Midterm Project
Applied Analytics

Group 10

Fall 2018

Group Members

Aryasomayajula Pawan Dixit

Divyank Garg

Mayank Mishra

Sajjaat Reyath

Shivam Solank

Outline

Problem Statement

A data set consisting of 30,000 bank records of credit card customers was provided and the target is to predict whether the customer defaults on credit card balance or not based on the data related to the customer providing his/her education, age, monthly bill, marital status etc.

Data Preprocessing

The dataset had 13607 values including NA, outlier (data dictionary was referred to set the limits for interval attributes and categories for the categorical attributes) and missing values.

(none) not Equal to ...

Columns: Label Mining

Name	Role	Level	Report	Order	Drop	Lower Limit	Upper Limit
Age	Input	Interval	No		No	.	.
Apr_Bill	Input	Interval	No		No	.	.
Apr_PayPercent	Input	Interval	No		No	.	.
Apr_Payment	Input	Interval	No		No	.	.
Apr_Status	Input	Interval	No		No	.	.
Credit_Limit	Input	Interval	No		No	.	.
Customer	ID	Interval	No		No	.	.
Default	Target	Binary	No		No	.	.
Education	Input	Nominal	No		No	.	.
Feb_Bill	Input	Interval	No		No	.	.
Feb_PayPercent	Input	Interval	No		No	.	.
Feb_Payment	Input	Interval	No		No	.	.
Feb_Status	Input	Interval	No		No	.	.
Gender	Input	Binary	No		No	.	.
Jan_Bill	Input	Interval	No		No	.	.
Jan_PayPercent	Input	Interval	No		No	.	.
Jan_Payment	Input	Interval	No		No	.	.
Jan_Status	Input	Interval	No		No	.	.
Jun_Bill	Input	Interval	No		No	.	.
Jun_PayPercent	Input	Interval	No		No	.	.
Jun_Payment	Input	Interval	No		No	.	.
Jun_Status	Input	Interval	No		No	.	.
Mar_Bill	Input	Interval	No		No	.	.
Mar_PayPercent	Input	Interval	No		No	.	.
Mar_Payment	Input	Interval	No		No	.	.
Mar_Status	Input	Interval	No		No	.	.
Marital_Status	Input	Nominal	No		No	.	.
May_Bill	Input	Interval	No		No	.	.
May_PayPercen	Input	Interval	No		No	.	.
May_Payment	Input	Interval	No		No	.	.
May_Status	Input	Interval	No		No	.	.
card_class	Input	Nominal	No		No	.	.

```
Attribute Counts
..................... Missing  Outliers
Default.........          0         0
Gender..........       3083         0
```



```
Education.......       4521         0
Marital_Status..          0         0
card_class......          0         0
Age.............       5999         0
Credit_Limit....          0         0
Jun_Status......          0         0
May_Status......          0         0
Apr_Status......          0         0
Mar_Status......          0         0
Feb_Status......          0         0
Jan_Status......          0         0
Jun_Bill........          0         1
May_Bill........          0         1
Apr_Bill........          0         1
Mar_Bill........          0         0
Feb_Bill........          0         0
Jan_Bill........          0         1
Jun_Payment.....          0         0
May_Payment.....          0         0
Apr_Payment.....          0         0
Mar_Payment.....          0         0
Feb_Payment.....          0         0
Jan_Payment.....          0         0
Jun_PayPercent..          0         0
May_PayPercent..          0         0
Apr_PayPercent..          0         0
Mar_PayPercent..          0         0
Feb_PayPercent..          0         0
Jan_PayPercent..          0         0
```

Interactive Replacement Interval Filter

Columns: ☐ Label ☐ Mining ☐ Basic ☐ Statistics

Name	Use	Limit Method	Replacement Lower Limit	Replacement Upper Limit	Replace Method	Lower Replacement Value	Up Va
Age	Default	User Specified	20	80	Missing	.	
Apr_Bill	Default	User Specified	-12000	32000	Missing	.	
Apr_PayPercent	Default	User Specified	0	1	Missing	.	
Apr_Payment	Default	User Specified	0	60000	Missing	.	
Apr_Status	Default	User Specified	-2	8	Missing	.	
Credit_Limit	Default	User Specified	100	80000	Missing	.	
Feb_Bill	Default	User Specified	-12000	32000	Missing	.	
Feb_PayPercent	Default	User Specified	0	1	Missing	.	
Feb_Payment	Default	User Specified	0	60000	Missing	.	
Feb_Status	Default	User Specified	-2	8	Missing	.	
Jan_Bill	Default	User Specified	-12000	32000	Missing	.	
Jan_PayPercent	Default	User Specified	0	1	Missing	.	
Jan_Payment	Default	User Specified	0	60000	Missing	.	
Jan_Status	Default	User Specified	-2	8	Missing	.	
Jun_Bill	Default	User Specified	-12000	32000	Missing	.	
Jun_PayPercent	Default	User Specified	0	1	Missing	.	
Jun_Payment	Default	User Specified	0	60000	Missing	.	
Jun_Status	Default	User Specified	-2	8	Missing	.	
Mar_Bill	Default	User Specified	-12000	32000	Missing	.	
Mar_PayPercent	Default	User Specified	0	1	Missing	.	
Mar_Payment	Default	User Specified	0	60000	Missing	.	
Mar_Status	Default	User Specified	-2	8	Missing	.	
May_Bill	Default	User Specified	-12000	32000	Missing	.	

Generate Summary   OK   Cancel

Output after the replacement

```
Data                                of                          Mode                  Mode2
Role     Variable Name      Role    Levels    Missing    Mode    Percentage    Mode2    Percentage

TRAIN    Education          INPUT     8         4521      2        39.72         1        29.81
TRAIN    Gender             INPUT     3         3083      2        54.20         1        35.53
TRAIN    Marital_Status     INPUT     4            0      2        53.21         1        45.53
TRAIN    card_class         INPUT     3            0      2        57.36         1        25.59
TRAIN    Default            TARGET    2            0      0        83.82         1        16.18



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

Data Role=TRAIN

Data     Variable                           Frequency
Role       Name       Role      Level           Count      Percent

TRAIN    Default     TARGET       0             25146       83.82
TRAIN    Default     TARGET       1              4854       16.18



Interval Variable Summary Statistics
(maximum 500 observations printed)

Data Role=TRAIN

                                          Standard       Non
Variable              Role      Mean      Deviation   Missing   Missing    Minimum     Median    Maximum   Skewness   Kurtosis

REP_Age               INPUT   35.49381    9.232328     24001      5999         21         34         79   0.741647   0.077663
REP_Apr_Bill          INPUT   1606.006    2350.196     29999         1   -5378.43     687.01   29243.94   2.741621   10.41638
REP_Apr_PayPercent    INPUT    0.32163    0.411748     30000         0          0     0.0617          1   0.921161   -1.01467
REP_Apr_Payment       INPUT   178.7184    602.1581     30000         0          0      61.56   30644.57   17.21664   564.3113
REP_Apr_Status        INPUT    -0.1662    1.196868     30000         0         -2          0          8   0.840682   2.084436
REP_Credit_Limit      INPUT    5725.52    4440.038     30000         0        300       4800      34200   0.991883   0.532664
REP_Feb_Bill          INPUT    1378.65    2079.263     30000         0   -2781.62     619.16   31709.25    2.87638   12.30588
REP_Feb_PayPercent    INPUT   0.340218    0.421385     30000         0          0     0.0602          1    0.81488   -1.21421
REP_Feb_Payment       INPUT   164.1391    522.5181     30000         0          0       51.3   14587.29   11.12742   180.0639
REP_Feb_Status        INPUT    -0.2662    1.133187     30000         0         -2          0          8   1.008197   3.989748
REP_Jan_Bill          INPUT   1328.362    2028.617     29999         1   -11614.4     583.73   23938.08   2.757149   10.57027
REP_Jan_PayPercent    INPUT   0.361249    0.429917     30000         0          0     0.0661          1   0.706188   -1.38834
REP_Jan_Payment       INPUT   178.3703    607.9893     30000         0          0       51.3   18080.38   10.64073   167.1614
REP_Jan_Status        INPUT    -0.2911    1.149988     30000         0         -2          0          8   0.948029   3.426534
REP_Jun_Bill          INPUT   1750.797    2511.923     29999         1   -5662.84     765.43   25541.04   2.621588   9.145481
REP_Jun_PayPercent    INPUT    0.31336    0.397166     30000         0          0     0.0744          1   0.999406    -0.8222
REP_Jun_Payment       INPUT   193.6945    566.4642     30000         0          0      71.82   29875.48   14.66837   415.2549
REP_Jun_Status        INPUT    -0.0167    1.123802     30000         0         -2          0          8   0.731975   2.720715
REP_Mar_Bill          INPUT   1479.593    2200.184     30000         0      -5814     651.58   30492.24   2.821965   11.30933
REP_Mar_PayPercent    INPUT    0.31924    0.413873     30000         0          0     0.0531          1    0.92226   -1.01926
REP_Mar_Payment       INPUT   165.0519    535.7827     30000         0          0       51.3    21238.2   12.90499   277.3338
REP_Mar_Status        INPUT   -0.22067    1.169139     30000         0         -2          0          8   0.999629   3.496983
REP_May_Bill          INPUT   1680.859    2427.175     29999         1   -2386.37     724.94   25443.77   2.653821   9.458058
```

All of these were replaced with missing values and finally those missing values were imputed by the following method

In SAS EM, <u>tree imputation method</u> was used for class variables and <u>mean imputation method</u> Interval variables

Property	Value
General	
Node ID	Impt
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
Nonmissing Variables	No
Missing Cutoff	50.0
Class Variables	
Default Input Method	Tree
Default Target Method	None
Normalize Values	Yes
Interval Variables	
Default Input Method	Mean
Default Target Method	None
Default Constant Value	
Default Character Value	
Default Number Value	.
Method Options	
Random Seed	12345
Tuning Parameters	...
Tree Imputation	

Imputation Output:

```
Variable Levels Summary
(maximum 500 observations printed)

                     Frequency
Variable    Role       Count

Customer    ID         30000
Default     TARGET         2



Class Variable Summary Statistics
(maximum 500 observations printed)

Data Role=TRAIN

                                  Number
Data                                of                       Mode               Mode2
Role     Variable Name    Role    Levels   Missing   Mode   Percentage   Mode2  Percentage

TRAIN    IMP_Education    INPUT      7        0        2      48.57        1      36.01
TRAIN    IMP_Gender       INPUT      2        0        2      62.86        1      37.14
TRAIN    Marital_Status   INPUT      4        0        2      53.21        1      45.53
TRAIN    card_class       INPUT      3        0        2      57.36        1      25.59
TRAIN    Default          TARGET     2        0        0      83.82        1      16.18



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

Data Role=TRAIN

Data     Variable                        Frequency
Role       Name      Role     Level        Count    Percent

TRAIN    Default    TARGET      0          25146     83.82
TRAIN    Default    TARGET      1           4854     16.18



Interval Variable Summary Statistics
(maximum 500 observations printed)

Data Role=TRAIN

                                              Standard      Non
Variable               Role        Mean      Deviation   Missing    Missing     Minimum     Median    Maximum    Skewness    Kurtosis

IMP_REP_Age            INPUT   35.49381     8.257783     30000         0          21    35.49381         79    0.829159    0.846927
IMP_REP_Apr_Bill       INPUT   1606.006     2350.156     30000         0    -5378.43      687.01   29243.94    2.741667    10.41682
IMP_REP_Jan_Bill       INPUT   1328.362     2028.583     30000         0    -11614.4      583.73   23938.08    2.757195    10.57072
IMP_REP_Jun_Bill       INPUT   1750.797     2511.881     30000         0    -5662.84      765.43   25541.04    2.621632    9.145886
IMP_REP_May_Bill       INPUT   1680.859     2427.135     30000         0    -2386.37      724.94   25443.77    2.653865    9.458473
REP_Apr_PayPercent     INPUT    0.32163     0.411748     30000         0           0      0.0617          1    0.921161    -1.01467
```

In Python, mean values were imputed for the interval attributes and mode values were imputed for the missing nominal attributes. After data cleaning and imputation, scaling of the interval attributes was done since the range of all the interval attributes are varying. This is done to make sure efficient running of Neural networks and to avoid any possible flaws in the feature selection techniques. Categorical attributes were encoded in Python by replacing each categorical value with one hot encoding.

```
 Replaced, Imputed and Encoded Data

        Age  Credit_Limit  Jun_Status  May_Status  Apr_Status  Mar_Sta
tus  \
0 -1.391899     -1.131883    1.794564    1.782348   -0.696663   -0.666
599
1 -1.149700     -0.366111   -0.874991    1.782348    0.138865    0.188
746
2 -0.180901     -0.591338    0.014861    0.111736    0.138865    0.188
746
3  0.182399     -0.906656    0.014861    0.111736    0.138865    0.188
746
4  2.604397     -0.906656   -0.874991    0.111736   -0.696663    0.188
746

   Feb_Status  Jan_Status  Jun_Bill  May_Bill     ...       Education1
  \
0   -1.530046   -1.486041 -0.643742 -0.648829     ...              0.0

1    0.234917    1.992316 -0.660503 -0.668231     ...              0.0

2    0.234917    0.253137 -0.298915 -0.494888     ...              0.0

3    0.234917    0.253137 -0.057224 -0.012891     ...              0.0

4    0.234917    0.253137 -0.579693 -0.612646     ...              0.0


   Education2  Education3  Education4  Education5  Marital_Status0  \
0         1.0         0.0         0.0         0.0              0.0
http://localhost:8888/notebooks/Midterm_656.ipynb#

3/6/2018                                     Midterm_656
1         1.0         0.0         0.0         0.0              0.0
2         1.0         0.0         0.0         0.0              0.0
3         1.0         0.0         0.0         0.0              0.0
4         1.0         0.0         0.0         0.0              0.0

   Marital_Status1  Marital_Status2  card_class0  card_class1
0              1.0              0.0          1.0          0.0
1              0.0              1.0          0.0          1.0
2              0.0              1.0          0.0          1.0
3              1.0              0.0          1.0          0.0
4              1.0              0.0          1.0          0.0
```

The cleaned, imputed, encoded and scaled data was then split for 10-fold cross validation which implies each time the model will be trained on 90% of the data and then the model will be assessed on the remaining 10% of the validation data or in other words, 90% of the data will be used for model fitting and remaining 10% data set will be held out for model assessment and this process will be carried out repeatedly until all the 10% held out test data has been utilized. The process for 10-fold cross validation is applied by running a loop. The screen shot of SAS code used is shown below:

Training Code

```
data MYLIB.selection;
call streaminit(12345);
set &em_import_data;
urand = rand('uniform');
proc sort data=MYLIB.selection;
by urand;
data &em_export_train;
drop fold_size urand;
set MYLIB.selection NOBS=nobs_;
fold_size = round(nobs_/ 10.0);
if _N_ <= fold_size then fold='A';
if _N_ > fold_size and _N_ <= 2*fold_size then fold='B';
if _N_ > 2*fold_size and _N_ <= 3*fold_size then fold='C';
if _N_ > 3*fold_size and _N_ <= 4*fold_size then fold='D';
if _N_ > 4*fold_size and _N_ <= 5*fold_size then fold='E';
if _N_ > 5*fold_size and _N_ <= 6*fold_size then fold='F';
if _N_ > 6*fold_size and _N_ <= 7*fold_size then fold='G';
if _N_ > 7*fold_size and _N_ <= 8*fold_size then fold='H';
if _N_ > 8*fold_size and _N_ <= 9*fold_size then fold='I';
if _N_ > 9*fold_size then fold='J';
run;
```

Training Code

```
data MyLib.temp1;
retain c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 0;
keep c1 c2 c3 c4 c5 c6 c7 c8 c9 c10;
set &em_import_data end=eof;
if fold='A' then c1=c1+1;
if fold='B' then c2=c2+1;
if fold='C' then c3=c3+1;
if fold='D' then c4=c4+1;
if fold='E' then c5=c5+1;
if fold='F' then c6=c6+1;
if fold='G' then c7=c7+1;
if fold='H' then c8=c8+1;
if fold='I' then c9=c9+1;
if fold='J' then c10=c10+1;
if eof then output;
data &em_export_validate;
drop c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 rfold;
retain rfold '0';
set MyLib.AllData_Train;
if rfold='0' then do;
   set MyLib.temp1;
   if c1=0 then rfold='A';
   if c2=0 then rfold='B';
   if c3=0 then rfold='C';
   if c4=0 then rfold='D';
   if c5=0 then rfold='E';
   if c6=0 then rfold='F';
   if c7=0 then rfold='G';
   if c8=0 then rfold='H';
   if c9=0 then rfold='I';
   if c10=0 then rfold='J';
end;
if fold=rfold then output;
run;
```

Model Fitting & Assessment

Since this is a classification problem, the modelling techniques applied to classify the customer into the default/non-default category are as follows:

1. Logistic Regression
2. Decision Tree
3. Neural Networks
4. Random Forest.

The biggest challenge was to choose the best model amongst these and within each category, parameters such as depth of the tree, number of layers and perceptrons for neural network was varied to achieve the optimized model selection.

Selecting the most important model metrics

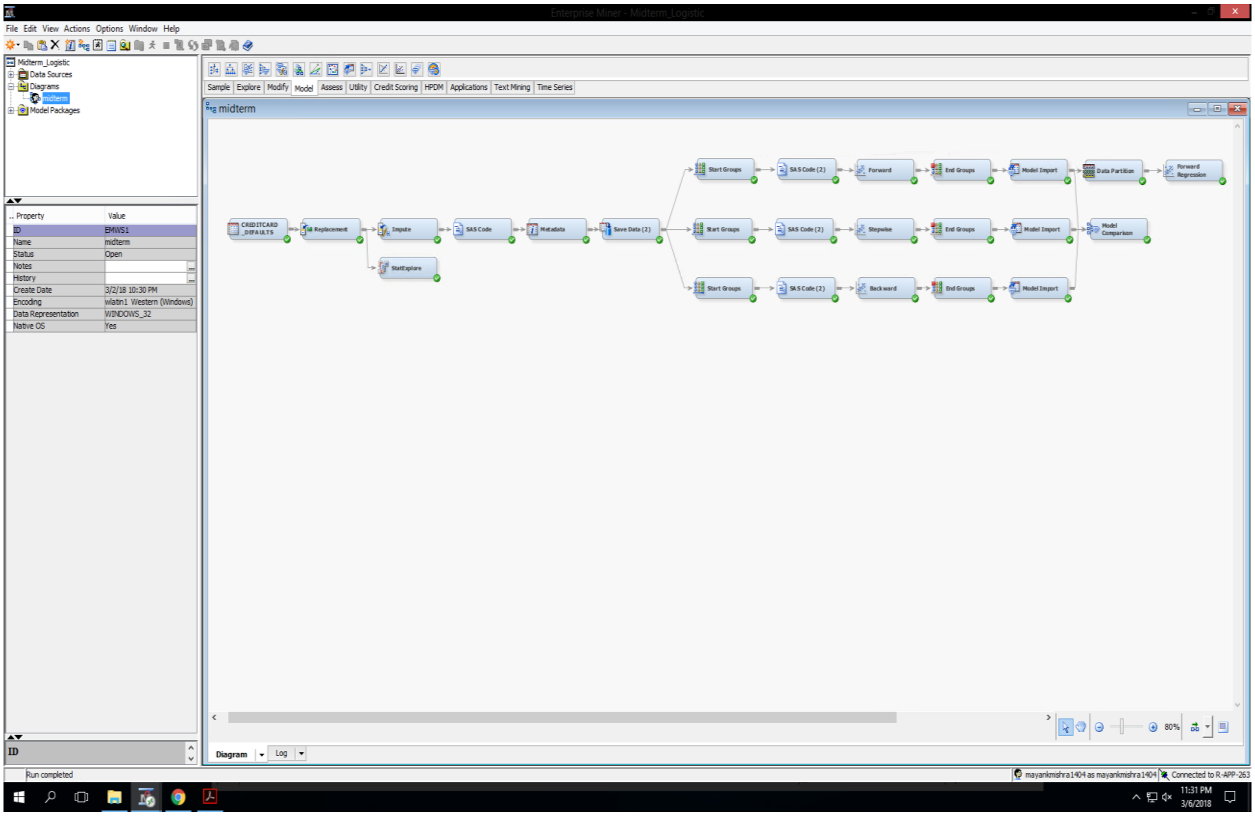
Since this data is from a bank, extra care would have been taken to particularly avoid incorrectly classifying a customer who will default, whereas incorrectly classifying a customer who will not default would be less problematic and would have less implications on the result. In other words, the value of False negatives has to be minimum and in order to achieve that, sensitivity/recall values have to optimized.
Now, for optimizing the values of sensitivity/recall, we used different methods for various classification models adopted.

For classifications models like Decision trees and Neural Networks, the parameters such as depth of the tree and layers of the neural network or the number of perceptrons in each layer of the neural networks were tuned. While optimizing the above-mentioned parameters of the respective classification model, the highest sensitivity/recall values were considered in selecting the best model while keeping in mind that the difference between training misclassification rate and the test misclassification rate should not be high enough to indicate overfitting of the model. This will in turn give poor results when the model will be used for a completely new data set in the future.

The dataset given presents us the problem of imbalanced classification since only 4584 out of 30000 customers are in default class which is 15.28% of the total number of customers. In such a scenario when one class outnumbers other class by large proportion, the machine learning algorithm doesn't get necessary information about the minority class so the TPR (True Positive Rate) values or the sensitivity/recall value would come out to be low until the case boosting of the minority class using the oversampling or any other method is carried out in order to eliminate the bias for the majority class ('not default' in this case).

Logistic Regression:

This model classifies the customers into a default or non-default class by modelling the probability that the customer belongs to a particular category. In SAS EM, forward, backward and stepwise variable selection methods were carried out to find out the best logistics regression model.



	Logistic Forward	Logistic Stepwise	Logistic Backward
MISC	0.136888889	0.136925926	0.138518519
sensivity/TPR	0.29532967	0.295100733	0.284340659
specificity	0.972693531	0.972693531	0.972870272
FPR	0.027306469	0.027306469	0.027129728
Precision	0.676100629	0.675930781	0.669181034
Accuracy	0.863111111	0.863074074	0.861481481
F1	0.411089866	0.410836653	0.399100257

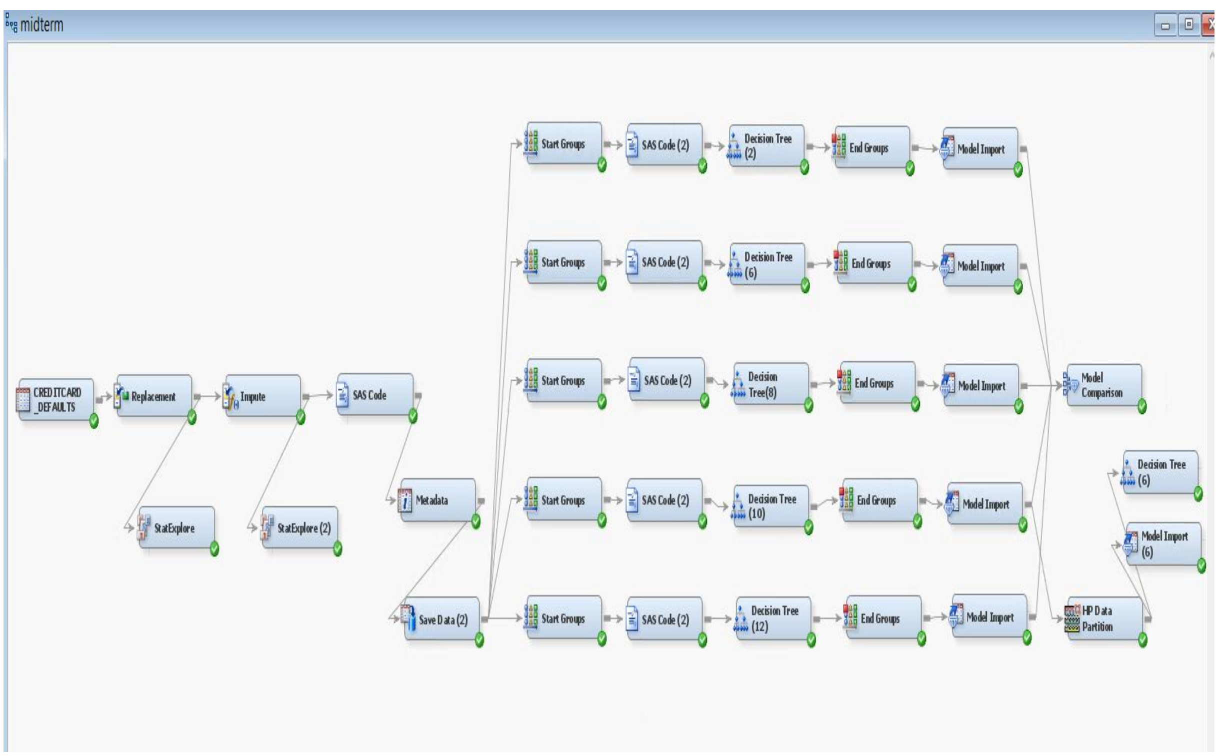
From the table, we observe that the **forward logistic regression is the best logistic regression model.**

Python output for logistic regression:

```
******** Logistic Regression ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8616    0.0033
recall....... 0.2829    0.0233
precision.... 0.6750    0.0367
f1........... 0.3976    0.0204
```

Decision Trees:

In this model, the data is split into branch like segments to form an inverted tree with root node at the top of the tree. Decision trees of two branches and different depths were tested in order to get the best model. The SAS code taught during lecture sessions were adopted to cross validate the data. Below is the SAS diagram used.



The model metrics for the various Decision Tree classification model are as follows:

	2 Branch 3 depth	2 Branch 6 depth	2 Branch 8 depth	2 Branch 10 depth	2 Branch 12 depth
MISC	0.125	0.120	0.117	0.113	0.111
sensivity/Recall	0.417	0.458	0.465	0.478	0.484
specificity	0.96	0.960	0.962	0.964	0.966
FPR	0.036	0.039	0.037	0.035	0.033
Precision	0.686	0.693	0.707	0.723	0.736
Accuracy	0.874	0.879	0.882	0.886	0.888
Recall	0.417	0.458	0.465	0.478	0.484
F1	0.519	0.551	0.561	0.576	0.584

From the model metrics, it is clear that the decision tree with 2 branches and depth=12 has the highest value of sensitivity/recall which is the most important parameter from the metrics. But when the misclassification error rate for the training data set and validation dataset were compared for the decision tree model of depth 12 are compared then it was observed that the model was overfitting the data. Therefore, **depth 10 was selected as the final best model for the decision tree in SAS EM**.

The output of the various decision tree models in Python are as follows:

```
**************** Decision Trees ****************

******** Decision Tree with Max_Depth =  1  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8732    0.0076
recall....... 0.4306    0.0330
precision.... 0.6686    0.0401
f1........... 0.5232    0.0314

******** Decision Tree with Max_Depth =  2  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8741    0.0065
recall....... 0.4081    0.0309
precision.... 0.6881    0.0386
f1........... 0.5116    0.0284

******** Decision Tree with Max_Depth =  3  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8750    0.0068
recall....... 0.4156    0.0309
precision.... 0.6894    0.0402
f1........... 0.5178    0.0291

******** Decision Tree with Max_Depth =  4  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8748    0.0077
recall....... 0.4565    0.0333
precision.... 0.6664    0.0408
f1........... 0.5410    0.0299

******** Decision Tree with Max_Depth =  5  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8741    0.0071
recall....... 0.4479    0.0403
precision.... 0.6675    0.0449
f1........... 0.5344    0.0301
```

```
******** Decision Tree with Max_Depth =  6  ********
Metric.......  Mean    Std. Dev.
```

ttp://localhost:8888/notebooks/Midterm_656.ipynb#

3/6/2018 Midterm_656

```
accuracy..... 0.8751    0.0067
recall....... 0.4516    0.0386
precision.... 0.6708    0.0368
f1........... 0.5386    0.0304

******** Decision Tree with Max_Depth =  7  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8738    0.0068
recall....... 0.4438    0.0355
precision.... 0.6673    0.0384
f1........... 0.5318    0.0280

******** Decision Tree with Max_Depth =  8  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8723    0.0072
recall....... 0.4425    0.0408
precision.... 0.6575    0.0382
f1........... 0.5278    0.0323

******** Decision Tree with Max_Depth =  9  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8712    0.0060
recall....... 0.4429    0.0296
precision.... 0.6508    0.0337
f1........... 0.5263    0.0236

******** Decision Tree with Max_Depth =  10  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8701    0.0084
recall....... 0.4444    0.0323
precision.... 0.6453    0.0476
f1........... 0.5252    0.0287
```

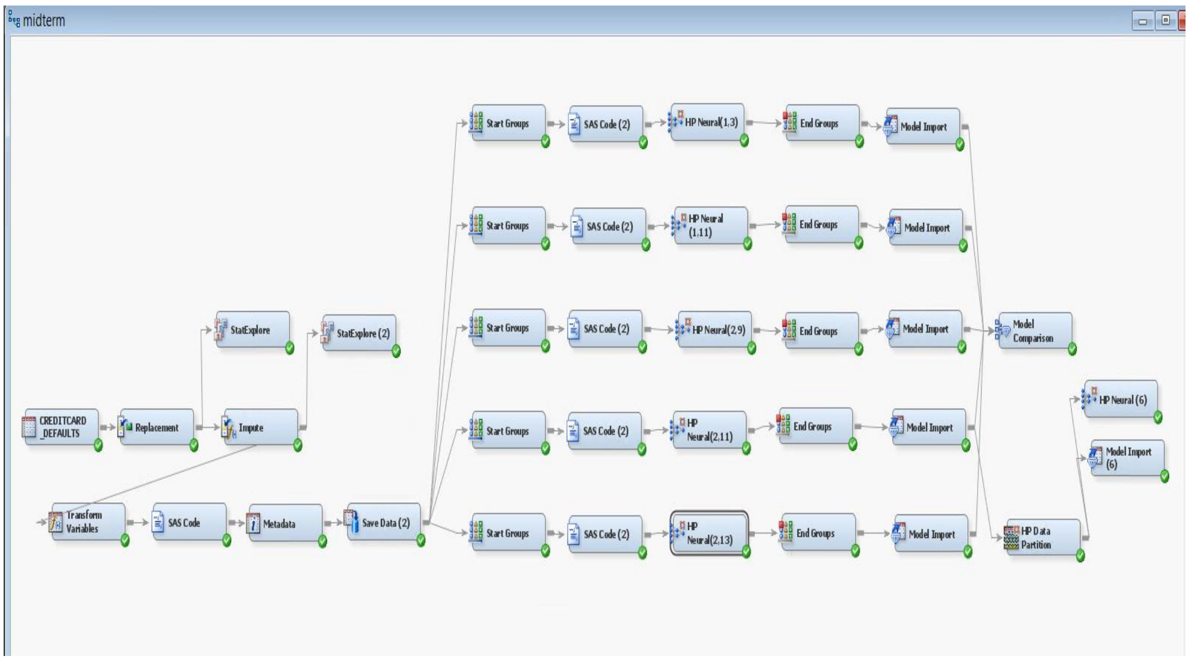
The decision tree with maximum depth=4 is the best model since it has the highest recall value.

Neural Networks:

These are a class of parametric models that accommodates a wider variety of nonlinear relationships between a set of predictors and a target variable. The network was defined with 3,5,7,9,11 perceptrons as instructed in the problem statement. The modeling was done using HP and non-HP nodes.

HP Neural Network:

In non-HP method, maximum of 2 layers and 13 neurons were adopted to model the data. Below is the diagram that was used in SAS enterprise miner.

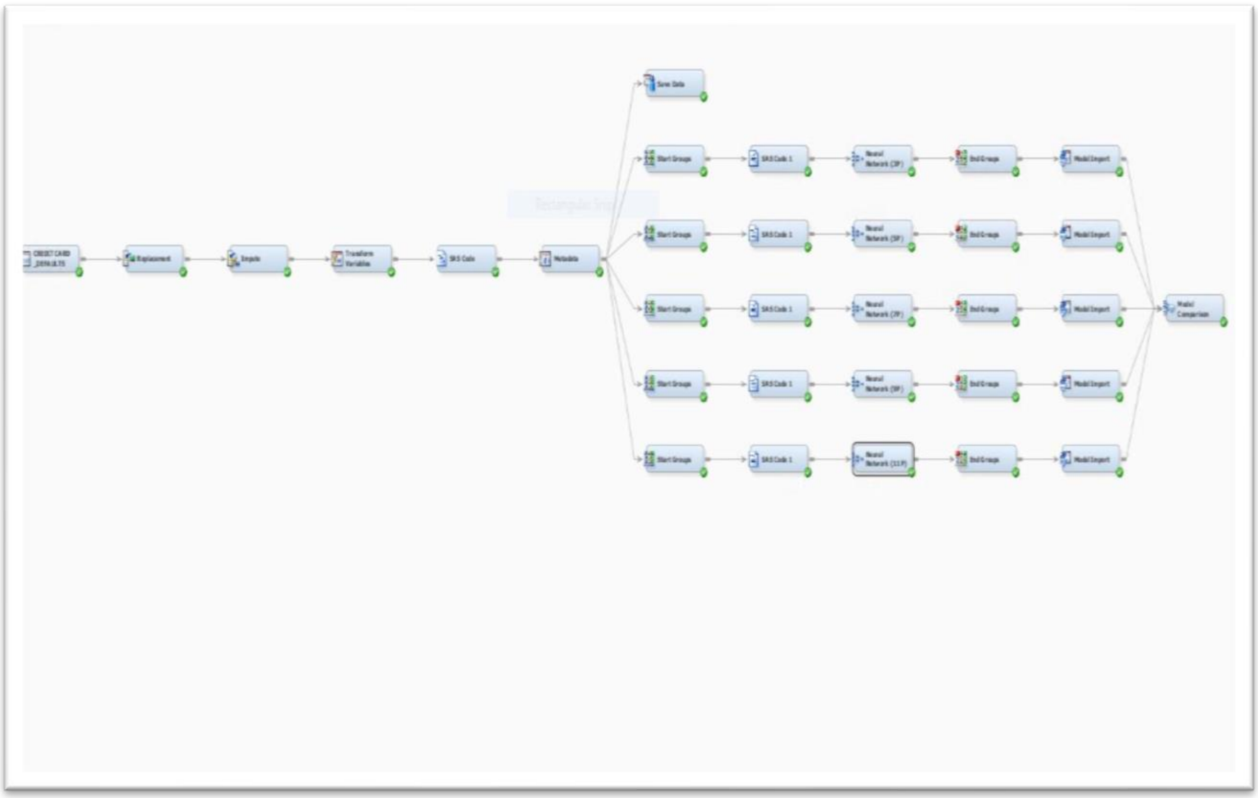


The model metrics for the various HP Neural Network classification model are shown below

	1 Layer 3 Neurons	1 Layer 11 Neurons	2 Layer 9 Neurons	2 Layer 11 Neurons	2 Layer 13 Neurons
MISC	0.136	0.135	0.135	0.136	0.138
Sensivity/Recall	0.414	0.393	0.436	0.441	0.427
specificity	0.949	0.955	0.946	0.945	0.945
FPR	0.050	0.044	0.053	0.054	0.054
Precision	0.614	0.629	0.612	0.609	0.601
Accuracy	0.863	0.864	0.864	0.863	0.861
Recall	0.414	0.393	0.436	0.441	0.427
F1	0.495	0.484	0.509	0.511	0.500

It is clear from the model metrics table that the sensitivity/recall, specificity, precision, accuracy, recall and F1 values are highest as well as FPR is the lowest for neural network with **2 layers and 11 neurons** therefore that neural network model is selected as the best neural network model in SAS EM.

Non-HP Neural Network:
In non-HP model, 3,5,7,9,11 perceptrons were adopted and the model was trained with 20 iterations and 30 runs with maximum run time for each node set to 1 hour.



The model metrics for the various non-HP Neural Network classification model are as shown below:

	3 Neurons	5 Neurons	7 Neurons	9 Neurons	11 Neurons
MISC	0.130	0.127	0.129	0.130	0.129
sensivity/Recall	0.388	0.415	0.398	0.380	0.388
specificity	0.963	0.961	0.963	0.964	0.964
Precision	0.666	0.672	0.674	0.672	0.673
Accuracy	0.870	0.873	0.872	0.870	0.871
F1	0.491	0.513	0.500	0.486	0.492
FPR	0.612	0.585	0.602	0.620	0.612

Comparing both the metric table, it is clearly evident that the HP neural network outperforms the non-HP neural network so we do not consider non-HP neural network while considering the final best model.

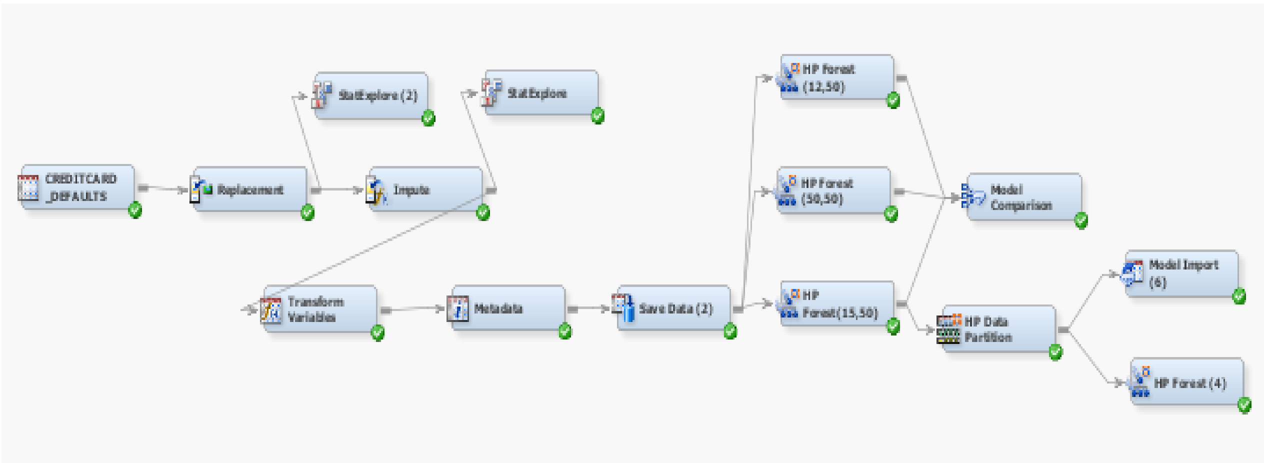
Output of different neural networks in python are as follows:-

```
**************** Neural Networks ****************

******** NEURAL NETWORK; 1 hidden layer with 3 perceptrons********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8720    0.0061
recall....... 0.4155    0.0346
precision.... 0.6707    0.0397
f1........... 0.5118    0.0270

******** NEURAL NETWORK; 1 hidden layer with 11 perceptrons********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8708    0.0075
recall....... 0.4429    0.0386
precision.... 0.6487    0.0388
f1........... 0.5253    0.0311

******** NEURAL NETWORK; 2 hidden layer with 5 & 4 perceptrons********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8719    0.0069
recall....... 0.4473    0.0326
precision.... 0.6553    0.0447
f1........... 0.5302    0.0238

******** NEURAL NETWORK; 2 hidden layer with 6 & 5 perceptrons********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8731    0.0063

p://localhost:8888/notebooks/Midterm_656.ipynb#

/2018                                         Midterm_656
recall....... 0.4627    0.0397
precision.... 0.6539    0.0363
f1........... 0.5405    0.0279

******** NEURAL NETWORK; 2 hidden layer with 7 & 6 perceptrons********

Metric.......  Mean    Std. Dev.
accuracy..... 0.8702    0.0064
recall....... 0.4524    0.0340
precision.... 0.6421    0.0361
f1........... 0.5295    0.0247
```

The best neural network model from python is the one with 2 hidden layers and 6&5 perceptrons.

Random Forest

Various Random Forest models were built by changing the number of depth and maximum number of trees.



Following are the model metrics-

Model Metrics	12Trees 50Depth	50Trees 50Depths	15Trees 50Depths
MISC	0.130	0.162	0.129
Sensivity/Recall	0.368	0.204	0.369
specificity	0.966	0.932	0.967
FPR	0.033	0.067	0.032
Precision	0.676	0.310	0.686
Accuracy	0.869	0.837	0.870
Recall	0.368	0.204	0.369
F1	0.476	0.246	0.480

From the metrics table it is observed that the Random Forest model with depth=50 and maximum number of trees=15 is the best model in SAS EM since the sensitivity/recall, specificity, precision, accuracy, recall and F1 values are the highest while the misclassification error rate and FPR values are the lowest.

The corresponding output of various random forest models in python are as follows:

```
**************** Random Forests ****************

********Number of Trees:  10  Max_features:  auto  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8715    0.0045
recall....... 0.3943    0.0323
precision.... 0.6787    0.0309
f1........... 0.4977    0.0244

********Number of Trees:  10  Max_features:  0.3  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8739    0.0070
recall....... 0.4028    0.0331
precision.... 0.6906    0.0441
f1........... 0.5078    0.0296

********Number of Trees:  10  Max_features:  0.5  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8776    0.0051
recall....... 0.4269    0.0316
precision.... 0.7017    0.0343
f1........... 0.5297    0.0235

********Number of Trees:  10  Max_features:  0.7  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8761    0.0058
recall....... 0.4209    0.0377
precision.... 0.6943    0.0344
f1........... 0.5229    0.0294

********Number of Trees:  15  Max_features:  auto  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8746    0.0051
recall....... 0.4425    0.0414
precision.... 0.6722    0.0297
f1........... 0.5321    0.0284

********Number of Trees:  15  Max_features:  0.3  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8766    0.0059
recall....... 0.4551    0.0331
precision.... 0.6785    0.0359
f1........... 0.5437    0.0238
```

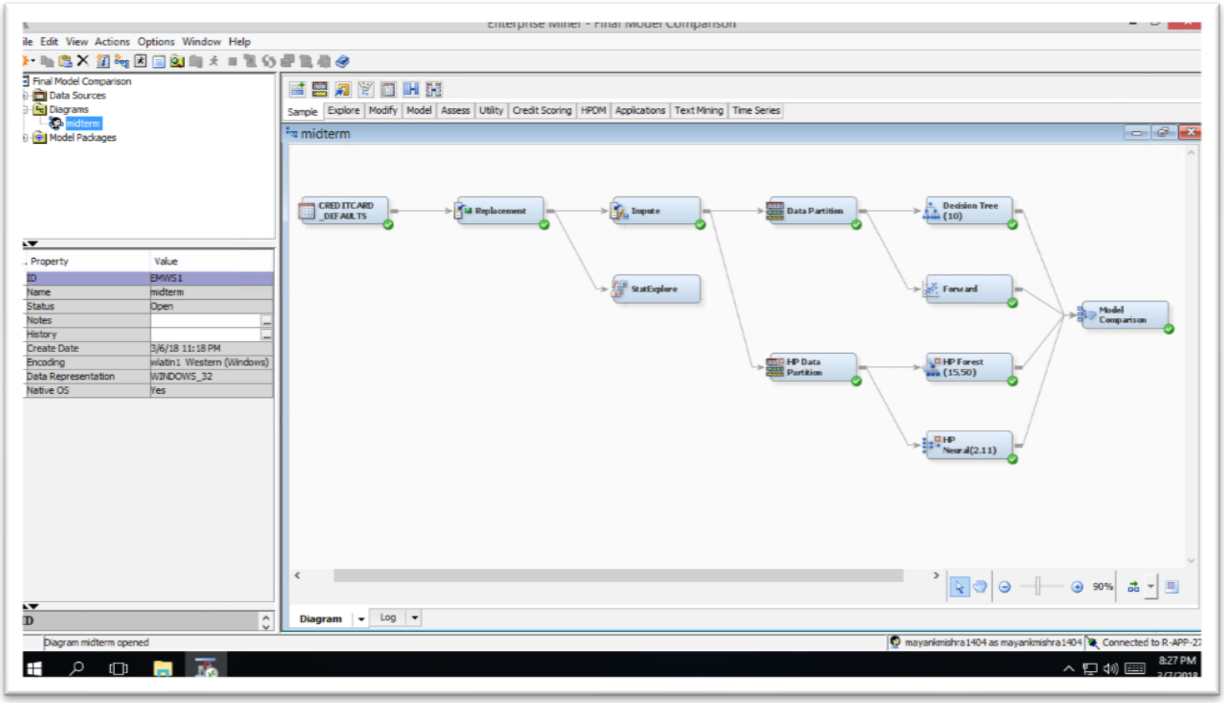
```
********Number of Trees:  15  Max_features:  0.5  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8800    0.0064
```



```
recall....... 0.4724    0.0345
precision.... 0.6911    0.0402
f1........... 0.5598    0.0242

********Number of Trees:  15  Max_features:  0.7  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8777    0.0063
recall....... 0.4679    0.0318
precision.... 0.6781    0.0358
f1........... 0.5528    0.0243

********Number of Trees:  20  Max_features:  auto  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8770    0.0057
recall....... 0.4244    0.0337
precision.... 0.6990    0.0374
f1........... 0.5270    0.0263

********Number of Trees:  20  Max_features:  0.3  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8789    0.0061
recall....... 0.4364    0.0357
precision.... 0.7044    0.0361
f1........... 0.5378    0.0290

********Number of Trees:  20  Max_features:  0.5  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8810    0.0061
recall....... 0.4485    0.0345
precision.... 0.7112    0.0393
f1........... 0.5489    0.0265

********Number of Trees:  20  Max_features:  0.7  ********
Metric.......  Mean    Std. Dev.
accuracy..... 0.8800    0.0060
recall....... 0.4510    0.0295
precision.... 0.7020    0.0336
f1........... 0.5485    0.0254
```

The best model from the above metric values appears to be the random forest model with maximum number of trees=15 and max number of features=0.7.

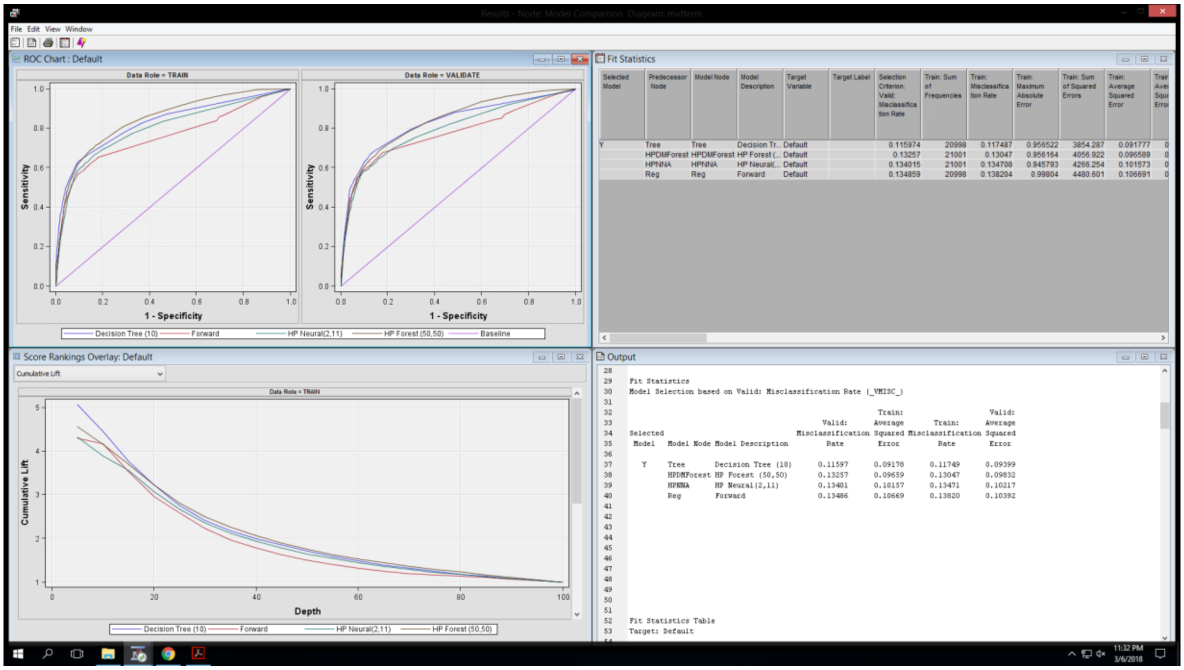
Model Comparison:



Comparing the various classification model from the SAS EM, we get the following table:-

Model Metrics	Decision Tree(2 branch, depth=10)	Neural Network(2 Layer 11 Neurons)	Random Forest(15 Tree depth=50)
MISC	0.113	0.136	0.129
sensivity/Recall	0.478	0.441	0.369
specificity	0.964	0.945	0.967
FPR	0.035	0.054	0.032
Precision	0.723	0.609	0.686
Accuracy	0.886	0.863	0.870
F1	0.576	0.511	0.480

Therefore, Decision Tree with 2 branch and depth=10 comes out to be the winner amongst all the SAS EM models.



Comparing the model output in Python we get,

Model Metrics	Logistic Regression	Decision Tree(depth=4)	Neural Network(2 Layer 11 Neurons)	Random Forest(15 Tree max_features=0.7)
Accuracy	0.8616	0.8478	0.8731	0.8777
sensivity/Recall	0.2829	0.4565	0.4627	0.4679
Precision	0.6750	0.6664	0.6539	0.6781
F1	0.3976	0.5410	0.5405	0.5528

Clearly, the winner amongst the python model is the Random Forest model with 15 tree and maximum features=0.7

```
                           accuracy_mean  accuracy_std  recall_mean  recall_
std  \
Logistic regression             0.861601      0.003348     0.282865    0.023
252
DT_Max_Depth_1                  0.873202      0.007624     0.430596    0.033
007
DT_Max_Depth_2                  0.874135      0.006458     0.408136    0.030
860
DT_Max_Depth_3                  0.874968      0.006826     0.415555    0.030
882
DT_Max_Depth_4                  0.874835      0.007739     0.456550    0.033
350
DT_Max_Depth_5                  0.874068      0.007116     0.447891    0.040
262
DT_Max_Depth_6                  0.875135      0.006730     0.451605    0.038
587
DT_Max_Depth_7                  0.873835      0.006753     0.443765    0.035
521
DT_Max_Depth_8                  0.872268      0.007167     0.442534    0.040
824
DT_Max_Depth_9                  0.871168      0.006009     0.442946    0.029
625
DT_Max_Depth_10                 0.870102      0.008368     0.444391    0.032
```



```
319
NN_1L_3P                        0.872035      0.006059     0.415544    0.034
600
NN_1L_11P                       0.870768      0.007493     0.442949    0.038
604
NN_2L_5P_4P                     0.871935      0.006865     0.447272    0.032
622
NN_2L_6P_5P                     0.873068      0.006316     0.462731    0.039
735
NN_2L_7P_6P                     0.870168      0.006354     0.452411    0.034
040
RF_10T_autoF                    0.871534      0.004451     0.394331    0.032
310
RF_10T_0.3F                     0.873868      0.006997     0.402779    0.033
059
RF_10T_0.5F                     0.877601      0.005080     0.426882    0.031
594
RF_10T_0.7F                     0.876102      0.005831     0.420910    0.037
696
RF_15T_autoF                    0.874568      0.005076     0.442541    0.041
359
RF_15T_0.3F                     0.876602      0.005939     0.455108    0.033
065
RF_15T_0.5F                     0.880001      0.006400     0.472407    0.034
539
RF_15T_0.7F                     0.877668      0.006299     0.467876    0.031
757
RF_20T_autoF                    0.877001      0.005727     0.424409    0.033
707
RF_20T_0.3F                     0.878935      0.006149     0.436359    0.035
670
RF_20T_0.5F                     0.880968      0.006073     0.448510    0.034
469
```

Logistic regression	0.674993	0.036698	0.397554	0.020424
DT_Max_Depth_1	0.668596	0.040104	0.523210	0.031426
DT_Max_Depth_2	0.688117	0.038631	0.511633	0.028421
DT_Max_Depth_3	0.689393	0.040202	0.517834	0.029109
DT_Max_Depth_4	0.666419	0.040791	0.541039	0.029913
DT_Max_Depth_5	0.667509	0.044878	0.534378	0.030089
DT_Max_Depth_6	0.670811	0.036808	0.538575	0.030402
DT_Max_Depth_7	0.667265	0.038383	0.531785	0.027998
DT_Max_Depth_8	0.657498	0.038156	0.527823	0.032266
DT_Max_Depth_9	0.650849	0.033704	0.526327	0.023551
DT_Max_Depth_10	0.645271	0.047575	0.525212	0.028696
NN_1L_3P	0.670669	0.039727	0.511788	0.027022
NN_1L_11P	0.648689	0.038787	0.525281	0.031061
NN_2L_5P_4P	0.655312	0.044721	0.530217	0.023778
NN_2L_6P_5P	0.653860	0.036262	0.540513	0.027915
NN_2L_7P_6P	0.642102	0.036107	0.529507	0.024688
RF_10T_autoF	0.678669	0.030947	0.497670	0.024417
RF_10T_0.3F	0.690578	0.044135	0.507753	0.029581
RF_10T_0.5F	0.701685	0.034258	0.529700	0.023515
RF_10T_0.7F	0.694317	0.034401	0.522910	0.029407
RF_15T_autoF	0.672159	0.029679	0.532125	0.028449
RF_15T_0.3F	0.678518	0.035902	0.543672	0.023816
RF_15T_0.5F	0.691078	0.040197	0.559817	0.024174
RF_15T_0.7F	0.678069	0.035791	0.552766	0.024319
RF_20T_autoF	0.699006	0.037374	0.526987	0.026349
RF_20T_0.3F	0.704398	0.036116	0.537778	0.028981
RF_20T_0.5F	0.711205	0.039282	0.548896	0.026461
RF_20T_0.7F	0.702002	0.033648	0.548511	0.025374

Conclusion:

Comparison between the best SAS EM Model and best Python model:

Model Metrics	Decision Tree(2 branch, depth=10)	Random Forest(15 Tree max_features=0.7)
Accuracy	0.886	0.878
sensivity/Recall	0.479	0.468
Precision	0.723	0.678
F1	0.576	0.553

From the following table, we can conclude that the Decision Tree model of SAS EM with 2 branches and depth=10 is by far the best model with highest Sensitivity/recall, Accuracy, Precision and F1 values.

The Sensitivity/recall value of the best model is 0.4785 which implies that the model correctly predicts whether a customer is going to default 47.8% of the time when the customer actually defaults.
The best model has an accuracy of 88.60% which depicts that the true events are correctly classified 88.60% of the time.

Best Model Output shown in SAS EM: