# Approach

The credit risk scoring is a very complicated process with a lot of due diligence on data, model reviews internal controls and sign offs. As a first step you could follow the steps outlined below with the accompanying code to create a straw man version of your approach.

The first step in building your prototype will be obtaining a sample dataset and performing high level analysis on it.

```
#setting up the data and performing high level analysis#
########################################################
#downloading the data
#https://raw.githubusercontent.com/obaidpervaizgill/CreditRiskModelling/master/credit.csv

#loading data
credit <- read.csv("credit.csv")

#identifying the structure of variables
str(credit)

#getting summary of the variables
summary(credit)

#getting the column names
colnames(credit)
#[1] "checking_balance"     "months_loan_duration" "credit_history"
"purpose"              "amount"               "savings_balance"
#[7] "employment_duration"  "percent_of_income"    "years_at_residence"   "age"
"other_credit"         "housing"
#[13] "existing_loans_count" "job"

#tabulating dependent variables
table(credit$default)

#No missing values in the data
#Note : I would have used "mice" package in R to impute missing values if there
were any

#Normalizing or standardizing data
#Note : I would have scaled the variables using standardization or minmax
normalization, but I havent done this here!

#Removing correlated features
#Note : I would have removed correlated feature based on an 80 percent
correlation rule in the correlation matrix

#spliting data into test and train
library(caTools)
split <- sample.split(credit$default, SplitRatio = 0.70)
train <- subset(cbind(credit,split), cbind(credit,split)$split == TRUE)
test <- subset(cbind(credit,split), cbind(credit,split)$split == FALSE)

#checking proportions across train and test
prop.table(table(train$default))
prop.table(table(test$default))
```

The second step in your prototype will be to train an explainable model, such as a logistic regression model so that you can identify and explain the driving variables.

```
#training a model using logistic regression#
############################################

#training a model
creditLogReg <- glm(train$default ~ ., data = train[,c(-17,-18)], family =
"binomial" ) #removing split feature and dependent variable
summary(creditLogReg) #summary of the model output
#Note: In theory I should rerun the model removing the non-significant features
but since I want to demonstrate multiple model usage I would let it slide

#predicing on test data
predCreditLogReg  0.5)
#Note: we want our model to be optimally sensitive hence we use 0.5 as the
threshold, redudcing the threshold will make the model more sensitive

#computing the accuracy of the model
accuracyCreditLogReg  0.5))[1,1]) + (as.matrix(table(test$default,
predCreditLogReg > 0.5))[2,2]))/nrow(test)

#computing the baseline model for comparison
baseLineAccuracy <- max(table(test$default))/nrow(test)

print(accuracyCreditLogReg)
print(baseLineAccuracy)
#Note : Our simple logistic regression model beats the baseline model

#assesing the robustness of model
library(ROCR)
rocrPredCreditLogReg <- prediction(predCreditLogReg,test$default)
areaUnderCurve <- as.numeric(performance(rocrPredCreditLogReg, "auc")@y.values)
#out of sample auc
print(areaUnderCurve)
#Note : Closer to 1 is better, 0.78 here is not bad for a first model
```

The third step in your prototype will be to train an more complicated model to assess if you can improve over your explainable model through additional tuning as well.

```
#training a model using decision trees#
#######################################
library("rpart")
library("rpart.plot")

#training a model
creditDecTree <- rpart(train$default ~ ., data = train[,c(-17,-18)], method =
"class", minbucket = 1) #min bucket is minimum number of observations in a
terminal nore
summary(creditDecTree) #summary of the model output

#plotting a decision tree to see splits
prp(creditDecTree)

#predicting on test data
predictCreditDecTree <- predict(creditDecTree, newdata = test[,c(-17,-18)], type
= "class") #getting classes rather than probability
```

```
#computing the accuracy of the model
table(test$default,predictCreditDecTree) #since we dont have a probability here
so we dont set a threshold

accuracyCreditDecTree <- ((as.matrix(table(test$default, predictCreditDecTree))
[1,1]) + (as.matrix(table(test$default, predictCreditDecTree))[2,2]))/nrow(test)

#computing the baseline model for comparison
baseLineAccuracy <- max(table(test$default))/nrow(test)

print(accuracyCreditDecTree)
print(baseLineAccuracy)
#Note: Our decision tree model beats the basline model in terms of accuracy

#assesing the robustness of model
library(ROCR)
rocrPredictCreditDecTree <- prediction((predict(creditDecTree, newdata =
test[,c(-17,-18)])[,2]), test$default) #getting probability and then picking
predicted class
areaUnderCurve <- as.numeric(performance(rocrPredictCreditDecTree,
"auc")@y.values) #out of sample auc
print(areaUnderCurve)

#tuning a model using decision trees#
#####################################
library(caret)

#tuning for complexity parameter, this penalizes model complexity and avoids
overfitting
tuneGridDecTree <- expand.grid(.cp=seq(0.01,0.5,0.01))

#creating a list of parameters to be passed onto the model
fitControlDecTree <- trainControl(method = "cv", number = 10)


tunedCreditDecTree <- train(train$default ~., data = train[,c(-17,-18)],
                            method = "rpart",
                            trControl = fitControlDecTree,
                            tuneGrid = tuneGridDecTree)

tunedPredictCreditDecTree <- predict(tunedCreditDecTree,
newdata=test[,c(-17,-18)], type="raw")

#copmuting the accuracy of the model
table(test$default,tunedPredictCreditDecTree) #since we dont have a probability
here so we dont set a threshold

accuracyTunedCreditDecTree <- ((as.matrix(table(test$default,
tunedPredictCreditDecTree))[1,1]) + (as.matrix(table(test$default,
tunedPredictCreditDecTree))[2,2]))/nrow(test)
```

The final step in your prototype will be to train using a highly robust and more black box model to assess if you can improve over your existing approaches, to see if it is worthwhile to pursue this path.

```
#training a model using random forest#
#######################################
library(randomForest)
```

```
#training a model
creditRandFor <- randomForest(as.factor(train$default) ~., data = train[,c(-17,-
18)],nodesize =25, ntree = 200)
summary(creditRandFor) #summary of the model output

#identifying the most important variables based on mean gini decrease
varImpPlot(creditRandFor)
#Note : Show how each split result in low impurities or increased homogeneity

#predicting on test data
predictCreditRandFor <- predict(creditRandFor, newdata = test[,c(-17,-18)])

#computing the accuracy of the model
table(test$default,predictCreditRandFor) #since we dont have a probability here
so we dont set a threshold

accuracyCreditRandFor <- ((as.matrix(table(test$default, predictCreditRandFor))
[1,1]) + (as.matrix(table(test$default, predictCreditRandFor))[2,2]))/nrow(test)

#computing the baseline model for comparison
baseLineAccuracy <- max(table(test$default))/nrow(test)

print(accuracyCreditRandFor)
print(baseLineAccuracy)
#Note: Our random forest model beats the basline model in terms of accuracy

#assesing the robustness of model
library(ROCR)
rocrPredictCreditRandFor <- prediction((predict(creditRandFor, newdata =
test[,c(-17,-18)], type = "prob")[,2]), test$default) #getting probability and
then picking predicted class
areaUnderCurve <- as.numeric(performance(rocrPredictCreditRandFor,
"auc")@y.values) #out of sample auc
print(areaUnderCurve)
#Note : Very high area under the curve but slighltly less than logistic
regression
#Note : Very high accuracy as good as logistic regression

#tuning a model using random forest#
#######################################
#Note : We can tune it using tuneRF package but repeated cross validation using
caret produces much better results
library(caret)

#tuning for mtry, this the number of variables randomly sampled for splits
tuneGridRandFor <- expand.grid(.mtry=c(1:sqrt(ncol(train[,c(-17,-18)]))))

#creating a list of parameters to be passed onto the model
fitControlRandFor <- trainControl(method = "repeatedcv",
                             number = 5, repeats = 3,
                             #fivefold cross validation repeated 10 times
                             classProbs = TRUE,
                             summaryFunction = twoClassSummary)

tunedCreditRandFor <- train(as.factor(train$default) ~., data = train[,c(-17,-
18)],
                            method = "rf",
```

```
                            trControl = fitControlRandFor,
                            verbose = TRUE,
                            metric = "ROC",
                            tuneGrid = data.frame(tuneGridRandFor),
                            importance = TRUE)

tunedPredictCreditRandFor <- predict(tunedCreditRandFor, newdata = test[,c(-17,-
18)])

#computing the accuracy of the model
table(test$default,tunedPredictCreditRandFor) #since we dont have a probability
here so we dont set a threshold

accuracyTunedCreditRandFor <- ((as.matrix(table(test$default,
tunedPredictCreditRandFor))[1,1]) + (as.matrix(table(test$default,
tunedPredictCreditRandFor))[2,2]))/nrow(test)
```

## Conclusion

Depending on the problem you are trying to solve, you could pick a model that serves your case, simplest is always the better unless the complicated one is significantly better. Also note that while there may be a temptation to jump into models, most improvement in model performance come from data wrangling and creating new features for your models.