Credit Card Risk Assessment-Kaggle Dataset

SUBHAM SENGUPTA

Executive summary

We are given a dataset on credit card risk assessment by a bank where given certain features the financial organisation wants to get well informed about their customer behaviours so that they can strategize their policy well enough for the next five years.

Let us begin by loading the libraries in R (the description of the libraries is attached along with):

#library

library(tidyverse) : For graphics through ggplot2, and EDA

library(dplyr): Required for EDA(Note that 'tidyverse' also contains 'dplyr')

library(caret): For predictive Data Analysis and data Pre-processing

library(randomForest): Implementing randomForest algorithm

library(e1071): Implementing svm algorithm

library(rpart): Implementing decision tree (rpart) algorithm

Now we import the dataset in R, and have a brief about the variables used:

1. ID: It is the serial number for each customer given in the table.

2. LIMIT_BAL: The balance limit for each credit card

3. SEX: categorical variable.

4. EDUCATION: categorical variable about the education status.

5. MARRIAGE: marital status, also a categorical variable.

6. AGE: (rounded off to nearest integer)

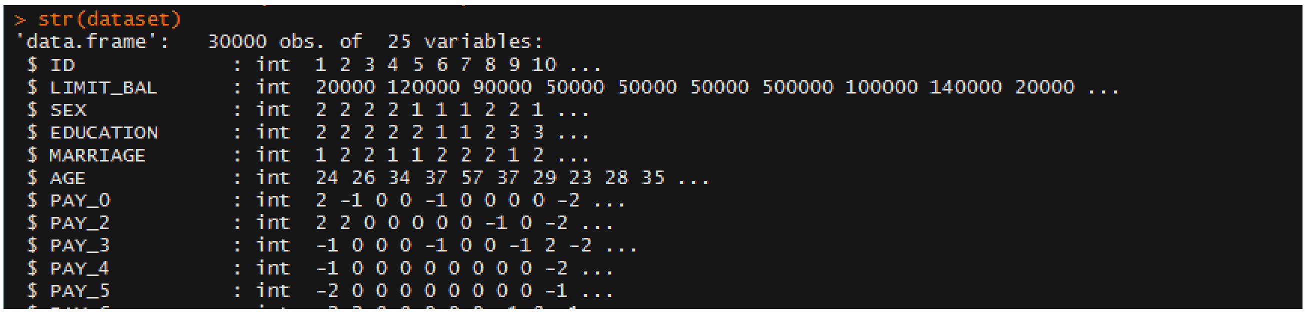
7. 'PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6': Most likely the payment status of the customer for the last 6 terms.

8.' BILL_AMT1, BILL_AMT2, BILL_AMT3, BILL_AMT4, BILL_AMT5, BILL_AMT6': Most likely the amount of fund credited in the last 6 terms.

9.' PAY_AMT1, PAY_AMT2, PAY_AMT3, PAY_AMT4, PAY_AMT5, PAY_AMT6': Most likely the amount paid back to the financial organisation in the last 6 terms.

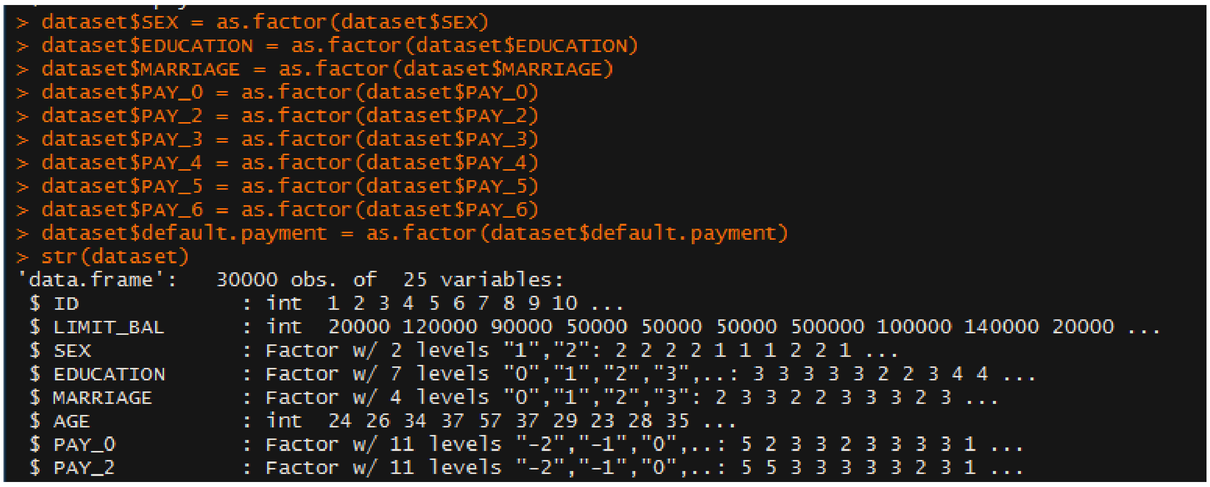
10. default.payment: binary variable classifying the customers into good or bad credit risks. '0' means not default and '1' means default.

Let us have a look at the structure of the dataset.



```
> str(dataset)
'data.frame':	30000 obs. of  25 variables:
 $ ID          : int  1 2 3 4 5 6 7 8 9 10 ...
 $ LIMIT_BAL   : int  20000 120000 90000 50000 50000 50000 500000 100000 140000 20000 ...
 $ SEX         : int  2 2 2 2 1 1 1 2 2 1 ...
 $ EDUCATION   : int  2 2 2 2 2 1 1 2 3 3 ...
 $ MARRIAGE    : int  1 2 2 1 1 2 2 2 1 2 ...
 $ AGE         : int  24 26 34 37 57 37 29 23 28 35 ...
 $ PAY_0       : int  2 -1 0 0 -1 0 0 0 0 -2 ...
 $ PAY_2       : int  2 2 0 0 0 0 0 -1 0 -2 ...
 $ PAY_3       : int  -1 0 0 0 -1 0 0 -1 2 -2 ...
 $ PAY_4       : int  -1 0 0 0 0 0 0 0 0 -2 ...
 $ PAY_5       : int  -2 0 0 0 0 0 0 0 0 -1 ...
```

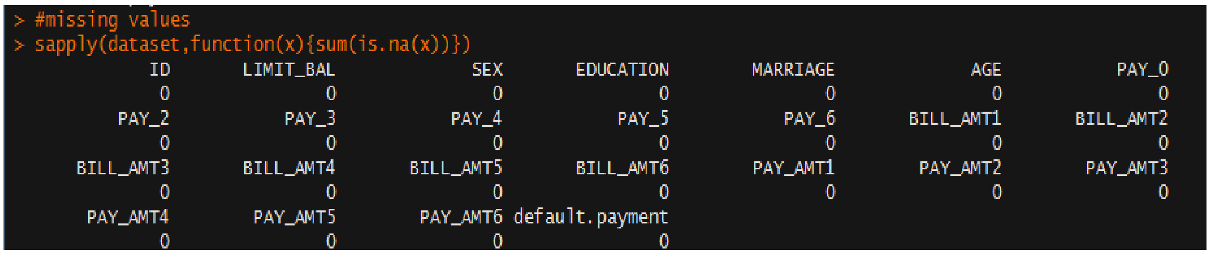
Clearly we can observe that all the variables are of integer type, so we need to convert the required features to categorical



```
> dataset$SEX = as.factor(dataset$SEX)
> dataset$EDUCATION = as.factor(dataset$EDUCATION)
> dataset$MARRIAGE = as.factor(dataset$MARRIAGE)
> dataset$PAY_0 = as.factor(dataset$PAY_0)
> dataset$PAY_2 = as.factor(dataset$PAY_2)
> dataset$PAY_3 = as.factor(dataset$PAY_3)
> dataset$PAY_4 = as.factor(dataset$PAY_4)
> dataset$PAY_5 = as.factor(dataset$PAY_5)
> dataset$PAY_6 = as.factor(dataset$PAY_6)
> dataset$default.payment = as.factor(dataset$default.payment)
> str(dataset)
'data.frame':	30000 obs. of  25 variables:
 $ ID          : int  1 2 3 4 5 6 7 8 9 10 ...
 $ LIMIT_BAL   : int  20000 120000 90000 50000 50000 50000 500000 100000 140000 20000 ...
 $ SEX         : Factor w/ 2 levels "1","2": 2 2 2 2 1 1 1 2 2 1 ...
 $ EDUCATION   : Factor w/ 7 levels "0","1","2","3",..: 3 3 3 3 3 2 2 3 4 4 ...
 $ MARRIAGE    : Factor w/ 4 levels "0","1","2","3": 2 3 3 2 2 3 3 3 2 3 ...
 $ AGE         : int  24 26 34 37 57 37 29 23 28 35 ...
 $ PAY_0       : Factor w/ 11 levels "-2","-1","0",..: 5 2 3 3 2 3 3 3 3 1 ...
 $ PAY_2       : Factor w/ 11 levels "-2","-1","0",..: 5 5 3 3 3 3 3 2 3 1 ...
```

Thus the necessary changes have been made.

Checking for missing values



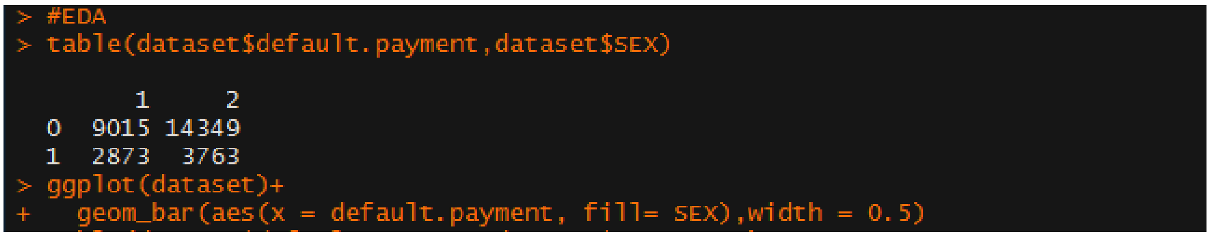
```
> #missing values
> sapply(dataset,function(x){sum(is.na(x))})
             ID       LIMIT_BAL             SEX       EDUCATION        MARRIAGE             AGE           PAY_0 
              0               0               0               0               0               0               0 
          PAY_2           PAY_3           PAY_4           PAY_5           PAY_6       BILL_AMT1       BILL_AMT2 
              0               0               0               0               0               0               0 
      BILL_AMT3       BILL_AMT4       BILL_AMT5       BILL_AMT6        PAY_AMT1        PAY_AMT2        PAY_AMT3 
              0               0               0               0               0               0               0 
       PAY_AMT4        PAY_AMT5        PAY_AMT6 default.payment 
              0               0               0               0 
```

There are no missing values in our dataset.

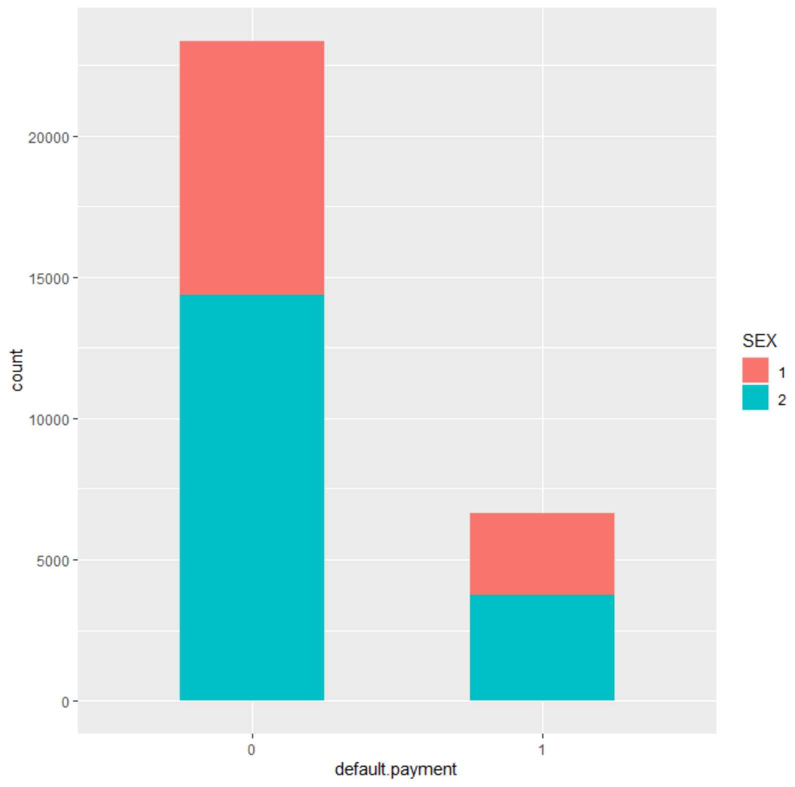
Exploratory data analysis

Brief: EDA or exploratory Data Analysis or descriptive summary of the data is utilizing the given data in hand to draw useful insights, for better understanding as well as to strategise further business value from the dataset. They can be graphical or tabulated to draw quick conclusions. Let us perform some EDA on our given dataset.

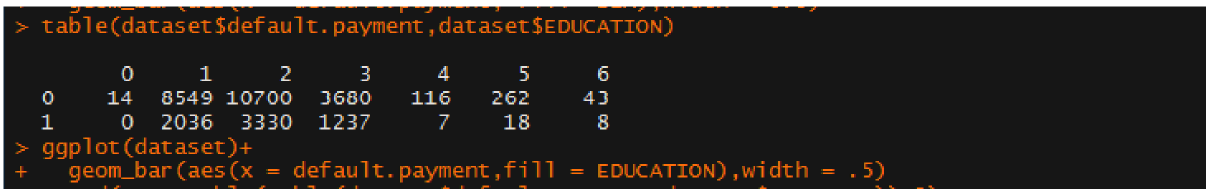
1.



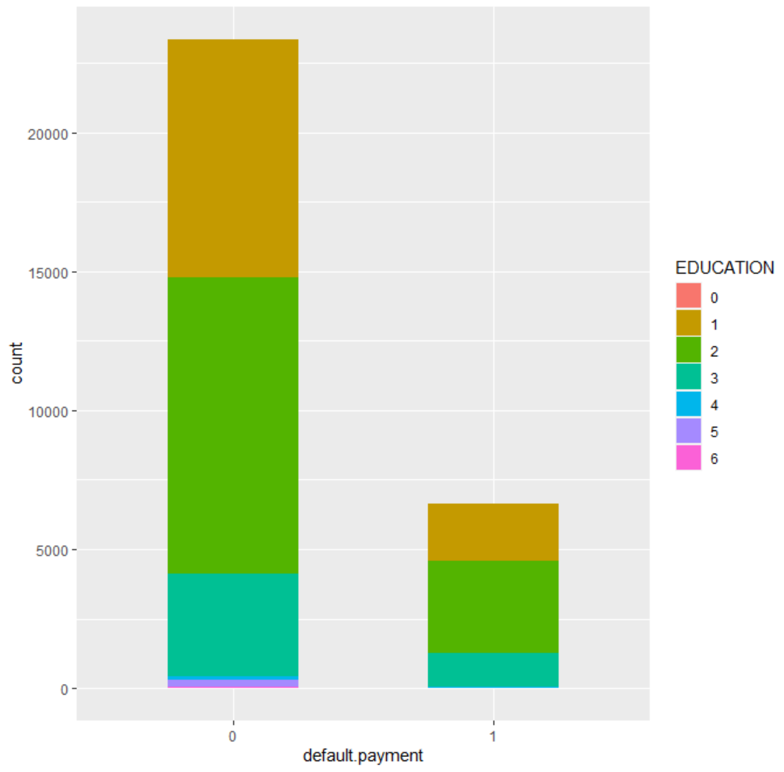
```
> #EDA
> table(dataset$default.payment,dataset$SEX)
   
        1     2
  0  9015 14349
  1  2873  3763
> ggplot(dataset)+
+   geom_bar(aes(x = default.payment, fill= SEX),width = 0.5)
```



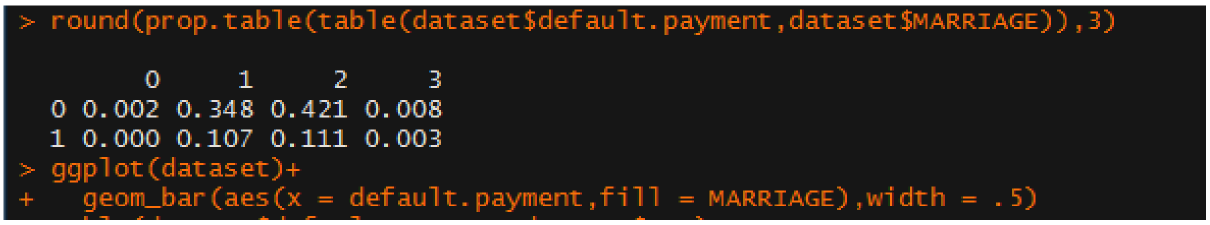
2.



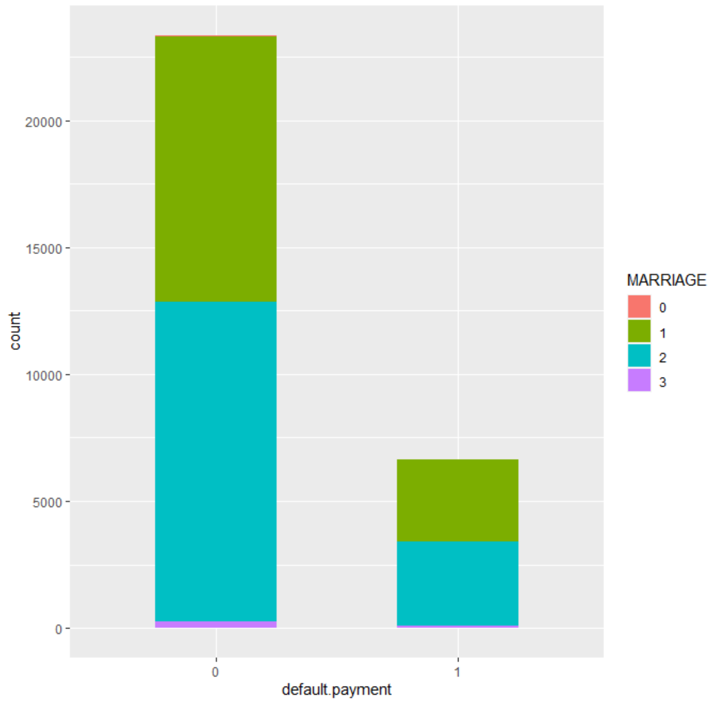
```
> table(dataset$default.payment,dataset$EDUCATION)

        0     1     2     3     4     5     6
  0    14  8549 10700  3680   116   262    43
  1     0  2036  3330  1237     7    18     8
> ggplot(dataset)+
+   geom_bar(aes(x = default.payment,fill = EDUCATION),width = .5)
```



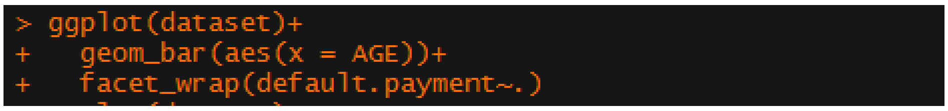
3.



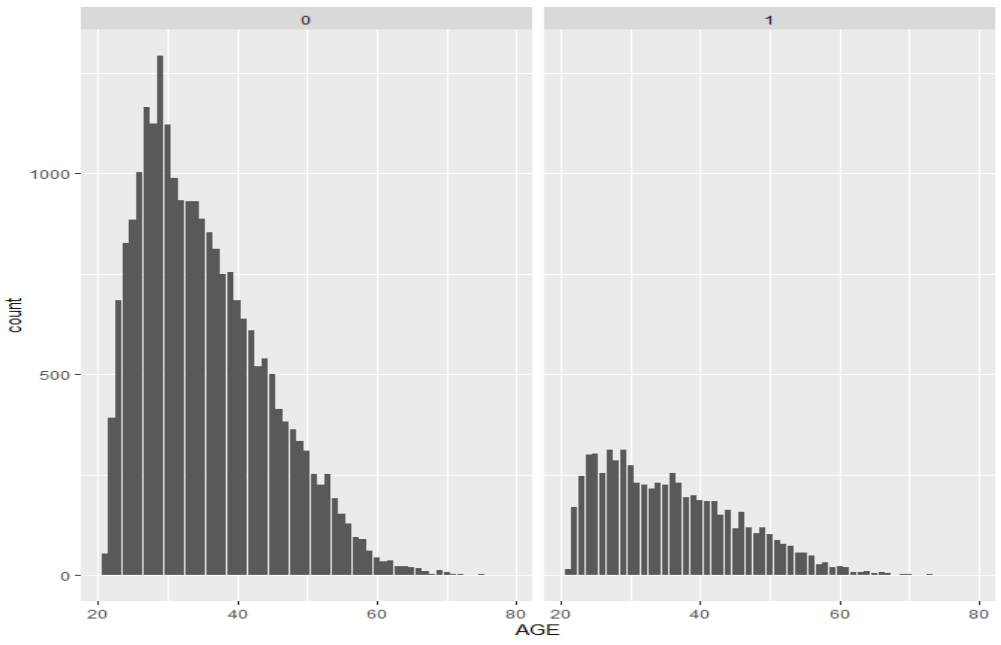
```
> round(prop.table(table(dataset$default.payment,dataset$MARRIAGE)),3)

        0     1     2     3
  0 0.002 0.348 0.421 0.008
  1 0.000 0.107 0.111 0.003
> ggplot(dataset)+
+   geom_bar(aes(x = default.payment,fill = MARRIAGE),width = .5)
```



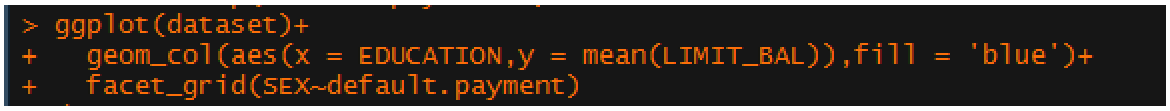
4.



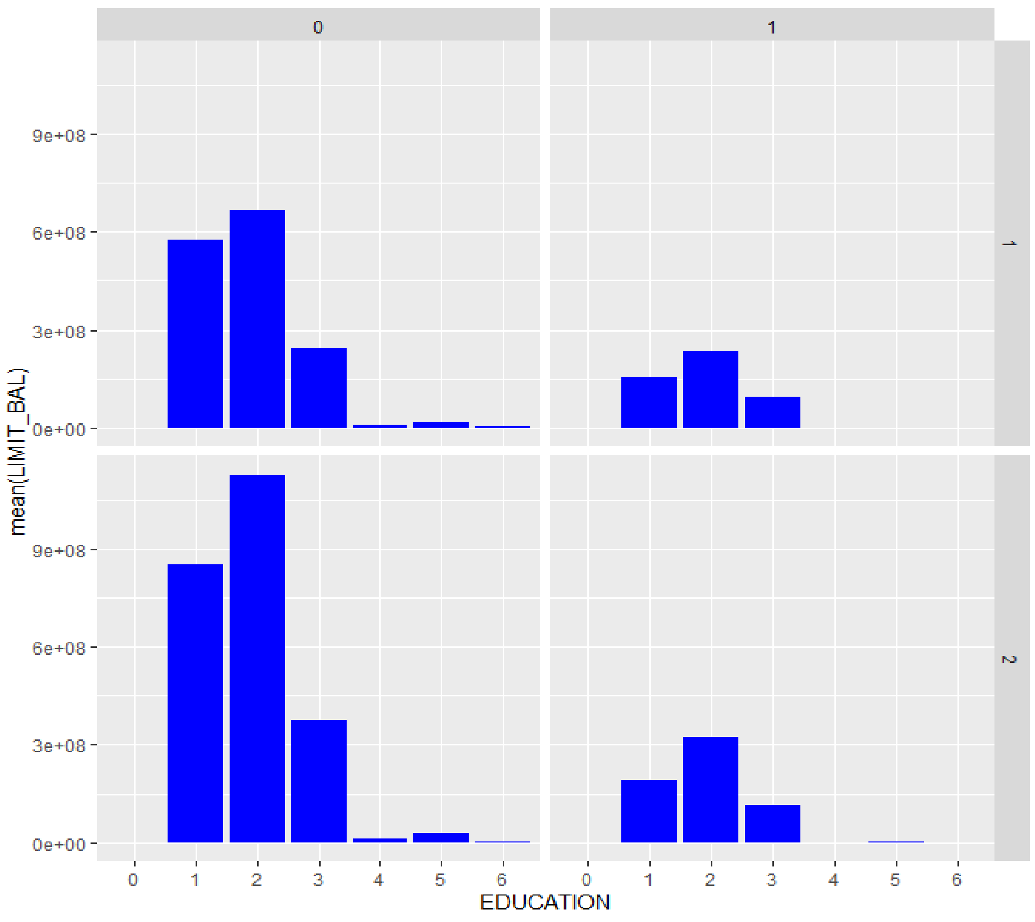
```
> ggplot(dataset)+
+   geom_bar(aes(x = AGE))+
+   facet_wrap(default.payment~.)
```



5.



```
> ggplot(dataset)+
+   geom_col(aes(x = EDUCATION,y = mean(LIMIT_BAL)),fill = 'blue')+
+   facet_grid(SEX~default.payment)
```



<u>Insights from EDA</u>

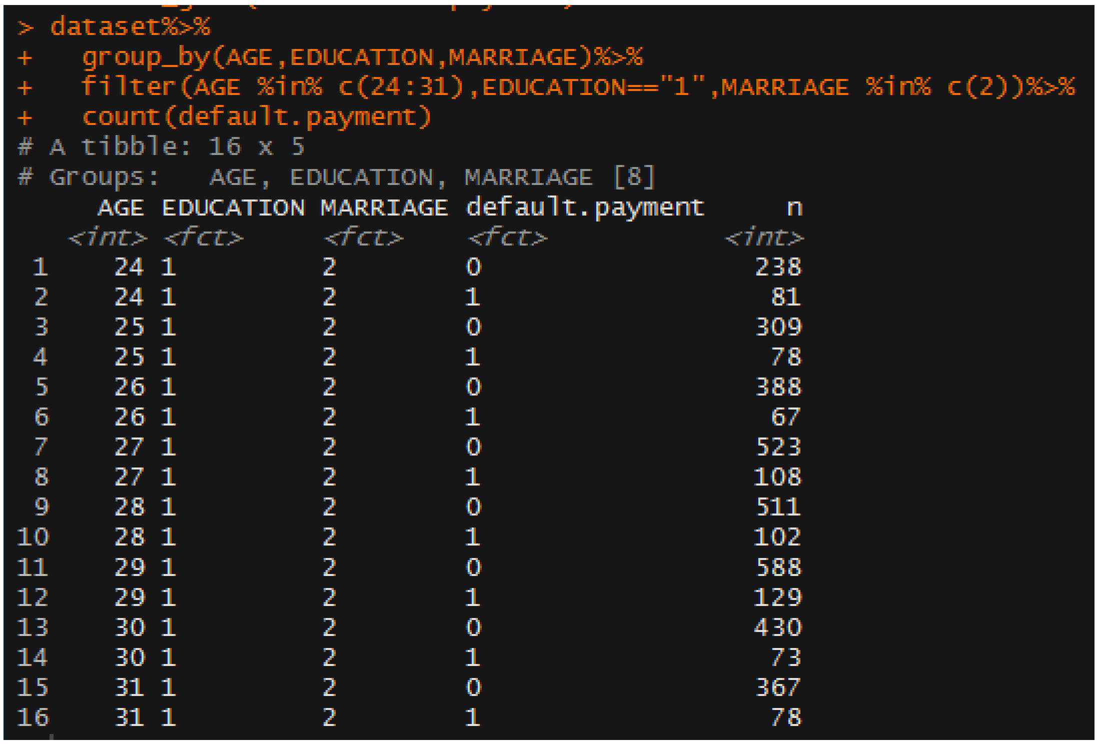
1. Clearly SEX category ‘2’ has more number of accounts in the financial organisation, but both the categories have almost the same proportion of good and bad credit risks.
2. EDUCATION categories ‘1’,’2’ and ‘3’ are more prevalent among the all the categories for all the customers. Out of the three categories, category ‘1’ includes most customers and also has a less proportion of bad credit risk.
3. Category ‘1’,’2’ are most prevalent when it comes to the feature MARRAIGE. Out of the two customers falling in ‘2’ is less likely to include bad credit risks.
4. When is come to AGE, customers falling in between age groups ’24-31’ are most likely to get a credit card, as shown by the plot above.
5. It is a combined case study about the average limit balance provided to the customers, which clearly depicts the average value allotted to good risk customers are higher.

Business Insight

‘One clear business insight which can be taken from the above analysis is that the financial organisation can target customers of:

- AGE group ’24-31’
- EDUCATION CATEGORY ‘1’
- MARRIAGE STATUS ‘2’

They are most likely to not default their credit card bills’. It is clearly depicted by the following:

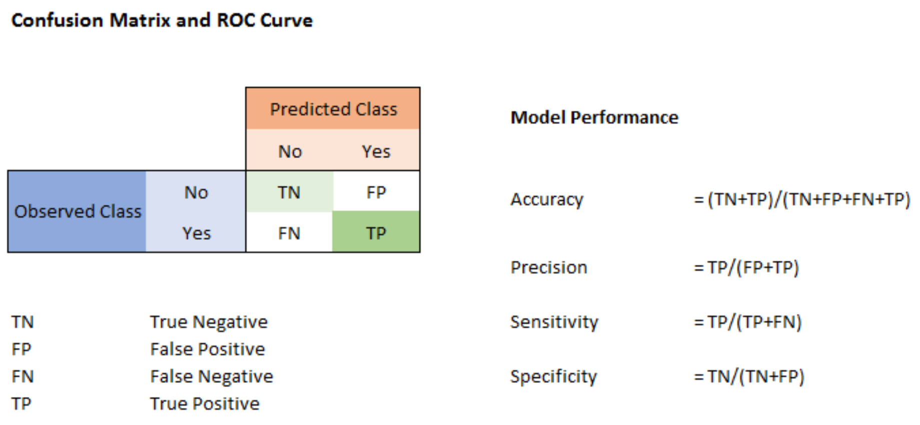


```
> dataset%>%
+   group_by(AGE,EDUCATION,MARRIAGE)%>%
+   filter(AGE %in% c(24:31),EDUCATION=="1",MARRIAGE %in% c(2))%>%
+   count(default.payment)
# A tibble: 16 x 5
# Groups:   AGE, EDUCATION, MARRIAGE [8]
     AGE EDUCATION MARRIAGE default.payment     n
   <int> <fct>     <fct>    <fct>           <int>
 1    24 1         2        0                 238
 2    24 1         2        1                  81
 3    25 1         2        0                 309
 4    25 1         2        1                  78
 5    26 1         2        0                 388
 6    26 1         2        1                  67
 7    27 1         2        0                 523
 8    27 1         2        1                 108
 9    28 1         2        0                 511
10    28 1         2        1                 102
11    29 1         2        0                 588
12    29 1         2        1                 129
13    30 1         2        0                 430
14    30 1         2        1                  73
15    31 1         2        0                 367
16    31 1         2        1                  78
```

Predictive Data Analysis

Now in order reach the proposed business insight above we build adequate models that accounts for the data in hand and also the related features. Now in the given dataset we have 23 features about the behaviour about the customer and one dependent variable. It is a problem of supervised machine learning classification. Hence we will use supervised machine learning algorithms. Some points to note:

- We will be using all the features in hand, though the business insight proposed to check for the AGE, EDUCATION, and MARRIAGE only but the rest features define the customer behaviour towards the fund, thus they are important too.
- We will use three classification machine learning algorithms.
  1. Ensemble Algorithm (Random Forest)
  2. SVM algorithm
  3. Decision tree algorithm
- Clearly we have seen that this an imbalanced dataset so we will be apply 10-fold cross validation technique for all the three algorithms.
- The model metrics. In a classification algorithm, the model metrics from confusion matrix are given as such


Confusion Matrix and ROC Curve

		Predicted Class	
		No	Yes
Observed Class	No	TN	FP
	Yes	FN	TP

TN	True Negative
FP	False Positive
FN	False Negative
TP	True Positive

Model Performance

Accuracy	=(TN+TP)/(TN+FP+FN+TP)
Precision	=TP/(FP+TP)
Sensitivity	=TP/(TP+FN)
Specificity	=TN/(TN+FP)

Image source: Google images

Now apart from accuracy, we have two more issues to deal with from the business perspective, in credit card risk assessment; the threat due to **FN is generally greater than FP**. And sensitivity or recall measures of such a threat. And to also consider for threat due to FP we will take into account the F1 score

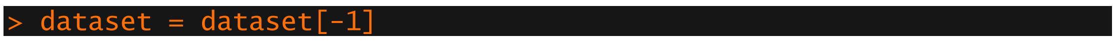
F1 score = harmonic mean (recall, precision)

Therefore the model metrics we will be use to evaluate the model are

1. Accuracy
2. Recall
3. F1 score

Data Pre-processing

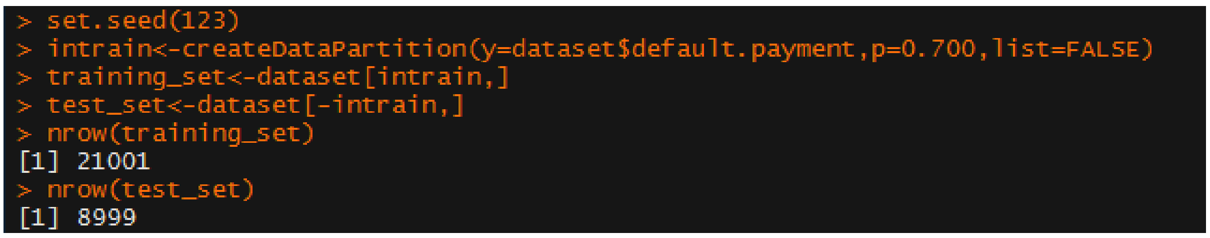
- First and foremost we remove the ID column, which is just a serial number for the customers



```
> dataset = dataset[-1]
```

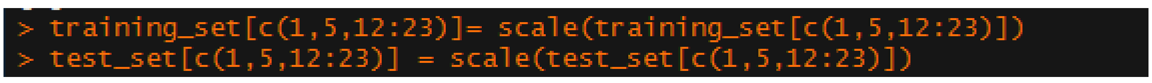
Now the dataset has 24 columns including the dependent variable, starting from the LIMIT_BAL feature.

- Splitting the dataset into training set and testing set:



```
> set.seed(123)
> intrain<-createDataPartition(y=dataset$default.payment,p=0.700,list=FALSE)
> training_set<-dataset[intrain,]
> test_set<-dataset[-intrain,]
> nrow(training_set)
[1] 21001
> nrow(test_set)
[1] 8999
```

- Scaling the features:



```
> training_set[c(1,5,12:23)]= scale(training_set[c(1,5,12:23)])
> test_set[c(1,5,12:23)] = scale(test_set[c(1,5,12:23)])
```

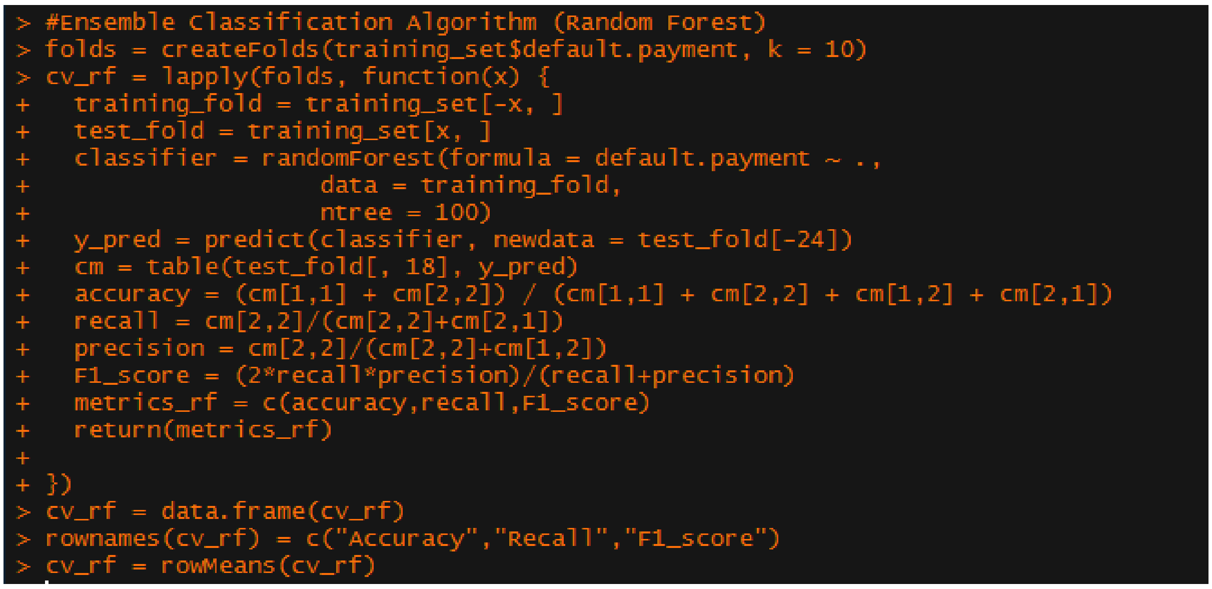
Points to note:

1. We have scaled only the numeric features, as we have already converted the categorical features into factors before.
2. Scaling is necessary for algorithms where GRADIENT DESCENT is considered. Among our three Machine Learning Algorithms, Only SVM algorithm requires feature scaling. Random forest and Decision Tree algorithm can work fine even without scaling. We applied scaling for all algorithms because we want to compare the metrics for the same data set.

1. **Ensemble method (Random Forest Algorithm)**

Random Forest classification algorithm is an ensemble machine learning algorithm, which applies multiple decision trees in its algorithm to predict the outcome, hence supposedly the name ‘Forest’. This ensemble method accounts for the variance of error caused due to one tree and hence has better accuracy.

Random forest is always better when there is imbalance in the dataset, and also the dataset has large dimensionality. Hence the choice of using random forest algorithm for this dataset. The code of the algorithm is as follow.

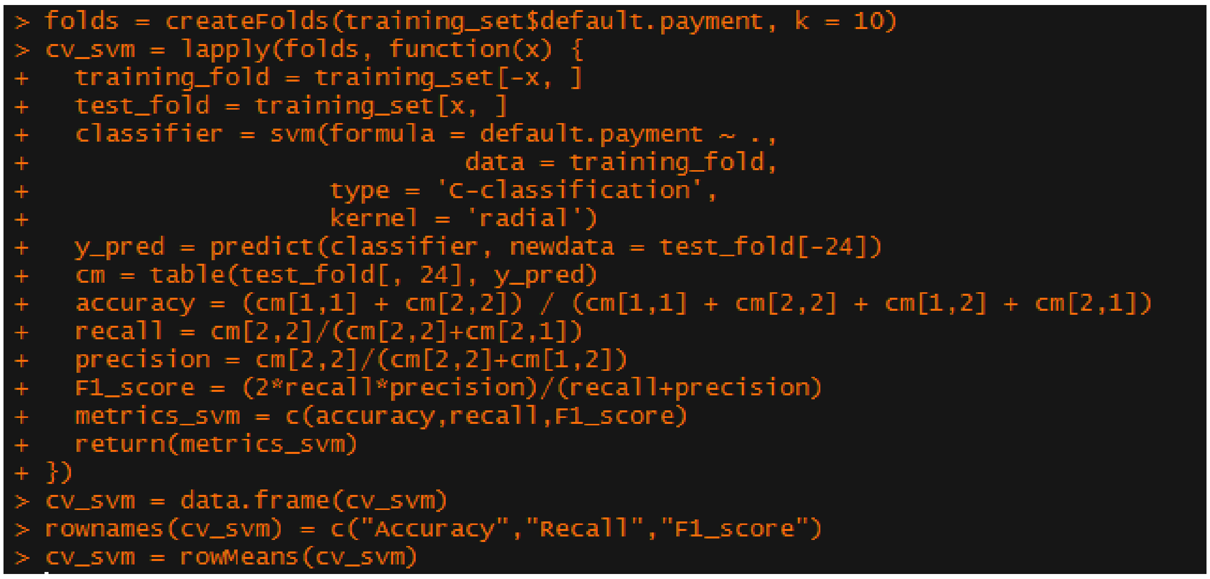


```
> #Ensemble Classification Algorithm (Random Forest)
> folds = createFolds(training_set$default.payment, k = 10)
> cv_rf = lapply(folds, function(x) {
+   training_fold = training_set[-x, ]
+   test_fold = training_set[x, ]
+   classifier = randomForest(formula = default.payment ~ .,
+                             data = training_fold,
+                             ntree = 100)
+   y_pred = predict(classifier, newdata = test_fold[-24])
+   cm = table(test_fold[, 18], y_pred)
+   accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
+   recall = cm[2,2]/(cm[2,2]+cm[2,1])
+   precision = cm[2,2]/(cm[2,2]+cm[1,2])
+   F1_score = (2*recall*precision)/(recall+precision)
+   metrics_rf = c(accuracy,recall,F1_score)
+   return(metrics_rf)
+
+ })
> cv_rf = data.frame(cv_rf)
> rownames(cv_rf) = c("Accuracy","Recall","F1_score")
> cv_rf = rowMeans(cv_rf)
```

2. **Support Vector Machines (SVM)**

Brief: Support Vector machines is a supervised machine learning algorithm, which can be applied for both linear and non-linear (through kernel functions) datasets. Support Vector machines are able to work with large number of features and also prevent overfitting, which is the sole purpose of using SVM for this dataset.

Apparently in SVM algorithm an infinite number of hyperplanes built that separate the data, and the most optimum hyperplane is the one that has largest distance to the nearest training data. For non-linear data, SVM utilises a technique known as the kernel trick, which projects the data in higher dimension to find the optimum hyperplane and then projects back the data in the actual dimension. For this dataset we have used ‘RBF or radial’ kernel. The code for the algorithm is as follows

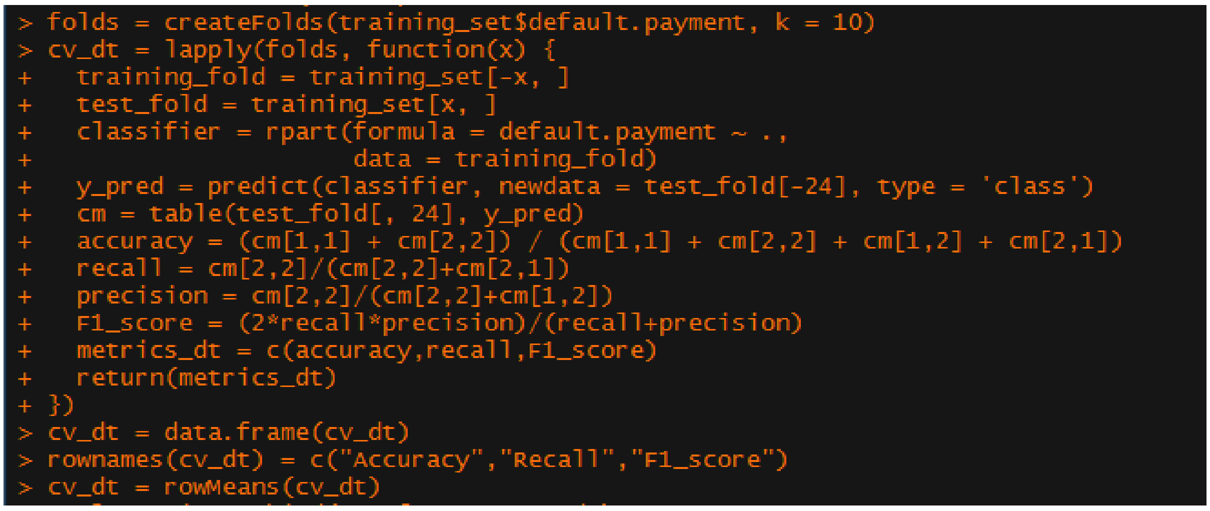

```
> folds = createFolds(training_set$default.payment, k = 10)
> cv_svm = lapply(folds, function(x) {
+   training_fold = training_set[-x, ]
+   test_fold = training_set[x, ]
+   classifier = svm(formula = default.payment ~ .,
+                              data = training_fold,
+                     type = 'C-classification',
+                     kernel = 'radial')
+   y_pred = predict(classifier, newdata = test_fold[-24])
+   cm = table(test_fold[, 24], y_pred)
+   accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
+   recall = cm[2,2]/(cm[2,2]+cm[2,1])
+   precision = cm[2,2]/(cm[2,2]+cm[1,2])
+   F1_score = (2*recall*precision)/(recall+precision)
+   metrics_svm = c(accuracy,recall,F1_score)
+   return(metrics_svm)
+ })
> cv_svm = data.frame(cv_svm)
> rownames(cv_svm) = c("Accuracy","Recall","F1_score")
> cv_svm = rowMeans(cv_svm)
```

3. **Decision Tree**

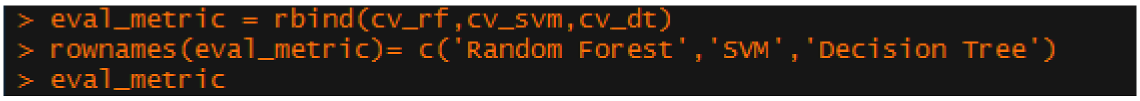
Brief: Now while performing the EDA we have realised that the dependent variable requires many factors or features to reach to a conclusion or insight. The same might be true for the ML algorithm. Taking into account all features we may need to decide the correct split of the dataset to boil down to the right conclusions or in other words we form a tree starting from the entire dataset and then branching and going down to the terminal node which is the classification feature ‘default.payment’. And therefore the use of decision tree algorithm for this dataset is justified.

In a decision tree algorithm we split the dataset based on the information entropy of the features at particular node and certainly form branches to ultimate dive in to the terminal node. The decision tree algorithm is simple to evaluate and also visualize or interpret.

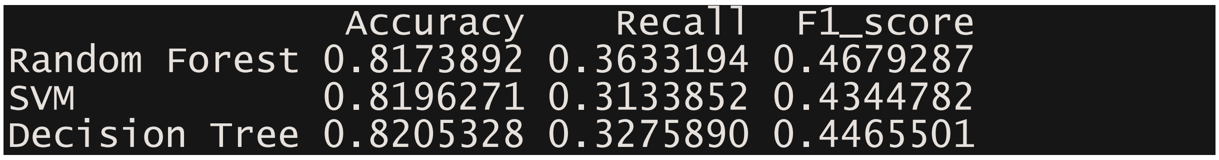
The code for the algorithm is as follows.


```
> folds = createFolds(training_set$default.payment, k = 10)
> cv_dt = lapply(folds, function(x) {
+   training_fold = training_set[-x, ]
+   test_fold = training_set[x, ]
+   classifier = rpart(formula = default.payment ~ .,
+                      data = training_fold)
+   y_pred = predict(classifier, newdata = test_fold[-24], type = 'class')
+   cm = table(test_fold[, 24], y_pred)
+   accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
+   recall = cm[2,2]/(cm[2,2]+cm[2,1])
+   precision = cm[2,2]/(cm[2,2]+cm[1,2])
+   F1_score = (2*recall*precision)/(recall+precision)
+   metrics_dt = c(accuracy,recall,F1_score)
+   return(metrics_dt)
+ })
> cv_dt = data.frame(cv_dt)
> rownames(cv_dt) = c("Accuracy","Recall","F1_score")
> cv_dt = rowMeans(cv_dt)
```

COMPARISON AND CONCLUSION



```
> eval_metric = rbind(cv_rf,cv_svm,cv_dt)
> rownames(eval_metric)= c('Random Forest','SVM','Decision Tree')
> eval_metric
```



```
               Accuracy    Recall  F1_score
Random Forest 0.8173892 0.3633194 0.4679287
SVM           0.8196271 0.3133852 0.4344782
Decision Tree 0.8205328 0.3275890 0.4465501
```

- The accuracy of ‘Decision Tree’ and ‘SVM’ are higher than ‘Random Forest’ algorithm
- But as we have mentioned earlier, from the business evaluation perspective metric ‘Recall’ is most important for evaluating this dataset and then ‘F1_score’.
- **Thus we will recommend the ‘Random Forest’ algorithm for this business problem to generate the correct predictions and utilise the insight stated above.**

NOTE THAT:

- No feature engineering techniques were used before predicting the results.
- Also no dimensionality reduction technique has been used.
- Also the models are based on the entire set of features for better interpretability. WE can always predict the important features first and then apply the model.
- We have only applied 10-fold cross validation for the dataset, and no other model metric evaluation or boosting techniques were used.
- Applying XGBoost ensemble method or grid search techniques can provide better model metrics.