Table of Contents

1 Problem Definition

Telemarketing is used in promoting business services or products, where the salesman or call center of a company contacts its customers for a direct marketing campaign. Direct marketing database contains information about potential customers, which can be used for advertisement, analysis, and communication. Nowadays, data-driven predictive approaches using data mining techniques have been adopted by many banks to classify potential customers before direct marketing, improving both the time-efficiency and cost-efficiency of telemarketing.

This data mining project focuses on the data related to a Portuguese bank telemarketing, which contains previous results of marketing campaigns. The business goal is to identify the potential customers who are more likely to subscribe to a term deposit account. The data mining goal is to use classification models to predict who will be potential customers based on previous marketing campaigns data obtained. In this project, SAS software will be used for data investigation, RStudio will be used for some data exploration and extra data preparation, and SAS Enterprise Miner will be used to develop a number of classification models and evaluate the results.

2 Data Exploration and Data Insights

2.1 Overview of the Dataset

The data was collected from 2008 (May) to 2010 (November), with 41188 instances, 20 inputs and 1 output. The description of the input and output variables are described in Table 1.

Table 1 Description of Input and Output Variables

Variable Name	***Variable Type***	***Variable Description***
Bank Client Data		
age	numeric	
job	categorical	'admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown'
marital	categorical	'divorced', 'married', 'single', 'unknown'
education	categorical	'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree', 'unknown'
default	categorical	'no', 'yes', 'unknown' (has credit in default ?)
housing	categorical	'no', 'yes', 'unknown' (has housing loan?)
loan	categorical	'no', 'yes', 'unknown' (has personal loan?)
Last Contact of the Current Campaign		
contact	categorical	'cellular', 'telephone'
month	categorical	'jan', 'feb', 'mar', … , 'nov', 'dec'
day_of_week	categorical	'mon', 'tue', 'wed', 'thu', 'fri'
duration	numeric	(in seconds)
campaign	numeric	(number of contacts performed during this campaign for this client)
pdays	numeric	(number of days since last contact)
previous	numeric	(number of contacts performed before this campaign for this client)

poutcome	categorical	'failure', 'nonexistent', 'success' (outcome of the previous campaign)
Social and Economic Context Attributes		
emp.var.rate	numeric	(employment variation rate)
cons.price.idx	numeric	(consumer price index)
cons.conf.idx	numeric	(consumer confidence index)
euribor3m	numeric	(euribor 3 month rate)
nr.employed	numeric	(number of employees)
Output Variable		
y	categorical	'yes', 'no' (has the client subscribed a term deposit?)

The original dataset was read in R to do the general exploration of the data, the categorical variables are changed into factors so we can have a general view of the data distribution among different categories. The summary of the data is shown in Figure 1:

```
      age                job            marital                    education        default
 Min.   :17.00   admin.     :10422   divorced: 4612   university.degree  :12168   no     :32588
 1st Qu.:32.00   blue-collar: 9254   married :24928   high.school        : 9515   unknown: 8597
 Median :38.00   technician : 6743   single  :11568   basic.9y           : 6045   yes    :    3
 Mean   :40.02   services   : 3969   unknown :   80   professional.course: 5243
 3rd Qu.:47.00   management : 2924                    basic.4y           : 4176
 Max.   :98.00   retired    : 1720                    basic.6y           : 2292
                 (Other)    : 6156                    (Other)            : 1749
    housing          loan            contact           month       day_of_week     duration
 no     :18622   no     :33950   cellular :26144   may    :13769   fri:7827    Min.   :   0.0
 unknown:  990   unknown:  990   telephone:15044   jul    : 7174   mon:8514    1st Qu.: 102.0
 yes    :21576   yes    : 6248                     aug    : 6178   thu:8623    Median : 180.0
                                                   jun    : 5318   tue:8090    Mean   : 258.3
                                                   nov    : 4101   wed:8134    3rd Qu.: 319.0
                                                   apr    : 2632               Max.   :4918.0
                                                   (Other): 2016
    campaign          pdays          previous            poutcome      emp.var.rate      cons.price.idx
 Min.   : 1.000   Min.   :  0.0   Min.   :0.000   failure    : 4252   Min.   :-3.40000   Min.   :92.20
 1st Qu.: 1.000   1st Qu.:999.0   1st Qu.:0.000   nonexistent:35563   1st Qu.:-1.80000   1st Qu.:93.08
 Median : 2.000   Median :999.0   Median :0.000   success    : 1373   Median : 1.10000   Median :93.75
 Mean   : 2.568   Mean   :962.5   Mean   :0.173                       Mean   : 0.08189   Mean   :93.58
 3rd Qu.: 3.000   3rd Qu.:999.0   3rd Qu.:0.000                       3rd Qu.: 1.40000   3rd Qu.:93.99
 Max.   :56.000   Max.   :999.0   Max.   :7.000                       Max.   : 1.40000   Max.   :94.77

 cons.conf.idx     euribor3m      nr.employed     y
 Min.   :-50.8   Min.   :0.634   Min.   :4964   no :36548
 1st Qu.:-42.7   1st Qu.:1.344   1st Qu.:5099   yes: 4640
 Median :-41.8   Median :4.857   Median :5191
 Mean   :-40.5   Mean   :3.621   Mean   :5167
 3rd Qu.:-36.4   3rd Qu.:4.961   3rd Qu.:5228
 Max.   :-26.9   Max.   :5.045   Max.   :5228
```

Figure 1 Summary of the Variables in Bank Telemarketing Dataset

From the summary we could see the output 'y' is largely imbalanced, where 36548 customers (88.7%) didn't subscribe during the campaign, only 4640 customers (11.3%) subscribed during the campaign.

2.2 Correlation of the Input Interval Variables

A correlation matrix was created to check if there is any strong correlation among the numeric independent variables (inputs), shown in Figure 2. Three variables have very strong correlation with each other (>0.7), which are: 'emp.var.rate', 'euribor3m', and 'nr.employed'.

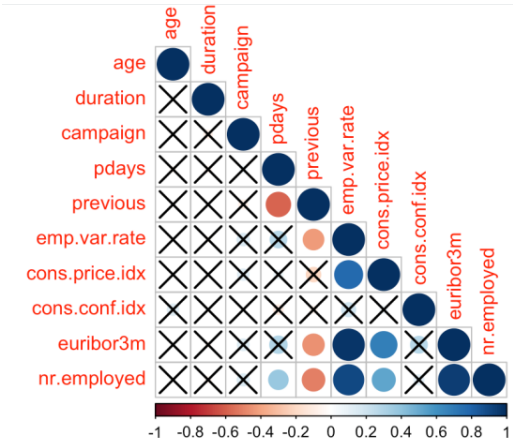

Figure 2 Correlation Matrix of Numeric Variables in the Dataset

2.3 Variable Worth of Inputs

The exploration in the SAS Enterprise Miner shows the ranking of chi-square values (Figure 3) and the variable worth (Figure 4) of the input variables. Together with the P-value of the Chi-Square test from the output window, it appears the all the numeric variables have relatively high Chi-Square statistics and high worth ranking. On the other hand, 'housing', and 'loan' related the least to the output target and ranked the least worthy variables too.

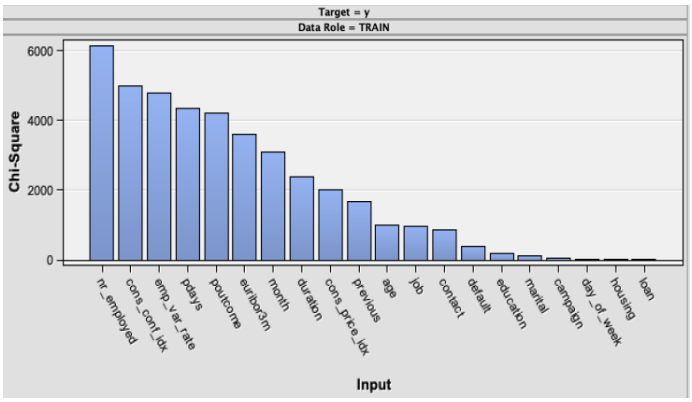

Figure 3 Chi-Square Ranking of the Input Variables

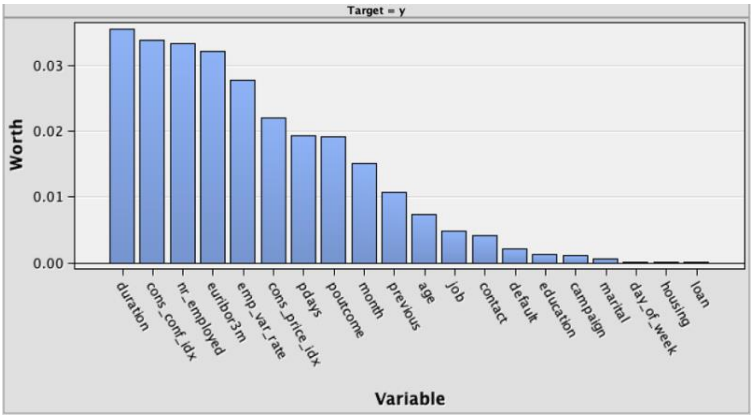

Figure 4 Variable Worth Ranking of the Input Variables

The interval variable summary statistics (Figure 5) in SAS also show that some of the variables have quite large standard deviation (>100), which may need to be transformed later before modelling.

Interval Variable Summary Statistics
(maximum 500 observations printed)

Data Role=TRAIN

Variable	Role	Mean	Standard Deviation	Non Missing	Missing
age	INPUT	40.02406	10.42125	41188	0
campaign	INPUT	2.567593	2.770014	41188	0
cons_conf_idx	INPUT	-40.5026	4.628198	41188	0
cons_price_idx	INPUT	93.57566	0.57884	41188	0
duration	INPUT	258.285	259.2792	41188	0
emp_var_rate	INPUT	0.081886	1.57096	41188	0
euribor3m	INPUT	3.621291	1.734447	41188	0
nr_employed	INPUT	5167.036	72.25153	41188	0
pdays	INPUT	962.4755	186.9109	41188	0
previous	INPUT	0.172963	0.494901	41188	0

Figure 5 Interval Variable Summary Statistics

3 Data Preparation and Feature Reduction

Feature reduction was done to reduce the possible misclassification rate during modelling and increase the modelling time-efficiency as well.

1. It is noted that the duration of the contact is not known before a contact is performed. The result is obviously known after the contact, which mean if the duration is “no”, the output will be “no”. Therefore, ‘duration’ should be excluded in our predictive model.
2. There are many ‘unknown” values in some variables shown in Table 2.

Table 2 Missing Values

Variable	***Count of Missing Value***
job	330
marital	80
education	1731
default	8597
housing	990
loan	990

These are all missing values, which is useless for our prediction, we should drop them. If we drop the rows, more than 1/4 of the data will be gone which will largely reduce the sample size for later model training. Instead, we use variable worth and Chi-Square statistics as reference, where we could see the ‘housing’, ‘loan’, ‘marital’, ‘education’, and ‘default’ don’t appear to influence the target value much, so we can drop those variables. For the ‘job’ input, there are only 330 instances unknown, which we can drop those rows.

3. By checking the correlation among interval values in previous section, we found three variables are highly correlated, ‘emp.var.rate’, ‘euribor3m’, and ‘nr.employed’. This is against the multicollinearity assumption of conducting logistic regression, so for logistic regression model, we will do this feature reduction before the regression. We can remove the two variables among them, which are ‘emp.var.rate’, and ‘euribor3m’, as the chi-square value and the variable worth of these two variables are lower than the variable ‘nr.employed’. (only for regression model)

In short, the 'housing', 'loan', 'marital', 'education', 'default', and 'duration' inputs are dropped, and the rows containing missing values in 'job' are removed. The dataset after first-stage preparation contains 40858 examples and 15 variables (including 14 inputs and 1 output).
The data partition node is used to separate the sample into training data and validation data. In our project, we first used 60/40 and then used 80/20, it turned out the results from the ratio 80/20 was better than 60/40, so the following content is based on the data partition to be 80/20 (80% data used for training, 20% data used for validation).

4 Data Mining Algorithms

4.1 Non-Parametric Models

4.1.1 Decision Tree

A decision tree is a type of flow chart which is used to visualize the decision-making process by mapping out different courses of action along with their potential outcomes. This outcome can then be used to drive business decisions.

Elements of Decision Tree Models:

Root Node - The main node at the top which describes the objective of the process
Branches - This is the steams coming out from the root node. Branches represents the different options that are used to make a particular decision.
Leaf Node - Leaf nodes are attached at the end of the branches and represents the outcome of the decision made in that branch.

In SAS, the Decision Tree models are created to classify observations based on the values of nominal or binary targets and to predict outcome of interval variables ("SAS Enterprise Miner," n.d.). In our model the objective of using the decision tree is to classify the observations as our target variable is binary.

Decision Tree Model Requirements:

At least one target or predicted variable. Data must be binary, interval or nominal.
At least one input or predictor variable.

Configurations:

We have used two types of interactive decision trees: Maximal Tree and Probability tree. The Configuration settings for both models (shown as Figure 6) are as below:

Nominal target splitting criterion was set to ProbChisq, specifies the method to use for splitting rules evaluation of nominal targets, which is the p-value of the F test associated with our node variance. *Interval target splitting criterion* was set to ProbChisq, which is the p-value of the Pearson Chi-square statistic for the target and the branch node. Since our target is categorical, so chi square test is appropriate. *Significance Level* was set to 0.2, means the maximum acceptable p value is 0.2. *Maximum Branch* was set to 2, means the maximum number of branches a splitting rule can produce is 2. *Assessment Measure* decides on which methods can be used to select the best tree based on the validation data. For the maximal tree we used misclassification method. The maximal tree model selects the tree that has the smallest misclassification rate. For our Probability Tree we have used Average Square error method to select the tree that has the smallest average square error.

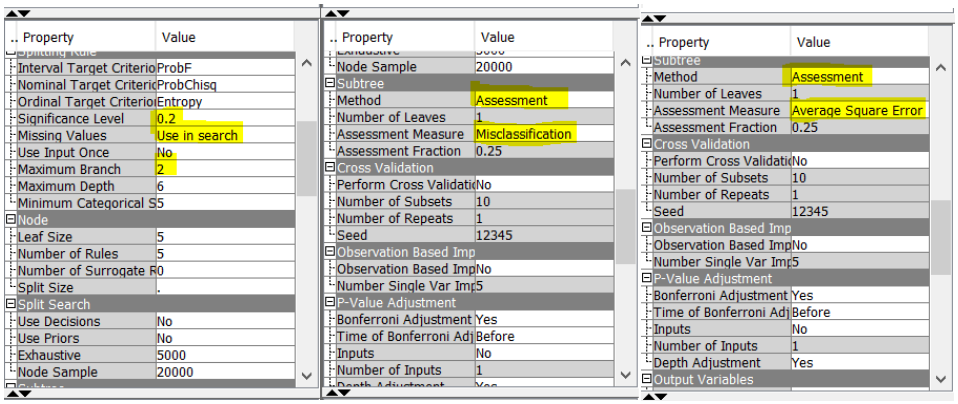

Figure 6 Configuration of Decision Tree Models

4.1.2 Gradient Boosting

Gradient boosting is a machine learning technique for classification problems, which produces a prediction model by assembling a bunch of weak prediction models, typically decision trees.

Model Methodology:

The model uses partitioning algorithm which searches the optimal partition in the data by using values of a single variable. The worth of the partitions is optimum when the distribution of target variables is similar within the segments. Gradient boosting resamples the analysis data set multiple times to generate a weighted average form of resampled data set. Then the data is used to create a series of decision trees and the accuracy of each trees are computed. These decision trees together form a predictive model. Since each successive sample is weighted according to the classification accuracy of the models, this approach is also known as stochastic gradient boosting.

Model Requirements:

At least one target or predicted variable. Data must be binary, interval or nominal.
At least one input or predictor variable.

Model Configuration:

We have used the below configuration for our gradient boosting model (shown as Figure 7): *N iterations* was set to 50, which means the number of trees will be 50. *Seed* was set as default (12345), which generates a random number to resample the analytic dataset. *Shrinkage* was set to reduce the prediction of each tree, which is 10% in our case. *Train proportion* is the proportion of training observations to train a decision tree. It changes with every iteration. For our first iteration we are taking the value as 60. *Maximum depth* defines the maximum number of splitting rules that can be applied. In our model maximum number of splitting rule can be applied is 10. *Assessment Measure* decides which methods to use to select the best tree, we used the decision method as in this method the tree that has the largest average 'yes' and smallest average 'no' will be selected.

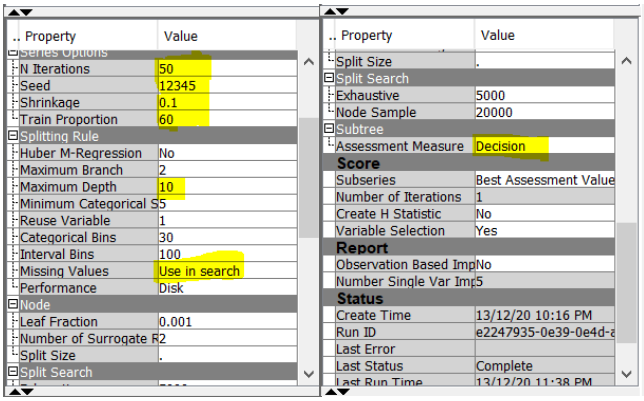

Figure 7 Configuration of Gradient Boosting Model

4.1.3 Memory-Based Reasoning (MBR)

Memory Based reasoning is a process that identifies similar cases and applies the information that is obtained from these cases to a new record. n Enterprise Miner, the Memory-Based Reasoning (MBR) node is a modelling tool that uses a k-nearest neighbour algorithm to categorize or predict observations.

Requirements of the Memory-Based Reasoning Model:

Requires exactly one target variable and can be binary, nominal, or interval. Also, model role of "input" is numeric, orthogonal to each other, and standardized.

Requirements of the Memory-Based Reasoning Model:

Method specifies which data representation can be used to store the training data. For our model we have selected the method as 'Scan'. As the number of observations in our training data is quite low, we wanted the model to scan through every observation in the data set and calculating its distance to a probe observation. *Number of Neighbors* is also known as K which is the number of nearest neighbours that can be used to categorize or predict observations, which was set to 3. *Number of Buckets* specifies the maximum number of buckets a leaf node can use to grow to before splitting into a branch with two new leaves, the value of this variable is our model is 8.

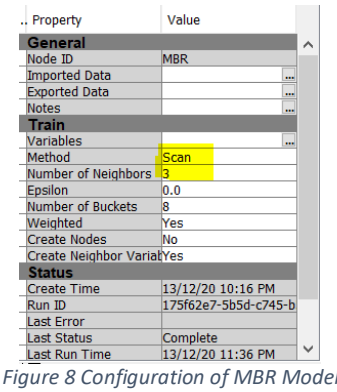

Figure 8 Configuration of MBR Model

4.2 Parametric Models

Pre-processing

The regression models and neural network models are sensitive to outlying or sensitive input values. Highly kurtotic or skewed data can be selected over the expected input data, which

leads to reduced prediction accuracy. Here we checked the distribution of the inputs before modelling (shown in Figure 9), finding that except the 'age' input others were with different level of skewness and kurtosis. Hence we used data transformation to induce transformed data for later modelling. The 'Log 10' function was used to transform the interval, and from the result (shown in Figure 10) we could see most of the variables were within the acceptable range (skewness and kurtosis within +/- 2) now, only two variables were remaining with high kurtosis and skewness (last two transformed variables in the Figure 10). The standard deviation of all the transformed interval variables are all within 1.

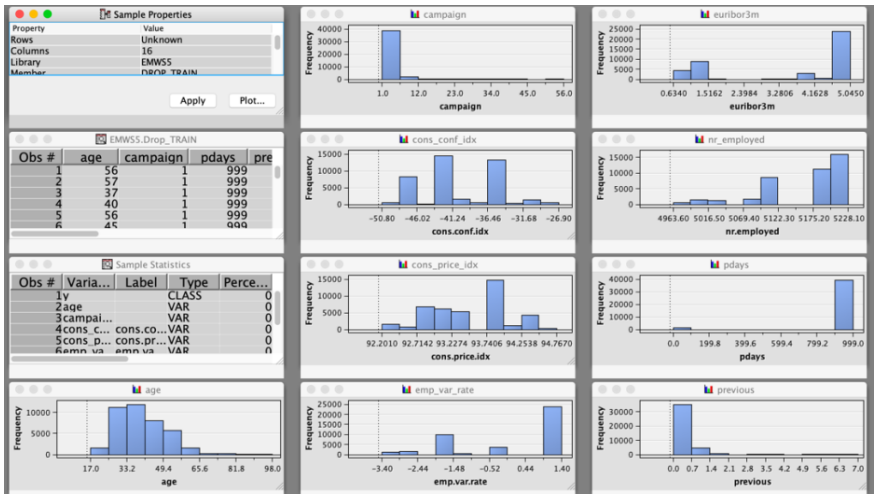

Figure 9 Distribution of Interval Variables before transformation

Source	Method	Variable Name	Formula	Number of Levels	Non Missing	Missing	Minimum	Maximum	Mean	Standard Deviation	Skewness	Kurtosis
Input	Original	campaign		.	20429	0	1	43	2.583386	2.815897	4.719861	35.44226
Input	Original	cons_con...		.	20429	0	-50.8	-26.9	-40.5023	4.657462	0.30558	-0.37115
Input	Original	cons_pric...		.	20429	0	92.201	94.767	93.56875	0.577674	-0.21956	-0.82606
Input	Original	emp_var_...		.	20429	0	-3.4	1.4	0.0648	1.578061	-0.70072	-1.09999
Input	Original	euribor3m		.	20429	0	0.634	5.045	3.598958	1.744322	-0.68118	-1.44773
Input	Original	nr_emplo...		.	20429	0	4963.6	5228.1	5166.131	72.87	-1.021	-0.06441
Input	Original	pdays		.	20429	0	0	999	961.4756	189.3554	-4.84833	21.5088
Input	Original	previous		.	20429	0	0	6	0.177493	0.501828	3.769521	19.26347
Output	Computed	LG10_ca...	log10(ca...	.	20429	0	0.30103	1.643453	0.486706	0.214901	1.364597	2.086212
Output	Computed	LG10_co...	log10(co...	.	20429	0	0	1.396199	1.009519	0.207992	-0.96028	1.785964
Output	Computed	LG10_co...	log10(co...	.	20429	0	1.969421	1.981216	1.97574	0.002655	-0.23002	-0.47284
Output	Computed	LG10_em...	log10(em...	.	20429	0	1.93E-16	0.763428	0.611444	0.200798	-1.22867	0.695619
Output	Computed	LG10_eur...	log10(eur...	.	20429	0	0.213252	0.781396	0.620012	0.20706	-0.76243	-1.26896
Output	Computed	LG10_nr_...	log10(nr_...	.	20429	0	3.695884	3.718427	3.713206	0.00617	-1.04006	-0.31939
Output	Computed	LG10_pd...	log10(pd...	.	20429	0	0	3	2.916447	0.423865	-4.93129	22.61323
Output	Computed	LG10_pre...	log10(pre...	.	20429	0	0	0.845098	0.04803	0.124413	2.548863	5.915645

Figure 10 Variable Transformation Results

4.2.1 Logistic Regression

Regression is a statistical method used to determine the relationship between a dependent variable and one or more independent variables.

Types of Regression:

1.Linear Regression: Linear regression is used to predict the value of interval target variables as a linear function of one or more independent predictor variables.

2.Logistic Regression: Logistic regression is used to predict the probability that a binary or nominal target variable will acquire the event as a function of one or more independent predictor variables.

In SAS both logistic and linear regression models are used.

Regression Model Requirements:

The model requires a minimum of one target variable. Data has to be an interval, ordinal, nominal, or binary target (response) variable.

The predictor or independent variable can be both continuous (interval) or discrete (binary, nominal, or ordinal).

Regression Model Configurations:

For our regression model, we have used the below configuration (shown as Figure 11). Since our target is a binary variable we changed the *Regression Type* to Logistic Regression and the *link function* as Logit. For the *Model Selection Property*, we used Stepwise model selection as we wanted to remove any effect already in the model, training continues until the Stay Significance Level is met. The *Selection Criterion* was set to Validation Misclassification, which leads to the model with the smallest misclassification rate is chosen in the end. *The highest degree of polynomial terms* to be used in a regression analysis was set to be 2. The *Entry Significance Level* was set as default (0.05), which specify the significance level when adding variables during stepwise regression. The *Stay Significance Level* was set as default (0.05), which specify the significance level when removing variables during stepwise regression.

Property	Value
Sequential Order	No
Entry Significance Level	0.05
Stay Significance Level	0.05
Start Variable Number	0
Stop Variable Number	0
Force Candidate Effects	0
Hierarchy Effects	Class
Moving Effect Rule	None
Maximum Number of Steps	0
Equation	
Main Effects	Yes
Two-Factor Interactions	No
Polynomial Terms	No
Polynomial Degree	2
User Terms	No

Class Targets	
Regression Type	Logistic Regression
Link Function	Logit
Model Options	
Suppress Intercept	No
Input Coding	Deviation
Model Selection	
Selection Model	Stepwise
Selection Criterion	Validation Misclassification
Use Selection Defaults	Yes
Selection Options	...

Figure 11 Configuration of Regression Model

4.2.2 Neural Network

Pre-processing

Since the Neural Network node in SAS Enterprise Miners doesn't have built-in approach to select useful inputs. We used 'Variable Selection' node to achieve similar function. We used default settings from Variable Selection node to perform the task, which will reject the variables with low R-square values. The selection results are shown in . From rejecting the input variables having low R-square scores, the input variables were reduced to six for analysis (shown in Figure 12).

Variable Name	Role ▲	Measurement Level	Type	Label
G_job	Input	Nominal	Numeric	Grouped...
G_month	Input	Nominal	Numeric	Grouped...
LG10_cons_price_idx	Input	Interval	Numeric	Transfor...
LG10_nr_employed	Input	Interval	Numeric	Transfor...
LG10_pdays	Input	Interval	Numeric	Transfor...
poutcome	Input	Nominal	Character	
LG10_campaign	Rejected	Interval	Numeric	Transfor...
LG10_cons_conf_idx	Rejected	Interval	Numeric	Transfor...
LG10_emp_var_rate	Rejected	Interval	Numeric	Transfor...
LG10_euribor3m	Rejected	Interval	Numeric	Transfor...
LG10_previous	Rejected	Interval	Numeric	Transfor...
age	Rejected	Interval	Numeric	
contact	Rejected	Nominal	Character	
day_of_week	Rejected	Nominal	Character	
job	Rejected	Nominal	Character	
month	Rejected	Nominal	Character	

Figure 12 Variable Selection Results

A neural network is a series of algorithms that are used to recognize the underlying relationships in a set of data through a process that mimics the way a system of neuron works.

In SAS the Neural Network model creates a multilayer neural network that passes the information from one layer to another to map an input to a predicted value.

Elements of Neural Network Models:

- Input Units — Have the values of input variables and standardize those values.
- Hidden Units — Perform internal computations and provide the nonlinearity that makes neural networks powerful.
- Output Units — Compute predicted values and compare those values with the values of the target variables.

Model Requirements:

Minimum one segment and one binary, nominal, or interval target variable is required.

Neural Network Model Configurations:

In our Neural Network Model, we have used the below configuration (shown as Figure 13). We changed the *Model Selection Criterion* to Misclassification which means we are focusing on the correct classification rate for model selection. *The number of hidden units* is set to be 5, which means our multilayer perception neural network will train with 5 units in the hidden layer. The *maximum iterations* are set to be 50, which allows maximum 50 iterations during network training. The *direction connection* enables a direct connection between input and output units (in addition to the connections made via hidden units).

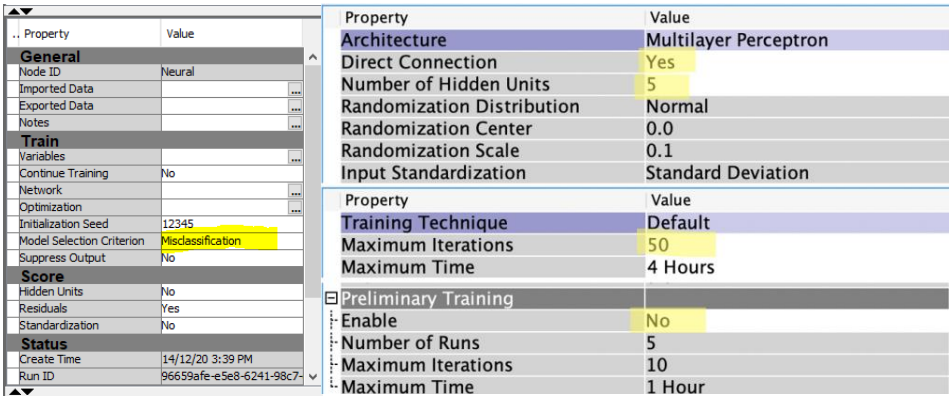

Figure 13 Configuration of Neural Network Model

4.2.3 Auto Neural

Auto neural node acts as an automated tool to find optimal configurations for a neural network model. To find the best-fitted network configuration auto neural networks conduct limited searches.

Auto Neural Network Architectures:

Auto neural nodes create different types of feed-forward network architectures. Each activation function and target function contain an optimized bias weight.

1. Multilayer Perceptron networks: these algorithms have been tuned to avoid overfitting but are not interpretable because of the highly nonlinear nature of combinations of activation functions.
2. Single hidden layer network: In this model new hidden neuron units are added one at a time and are selected according to the most beneficiary activation functions.
3. Funnel network: Same as single hidden layer network but the neuron units form a funnel pattern.
4. Block Network: In this model new hidden neuron units are added as new layers in the network.

5. Cascade network: In this network newly added hidden neuron units are connected from all existing input and hidden neurons.

Data Set Requirements:

The training data must contain a minimum of one target variable and one input variable.

Auto Neural Node Properties:

For our Auto Neural node, we have done the below configuration (shown as Figure 14). We have used most of the default configuration settings, only changed the number of hidden unit to 1. The *train action* was set to search, to sequentially increase the network complexity. *Number of Hidden Units* was set to 1, so each iteration adds one hidden unit. The *Tolerance* was set to low so the preliminary training can be avoided. The *direct* was set to no which cut the connection between the inputs and the target. The *Sine* and *Normal* were set to 0, so the normal distribution activation function and sine activation function were deactivated. The training will stop when *overfitting* happens.

Activation Functions	
Direct	No
Exponential	No
Identity	No
Logistic	No
Normal	No
Reciprocal	No
Sine	No
Softmax	No
Square	No

Increment and Searc	
Adjust Iterations	Yes
Freeze Connections	No
Total Number of Hid	30
Final Training	Yes
Final Iterations	5
Model Options	
Architecture	Single Layer
Termination	Overfitting
Train Action	Search
Target Layer Error Fu	Default
Maximum Iterations	8
Number of Hidden U	1
Tolerance	Low
Total Time	One Hour

Figure 14 Configuration of Auto Neural Model

5 Evaluation of the Models

5.1 Evaluation Parameters

Precision:

Precision can be used as a measure of a classifier's exactness. A low precision can indicate large number of False Positives.

$$Precision = \frac{TP}{TP + FP}$$

Accuracy:

Accuracy can be used as a measure of a classifier's correctness. It determines how many TP, TN, FP, FN were correctly classified.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Recall:

Recall can be used as a measure of a classifier's completeness. A low recall indicates many False Negatives.

$$Recall = \frac{TP}{TP + FN}$$

F1 score:

F1 score conveys the balance between the precision and the recall. F1 score helps us to identify the best possible combination of precision and recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

ROC:

ROC curve shows the relationship of the sensitivity (or true positive rate) against specificity (or false positive rate). The more closer the curve to the top-left corner, the better performance the classifier is given.
The ROC Index indicates the AUC value, which is the area under curve. The higher value indicates better performance of a classifier.

$$TPR = \frac{TP}{TP + FN}$$
$$FPR = \frac{FP}{FP + TN}$$

5.2 Evaluation of Models

Model comparison was conducted in SAS Enterprise Miner, misclassification rate, ROC Index, and Average Square Error can be directly obtained from the summary statistics in the output window. Precision, Recall, and F1 score are derived from the TP, TN, FP, and FN values of each model in the output window.
The classification table (see in Figure 15) was used to check the detailed classification rate for the target value (here we choose validation data to analyse the result). Although the misclassification rate in general for each model is not high, the misclassification of the target value "yes" is quite large for all the models which can be derived from the classification table. Misclassification rate for target value "yes" is even higher than 50% for MBR model, and one of the reasons may be insufficient sample data targeted to 'yes'. If we want to lower the misclassification rate for "yes", we need more sample data which contains the target value to be "yes".

Event Classification Table
Model Selection based on Train: Roc Index (_AUR_)

Model Node	Model Description	Data Role	Target	Target Label	False Negative	True Negative	False Positive	True Positive
MBR	MBR	TRAIN	y		2028	28398	606	1653
MBR	MBR	VALIDATE	y		661	6964	287	261
Tree2	probabilty tree	TRAIN	y		2616	28418	586	1065
Tree2	probabilty tree	VALIDATE	y		667	7115	136	255
Boost	Gradient Boosting	TRAIN	y		2446	28637	367	1235
Boost	Gradient Boosting	VALIDATE	y		675	7115	136	247
Tree	Maximal tree	TRAIN	y		2740	28527	477	941
Tree	Maximal tree	VALIDATE	y		692	7146	105	230
AutoNeural	AutoNeural	TRAIN	y		2920	28648	356	761
AutoNeural	AutoNeural	VALIDATE	y		729	7169	82	193
Neural	Neural Network	TRAIN	y		2909	28650	354	772
Neural	Neural Network	VALIDATE	y		733	7171	80	189
Reg	Regression	TRAIN	y		2816	28576	428	865
Reg	Regression	VALIDATE	y		705	7147	104	217

Figure 15 TP, TN, FP, FN of the Models (Using Validation Data)

For training data, the results are shown in Table 3. The MBR performed the best in general during for the training stage, with the lowest misclassification rate, highest accuracy, highest recall, and highest AUC.

Table 3 Results of Models on Training Data

	MBR	Gradient Boost	Probability Tree	Neural Network	Regression	Auto Neural	Maximal tree
Misclassification Rate	0.08	0.09	0.10	0.10	0.10	0.10	0.10
Roc Index (Area Under The curve)	0.89	0.87	0.80	0.79	0.79	0.79	0.76

Precision	0.731	0.770	0.645	0.685	0.668	0.681	0.663
Accuracy	0.919	0.913	0.903	0.900	0.900	0.899	0.901
Recall	0.449	0.335	0.289	0.219	0.235	0.206	0.255
F1 Score	0.556	0.467	0.399	0.331	0.349	0.321	0.357

For validation data, the results are shown in Table 4. The best performed model according to each evaluation parameter is hard to decide, as nearly each model has one or more best performed aspects. For example, Auto Neural Model has the lowest misclassification rate, but lowest recall. MBR has the highest recall, but AUC and precision are the lowest. Probability Tree model can be considered as the best model as it has the lowest classification rate, highest F1 score, the second highest AUC, the second highest accuracy, and the second highest recall.

Table 4 Results of Models on Validation Data

	MBR	Gradient Boost	Probability Tree	Neural Network	Regression	Auto Neural	Maximal tree
Misclassification Rate	0.12	0.10	0.10	0.10	0.10	0.10	0.10
Roc Index (Area Under The curve)	0.69	0.80	0.79	0.79	0.79	0.79	0.75
Precision	0.476	0.644	0.652	0.702	0.676	0.701	0.686
Accuracy	0.884	0.900	0.901	0.900	0.902	0.900	0.901
Recall	0.283	0.267	0.276	0.216	0.235	0.209	0.249
F1 Score	0.354	0.377	0.387	0.330	0.348	0.321	0.366

ROC Curves of training data and validation data are shown in Figure 16. In the training stage, the MBR model had the best performance, while in the validation stage, it dropped down to the lowest. Gradient Boosting model dropped around 7% in the validation stage as well. Performance of other models were rather stable at both stages.

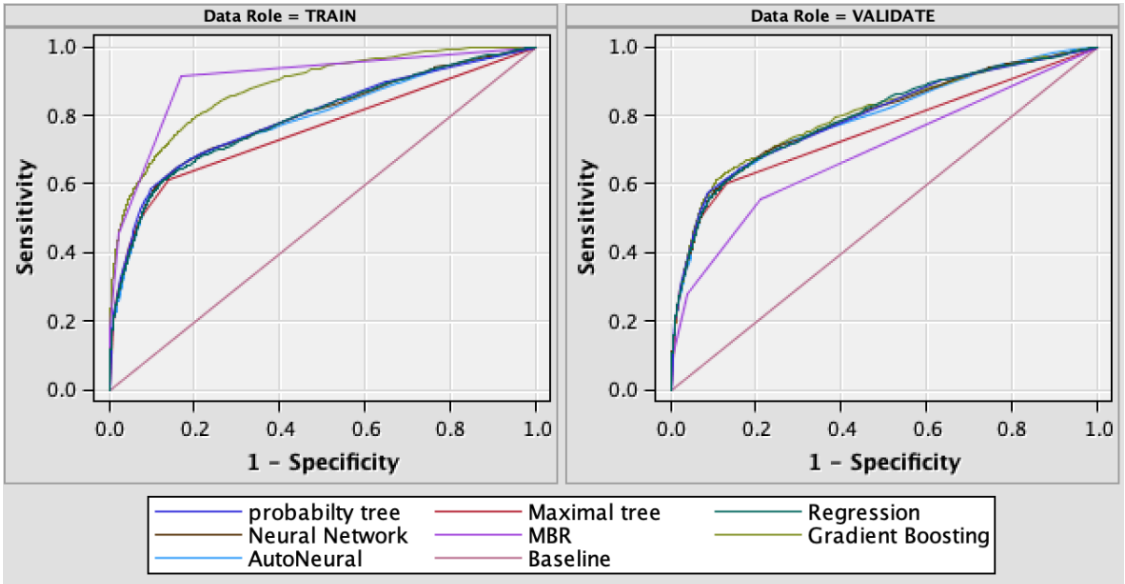

Figure 16 ROC Chart of the Models (Training Data and Validation Data)

To choose the best model, we are looking for a model with lowest misclassification rate, highest ROC index, highest precision, highest recall. However, on the one hand, the parameter "recall" is not quite suitable to be a reference in our case, since the sample data is largely imbalanced (on target value 'y'), and a meaningful recall value is generally based on balanced data. On the other hand, in the evaluation results, the best result for each parameter fell onto different models. Hence we need to find one model that has "generally" best results. It means for some parameter having the best results, the model has relatively high performance for other parameters as well. Based on this, we choose ***Probability Tree model*** to be our best model, which has lowest classification rate (0.1), highest F1 score (0.387), the second highest AUC (0.79), the second highest accuracy (0.901), and the second highest recall (0.276).

6 Discussion

For our best model, ***Probability Tree model*** has an accuracy of 90.1%, a misclassification rate 0.1, precision of 65.2%, recall of 27.6%, and an AUC of 0.79. It means we may to contact with 65.2% of the potential customers, and 27.6% of them are likely to result in a success. The recall value is very low is because of the largely imbalanced sample data.

6.1 Comparing with Work of Muneeb Asif

For the models established by Muneeb Asif (Asif, 2018), which included Support Vector Machine (SVM), Decision Trees, Random Forest, and Artificial Neural Network. The results are shown in Table 5. It worked on the old dataset (bank-full.csv), which has 45211 instances and 17 attributes.

Table 5 Evaluation Results of Models done by Muneeb Asif (full model)

Algorithms	***Accuracy***	***AUC***
Support Vector Machine	89.6%	0.773
Decision Trees	89.6%	0.785
Random Forest	90.6%	0.937
Artificial Neural Network	89.0%	0.893

The accuracy of our best model (Probability Tree Model) is slightly lower than the best model in his work (by 0.5%), which is higher than all the rest of his models. The AUC is lower than his best model and the second of his model, which is artificial neural network. The accuracy of our neural network is higher than his neural network model, this may be because we did feature selection before the neural network modelling, which may help reduce the noise. The performance of Neural Network model is much higher than our model, this may be because of the 'duration' variable which he didn't exclude. If 'duration' is not excluded, the AUC will be higher than it should be, because when duration is 0 the output is definitely 'no', which can largely reduce the false negative rate and increase the AUC.

The best model in his work is Random Forest, which is an ensemble method including a large number of trees, using 'majority rules' or averages to join at the end of the process. Our similar model with Random Forest is the Gradient Boosting model, which also contains a set of decision trees. The accuracy of our Gradient Boosting Model after training stage is higher than the random forest model in his work. But in the validation stage, our Gradient Boosting Model dropped largely on almost all the parameters, being lower than any other models. This

may be because the training data is lack of representative instances and has a lot of noise, which caused overfitting in training stage.

6.2 Comparing with Work of Moro, Cortez, & Rita

For the models established by Sergio Moro etc (Moro, Cortez, & Rita, 2014), which included Support Vector Machine (SVM), Decision Trees, neural network, and logistic regression. The results are shown in Table 6. They used a larger dataset which contains 52994 cases, with 6557 (12.38%) records related with successes.

Table 6 Evaluation Results of Models done by Moro, Cortez, & Rita (modelling phase)

Algorithms	***AUC***	***ALIFT***
Support Vector Machine	89.1%	0.844
Decision Trees	83.3%	0.756
Logistic Regression	90%	0.849
Neural Network	92.9%	0.878

In their work, the best model is the neural network algorithm, with the highest AUC (92.9%), Our best model with Probability Tree algorithm has an AUC of 0.79, which is higher than their Decision Tree algorithm but lower than their other three algorithms. Even their model with lowest AUC (Decision Trees) is still better than all of our models regarding general performance.

In their work, the variable 'duration' was not excluded. The 'duration' variable may contribute to the high accuracy and precision in all the models. As we mentioned before, If 'duration' is not excluded, it can largely reduce the false negative rate and increase the precision and accuracy.

They used 51651 examples to be training data, and 1293 cases with the newest data to be testing data. The ratio is almost 40/1, much bigger than our data partition, which was 8/2. From this aspect, maybe we should put more data to training stage and less data for validation stage.

In their work they conducted data enrichment by gathering external data online, leading to a total of 150 attributes. Then they used factor of analysis and forward selection method to do the feature selection for dropping irrelevant variables. From this aspect, maybe we should do extra data enrichment to gather more variables, as the variables in our dataset may not be the most contributed factor to the target value.

6.3 Comparing with Work of Palaniappan, Mustapha, Foozy, & Atan

For the models established by Palaniappan etc (Palaniappan, Mustapha, Foozy, & Atan, 2017), which included Naïve Bayes, Decision Trees, and Random Forest. The results are shown in Table 7. They used the same dataset as we did in our project (but they used it in time ordered sequence).

Table 7 Evaluation Results of Models done by Palaniappan, Mustapha, Foozy, & Atan

Algorithms	***Accuracy***	***Precision***	***Recall***
Naïve Bayes	86.3%	0.687	0.755
Random Forest	88.8%	0.823	0.505

Decision Tree	90.7%	0.778	0.697

In their work, the best model is the decision tree algorithm, with the highest accuracy (90.7%), and average precision (0.778) and recall (0.697). Our best model with Probability Tree algorithm has accuracy of 90.1%, which is lower than their Decision Tree algorithm but higher than their other two algorithms. The precision (0.652) is lower than all of their algorithms. The recall of all of our algorithms are much lower than their models.
In their work, the variable 'duration' was not excluded. The 'duration' variable may contribute to the high accuracy and precision in all the models. As we mentioned before, If 'duration' is not excluded, it can largely reduce the false negative rate and increase the precision and accuracy.
For the data preparation and cleaning they used Z-transformation method to normalize the data. But didn't clean the missing values or do feature selection. The reason why the Naïve Bayes algorithm has the lowest accuracy and precision may be because Naïve Bayes based on the assumption that features are independent from each other, which is not the case in this dataset. As earlier we conducted the correlation matrix and found statistically significant strong correlation among three variables 'emp.var.rate', 'euribor3m', and 'nr.employed'. A high recall and relatively low precision for Naïve Bayes means the model returns many results but many of them are incorrectly predicted results.

6.4 Comparing with Balanced Data

To check how can the balanced data help to improve the models, we used RStudio to balance the original dataset. We used the *ovun.sample* function in R ("Ovun.Sample Function | R Documentation," n.d.) to do over-sampling, and under-sampling. This function helps to generates possibly balanced samples by random under-sampling majority examples, and/or over-sampling minority examples. The command we used:

```
data_balance <- ovun.sample(y~.,data =BankDataCleaned_1,
                            N=nrow(BankDataCleaned_1),p=0.5,seed = 1,
                            method = "both")$data
```

Results of the Models on Training Data is shown in Table 8.

Table 8 Results of Models on Training Data

	MBR	Gradient Boost	Probability Tree	Neural Network	Regression	Auto Neural	Maximal tree
Misclassification Rate	0.15	0.17	0.24	0.25	0.26	0.25	0.24
Roc Index (Area Under The curve)	0.95	0.92	0.80	0.80	0.79	0.79	0.80
Precision	0.76	0.87	0.82	0.82	0.81	0.82	0.81
Accuracy	0.84	0.82	0.75	0.74	0.73	0.74	0.75
Recall	0.99	0.76	0.65	0.62	0.61	0.62	0.65
F1 Score	0.85	0.81	0.72	0.70	0.69	0.70	0.72

Results of the Models on Validation Data is shown in Table 9.

Table 9 Results of Models on Validation Data

	MBR	Gradient Boost	Probability Tree	Neural Network	Regression	Auto Neural	Maximal tree
Misclassification Rate	0.20	0.20	0.25	0.25	0.26	0.26	0.25
Roc Index (Area Under The curve)	0.89	0.89	0.80	0.80	0.79	0.79	0.80
Precision	0.60	0.84	0.81	0.81	0.80	0.80	0.81
Accuracy	0.79	0.79	0.75	0.74	0.73	0.74	0.75
Recall	0.95	0.73	0.65	0.63	0.61	0.63	0.65
F1 Score	0.73	0.78	0.72	0.71	0.69	0.70	0.72

We could compare the above results based on balanced data with our results based on the original imbalanced data. It appears the misclassification rate all dropped by doubled, accuracy dropped more than 10% as well. This is because of the increased positive cases and decreased negative creases to the sample data, which will lead to the result of decreasing the number of true negative, and increasing the number of false positive. But the recall and F1 score has largely improved, which means the precision and recall are much more balanced than before. Which means the predictive ability of the "right" customer has been largely improved.

We can generate the conclusion from this comparison that the balanced data is important for prediction, we didn't used this function in the original data because we don't know the reliability of these random generated cases. In the future, in order to improve the model to targeting more efficiently at "right" customer, we need more real data with the customers who subscribed during the campaign.

7 Reference

Asif, M. (2018). *Predicting the Success of Bank Telemarketing using various Classification Algorithms*.

Moro, S., Cortez, P., & Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, *62*, 22–31.

Ovun.sample function | R Documentation. (n.d.). Retrieved December 16, 2020, from https://www.rdocumentation.org/packages/ROSE/versions/0.0-3/topics/ovun.sample

Palaniappan, S., Mustapha, A., Foozy, C. F. M., & Atan, R. (2017). Customer profiling using classification approach for bank telemarketing. *JOIV: International Journal on Informatics Visualization*, *1*(4–2), 214–217.

SAS Enterprise Miner. (n.d.). Retrieved December 16, 2020, from https://support.sas.com/en/software/enterprise-miner-support.html

8 Appendix

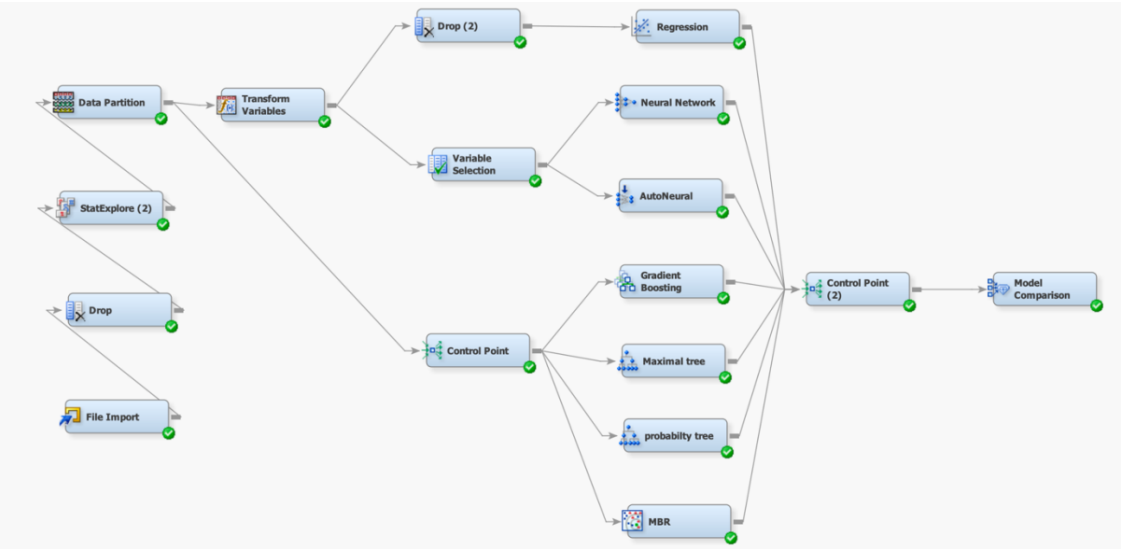

Figure 17 Diagram of the Project (Starting from File Import)