ML in R

Dataset: Campus Recruitment from Kaggle

This notebook will build a model to predict if a student would get employed. The purpose is to learn how to build basic ML models in R using the `caret` package.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(GGally)
```

```
## Warning: package 'GGally' was built under R version 4.2.2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.2
```

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

```
library(patchwork)
```

Load and inspect dataset

```
(df <- read_csv('../data/Placement_Data_Full_Class.csv'))
```

```
## Rows: 215 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (8): gender, ssc_b, hsc_b, hsc_s, degree_t, workex, specialisation, status
## dbl (7): sl_no, ssc_p, hsc_p, degree_p, etest_p, mba_p, salary
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## # A tibble: 215 x 15
##    sl_no gender ssc_p ssc_b   hsc_p hsc_b   hsc_s degre~1 degre~2 workex etest_p
##    <dbl> <chr>  <dbl> <chr>   <dbl> <chr>   <chr>   <dbl> <chr>   <chr>    <dbl>
##  1     1 M       67   Others   91   Others  Comm~    58   Sci&Te~ No        55
##  2     2 M       79.3 Central  78.3 Others  Scie~    77.5 Sci&Te~ Yes       86.5
##  3     3 M       65   Central  68   Central Arts     64   Comm&M~ No        75
##  4     4 M       56   Central  52   Central Scie~    52   Sci&Te~ No        66
##  5     5 M       85.8 Central  73.6 Central Comm~    73.3 Comm&M~ No        96.8
##  6     6 M       55   Others   49.8 Others  Scie~    67.2 Sci&Te~ Yes       55
##  7     7 F       46   Others   49.2 Others  Comm~    79   Comm&M~ No        74.3
##  8     8 M       82   Central  64   Central Scie~    66   Sci&Te~ Yes       67
##  9     9 M       73   Central  79   Central Comm~    72   Comm&M~ No        91.3
## 10    10 M       58   Central  70   Central Comm~    61   Comm&M~ No        54
## # ... with 205 more rows, 4 more variables: specialisation <chr>, mba_p <dbl>,
## #   status <chr>, salary <dbl>, and abbreviated variable names 1: degree_p,
## #   2: degree_t
```

Dataset contains 15 columns:

- sl_no: Serial Number
- gender: Gender- Male='M',Female='F'
- ssc_p: Secondary Education percentage- 10th Grade
- ssc_b: Board of Education- Central/ Others
- hsc_p: Higher Secondary Education percentage- 12th Grade
- hsc_b: Board of Education- Central/ Others
- hsc_s: Specialization in Higher Secondary Education
- degree_p: Degree Percentage
- degree_t: Under Graduation(Degree type)- Field of degree education
- workex: Work Experience
- etest_p: Employability test percentage (conducted by college)
- specialization: Post Graduation(MBA)- Specialization
- mba_p: MBA percentage
- status: Status of placement- Placed/Not placed
- salary: Salary offered by corporate to candidates

Note: percentage refers to their performance on that grade. For example, a student scores 120/200 in his 12th grade. The percentage (hsc_p) would be 60%

```
glimpse(df)
```

```
## Rows: 215
## Columns: 15
## $ sl_no          <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ~
## $ gender         <chr> "M", "M", "M", "M", "M", "M", "F", "M", "M", "M", "M", ~
## $ ssc_p          <dbl> 67.00, 79.33, 65.00, 56.00, 85.80, 55.00, 46.00, 82.00,~
## $ ssc_b          <chr> "Others", "Central", "Central", "Central", "Central", "~
## $ hsc_p          <dbl> 91.00, 78.33, 68.00, 52.00, 73.60, 49.80, 49.20, 64.00,~
## $ hsc_b          <chr> "Others", "Others", "Central", "Central", "Central", "O~
## $ hsc_s          <chr> "Commerce", "Science", "Arts", "Science", "Commerce", "~
## $ degree_p       <dbl> 58.00, 77.48, 64.00, 52.00, 73.30, 67.25, 79.00, 66.00,~
## $ degree_t       <chr> "Sci&Tech", "Sci&Tech", "Comm&Mgmt", "Sci&Tech", "Comm&~
## $ workex         <chr> "No", "Yes", "No", "No", "No", "Yes", "No", "Yes", "No"~
## $ etest_p        <dbl> 55.00, 86.50, 75.00, 66.00, 96.80, 55.00, 74.28, 67.00,~
## $ specialisation <chr> "Mkt&HR", "Mkt&Fin", "Mkt&Fin", "Mkt&HR", "Mkt&Fin", "M~
## $ mba_p          <dbl> 58.80, 66.28, 57.80, 59.43, 55.50, 51.58, 53.29, 62.14,~
## $ status         <chr> "Placed", "Placed", "Placed", "Not Placed", "Placed", "~
## $ salary         <dbl> 270000, 200000, 250000, NA, 425000, NA, NA, 252000, 231~
```

```
sapply(df, function(x) if (is.character(x)) unique(x)) %>% discard(is.null)
```

```
## $gender
## [1] "M" "F"
##
## $ssc_b
## [1] "Others"  "Central"
##
## $hsc_b
## [1] "Others"  "Central"
##
## $hsc_s
## [1] "Commerce" "Science"  "Arts"
##
## $degree_t
## [1] "Sci&Tech"  "Comm&Mgmt" "Others"
##
## $workex
## [1] "No"  "Yes"
##
## $specialisation
## [1] "Mkt&HR"  "Mkt&Fin"
##
## $status
## [1] "Placed"     "Not Placed"
```

Convert all character vector to factor

```
df <- df %>%
  mutate_if(is.character, as.factor)
```

All Data are completed. Salary will be null if and only if their status is “Not Placed”

```
sapply(df, function(x) sum(is.na(x)))
```

```
##          sl_no         gender          ssc_p          ssc_b          hsc_p
##             0              0              0              0              0
##          hsc_b          hsc_s       degree_p       degree_t         workex
##             0              0              0              0              0
##        etest_p specialisation          mba_p         status         salary
##             0              0              0              0             67
```

```
df %>%
  filter(status == 'Placed') %>%
  summarise(null_salary = sum(is.na(salary)))
```

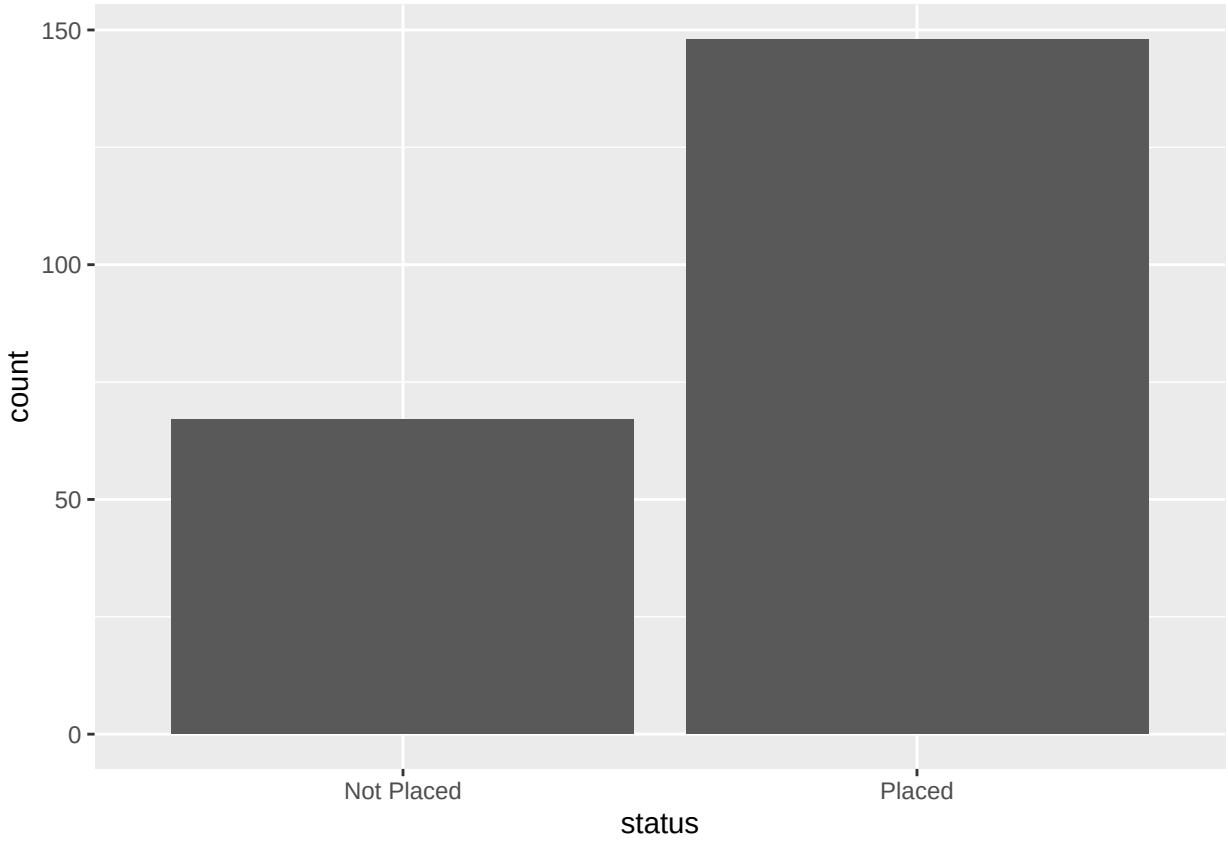
```
## # A tibble: 1 x 1
##   null_salary
##         <int>
## 1           0
```

EDA

```
df %>%
  count(status) %>%
  mutate(prop = n / sum(n))
```

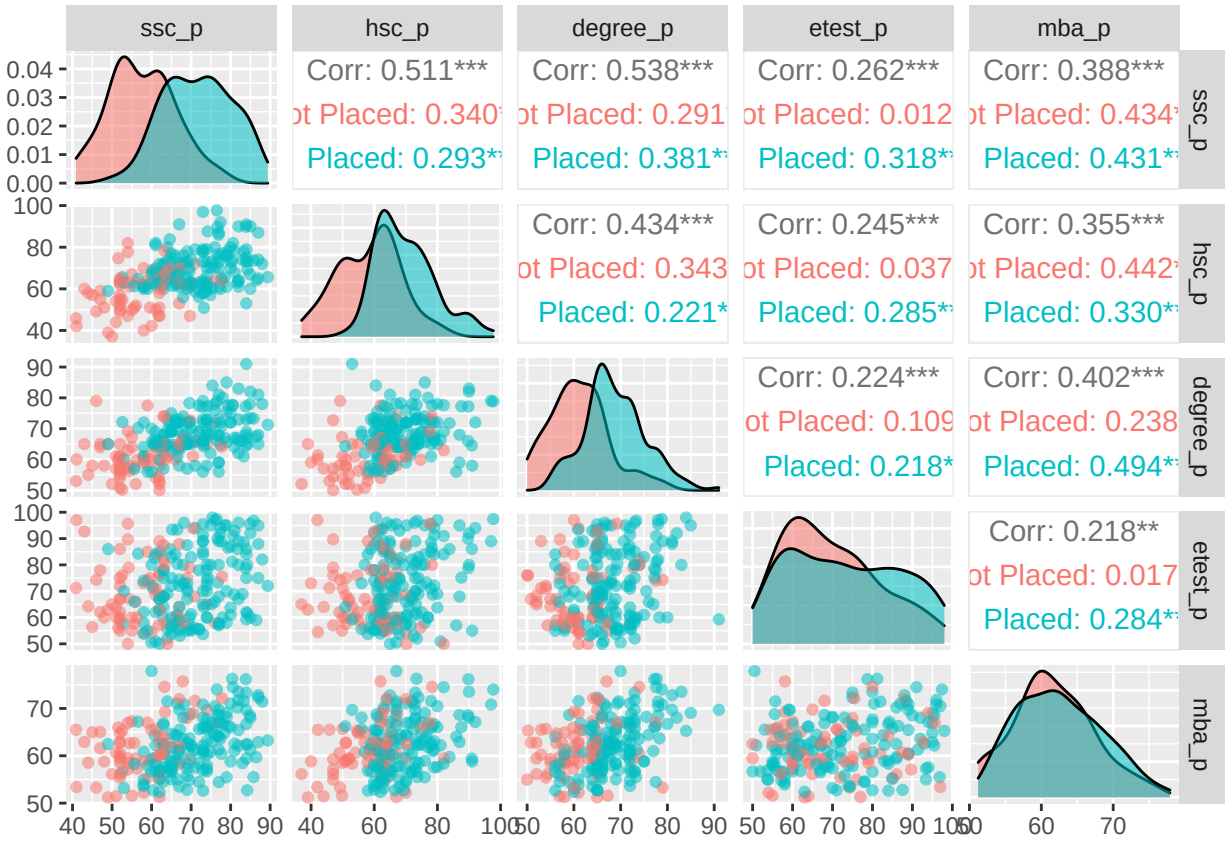
```
## # A tibble: 2 x 3
##   status         n  prop
##   <fct>      <int> <dbl>
## 1 Not Placed    67 0.312
## 2 Placed       148 0.688
```

```
ggplot(df, aes(status)) +
  geom_bar()
```



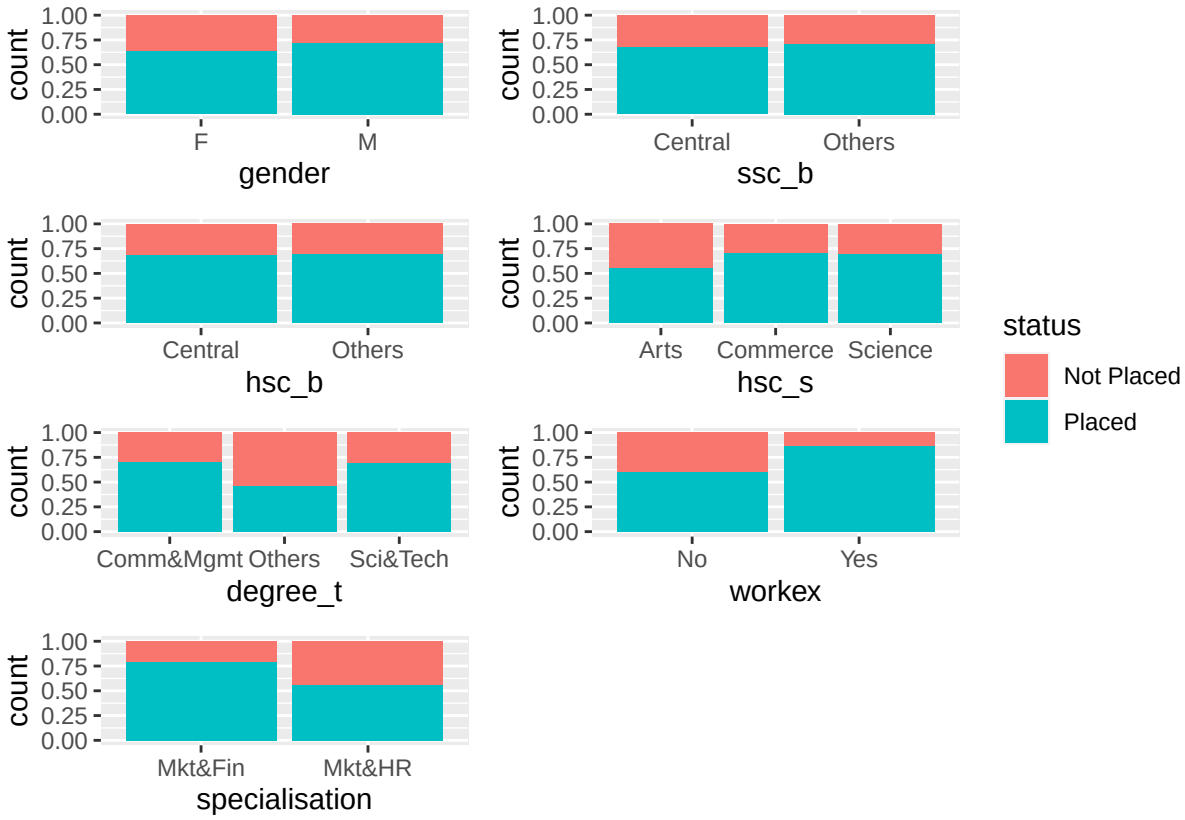
It seems like there is not much different between etest_p and mba_p among employed and unemployed students.

```
ggpairs(df,
        mapping = aes(color = status, alpha = .5),
        columns = which(sapply(names(df), function(x) endsWith(x, '_p'))))
```



Both values of ssc_b and hsc_b has similar a proportion of employed students. Similarly, there is a subtle difference between "Commerce" and "Science" (hsc_s) as well as "Comm&Mgmt" and "Sci&Tech" (degree_t).

```
cols <- c('gender', 'ssc_b', 'hsc_b', 'hsc_s', 'degree_t', 'workex', 'specialisation')
plot_list <- lapply(cols, function(x) {
  ggplot(df, aes(get(x), fill = status)) +
    geom_bar(position = 'fill') +
    labs(x = x)
})
wrap_plots(plot_list, ncol = 2, guides = 'collect')
```



Training Models

```
set.seed(7)

prep_df <- df %>% select(-c(sl_no, salary))

n <- nrow(prep_df)
id <- createDataPartition(prep_df$status, p = .8, list = FALSE)
train_df <- prep_df[id,]
test_df <- prep_df[-id,]
nrow(train_df)
```

```
## [1] 173
```

```
nrow(test_df)
```

```
## [1] 42
```

```
train_df %>%
  group_by(status) %>%
  count() %>%
  ungroup() %>%
  mutate(freq = n / sum(n))
```

```
## # A tibble: 2 x 3
```

```
##   status         n  freq
##   <fct>      <int> <dbl>
## 1 Not Placed    54 0.312
## 2 Placed       119 0.688
```

```
test_df %>%
  group_by(status) %>%
  count() %>%
  ungroup() %>%
  mutate(freq = n / sum(n))
```

```
## # A tibble: 2 x 3
##   status         n  freq
##   <fct>      <int> <dbl>
## 1 Not Placed    13 0.310
## 2 Placed        29 0.690
```

Predict using logistic model

```
set.seed(7)

ctrl <- trainControl(method = 'cv',
                     number = 5)

logit_model <- train(status ~ . - hsc_b - ssc_b,
                     data = train_df,
                     method = 'glm',
                     trControl = ctrl)

logit_model
```

```
## Generalized Linear Model
##
## 173 samples
##  12 predictor
##   2 classes: 'Not Placed', 'Placed'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 139, 138, 138, 139, 138
## Resampling results:
##
##   Accuracy   Kappa
##   0.8615126  0.6775211
```

```
logit_model$finalModel
```

```
##
## Call:  NULL
##
```

```
## Coefficients:
##             (Intercept)                 genderM                   ssc_p
##              -16.661453                1.063995                0.186449
##                   hsc_p           hsc_sCommerce            hsc_sScience
##                0.113165               -0.762667                -0.163891
##                degree_p          degree_tOthers      `degree_tSci&Tech`
##                0.154734               -1.063349                -1.301606
##              workexYes                 etest_p  `specialisationMkt&HR`
##                2.287810               -0.007809                -0.119293
##                   mba_p
##              -0.187684
##
## Degrees of Freedom: 172 Total (i.e. Null);  160 Residual
## Null Deviance:       214.8
## Residual Deviance: 91.39     AIC: 117.4
```

```
varImp(logit_model)
```

```
## glm variable importance
##
##                         Overall
## ssc_p                   100.000
## mba_p                    69.546
## hsc_p                    66.642
## workexYes                65.477
## degree_p                 60.001
## `degree_tSci&Tech`       35.358
## genderM                  32.862
## degree_tOthers           14.668
## hsc_sCommerce             9.642
## etest_p                   5.058
## `specialisationMkt&HR`    2.124
## hsc_sScience              0.000
```

Predict using decision tree

```
set.seed(7)

tree_model <- train(status ~ . - etest_p - hsc_b - ssc_b,
                    data = train_df,
                    method = 'rpart2',
                    trControl = ctrl,
                    tuneGrid = data.frame(maxdepth = seq(1, 5, 1)))
tree_model
```

```
## CART
##
## 173 samples
##  12 predictor
##   2 classes: 'Not Placed', 'Placed'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 139, 138, 138, 139, 138
## Resampling results across tuning parameters:
##
##   maxdepth  Accuracy   Kappa
##   1         0.7689076  0.4559131
##   2         0.8097479  0.5263333
##   3         0.8154622  0.5517055
##   4         0.8152941  0.5371019
##   5         0.8152941  0.5371019
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was maxdepth = 3.
```

```
set.seed(7)

grid <- expand.grid(
  mtry = seq(2, 12, 2),
  min.node.size = seq(3, 9, 2),
  splitrule = c('gini', 'extratrees')
)

rf_model <- train(status ~ .,
                      data = train_df,
                      method = 'ranger',
                      trControl = ctrl,
                      tuneGrid = grid)

rf_model
```

```
## Random Forest
##
## 173 samples
##  12 predictor
##   2 classes: 'Not Placed', 'Placed'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 139, 138, 138, 139, 138
## Resampling results across tuning parameters:
##
##   mtry  min.node.size  splitrule   Accuracy   Kappa
##    2    3              gini        0.8554622  0.6296567
##    2    3              extratrees  0.8265546  0.5329104
##    2    5              gini        0.8668908  0.6598943
##    2    5              extratrees  0.8151261  0.4921747
##    2    7              gini        0.8497479  0.6115198
##    2    7              extratrees  0.8210084  0.5099508
##    2    9              gini        0.8615126  0.6472679
##    2    9              extratrees  0.8211765  0.5086043
##    4    3              gini        0.8615126  0.6650903
##    4    3              extratrees  0.8615126  0.6471618
##    4    5              gini        0.8556303  0.6476418
```

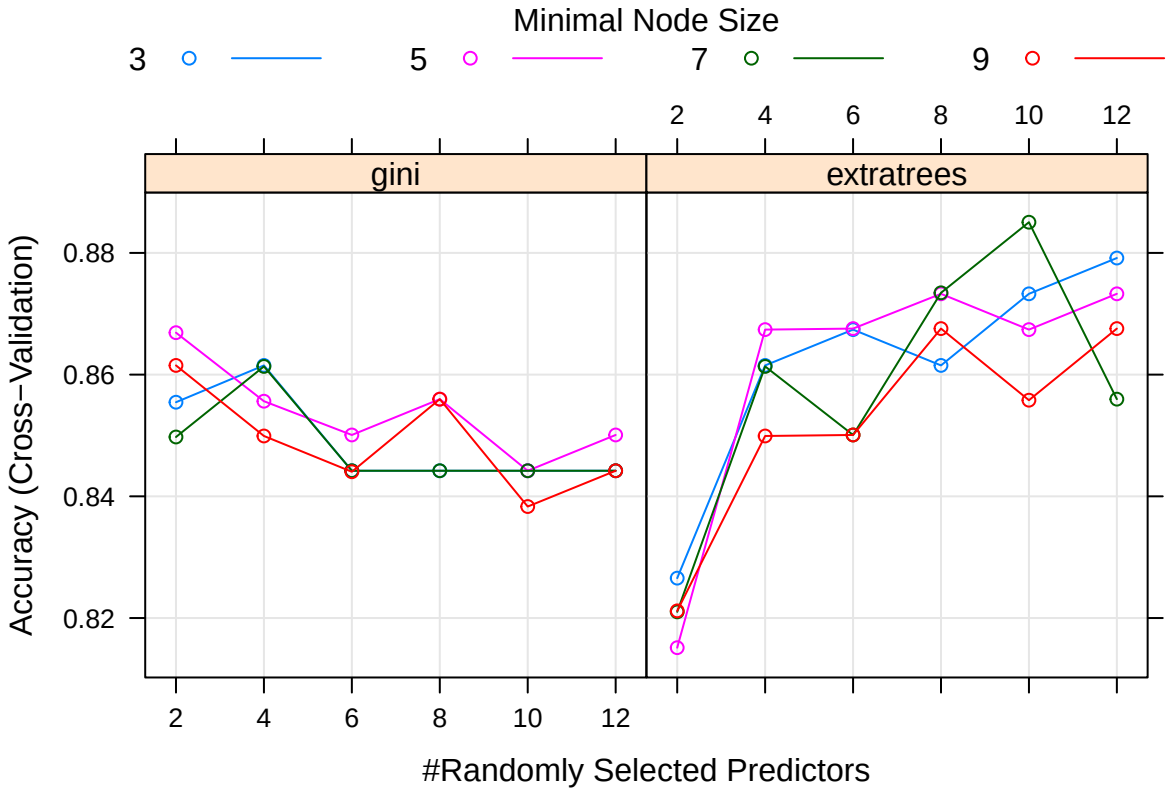
```
##    4    5             extratrees  0.8673950  0.6622978
##    4    7             gini        0.8613445  0.6618761
##    4    7             extratrees  0.8613445  0.6387465
##    4    9             gini        0.8499160  0.6370343
##    4    9             extratrees  0.8499160  0.6113259
##    6    3             gini        0.8442017  0.6234512
##    6    3             extratrees  0.8673950  0.6691774
##    6    5             gini        0.8500840  0.6341190
##    6    5             extratrees  0.8675630  0.6724782
##    6    7             gini        0.8442017  0.6170697
##    6    7             extratrees  0.8500840  0.6258508
##    6    9             gini        0.8440336  0.6246399
##    6    9             extratrees  0.8500840  0.6281634
##    8    3             gini        0.8442017  0.6197463
##    8    3             extratrees  0.8615126  0.6582078
##    8    5             gini        0.8559664  0.6509816
##    8    5             extratrees  0.8732773  0.6835955
##    8    7             gini        0.8442017  0.6197463
##    8    7             extratrees  0.8734454  0.6888784
##    8    9             gini        0.8559664  0.6519251
##    8    9             extratrees  0.8675630  0.6743689
##   10    3             gini        0.8442017  0.6197463
##   10    3             extratrees  0.8732773  0.6900744
##   10    5             gini        0.8442017  0.6197463
##   10    5             extratrees  0.8673950  0.6756564
##   10    7             gini        0.8442017  0.6208194
##   10    7             extratrees  0.8850420  0.7195869
##   10    9             gini        0.8383193  0.5996429
##   10    9             extratrees  0.8557983  0.6456007
##   12    3             gini        0.8442017  0.6197463
##   12    3             extratrees  0.8791597  0.7064746
##   12    5             gini        0.8500840  0.6265203
##   12    5             extratrees  0.8732773  0.6900744
##   12    7             gini        0.8442017  0.6197463
##   12    7             extratrees  0.8559664  0.6444221
##   12    9             gini        0.8442017  0.6140156
##   12    9             extratrees  0.8675630  0.6724782
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 10, splitrule = extratrees
##  and min.node.size = 7.
```

```
p <- predict(rf_model, newdata = test_df)
confusionMatrix(p,
                              test_df$status,
                            positive = 'Placed',
                              mode = 'prec_recall')
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Not Placed Placed
##   Not Placed         10      0
##   Placed              3     29
```

```
##
##                Accuracy : 0.9286
##                  95% CI : (0.8052, 0.985)
##     No Information Rate : 0.6905
##     P-Value [Acc > NIR] : 0.0002153
##
##                   Kappa : 0.8215
##
##  Mcnemar's Test P-Value : 0.2482131
##
##               Precision : 0.9062
##                  Recall : 1.0000
##                      F1 : 0.9508
##              Prevalence : 0.6905
##          Detection Rate : 0.6905
##    Detection Prevalence : 0.7619
##       Balanced Accuracy : 0.8846
##
##        'Positive' Class : Placed
##
```

```
plot(rf_model)
```




```
model_list <- list(logistic = logit_model,
                           tree = tree_model,
```

```
                          rf = rf_model)
resamp <- resamples(model_list)
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: logistic, tree, rf
## Number of resamples: 5
##
## Accuracy
##               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## logistic 0.8000000 0.8571429 0.8823529 0.8615126 0.8823529 0.8857143    0
## tree     0.7428571 0.7714286 0.8529412 0.8154622 0.8529412 0.8571429    0
## rf       0.8285714 0.8285714 0.8857143 0.8850420 0.9117647 0.9705882    0
##
## Kappa
##               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## logistic 0.5679012 0.6601942 0.6991150 0.6775211 0.7058824 0.7545126    0
## tree     0.3883495 0.4117647 0.6118721 0.5517055 0.6601942 0.6863469    0
## rf       0.5588235 0.6209386 0.7058824 0.7195869 0.7811159 0.9311741    0
```

Choose logistic model as a final model because it is simple and has the best performance on the test set

```
for (model in model_list) {
  p <- predict(model, newdata = test_df)
  print(model$modelInfo$label)
  print(confusionMatrix(p,
                                test_df$status,
                               positive = 'Placed',
                                mode = 'prec_recall'))
}
```

```
## [1] "Generalized Linear Model"
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Not Placed Placed
##   Not Placed         11      0
##   Placed              2     29
##
##                Accuracy : 0.9524
##                  95% CI : (0.8384, 0.9942)
##     No Information Rate : 0.6905
##     P-Value [Acc > NIR] : 3.384e-05
##
##                   Kappa : 0.8837
##
##  Mcnemar's Test P-Value : 0.4795
##
##               Precision : 0.9355
```

```
##                  Recall : 1.0000
##                      F1 : 0.9667
##              Prevalence : 0.6905
##          Detection Rate : 0.6905
##    Detection Prevalence : 0.7381
##       Balanced Accuracy : 0.9231
##
##        'Positive' Class : Placed
##
## [1] "CART"
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Not Placed Placed
##   Not Placed          8      1
##   Placed              5     28
##
##                Accuracy : 0.8571
##                  95% CI : (0.7146, 0.9457)
##     No Information Rate : 0.6905
##     P-Value [Acc > NIR] : 0.01118
##
##                   Kappa : 0.6348
##
##  Mcnemar's Test P-Value : 0.22067
##
##               Precision : 0.8485
##                  Recall : 0.9655
##                      F1 : 0.9032
##              Prevalence : 0.6905
##          Detection Rate : 0.6667
##    Detection Prevalence : 0.7857
##       Balanced Accuracy : 0.7905
##
##        'Positive' Class : Placed
##
## [1] "Random Forest"
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Not Placed Placed
##   Not Placed         10      0
##   Placed              3     29
##
##                Accuracy : 0.9286
##                  95% CI : (0.8052, 0.985)
##     No Information Rate : 0.6905
##     P-Value [Acc > NIR] : 0.0002153
##
##                   Kappa : 0.8215
##
##  Mcnemar's Test P-Value : 0.2482131
##
##               Precision : 0.9062
```

```
##                  Recall : 1.0000
##                      F1 : 0.9508
##              Prevalence : 0.6905
##          Detection Rate : 0.6905
##    Detection Prevalence : 0.7619
##       Balanced Accuracy : 0.8846
##
##        'Positive' Class : Placed
##
```