Predicting Sales Win or Lose

Herman Toeante

February 12, 2019

Machine Learning for predicting sales win or loss

The goal of this project is to use machine learning to predict whether or not a sales lead is either win or loss and the probability of each class. This project is a supervised classification problem.

Two models will be used to test the data; Logistic regression and Random Forest. These two models are part of the caret package or R. Before preparing the model, the first step is to choose the independent variable that will be used in the model.

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.5.2
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.2
```

```
sales_win_loss <- read_csv("DATA/WA_Fn-UseC_-Sales-Win-Loss.csv")

colnames(sales_win_loss) <- c("ID","SuppliesSubgroup","SuppliesGroup","Region", "Route",
                              "ElapsedDays", "Result","SalesStageCount",
                              "TotalDaysClosing","TotalDaysQualified",
                              "Opportunity","ClientSizeRev","ClientSizeCount",
                              "Revenue","Competitor","RDaysIdentified",
                              "RDaysValidated","RDaysQualified",
                              "DealSize")


ModelData <- sales_win_loss %>% select(SuppliesSubgroup, Region, Route, TotalDaysClosing,
                                       TotalDaysQualified, Opportunity, ClientSizeRev,
                                       ClientSizeCount, Competitor, DealSize, Result)
```

The next step is partitioning the data set into training, validation, and testing dataset. We start with the seed for reproducibility followed by using createDataPartition function from caret.

```
set.seed(3456)
TrainingValidationIndex <- caret::createDataPartition(ModelData$Result, p = .80,
                                                      list = FALSE,
                                                      times = 1)
str(TrainingValidationIndex)
```

```
##  int [1:62421, 1] 3 6 8 9 10 11 13 14 15 16 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr "Resample1"
```

```r
TrainingValidation <- ModelData[ TrainingValidationIndex,]
TrainingIndex <- caret::createDataPartition(TrainingValidation$Result, p = .75,
                                            list = FALSE,
                                            times = 1)

Training <- TrainingValidation[TrainingIndex,]
Validation <- TrainingValidation[-TrainingIndex,]
Testing  <- ModelData[-TrainingValidationIndex,]
```

Using glimpse function, we can see the selected variables for the model.

```r
glimpse(Training)
```

```
## Observations: 46,817
## Variables: 11
## $ SuppliesSubgroup   <chr> "Shelters & RV", "Batteries & Accessories",...
## $ Region             <chr> "Pacific", "Northwest", "Pacific", "Northwe...
## $ Route              <chr> "Reseller", "Fields Sales", "Reseller", "Fi...
## $ TotalDaysClosing   <int> 114, 156, 50, 165, 31, 208, 138, 32, 130, 1...
## $ TotalDaysQualified <int> 0, 156, 50, 165, 31, 208, 138, 32, 130, 125...
## $ Opportunity        <int> 232522, 250000, 55003, 0, 10000, 232522, 20...
## $ ClientSizeRev      <int> 5, 1, 1, 1, 2, 1, 4, 5, 4, 1, 3, 1, 1, 5, 1...
## $ ClientSizeCount    <int> 1, 5, 1, 2, 1, 1, 5, 1, 3, 5, 5, 1, 4, 3, 5...
## $ Competitor         <chr> "Unknown", "None", "Unknown", "Unknown", "U...
## $ DealSize           <int> 5, 6, 4, 1, 2, 5, 5, 1, 4, 5, 4, 3, 4, 4, 7...
## $ Result             <chr> "Loss", "Loss", "Loss", "Loss", "Loss", "Lo...
```

Next, we setup the model parameter

```r
control <- caret::trainControl(method = "cv", number = 2, classProbs = TRUE)
seed <- 7
metric <- "Accuracy"
set.seed(seed)
```

Training Logistic with training dataset

```r
GLMModel <- caret::train(
  Result ~ SuppliesSubgroup + Region + Route + TotalDaysClosing + TotalDaysQualified +
    Opportunity + ClientSizeRev + ClientSizeCount + Competitor + DealSize,
  data = Training,
  method = "glm",
  trControl = control
)
```

Training Random Forest model with training dataset

```r
RFModel <- caret::train(
  Result ~ SuppliesSubgroup + Region + Route + TotalDaysClosing + TotalDaysQualified +
    Opportunity + ClientSizeRev + ClientSizeCount + Competitor + DealSize,
  data = Training,
  method = "rf",
  trControl = control
)
```

Using predict function from caret package, I will use the result to find the best model. Caret package also have the confusion matrix function to calculate sensitivity, specificity, negative predictied value, positive predicted values, and F1 score. The F1 score is a harmonic average of precision and recall to select the best model.

```
PredGLM <- predict(GLMModel, Validation)

ConfMatGLM <- caret::confusionMatrix(
  PredGLM, factor(Validation$Result), positive = "Won",
  mode = "everything")

ConfMatGLM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  Loss   Won
##       Loss 11604  2703
##       Won    475   822
##
##                Accuracy : 0.7963
##                  95% CI : (0.7899, 0.8026)
##     No Information Rate : 0.7741
##     P-Value [Acc > NIR] : 9.572e-12
##
##                   Kappa : 0.2498
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.23319
##             Specificity : 0.96068
##          Pos Pred Value : 0.63377
##          Neg Pred Value : 0.81107
##               Precision : 0.63377
##                  Recall : 0.23319
##                      F1 : 0.34094
##              Prevalence : 0.22590
##          Detection Rate : 0.05268
##    Detection Prevalence : 0.08312
##       Balanced Accuracy : 0.59693
##
##        'Positive' Class : Won
##
```

The F1 score for the Logistic model is 0.3409374 Next, we did the same thing using Random Forest model with validation data set

```
PredRF <- predict(RFModel, Validation)

ConfMatRF <- caret:: confusionMatrix(PredRF, factor(Validation$Result), positive = "Won",
                                     mode = "everything")

ConfMatRF
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction  Loss   Won
##       Loss 11233  1893
##       Won    846  1632
##
##                Accuracy : 0.8245
##                  95% CI : (0.8184, 0.8304)
##     No Information Rate : 0.7741
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4391
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4630
##             Specificity : 0.9300
##          Pos Pred Value : 0.6586
##          Neg Pred Value : 0.8558
##               Precision : 0.6586
##                  Recall : 0.4630
##                      F1 : 0.5437
##              Prevalence : 0.2259
##          Detection Rate : 0.1046
##    Detection Prevalence : 0.1588
##       Balanced Accuracy : 0.6965
##
##        'Positive' Class : Won
##
```

The F1 score for the Random Forest model is 0.5437281

Because the F1 score for Random Forest model is higher than the logistic model. The randome forest model is selected as the better model. The next step is to evaluate the Random Forest model using testing dataset.

```
FinalPredRF <- predict(RFModel, Testing)

ConfMatFinalRF <- caret::confusionMatrix(FinalPredRF, factor(Testing$Result), positive = "Won", mode =

ConfMatFinalRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  Loss   Won
##       Loss 11256  1855
##       Won    823  1670
##
##                Accuracy : 0.8284
##                  95% CI : (0.8224, 0.8343)
##     No Information Rate : 0.7741
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4525
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
##             Sensitivity : 0.4738
##             Specificity : 0.9319
##          Pos Pred Value : 0.6699
##          Neg Pred Value : 0.8585
##               Precision : 0.6699
##                  Recall : 0.4738
##                      F1 : 0.5550
##              Prevalence : 0.2259
##          Detection Rate : 0.1070
##    Detection Prevalence : 0.1598
##       Balanced Accuracy : 0.7028
##
##        'Positive' Class : Won
##
```

The final prediction using testing data set (0.5550017) appears to have about the same F1 score as using the validation data set (0.5437281)