CREDIT RISK ANALYSIS AND PREDICTIVE MODELING

ISHU DEWAN

Graduate Student

Department of Information & Decision Science

University of Illinois at Chicago

Table of Contents

ABSTRACT

This report entails the analysis on German Credit dataset, which has 1000 past credit applicants, described by 30 variables. Each applicant is rated as 'Good' or 'Bad' credit (encoded as 1 and 0 respectively). It's important to look at various characteristics of an applicant and develop a credit scoring rule that can help determine if a loan applicant can be a defaulter at a later stage so that they can go ahead and grant the loan or not.

The models used for predictive modeling are: Decision Trees, AdaBoost and Random Forest.
The performance measures used to evaluate the models are: Confusion Matrix, ROC and Area under the curve (AUC).

The results of the model have been quantified for it to make sense for real businesses, hence it also contains the analysis on the cost incurred by financial institutions assuming:

- Cost incurred by financial institution if the good applicant gets classified as bad applicant = 100 DM
- Cost incurred by financial institution if the bad applicant gets classified as good applicant = 500 DM

The best performing model based on the above measures is then deployed using RShiny as an interactive application.

DATA PREPARATION

Data Description

Variable	Variable Type	Description
CHK_ACCT	Categorical	Status of existing Checking account 0 - < 0DM 1 - > 0DM and < 200DM 2 - >= 200DM 3 - No Checking Account
Duration	Integer	Duration (in month)
History	Categorical	Credit History 0 – no credits taken, and all credits paid back duly 1 – all credits at this bank paid back duly 2 – existing credits, paid back duly till now 3 – delay in paying off in the past 4 – critical amount / other credits existing (not at this bank)
New Car	Categorical	0 – No 1 – Yes
Used Car	Categorical	0 – No 1 – Yes
Furniture	Categorical	0 – No 1 – Yes
Radio/TV	Categorical	0 – No 1 – Yes
Education	Categorical	0 – No 1 – Yes
Retraining	Categorical	0 – No

		1 – Yes
Amount	Integer	Credit Amount
SAV_ACCT	Categorical	Status of existing Saving account 0 - < 100 DM 1 - > 100 DM and < 500 DM 2 - > 500 DM and <1000 DM 3 - >= 1000DM 4 – unknown/no savings account
Employment	Categorical	Present Employment since: 0 – unemployed 1 - < 1 year 2 - > 1 year and < 4 years 3 - > 4 years and < 7 years 4 - >= 7 years
Install_Rate	Integer	Installment rate in percentage of disposable income
Male_Div	Categorical	Male Divorced or Separated 0 – No 1 – Yes
Male_Single	Categorical	0 – No 1 – Yes
Male_Mar_or_Wid	Categorical	Male Married or Widowed 0 – No 1 – Yes
Co-applicant	Categorical	0 – No 1 – Yes
Guarantor	Categorical	0 – No 1 – Yes
Present Resident	Categorical	Present Resident
Real Estate	Categorical	0 – No 1 – Yes
Prop_Unkn_none	Categorical	Unknown or no property 0 – No 1 – Yes
Age	Integer	Age (In Years)
Other_Install	Categorical	Other Installment Plans 0 – No 1 – Yes
Rent	Categorical	Housing as Rent: 0 – No 1 – Yes
Own_Res	Categorical	Housing as Own Residence: 0 – No 1 – Yes
Num_credits	Integer	Number of existing credits at this bank
Job	Categorical	0 – unemployed/unskilled – non-resident 1 – unskilled – resident 2 – skilled employee/official

		3 – management/self-employed/highly qualified employee/officer
Num_dependents	Integer	Number of people being liable to provide maintenance for
Telephone	Categorical	0 – No 1 – Yes
Foreign	Categorical	Foreign Worker 0 – No 1 – Yes
Response	Categorical	0 – Bad 1 – Good

Data Cleaning

Impute Missing Values

- There are missing values in the fields related to purpose of credit: NEW_CAR, USED_CAR, FURNITURE, RADIO.TV, EDUCATION, and RETRAINING. For these variables, the values were either '1' or 'NA' so, NA has been replaced with '0'.
- For the variable AGE, the missing values has been replaced with the average age.

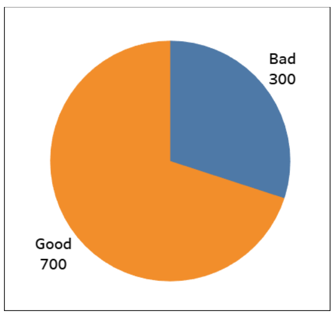
Drop the variables

- Observation# - It is not required to build the predictive model.

EXPLORATORY DATA ANALYSIS

Response variable

Let's start by looking at the response variables. As depicted below, there are 700 cases of good credit and 300 cases of bad credit.

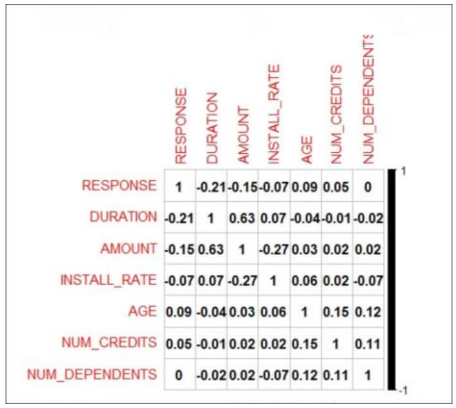


Continuous variables

- Descriptive Statistics

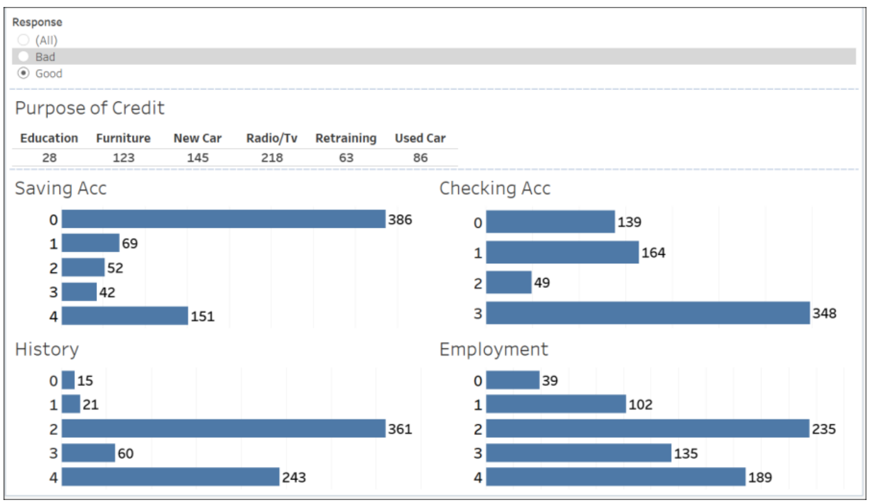
	Age	Amount	Duration	Install Rate
Average	35	3,271	21	3
Median	33	2,320	18	3
Standard Deviation	11	2,823	12	1
Minimum	19	250	4	1
Maximum	75	18,424	72	4
25th Percentile	27	1,366	12	2
75th Percentile	42	3,972	24	4

- Correlation matrix

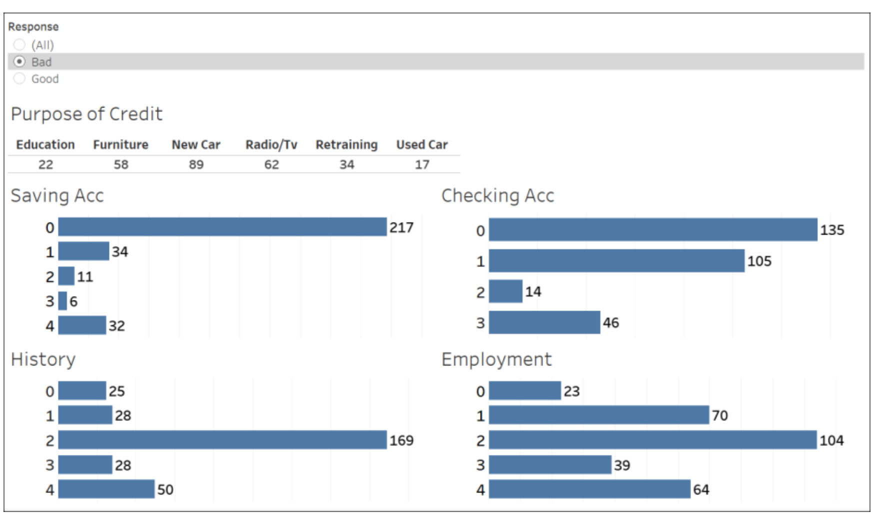


Categorical variables

- Applicants with credit risk as 1 (Good):



- Applicants with credit risk as 0 (Bad):



Interpretation:

- More than 50% of the applicants don't have checking count or have no money in their checking account
- Based on history, the maximum number of good credit applicants have existing credits and have been paying duly till now
- 78.3% of the applicants either have no savings account or have money less than 100DM
- 76.6% of the applicants have been employed for more than a year

INTERESTING VARIABLES and WHY?

- CHK_ACCT & SAV_ACCT- There are almost equal 'good' and 'bad' cases with the checking account of < 0 DM and SAV_ACCT < 100 DM. It'll be interesting to see what other factors help determine the credit risk.
- Out of the 'purpose of credit' variables, new_car, used_car and education seem interesting because others are relatively low priced.
- Employment and job: Both job and employment are interesting because the credit rating will depend on the number of years of employment and whether the individual is skilled professional or unskilled professional.

MODEL BUILDING

The dataset has been split into 70:30 ratio.

Decision Tree

Decision Tree is a tree-based algorithm used for both regression and classification. In this case, the classification tree has been used to classify the applicant as good or bad. The criteria used to split the tree are:

- **minsplit** = 20 : The minimum number of observations that must exist in a node in order for a split to be attempted. With the increase in minsplit value, the performance decreases.
- **minbucket** = 7 : The minimum number of observations in any terminal node. With the increase in minbucket value, the performance decreases. It is because, with higher number of values in the root, it's highly likely to get high proportion of defaulters as well.
- **cp** = 0.01 (default value) : Complexity parameter. Any split that does not decrease the overall lack of fit by a factor of cp is not attempted. In this case, the performance decreases with the increase in complexity parameter
- **Xval** = 10 : The number of cross validations.

```
Classification tree:
rpart(formula = RESPONSE ~ ., data = dataset_train, method = "class",
    control = rpart.control(minsplit = 20, minbucket = 7, cp = 0.01,
        xval = 10, parms = list(split = "infomation")))

Variables actually used in tree construction:
[1] AMOUNT          CHK_ACCT        DURATION        EMPLOYMENT      GUARANTOR       HISTORY         PRESENT_RESIDENT SAV_ACCT

Root node error: 240/800 = 0.3

n= 800

        CP nsplit rel error  xerror     xstd
1 0.044444      0   1.00000 1.00000 0.054006
2 0.036111      3   0.86667 0.95833 0.053339
3 0.029167      7   0.71667 0.92500 0.052770
4 0.019444      8   0.68750 0.91250 0.052548
5 0.018750     11   0.62917 0.91250 0.052548
6 0.016667     13   0.59167 0.91250 0.052548
7 0.010000     15   0.55833 0.84583 0.051284
```

Confusion Matrix and Statistics

The basic terms in the confusion matrix are:

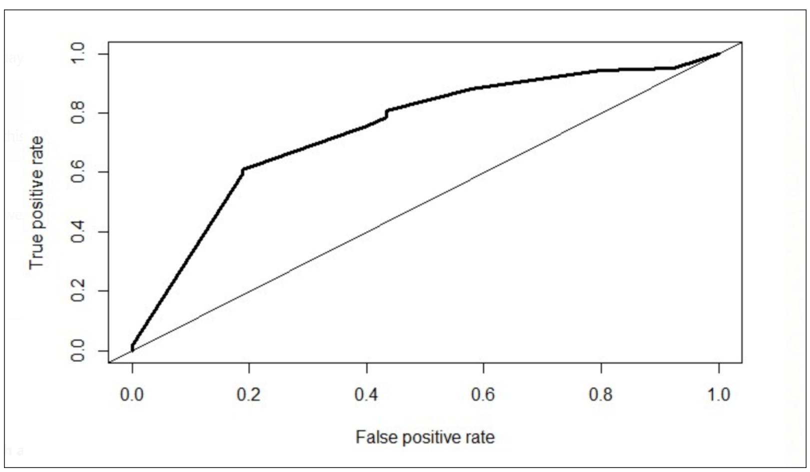
- True Positives (TP): Actual good applicant and predicted as good applicant
- True negatives (TN): Actual bad applicant and predicted as bad applicant
- False Positives (FP): Actual bad applicant and predicted as good applicant
- False Negatives (FN): Actual good applicant and predicted as bad applicant

All performance measure devised from confusion matrix can be found here.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  38  25
         1  52 185

               Accuracy : 0.7433
                 95% CI : (0.69, 0.7918)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : 0.056076

                  Kappa : 0.3316
 Mcnemar's Test P-Value : 0.003047

            Sensitivity : 0.4222
            Specificity : 0.8810
         Pos Pred Value : 0.6032
         Neg Pred Value : 0.7806
             Prevalence : 0.3000
         Detection Rate : 0.1267
   Detection Prevalence : 0.2100
      Balanced Accuracy : 0.6516

       'Positive' Class : 0
```

ROC Curve

ROC curve is the most commonly used way to visualize the performance of a binary classifier, and AUC is the best way to summarize its performance in a single number. [Source]



The area under curve (AUC) for this model is 0.744.

Decision Tree: Stable?

A stable learning algorithm is one for which the prediction does not change much when the training data is modified slightly. Decision trees are inherently unstable as with slight change in input, not only the output changes but it gives rise to a completely different tree structure. [Source]

AdaBoost

Boosting is an ensemble technique that attempts to create a strong classifier from several weak classifiers. AdaBoost is best used to boost the performance of decision trees on binary classification problems. [Source]

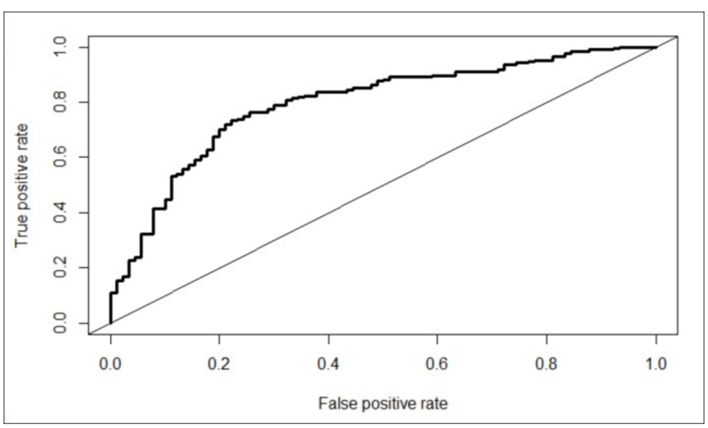
```
Call:
ada(RESPONSE ~ ., data = dataset_train)

Loss: exponential Method: discrete   Iteration: 50

Training Results

Accuracy: 0.911 Kappa: 0.777
```

- Loss = exponential (default) : Loss under exponential loss.
- Iter = 50: Number of boosting iterations to perform.

Confusion Matrix and Statistics

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  41  23
         1  49 187

               Accuracy : 0.76
                 95% CI : (0.7076, 0.8072)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : 0.012488

                  Kappa : 0.3772
 Mcnemar's Test P-Value : 0.003216

            Sensitivity : 0.4556
            Specificity : 0.8905
         Pos Pred Value : 0.6406
         Neg Pred Value : 0.7924
             Prevalence : 0.3000
         Detection Rate : 0.1367
   Detection Prevalence : 0.2133
      Balanced Accuracy : 0.6730

       'Positive' Class : 0
```

ROC Curve



The area under curve (AUC) for this model is 0.795.

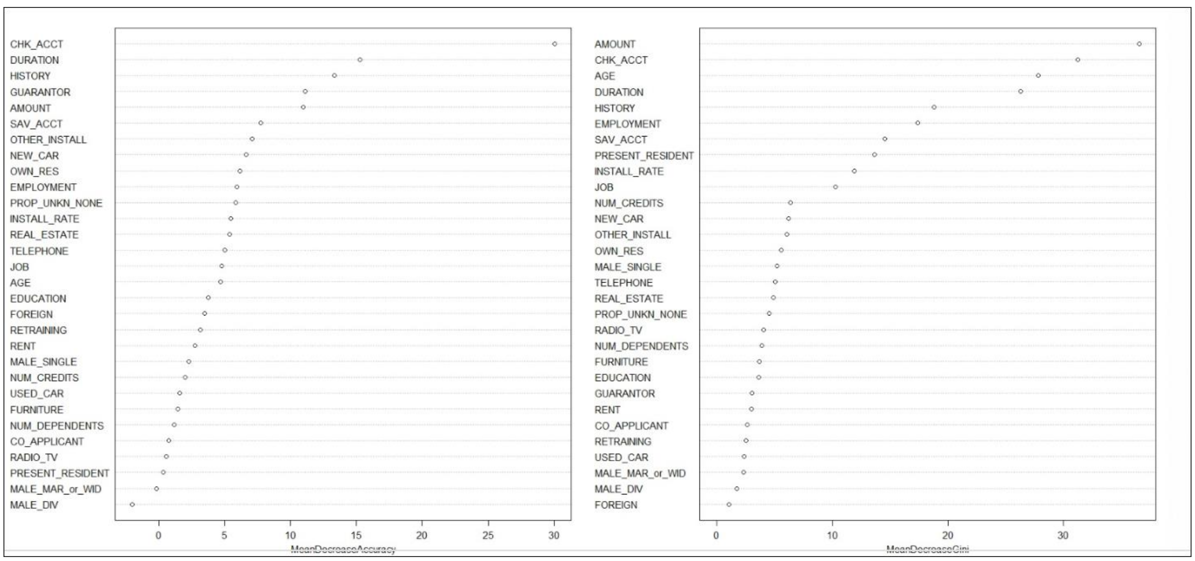
Random Forest

Random Forest is an ensemble of Decision Trees, most of the time trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. [Source]

```
Call:
 randomForest(formula = RESPONSE ~ ., data = dataset_train, ntree = 700,      importance = T)
               Type of random forest: classification
                     Number of trees: 700
No. of variables tried at each split: 5

        OOB estimate of  error rate: 23.71%
Confusion matrix:
   0   1 class.error
0 86 124  0.59047619
1 42 448  0.08571429
```

Variable Importance

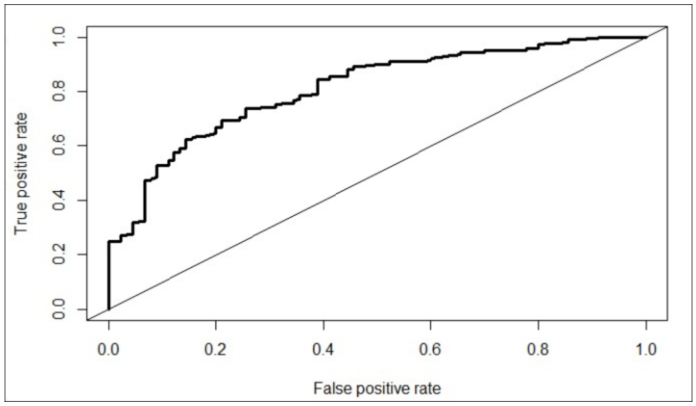
The excellent quality of the random forest algorithm is that it is very easy to measure the relative importance of each feature on the prediction, as depicted below:



Confusion Matrix and Statistics

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  32  14
         1  58 196

               Accuracy : 0.76
                 95% CI : (0.7076, 0.8072)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : 0.01249

                  Kappa : 0.3358
 Mcnemar's Test P-Value : 4.029e-07

            Sensitivity : 0.3556
            Specificity : 0.9333
         Pos Pred Value : 0.6957
         Neg Pred Value : 0.7717
             Prevalence : 0.3000
         Detection Rate : 0.1067
   Detection Prevalence : 0.1533
      Balanced Accuracy : 0.6444

       'Positive' Class : 0
```

ROC Curve



The area under curve (AUC) for this model is 0.809.

QUANTIFYING MISCLASSIFICATION RATE - CALCULATING COST

The cost has been calculated based on the following assumptions:

- Cost incurred by financial institution if the good applicant gets classified as bad applicant = 100 DM
- Cost incurred by financial institution if the bad applicant gets classified as good applicant = 300 DM

Decision Tree

	Reference	
Prediction	0	1
0	38	25
1	52	185

Loss incurred by the financial institution using this model = (-300*25) + (-100*52) = -12,700

AdaBoost

	Reference	
Prediction	0	1
0	41	23
1	49	187

Loss incurred by the financial institution using this model = (-300*23) + (-100*49) = -11,800

Random Forest

	Reference	
Prediction	0	1
0	32	14
1	58	196

Loss incurred by the financial institution using this model = (-300*14) + (-100*58) = -10,000

MODEL DEPLOYMENT USING RSHINY

Based on the above results – Accuracy, Area under ROC curve and misclassification cost of different models, it can be deduced that Random Forest performs the best.

The model has been used for deployed using RShiny and below screenshot is the UI of the application: