Default of Credit Card Clients

Deepika Srinivasan

ACKNOWLEDGEMENTS

We would like to thank our professor LIANWEN Wang for guiding us and encouraging us with valuable feedback.

In addition, we also offer our sincere thanks to UCI for documenting data and metadata on this dataset.
UCI Default of Credit card client Dataset
https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

Source:
Name: I-Cheng Yeh
email addresses: (1) icyeh **'@'** chu.edu.tw (2) 140910 **'@'** mail.tku.edu.tw
institutions: (1) Department of Information Management, Chung Hua University, Taiwan. (2) Department of Civil Engineering, Tamkang University, Taiwan.

Yeh, I. C., & Lien, C. H. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Systems with Applications, 36(2), 2473-2480.

Contents

2. INTRODUCTION

Problem Statement

Credit card industry is a great boon in today's world and it has played a major role in everyday life. The average American owns a number of credit cards with different balances reaching a huge amount sometimes in thousands of dollars. However, care should be taken when handling these cards as it might bring a heavy toll on credit score and credit history if not maintained well.

Default

When the customer fails to make a payment on the debt by the specified deadline it's said that he has defaulted the payment. With respect to credit cards, creditors may raise interest rates, issue penalty rate or sometimes may even decrease the line of credit. During the circumstances of serious cases, card issuer may even take legal action against customer to enforce payment.

In this report, we will work on Default of credit card client's dataset involving Taiwanese credit customers and use it to predict whether they would default in the consecutive month based on credit balance, previous bill payments, bill statements and their demographic factors like gender, marital status, education etc. We will make use of various classifiers like Decision trees using Holdout, Bagging and Random forest, Naïve Bayes and SVM models in order to find the best classifier with minimum test error rate.

Dataset Overview

The dataset used in this report is taken from UCI- Machine Learning Repository - (https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients)

Default of Credit card clients dataset consists of data of 30000 Taiwan customers with demographic factors, credit balance data, payment history and bill statements from April 2005- September 2005.In addition, there exists one target column which will indicate whether the customer will default or not?

We would also highlight the fact that some levels not listed as part of the dataset also existed in dataset. Since the ratio of these values were significant we could not remove them from the dataset for our analysis. In sum, we have included the dataset in as is condition.

Attribute Information:

ATTRIBUTES	VALUES
LIMIT_BAL	Credit limits vary widely depending on a consumer's creditworthiness Amount of the given credit including consumer credit and his/her family (supplementary) credit.
GENDER	1 = male; 2 = female
EDUCATION	1 = graduate school; 2 = university; 3 = high school; 4 = others
MARITAL STATUS	1 = married; 2 = single; 3 = others
PAYMENT HISTORY –APRIL 2005 TO SEPTEMBER 2005	The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.
BILL AMOUNT – APRIL 2005 TO SEPTEMBER 2005	Amount of bill statement
PAY AMOUNT-APRIL 2005 TO SEPTEMBER 2005	Amount of previous payment
DEFAULT PAYMENT	1=Default 0=Non-default

Data load:

We load the data from excel into R by making use of readxl library package and analyze few rows to get a basic understanding of the data

```
> head(data,5)
  ID LIMIT_BAL SEX EDUCATION MARRIAGE AGE PAY_0 PAY_2 PAY_3 PAY_4 PAY_5 PAY_6 BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4 BILL_AMT5 BILL_AMT6 PAY_AMT1 PAY_AMT2 PAY_AMT3
1  1     20000   2         2        1  24     2     2    -1    -1    -2    -2      3913      3102       689         0         0         0        0       689         0
2  2    120000   2         2        2  26    -1     2     0     0     0     2      2682      1725      2682      3272      3455      3261        0      1000      1000
3  3     90000   2         2        2  34     0     0     0     0     0     0     29239     14027     13559     14331     14948     15549     1518      1500      1000
4  4     50000   2         2        1  37     0     0     0     0     0     0     46990     48233     49291     28314     28959     29547     2000      2019      1200
5  5     50000   1         2        1  57    -1     0    -1     0     0     0      8617      5670     35835     20940     19146     19131     2000     36681     10000
  PAY_AMT4 PAY_AMT5 PAY_AMT6 default_payment_next_month
1        0        0        0                          1
2     1000        0     2000                          1
3     1000     1000     5000                          0
4     1100     1069     1000                          0
5     9000      689      679                          0
```

Data Transformation:

As per our analysis, attributes Sex, Education, Marriage and PAY_0 to PAY_6 are categorical variables. However, we find that there are many miscoded values and removing them will result in data loss. So, we have not converted them as factors.

default_payment_next_month is also indicated as numerical variable in the dataset which is modified as factor for our further analysis.

data$default_payment_next_month<-as.factor(data$default_payment_next_month)

Structure of dataset after modification of default payment attribute:

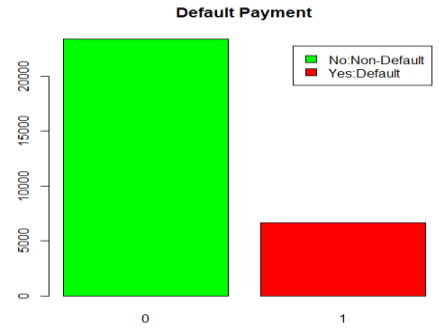
```
> str(data)
Classes 'tbl_df', 'tbl' and 'data.frame':       30000 obs. of  25 variables:
 $ ID                        : num  1 2 3 4 5 6 7 8 9 10 ...
 $ LIMIT_BAL                 : num  20000 120000 90000 50000 50000 50000 500000$
 $ SEX                       : num  2 2 2 2 1 1 1 2 2 1 ...
 $ EDUCATION                 : num  2 2 2 2 2 1 1 2 3 3 ...
 $ MARRIAGE                  : num  1 2 2 1 1 2 2 2 1 2 ...
 $ AGE                       : num  24 26 34 37 57 37 29 23 28 35 ...
 $ PAY_0                     : num  2 -1 0 0 -1 0 0 0 0 -2 ...
 $ PAY_2                     : num  2 2 0 0 0 0 0 -1 0 -2 ...
 $ PAY_3                     : num  -1 0 0 0 -1 0 0 -1 2 -2 ...
 $ PAY_4                     : num  -1 0 0 0 0 0 0 0 0 -2 ...
 $ PAY_5                     : num  -2 0 0 0 0 0 0 0 0 -1 ...
 $ PAY_6                     : num  -2 2 0 0 0 0 0 -1 0 -1 ...
 $ BILL_AMT1                 : num  3913 2682 29239 46990 8617 ...
 $ BILL_AMT2                 : num  3102 1725 14027 48233 5670 ...
 $ BILL_AMT3                 : num  689 2682 13559 49291 35835 ...
 $ BILL_AMT4                 : num  0 3272 14331 28314 20940 ...
 $ BILL_AMT5                 : num  0 3455 14948 28959 19146 ...
 $ BILL_AMT6                 : num  0 3261 15549 29547 19131 ...
 $ PAY_AMT1                  : num  0 0 1518 2000 2000 ...
 $ PAY_AMT2                  : num  689 1000 1500 2019 36681 ...
 $ PAY_AMT3                  : num  0 1000 1000 1200 10000 657 38000 0 432 0 ...
 $ PAY_AMT4                  : num  0 1000 1000 1100 9000 ...
 $ PAY_AMT5                  : num  0 0 1000 1069 689 ...
 $ PAY_AMT6                  : num  0 2000 5000 1000 679 ...
 $ default_payment_next_month: Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 $
```

Exploratory Data Analysis:

We have made use of R to prepare the following dashboards for better understanding the data and to derive prediction based on the variables.

Attribute: Default Payment

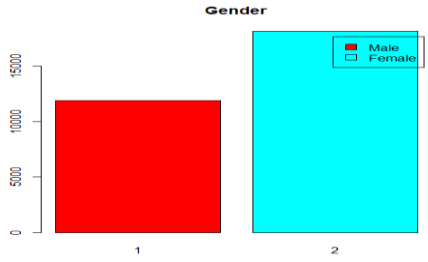
```
> table(data$default_payment_next_month)

    0     1
23364  6636
```



- 77.88% customers will not default while 22.12% of them are defaulters.

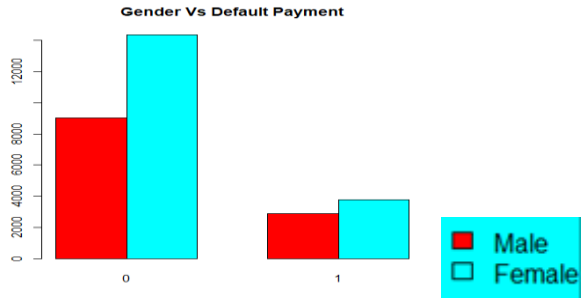
Attribute:Gender

```
> table(data$SEX)

    1     2
11888 18112
```



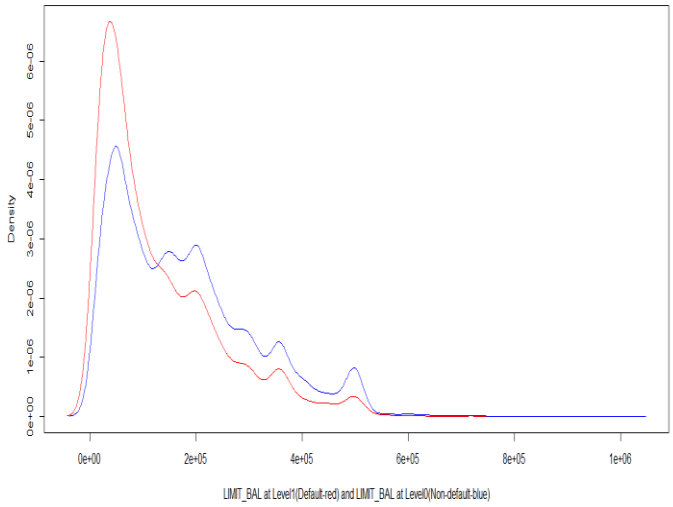
- 60.37% of female customers and 39.62% male customers

```
> table(data$SEX,data$default_payment_next_month)

      0    1
1  9015 2873
2 14349 3763
```



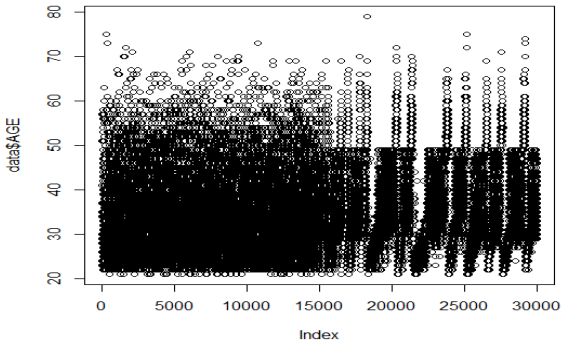
- Female outrank Male customers count in both Default and Non-Default category. In contrast with Defaulting customers, Non-defaulting Female customers count is significantly higher than Male customers.

Attribute: LIMIT_BAL

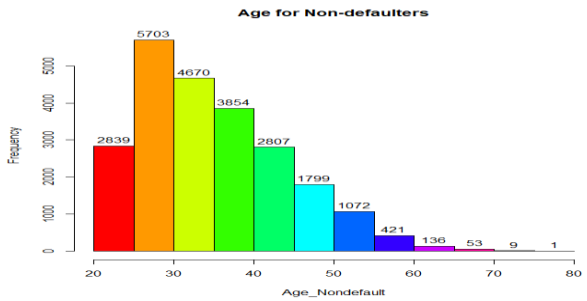


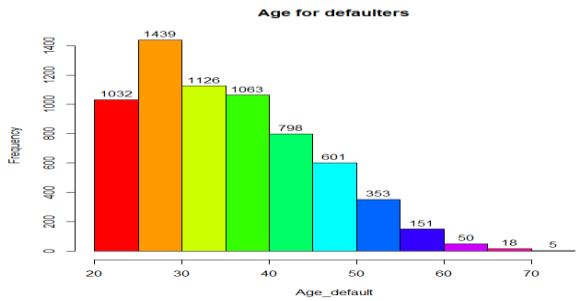
- People who default have lesser credit limit compared to those who pay on time.

Attribute:Age

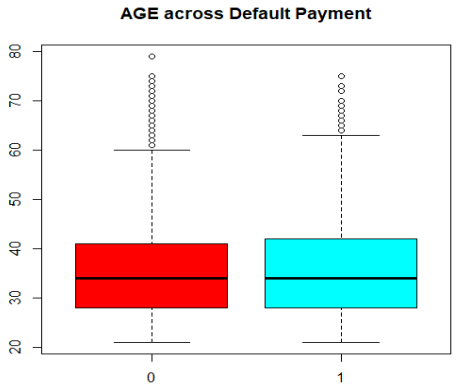


- Trend in age(order by age) and less dense for ID greater than 15000.





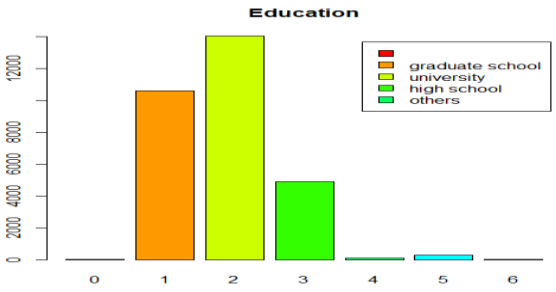
- Number of Non-defaulters belonging to age group 30-40 is higher.



- Boxplots looks identical for both category and therfore Age attribute doesn't play a major role in differentiating Default Vs Non Default customers.

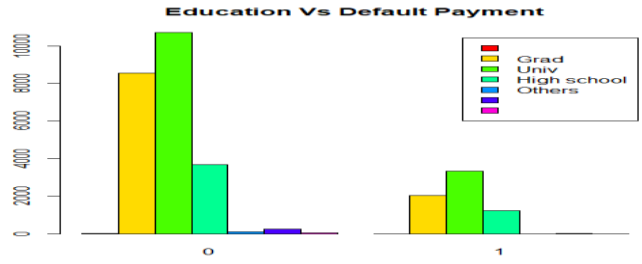
Attribute: Education

```
> table(data$EDUCATION)

    0     1     2     3     4     5     6
   14 10585 14030  4917   123   280    51
```



- More customers are highly educated(Graduate,University Level)
- We can also infer that level 0,5,6 are irrelevent levels but can be ignored as its not that significant when compared to other levels.

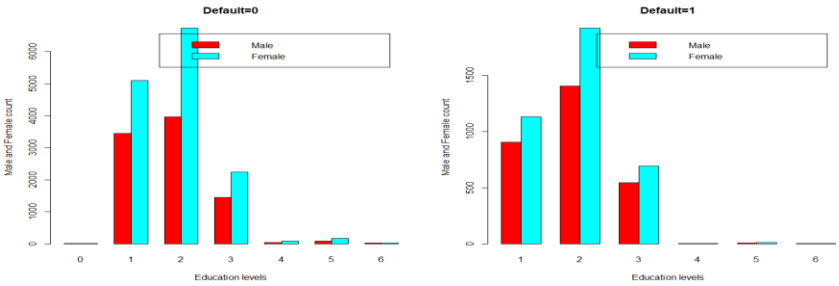
```
> table(data$EDUCATION,data$default_payment_next_month)

      0    1
0    14    0
1  8549 2036
2 10700 3330
3  3680 1237
4   116    7
5   262   18
6    43    8
```



- Customers who had University level education ranks topmost in either categories. The proportion of highly educated(Grad and University level) is higher in Non-default category of customers.

Relationship between Gender and Education

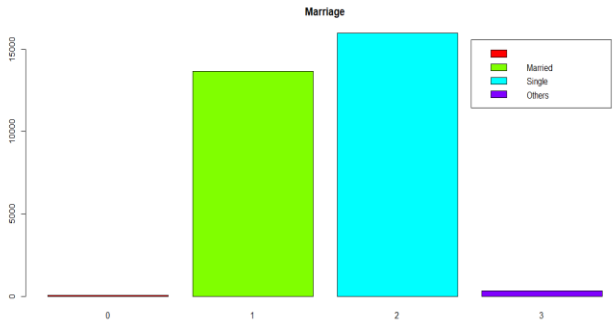
```
> table(level2$SEX, level2$EDUCATION)

      0    1    2    3    4    5    6
  1   8 3448 3966 1445   38   89   21
  2   6 5101 6734 2235   78  173   22
```



- In all the levels of education,dataset contains more female customers than male customers

Attribute:MARRIAGE

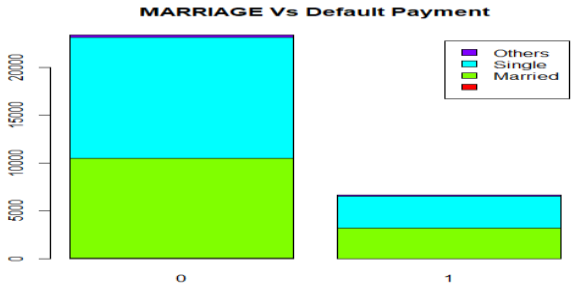
```
> table(data$MARRIAGE)

    0     1     2     3
   54 13659 15964   323
```



- Dataset has more Single customers when compared to Married followed by an insignificant count in others category. In addition, we also find a level 0 that is not part of the Marriage level.

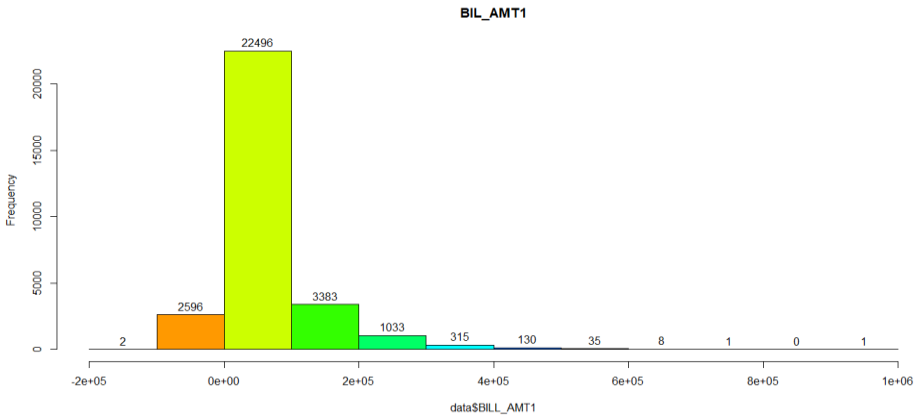
Relationship between Marital status and Default Payment

```
> table(data$MARRIAGE,data$default_payment_next_month)

      0     1
0    49     5
1 10453  3206
2 12623  3341
3   239    84
```

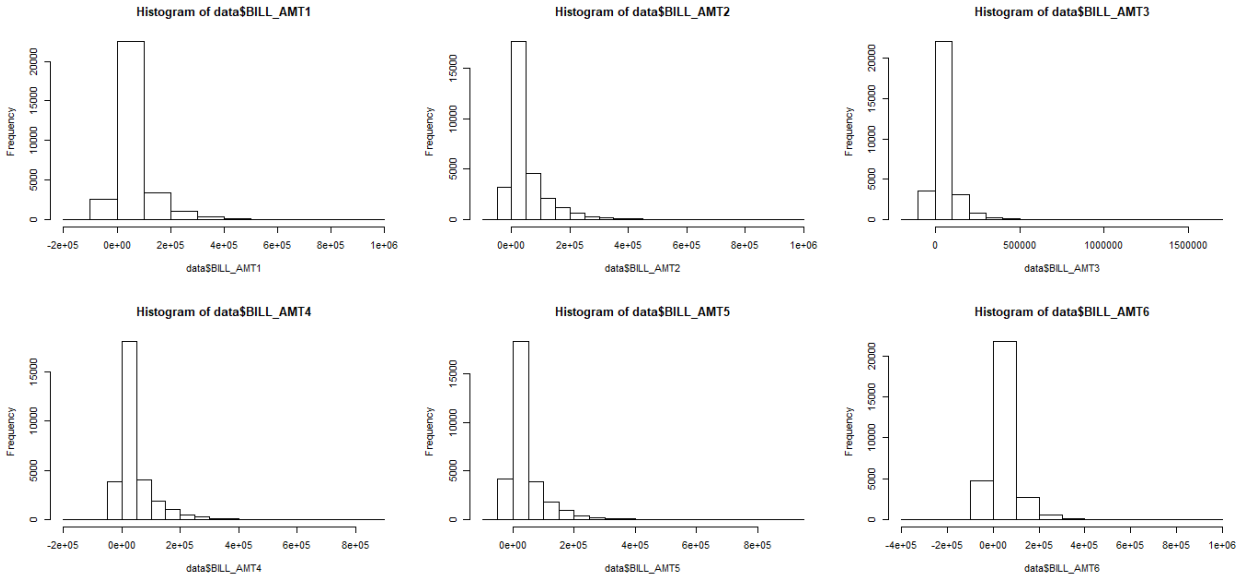


- Single customers rank the highest in both Default and Non-Default category.
- 79.07% Single customers and 76.53% Married Customers belong to Non-default category.

Atrribute:BILL_AMOUNT



From the above graph, we find that Bill statement for September 2005 had remarkably higher number of customers(22496) with amount in the range of 0 to 1e+05(100,000$).We find that same argument holds for BILL_AMT3 and BIL_AMT6.



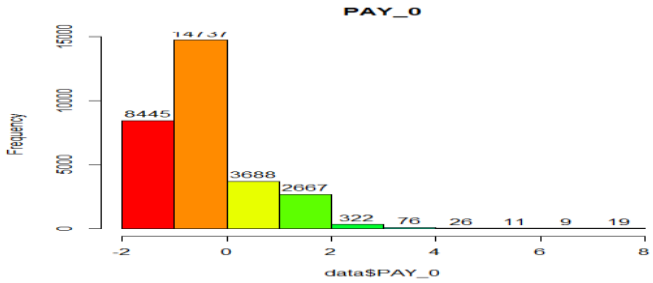
- Data Distribution is positively skewed(right tail is longer) towards higher amount in Bill_amt variable.

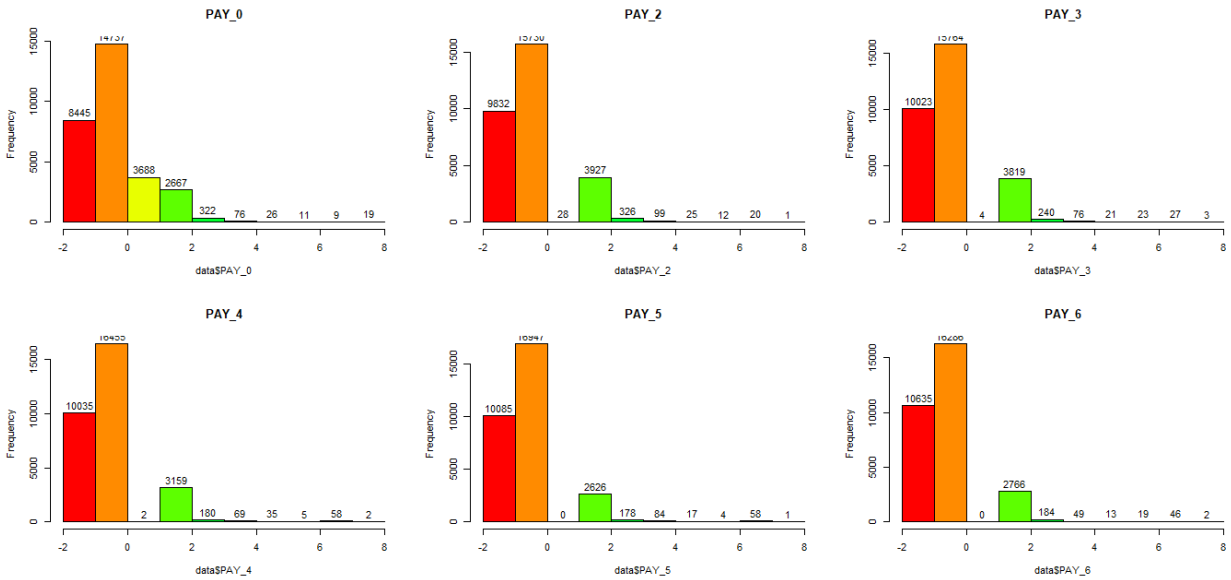
Attribute: PAY_0 –Repayment Status

In the dataset there are many levels to indicate whether customer has made payment for the particular month and if not payment delay is also mentioned in terms of months.

For example -1 denotes “Pay duly” and 1 denoted “Payment delay for 1 month”

However, from below graph we can see that some irrelevant levels are also picked up like 0 and -2.Since the count is not negligible we have not filtered the data on these irrelevant levels. In sum, we can consider 0 and -2 levels as data error.

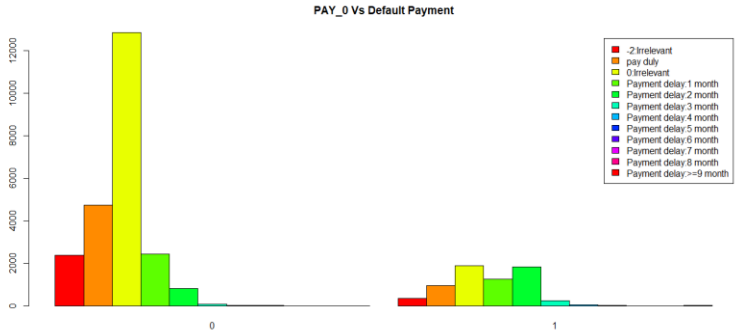




- Number of customers with more than 2 months payment is considerably lesser.
- More customer pay duly across all the months when compared to other categry levels.

Relationship between Repayment status and Default Payment

```
> table(data$PAY_0,data$default_payment_next_month)

        0     1
  -2 2394   365
  -1 4732   954
  0 12849  1888
  1  2436  1252
  2   823  1844
  3    78   244
  4    24    52
  5    13    13
  6     5     6
  7     2     7
  8     8    11
```



- Non-default category customers have done the payment duly when compared to Non-defaulting customers. This is a strong relationship as paying duly will help the customer not to default in the consecutive month. Thus PAY_X attribute is quite important when compared to others.
- Cutomers who have payment delay for greater than 2 months seems to be drastically less in both Default and Non-default category.

Demographics:

Based on the demographics, we can predict that being a Female,highly educated,single and between 30-40 years would more likely pay on time and will belong to Non-default class.

Feature Selection

We have 30000 observation and 25 variables.Some of the attributes had irrelevent or noisy data which has not been removed for our analysis.

Another interesting observation is that our target variable is in the ratio of 77.28:22.12 so without making use of classification we can easily predict ~78% of the time that data belongs to Non-defaulters group.

Decision Tree

Decision Trees are a type of Supervised Machine learning in which we split the data based on a specific attribute values.The resulting tree comprises of two parts, namely decision nodes and leaves.Decision nodes are the attributes on which data will be split while the leaves are final decision or the outcome derived.

Decision trees are classified as two tyoes

1.Classification Trees(Yes/No Types)

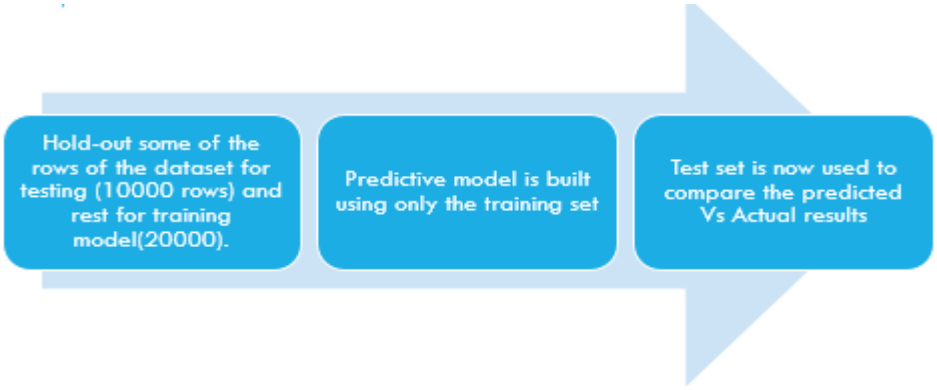
We have made use of the dataset as an example for Classification based Decision tree where we find whether the customers will default or not based on the other attributes.

2.Regression Trees(Continuous Data types)

In this type of trees target variable is continuous.

Classification Techniques on Default of Credit card clients Dataset

3.DECISION TREE-HOLDOUT



	Number of Observations	Number of Attributes
Training set	**20000**	**25**
Test Set	**10000**	**25**

Decision tree is built and it helps to classify the attribute default_payment_next_month based on all other attributes.As indicated, we will build the model on our training data.

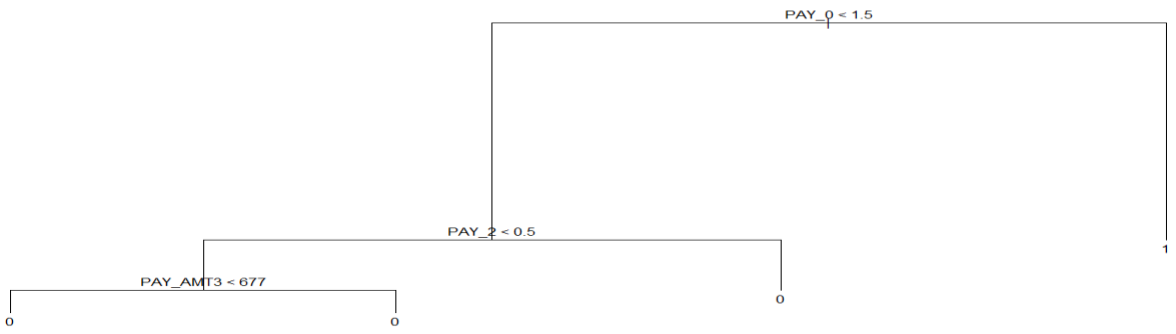
tree.data<-tree(default_payment_next_month~.,data=data,subset=train)

Summary() function will give us a detailed information on the tree built like the number of terminal nodes,training error rate.

```
> summary(tree.data)

Classification tree:
tree(formula = default_payment_next_month ~ ., data = data, subset = train)
Variables actually used in tree construction:
[1] "PAY_0"    "PAY_2"    "PAY_AMT3"
Number of terminal nodes:  4
Residual mean deviance:  0.8913 = 17820 / 20000
Misclassification error rate: 0.1801 = 3602 / 20000
```

From the summary(tree.data) we can see that only 3 variables are considered important when compared to other variables.

Below is the tree structure



In order to get split criterion,number of observations in the branch,deviance,overall prediction we make use of command tree.data

```
> tree.data
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

1) root 20000 21250 0 ( 0.7766 0.2234 )
  2) PAY_0 < 1.5 17910 16160 0 ( 0.8331 0.1669 )
    4) PAY_2 < 0.5 16373 13470 0 ( 0.8564 0.1436 )
      8) PAY_AMT3 < 677 4488  4713 0 ( 0.7814 0.2186 ) *
      9) PAY_AMT3 > 677 11885  8495 0 ( 0.8847 0.1153 ) *
    5) PAY_2 > 0.5 1537  2087 0 ( 0.5843 0.4157 ) *
  3) PAY_0 > 1.5 2090  2527 1 ( 0.2928 0.7072 ) *
```

Thus we find that 1.5,0.5 and 677 are the critical values for PAY_0,PAY_2 and PAY_AMT3 respectively

For example lets take one of the branch

```
  2) PAY_0 < 1.5 17910 16160 0 ( 0.8331 0.1669 )
```

We can say that probability of Non-defaulters is 83.31% for the split PAY_0 < 1.5 and therefore overall prediction is “Non-Default” i.e 0

Training error – We predict the target variable in training set using the tree built and obtain the training error.

Test Error- We predict the target variable(Test set) outcome using the model built and obtain the test error.

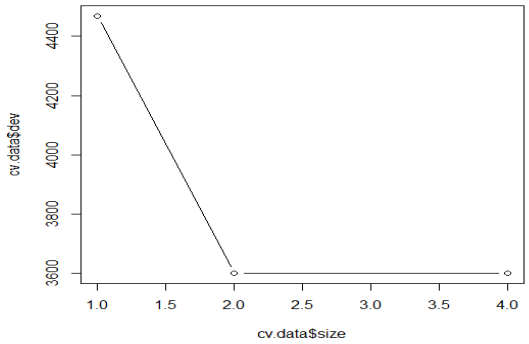
Train Error	Test Error	Accuracy(Test)
0.1801	**0.181**	**81.9%**

- Holdout test error is slightly higher than Training error.

Now we try to see if pruning the tree will result in better test error rate

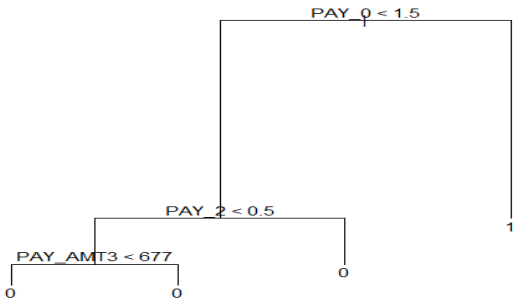
cv.tree() performs cross-validation in order to determine the optimal level of tree complexity.

```
> cv.data
$size
[1] 4 2 1

$dev
[1] 3602 3602 4468

$k
[1] -Inf    0  866

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"
```



Optimal tree size is 2 or 4 since the deviation is the smallest

Tree after pruning



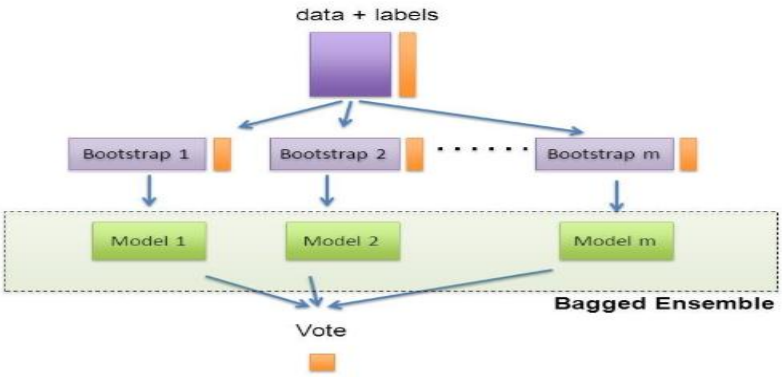
Since the tree is the same after Pruning there is no change in Training and test error

4.DECISION TREE-BAGGING

Analogy: Diagnosis based on multiple doctor's majority vote.

Training- We build classifier model Mi using training set Di which has d tuples from dataset D with replacement.

Classification – Each classifier Mi will return class prediction for the unknown sample and majority vote from all classifiers are used to determine the final class for sample.



We make use of randomforest library for performing Bagging operation

library(randomForest)

Bagging – Default mtry and ntrees=500

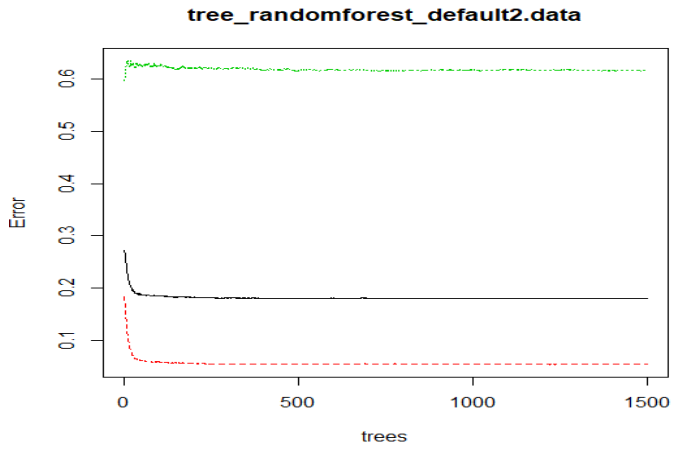
```
> print(tree_randomforest_default1.data)

Call:
 randomForest(formula = default_payment_next_month ~ ., data = data,      ntree = 500, subset = train)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 18.02%
Confusion matrix:
      0    1 class.error
0 14681  851  0.05479011
1  2753 1715  0.61615936
```

Test error rate is 17.9% for Bagging with ntrees=500

Bagging –Default mtry and ntrees=1500

```
> print(tree_randomforest_default2.data)

Call:
 randomForest(formula = default_payment_next_month ~ ., data = data,      ntree = 1500, subset = train)
               Type of random forest: classification
                     Number of trees: 1500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 17.98%
Confusion matrix:
      0    1 class.error
0 14695  837  0.05388875
1  2760 1708  0.61772605
```

Test error is 17.86% for Bagging with 1500 trees.Thus there is no much improvement in test error rate by increasing the number of trees from 500 to 1500 which can also inferred in the below graph.



Bagging – Mtry:24,ntree:500

```
> print(tree_randomforest_24.data)

Call:
 randomForest(formula = default_payment_next_month ~ ., data = data,      ntree = 500, mtry = 24, subset = train)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 24

        OOB estimate of  error rate: 18.15%
Confusion matrix:
      0    1 class.error
0 14637  895  0.05762297
1  2736 1732  0.61235452
```

Test error rate is 18.25%

Bagging –Mtry 24,ntrees 1500

```
> print(tree_randomforest_24_1500.data)

Call:
 randomForest(formula = default_payment_next_month ~ ., data = data,      ntree = 1500, mtry = 24, subset = train)
               Type of random forest: classification
                     Number of trees: 1500
No. of variables tried at each split: 24

        OOB estimate of  error rate: 18.08%
Confusion matrix:
      0    1 class.error
0 14656  876  0.05639969
1  2740 1728  0.61324978
```

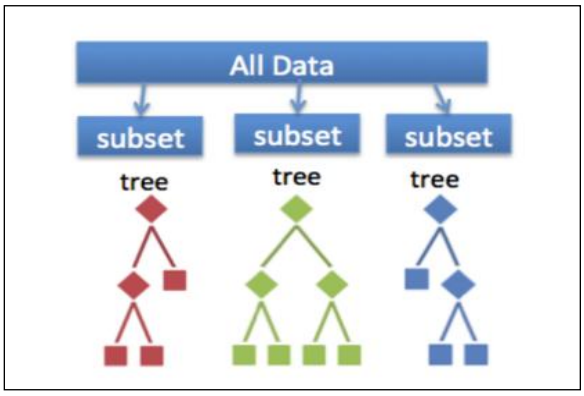
Test Error rate is 18.35%

Below table illustrates Test error obtained in Bagging method

Mtry	Ntrees	Test Error	Accuracy(Test)
Default-4	**500**	**0.179**	**82.1%**
Default-4	**1500**	**0.1786**	**82.14%**
24	**500**	**0.1825**	**81.75%**
24	**1500**	**0.1835**	**81.65%**

- Thus we find that test error reduced slightly when number of trees increased from 500 to 1500 in-case of default mtry. However, for mtry 24, test error increased when number of trees increased.

5. DECISION TREE-RANDOMFOREST



Random forest is generated similar to Bagging where we create number of decision trees based on random selection of data and attributes. Each classifier in the ensemble is a decision tree and during the classification we obtain tree votes from each classifier and most popular vote is assigned to the unknown sample.

By default, randomforest() uses sqrt(attributes count) variables when building a random forest of classification trees.

OOB(Out-of Bag) error estimate is calculated for cases which were not used for tree building process.

We will focus on random forest analysis for mtry=4,5,6 and ntrees=500.

In the below analysis, we build the randomforest with the training data and test the model constructed using our test data.

Mtry=4 ntree=500

```
> print(tree_randomforest_4.data)

Call:
 randomForest(formula = default_payment_next_month ~ ., data = data,       ntree = 500, mtry = 4, subset = train)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 18.09%
Confusion matrix:
      0    1 class.error
0 14689  843  0.05427505
1  2774 1694  0.62085944
```

Thus the above model has OOB 18.09% error rate so it can predict the class with 81.91% accuracy

After conducting our prediction using test dataset, Test error rate =0.1792

Mtry=5,ntree=500

```
> print(tree_randomforest_5.data)

Call:
 randomForest(formula = default_payment_next_month ~ ., data = data,
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 5

        OOB estimate of  error rate: 18.07%
Confusion matrix:
      0    1 class.error
0 14672  860  0.05536956
1  2753 1715  0.61615936
```

Test error rate =0.1809

Mtry=6,ntree=500

```
> print(tree_randomforest_6.data)

Call:
 randomForest(formula = default_payment_next_month ~ ., data = data,
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 6

        OOB estimate of  error rate: 18.2%
Confusion matrix:
      0    1 class.error
0 14662  870  0.05601339
1  2769 1699  0.61974038
```

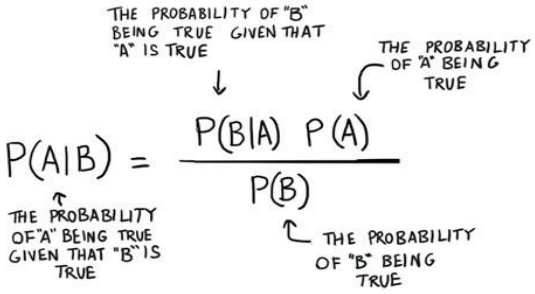
Test error rate =0.1813

Mtry	Ntrees	Test Error	Accuracy(Test)	OOB Rate
4	**500**	**0.1792**	**82.08%**	**18.09%**
5	**500**	**0.1809**	**81.91%**	**18.07%**
6	**500**	**0.1813**	**81.87%**	**18.2%**

- Randomforest with mtry=4 resulted in the lowest test error on comparison with other mtry.

6. NAIVE BAYES CLASSIFIER

This classification technique is based on Bayes theorem with an assumption of independence among predictors. This model assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes also outperforms highly sophisticated classification methods.



Pros:

- Computationally fast
- Works well with high dimensions
- Can be easily trained using a small data set

Cons:

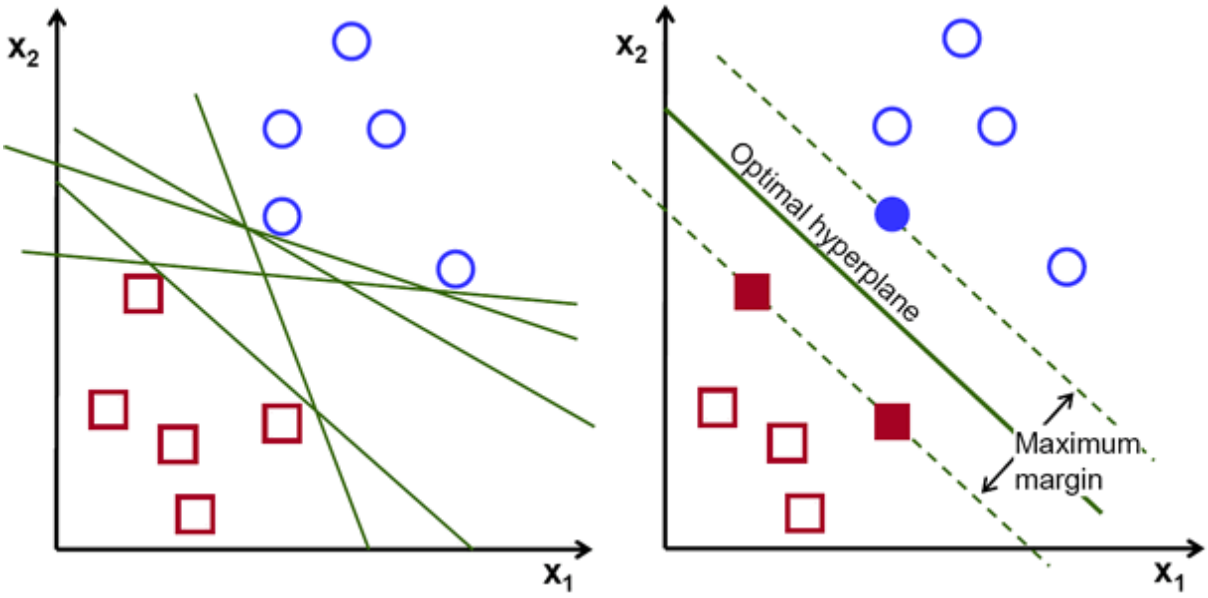
- Zero conditional probability problem for features having zero frequency, the total probability also becomes zero. There are several sample correction techniques to fix this problem such as Laplacian Correction.
- Another disadvantage is the very strong assumption of independence class features, which is near to impossible to find such data sets in real life.

Naive Bayes Results:

Train Error	Test Error	Accuracy
0.3528	**0.3508**	**64.92**

Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for both classification and regression challenges, mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.



In addition to performing linear classification, SVM can efficiently perform a non-linear classification using kernel trick, implicitly mapping points into high-dimensional feature spaces and finds the hyper-plane that separates the two classes. Decision boundaries are dependent on support vectors. SVM is robust to outliers.

Tuning Parameters:

Kernel: There are different kernels to choose for separating hyper-plane based on the dataset. Some of them are Linear, Radial, Polynomial, etc.

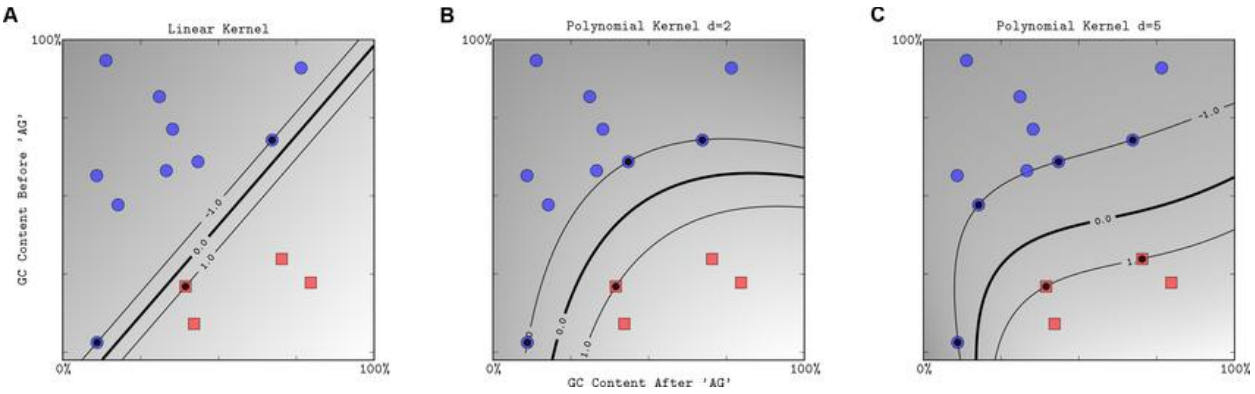
Cost: The cost parameter trades off correct classification of training set against maximization of the decision function's margin. For large values of Cost, SVM will choose a smaller-margin hyper-plane if

that does a better job at getting all the training points classified correctly. For a very small value of Cost, SVM look for a larger-margin separating hyper-plane, even that hyper-plane misclassifies more points.

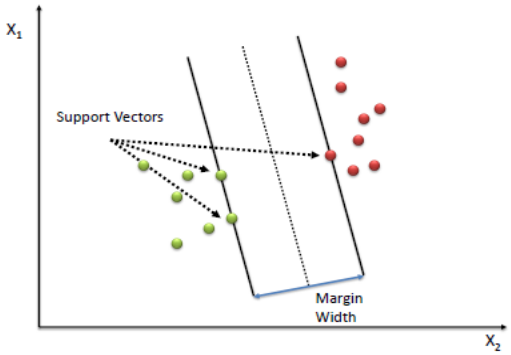
Gamma: Low value means that every point has a far reach and conversely, high value means that every point has close reach i.e. if gamma has a very high value, then the decision boundary is just going to be dependent upon the points that are very close to the line which results in ignoring some of the points that are far from the decision boundary. If gamma value is low even far away points are considered for decision boundary.

Default value of gamma is the inverse of number of attributes in a dataset.

Degree: The degree of a polynomial controls the flexibility of decision boundary. The lowest degree polynomial with degree equal to one is same as linear kernel. As the degree increases, hyper-plan is more flexible for separating the classes.



Support vectors are the data points that are closer to the hyper-plane and influence the position and orientation of the hyper-plane. Using these support vectors, we maximize the margin of the classifier.



Deleting the support vectors will change the position of the hyper-plane. These are the points that helps to build the Support Vector Machine (SVM).

SVM for Default of Credit Card Clients

7. SVM LINEAR

We use library(e1071) for SVM function in R.

SVM function:

svmfit=svm(default_payment_next_month~.,data=data,subset=train,kernel="linear")

default_payment_next_month is the predicate for default of credit card clients dataset.

For knowing the Train error:

First we predict the label for Train dataset using predict function from model we build, later will test again the actual value.

default_payment_next_month_pred_svm_train=predict(svmfit,data.train)

table function is used to provide confusion matrix.

table(default_payment_next_month_pred_svm_train,default_payment_next_month.train)

Below statement will provide Train error rate

mean(default_payment_next_month_pred_svm_train!=default_payment_next_month.train)

Similarly we follow same steps for Test error rate

Error rates when SVM performed with below parameters:

Kernel = linear, cost = 0.001

COST	SUPPORT VECTORS	TRAIN ERROR	TESTERROR
0.001	**9188**	**0.215**	**0.2118**

Kernel = linear, cost = 0.01

COST	SUPPORT VECTORS	TRAIN ERROR	TESTERROR
0.01	**9228**	**0.19045**	**0.1907**

Kernel = linear, cost = 0.1

COST	SUPPORT VECTORS	TRAIN ERROR	TESTERROR
0.1	**9487**	**0.1904**	**0.1907**

Kernel = linear

COST	SUPPORT VECTORS	TRAIN ERROR	TESTERROR
Default(1)	**9772**	**0.1898**	**1903**

Observations:

- As cost increased number of support vectors also increased.
- With High cost SVM concentrates on correct classification, so as cost increased Train Error rate decreased.
- With decrease in Train Error rate, Test error rate also decreased.

8. SVM RADIAL

Error rates when SVM performed with below parameters

Kernel = radial, cost = 0.1, gamma = 0.01

COST	GAMMA	TRAIN ERROR	TESTERROR
0.1	**0.01**	**0.18965**	**0.19**

Kernel = radial, cost = 0.1

COST	GAMMA	TRAIN ERROR	TESTERROR
0.1	**Default(0.0417)**	**0.18195**	**0.1852**

Kernel = radial, cost = 0.1, gamma = 0.07

COST	GAMMA	TRAIN ERROR	TESTERROR
0.1	**0.07**	**0.1809**	**0.1844**

Kernel = radial, cost = 0.1, gamma = 0.1

COST	GAMMA	TRAIN ERROR	TESTERROR
0.1	**0.1**	**0.18175**	**0.185**

Kernel = radial, cost = 0.1, gamma = 1

COST	GAMMA	TRAIN ERROR	TESTERROR
0.1	**1**	**0.2234**	**0.2168**

Kernel = radial, cost = 0.1, gamma = 3

COST	GAMMA	TRAIN ERROR	TESTERROR
0.1	**3**	**0.2168**	**0.2168**

Kernel = radial, cost = 1, gamma = 0.01

COST	GAMMA	TRAIN ERROR	TESTERROR
1	**0.01**	**0.1807**	**0.1821**

Kernel = radial, cost = 1

COST	GAMMA	TRAIN ERROR	TESTERROR
1	**Default(0.0417)**	**0.1729**	**0.1812**

Kernel = radial, cost = 1, gamma = 0.07

COST	GAMMA	TRAIN ERROR	TESTERROR
1	**0.07**	**0.16755**	**0.1827**

Kernel = radial, cost = 1, gamma = 0.1

COST	GAMMA	TRAIN ERROR	TESTERROR
1	**0.1**	**0.16235**	**0.1823**

Kernel = radial, cost = 1, gamma = 1

COST	GAMMA	TRAIN ERROR	TESTERROR
1	**1**	**0.06305**	**0.2073**

Kernel = radial, cost = 1, gamma = 3

COST	GAMMA	TRAIN ERROR	TESTERROR
1	**3**	**0.02925**	**0.2184**

Kernel = radial, cost = 10, gamma = 0.01

COST	GAMMA	TRAIN ERROR	TESTERROR
10	**0.01**	**0.1754**	**0.1798**

Kernel = radial, cost = 10

COST	GAMMA	TRAIN ERROR	TESTERROR
10	**Default(0.0417)**	**0.15575**	**0.1835**

Kernel = radial, cost = 10, gamma = 0.07

COST	GAMMA	TRAIN ERROR	TESTERROR
10	**0.07**	**0.13315**	**0.1885**

Kernel = radial, cost = 10, gamma = 0.1

COST	GAMMA	TRAIN ERROR	TESTERROR
10	**0.1**	**0.1131**	**0.1939**

Kernel = radial, cost = 10, gamma = 1

COST	GAMMA	TRAIN ERROR	TESTERROR
10	**1**	**0.0131**	**0.2332**

Kernel = radial, cost = 10, gamma = 3

COST	GAMMA	TRAIN ERROR	TESTERROR
10	**3**	**0.00165**	**0.2319**

Observations:

- As cost increased Train error decreased.
- For cost = 0.1 we can see from gamma 0.01 to 0.07 Train and Test error rates decreased and gamma 0.1 to 3 Train and Test increased. Here gamma dominated cost parameter.

  For low value of cost SVM concentrates on maximizing the margin, even though there is some misclassification. For high value of gamma only near values are considered to make decision boundaries which may lead to choose hyper-plane with less margin and increase in Train and Test error rates.
- For cost 1 and 10, as gamma increased Train error decreased i.e., classification for train data set is done correctly (misclassification is very less, captured almost all points of train data). Here cost dominated gamma parameter. Test error increased, for high cost we may have less may have irregular decision boundary (over-fitting the data) and chances of misclassification for test data set is more.

9. SVM POLYNOMIAL

Error rates when SVM performed with below parameters:

Kernel = polynomial, cost = 0.1

COST	DEGREE	GAMMA	TRAIN ERROR	TESTERROR
0.1	**Default(3)**	**Default(0.0417)**	**0.19865**	**0.2013**

Kernel = polynomial, cost = 0.1, gamma = 0.07

COST	DEGREE	GAMMA	TRAIN ERROR	TESTERROR
0.1	**Default(3)**	**0.07**	**0.18925**	**0.194**

Kernel = polynomial, cost = 0.1, gamma = 0.1

COST	DEGREE	GAMMA	TRAIN ERROR	TESTERROR
0.1	**Default(3)**	**0.1**	**0.18135**	**0.1927**

Kernel = polynomial, cost = 0.1, Degree = 2

COST	DEGREE	TRAIN ERROR	TESTERROR
0.1	**2**	**0.2203**	**0.216**

Kernel = polynomial, cost = 0.1

COST	DEGREE	TRAIN ERROR	TESTERROR
0.1	**Default(3)**	**0.19865**	**0.2013**

Kernel = polynomial, cost = 0.1, Degree = 4

COST	DEGREE	TRAIN ERROR	TESTERROR
0.1	**4**	**0.20275**	**0.2065**

Kernel = polynomial, cost = 0.1, Degree = 5

COST	DEGREE	TRAIN ERROR	TESTERROR
0.1	**5**	**0.19585**	**0.2065**

Kernel = polynomial, cost = 1, Degree = 2

COST	DEGREE	TRAIN ERROR	TESTERROR
1	**2**	**0.213**	**0.2138**

Kernel = polynomial, cost = 1

COST	DEGREE	TRAIN ERROR	TESTERROR
1	**Default(3)**	**0.18395**	**0.1923**

Kernel = polynomial, cost = 1, Degree = 4

COST	DEGREE	TRAIN ERROR	TESTERROR
1	**4**	**0.1818**	**0.1995**

Kernel = polynomial, cost = 10, Degree = 2

COST	DEGREE	TRAIN ERROR	TESTERROR
10	**2**	**0.20655**	**0.2079**

Kernel = polynomial, cost = 10

COST	DEGREE	TRAIN ERROR	TESTERROR
10	**Default(3)**	**0.1682**	**0.1932**

Kernel = polynomial, cost = 10, Degree = 4

COST	DEGREE	TRAIN ERROR	TESTERROR
10	**4**	**0.15375**	**0.2015**

Observations:

- For cost 0.1, as gamma increased Train and Test error rates decreased. Best results are produced with default (3) degree.
- For costs 1 and 10, as degree increased Train error rate decreased.
- As cost increased Train error rates decreased.

10. COMPARISON OF MULTIPLE CLASSIFICATION TECHNIQUES

(a) Comparison - Decision Tree Techniques

The below graph represents the comparison between various decision tree techniques such as Holdout, Bagging and Random forest. The Bagging technique has the least error rate followed by Random forest and Holdout method. However, all these techniques show only negligible differences in their error rates. Overall, Decision tree techniques provides 82% average accuracy rate for this data set.

(b) Comparison – Support Vector Machines Techniques

The below graph illustrates the comparison between different kernels of Support Vector Machine techniques such as Linear, Radial and Polynomial. All the kernels are compared against cost 1, out of which the Radial kernel has outperformed others with the least error rate, followed by Linear and Polynomial kernels. Overall, Support Vector Machines provides 81% average accuracy rate for this data set which is slightly lesser then the Decision Tree techniques.

(c) Comparison – Multiple Classification Techniques

The below graph illustrates the comparison of all the techniques such as Holdout, Bagging , Random forest, Support vector machines - Linear,Radial,Polynomial and Naive Bayes. All the techniques except Naive Baiyes follow a similar trend with negligible differences between their error rates.However, Naive Bayes is roughly twice the error rate of other techniques and reports with the least accuracy rate of 65%.On the otherhand, Bagging provides the highest accuracy rate of 82% . Overall, all the techniques except Naive Bayes holds good accuracy rates for this data set.

11. POTENTIAL PERFORMANCE ISSUES AND POSSIBLE FUTURE STUDY

Performance Issues:

Support vector machines with linear kernel exhibited slow performance during execution and the possible reasons are highlighted below:

- Larger dataset
- Increased number of support vectors
- Intensity of the algorithm

Possible Future Study

Emphasis on Data Quality (Preprocessing):

Data quality rules can be applied to the data set for obtaining cleansed and deduplicated data. The data quality check includes the following process:

- **Data profiling & assessment:**

  Gather the statistics of existing data set and derive cleansing and matching rules.

- **Data cleansing & standardization:**

  Perform data cleansing and standardization wherever applicable.

- **Data Deduplication:**

  Perform data deduplication in case of redundant data set.

Test with other different data mining techniques

- Benchmark their performances

12. PROJECT CONCLUSION

Best classification technique:

Bagging technique with highest accuracy rate of **82%** is considered the best technique for this data set. The possible reasons are listed below:

- Reduces variance and prevents overfitting.
- Majority of the project dataset attributes are qualitative and this technique easily handles qualitative (categorical) features.

Poor classification technique:

Naive Bayes technique with least accuracy rate of **65%** is considered the poor technique for this data set.

The possible reasons are listed below:

- Relies on independence assumption and performs poorly if this assumption is not met.
- Naive Bayes best suits for text classification and the project data set is not based on text classification.

13. APPENDIX

1. R - Source code with all classification techniques **– Rcode-Classification.txt**
2. R - Source code with all data analysis - **Rcode-Data Analysis.txt**