Chapter 19: Use Case

Ram Gopal, Dan Philps, and Tillman Weyde

2022

Contents

USe Case: Credit Risk - Identifying Bad Credits

The dataset used for the use case is from https://datahub.io/machine-learning/credit-g and is based on Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Loading libraries

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(rattle)
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
if(!require("pacman")){
  install.packages("pacman")
}
```

```
## Loading required package: pacman
```

```
pacman::p_load("remotes", "RnavGraphImageData", "magrittr", "nnet", "data.table",
               "caret", "rpart", "tictoc", "naivebayes", "RSNNS", "mvtnorm", "patchwork")
```

Read the data file

- Useful to note that reading categorical data as factors makes data wrangling much simpler in R. While you may want to illustrate one-hot encoding for factor variables, it is not necessary to do so for the models in this use case.

```
df <- read.csv("credit-g.csv", stringsAsFactors=TRUE)
```

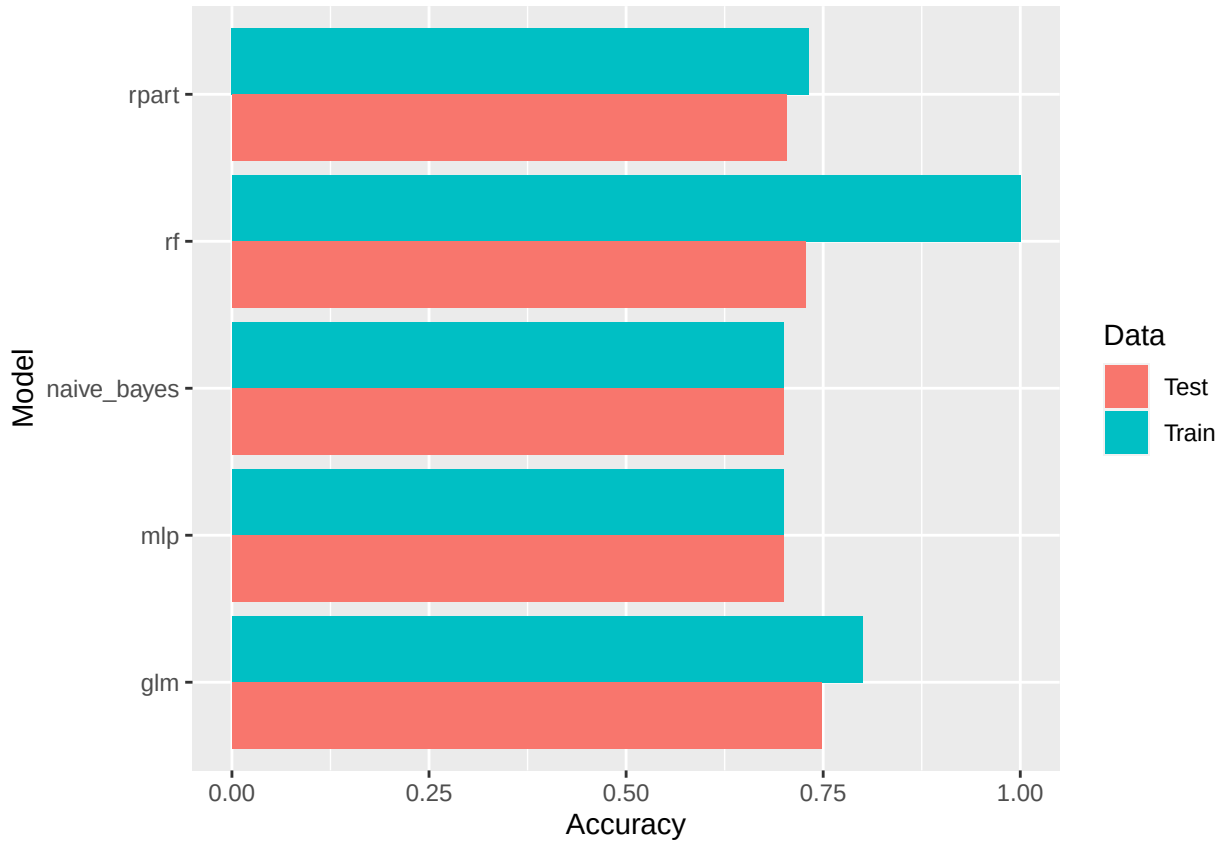
Build models

- Looping through 5 different prediction algorithms

```
index <- caret::createDataPartition(df$class,p=0.5,list=FALSE)
train_df <- df[index,]
test_df <- df[-index,]
trControl <- trainControl(method  = "cv", number  = 2)
results_df = data.frame()

for (mdl in c("rpart","naive_bayes","glm","rf","mlp"))
{
     mdl_model <- caret::train(class ~ .,
             method=mdl,
             trControl  = trControl,
             data = train_df,
             metric     = "Accuracy")
     mdl_pred_test= predict(mdl_model,test_df)
     mdl_pred_train= predict(mdl_model,train_df)
    train_accuracy =
      caret::confusionMatrix(mdl_pred_train,train_df$class)$overall[1]
    test_accuracy =
      caret::confusionMatrix(mdl_pred_test,test_df$class)$overall[1]
    print(paste("Model ", mdl, " Accuracy: ",
                "Training = ", train_accuracy,
                " Test = ",test_accuracy))
    results_df = rbind(results_df,c(mdl,"Train",train_accuracy))
    results_df = rbind(results_df,c(mdl,"Test",test_accuracy))

}
```

```
## [1] "Model  rpart  Accuracy:  Training =  0.732  Test =  0.704"
## [1] "Model  naive_bayes  Accuracy:  Training =  0.7  Test =  0.7"
## [1] "Model  glm  Accuracy:  Training =  0.8  Test =  0.748"
## [1] "Model  rf  Accuracy:  Training =  1  Test =  0.728"
## [1] "Model  mlp  Accuracy:  Training =  0.7  Test =  0.7"
```

```
colnames(results_df) = c("Model","Data","Accuracy")
results_df$Accuracy = as.numeric(results_df$Accuracy)

ggplot(results_df,aes(Model,Accuracy,fill=Data)) +
  geom_col(position = "dodge")+ coord_flip()
```



- Useful to point out the functions to compute sensitivity, specificity,and F1 scores.

```
    mdl = c("glm")
   mdl_model <- caret::train(class ~ .,
           method=mdl,
           trControl  = trControl,
           data = train_df,
           metric     = "Accuracy")
   mdl_pred_test= predict(mdl_model,test_df)
   mdl_pred_train= predict(mdl_model,train_df)
  train_accuracy =
    caret::confusionMatrix(mdl_pred_train,train_df$class)$overall[1]
  test_accuracy =
```

```
      caret::confusionMatrix(mdl_pred_test,test_df$class)$overall[1]
print("Train")
```

```
## [1] "Train"
```

```
paste("Accuracy = ",train_accuracy)
```

```
## [1] "Accuracy =  0.8"
```

```
paste("Sensitivity = ",sensitivity(mdl_pred_train,train_df$class))
```

```
## [1] "Sensitivity =  0.553333333333333"
```

```
paste("Specificity = ",specificity(mdl_pred_train,train_df$class))
```

```
## [1] "Specificity =  0.905714285714286"
```

```
paste("F1 Score = ",F_meas(mdl_pred_train,train_df$class))
```

```
## [1] "F1 Score =  0.62406015037594"
```

```
caret::confusionMatrix(mdl_pred_train,train_df$class)$table
```

```
##           Reference
## Prediction bad good
##       bad   83   33
##       good  67  317
```

```
print("Test")
```

```
## [1] "Test"
```

```
paste("Accuracy = ",test_accuracy)
```

```
## [1] "Accuracy =  0.748"
```

```
paste("Sensitivity = ",sensitivity(mdl_pred_test,test_df$class))
```

```
## [1] "Sensitivity =  0.466666666666667"
```

```
paste("Specificity = ",specificity(mdl_pred_test,test_df$class))
```

```
## [1] "Specificity =  0.868571428571429"
```

```
paste("F1 Score = ",F_meas(mdl_pred_test,test_df$class))
```

```
## [1] "F1 Score =  0.526315789473684"
```

```
caret::confusionMatrix(mdl_pred_test,test_df$class)$table
```

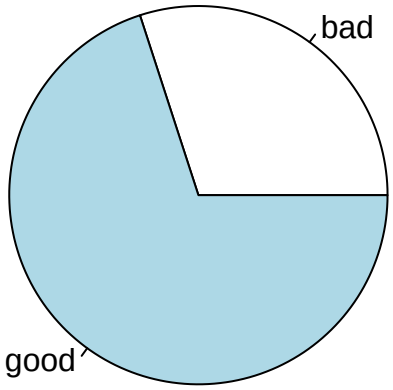
```
##           Reference
## Prediction bad good
##       bad   70   46
##       good  80  304
```

Data Imbalance

```
table(train_df$class)
```

```
##
##  bad good
##  150  350
```

```
pie(table(train_df$class))
```



Upsample

- You may to discuss the logic that determines the size of the up-sampled training data. It should be emphasized that only the training data is altered, but not the test data.

```
prop.table(table(train_df$class))
```

```
##
##  bad good
##  0.3  0.7
```

```
trainup<-upSample(x=train_df[,-ncol(train_df)],
                  y=train_df$class)
colnames(trainup) = c(colnames(trainup[-21]),"class")
prop.table(table(trainup$class))
```

```
##
##  bad good
##  0.5  0.5
```

```
nrow(train_df)
```

```
## [1] 500
```

```
nrow(trainup)
```

```
## [1] 700
```

- Logistic regression - useful to compare the results with the original results.

```
  mdl = c("glm")
     mdl_model <- caret::train(class ~ .,
              method=mdl,
              trControl  = trControl,
              data = trainup,
              metric     = "Accuracy")
     mdl_pred_test= predict(mdl_model,test_df)
     mdl_pred_train= predict(mdl_model,trainup)
    train_accuracy =
      caret::confusionMatrix(mdl_pred_train,trainup$class)$overall[1]
    test_accuracy =
      caret::confusionMatrix(mdl_pred_test,test_df$class)$overall[1]
print("Train")
```

```
## [1] "Train"
```

```
paste("Accuracy = ",train_accuracy)
```

```
## [1] "Accuracy =  0.755714285714286"
```

```
paste("Sensitivity = ",sensitivity(mdl_pred_train,trainup$class))
```

```
## [1] "Sensitivity =  0.762857142857143"
```

```
paste("Specificity = ",specificity(mdl_pred_train,trainup$class))
```

```
## [1] "Specificity =  0.748571428571429"
```

```
paste("F1 Score = ",F_meas(mdl_pred_train,trainup$class))
```

```
## [1] "F1 Score =  0.757446808510638"
```

```
caret::confusionMatrix(mdl_pred_train,trainup$class)$table
```

```
##           Reference
## Prediction bad good
##       bad  267   88
##       good  83  262
```

```
print("Test")
```

```
## [1] "Test"
```

```
paste("Accuracy = ",test_accuracy)
```

```
## [1] "Accuracy =  0.698"
```

```
paste("Sensitivity = ",sensitivity(mdl_pred_test,test_df$class))
```

```
## [1] "Sensitivity =  0.666666666666667"
```

```
paste("Specificity = ",specificity(mdl_pred_test,test_df$class))
```

```
## [1] "Specificity =  0.711428571428571"
```

```
paste("F1 Score = ",F_meas(mdl_pred_test,test_df$class))
```

```
## [1] "F1 Score =  0.56980056980057"
```

```
caret::confusionMatrix(mdl_pred_test,test_df$class)$table
```

```
##           Reference
## Prediction bad good
##       bad  100  101
##       good  50  249
```

SMOTE

- It may be useful to discuss the key difference with up-sampling and down-sampling. SMOTE employs KNN and interpolation to create synthetic data. Other, more advanced approaches have been developed. Time permitting, some of these can be discussed.

```
library(UBL)
```

```
## Loading required package: MBA

## Loading required package: gstat

## Loading required package: automap

## Loading required package: sp

## Loading required package: randomForest

## randomForest 4.7-1.1

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:rattle':
##
##     importance

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

```
set.seed(111)
trainsmote = UBL::SmoteClassif(class~.,train_df, dist="HEOM")
table(trainsmote$class)
```

```
##
##  bad good
##  249  250
```

```
 mdl = c("glm")
     mdl_model <- caret::train(class ~ .,
             method=mdl,
             trControl  = trControl,
             data = trainsmote,
             metric     = "Accuracy")
     mdl_pred_test= predict(mdl_model,test_df)
     mdl_pred_train= predict(mdl_model,trainsmote)
    train_accuracy =
      caret::confusionMatrix(mdl_pred_train,trainsmote$class)$overall[1]
    test_accuracy =
      caret::confusionMatrix(mdl_pred_test,test_df$class)$overall[1]
print("Train")
```

```
## [1] "Train"
```

```
paste("Accuracy = ",train_accuracy)
```

```
## [1] "Accuracy =  0.767535070140281"
```

```
paste("Sensitivity = ",sensitivity(mdl_pred_train,trainsmote$class))
```

```
## [1] "Sensitivity =  0.783132530120482"
```

```
paste("Specificity = ",specificity(mdl_pred_train,trainsmote$class))
```

```
## [1] "Specificity =  0.752"
```

```
paste("F1 Score = ",F_meas(mdl_pred_train,trainsmote$class))
```

```
## [1] "F1 Score =  0.770750988142292"
```

```
caret::confusionMatrix(mdl_pred_train,trainsmote$class)$table
```

```
##            Reference
## Prediction bad good
##       bad  195   62
##       good  54  188
```

```
print("Test")
```

```
## [1] "Test"
```

```
paste("Accuracy = ",test_accuracy)
```

```
## [1] "Accuracy =  0.704"
```

```
paste("Sensitivity = ",sensitivity(mdl_pred_test,test_df$class))
```

```
## [1] "Sensitivity =  0.646666666666667"
```

```
paste("Specificity = ",specificity(mdl_pred_test,test_df$class))
```

```
## [1] "Specificity =  0.728571428571429"
```

```
paste("F1 Score = ",F_meas(mdl_pred_test,test_df$class))
```

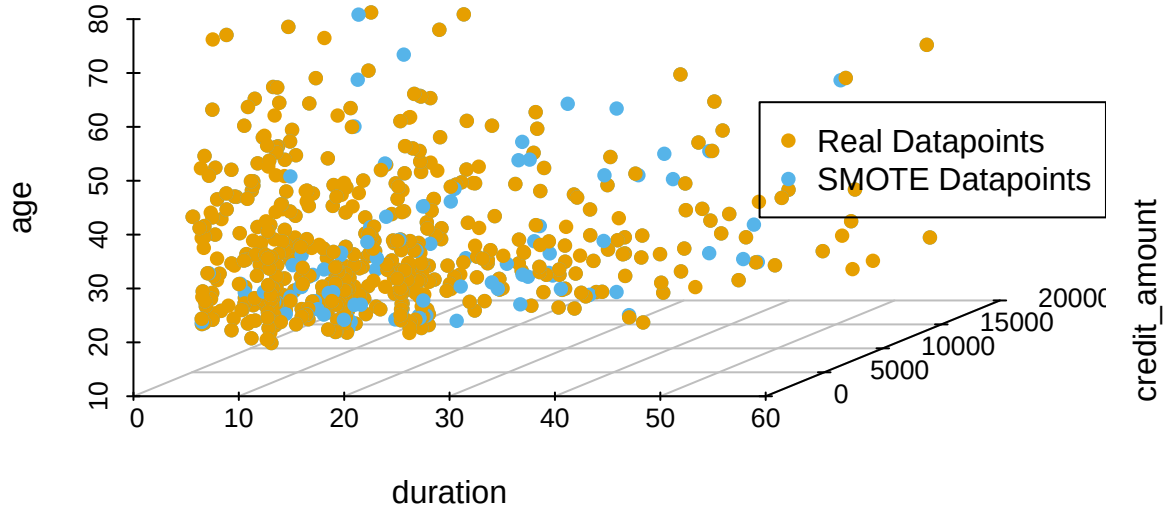
```
## [1] "F1 Score =  0.567251461988304"
```

```
caret::confusionMatrix(mdl_pred_test,test_df$class)$table
```

```
##            Reference
## Prediction bad good
##       bad   97   95
##       good  53  255
```

- Useful to compare and contrast the results with the previous approaches.
- The following provides a good graphical illustration of the original and SMOTE data points for numeric data. You can ask students to think about how to compare factor/categorical data similarities.

```
library(scatterplot3d)
x = train_df[,c(2,5,13)]
x$type = "Real Datapoints"
x1 = trainsmote[,c(2,5,13)]
x1$type = "SMOTE Datapoints"
y = rbind(x,x1)
y$type = factor(y$type)

attach(y)
colors <- c("#E69F00", "#56B4E9")
colors <- colors[as.numeric(type)]
scatterplot3d(x = duration, y = credit_amount, z = age, color = colors,pch=16,
              grid=TRUE, box=FALSE)
legend("right", legend = levels(type),
      col =  c("#E69F00", "#56B4E9"), pch = 16)
```




```
detach(y)
```

Down-sampling

- Results in smaller training data.

```
prop.table(table(train_df$class))
```

```
##
##  bad good
##  0.3  0.7
```

```
traindown<-downSample(x=train_df[,-ncol(train_df)],
                      y=train_df$class)
colnames(traindown) = c(colnames(traindown[-21]),"class")
prop.table(table(traindown$class))
```

```
##
##  bad good
##  0.5  0.5
```

```
  mdl = c("glm")
     mdl_model <- caret::train(class ~ .,
             method=mdl,
             trControl  = trControl,
             data = traindown,
             metric     = "Accuracy")
     mdl_pred_test= predict(mdl_model,test_df)
     mdl_pred_train= predict(mdl_model,traindown)
    train_accuracy =
      caret::confusionMatrix(mdl_pred_train,traindown$class)$overall[1]
    test_accuracy =
      caret::confusionMatrix(mdl_pred_test,test_df$class)$overall[1]
print("Train")
```

```
## [1] "Train"
```

```
paste("Accuracy = ",train_accuracy)
```

```
## [1] "Accuracy =  0.826666666666667"
```

```
paste("Sensitivity = ",sensitivity(mdl_pred_train,traindown$class))
```

```
## [1] "Sensitivity =  0.826666666666667"
```

```
paste("Specificity = ",specificity(mdl_pred_train,traindown$class))
```

```
## [1] "Specificity =  0.826666666666667"
```

```
paste("F1 Score = ",F_meas(mdl_pred_train,traindown$class))
```

```
## [1] "F1 Score =  0.826666666666667"
```

```
caret::confusionMatrix(mdl_pred_train,traindown$class)$table
```

```
##           Reference
## Prediction bad good
##       bad  124   26
##       good  26  124
```

```
print("Test")
```

```
## [1] "Test"
```

```
paste("Accuracy = ",test_accuracy)
```

```
## [1] "Accuracy =  0.69"
```

```
paste("Sensitivity = ",sensitivity(mdl_pred_test,test_df$class))
```

```
## [1] "Sensitivity =  0.72"
```

```
paste("Specificity = ",specificity(mdl_pred_test,test_df$class))
```

```
## [1] "Specificity =  0.677142857142857"
```

```
paste("F1 Score = ",F_meas(mdl_pred_test,test_df$class))
```

```
## [1] "F1 Score =  0.582210242587601"
```

```
caret::confusionMatrix(mdl_pred_test,test_df$class)$table
```

```
##           Reference
## Prediction bad good
##       bad  108  113
##       good  42  237
```

Final Model with SMOTE

- Useful to compare the SMOTE results with the original, unbalanced training data.

```
train_df = trainsmote
trControl <- trainControl(method  = "cv", number  = 2)
results_df = data.frame()

for (mdl in c("rpart","naive_bayes","glm","rf","mlp"))
{
     mdl_model <- caret::train(class ~ .,
             method=mdl,
             trControl  = trControl,
             data = train_df,
             metric     = "Accuracy")
     mdl_pred_test= predict(mdl_model,test_df)
     mdl_pred_train= predict(mdl_model,train_df)
    train_accuracy =
      caret::confusionMatrix(mdl_pred_train,train_df$class)$overall[1]
    test_accuracy =
      caret::confusionMatrix(mdl_pred_test,test_df$class)$overall[1]
    print(paste("Model ", mdl, " Accuracy: ",
                "Training = ", train_accuracy,
                " Test = ",test_accuracy))
    results_df = rbind(results_df,c(mdl,"Train",train_accuracy))
    results_df = rbind(results_df,c(mdl,"Test",test_accuracy))

}
```

```
## [1] "Model  rpart  Accuracy:  Training =  0.653306613226453  Test =  0.618"
## [1] "Model  naive_bayes  Accuracy:  Training =  0.74749498997996  Test =  0.704"
## [1] "Model  glm  Accuracy:  Training =  0.767535070140281  Test =  0.704"
## [1] "Model  rf  Accuracy:  Training =  1  Test =  0.738"
## [1] "Model  mlp  Accuracy:  Training =  0.501002004008016  Test =  0.7"
```

```
colnames(results_df) = c("Model","Data","Accuracy")
results_df$Accuracy = as.numeric(results_df$Accuracy)

ggplot(results_df,aes(Model,Accuracy,fill=Data)) +
  geom_col(position = "dodge")+ coord_flip()
```