Data Mining Assignment

Performance comparison of prediction models on Portuguese Bank dataset

Harsh Vardhan Rai

Mina Jamshidian

Table of Content

Problem Definition

The data which was analysed comes from a Portugal based bank which did a direct marketing campaign to sell a term deposit product by making phone calls. The data has been generated based on the campaign targeting the potential individuals.

A term deposit is a cash investment to a financial institution for an agreed interest rate over a fixed time period. The goal of this analysis is determining whether a customer will subscribe or not and to compare some classification models performance in SAS Enterprise Miner.

Data Description

This dataset has 21 variables and 41188 records that we found that it contains 9 numerical variables and 11 categorical variables. The y variable would be used as the target variable for the models that we are attempting to predict. The variables of dataset were divided to five parts:

- Bank client data: 1- age, 2- job, 3- marital, 4- education, 5- default, 6- housing, 7- loan.
- Depend on the last contact of the current campaign: 8- contact, 9- month, 10- day_of_week, 11- duration.
- Other attributes: 12- campaign,13- pdays, 14- previous, 15- poutcome.
- Social and Economic Context Attributes: 16- emp.var.rate, 17- cons.price.idx, 18- cons.conf.idx, 19- euribor3m, 20- nr.employed.
- Target Variable: 21- y

Link to dataset: http://archive.ics.uci.edu/ml/datasets/Bank+Marketing

Data Input, Exploration & Analysis

For this part, R programming was used. The dataset is in the CSV format. So, for reading this format and first analysing the below code was used:

For more exploration about all the variables, the *CrossTable()* function was used that its output was similar to S-Plus crosstabs() and SAS Proc Freq (or SPSS format) with Chi-square, Fisher and McNemar tests of the independence of all table factors with the target variable(y). Here we will look only via Chi-square tests.

```
clean_bank_add_full = read.csv("bank-additional-full.csv",sep = ";")
```

Missing & Unknown values

- After checking the missing value and “unknown” value as you can see in Figure 1, we found there wasn't any “NA” value. But some variables have a lot of “unknown” value. So, we have sorted out the column names with their sum of unknown.

```
colSums(is.na(clean_bank_add_full))
colSums(clean_bank_add_full == "unknown")
```

```
colSums(is.na(clean_bank_add_full))
      age           job       marital     education       default       housing          loan       contact         month   day_of_week      duration      campaign         pdays      previous
        0             0             0             0             0             0             0             0             0             0             0             0             0             0
 poutcome  emp.var.rate cons.price.idx  cons.conf.idx     euribor3m   nr.employed             y
        0             0             0             0             0             0             0
colSums(clean_bank_add_full == "unknown")
      age           job       marital     education       default       housing          loan       contact         month   day_of_week      duration      campaign         pdays      previous
        0           330            80          1731          8597           990           990             0             0             0             0             0             0             0
 poutcome  emp.var.rate cons.price.idx  cons.conf.idx     euribor3m   nr.employed             y
        0             0             0             0             0             0             0
```

Figure 1

Wrangling & Exploration

- It has been found that the value of the target variable which is "y" is yes and no, so it was decided to convert the values to 0,1 by adding a new column to the dataset with the name of "y_new".

```
clean_bank_add_full = clean_bank_add_full %>% mutate(y_new= ifelse(y== "no",0,1))
```

- Cleaning the unknown values for 5 variables (job, marital, education, housing, default) based on their correlation with the target variable (dependent variable) which is "y_new".

    1. For job, we found that the response was highest among students around 31% while other groups were between 6-15%. Also, we could easily ignore the unknown values here.

    2. For marital, singles were making the highest percentage while other were more prone to say no. We can ignore the unknown values here.

    3. For education, we can easily see that as the level of education increases people are more likely to subscribe to the term deposit. Also, the number of people opting for term deposit in the unknown are more so we can't ignore these values, instead through CrossTable we found that the most proportionate data of unknown was resembling the university level. So, we have replaced the unknown with university holders.

    4. For housing, there were no proper variations and the chi-square proportions suggests that we can ignore these values.

    5. For default, the proportions were quite abnormal. People opting for term deposit were just 3 while 32000 people opted for no. So, it'll be better to remove this column.

```
CrossTable(clean_bank_add_full$job,clean_bank_add_full$y_new)
clean_bank_add_full = clean_bank_add_full %>% filter( job!= "unknown")

CrossTable(clean_bank_add_full$marital,clean_bank_add_full$y_new)
clean_bank_add_full = clean_bank_add_full %>% filter( marital!= "unknown")

CrossTable(clean_bank_add_full$education,clean_bank_add_full$y_new)
clean_bank_add_full$education[clean_bank_add_full$education=="unknown"]="university.degree"

CrossTable(clean_bank_add_full$housing,clean_bank_add_full$y_new)
clean_bank_add_full = clean_bank_add_full %>% filter( housing!= "unknown")
```

Figure 2

- We found the value of the “age” variable would be better if divided into several groups. So, we have divided it into (1- age > 60,"senior-citizen, 2- age >45,"mid-old" 3- age>30,"Mid-age" 4- age>15,"Young" 5- age<15 "Children"). Then add it as a new column with the name of “age_label” to the dataset.

```
CrossTable(clean_bank_add_full$age,clean_bank_add_full$y_new)

clean_bank_add_full= clean_bank_add_full%>% mutate(age_label= if_else(age > 60,"senior-citizen",if_else(age
>45,"mid-old",if_else(age>30,"Mid-age",if_else(age>15,"Young","Children")))))
```

- The column campaign gives us the value of how many times has the individual being contacted. It was surprising that they were being called 20-30 times but still there were making no changes to the proportion of people saying yes to the term deposit. So, the value of the campaign variable should be filtered by the less than 10.

```
CrossTable(clean_bank_add_full$campaign,clean_bank_add_full$y_new)
clean_bank_add_full= clean_bank_add_full%>%
  filter(campaign<10)
```

- It was found a lot of values of the “pday” variable is 999, this number shows the number of days that passed by after the client was last contacted from a previous campaign that means client was not previously contacted so this value was replaced by -1 value to increase the normality. Then it was added to a new column with the name of “pdays_new”.

```
CrossTable(clean_bank_add_full$pdays,clean_bank_add_full$y_new)
clean_bank_add_full=clean_bank_add_full%>%
  mutate(pdays_new=ifelse(pdays==999,"0","1"))
```

- Analysing and plotting other variables by CrossTable() as shown in Figure 3.

```
CrossTable(clean_bank_add_full$age_label,clean_bank_add_full$y_new)
CrossTable(clean_bank_add_full$contact,clean_bank_add_full$y_new)
CrossTable(clean_bank_add_full$month,clean_bank_add_full$y_new)
CrossTable(clean_bank_add_full$day_of_week,clean_bank_add_full$y_new)
CrossTable(clean_bank_add_full$duration,clean_bank_add_full$y_new)
CrossTable(clean_bank_add_full$previous,clean_bank_add_full$y_new)
CrossTable(clean_bank_add_full$poutcome,clean_bank_add_full$y_new)
CrossTable(clean_bank_add_full$y_new)
```

```
g1<-ggplot(clean_bank_add_full,aes(x=contact,fill=y))+geom_bar()
g2<-ggplot(clean_bank_add_full,aes(x=month,fill=y))+geom_bar()
g3<-ggplot(clean_bank_add_full,aes(x=day_of_week,fill=y))+geom_bar()
g5<-ggplot(clean_bank_add_full,aes(x=duration,fill=y))+geom_bar()
g6<-ggplot(clean_bank_add_full,aes(x=campaign,fill=y))+geom_bar()
g7<-ggplot(clean_bank_add_full,aes(x=campaign,fill=y))+geom_bar()
g8<-ggplot(clean_bank_add_full,aes(x=previous,fill=y))+geom_bar()
g9<-ggplot(clean_bank_add_full,aes(x=poutcome,fill=y))+geom_bar()

# Checking the Distribution of variables by bar plot
grid.arrange(g1,g2,g3,g5,g6,g7,g8,g9 ,nrow=4)
```

```
# Checking the outliers by Box plot
g0<-ggplot(clean_bank_add_full)+aes(x = "", y = age)+geom_boxplot(fill = "red")
g4<-ggplot(clean_bank_add_full)+aes( y = duration)+geom_boxplot(fill = "red")
grid.arrange(g0,g4,ncol=2)
```

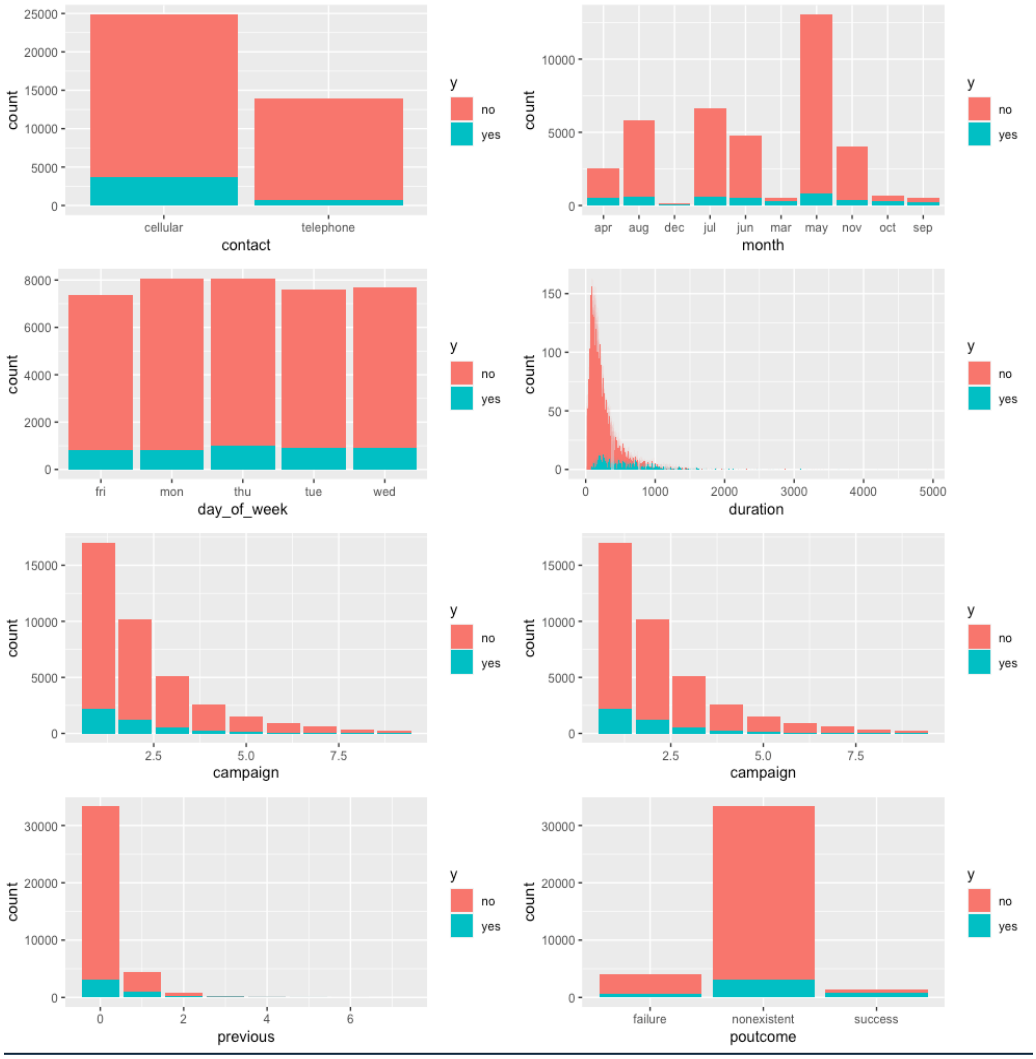


Figure 3

- For socio-economic attributes to check the correlation we have plotted a correlation matrix as shown in Figure 4. Three variables showed a correlation greater than 0.9.

```
cor_matrix <- clean_bank_add_full[, c(16,17,18,19,20)]
chart.Correlation(cor_matrix, histogram=TRUE, pch=19)
```

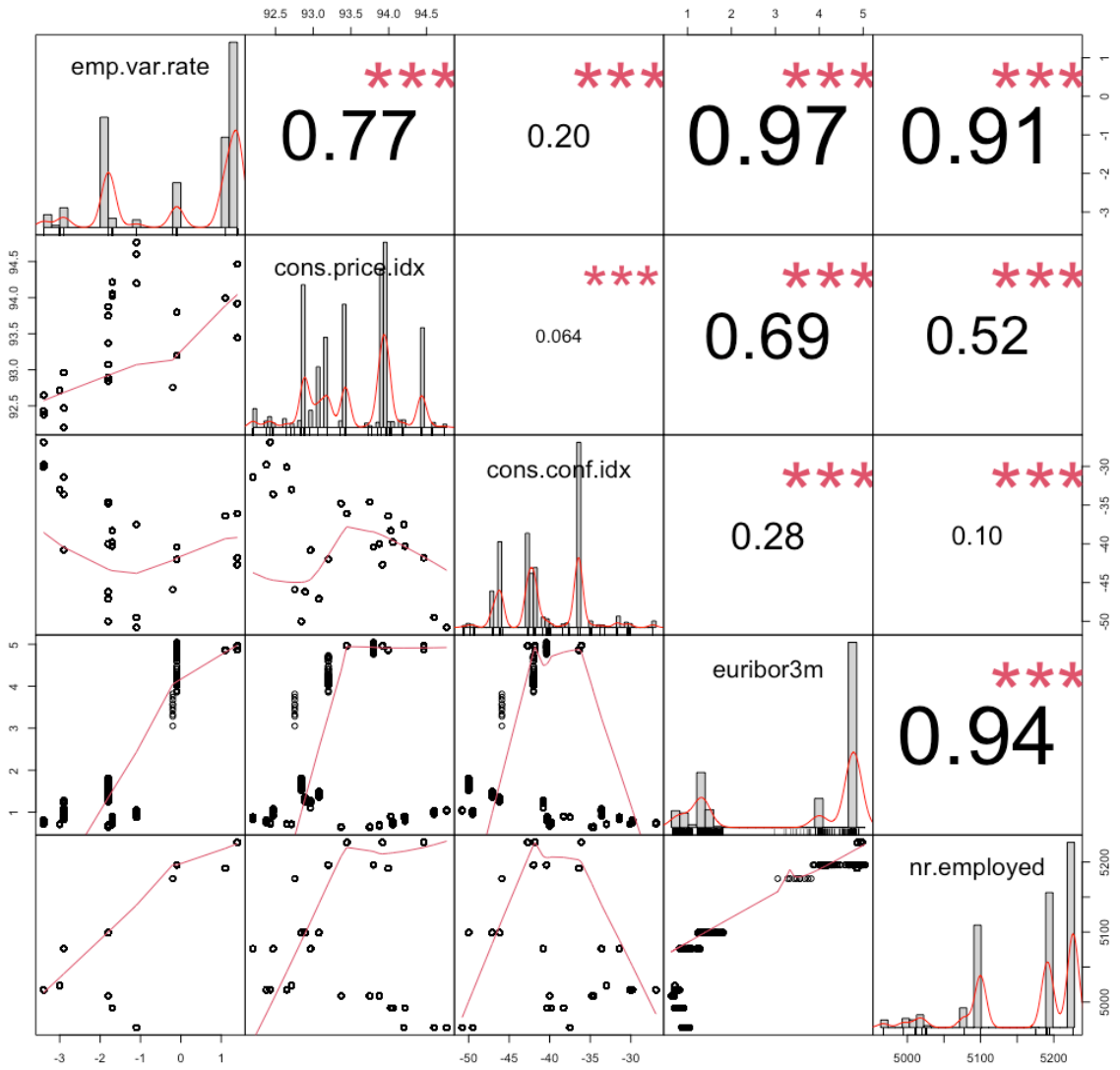


Figure 4

- Removing column default as per our analysis.

```
clean_bank_add_full = clean_bank_add_full[-c(5)]
```

- The nr.employed variable, shows the number of employees in the bank. It could not be a decimal therefore this variable rounded by the round function and added as a new variable to the dataset with the name “nr.employed_new” and the nr.employed was removed from the dataset.

```
clean_bank_add_full$nr.employed_new = round(clean_bank_add_full$nr.employed)
clean_bank_add_full = clean_bank_add_full[-c(19)]

# Removing pday variable
clean_bank_add_full = clean_bank_add_full[-c(12)]
```

Summary Statistics

- Summarizing the categorical variables by *HMISC:describe()*. Some of the variables are shown in the figure 5.

```
detach("package:psych", unload = TRUE)

des_categoriaval <- clean_bank_add_full  %>%
select(contact,job,housing,day_of_week,education,loan,marital,month,poutcome,y_new)
describe(des_categoriaval)
```

Figure 5

```
> describe(clean_bank_add_full$education)
clean_bank_add_full$education
       n  missing distinct
   39803        0        7

lowest : basic.4y            basic.6y            basic.9y            high.school         illiterate
highest: basic.9y            high.school         illiterate          professional.course university.degree

Value                basic.4y            basic.6y            basic.9y         high.school          illiterate
Frequency                4002                2204                5856                9244                  18
Proportion              0.101               0.055               0.147               0.232               0.000

Value      professional.course   university.degree
Frequency                 5100               13379
Proportion               0.128               0.336
> describe(clean_bank_add_full$loan)
clean_bank_add_full$loan
       n  missing distinct
   39803        0        2

Value         no   yes
Frequency  33620  6183
Proportion 0.845 0.155
> describe(clean_bank_add_full$marital)
clean_bank_add_full$marital
       n  missing distinct
   39803        0        3

Value      divorced  married   single
Frequency      4478    24110    11215
Proportion    0.113    0.606    0.282
```

- Summarizing the numerical variables *PSYCH:describe()*

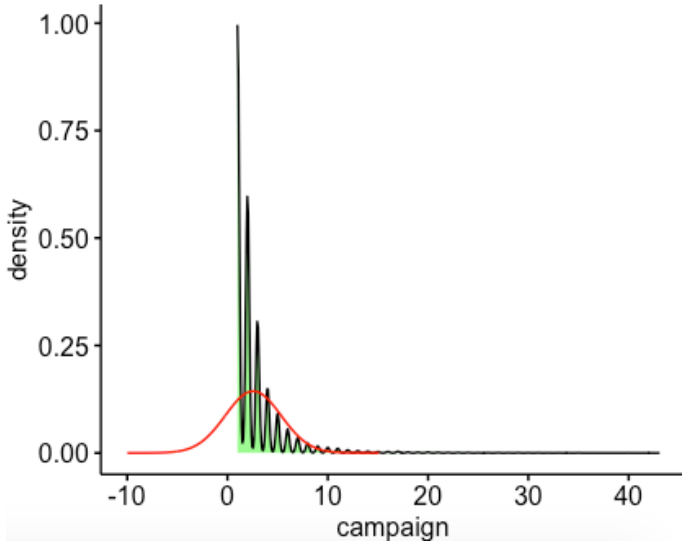
```
detach("package:Hmisc", unload = TRUE)
des_numerical <- clean_bank_add_full  %>%
select(age,campaign,pdays_new,previous,cons.price.idx,cons.conf.idx,euribor3m,nr.employed)
describe(des_numerical)
```

```
> describe(clean_bank_add_full$age)
   vars     n  mean    sd median trimmed   mad min max range skew kurtosis   se
X1    1 39803 39.98 10.4     38   39.25 10.38  17  98    81 0.79     0.82 0.05
> describe(clean_bank_add_full$campaign)
   vars     n mean   sd median trimmed  mad min max range skew kurtosis   se
X1    1 39803 2.57 2.76      2    1.99 1.48   1  43    42 4.69    34.85 0.01
> describe(clean_bank_add_full$previous)
   vars     n mean   sd median trimmed mad min max range skew kurtosis se
X1    1 39803 0.17 0.49      0    0.05   0   0   7     7 3.84    20.31  0
> describe(clean_bank_add_full$cons.price.idx)
   vars     n  mean   sd median trimmed  mad  min   max range  skew kurtosis se
X1    1 39803 93.57 0.58  93.75   93.58 0.56 92.2 94.77  2.57 -0.22    -0.83  0
> describe(clean_bank_add_full$cons.conf.idx)
   vars     n   mean   sd median trimmed  mad   min   max range skew kurtosis   se
X1    1 39803 -40.52 4.63  -41.8  -40.62 6.52 -50.8 -26.9  23.9 0.31    -0.36 0.02
> describe(clean_bank_add_full$euribor3m)
   vars     n mean   sd median trimmed  mad  min  max range  skew kurtosis   se
X1    1 39803 3.62 1.73   4.86    3.8 0.16 0.63 5.04  4.41 -0.71    -1.41 0.01
> describe(clean_bank_add_full$nr.employed)
   vars     n    mean   sd median trimmed mad    min    max range  skew kurtosis   se
X1    1 39803 5167.03 72.2   5191 5178.39  55 4963.6 5228.1 264.5 -1.04    -0.01 0.36
```

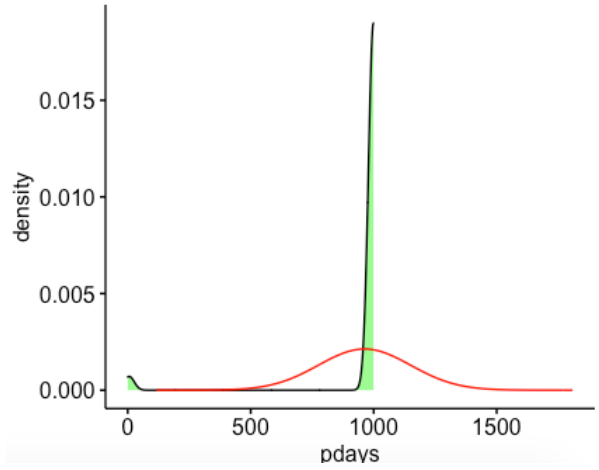
Density curves for introducing normality

- Checking the density functions and the normal distribution of some variables as they had skewness more than the normal value as described during summarising the numerical variables.

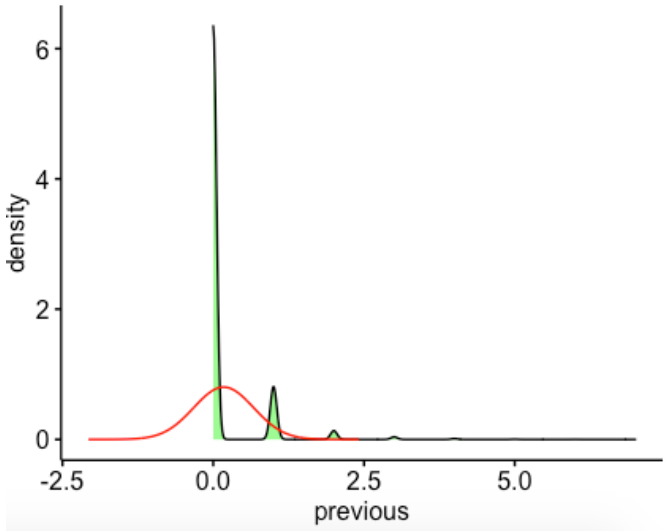
```
#PLOT1 :variable : "Camapign"
ggdensity(clean_bank_add_full,x= "campaign",fill = "green")+
  stat_overlay_normal_density(color = "red")
```




```
#PLOT2 :variable : "pdays"
ggdensity(clean_bank_add_full,x= "pdays",fill = "green")+
  stat_overlay_normal_density(color = "red")
```




```
#PLOT3 :variable : "previous"
ggdensity(clean_bank_add_full,x= "previous",fill = "green")+
  stat_overlay_normal_density(color = "red")
```



Data Balancing through ROSE (Random Over Sampling)

- When we summarized our target variable, we found that there was a major skewness towards the target variable for "No". So, data balancing needs to be performed for removing this skewness. We over sampled our data with respect to the no values in the target variable.

```
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  39803

          |         0 |         1 |
          |-----------|-----------|
          |     35316 |      4487 |
          |     0.887 |     0.113 |
          |-----------|-----------|
```

```
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  68602

          |         0 |         1 |
          |-----------|-----------|
          |     35316 |     33286 |
          |     0.515 |     0.485 |
          |-----------|-----------|
```

```
data_bal=ovun.sample(y_new ~ ., data = clean_bank_add_full, method = "over",N = 68602)$data
data_bal = data_bal[-c(20)]
```

Exporting the final data frame

- At last when all the analysis was carried out and some of the nonperforming variables were removed, the data was written onto a .CSV file which will serve as input to our SAS.

```
write_csv(data_bal,"clean_data_final.csv")
```

SAS Enterprise Miner (Models)

Importing the cleaned dataset from R we initially found the variable summary by Stat Explore node. It'll give all the necessary information about the dataset.
So, after cleaning we found that we have 10 interval variables, 10 nominal variables and 1 binary target variable as shown in the figure 6.

Variable Summary

Role	Measurement Level	Frequency Count
INPUT	INTERVAL	10
INPUT	NOMINAL	10
TARGET	BINARY	1

Variable Levels Summary
(maximum 500 observations printed)

Variable	Role	Frequency Count
y_new	TARGET	2

Figure 6

Distribution of the target variable can be found in the below figure 7. As per oversampling done for our target variable, we are now having proportionate dataset to build the models.

Distribution of Class Target and Segment Variables
(maximum 500 observations printed)

Data Role=TRAIN

Data Role	Variable Name	Role	Level	Frequency Count	Percent
TRAIN	y_new	TARGET	0	34301	50
TRAIN	y_new	TARGET	1	34301	50

Figure 7

Below is the summary for all the class variables in figure 8 and for interval variables in figure 9.

Class Variable Summary Statistics
(maximum 500 observations printed)

Data Role=TRAIN

Data Role	Variable Name	Role	Number of Levels	Missing	Mode	Mode Percentage	Mode2	Mode2 Percentage
TRAIN	age_label	INPUT	4	0	Mid-age	49.86	mid-old	24.45
TRAIN	contact	INPUT	2	0	cellular	72.56	telephone	27.44
TRAIN	day_of_week	INPUT	5	0	thu	21.51	wed	20.15
TRAIN	education	INPUT	7	0	university.degree	36.98	high.school	22.77
TRAIN	housing	INPUT	2	0	yes	54.14	no	45.86
TRAIN	job	INPUT	11	0	admin.	27.08	blue-collar	18.72
TRAIN	loan	INPUT	2	0	no	84.67	yes	15.33
TRAIN	marital	INPUT	3	0	married	58.03	single	31.07
TRAIN	month	INPUT	10	0	may	27.42	jul	15.72
TRAIN	y_new	TARGET	2	0	0	50.00	1	50.00

Figure 8

Interval Variable Summary Statistics
(maximum 500 observations printed)

Data Role=TRAIN

Variable	Role	Mean	Standard Deviation	Non Missing	Missing	Minimum	Median	Maximum	Skewness	Kurtosis
age	INPUT	40.33465	12.03883	68602	0	17	38	98	0.997895	1.139506
campaign	INPUT	2.123335	1.523677	68602	0	1	2	9	1.882231	3.908776
cons_conf_idx	INPUT	-40.2053	5.38469	68602	0	-50.8	-41.8	-26.9	0.341557	-0.46462
cons_price_idx	INPUT	93.46896	0.632497	68602	0	92.201	93.444	94.767	-0.10006	-0.96884
duration	INPUT	388.2213	359.1051	68602	0	0	267	4918	2.277534	9.139397
emp_var_rate	INPUT	-0.51625	1.719806	68602	0	-3.4	-1.1	1.4	-0.12908	-1.54943
euribor3m	INPUT	2.943465	1.887345	68602	0	0.634	1.811	5.045	-0.00049	-1.92099
nr_employed_new	INPUT	5134.829	86.51054	68602	0	4964	5099	5228	-0.40869	-1.20566
pdays_new	INPUT	0.112285	0.31572	68602	0	0	0	1	2.456142	4.03275
previous	INPUT	0.31314	0.697482	68602	0	0	0	7	2.901476	10.59434

Figure 9

Data Partition

Data is typically split into separate chunks for preparation, validating and checking classifiers to improve the classification efficiency of models. Since the test partition is primarily used to measure fit statistics after modelling and model selection is completed, it is regarded as wasting data by sub-setting this way.

Also, we can boost the model stability by increasing the observations to some degree in train results. Based on these details, for this analysis, we have partitioned the data (Figure 11) into trains and validated chunks in the 75:25 ratio in SAS Enterprise Miner. The setting for this partition can be seen in the (Figure 10). This makes it possible to include 51452 observations for Train and 17150 observations for validation sets.

Train	
Variables	
Output Type	Data
Partitioning Method	Default
Random Seed	12345
Data Set Allocations	
Training	75.0
Validation	25.0
Test	0.0
Report	
Interval Targets	Yes
Class Targets	Yes

Figure 10

Partition Summary

Type	Data Set	Number of Observations
DATA	EMWS3.Stat_TRAIN	68602
TRAIN	EMWS3.Part_TRAIN	51452
VALIDATE	EMWS3.Part_VALIDATE	17150

Figure 11

Data Mining Models & Configuration Settings

1. Decision Tree

At first, we are generating a Maximal Decision Tree by training the node to automatically break and generate a tree. Secondly, we are trying to create an optimized tree with some configuration settings.

In order to check the output of the created tree, a Subtree Assessment plot was analysed for the parameter Misclassification Rate. The number of leaf nodes produced for the Maximal Tree has been observed to be 60. For the train, the misclassification rate curve is found to diverge and validate data chunks, indicating a model output for the Maximal Decision Tree (Figure 12).

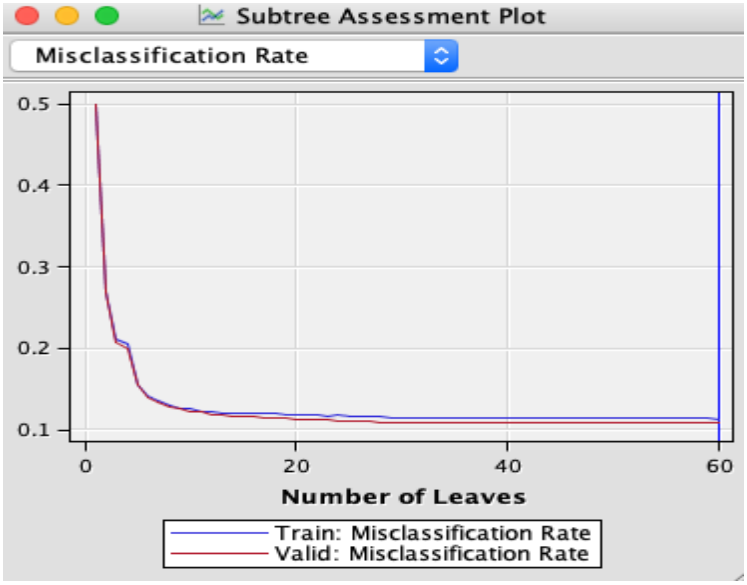


Figure 12

From the optimized plot, it has been shown that the train and validate chunks model has a trend in its Misclassification Rate for the number of leaves > 30, above which there is no further change as the curve is either constant or divergent as seen in Figure 13.

Settings: The only setting changed was in the subtree method where in the maximal it's the Largest while in the optimized the method will change to Assessment.

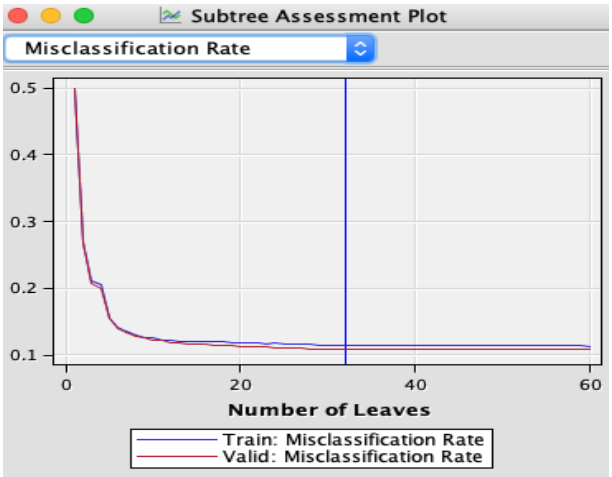


Figure 13

- Model Comparison (Maximal Decision Tree & Optimized Decision Tree)

AUR: The values for Maximal Decision Tree have been found to be 0.947 for the testing and 0.946 for validating classifiers. While the AUR values for Optimized Decision Tree have been found to be 0.935 for the training and 0.936 for validating classifiers as in Figure 15. The ROC curve is described as in Figure 14.

This clearly shows that the Maximal Decision Tree has more accuracy. The Maximal Decision Tree has been considered for its better results for further model comparison at later stages in this study.

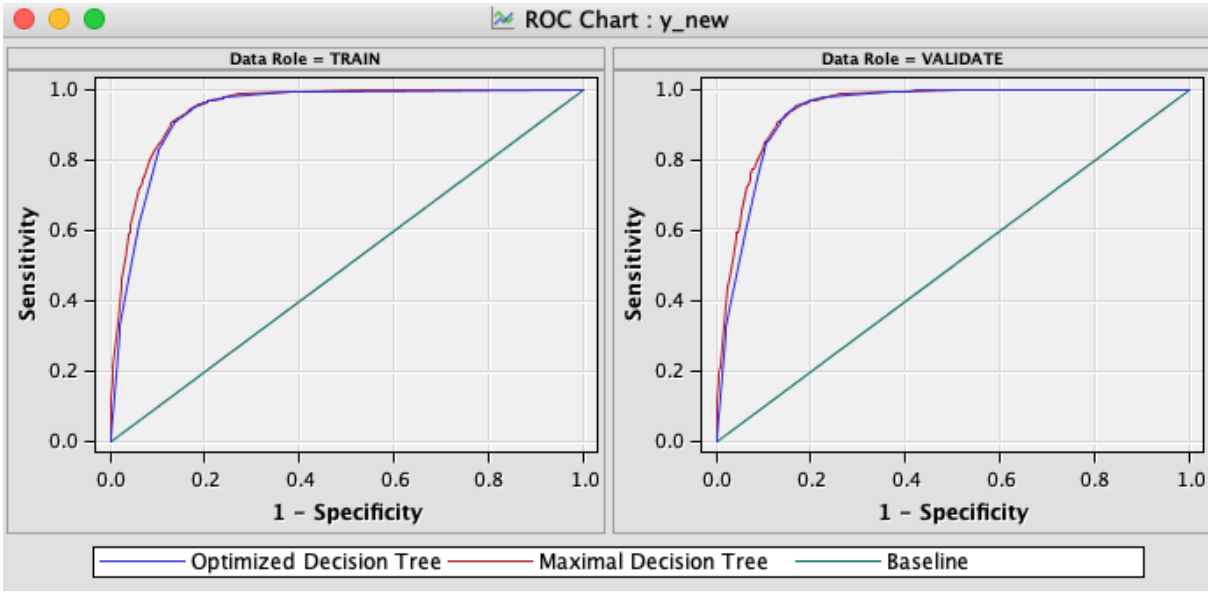


Figure 14

Fit Statistics
Model Selection based on Valid: Misclassification Rate (_VMISC_)

Selected Model	Model Node	Model Description	Valid: Misclassification Rate	Train: Average Squared Error	Train: Misclassification Rate	Valid: Average Squared Error
Y	Tree2	Optimized Decision Tree	0.10875	0.088014	0.11506	0.085920
	Tree	Maximal Decision Tree	0.10910	0.083936	0.11284	0.083276

Figure 15

Data Role=Train

Statistics	Tree2	Tree
Train: Bin-Based Two-Way Kolmogorov-Smirnov Probability Cutoff	0.59	0.58
Train: Kolmogorov-Smirnov Statistic	0.77	0.77
Train: Average Squared Error	0.09	0.08
Train: Roc Index	0.94	0.95

Data Role=Valid

Statistics	Tree2	Tree
Valid: Kolmogorov-Smirnov Statistic	0.78	0.78
Valid: Average Squared Error	0.09	0.08
Valid: Roc Index	0.94	0.95

2. *Logistic Regression*

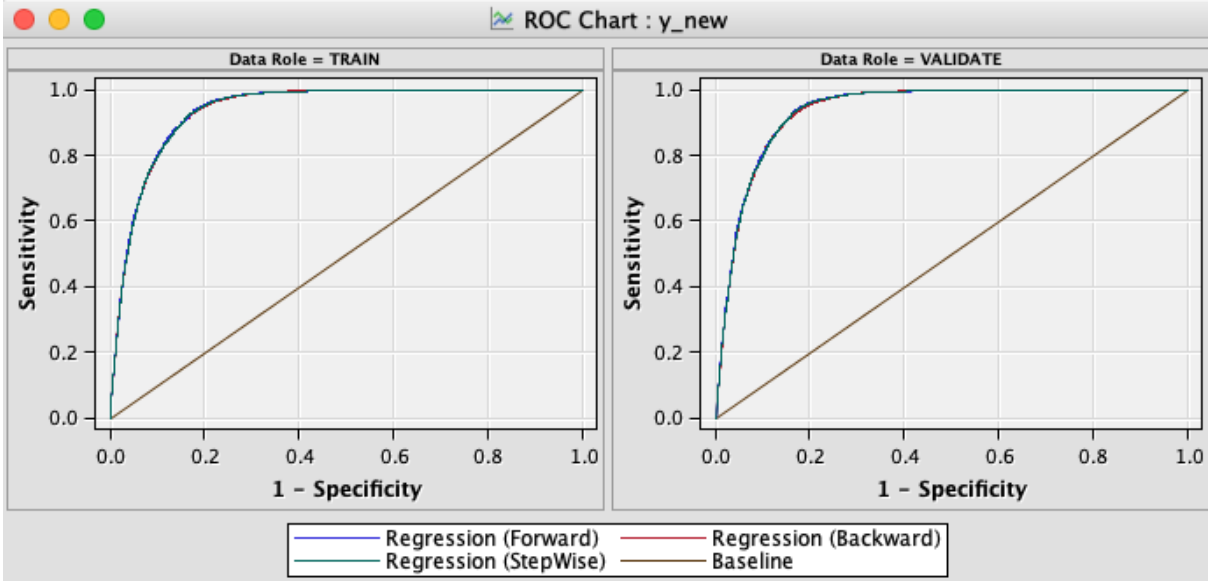
We have conducted Logistic Regression for this analysis as the output variable is categorical. Three different forms, including Forward, Backward and Stepwise, have been discussed within this and three different models for respective regressions have been generated appropriately.

Settings: The only setting changed was in the selecting the model for our three different regression.

- Model Comparison

The AUR values for Forward, Backward and Stepwise Logistic Regressions have been found to be 0.938,0.937 and 0.937 for the training and 0.938,0.937 and 0.937 for validating classifiers respectively. This clearly shows that all of them have the same accuracy.

But due to a point we could select Forward logistic regression for further model comparison at later stages in this study.



Fit Statistics
Model Selection based on Valid: Misclassification Rate (_VMISC_)

Selected Model	Model Node	Model Description	Valid: Misclassification Rate	Train: Average Squared Error	Train: Misclassification Rate	Valid: Average Squared Error
Y	Reg	Regression (StepWise)	0.12694	0.094456	0.13028	0.093593
	Reg3	Regression (Forward)	0.12799	0.093680	0.12921	0.092898
	Reg2	Regression (Backward)	0.12898	0.094667	0.13117	0.094160

Data Role=Train

Statistics	Reg	Reg3	Reg2
Train: Bin-Based Two-Way Kolmogorov-Smirnov Probability Cutoff	0.47	0.47	0.47
Train: Kolmogorov-Smirnov Statistic	0.76	0.76	0.75
Train: Akaike's Information Criterion	33628.25	33441.88	33655.15
Train: Average Squared Error	0.09	0.09	0.09
Train: Roc Index	0.94	0.94	0.94

Data Role=Valid

Statistics	Reg	Reg3	Reg2
Valid: Kolmogorov-Smirnov Statistic	0.76	0.77	0.76
Valid: Average Squared Error	0.09	0.09	0.09
Valid: Roc Index	0.94	0.94	0.94

3. *Neural Networks*

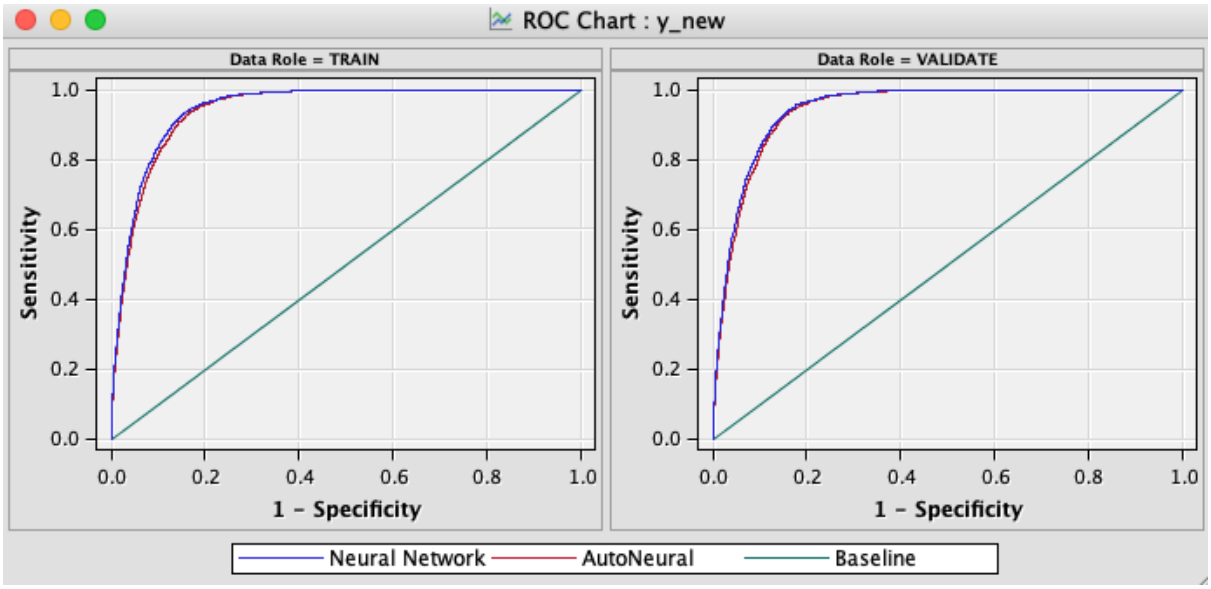
In SAS Enterprise Miner, the Neural Network was initially built using the default settings. The misclassification rate was indicated to be high for both train and validate part, and the gap between the two values was also noted to be greater. In order to maximize the Neural Network's output, only a subset of input variables must be used for classification. By using variable selection , we reduced the number of input variables and moved their results to the Neural Network, where only relevant input variables are considered.

We changed the number of hidden units to 64 for getting the lowest misclassification rate for validation data to further improve the performance.

- Model Comparison

The AUR values for Neural Network have been found to be 0.946 for the training and 0.946 for validating classifiers. And the AUR values for Auto Neural have been found to be 0.942 for the training and 0.942 for validating classifiers. This clearly shows that the Neural Network has more accuracy.

Neural Network was chosen for further model comparison at later stages in this study.



Fit Statistics
Model Selection based on Valid: Misclassification Rate (_VMISC_)

Selected Model	Model Node	Model Description	Valid: Misclassification Rate	Train: Average Squared Error	Train: Misclassification Rate	Valid: Average Squared Error
Y	Neural	Neural Network	0.11224	0.084726	0.11263	0.084465
	AutoNeural	AutoNeural	0.11324	0.088300	0.11737	0.087136

Data Role=Train

Statistics	Neural	Auto Neural
Train: Bin-Based Two-Way Kolmogorov-Smirnov Probability Cutoff	0.56	0.58
Train: Kolmogorov-Smirnov Statistic	0.78	0.77
Train: Akaike's Information Criterion	31162.50	30239.77
Train: Average Squared Error	0.08	0.09
Train: Roc Index	0.95	0.94

Data Role=Valid

Statistics	Neural	Auto Neural
Valid: Kolmogorov-Smirnov Statistic	0.78	0.77
Valid: Average Squared Error	0.08	0.09
Valid: Roc Index	0.95	0.94

4. *Gradient Boosting*

We have chosen Gradient Boosting for this analysis as the output variable is categorical. Also, it's a type of greedy algorithm which increases the accuracy of the model by reducing overfitting.

Settings: For subtree we have chosen assessment measure as misclassification.

- Model Comparison

The AUR values for Gradient Boosting have been found to be 0.797 for the testing and 0.8 for validating classifiers.

Fit Statistics
Model Selection based on Valid: Misclassification Rate (_VMISC_)

Selected Model	Model Node	Model Description	Valid: Misclassification Rate	Train: Average Squared Error	Train: Misclassification Rate	Valid: Average Squared Error
Y	Boost	Gradient Boosting	0.12426	0.092479	0.12515	0.092288

Fit Statistics Table
Target: y_new

Data Role=Train

Statistics	Boost
Train: Bin-Based Two-Way Kolmogorov-Smirnov Probability Cutoff	0.54
Train: Kolmogorov-Smirnov Statistic	0.76
Train: Average Squared Error	0.09
Train: Roc Index	0.94

Data Role=Valid

Statistics	Boost
Valid: Kolmogorov-Smirnov Statistic	0.76
Valid: Average Squared Error	0.09
Valid: Roc Index	0.94

5. *HP Support Vector Machine*

The HP SVM was used as a model for our dataset. SVM can be used for both classification as well as regression problems and also it is a binary classifier.

The AUR values for Gradient Boosting have been found to be 0.794 for the testing and 0.799 for validating classifiers.

Fit Statistics
Model Selection based on Train: Roc Index (_AUR_)

Selected Model	Model Node	Model Description	Train: Roc Index	Train: Average Squared Error	Train: Misclassification Rate	Valid: Average Squared Error	Valid: Misclassification Rate
Y	HPSVM	HP SVM	0.938	0.16064	0.12903	0.15969	0.12694

Fit Statistics Table
Target: y_new

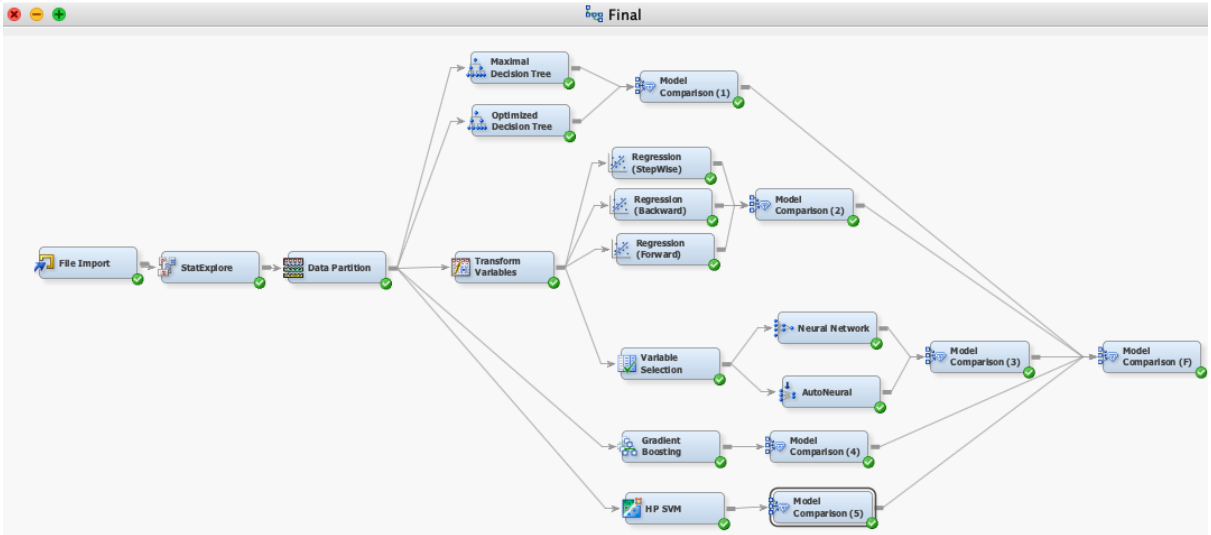
Data Role=Train

Statistics	HPSVM
Train: Bin-Based Two-Way Kolmogorov-Smirnov Probability Cutoff	0.48
Train: Kolmogorov-Smirnov Statistic	0.75
Train: Average Squared Error	0.16
Train: Roc Index	0.94

Data Role=Valid

Statistics	HPSVM
Valid: Kolmogorov-Smirnov Statistic	0.76
Valid: Average Squared Error	0.16
Valid: Roc Index	0.94

Final Workflow Diagram



Model Results Comparison

AUR Values according to fit statistics, the parameter values found for the Maximal Decision Tree,all the Logistic Regression, Neural Networks, Gradient Boosting and HP SVM Models are 0.95,0.94,0.95,0.94,0.94 (for train) and same for validate.

The Maximal Decision Tree and Neural Network are the more accurate at predicting outcome.

		AUR Values	AUR Values
		Train	Validate
Decision Tree	Maximal Decision	0.95	0.95
	Optimized Tree	0.94	0.94
Logistic Regression	Stepwise	0.94	0.94
	Backward	0.94	0.94
	Forward	0.94	0.94
Neural Networks	Auto NN	0.94	0.94
	NN	0.95	0.95
Gradient Boosting		0.94	0.94
HP SVM		0.94	0.94

Discussion

- In the main paper (Moro, Cortez & Rita, 2014), semi automated feature selection was done on the dataset for the original research carried out on the same dataset, where some of the features were personally selected. Data partitions were also rendered for train and validate parts in the ratio of 65:35 respectively. Different models were compared on the basis of the parameter called AUC (Area under Curve) and Area of LIFT Cumulative Curve, including Decision Tree, Support Vector Machine, Logistic Regression and Neural Network.
  By choosing various samples from the dataset, including 5, 10, 20, 30, 40, 50, 60 and 70 percent, the models considered were compared and the effects of the parameters were considered for analysis.
  In the original study it was found that the Neural Network provided the best results for the input dataset for all sample size values.

- In the paper(Wisaeng, 2013) four algorithms were used such as J48-graft algorithm, LAD tree (LADT) algorithm, radial basis function network (RBFN, and support vector machine (SVM). And The experimental study showed that the support vector machine achieves the highest sensitivity, specificity and accuracy of all classifiers.

- In the paper (Moro, Cortez & Rita, 2017) in order to tune out the DM model outcomes, three variations of the CRISP-DM technique were used. In effect, each iteration of CRISP-DM has proven to be of great benefit, as predictive performance achieved has improved. High predictive performance was achieved by the best model materialized by a Support Vector Machine (SVM).

- This paper (A.Elsalamony, 2014) analyzed and contrasted the classification efficiency of the MPLNN, TAN, LR and C5.0 models of four different data mining techniques on the bank direct marketing data set for bank deposit subscription classification.Three statistical methods have been applied to the classification performance of the four models: classification accuracy, sensitivity and specificity.
  The ratio of 70% and 30%, respectively, was divided into training and testing in this data collection. The efficacy of models has been shown by experimental results. Slightly better performance was achieved with C5.0 than with MLPNN, LR and TAN.

- The predicted results of four classifiers, MLPNN, Decision Tree (C4.5), Logistic Regression and Random Forest, are used and compared in this paper (Asare-Frempong & Jayabalan, 2017). Results from the research showed that among the four classifiers with Decision Tree (C4.5) coming second in terms of prediction ability, the Random Forest Classifier has the better predictive ability.

- For this project analysis for data partitions, the train and validate parts in the ratio of 75:25 respectively were rendered. The AUR value is the parameters considered for evaluation. The performance of different models such as Decision Tree, Logistic Regression, Neural Network, Gradient Boosting and HP SVM were compared. In determining the outcome variable, we considered the Maximal Decision Tree has the highest accuracy.

- In Total, it can be assumed that there is no way to prove that one model chosen would be better than the other considering different analyses in a specific analysis. It relies on various conditions considered for model performance testing because the model's accuracy is determined by data cleaning, feature selection, etc.

References

Moro, S., Cortez, P., & Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. Decision Support Systems, 62, 22-31. doi: 10.1016/j.dss.2014.03.001

Wisaeng, K. (2013). An Empirical Comparison of Data Mining Techniques in Medical Databases. International Journal Of Computer Applications, 77(7), 23-27. doi: 10.5120/13408-1061

Moro, S., Cortez, P., & Rita, P. (2017). A divide-and-conquer strategy using feature relevance and expert knowledge for enhancing a data mining approach to bank telemarketing. Expert Systems, 35(3), e12253. doi: 10.1111/exsy.12253

A.Elsalamony, H. (2014). Bank Direct Marketing Analysis of Data Mining Techniques. International Journal Of Computer Applications, 85(7), 12-22. doi: 10.5120/14852-3218

Asare-Frempong, J., & Jayabalan, M. (2017). Predicting customer response to bank direct telemarketing campaign. 2017 International Conference On Engineering Technology And Technopreneurship (ICE2T). doi: 10.1109/ice2t.2017.8215961