

**Semester 1, 2023/2024**

**WIE3007 Data Mining and Warehousing**

**Group Assignment**

**Mobile Price Prediction**

**Lecturer: Prof. Dr. Teh Ying Wah**

| **NAME** | **MATRIC NUMBER** |
| --- | --- |
| Lim Zheng Yu | U2102809 |
| Guan Ming Li | U2102768 |
| Khoo Zi Wei | U2102814 |
| Janice Chong See Wai | S2132420 |
| Jasmine Chong See Yan | S2132419 |

**Link to datasets and code:**

<https://github.com/millivan/WIE3007-Data-Mining-Warehousing-Group-Assignment>

**Link to video:** <https://drive.google.com/drive/folders/1zsV3PDJf8pamb9Nom34p5aFY7X0Uckew?usp=sharing>

## Table of Content

[1.0 Introduction of Dataset 4](#_rzx4dppemiz4)

[2.0 Objectives 5](#_j5rgkvtce2xc)

[3.0 Data Preprocessing 6](#_x9ipn24crc4y)

[3.1 Concatenating the datasets 6](#_jqji5typ01f8)

[3.2 Eliminating Duplicate Entries 7](#_oi8c5ec2tdvz)

[3.3 Standardising Data Formats 8](#_e3pmchnnlf8i)

[3.4 Imputation 8](#_qfiwf2ktm6lk)

[4.0 SAS SEMMA Methodology 12](#_dt6zyibsizza)

[4.1 Sample 12](#_o0pwwjep05tx)

[4.2 Explore 13](#_qrjy2nb15pw)

[4.2.1 EDA Visualization 13](#_lb7yadvary22)

[4.2.1.1 StatExplore 13](#_yctzfhg97y9m)

[4.2.1.2 Graph Explore 16](#_liz7dyxjj84v)

[4.2.1.3 4D Visualisation 20](#_7l5jtlukrmkj)

[4.2.2 Correlation Analysis 22](#_t33b0a522thh)

[4.2.2.1 Variable Importance 24](#_6ofix0sypo05)

[4.2.2.2 R2 Values 24](#_l0f6q486zhgl)

[4.2.2.3 Variable Selection Table 25](#_c28j9ka1bl6p)

[4.2.2.4 Effects in the Model 26](#_v2h9whoksvkx)

[4.2.2.5 Output Window 26](#_ir298ke102j4)

[4.2.3 Association Rule Mining 30](#_6u0ikmk5os8f)

[4.2.3.1 Dataset Used 30](#_rdvdryj6ipyy)

[4.2.3.2 Process Flow & Variable Selection 30](#_oqu3k4p6hbu9)

[4.2.3.3 Output Window 31](#_lmhhcc508ua)

[4.2.4 Sequence Analysis 35](#_gtcrpxfp5wcv)

[4.2.4.1 Dataset Used 35](#_rtrfdpk7ayqy)

[4.2.4.2 Process Flow & Variable Selection 35](#_y1bdbcm48zbu)

[4.2.4.3 Output Window 36](#_ahl9qosb2hjo)

[4.2.5 Time Series (TS) Clustering 39](#_qrdb73epjmlx)

[4.2.5.1 Battery Power 40](#_xdh3hynu7bne)

[4.2.5.2 Internal Memory 41](#_o5aaazfdxg1)

[4.2.5.3 Primary Camera 41](#_5j4qrpmx568s)

[4.2.5.4 RAM 42](#_ptigdds8soky)

[4.2.5.5 Number of Cores 43](#_ncq7wgj9josu)

[4.2.5.6 Overall Results 44](#_5zql24tna68y)

[4.2.6 DBSCAN 45](#_x81ow4pinvx5)

[4.3 Modify 47](#_hu5lkr11tj0m)

[4.3.1 Data Type Conversion and Label Encoding 47](#_4x3oyhdq1c2h)

[4.3.2 Feature Engineering 49](#_wmyp0p1kg5z9)

[4.3.2.1 Dataset Examination 49](#_jebyizf4257v)

[4.3.2.2 Implementation of FeatureTools 53](#_re8h4rtl605z)

[4.3.2.3 Perform Deep Feature Synthesis 55](#_pdgutoix00my)

[4.3.3 Developing Star Schema Structure 56](#_hywzshg0nmuh)

[4.3.3.1 Design Data Model 56](#_iu5no7vup10k)

[4.3.3.2 Define Business Rules 57](#_vszkt7k08epz)

[4.3.3.3 Create Star Schema 57](#_6l5xbsdncr8n)

[4.3.3.4 Create Data Dictionary 58](#_46aqg6j35vh)

[4.3.3.5 Gather Insightful Information 60](#_4h7tya50zxyh)

[4.3.4 Splitting Data for Training and Validation 61](#_yzkcunrf5sdv)

[4.3.5 Feature Selection 62](#_kp70wdeiws9r)

[4.4 Model 65](#_qh3bxs1it91f)

[4.4.1 AutoNeural 65](#_x58ba5lbdv21)

[4.4.2 Decision Tree 66](#_jdykmvxwt2f3)

[4.4.3 Gradient Boosting 67](#_7eitkuavndjh)

[4.4.4 Random Forest 68](#_xqvzgxy2v7jd)

[4.4.5 Ensemble Models: Bagging 69](#_v6sz3ryz72s4)

[4.4.6 Ensemble Models: Boosting 69](#_dh0ndc707x7h)

[4.4.7 Ensemble Node Methods 70](#_fyu40985005h)

[4.4.7.1 Average 70](#_wk1lpqo9cff5)

[4.4.7.2 Voting 70](#_cw4ccbwbskq9)

[4.5 Assess 71](#_kck5vy78qv1v)

[4.5.1 Misclassification Rate 71](#_krpr19lcnshr)

[4.5.2 Precision, Recall, F1-Score 72](#_n0wl38o86ifm)

[4.5.3 Score 73](#_3qifshmgjh4l)

[5.0 Conclusion 74](#_uuolqcblort8)

[6.0 References 75](#_x51h07s91gy8)

## 1.0 Introduction of Dataset

The project is based on the Mobile Price Classification dataset, which can be accessed on Kaggle. This dataset gathers sales information for mobile phones from various brands and includes a range of details about the mobile phones’ features and specifications. By using Talend Data Prep, Talend Integration, FeatureTools, SAS Enterprise Miner and KNIME, our analysis focuses on uncovering patterns that link the characteristics of mobile phones to their price ranges. Our goal is not to determine the exact price but to categorise the mobile phones into different price levels, helping us to understand how features may influence the pricing category within the mobile phone market.

Our project works with two datasets named “mobile\_price\_1” and “mobile\_price\_2”. Each dataset contains 22 attributes that describe the features of various mobile phones and also record the purchase date for each device. The “mobile\_price\_1” dataset includes 1028 entries, and “mobile\_price\_2” has 2064 entries. Table 1 below provides a list of these attributes and a brief explanation of each attribute.

| **Attribute** | **Description** |
| --- | --- |
| date | Purchase date of the mobile phone |
| pc | Megapixels of the primary camera |
| fc | Megapixels of the front camera |
| sc\_h | Screen height in centimetres |
| sc\_w | Screen width in centimetres |
| m\_dep | Depth of the mobile phone in centimetres |
| px\_width | Pixel resolution width |
| px\_height | Pixel resolution height |
| ram | Random Access Memory in megabytes |
| int\_memory | Internal memory in gigabytes |
| four\_g | Availability of 4G |
| three\_g | Availability of 3G |
| dual\_sim | Support for dual SIM |
| battery\_power | Battery capacity in milliampere-hours (mAh) |
| touch\_screen | Presence of a touch screen |
| clock\_speed | Microprocessor’s speed of executing instructions |
| n\_cores | Number of processor cores |
| wifi | Availability of wifi |
| blue | Availability of Bluetooth |
| mobile\_wt | Weight of the mobile phone in grams |
| talk\_time | Maximum duration of call in hours after a full charge |
| price\_range | Categorised price range (low cost, medium cost, high cost, very high cost) |

Table 1: Dataset Description

## 2.0 Objectives

In this project, we aim to accomplish several objectives to enhance our analysis of the datasets:

1. To identify and visualise the purchasing trends and common patterns in mobile phone sales
2. To discover the relationships between various mobile phone characteristics through association rule mining
3. To identify and evaluate the impact of specific features on the classification of mobile phones by price range
4. To explore additional features that could be beneficial for the modelling phase using feature engineering
5. To apply and compare different classification techniques to predict the mobile phone’s price range based on its attributes
6. To assess and report the performance of various predictive models

## 3.0 Data Preprocessing

Before employing the SEMMA methodology for data mining, it is crucial to perform data cleaning and preprocessing. This ensures that the datasets are primed for exploration and can yield more valuable insights. Untidy datasets may introduce bias and inconsistency during the modelling stage, potentially skewing predictive results. The data preprocessing phase encompasses four main steps: merging the two datasets, eliminating any duplicate entries, standardising the data format, and imputing missing values.

### 3.1 Concatenating the datasets

Initially, our project utilises two separate datasets: “mobile\_price\_1” and “mobile\_price\_2”. To streamline our analysis, we plan to concatenate these two datasets into a single dataset since they share the same structure and have an identical count of 22 features. Combining these datasets is essential for ensuring a comprehensive analysis, as it enlarges the data pool and provides a more robust basis for identifying the patterns of the data.

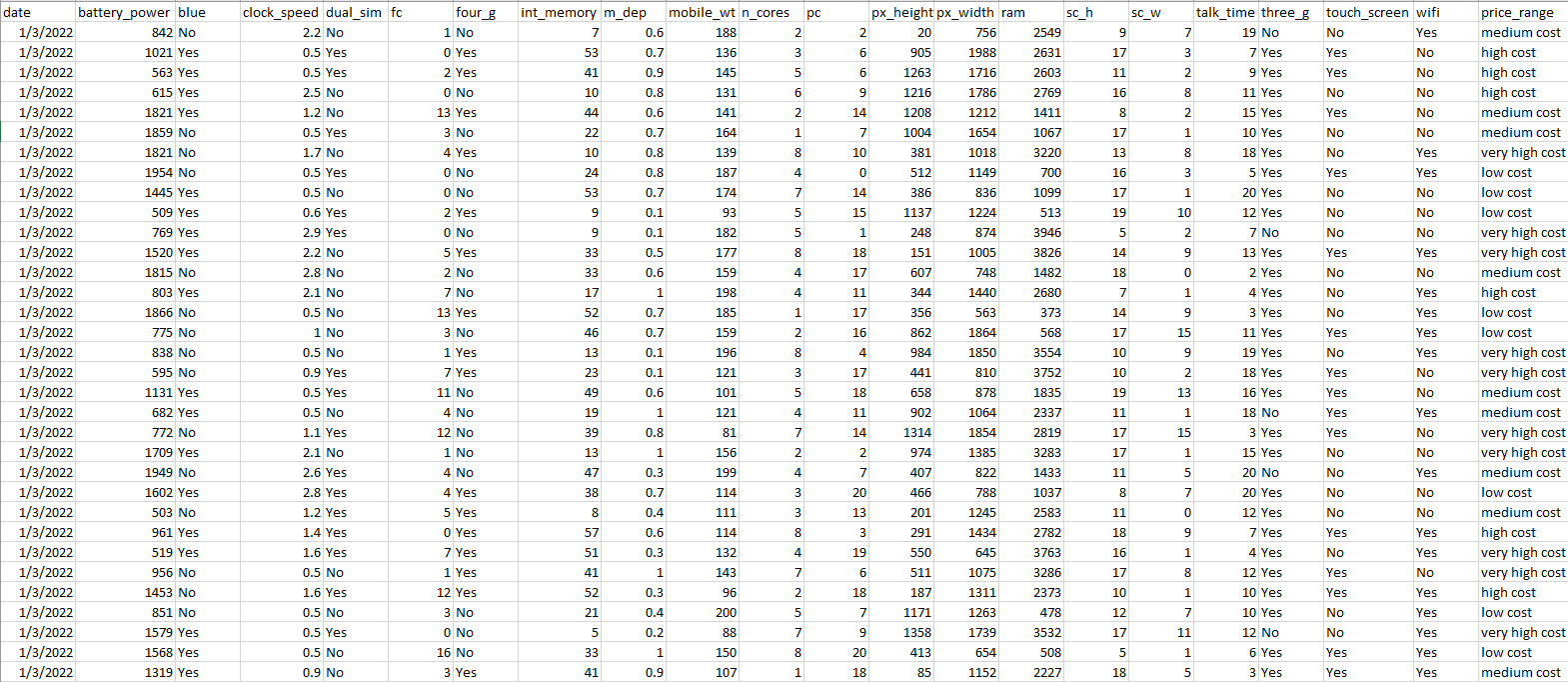
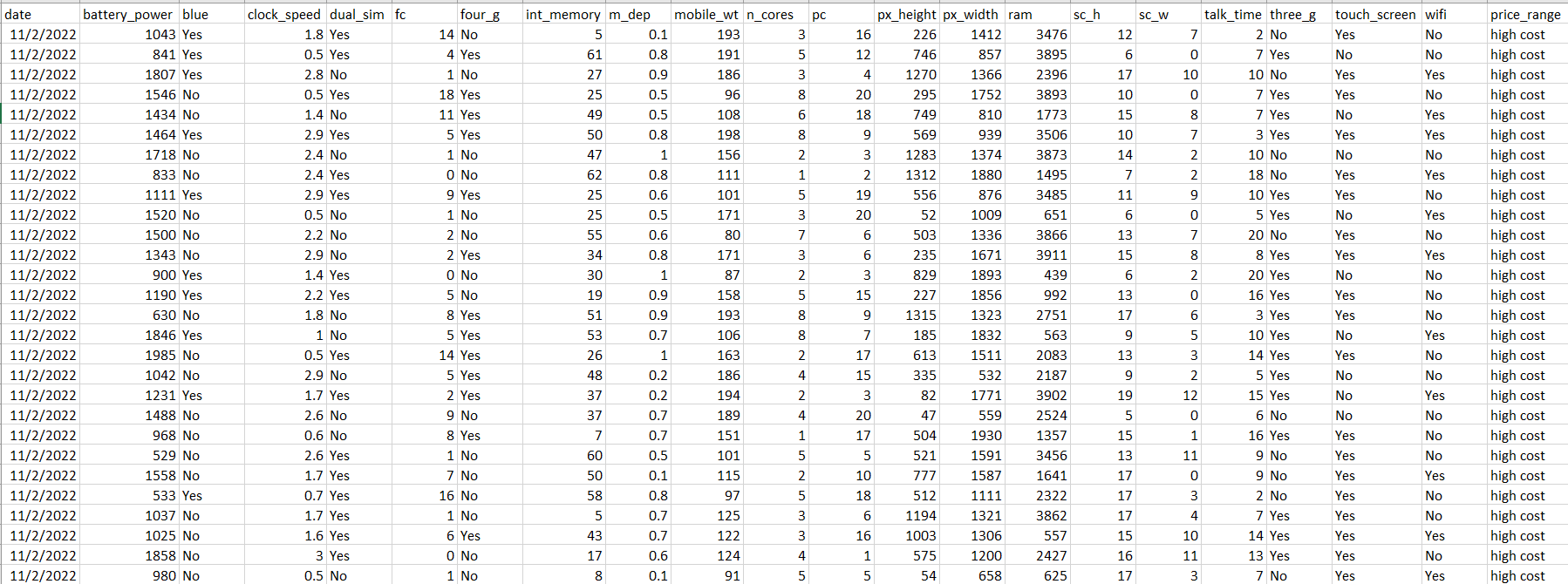


Figure 1 & 2: “mobile\_price\_1” and “mobile\_price\_2” Datasets

To concatenate the two datasets, we will be using Talend Integration. First, we set up new metadata for both “mobile\_price\_1” and “mobile\_price\_2” to import them. Once the datasets are imported, we read them by bringing them into the job design space as tFileInputDelimited components.

For the actual concatenating, we use a tUnite component. We drag this into the workspace and link both datasets to it. We run the process after confirming the inputs and ensuring the columns align properly. Talend automatically matches the columns since the datasets are structured the same.

Finally, to get our concatenated dataset, we use a tFileOutputDelimited component for saving the output. We connect the tUnite component to this tFileOutputDelimited component and save the output result in a CSV file called “concatenated\_dataset.csv”. This new file now has 3092 entries, with 22 features from the original datasets.



Figure 3: Concatenating The Datasets using Talend Integration

### 3.2 Eliminating Duplicate Entries

In our concatenated dataset, we found out that there are several duplicate rows. Duplicates can skew results and generally lower the quality of any insights drawn from the data. Therefore, removing these duplicates is crucial to maintain the integrity of our analysis.

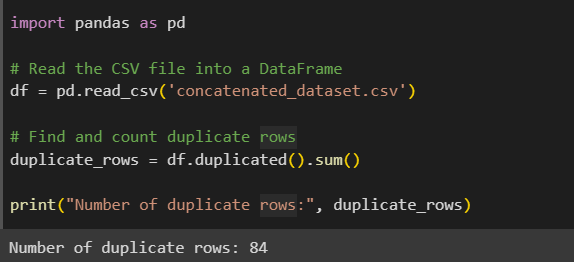


Figure 4: Count of Duplicate Rows using Python

To remove the duplicate rows from our dataset, we will use Talend Integration again. We begin by creating new metadata for the concatenated dataset and importing it. Once imported, we read the data as a tFileInputDelimited component in the job design space.

Next, we tackle the duplicate rows with a tUniqRow component. After dragging it into our workspace and connecting it to our data, we will specify the columns for comparison, which in our case, involves all of them as Talend does this by default.

To finalise and save our refined dataset without duplicates, we use a tFileOutputDelimited component. Connecting the tUniqRow component to the tFileOutputDelimited component, we define where to save the file and name it “noduplicate.csv”. Our cleaned file will then contain 3008 entries, reflecting the removal of 84 duplicate rows, and preserving the original 22 features.

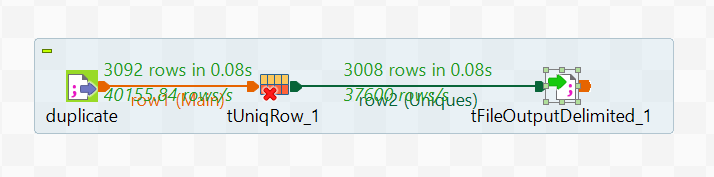


Figure 5: Removing The Duplicate Rows using Talend Integration

### 3.3 Standardising Data Formats

Additionally, we noticed that the “date” column in our dataset has inconsistent formatting, with dates appearing in two different formats: ‘12/2/2022’ (DD/MM/YYYY) and ‘13-02-2022’ (DD-MM-YYYY), which are shown in Figure 6. Standardising the date format is important to ensure consistency, which is vital for accurate analysis and comparison.

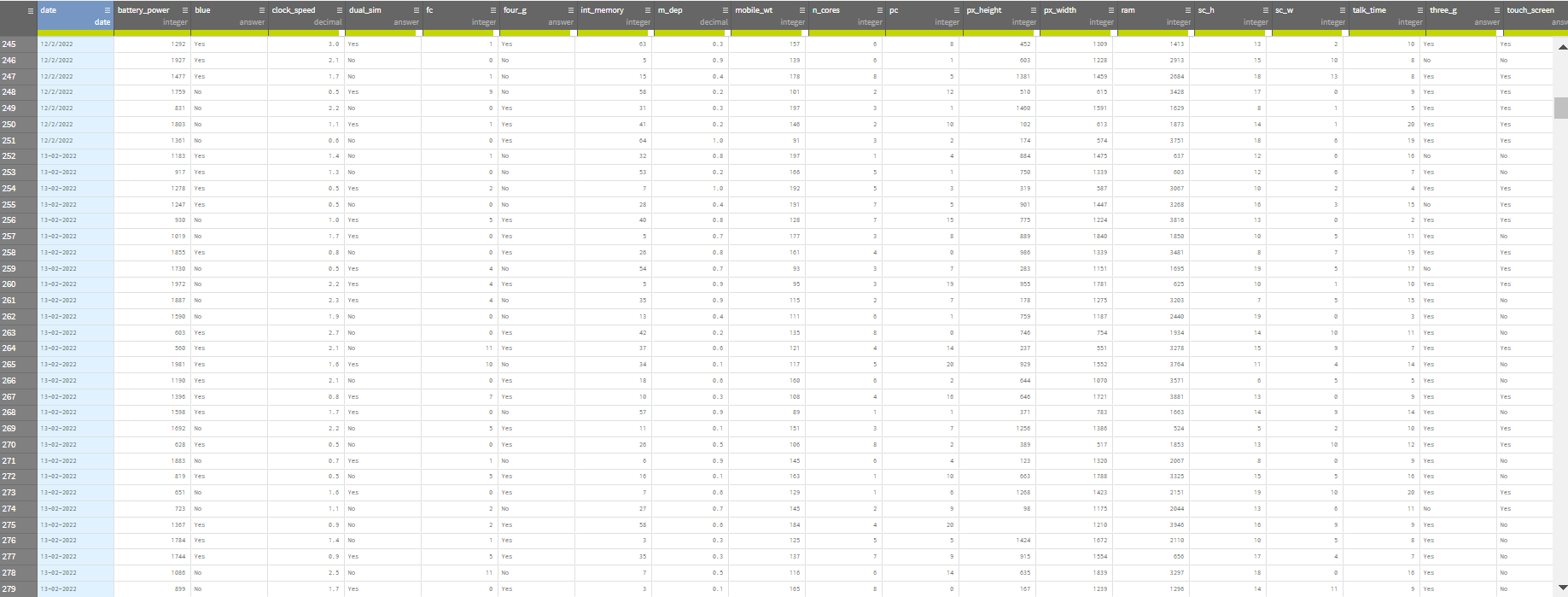


Figure 6: Inconsistent Data Format in “date” Column

To standardise the date format in our dataset, we will employ Talend Data Prep. We start by loading the “noduplicate.csv” file into our workspace. Then, we change all dates into a single format. Specifically, we convert the dates with the ‘dd-MM-yyyy’ format to the ‘dd/MM/yyyy’ format, ensuring that every date in the dataset follows this uniform style. Once all the dates are uniform, we save the updated dataset as a new CSV file named “standardised.csv”.

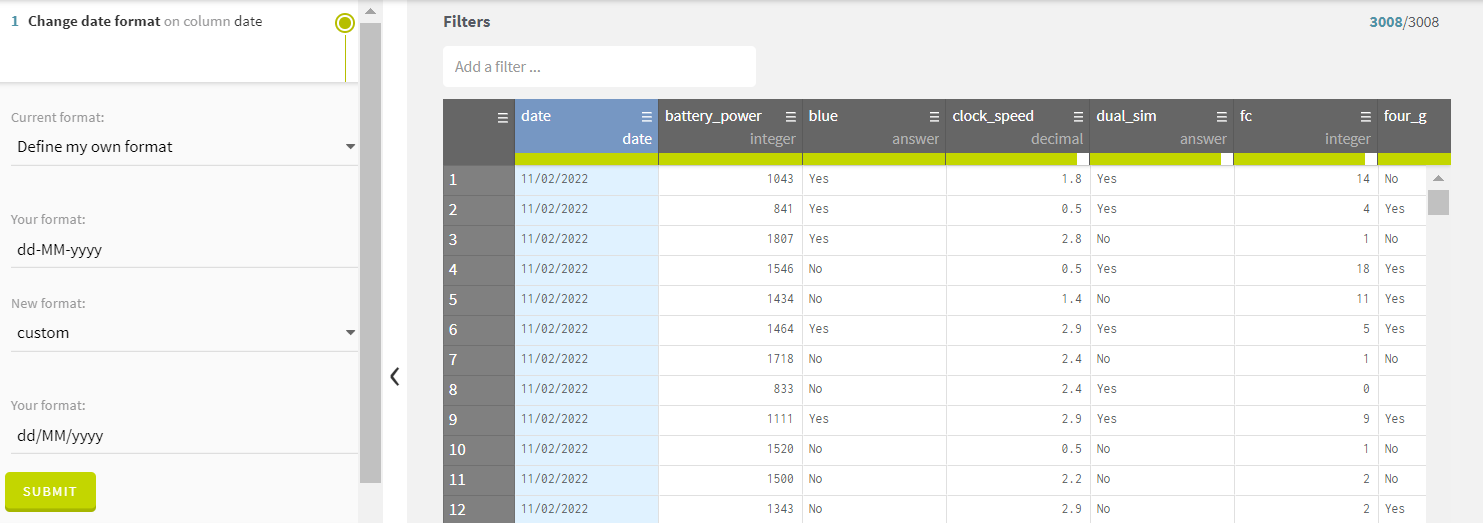


Figure 7: Standardising The Date Format using Talend Data Prep

### 3.4 Imputation

After we have adjusted the date formats using Talend Data Prep, we observe that some columns still have null values. This is indicated by the incomplete green bars beneath the column headers, which show the gaps where data should be. To address these gaps, we need to perform imputation. Imputation is important because it allows us to fill in these missing values with reasonable estimates, ensuring that our dataset is complete and maintaining the quality of our dataset.

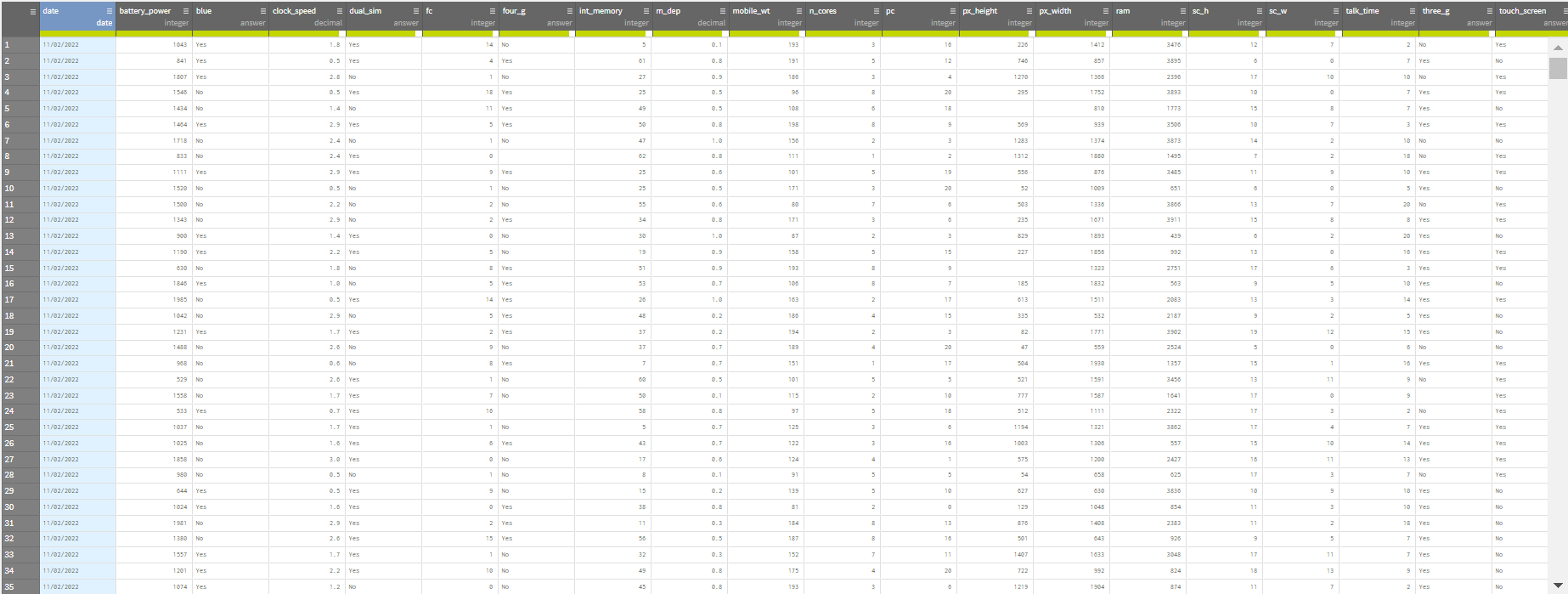


Figure 8: Checking Missing Values using Talend Data Prep

To carry out imputation and handle the missing values in our dataset, we will turn to SAS Enterprise Miner. We start by bringing our dataset into the program using the File Import Node and placing it into the workspace. Once we run the File Import Node, we will get a summary of our dataset, which will indicate that all the features fall into nominal and interval levels, as shown in Figure 9.

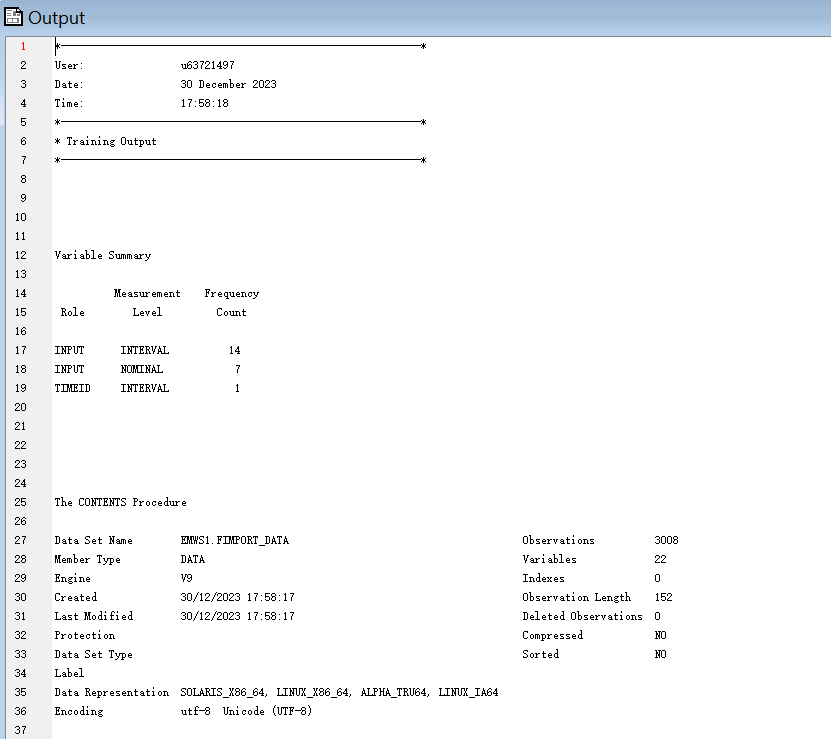


Figure 9: Summary of Dataset

By connecting the File Import Node to the StatExplore Node, we can see that our dataset has a considerable number of missing values. The output from the StatExplore Node details which columns have missing entries, dividing them into two categories: nominal level and interval level. Below are two images that display the number of missing values for each column in the dataset.

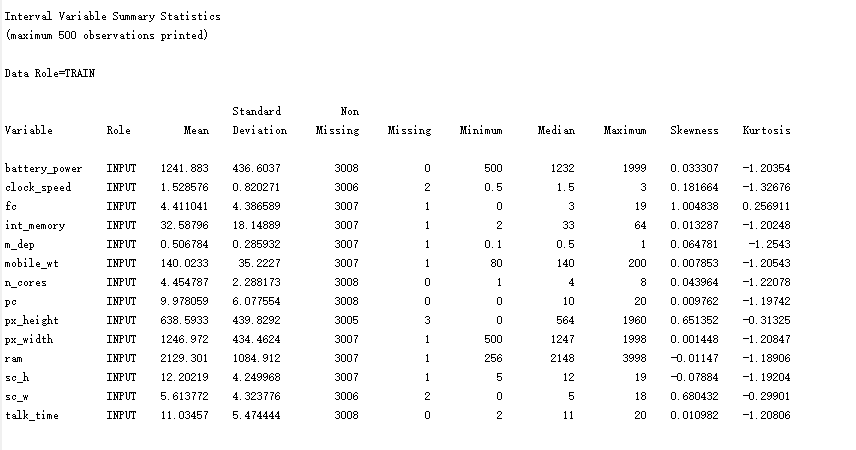
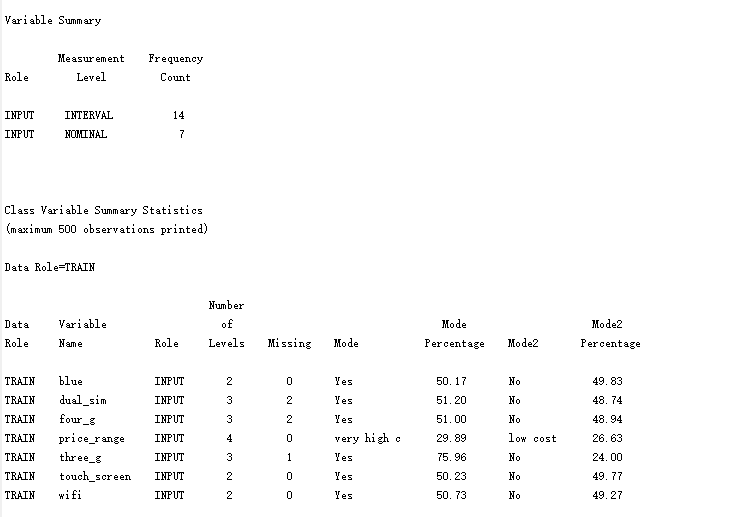


Figure 10 & 11: Count of Missing Values of Each Column

Next, we will tackle the missing values by using the Impute Node in SAS Enterprise Miner. We add this node to our workspace and connect it to our dataset. For numerical, or interval data, we use the “Mean” method for imputation. This method is straightforward as it replaces missing numbers with the average value of the rest, which helps keep the data’s overall pattern true to the original. For categorical or nominal data, we use the “Count” method. This method fills in missing values with the most common category (mode) found in the dataset, a simple way to ensure our categorical data stays consistent.

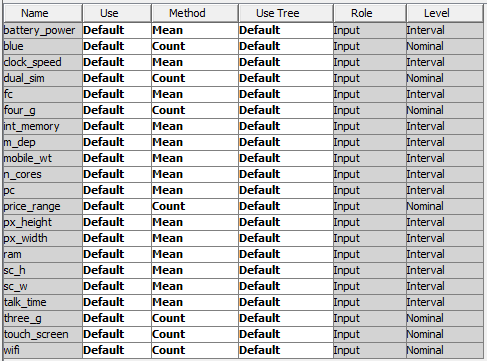


Figure 12: Method Used for Imputation in Each Column

After applying the imputation methods, every missing value has been replaced with an appropriate imputed value. The outcomes of this process are clearly outlined in the imputation summary below, where we can see the imputed values for each column that had missing data.

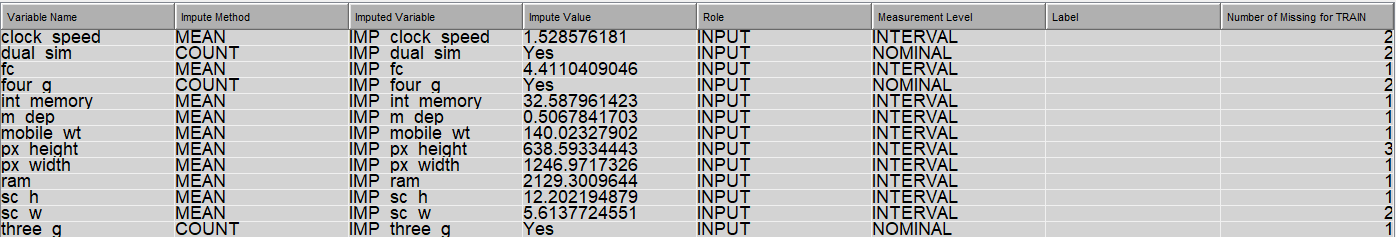


Figure 13: Imputation Summary

To save the datasets after imputation, we link up with the Save Data Node. Then, we set the location where we want to save the file and named the new CSV file “imputed.csv”.

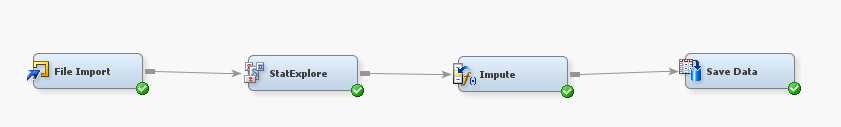


Figure 14: Imputation Process using SAS Enterprise Miner

## 4.0 SAS SEMMA Methodology

In our data mining project, we are going to use the SEMMA approach provided by SAS to help us extract valuable insights and apply them to our predictive classification models. This method includes five stages: Sample, Explore, Modify, Model, and Assess.

### 4.1 Sample

In the Sample step, the goal is to sample a dataset that is big enough for meaningful results but small enough to manage efficiently. In our project, sampling is not necessary since our dataset is relatively compact, containing only 3008 records. It contains all the necessary data, therefore simply trimming it down could risk omitting important information and potentially lead to unreliable and inconsistent results in the modelling stage.

The target variable of our dataset, “price\_range”, divides mobile phones into four categories which are “low cost”, “medium cost”, “high cost” and “very high cost”. This categorisation is crucial for understanding pricing strategies in data analysis.

In this stage, we can use the File Import Node to load our dataset. Once we run this node, it gives us a summary of the dataset, detailing the data types for each attribute as shown in the image below.

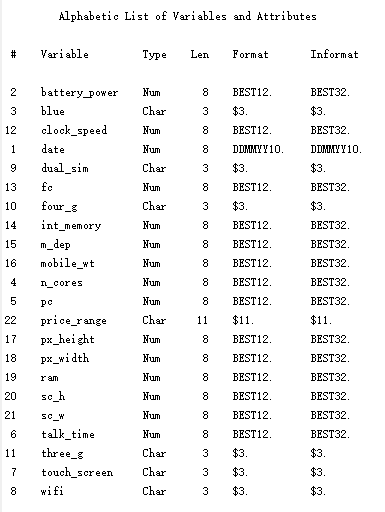


Figure 15: Summary of Dataset

### 

### 4.2 Explore

In the Explore phase, we will examine the datasets closely to observe for any unexpected trends, relationships or patterns and get a good grasp of the data. During this stage, we will focus on six main steps: conducting exploratory data analysis (EDA) with visualisations that may include 3D and 4D charts, analysing how variables correlate with each other, searching for association rules among the data, performing sequence analysis, implementing time series clustering and DBSCAN.

#### 4.2.1 EDA Visualization

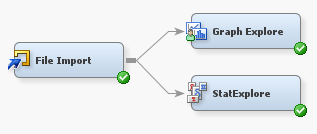


Figure 16: Flow of EDA Visualization

Before running the ‘Graph Explore’ and ‘StatExplore’ nodes, the Role of the ‘price\_range’ variable is changed to Target.

##### 4.2.1.1 StatExplore

Using the ‘StatExplore’ node, we can perform Summary Statistics on our dataset.

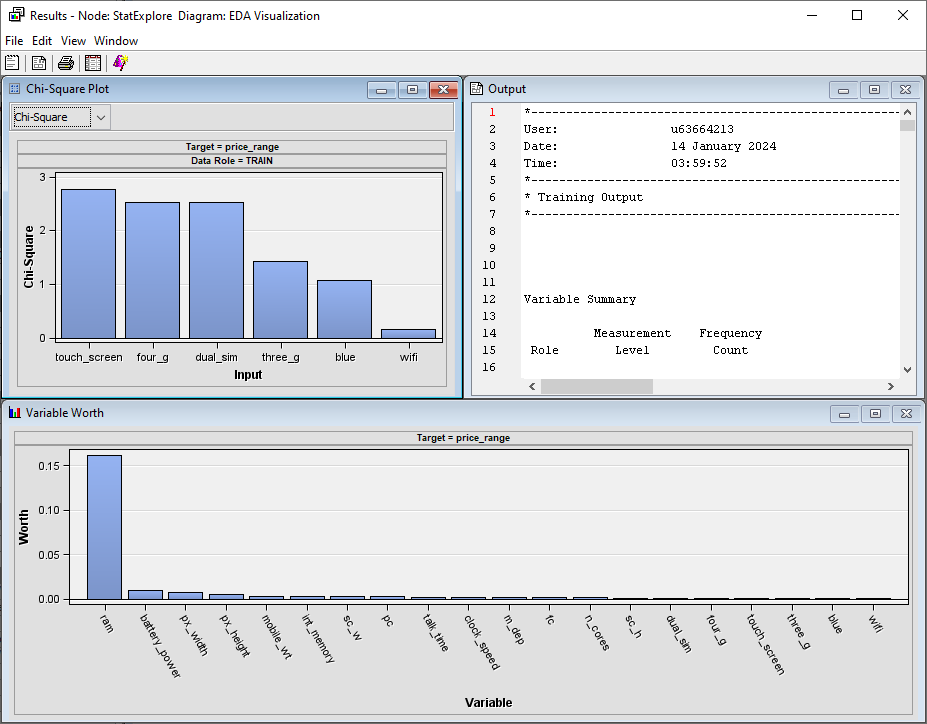


Figure 17: Output of ‘StatExplore’ Node

From the Results window, go to the ‘View’ tab and choose ‘Plots’ to explore all possible visualisations by SAS Enterprise Miner.

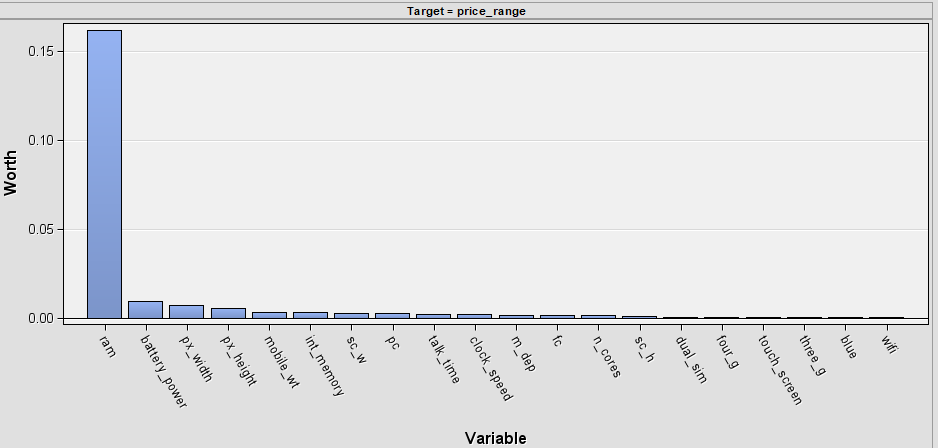
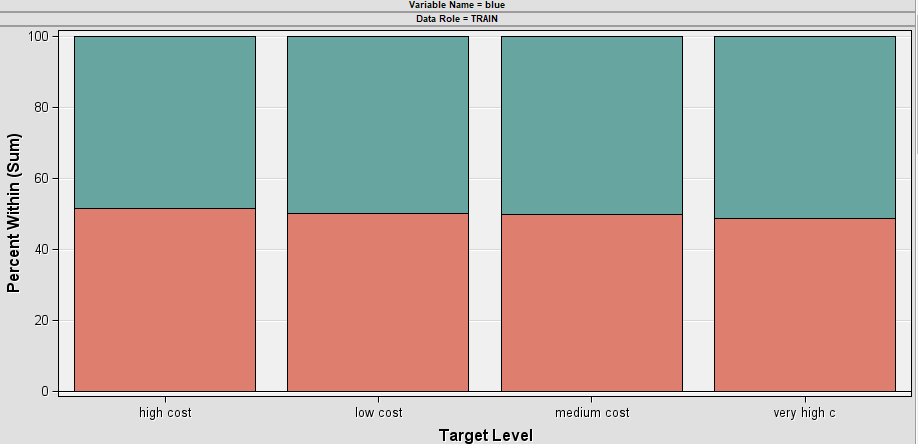


Figure 18: Plot of Variable Worth

From the **‘Variable Worth’** window, we can see that the most important variable is **‘ram’** with a worth of 0.1619. The other variables are far behind, with **‘battery\_power’** in second with a worth of a meagre 0.0095. This means that from this dataset, only **‘ram’** has the strongest predicting power.



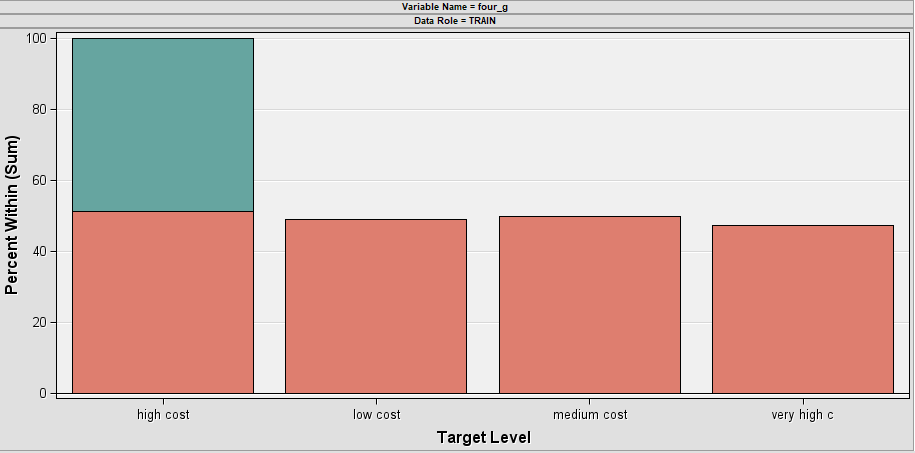
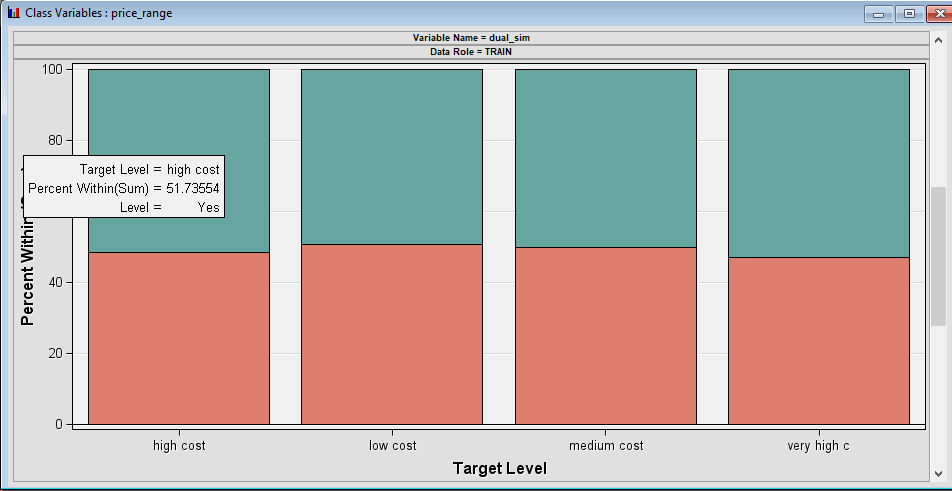


Figure 19 - 21: Distribution of Categorical Variables Across All Levels of Target Variable

From the ‘Class Variables : price\_range’ window, we can observe that the data is very balanced as all categorical variables against all levels of the target variable are plotted around the 50% mark.



For example, 51.73554% of data which belong to the ‘high cost’ level of the ‘price\_range’ target variable have dual\_sim.

Note: The last plot may be a system bug. Further checking is done on the dataset and it is confirmed that the distribution is accurate.

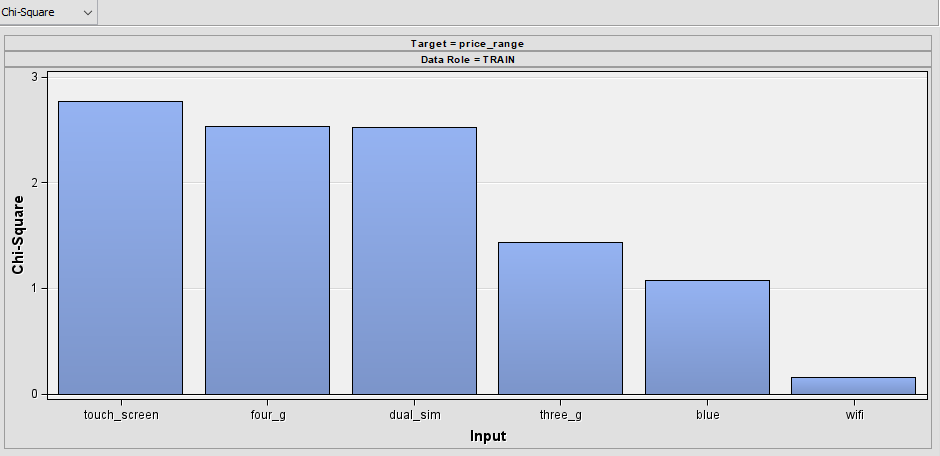


Figure 22: Plot of Chi-Square Values

The **‘Chi-Square Plot’** window shows the Chi-Square values between the target variable ‘price\_range’ and categorical variables in the dataset.

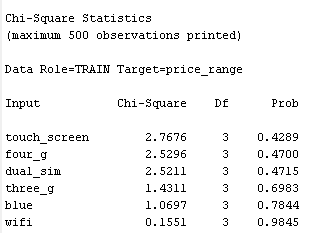


Table 23: Chi-Square Statistics

We can observe that none of these variables resulted in a value of 0.05 or less in the ‘Prob’ column. Hence, our null hypotheses that there is no correlation between these categorical variables and the target variable are accepted. In other words, these categorical variables do not have any relationship with the target variable. This explains why these variables have such a low worth in the previous visualisation.

##### 4.2.1.2 Graph Explore

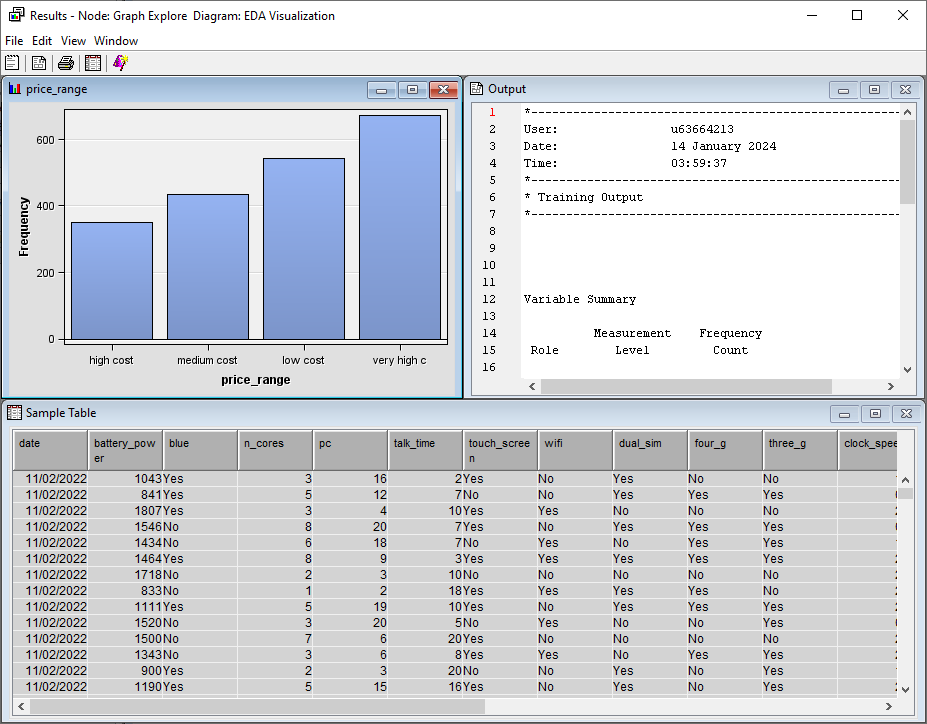


Figure 24: Output of ‘Graph Explore’ Node



Figure 25: Distribution of ‘price\_range’ Target Variable

Distribution of the ‘price\_range’ variable are as follows:

| **Label** | **Count** |
| --- | --- |
| Low cost | 543 |
| Medium cost | 434 |
| High cost | 350 |
| Very high cost | 673 |

Table 2: Distribution of “price\_range” Target Variable

From the table, we can say that the target variable is slightly imbalanced. This should be made aware of before proceeding with the Model stage.

From the StatExplore section, we noticed that **‘ram’** is one of the most important variables in the dataset. Let us explore its relationship with **‘price\_range’**.

By default, Graph Explore does not generate boxplots. To do this, go to the ‘View’ tab and select ‘Plot…’

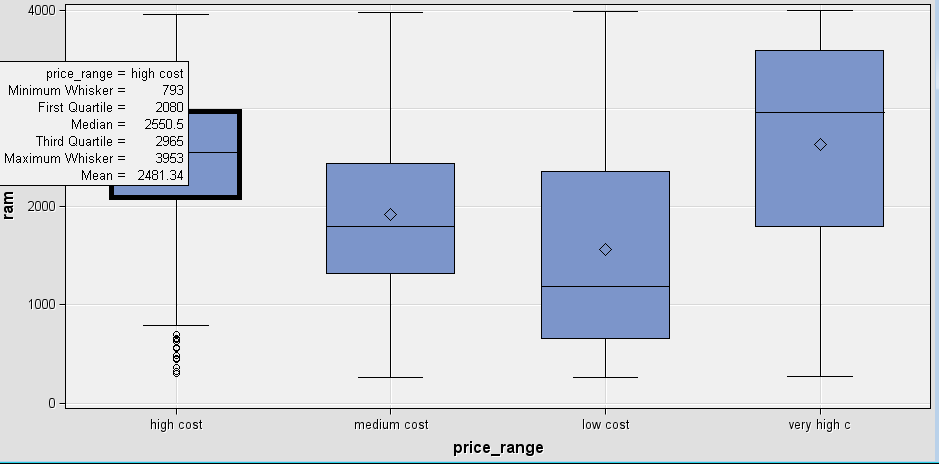


Figure 26: Boxplot of ‘ram’ Across Different Price Ranges

We can observe that there are no outliers for all except the ‘high cost’ price range. The minimum whisker for this boxplot is 793, hence, phones with ‘ram’ of less than 793 will be considered outliers.

To explore more on the specifications of the phones, here is the density plot of ‘battery\_power’ against ‘ram’:

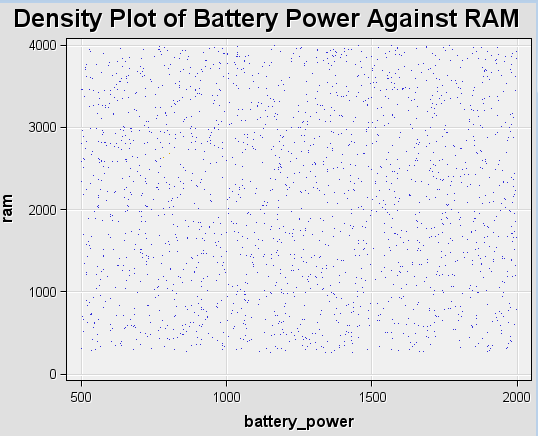


Figure 27: Density Plot of ‘battery\_power’ Against ‘ram’

We can say that the distribution of phones based on their ‘battery\_power’ and ‘ram’ is quite even as there are no obvious regions which are particularly dense from the plot.

#### 

Figure 28: Histogram of Distribution of Phones based on RAM

We can see that the distribution of phones based on their ‘ram’ size is also quite even, with no bins having extreme values. The minimum value for ‘ram’ is 258 while that maximum is 3998. Each bin has a percentage of between 8.85% and 11.20%.

#### 

Figure 29: 3D Grid Chart between ‘ram’, ‘battery\_power’, and ‘talk\_time'

The 3D grid chart shows that in general, the higher the ‘battery\_power’ and the larger the ‘ram’, the longer the ‘talk\_time’. However, if we look at only ‘ram’ against ‘talk\_time’, there doesn’t seem to be a clear direction. Hence, we can deduce that the larger contributing factor to a longer ‘talk\_time’ is due to the larger battery.

##### 4.2.1.3 4D Visualisation

##### 

Figure 30: 4D Scatter Plot

The features used for this plot are ‘ram’, ‘battery\_power’, ‘int\_memory’ and ‘price\_range’ (price\_range is an ordinal variable from 0 to 3, 0 being the cheapest and 3 being the most expensive).



Figure 31 & 32: 4D Scatter Plot from the Angle of ‘ram’

We can see that at the lower spectrum of ‘ram’ are where most of the labels are in blue colour (cheapest category). The distribution of red points (most expensive category) are denser at the higher spectrum of ‘ram’. From this visualisation, we can say that phones with small ‘ram’ are generally cheaper than phones with large ‘ram’.

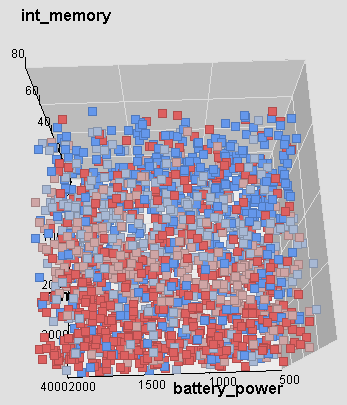


Figure 33: 4D Scatter Plot from the angle of ‘battery\_power’

A top-down view of the visualisation enables us to see how the ‘battery\_power’ variable affects the prices of phones. Unfortunately, there is no relationship that can be discovered through observation as it appears that there is no clear segmentation of the phones’ prices based on ‘battery\_power’. It seems that phones of all price ranges are equally distributed across all ranges of ‘battery\_power’.

#### 

Figure 34: 4D Scatter Plot from the angle of ‘int\_memory’

From the angle of ‘int\_memory’, it is also difficult to discover any obvious layers horizontally. We can say that the ‘price\_range’ target variable is not mainly affected by the phone’s ‘int\_memory’.

Therefore, we can conclude the main findings from this 4D Visualisation: it is now apparent that the ‘price\_range’ target variable has a relatively strong relationship with ‘ram’. Hence, it is now understandable why only the ‘ram’ variable has the highest worth from Section 4.2.1.1 StatExplore, as even the variable ‘battery\_power’ with the second highest worth does not show any obvious relationship with the target.

#### 4.2.2 Correlation Analysis



Figure 35: Flow of Correlation Analysis

It is imperative that a correlation analysis is done before proceeding with more complex work. Correlation analysis helps us understand the relationships between features. We are more concerned about the relationship between our target variable and the other variables/ features. We first use the **‘File Import’** node (renamed to ‘Cleaned Dataset’) to read our cleaned dataset.

Under the ‘Cleaned Dataset’ node, ‘Edit Variables…’ is chosen to change the role of the ‘price\_range’ variable to ‘Target’. Other variables remain the same.



Figure 36: Change ‘price\_range’ Role to Target

The data is then passed into the **‘Variable Selection’** node to perform the correlation analysis.

Before running the node, under the ‘Properties’ panel, the ‘Target Model’ property is changed from ‘Default’ to ‘R and Chi-square’. The reason ‘R and Chi-square’ is chosen is because the target variable is categorical, but at the same time, there are continuous features to consider too.



Figure 37: Change Target Model from Default to R and Chi-square

Below are the results after running the **‘Variable Selection’** node:

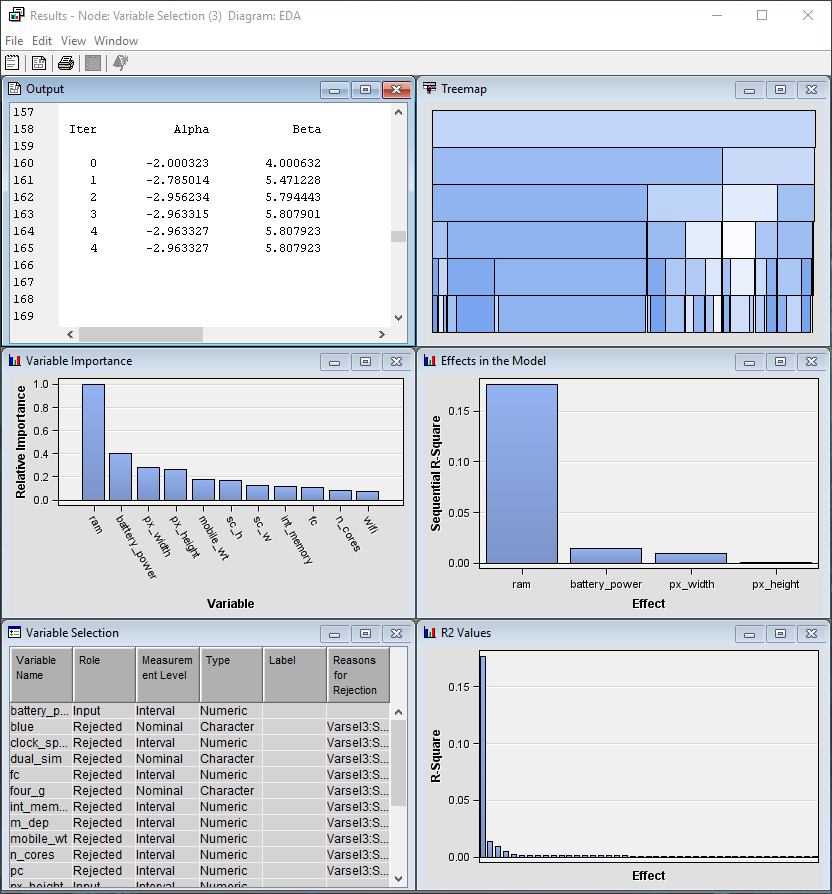


Figure 38: Results after Running ‘Variable Selection’ Node

##### 4.2.2.1 Variable Importance

Let’s first look at the ‘Variable Importance’ window:

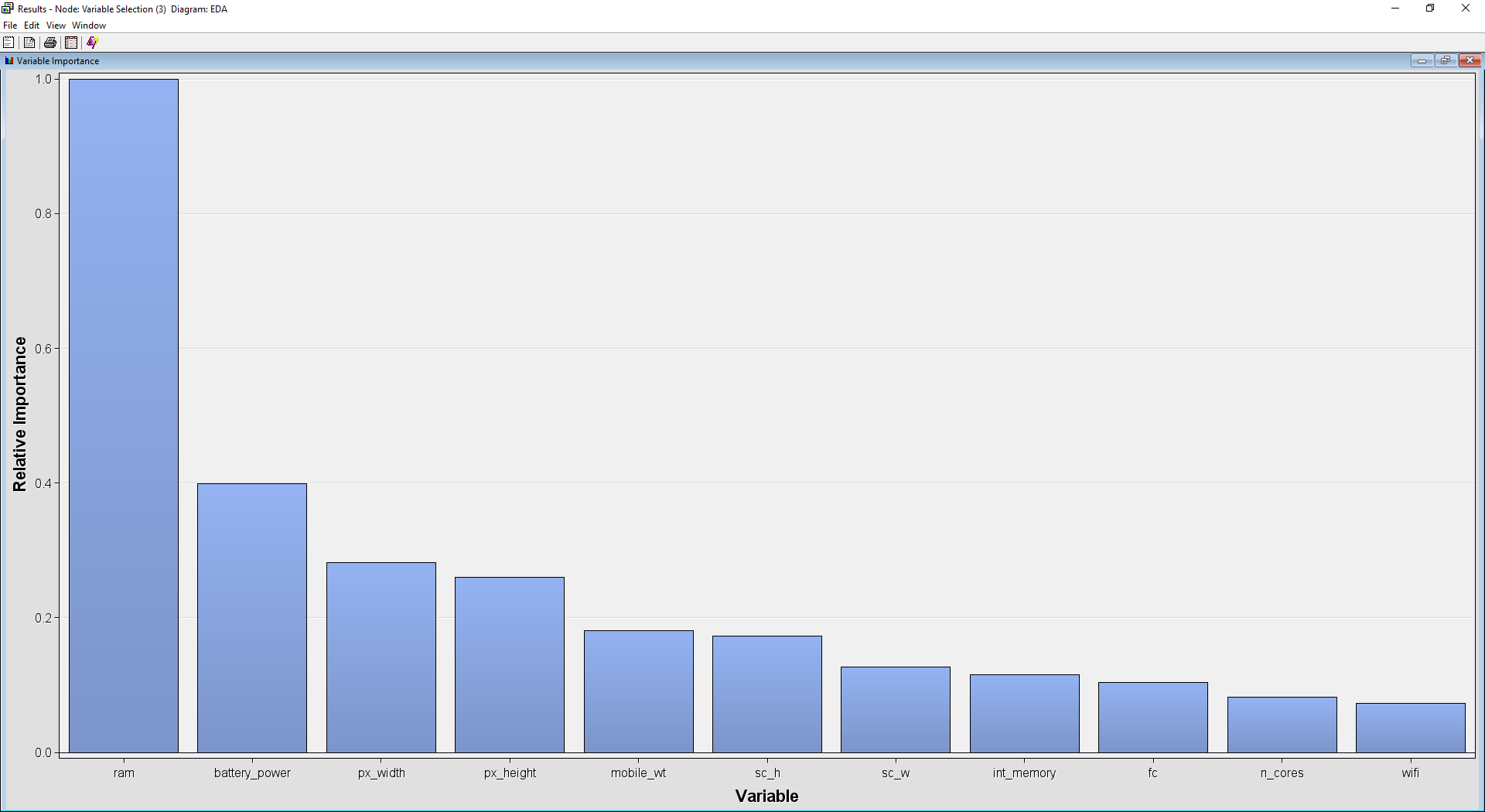


Figure 39: ‘Variable Importance’ Window

This window shows the variables relative importance based on their chi-squared scores in descending order. In our case, the variable ‘ram’ has the highest relative importance (by a wide margin) of 1, followed by battery\_power (0.3987), px\_width (0.2813), and px\_height (0.2599).

##### 4.2.2.2 R2 Values

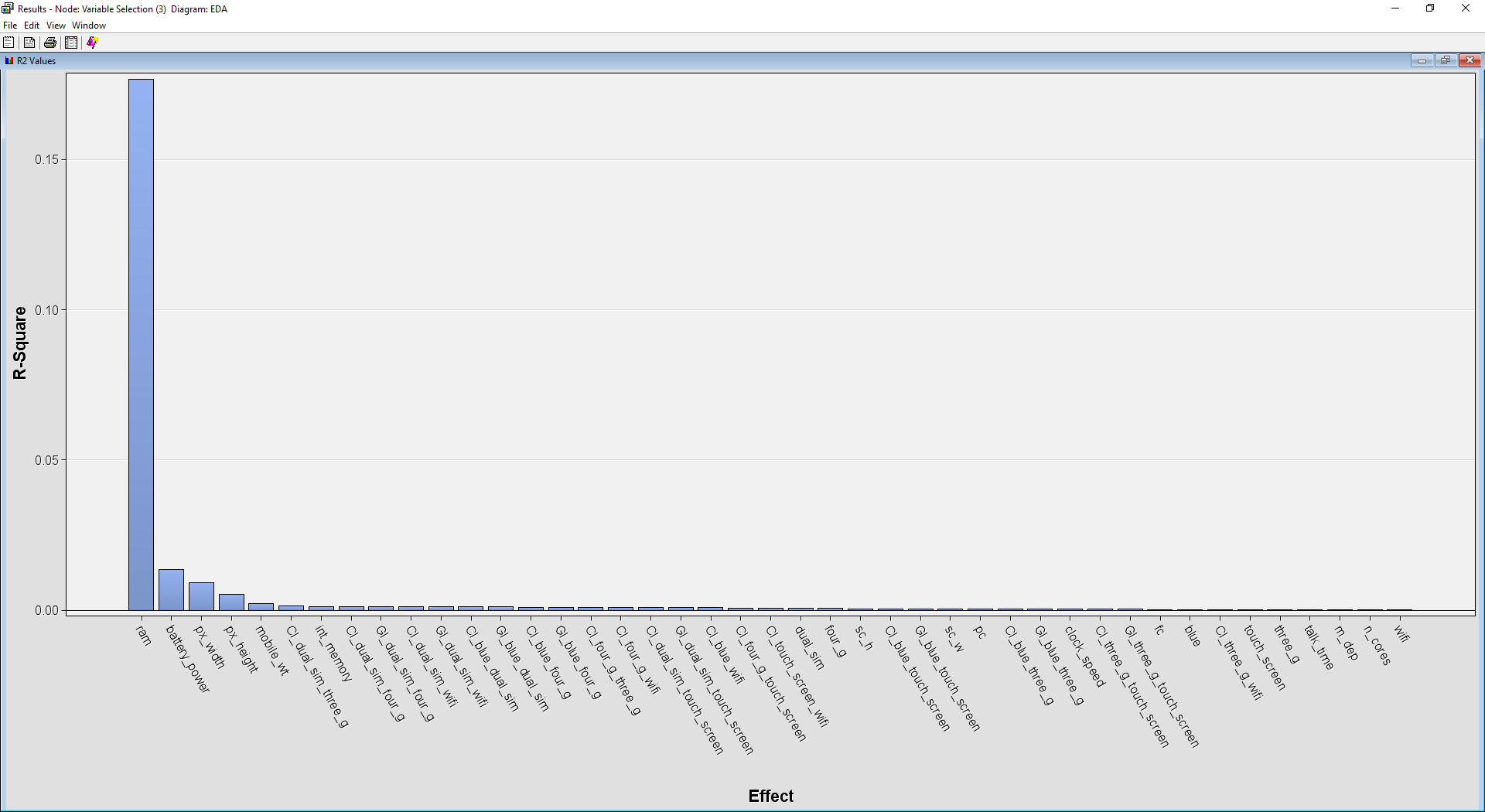


Figure 40: ‘R2 Values’ Window

We can observe that the variable ‘ram’ also has the highest R-Square value of 0.1768 (also by a wide margin). The following few variables are ‘battery\_power’ (0.0136), ‘px\_width’ (0.0092), ‘px\_height’ (0.0052), and ‘mobile\_wt’ (0.0023). All the variables have very low R-squared values. Hence, it is almost certain that these features are not sufficient and feature engineering is needed in the later processes.

##### 4.2.2.3 Variable Selection Table

##### 

Figure 41: ‘Variable Selection’ Window

This round of correlation analysis shows that only ‘battery\_power’, ‘px\_height’, ‘px\_width’, and ‘ram’ will be selected if we were to proceed with the ‘Model’ stage with this dataset. In the documentation, although it is stated that variables with a relative importance of greater than or equal to 0.05 will be assigned an Input role by default, some variables are still rejected because of this reason ‘Varsel3:Small R-square value’. The remaining variables are rejected too due to the small Chi-square value as well.

##### 4.2.2.4 Effects in the Model

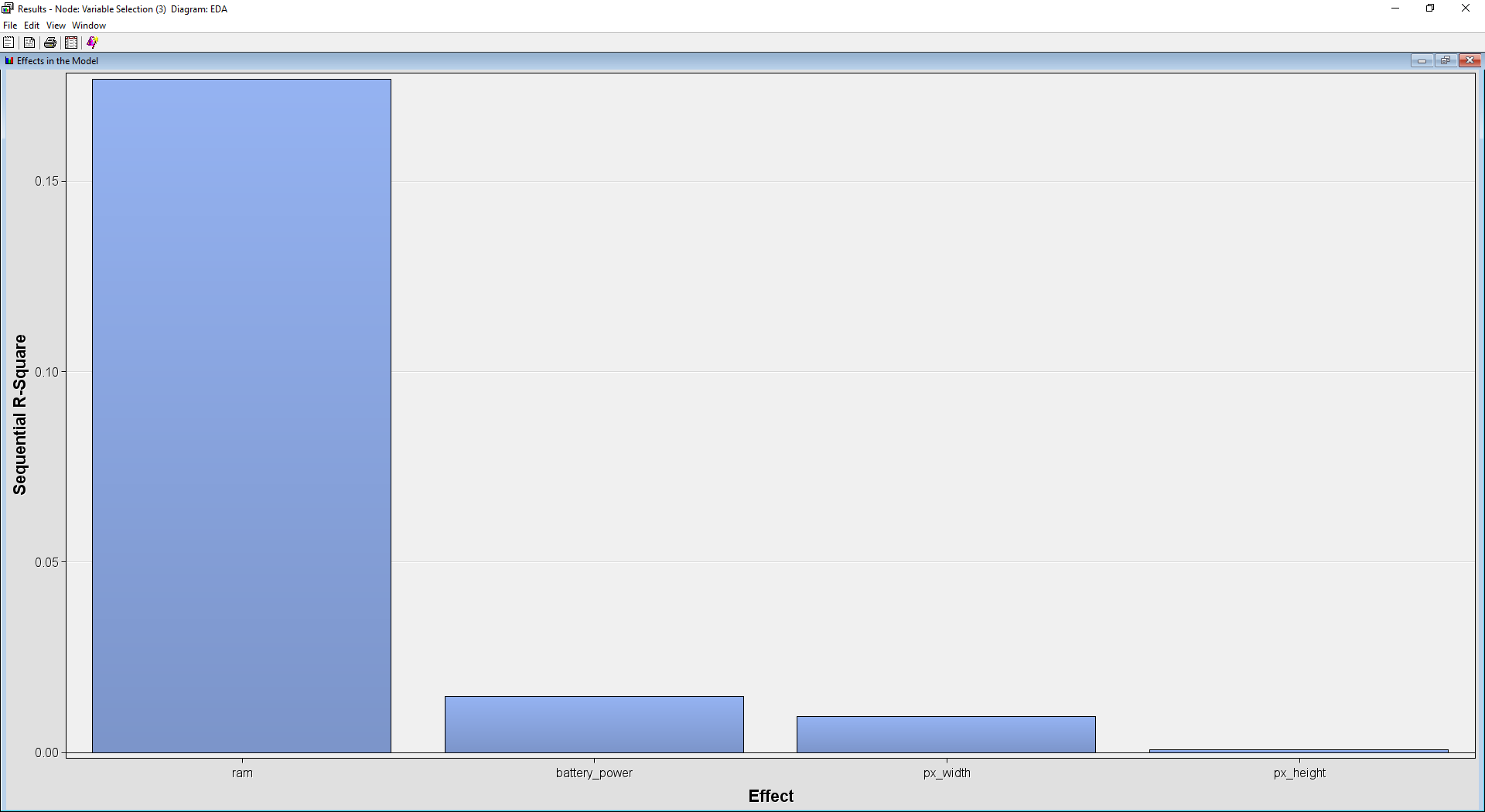


Figure 42: ‘Effects in the Model’ Window

This window only shows the variables which were assigned the role Input. It displays the sequential R-squared values from highest to lowest: ‘ram’ (0.1768), ‘battery\_power’ (0.0147), ‘px\_width (0.0094)’, ‘px\_height’ (0.0006).

##### 4.2.2.5 Output Window

Finally, the ‘Output Window’ displays the detailed information about the ‘Variable Selection’ run.

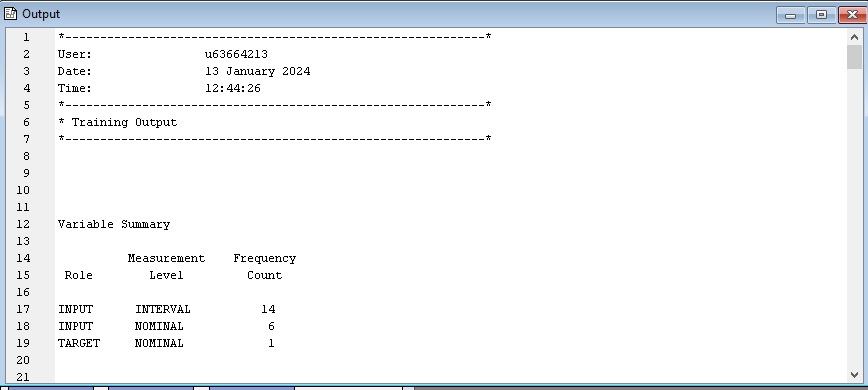


Figure 43: ‘Output’ Window

Here are some of its more important details:

* Variable Summary

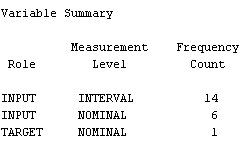


Table 44: Variable Summary

This section describes the role of the variables, its measurement type, and the number of such variables.

* R-Squares for Target Variable

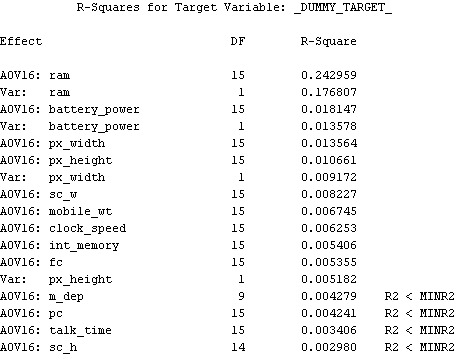


Table 45: R-Squares for Target Variable

This table describes the R-Square values for the target variable and has the columns for Effect, Degrees of Freedom (DF), and R-square scores. If a variable effect’s R-square value is less than the predefined R-square minimum, the information is noted in the last column (R2 < MINR2).

* Effects Chosen for Target

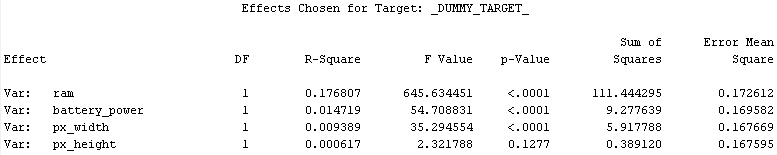


Table 46: Effects Chosen for Target

This table describes the effects or variables which have been chosen as the target variable’s predictors. It displays the variable name, DF, R-square, F and p-values, Sum of Squares and Error Mean Square of each of the selected effects.

* ANOVA Table for Target

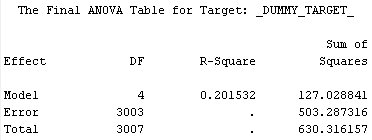


Table 47: Final ANOVA Table for Target

This ANOVA table shows the DF, R-Square values, and Sum of Squares for the model.

* Estimating Logistic

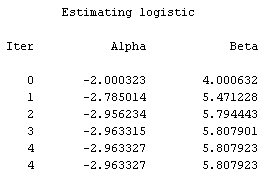


Table 48: Estimating Logistic

This table lists the alpha and beta estimators for early iterations.

* Classification Table for Cutoff

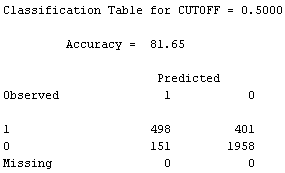


Table 49: Classification Table for CUTOFF = 0.5000

The cutoff for this table is 0.5 and the predictions are 81.65% accurate compared to the training data. The table shows observed vs predicted values for the target variable using the training data.

* Effect Summary

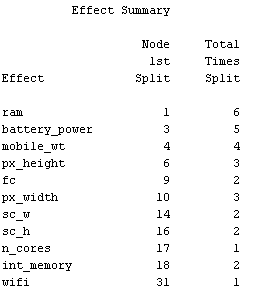


Table 50: Effect Summary

This table has the Effect, Node 1st Split and Total Times Split columns.

It is worth noting that the correlation analysis here only serves as a basic overview of the relationship between the target variable and other variables in the dataset. In later stages, feature engineering is conducted using FeatureTools and more features will be introduced. Hence, the findings in this stage are merely for exploration purposes.

#### 4.2.3 Association Rule Mining

Association rule mining is a data mining technique that aims to discover interesting relationships, patterns, and correlations within large datasets. It focuses on identifying strong associations between different items or variables in the data. This process entails revealing relationships between variables and leveraging these relationships for predictive analysis or decision-making. By identifying associations between variables, association rule mining can help users understand the relationships between different variables and how those variables may be related to one another.

##### 4.2.3.1 Dataset Used

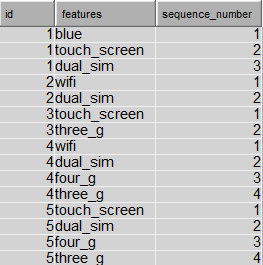


Figure 51: Transformed Data for Association Rule Mining and Sequence Analysis

Before importing the data source into SAS Enterprise Miner, the data is transformed to a format that is more suitable for association rule mining and sequence analysis. The original data had columns representing different features (blue, touch\_screen, wifi, dual\_sim, four\_g, three\_g) for each ID. Each row indicated whether the feature was present (TRUE) or absent (FALSE). However, it is not suitable for association rule mining and as each row in the original data represents an ID, and the features are not inherently transactional. It is also not suitable for sequence analysis because it lacks a temporal or sequential order. It doesn't provide information about the sequence in which features were activated. Hence, a new data is created with 3 columns, **‘id’**, **‘features’**, and **‘sequence\_number’**. For each row in the original data, if the feature was present (TRUE), the feature will be included in the transformed data. Sequence number is assigned to each feature within a specific ID, representing the order or sequence in which the features occur within an ID.

##### 4.2.3.2 Process Flow & Variable Selection



Figure 52: Association Rule Mining Process Flow

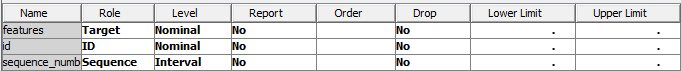


Figure 53: Variables in ‘File Import’ node

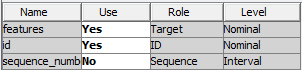


Figure 54: Variables in ‘Association’ node

The **‘File Import’** node and **‘Association’** node are used for Association Rule Mining. Firstly, the role for each variable is assigned in the ‘File Import’ node. The ‘id’ column is set as ‘ID’ role, ‘features’ column as ‘Target’ role and ‘sequence\_number’ column as ‘Sequence’ role. In the ‘Association’ node, the use of ‘sequence\_number’ is set to no as only two variables are used for association rule mining which are the **‘id’** and **‘features’**.

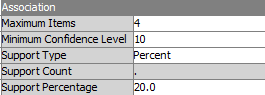


Figure 55: Properties Panel for Association

The properties of association are set as default except the support percentage. The support percentage is changed to **20%** instead of the default at 5%. The increase in support percentage indicates a higher level of stringency or strictness in the rule generation process. This emphasises the focus on stronger and more reliable patterns in the data. As the support percentage increases, the number of rules will decrease as well, a smaller set of rules can be more manageable and interpretable, making it easier to focus on the most relevant associations.

##### 4.2.3.3 Output Window

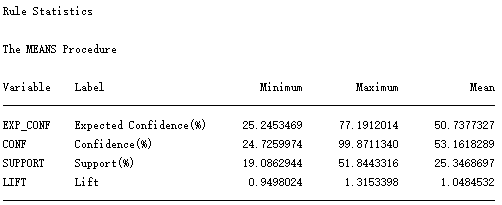


Figure 56: Association Rule Statistic

The statistics above are the measures that are commonly used to evaluate association rule mining. Expected Confidence represents the expected likelihood that the consequent of a rule is true given the antecedent. Confidence indicates how frequently the if-then rule is found to be true in the dataset. Support indicates how frequently the items occur in the dataset. Lift is the ratio of confidence to the proportion of all samples that are covered by the consequence.

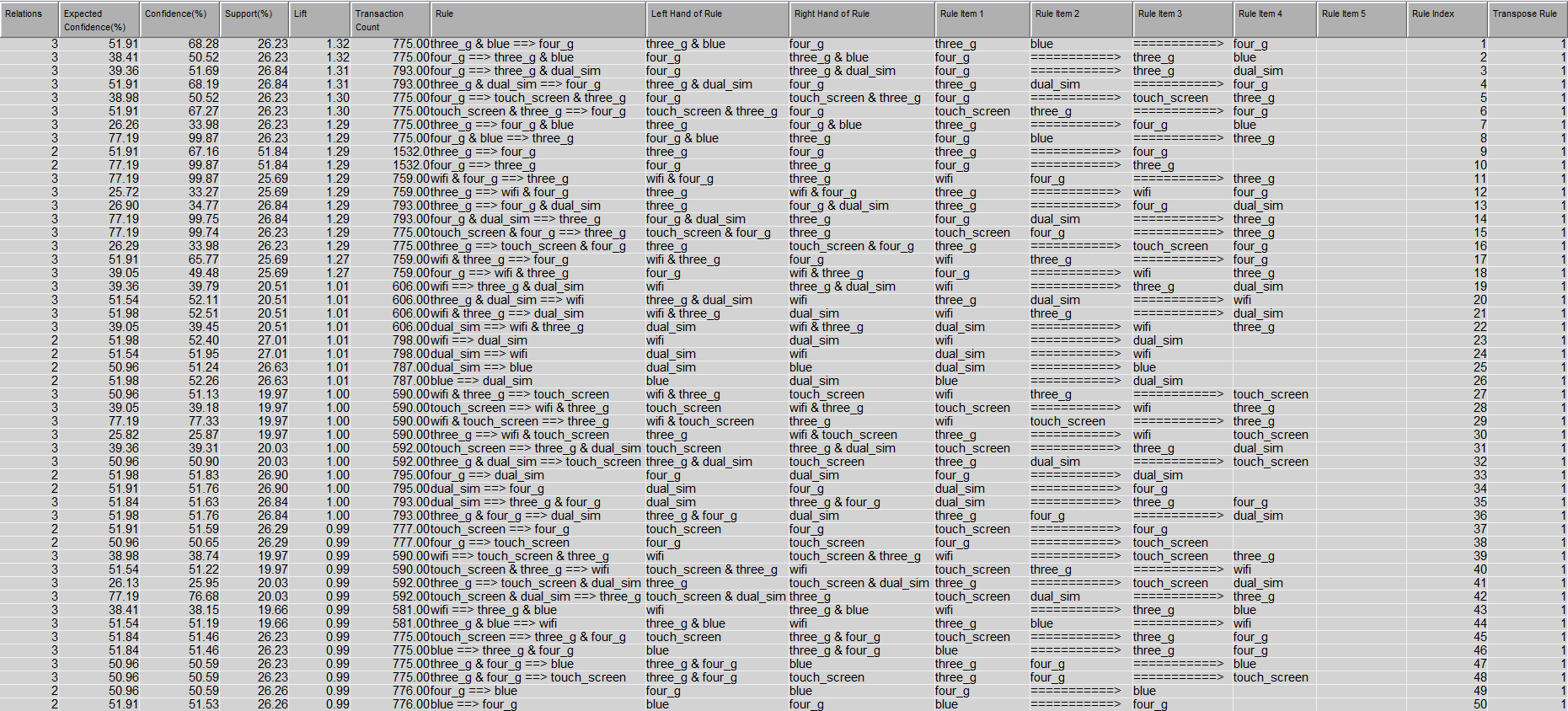




Figure 57 & 58: Association Rule Table

The association rule table contains information about association rules, including measures such as expected confidence, confidence, support, lift, and the count of transactions that support each rule. There are a total of 90 association rules generated. There are a total of 6 features included in the association rules, which are:

1. dual\_sim
2. blue (bluetooth)
3. three\_g (3G)
4. four\_g (4G)
5. touch\_screen
6. wifi

The rules are arranged according to the lift in descending order. There are a total of **26 rules** with positive association in which the lift is greater than 1 while another 64 rules suggest a negative association or no association. The following is the explanation for some association rules:

Rule 1:



This rule suggests that when a mobile has features "three\_g" and "blue", there is an expected confidence of 51.91% that it will also have the feature "four\_g". The observed confidence is 68.28%, meaning that in 68.28% of cases where "three\_g" and "blue" are present, "four\_g" is also present. The support of 26.23% indicates that this rule is true for 26.23% of transactions. The lift of 1.32 suggests a positive association between the antecedent and consequent.

Rule 5:



This rule implies that when a mobile has the feature "four\_g", there is an expected confidence of 38.98% that it will also have features "touch\_screen" and "three\_g". 50.52% of cases where "four\_g" is present, "touch\_screen" and "three\_g" are also present. This rule is true for 26.23% of transactions.

Rule 24:



This rule implies that when a mobile has the feature "wifi", there is an expected confidence of 51.98% that it will also have the feature "dual\_sim". In 52.40% of cases where "wifi" is present, "dual\_sim" is also present. This rule is true for 27.01% of transactions.

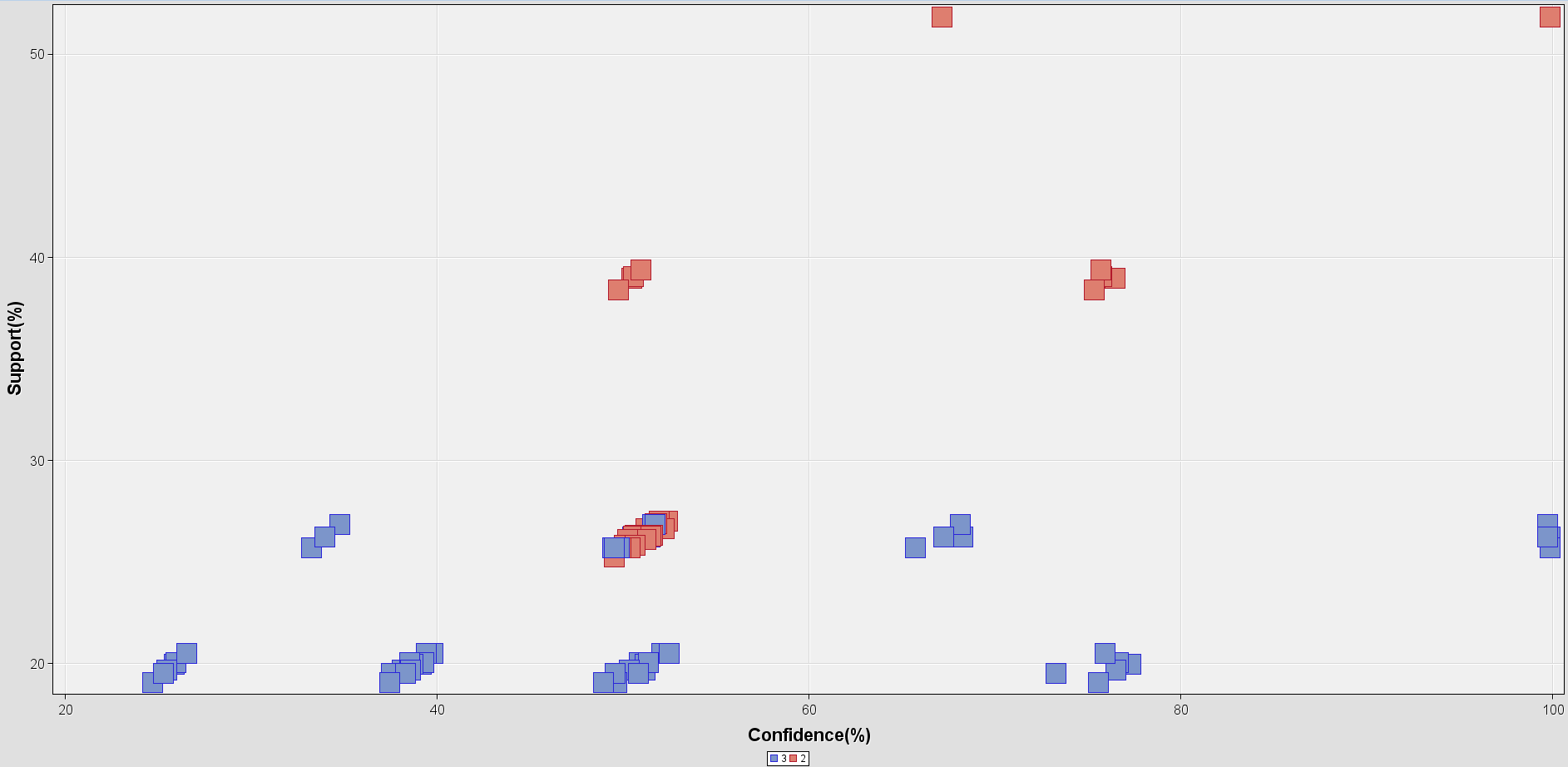


Figure 59: Statistics Plot

The figure above shows the statistics plot of the association rule mining. The labels below are the number of relations. The rule on the top right corner has the highest confidence level (99.87%) and support percentage (51.84%), which is **“four\_g ⇒ three\_g”**. This indicates a strong relationship between having the feature "four\_g" and also having the feature "three\_g".

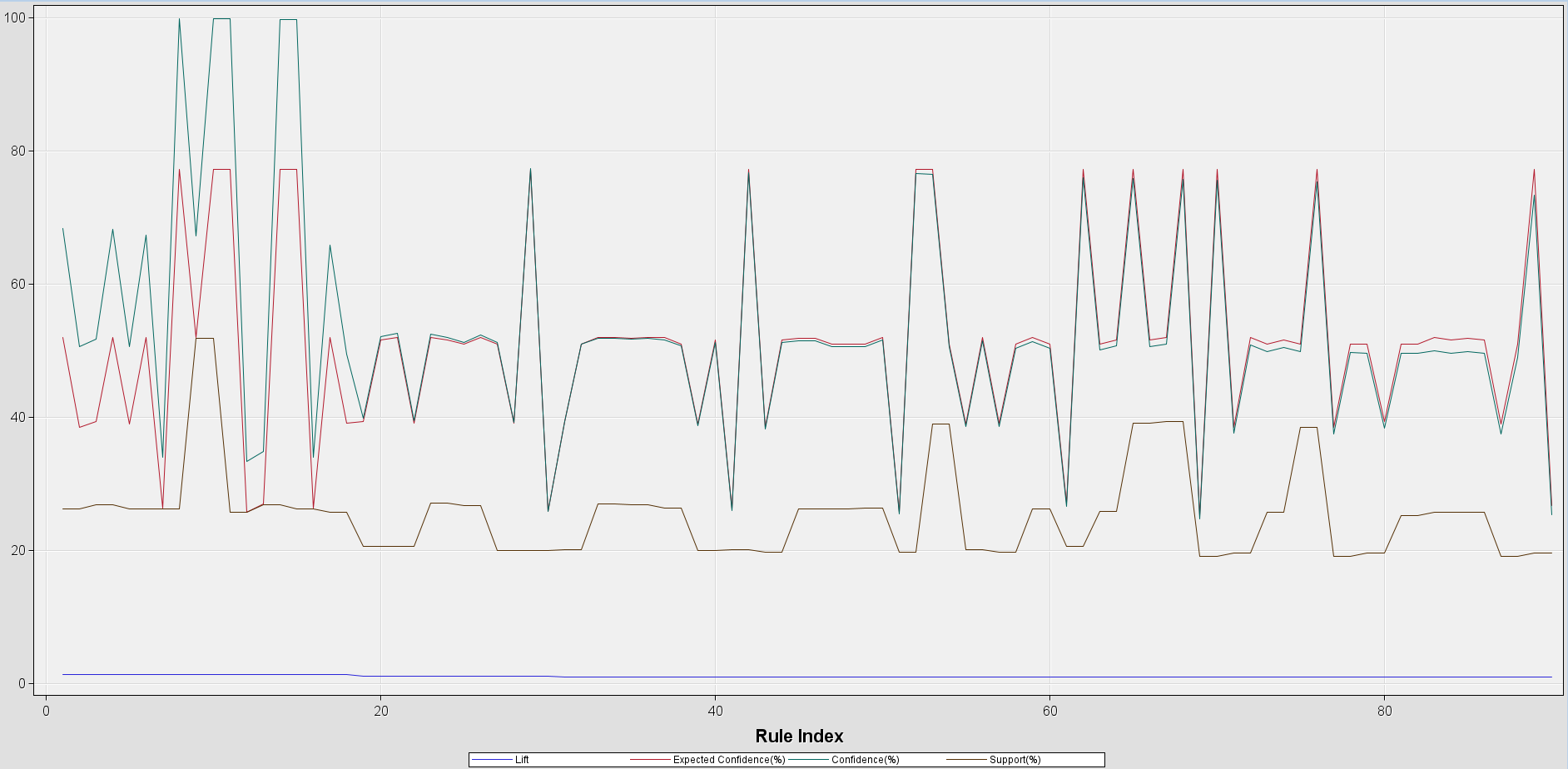


Figure 60: Statistics Line Plot

A statistics line plot for association rule mining typically visualises key metrics such as lift, expected confidence, confidence and support across a range of rules.

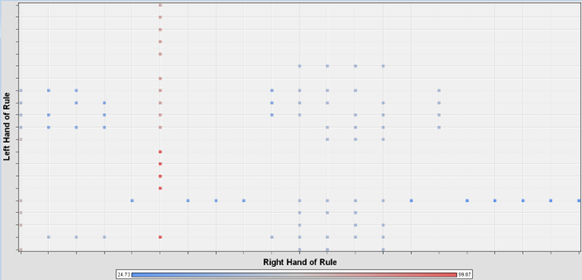


Figure 61: Association Rule Matrix

In association rule matrix, each row typically represents a unique antecedent (left hand of rule), and each column represents a unique consequent (right hand of rule).The cell at the intersection of a row and column contains the association rule that involves the corresponding antecedent and consequent. This matrix layout can quickly identify which antecedents are associated with which consequent and examine the associated statistical measures such as confidence for each rule. The intensity of the colour indicates the level of confidence for each rule, in which the dots with higher colour intensity have higher confidence level. In the above rule matrix, we can see that the left hand of rule **"four\_g"** is associated with many right hand of rules such as "three\_g & blue", "wifi", "touch\_screen & three\_g" and more.

#### 4.2.4 Sequence Analysis

Sequence analysis is a technique used to discover patterns, trends, and relationships within sequential data, providing valuable insights for decision-making and process improvement. Sequential data consists of ordered sequences of events or items, such as time-stamped transaction data.

##### 4.2.4.1 Dataset Used

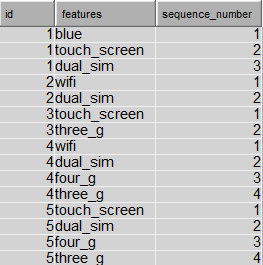


Figure 62: Transformed Data for Association Rule Mining and Sequence Analysis

The data used for sequence analysis is the same as the association rule. The data transformation steps are mentioned in the association rule mining part. The original data is not suitable for sequence analysis as it lacks temporal or sequential order. It doesn't provide information about the sequence in which features were activated. While for the transformed data, sequence number is assigned to each feature within a specific ID, representing the order or sequence in which the features occur within an ID.

##### 4.2.4.2 Process Flow & Variable Selection



Figure 63: Sequence Analysis Process Flow

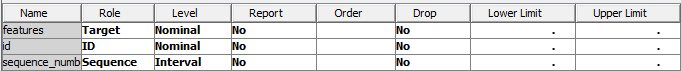


Figure 64: Variables in ‘File Import’ node

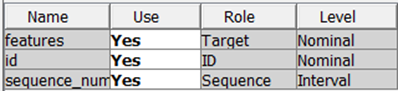


Figure 65: Variables in ‘Association’ node

The **‘File Import’** node and **‘Association’** node are used for Sequence Analysis. Firstly, the role for each variable is assigned in the ‘File Import’ node. The **‘id’** column is set as ‘ID’ role, **‘features’** column as ‘Target’ role and **‘sequence\_number’** column as ‘Sequence’ role. In the ‘Association’ node, all three variables are used for sequence analysis.

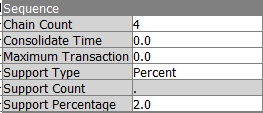


Figure 66: Properties Panel for Sequence Analysis

The properties of sequence analysis are set as default except the chain count. The chain count is increased to **4** instead of the default at 3. This is because longer chain count indicates longer length of sequences being analysed. Longer sequences may capture more complex and informative patterns in the data, providing more specific insights into the order and relationships between events. Longer sequences may also lead to more refined association rules. Rules generated from longer sequences are likely to be more specific and accurate, as they consider a greater number of events in the context of each other.

##### 4.2.4.3 Output Window

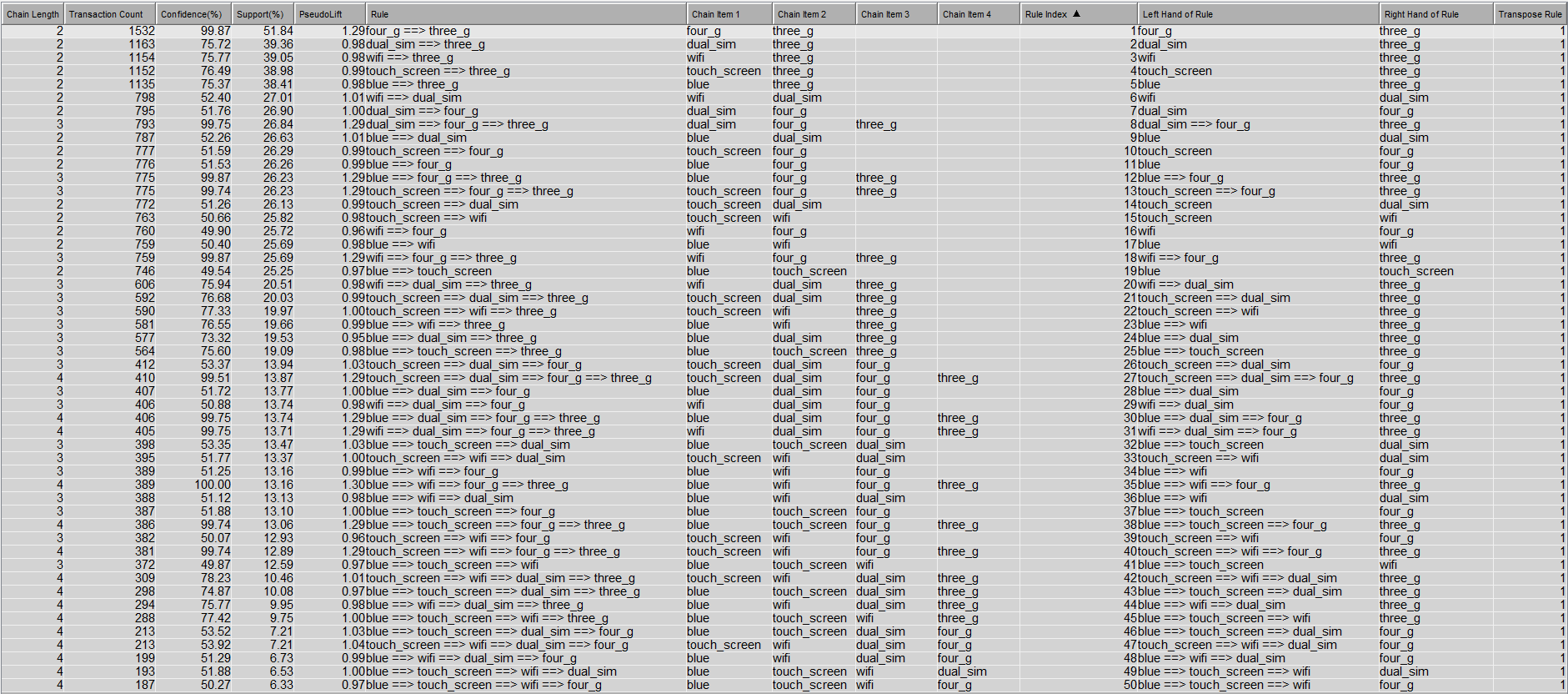


Figure 67: Sequence Analysis Rule Table

The sequence analysis rule table contains information about sequential patterns or sequences of events discovered in a dataset, including measures such as chain length, transaction count, confidence, support and pseudo lift that support each rule. Chain length indicates the length of the sequence or the number of items in each sequence. There are a total of 50 sequential rules generated. There are a total of **18 rules** with positive association in which the lift is greater than 1 while another 32 rules suggest a negative association or no association. The following is the explanation for some sequential rules:

Rule 1:



The sequence "four\_g ==> three\_g" is a common pattern, appearing in 51.84% of the transactions. The extremely high confidence (99.87%) indicates a very reliable association. If "four\_g" is present, there is almost a guaranteed presence of "three\_g" in the same transaction. The pseudo lift of 1.29 further emphasizes the strength of the association, indicating a positive relationship between the two events.

Rule 32:



The sequence "blue ==> touch\_screen ==> dual\_sim" is present in approximately 13.47% of transactions, indicating a moderate level of occurrence. The confidence of 53.35% suggests a moderate reliability in the sequential relationship. If "blue" is present, there is a likelihood of both "touch\_screen" and "dual\_sim" being present in the same transaction. The pseudo lift of 1.03 indicates a slight positive association, suggesting a slightly increased likelihood of the subsequent items in the sequence occurring together.

Rule 40:



The sequence "touch\_screen ==> wifi ==> four\_g ==> three\_g" is present in approximately 12.89% of transactions, indicating a notable occurrence. The extremely high confidence of 99.74% suggests an exceptionally reliable sequential relationship. If the initial items are present, there is almost a guaranteed presence of "three\_g" in the same transaction.

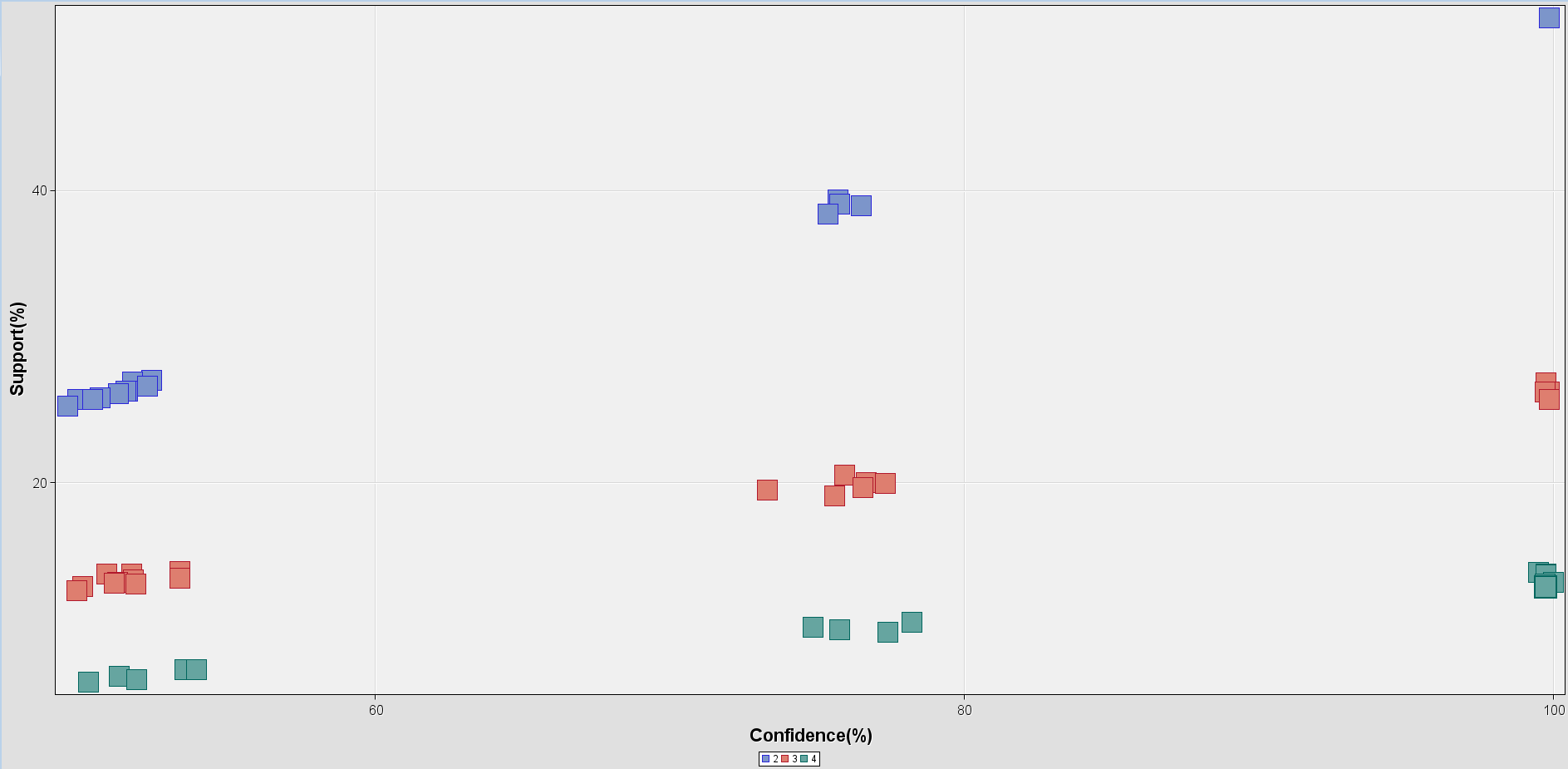


Figure 68: Statistics Plot

The figure above shows the statistics plot of the sequence analysis. The labels below are the chain length. The rule on the top right corner has the highest confidence level (99.87%) and support percentage (51.84%), which is **“four\_g ⇒ three\_g”**. This indicates a strong sequential relationship between "four\_g" and "three\_g".

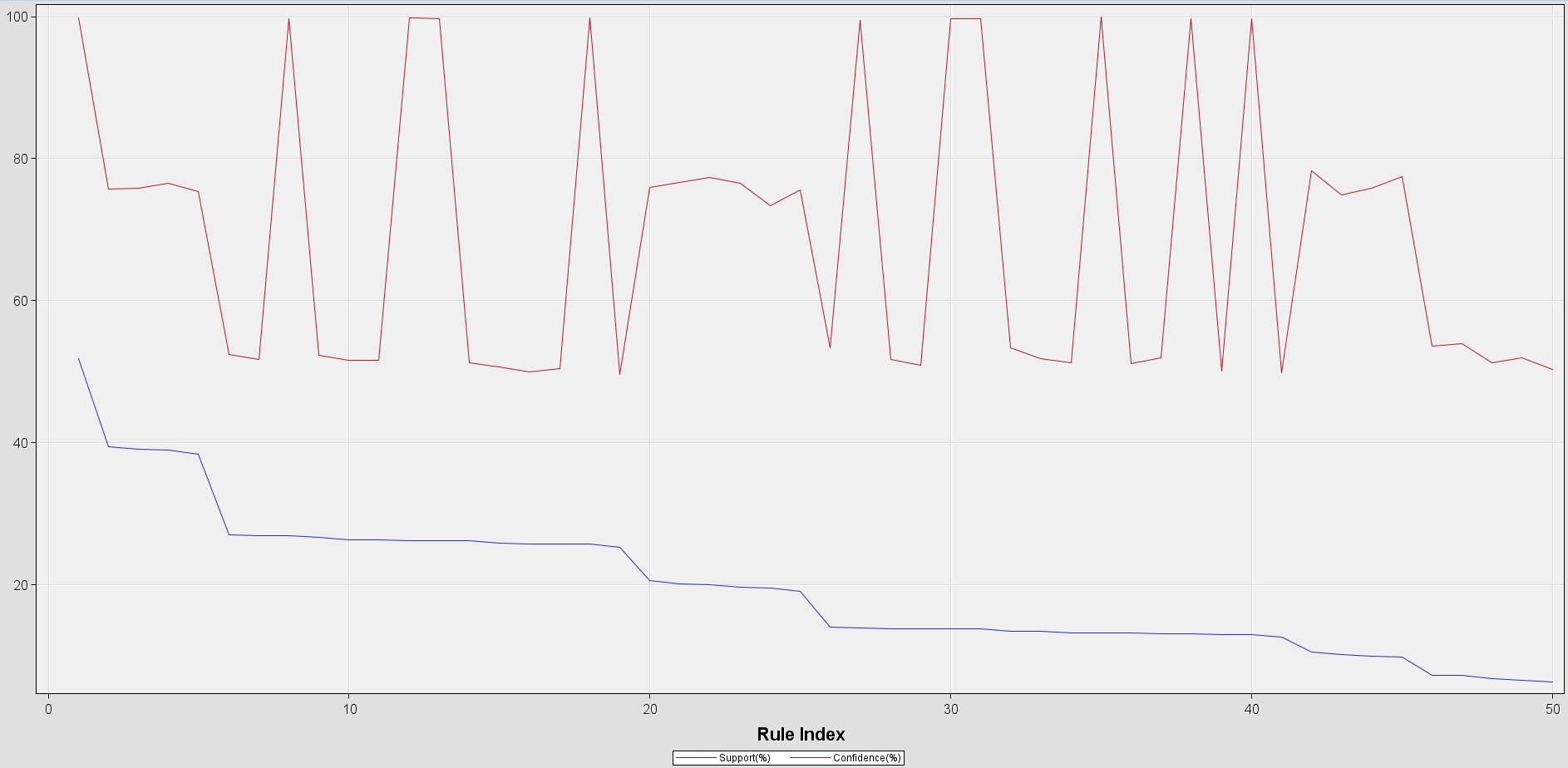


Figure 69: Statistics Line Plot

A statistics line plot for sequence analysis typically visualises key metrics such as confidence and support across a range of sequential rules.

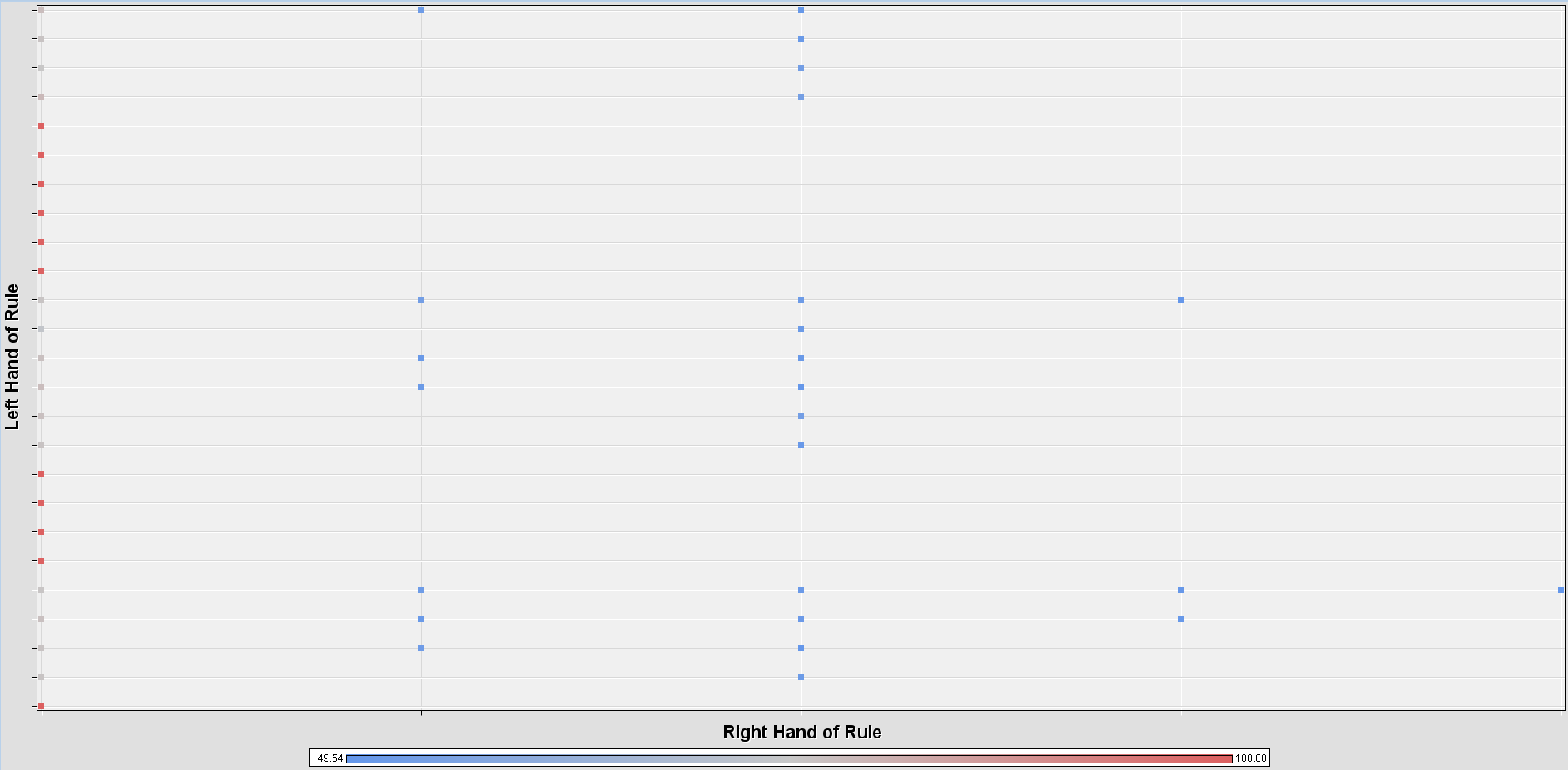


Figure 70: Sequential Rule Matrix

In the sequential rule matrix above, we can see that the right hand of rule **"four\_g"** is associated with many left hand of rules such as "blue ⇒ touch\_screen ⇒ wifi", "blue ⇒ dual\_sim", "touch\_screen" and more.

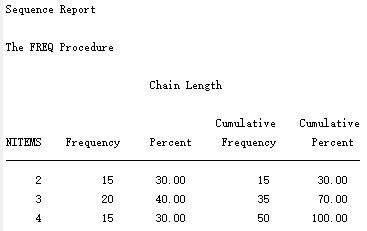


Figure 71: Sequence Analysis Report

The report summarises the distribution of chain lengths in a dataset obtained through a sequence analysis. NITEMS represents the number of items in each chain or sequence. Frequency indicates how many sequences have the corresponding NITEMS. Percent represents the percentage of sequences that have the given number of items relative to the total number of sequences. From the report, we can see that there are 15 sequences in the dataset with a length of 2 items. These sequences account for **30.00%** of all sequences. This is the smallest chain length category. While 20 sequences with a length of 3 items which account for **40.00%** of all sequences. When combined with the sequences of length 2, the cumulative percentage reaches 70.00%. Lastly, there are 15 sequences with a length of 4 items which account for **30.00%** of all sequences. When combined with the sequences of lengths 2 and 3, the cumulative percentage reaches 100.00%.

#### 4.2.5 Time Series (TS) Clustering

The time series clustering analyses sequences of data points over time and identifies similarities or patterns among these sequences. Through time series clustering, the similarities or shared characteristics among mobile price ranges can be observed. The time series clustering is applied to 5 features (i.e., battery power, internal memory, primary camera, RAM and the number of cores) to compare how they differ among the mobile price range.

There are 4 nodes which were utilised in building the time series clustering path, namely the ‘File Import’ node, the ‘TS Data Preparation’ node, the ‘Metadata’ node and the ‘TS Similarity’ node (refer to Figure 72).

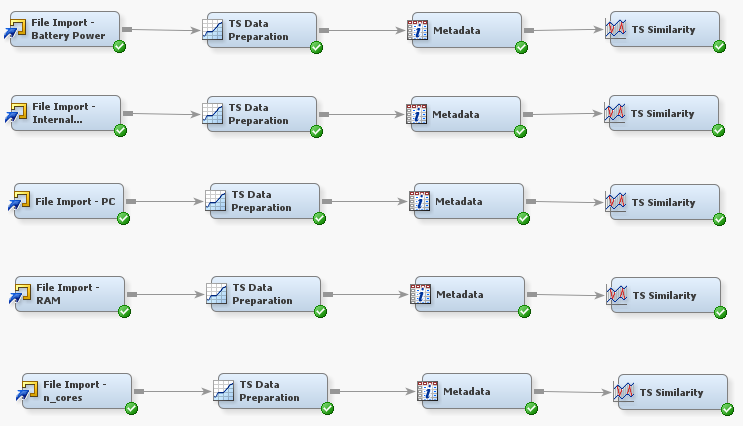


Figure 72: TS clustering process flow

Firstly, the roles of each feature are assigned at the ‘File Import’ node. The ‘Date’ and the ‘price\_range’ features are assigned ‘Time ID’ and ‘CrossID’ as their roles respectively. Whereas, the role of the 5 selected features will be assigned as ‘Target’ while the rest of the features will be assigned as ‘Rejected’. Next, the ‘Transpose’ option in the ‘TS Data Preparation’ node is set to ‘Yes’ (refer to Figure 73).

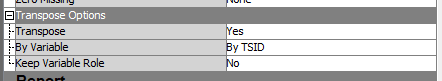


Figure 73: TS Data Preparation node’s transpose option

The similarities analysis is performed between the target series and the input series. The ‘Low Cost’ time series is the target in the ‘Metadata’ node (refer to Figure 74). Lastly, the ‘TS Similarity’ node is run and the results are obtained, which will be discussed in the section below.

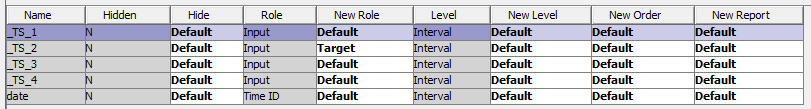


Figure 74: TS metadata node

##### 4.2.5.1 Battery Power

| **Image Description** | **Image** |
| --- | --- |
| Process flow |  |
| Roles assignment |  |
| TSID Map Table |  |
| Target series vs input series |  |
| Similarity measure |  |

Table 3: TS Clustering Evidence and Results - battery power

##### 4.2.5.2 Internal Memory

| **Image Description** | **Image** |
| --- | --- |
| Process flow |  |
| Roles assignment |  |
| TSID Map Table |  |
| Target series vs input series |  |
| Similarity measure |  |

Table 4: TS Clustering Evidence and Results - internal memory

##### 4.2.5.3 Primary Camera

| **Image Description** | **Image** |
| --- | --- |
| Process flow |  |
| Roles assignment |  |
| TSID Map Table |  |
| Target series vs input series |  |
| Similarity measure |  |

Table 5: TS Clustering Evidence and Results - primary camera

##### 4.2.5.4 RAM

| **Image Description** | **Image** |
| --- | --- |
| Process flow |  |
| Roles assignment |  |
| TSID Map Table |  |
| Target series vs input series |  |
| Similarity measure |  |

Table 6: TS Clustering Evidence and Results - RAM

##### 4.2.5.5 Number of Cores

| **Image Description** | **Image** |
| --- | --- |
| Process flow |  |
| Roles assignment |  |
| TSID Map Table |  |
| Target series vs input series |  |
| Similarity measure |  |

Table 7: TS Clustering Evidence and Results - number of cores

##### 4.2.5.6 Overall Results

Based on Table 3, 4, 5, 6 and 7, the ‘TSID Map Table’ shows the input variables grouping associated with each time series. Thus, ‘TS\_1’ is labelled as ‘battery\_power\_1’, ‘int\_memory\_1’, ‘pc\_1’, ‘ram\_1’ and ‘n\_cores\_1’ for each of the target features selected representing the ’high cost’ price range variable, ‘TS\_2’ is labelled as ‘battery\_power\_2’, ‘int\_memory\_2’, ‘pc\_2’, ‘ram\_2’ and ‘n\_cores\_2’ for each of the target features selected representing the ‘low cost’ price range variable, ‘TS\_3’ is labelled as ‘battery\_power\_3’, ‘int\_memory\_3’, ‘pc\_3’, ‘ram\_3’ and ‘n\_cores\_3’ for each of the target features selected representing the ‘medium cost’ price range variable and ‘TS\_4’ is labelled as ‘battery\_power\_4’, ‘int\_memory\_4’, ‘pc\_4’, ‘ram\_4’ and ‘n\_cores\_4’ for each of the target features selected representing the ‘very high cost’ price range variable.

The ‘similarity measure’ is a method used to quantify the similarities of time series data to help identify relationships between the time series variables. The similarity measure used in this project is the squared deviation which is the default selection to calculate the distance between the time series. The smaller the similarity measure, the closer the similarities between the variables. In Table 3, 4, 5, 6 and 7, ‘TS\_2’ and ‘TS\_4’ have the closest similarities.

In conclusion, it can be observed that the low-cost mobile (TS\_2) and very high-cost mobile (TS\_4) have similar features despite being the lowest and the highest price range respectively. Despite having the highest mobile price range, it has very similar battery power, internal memory, primary camera, RAM and the number of cores as the lowest mobile price range. Therefore, despite the price difference, the features of the mobile phones are highly similar.

#### 4.2.6 DBSCAN

DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. It groups data points into clusters, based on their density and distance between points, and isolates points that are not part of the cluster (noise). The key terms in DBSCAN are ‘core points’, ‘border points and ‘noise’. These are determined based on epsilon, and MinPt values. The DBSCAN is performed using the KNIME Analytics Platform (refer to Figure 75).

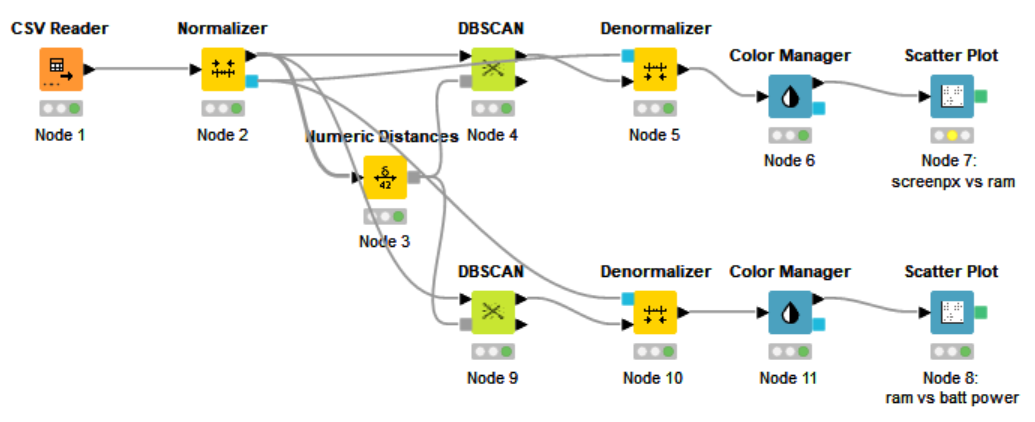


Figure 75: Workflow of DBSCAN

The relationship between the feature ‘Pixel Resolution Height’ and ‘RAM’ is explored. In the DBSCAN node, the epsilon is set to 1.25 and the minimum points are set to 3 (refer to Figure 76). The results are represented in a scatter plot (Figure 78). Data points are grouped into 4 clusters. Both low and high RAM show variability in Pixel Resolution Height, with lower Pixel Resolution Height having denser clusters.

Additionally, the relationship between the feature ‘Battery Power’ and ‘RAM’ is explored. In the DBSCAN node, the epsilon is set to 1.41 and the minimum points are set to 5 (refer to Figure 77). The results are represented in a scatter plot (Figure 79). Data points are grouped into 2 clusters. Both low and high RAM show similar variability in battery power, with lower RAM having a larger cluster (Cluster\_1).

Based on the results obtained in both of the scatter plots, the clusters primarily differentiate mobile phones based on their RAM characteristics. This suggests that RAM plays a more significant role in determining the grouping of devices in this dataset as compared to the other 2 variables. Therefore, it can be concluded that the mobile phone price determination is primarily based on the RAM.

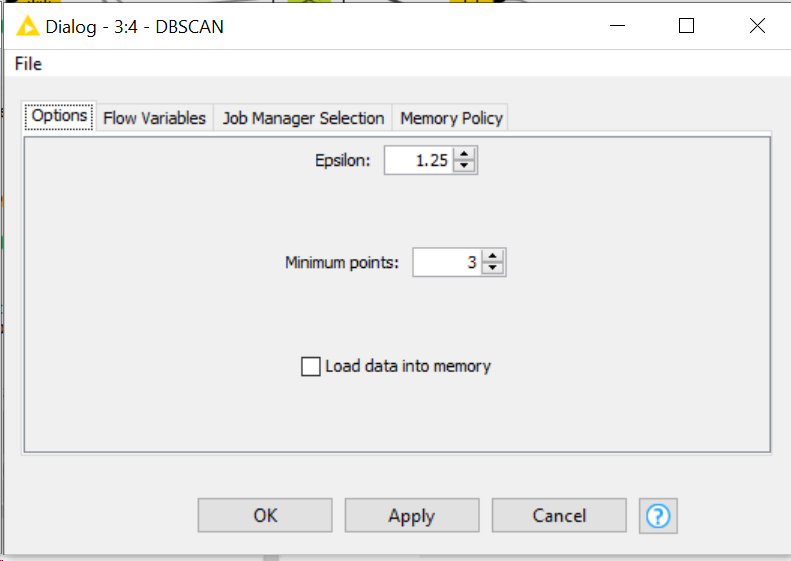
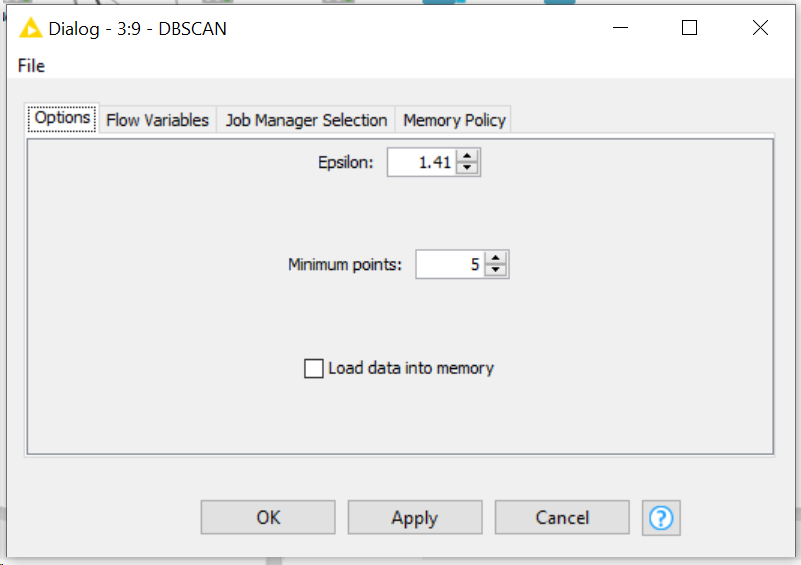
 

Figure 76 & 77: DBSCAN parameters

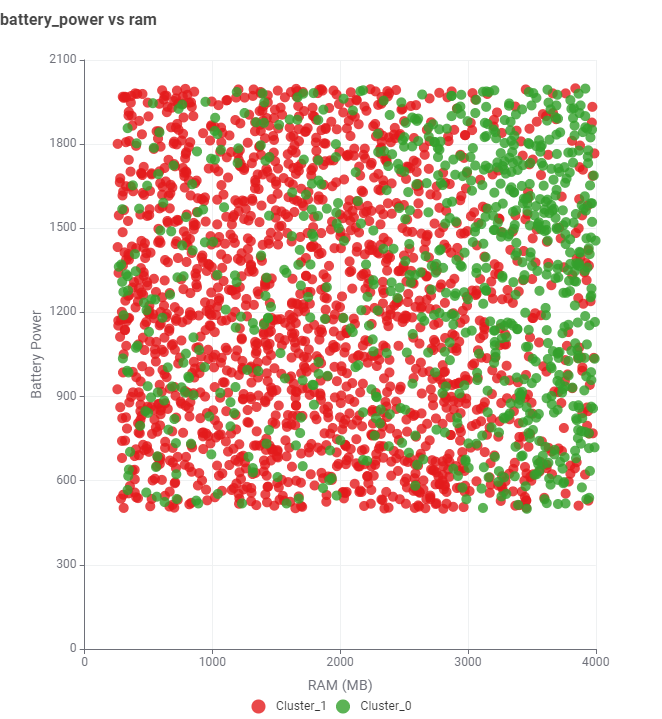
 

Figure 78 & 79: Results of DBSCAN clustering

### 

### 4.3 Modify

In the Modify phase, we will prepare the data for deeper analysis and modelling, this involves transforming existing variables as well as filtering the data. We will concentrate on five main steps during this stage: converting data types and encoding labels, engineering new features, creating a star schema for data organisation, dividing the data into training and validation sets and selecting the most useful features for our model.

#### 4.3.1 Data Type Conversion and Label Encoding

Before moving to the modelling phase, we notice that certain categorical variables are in string format. Specifically, columns like “blue”, “touch\_screen”, “wifi”, “dual\_sim”, “four\_g”, and “three\_g” are strings indicating Yes or No. Meanwhile, “price\_range” is divided into four categories: “low cost”, “medium cost”, “high cost”, and “very high cost”. Converting these string values to numerical form is crucial for modelling as most algorithms work better with numbers. Numerical representation allows for more efficient computation and is necessary for accurately interpreting the relationship between features and the target variable in predictive modelling.

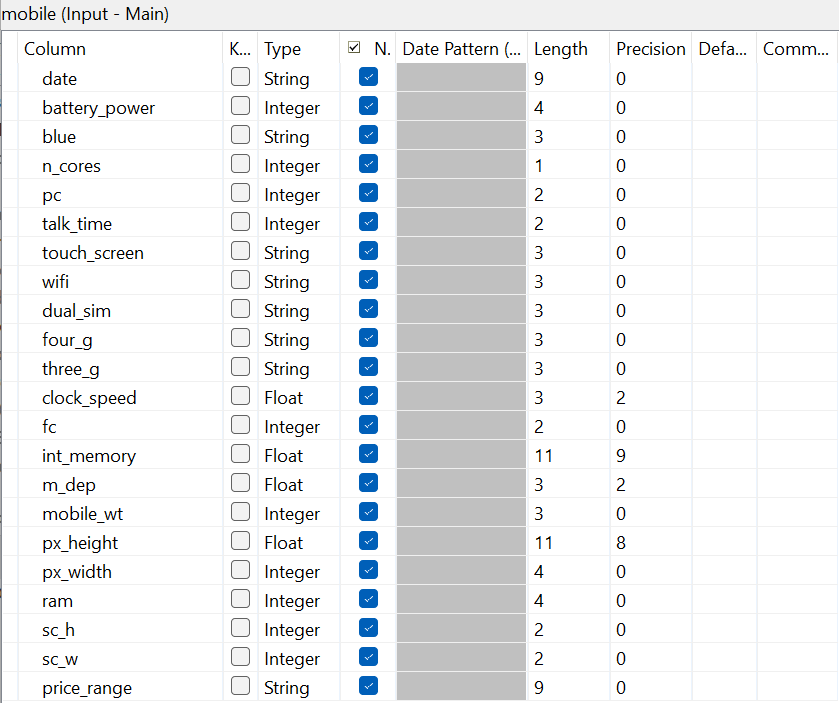


Figure 80: Summary of Imputed Dataset

To change string data into boolean values and numerically encode the “price\_range” categories, we will use Talend Integration. We start by setting up new metadata for the dataset we have imputed and then import it. After it is imported, we read the data using a tFileInputDelimited component in the job design area.

Next, we will use a tMap component to handle data type conversion and label encoding. We drag this into our workspace and link it to our data, then identify the columns needing conversion. For our project, this includes “blue”, “touch\_screen”, “wifi”, “dual\_sim”, “four\_g”, “three\_g” and “price\_range”. The remaining columns will remain as they are. The image below shows the setup for the mapping process in Talend Integration.

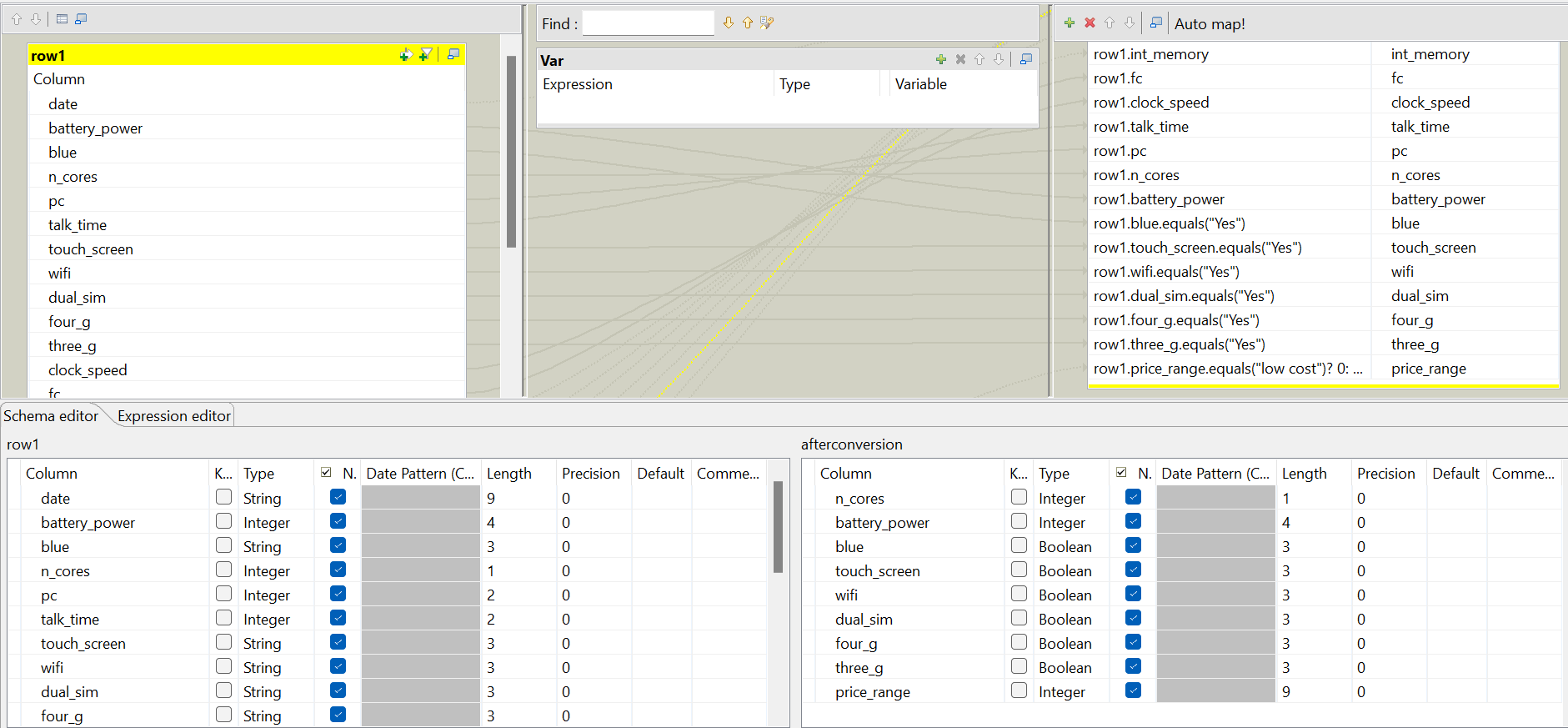


Figure 81: Mapping Configuration Setup using Talend Integration

To carry out the data type conversion and label encoding, we have to input the appropriate code expressions during the mapping stage. For changing to boolean data types, we use the boolean expressions shown in Figure 82. For the label encoding, we apply if-else statements as shown in Figure 83.

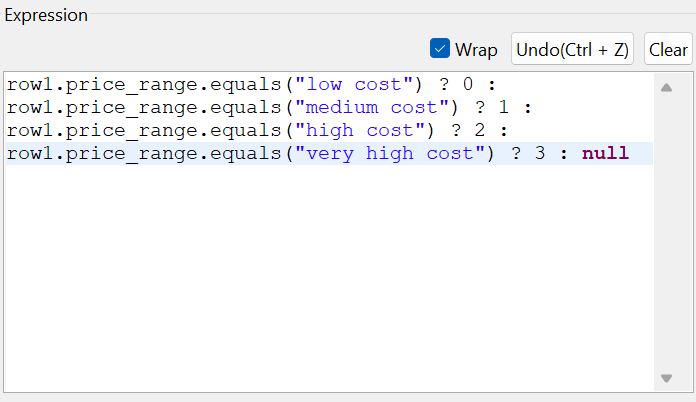
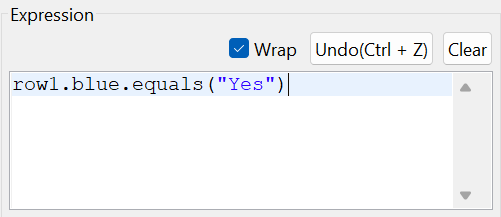


Figure 82 & 83: Code Snippets for Data Type Conversion and Label Encoding

During the process of converting data types, we change the “blue”, “touch\_screen”, “wifi”, “dual\_sim”, “four\_g” and “three\_g” columns from the string ‘Yes’ to the boolean TRUE and ‘No’ to FALSE. For label encoding, we assign numbers to the “price\_range” categories: 0 for “low cost”, 1 for “medium cost”, 2 for “high cost” and 3 for “very high cost”.

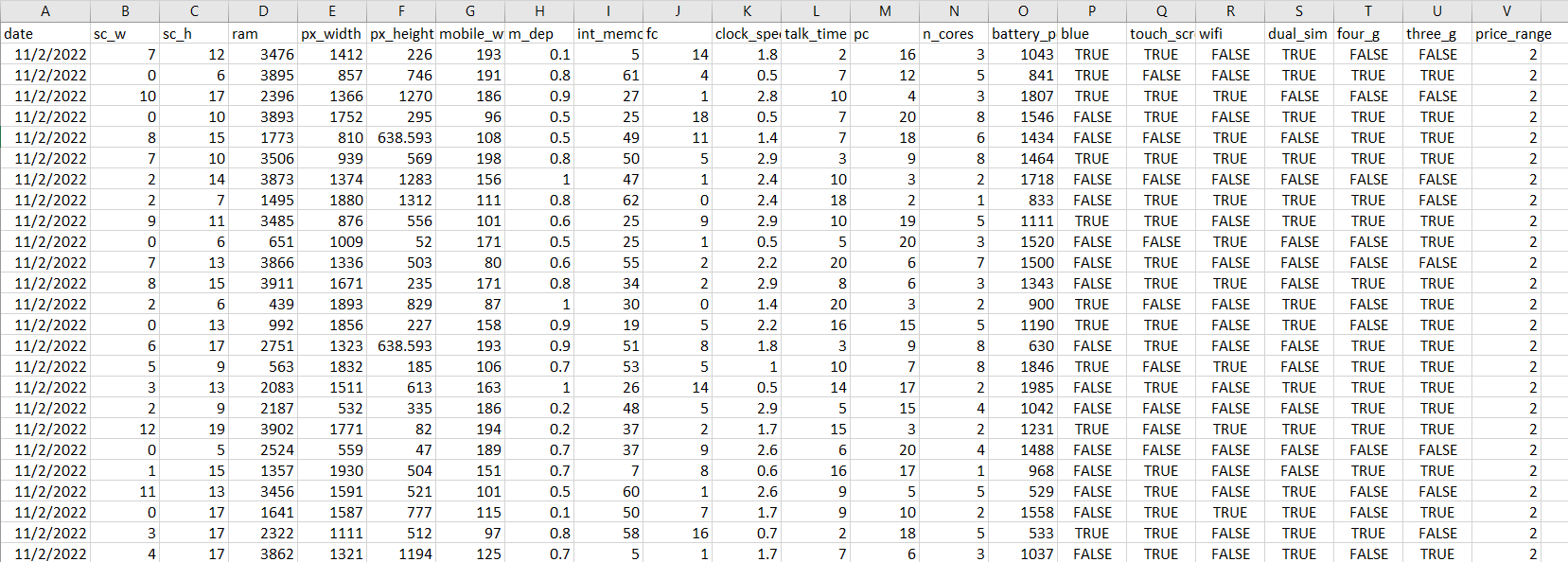


Figure 84: Transformed Dataset After Data Type Conversion and Label Encoding

To save our updated dataset, we will utilise a tFileOutputDelimited component to set the save location and file name, which we call "conversion.csv".

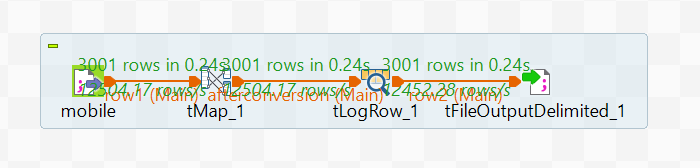


Figure 85: Data Type Conversion and Label Encoding using Talend Integration

#### 4.3.2 Feature Engineering

For the feature engineering stage, we will use FeatureTools, a Python library, to automate the creation of new attributes. This approach will help us enhance our dataset by unveiling additional insights that could improve the performance of our models. FeatureTools simplifies the process of generating new features from existing data, making it a practical choice for efficiently expanding our dataset with meaningful attributes.

##### 4.3.2.1 Dataset Examination

1. **Dataset**

A total of five data frames have been created from the dataset, specifically named “Camera”, “Memory”, “Screen”, “TechSpec” and “Sales”.

1. Camera:

The “Camera” data frame contains 3001 entries, each detailing various aspects of mobile phone cameras, such as CameraID, primary camera megapixels (pc), and front camera megapixels (fc), offering a comprehensive dataset on mobile phone camera specifications.

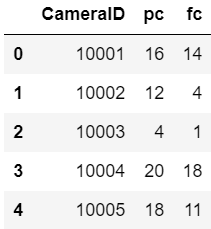


Figure 86: Camera Data Frame

1. Screen:

The “Screen” data frame contains 3001 entries, cataloguing distinct characteristics of mobile phone screens including ScreenID, screen height (sc\_h), screen width (sc\_w), pixel resolution width (px\_width), and pixel resolution height (px\_height), thus providing detailed information on mobile phone screen specifications.

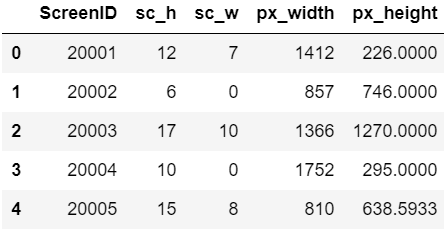


Figure 87: Screen Data Frame

1. Memory:

The “Memory” data frame contains 3001 records, each providing specific details about the memory features of mobile phones, such as MemoryID, RAM capacity (ram), and internal storage (int\_memory), presenting an extensive compilation of data on mobile phone memory attributes.

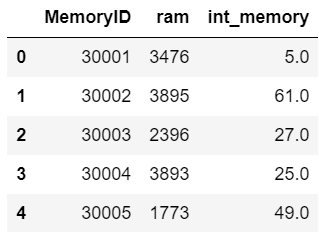


Figure 88: Memory Data Frame

1. TechSpec:

The “TechSpec” data frame includes 3001 entries, each providing a range of technical specs of mobile phones, such as TechSpecID, weight (mobile\_wt), depth (m\_depth), battery capacity (battery\_power), touchscreen availability (touch\_screen), processor speed (clock\_speed), number of cores (n\_cores), and connectivity options like WiFi (wifi), Bluetooth (blue), dual-sim (dual\_sim), 4G (four\_g), 3G (three\_g) and battery talk time (talk\_time), providing thorough information of mobile phone technical features.

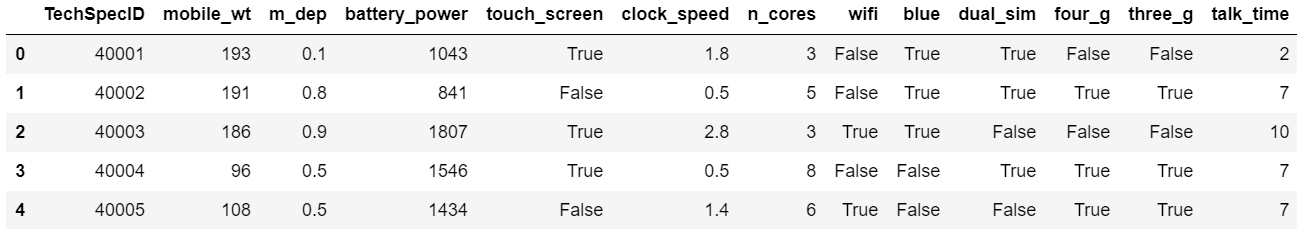


Figure 89: TechSpec Data Frame

1. Sales:

The “Sales” data frame, consisting of 3001 entries, acts as detailed transactions for mobile phone purchases. It incorporates a primary key (SalesID) and links to other data sets through foreign keys (CameraID, ScreenID, MemoryID, and TechSpecID). The table also notes the purchase date and price range, showing when each phone was sold and its cost category.

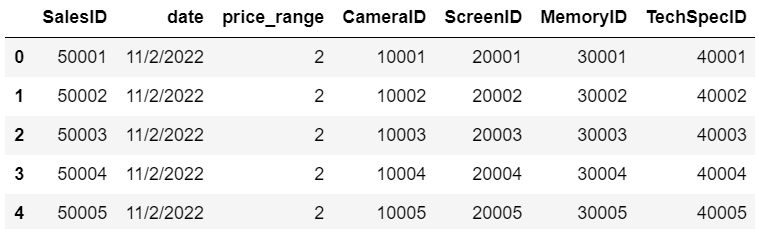


Figure 90: Sales Data Frame

1. **Business Rules**:

* Each camera record can be associated with one or more sales records.
* Each sales record is linked to one and only one camera record.
* Each memory record can be associated with one or more sales records.
* Each sales record is linked to one and only one memory record.
* Each screen record can be associated with one or more sales records.
* Each sales record is linked to one and only one screen record.
* Each techspec record can be associated with one or more sales records.
* Each sales record is linked to one and only one techspec record.

1. **Relationships**:
2. Camera:

| **Attribute** | **Data Type** | **Constraint** |
| --- | --- | --- |
| CameraID | INTEGER | PRIMARY KEY |
| pc | INTEGER |  |
| fc | INTEGER |  |

| **Relationship** | **Cardinality** |
| --- | --- |
| Camera has one-to-many relationship with Sales | (1,1) to (1,M) |

1. Screen:

| **Attribute** | **Data Type** | **Constraint** |
| --- | --- | --- |
| ScreenID | INTEGER | PRIMARY KEY |
| sc\_h | INTEGER |  |
| sc\_w | INTEGER |  |
| px\_width | INTEGER |  |
| px\_height | FLOAT |  |

| **Relationship** | **Cardinality** |
| --- | --- |
| Screen has one-to-many relationship with Sales | (1,1) to (1,M) |

1. Memory:

| **Attribute** | **Data Type** | **Constraint** |
| --- | --- | --- |
| MemoryID | INTEGER | PRIMARY KEY |
| ram | INTEGER |  |
| int\_memory | FLOAT |  |

| **Relationship** | **Cardinality** |
| --- | --- |
| Memory has one-to-many relationship with Sales | (1,1) to (1,M) |

1. TechSpec:

| **Attribute** | **Data Type** | **Constraint** |
| --- | --- | --- |
| TechSpecID | INTEGER | PRIMARY KEY |
| mobile\_wt | INTEGER |  |
| m\_dep | FLOAT |  |
| battery\_power | INTEGER |  |
| touch\_screen | BOOLEAN |  |
| clock\_speed | FLOAT |  |
| n\_cores | INTEGER |  |
| wifi | BOOLEAN |  |
| blue | BOOLEAN |  |
| dual\_sim | BOOLEAN |  |
| four\_g | BOOLEAN |  |
| three\_g | BOOLEAN |  |
| talk\_time | INTEGER |  |

| **Relationship** | **Cardinality** |
| --- | --- |
| TechSpec has one-to-many relationship with Sales | (1,1) to (1,M) |

1. Sales:

| **Attribute** | **Data Type** | **Constraint** |
| --- | --- | --- |
| SalesID | INTEGER | PRIMARY KEY |
| date | DATE |  |
| price\_range | INTEGER |  |
| CameraID | INTEGER | FOREIGN KEY |
| ScreenID | INTEGER | FOREIGN KEY |
| Memory\_ID | INTEGER | FOREIGN KEY |
| TechSpec\_ID | INTEGER | FOREIGN KEY |

| **Relationship** | **Cardinality** |
| --- | --- |
| Sales has one-to-many relationship with Camera | (1,1) to (1,1) |
| Sales has one-to-many relationship with Screen | (1,1) to (1,1) |
| Sales has one-to-many relationship with Memory | (1,1) to (1,1) |
| Sales has one-to-many relationship with TechSpec | (1,1) to (1,1) |

##### 4.3.2.2 Implementation of FeatureTools

1. **Define Entities and EntitySet**:
2. Import necessary library:

Import the FeatureTools library and name it as “ft” so that we can use it for automatic feature creation and managing data that are connected.



Figure 91: Import FeatureTools Library

1. Implement denormalization:

This code combines different data frames like “Sales”, “Camera”, “Screen”, “Memory” and “TechSpec” into one simple table using primary keys. It is created to put all the data in one data frame, making it easier to analyse and create new features, and follows a star schema layout.



Figure 92: Implement Denormalization

1. Create an entity set:

Initialise an empty EntitySet named ‘mobile’ to organise and manage related data entities for feature engineering.



Figure 93: Create Entity Set

1. Define entities:

This code incorporates the “Sales” data frame, which encompasses data from “Camera”, “Screen”, “Memory” and “TechSpec” into an EntitySet, defining it as an entity within the entity set. It employs the “denormalized” data frame with the “SalesID” column as a unique identifier.



Figure 94: Create Entities

The result is as below:

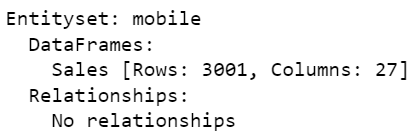


Figure 95: Outcome of ‘mobile’ Entity Set

1. **Establish Relationships**:

This code performs data normalisation to set up relationships between different sets of data. It breaks down and organises the data into separate data frames. The “Sales” data frame gets split into “Camera”, “Screen”, “Memory” and “TechSpec” data frames, and then links them together. This method makes it easier to connect and show complicated relationships, which is good for analysis and building models, much like in a star schema.

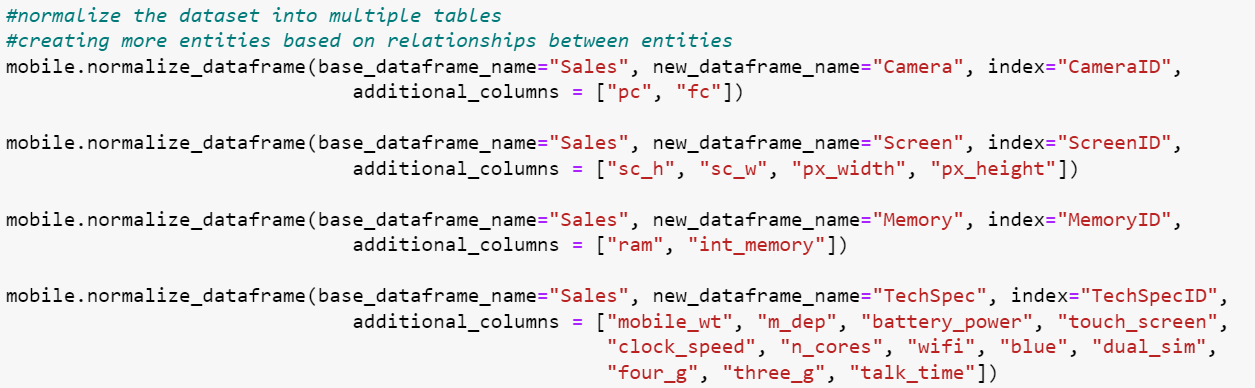


Figure 96: Normalisation from Entity Set

The result is as below:

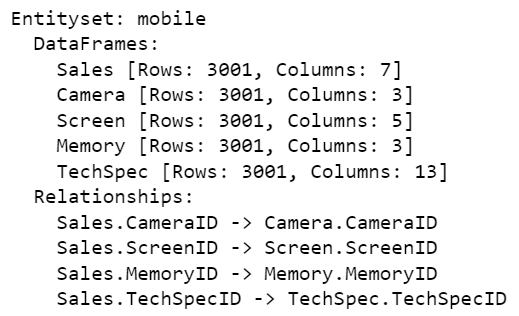


Figure 97: Normalisation from Entity Set

##### 4.3.2.3 Perform Deep Feature Synthesis

Deep Feature Synthesis (DFS) is a method for automatically creating complex and useful features from raw data, made possible by the FeatureTools library. It looks into the relationships in an EntitySet to come up with new features that reveal hidden patterns and links in the data. DFS considers various changes, groupings, and interactions between entities and their details to build a broad set of features.

DFS is specifically applied to the “Sales” data frame, with a ‘max\_depth’ parameter set at 2. This parameter constrains the complexity of feature generation, limiting it to two levels of aggregation or transformation. Additionally, the ‘n\_jobs’ parameter leverages up to three CPU cores for efficient parallel processing.

After running DFS, 62 features are created that bring out important information from the data. The outcomes, including these 35 features, are as shown below for further study.

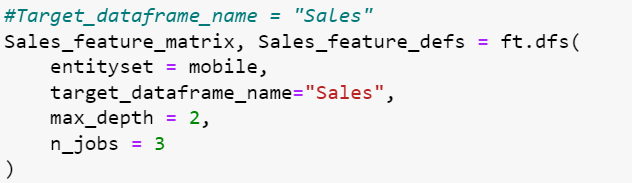


Figure 98: Implement Deep Features Synthesis

The result is as shown below:

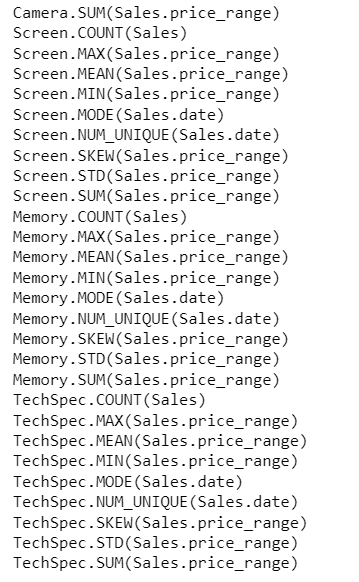
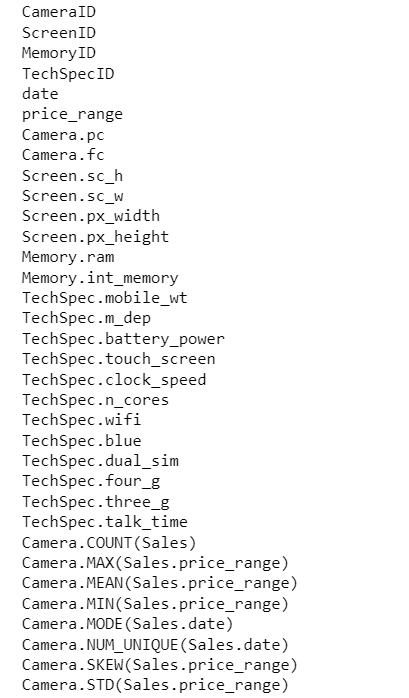


Figure 99 & 100: Features Generated from DFS

#### 4.3.3 Developing Star Schema Structure

In data warehousing, the star schema is a key structure that places a central fact table surrounded by many dimension tables. This layout is crucial for effectively organising complex data into a format that is easy to use and great for analysis. The star schema allows for faster query performances, simplifies business reporting, and can be easily understood by business users, facilitating better decision-making. It effectively separates process-oriented data in the fact table from descriptive data in dimension tables, making it a popular choice for database design.

##### 4.3.3.1 Design Data Model

At the core of the star schema is the fact table, which is central to the data model. It records measurable and quantifiable data that show how the business is doing. In this dataset, the “**Sales**” entity acts as this main **fact table**, keeping track of key transaction details.

Surrounding the fact table are the dimension tables, each representing a key business entity that provides context to the facts. These dimensions include:

* **Camera Dimension**: Identified by **CameraID**, this dimension enhances the dataset with specifics about camera quality, including pc (Primary Camera megapixels) and fc (Front Camera megapixels). This information aids in understanding the photographic capabilities and features of different mobile phones.
* **Screen Dimension**: Identified by **ScreenID**, this dimension contributes to the dataset with details about screen characteristics, including sc\_h (Screen Height), sc\_w (Screen Width), px\_width (Pixel Resolution Width), and px\_height (Pixel Resolution Height). These attributes help in analysing the display features and dimensions of various mobile phones.
* **Memory Dimension**: Identified by **MemoryID**, this dimension augments the dataset with details on memory capacity, including RAM (Random Access Memory in MegaBytes) and int\_memory (Internal Memory in GigaBytes). This information is vital for understanding the storage capabilities and performance potential of different mobile phones.
* **TechSpec Dimension**: Identified by **TechSpecID**, this dimension enriches the dataset with detailed technical specifications, including mobile\_wt (Weight of mobile phone), m\_dep (Mobile Depth), battery\_power, touch\_screen availability, clock\_speed, n\_cores (Number of processor cores), wifi, blue (Bluetooth), dual\_sim, four\_g, three\_g networks compatibility, and talk\_time. These attributes offer a comprehensive view of the mobile phones’ technical and functional capabilities, impacting user experience and performance.

##### 4.3.3.2 Define Business Rules

The following are the business rules we want to accomplish by utilising this star schema:

* To determine the popularity or demand of various mobile phone specifications by analysing the count of sales for each dimension
* To assess how different technical specifications of mobile phones correlate with their market price segments by examining the average price range
* To identify pricing volatility for different specifications by evaluating the standard deviation in price ranges
* To strategize inventory and marketing efforts by combining popularity and pricing data of various specifications.

##### 4.3.3.3 Create Star Schema

Figure 101 below displays the star schema with the “Sales” fact table and “Camera”, “Screen”, “TechSpec” and “Memory” dimension tables, including the newly created features.

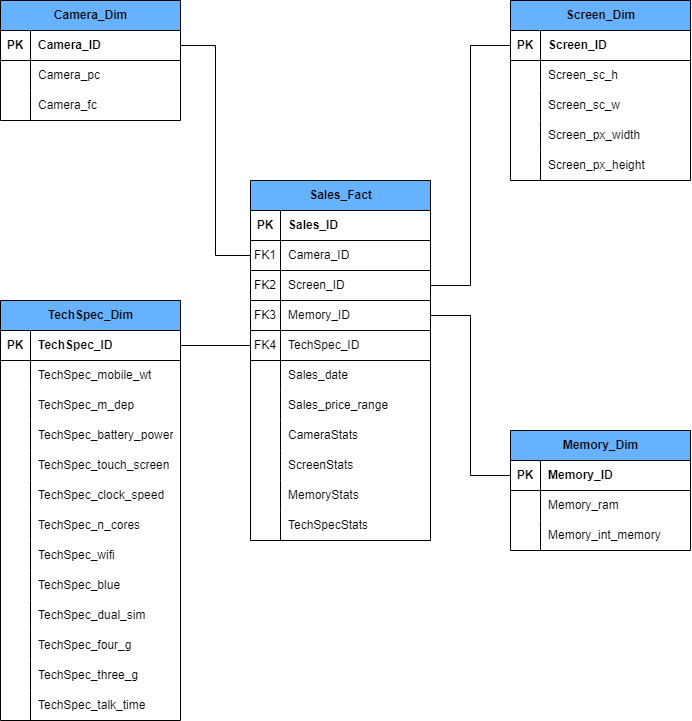


Figure 101: Star Schema

**Link to Star Schema**: <https://drive.google.com/file/d/1KBmCVnzG2P-VaZFReCb-P-dTvPesqtbH/view?usp=sharing>

##### 4.3.3.4 Create Data Dictionary

A data dictionary is a tool used in managing data that provides detailed information about the contents of a dataset or database. It includes things like the types of data, names of fields, and their descriptions, helping users better understand the structure and meaning of the data.

1. **Fact Table: Sales\_Fact**

| **Field Name** | **Data Type** | **Description** |
| --- | --- | --- |
| Sales\_ID (PK) | Integer | Primary key identifying each sale |
| Camera\_ID (FK1) | Integer | Foreign key that links to the Camera table, identifying the camera used |
| Screen\_ID (FK2) | Integer | Foreign key that links to the Screen table, identifying the screen used |
| Memory\_ID (FK3) | Integer | Foreign key that links to the Memory table, identifying the memory used |
| TechSpec\_ID (FK4) | Integer | Foreign key that links to the TechSpec table, identifying the technical specifications used |
| Sales\_date | Date | The date when the sale occurred |
| Sales\_price\_range | Integer | Categorises the price into a range indicating how expensive the sale was |
| **CameraStats** -  Camera.COUNT(Sales), Camera.MEAN(Sales.price\_range), Camera.STD(Sales.price\_range) | Float | Aggregated metrics related to sales involving specific camera types. It includes the total number of sales, the average price range of those sales, and the variability in those price ranges |
| **ScreenStats** -  Screen.COUNT(Sales), Screen.MEAN(Sales.price\_range), Screen.STD(Sales.price\_range) | Float | Aggregated metrics related to sales involving specific screen types. Includes total number of sales, the average price range of those sales, and the variability in those price ranges |
| **MemoryStats** -  Memory.COUNT(Sales), Memory.MEAN(Sales.price\_range), Memory.STD(Sales.price\_range) | Float | Aggregated metrics related to sales involving specific memory configurations. Comprises the total number of sales, the average price range of those sales, and the variability in those price ranges |
| **TechSpecStats** -  TechSpec.COUNT(Sales), TechSpec.MEAN(Sales.price\_range), TechSpec.STD(Sales.price\_range) | Float | Aggregated metrics related to sales involving various technical specifications. This includes the total number of sales, the average price range of those sales, and the variability in those price ranges |

1. **Dimension Table: Camera\_Dim**

| **Field Name** | **Data Type** | **Description** |
| --- | --- | --- |
| Camera\_ID (PK) | Integer | Primary key for each camera specification entry |
| Camera\_pc | Integer | Represents the megapixels of the primary camera in the mobile phone |
| Camera\_fc | Integer | Represents the megapixels of the front camera in the mobile phone |

1. **Dimension Table: Screen\_Dim**

| **Field Name** | **Data Type** | **Description** |
| --- | --- | --- |
| Screen\_ID (PK) | Integer | Primary key for each screen specification entry |
| Screen\_sc\_h | Integer | Height of the mobile phone screen in centimetres |
| Screen\_sc\_w | Integer | Width of the mobile phone screen in centimetres |
| Screen\_px\_width | Integer | The pixel resolution width of the mobile phone screen |
| Screen\_px\_height | Float | The pixel resolution height of the mobile phone screen |

1. **Dimension Table: Memory\_Dim**

| **Field Name** | **Data Type** | **Description** |
| --- | --- | --- |
| Memory\_ID (PK) | Integer | Primary key for each memory specification entry |
| Memory\_ram | Integer | Represents the Random Access Memory (RAM) in Megabytes |
| Memory\_int\_memory | Float | Represents the internal storage memory of the phone in Gigabytes |

1. **Dimension Table: TechSpec\_Dim**

| **Field Name** | **Data Type** | **Description** |
| --- | --- | --- |
| TechSpec\_ID | Integer | Primary key for each set of technical specifications |
| mobile\_wt | Integer | The weight of the mobile phone in grams |
| m\_dep | Float | The depth of the mobile phone in centimetres |
| battery\_power | Integer | The total energy the battery can store at one time, measured in milliamp-hours (mAh) |
| touch\_screen | Boolean | Indicates whether the phone has a touch screen (True) or not (False) |
| clock\_speed | Float | The speed at which the phone’s processor executes instructions, measured in gigahertz (GHz) |
| n\_cores | Integer | The number of cores in the phone’s processor |
| wifi | Boolean | Indicates whether the phone has WiFi capability (True) or not (False) |
| blue | Boolean | Indicates whether the phone has Bluetooth capability (True) or not (False) |
| dual\_sim | Boolean | Indicates whether the phone supports dual SIM cards (True) or not (False) |
| four\_g | Boolean | Indicates whether the phone supports 4G network connectivity (True) or not (False) |
| three\_g | Boolean | Indicates whether the phone supports 3G network connectivity (True) or not (False) |
| talk\_time | Integer | The maximum length of time the phone can be used to talk before the battery runs out, measured in hours |

##### 4.3.3.5 Gather Insightful Information

1. **Identifying Popular Specifications to Drive Inventory Strategy**:

* By looking at how many sales each phone feature has (like Camera.COUNT(Sales), Screen.COUNT(Sales), Memory.COUNT(Sales) and TechSpec.COUNT(Sales)), we can figure out which phone specs are most and least popular. This information lets the business know what customers want, helping them adjust their inventory and what they make to include features that are in high demand.

1. **Assessing Price Range for Market Positioning and Competitive Strategy**:

* By using the average price for sales linked to each phone feature (like Camera.MEAN(Sales.price\_range), Screen.MEAN(Sales.price\_range), Memory.MEAN(Sales.price\_range) and TechSpec.MEAN(Sales.price\_range)), the company can see where different features fit in the market, from low-cost to very high-cost. This information helps them place their products in the right market category and set prices that match what they want to achieve in the market, ensuring competitive advantage and customer satisfaction.

1. **Understanding Price Variability to Formulate Pricing and Promotion Strategies**:

* Looking at the range of prices (like Camera.STD(Sales.price\_range), Screen.STD(Sales.price\_range), Memory.STD(Sales.price\_range) and TechSpec.STD(Sales.price\_range)) helps the business see how much prices change based on different features. If the price doesn’t change much (low standard deviation), it means the prices are pretty consistent. If there is a big change (high standard deviation), prices vary a lot. This information is critical for the business to understand the pricing dynamics and can be used to formulate more effective pricing strategies and promotions targeting specific customer segments or market needs.

#### 4.3.4 Splitting Data for Training and Validation

After using FeatureTools to create new features, we need to make sure our data is ready for making models. Before splitting the data into training and test sets, it is good to check if all the different classes in our data are roughly equal in size. If they are not, we might have to use oversampling to even things out. Looking at the pie chart in Figure 102, we see that the classes are divided as follows: 26.62%, 23.39%, 20.03%, and 29.96%. This spread is pretty balanced, meaning that no one class takes up most of the data. Since each class is at least 20% of the data, we do not need to worry about them being imbalanced and probably do not need to do oversampling.

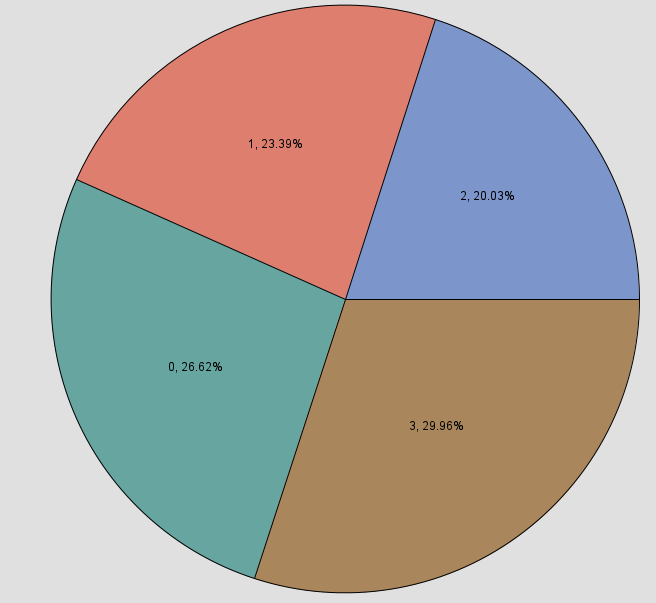


Figure 102: Pie Chart of Classes Distribution

Splitting the data is important for data analysis because it helps make sure our model can work well on new, unseen data, not just the data it learned from. We split our data into a training set and a validation set using SAS Enterprise Miner. The training set, which is 70% of the data, is what we use to train the model. The remaining 30% is the validation set, which we use to test how well our model works. We use a specific random seed (12345) so we can get the same split every time we run it. The settings we have used to divide the data are shown in Figure 103 in the Data Partition node’s properties.

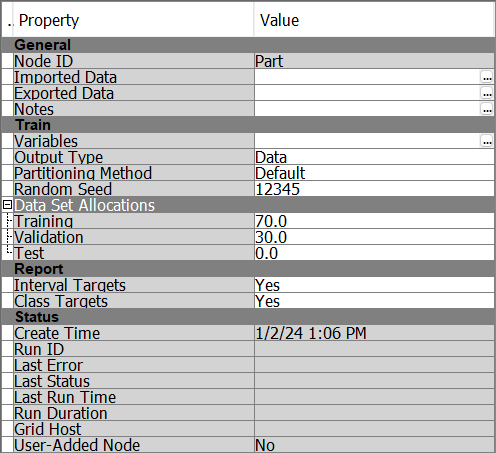


Figure 103: Properties of Data Partition Node

Figure 104 displays the outcome of splitting the data using the Data Partition Node. The image reveals that out of the original 3001 records, 2101 are allocated for the training set and the remaining 900 make up the validation set.

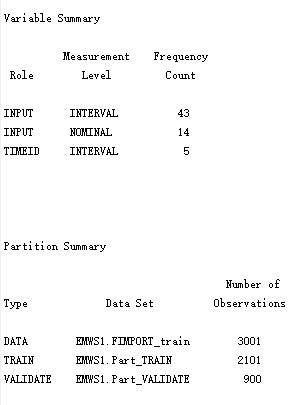


Figure 104: Partition Summary

#### 4.3.5 Feature Selection

Before we started building models, we noticed our dataset was quite big with 62 features, making it too complicated to create prediction models. So, we need to reduce the number of features to make things simpler. Choosing the right features is important because it helps the model focus on the most relevant information, which can lead to better predictions and faster performance. We use the Variable Selection node to pick out the most important features that help us classify the price range of mobile phones. This makes our model easier to understand and often improves its accuracy.

Figure 105 details the Variable Selection Node’s configuration in SAS Enterprise Miner. It manages up to 100 classes and accepts features with up to 50% missing data. The default setting automatically selects between Chi-Square and R-Square methods for feature selection, tailored to the data’s structure. Chi-Square is determined with a minimum value of 3.84, while R-Square considers up to 3000 variables with a minimum threshold of 0.005. Variables not chosen are hidden in the final output, streamlining the feature set for modelling purposes.

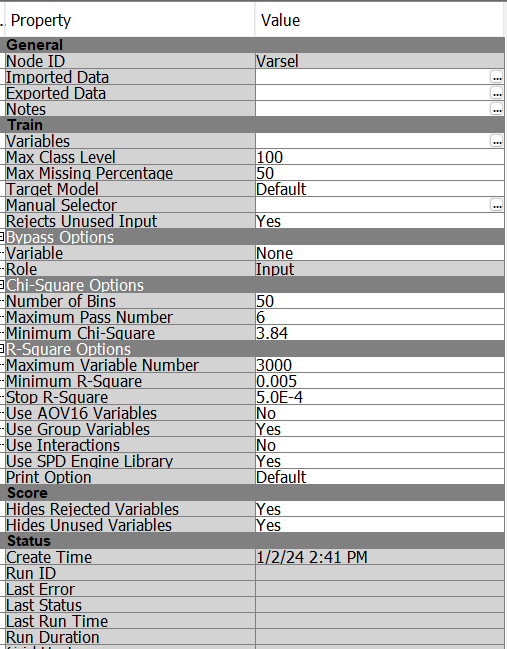


Figure 105: Properties of Variable Selection Node

After using the Variable Selection Node, 53 features were selected. Figure 106 displays all the features chosen by the Variable Selection Node.

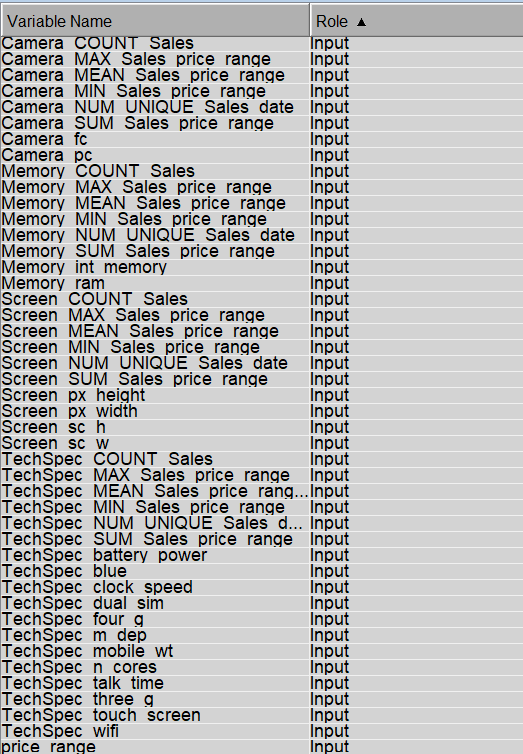


Figure 106: Features Selected by Variable Selection Node

Figure 107 below shows how we divided the data into training and validation sets and selected important features using SAS Enterprise Miner.

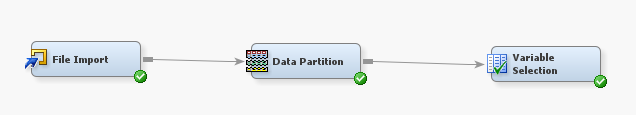


Figure 107: Data Partition and Variable Selection using SAS Enterprise Miner

### 

### 4.4 Model

SAS Enterprise Miner is utilised to build and train the models (refer to Figure 108). In this project, 3 main classification models are utilised: decision tree, random forest, AutoNeural and Gradient Boosting. On top of that, ensemble models, i.e., bagging and boosting, are also utilised and applied to the decision tree. Furthermore, 2 ensemble learning methods (i.e., ensemble average and ensemble voting) are also used in this project.

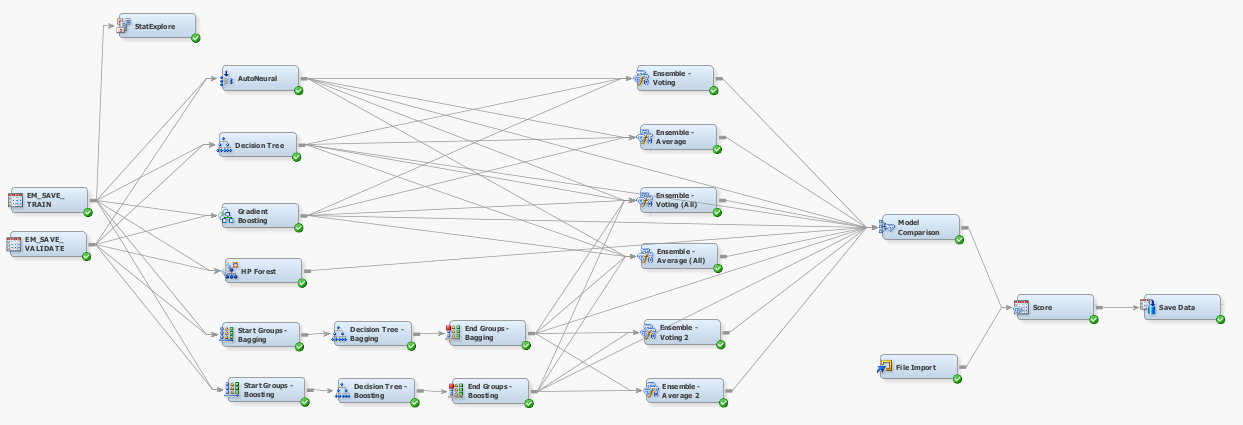


Figure 108: Architecture Diagram of the Modelling Nodes

The models will be trained with the training dataset and validated with the validation dataset that was obtained after splitting (in Section 4.3.4) and feature selection (in Section 4.3.5) by specifying their respective roles as ‘train’ and ‘validate’ (refer to Figure 109 & 110). Further explanation and discussion are in the following section below. The models' performance comparison will be discussed in Section 4.5.

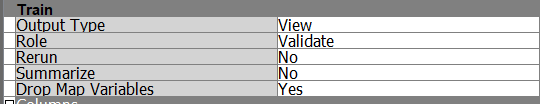
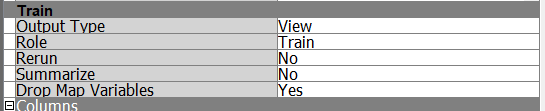


Figure 109 & 110: Role Specification of Train And Validate Datasets

#### 4.4.1 AutoNeural

AutoNeural is a neural network model in SAS Enterprise Miner which is an automated tool that can help find the optimal configurations for a neural network model (SAS Institute Inc., 2017).

The default parameters were also set. The key parameter to highlight is that the ‘Termination Methods’ chosen is ‘Overfitting’. This means that the model training will stop when overfitting is detected. Besides, the ‘Maximum Iteration’ is set to 8 with the ‘Number of Hidden Units’ is set to 2. This implies that for every 2 hidden units added, 8 iterations will be performed to find the most optimal weights. Figure 111 shows the full parameters of the node. Figure 112 shows that the best AutoNeural has a 0 misclassification rate.

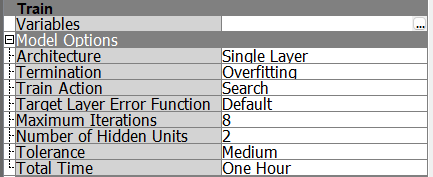


Figure 111: AutoNeural Node’s Parameters

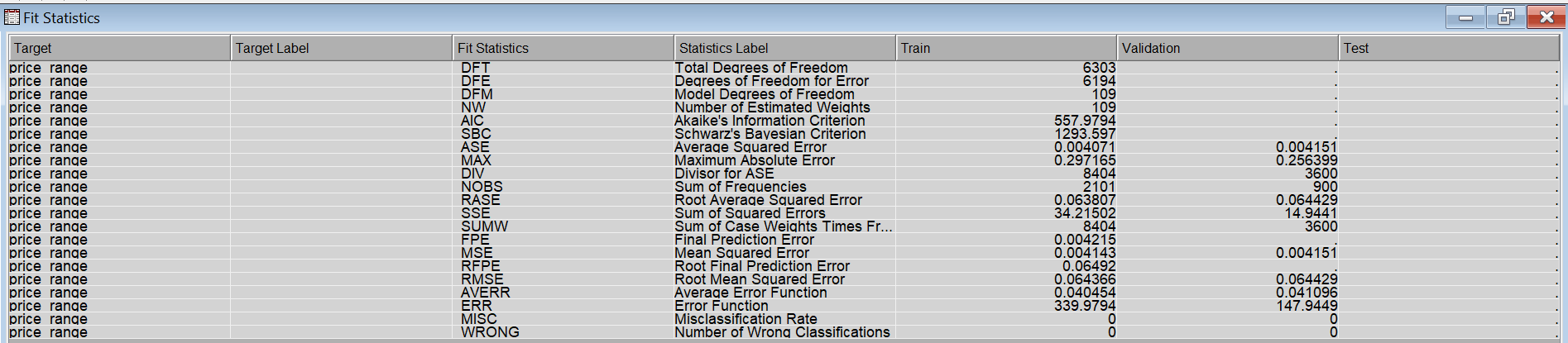


Figure 112: Autoneural Node’s Fit Statistics Output

#### 4.4.2 Decision Tree

A decision tree is a supervised machine learning algorithm for classification and regression. It is a non-parametric algorithm featuring a hierarchical structure composed of root nodes, branches, internal nodes and leaf nodes (IBM, n.d.).

The default parameters were used for the tree. One important note to highlight is the parameters in the SubTree section whereby the ‘Method’ is selected as the ‘Largest’ to select the largest tree, and the ‘Assessment Method’ is set to ‘Misclassification’ so that the model aims to minimise the misclassification rate. Additionally, the cross-validation parameter is set to ‘Yes’ to allow for a more reliable evaluation of the model’s performance. Figure 113 shows the full parameters of the tree and Figure 114 shows the tree. Figure 115 shows that the decision tree has a 0 misclassification rate.

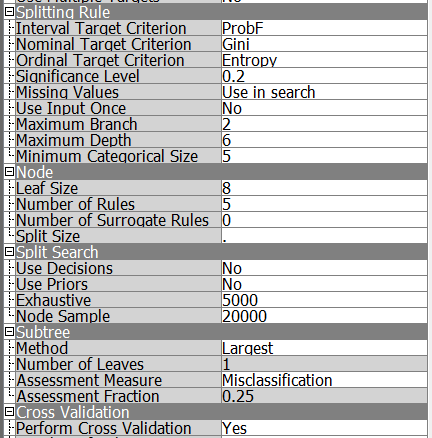


Figure 113: Decision Tree Node’s Parameters

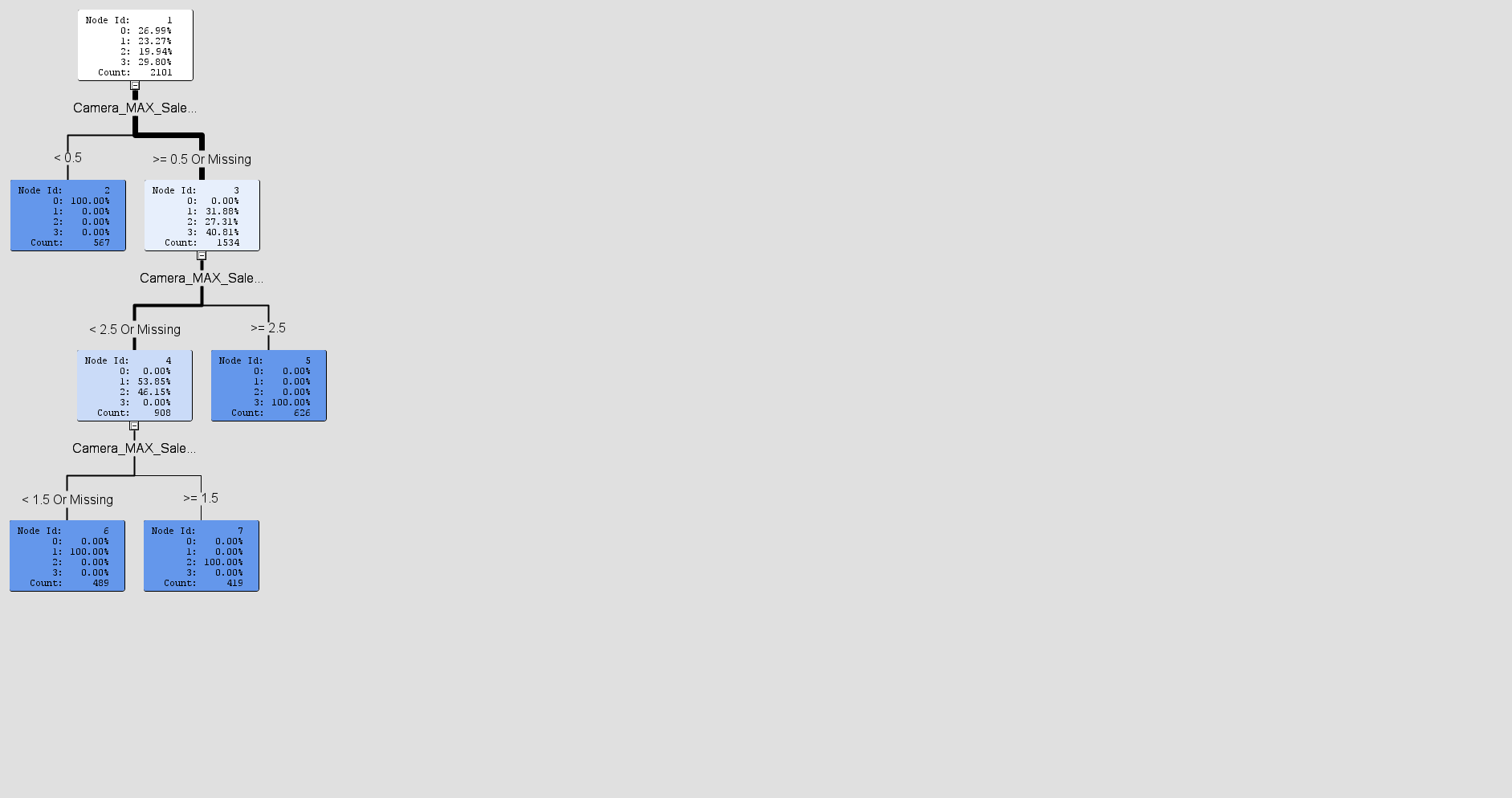


Figure 114: Architecture of Decision Tree

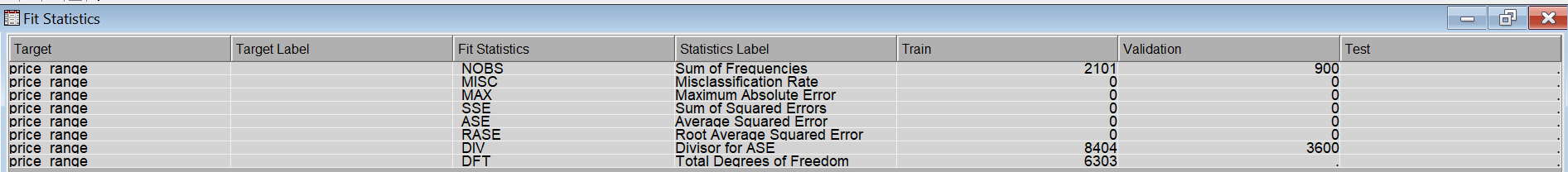


Figure 115: Decision Tree Node’s Fit Statistics Output

#### 4.4.3 Gradient Boosting

Gradient boosting builds a predictive model by creating a series of decision trees. It repeatedly samples the data to generate predictions and combines them to form a weighted average. Each tree in the series is trained to predict the difference between actual and predicted values from previous trees, focusing on areas where earlier predictions were less accurate.

The default parameters were also used. Some key parameters are the ’Number of iterations’ is set to 50 and the ‘train proportion’ is set to 60. ‘Maximum Branch’ and ‘Maximum depth’ are both set to 2. Figure 116 shows the full parameters of the node and Figure 117 shows that the gradient boosting has a 0 misclassification rate.

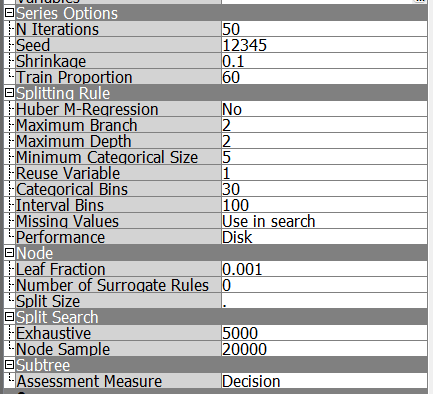


Figure 116: Gradient Boost Node’s Parameters

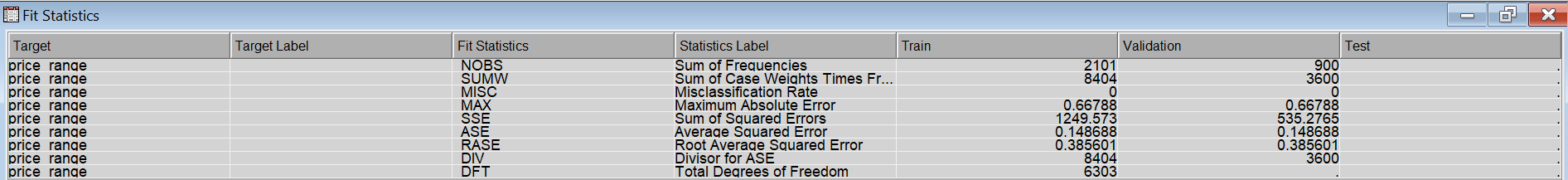


Figure 117: Gradient Boost Node’s Fit Statistics Output

#### 4.4.4 Random Forest

Random Forest is a machine learning algorithm which combines the output of multiple decision trees to reach a single result (Sruthi, 2023). In SAS Enterprise Miner, the HP Forest node was used to build the Random Forest model.

Similarly, the default parameters were used. Only the ‘Maximum Number Of Trees’ and the ‘Maximum Number Of Depth’ were reduced to 50 and 25, instead of the default value of 100 and 50 respectively to prevent overfitting. A large number of trees can improve generalisation. However, it is more prone to overfitting and, thus, unable to generalise well with new, unseen data. Additionally, a large number of trees and depth results in a longer training time. Figure 118 shows the full parameters of the node and Figure 119 shows that the HP Forest has a 0 misclassification rate.

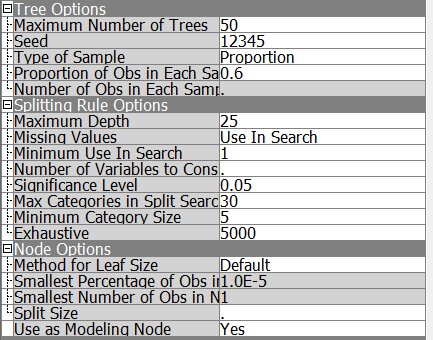


Figure 118: HP Forest Node’s Parameters

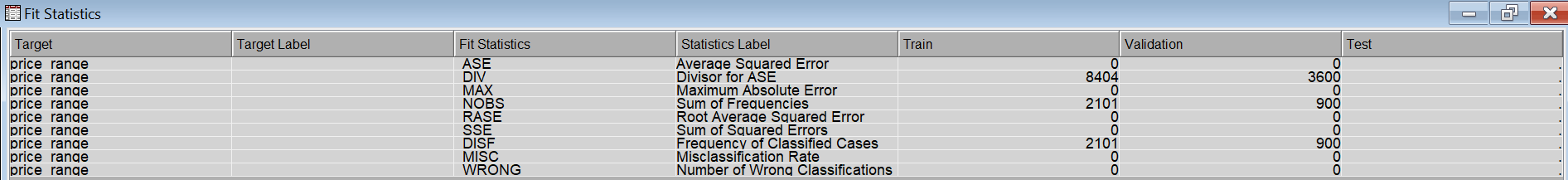


Figure 119: HP Forest Node’s Fit Statistics Output

#### 4.4.5 Ensemble Models: Bagging

Bagging stands for bootstrap aggregating. It consists of creating several samples to train models in parallel and combining the predicted probabilities (Maldonado, M. et al., 2014). In classification problems, bagging takes a majority vote of the predictions made from all of the models. Each model votes for a class and the class with a majority vote will be used as the final prediction. The majority vote approach can help reduce variance and bias in the predictions. Bagging is effective because it allows for a better performance than any single model.

Similarly, the default parameters were used. The ‘Index Count’ is used to specify the number of iterations to perform. In SAS Enterprise Miner, the bagging method is applied to the decision tree model using the ‘Start Group’ node and the ‘End Group’ node. Figure 120 shows the full parameters of the node.

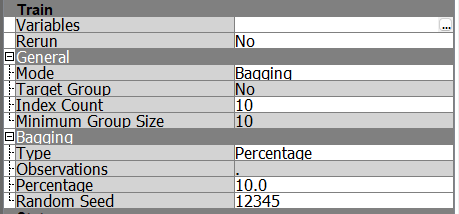


Figure 120: Bagging Node’s Parameters

#### 4.4.6 Ensemble Models: Boosting

According to Rocca (2021), boosting, akin to bagging, builds a group of models for stronger performance. However, while bagging aims to reduce variance, boosting involves training the machine learning models sequentially whereby each subsequent model tries to correct the errors of its predecessor. The combination of the models will give a weighted sum based on their accuracy.

Boosting involves readjusting the weights assigned for misclassified observations in each iteration, assigning a much higher weight to observations that are often misclassified than observations that are correctly classified. This weight readjustment is performed as long as it minimises the loss function.

Similarly to bagging, the default parameters were used. In SAS Enterprise Miner, the boosting method is also applied to the decision tree model using the ‘Start Group’ node and the ‘End Group’ node. Figure 121 shows the full parameters of the node.

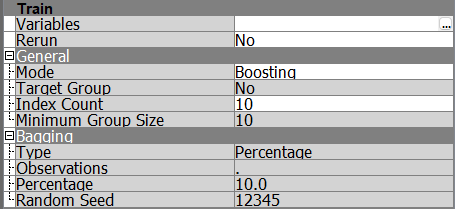


Figure 121: Boosting Node’s Parameters

#### 4.4.7 Ensemble Node Methods

The Ensemble node takes the probabilities for different categories from several models built individually and combines them to create a new model. This new model is then used to evaluate or make predictions on the new data.

##### 4.4.7.1 Average

This method is suitable for categorical and interval targets. This method takes the average of the posterior probabilities (predictions) of each model. Figure 122 shows the full parameters of the node.

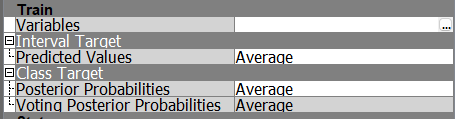


Figure 122: Ensemble Average Node’s Parameters

##### 4.4.7.2 Voting

This method is available for categorical targets only. There are 2 methods to compute the posterior probabilities which are the ‘Average’ and ‘Proportion’.

Figure 123 shows the full parameters of the node. The key parameter to highlight is the ‘Voting Posterior Probabilities’ This parameter is set to ‘Average’. This implies that the ensemble will calculate the average predictions of all individual models based on the majority of similar predictions, ignoring the predictions which are different.

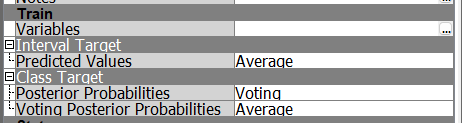


Figure 123: Ensemble Voting Node’s Parameters

### 

### 4.5 Assess

The ‘Model Comparison’ node is used to compare, assess, and identify the best-performing model. Figure 124 shows the model comparisons while Figure 125 shows the classification tables. Additionally, Figure 126 shows the classification charts of all models.

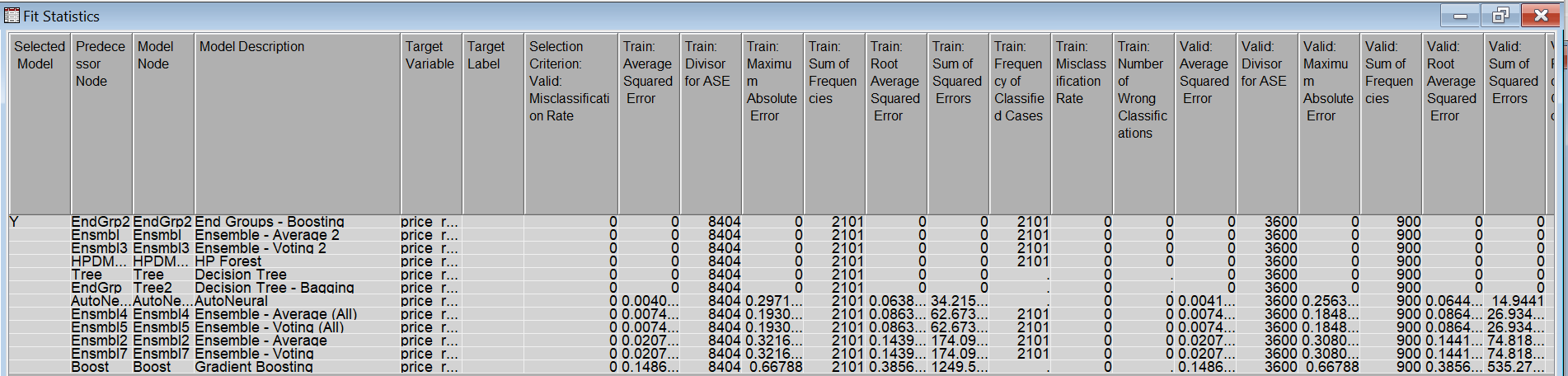


Figure 124: Modal Comparison Output of All Models - 1

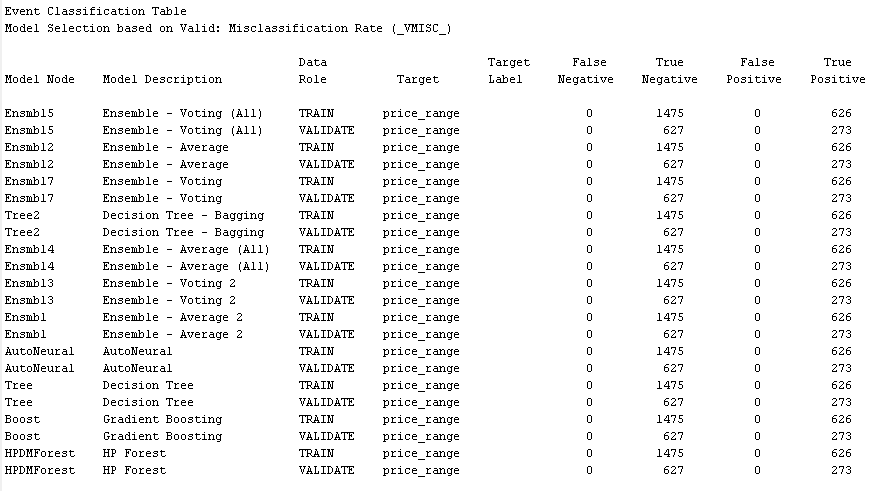


Figure 125: Modal Comparison Output of All Models - 2

#### 4.5.1 Misclassification Rate

The misclassification rate refers to the proportion of inaccurate predictions. The lower the misclassification rate, the better the performance of the model. Figure 126 shows the classification charts of each model.

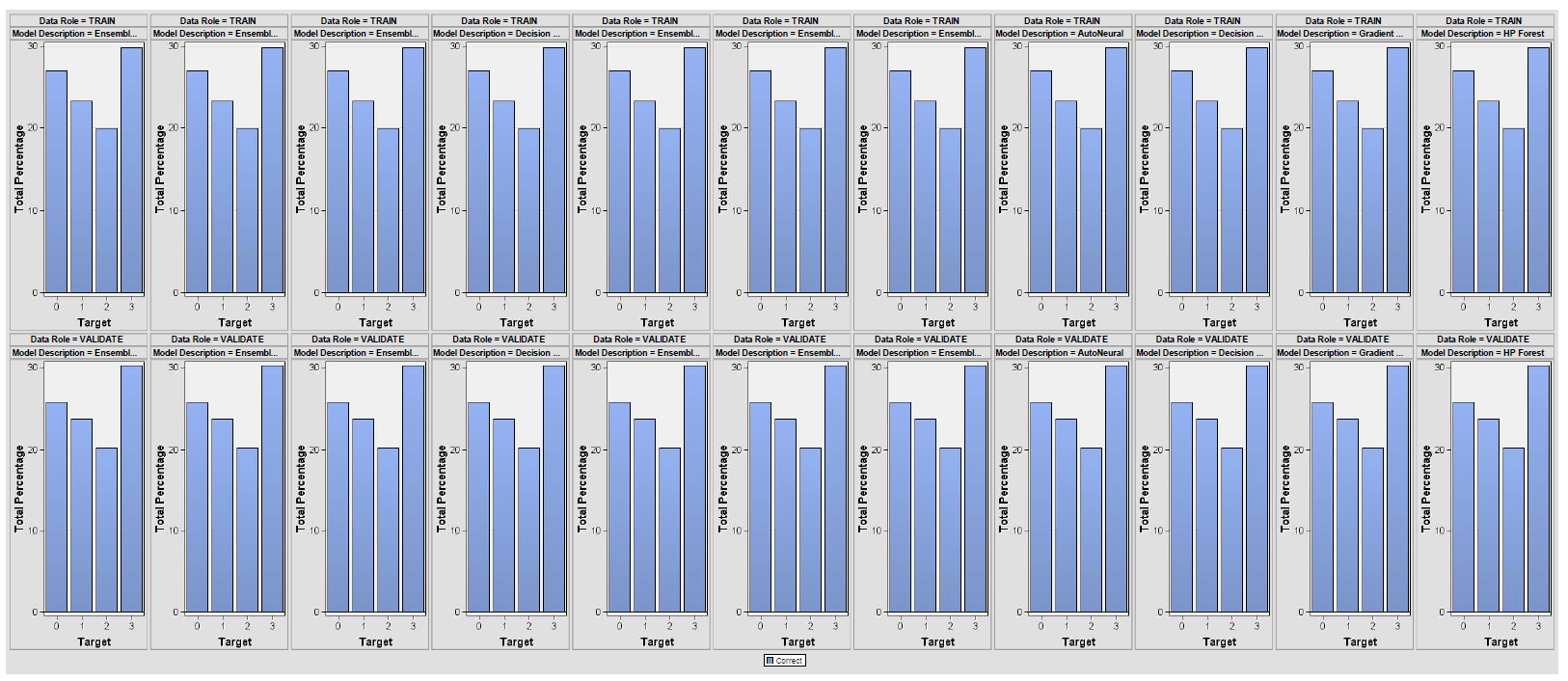


Figure 126: Classification Charts of All Models

To conclude, the charts from Figure 126 show that there is no misclassification in all of the trained models.

#### 4.5.2 Precision, Recall, F1-Score

Precision refers to out of all positive predictions of the model, how many are actually positive. Precision measures the reliability of the model. The equation is represented below.

Recall refers to out of all actual positives, how many the model predicted as positive. Recall measures the ability of the model to predict positive samples. The equation is represented below.

F1-score is the balance between precision and recall. It is useful when we want to consider both false positives and false negatives. A higher F1 score indicates a better model performance. The equation is represented below.

In short, based on Figure 125 above, the precision, recall and f1 score can be calculated using the values of ‘False Negative’, ‘True Negative’, ‘False Positive’ and ‘True positive’. It can be observed that all false positives and false negatives are 0s. Thus, precision, recall and f1 score are equal to 1.

#### 4.5.3 Score

The ‘Score’ node is utilised to evaluate how well the model is performing on new data. Similarly, a new data is imported using the ‘File Import’ node and then tested with the best model selected by the ’Modal Comparison’ node. The results are then evaluated (refer to Figure 127).

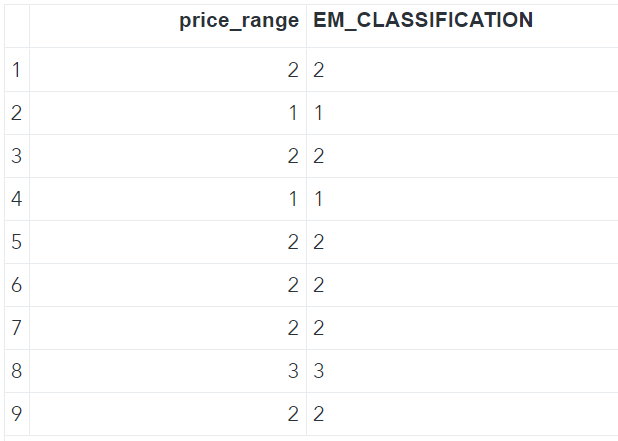


Figure 127: Actual Data vs Predicted Data

## 5.0 Conclusion

In conclusion, our data mining project provided deep insights into mobile phone sales. We used a detailed method, from collecting and preprocessing data to applying SAS SEMMA Methodology by using the advanced tools like Talend Integration, Talend Data Preparation, FeatureTools, SAS Enterprise Miner, and KNIME. This approach helped us to maintain high data quality as well as set the stage for effective model development and analysis.

To enhance our model’s performance, we created new features with FeatureTools and developed a star schema for organised data warehousing. This improved storage efficiency and simplified data queries. The schema also helped split our data into training and validation sets, which are essential for assessing the model’s predictive accuracy.

In the modelling phase, we utilised various algorithms and observed that RAM, battery power, and screen size were significant predictors of a mobile phone’s price range. Using DBSCAN clustering, we identified clear groupings based on these features. Additionally, we implemented Time Series Clustering, which further enhanced our understanding of how phone prices change over time, providing valuable insights into market trends and consumer behaviour.

For future work, we recommend exploring additional machine learning models and perhaps integrating more advanced ensemble methods to improve predictive accuracy. Neural networks could also be a great fit due to the diverse features in our data. Adding detailed data like customer reviews and specific sales numbers might give us a clearer picture of what customers want and their spending patterns.

Regularly updating the data warehouse and retraining models are also key steps. This keeps our findings relevant and useful, especially important in the fast-paced world of mobile phones where trends and technology are always changing. By staying up-to-date and refining our models, we can keep up with these shifts and continue to provide valuable insights into the mobile phone market.

## 6.0 References

Ali, M. (2023). *Association Rule Mining in Python Tutorial*.

<https://www.datacamp.com/tutorial/association-rule-mining-python>

Dutta, G. (2022). *Mobile price Classification*. Kaggle. <https://www.kaggle.com/datasets/gauravduttakiit/mobile-price-classification?select=sample.csv>

IBM. (n.d.). *What is a Decision Tree | IBM*. <https://www.ibm.com/topics/decision-trees#:~:text=data%20mining%20solutions-,Decision%20Trees,internal%20nodes%20and%20leaf%20nodes>.

Maldonado, M., Dean, J., Czika, W., & Haller, S. (2014). *Leveraging Ensemble Models in SAS® Enterprise MinerTM*. <https://support.sas.com/resources/papers/proceedings14/SAS133-2014.pdf>

Matignon, R. (2007). *Data Mining Using SAS Enterprise Miner*. John Wiley & Sons. <https://www.lexjansen.com/wuss/2007/AnalyticsStatistics/ANL_Matignon_DataMining.pdf>

Rocca, J. (2021). Ensemble methods: bagging, boosting and stacking - Towards Data Science. *Medium*. <https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205>

SAS Institute Inc. (2017). *SAS Help Center*. <https://documentation.sas.com/doc/en/emref/14.3/n1rk56ji9xtxxnn1q40urbumsno3.htm>

SAS Institute Inc. (2018). *SAS Help Center*. <https://documentation.sas.com/doc/en/emcs/14.3/n0pejm83csbja4n1xueveo2uoujy.htm#n0qt24wy1nhibyn104sv1lr4z51r>

Sruthi. (2023). *Understand random forest algorithms with examples (Updated 2024)*. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>