1. INTRODUCTION

Marketing selling campaigns in enterprises are a form of organising marketing activities of product/services. By carrying out campaigns, it can drive more profit to the company that otherwise may not happen. Banking sector is one of the industries which uses such campaigning activities to provide monetary services to people and make profit. To do this, the banks engage in direct marketing by investing a lot of money and infra structure to gain capital. One way of direct marketing in selling is to contact the customers through phone. A call centre allows the bank to communicate with clients from different places.

The aim of our study in this project is to reduce the bank's investment of resources towards marketing by predicting the right number of customers to approach for subscribing to a new bank product like **term deposits**. Here we are using a **Portuguese bank** market campaign dataset who sell term deposits. The project should be able to select the best of clients so that instead of spending on approaching all the customers, the bank can aim selected customers who are more likely to subscribe for the term deposits.

2. DATASET

The dataset used in this project is related to telemarketing campaigns of a Portuguese Bank Institution, University of California, Irvine (UCI). It is located in the machine learning repository of UCI at URL: https://archive.ics.uci.edu/ml/datasets/Bank+Marketing . The link contains four datasets from which we are using 2 datasets namely bank-additional-full.csv with all observations (20 inputs) and bank-additional.csv with 10% of full data (20 inputs). It contains examples from May 2008 to November 2010. The marketing campaign was based on calls and in order to know if the term deposit would be subscribed or not, same customer was contacted more than once.

As mentioned earlier the data set contains 20 input variables including client's personal data like age, job, marital status, loan background etc., contact details like duration, last contacted details and social & economic attributes. The 21st variable will be the binary outcome variable **'y'** which determines whether the customer subscribed to term deposit or not with 2 possible values 'yes', 'no'. The classification goal here is to predict if the client will buy the term deposit.

3. DATA EXPLORATION

To build any data model it is very important to first understand the data completely to maximum extent. The obtained data which is imported into SAS enterprise Miner has to be explored to get insights about what kind of data we have and what else is missing from the data since efficient models cannot be built from weak data.

A statistical examination of data gives us a clear picture about the dataset in our study, which would be helpful for our further analysis. To do the same, a descriptive statistic is drawn on the dataset using nodes like **StatExplore, Multiplot** which are available from the explore tab on the Toolbar of SAS enterprise Miner. Descriptive statistics are very useful in studying the important and basic patterns of data of our interest. They provide simple summaries about each of the variables which resides in the dataset. Along with descriptive statistics, charts for every variable are also plotted which helps an individual to visually observe possible patterns. Rather than presenting the statistics in numbers, graphs allow easy interpretation of facts about data, especially if the data is very large. We shall start finding insights about each of the variables in the dataset.

3.1.1 INTERVAL VARIABLES

We have 9 interval variables in the dataset. None of them have missing values, hence no imputation of values is required. The histograms and bar plots by target variable 'y' for all variables is provided in the Appendix section from Figure 1 to Figure 19. All the interval variables of bank dataset are as explained below:

- *age*: It indicates age of the client. It seems that age does not have much impact on the target variable 'y' since the variance between the two responses of 'y' for age is very less.
  Figure 1 in appendix represents the variable age.
- *Campaign*: It indicates number of contacts performed for this campaign for a client. The value ranges from minimum of 1 to maximum of 56. Figure 2 in appendix represents the variable campaign and it seen that the distribution is skewed. For further analysis we may have to consider this variable for transformation.
- *pdays*: It indicates the number of days after the client was contacted for the last time from previous campaign. The values range from 0 to 27 days and also it has value **'999'** which indicates that the customer was not contacted at all. Around **96% of the examples say that the client was not contacted**. This data makes no sense to training the model since we are required to train a model with data where the customer was contacted and corresponding observations are taken into consideration. Hence, going further we have to either ignore the observations where *pdays=999* or we can replace them with mean/median of variable. Let us select the second option for this study where pdays=999 are replaced and imputed with **median of** *pdays* **= 6.** Figure 17 in appendix represents the plots for this variable.
- *cons.conf.idx*: It is a monthly indicator which determines consumer confidence index. The value ranges from minimum of -50.8 to -26.9. It explains about the country's current and future economic situation each month. Figure 3 in appendix section represents the consumer confidence index. The data does not seem to have even spread throughout the histogram.
- *cons.price.idx*: It is a monthly indictor which determines consumer price index. The value ranges in decimal points from 92 to 95. It explains the changes in prices paid by consumers for goods and services each month. Figure 4 in appendix section represents the consumer price index. The data does not seem to have even spread throughout the histogram.
- *emp.var.rate*: It is a quarterly indicator which determines employment variation rate in the country. The value ranges from -3.4 to 1.4. Figure 5 in appendix section represents employment variation rate in country.
- *euribor3m:* It is a daily indicator which determines the average interest with which the all European banks borrow from each other that mature after 3 months. Figure 6 represents the Euribor Interbank offered rate. The value varies from 0.63 to 5.04.
- *nr.employed*: It indicates the number of persons employed and is a quarterly indicator. The values vary from 4963 to 5228 and the histogram in figure 7 of appendix represent the number of persons employed.
- *previous*: Indicates the number of contacts performed before the current campaign and for a particular client. The value ranges from 0 to 7. Figure 19 in appendix shows the histogram and the distribution w.r.t target variable y. From histogram, we can see that the variables values are highly skewed and can be considered for transformation of variable.
- *duration*: The duration variable has been **rejected** in this analysis because if duration='0' then y='no' which means that we will not know the duration of call without calling the customer and this variable highly effects the target variable 'y'.

3.1.2 NOMINAL VARIABLES

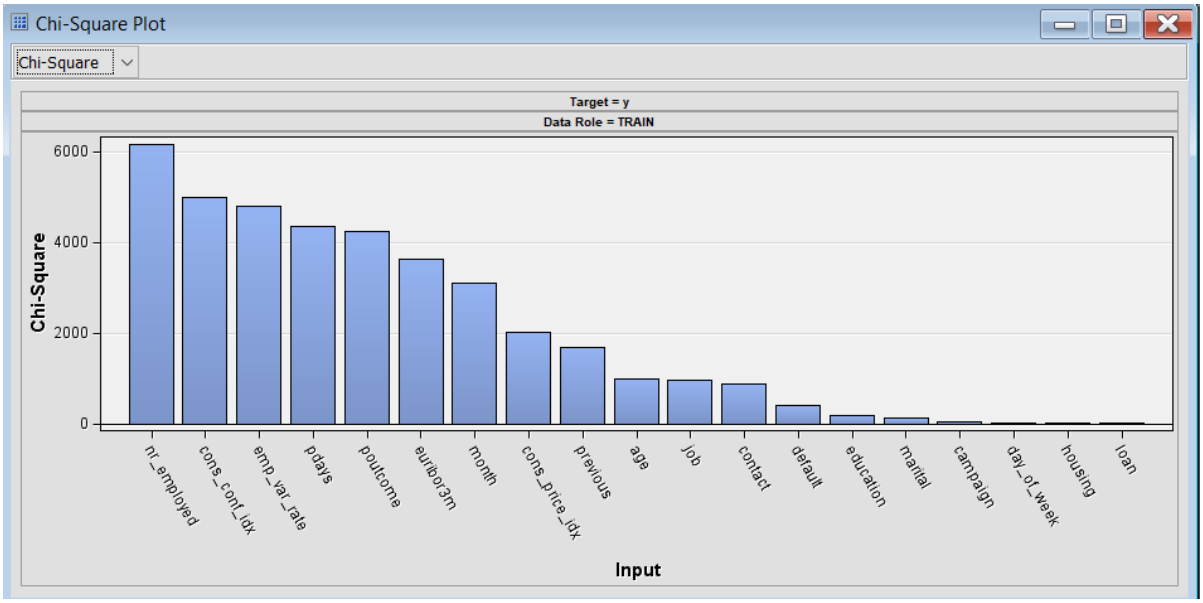
We have 10 nominal variables in our dataset and there are no missing categorical values and thus no imputation is needed. A review of all the nominal variables plot reveals the following observations.

- *contact*: The variable indicates the communication type of the campaign for a particular client. It has 2 possible values “cellular” and “telephone”. Around 63% of clients were contacted through “cellular” and around 37% of clients were contacted through “telephone”. The clients who were contacted through cell phone had more “yes” responses than who were contacted through telephone. Figure 5 in appendix represents the contact variable.

- *day_of_week*: The variable tells the last weekday when the client was contacted. The possible values are 5 weekdays from “monday” to “friday”. The figure 6 in appendix says that the variable did not vary much with respect to target variable ‘y’. It seems that most of the clients were contacted on “mondays” and “thursdays”.
- *default*: Indicates whether a client has a credit card provided by bank or not. The possible values are “yes”,”no” and “uknown”. From the graph in figure 7 of appendix it seems that most of the clients who didn’t have a credit card subscribed for term deposit.
- *education*: The categorical variable indicates the educational qualification of the client. It had 8 possible educational qualifications and it is found from the graph that the parameter did not affect the response variable ‘y’. Figure 8 in appendix reflects the same.
- *housing*: Indicates whether the client already has a house loan borrowed from bank. Includes 3 possible values with “yes”,”no” and “unknown”. Figure 11 in appendix shows that there was slight difference in clients having house loan and not having house loan, getting subscribed for term deposit.
- *job:* Indicates the type of job that a client is employed with. The bank had more clients who were working in admin and blue-collar jobs and its likely to see that they were the ones with more “yes” response for term deposit. Same has been plotted in figure 12.
- *loan:* The variable determines whether the client already has personal loan borrowed from the bank. The possible values being “yes”,”no” and “unknown”, customers who do not have prior personal loans in bank seems to buy the term deposits than the customers who already have personal loan.
- *month:* Indicates the last contact month of the year. Values ranging from Jan to Dec. Figure 14 explains the distribution of month.
- *marital*: Determines the marital status of bank clients. The bank has got more customers who are married and figure 16 demonstrates the same, it is obvious that more “yes” *responses* are from customers whose status is married.
- *poutcome*: It is the outcome variable of previous campaign. The possible values include “success”,”failure” and “non-existent”. Figure 18 in appendix shows the distribution of outcome of previous campaign.
- *y:* In the given dataset, it is observed that the target response ‘y’ has around **89% of “yes”** responses and around **11% of “no”** responses. The same has been plotted in figure 20 of appendix section. This should be taken into consideration while modelling since a biased dataset can lead to create a biased data model. We will further explore the relationships and summary statistics of each of the variables in the dataset using SAS enterprise miner.

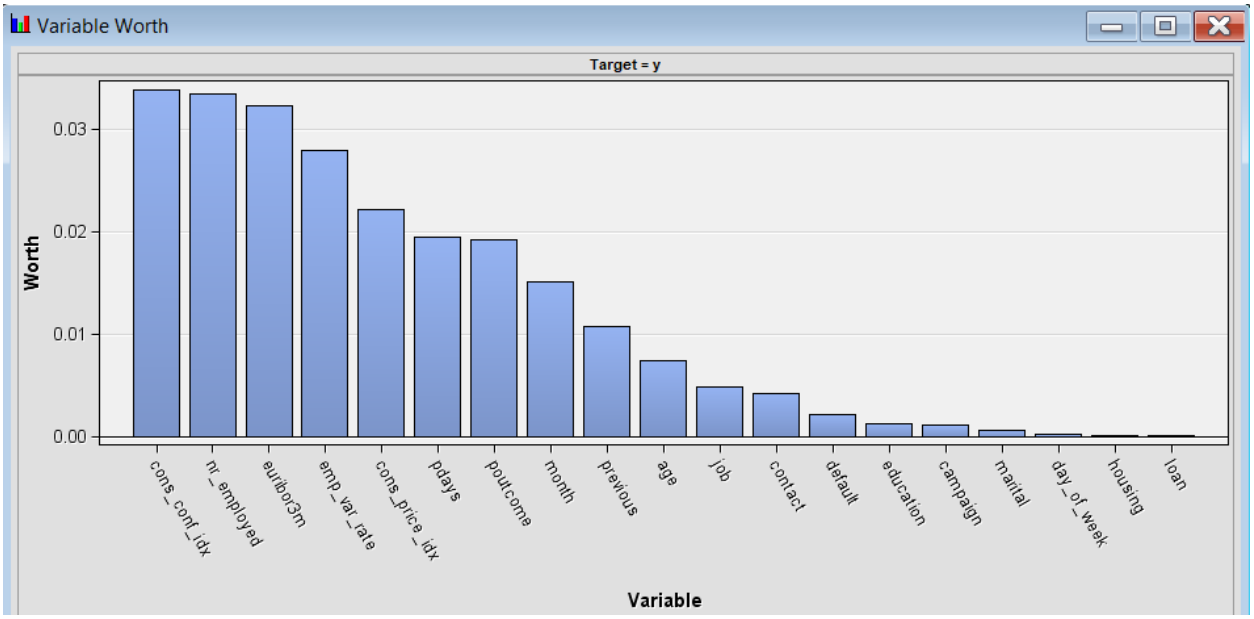
- Using the StatExplore node the following observations were done:

1. Chi-Square Plot



A Chi-square statistic gives us the relationship between each of the variables and the target variable. The Chi-square statistic is very similar to the co-efficient of determination, R-squared value. It the measure of how much a variable can explain the variance in target variable. Since our target variable is categorical Chi-square statistic would be the best way of assessing variable relationships. Reviewing the above plot, we can select variables for our data models and thus do feature selection. From the graph it is evident that *nr.employment* variable has the highest Chi-square statistic followed by other variables like *cons.conf.idx*, *emp.var.rate*, *pdays*, *poutcome*, *euribor3m, month* and so on.

2. Variable Worth



The plot above present a graph through we can easily understand which variable is most related to the target variable and helps to select the best predictor variables. It gives each of the variable's worth in predicting the target variable.

Here, we can see that consumer confidence index (cons_conf_idx) is serving as the best predictor and it is very likely that a customer's spending activity and income stability accounts to predict whether the customer gets subscribed to a term deposit account or not. Further nr_employed followed by euribor3m, emp_var_rate, pdays, poutcome, month are determined as best predictors, in descending order.

3. Output

The output window provides a complete summary of the data in the form of numbers. The output window includes information like the measurement levels of variables and corresponding frequencies, summary statistics for each of the variable types separately and mainly the Target variable statistics with respect to each of the variables present in the dataset.

By examining this summary, we can get to know about what kind of data we have and if there are any missing values in the dataset, we can find them and later work on them to be replaced for better analysis.

Following are the observations made for the current BANK dataset, from the output window:

- From the dataset, 9 interval variables and 10 nominal variables were found.

ROLE	MEASUREMENT LEVEL	FREQUENCY COUNT	VARIABLES NAME	MISSING VALUES
OUTPUT	BINARY	1	y	0
INPUT	INTERVAL	9	age, campaign, cons_conf_idx, cons_price_idx , emp_var_rate, euribor3m, nr_employed, pdays , previous	0
INPUT	NOMINAL	10	Contact, day_of_week, default education, housing, job, loan, marital, month, poutcome	0

- Target variable relationship with other variables

Target Variable	Variable Name	Variable Importance	Variable Worth	Variable Labels
y	cons_conf_idx	1	0.034410	cons.conf.idx
y	nr_employed	2	0.033610	nr.employed
y	euribor3m	3	0.033266	euribor3m
y	emp_var_rate	4	0.026378	emp.var.rate
y	pdays	5	0.022131	pdays
y	poutcome	6	0.021517	poutcome
y	cons_price_idx	7	0.020643	cons.price.idx
y	month	8	0.013983	Month
y	previous	9	0.013096	Previous
y	age	10	0.005833	age
y	contact	11	0.003682	Contact
y	job	12	0.003236	Job
y	campaign	13	0.001408	campaign
y	default	14	0.001143	Default
y	education	15	0.001038	education
y	marital	16	0.000487	Marital
y	loan	17	0.000053	Loan
y	housing	18	0.000030	Housing
y	day_of_week	19	0.000024	day_of_week

The table provides a brief of how much each of the variables can account in predicting the Target variable ‘y’. The column Variable Worth contains the log worth values of variables corresponding to the target variable. Higher the value more the variable can explain the response the variable. The plot in the previous section also explains the same numeric in the form of graph.

The column Variable Importance ranks each of the variable for predicting the Target variable, depending on the variable worth values.

From the table, we can see that variable “cons_conf_idx” can highly contribute than any other variables, to predict the response variable and variable “day_of_week” contributes the least.

4. DATA ANALYSIS AND PREPARATION

4.1.1 Data Partition

The raw bank dataset is now set to divide into train data and validation dataset. Separating data into train, test and validation data is very important when we build data models and very useful during the evaluation of data models. Typically, when we separate a data, a larger partition is used for training the model and smaller partition is used for validating/testing the model and later we check the built model with validation data to make sure there is no existence of overfitting of model. In our project, we are using only train and validation data partitions since testing is not included here.

- To carry out data partition in SAS enterprise Miner, we have a node named **Data Partition** in Sample section of SEMMA tool bar.
- In the properties panel of Data Partition, data allocations tab has been set to **55% of training** data and **45% of validation** data.
- The data partition method used here is default which selects to **stratify** the data on target variable. Since we have 89% of “no” responses and 11% of “yes” responses in target variable, Stratifying ensures that both responses “no” and “yes” are **well-represented** in data partitions.
- Once we run the node, the data gets partitioned with proportions as mentioned above and we can see the same in results. In addition, we can also see the proportion of each variable which got partitioned into train and validate data.

Partition Summary

Type	Data Set	Number of Observations
DATA	EMWS1.Stat_TRAIN	41188
TRAIN	EMWS1.Part_TRAIN	22652
VALIDATE	EMWS1.Part_VALIDATE	18536

4.1.2 Data Replacement

As discussed earlier in Data analysis part, we observed that the variable *pdays* contains a value ‘999’ which needs to be replaced. Here, we are replacing these values with missing values(.) in both training and test data sets.

- For data replacement in SAS Miner, we have a node called **Replacement in the Modify** section of SEMMA tool bar. The partitioned data node is connected to replacement node.
- The replacement window of interval variables has been customised to add **missing** values as replacement for *pdays*=999 in properties panel and Default limit methods has been switched to **none** since we do not want to enforce the replacement and we will do it manually.
- In replacement editor of interval variable, we specify the variable to be replaced and what values to be replaced.

Interactive Replacement Interval Filter

Columns: Label Mining Basic Statistics

Name	Use	Limit Method	Replacement Lower Limit	Replacement Upper Limit	Replace
age	Default	Default	.	.	Default
campaign	Default	Default	.	.	Default
cons_conf_idx	Default	Default	.	.	Default
cons_price_idx	Default	Default	.	.	Default
emp_var_rate	Default	Default	.	.	Default
euribor3m	Default	Default	.	.	Default
nr_employed	Default	Default	.	.	Default
pdays	Default	User Specified	.	998	Default
previous	Default	Default	.	.	Default

Since we want all '999' to be replaced we will specify the Replacement upper limit as '998' which replaces all values greater than '998'

- Once we run the node, the output window gives the number of values replaced in both training and validation dataset.

Total Replacement Counts

Variable	Label	Role	Train	Validation
pdays	pdays	INPUT	21857	17816

A total of (21857+17816) 39673 values in *pdays* have been replaced as missing values(.)
We can also see an added column in the dataset named *Replacement: pdays* which holds the replaced column values of *pdays*

EMWS1.Repl_TRAIN

contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	euribor3m	y	emp.var.rate	cons.price.idx	cons.conf.idx	nr.employed	Replacement: pdays
elephone	may	mon	261	1	999	0	nonexistent	4.857	no	1.1	93.994	-36.4	5191	
elephone	may	mon	151	1	999	0	nonexistent	4.857	no	1.1	93.994	-36.4	5191	
elephone	may	mon	307	1	999	0	nonexistent	4.857	no	1.1	93.994	-36.4	5191	
elephone	may	mon	139	1	999	0	nonexistent	4.857	no	1.1	93.994	-36.4	5191	
elephone	may	mon	55	1	999	0	nonexistent	4.857	no	1.1	93.994	-36.4	5191	
elephone	may	mon	222	1	999	0	nonexistent	4.857	no	1.1	93.994	-36.4	5191	
elephone	may	mon	137	1	999	0	nonexistent	4.857	no	1.1	93.994	-36.4	5191	
elephone	may	mon	293	1	999	0	nonexistent	4.857	no	1.1	93.994	-36.4	5191	
elephone	may	mon	146	1	999	0	nonexistent	4.857	no	1.1	93.994	-36.4	5191	

Now the data is ready for data modelling. In the next section, we shall start implementing different machine learning algorithms to build predictive models.

5. DATA MODELLING

The aim of our project is to build predictive models for the bank dataset by using all relevant algorithms available in SAS enterprise miner and evaluate each of them to find the best model for predicting the target variable 'y'. So here we are starting with training a decision tree model since this model does not need imputation to deal with missing values.

1. DECISION TREE:

Decision Tree is a supervised learning algorithm which is easily understood and interpreted among all classification algorithms. It is mostly used for classification problems but also for regression problems. When training a dataset to classify a variable, the decision tree makes

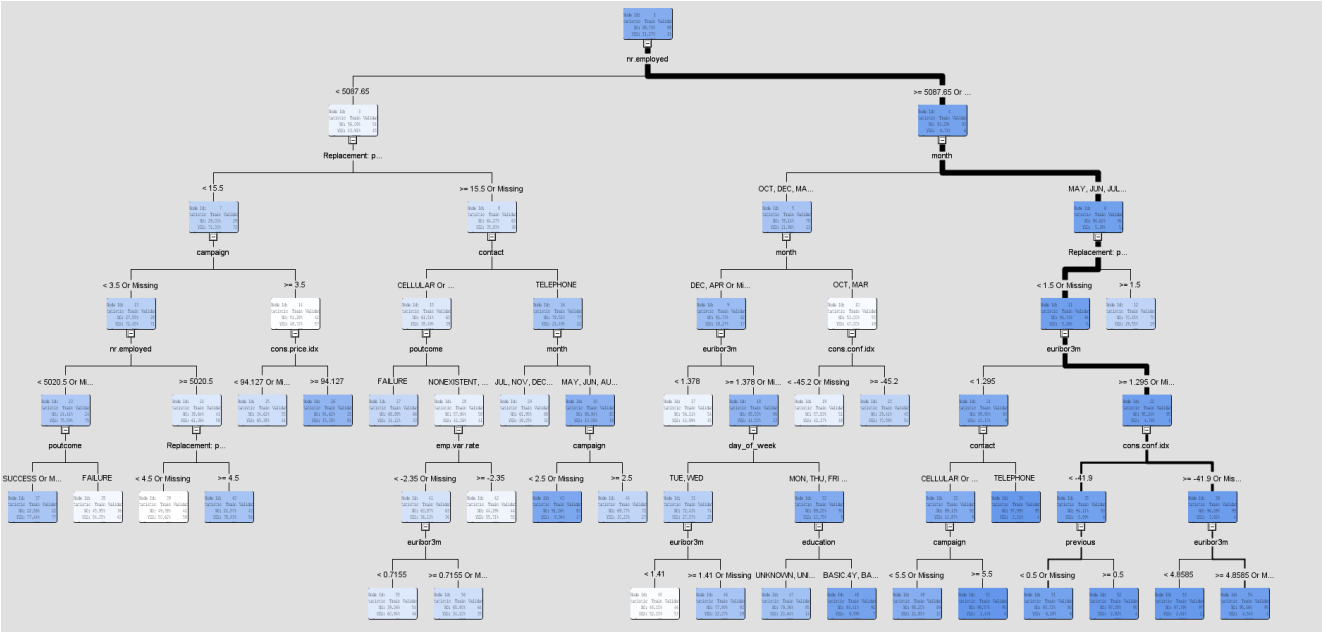
smaller subsets of data based on certain node rules until the target variable falls under one category. The selection of variables to be splitted is calculated by the system based on parameters like maximum information gain.

Here in this project we are building 2 models using decision tree, one is the interactive or maximal decision tree model and another one is an auto-pruned decision tree model. To manage the missing values in decision trees, we have customised the algorithm to use up to **4 surrogate rules** when it encounters an input value with missing observations.

1a. **Maximal Decision tree:**

The Maximal decision tree works as an interactive decision tree where a user can select the variables to be splitted at each node of the tree. The basic idea of maximal tree is the decision tree is splitted upto maximum extent using all of the variables resulting in a massive decision tree. The algorithm is **Chi-square** driven and the selection of variables at nodes depend on the Chi-square driven statistic value **-log(p)** of the variables. Since the model is interactive, we can see which variables have the best Chi-square statistic and also the best point at which the node gets splitted to. Each time a variable is selected and node is created, for the next node the variables -log(p) decreases which explains that the current variable becomes the leaf node for the first selected variable.

In SAS enterprise we have got number of algorithms to use in which we can choose **Decision tree** for now. The maximal decision tree is created by using the **Train node** option which splits the decision tree to maximum number of branches. Below is the maximal tree created.

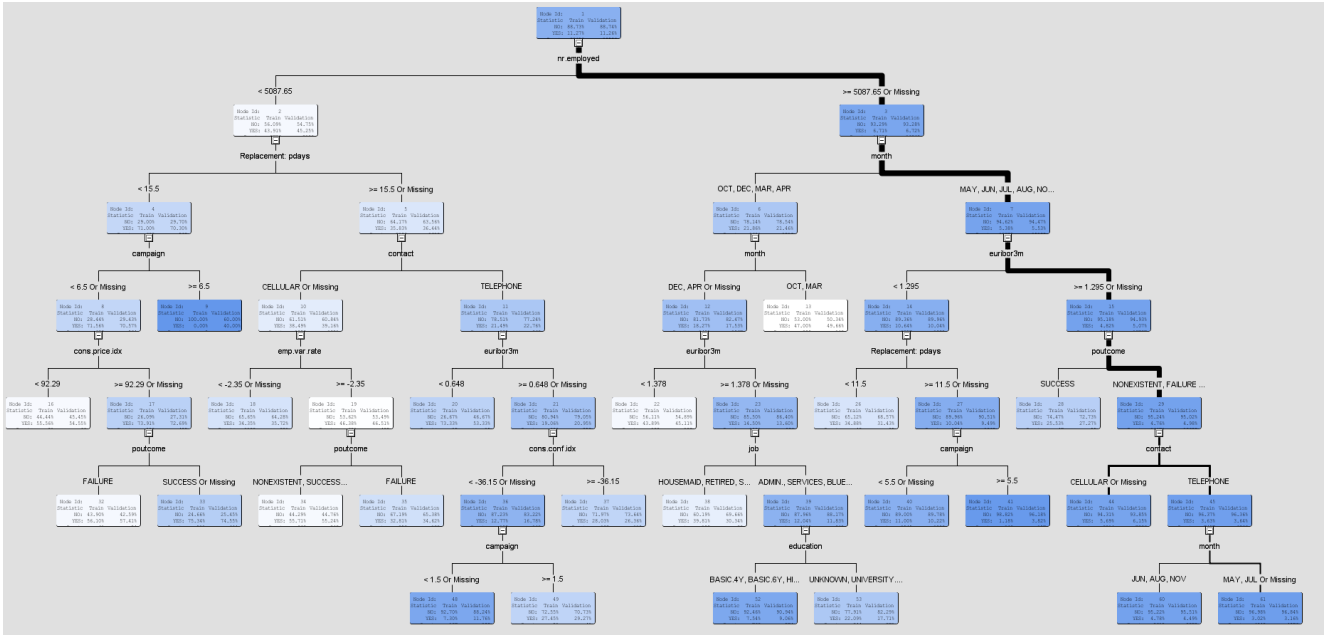


As we can see, the tree contains a number of branches and leaf nodes. The tree in total contains **54 leaf nodes**. The misclassification rates for the model are **0.0972** and **0.0991** respectively for train and validate data.

The maximal tree is not best decision tree to use, the reason is that specifying too many splits in the decision tree results in overfitting of the data model to the current dataset which should not happen. If we happen to select a smaller number of variables then we will get a higher accurate predictive data model when it is exposed to an unknown dataset. Hence, the decision tree needs to be pruned and it will be done in our next model.

1b. **Regular decision tree (Auto-pruned)**:

In regular decision tree the **subtree method** is selected as **Assessment**, which means that the decision tree first creates a maximal tree and then prunes the same to get a final effective decision tree with reduced set of branches. Here, we know that the target outcome should be '1' or '0' and thus we select the **assessment measure** as **Decision.** This tries to optimise the branches that would decide to predict a '1' or '0' for target variable.



As we can see from above built model, the original decision tree is trimmed to **44 leaf nodes**. The leaf nodes have been **decreased by 10 numbers** which results in better performance when it is used for a new dataset. The misclassification rate is found to be **0.0986** and **0.099** for train and validation data respectively.

2. REGRESSION

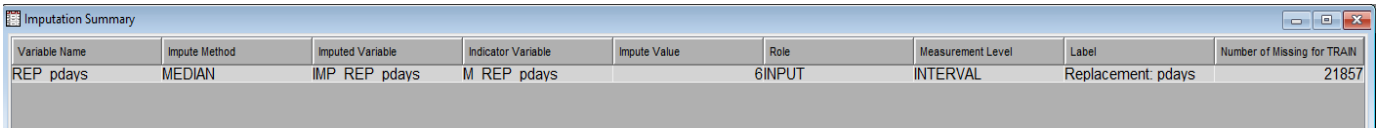
Before doing regression, we have to consider that whether our data meets the criteria for being analysed using regression. The things we have to check about is missing values in the columns of data and the considering the transformation of variables which are not normal.

a. Handling missing values

In case of **decision trees handling the missing values** is not a problem since we have **surrogate rules** for rescue. This specification allows SAS enterprise miner to use upto the mentioned number of surrogate rules in each non-leaf node if the splitting rule depends on an input having missing values. But in regression, the case is that the algorithm automatically ignores the observations which contains 50% of missing values. Hence if we want to make use of all the data, we have to impute the missing values. Until now we have got missing values in only one variable i.e. *pdays*. In this study, these **missing values are imputed** with the **median** of the non-missing values of variable *pdays.*The missing value **cut off is set to 97%** since we have 96% missing.

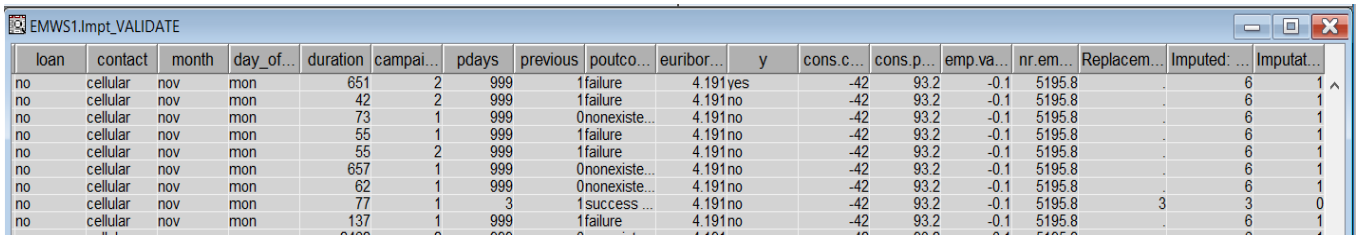
- Before regression, an **impute** node has been used in SAS enterprise miner to replace missing values. The default input method is selected as median and hence median of *pdays* i.e. '**6'** will be imputed in place of all missing values. By default, the missing cut

- Further we will be creating a new column with imputed values and use it as an input in the data modelling. For this we have to do personalise the settings for indicator variable with **Type** set as **Unique** and **Role** set to **input**. In addition, we will get a variable which indicates whether a value is imputed or not: '1' for imputed and '0' for not imputed, and this will also be an input variable. Both of those become inputs to be tested whether or not they affect the outcome variable.
- Once we run the node, the values get imputed and we can see two additional columns in the data as shown below:


Imputation Summary

Variable Name	Impute Method	Imputed Variable	Indicator Variable	Impute Value	Role	Measurement Level	Label	Number of Missing for TRAIN
REP_pdays	MEDIAN	IMP_REP_pdays	M_REP_pdays	6	INPUT	INTERVAL	Replacement: pdays	21857

For example, we can see that in train data 21857 values are replaced and the same happens in validation data which replaces 17816 values.


EMWS1.Impt_VALIDATE

loan	contact	month	day_of...	duration	campai...	pdays	previous	poutco...	euribor...	y	cons.c...	cons.p...	emp.va...	nr.em...	Replacem...	Imputed: ...	Imputat...
no	cellular	nov	mon	651	2	999	1	failure	4.191	yes	-42	93.2	-0.1	5195.8	.	6	1
no	cellular	nov	mon	42	2	999	1	failure	4.191	no	-42	93.2	-0.1	5195.8	.	6	1
no	cellular	nov	mon	73	1	999	0	nonexiste...	4.191	no	-42	93.2	-0.1	5195.8	.	6	1
no	cellular	nov	mon	55	1	999	1	failure	4.191	no	-42	93.2	-0.1	5195.8	.	6	1
no	cellular	nov	mon	55	2	999	1	failure	4.191	no	-42	93.2	-0.1	5195.8	.	6	1
no	cellular	nov	mon	657	1	999	0	nonexiste...	4.191	no	-42	93.2	-0.1	5195.8	.	6	1
no	cellular	nov	mon	62	1	999	0	nonexiste...	4.191	no	-42	93.2	-0.1	5195.8	.	6	1
no	cellular	nov	mon	77	1	3	1	success ...	4.191	no	-42	93.2	-0.1	5195.8	3	3	0
no	cellular	nov	mon	137	1	999	1	failure	4.191	no	-42	93.2	-0.1	5195.8	.	6	1

In the above figure we can see that there are totally 3 columns: **replaced column**, **imputed column** and **imputation indicator column**. The missing values are replaced by value **'6'**. Once the values are imputed the original *pdays* column **role** becomes **rejected**.

b. Transformation of variables

After imputing the variables, we should not consider transforming some of our variables which does not look normal. Transforming the data before modelling always improves the response of model by stabilizing its variance, removing non-linearity and handle the data which is not normal. We can opt to transform one or more variables in our dataset.

Here we are selecting two variables for transformation: ***campaign*** and ***previous*** since these interval variables does not seem to be normal when checked for skew and kurtosis.

The **common log transformation** method is used to control the skewness of the variable. As a result, 2 new transformed variable columns are added and we can see that the common log transformation tries to normalise both the variables.

Variable Name ▲	Skewness	Kurtosis
campaign	4.460463	30.70516
LG10_campaign	1.361957	2.056726
LG10_previous	2.561209	5.964269
previous	3.740137	18.76213

Now our data is ready for building a regression model. We drag a regression model into the workspace and connect the impute node to it and directly run the model.

- The default settings of regression node build a model where all variables are used to build the model and the results are as shown below. The significance level of accepting any model as significant is considered as **0.05.**

Likelihood Ratio Test for Global Null Hypothesis: BETA=0

-2 Log Likelihood Intercept Only	Intercept & Covariates	Likelihood Ratio Chi-Square	DF	Pr > ChiSq
15948.963	12484.936	3464.0276	51	<.0001

From the above screenshot we can say that our regression model is statistically significant **(<0.0001)** and it is capable of explaining the target variable 'y'.

- Since we have used the default setting of regression the algorithm uses all variables to build the model and we can see from the results that not all variables are significant in explaining the target variable 'y'.

Type 3 Analysis of Effects

Effect	DF	Wald Chi-Square	Pr > ChiSq
IMP_REP_pdays	1	0.2077	0.6486
LG10_campaign	1	4.6320	0.0314
LG10_previous	1	2.6314	0.1048
M_REP_pdays	1	17.8411	<.0001
age	1	0.4796	0.4886
cons_conf_idx	1	9.2160	0.0024
cons_price_idx	1	161.2519	<.0001
contact	1	73.5709	<.0001
day_of_week	4	24.4248	<.0001
default	2	8.8778	0.0118
education	7	10.7133	0.1516
emp_var_rate	1	96.3661	<.0001
euribor3m	1	16.3261	<.0001
housing	2	0.4312	0.8061
job	11	15.5014	0.1607
loan	1	0.1106	0.7395
marital	3	0.9982	0.8017
month	9	227.9606	<.0001
nr_employed	0	0.0000	.
poutcome	2	6.2854	0.0432

- Here comes the use of **Stepwise** model selection where the algorithm automatically selects the variables based on their Chisq value stepwise and runs the model. The results of this model are as shown below:

Likelihood Ratio Test for Global Null Hypothesis: BETA=0

-2 Log Likelihood Intercept Only	Intercept & Covariates	Likelihood Ratio Chi-Square	DF	Pr > ChiSq
15948.963	15948.963	0.0000	0	.

Type 3 Analysis of Effects

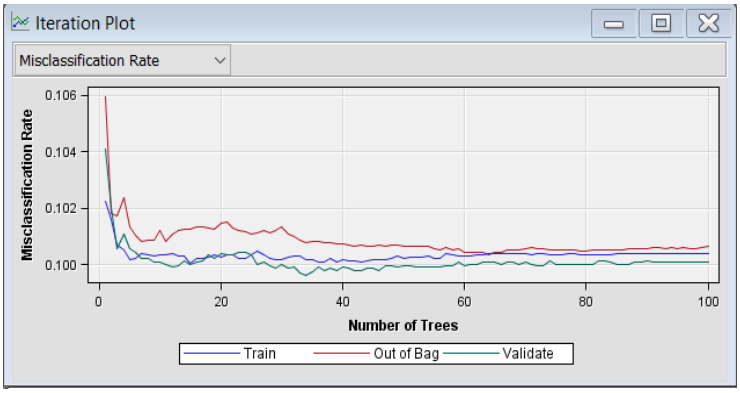
Effect	DF	Wald Chi-Square	Pr > ChiSq
LG10_campaign	1	4.5329	0.0332
M_REP_pdays	1	17.3044	<.0001
cons_conf_idx	1	10.0781	0.0015
cons_price_idx	1	165.6903	<.0001
contact	1	73.0560	<.0001
day_of_week	4	24.4489	<.0001
default	2	10.5338	0.0052
emp_var_rate	1	100.1223	<.0001
euribor3m	1	17.4326	<.0001
month	9	243.4806	<.0001
poutcome	2	79.3880	<.0001

The process uses **21 steps** to select the best variables for the model with **number of variables reduced to 11**, capable of efficiently predicting the target variable 'y'. The misclassification rate was found to be **0.0992** for both train and validation data.

3. RANDOM FOREST

Random forest is classifier algorithm which uses a number of decision tress to build a best predict data model. The algorithm selects the variables depending on the variable importance calculated and several number of weak decision trees are created. As the number of tress increase the model becomes gradually efficient with decreasing error rate. Some of the interesting observations in the random forest model as are explained below:

Variable Name	Number of Splitting Rules	Train: Gini Reduction	Train: Margin Reduction	OOB: Gini Reduction	OOB: Margin Reduction	Valid: Gini Reduction	Valid: Margin Reduction	Label
euribor3m	253	0.007169	0.014337	0.00679	0.01391	0.00705	0.014831	
nr_emplo...	217	0.009512	0.019023	0.00903	0.01864	0.00942	0.019891	nr.emplo...
month	210	0.003555	0.007109	0.00336	0.00692	0.00321	0.007146	
cons_co...	200	0.003163	0.006327	0.00285	0.00600	0.00288	0.006454	cons.con...
cons_pri...	157	0.001065	0.002130	0.00096	0.00200	0.00101	0.002254	cons.pric...
previous	153	0.000419	0.000838	0.00022	0.00064	0.00010	0.000433	
poutcome	145	0.004598	0.009196	0.00431	0.00892	0.00439	0.008736	
contact	137	0.000645	0.001290	0.00053	0.00118	0.00049	0.001390	
M_REP_...	129	0.007458	0.014915	0.00745	0.01490	0.00756	0.014546	Imputatio...
emp_var...	123	0.002579	0.005159	0.00246	0.00502	0.00244	0.005524	emp.var.r...
job	109	0.000640	0.001281	0.00026	0.00088	0.00036	0.000946	
age	102	0.000458	0.000915	0.00012	0.00058	0.00021	0.000632	
day_of_w...	87	0.000413	0.000826	0.00020	0.00062	0.00021	0.000516	
education	77	0.000187	0.000374	0.00001	0.00025	-0.00003	0.000200	
IMP_RE...	58	0.001237	0.002474	0.00112	0.00235	0.00115	0.002297	Imputed: ...
campaign	55	0.000109	0.000219	-0.00005	0.00006	-0.00001	0.000126	
marital	42	0.000062	0.000124	-0.00002	0.00005	-0.00002	0.000057	
default	33	0.000056	0.000111	0.00004	0.00008	0.00003	0.000066	
housing	27	0.000036	0.000073	-0.00005	-0.00002	-0.00003	0.000017	
loan	20	0.000017	0.000034	-0.00002	0.00000	-0.00001	0.000017	



The table in the left gives the variable importance indicating *euribor3m, nr.employed, month* as the variable with most importance and *loan* with least. Depending on this calculation the algorithm tries to build number of weak decision tree models. Also, the second graph indicates that there were **100 weak decision trees built** and **as number of trees increased the Misclassification error rate decreased** trying to make the model to behave as the best fit. Until 20 trees the misclassification rate gets decreased and further the trend becomes stagnant and finally at 100th decision tree the misclassification rate is **0.0994 and 0.0998** for train and validation data respectively, which is a good measure comparatively.

4. SUPPORT VECTOR MACHINE (SVM)

A support vector machine is a binary classifier algorithm where it separates the observations into two classes. There is a margin drawn in between the two observations, the more we can separate these two models by increasing the margin, we will have a better and more accurate model when applied to an unknown data. Since all data cannot be separated with a 1-D margin we use kernel functions where a higher-dimensional margin is placed between the observations. A selected kernel and its parameters accounts in how well a classification model is built.

Here, in our SVM model we tried different kernel functions and have selected **linear kernel** with **2 polynomial degree** as the **best option** to deal with.

The misclassification rate of the model is found as **0.091** and **0.101** for train and validation data respectively. We can see that there is an increase in misclassification rate in validation data which says that the model is slightly fit to the train data which has less misclassification rate. Also, we can see that the true positive numbers have decreased from Train data to Validation data as shown beside:

Event Classification Table

Data Role=TRAIN Target=y Target Label=' '

False Negative	True Negative	False Positive	True Positive
1799	19832	268	753

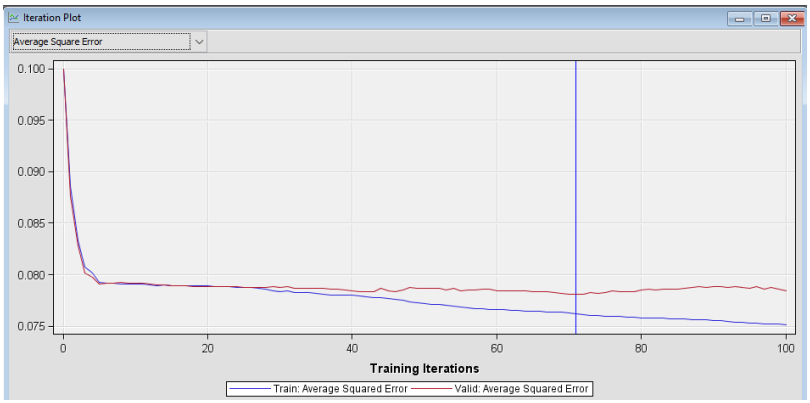
Data Role=VALIDATE Target=y Target Label=' '

False Negative	True Negative	False Positive	True Positive
1587	16152	296	501

5. NEURAL NETWORK

Neural Network is an algorithm which can build model having non-linear relationships between the predictor and target variables. Here the processing units called **neurons** are arranged in layers and we will be having 3 layers **input, hidden** and **an output layer**.
For neural network models it is better if we do variable selection since a large number of inputs enforces more error rate into the neural network model. Hence, we will be using a **variable selection** node, here the variables which have **low Chi-square value are rejected** so that we will be left with less number and effective input variables. As a result, a **reduced set of 11 variables** are fed to the neural network.
In this neural network we have made a direct connection to between the input and output layer in addition to the connection of hidden and output layers. The **number of hidden layers** here are selected as **6** with an estimation of number of layers = number of inputs/2.

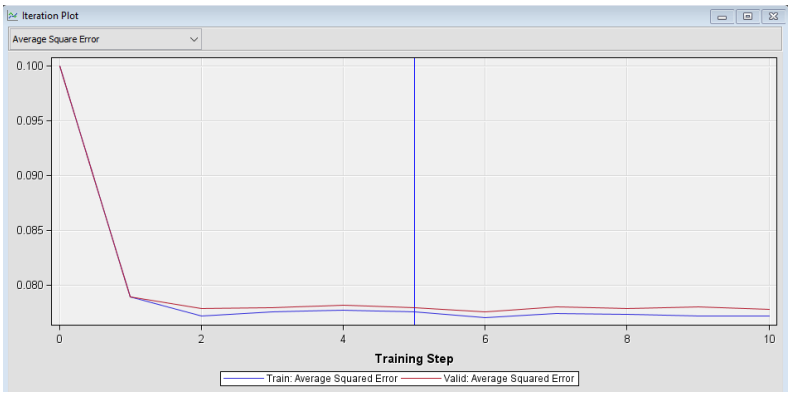
The iteration plot explains the measurement of average squared error with respect to number of optimal iterations.



The iteration plot shows that even after doing variable selection, there in an **increase in average square error rate from train data to validation data**. The **misclassification rate** is also recorded as **0.098** and **0.1** for train and validation data respectively, which does not show much difference.

6. AUTO-NEURAL NETWORK

In auto-neural network, instead of setting the number of hidden layers manually the algorithm calculates the **best number of hidden layers** to the model. The algorithm performs number of **iterations** adding the hidden layers one-by-one by to get the best model fit. For this we have to mention the algorithm that number of hidden layers to be incremented for each iteration as '1' and we will de activate the direct connection between input and output layer. Below is the result of auto neural network that we have built, here we are using the same input from variable selection node as in neural network model.



Here, we can see and try to compare the iteration plots of neural-network model and auto neural network model. In this plot, after **fourth iteration with 4 hidden layers** there is a **minimal difference** between the **average square error** of train and validation data. The misclassification rate is also almost same for train and validation data being **0.0985**.

6. MODEL COMPARISION

Each of the data models which are built in previous section are evaluated to check which model performs the best in explaining the target variable 'y' and that becomes the best data model for predicting the target variable.
When we run a data model, we will get different number of statistics which explains the fitness and usefulness of each model. Here we are using two such parameters to assess the best predicting model for our data set, namely **Misclassification rate** and **ROC index/Area Under Curve (AUC).**

- **Misclassification rate:**

The assessment of data model here depends on the concept of **confusion matrix**. Below is the explanation for confusion matrix.

	PREDICTED (0)	**PREDICTED (1)**
ACTUAL (0)	**TRUE NEGATIVE (TN)**	**FALSE POSITIVE (FP)**
ACTUAL (1)	**FALSE NEGATIVE (FN)**	**TRUE POSITIVE (FN)**

- TRUE POSITIVE: It represents correct prediction of actual value as positive by the model.
- TRUE NEGATIVE: It represents correct prediction of actual value as negative.
- FALSE POSITIVE: It represents incorrect prediction of actual value as positive.
- FALSE NEGATIVE: It represents incorrect prediction of actual value as negative.

From the confusion matrix, classification error= (FP+FN) / (TP+TN+FP+FN). This is called as Misclassification error rate which indicates the **overall incorrectness of the classifier model**.

The table 1 in appendix shows misclassification rate for each of the built models. It is always good to have low values of mis classification error rate.

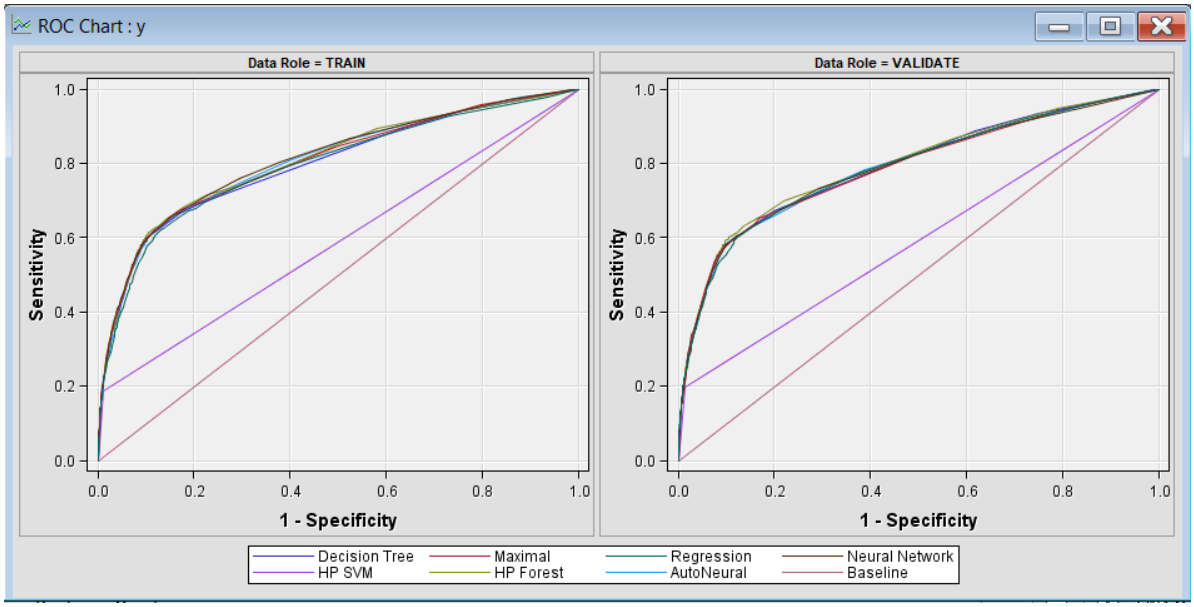
- **ROC (Receiver Operating Characteristic) Index / AUC (Area Under Curve)**

From above knowledge of confusion matrix, we draw a graph of Receiver Operating Characteristic. We consider 2 terminologies here: Sensitivity and Specificity.

- Sensitivity: It detects how often a model correctly predicts the actual value as positive.
- Specificity: It detects how often a model correctly predicts the actual value as negative.

Therefore, ROC can be explained as a graph of **Sensitivity** in y-axis and **1-Specificity** in x-axis. Basically, ROC curve is a plot between True positives and false positives of a data model. **Area Under ROC Curve** indicates that as the ROC curve gets pulled towards '1' in y-axis (AUC of '1' makes an ideal model), the area under curve increases and means that the model is capable of predicting more True Positives than false positives. As a result, the accuracy of predicting model also increases.

The measures of misclassification rate and ROC Index for each of the models built are as tabulated below and figure (a) below represent a visual representation of ROC Index for both train and validation data.



figure(a): Comparison of ROC charts for all the data models

MODEL NAME	MISCLASSIFICATION RATE	ROC INDEX
Auto Neural Network	***0.098511***	***0.806***
Decision tree	0.09905	0.797
HP Forest	0.100129	0.089
HP SVM	0.102773	0.795
Maximal Tree	0.099158	0.803
Neural Network	***0.100183***	***0.81***
Regression	0.099158	0.794

As per the observations above, we can say that among all the predictive models Auto Neural Network has got the least misclassification and SVM has got highest misclassification error rate. When it comes to ROC, Neural Network has got the highest Area Under Curve and SVM has got the lowest Area Under Curve. Therefore, after reviewing the results we can say that **Auto Neural Networks** and **Neural Networks** as the **best predicting data models** for our Portuguese bank dataset **to predict the customers** who will get subscribed to the bank's term deposit service.

The possible reason that Auto Neural Network and Neural Network were built as best models is the kind of input variables fed into the data models. Unlike from the inputs for other data models, here we have used variable selection method to cut down all the variables which do not potentially have the capability of explaining the variation in target variable and the inputs were transformed to yield the normalised form of the variables which gets added to the efficiency of model. However, when it comes to neural network models, it becomes a black box and interpreting the output of a neural network is nearly not possible. But still there are many approaches from which we can interpret the working and an example of one such approach is convolutional neural networks.

7. COMPARISION OF RELATED WORK

A related study for this work was carried out from a Portugal research unit. In this paper they proposed the same approach of using data mining models for predicting the customers who get subscribed to the term deposit of Portuguese bank.

When we compare our work with the paper following observations were made:

RELATED WORK	CURRENT APPROACH
The paper emphasised more on feature selection, feature engineering since they have 150 attributes to filter and finally used 22 relevant features	In our study we had 20 inputs to work in the dataset provided
Feature selection was done both manually and adapted forward selection method	In our study we used all the variables except *duration* which was trimmed manually
Four models were compared namely Logistic regression, Decision trees, Neural Networks and Support Vector Machines	Seven models are compared namely Decision tree, Maximal decision tree, Logistic regression, Random forest, Support Vector Machines, Neural Networks, Auto-Neural Networks
There is no information on variable transformation	The common log transformation method was used for variable transformation
No particular information about the values of varaible *pdays* is available in the paper	We have replaced and imputed (median imputation) the values of *pdays*='999' which means that the customer was not contacted.
The performance of models was assessed using two measurements: Area Under Curve (AUC) and Area of the LIFT cumulative curve (ALIFT)	The performance of models is assessed by Area Under curve (AUC) and misclassification rate
Data has been partitioned as train and test data.	Data is partitioned to train and validation data.
The model is realistic and gets updated frequently by using a procedure called rolling window evaluation method where new contacts gets added to the test data.	The model is not designed to get updated.
Finally, Neural Network was evaluated as the best model with AUC of 0.80 and ALIFT of 0.67	The final best model was assessed as Auto-Neural network with AUC of 0.80 and misclassification rate of 0.0985

8. CONCLUSION

In this study, different predictive models have been built to predict the bank customers those who gets subscribed to the new product of the Portuguese bank, "term deposits", so that the bank can target those customers during campaigning. The bank dataset we used contains 41188 observations with 20 input variables. We used 19 input variables for the data modelling rejecting the duration variable since it highly affects the outcome variable when customer is not at all contacted. Seven predictive models are built in this study namely Decision tree, Maximal decision tree, Logistic regression, Random forest, Support Vector Machines, Neural Networks and Auto-Neural Networks. The models were evaluated using two model assessment measures Area Under Curve (AUC) and misclassification rate. For both the metrics, Auto Neural Networks and Neural Networks was assessed as the best fits with AUC of 0.80 and misclassification rate of 0.0985. The workflow of the data model building process using SAS Enterprise Miner, can be visualised from Figure 21 in Appendix.

9. Appendix

Figure 1: **Distribution of age** & **age by 'y'(- NO - YES)**

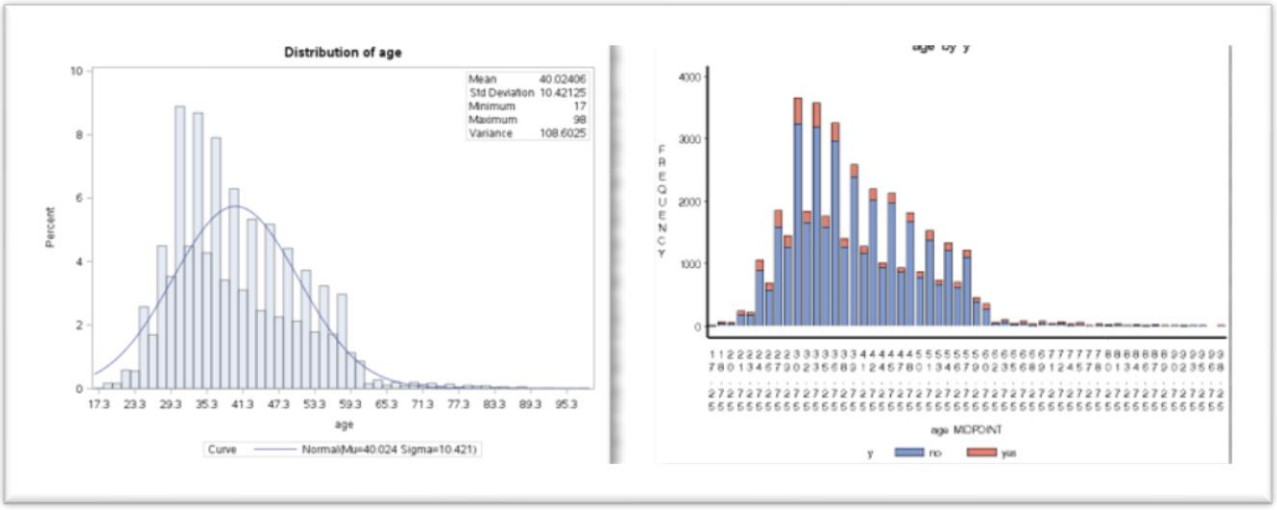


Figure 2: **Distribution of campaign** & **campaign by 'y'**

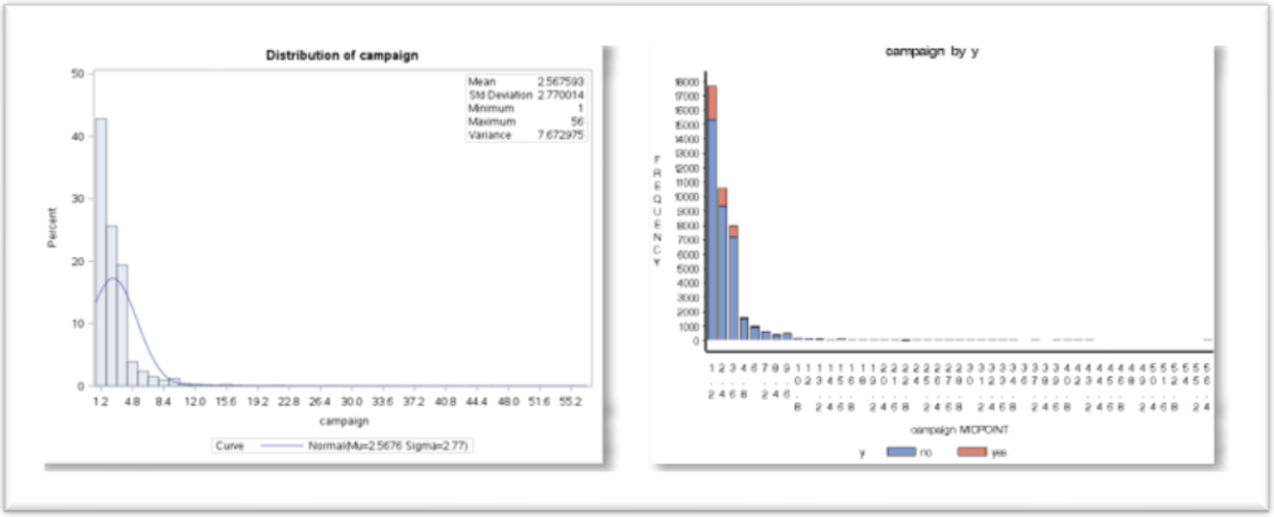


Figure 3: **Distribution of cons.conf.idx** & **cons.conf.idx by 'y'**

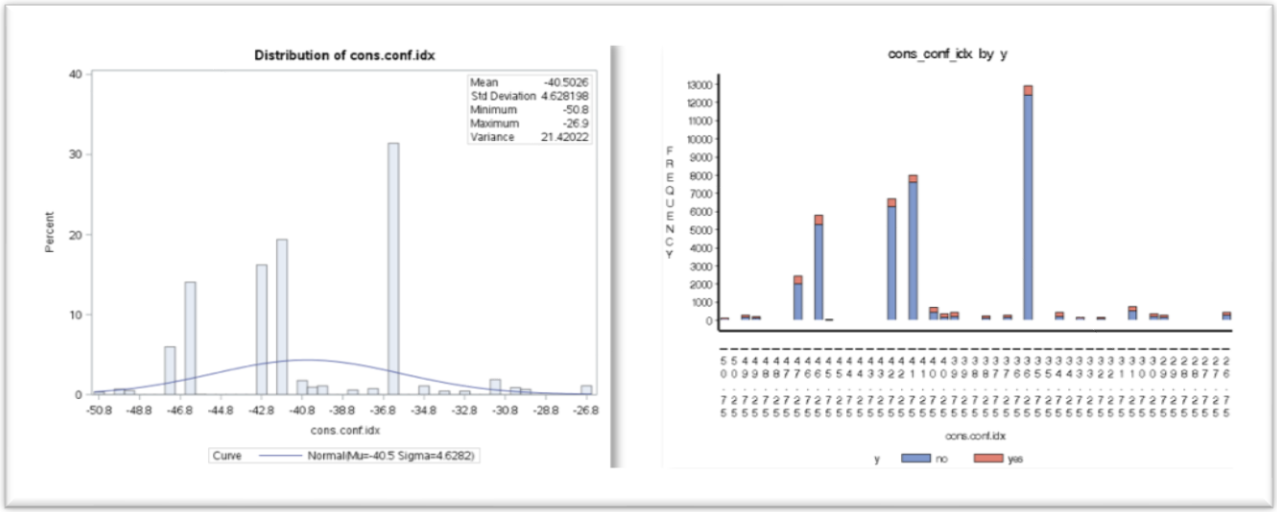


Figure 4: **Distribution of cons.price.idx** and **cons.price.idx by 'y'**

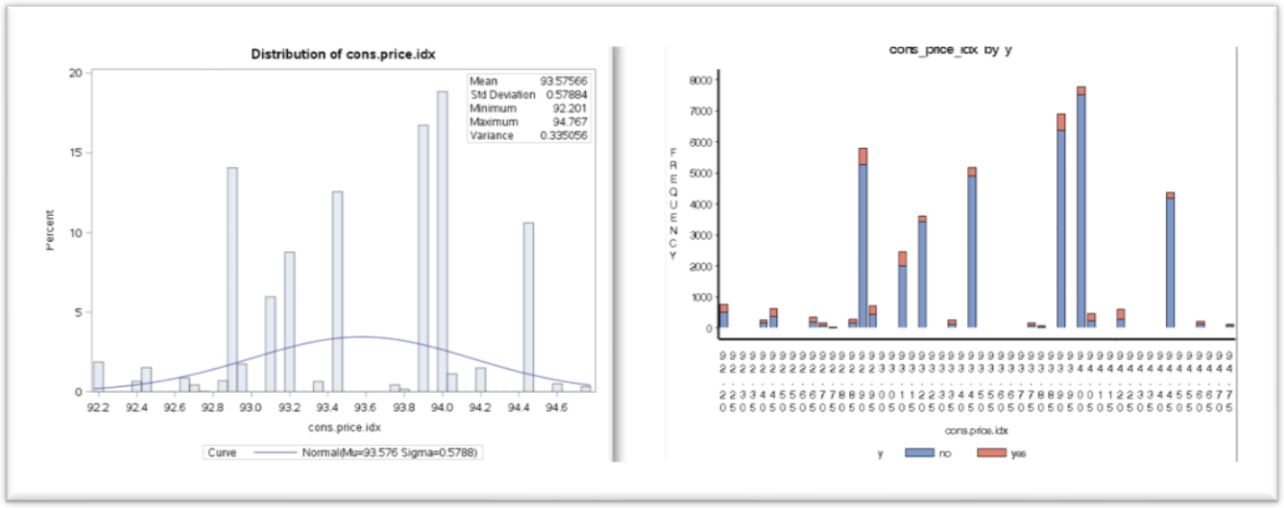


Figure 5: **Distribution of contact** & **contact by 'y'**

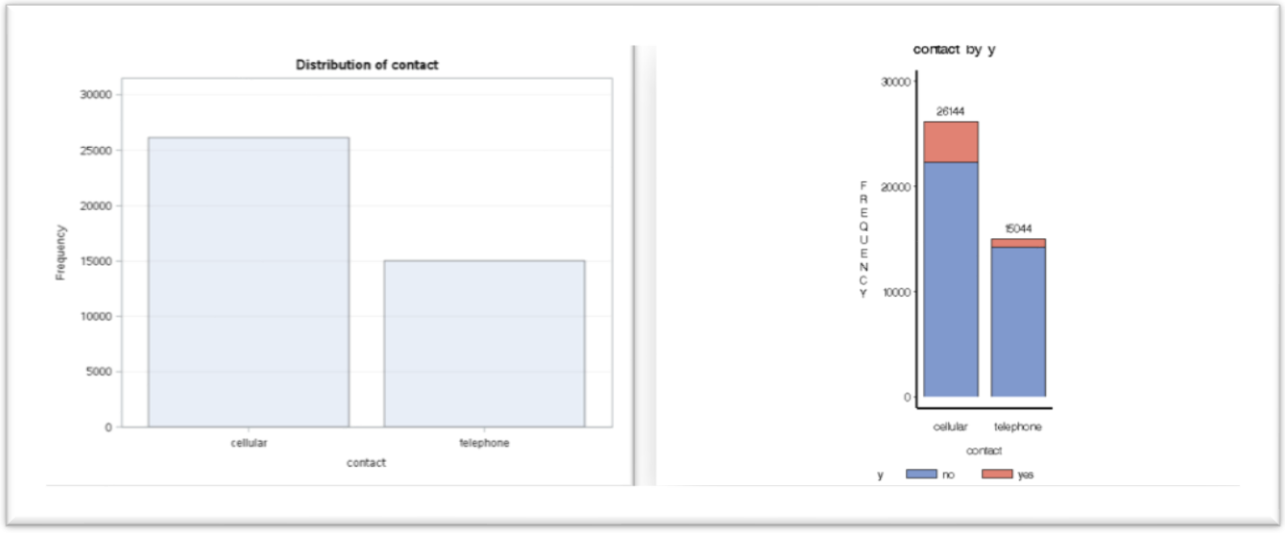


Figure 6: **Distribution of day_of_week** & **day_of_week by 'y'**

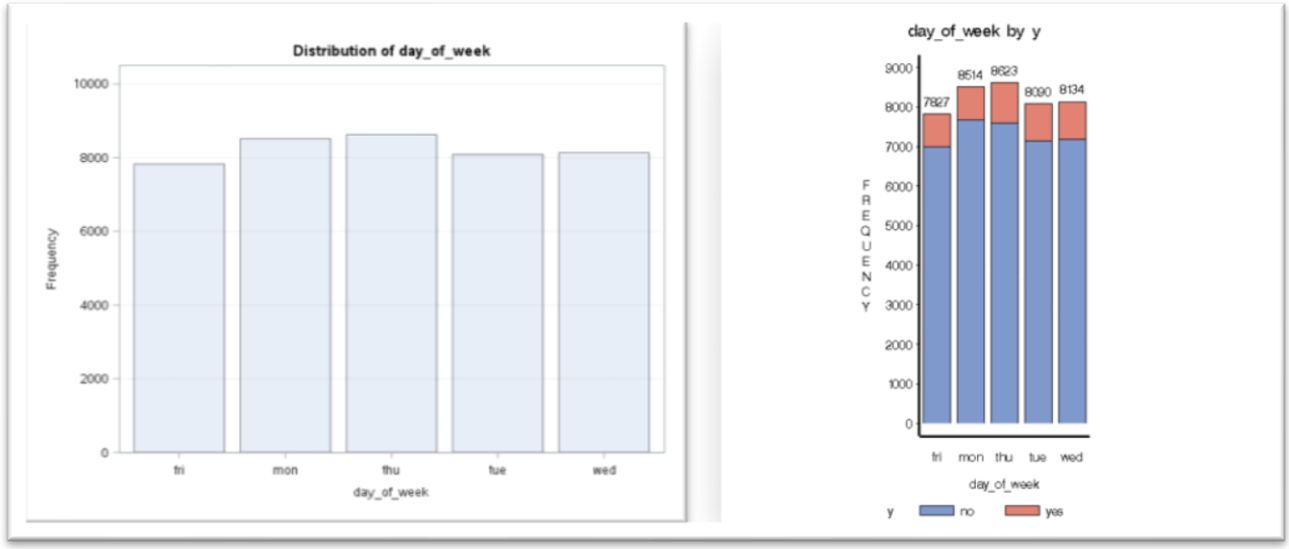


Figure 7: **Distribution of default** & **default by 'y'**

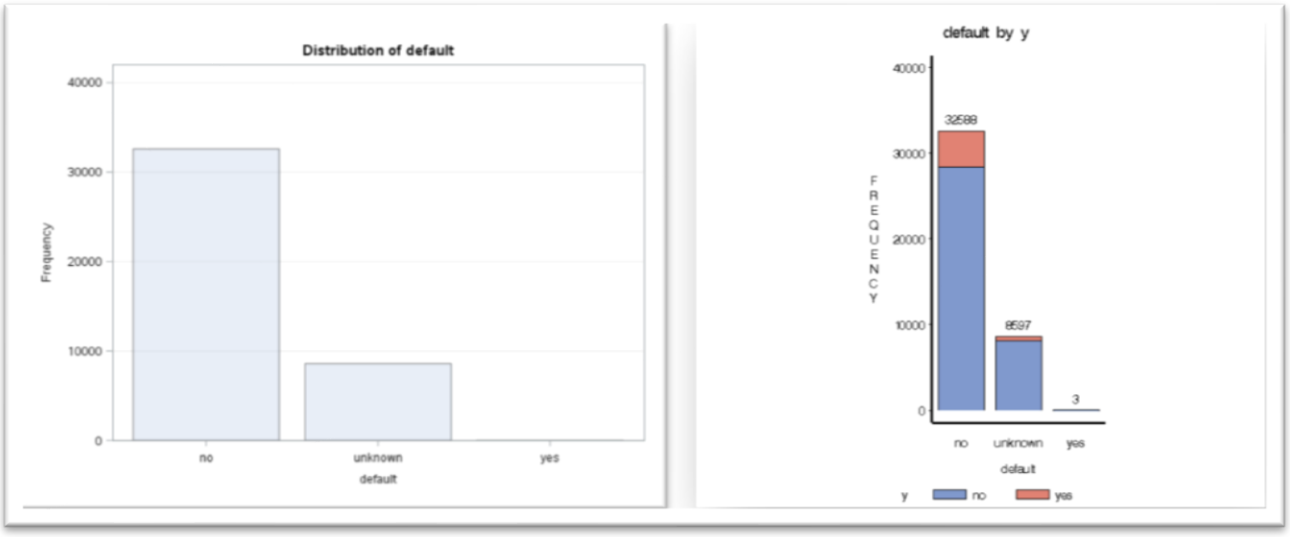


Figure 8: **Distribution of education** & **education by 'y'**

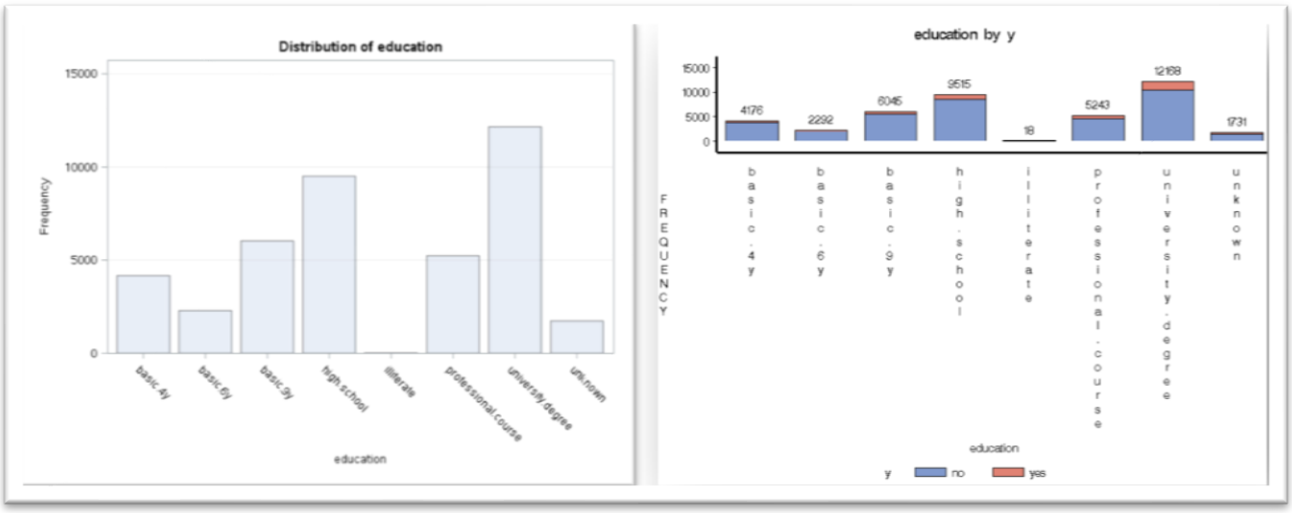


Figure 9: **Distribution of emp.var.rate** & **emp.var.rate by 'y'**

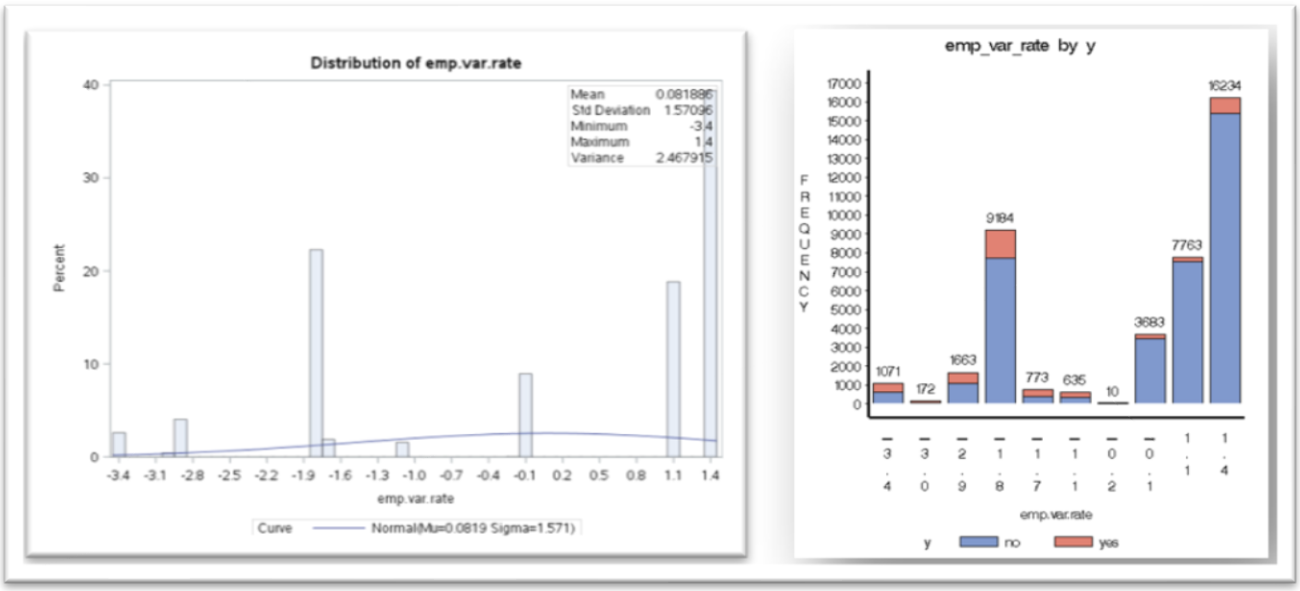


Figure 10: **Distribution of euribor3m** & **euribor3m by 'y'**

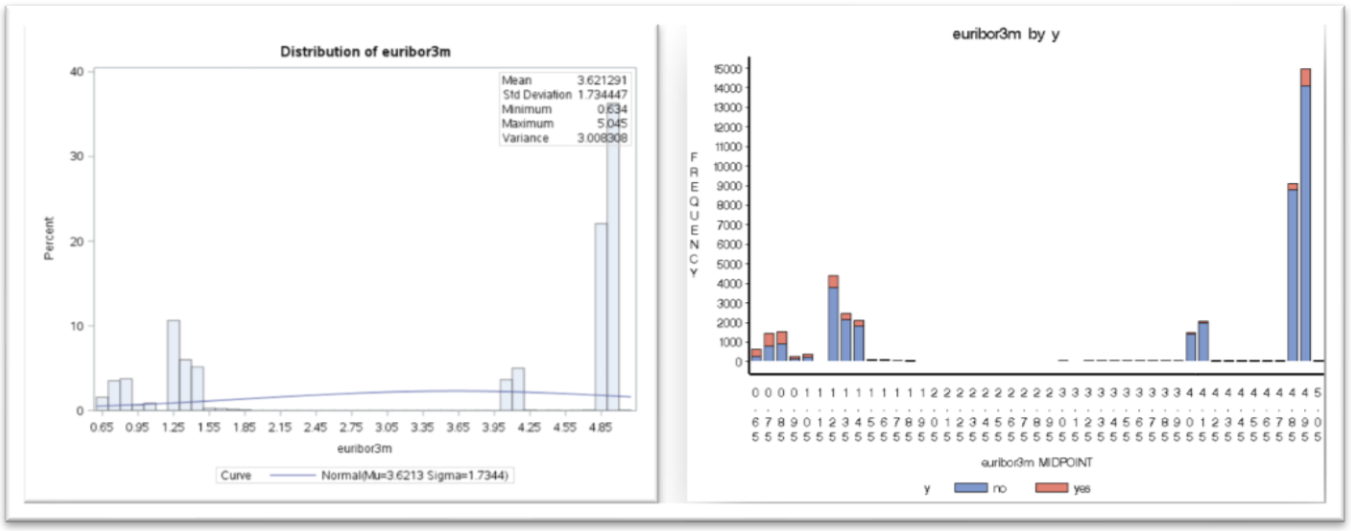


Figure 11: **Distribution of housing** & **housing by 'y'**

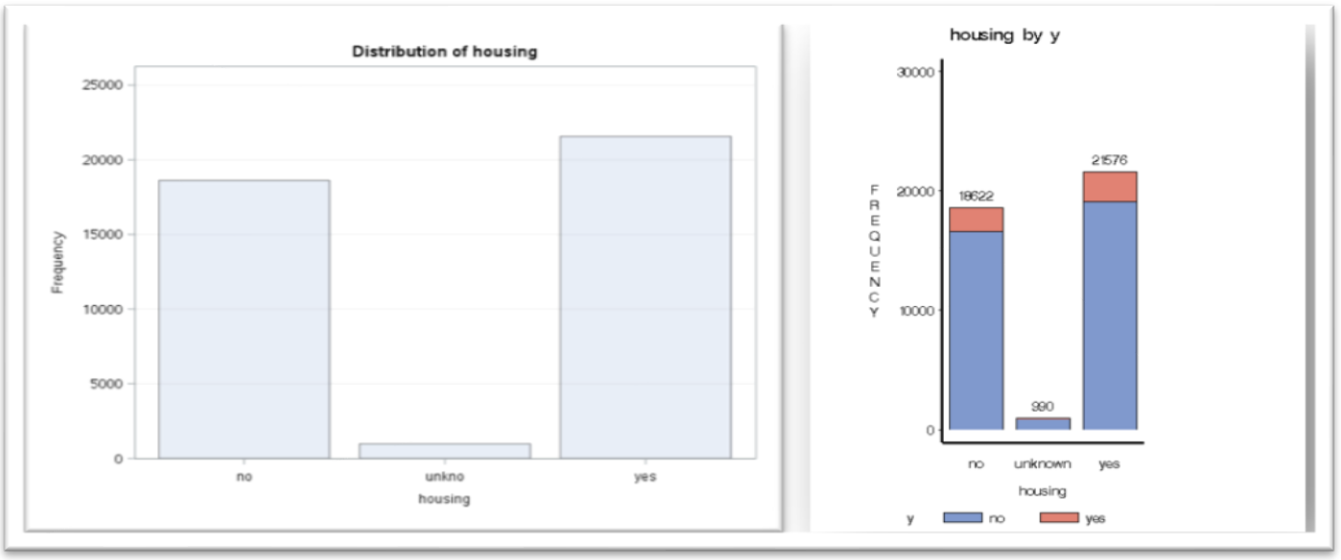


Figure 12: **Distribution of job** & **job by 'y'**

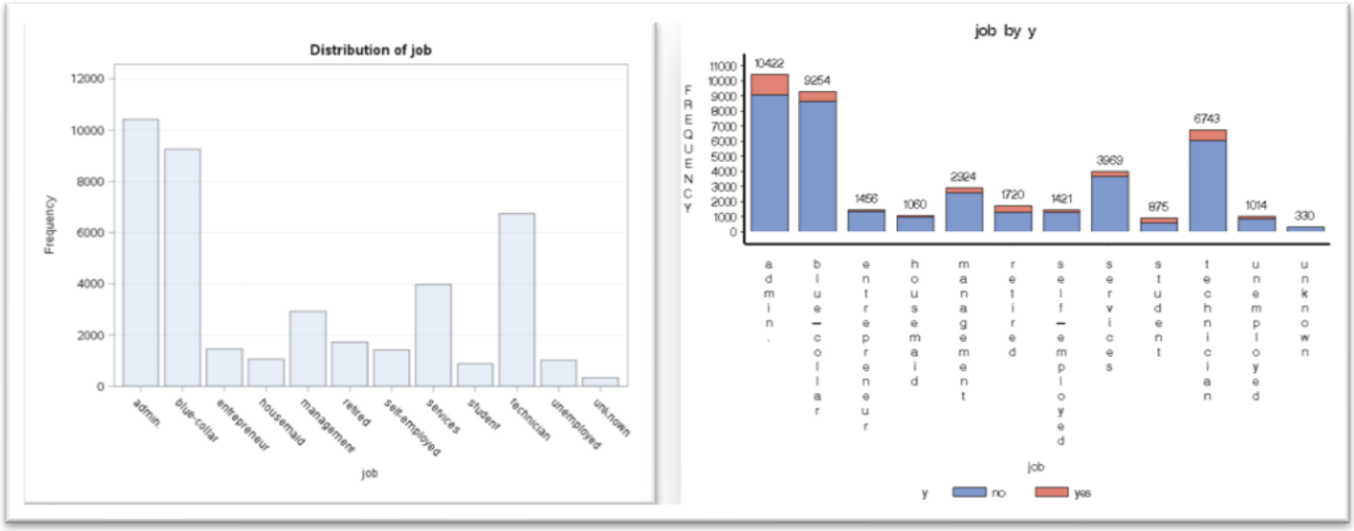


Figure 13: **Distribution of loan** & **loan by 'y'**

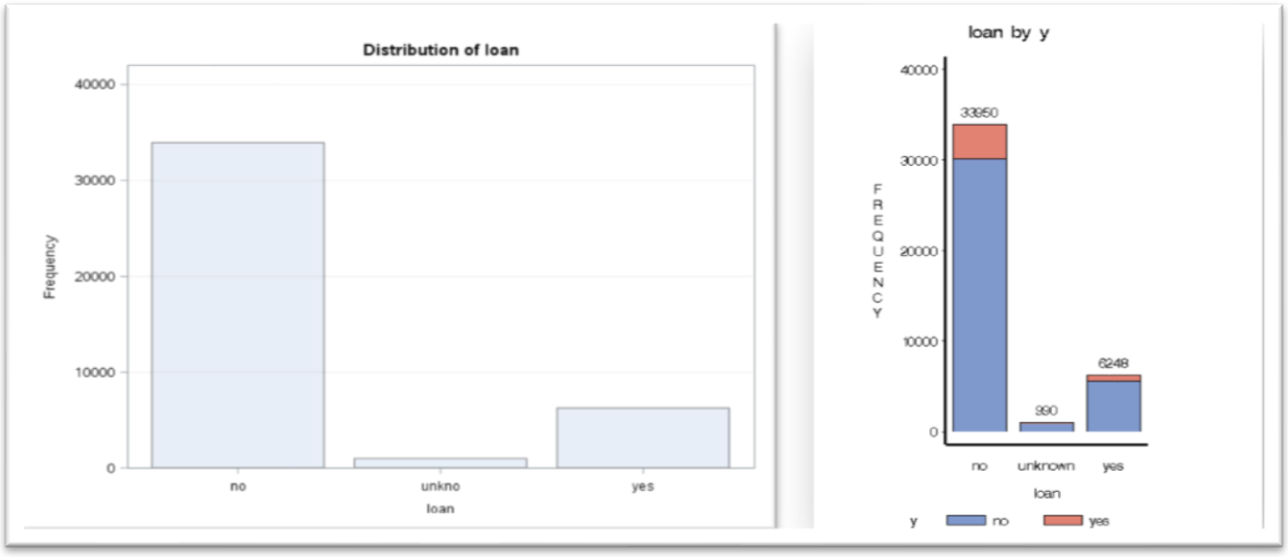


Figure 14: **Distribution of month** & **month by 'y'**

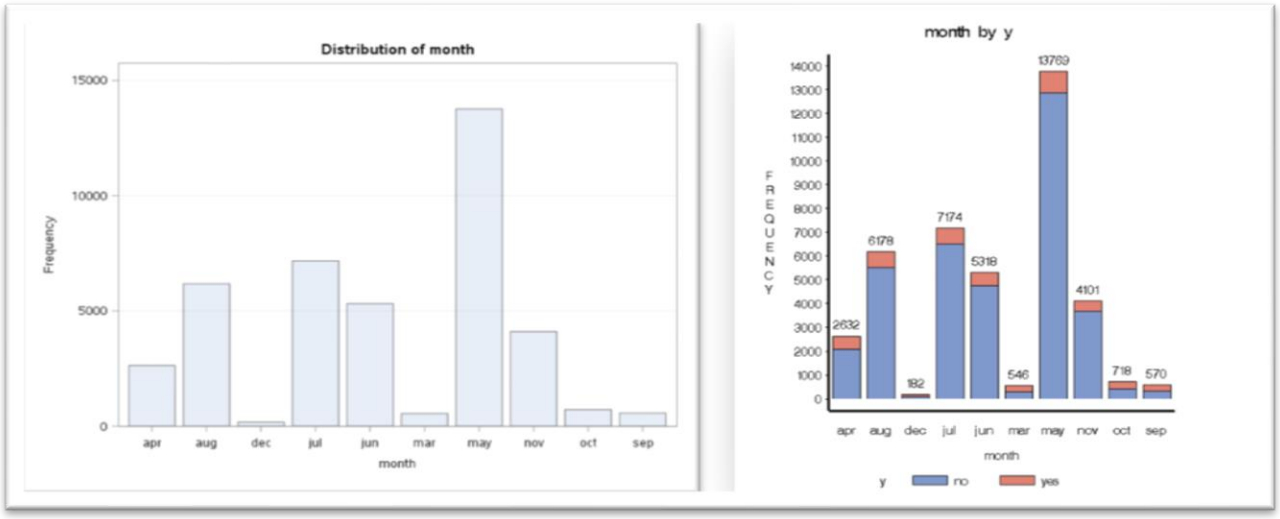


Figure 15: **Distribution of marital** & **marital by 'y'**

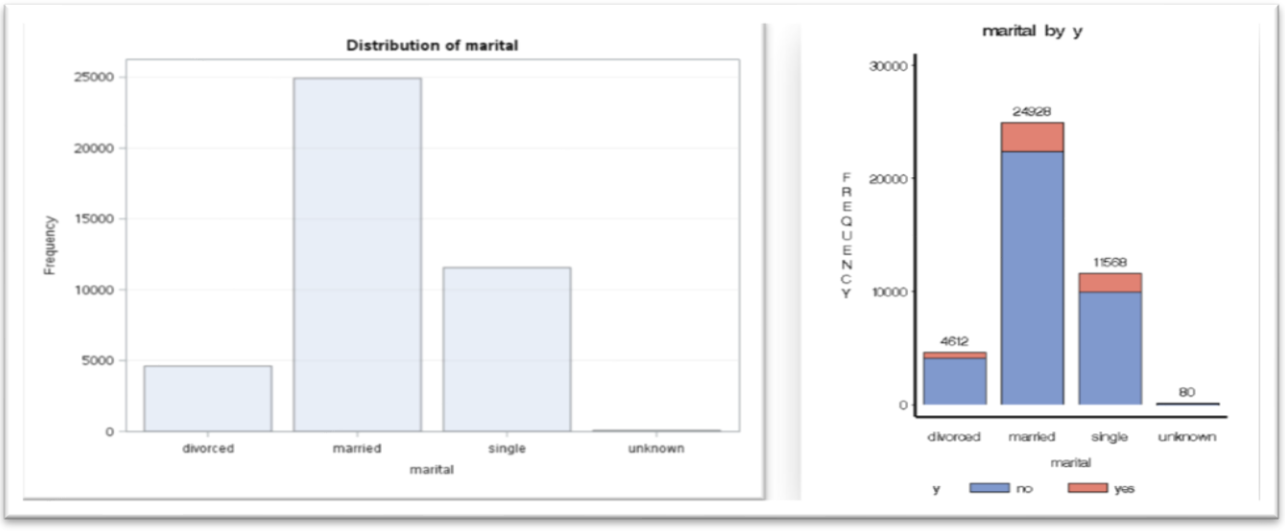


Figure 16: **Distribution of nr.employed** & **nr.employed by 'y'**

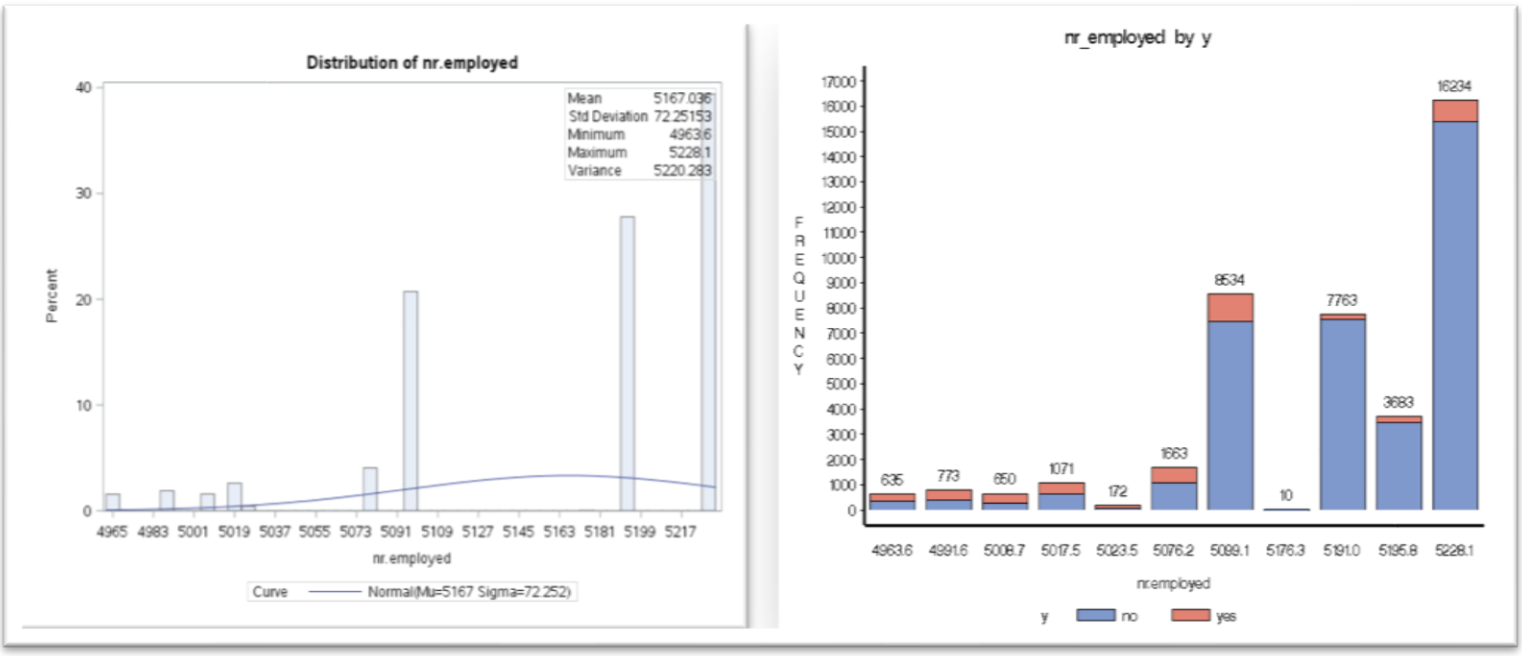


Figure 17: **Distribution of pdays** & **pdays by 'y'**

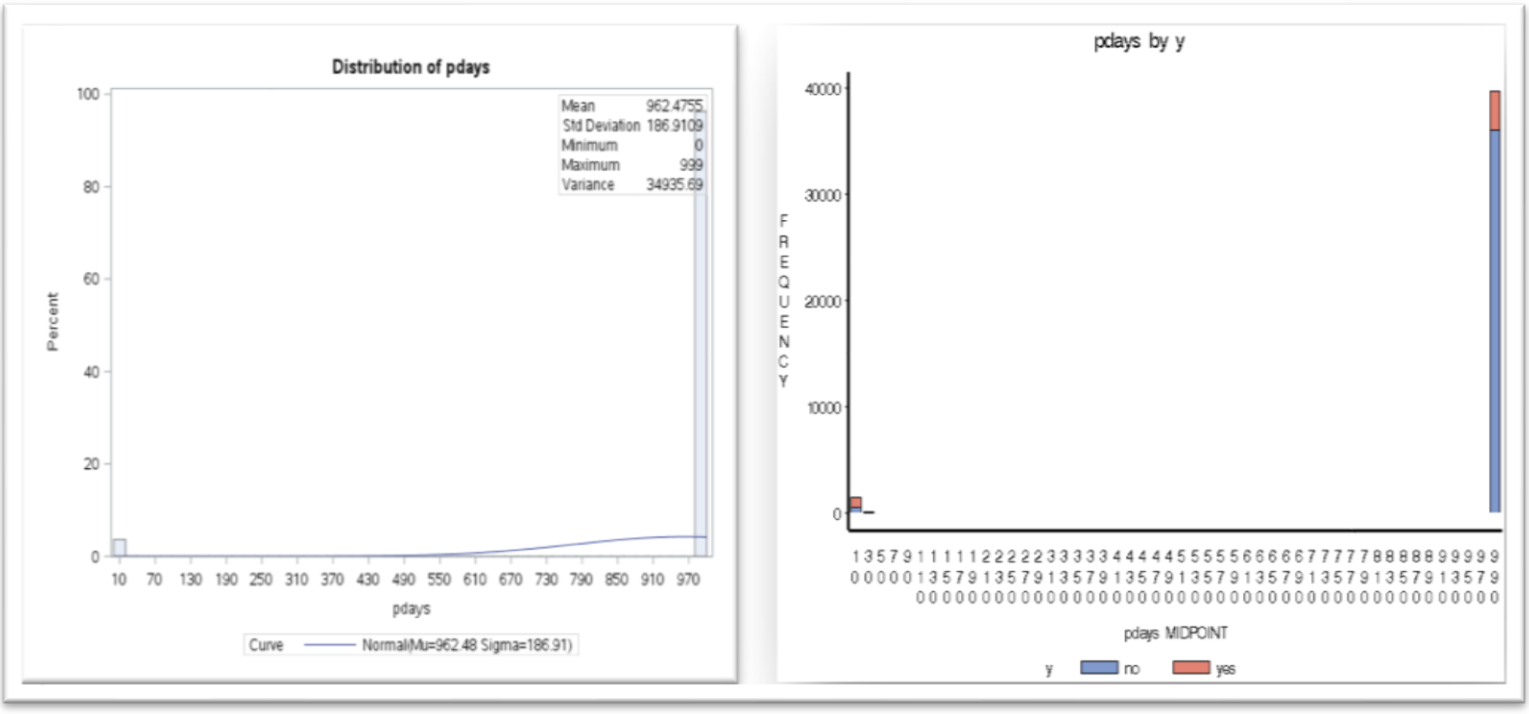


Figure 18: **Distribution of poutcome** & **poutcome by 'y'**

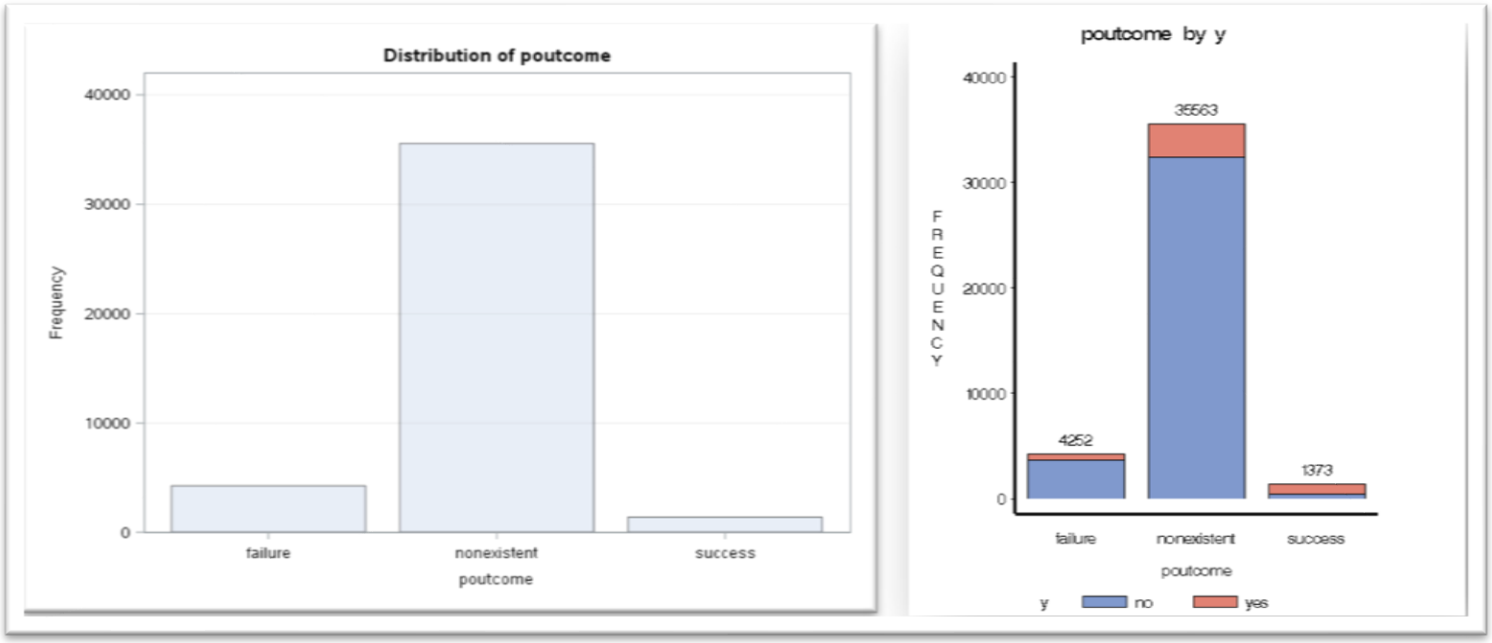


Figure 19: **Distribution of previous** & **previous by 'y'**

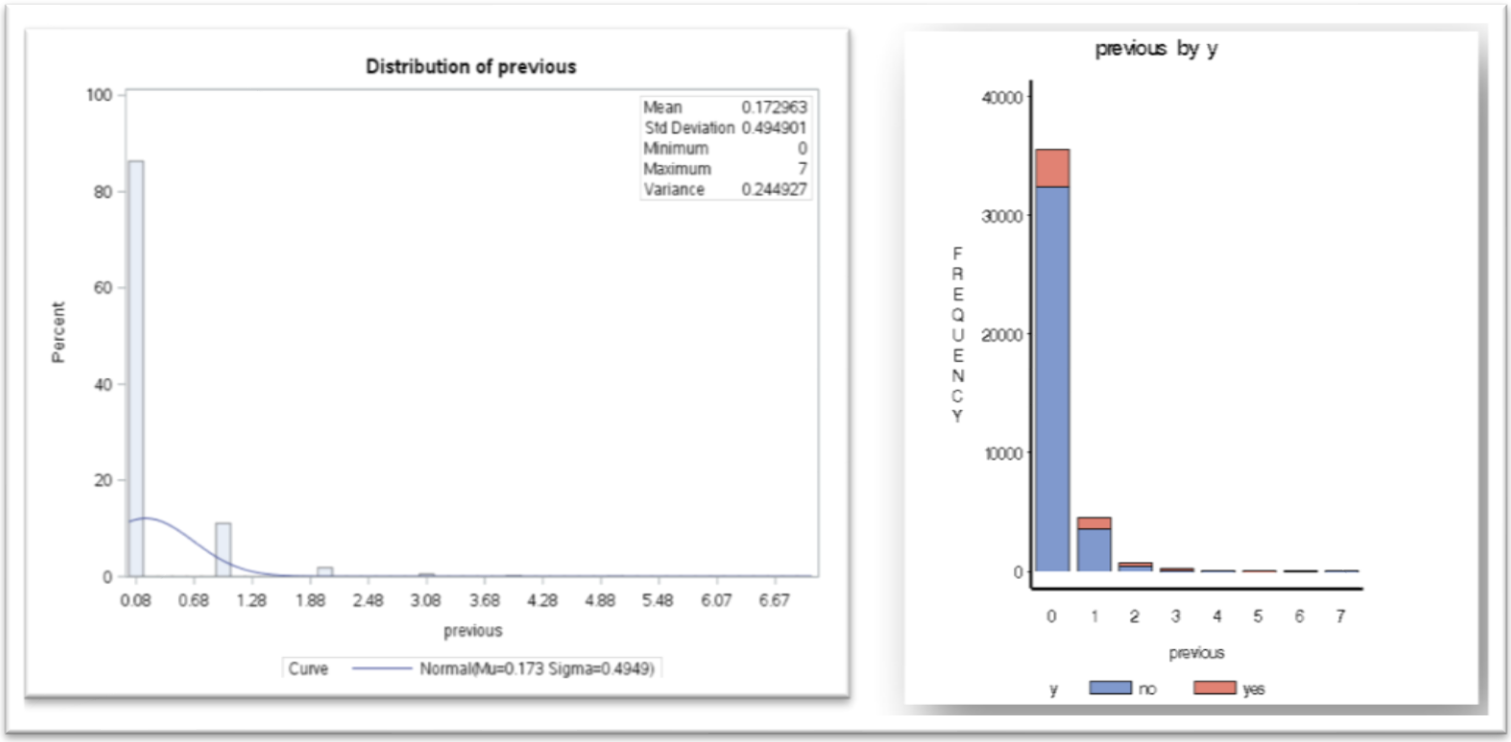


Figure 20: **Distribution of Target variable ‘y’**

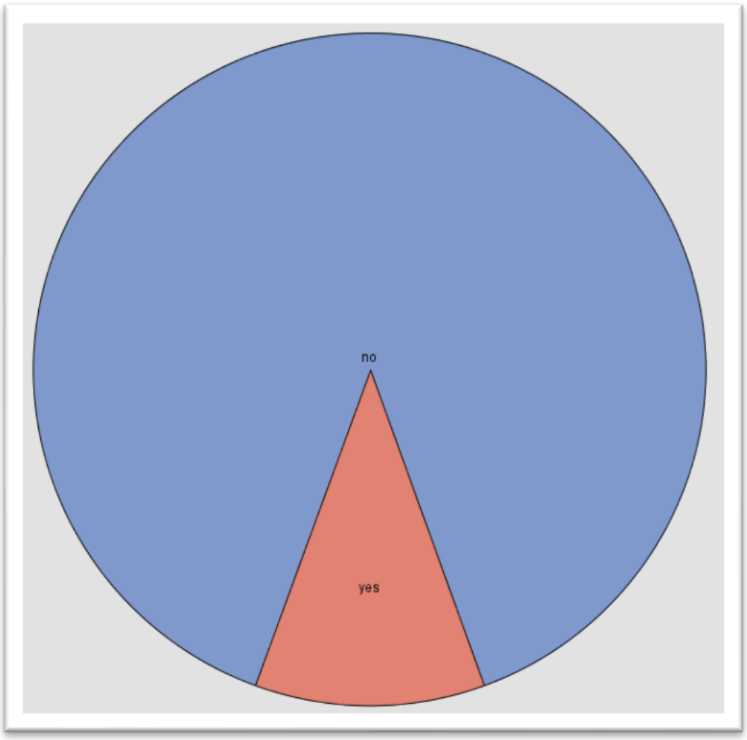


Figure 21: **WORKFLOW IN SAS ENTERPRISE MINER**