Warner Internship Code

Yiting Zhang

2022-08-24

Project Summary

Study time-tracking trends and patterns across U.S Non-exempt population at Warner Bros. Discovery. We investigate the different variables that could potentially contribute to employee non-compliance including timesheet submission after deadline (overdue).

Steps

1. Collect/Pull data from Workday system
2. Data validation and data cleaning: create variables overdue based on Monday deadline
3. Exploratory Analysis
4. Machine learning: KNN, Random Forest, and Mixed Effect Models

Cool Libraries

- library(fuzzyjoin)
  - for inexact matching
- library(stringdist)
  - similar to fuzzy match
  - stringdist(a=variable, b=variable that contains the string to search, mathod= "lv)
- library(stringr)
  - Similar to fuzzy match, use str_detect to detect the presence of a pattern match in a stirng
  - str_detect(fruit, "a")
- library(kableExtra)
  - Build complex tables and manipulate differnet table styles
  - kbl() %>% kable_classsic(full_width=F, html_font="Cambria")
- library(rcompanion)
  - pairwiseNominalIndependence(table)
  - See if different combinations of values in the table would generate significance (small P-value)
  - Look at the adjusted value

Skeleton Code

Concepts

- Mapping data
  - eg. if your data is missing a lot of values in a specific variable, you can create a mapping data that contains the value from other rows and match to fill in the blanks/NAs. You can match the Worker ID for example.
  - Use left_join() to join the mapping data with the original data

Data manipulation

- If combining two conditions in the dataset[] does not work, try to use dataset[which(dataset$var == “some condition”),]
- str_remove() to remove unwanted portion of the string
  - For example, removing “Terminiated” from Worker Variable
  - time_tracking[str_remove(time_tracking$Worker, “Terminated”),]
- Date
  - as.Date() only do date
  - as.POSIXt() or strptime() can include time into date
  - Manual Date by hand: as.Date(paste(“2022”,8,26,sep=“-”))
- Retrieve a part of the string
  - substr(variable,beginning, ending)
  - substr(time_tracking$Period, 1,10)
- Substitute a part of the string
  - gsub(“things you want to replace”,“things you want to overwrite the original”, dataset$variable)
- Remove duplicate
  - dataset <- dataset[!duplicated(dataset$var),]
- Remove something in a parentheses
  - overdue_data <- overdue_data %>% mutate(Period=str_remove(Period, “\s*\([^\\]+\)“))
- check which column/variable has NA and how many are there
  - colSums(is.na(dataset))

Creating Variables

- dataset$new_variable <- ifelse(condition 1, what happen next, if condition didn't satisfied what happen next)
  - can include an ifelse() in and ifelse()
  - Sometimes different functions in different packages do not work well together, so it will be better to use the functions under the same package
  - time_tracking <- time_tracking %>% mutate(manager_approval = ifelse())
- Creating groups as a new variable
  - e.g.

```
# time_tracking$hire_range <- ifelse(as.Date(paste("2022",8,26,sep="-"))- time_tracking$hire_date <= 36
```

Creating tables and data frames

- Change header names in a data frame
  - For one column: names(dataset)[column number] <- "new name"
  - For all columns: names(dataset) <- c("new name 1", new name 2")
- Delete columns/rows in a dataset
  - dataset <- dataset[delete rows here, delete columns here]
  - eg. state_data <- state_data[,-c(2:5)]
- Remove NAs in a column/variable
  - dataset[is.na(dataset$car)==FALSE,]
- Join data together
  - inner_join()
  - merge(dataset1,dataset2, by="Var")
- Add frequency to the dataset
  - If need a frequncy value in order to plot, then we can manually add a freq column where everyhing is 1
  - dataset$freq <- 1
- Create a table and convert it to dataset in wide form +e.g.

```
# od_worker <- table(overdue_data$Worker, overdue_data$overdue)
#
# od_worker_tab <- od_worker %>%                # Order table with dplyr
#
#   as.data.frame() %>%
#
#   arrange(desc(Freq))
#
# wider_od_worker_tab <- pivot_wider(od_worker_tab, names_from = Var2, values_from = Freq )
```

- barplot with both x and y axis

```
# ggplot(state_tab_sort, aes(x=reorder(State,`Past deadline head counts`), y=`Past deadline head counts
```

ggplot

- barplot with percentage label on each bar and no y axis value

```
# ggplot(data=demo_data, aes(x=hire_range)) +geom_bar(aes(y = (..count..)/sum(..count..)),fill="blue2")
#
#   geom_text(aes(y = ((..count..)/sum(..count..)), label = scales::percent((..count..)/sum(..count..),
```

- barplot with count label on each bar and no y axis value

```
# ggplot(data=demo_data, aes(x=`Production Type`))+
#
# geom_bar(fill="skyblue") +
#
# geom_text(aes(label=..count..),  stat="count",vjust=-0.25)+ ggtitle("Distribution of Non-exempt Emplo
```

- barplot to change the order of the x axis value

```
# combined_tab_od$Approval_Freq <- factor(combined_tab_od$Approval_Freq, levels=c("<10", "10-40", "41-1
#
# ggplot(data=combined_tab_od, aes(x=Approval_Freq)) +
#
# geom_bar(aes(y = (..count..)/sum(..count..)), fill="blue2") +
#
#   geom_text(aes(y = ((..count..)/sum(..count..)), label = scales::percent((..count..)/sum(..count..),
#
#   scale_y_continuous(labels = scales::percent)+  ggtitle("Overdue By Frequency")+ xlab("Submission St
```

Statistical test

When analyzing data, it is useful to check whether there is actually significance between two variables.

A function that I created is:

```
# stat_test <- function(table){
#   proportion_table <- rprop(table, digits=1, percent=TRUE, total=FALSE, n=FALSE)  #find percentage of
#   prop_test <- prop.test(table)  #find p-value
#   chisq_test <- chisq.tes(table, simulate.p.value = TRUE)$stdres  #find the standard residuals
#   pairwise_test <- pairwiseNominalIndependence(table)
#   return(list(table, proportion_table,prop_test, chisq_test, pairwise_test))
# }
```

A way to look at standard residuals from chi-square test:

```
      |  On Time | Past deadline
```

——— | ——— | ———— HBO | -8 | 8 Otter | -1.5 | 1.5 Turner | 0.1 | -0.1 Warner M | 6.4 | -6.4

Positive number means it is likely to happen, and the bigger the number it is, most likely that it is going to happen. For example, HBO is most likely to submit timesheet pass deadline where Warner Media is least likely to submit past deadline.

Modeling

For clustering models (Rf, KNN, Gradient Boosting), we need to use aggregate data (So instead of saying "yes" "no" under the variable, we need to aggregate the freq of the variable under the same employee).

e.g.

Worker	Overdue
1	Yes
2	No
3	NO
4	Yes

And so on...

And change this to

Worker	Overdue_count
1	3 #3 if this worker was overdued 3 times in the entire dataset
2	6
3	8
4	1

- Preprocessing
  - for clustering models, everything has to be in factor or numeric forms
  - Can be either ordinal factor (there's a rank, or potentially from highest to lowest) to just normal factor
  - Normal Factor: as.factor()
  - Ordinal Factor: factor(dataset$var, order= TRUE, level=c("biggest","2nd","3rd","smallest"))
  - Use str() to check the variable type
- Imputation
  - After the data is cleaned, we need to impute all the NA values, so either replace them with median or mean or delete the column/row
  - e.g.

```
# #create a function that calculates the mode
# calc_mode <- function(x){
#   # List the distinct / unique values
#
#   distinct_values <- unique(x)
#   # Count the occurrence of each distinct value
#   distinct_tabulate <- tabulate(match(x, distinct_values))
#   # Return the value with the highest occurrence
#   distinct_values[which.max(distinct_tabulate)]
# }


# use this function to replace the NAs with mode
# aggregate_norm$Manager_level[is.na(aggregate_norm$Manager_level)] <- calc_mode(aggregate_norm$Manager
```

- Standardization/ Normalization
  - After imputing NAs, we also need to normalize the data
  - Categorical variable can also be normalized
  - e.g.

```
#method 1: use preProcess() and predict()
# aggregate_test <- preProcess(as.data.frame(aggregate_data), method="knnImpute")
#
# aggregate_norm <- predict(aggregate_test, newdata= aggregate_data)
#
# head(aggregate_norm)



#method 2: use function
min_max_norm <- function(x) {
    (x - min(x)) / (max(x) - min(x))
  }

# apply Min-Max normalization to first four columns in iris dataset
# aggregate_norm <- as.data.frame(lapply(aggregate_data[1:4], min_max_norm))
```

- Random Forest
  - Run the function
  - Tune the tree to find the best mtry or # of trees
  - Find variable of Importance by plotting VarImpPLot()
  - Find mse or rmse by rf.model$mseorsqrt(rf.model$mse)
  - Find the # of trees with minimum mse
  - partially dependence plot to look at the first few important variables
  - e.g.

```
#tune the tree

# rf.model_tuned <- tuneRF(
#                x=aggregate_rf[,-1], #define predictor variables
#                y=aggregate_rf$Overdue_count, #define response variable
#                ntreeTry=500,
#                mtryStart=5,
#                stepFactor=1.5,
#                improve=0.01,
#                trace=FALSE #don't show real-time progress
#                )


# or

# control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
# set.seed(123)
# tunegrid <- expand.grid(.mtry=c(1:15))
# rf_gridsearch <- train(Overdue_count~., data=aggregate_rf, method="rf", metric="RMSE", tuneGrid=tuneg
# print(rf_gridsearch)
# plot(rf_gridsearch)
# rf_gridsearch$results
# rf.model <- randomForest(Overdue_count ~ ., data=aggregate_rf, mtry=6, proximity=TRUE)
# rf.model
#
# #find the # of trees with min mse
# plot(rf.model)
# which.min(rf.model$mse)
#
# #find RMSE of best model
# sqrt(rf.model$mse[which.min(rf.model$mse)])
#
# #variable of importance
# rf.model$importance
#
# #manager delinquency cout has the highest importance,followed by Monday and Tuesday
# varImpPlot(rf.model)
```

```
#partially dependence plot

# rf.model <- randomForest(Overdue_count ~ ., data=aggregate_rf, proximity=TRUE)
# pdp::partial(rf.model, pred.var = "State",  plot = TRUE)
# rf.model %>% partial(pred.var = "State") %>% plotPartial(smooth = TRUE, lwd = 2,las=3, ylab = express
```

- KNN
  - It should be able to output variable of importance but it did not work in this project

```
# #create train test data
# train_index <- sample(row.names(aggregate_rf), 0.7*dim(aggregate_rf)[1])
# test_index <- setdiff(row.names(aggregate_rf),train_index)
# train_data <- aggregate_rf[train_index, ]
# test_data <- aggregate_rf[test_index, ]
```

```
# knn.model <- knn(train= train_data, test= test_data, cl= train_data$Overdue_count, k=5)


# or


# train_data %>% as.data.frame()
# knn.model1 <- train(Overdue_count ~ .,
#                   data = train_data,
#                   method = "knn",
#                   trControl = trainControl(method = "repeatedcv", number=10,repeats=3, search="grid"),
#                   metric = "RMSE",
#                   tunelength= 10,
#                   verbose = FALSE)
#
# knn.model1
# knn.model1$results
# plot(knn.model1)
# print(knn.model1)
#
# #find the k that has the highest accuracy
```

```
#
# #variables of importance
#
# # varImp(knn.model1)
#
# #predict
#
# knn.pred<- predict(knn.model1, newdata = test_data)
#
# RMSE(knn.pred, test_data$Overdue_count)
```