Exercise 3

Team members:

- Tashfeen Ahmed
- Adrian Alarcon
- Yvan Kammelu
- Zhicheng Zhong

Data Preparation

Data Source

To do this exercise, we will use the dataset produced in this file. This was the base of the previous exercise.

Aggregating at quarterly level

First, to start this exercise, the dataset has been aggregated at quarter level

Variable generation

Then, we generate variables at quarter level, including `new_applications`, `abandoned_applications`, `patents_issued`, `in_process_applications` an `current_art_unit`.

After generating those variables, variables about art unit were generated, considering `number_of_woman` and `number_people_art_unit`.

```
art_unit_info = applications %>%
  group_by(filing_year_quarter, examiner_art_unit) %>%
  summarise(
    num_people_in_art_unit = n_distinct(examiner_id),
    num_women_in_art_unit = sum(gender.x == "female", na.rm = TRUE),
    .groups = 'drop'
  )


panel_data =  panel_data %>%
  left_join(art_unit_info, by = c("filing_year_quarter", "current_art_unit" =
"examiner_art_unit"))
```

Generating target variables

By comparing the last 5 quarters, we generated target variable: `separation_indicator`, which means that the examiner left the company

```
panel_data = panel_data %>%
  group_by(examiner_id) %>%
  mutate(
    # Get a list of the last five quarters of activity for each examiner
    last_five_quarters = list(tail(sort(unique(filing_year_quarter)), 5))
  ) %>%
  ungroup() %>%
  mutate(
    # Check if the current quarter is in the last five quarters of activity
    separation_indicator = if_else(map_lgl(filing_year_quarter, ~ .x %in% las
t_five_quarters[[1]]), 1, 0)
  )
```

Aggregate data at examiner level

What the model will predict is if an examiner is going to leave the company. To achieve this, we need to aggregate the data at examiner level. Additionally, performance variables were created including the average of the applications issued per quarter, the average of the abandoned applications, and the totals of each variables.

```
examiner_data = panel_data %>%
                  group_by(examiner_id) %>%
                  summarise(
                    #mean_new_applications_qt = mean(num_new_applications),
                    #total_new_applications_qt = sum(num_new_applications),
                    mean_abandoned_applications_qt = mean(num_abandoned_appli
cations),
                    #total_abandoned_applications = sum(num_abandoned_applica
tions),
                    mean_issued_patents_qt = mean(num_issued_patents),
                    #total_issued_patents = sum(num_issued_patents),
                    mean_in_process_applications_qt = mean(num_in_process_app
lications),
                    separation_indicator = max(separation_indicator)
                  )

examiner_data_info = applications %>% group_by(examiner_id) %>%
                        summarise(
                          gender = max(gender.x),
                          race = max(race.x),
                          mean_tenure = mean(tenure_days.x)
                          )

examiner_data_final = merge(examiner_data, examiner_data_info, by = 'examiner
_id', all = TRUE)
```

Since there are some data without `examiner_id`, we filtered those values where that field showed NULL values

```
examiner_data_final = examiner_data_final %>%
                        filter(!is.na(!!sym('examiner_id')))
```

Handling categorical variables

Since there are two categorical variables: `race` and `gender`, it was necessary to handle them. To do this, one hot encoding was performed.

Because of NULL values in gender, for those rows, the dummy vars would be 0.

Finally, we drop `examiner_id` since it is an id and it should not enter into the predictive modeling

```
final_data = predict(dummyVars('~.', data = examiner_data_final), newdata = e
xaminer_data_final)

final_data = data.frame(final_data)



final_data = final_data %>%
                mutate(genderfemale = ifelse(is.na(genderfemale),0,genderfema
le),
                       gendermale = ifelse(is.na(gendermale),0,gendermale))

final_data = final_data %>% select(-c('examiner_id'))
#final_data$separation_indicator = as.factor(final_data$separation_indicator)
data_subset = head(final_data, 5)
data_subset = data_subset[,1:5]
kable(data_subset, format = "latex", booktabs = TRUE, longtable = FALSE, capt
ion = "Final data") %>%
  kable_styling(latex_options = "hold_position")
```

Modeling

For modeling purposes, we decided to experiment with two algorithms:

- Logistic Regression
- Random Forest

To evaluate metrics, we splited the dataset into two subdatasets: one for training purposes, containing 70% of the data, and one for test purposes, that is going to be used to validate metrics.

```
set.seed(123)
splitIndex = createDataPartition(final_data$separation_indicator, p = 0.7, li
st = FALSE)

# Create training set
train_data = final_data[splitIndex, ]
```

```
# Create testing set
test_data = final_data[-splitIndex, ]
```

Logistic Regression

Here it is showed the results of the logistic regression

```
logistic_model = glm(separation_indicator ~ ., data = train_data, family = "b
inomial", maxit = 1000)

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(logistic_model)

##
## Call:
## glm(formula = separation_indicator ~ ., family = "binomial",
##     data = train_data, maxit = 1000)
##
## Coefficients: (1 not defined because of singularities)
##                                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     -3.364e+00  1.916e-01 -17.559  < 2e-16 ***
## mean_abandoned_applications_qt   6.442e-01  3.985e-02  16.164  < 2e-16 ***
## mean_issued_patents_qt          -1.221e-02  1.462e-02  -0.835  0.40380
## mean_in_process_applications_qt  2.040e-01  2.564e-02   7.957 1.76e-15 ***
## genderfemale                    -1.777e-01  1.435e-01  -1.238  0.21553
## gendermale                      -7.836e-02  1.275e-01  -0.615  0.53867
## raceAsian                        6.167e-01  9.771e-02   6.312 2.76e-10 ***
## raceblack                        6.088e-01  2.154e-01   2.826  0.00471 **
## raceHispanic                     3.780e-01  2.028e-01   1.864  0.06233 .
## raceother                        1.262e+01  3.043e+02   0.041  0.96692
## racewhite                               NA         NA      NA       NA
## mean_tenure                      5.667e-04  3.294e-05  17.201  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5029.4  on 3953  degrees of freedom
## Residual deviance: 3523.0  on 3943  degrees of freedom
## AIC: 3545
##
## Number of Fisher Scoring iterations: 12

logistic_predictions_probas = predict(logistic_model, test_data, type = 'resp
onse')

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful case
s
```

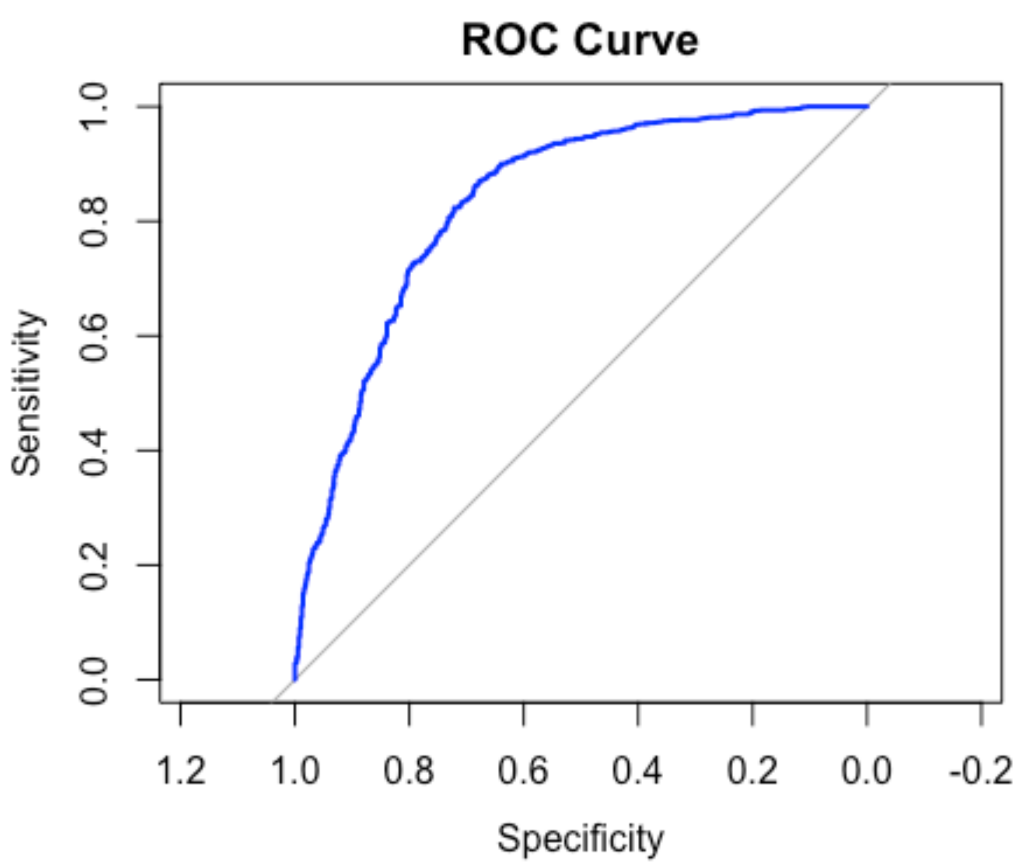
```
predictions = ifelse(logistic_predictions_probas>0.5,1,0)
```

Confusion matrix for logistic regression:

```
conf_matrix = confusionMatrix(table(predictions, test_data$separation_indicat
or))
conf_matrix_data = as.matrix(conf_matrix$table)
print(conf_matrix_data)

##
## predictions    0    1
##           0  365  112
##           1  207 1010
```

ROC Curve and AUC for logistic regression

```
roc_curve = roc(test_data$separation_indicator, logistic_predictions_probas)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

roc_plot = plot(roc_curve, main = "ROC Curve", col = "blue")
```




```
print(auc(roc_curve))
```

```
## Area under the curve: 0.8341
```

```
roc_plot
```

```
##
## Call:
## roc.default(response = test_data$separation_indicator, predictor = logisti
c_predictions_probas)
##
## Data: logistic_predictions_probas in 572 controls (test_data$separation_in
dicator 0) < 1122 cases (test_data$separation_indicator 1).
## Area under the curve: 0.8341
```

RandomForest

Here it is showed the results of the Random Forest

```
rf_model = randomForest(separation_indicator ~., data = train_data, class.f =
TRUE)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values.  Are you sure you want to do regression?
```

```
predictions_rf = predict(rf_model, test_data, type = 'response')
```

Confusion matrix for Random Forest

```
conf_matrix_rf = confusionMatrix(table(ifelse(predictions_rf>0.5,1,0), test_d
ata$separation_indicator))
conf_matrix_data_rf = as.matrix(conf_matrix_rf$table)
print(conf_matrix_data_rf)
```

```
##
##         0    1
##   0   466   82
##   1   106 1040
```

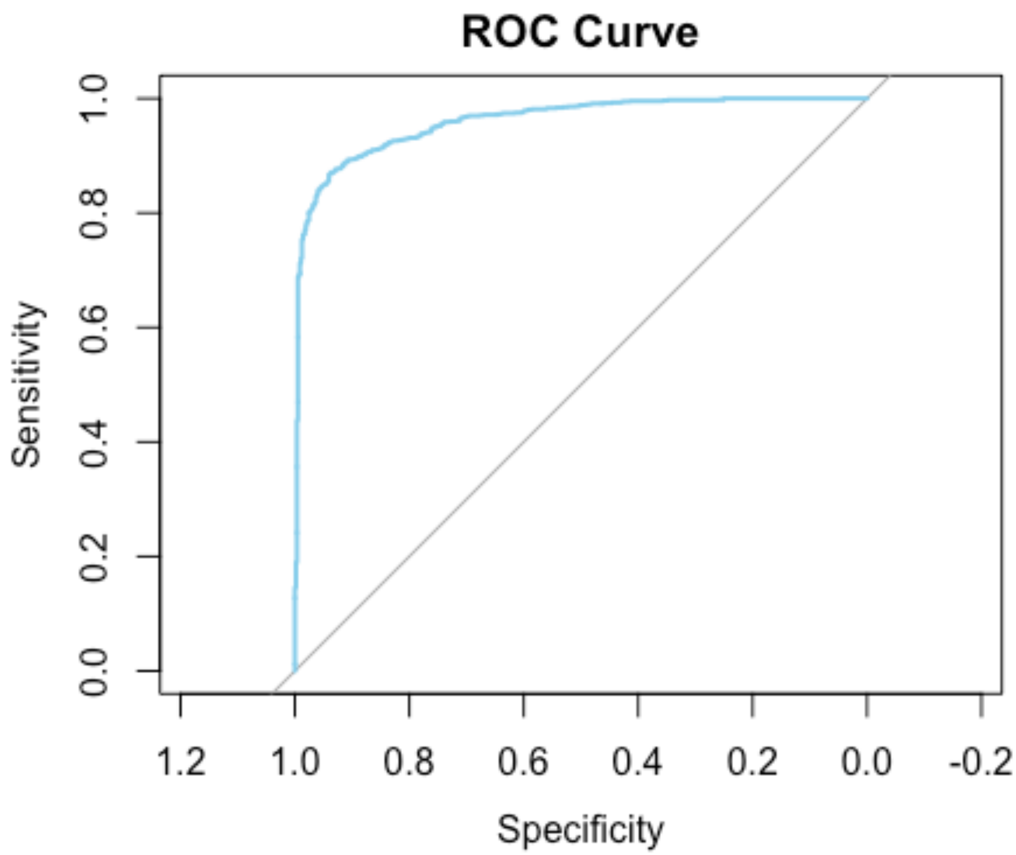
ROC Curve and AUC for Random Forest

```
roc_curve_rf = roc(test_data$separation_indicator, predictions_rf)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
roc_plot_rf = plot(roc_curve_rf, main = "ROC Curve", col = "skyblue")
```

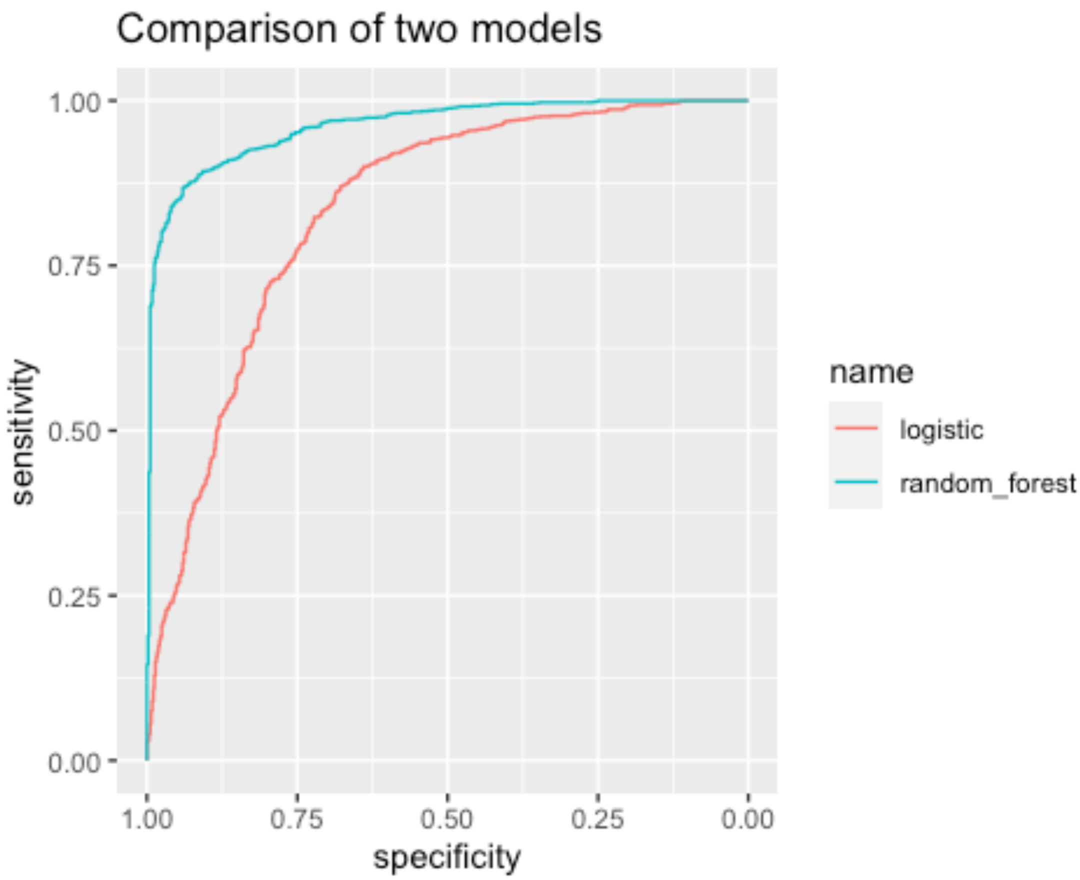



```
print(auc(roc_curve_rf))

## Area under the curve: 0.9612

roc_plot_rf

##
## Call:
## roc.default(response = test_data$separation_indicator, predictor = predict
ions_rf)
##
## Data: predictions_rf in 572 controls (test_data$separation_indicator 0) <
1122 cases (test_data$separation_indicator 1).
## Area under the curve: 0.9612
```

Comparing models

Here a comparison in terms of ROC curve

```
ggroc(list(logistic = roc_curve, random_forest = roc_curve_rf))+
  labs(title = "Comparison of two models")
```




```
print('Logistic Regression:')

## [1] "Logistic Regression:"

print(auc(roc_curve))

## Area under the curve: 0.8341

print('Random Forest:')

## [1] "Random Forest:"

print(auc(roc_curve_rf))

## Area under the curve: 0.9612
```