TEXAS A&M UNIVERSITY

STAT 656 | Applied Analytics
Mid Term Exam | Final Report

GROUP 13

Vinay Sairam Sashank Bandhakavi

Hari Prasad Anbalagan

Qiongyu Shi

Nishu Kurup

Boyan Zhang

8th March, 2018

Table of Contents

List of Figures

List of Tables

1 Introduction

This document summarizes the problem, method and results for the midterm exam for the course STAT 656 (Applied Analytics). It involves the analysis a credit card database for a bank that experiences high number of credit card default. By using the machine learning techniques taught in class which includes the decision tree, regression, neural networks and random forest a model that successfully predicts the probability of default. Both SAS EM and Python were utilized for performing the analysis.

The team was chosen by the instructor and consisted of 5 members. The teams divided up the work and the final result is shown in the sections below.

2 General

2.1 Software and Systems

The following software were utilized during the implementation of this project.

1. Microsoft Word/Excel 2016: For documentation and tabulations
2. SAS Enterprise Miner Workstation 14.3
3. Python 3.6.3 (Anaconda Spyder IDE 3.2.4

3 Problem Statement

The problem statement for the mid term exam was uploaded to course ecampus website. The data consists of 30,000 bank records for credit card customers for a bank that experiences a high level of default. The target attributes is a binary outcome - 'default', indicating whether the customer defaulted on credit card balance (default=1) or not (default=0). The problem is to build and validate the best model for predicting the probability of a customer default based upon their payment record for the past 6 months and their demographic and other bank information. The techniques discussed in class should be used to analyse the probelm. The data should be preprocessed and then hyperparameter optimization should be used to configure the best solutions from:

1. logistic regression,
2. decision tree,
3. neural network, and
4. random forest

Once the best configuration is selected for each of these models, the best model among the 4 should be selected using a simple 70/30, train/validate, model comparison.

4 Project Management

For successful implementation of the project a project management plan was adopted. This involved creating a workflow procedure for both SAS and Python solutions. Frequent meetings were conducted to update each other of the progress and to examine any roadblocks towards the

successful implementation of the midterm. In addition to the group discussion board provided on ecampus, a Google team drive was also created to facilitate the easy exchange of files. A team group was also created in the messaging system GroupMe for discussions and exchange of ideas among team members.

The distribution of tasks was based on a division of labor methodology that leveraged the skills of the individual team members based on the level of experience in the relevant area . The allocation of manpower was based on skills, time required, interests and equitable loading on all members. Independent verification of the results were also performed to ensure data and analysis fidelity. A document version control was also adopted for ensuring that the latest information was always present in the relevant document.

5 Methodology

5.1 Business Condition

The False negatives should be given importance over false positives because, misclassification of non-defaulted users can be accepted but misclassification of defaulted users cannot be accepted. Hence sensitivity should have high importance in comparing various models over precision. Model can also be selected based on a combined metric of precision and sensitivity i.e. f1 score. Precision answers the following question: out of all the examples the classifier labeled as Default, what fraction were correct? On the other hand, recall answers: out of all the Default examples there were, what fraction did the classifier pick up? The best model will have relatively a high f1 score and high sensitivity score.

6 Exploratory Data Analysis

6.1 Data Exploration

The data seems to be unbalanced. Although this is not considered as a rare event problem. So, we forecasted a low sensitivity, high accuracy and high specificity scores.

Table 1 Proportion of target values in data set

Default (1)	No-Default (0)
0.1618	0.8382

6.2 Correlation Matrix

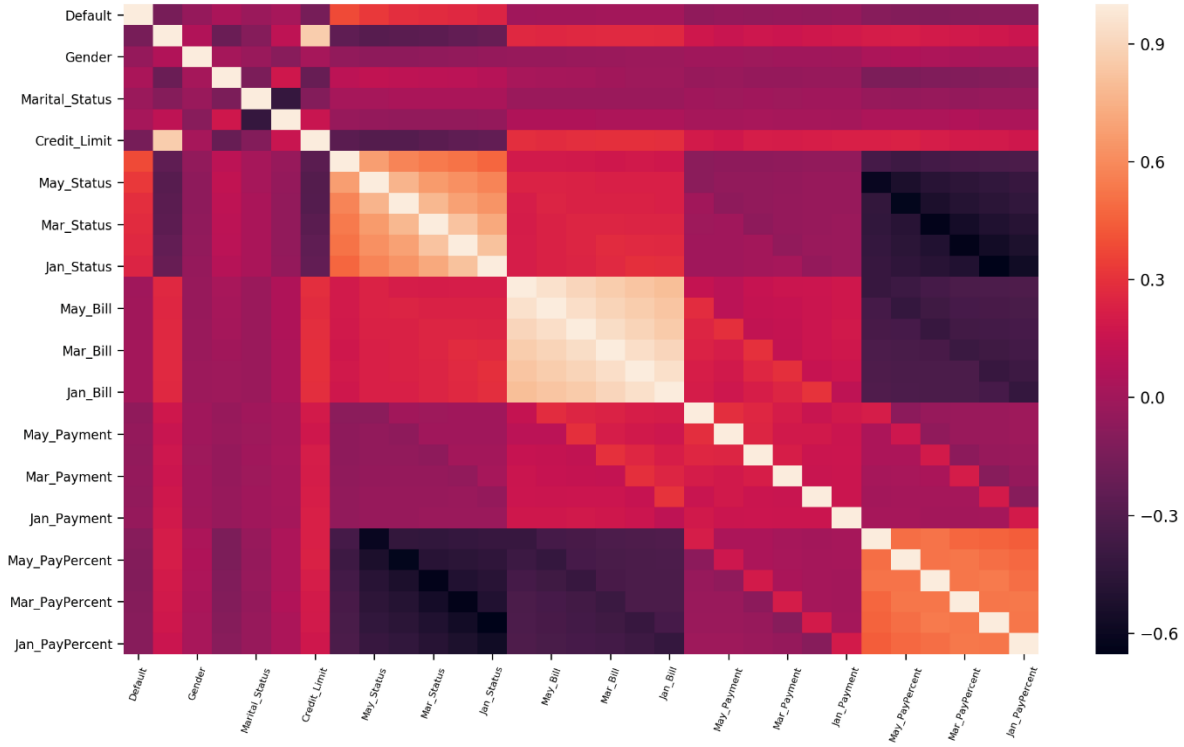

Figure 1: Correlation Matrix

The correlation matrix above shows that there is a high correlation among Jan_Bill, Feb_Bill, Mar_Bill, Apr_Bill, May_Bill, and Jun_Bill. Also, it shows high correlation among Jan_Status, Feb_Status, Mar_Status, Apr_Status, May_Status, and Jun_Status. And there are few other highly correlated pairs. It can be said that when a forward step wise regression is done on the data set, the final subset will not contain these highly correlated variables. Decision Trees, Random Forest, & Neural Networks will handle any correlations present. So, no need to create any subsets. Subsets should be created for these models to only reduce the processing time.

7 SAS EM Solution

The SAS EM solution consists of data preprocessing to ensure that the missing values and outliers were replaced and imputed. This was followed by the analysis procedures for each of the 4 techniques to be used

7.1 Pre-processing

The data is imported using 'INPUT DATA' node in SAS EM (A new library is to be created and the SAS data file to be loaded in the same location). As the variable named customer is an ID, the type is changed to ID on importing and is rejected from analysis in the nodes to the right.

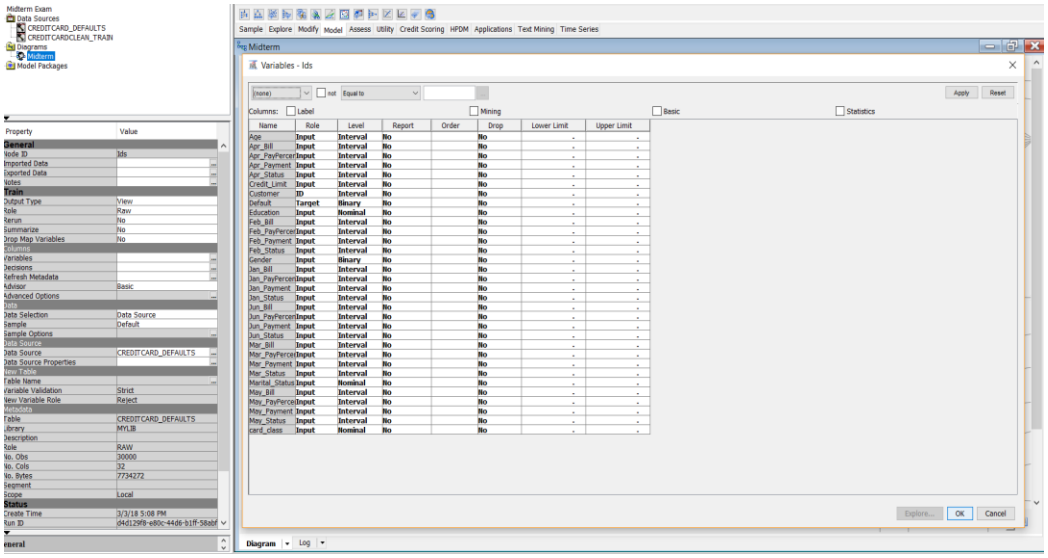

Figure 2: Variable settings

The replacement node identifies outliers as per the data dictionary and changes them to missing. This is done for both the interval and class variables.

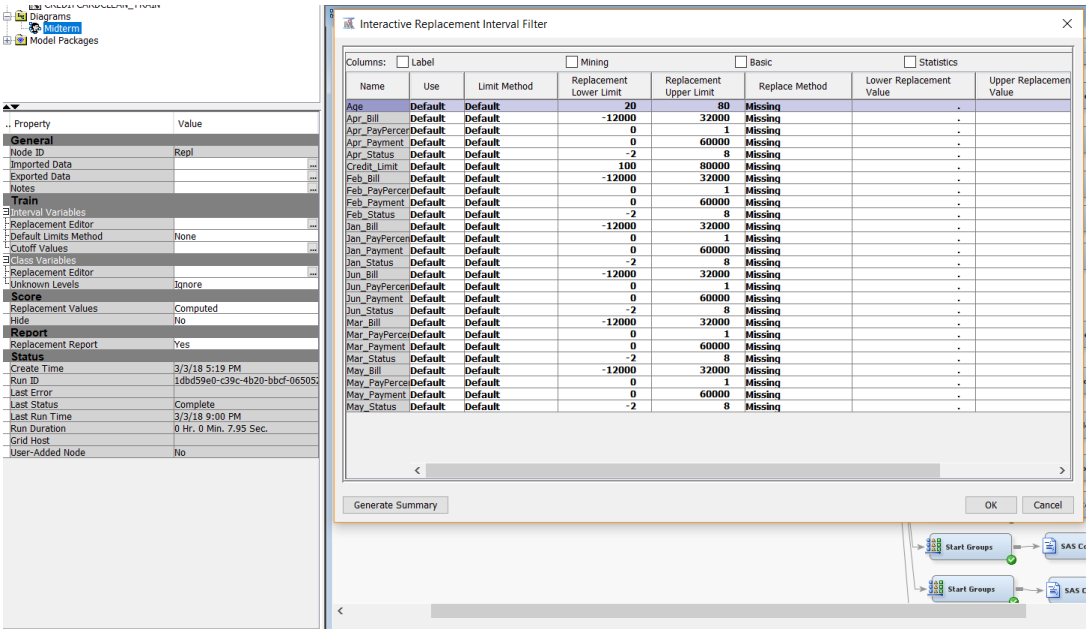

Figure 3: Define interval outliers

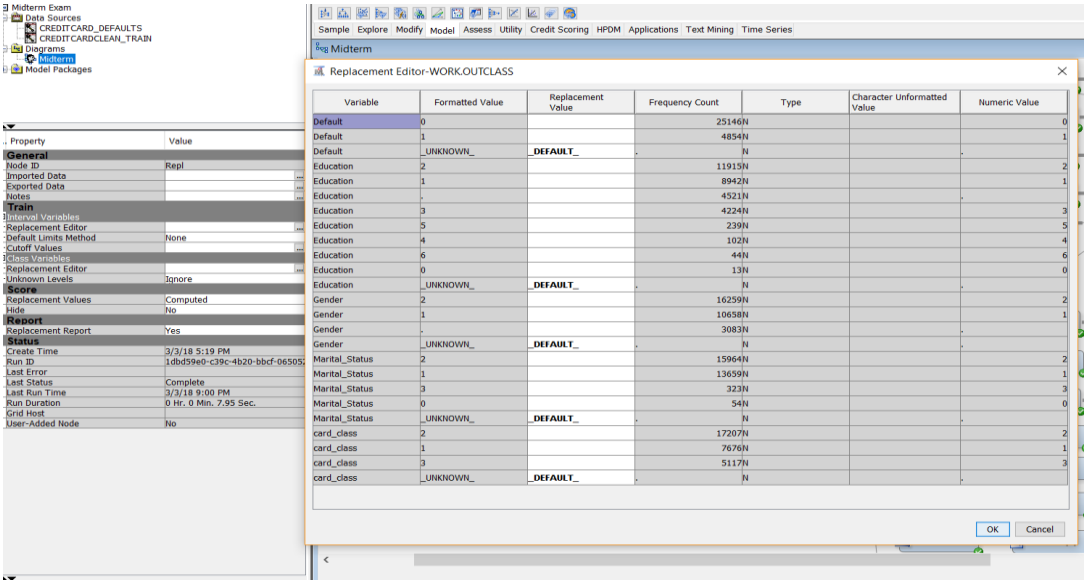

Figure 4: Class variables tab

Both the class variables and interval variables are then imputed using the impute node by setting the imputation method to tree for both the set of variables. The results after imputation are as below (obtained from STATEXPLORE RESULTS)

Variable	Role	Mean	Standard Deviation	Non Missing	Missing	Minimum	Median	Maximum	Skewness	Kurtosis
Apr_Bill	INPUT	1607.85	2371.749	30000	0	-5378.43	687.01	56911.84	3.08783	19.78325
Apr_PayPercent	INPUT	0.32163	0.411748	30000	0	0	0.0617	1	0.921161	-1.01467
Apr_Payment	INPUT	178.7184	602.1581	30000	0	0	61.56	30644.57	17.21664	564.3113
Apr_Status	INPUT	-0.1662	1.196868	30000	0	-2	0	8	0.840682	2.084436
Credit_Limit	INPUT	5725.52	4440.038	30000	0	300	4800	34200	0.991883	0.532664
Feb_Bill	INPUT	1378.65	2079.263	30000	0	-2781.62	619.16	31709.25	2.87638	12.30588
Feb_PayPercent	INPUT	0.340218	0.421385	30000	0	0	0.0602	1	0.81488	-1.21421
Feb_Payment	INPUT	164.1391	522.5181	30000	0	0	51.3	14587.29	11.12742	180.0639
Feb_Status	INPUT	-0.2662	1.133187	30000	0	-2	0	8	1.008197	3.989748
IMP_Age	INPUT	35.24038	8.547893	30000	0	21	34	79	0.806945	0.488004
Jan_Bill	INPUT	1329.414	2036.75	30000	0	-11614.4	583.73	32888.91	2.846645	12.2707
Jan_PayPercent	INPUT	0.361249	0.429917	30000	0	0	0.0661	1	0.706188	-1.38834
Jan_Payment	INPUT	178.3703	607.9893	30000	0	0	51.3	18080.38	10.64073	167.1614
Jan_Status	INPUT	-0.2911	1.149988	30000	0	-2	0	8	0.948029	3.426534
Jun_Bill	INPUT	1751.838	2518.346	30000	0	-5662.84	765.43	32986.28	2.663861	9.80629
Jun_PayPercent	INPUT	0.31336	0.397166	30000	0	0	0.0744	1	0.999406	-0.8222
Jun_Payment	INPUT	193.6945	566.4642	30000	0	0	71.82	29875.48	14.66837	415.2549
Jun_Status	INPUT	-0.0167	1.123802	30000	0	-2	0	8	0.731975	2.720715
Mar_Bill	INPUT	1479.593	2200.184	30000	0	-5814	651.58	30492.24	2.821965	11.30933
Mar_PayPercent	INPUT	0.31924	0.413873	30000	0	0	0.0531	1	0.92226	-1.01926
Mar_Payment	INPUT	165.0519	535.7827	30000	0	0	51.3	21238.2	12.90499	277.3338
Mar_Status	INPUT	-0.22067	1.169139	30000	0	-2	0	8	0.999629	3.496983
May_Bill	INPUT	1681.924	2434.143	30000	0	-2386.37	724.94	33650.44	2.705221	10.30295
May_PayPercent	INPUT	0.327227	0.40639	30000	0	0	0.0756	1	0.919102	-0.99937
May_Payment	INPUT	202.5039	787.9978	30000	0	0	68.71	57601.66	30.45382	1641.632
May_Status	INPUT	-0.13377	1.197186	30000	0	-2	0	8	0.790565	1.570418

Figure 5: STATEXPLORE results

7.2 Analysis of Techniques

The analysis of the different machine learning techniques along with the metrics are presented in this section. These include regression, decision tree, artificial neural networks and random forest.

7.2.1 Regression

For the regression analysis data partition is performed to the imputed data using the Data Partition Node and HP Data Partition Node. The Forward, Backward, Stepwise Regressions nodes are added for the Data Partition Node and HP Forward, HP Backward and HP Stepwise Data Partition Node and the results are compared using the Model Comparison Node.

Table 2 Confusion Matrix For Regression

Confusion Matrix	Validation Data Partition: 70/30					
	Logistic Regression Forward	Logistic Regression Stepwise	Logistic Regression Backward	HP Logistic Regression Forward	HP Logistic Regression Stepwise	HP Logistic Regression Backward
FN	1032	1032	1032	1049	1049	1048
TN	7364	7364	7365	7360	7361	7360
FP	181	181	180	184	183	184
TP	425	425	425	406	405	407

Table 3 Selection Metrics For Regression

Metrics	Validation Data Partition: 70/30					
	Logistic Regression Forward	Logistic Regression Stepwise	Logistic Regression Backward	HP Logistic Regression Forward	HP Logistic Regression Stepwise	HP Logistic Regression Backward
Recall	0.2917	0.2917	0.2917	0.2790	0.2785	0.2797
Precision	0.7013	0.7013	0.7025	0.6881	0.6888	0.6887
Accuracy	0.8653	0.8653	0.8654	0.8630	0.8631	0.8631
F1	0.2917	0.2917	0.2917	0.2790	0.2785	0.2797

7.2.2 Decision Trees

For the Decision Tree analysis, HP and Non-HP decision tree nodes are separately added and the tree depth is varied until the best output (based on metrics) is obtained based on Gini for nominal target attribute. Later the other parameters such as branches, leaf size, min categorical size are varied until the metrics no longer improve to identify the best model. The critical metrics are presented in the tables below.

Table 4 10 Fold Cross Validation Metrics for Several Depths

Metrics	Validation Data 10 Fold Cross Validation					
	Decision Tree Depth 12 Branch 2 Leaf Size 5 Category Size 5 Random Seed 12345	Decision Tree Depth 15 Branch 3 Leaf Size 5 Category Size 5 Random Seed 12345	Decision Tree Depth 12 Branch 2 Leaf Size 5 Category Size 5 Random Seed 12345	Decision Tree Depth 15 Branch 3 Leaf Size 5 Category Size 5 Random Seed 12345	Decision Tree Depth 20 Branch 4 Leaf Size 5 Category Size 5 Random Seed 12345	Decision Tree Depth 20 Branch 3 Leaf Size 5 Category Size 5 Random Seed 12345
Recall	0.4878	0.4878	0.4547	0.4475	0.4889	0.4936
Precision	0.7354	0.7354	0.6810	0.6788	0.7441	0.7423
Accuracy	0.8887	0.8887	0.8773	0.8763	0.8901	0.8903
F1	0.5866	0.5866	0.5453	0.5394	0.5901	0.5929

Table 5 10 Fold Cross Validation Metrics for Several Depths

Metrics	Validation Data 10 Fold Cross Validation			
	Decision Tree Depth 5 Branch 2 Leaf Size 5 Category Size 5 Random Seed 12345	Decision Tree Depth 6 Branch 2 Leaf Size 5 Category Size 5 Random Seed 12345	Decision Tree Depth 8 Branch 2 Leaf Size 5 Category Size 5 Random Seed 12345	Decision Tree Depth 10 Branch 2 Leaf Size 5 Category Size 5 Random Seed 12345
Recall	0.4479	0.4575	0.4674	0.4776
Precision	0.6858	0.6936	0.7088	0.7240
Accuracy	0.8774	0.8796	0.8827	0.8859
F1	0.5410	0.5511	0.5627	0.5749

Table 6 10 Fold Cross Validation Metrics for best DT for Several random seeds

Metrics	Validation Data Partition: 70/30					Range
	Decision Tree Depth 20 Branch 3 Leaf Size 3 Category Size 3 Random Seed 12345	Decision Tree Depth 20 Branch 3 Leaf Size 3 Category Size 3 Random Seed 123	Decision Tree Depth 20 Branch 3 Leaf Size 3 Category Size 3 Random Seed 5	Decision Tree Depth 20 Branch 3 Leaf Size 3 Category Size 3 Random Seed 100	Decision Tree Depth 20 Branch 3 Leaf Size 3 Category Size 3 Random Seed 123456	
MISC	0.1174	0.1169	0.1175	0.1159	0.1155	0.1155 - 0.1174
Sensitivity	0.5120	0.4839	0.4626	0.4842	0.4732	0.4732 - 0.5120
Specificity	0.9541	0.9602	0.9636	0.9613	0.9638	0.9638 - 0.9541
FPR	0.0459	0.0398	0.0364	0.0387	0.0362	0.0362 - 0.0459
Precision	0.6832	0.7015	0.7102	0.7071	0.7162	0.7162 - 0.6832
Accuracy	0.8826	0.8831	0.8825	0.8841	0.8845	0.8845 - 0.8826
F1	0.5853	0.5727	0.5603	0.5748	0.5699	0.5699 - 0.5853

7.2.3 Neural Networks

Neural Networks (HP and Non-HP) nodes are separately added to the diagram. For the Non-HP NN, the perceptron are varied until the best results are obtained. For HP NN, the number of hidden layers is set to two layers and the perceptron are varied until best results are obtained. The critical metrics are presented in the tables below.

Table 7 10 Fold Cross Validation Metrics for Several Perceptrons

Metrics	Validation Data Partition: 10 Fold Cross Validation			
	Neural Networks Layer 1 Perceptron 11	HP Neural Networks Layer 1 Perceptron 17	Neural Networks Layer 1 Perceptron 15	HP Neural Networks Layer 1 Perceptron 9
Recall	0.4368	0.4464	0.4230	0.4450
Precision	0.6456	0.6657	0.6430	0.6669
Accuracy	0.8701	0.8742	0.8686	0.8742
F1	0.5210	0.5345	0.5103	0.5338

7.2.4 Random Forest

For Random Forest analysis random forest nodes are added to the imputed nodes with varying proportions of each observation in each sample. The results are recorded, and the best proportion number is picked after comparing using the model comparison node. The critical metrics are presented in the tables below.

Table 8 10 Fold Cross Validation Metrics for Several Perceptrons

Metrics	Validation Data Partition: 10 Fold Cross Validation				
	Random Forest 0.8	Random Forest 0.9	Random Forest 0.5	Random Forest 0.6	Random Forest 0.7
Recall	0.4561	0.4586	0.4168	0.4159	0.4306
Precision	0.6823	0.6822	0.6802	0.6819	0.6866
Accuracy	0.8776	0.8778	0.8739	0.8741	0.8761
F1	0.5467	0.5485	0.5169	0.5167	0.5292

7.3 Comparison Of SAS Solutions

All the results of the various iterations of the cross-validation loops are then compared using model comparison node. The best results of the Decision Tree and Neural Network nodes are then applied to data partition node (created earlier) to study the results and identify the best model to be used. The comparison table for the different models are shown below.

Table 9 10 Comparison of different techniques for the 70/30 partition

Metrics	Validation Data Partition: 70/30			
	Decision Tree Depth 20 Branch 3 Leaf Size 3 Category Size 3 Random Seed 12345	**Random Forest** 0.9	**Neural Networks** Layer 2 Perceptron (8,9)	**Logistic Regression** Forward
MISC	0.1174	0.1222	0.1258	0.1347
Sensitivity	0.5120	0.4586	0.4464	0.2917
Specificity	0.9541	0.9588	0.9567	0.9760
FPR	0.0459	0.0412	0.0433	0.0240
Precision =	0.6832	0.6822	0.6657	0.7013
Accuracy	0.8826	0.8778	0.8742	0.8653
F1	0.5853	0.5485	0.5345	0.4120

8 Python Solution

The python solution is presented in this section. The python code was developed for all the techniques and is presented in Appendix A. The preprocessing and the techniques are presented and model comparison is performed to obtain the best model.

8.1 Pre-processing

An attribute map is created with key as the name of columns in the data frame. The first number indicates 0= interval, 1=binary, and 2=nominal. The 1st tuple for interval attributes is their lower

and upper bounds. The 1st tuple for categorical attributes is their allowed categories. The 2nd tuple contains the number of missing and no of outliers. Data preprocessing step is used to identify outliers and set missing values. Columns Gender, Education, and Age have missing values. Data preprocessing class was used to identify 1 outlier each for Jun_Bill, May_Bill, Apr_Bill, and Jan_Bill.

Table 10 List of Missing values and outliers

```
********** Data Preprocessing ***********
Features Dictionary Contains:
26 Interval,
2 Binary, and
3 Nominal Attribute(s).

Data contains 30000 observations & 31 columns.


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
```

8.1 Analysis of Techniques

The description of the different analysis performed and the results are presented in this section.

8.1.1 Logistic Regression

Forward Stepwise Regression was used to find the optimal subset of the dataset that gives the best prediction. The result of 10-fold cross validation is that the data set should be reduced to

the following columns 'Jun_Status', 'May_Status', 'Jun_PayPercent', 'card_class0'. The regression random state 12345 was used to do data modelling.

Table 11 Optimal Subset of dataset

Data Subsets	Validation Data Partition: 10-Fold Cross Validation Logistic Regression Forward Step wise F1 Score
Jun_Status	0.3156
Jun_Status, May_Status,	0.4208
Jun_PayPercent,	0.4236
Jun_PayPercent,	0.4282
MAX MEAN	0.4282
TN	7364.0000
FP	181.0000
TP	425.0000

8.1.2 Decision Trees

For the decision trees analysis, max_depths: 5,6,8,10,12 were used based on previous experience. Also, considering the amount of observations there are, the depth was increased to 50. The results show that max_depth has the largest influence upon metrics score while min_samples_split only has a little. When max_depth is below 20, the metrics score slightly increases with the increase in min_samples_leaf value. However, when the max_depth is larger than 20, metrics score decreases as the min_samples_leaf becomes larger. Since there isn't much difference between depth 20,25,and 50, there was no need to check with other depths. The best model can be *Maximum Tree Depth:25, Min_samples_leaf:3, Min_samples_split:5* or *Maximum Tree Depth:6, Min_samples_leaf:7, Min_samples_split:5* after considering F1 Score, Accuracy, & Recall.

Table 12 10 Fold Cross Validation Metrics for Several Depths

Validation Data Partition: 70/30									
Metrics	Decision Tree Depth 5 Leaf Size 3	Decision Tree Depth 5 Leaf Size 5	Decision Tree Depth 5 Leaf Size 7	Decision Tree Depth 6 Leaf Size 3	Decision Tree Depth 6 Leaf Size 5	Decision Tree Depth 6 Leaf Size 7	Decision Tree Depth 8 Leaf Size 3	Decision Tree Depth 8 Leaf Size 5	Decision Tree Depth 8 Leaf Size 7
Recall	0.4477	0.4479	0.4479	0.4498	0.4514	0.4524	0.4411	0.4411	0.4442
Precision	0.6671	0.6675	0.6671	0.6707	0.6710	0.6710	0.6548	0.6560	0.6560
Accuracy	0.8740	0.8741	0.8740	0.8750	0.8751	0.8751	0.8717	0.8719	0.8722
F1	0.5341	0.5344	0.5343	0.5372	0.5385	0.5390	0.5261	0.5263	0.5285
Metrics	Decision Tree Depth 10 Leaf Size 3	Decision Tree Depth 10 Leaf Size 5	Decision Tree Depth 10 Leaf Size 7	Decision Tree Depth 12 Leaf Size 3	Decision Tree Depth 12 Leaf Size 5	Decision Tree Depth 12 Leaf Size 7	Decision Tree Depth 15 Leaf Size 3	Decision Tree Depth 15 Leaf Size 5	Decision Tree Depth 15 Leaf Size 7
Recall	0.4411	0.4423	0.4434	0.4442	0.4446	0.4419	0.4425	0.4446	0.4500
Precision	0.6394	0.6453	0.6446	0.6131	0.6161	0.6201	0.5563	0.5682	0.5726
Accuracy	0.8688	0.8700	0.8700	0.8643	0.8648	0.8654	0.8525	0.8549	0.8563
F1	0.5209	0.5238	0.5245	0.5142	0.5154	0.5150	0.4924	0.4979	0.5031
Metrics	Decision Tree Depth 20 Leaf Size 3	Decision Tree Depth 20 Leaf Size 5	Decision Tree Depth 20 Leaf Size 7	Decision Tree Depth 25 Leaf Size 3	Decision Tree Depth 25 Leaf Size 5	Decision Tree Depth 25 Leaf Size 7	Decision Tree Depth 50 Leaf Size 3	Decision Tree Depth 50 Leaf Size 5	Decision Tree Depth 50 Leaf Size 7
Recall	0.4633	0.4502	0.4475	0.4660	0.4549	0.4493	0.4650	0.4541	0.4462
Precision	0.4936	0.5152	0.5274	0.4745	0.5069	0.5292	0.4872	0.5066	0.5280
Accuracy	0.8359	0.8420	0.8452	0.8296	0.8397	0.8457	0.8338	0.8397	0.8454
F1	0.4775	0.4800	0.4835	0.4697	0.4790	0.4854	0.4754	0.4785	0.4831

8.1.3 Neural Networks

For neural networks, the same strategy was followed for doing hyper parameter optimization. Hyper parameter values were chosen based on homework assignments. Later new values were added to improve the tuning. As show in results, there isn't much difference with different layers/perceptrons. It can be determined that (7,6) are the optimal hyper parameters by considering the recall and f1 score values.

Table 13 10 Fold Cross Validation Metrics for Several Layers and Perceptrons

Validation Data								
Metrics	Neural Networks Layer 1 Perceptron 3	Neural Networks Layer 1 Perceptron 17	Neural Networks Layer 2 Perceptron (5,4)	Neural Networks Layer 2 Perceptron (6,5)	Neural Networks Layer 2 Perceptron (7,6)	Neural Networks Layer 2 Perceptron (8,7)	Neural Networks Layer 2 Perceptron (9,8)	Neural Networks MAX MEAN
Recall	0.4397	0.4271	0.4419	0.4411	0.4565	0.4308	0.4458	0.4565
Precision	0.6629	0.6475	0.6597	0.6568	0.6431	0.6345	0.6300	0.6629
Accuracy	0.8729	0.8693	0.8724	0.8721	0.8704	0.8673	0.8677	0.8729
F1	0.5279	0.5137	0.5283	0.5265	0.5325	0.5116	0.5217	0.5325

8.1.4 Random Forest

For the random forest techniques hyper parameter optimization n_estimators (the number of trees created) and max_features (proportion of features used in the model) are chosen as these two are the most important and frequently parameters to adjust. A broad range was chosen for both parameters at the inception. As shown in results, metrics scores increased when n_estimators and max_features had increased. However, when the n_estimators was increased to 70, the metrics scores didn't show much difference from the one with 50 estimators. It is obvious since the total no of estimators is less than 50. So we stop it to increase more. "auto" in

max_feature usually gives good results, but it didn't so in this case. Finally, the best model contains *Number of Trees:70, Max_features:0.5* as it has high recall, accuracy, and f1 score value.

Table 14 10 Fold Cross Validation Metrics for Several Trees and Features

Metrics	Validation Data Partition: 10 Fold Cross Validation					
	Random Forest No of Trees 10 Features: Auto	Random Forest No of Trees 10 Features: 0.5	Random Forest No of Trees 10 Features: 0.7	Random Forest No of Trees 30 Features: Auto	Random Forest No of Trees 30 Features: 0.5	Random Forest No of Trees 30 Features: 0.7
Recall	0.3853	0.4173	0.4236	0.4252	0.4565	0.4609
Precision	0.6886	0.6940	0.6899	0.7018	0.7086	0.7111
Accuracy	0.8721	0.8758	0.8757	0.8776	0.8815	0.8821
F1	0.4932	0.5201	0.5240	0.5287	0.5544	0.5580
Metrics	**Random Forest** No of Trees 50 Features: Auto	**Random Forest** No of Trees 50 Features: 0.5	**Random Forest** No of Trees 50 Features: 0.7	**Random Forest** No of Trees 70 Features: Auto	**Random Forest** No of Trees 70 Features: 0.5	**Random Forest** No of Trees 70 Features: 0.7
Recall	0.4361	0.4654	0.4706	0.4444	0.4706	0.4695
Precision	0.7075	0.7102	0.7094	0.7088	0.7109	0.7090
Accuracy	0.8793	0.8826	0.8829	0.8803	0.8832	0.8827
F1	0.5385	0.5612	0.5648	0.5453	0.5653	0.5638

8.2 Comparison

The entire data was split in the ratio train: validation - 70:30. The best models that were obtained after 10-folds cross validation were trained on the train dataset and later were applied on validation datasets. The logistic regression is not chosen as the best model even though it has high interpretability because its metrics are low compared to Neural Network metrics. The results also showed that both Decision tree (depth=20, min_samples_leaf=3) and Random forest (n_estimator=60, max_feature=0.5) didn't do better than Neural-Network (two-layers 7,6). Neural Network has the highest recall, accuracy and f1 score. As it can identify more defaults it is the best model.

Table 15 Comparison of Model Metrics

Metrics	Validation Data Partition: 70/30			
	Logistic Regression (Jun_Status, May_Status, Jun_PayPercent, card_class0)	**Decision Tree** Depth 20 Leaf Size 3	**Random Forest** No of Trees 60 Features: 0.5	**Neural Networks** Layer 2 Perceptron (7,6)
Recall	0.4210	0.4599	0.4710	0.5115
Precision	0.6748	0.4612	0.7031	0.7367
Accuracy	0.8754	0.8290	0.8841	0.8931
F1	0.5178	0.4605	0.5641	0.6040

9 Discussion of Results and Conclusion

It is seen that after imputation, the percent of defaulters in the data set is about 16%.

Table 16 Comparison of Model Metrics with different seeds

Data Role	Variable Name	Role	Level	Frequency Count	Percent
TRAIN	Default	TARGET	0	25146	83.82
TRAIN	Default	TARGET	1	4854	16.18

Given this, in a data set of 30000 there is almost a 1:5 distribution between defaulters and non-defaulters. Due to the skewed distribution of the data, predicting the best model just by average misclassification rate as even a bad model predicting all 0, has a good chance of producing a MISC of 16%. Hence to determine the best model, we will go by how well the model predicts defaulters (TP) which is measured by sensitivity, F1 score and precision.

Based on the the sensitivity of the models and F1 score the decision tree model with the parameters (depth=2,branch=2 and leaf size=2) appears to be the best model (with lowest MISC as well) with random forests appearing very close on these parameters to the decision tree output.

However interestingly on varying the random seed number the sensitivity fluctuated significantly for decision tree model .The results are tabulated below for both SAS and Python.

Table 17 Comparison of Model Metrics with different seeds (SAS)

Metrics	Validation Data Partition: 70/30					Range
	Decision Tree	**Decision Tree**	**Decision Tree**	**Decision Tree**	**Decision Tree**	
	Depth 20 Branch 3 Leaf Size 3 Category Size 3 Random Seed 12345	Depth 20 Branch 3 Leaf Size 3 Category Size 3 Random Seed 123	Depth 20 Branch 3 Leaf Size 3 Category Size 3 Random Seed 5	Depth 20 Branch 3 Leaf Size 3 Category Size 3 Random Seed 100	Depth 20 Branch 3 Leaf Size 3 Category Size 3 Random Seed 123456	
MISC	0.1174	0.1169	0.1175	0.1159	0.1155	0.1155 - 0.1174
Sensitivity	0.5120	0.4839	0.4626	0.4842	0.4732	0.4732 - 0.5120
Specificity	0.9541	0.9602	0.9636	0.9613	0.9638	0.9638 - 0.9541
FPR	0.0459	0.0398	0.0364	0.0387	0.0362	0.0362 - 0.0459
Precision	0.6832	0.7015	0.7102	0.7071	0.7162	0.7162 - 0.6832
Accuracy	0.8826	0.8831	0.8825	0.8841	0.8845	0.8845 - 0.8826
F1	0.5853	0.5727	0.5603	0.5748	0.5699	0.5699 - 0.5853

A comparison of models with different random seeds was performed and it is seen that the Neural Network is the best predictive model.

Table 18 Comparison of Model Metrics with different seeds (Python)

Validation Data Partition: 70/30									
Metrics	Decision Tree random seed 5	Neural Network random seed 5	Random Forest random seed 5	Decision Tree random seed 7	Neural Network random seed 7	Random Forest random seed 7	Decision Tree random seed 12	Neural Network random seed 12	Random Forest random seed 12
Recall	0.4477	0.4479	0.4479	0.4498	0.4514	0.4524	0.4411	0.4411	0.4442
Precision	0.6671	0.6675	0.6671	0.6707	0.6710	0.6710	0.6548	0.6560	0.6560
Accuracy	0.8740	0.8741	0.8740	0.8750	0.8751	0.8751	0.8717	0.8719	0.8722
F1	0.5341	0.5344	0.5343	0.5372	0.5385	0.5390	0.5261	0.5263	0.5285
Metrics	Decision Tree random seed 123	Neural Network random seed 123	Random Forest random seed 123	Decision Tree random seed 12345	Neural Network random seed 12345	Random Forest random seed 12345			
Recall	0.4411	0.4423	0.4434	0.4442	0.4446	0.4419			
Precision	0.6394	0.6453	0.6446	0.6131	0.6161	0.6201			
Accuracy	0.8688	0.8700	0.8700	0.8643	0.8648	0.8654			
F1	0.5209	0.5238	0.5245	0.5142	0.5154	0.5150			

The range of sensitivities however are still better than the best model of random forest and neural network in SAS. This makes it the best model from SAS irrespective of the Random Seed value.

9.1 BEST MODEL SELECTION

As python can generate a neural network output with sensitivities varying from 0.49-0.56 based on random seed number the python model neural network with 2 hidden layers and (7,6) perceptron is selected and is selected as the best model from our analysis.

Table 19 Comparison of Model Metrics between SAS and Python

PYTHON VS SAS(Best Models)		
Metrics	SAS	Python
	Decision Tree Depth 20 Branch 3 Leaf Size 3	Neural Networks Layer 2 Perceptron (7,6)
Recall	0.5120	0.5115
Precision	0.6832	0.7367
Accuracy	0.8826	0.8931
F1	0.5853	0.6040

9.2 POSSIBILITY FOR VARIANCE IN DECISION TREE MODEL

As the tree is based on a single split of the data set (70/30), the tree has a fixed training and validation data set. Every time the random seed is changed the data set changes and due to the skew associated with the data (as mentioned earlier), the reproducibility in results is not high. Random forest model lacks this disadvantage as they are comprised of multiple bags of decision trees ensuring homogeneity in the sample spaces. This is also reflected in lower ASE (Average

Squared Error) values (approx. 0.094) for random forest models as compared to the decision tree models (0.096) despite having slightly poorer metric performances as compared to the decision tree models.

9.3 DEEP DIVE INTO METRICS

The credit card company may have its own set of interpretation of 'good' model based on financial risks and P&L as opposed to a pure Statistics view point. The company may not mind higher false positives as opposed to higher false negatives as false negatives may have a bigger financial risk to a company. As these metrics have not been clearly defined in the problem statement, actual selection of good model in reality may be determined by combing 'weights' associated false positives, false negatives and computing the loss matrix and the lowest loss model shall be the best to use.

9.4 BALANCING THE DATA

As mentioned above the data is slightly skewed with limited (16%) defaults. To ensure the models are more accurate, over-sampling or under-sampling may be performed on the data set. As this project does not require do to data transformations this is not pursued in this report. Balancing the data may provide more robust models but may also introduce bias into these models.

Appendix A : Python Code

```
# coding: utf-8

"""
Created on Sat Mar  3 20:36:34 2018

@authors: Sashank & Olivia

"""
# ================================================================================
# #import Class Libraries
# ================================================================================

# class for logistic regression
from sklearn.linear_model import LogisticRegression

# class for decision tree
from Class_tree import DecisionTree
from sklearn.tree import DecisionTreeClassifier
#class for neural network
from Class_FNN import NeuralNetwork
from sklearn.neural_network import MLPClassifier
#class for random forest
from sklearn.ensemble import RandomForestClassifier
#other needed classes
from Class_replace_impute_encode import ReplaceImputeEncode
from sklearn.model_selection import cross_validate
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix
# show plots
import seaborn as sns


# ================================================================================
# # Functions Used
# ================================================================================
# Get classification metrics
def getClassificationMetrics(tn, fp, fn, tp, y_test, pred_test):

    #sensitivity
    Recall=tp/(tp+fn);
    print("sensitivity: ", Recall)
    #specificity
    Specificity= tn/(tn+fp)
    print("specificity: ", Specificity)
    #accuracy
    print("accuracy: ", (tp+tn)/(tp+fn+tn+fp))
    #precision
    Precision=tp/(tp+fp);
    print("precision: ", Precision)
    #f1 score
    print("f1_score: ",  (2*Recall*Precision)/(Recall + Precision))
    #misclassification
    print("Misc: ", (fp+fn)/(tp+fn+tn+fp))
    #False Positive Rate
```

```
    print("FPR: ", 1-Specificity)

    return


# ================================================================================
# #Get Data from excel file
# ================================================================================

file_path = "/Users/sasha/Library/Mobile Documents/com~apple~CloudDocs/STAT 656/Midterm/CreditCard_Defaults.xlsx"
df = pd.read_excel(file_path)
df.head()

# create attribute map
# Attribute Map:  the key is the name in the DataFrame
# The first number of 0=Interval, 1=binary and 2=nomial
# The 1st tuple for interval attributes is their lower and upper bounds
# The 1st tuple for categorical attributes is their allowed categories
# The 2nd tuple contains the number missing and number of outliers
attribute_map = {
    "Default":[1,(0,1),[0,0]],
   "Gender":[1,(1,2),[0,0]],
  "Education":[2,(0,1,2,3,4,5,6),[0,0]],
   "Marital_Status":[2,(0,1,2,3),[0,0]],
    "card_class":[2,(1,2,3),[0,0]],
    "Age":[0,(20,80),[0,0]],
    "Credit_Limit":[0,(100,80000),[0,0]],
    "Jun_Status":[0,(-2,8),[0,0]],
   "May_Status":[0,(-2,8),[0,0]],
   "Apr_Status":[0,(-2,8),[0,0]],
    "Mar_Status":[0,(-2,8),[0,0]],
    "Feb_Status":[0,(-2,8),[0,0]],
    "Jan_Status":[0,(-2,8),[0,0]],
     "Jun_Bill":[0,(-12000,32000),[0,0]],
   "May_Bill":[0,(-12000,32000),[0,0]],
    "Apr_Bill":[0,(-12000,32000),[0,0]],
    "Mar_Bill":[0,(-12000,32000),[0,0]],
    "Feb_Bill":[0,(-12000,32000),[0,0]],
    "Jan_Bill":[0,(-12000,32000),[0,0]],
    "Jun_Payment":[0,(0,60000),[0,0]],
    "May_Payment":[0,(0,60000),[0,0]],
          "Apr_Payment":[0,(0,60000),[0,0]],
          "Mar_Payment":[0,(0,60000),[0,0]],
          "Feb_Payment":[0,(0,60000),[0,0]],
          "Jan_Payment":[0,(0,60000),[0,0]],
          "Jun_PayPercent":[0,(0,1),[0,0]],
          "May_PayPercent":[0,(0,1),[0,0]],
          "Apr_PayPercent":[0,(0,1),[0,0]],
          "Mar_PayPercent":[0,(0,1),[0,0]],
          "Feb_PayPercent":[0,(0,1),[0,0]],
          "Jan_PayPercent":[0,(0,1),[0,0]]

      }

      df = df.drop("Customer", axis=1)
      # drop=False - used for Decision tree
      rie_1 = ReplaceImputeEncode(data_map=attribute_map, nominal_encoding='one-hot', interval_scale = 'std',drop = False, display=True)
      encoded_df = rie_1.fit_transform(df)
      #create X and y
```

```
118.        y = encoded_df["Default"]
119.        X = encoded_df.drop("Default", axis=1)
120.        np_y = np.ravel(y)
121.
122.
123.
124.        # ================================================================================
125.        # #Initial Analysis
126.        # ================================================================================
127.        #Correlation Plot of entire dataset
128.        # Deductions: All the columns indicating customers bills are correlated
129.        #All columns indicating months behind payement are correlated
130.        #
131.
132.        corr = df.corr()
133.        g=sns.heatmap(corr,
134.                      xticklabels="auto",
135.                      yticklabels="auto")
136.        g.set_yticklabels(g.get_yticklabels(), rotation = 0, fontsize = 8)
137.        g.set_xticklabels(g.get_yticklabels(), rotation = 70, fontsize = 6)
138.
139.
140.
141.        # ================================================================================
142.        # # Hyper Parameter Optimization with 10-fold Cross Validation
143.        # ================================================================================
144.
145.        #Cross Validation for Logistic Regression
146.
147.        #Do Forward stepwise regression with kfold cross validation
148.        remaining = set(X.columns)
149.        response='Default'
150.        selected = []
151.        current_score, best_new_score = 0.0, 0.0
152.        while remaining and current_score == best_new_score:
153.            scores_with_candidates = []
154.            for candidate in remaining:
155.                #forward step wise columns
156.                formula = selected + [candidate]
157.
158.                X_Forward= encoded_df[formula]
159.                model = LogisticRegression()
160.                kf = KFold(n_splits=10,random_state=12345)
161.                scoring = 'f1'
162.                results = cross_val_score(model, X_Forward, np_y, cv=kf, scoring=scoring
    )
163.                score= results.mean()
164.                scores_with_candidates.append((score, candidate))
165.
166.            scores_with_candidates.sort()
167.            best_new_score, best_candidate = scores_with_candidates.pop()
168.            if(best_new_score==current_score):
169.                break;
170.            if current_score < best_new_score:
171.                print(best_new_score,current_score)
172.                remaining.remove(best_candidate)
173.                selected.append(best_candidate)
```

```
                print(selected)
                current_score = best_new_score

        # ================================================================================

        # #Output
        # 0.315606776734 0.0
        # ['Jun_Status']
        # 0.461891482261 0.315606776734
        # ['Jun_Status', 'card_class2']
        # 0.463242620061 0.461891482261
        # ['Jun_Status', 'card_class2', 'Marital_Status3']
        # 0.46343712211 0.463242620061
        # ['Jun_Status', 'card_class2', 'Marital_Status3', 'Education6']
        # 0.463632243445 0.46343712211
        # ['Jun_Status', 'card_class2', 'Marital_Status3', 'Education6', 'Education5']

        #
        # ================================================================================


        #Best model is at ['Jun_Status', 'card_class2', 'Marital_Status3', 'Education6',
     'Education5']



        # ================================================================================

        # # Cross Validation for decision tree:
        # ================================================================================

        '''''
        The results show that max_depth has the most influence while min_samples_split w
    on't much have inflence on metrics score.
        When max_depth is below 20, the metrics score do slightly increase when
        the min_samples_leaf increases. When the max_depth is bigger than 20, then actua
    lly increase of
        min_samples_leaf reduces the metrics score. Since there isn't much difference be
    tween depth 20,25,50, I stop increase
        this number.We can try to use Maximum Tree Depth:25, Min_samples_leaf:3, Min_sam
    ples_split:5 or Maximum Tree Depth:6, Min_samples_leaf:7,
        Min_samples_split:5 in the 70/30 validation.
        '''
        depth_list = [6]
        #[5,6,7,8,10, 12, 15, 20, 25, 50]
        minSamplesLeaf= [3,5,7]
        minSamplesSplit=[3,5,7]


        score_list = ['accuracy', 'recall', 'precision', 'f1']
        for d in depth_list:
            for l in minSamplesLeaf:
                for s in minSamplesSplit:
                    print("\nMaximum Tree Depth: ", d, "Min_samples_leaf", l, "Min_sampl
    es_split", s)
                    dtc = DecisionTreeClassifier(max_depth=d, min_samples_leaf=l,  min_s
    amples_split=s, random_state=12345)
                    dtc = dtc.fit(X,np_y)
```

```
222.                    scores = cross_validate(dtc, X, np_y, scoring=score_list, return_tra
    in_score=False, cv=10)
223.
224.                    print("{:.<13s}{:>6s}{:>13s}".format("Metric", "Mean", "Std. Dev."))

225.                    for s in score_list:
226.                        var = "test_"+s
227.                        mean = scores[var].mean()
228.                        std  = scores[var].std()
229.                        print("{:.<13s}{:>7.4f}{:>10.4f}".format(s, mean, std))
230.
231.
232.        #Maximum Tree Depth:  25 Min_samples_leaf 3 Min_samples_split 5
233.        #Metric.......  Mean    Std. Dev.
234.        #accuracy..... 0.8296    0.0108
235.        #recall....... 0.4660    0.0241
236.        #precision.... 0.4745    0.0336
237.        #f1........... 0.4697    0.0241
238.
239.        #Maximum Tree Depth:  6 Min_samples_leaf 7 Min_samples_split 5
240.        #Metric.......  Mean    Std. Dev.
241.        #accuracy..... 0.8751    0.0072
242.        #recall....... 0.4524    0.0389
243.        #precision.... 0.6701    0.0385
244.        #f1........... 0.5390    0.0319
245.
246.
247.
248.
249.
250.        # ================================================================================

251.        # # Cross-
    Validation for neural network: There isn't much difference going on with different
252.        # # layers/perceptrons. Best Neural Network is (7,6)
253.        # ================================================================================

254.        network_list = [(3), (11), (5,4), (6,5), (7,6), (8,7),(9,8)]
255.        score_list = ['accuracy', 'recall', 'precision', 'f1']
256.        for nn in network_list:
257.            print("\nNetwork: ", nn)
258.            fnn = MLPClassifier(hidden_layer_sizes=nn, activation='logistic',
               solver='lbfgs', max_iter=1000, random_state=12345)
259.            fnn = fnn.fit(X,np_y)
260.            scores = cross_validate(fnn, X, np_y, scoring=score_list,
               return_train_score=False, cv=10)
261.
262.            print("{:.<13s}{:>6s}{:>13s}".format("Metric", "Mean", "Std. Dev."))
263.            for s in score_list:
264.                var = "test_"+s
265.                mean = scores[var].mean()
266.                std  = scores[var].std()
267.                print("{:.<13s}{:>7.4f}{:>10.4f}".format(s, mean, std))
268.        #Network:  (7, 6)
269.        # ================================================================================

270.        # Metric.......  Mean    Std. Dev.
271.        # accuracy..... 0.8695    0.0079
272.        # recall....... 0.4537    0.0268
273.        # precision.... 0.6382    0.0456
274.        # f1........... 0.5293    0.0240
```

```
275.        # ========================================================================

276.
277.
278.
279.
280.        # ========================================================================

281.        # # Cross-Validation for random forest
282.        # ========================================================================

283.        '''''
284.        Metrics scores increase when we enlarge the n_estimators and max_features. Howev
    er, when we
285.        further increase the n_estimators to 70, the metrics scores didn't show much dif
    ference from the
286.        one with 50 estimators. So we stop it to increase more. "auto" in max_feature us
    ually gives
287.        good results, but it's not the case here. Finally, we would use the model with N
    umber of Trees:70,
288.        Max_features:0.5 for 70/30 split validation
289.        '''
290.        estimators_list   = [10, 30, 50, 70]
291.        max_features_list = ['auto', 0.5, 0.7]
292.        score_list = ['accuracy', 'recall', 'precision', 'f1']
293.        max_f1 = 0
294.        for e in estimators_list:
295.            for f in max_features_list:
296.                print("\nNumber of Trees: ", e, " Max_features: ", f)
297.                rfc = RandomForestClassifier(n_estimators=e, criterion="gini", max_depth
    =None, min_samples_split=2, min_samples_leaf=1, max_features=f,                    n_j
    obs=1, bootstrap=True, random_state=12345)
298.                rfc= rfc.fit(X, np_y)
299.                scores = cross_validate(rfc, X, np_y, scoring=score_list, return_train_s
    core=False, cv=10)
300.
301.                print("{:.<13s}{:>6s}{:>13s}".format("Metric", "Mean", "Std. Dev."))
302.                for s in score_list:
303.                    var = "test_"+s
304.                    mean = scores[var].mean()
305.                    std  = scores[var].std()
306.                    print("{:.<13s}{:>7.4f}{:>10.4f}".format(s, mean, std))
307.
308.
309.        #Number of Trees:  70  Max_features:  0.5
310.        #Metric.......  Mean    Std. Dev.
311.        #accuracy..... 0.8832    0.0062
312.        #recall....... 0.4706    0.0367
313.        #precision.... 0.7109    0.0330
314.        #f1........... 0.5653    0.0287
315.
316.
317.
318.
319.        # ========================================================================

320.        # # Evaluate the best model:
321.        # ========================================================================

322.        '''''
```

```
323.        Decision tree with(Maximum Tree Depth: 25 Min_samples_leaf:3 Min_samples_split:
    5) is overfitting
324.        the data. So does the random forest (n_estimator:70, max_feature:0.5). The best
    model is neural
325.        network (7,6), which scores higher across all metrics. Please refer report for f
    urther details.
326.
327.        '''
328.
329.
330.        #Best Model forward step wise selection - 0.42 model score
331.
332.        X_ForwardStepWise=X[['Jun_Status', 'card_class2', 'Marital_Status3', 'Education6
    ', 'Education5']]
333.
334.        #Split Data 70-30
335.        #split 70, 30 and use X_train and y_train for doing 70/30 cross validation
336.        X_train, X_test, y_train, y_test = train_test_split(X_ForwardStepWise,np_y,test_
    size = 0.3, random_state=12345)
337.
338.        #train model
339.        logModel= LogisticRegression()
340.        logModel.fit(X_train, y_train)
341.
342.        #test model
343.        pred_test = logModel.predict(X_test)
344.        pred_train = logModel.predict(X_train)
345.        #get Metrics
346.        tn, fp, fn, tp = confusion_matrix(y_test, pred_test).ravel()
347.        tn_train, fp_train, fn_train, tp_train = confusion_matrix(y_train,pred_train ).r
    avel()
348.        (tn, fp, fn, tp)
349.
350.        getClassificationMetrics(tn,fp,fn,tp,y_test,pred_test)
351.        getClassificationMetrics(tn_train, fp_train, fn_train, tp_train,y_train,pred_tra
    in)
352.
353.
354.        # ================================================================================

355.        # sensitivity_test/recall_test/TPR_test: 0.420097697139
356.        # specificity_test:  0.961675697106
357.        # accuracy_test:  0.875444444444
358.        # precision_test:  0.674887892377
359.        # f1_score:  0.517849462366
360.        # misc_test:  0.124555555556
361.        # FPR:  0.0383243028941
362.        # ================================================================================

363.
364.
365.
366.        X_train, X_validate, y_train, y_validate =  train_test_split(X, np_y,test_size =
     0.3, random_state=12345)
367.
368.
369.        #Decison Tree
370.        dtc = DecisionTreeClassifier(max_depth=25, min_samples_leaf=3, min_samples_split
    =5, random_state=12345)
371.        dtc = dtc.fit(X_train,y_train)
```

```
372.        DecisionTree.display_binary_split_metrics(dtc, X_train, y_train,X_validate, y_va
    lidate)
373.        features = list(X)
374.        classes = ['No-Default', 'Default']
375.
376.        DecisionTree.display_importance(dtc, features)
377.
378.
379.        # ================================================================================

380.        # #Output
381.        # Model Metrics.....        Training     Validation
382.        # Observations.....           21000           9000
383.        # Features.............          41             41
384.        # Maximum Tree Depth.....       25             25
385.        # Minimum Leaf Size......        3              3
386.        # Minimum split Size.....        5              5
387.        # Mean Absolute Error....     0.0552         0.1807
388.        # Avg Squared Error......     0.0276         0.1502
389.        # Accuracy...............     0.9580         0.8284
390.        # Precision..............     0.8924         0.4612
391.        # Recall (Sensitivity)...     0.8439         0.4599
392.        # F1-score...............     0.8675         0.4605
393.        # MISC (Misclassification)...   4.2%          17.2%
394.        #      class 0...............   2.0%          10.2%
395.        #      class 1...............  15.6%          54.0%
396.        #
397.        #
398.        # Training
399.        # Confusion Matrix  Class 0   Class 1
400.        # Class 0.....     17231       348
401.        # Class 1.....       534      2887
402.        #
403.        #
404.        # Validation
405.        # Confusion Matrix  Class 0   Class 1
406.        # Class 0.....      6797       770
407.        # Class 1.....       774       659
408.        #
409.        # FEATURE......... IMPORTANCE
410.        # Jun_Status......   0.2742
411.        # May_Status......   0.0498
412.        # Credit_Limit....   0.0490
413.        # Age.............   0.0454
414.        # Jun_Bill........   0.0395
415.        # Jan_Bill........   0.0374
416.        # May_Bill........   0.0315
417.        # May_PayPercent..   0.0306
418.        # Apr_Bill........   0.0289
419.        # Jan_Payment.....   0.0287
420.        # May_Payment.....   0.0287
421.        # Apr_PayPercent..   0.0282
422.        # Jun_PayPercent..   0.0279
423.        # Mar_Bill........   0.0276
424.        # Apr_Payment.....   0.0275
425.        # Mar_PayPercent..   0.0273
426.        # Feb_Bill........   0.0271
427.        # Jun_Payment.....   0.0252
428.        # Feb_PayPercent..   0.0243
429.        # Jan_PayPercent..   0.0230
430.        # Feb_Payment.....   0.0225
```

```
431.        # Mar_Payment.....   0.0219
432.        # Mar_Status......   0.0164
433.        # Jan_Status......   0.0153
434.        # Apr_Status......   0.0087
435.        # Gender..........   0.0074
436.        # Education2......   0.0059
437.        # Feb_Status......   0.0043
438.        # Education3......   0.0042
439.        # Education1......   0.0041
440.        # Marital_Status2.   0.0028
441.        # Marital_Status1.   0.0023
442.        # card_class1.....   0.0009
443.        # card_class0.....   0.0008
444.        # card_class2.....   0.0004
445.        # Education5......   0.0003
446.        # Marital_Status3.   0.0001
447.        # Education0......   0.0000
448.        # Education4......   0.0000
449.        # Education6......   0.0000
450.        # Marital_Status0.   0.0000
451.        # ============================================================================

452.
453.
454.        #Neural Network
455.        fnn = MLPClassifier(hidden_layer_sizes=(7,6), activation='logistic', solver='lbf
    gs', max_iter=1000, random_state=12345)
456.        fnn = fnn.fit(X_validate,y_validate)
457.        NeuralNetwork.display_binary_split_metrics(fnn, X_train, y_train, X_validate, y_
    validate)
458.        # ============================================================================

459.        # #output
460.        # Model Metrics.....        Training     Validation
461.        # Observations.....           21000           9000
462.        # Features.............          41             41
463.        # Number of Layers.....          2              2
464.        # Number of Outputs......        1              1
465.        # Number of Weights......      349            349
466.        # Activation Function....   logistic       logistic
467.        # Mean Absolute Error....     0.1913         0.1602
468.        # Avg Squared Error......     0.1111         0.0798
469.        # Accuracy...............     0.8602         0.8931
470.        # Precision..............     0.6065         0.7367
471.        # Recall (Sensitivity)...     0.4046         0.5115
472.        # F1-score...............     0.4854         0.6038
473.        # MISC (Misclassification)...  14.0%          10.7%
474.        #      class 0.............     5.1%           3.5%
475.        #      class 1.............    59.5%          48.8%
476.        #
477.        #
478.        # Training
479.        # Confusion Matrix  Class 0   Class 1
480.        # Class 0.....     16681       898
481.        # Class 1.....      2037      1384
482.        #
483.        #
484.        # Validation
485.        # Confusion Matrix  Class 0   Class 1
486.        # Class 0.....      7305       262
487.        # Class 1.....       700       733
```

```
488.        # ============================================================================

489.
490.
491.        #Random Forest
492.        rfc = RandomForestClassifier(n_estimators=70, criterion="gini", max_depth=None,
    min_samples_split=2, min_samples_leaf=1, max_features=0.5, n_jobs=1, bootstrap=True, ra
    ndom_state=12345)
493.        rfc= rfc.fit(X_train, y_train)
494.        DecisionTree.display_importance(rfc, features)
495.        rfc_validate = rfc.predict(X_validate)
496.        rfc_train = rfc.predict(X_train)
497.        tn, fp, fn, tp = confusion_matrix(y_validate, rfc_validate).ravel()
498.        tn_t, fp_t, fn_t, tp_t = confusion_matrix(y_train, rfc_train).ravel()
499.
500.
501.        getClassificationMetrics(tn,fp,fn,tp,y_validate,rfc_validate)
502.        getClassificationMetrics(tn_t,fp_t,fn_t,tp_t,y_train,rfc_train)
503.
504.        #Output
505.        # ============================================================================

506.        # FEATURE......... IMPORTANCE
507.        # Jun_Status......   0.1649
508.        # May_Status......   0.0750
509.        # Age.............   0.0490
510.        # Jun_Bill........   0.0416
511.        # Credit_Limit....   0.0410
512.        # Jan_Bill........   0.0350
513.        # May_Bill........   0.0347
514.        # Apr_Payment.....   0.0346
515.        # Jan_Payment.....   0.0328
516.        # Feb_Bill........   0.0320
517.        # Mar_Bill........   0.0309
518.        # Apr_Bill........   0.0308
519.        # Jun_Payment.....   0.0307
520.        # Apr_PayPercent..   0.0294
521.        # Jun_PayPercent..   0.0290
522.        # May_PayPercent..   0.0290
523.        # May_Payment.....   0.0287
524.        # Mar_Payment.....   0.0284
525.        # Feb_Payment.....   0.0272
526.        # Mar_PayPercent..   0.0270
527.        # Jan_PayPercent..   0.0269
528.        # Feb_PayPercent..   0.0258
529.        # Apr_Status......   0.0233
530.        # Feb_Status......   0.0167
531.        # Mar_Status......   0.0164
532.        # Jan_Status......   0.0129
533.        # Gender..........   0.0077
534.        # Education2......   0.0059
535.        # Marital_Status2.   0.0058
536.        # Education1......   0.0055
537.        # Education3......   0.0054
538.        # Marital_Status1.   0.0052
539.        # card_class1.....   0.0033
540.        # card_class0.....   0.0027
541.        # card_class2.....   0.0018
542.        # Marital_Status3.   0.0017
543.        # Education5......   0.0007
544.        # Marital_Status0.   0.0003
```

```
545.        # Education4......   0.0002
546.        # Education6......   0.0002
547.        # Education0......   0.0000
548.        #
549.        # for validation data
550.        # sensitivity:  0.471039776692
551.        # specificity:  0.962336460949
552.        # accuracy:  0.884111111111
553.        # precision:  0.703125
554.        # f1_score:  0.564145424154
555.        # Misc:  0.115888888889
556.        # FPR:  0.0376635390511
557.        #
558.        # for train data
559.        # sensitivity:  0.999415375621
560.        # specificity:  1.0
561.        # accuracy:  0.999904761905
562.        # precision:  1.0
563.        # f1_score:  0.999707602339
564.        # Misc:  9.52380952381e-05
565.        # FPR:  0.0
566.        # ================================================================================
```