402 Final

2024-12-02

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
df <- read.csv("~/Downloads/Loan_default.csv", stringsAsFactors = T)
df <- df[,-1] #Drop ID as it is not necessary
df$Education <- factor(df$Education,levels = c("High School","Bachelor's","Master's","PhD")) #Reorder L
df$MaritalStatus <- factor(df$MaritalStatus,levels = c("Single","Married","Divorced"))
```

```r
set.seed(123)
idx <- sample(1:225694,replace = F, size = 29653)
balance <- df[df$Default == 0,]
df <- rbind(df[df$Default == 1,],balance[idx,]) #balance the dataset for the outcome variable
```

```r
set.seed(123)
index <- createDataPartition(df$Default, p=.8, list=FALSE, times=1) #Make sure train and test are also
train <- df[index,]
test <- df[-index,]
```

#Random Forest

Random Forest is like asking a group of experts for their opinion and combining their answers to make a better decision. It's an ensemble learning method that creates multiple decision trees (like individual experts), and their predictions are combined to make the final decision. Each opinion is like a decision tree, they may not always be right. But when you combine their votes , you'll likely make a better prediction. Random Forest is a powerful ensemble learning method used primarily for classification and regression tasks. It creates multiple decision trees by: Bootstrapping the training data to form subsets. Randomly selecting features at each split in the trees. Aggregating the results from all trees (majority vote for classification or average for regression). This approach helps Random Forest reduce overfitting and improve model accuracy by introducing diversity into the individual decision trees.

What is Grid Search? Grid Search is a systematic method to tune the hyperparameters of a machine learning model. It helps find the best combination of parameters by exhaustively trying all possible values within a specified range.

Why Do We Use Grid Search? Hyperparameters, like the number of trees (ntree) or the number of features considered at each split (mtry) in Random Forest, can significantly impact a model's performance. Grid Search helps identify the combination of hyperparameters that yields the best results.

```r
library(caret)
library(randomForest)
```

```
## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
# Make sure the outcome is a factor in the training dataset
train$Default <- as.factor(train$Default)

# Define the grid of hyperparameters to tune
grid <- expand.grid(
  mtry = c(2, 3, 4, 5)        # Number of variables tried at each split
)

# Train the model using caret's train() function with Random Forest and grid search
rf_model <- train(
  Default ~ .,
  data = train,
  method = "rf",              # Random Forest
  tuneGrid = grid,            # Pass the hyperparameter grid
  trControl = trainControl(method = "cv", number = 5),  # 5-fold cross-validation
  importance = TRUE           # Ensure variable importance is calculated
)

# View the best combination of hyperparameters
print(rf_model$bestTune)
```

```
##   mtry
## 1    2
```

```
# Print the final model with the best hyperparameters
print(rf_model$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 31.89%
## Confusion matrix:
##       0     1 class.error
## 0 16285  7438   0.3135354
## 1  7692 16031   0.3242423
```

```
# Make predictions with the optimized model
predictions <- predict(rf_model, newdata = test)
```

Number of Trees (ntree):

A Random Forest consists of many decision trees, each contributing to the overall prediction. We used grid search to test multiple values for ntree (e.g., 100, 200, 500) and found that 500 trees provided the best balance of: Model performance: Adding more trees reduced the OOB (Out-of-Bag) error rate. Computational cost: 500 trees maintained a reasonable runtime. Why stop at 500? Beyond a certain point, adding more trees gives diminishing returns while significantly increasing computation time. Result: 500 trees helped achieve an OOB error rate of 31.89%, showing good generalization.

Number of Variables Tried at Each Split (mtry): When building each decision tree, a Random Forest algorithm randomly selects a subset of features to consider for splits at each node. Using grid search, we tested different values of mtry (e.g., 2, 3, 4) and found that 2 features at each split worked best. Why 2? It introduced diversity among the trees (as each tree used different features for splitting). It maintained good predictive power, avoiding underfitting or overfitting.

```
train$Default <-as.factor(train$Default)

rf_model =randomForest(Default~., data = train, ntree = 500, mtry = 2, importance = TRUE)

print(rf_model)
```

```
##
## Call:
##  randomForest(formula = Default ~ ., data = train, ntree = 500,      mtry = 2, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 31.95%
## Confusion matrix:
##       0     1 class.error
## 0 16228  7495   0.3159381
## 1  7664 16059   0.3230620
```

```r
# Check predictions
table(predictions)
```

```
## predictions
##    0    1
## 6015 5845
```

```r
# Check test$Default
table(test$Default)
```

```
##
##    0    1
## 5930 5930
```

```r
# Compare levels
levels(predictions)
```

```
## [1] "0" "1"
```

```r
levels(test$Default)
```

```
## NULL
```

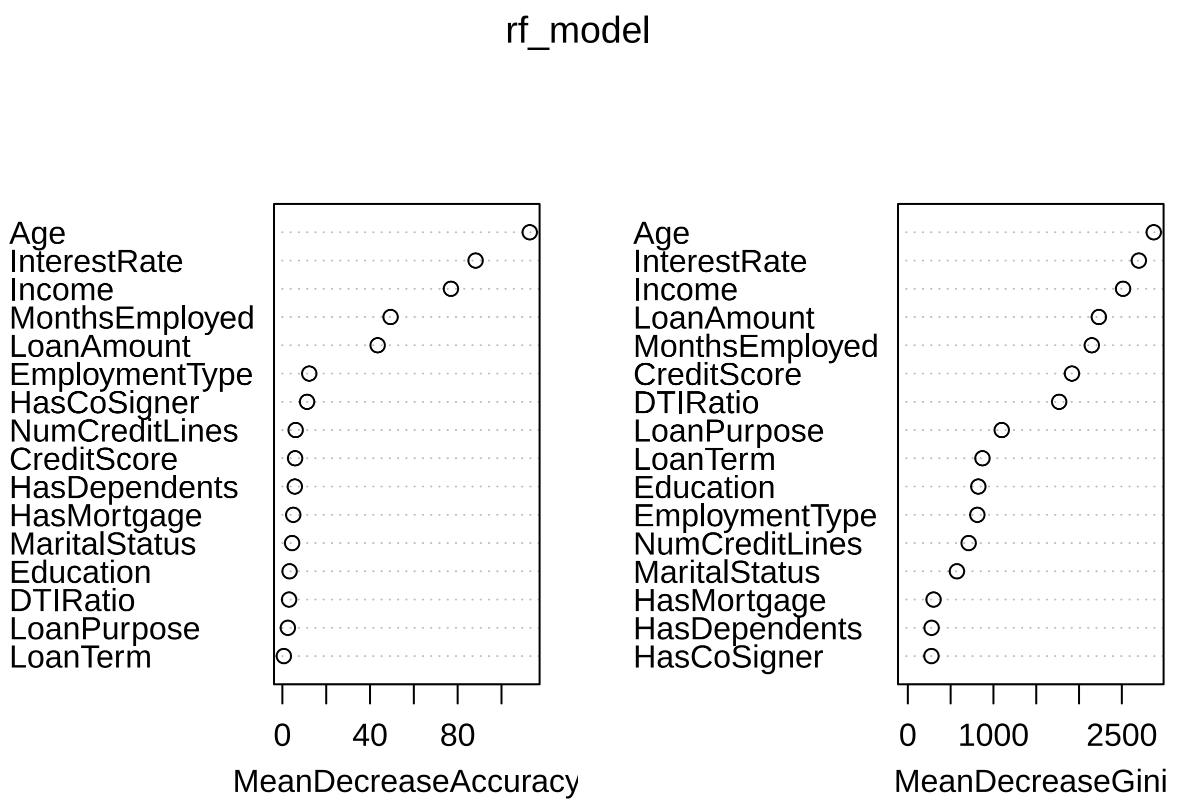
```r
df$Default <- factor(ifelse(df$Default == 1, "Default", "Non-Default"))
df$Default <- relevel(df$Default, ref = "Non-Default")

predictions <- predict(rf_model, newdata = test)

# Ensure predictions are factors
predictions <- factor(predictions)
test$Default <- factor(test$Default)

# Calculate the confusion matrix
conf_matrix <- confusionMatrix(predictions, test$Default)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 4021 1947
##          1 1909 3983
##
##                Accuracy : 0.6749
##                  95% CI : (0.6664, 0.6833)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.3497
##
```

```
##  Mcnemar's Test P-Value : 0.5513
##
##             Sensitivity : 0.6781
##             Specificity : 0.6717
##          Pos Pred Value : 0.6738
##          Neg Pred Value : 0.6760
##              Prevalence : 0.5000
##          Detection Rate : 0.3390
##    Detection Prevalence : 0.5032
##       Balanced Accuracy : 0.6749
##
##        'Positive' Class : 0
##
```

Important Predictors 1. Mean Decrease in Accuracy This metric evaluates how much model accuracy would decrease if a specific predictor were removed. Higher values indicate greater importance. Age: Most significant predictor, suggesting a strong relationship between a person's age and their likelihood of defaulting on a loan. Interest Rate: Reflects the impact of borrowing cost; higher interest rates likely correlate with increased risk of default. Income: Indicates that higher income reduces the probability of default. Months Employed: Job stability plays a role in financial security. Loan Amount: Larger loans are associated with higher default risk. 2. Mean Decrease in Gini This measures the contribution of each variable to reducing impurity (classification error) in the trees. Higher values indicate more influential variables. Age Interest Rate Income: Loan Amount Months Employed In conclusion, Age, Interest Rate, Income, Loan Amount, and Months Employed emerge as the most important predictors, these variables can guide loan underwriting processes and risk assessment strategies.

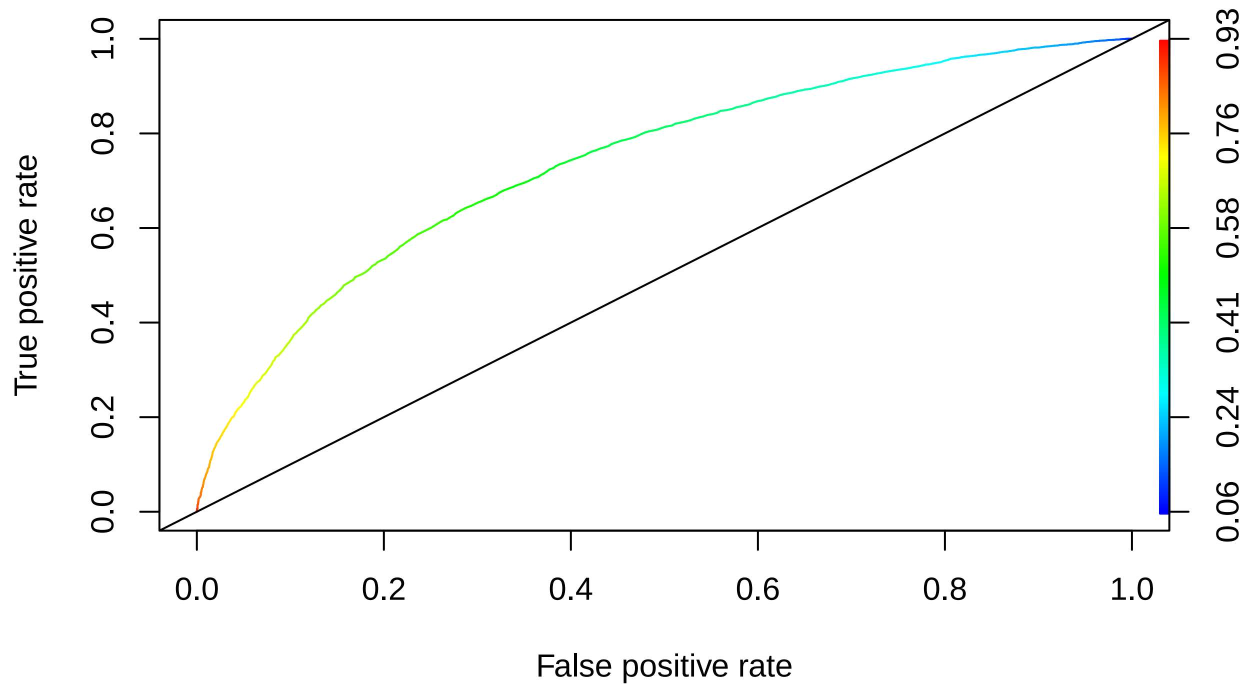
```r
# Variable importance plot
varImpPlot(rf_model)
```



```r
#Calculate the probability
probability <- predict(rf_model, test, "prob")

probability <- probability[, -which(colnames(probability) == "0")]

head(probability)
```

```
##    12    19    28    61   178   204
## 0.410 0.566 0.596 0.696 0.410 0.594
```

```r
#Draw the ROC Curve

library(ROCR)

pred_m2 <- prediction(probability, test$Default)
roc_curve <- performance(pred_m2, "tpr","fpr")
plot(roc_curve, colorize=T)
abline(0, 1)
```



```r
auc_ROCR <- performance(pred_m2, measure = "auc")
(auc_ROCR <- auc_ROCR@y.values[[1]])
```

```
## [1] 0.7385253
```