Talent Analytics Exercise 3 Compiled

2024-01-29

The first few sections of this markdown document reiterates the processing steps highlighted in Exercise 2 with some improvments to the code. For details on Exercise 3, skip to page 8 for the regression analysis.

Initialisation of libraries and dataset

Import Libraries

```
knitr::opts_chunk$set(tidy.opts=list(width.cutoff=60), format='latex', echo=TRUE)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(lubridate)
library(arrow)
```

```
##
## Attaching package: 'arrow'
##
## The following object is masked from 'package:lubridate':
##
##     duration
##
## The following object is masked from 'package:utils':
##
##     timestamp
```

Import Dataset

```
data_path <- "/Users/chien/Library/CloudStorage/OneDrive-McGillUniversity/5c_Talent_Analytics/ta-assign
applications <- arrow::read_feather(data_path)
```

Processing of dataset to include gender and race for examiners

Adding gender.y to dataset based on surnames library

```r
library(gender)
examiner_names <- applications %>%
        distinct(examiner_name_first)

examiner_names_gender <- examiner_names %>%
        do(results = gender(.$examiner_name_first, method = "ssa")) %>%
        unnest(cols = c(results), keep_empty = TRUE) %>%
        select(
                examiner_name_first = name,
                gender,
                proportion_female)

# remove extra colums from the gender table
examiner_names_gender <- examiner_names_gender %>%
        select(examiner_name_first, gender)

# joining gender back to the dataset
applications <- applications %>%
        left_join(examiner_names_gender, by = "examiner_name_first")

# cleaning up
rm(examiner_names)
rm(examiner_names_gender)
gc()
```

```
##            used  (Mb) gc trigger  (Mb) limit (Mb)  max used  (Mb)
## Ncells  4444105 237.4    7901380 422.0         NA   4463595 238.4
## Vcells 59481087 453.9  124343714 948.7      16384 103926350 792.9
```

Adding race.y to dataset using surnames library

```r
library(wru)

examiner_surnames <- applications %>%
        select(surname = examiner_name_last) %>%
        distinct()

examiner_race <- predict_race(voter.file = examiner_surnames, surname.only = T) %>%
        as_tibble()
```

```
## Warning: Unknown or uninitialised column: `state`.

## Proceeding with last name predictions...

## i All local files already up-to-date!

## 701 (18.4%) individuals' last names were not matched.
```

```r
examiner_race <- examiner_race %>%
        mutate(max_race_p = pmax(pred.asi, pred.bla, pred.his, pred.oth, pred.whi)) %>%
        mutate(race = case_when(
                max_race_p == pred.asi ~ "Asian",
                max_race_p == pred.bla ~ "black",
                max_race_p == pred.his ~ "Hispanic",
                max_race_p == pred.oth ~ "other",
                max_race_p == pred.whi ~ "white",
                TRUE ~ NA_character_
        ))

# removing extra columns
examiner_race <- examiner_race %>%
        select(surname,race)

applications <- applications %>%
        left_join(examiner_race, by = c("examiner_name_last" = "surname"))

rm(examiner_race)
rm(examiner_surnames)
gc()
```

```
##            used  (Mb) gc trigger  (Mb) limit (Mb)  max used  (Mb)
## Ncells  4628415 247.2    7901380 422.0         NA   6796160 363.0
## Vcells 61822469 471.7  124343714 948.7      16384 124029298 946.3
```

Adding dates-related data to calculate tenure days

```
library(lubridate) # to work with dates

examiner_dates <- applications %>%
        select(examiner_id, filing_date, appl_status_date)

examiner_dates <- examiner_dates %>%
        mutate(start_date = ymd(filing_date), end_date = as_date(dmy_hms(appl_status_date)))

examiner_dates <- examiner_dates %>%
        group_by(examiner_id) %>%
        summarise(
                earliest_date = min(start_date, na.rm = TRUE),
                latest_date = max(end_date, na.rm = TRUE),
                tenure_days = interval(earliest_date, latest_date) %/% days(1)
        ) %>%
        filter(year(latest_date)<2018)

applications <- applications %>%
        left_join(examiner_dates, by = "examiner_id")

rm(examiner_dates)
gc()
```

```
##            used  (Mb) gc trigger   (Mb) limit (Mb)  max used  (Mb)
## Ncells  4637225 247.7    7901380  422.0         NA   7901380 422.0
## Vcells 67898261 518.1  149292456 1139.1      16384 124343712 948.7
```

Creating panel data

Cleaning noisy data

```
# Checking for number of unique values in categorical data

cat_columns <- c("disposal_type", "race.x", "gender.x", "race.y", "gender.y")
result_list <- lapply(cat_columns, function(col_name) {
    counts <- table(applications[[col_name]], useNA = "ifany")
    data.frame(Column = col_name, Value = names(counts), Count = as.integer(counts))
})
print(result_list)
```

```
## [[1]]
##          Column Value   Count
## 1 disposal_type   ABN  601411
## 2 disposal_type   ISS 1087306
## 3 disposal_type  PEND  329760
##
## [[2]]
##   Column    Value   Count
## 1 race.x    Asian  591644
## 2 race.x    black   89559
## 3 race.x Hispanic   58856
## 4 race.x    other    1891
## 5 race.x    white 1276527
##
## [[3]]
##     Column  Value   Count
## 1 gender.x female  571227
## 2 gender.x   male 1143391
## 3 gender.x   <NA>  303859
##
## [[4]]
##   Column    Value   Count
## 1 race.y    Asian  591644
## 2 race.y    black   89559
## 3 race.y Hispanic   58856
## 4 race.y    other    1891
## 5 race.y    white 1276527
##
## [[5]]
##     Column  Value   Count
## 1 gender.y female  571227
## 2 gender.y   male 1143391
## 3 gender.y   <NA>  303859
```

Removing NA values in gender.x

The column gender.x and gender.y has 303859 NA values (constituting about 15% of the dataset). These values will be dropped to facilitate analysis.

```
applications <- applications[!is.na(applications$gender.x), ]
counts <- table(applications$gender.x, useNA = "ifany")
print(counts)
```

```
##
##  female    male
##  571227 1143391
```

Cleaning data types

```
# Convert filing_date to Date format and create a quarter variable
applications$filing_date <- as.Date(applications$filing_date)
applications$quarter <- paste0(year(applications$filing_date), "/", quarter(applications$filing_date))

# Aggregate applications by quarter and examiner

## count the number of distinct applications and aggregate by each examiner
applications <- applications %>%
        group_by(quarter, examiner_id) %>%
        mutate(new_applications = n_distinct(application_number)) %>%
        ungroup()

applications <- applications %>%
        group_by(quarter, examiner_id) %>%
        mutate(ISSUED_applications = sum(disposal_type == "ISS" & !duplicated(application_number)))

applications <- applications %>%
        group_by(quarter, examiner_id) %>%
        mutate(abn_applications = sum(disposal_type == "ABN" & !duplicated(application_number)))

applications <- applications %>%
        group_by(quarter, examiner_id) %>%
        mutate(PEN_applications = sum(disposal_type == "PEND" & !duplicated(application_number)))

applications <- applications %>%
        group_by(quarter,examiner_art_unit) %>%
        mutate(examiner_art_unit_num =  n_distinct(examiner_id))%>%
        ungroup()

applications <- applications %>%
        group_by(quarter, examiner_art_unit) %>%
        mutate(women_in_art_unit  = sum(gender.y == "female" & !duplicated(examiner_id)))

applications <- applications %>%
        group_by(quarter, examiner_art_unit) %>%
        mutate(Asian_in_art_unit  = sum(race.y == "Asian" & !duplicated(examiner_id)))

applications <- applications %>%
        group_by(quarter, examiner_art_unit) %>%
        mutate(Black_in_art_unit  = sum(race.y == "black" & !duplicated(examiner_id)))

applications <- applications %>%
        group_by(quarter, examiner_art_unit) %>%
        mutate(Hispanic_in_art_unit  = sum(race.y == "Hispanic" & !duplicated(examiner_id)))

applications <- applications %>%
        group_by(quarter, examiner_art_unit) %>%
        mutate(Other_in_art_unit  = sum(race.y == "other" & !duplicated(examiner_id)))
```

```
applications <- applications %>%
        group_by(quarter, examiner_art_unit) %>%
        mutate(White_in_art_unit  = sum(race.y == "white" & !duplicated(examiner_id)))
```

Sorting applications by examiner and quarter

```
# sort by examiner_id and quarter
applications <- applications %>%
        arrange(examiner_id, quarter)

applications_selected <- applications %>%
        select(
                application_number,
                examiner_id,
                examiner_name_first,
                examiner_name_middle,
                examiner_name_last,
                tc,
                quarter,
                new_applications,
                ISSUED_applications,
                abn_applications,
                PEN_applications,
                examiner_art_unit,
                women_in_art_unit,
                Asian_in_art_unit,
                Black_in_art_unit,
                Other_in_art_unit,
                White_in_art_unit,
                ends_with(".x")  # Select columns that end with '_x'
        ) %>%
        rename_with(~ str_remove(., ".x"), ends_with(".x"))  # Remove the '_x' suffix
```

Introducing separation and AU move indicator

```
# find the latest time quarter for each examiner
overall_max_quarter <- "2017/1"

# filter dataset to exclude the latest quarter
applications_selected <- applications_selected %>%
        filter(quarter <= overall_max_quarter)

# add the separation indicator variable
applications_selected <- applications_selected %>%
        group_by(examiner_id) %>%
        mutate(max_quarter_examiner = max(quarter)) %>%
        ungroup() %>%
        mutate(separation_indicator = if_else(max_quarter_examiner < overall_max_quarter, 1, 0))

# AU move indicator
applications_selected <- applications_selected %>%
  group_by(examiner_id) %>%
  mutate(au_move_indicator = if_else(examiner_art_unit != lag(examiner_art_unit), 1, 0)) %>%
  ungroup()

# Fill NA for the au_move_indicator
applications_selected <- applications_selected %>%
  mutate(au_move_indicator = if_else(is.na(au_move_indicator), 0, au_move_indicator))

# Drop columns that are not needed
applications_selected <- applications_selected %>%
  select(-c(max_quarter_examiner, earliest_date, latest_date, tc))

# Rename applications_selected as df
df <- applications_selected
```

Aggregating panel data (quarterly)

```
# individual level data
indi_attributes <- df %>%
  select(gender, race, examiner_id) %>%
  distinct(examiner_id, .keep_all = TRUE)

panel_df <- df %>%
  group_by(examiner_id, quarter) %>%
  summarize(
    new_applications = mean(new_applications, na.rm = TRUE),
    ISSUED_applications = mean(ISSUED_applications, na.rm = TRUE),
    total_abn_applications = mean(abn_applications, na.rm = TRUE),
    total_PEN_applications = mean(PEN_applications, na.rm = TRUE),
    tenure_days = mean(tenure_days, na.rm = TRUE),
    women_in_art_unit = mean(women_in_art_unit, na.rm = TRUE),
    Asian_in_art_unit = mean(Asian_in_art_unit, na.rm = TRUE),
    Black_in_art_unit = mean(Black_in_art_unit, na.rm = TRUE),
    Other_in_art_unit = mean(Other_in_art_unit, na.rm = TRUE),
    White_in_art_unit = mean(White_in_art_unit, na.rm = TRUE),
    separation_indicator = mean(separation_indicator, na.rm = TRUE),
    au_move_indicator = sum(au_move_indicator, na.rm = TRUE)
  )
```

```
## `summarise()` has grouped output by 'examiner_id'. You can override using the
## `.groups` argument.
```

```
panel_df <- panel_df %>%
  left_join(indi_attributes, by = "examiner_id")

panel_df <- panel_df %>%
  mutate(
    examiner_id = as.integer(examiner_id),
    quarter = as.character(quarter),  # or you could separate into year and quarter
    tenure_days = as.numeric(tenure_days),  # Assuming you keep the .x column
    separation_indicator = as.integer(separation_indicator),
    au_move_indicator = as.integer(au_move_indicator),
    gender = as.factor(gender),
    race = as.factor(race)
  )
```

```
# to find maximum quarter
max(panel_df$quarter)
```

```
## [1] "2017/1"
```

```
# for those with separation indicator = 1, make their last quarter = 1 and the rest 0.
panel_df <- panel_df %>%
  group_by(examiner_id) %>%
  mutate(
    last_observation = ifelse(row_number() == n(), 1, 0), # Identify the last observation
    separation_indicator = ifelse(last_observation == 1 & any(separation_indicator == 1), 1, 0)
  ) %>%
  select(-last_observation) %>% # Remove the helper column
  ungroup()

# change the au_move_indicator - if > 1 then 1
panel_df$au_move_indicator[panel_df$au_move_indicator > 1] <- 1
```

Prediction Model (Exercise 3)

Our logistic regression model takes the panel data that is aggregated by examiner id and quarter to predict turnover.

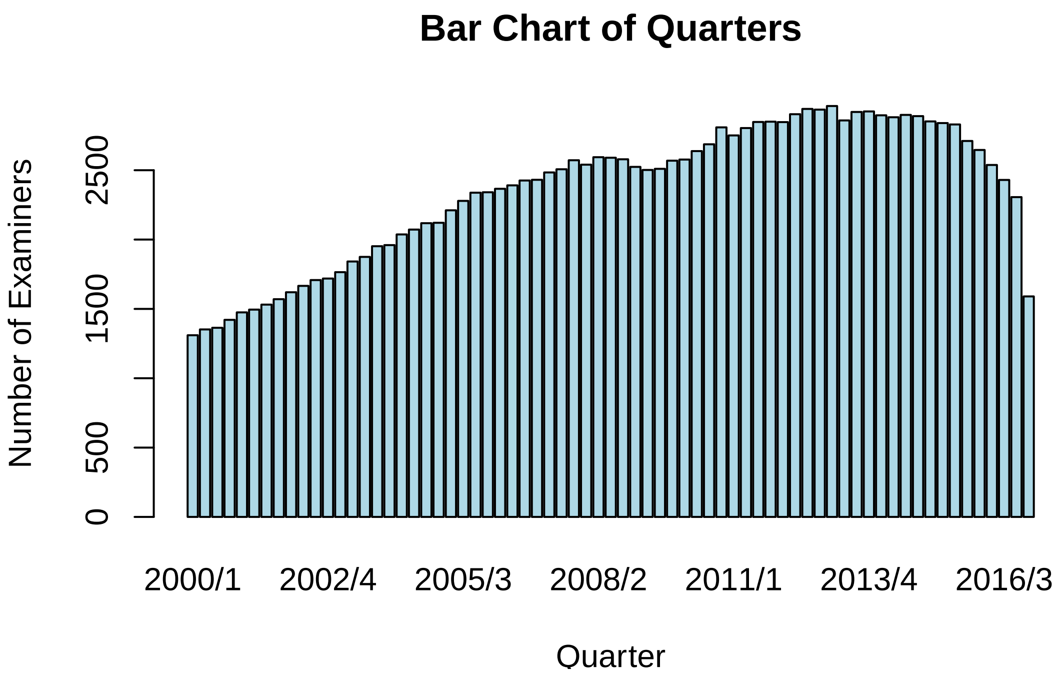
Exploratory Data Analysis of panel data

The panel data consists of the following columns.

```
print(colnames(panel_df))
```

```
##  [1] "examiner_id"            "quarter"                "new_applications"
##  [4] "ISSUED_applications"    "total_abn_applications" "total_PEN_applications"
##  [7] "tenure_days"            "women_in_art_unit"      "Asian_in_art_unit"
## [10] "Black_in_art_unit"      "Other_in_art_unit"      "White_in_art_unit"
## [13] "separation_indicator"   "au_move_indicator"      "gender"
## [16] "race"
```

Some exploratory data analysis was done on the data.



Due to the large number of categories in column "quarter", we dropped the column "quarter" to facilitate analysis. Additionally, "examiner_id" was dropped too. As we are predicting turnover, the column "au_move_indicator" was dropped too.

Training and Testing Logistic Regression Model

The model is trained with 80% of the data, holding 20% of the data as the test set.The model takes in features such as number of new applications, number of issued applications, total applications abandoned and tenure days to predict turnover.

```r
library(caret)
library(ROCR)
library(gtsummary)

# Create a subset of the panel_df without some columns
data <- subset(panel_df, select = -c(examiner_id, quarter, au_move_indicator))
data$gender <- as.factor(data$gender)
data$race <- as.factor(data$race)

# Split data into training and testing sets
set.seed(123) # for reproducibility
trainingIndex <- createDataPartition(data$separation_indicator, p = .8, list = FALSE)
trainingData <- data[trainingIndex,]
testingData <- data[-trainingIndex,]

# Train logistic regression model and print results
model <- glm(separation_indicator ~ ., data = trainingData, family = binomial())
model_summary <- tbl_regression(model)
model_summary
```

Characteristic	log(OR)	95% CI	p-value
new_applications	-0.18	-0.20, -0.17	<0.001
ISSUED_applications	-0.47	-0.52, -0.42	<0.001

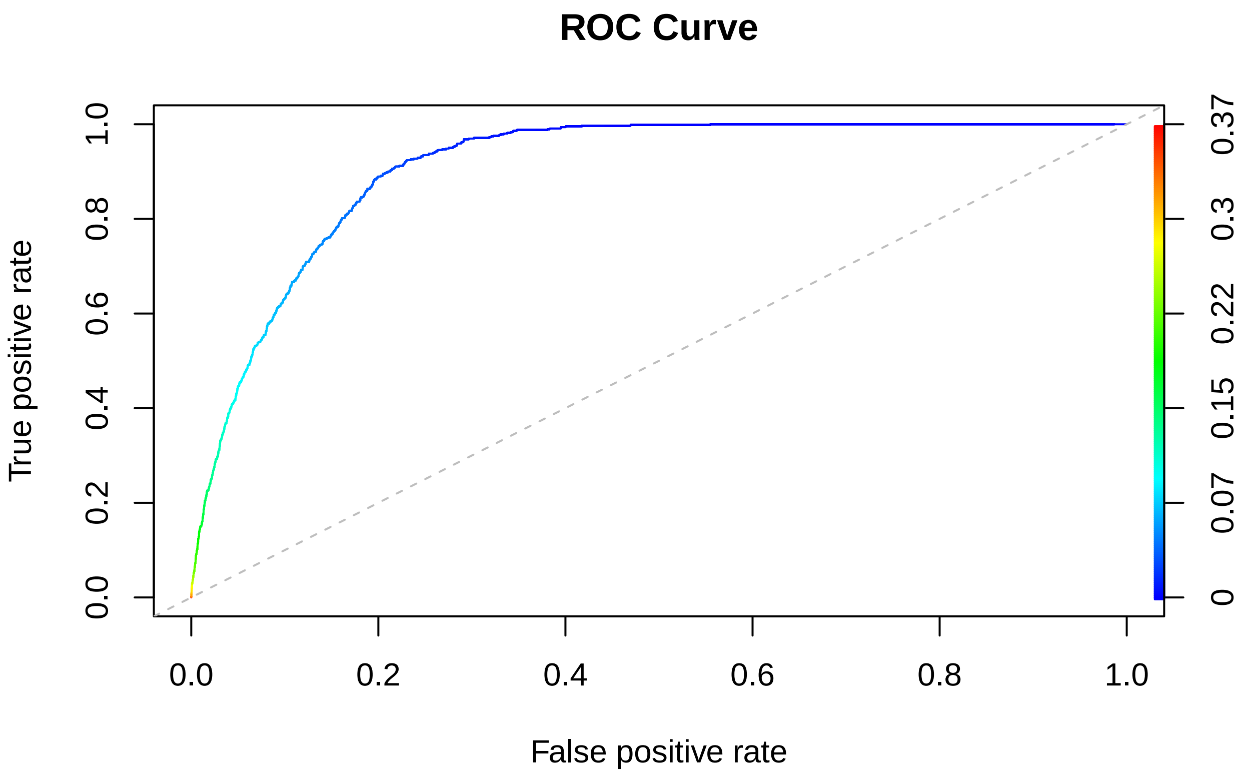
Characteristic	**log(OR)**	**95% CI**	**p-value**
total_abn_applications	-0.83	-0.90, -0.77	<0.001
total_PEN_applications			
tenure_days	0.00	0.00, 0.00	<0.001
women_in_art_unit	0.02	0.00, 0.04	0.031
Asian_in_art_unit	-0.02	-0.04, -0.01	0.009
Black_in_art_unit	-0.04	-0.10, 0.02	0.2
Other_in_art_unit	0.29	-0.13, 0.67	0.2
White_in_art_unit	0.01	0.00, 0.02	0.3
gender			
male	—	—	
female	-0.09	-0.18, 0.00	0.062
race			
white	—	—	
Asian	-0.17	-0.27, -0.07	0.001
black	0.02	-0.22, 0.24	0.9
Hispanic	-0.23	-0.43, -0.03	0.028
other	-0.05	-3.0, 1.6	>0.9

Predict on testing set

```r
predictions <- predict(model, testingData, type = 'response')
actualClasses <- testingData$separation_indicator

# Convert probabilities to predicted scores for ROC analysis
predictionScores <- prediction(predictions, actualClasses)
```

Plotting ROC Curve



```r
# Calculate AUC
auc <- performance(predictionScores, "auc")
aucValue <- auc@y.values[[1]]
cat("AUC:", aucValue, "\n")
```

```
## AUC: 0.908498
```

Results and Discussion

Of the different features, number of new_applications, issued applications, total abandoned applications and tenure days are highly significant in predicting turnover with a small p-value (<0.001). The negative log(OR) value suggests that higher values of this predictor are associated with lower odds of the outcome occurring (i.e. lower probability of turnover). The other features are less significant in predicting turnover rates.

Recommendations

This model, with an AUC value close to 1, is very good at identifying which employees might leave the company. It suggests that if an examiner processes fewer applications in a quarter, they might be thinking about leaving. This drop in applications could mean the examiner is less motivated and not working as much. So, the company can use the number of applications an examiner handles as a sign to see if they might quit. If they notice an examiner with fewer applications, they can act early to try and keep them, especially if keeping an employee is cheaper than hiring a new one.