Classification Case-Study

Martin Hanewald

1 August 2018

1 Abstract

This analysis is about a classification problem in marketing: how can we find the customer group with the highest potential value, on which to focus ones marketing activities.

This question is answered in two steps:

1. A cluster analysis is conducted on a dataset of historical transactions of the existing customer portfolio in order two separate two groups: **high** and **low** potential customers.
2. A second dataset with demographic information about the existing customers is used to train a classification algorithm to predict the group membership.

2 Configuration and libraries

```
library(tidyverse)
library(caret)
library(lubridate)
library(GGally)
library(rpart.plot)
```

3 Clustering

3.1 Loading data

```
orders <- read_csv2('data/orders_sim.csv')
orders
```

TransactionID <int>	ProductID <int>	CustomerID <int>	Date <date>	OrderTotal <int>
10195	9256	2088	2018-08-17	86
10756	9257	2088	2018-07-25	63
10309	9258	2088	2018-07-18	119
10430	9257	2088	2018-06-13	181
10645	9252	2088	2017-11-06	101
10255	9258	2088	2018-06-08	119
10330	9258	2088	2018-03-30	122
10527	9259	2088	2018-06-01	182
10848	9256	2088	2018-03-12	90
10469	9254	1654	2017-12-06	77

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 ... 1000 Next

3.2 Preprocessing

3.2.1 Feature engineering

We create three new features from the transactional data via summarising on the CustomerID's.

1. Recency: Days since last transaction
2. Frequency: Total transaction count
3. Market Value: Total revenue

These are the so called RFM metrics.

```
RFM <- orders %>%
    mutate_at(vars(CustomerID), as.factor) %>%
    group_by(CustomerID) %>%
    summarise(R = as.integer(Sys.Date() - max(Date)),
              F = n(),
              M = sum(OrderTotal))
RFM
```

CustomerID <fctr>	**R** <int>	**F** <int>	**M** <int>
1002	66	9	1165
1003	96	11	1313
1004	110	9	1174
1005	82	9	1184
1006	97	10	1067
1007	60	17	1710
1008	73	15	1616
1009	104	16	2079
1010	77	10	1250
1011	91	9	1127

1-10 of 1,089 rows Previous **1** 2 3 4 5 6 ... 109 Next

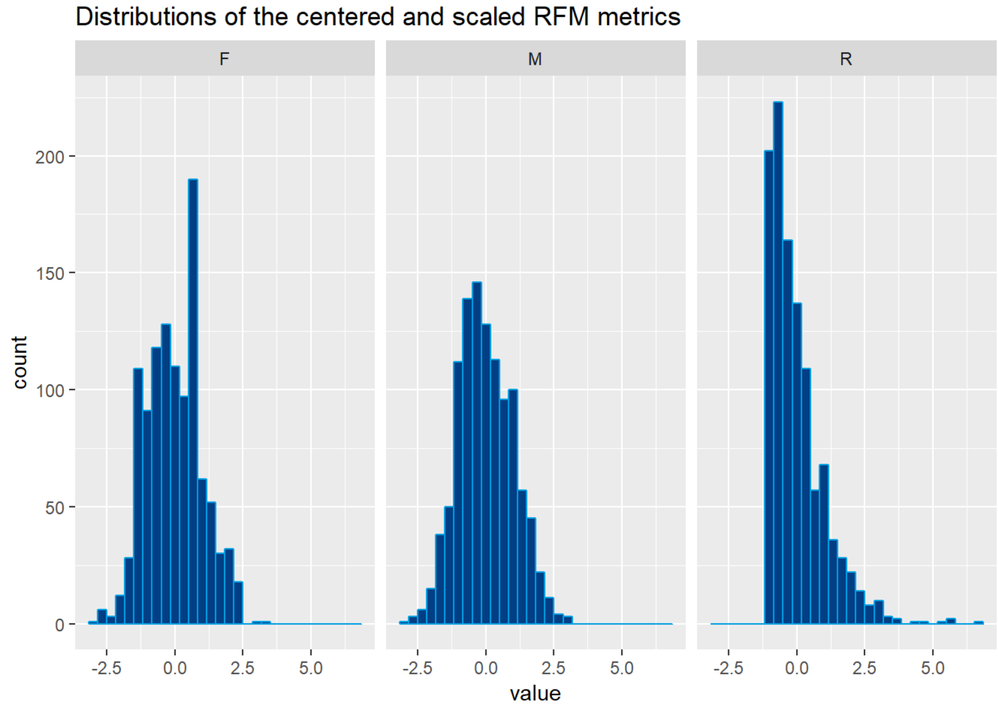
3.2.2 Centering and scaling

For more meaningful exploratory plots we center and scale the RFM matrix.

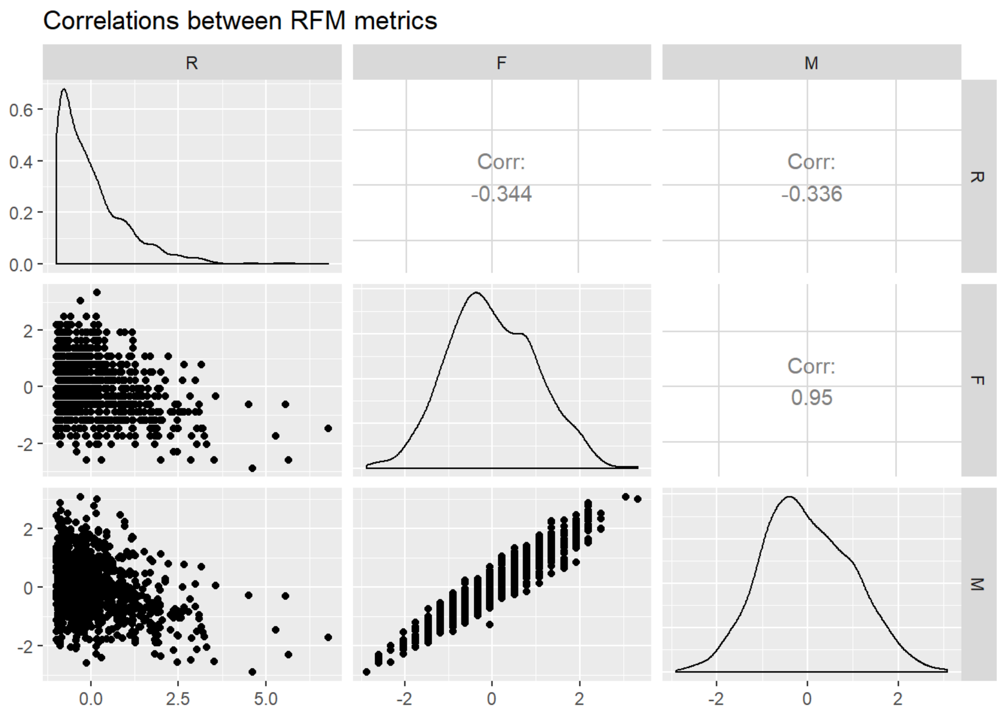
```
pre <- preProcess(RFM)
RFM_centered <- predict(pre, RFM)
```

3.3 Exploratory analysis

```
RFM_centered %>% gather(metric, value, -CustomerID) %>%
    ggplot(aes(x=value)) +
        geom_histogram(color = qpal[1], fill = qpal[2]) +
        facet_grid(.~metric) +
        labs(title='Distributions of the centered and scaled RFM metrics')
```




```
RFM_centered %>%
    select(-CustomerID) %>%
    ggpairs(title='Correlations between RFM metrics')
```



3.4 k-means Clustering

Based on the RFM metrics we conduct a k-means clustering with k = 2.

```
cluster <- RFM %>%
    select(-CustomerID) %>%
    kmeans(2)

cluster
```

```
## K-means clustering with 2 clusters of sizes 481, 608
##
## Cluster means:
##          R         F        M
## 1 82.00624 14.307692 1805.607
## 2 98.72368  8.735197 1051.641
##
## Clustering vector:
##    [1] 2 2 2 2 2 1 1 1 2 2 1 2 1 2 1 2 1 2 2 2 2 2 2 2 1 1 1 1 1 2 1 1 2
##   [35] 1 1 2 1 2 1 1 2 2 1 1 1 1 1 1 2 2 1 2 2 2 2 1 2 2 2 2 2 1 1 1 2 2 2
##   [69] 1 2 1 2 2 1 2 1 1 2 1 2 1 2 2 1 2 2 2 2 1 1 2 2 1 2 1 2 1 1 2 2 1 1
##  [103] 2 2 1 1 1 1 2 1 2 2 1 2 1 1 2 1 2 2 2 1 2 1 2 2 2 2 2 2 2 1 2 2 1 2
##  [137] 1 2 2 2 2 1 1 1 1 1 2 2 1 2 2 2 2 1 1 2 1 2 2 2 2 2 2 2 1 2 2 1 1 2
##  [171] 1 2 1 2 2 1 2 1 1 1 1 1 1 1 1 2 2 2 1 2 1 1 2 2 2 1 1 1 2 1 1 2 2 1
##  [205] 1 2 2 2 1 2 1 2 2 1 2 2 2 1 1 2 2 2 2 1 2 1 1 2 2 2 2 1 2 2 1 1 2 2
##  [239] 2 1 2 2 1 2 2 2 1 2 1 2 2 1 2 1 1 1 2 1 2 2 2 1 2 2 1 2 2 2 2 2 1 2
##  [273] 2 1 2 1 2 2 1 1 1 2 2 2 2 1 1 2 1 2 1 2 1 2 1 1 2 1 2 1 1 2 2 2 1 2
##  [307] 2 2 1 2 2 2 2 1 1 2 1 1 1 2 1 1 2 1 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2
##  [341] 1 2 2 1 1 1 2 1 2 2 2 2 2 2 1 1 2 1 2 2 1 2 1 2 2 2 2 1 2 2 2 2 2 2
##  [375] 2 1 1 1 2 1 1 2 1 1 2 2 1 2 2 1 1 1 1 2 2 2 2 1 1 2 2 2 2 2 2 1 2 2
##  [409] 2 2 1 2 1 1 1 1 2 1 2 1 2 2 1 1 2 1 2 2 1 2 2 1 1 2 2 2 1 1 2 1 2 1
##  [443] 1 1 2 2 2 2 2 2 2 2 2 2 1 1 1 1 2 2 1 2 2 1 1 1 2 2 2 2 1 2 1 2 2 2
##  [477] 2 2 1 2 2 2 2 2 2 2 1 2 2 1 1 1 1 2 2 2 1 1 2 2 2 1 1 2 1 1 2 2 2 2
##  [511] 2 2 1 2 2 2 1 1 1 1 2 2 2 2 1 2 1 2 1 1 1 2 2 2 1 1 1 2 2 2 1 2 1 2
##  [545] 1 1 1 1 2 2 2 1 1 1 1 2 2 2 2 2 2 1 1 1 2 2 1 2 1 2 2 1 2 2 1 1 1 2
##  [579] 1 1 1 2 1 2 2 1 2 2 1 2 1 1 2 1 2 2 1 1 2 1 2 2 1 1 2 1 2 2 1 2 1 1
##  [613] 1 2 1 1 1 1 2 2 2 2 2 2 2 2 2 1 2 2 2 1 2 1 2 2 2 1 2 1 2 1 1 1 2 1
##  [647] 2 2 2 1 1 2 2 2 2 1 1 1 2 1 2 1 2 2 1 1 2 1 2 2 2 1 1 1 1 2 1 2 1 1
##  [681] 2 2 1 1 2 1 2 2 1 2 2 2 1 1 1 1 2 1 1 1 1 2 2 1 2 2 2 1 2 1 2 2 2 1
##  [715] 2 2 1 1 1 1 2 2 2 1 1 1 2 2 2 2 1 1 1 2 1 1 2 2 2 1 1 2 1 2 1 2 2 1
##  [749] 1 1 2 1 1 2 2 2 1 2 2 2 1 1 1 1 1 2 1 2 1 2 2 1 2 2 1 1 2 2 2 1 2 1
##  [783] 2 2 1 2 1 2 2 2 2 2 2 1 2 1 1 1 1 2 1 1 2 2 1 2 1 1 2 1 2 2 2 2 2 2
##  [817] 2 2 2 2 2 1 1 1 1 1 2 1 1 1 1 1 2 1 2 2 2 2 1 1 2 2 2 2 2 2 2 2 1 1
##  [851] 1 1 1 1 2 1 2 2 1 2 2 2 2 1 2 2 2 1 1 2 1 1 1 2 2 1 1 1 2 2 1 1 2 1
##  [885] 2 2 1 2 1 1 2 1 1 1 2 2 2 1 1 1 1 2 2 2 2 2 2 1 2 2 1 1 1 1 1 1 2 2
##  [919] 1 1 1 1 1 2 2 1 2 2 1 2 2 2 1 2 2 2 1 2 1 2 1 1 1 2 1 2 2 2 1 1 1 2
##  [953] 2 1 2 2 2 1 1 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 1 1 1 1 2 1 2 1 2 2 2 2
##  [987] 1 1 2 2 1 2 2 2 1 1 2 2 2 2 1 1 1 1 1 1 2 2 2 2 2 1 1 2 1 1 1 2 1 2
## [1021] 2 1 1 2 2 1 2 1 1 1 2 1 1 1 2 2 2 1 2 2 1 1 1 2 1 2 2 2 1 2 1 2 1 2
## [1055] 1 2 1 2 1 1 1 2 2 2 2 1 1 2 2 2 1 2 1 2 1 2 2 2 1 2 1 2 1 2 1 1 2 2
## [1089] 1
##
## Within cluster sum of squares by cluster:
## [1] 36562548 40540396
##  (between_SS / total_SS =  66.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Joining the cluster assignments to the RFM table.

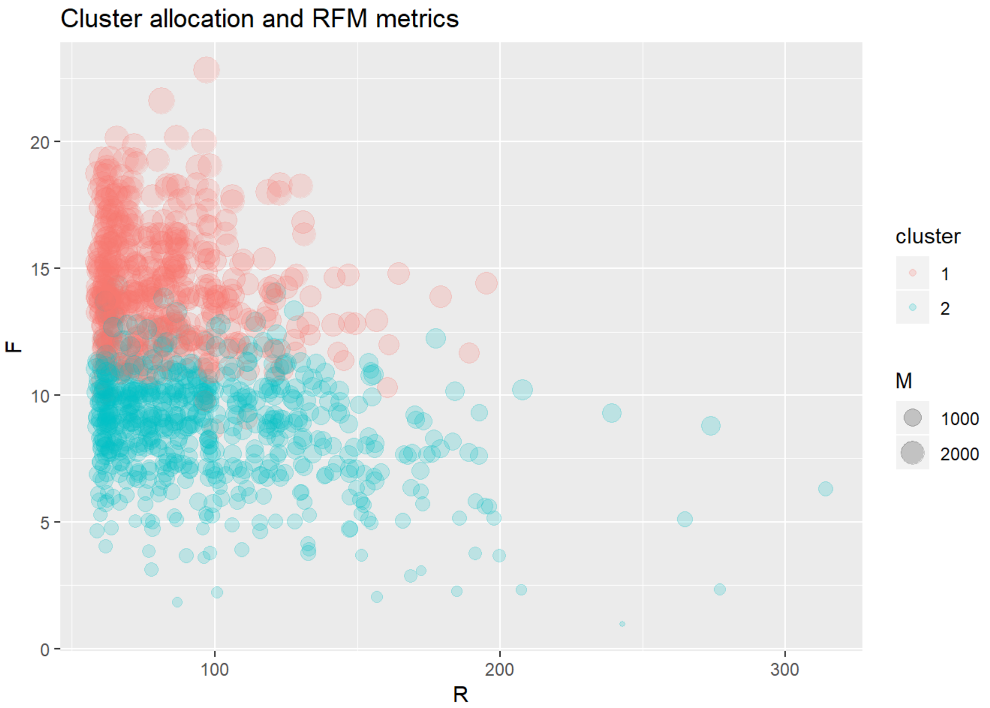
```
RFM_clust <- RFM %>%
    mutate(cluster = as.factor(cluster$cluster))

RFM_clust
```

CustomerID	R	F	M	cluster
<fctr>	<int>	<int>	<int>	<fctr>
1002	66	9	1165	2
1003	96	11	1313	2
1004	110	9	1174	2
1005	82	9	1184	2
1006	97	10	1067	2
1007	60	17	1710	1
1008	73	15	1616	1
1009	104	16	2079	1

CustomerID	R	F	M	cluster
<fctr>	<int>	<int>	<int>	<fctr>
1010	77	10	1250	2
1011	91	9	1127	2

1-10 of 1,089 rows Previous **1** 2 3 4 5 6 ... 109 Next

```
RFM_clust %>%
    ggplot(aes(x = R, y = F, size = M,color = cluster)) +
    geom_jitter(alpha = .2) + #scale_size(range=c(1,10)) +
    labs(title = 'Cluster allocation and RFM metrics')
```



The interpretation of the two clusters can be seen as **high-potential** and **low-potential** customers.

4 Classification

4.1 Load data

In order to predict cluster allocation of future customers, we need a demographic dataset about our existing customer portfolio.

We load the following set consisting of the attributes

1. Marital Status
2. Age
3. Sex
4. Education

We try to find a relationship between these attributes and the cluster segment to which a customer belongs via training a supervised learning algorithm.

```
customers <- read_csv2('data/customers_sim.csv') %>%
    mutate_at(vars(CustomerID, MaritalStatus, Sex, Education), as.factor)
customers
```

CustomerID	Age	MaritalStatus	Sex	Education
<fctr>	<int>	<fctr>	<fctr>	<fctr>
1002	42	Married	Male	High School
1003	71	Married	Female	Master
1004	26	Married	Female	High School
1005	25	Married	Male	High School
1006	65	Married	Male	High School

CustomerID	Age	MaritalStatus	Sex	Education
<fctr>	<int>	<fctr>	<fctr>	<fctr>
1010	48	Married	Male	High School
1011	79	Married	Female	High School
1013	52	Married	Female	Bachelor
1015	45	Divorced	Male	Bachelor
1017	97	Married	Male	High School

1-10 of 1,089 rows Previous **1** 2 3 4 5 6 ... 109 Next

Structure of the dataset:

```
str(customers)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    1089 obs. of  5 variables:
##  $ CustomerID   : Factor w/ 1089 levels "1002","1003",..: 1 2 3 4 5 9 10 12 14 16 ...
##  $ Age          : int  42 71 26 25 65 48 79 52 45 97 ...
##  $ MaritalStatus: Factor w/ 3 levels "Divorced","Married",..: 2 2 2 2 2 2 2 2 1 2 ...
##  $ Sex          : Factor w/ 2 levels "Female","Male": 2 1 1 2 2 2 1 1 2 2 ...
##  $ Education    : Factor w/ 3 levels "Bachelor","High School",..: 2 3 2 2 2 2 2 1 1 2 ...
```

4.2 Preprocessing

4.2.1 Joining the cluster allocation

We need to add the column **cluster** from our previous analysis to the demographic dataset. The join is done over the common field **CustomerID**.

```
modeldat <- customers %>%
    inner_join(RFM_clust) %>%
    select(-CustomerID, -R, -F, -M)
modeldat
```

Age	MaritalStatus	Sex	Education	cluster
<int>	<fctr>	<fctr>	<fctr>	<fctr>
42	Married	Male	High School	2
71	Married	Female	Master	2
26	Married	Female	High School	2
25	Married	Male	High School	2
65	Married	Male	High School	2
48	Married	Male	High School	2
79	Married	Female	High School	2
52	Married	Female	Bachelor	2
45	Divorced	Male	Bachelor	2
97	Married	Male	High School	2

1-10 of 1,089 rows Previous **1** 2 3 4 5 6 ... 109 Next

4.2.2 Splitting

We split the joined dataframe in training- and testset using a stratified splitting strategy on variable **cluster**, in order to retain an even distribution of both clusters in both sets.

```
intrain <- createDataPartition(modeldat$cluster, p = .8, list=FALSE)

training <- modeldat[intrain,]
testing <- modeldat[-intrain,]
```

4.3 Model training

We activate 10-fold Cross-Validation for the training procedures.

```r
trControl <- trainControl(method = 'cv')
fit <- list()
```

We will fit three different models.

4.3.1 Logistic regression

```r
fit$glm <- train(cluster ~ .,
                       method="glm",
                       family="binomial",
                       trControl = trControl,
                       data = training)

fit$glm
```

```
## Generalized Linear Model
##
## 872 samples
##   4 predictor
##   2 classes: '1', '2'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 785, 784, 785, 785, 785, 785, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9128265  0.823225
```

4.3.2 Partition tree

```r
fit$rpart <- train(cluster ~ .,
                       method="rpart",
                       trControl = trControl,
                       data = training,
                       tuneLength = 10)

fit$rpart
```

```
## CART
##
## 872 samples
##   4 predictor
##   2 classes: '1', '2'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 785, 785, 784, 785, 785, 785, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.00000000  0.9118582  0.8213897
##   0.06291486  0.8923170  0.7834229
##   0.12582973  0.8923170  0.7834229
##   0.18874459  0.8798170  0.7593154
##   0.25165945  0.8086556  0.6234743
##   0.31457431  0.8086556  0.6234743
##   0.37748918  0.8086556  0.6234743
##   0.44040404  0.8086556  0.6234743
##   0.50331890  0.8086556  0.6234743
##   0.56623377  0.6658061  0.2762678
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.
```

4.3.3 Boosted trees

```
fit$gbm <- train(cluster ~ .,
                       method="gbm",
                       trControl = trControl,
                       data = training,
                       tuneLength = 10,
                 verbose =F)

fit$gbm
```

```
## Stochastic Gradient Boosting
##
## 872 samples
##   4 predictor
##   2 classes: '1', '2'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 784, 785, 784, 784, 785, 785, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##    1                  50      0.9164419  0.8305919
##    1                 100      0.9175385  0.8332359
##    1                 150      0.9117904  0.8217131
##    1                 200      0.9152129  0.8286095
##    1                 250      0.9152262  0.8284304
##    1                 300      0.9152129  0.8285262
##    1                 350      0.9140768  0.8260313
##    1                 400      0.9140765  0.8259965
##    1                 450      0.9117512  0.8213206
##    1                 500      0.9140370  0.8258739
##    2                  50      0.9072723  0.8116761
##    2                 100      0.9117777  0.8212899
##    2                 150      0.9140768  0.8258689
##    2                 200      0.9140370  0.8256667
##    2                 250      0.9151995  0.8281301
##    2                 300      0.9129271  0.8233804
##    2                 350      0.9140504  0.8258908
##    2                 400      0.9128879  0.8233228
##    2                 450      0.9094654  0.8167128
##    2                 500      0.9083160  0.8141980
##    3                  50      0.9095317  0.8165134
##    3                 100      0.9106945  0.8188698
##    3                 150      0.9140768  0.8256387
##    3                 200      0.9117515  0.8210686
##    3                 250      0.9106018  0.8189494
##    3                 300      0.9060035  0.8095279
##    3                 350      0.9060430  0.8094760
##    3                 400      0.9071660  0.8121910
##    3                 450      0.9060166  0.8097875
##    3                 500      0.9048674  0.8071522
##    4                  50      0.9140768  0.8256984
##    4                 100      0.9106550  0.8184887
##    4                 150      0.9117646  0.8211767
##    4                 200      0.9083160  0.8140776
##    4                 250      0.9083294  0.8141013
##    4                 300      0.9083686  0.8145254
##    4                 350      0.9014711  0.8002677
##    4                 400      0.8980754  0.7937154
##    4                 450      0.8969390  0.7909928
##    4                 500      0.8934771  0.7843718
##    5                  50      0.9175649  0.8328207
##    5                 100      0.9151998  0.8281958
##    5                 150      0.9174725  0.8330337
##    5                 200      0.9060694  0.8098135
##    5                 250      0.9026737  0.8030421
##    5                 300      0.8981152  0.7937425
##    5                 350      0.8992251  0.7960212
##    5                 400      0.8934904  0.7846125
##    5                 450      0.8866331  0.7704846
##    5                 500      0.8900947  0.7774450
##    6                  50      0.9129271  0.8234305
##    6                 100      0.9129010  0.8234737
##    6                 150      0.9048407  0.8073437
##    6                 200      0.9014183  0.8005295
##    6                 250      0.8979830  0.7933903
##    6                 300      0.8968600  0.7910824
##    6                 350      0.8945873  0.7866861
##    6                 400      0.8911254  0.7794348
##    6                 450      0.8910989  0.7793779
##    6                 500      0.8819685  0.7608434
##    7                  50      0.9187143  0.8352851
##    7                 100      0.9094919  0.8165440
##    7                 150      0.9026211  0.8027927
##    7                 200      0.9014186  0.8003361
```

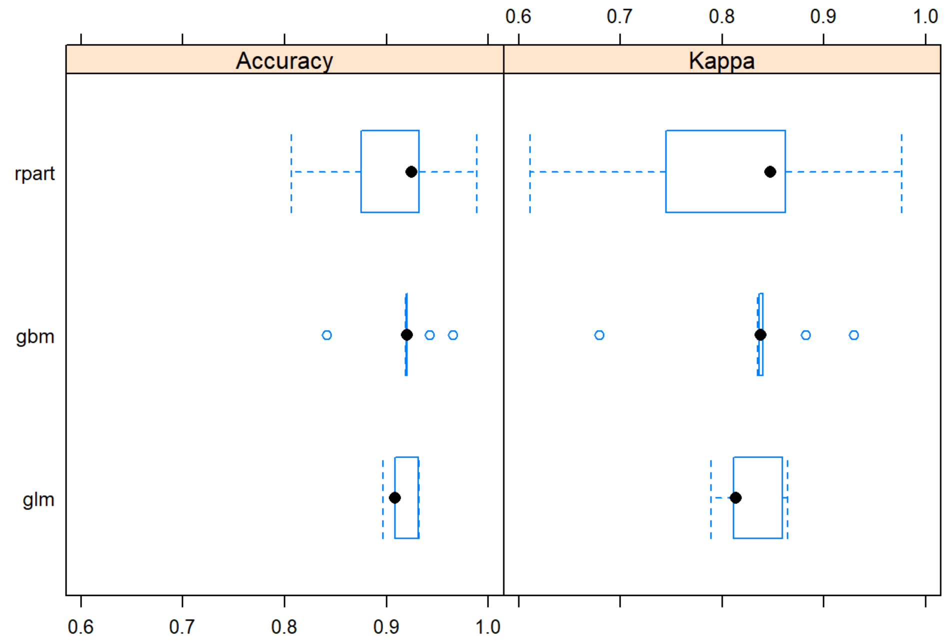
```
##    7                 250      0.8968467  0.7911750
##    7                 300      0.8923146  0.7822275
##    7                 350      0.8923015  0.7819296
##    7                 400      0.8934509  0.7842158
##    7                 450      0.8888924  0.7751615
##    7                 500      0.8866197  0.7703763
##    8                  50      0.9095581  0.8165194
##    8                 100      0.9140243  0.8256517
##    8                 150      0.9025813  0.8029741
##    8                 200      0.8979964  0.7934607
##    8                 250      0.8991592  0.7958589
##    8                 300      0.8980623  0.7936079
##    8                 350      0.8900683  0.7773836
##    8                 400      0.8900813  0.7776185
##    8                 450      0.8912046  0.7800488
##    8                 500      0.8900552  0.7776674
##    9                  50      0.9152660  0.8285303
##    9                 100      0.9083558  0.8144241
##    9                 150      0.9048674  0.8076700
##    9                 200      0.8957370  0.7891649
##    9                 250      0.8912050  0.7798946
##    9                 300      0.8888794  0.7753132
##    9                 350      0.8911785  0.7795098
##    9                 400      0.8911785  0.7795098
##    9                 450      0.8912050  0.7795354
##    9                 500      0.8934774  0.7843910
##   10                  50      0.9106152  0.8187641
##   10                 100      0.9106018  0.8189916
##   10                 150      0.9003350  0.7981628
##   10                 200      0.8946268  0.7869534
##   10                 250      0.8934904  0.7845341
##   10                 300      0.8889583  0.7750633
##   10                 350      0.8900947  0.7775270
##   10                 400      0.8912441  0.7797228
##   10                 450      0.8854967  0.7678850
##   10                 500      0.8912043  0.7799812
## 
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
## 
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 50, interaction.depth
##  = 7, shrinkage = 0.1 and n.minobsinnode = 10.
```

5 Model Evaluation

5.1 Performance on training set

We collect the resamples of all three fitted models and plot the distribution of the performance statistics **Accuracy** and **Cohen's Kappa**.

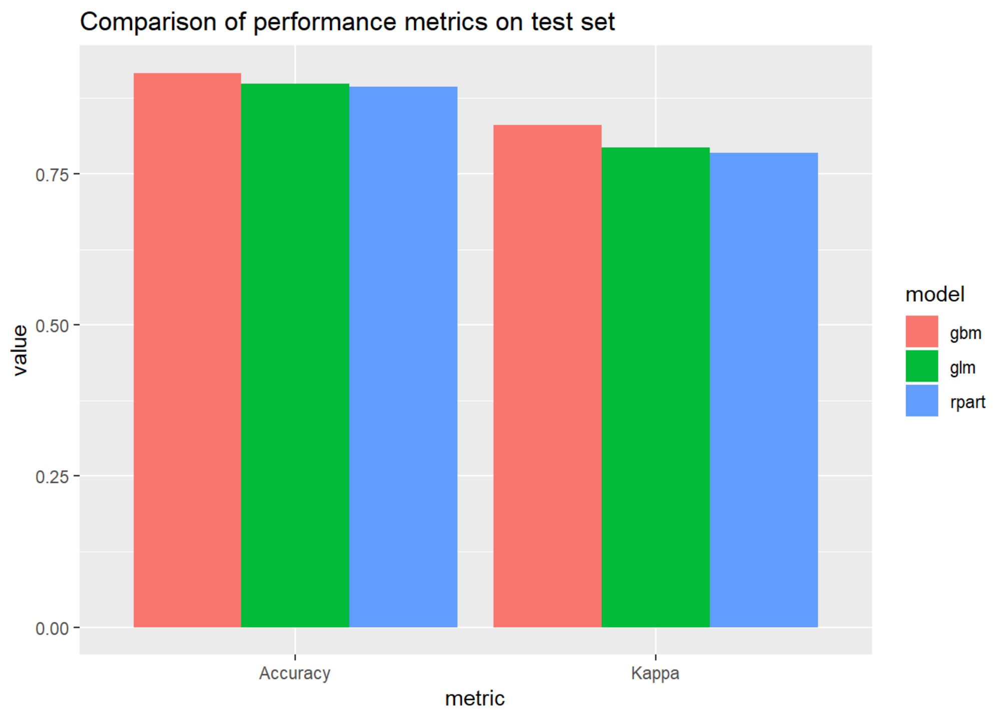
```
rs <- resamples(fit)
bwplot(rs)
```



All three models perform almost similarly. Although the boosted model scores the highest mean accuracy, it also has a high variance in the metrics. Therefore it might not be the most robust choice and we could still stick with a mucher easier decision tree model, while retaining some interpretability.

5.2 Performance on test set

```
prediction <- lapply(fit, predict, newdata = testing) %>%
    bind_cols()

prediction %>%
    sapply(postResample, obs = testing$cluster) %>%
    as.data.frame() %>%
    rownames_to_column('metric') %>%
    gather(model, value, -metric) %>%
    ggplot(aes(x=metric, y = value, fill = model)) +
    geom_col(position='dodge') +
    labs(title='Comparison of performance metrics on test set')
```



Showing the confusion matrix of the decision tree model on the test set.

```
predict(fit$rpart, testing) %>%
    confusionMatrix(testing$cluster)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1  85  12
##          2  11 109
##
##                Accuracy : 0.894
##                  95% CI : (0.8452, 0.9316)
##     No Information Rate : 0.5576
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7854
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8854
##             Specificity : 0.9008
##          Pos Pred Value : 0.8763
##          Neg Pred Value : 0.9083
##              Prevalence : 0.4424
##          Detection Rate : 0.3917
##    Detection Prevalence : 0.4470
##       Balanced Accuracy : 0.8931
##
##        'Positive' Class : 1
##
```

Visualizing the decision tree.

```
fit$rpart$finalModel %>% prp()
```