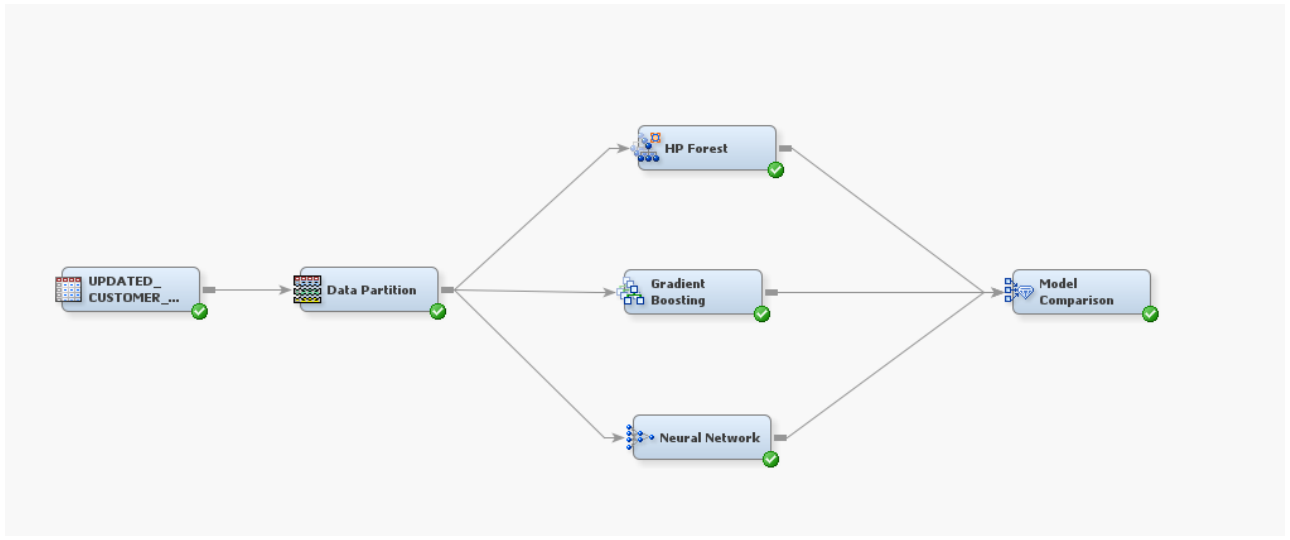
ADVANCED DATA EXPLORATION TECHNIQUES PROJECT

Submitted by:

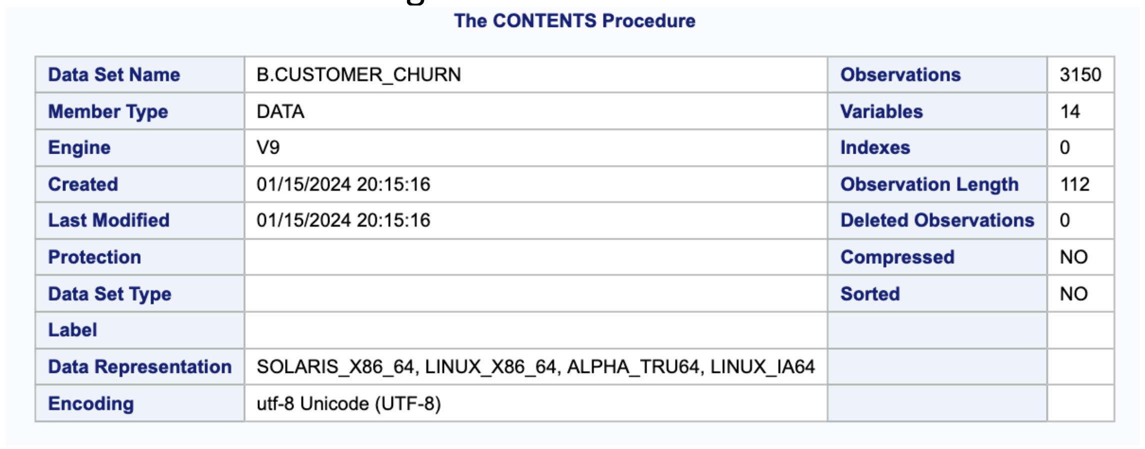
Emilian Loric

Andi Mutiah Azzahra



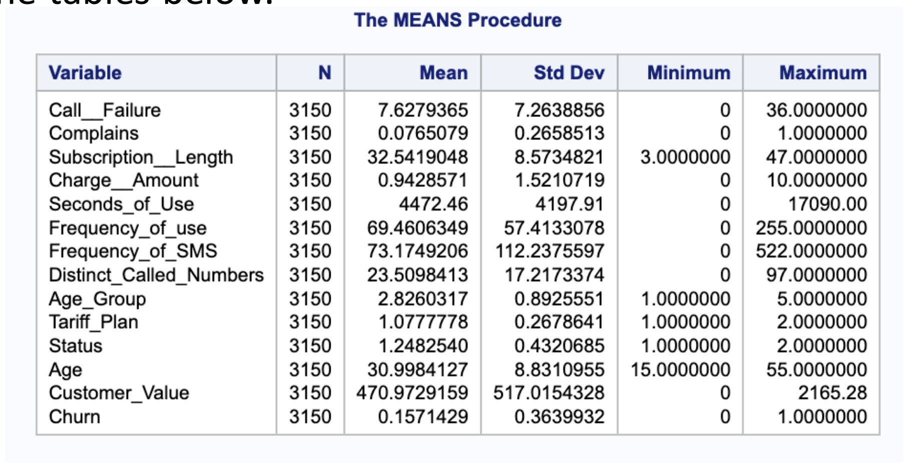
Data description

- The data used for the project is the "Iranian Customer Churn" data from https://archive.ics.uci.edu/dataset/563/iranian+churn+dataset.
- The original data has 14 features and 3150 observations with "Churn" which is a binary classification as the target variable.
- However, two variables are eliminated in the machine learning model deployment. Firstly, "Customer Value" is excluded due to no clear information provided of what the value exactly represents. Secondly, "Age Group", since it is treated the same as "Age" variable.
- The data has no missing value. More information can be seen in the tables below.



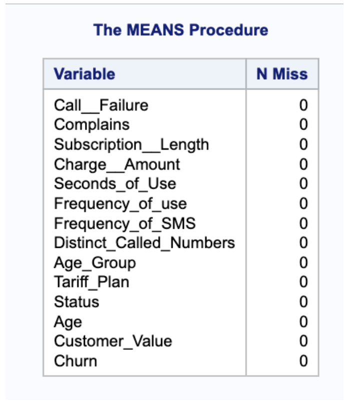
The CONTENTS Procedure

Data Set Name	B.CUSTOMER_CHURN	Observations	3150
Member Type	DATA	Variables	14
Engine	V9	Indexes	0
Created	01/15/2024 20:15:16	Observation Length	112
Last Modified	01/15/2024 20:15:16	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64		
Encoding	utf-8 Unicode (UTF-8)		



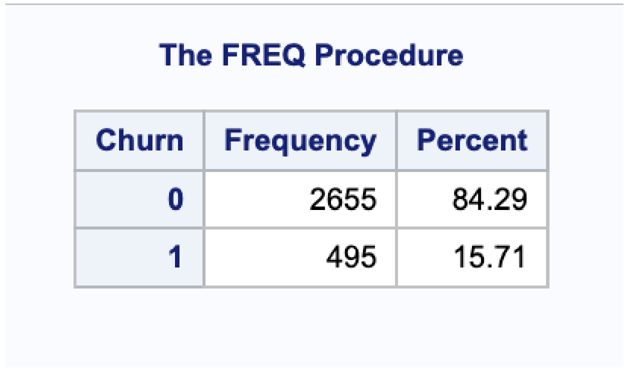
The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Call__Failure	3150	7.6279365	7.2638856	0	36.0000000
Complains	3150	0.0765079	0.2658513	0	1.0000000
Subscription__Length	3150	32.5419048	8.5734821	3.0000000	47.0000000
Charge__Amount	3150	0.9428571	1.5210719	0	10.0000000
Seconds_of_Use	3150	4472.46	4197.91	0	17090.00
Frequency_of_use	3150	69.4606349	57.4133078	0	255.0000000
Frequency_of_SMS	3150	73.1749206	112.2375597	0	522.0000000
Distinct_Called_Numbers	3150	23.5098413	17.2173374	0	97.0000000
Age_Group	3150	2.8260317	0.8925551	1.0000000	5.0000000
Tariff_Plan	3150	1.0777778	0.2678641	1.0000000	2.0000000
Status	3150	1.2482540	0.4320685	1.0000000	2.0000000
Age	3150	30.9984127	8.8310955	15.0000000	55.0000000
Customer_Value	3150	470.9729159	517.0154328	0	2165.28
Churn	3150	0.1571429	0.3639932	0	1.0000000



The MEANS Procedure

Variable	N Miss
Call__Failure	0
Complains	0
Subscription__Length	0
Charge__Amount	0
Seconds_of_Use	0
Frequency_of_use	0
Frequency_of_SMS	0
Distinct_Called_Numbers	0
Age_Group	0
Tariff_Plan	0
Status	0
Age	0
Customer_Value	0
Churn	0



The FREQ Procedure

Churn	Frequency	Percent
0	2655	84.29
1	495	15.71

Variables Explanation

1. Call Failures: number of call failures
2. Complains: binary (0: No complaint, 1: complaint)
3. Subscription Length: total months of subscription
4. Charge Amount: Ordinal attribute (0: lowest amount, 9: highest amount)
5. Seconds of Use: total seconds of calls
6. Frequency of use: total number of calls
7. Frequency of SMS: total number of text messages
8. Distinct Called Numbers: total number of distinct phone calls
9. Age Group: ordinal attribute (1: younger age, 5: older age)
10. Tariff Plan: binary (1: Pay as you go, 2: contractual)
11. Status: binary (1: active, 2: non-active)
12. Age : Age of the customer
13. Churn: binary (1: churn(unsubscribe), 0: non-churn(stay subscribed) - Target variable
14. Customer Value: The calculated value of customer

Code to execute data in SAS

```
libname b "/home/u62564367/ADET/project1";


/* Importing the data */
FILENAME  REFFILE '/home/u62564367/ADET/project1/Customer  Churn.csv';
RUN;
options validvarname=v7;
PROC IMPORT DATAFILE=REFFILE
   DBMS=CSV
   OUT=b.customer_churn;
   GETNAMES=YES;
RUN;
PROC CONTENTS  DATA=b.customer_churn;
RUN;

/* ----------------- DATA  PREPROCESSING ------------------- */
/* Creating table */
data b.customer_churn;
   set b.customer_churn;
run;
/* Showing statistical summary */
proc means data=b.customer_churn;
run;
/* Checking missing values */
proc means data=b.customer_churn nmiss;
run;
/* Checking the proportion of target variable churn */
proc freq data=b.customer_churn;
  tables churn/ nocum;
run;
```

```
/* some of the variables do not have the appropriate format */
/*
Call__Failure
Complains
Subscription__Length
Charge__Amount
Seconds_of_Use
Frequency_of_use
Frequency_of_SMS
Distinct_Called_Numbers
Age_Group
Tariff_Plan
Status
Age
Customer_Value
Churn
*/
/* Rename variables */
data b.customer_churn;
  set b.customer_churn(rename=(Call__Failure = Call_Failure Subscription__Length  = Subscription_Length
Charge__Amount  = Charge_Amount  ));
run;

/* Creating standard binary classification */
data b.customer_churn;
   set b.customer_churn;
   /*Changing the value of customer status from 1 = active, 2 = inactive to 1 = active and 0 =inactive */
   if status = 2 then status = 0;
   /*Cahnging the value of customer tariff_plan from 1 = pay as you go , 2 = contractual to 1 = pay as you go and 0 =
contractual */
   if tariff_plan = 2 then tariff_plan = 0;
run;
```

l

Additional Features

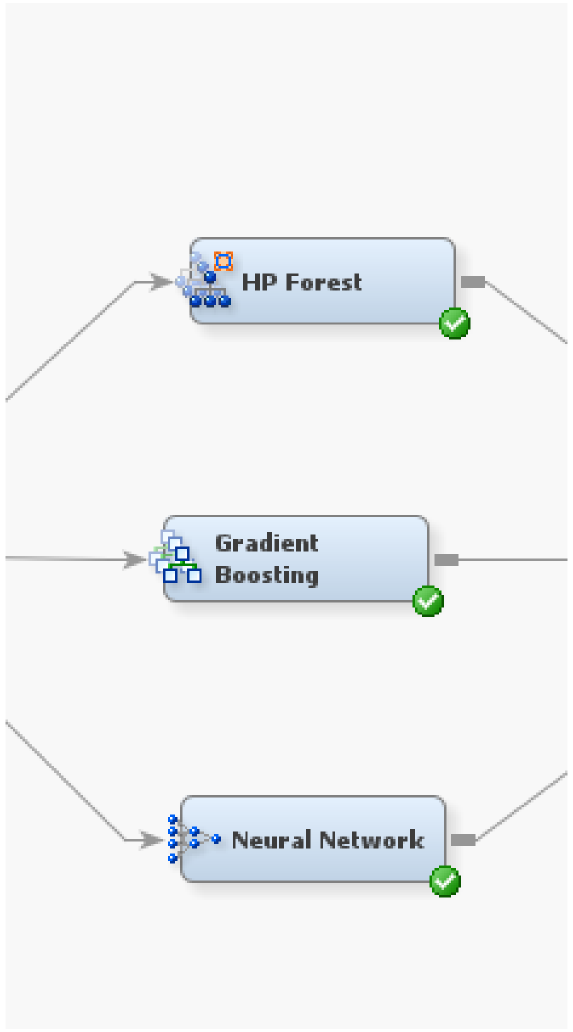
Based on the data provided, 7 features are created which are catagorized into 3 segments:

- **Usage Intensity Features:**
  - **call_failure_rate:** Call failures divided by frequency of use to give a failure rate per call made
  - **average_call_length:** Seconds of use divided by the frequency of use to see the average length per call
  - **average_monthly_usage_ratio:** seconds of use divided by subscription length to see how long, in average, customer spent on call per month.
  - **average_monthly_charge:** charge amount divided by subscription length to see the average of how much the customer is charged per month
  - **text_to_call_ratio:** dividing frequency of sms with frequency of use to understand wether the customer prefers texting or calling
- **Customer Interaction Features:**
  - **average_calls_per_contact:** Frequency of use divided by distinct called numbers to see how often the customer calls the same number.
- **Loyalty and Satisfaction Features:**
  - **Charge to Usage Ratio:** The charge amount divided by seconds of use to understand the cost to usage relation.

Code of Additional Feature on SAS

```
/* ------------------ FEATURE ENGINEERING ---------------– */
data b.updated_customer_churn;
  set b.customer_churn;
  /* Usage Intensity Features */
  call_failure_rate = round(call_failure / frequency_of_use, 0.01);
  average_call_length = round(seconds_of_use / frequency_of_use, 0.01);
  average_monthly_usage_ratio = round(seconds_of_use / subscription_length, 0.01);
  average_monthly_charge = round(charge_amount  / subscription_length, 0.001);
  text_to_call_ratio = round(frequency_of_sms  / frequency_of_use, 0.01);

  /* Customer Interaction Features */
  average_calls_per_contact = round(frequency_of_use  / distinct_called_numbers, 0.01);

  /* Loyalty and Satisfaction Features */
  charge_to_usage_ratio = round(charge_amount  / seconds_of_use, 0.00001);

run;
```

Methods compared



- Random Forest

In classification tasks, a Random Forest operates by constructing multiple decision trees during training and outputting the class that is the mode of the classes (classification) of the individual trees. It adds randomness to model building to ensure diverse trees, which makes the ensemble robust against overfitting.

- Gradient Boosting

Gradient Boosting builds an ensemble of weak prediction models, typically decision trees, in a sequential manner. Each tree is trained to correct the mistakes of the previous one, gradually improving the model's accuracy. This method combines the predictions from multiple trees to make more accurate and robust predictions than any single tree.

- Neural Network

A Neural Network, consists of layers of interconnected nodes (neurons) where each connection represents a weight. During training, these weights are adjusted through backpropagation to minimize the error in predictions. Neural Networks are adept at learning complex patterns and relationships in data.

Random Forest

Why ? We decided to choose Random Forest because its major strength lies in its robustness and simplicity. Also, Random Forest is less prone to overfitting compared to many other algorithms.

Robustness and Simplicity: Random Forest is less likely to overfit than many other classifiers.

Feature Importance: It provides a straightforward indication of feature importance.

Good for a Wide Range of Problems: Performs well across a wide range of classification tasks and datasets.

Handling Different Types of Data: It can handle mixed types of data (numeric and binary in our case)

Interpretability: While better than neural networks, Random Forests still lack the interpretability of simpler models like decision trees.

Gradient Boosting

Why ? The key benefit of Gradient Boosting is its high performance and accuracy. It is particularly effective due to its approach of building an ensemble of weak prediction models in a sequential manner, which often results in superior predictive accuracy

High Performance: Gradient Boosting often provide very competitive accuracy.

Good with Imbalanced Data: It can handle class imbalance by constructing successive training sets based on incorrectly classified examples.

Handling Different Types of Data: It can handle mixed types of data (numerical and categorical).

Parameter Tuning: Requires careful tuning of parameters and may be sensitive to parameter settings.

Neural Network

Why ? The primary advantage of Neural Networks is their ability to model complex, non-linear relationships in data. Even if we knew that it would probably performs much less than Random Forest and Gradient Boosting in our case, we personally found it interesting to compare this methods with the other ones.

Flexibility and Complexity: Neural networks can model complex, non-linear relationships in data.

Scalability: It scales well with large amounts of data and can benefit from GPU acceleration.

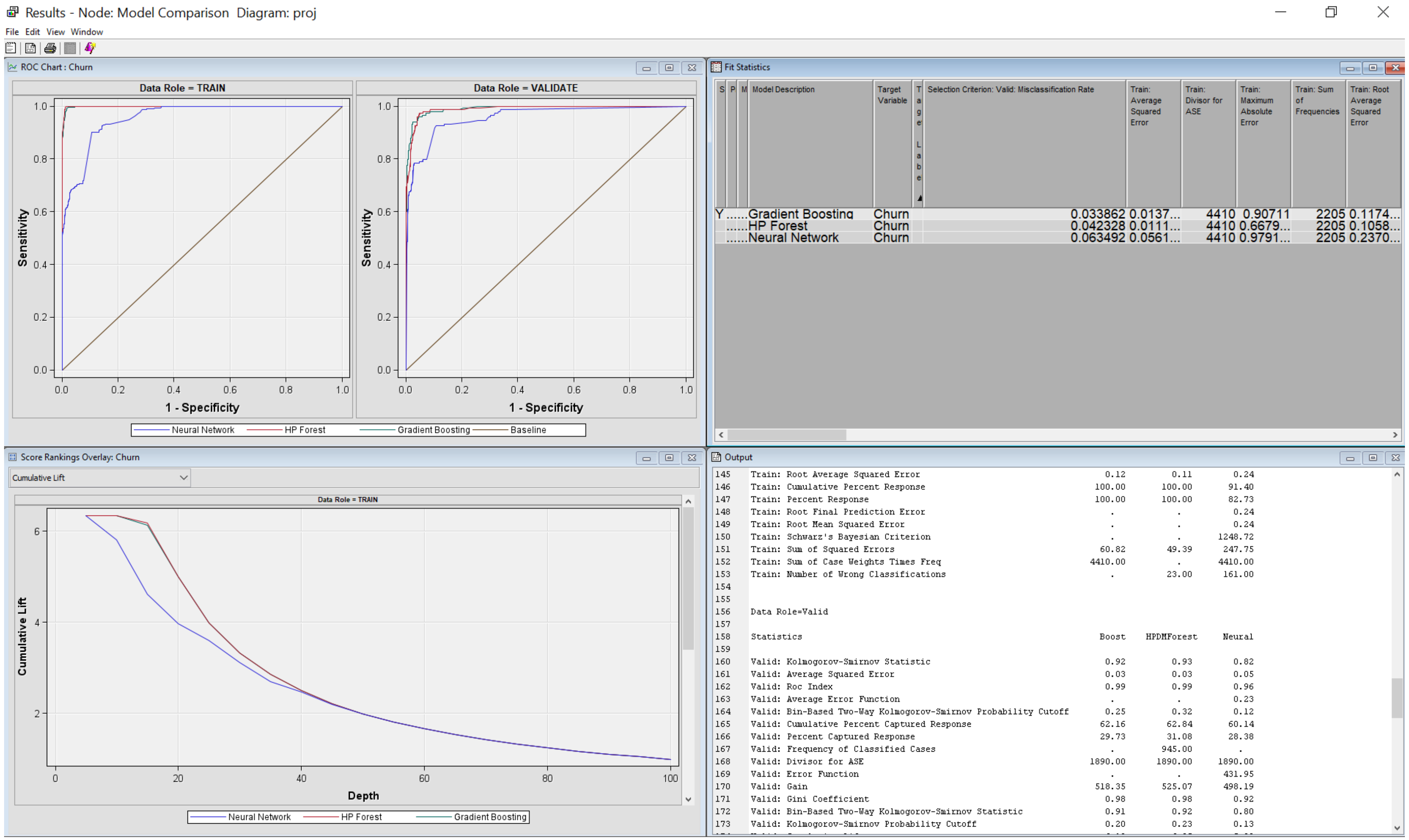
Data Hungry: It generally required large amounts of data to perform well and avoid overfitting.

Computational Intensity: Training can be computationally expensive and time-consuming.

Interpretability: It is seen as "black boxes," with limited interpretability regarding how decisions are made.

Tuning and Complexity: Setting up a neural network can be complex and requires expertise.

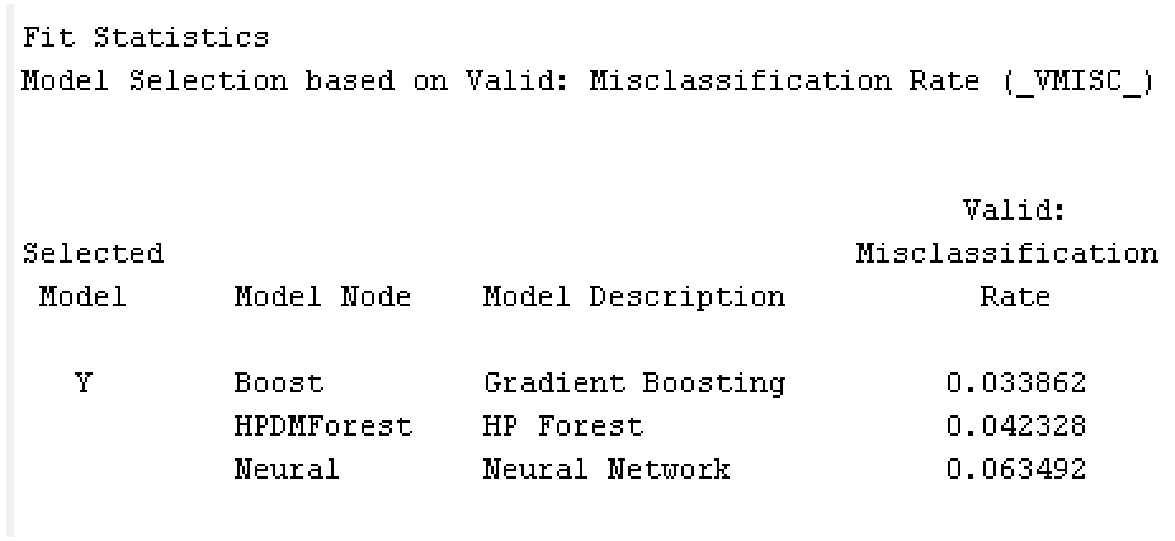
Models' assessment and comparison



S	P	M	Model Description	Target Variable	Target Label	Selection Criterion: Valid: Misclassification Rate	Train: Average Squared Error	Train: Divisor for ASE	Train: Maximum Absolute Error	Train: Sum of Frequencies	Train: Root Average Squared Error
Y	...	...	Gradient Boosting	Churn		0.033862	0.0137...	4410	0.90711	2205	0.1174...
	...	...	HP Forest	Churn		0.042328	0.0111...	4410	0.6679...	2205	0.1058...
	...	...	Neural Network	Churn		0.063492	0.0561...	4410	0.9791...	2205	0.2370...

```
Train: Root Average Squared Error                               0.12        0.11        0.24
Train: Cumulative Percent Response                            100.00      100.00       91.40
Train: Percent Response                                       100.00      100.00       82.73
Train: Root Final Prediction Error                               .           .         0.24
Train: Root Mean Squared Error                                   .           .         0.24
Train: Schwarz's Bayesian Criterion                              .           .      1248.72
Train: Sum of Squared Errors                                   60.82       49.39      247.75
Train: Sum of Case Weights Times Freq                        4410.00         .      4410.00
Train: Number of Wrong Classifications                           .         23.00      161.00


Data Role=Valid

Statistics                                                     Boost   HPDMForest     Neural

Valid: Kolmogorov-Smirnov Statistic                             0.92        0.93        0.82
Valid: Average Squared Error                                    0.03        0.03        0.05
Valid: Roc Index                                                0.99        0.99        0.96
Valid: Average Error Function                                    .           .         0.23
Valid: Bin-Based Two-Way Kolmogorov-Smirnov Probability Cutoff  0.25        0.32        0.12
Valid: Cumulative Percent Captured Response                    62.16       62.84       60.14
Valid: Percent Captured Response                               29.73       31.08       28.38
Valid: Frequency of Classified Cases                             .        945.00         .
Valid: Divisor for ASE                                       1890.00     1890.00     1890.00
Valid: Error Function                                            .           .       431.95
Valid: Gain                                                   518.35      525.07      498.19
Valid: Gini Coefficient                                         0.98        0.98        0.92
Valid: Bin-Based Two-Way Kolmogorov-Smirnov Statistic           0.91        0.92        0.80
Valid: Kolmogorov-Smirnov Probability Cutoff                    0.20        0.23        0.13
```

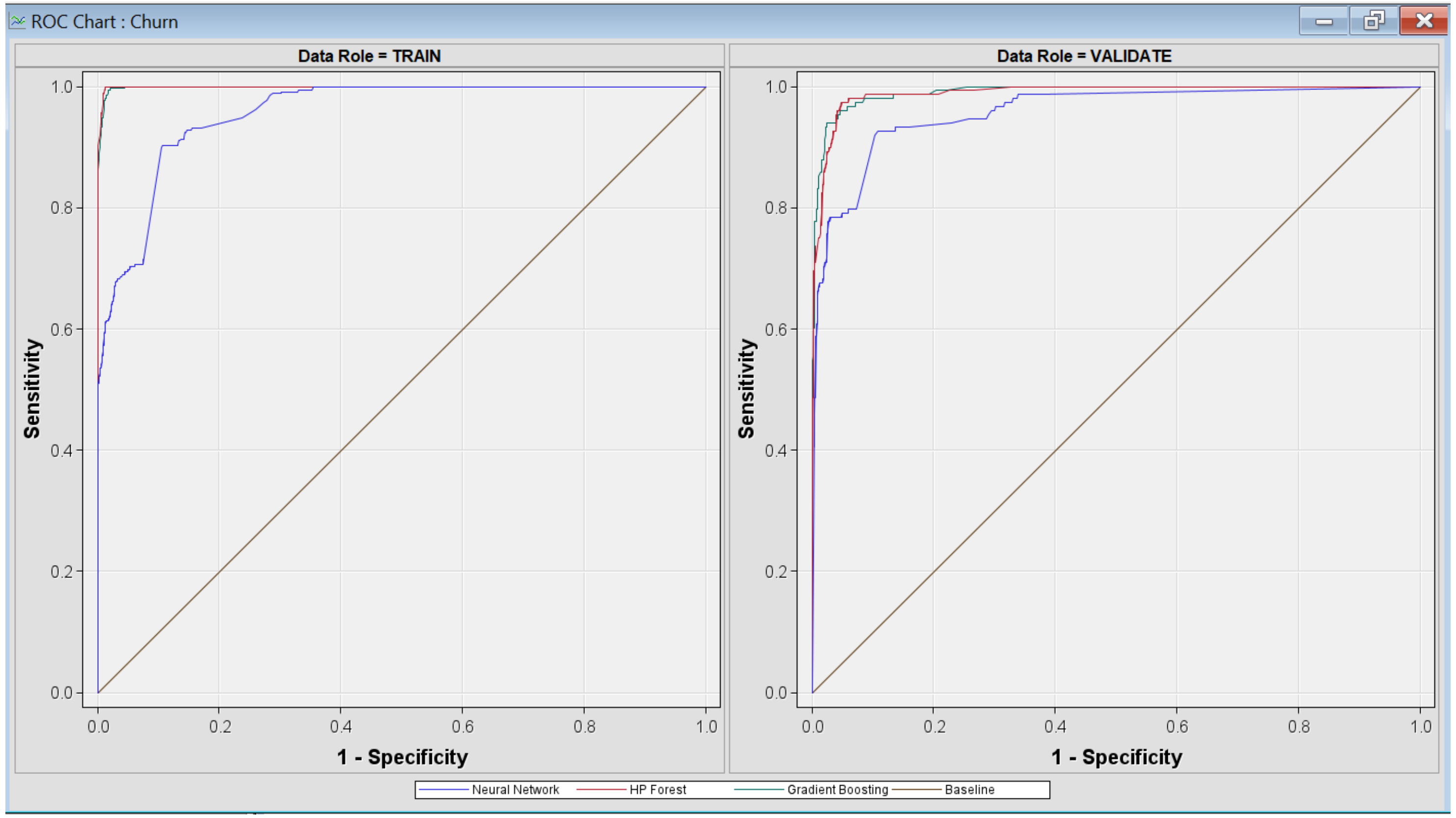
Misclassification rate



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



                                                         Valid:
Selected                                           Misclassification
 Model       Model Node    Model Description             Rate

   Y         Boost         Gradient Boosting           0.033862
             HPDMForest    HP Forest                   0.042328
             Neural        Neural Network              0.063492
```

- Misclassification rate represents the proportion of the misclassified observations over the entire dataset. The lowest is the rate, the better the model performance is.
- It is obtained by calculating number of miscalulated instances divided by the total instances in the dataset
- In terms of misclassification, Gradient Boosting model performs the lowest misclassification rate of 0.0339 which indicates the best performance, and Neural Network performs the highest rate of 0.0634 compared to all models.

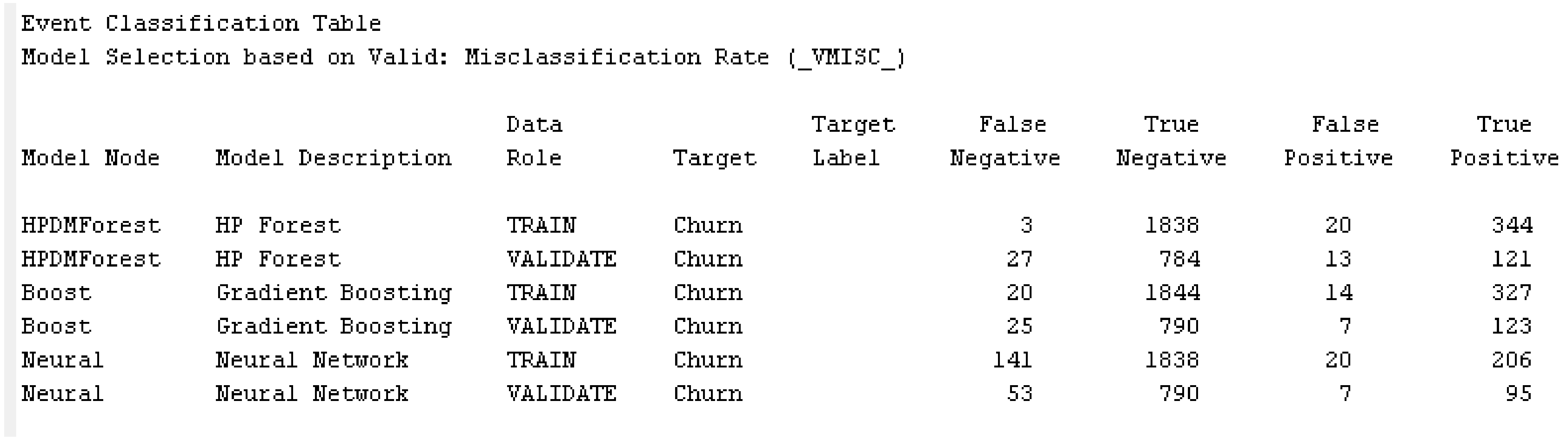
ROC curve



```
Data Role=Valid

Statistics                                                Boost    HPDMForest     Neural

Valid: Kolmogorov-Smirnov Statistic                        0.92         0.93        0.82
Valid: Average Squared Error                               0.03         0.03        0.05
Valid: Roc Index                                           0.99         0.99        0.96
```

- Here, we analyze the ROC curve of our 3 models. We see that the one of neural network (blue) is the worst one compared to the others. Then, we see ROC curve of Gradient boosting and Random Forest are very close and it is difficult to know which one minimize the Euclidean distance with the top left corner (sensitivity = 1 and 1-specificity = 0).
- So, we get a look on the ROC index from the statistical part, and there we see gradient boosting and random forest get the same result (0.99) which is excellent.

Confusion matrix



```
Event Classification Table
Model Selection based on Valid: Misclassification Rate (_VMISC_)

                                 Data                  Target    False     True      False     True
Model Node   Model Description   Role       Target    Label     Negative  Negative  Positive  Positive

HPDMForest   HP Forest           TRAIN      Churn                    3      1838        20       344
HPDMForest   HP Forest           VALIDATE   Churn                   27       784        13       121
Boost        Gradient Boosting   TRAIN      Churn                   20      1844        14       327
Boost        Gradient Boosting   VALIDATE   Churn                   25       790         7       123
Neural       Neural Network      TRAIN      Churn                  141      1838        20       206
Neural       Neural Network      VALIDATE   Churn                   53       790         7        95
```

About confusion matrix, we focus on the entire dataset which is both train and validate to evaluate the model better.
At this stage, we clearly see that Gradient Boosting is better than Neural Network and Random Forest on the absolute number of misclassifications.

Model Performance Evaluation

To calculate the validation score of accuracy performed below, both training and validation data are evaluated separately to analyze the performance of the model to the whole dataset. We focus more on minimizing false negative since we want to prevent the customer to unsubscribe to the service.

Therefore, higher recall score will be preferable. In this case, Gradient Boosting performs the best.

HP Forest (Training)

		predicted class	
		1	0
Real class	1	344	3
	0	20	1838

Accuracy	99%
Miss class error	1%
Precision Score	95%
Recall	99%
F1 score	97%
Sensitivity(1)	99%
Specificity(0)	99%

Gradient Boosting (Training)

		predicted class	
		1	0
Real class	1	327	20
	0	14	1844

Accuracy	98%
Miss class error	2%
Precision Score	96%
Recall	94%
F1 score	95%
Sensitivity(1)	94%
Specificity(0)	99%

Neural Network (Training)

		predicted class	
		1	0
Real class	1	206	141
	0	20	1838

Accuracy	93%
Miss class error	7%
Precision Score	91%
Recall	59%
F1 score	72%
Sensitivity(1)	59%
Specificity(0)	99%

HP Forest (Validation)

		predicted class	
		1	0
Real class	1	121	27
	0	13	784

Accuracy	96%
Miss class error	4%
Precision Score	90%
Recall	82%
F1 score	86%
Sensitivity(1)	82%
Specificity(0)	98%

Gradient Boosting(Validation)

		predicted class	
		1	0
Real class	1	123	25
	0	7	790

Accuracy	97%
Miss class error	3%
Precision Score	95%
Recall	83%
F1 score	88%
Sensitivity(1)	83%
Specificity(0)	99%

Neural Network (Validation)

		predicted class	
		1	0
Real class	1	95	53
	0	7	790

Accuracy	94%
Miss class error	6%
Precision Score	93%
Recall	64%
F1 score	76%
Sensitivity(1)	64%
Specificity(0)	99%

Conclusion

- Because our dataset is imbalanced, instead of looking at the accuracy, we focus more on evaluating F1-Score which is a weighted harmonic mean of precision and recall.
- In our business case, we want to minimize the customers who are susceptible to quit soon. For that, we want to minimize the False Negative. The corresponding metric is called recall (same as sensitivity).
- Finally, the model that gets the highest performance of F1-Score and minimizes the false negative classification (recall) in the validation data is the Gradient Boosting model which accounts for 83% and 88% of recall and F1-score respectively.