Assignment: Supervised Learning

Dunja Novaković

2021-08-01

Instructions

This assignment reviews the *Supervised Learning* analytical lecture. You will use the *supervised_learning.Rmd* file I reviewed in the video lectures to complete this assignment. You will *copy and paste* relevant code from that file and update it to answer the questions in this assignment. You will respond to questions in each section after executing relevant code to answer a question. You will submit this assignment to its *Submissions* folder on *D2L*. You will submit this *(1)* completed **R Markdown** script and *(2)* a *PDF*, *Word*, or *HTML* rendered version of it to *D2L* by the due date and time. As a first option, if you installed `TinyTeX` successfully, then I prefer a *PDF* version. As a second option, if you have *Microsoft Word*, then I prefer a *Word* version. As a third option, you can knit to *HTML*. The first two options work better with *D2L*.

To start:

For any analytical project, you want to create a clear project directory structure.
All materials from this course should exist in one folder on your computer. Inside of that main course folder, you should create folders to store course documentation, lecture analytical projects, assignments analytical projects, etc. Inside of your folder for assignments analytical projects, you should create folder for this assignment named *supervised_learning*.

Any analytical project folder should contain inside it at least three additional folders named *scripts*, *data*, and *plots*. Store this script in the *scripts* folder, the data for this assignment in the *data* folder, and any requested plots in the *plots* folder. Each analytical project should also contain a **.Rproj** file in its top-level directory. Go to the *File* menu in *RStudio*, select *New Project…*, choose *Existing Directory*, go to the folder you created to contain this analytical project. Select it as the top-level directory for this **RStudio Project**.

Global Settings

The first code chunk sets the global settings for the remaining code chunks in the document. Do *not* change anything in this code chunk.

Load Packages

In this code chunk, we load packages we need for this assignment:

1. **here**,
2. **tidyverse**,
3. **skimr**,
4. **rpart**,
5. **rattle**,
6. **randomForest**,
7. **gbm**, and

8. **caret**.

We will use functions from these packages to import the data, examine the data, calculate summaries on the data, build logistic regression models, and create visualizations from the data. Do *not* change anything in this code chunk.

```
### load libaries for use in current working session
## here for workflow
library(here)
```

```
## here() starts at C:/Users/novak/OneDrive/Desktop/MGT 591/Assignments/supervised_learning
```

```
## tidyverse for data manipulation and plotting
# loads eight different libraries simultaneously
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
## skimr for summary statistics
library(skimr)

## rpart to build single decision trees
library(rpart)

## rattle to plot decision trees
library(rattle)
```

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

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
## randomForest for random forests
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## 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:dplyr':
##
##     combine
```

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

```
## gbm for generalized boosted models
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
## caret for supervised learning
library(caret)
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

Task 1: Load, Clean, and Examine Data

Load the **emp_job_info.rdata** data file with the correct functions. Left join **emp_job_info_1_full** and **emp_job_info_2_full** by **emp_id**. Name the joined data: **emp_data**. Remove all other objects from your global environment (i.e., keep **emp_data** but remove the other four data objects). Clean the data like in the analytical lecture. Run **skim_without_charts()** on **left** and **last_evaluation** in **emp_data** while grouping by **department**.

Question 1.1: How many employees left and remain in the **technical** department? What is the average evaluation of employees in the **marketing** department?

Response 1.1: *Remain: 2023, Left: 697. Avg. evaluation: 71.6.*

Produce a density plot of **last_evaluation** filled by **left**. Use a facet wrap for **department**. Label the axes and fill appropriately.

Question 1.2: What do you notice about the density curves for those who left versus those who remain regardless of department?

Response 1.2: *Density curves for those who left have a dip for evaluation scores between 60 and 80, regardless of department.*

Produce a horizontal bar plot with **department** represented on the y-axis and percentage of employees in each **salary** category filling the bars. You will need to group **emp_data** by **department** and **salary** first and count the number of employees in combination of those groups. Then, you will need to group just by **department** to calculate the percentage of employees in each department at each salary level. Then, you will pass this data into **ggplot**. The y-axis should represent **department** and x-axis should represent percentage of employees. Use **coord_flip** to appropriately create the horizontal bar plot.

Question 1.3: Does **IT** have more employees with a **low** or **medium** salary? Which department has the highest percentage of highly-paid employees?

Response 1.3: *IT has more employees with a low salary. Management has the highest percentage of highly-paid employees.*

```
#### Q1.1
### load data via the load and here functions
load(here("data", "emp_job_info.rdata"))

### join data tables
## create joined data table
emp_data <- emp_job_info_1_full %>%
  ## left join first and second data tables
  left_join(emp_job_info_2_full, by = "emp_id")

## clean global environment
# remove unnecessary data tables
rm(emp_job_info_1_full, emp_job_info_2_full,
   emp_job_info_1_samp, emp_job_info_2_samp)

### clean data
emp_data <- emp_data %>%
  ## change particular variables to factors
  mutate_at(vars(department, salary, promotion_last_5_years,
                 work_accident, left), as_factor) %>%
  ## assign levels of particular factors
  mutate_at(vars(promotion_last_5_years, work_accident, left),
            ~ fct_recode(., `No` = "0", `Yes` = "1")) %>%
  ## rescale outcomes
  mutate_at(vars(job_sat, last_evaluation), ~ 100*.)

### explore data
## call data
emp_data %>%
  ## group by department and salary
  group_by(department) %>%
  ## remove id variable
  select(left, last_evaluation) %>%
  ## summarize
  skim_without_charts()
```

```
## Adding missing grouping variables: 'department'
```

Table 1: Data summary

Name	Piped data
Number of rows	14999

Number of columns	3
Column type frequency:	
factor	1
numeric	1
Group variables	department

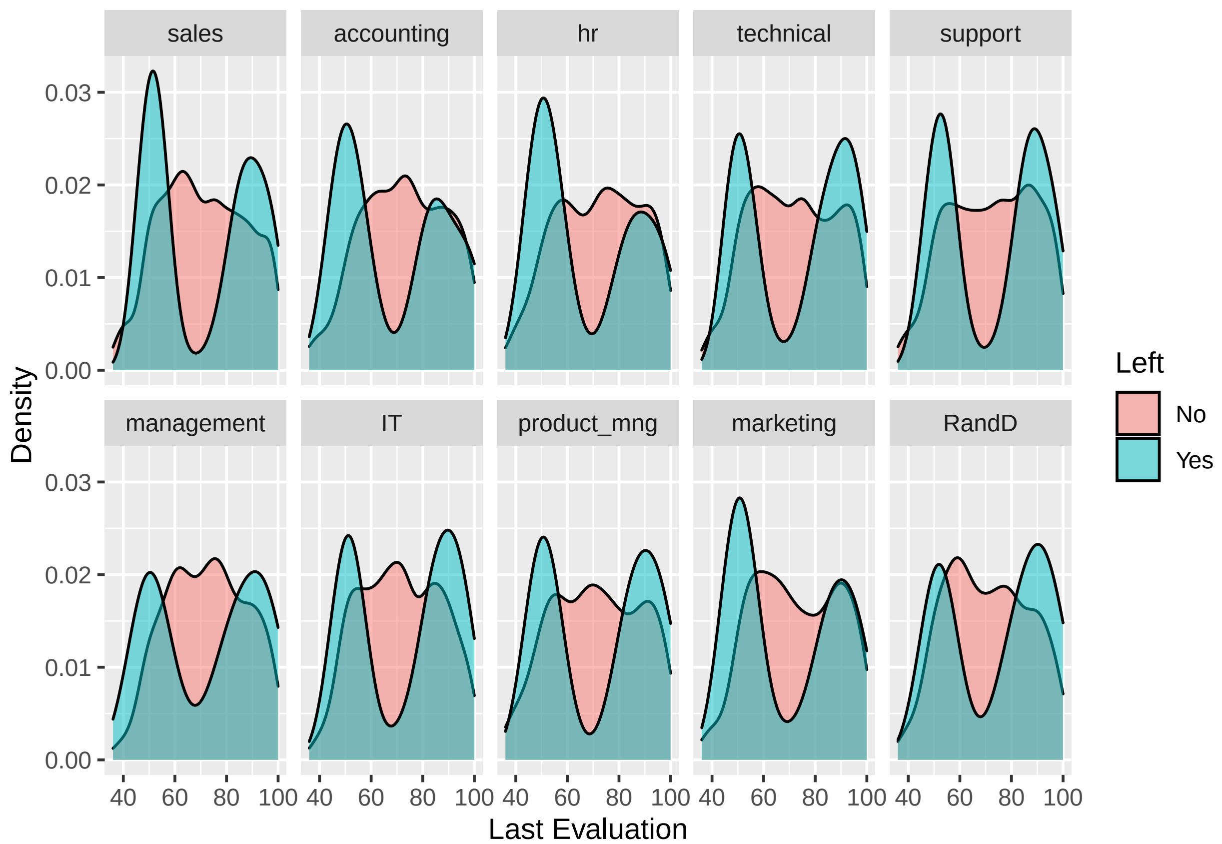
Variable type: factor

skim_variable	department	n_missing	complete_rate	ordered	n_unique	top_counts
left	sales	0	1	FALSE	2	No: 3126, Yes: 1014
left	accounting	0	1	FALSE	2	No: 563, Yes: 204
left	hr	0	1	FALSE	2	No: 524, Yes: 215
left	technical	0	1	FALSE	2	No: 2023, Yes: 697
left	support	0	1	FALSE	2	No: 1674, Yes: 555
left	management	0	1	FALSE	2	No: 539, Yes: 91
left	IT	0	1	FALSE	2	No: 954, Yes: 273
left	product_mng	0	1	FALSE	2	No: 704, Yes: 198
left	marketing	0	1	FALSE	2	No: 655, Yes: 203
left	RandD	0	1	FALSE	2	No: 666, Yes: 121

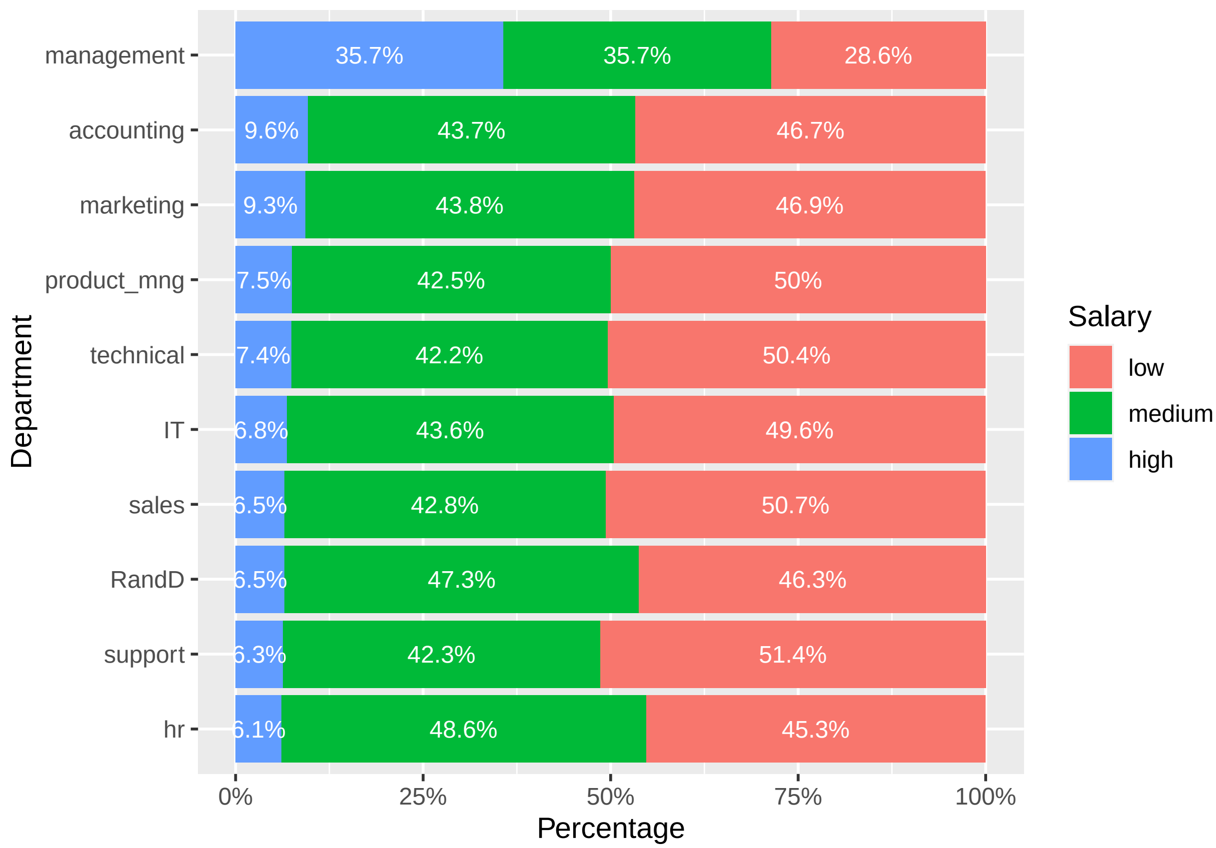
Variable type: numeric

skim_variable	department	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
last_evaluation	sales	0	1	70.97	17.15	36	55.0	70	86.0	100
last_evaluation	accounting	0	1	71.77	17.19	36	56.0	73	86.0	100
last_evaluation	hr	0	1	70.88	17.46	37	55.0	72	86.5	100
last_evaluation	technical	0	1	72.11	17.34	36	56.0	73	88.0	100
last_evaluation	support	0	1	72.31	17.12	36	56.0	74	87.0	100
last_evaluation	management	0	1	72.40	16.03	37	59.0	73	86.0	100
last_evaluation	IT	0	1	71.68	16.45	37	56.0	72	86.0	100
last_evaluation	product_mng	0	1	71.48	17.81	36	55.0	72	88.0	100
last_evaluation	marketing	0	1	71.59	17.34	36	55.0	71	87.0	100
last_evaluation	RandD	0	1	71.21	16.51	36	56.5	71	86.0	100

```r
#### Q1.2
### plot data
## density distributions for performance
# call data and set aesthetics
ggplot(emp_data, aes(x = last_evaluation, fill = left)) +
  # density geometry
  geom_density(alpha = 0.5) +
  # facet by boss and employee gender
  facet_wrap(~ department, nrow=2) +
  # aesthetic labels
  labs(x = "Last Evaluation", y = "Density", fill = "Left")
```




```
#### Q1.3
## bar plots for salary
# call data
emp_data %>%
  ## group by two variables
  group_by(department, salary) %>%
  ## count
  count() %>%
  ## group by one variable
  group_by(department) %>%
  ## calculate percentage
  mutate(pct = round(n/sum(n), digits = 3)) %>%
  ## call plot
  ggplot(aes(x = fct_rev(fct_reorder2(department, salary, pct)),
             y = pct, fill = salary)) +
    ## bar geometry
    geom_bar(position = "fill", stat = "identity") +
    ## text geometry
    geom_text(aes(label = paste0(pct*100, "%")), size = 3,
           position = position_stack(vjust = 0.5), color = "white") +
    ## y-axis
    scale_y_continuous(labels = scales::percent_format()) +
    ## aesthetic labels
    labs(x = "Department", y = "Percentage", fill = "Salary") +
    ## flip coordinates
    coord_flip()
```



Task 2: Single Decision Trees

Set the random seed to *301*.
Create a training and test data set such that the training data set consists of *65%* of the full sample and the testing data set consists of the other *35%* of the full sample. Name the training and testing data sets **emp_train** and **emp_test**, respectively. Estimate a single decision tree model using **rpart** on the training data where all other variables except for **emp_id** predict **left**. You can use this formula input inside of **rpart**: **left ~ . - emp_id**. Save the model as **mod_1_train**. Use **fancyRpartPlot** to plot the resulting tree, set **cex = 0.4**, and make sure your plotting window is large before you execute the code. Making the plotting window large before you execute the code will make it easier to read the tree. Apply **summary** to the model.

Question 2.1: What is the prediction on **left** for an employee who has a job satisfaction greater than or equal to 47, a tenure greater than or equal to 4.5, and a last evaluation less than 82? Which variable is the most important predictor?

Response 2.1: *The prediction is that the employee will not leave the company. The most important predictor is job_sat.*

Calculate the *class* predictions on **left** in the testing data set. Save the *class* predictions as **mod_1_test_class**. Use **confusionMatrix()** to evaluate model accuracy.

Question 2.2: What is the sensitivity accuracy? What is the positive predictive value accuracy?

Response 2.2: *Sensitivity accuracy: 0.9853. Positive predictive value accuracy: 0.9719.*

Estimate a single decision tree model using **rpart** on the training data where all other variables except for **emp_id** predict **last_evaluation**. You can use this formula input inside of **rpart**: **last_evaluation ~ .**

- emp_id. Save the model as **mod_2_train**. Use **fancyRpartPlot** to plot the resulting tree, set **cex = 0.4**, and make sure your plotting window is large before you execute the code. Making the plotting window large before you execute the code will make it easier to read the tree. Apply **summary** to the model.

Question 2.3: What is the prediction on **last_evaluation** for an employee who has completed greater than or equal to 2.5 projects and did not leave the company? Which variable is the most important predictor?

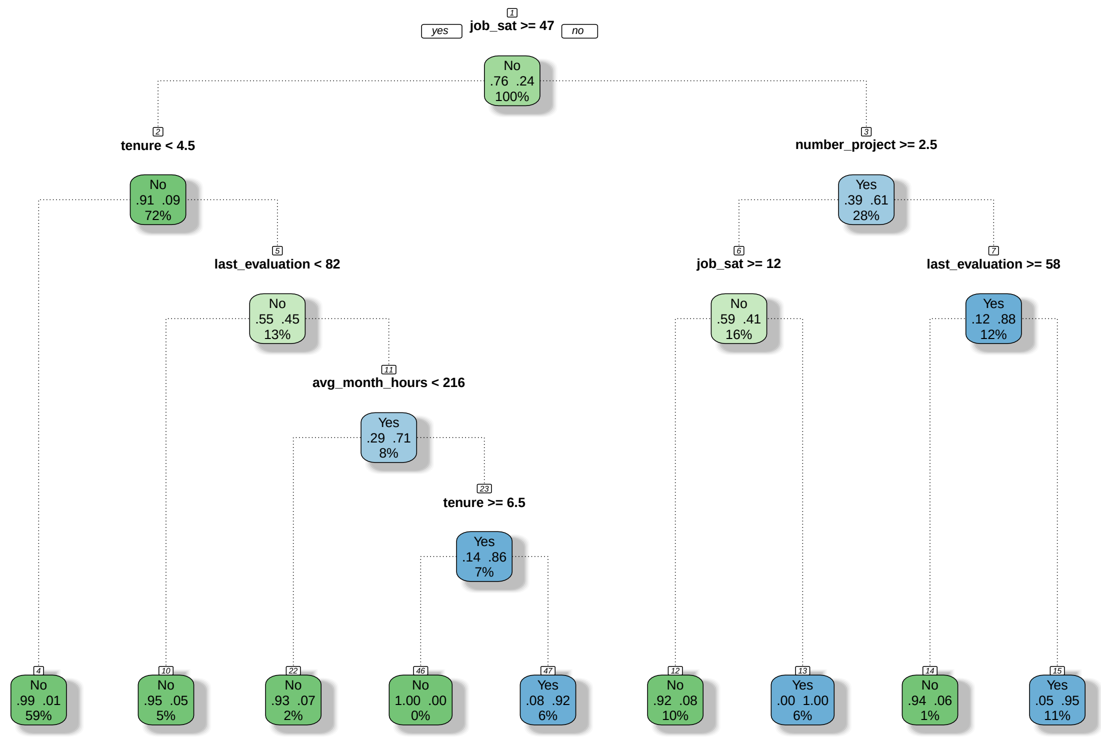
Response 2.3: *Prediction: 72. The most important predictor is number_project.*

Calculate the predictions of **last_evaluation** in the testing data set. Save the predictions as **mod_2_test_pred**. Use **postResample()** to evaluate model accuracy.

Question 2.4: What is the root mean squared error? What is the R-squared?

Response 2.4: *RMSE: 14.7029180. Rsquared: 0.2561656.*

```
#### Q2.1
### training and testing data
## set seed
set.seed(301)

## training data
emp_train <- emp_data %>%
  ## sample a fraction
  sample_frac(0.65)

## testing data
emp_test <- emp_data %>%
  ## find the difference between data
  setdiff(emp_train)

### estimate a single classification tree
## training model
mod_1_train <- rpart(left ~ . - emp_id,
                     data = emp_train)

## plot
fancyRpartPlot(mod_1_train, sub = NULL, type = 1, cex = 0.4)
```




```
## summary
summary(mod_1_train)
```

```
## Call:
## rpart(formula = left ~ . - emp_id, data = emp_train)
##   n= 9749
##
##           CP nsplit rel error    xerror        xstd
## 1 0.24935401      0 1.0000000 1.0000000 0.018113216
## 2 0.18669251      1 0.7506460 0.7506460 0.016293494
## 3 0.07170543      3 0.3772610 0.3772610 0.012160321
## 4 0.05684755      5 0.2338501 0.2338501 0.009751988
## 5 0.03186908      6 0.1770026 0.1774332 0.008554803
## 6 0.01722653      7 0.1451335 0.1455642 0.007779183
## 7 0.01000000      8 0.1279070 0.1283376 0.007319894
##
## Variable importance
##         job_sat  number_project avg_month_hours last_evaluation          tenure
##              35              18              17              16              13
##
## Node number 1: 9749 observations,    complexity param=0.249354
##   predicted class=No    expected loss=0.2381783  P(node) =1
##     class counts:  7427  2322
##    probabilities: 0.762 0.238
##   left son=2 (7012 obs) right son=3 (2737 obs)
##   Primary splits:
```

```
##       job_sat         < 46.5  to the right, improve=1028.3970, (0 missing)
##       number_project  < 2.5   to the right, improve= 636.5861, (0 missing)
##       tenure          < 2.5   to the left,  improve= 260.8862, (0 missing)
##       avg_month_hours < 275.5 to the left,  improve= 250.7124, (0 missing)
##       last_evaluation < 57.5  to the right, improve= 155.9336, (0 missing)
##   Surrogate splits:
##       number_project  < 2.5   to the right, agree=0.792, adj=0.259, (0 split)
##       avg_month_hours < 275.5 to the left,  agree=0.751, adj=0.115, (0 split)
##       last_evaluation < 48.5  to the right, agree=0.740, adj=0.073, (0 split)
##
## Node number 2: 7012 observations,    complexity param=0.07170543
##   predicted class=No   expected loss=0.09469481  P(node) =0.7192533
##     class counts:  6348   664
##    probabilities: 0.905 0.095
##   left son=4 (5706 obs) right son=5 (1306 obs)
##   Primary splits:
##       tenure          < 4.5   to the left,  improve=405.74000, (0 missing)
##       last_evaluation < 82.5  to the left,  improve=141.72170, (0 missing)
##       avg_month_hours < 216.5 to the left,  improve=112.39750, (0 missing)
##       number_project  < 4.5   to the left,  improve= 76.64600, (0 missing)
##       job_sat         < 71.5  to the left,  improve= 53.98825, (0 missing)
##   Surrogate splits:
##       last_evaluation < 99.5  to the left,  agree=0.822, adj=0.044, (0 split)
##       avg_month_hours < 300   to the left,  agree=0.814, adj=0.002, (0 split)
##
## Node number 3: 2737 observations,    complexity param=0.1866925
##   predicted class=Yes  expected loss=0.3942273  P(node) =0.2807467
##     class counts:  1079  1658
##    probabilities: 0.394 0.606
##   left son=6 (1601 obs) right son=7 (1136 obs)
##   Primary splits:
##       number_project  < 2.5   to the right, improve=296.4543, (0 missing)
##       job_sat         < 11.5  to the right, improve=229.7597, (0 missing)
##       tenure          < 4.5   to the right, improve=223.3573, (0 missing)
##       avg_month_hours < 160.5 to the right, improve=111.3998, (0 missing)
##       last_evaluation < 57.5  to the right, improve=108.5676, (0 missing)
##   Surrogate splits:
##       job_sat         < 35.5  to the left,   agree=0.876, adj=0.702, (0 split)
##       avg_month_hours < 161.5 to the right,  agree=0.859, adj=0.659, (0 split)
##       last_evaluation < 57.5  to the right,  agree=0.855, adj=0.651, (0 split)
##       tenure          < 3.5   to the right,  agree=0.839, adj=0.613, (0 split)
##       department      splits as  LLRLLLLLLL, agree=0.586, adj=0.003, (0 split)
##
## Node number 4: 5706 observations
##   predicted class=No   expected loss=0.01331931  P(node) =0.5852908
##     class counts:  5630    76
##    probabilities: 0.987 0.013
##
## Node number 5: 1306 observations,    complexity param=0.07170543
##   predicted class=No   expected loss=0.4502297  P(node) =0.1339625
##     class counts:   718   588
##    probabilities: 0.550 0.450
##   left son=10 (513 obs) right son=11 (793 obs)
##   Primary splits:
```

```
##       last_evaluation < 81.5  to the left,  improve=272.3839, (0 missing)
##       avg_month_hours < 215.5 to the left,  improve=251.2580, (0 missing)
##       tenure          < 6.5   to the right, improve=171.1256, (0 missing)
##       job_sat         < 71.5  to the left,  improve=151.9451, (0 missing)
##       number_project  < 3.5   to the left,  improve=132.7317, (0 missing)
##   Surrogate splits:
##       avg_month_hours < 215.5 to the left,  agree=0.738, adj=0.333, (0 split)
##       number_project  < 3.5   to the left,  agree=0.713, adj=0.269, (0 split)
##       job_sat         < 71.5  to the left,  agree=0.697, adj=0.228, (0 split)
##       tenure          < 6.5   to the right, agree=0.677, adj=0.177, (0 split)
##       work_accident   splits as  RL,        agree=0.641, adj=0.086, (0 split)
##
## Node number 6: 1601 observations,    complexity param=0.1866925
##   predicted class=No   expected loss=0.4097439  P(node) =0.164222
##     class counts:   945   656
##    probabilities: 0.590 0.410
##   left son=12 (1023 obs) right son=13 (578 obs)
##   Primary splits:
##       job_sat         < 11.5  to the right, improve=630.3104, (0 missing)
##       avg_month_hours < 242.5 to the left,  improve=360.5450, (0 missing)
##       number_project  < 5.5   to the left,  improve=337.7908, (0 missing)
##       last_evaluation < 76.5  to the left,  improve=260.6734, (0 missing)
##       tenure          < 3.5   to the left,  improve=109.6993, (0 missing)
##   Surrogate splits:
##       avg_month_hours < 242.5 to the left,  agree=0.858, adj=0.607, (0 split)
##       number_project  < 5.5   to the left,  agree=0.838, adj=0.550, (0 split)
##       last_evaluation < 76.5  to the left,  agree=0.780, adj=0.391, (0 split)
##
## Node number 7: 1136 observations,    complexity param=0.03186908
##   predicted class=Yes  expected loss=0.1179577  P(node) =0.1165248
##     class counts:   134  1002
##    probabilities: 0.118 0.882
##   left son=14 (84 obs) right son=15 (1052 obs)
##   Primary splits:
##       last_evaluation < 57.5  to the right, improve=122.73350, (0 missing)
##       avg_month_hours < 162   to the right, improve=107.02160, (0 missing)
##       job_sat         < 35.5  to the left,  improve= 91.24511, (0 missing)
##       tenure          < 3.5   to the right, improve= 51.84971, (0 missing)
##       work_accident   splits as  RL,        improve=  5.61825, (0 missing)
##   Surrogate splits:
##       avg_month_hours < 162   to the right, agree=0.945, adj=0.250, (0 split)
##       tenure          < 3.5   to the right, agree=0.940, adj=0.190, (0 split)
##       job_sat         < 34.5  to the left,  agree=0.937, adj=0.143, (0 split)
##
## Node number 10: 513 observations
##   predicted class=No   expected loss=0.04873294  P(node) =0.05262078
##     class counts:   488    25
##    probabilities: 0.951 0.049
##
## Node number 11: 793 observations,    complexity param=0.05684755
##   predicted class=Yes  expected loss=0.2900378  P(node) =0.08134168
##     class counts:   230   563
##    probabilities: 0.290 0.710
##   left son=22 (152 obs) right son=23 (641 obs)
```

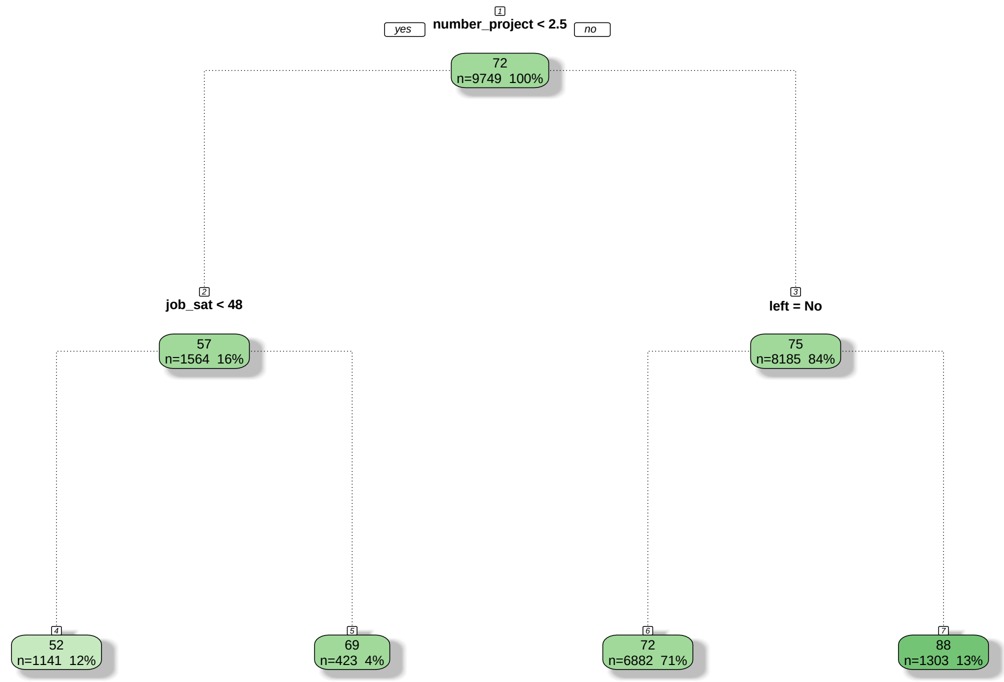
```
##   Primary splits:
##       avg_month_hours < 215.5 to the left,  improve=156.06060, (0 missing)
##       tenure          < 6.5   to the right, improve=134.21740, (0 missing)
##       job_sat         < 71.5  to the left,  improve=115.70150, (0 missing)
##       number_project  < 3.5   to the left,  improve= 76.20733, (0 missing)
##       salary          splits as  RLL,       improve= 23.33242, (0 missing)
##   Surrogate splits:
##       tenure         < 6.5   to the right, agree=0.851, adj=0.224, (0 split)
##       job_sat        < 71.5  to the left,  agree=0.849, adj=0.211, (0 split)
##       number_project < 3.5   to the left,  agree=0.837, adj=0.151, (0 split)
##       salary         splits as  RRL,       agree=0.810, adj=0.007, (0 split)
##
## Node number 12: 1023 observations
##   predicted class=No   expected loss=0.07624633  P(node) =0.1049338
##     class counts:   945    78
##    probabilities: 0.924 0.076
##
## Node number 13: 578 observations
##   predicted class=Yes  expected loss=0  P(node) =0.05928813
##     class counts:     0   578
##    probabilities: 0.000 1.000
##
## Node number 14: 84 observations
##   predicted class=No   expected loss=0.05952381  P(node) =0.008616268
##     class counts:    79     5
##    probabilities: 0.940 0.060
##
## Node number 15: 1052 observations
##   predicted class=Yes  expected loss=0.05228137  P(node) =0.1079085
##     class counts:    55   997
##    probabilities: 0.052 0.948
##
## Node number 22: 152 observations
##   predicted class=No   expected loss=0.06578947  P(node) =0.01559134
##     class counts:   142    10
##    probabilities: 0.934 0.066
##
## Node number 23: 641 observations,    complexity param=0.01722653
##   predicted class=Yes  expected loss=0.1372855  P(node) =0.06575033
##     class counts:    88   553
##    probabilities: 0.137 0.863
##   left son=46 (40 obs) right son=47 (601 obs)
##   Primary splits:
##       tenure                < 6.5   to the right, improve=63.504970, (0 missing)
##       job_sat               < 71    to the left,  improve=58.148360, (0 missing)
##       number_project        < 3.5   to the left,  improve=46.337750, (0 missing)
##       salary                splits as  RRL,       improve=10.913580, (0 missing)
##       promotion_last_5_years splits as  RL,        improve= 8.817611, (0 missing)
##   Surrogate splits:
##       job_sat               < 59    to the left,  agree=0.952, adj=0.225, (0 split)
##       promotion_last_5_years splits as  RL,        agree=0.941, adj=0.050, (0 split)
##
## Node number 46: 40 observations
##   predicted class=No   expected loss=0  P(node) =0.004102985
```

```
##     class counts:    40     0
##    probabilities: 1.000 0.000
##
## Node number 47: 601 observations
##   predicted class=Yes  expected loss=0.07986689  P(node) =0.06164735
##     class counts:    48   553
##    probabilities: 0.080 0.920
```

```r
#### Q2.2
## testing model
# class predictions
mod_1_test_class <- predict(mod_1_train, newdata = emp_test, type = "class")

### evaluate predictions
## confusion matrix
confusionMatrix(mod_1_test_class, emp_test$left)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  3942  114
##        Yes   59 1135
##
##                Accuracy : 0.967
##                  95% CI : (0.9619, 0.9717)
##     No Information Rate : 0.7621
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9077
##
##  Mcnemar's Test P-Value : 4.034e-05
##
##             Sensitivity : 0.9853
##             Specificity : 0.9087
##          Pos Pred Value : 0.9719
##          Neg Pred Value : 0.9506
##              Prevalence : 0.7621
##          Detection Rate : 0.7509
##    Detection Prevalence : 0.7726
##       Balanced Accuracy : 0.9470
##
##        'Positive' Class : No
##
```

```r
#### Q2.3
### estimate a single regression tree
## training model
mod_2_train <- rpart(last_evaluation ~ . - emp_id, data = emp_train)

## plot
fancyRpartPlot(mod_2_train, sub = NULL, type = 1, cex = 0.4)
```




```r
## summary
summary(mod_2_train)
```

```
## Call:
## rpart(formula = last_evaluation ~ . - emp_id, data = emp_train)
##   n= 9749
##
##           CP nsplit rel error    xerror        xstd
## 1 0.14508949      0 1.0000000 1.0002771 0.008833497
## 2 0.09056112      1 0.8549105 0.8552047 0.009636740
## 3 0.02809428      2 0.7643494 0.7647767 0.009522871
## 4 0.01000000      3 0.7362551 0.7367388 0.009051419
##
## Variable importance
##  number_project            left         job_sat avg_month_hours          tenure
##              43              29              18               8               2
##
## Node number 1: 9749 observations,    complexity param=0.1450895
##   mean=71.7611, MSE=294.3394
##   left son=2 (1564 obs) right son=3 (8185 obs)
##   Primary splits:
##       number_project  < 2.5   to the left,  improve=0.145089500, (0 missing)
##       avg_month_hours < 162.5 to the left,  improve=0.101269400, (0 missing)
##       tenure          < 3.5   to the left,  improve=0.060869890, (0 missing)
##       job_sat         < 49.5  to the left,  improve=0.050798960, (0 missing)
##       department      splits as  LLLRRRLLRL, improve=0.001283797, (0 missing)
```

```
##
## Node number 2: 1564 observations,    complexity param=0.02809428
##   mean=56.81138, MSE=180.4395
##   left son=4 (1141 obs) right son=5 (423 obs)
##   Primary splits:
##       job_sat         < 47.5  to the left,  improve=0.28566560, (0 missing)
##       left            splits as  RL,        improve=0.27388040, (0 missing)
##       avg_month_hours < 161.5 to the left,  improve=0.25255650, (0 missing)
##       tenure          < 3.5   to the left,  improve=0.08882063, (0 missing)
##       work_accident   splits as  LR,        improve=0.02513367, (0 missing)
##   Surrogate splits:
##       left                    splits as  RL,        agree=0.900, adj=0.631, (0 split)
##       avg_month_hours         < 161.5 to the left,  agree=0.864, adj=0.496, (0 split)
##       tenure                  < 3.5   to the left,  agree=0.795, adj=0.243, (0 split)
##       work_accident           splits as  LR,        agree=0.735, adj=0.021, (0 split)
##       promotion_last_5_years splits as  LR,        agree=0.730, adj=0.002, (0 split)
##
## Node number 3: 8185 observations,    complexity param=0.09056112
##   mean=74.61772, MSE=265.2377
##   left son=6 (6882 obs) right son=7 (1303 obs)
##   Primary splits:
##       left            splits as  LR,        improve=0.11970060, (0 missing)
##       job_sat         < 11.5  to the right, improve=0.04545700, (0 missing)
##       avg_month_hours < 218.5 to the left,  improve=0.04141328, (0 missing)
##       tenure          < 3.5   to the left,  improve=0.03109584, (0 missing)
##       number_project  < 4.5   to the left,  improve=0.01575521, (0 missing)
##   Surrogate splits:
##       job_sat         < 11.5  to the right, agree=0.911, adj=0.444, (0 split)
##       number_project  < 5.5   to the left,  agree=0.873, adj=0.200, (0 split)
##       avg_month_hours < 275.5 to the left,  agree=0.870, adj=0.185, (0 split)
##
## Node number 4: 1141 observations
##   mean=52.43996, MSE=60.17102
##
## Node number 5: 423 observations
##   mean=68.60284, MSE=314.2678
##
## Node number 6: 6882 observations
##   mean=72.16594, MSE=257.6182
##
## Node number 7: 1303 observations
##   mean=87.56715, MSE=106.0444
```

```r
#### Q2.4
## testing model
# regression predictions
mod_2_test_pred <- predict(mod_2_train, newdata = emp_test)

## three measures with one function
postResample(mod_2_test_pred, emp_test$last_evaluation)
```

```
##       RMSE   Rsquared        MAE
## 14.7029180  0.2561656 11.9109397
```

Task 3: Random Forest

Estimate a random forest model using **randomForest** on the training data where all other variables except for **emp_id** predict **left**. You can use this formula input inside of **randomForest**: **left ~ . - emp_id**. Save the model as **mod_3_train**. Print the variable importance calculations.

Question 3.1: Which variable is most important?

Response 3.1: *The most important variable is job_sat.*

Calculate the *class* predictions on **left** in the testing data set. Save the *class* predictions as **mod_3_test_class**. Use **confusionMatrix()** to evaluate model accuracy.

Question 3.2: What is the specificity accuracy? What is the negative predictive value accuracy?

Response 3.2: *Specificity accuracy: 0.9640. Negative predictive value accuracy: 0.9942.*

Estimate a random forest model using **randomForest** on the training data where all other variables except for **emp_id** predict **last_evaluation**. You can use this formula input inside of **randomForest**: **last_evaluation ~ . - emp_id**. Also add **ntree = 100** inside the **randomForest** function to reduce computation time. Save the model as **mod_4_train**. This model may take 1-3 minutes to run on your computer. Wait patiently. Print the variable importance calculations.

Question 3.3: Which variable is the most important predictor?

Response 3.3: *The most important predictor is avg_month_hours.*

Calculate the predictions of **last_evaluation** in the testing data set. Save the predictions as **mod_4_test_pred**. Use **postResample()** to evaluate model accuracy.

Question 3.4: What is the root mean squared error? What is the R-squared?

Response 3.4: *RMSE: 13.6536229. RSquared: 0.3589448.*

```r
#### Q3.1
### estimate a random forest classification
## training model
mod_3_train <- randomForest(left ~ . - emp_id,
                            data = emp_train)

## variable importance
mod_3_train$importance
```

```
##                        MeanDecreaseGini
## department                    57.708591
## salary                        29.382778
## promotion_last_5_years         3.221672
## work_accident                 22.994183
## tenure                       635.889482
## number_project               638.877775
## avg_month_hours              516.698749
## job_sat                     1192.430997
## last_evaluation              421.400525
```

```r
#### Q3.2
## testing model
# class predictions
mod_3_test_class <- predict(mod_3_train, newdata = emp_test)
```

```r
### evaluate predictions
## confusion matrix
confusionMatrix(mod_3_test_class, emp_test$left)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  3994   46
##        Yes    7 1203
##
##                Accuracy : 0.9899
##                  95% CI : (0.9868, 0.9924)
##     No Information Rate : 0.7621
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9719
##
##  Mcnemar's Test P-Value : 1.792e-07
##
##             Sensitivity : 0.9983
##             Specificity : 0.9632
##          Pos Pred Value : 0.9886
##          Neg Pred Value : 0.9942
##              Prevalence : 0.7621
##          Detection Rate : 0.7608
##    Detection Prevalence : 0.7695
##       Balanced Accuracy : 0.9807
##
##        'Positive' Class : No
##
```

```r
#### Q3.3
### estimate a random forest regression
## training model
mod_4_train <- randomForest(last_evaluation ~ . - emp_id, data = emp_train, ntree = 100)

## variable importance
mod_4_train$importance
```

```
##                         IncNodePurity
## department                  235585.68
## salary                       84441.46
## promotion_last_5_years       16118.58
## work_accident                44103.98
## left                        235294.34
## tenure                      203674.34
## number_project              355057.47
## avg_month_hours             586482.92
## job_sat                     533121.63
```

```r
#### Q3.4
## testing model
# regression predictions
mod_4_test_pred <- predict(mod_4_train, newdata = emp_test)

### evaluate predictions
## three measures with one function
postResample(mod_4_test_pred, emp_test$last_evaluation)
```

```
##       RMSE   Rsquared        MAE
## 13.6536229  0.3589448 10.7898561
```

Task 4: Gradient Boosted Machine

Create a new variable inside of **emp_train** and **emp_test** named **left_num** just like in the analytical lecture. Estimate a gradient boosted machine using **gbm** on the training data where all other variables except for **emp_id** and **left** predict **left_num**. You can use this formula input inside of **gbm**: **left_num ~ . - emp_id - left**. Note that you are predicting **left_num** without **left** in the model. Use *500* trees and the *bernoulli* distribution. Save the model as **mod_5_train**. Apply **summary** to the model.

Question 4.1: Which variable is most important?

Response 4.1: *The most important variable is job_sat.*

Calculate the *class* predictions on **left_num** in the testing data set. Save the *class* predictions as **mod_5_test_class**. Use **confusionMatrix()** to evaluate model accuracy.

Question 4.2: What is the specificity accuracy? What is the negative predictive value accuracy?

Response 4.2: *Specificity : 0.8855. Neg Pred Value : 0.9279.*

Estimate a gradient boosted machine using **gbm** on the training data where all other variables except for **emp_id** and **left** predict **last_evaluation**. You can use this formula input inside of **gbm**: **last_evaluation ~ . - emp_id - left**. Note that you are predicting **left_num** without **left** in the model. Use *500* trees. Save the model as **mod_6_train**. Apply **summary** to the model.

Question 4.3: Which variable is the most important predictor?

Response 4.3: *The most important predictor is number_project.*

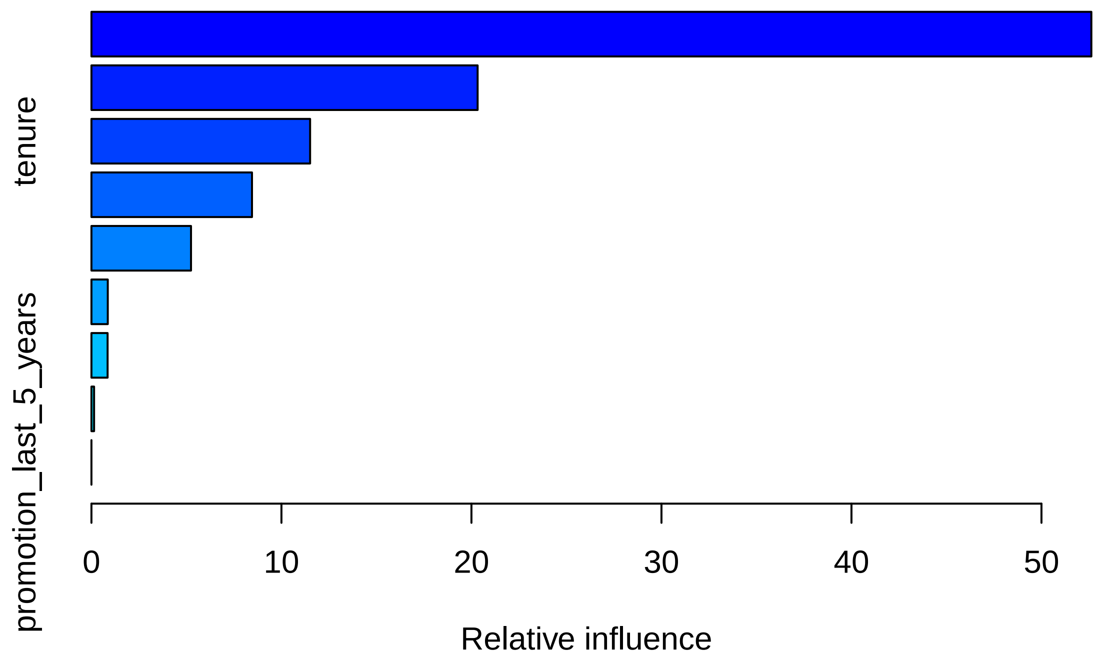
Calculate the predictions of **last_evaluation** in the testing data set. Save the predictions as **mod_6_test_pred**. Use **postResample()** to evaluate model accuracy.

Question 4.4: What is the root mean squared error? What is the R-squared?

Response 4.4: *RMSE: 14.6095396. Rsquared: 0.2657993.*

```r
#### Q4.1
### adjust categorical outcome
## training data
emp_train <- emp_train %>%
  ## create new variable
  mutate(left_num = as.numeric(left) - 1)

## testing data
emp_test <- emp_test %>%
    ## create new variable
  mutate(left_num = as.numeric(left) - 1)
```

```r
### estimate a gradient boosted machine classification
## training model
mod_5_train <- gbm(left_num ~ . - emp_id - left,
                   # specify data and distribution
                   data = emp_train, distribution = "bernoulli",
                   # specify number of trees
                   n.trees = 500)

## summary
summary(mod_5_train)
```



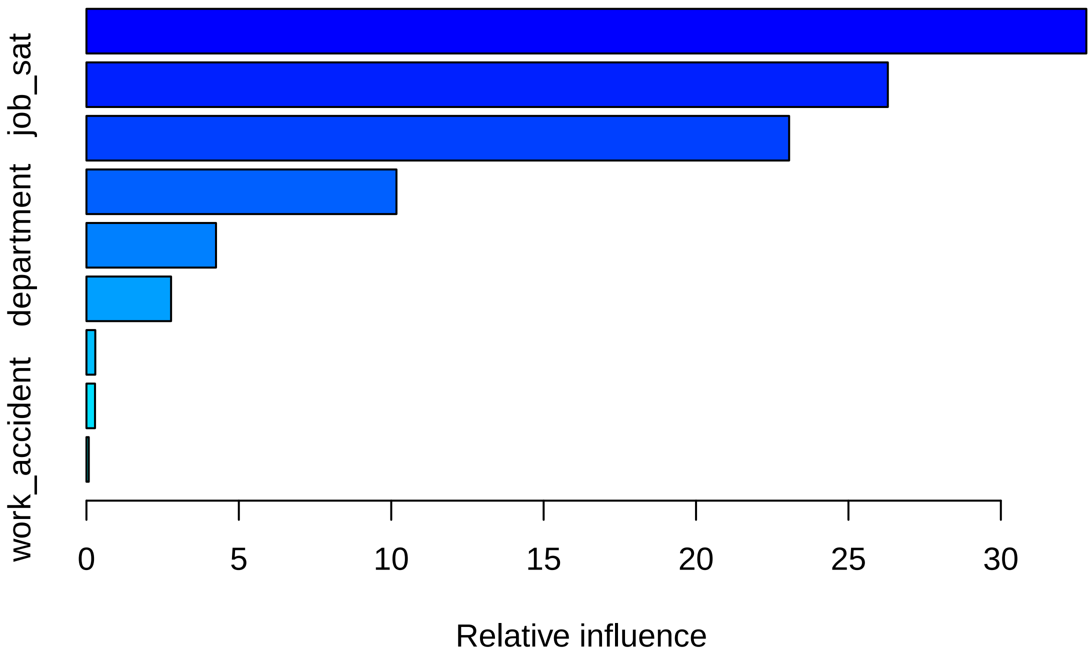

```
##                                           var    rel.inf
## job_sat                               job_sat 52.6256336
## number_project                 number_project 20.3255563
## tenure                                 tenure 11.5079490
## last_evaluation               last_evaluation  8.4497954
## avg_month_hours               avg_month_hours  5.2410505
## salary                                 salary  0.8605830
## work_accident                   work_accident  0.8507389
## department                         department  0.1386932
## promotion_last_5_years promotion_last_5_years  0.0000000
```

```r
#### Q4.2
## testing model
```

```r
# probability predictions
mod_5_test_pred <- predict(mod_5_train, newdata = emp_test, n.trees = 500,
                           type = "response")

# class predictions
mod_5_test_class <- as.factor(if_else(mod_5_test_pred < 0.5, "No", "Yes"))

### evaluate predictions
## confusion matrix
confusionMatrix(mod_5_test_class, emp_test$left)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  3915  143
##        Yes   86 1106
##
##                Accuracy : 0.9564
##                  95% CI : (0.9505, 0.9617)
##     No Information Rate : 0.7621
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8778
##
##  Mcnemar's Test P-Value : 0.0002151
##
##             Sensitivity : 0.9785
##             Specificity : 0.8855
##          Pos Pred Value : 0.9648
##          Neg Pred Value : 0.9279
##              Prevalence : 0.7621
##          Detection Rate : 0.7457
##    Detection Prevalence : 0.7730
##       Balanced Accuracy : 0.9320
##
##        'Positive' Class : No
##
```

```r
#### Q4.3
### estimate a single regression tree
## training model
mod_6_train <- gbm(last_evaluation ~ . - emp_id - left,
                   # specify data and distribution
                   data = emp_train, distribution = "gaussian",
                   # specify number of trees
                   n.trees = 500)

## summary
summary(mod_6_train)
```




```
##                                             var     rel.inf
## number_project                   number_project 32.80520475
## job_sat                                 job_sat 26.29296071
## avg_month_hours                 avg_month_hours 23.05354683
## tenure                                   tenure 10.17159930
## department                           department  4.25090686
## left_num                               left_num  2.77701522
## promotion_last_5_years promotion_last_5_years  0.29030856
## salary                                   salary  0.28187723
## work_accident                     work_accident  0.07658053
```

```
#### Q4.4
## testing model
# regression predictions
mod_6_test_pred <- predict(mod_6_train, newdata = emp_test, n.trees = 500)

### evaluate predictions
## three measures with one function
postResample(mod_6_test_pred, emp_test$last_evaluation)
```

```
##       RMSE   Rsquared        MAE
## 14.6095396  0.2657993 11.8388071
```

Task 5: Prediction Plots

Create a new *tibble* data object named **last_eval_preds**. Name the first column **last_evaluation** and set it equal to **emp_test$last_evaluation**. Name the second column **dt_preds** and set it equal to **mod_2_test_pred**. Name the third column **rf_preds** and set it equal to **mod_4_test_pred**. Name the fourth column **gbm_preds** and set it equal to **mod_6_test_pred**.

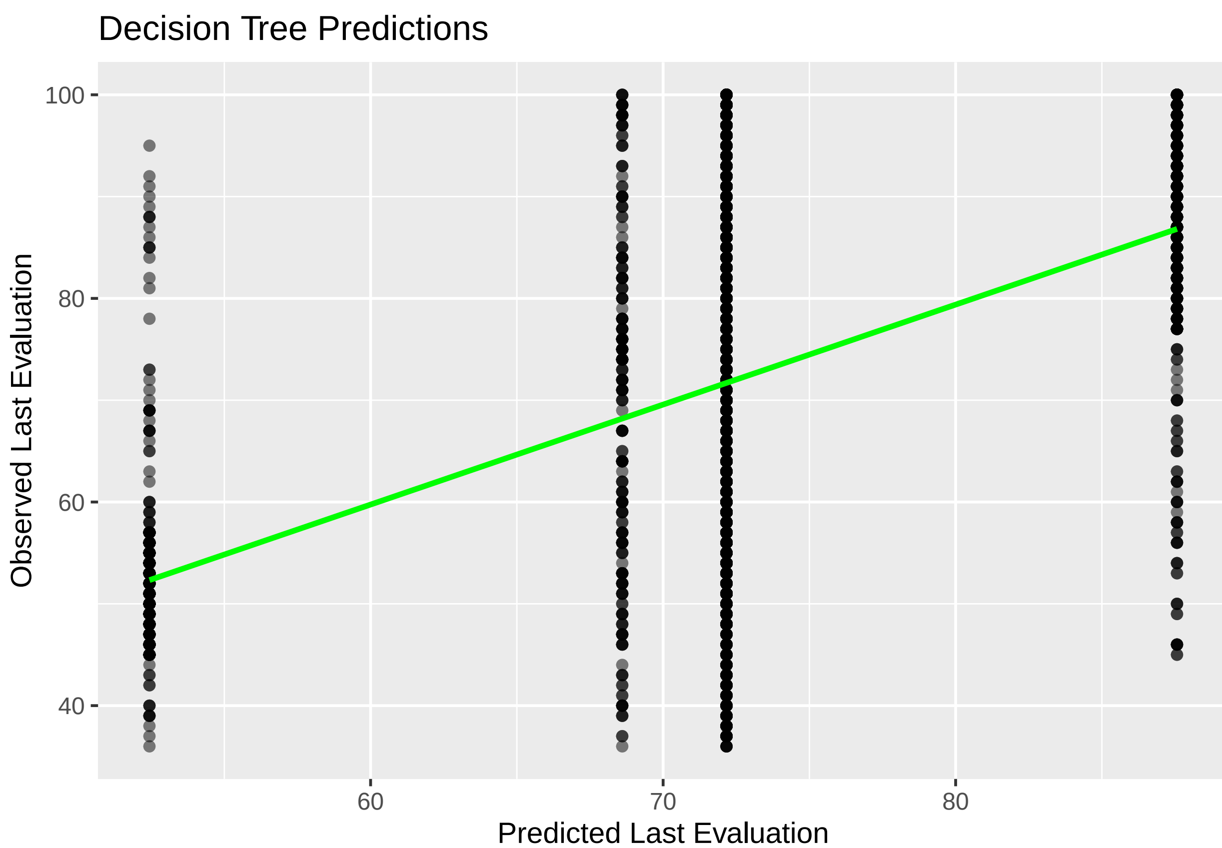
Produce three scatterplots using **ggplot()**. Set the data to **last_eval_preds**. Map **dt_preds**, **rf_preds**, and **gbm_preds**, respectively, to the x-axis in the three separate scatterplots. Map **last_evaluation** to the y-axis. Add the point geometry. Add the smooth geometry with **method** set to **lm**, **se** set to **FALSE**, and **color** set to **green**. Add appropriate axes labels and plot title.

Question 5.1: From which two algorithms do the predictions look the most similar?

Response 5.1: *Random forest and Gradient Boosted Machine.*

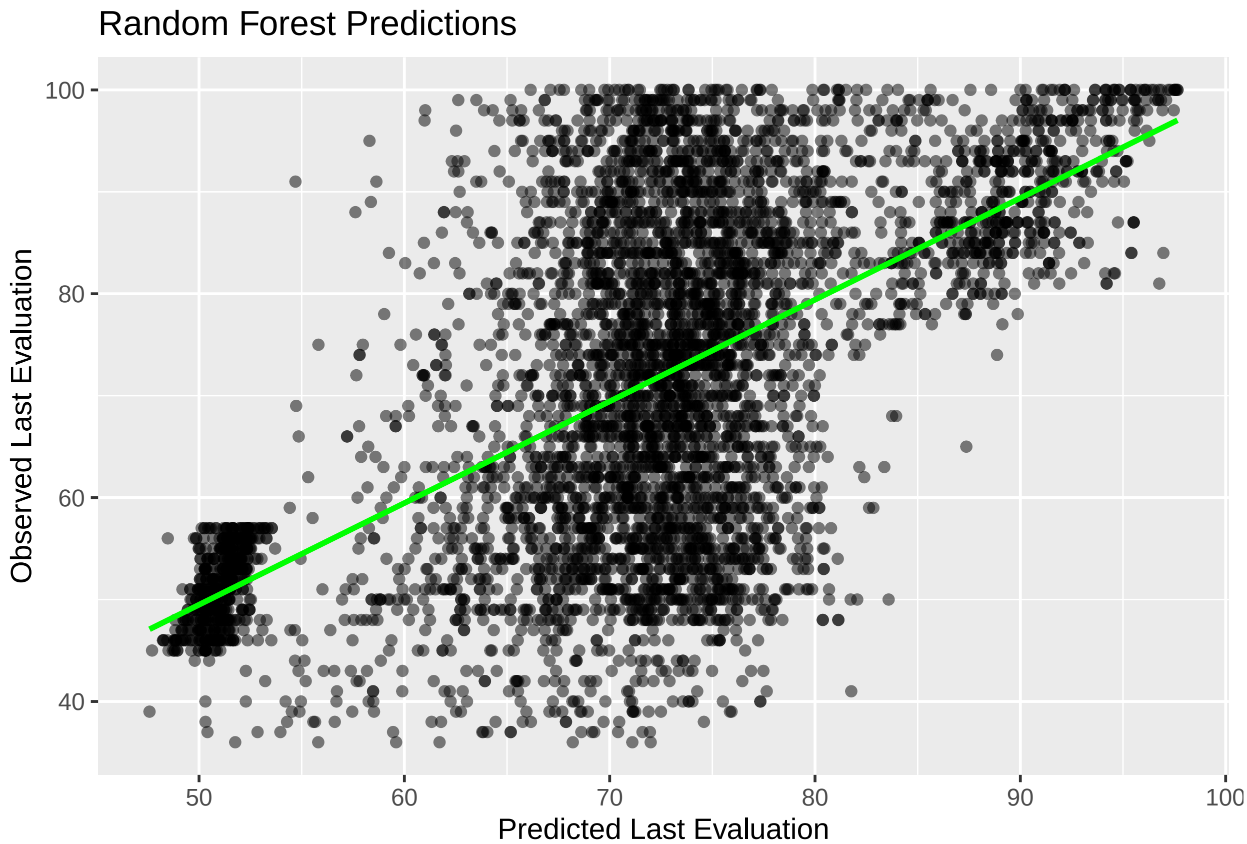
```r
#### Q5.1
### create data object with observed and predicted values
## name data
last_eval_preds <- tibble(
  ## observed values of last_eval
  last_evaluation = emp_test$last_evaluation,
  ## decision tree predicted values
  dt_preds = mod_2_test_pred,
  ## random forest predicted values
  rf_preds = mod_4_test_pred,
  ## gradient boosted machine predicted values
  gbm_preds = mod_6_test_pred,
)

### plot decision tree predictions
## call data and mapping
ggplot(last_eval_preds, aes(x = dt_preds, y = last_evaluation)) +
  ## point geometry
  geom_point(alpha = 0.5) +
  ## smooth geometry
  geom_smooth(method = "lm", se = FALSE, color = "green") +
  ## labs
  labs(x = "Predicted Last Evaluation", y = "Observed Last Evaluation") +
  ## title
  ggtitle("Decision Tree Predictions")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



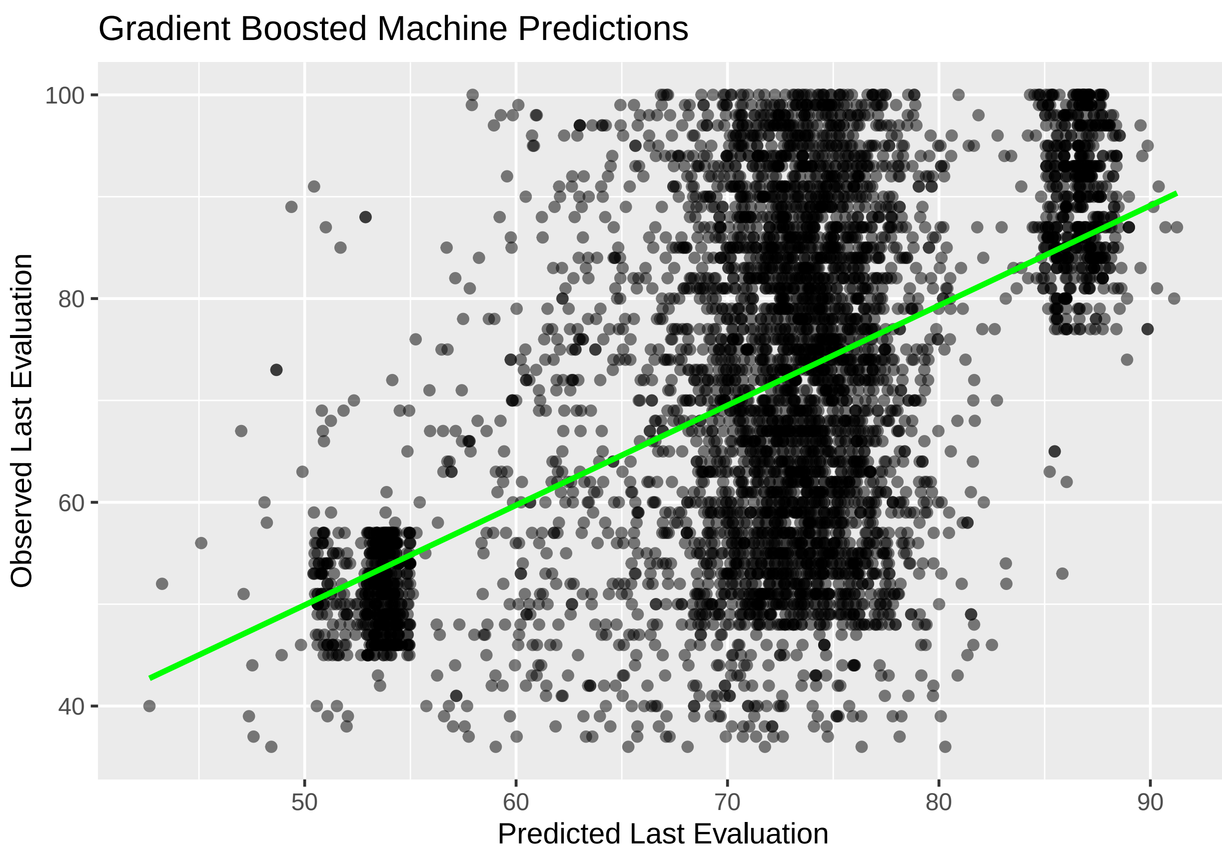

```
### plot random forest predictions
## call data and mapping
ggplot(last_eval_preds, aes(x = rf_preds, y = last_evaluation)) +
  ## point geometry
  geom_point(alpha = 0.5) +
  ## smooth geometry
  geom_smooth(method = "lm", se = FALSE, color = "green") +
  ## labs
  labs(x = "Predicted Last Evaluation", y = "Observed Last Evaluation") +
  ## title
  ggtitle("Random Forest Predictions")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
### plot gbm predictions
## call data and mapping
ggplot(last_eval_preds, aes(x = gbm_preds, y = last_evaluation)) +
  ## point geometry
  geom_point(alpha = 0.5) +
  ## smooth geometry
  geom_smooth(method = "lm", se = FALSE, color = "green") +
  ## labs
  labs(x = "Predicted Last Evaluation", y = "Observed Last Evaluation") +
  ## title
  ggtitle("Gradient Boosted Machine Predictions")
```

```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



Task 6: Supervised Learning Meta-Engine

Use the `caret` work flow to predict **last_evaluation** from all predictors except **emp_id** and **left_num** in the **emp_train** data using **lm** (i.e., ordinary least-squares regression), **rpart**, **ranger** (i.e., an alternate random forest algorithm), and **gbm**. Note, you will remove both **emp_id** and **left_num** as predictors of **last_evaluation**. Set-up a training control object with *3-fold cross-validation repeated 3 times* and name it **train_control**. Name the ordinary least-squares regression model: **mod_reg_lm**. Name the single tree regression model: **mod_reg_rpart**. You can ignore any warning messages from running **rpart**. Name the random forest regression model: **mod_reg_rf**. For the random forest model, make sure to use **ranger** as the method and set **num.trees = 100**. Your computer will take 1-3 minutes to run the random forest model. Wait patiently. Name the gradient boosted regression machine: **mod_reg_gbm**. Calculate the predictions for each model. Name the predictions for the ordinary least-squares regression model: **mod_reg_lm_test**. You can keep the **type = "raw"** argument in the **predict()** function. Name the predictions for the single decision tree regression model: **mod_reg_rpart_test**. Name the predictions for the random forest regression model: **mod_reg_rf_test**. Name the predictions for the gradient boosted regression machine: **mod_reg_gbm_test**. Evaluate each model's accuracy with **postResample()**.

Question 6.1: What is the R-squared value of the ordinary least-squares regression model? Which model performed the best?

Response 6.1: *Rsquared: 0.1924587. Random forest model performed the best.*

```
#### Q6.1
### train models
## train controls
train_control <- trainControl(
```

```r
  # repeated cross-validation
  method = "repeatedcv",
  # 3-fold cross-validation
  number = 3,
  # cross-validation repeated 3 times
  repeats = 3
  )

## model ordinary least-squares regression
mod_reg_lm <- train(
  # model equation
  last_evaluation ~ . - emp_id - left_num,
  # specify data, method, and controls
  data = emp_train, method = "lm", trControl = train_control)

## model decision tree
mod_reg_rpart <- train(
  # model equation
  last_evaluation ~ . - emp_id - left_num,
  # specify data, method, and controls
  data = emp_train, method = "rpart", trControl = train_control)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```r
## model random forest
mod_reg_rf <- train(
  # model equation
  last_evaluation ~ . - emp_id - left_num,
  # specify data, method, and controls
  data = emp_train, method = "ranger", num.trees = 100, trControl = train_control)

## model gradient boosted machine
mod_reg_gbm <- train(
  # model equation
  last_evaluation ~ . - emp_id - left_num,
  # specify data and method
  data = emp_train, method = "gbm",
  # specify verbosity and controls
  verbose = FALSE, trControl = train_control)

### test models
## ordinary least-squares regression
mod_reg_lm_test <- predict(mod_reg_lm, newdata = emp_test, type = "raw")

# post resample
postResample(mod_reg_lm_test, emp_test$last_evaluation)
```

```
##       RMSE   Rsquared        MAE
## 15.3208803  0.1924587 12.6252348
```

```r
## decision tree
mod_reg_rpart_test <- predict(mod_reg_rpart, newdata = emp_test, type = "raw")

# post resample
postResample(mod_reg_rpart_test, emp_test$last_evaluation)
```

```
##       RMSE   Rsquared        MAE
## 14.9890920  0.2269692 12.2378553
```

```r
## random forest
mod_reg_rf_test <- predict(mod_reg_rf, newdata = emp_test, type = "raw")

# post resample
postResample(mod_reg_rf_test, emp_test$last_evaluation)
```

```
##       RMSE   Rsquared        MAE
## 13.7400299  0.3522209 10.6331642
```

```r
## gradient boosted machine
mod_reg_gbm_test <- predict(mod_reg_gbm, newdata = emp_test, type = "raw")

# post resample
postResample(mod_reg_gbm_test, emp_test$last_evaluation)
```

```
##       RMSE   Rsquared        MAE
## 14.3981300  0.2868043 11.6599562
```