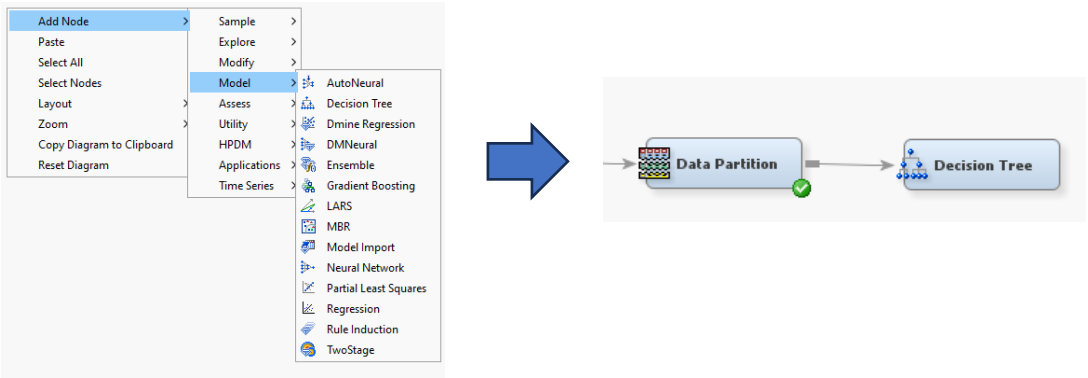


After connecting the node to the workflow, we click on the data partition node for configurations. The setting can be configured on the left, we change the percentage of data set allocations where training set will be having 80% of the dataset while validation set will be having 20% of the dataset. After the configuration, we proceed to execute the node. Then we proceed to view the results, from the results at the partition summary part, we are able to observe that whole dataset having 1000 observations has been split into 800 for train and 200 for validate. The steps and results are shown in the figures below.



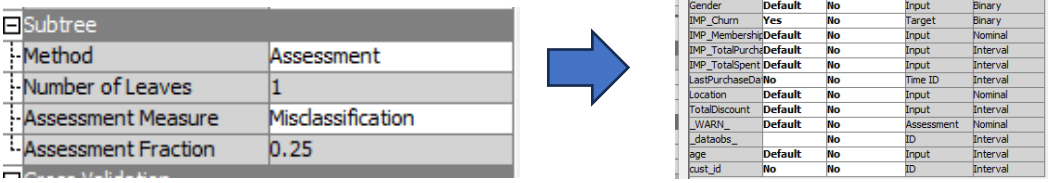
2. Optimal Decision Tree

After done splitting the data, we proceed to building the decision tree. First, we right-click on the diagram and hover over to add node followed by model and look for the decision tree. After added the decision tree node, the node is connected to the Data Partition node.

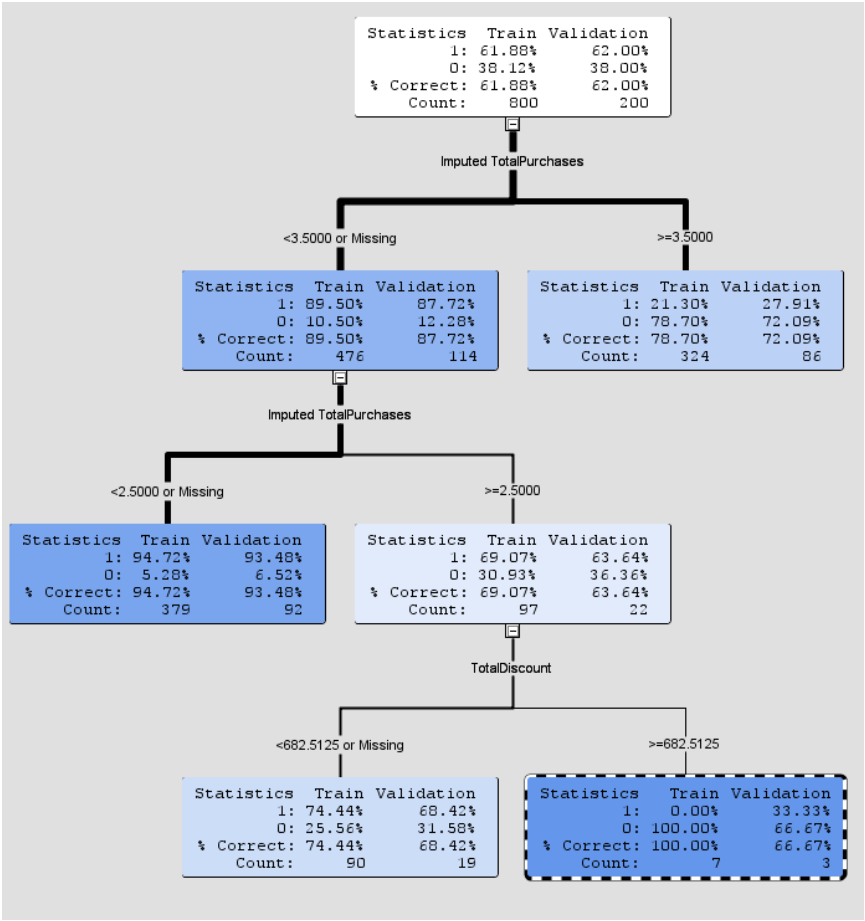


After connecting the node, we proceed to configure the settings in the decision tree node's properties panel. Change the Assessment measure to misclassification. As we are building the decision tree autonomously at first, other settings just remain as default and execute the node. The figures below show the steps to configure the settings. We also check back the variables that we are using, for those variables that does not provide any meaning to the analysis can be

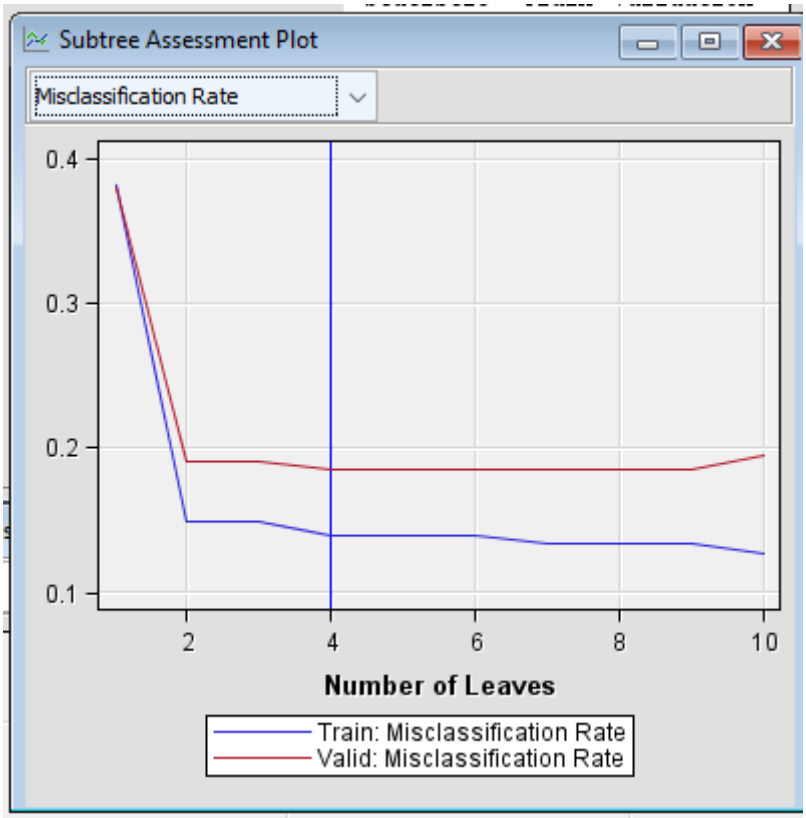
dropped. The steps are done in the figure below. Under the use column for the variables window, we choose the variable that we not going to use and select the choice to No. The LastPurchaseDate and cust_id was placed to No as the LastPurchaseDate is used to generate the Churn which is the target of the variables. It may result in bias situation and including of the LastPurchaseDate may lead to overfitting. For the cust_id, it did not give any meaning to the analysis, hence both the variables are not included in the analysis.



Then we proceed to executing the node. The execution will provide us the optimal decision tree where the tree has 2 branches and 4 leaf nodes. The tree is shown in the figure below.



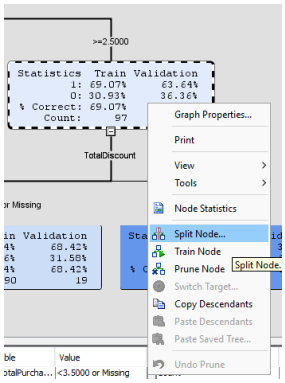
The figure below shows the subtree assessment plot of the decision tree. It shows the misclassification rate of the decision tree when increasing of the number of leaves. The vertical line serves as the noteworthy indicator where the performance of the decision tree is at optimal. Increasing the number of leaves may not decrease the misclassification rate much. The vertical line has pointed at 4 which indicates that 4 is the optimal number of leaves for the decision tree.

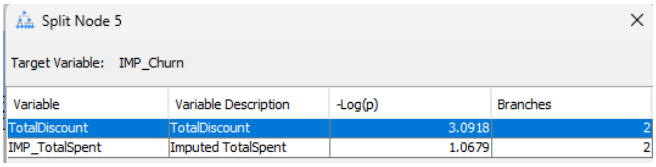


The figure below shows the importance of the variable used in the decision tree. In the decision tree, we used two variables which are IMP_TotalPurchases and TotalDiscount. The importance of the IMP_TotalPurchases is 1.0000 while the importance of the TotalDiscount is 0.1949. Further verification can be done by selecting the interactive at the decision tree node's properties panel. Right-clicking at the leaf right before the TotalDiscount was used, and hover to split node. Then a split node window will appear and shows the variables with the highest importance level in sequence. The steps are shown in the figures below. We are able to observe that the TotalDisocunt has higher -Log(p) value compared to IMP_TotalSpent. Hence, TotalDiscount was selected. From the decision tree created, we are able to induce that these 2 features influence the spending behaviour of the customer. Whether the customer will continue to purchase through e-commerce strongly depends on their past spending habit and also the discount provided by the seller.

Variable Importance

Variable Name	Label	Number of Splitting Rules	Importance	Validation Importance	Ratio of Validation to Training Importance
IMP_TotalPurchases	Imputed TotalPurchases	2	1.0000	1.0000	1.0000
TotalDiscount		1	0.1949	0.0000	0.0000





The figure below shows the rule node of the decision tree. Based on the figure below, for rule node 3, we are able to interpret that this node identifies a segment of customers whose imputed total purchases are relatively high. The model predicts a low probability of churned customers for this group. For rule node 4, this node captures the customers with low imputed total purchases. The model predicts a high probability of churned customer for this group. The rule node 10 further refines the conditions by considering the total discount. It predicts a moderate probability of churn for customers with specific total purchase and discount characteristics. Lastly, the rule node 11 represents a group of customers with specific total purchase and discount characteristics that indicate a very low probability of churn. These rule nodes help to segment the dataset based on the features selected and provide meaningful insights into the factor influencing the predicted churn probability for different customer groups. This can assist the seller to derive actionable insights for customer retention strategies.

```
*------------------------------------------------------------*
 Node = 3
*------------------------------------------------------------*
if Imputed TotalPurchases >= 3.5
then
 Tree Node Identifier   = 3
 Number of Observations = 324
 Predicted: IMP_Churn=1 = 0.21
 Predicted: IMP_Churn=0 = 0.79

*------------------------------------------------------------*
 Node = 4
*------------------------------------------------------------*
if Imputed TotalPurchases < 2.5 or MISSING
then
 Tree Node Identifier   = 4
 Number of Observations = 379
 Predicted: IMP_Churn=1 = 0.95
 Predicted: IMP_Churn=0 = 0.05

*------------------------------------------------------------*
 Node = 10
*------------------------------------------------------------*
if TotalDiscount < 682.513 or MISSING
AND Imputed TotalPurchases < 3.5 AND Imputed TotalPurchases >= 2.5
then
 Tree Node Identifier   = 10
 Number of Observations = 90
 Predicted: IMP_Churn=1 = 0.74
 Predicted: IMP_Churn=0 = 0.26

*------------------------------------------------------------*
 Node = 11
*------------------------------------------------------------*
if TotalDiscount >= 682.513
AND Imputed TotalPurchases < 3.5 AND Imputed TotalPurchases >= 2.5
then
 Tree Node Identifier   = 11
 Number of Observations = 7
 Predicted: IMP_Churn=1 = 0.00
 Predicted: IMP_Churn=0 = 1.00
```

From the fit statistics show in the figure below, the optimal decision tree built has a training misclassification rate of 0.14 and validation misclassification rate of 0.185.

Fit Statistics

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
IMP_Churn	Imputed Churn	_NOBS_	Sum of Frequencies	800	200	.
IMP_Churn	Imputed Churn	_MISC_	Misclassification Rate	0.14	0.185	.
IMP_Churn	Imputed Churn	_MAX_	Maximum Absolute Err...	0.94723	1	.
IMP_Churn	Imputed Churn	_SSE_	Sum of Squared Errors	180.7447	56.9506	.
IMP_Churn	Imputed Churn	_ASE_	Average Squared Error	0.112965	0.142377	.
IMP_Churn	Imputed Churn	_RASE_	Root Average Squared...	0.336103	0.377328	.
IMP_Churn	Imputed Churn	_DIV_	Divisor for ASE	1600	400	.
IMP_Churn	Imputed Churn	_DFT_	Total Degrees of Free...	800	.	.

3. Underfitting Decision Tree

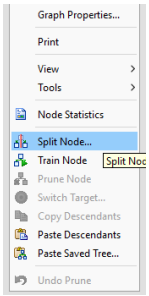
After the autonomous decision tree, we proceed with interactive decision tree to depict the underfitting situation. To create the underfitting situation, we configure the number of leaves to be less than the optimal number obtained in previous section. We added another decision tree node and rename the node to underfitting tree.

At the Interactive and edit by clicking the '…' icon. It displays a window that shows the interactive decision tree in which allowing us to add the leaf node ourselves. The figures below show the steps.

General	
Node ID	Tree2
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Interactive	
Import Tree Model	No
Tree Model Data Set	
Use Frozen Tree	No
Use Multiple Targets	No

Statistics	Train	Validation
1:	61.88%	62.00%
0:	38.12%	38.00%
% Correct:	61.88%	62.00%
Count:	800	200

Then we proceed to split the node by right-clicking at the root node and select split node option. It pops a split node window allowing us to choose the variable that we would like to use. The variables are shown in a sequence from the most importance to the least. Then by selecting the variable we want and apply it added the node automatically. This process is repeated several times to create an underfitting decision tree. To have an underfitting tree, we should ensure the number of leaves to be less than 4.

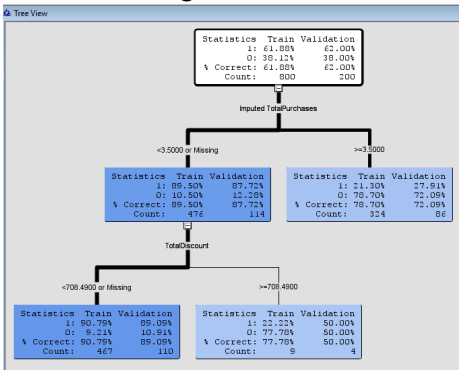


Split Node 1

Target Variable: IMP_Churn

Variable	Variable Description	-Log(p)	Branches
IMP_TotalPurchases	Imputed TotalPurchases	82.5847	2
IMP_MembershipLevel	Imputed MembershipLevel	44.6701	2
IMP_TotalSpent	Imputed TotalSpent	38.5718	2
TotalDiscount	TotalDiscount	34.0824	2
FavoritePaymentMethod	FavoritePaymentMethod	10.4164	2
Location	Location	1.0299	2
Gender	Gender	0.1641	2
FavoriteCategory	FavoriteCategory	0.0933	2
age	age	0.0	2

The underfitting tree is created as the figure below.



An underfitting tree is too simple to capture the underlying patterns in the data. In the decision tree built above, it only uses the 'Imputed Total Purchases' as the predictor. This causes the lack of information capture in the decision tree. From rule node 3, the node represents a split based on the total purchases. If the imputed total purchases are below 3.5 or missing, the tree follows the branch to Node 3. Rule node 4 represents the alternative split. Rule node 28 introduces an additional condition related to total discount. It further splits the subset of data where total purchases are less than 3.5 or missing. Lastly, the node 29 represents another split based on conditions related to total discount. From the interpretation of the rule node, the tree seems to be reasonably simple with a limited number of conditions, which contributes to an underfitting model. The conditions focus on 'Imputed TotalPurchases' and 'TotalDiscount,' but there could be room for more complexity or additional features to capture more patterns.

In order to address this underfitting issue, we can evaluate the model's performance on a validation dataset to ensure it captures meaningful patterns, else, consider adding more features or adjusting the parameters to allow the model to capture more complexity if needed.

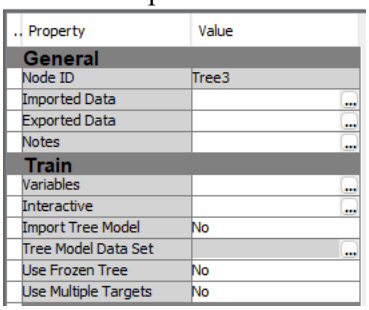
The figure below shows the misclassification rate of the underfitting tree. For the train dataset, its misclassification rate is 0.1425 while for the validation set is 0.19 which is higher compared to the optimal decision tree.

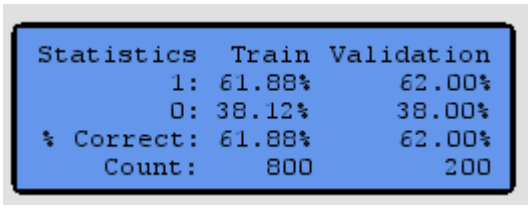
Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
IMP_Churn	Imputed Churn	_NOBS_	Sum of Frequencies	800	200	.
IMP_Churn	Imputed Churn	_MISC_	Misclassification Rate	0.1425	0.19	.
IMP_Churn	Imputed Churn	_MAX_	Maximum Absolute Err...	0.907923	0.907923	.
IMP_Churn	Imputed Churn	_SSE_	Sum of Squared Errors	189.8036	59.4191	.
IMP_Churn	Imputed Churn	_ASE_	Average Squared Error	0.118627	0.148548	.
IMP_Churn	Imputed Churn	_RASE_	Root Average Squared...	0.344423	0.385419	.
IMP_Churn	Imputed Churn	_DIV_	Divisor for ASE	1600	400	.
IMP_Churn	Imputed Churn	_DFT_	Total Degrees of Free...	800	.	.

4. Overfitting Decision Tree

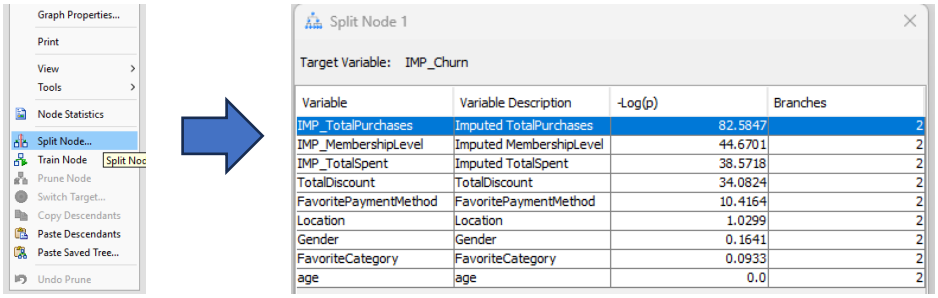
After the underfitting decision tree, we proceed with interactive decision tree to depict the underfitting situation. To create the underfitting situation, we configure the number of leaves to be more than the optimal number obtained in previous section. We added another decision tree node and rename the node to overfitting tree.

At the Interactive and edit by clicking the '…' icon. It displays a window that shows the interactive decision tree in which allowing us to add the leaf node ourselves. The figures below show the steps.



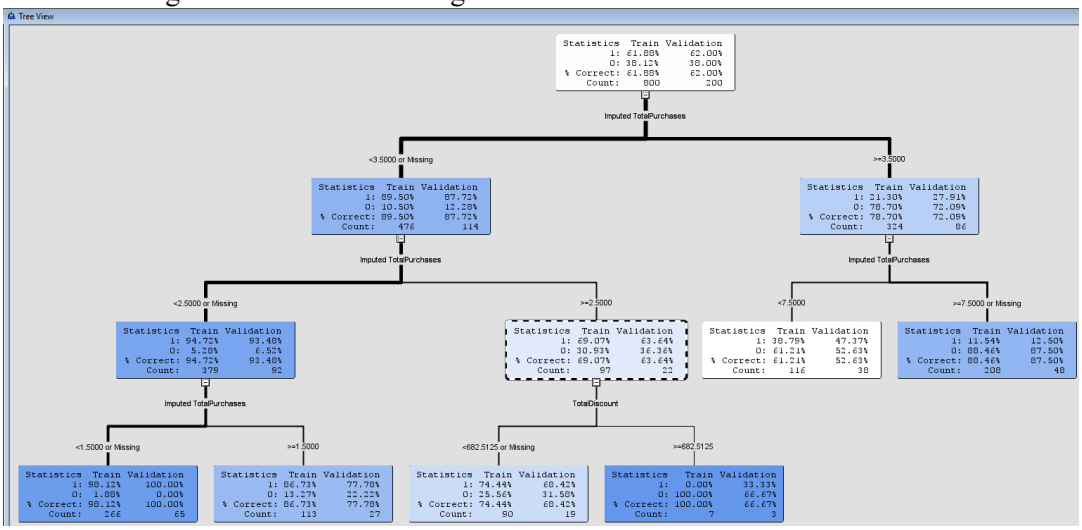


Then we proceed to split the node by right-clicking at the root node and select split node option. It pops a split node window allowing us to choose the variable that we would like to use. The variables are shown in a sequence from the most importance to the least. Then by selecting the variable we want and apply it added the node automatically. This process is repeated several times to create an underfitting decision tree. To have an overfitting tree, we should ensure the number of leaves to be more than 4.



Variable	Variable Description	-Log(p)	Branches
IMP_TotalPurchases	Imputed TotalPurchases	82.5847	2
IMP_MembershipLevel	Imputed MembershipLevel	44.6701	2
IMP_TotalSpent	Imputed TotalSpent	38.5718	2
TotalDiscount	TotalDiscount	34.0824	2
FavoritePaymentMethod	FavoritePaymentMethod	10.4164	2
Location	Location	1.0299	2
Gender	Gender	0.1641	2
FavoriteCategory	FavoriteCategory	0.0933	2
age	age	0.0	2

The overfitting tree is created as the figure below.



The decision tree created has multiple nodes with various conditions, indicating potential overfitting due to the complexity of the model. Rule node 6 represents a split based on a narrow range of total purchases. It predicts a relatively high churn probability for the observations falling into this category. Rule node 7 introduces another split based on the wide range of total purchases. The predicted churn probability is lower compared to Node 6. Rule node 8 represents a split for high values of total purchases or missing values. It predicts a very low churn probability for this segment. Rule node 9 introduce a split for very low values of total purchases or missing values. It predicts a very high churn probability for this segment. The rule node 10 represents another split within a specific range of total purchases that also predicts a high churn probability for this segment. The tree appears to be capturing very specific ranges of total purchases, leading to high granularity. The high churn probabilities in some nodes may be indicative of overfitting, especially if these patterns do not generalize well to new, unseen data. The feature used in this decision tree is only TotalPurchase, this leads to overfitting as it captures the noise or variability present in the training data. Besides, the tree tends to memorise the training set and leads to poor generalisation on new data which works poorly in foreseen future market trends.

To address the overfitting issue, consider simplifying the tree by pruning or reducing the maximum depth to avoid overfitting. Besides, evaluates the model performance on a validation dataset to ensure it generalises well else it is not reliable. The model is unable to assist in decision-making as it does not truly reflect the real situation. Lastly, we can experiment with different hyperparameters to find a more balanced and generalised model.

The overfitting decision tree created fit statistics is shown as below. The misclassification rate of train dataset is 0.14875 and validation set is 0.19 which is slightly higher than the optimal decision tree.

Fit Statistics

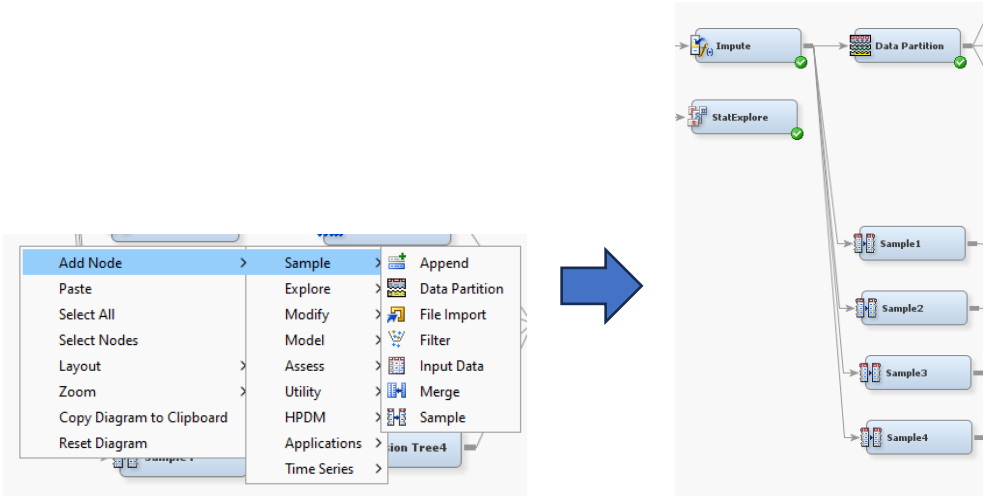
Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
IMP_Churn	Imputed Churn	_NOBS_	Sum of Frequencies	800	200	.
IMP_Churn	Imputed Churn	_MISC_	Misclassification Rate	0.14875	0.19	.
IMP_Churn	Imputed Churn	_MAX_	Maximum Absolute Err...	0.981203	0.884615	.
IMP_Churn	Imputed Churn	_SSE_	Sum of Squared Errors	174.8208	50.13856	.
IMP_Churn	Imputed Churn	_ASE_	Average Squared Error	0.109263	0.125346	.
IMP_Churn	Imputed Churn	_RASE_	Root Average Squared...	0.33055	0.354043	.
IMP_Churn	Imputed Churn	_DIV_	Divisor for ASE	1600	400	.
IMP_Churn	Imputed Churn	_DFT_	Total Degrees of Free...	800	.	.

Ensemble Methods

There are two ensemble methods which are bagging and boosting. For this case study, we will be using random forest algorithm for the bagging and gradient boosting for the boosting.

1. **Random Forest Algorithm**

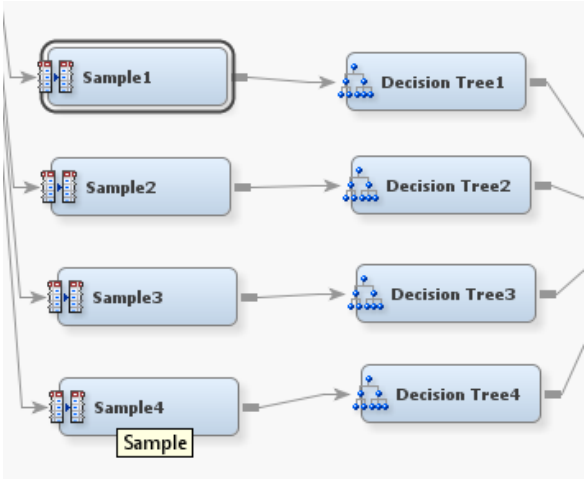
   To create the random forest model, we will be using sample node, decision tree node and ensemble node. The steps to add the sample node are shown in the figure below. We created 4 sample nodes in which each sample node will be taking 25% of the data adding up to the whole dataset. Connect all the nodes to the impute node to access the data.



After connecting the node, we proceed to configure the property setting in the left panel. The figure below shows the setting for one of the samples. The percentage size of the node is set to 25. This step is repeated for all other sample nodes. The other samples nodes will be repeating the same step but modifying the random seed number. This is to ensure that the sample data that generated by each sample node is different else it will lead to bias and overfitting situation.

General	
Node ID	Smpl
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
Output Type	Data
Sample Method	Default
Random Seed	12345
Size	
Type	Percentage
Observations	.
Percentage	25.0
Alpha	0.01
PValue	0.01
Cluster Method	Random

After the sample node, we will be adding for decision tree node. The decision trees are added the same way as we done previously. Each decision tree is then connected to the sample node respectively as shown in the figure below.



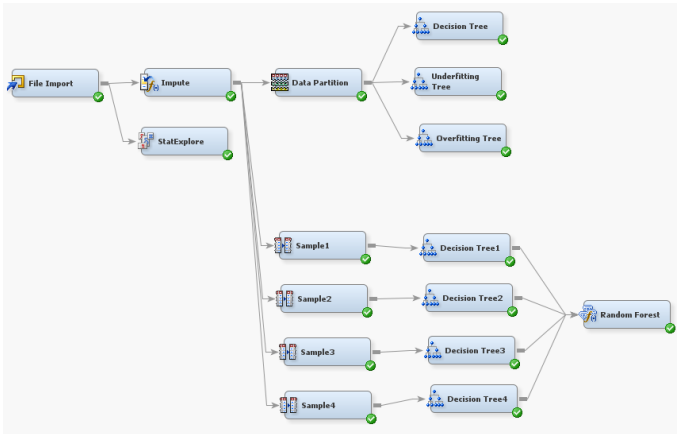
The property setting for all the decision tree are the same. We just modified the subtree assessment measure to misclassification. Other settings are left as default. The variables in each decision tree are checked to match with the variables in the optimal decision tree.

Subtree	
Method	Assessment
Number of Leaves	1
Assessment Measure	Misclassification

After done configuring the decision tree node, proceed to add the ensemble node. The ensemble node can be found in the model tab. We connected all the decision trees to the ensemble node. Then we will configure the property of the ensemble node, ensuring all the predicted values and posterior probabilities are set to average. This allows the model to obtain the average misclassification rate of the 4 decision trees instead of obtaining from the highest. After configuring the property as shown in the figure below, we proceed to check the variables then execute the node.

Train	
Variables	...
Interval Target	
Predicted Values	Average
Class Target	
Posterior Probabilities	Average
Voting Posterior Probabilities	Average

The workflow for the random forest is depict as the figure in below.



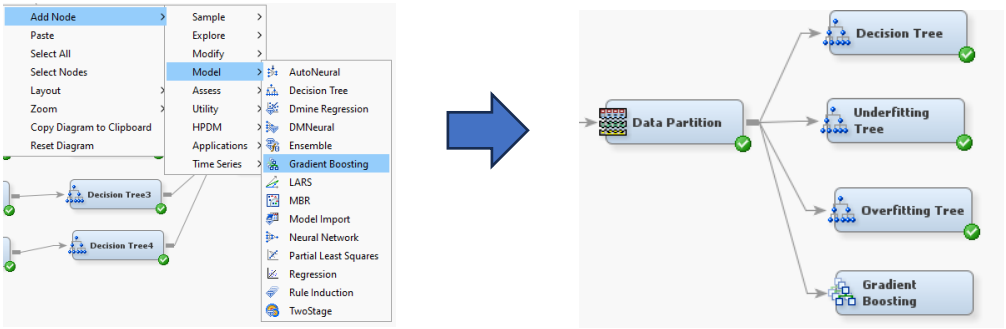
The figure below shows the fit statistics of the random forest. It shows the misclassification rate of the train dataset is 0.088 which largely lower than the optimal decision tree. It shows the ability to perform well in any unseen situations with low average squared error score.

Fit Statistics

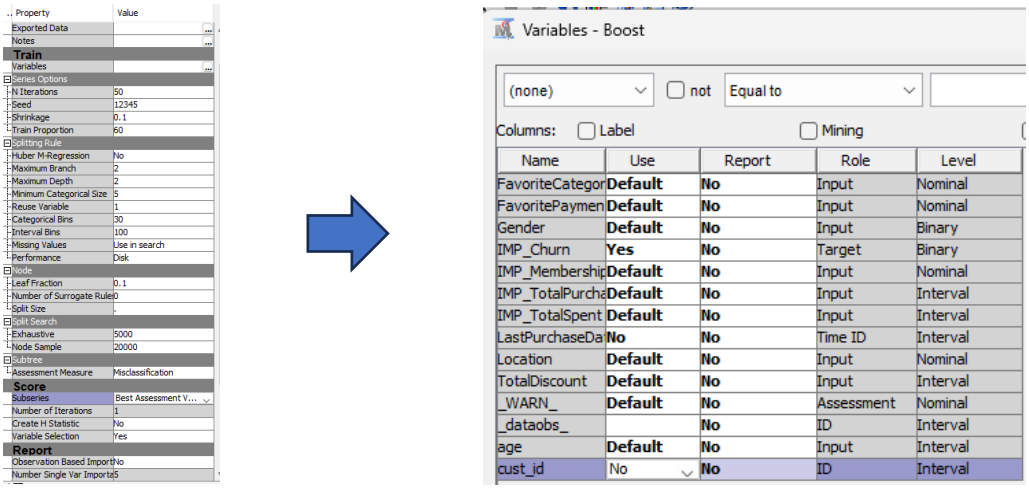
Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
IMP_Churn	Imputed Churn	_ASE_	Average Squared Error	0.07506		
IMP_Churn	Imputed Churn	_DIV_	Divisor for ASE	500		
IMP_Churn	Imputed Churn	_MAX_	Maximum Absolute Err...	0.956881		
IMP_Churn	Imputed Churn	_NOBS_	Sum of Frequencies	250		
IMP_Churn	Imputed Churn	_RASE_	Root Average Squared...	0.273971		
IMP_Churn	Imputed Churn	_SSE_	Sum of Squared Errors	37.52996		
IMP_Churn	Imputed Churn	_DISF_	Frequency of Classifie...	250		
IMP_Churn	Imputed Churn	_MISC_	Misclassification Rate	0.088		
IMP_Churn	Imputed Churn	_WRONG_	Number of Wrong Cla...	22		

2. Gradient Boosting

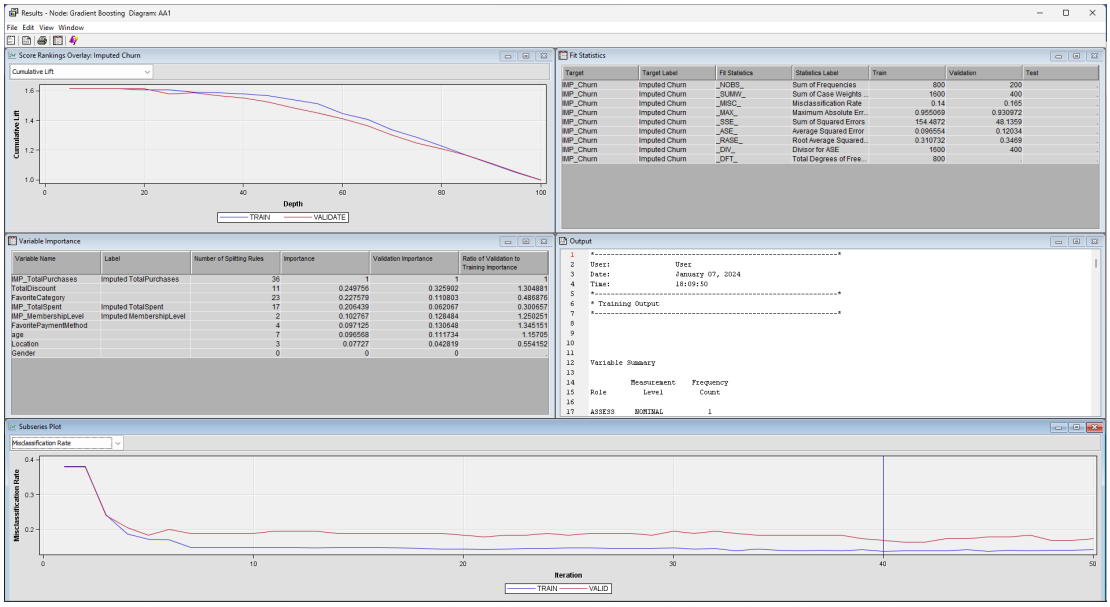
Moving on into creating gradient boosting model, we added the gradient boosting node to the diagram. The steps are shown in the figures below. Connect the gradient boosting node to the data partition node.



Proceed to configuring the node property by modifying the assessment measure to misclassification and leave the rest as default. Before executing the node, we edit the variable to select the variables that will not be included in the model. LastPurchaseDate and cust_id will be dropped just as what we have done in the decision tree part. Then the node is ready to be executed.



After execution of the node, we viewed the results. The figure below shows the results window. From the result window, we can observe the misclassification rate graph, variable importance table, classification table and the fit statistics.

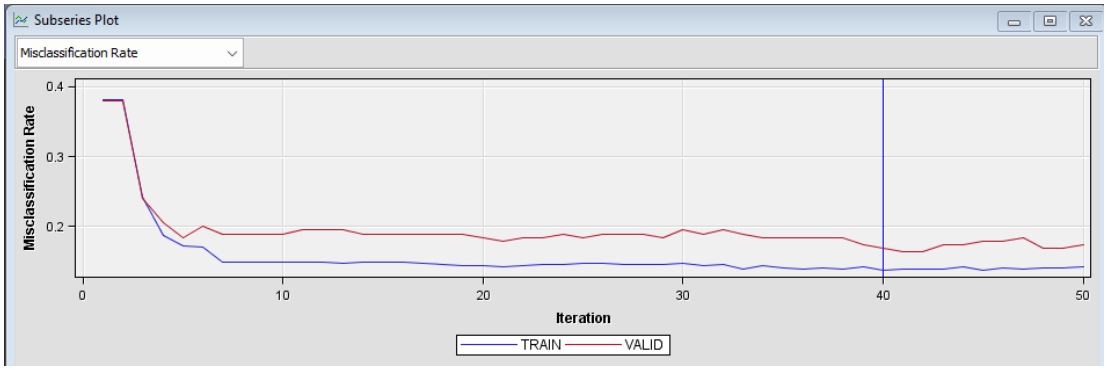


The figure below shows the variable importance of the variables used in the gradient boosting. We observed the imputed TotalPurchases which has the highest importance was involves in 36 splitting rules while the gender which gives 0 importance are not used in the gradient boosting.

Variable Importance

Variable Name	Label	Number of Splitting Rules	Importance	Validation Importance	Ratio of Validation to Training Importance
IMP_TotalPurchases	Imputed TotalPurchases	36	1	1	1
TotalDiscount		11	0.249756	0.325902	1.304881
FavoriteCategory		23	0.227579	0.110803	0.486876
IMP_TotalSpent	Imputed TotalSpent	17	0.206439	0.062067	0.300657
IMP_MembershipLevel	Imputed MembershipLevel	2	0.102767	0.128484	1.250251
FavoritePaymentMethod		4	0.097125	0.130648	1.345151
age		7	0.096568	0.111734	1.15705
Location		3	0.07727	0.042819	0.554152
Gender		0	0	0	.

The figure below shows the graph misclassification rate against iterations. From the line graph, we can observe that the model achieved its optimum at 40 iterations. The vertical line serve as the noteworthy indicator.



The figure below is the fit statistics of the gradient boosting. It shows the misclassification rate of 0.14 and 0.165 for training set and validation set respectively. The root average squared error is also lower by 0.02 when compared with the optimal decision tree model.

Fit Statistics

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
IMP_Churn	Imputed Churn	_NOBS_	Sum of Frequencies	800	200	.
IMP_Churn	Imputed Churn	_SUMW_	Sum of Case Weights ...	1600	400	.
IMP_Churn	Imputed Churn	_MISC_	Misclassification Rate	0.14	0.165	.
IMP_Churn	Imputed Churn	_MAX_	Maximum Absolute Err...	0.955069	0.930972	.
IMP_Churn	Imputed Churn	_SSE_	Sum of Squared Errors	154.4872	48.1359	.
IMP_Churn	Imputed Churn	_ASE_	Average Squared Error	0.096554	0.12034	.
IMP_Churn	Imputed Churn	_RASE_	Root Average Squared...	0.310732	0.3469	.
IMP_Churn	Imputed Churn	_DIV_	Divisor for ASE	1600	400	.
IMP_Churn	Imputed Churn	_DFT_	Total Degrees of Free...	800	.	.

The figure below shows the classification table. This classification table can be used to calculate the performance of the model based on several evaluation metrics such as accuracy, recall, precision and F1-score.

Event Classification Table

Data Role=TRAIN Target=IMP_Churn Target Label=Imputed Churn

False Negative	True Negative	False Positive	True Positive
58	251	54	437

Data Role=VALIDATE Target=IMP_Churn Target Label=Imputed Churn

False Negative	True Negative	False Positive	True Positive
19	62	14	105

Analysis between decision tree and ensemble methods

Evaluation Metrics	Misclassification Rate		Root Average Squared Error	
Model	train	validate	train	validate
Decision Tree	0.14	0.185	0.3361	0.3773
Random Forest	0.088	-	0.2740	-
Gradient Boosting	0.14	0.165	0.3107	0.3469

The Misclassification Rate measures the proportion of incorrectly classified instances. A lower misclassification rate indicates better model performance. Both Random Forest and Gradient Boosting outperform the Decision Tree on both the training and validation sets. They have lower misclassification rates, suggesting improved accuracy.

The Root Average Squared Error is a measure of the average magnitude of errors. A lower value indicates better predictive accuracy. Similar to the misclassification rate, both Random Forest and Gradient Boosting have lower root average squared errors compared to the Decision Tree, indicating improved predictive accuracy.

Both Random Forest and Gradient Boosting demonstrate better performance (lower misclassification rates and root average squared errors) compared to the Decision Tree. Ensemble methods, such as Random Forest and Gradient Boosting, often outperform individual decision trees by combining multiple models to reduce overfitting and improve generalization.

Ensemble methods combine multiple models, reducing overfitting and improving the overall predictive accuracy. This is crucial in e-commerce to accurately predict customer behaviour, such as identifying potential churners or understanding purchase patterns. Besides, E-commerce datasets can be complex, with numerous interacting factors influencing customer behaviour. Ensemble methods are capable of capturing complex relationships and interactions between variables, leading to more accurate predictions. In addition to that, it can avoid from overfitting while capturing more complex relationships and interactions between variable. Decision trees, when used individually, can be prone to bias and overfitting. Ensemble methods help mitigate these issues by combining multiple weak learners into a strong model, resulting in more balanced and unbiased predictions. Ensemble methods can uncover hidden patterns and segments within the customer base. This segmentation is valuable for tailoring

marketing strategies, personalized recommendations, and targeted promotions to different customer groups.

Reflection

First of all, from this case study, we are able to interpret the customer behaviour on the E-commerce website. Additional to that is the case study requires us to select our own dataset while having a guide in the dataset structure. It serves as the additional task which tested us on the knowledge on the e-commerce datasets we collected instead of just blindly selecting the datasets and applying the steps to fulfil the task. Besides, it trained my python programming skills as the dataset I collected does not matches the dataset structure fix directly, hence, utilising the python libraries, I am able to aggregate a datasets which truly reflect the customer behaviour in the real situation.

Moving on it also help to revised back all the tools taught during the Data Mining course such as SAS Enterprise Miner, Talend Open Studio for Data Integration and Talend Data Preparation Free Desktop. During the case study, I faced a few problems due to the version and subscription of the software. As shown during the class, the duplication of data is unable to remove in the Talend Data Preparation with the deduplicate with row match function. Going through the documentation, this function only supported in the Talend Data Preparation version 7.3 and 8.0 Cloud version in which both versions are needed with subscription. However, I am able to remove the duplicates with the Talend Open Studio for Data Integration with the use of tUniqRow node. Another issue faced in the Talend Data Preparation is that the data we exported to csv did not follow the data type we set in the Talend Data Preparation.

Lastly, from this case study, I learnt time managing skills beside the technical skills as this task is given one day including our sleep time and I am having another class on the next day. Hence, proper time management will ensure the task are able to completed on time.