Project

2024-11-22

#Libraries

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## 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 errors
```

```
library(lubridate)
library(dplyr)
library(rpart)
library(rattle)
```

```
## Loading required package: bitops
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## randomForest 4.7-1.2
## 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
```

```
lafd <- read.csv("lafd.csv")

str(lafd)
```

```
## 'data.frame':    7201913 obs. of  11 variables:
##  $ Randomized.Incident.Number  : num  2.02e+11 2.02e+11 2.02e+11 2.02e+11 2.02e+11 ...
##  $ First.In.District           : int  38 79 93 2 33 9 2 72 93 94 ...
##  $ Emergency.Dispatch.Code     : chr  "Emergency" "Emergency" "Emergency" "Emergency" ...
##  $ Dispatch.Sequence           : int  4 4 2 5 2 1 3 7 1 1 ...
##  $ Dispatch.Status             : chr  "QTR" "QTR" "QTR" "QTR" ...
##  $ Unit.Type                   : chr  "E - ENGINE" "E - ENGINE" "E - ENGINE" "E - ENGINE" ...
##  $ PPE.Level                   : chr  "EMS" "EMS" "EMS" "EMS" ...
##  $ Incident.Creation.Time..GMT.: chr  "23:20:05" "14:00:16" "22:11:17" "00:59:16" ...
##  $ Time.of.Dispatch..GMT.      : chr  "23:20:51" "14:01:31" "22:12:25" "00:59:35" ...
##  $ En.Route.Time..GMT.         : chr  "23:21:15" "14:02:14" "22:13:24" "01:00:38" ...
##  $ On.Scene.Time..GMT.         : chr  "23:24:53" NA "22:16:51" "01:02:29" ...
```

##Data Clean

```
lafd <- lafd %>%
  rename(
    randomincidentnumber = `Randomized.Incident.Number`,
    firstindistrict = `First.In.District`,
    emergencycode = `Emergency.Dispatch.Code`,
    dispatchsequence = `Dispatch.Sequence`,
    dispatchstatus = `Dispatch.Status`,
    unittype = `Unit.Type`,
    ppelevel = `PPE.Level`,
    incidentcreationtime = `Incident.Creation.Time..GMT.`,
    timeofdispatch = `Time.of.Dispatch..GMT.`,
    enroutetime = `En.Route.Time..GMT.`,
    onscenetime = `On.Scene.Time..GMT.`
  )
```

##Fix the next day issue | Data Clean

```
lafd_filtered <- lafd

lafd_filtered <- lafd[which(lafd$incidentcreationtime <= lafd$timeofdispatch &
                            lafd$incidentcreationtime <= lafd$enroutetime &
                            lafd$incidentcreationtime <= lafd$onscenetime), ]
```

##New Features/Columns

```
lafd_filtered <- lafd_filtered %>%
  mutate(across(c(incidentcreationtime, timeofdispatch, enroutetime, onscenetime),
                ~as.POSIXct(., format = "%H:%M:%S", tz = "UTC"))) %>% mutate(
    notify_time = as.numeric(difftime(timeofdispatch, incidentcreationtime, units = "secs")),
    route_response = as.numeric(difftime(enroutetime, incidentcreationtime, units = "secs")),
    arrivial_response = as.numeric(difftime(onscenetime, incidentcreationtime, units = "secs")),
    response_time = as.numeric(difftime(enroutetime, timeofdispatch, units = "secs"))
  )
```

```r
lafd_filtered <- lafd_filtered %>%
  filter(!is.na(notify_time) & !is.na(route_response) &
         !is.na(arrivial_response) & !is.na(response_time))
```

##Speed Rank Feature

```r
##Creates a categorical instead of a continuous numerical variable

lafd_filtered_speed <- lafd_filtered %>% mutate(speed_rank = case_when(
      response_time %in% c(28:38) ~ "Fast",
       response_time %in% c(38:48) ~ "Moderately Fast",
       response_time %in% c(48:58) ~ "Moderately Slow",
       response_time %in% c(58:68) ~ "Slow"))

##creates a non na speed rank column, gets rid of outliers

lafd_filtered_speed <- lafd_filtered[!is.na(lafd_filtered$speed_rank), ]
```

##Graphs

```r
#dispatch sequence vs route respone | scatterplot

ggplot(lafd_filtered, aes(x = dispatchsequence, y=route_response, color = emergencycode, alpha = .05)) +
  geom_point() +
  labs(
    title = "dispatch sequence vs Route Response",
    x = "dispatch sequence",
    y = "en route time minus incident creation time"
  ) +
  theme_minimal()

##Histogram of Route Response
lafd_filtered$route_response <- as.numeric(lafd_filtered$route_response)

ggplot(lafd_filtered, aes(x = log(route_response + 1))) +
  geom_histogram(
    binwidth = 0.2,
    color = "black",
    fill = "skyblue",
    alpha = 0.7
  ) +
  labs(
    title = "Log-transformed Histogram of Time Differences",
    x = "Log(Time Difference + 1) (seconds)",
    y = "Frequency"
  ) +
  theme_minimal()


#Incident creation time vs route response
```

```
ggplot(lafd_filtered, aes(x = incidentcreationtime,y=route_response, alpha = .01)) +
  geom_point() +
  labs(
    title = "incident creation time vs response time",
    x = "incident creation time",
    y = "en route time minus incident creation time"
  ) +
  theme_minimal()

#Route Response vs incidentcreationtime (grouped by ppelevel)

ggplot(na.omit(lafd_filtered), aes(x = route_response, y = incidentcreationtime, color = ppelevel)) + g

#First in district vs route response

ggplot(na.omit(lafd_filtered), aes(x = firstindistrict, y = route_response, alpha = .01)) + geom_point()


#Incidentcreationtime boxplot (grouped by unittype)

ggplot(lafd_filtered, aes(x=incidentcreationtime, color = unittype)) + geom_freqpoly()

#Route Response by dispatch status

lafd_filtered %>% ggplot(aes(dispatchstatus,log(route_response + 1))) + geom_boxplot() +
  labs(title= "response time by dispatch status", x= 'dispatch status', y='response time')


#Incidentcreationtime frequency

ggplot(lafd_filtered, aes(x=incidentcreationtime, color = dispatchstatus)) + geom_freqpoly()

#First In District Mean response time graph

lafd_filtered$firstindistrict<- as.integer(lafd_filtered$firstindistrict)

mean_First <- lafd_filtered %>% group_by(firstindistrict) %>% summarize(mean_value = mean(response_time
mean_First %>% ggplot(aes(x = firstindistrict, y = mean_value)) + geom_point()
```

#Time of Day Feature and Historam

```
hour <- hour(lafd_filtered$incidentcreationtime)
lafd_filtered <- lafd_filtered %>% mutate(Hour = hour)
lafd_filtered <- lafd_filtered %>% mutate(incident_period = case_when(
      Hour %in% c(0, 1, 2, 3) ~ "Early Morning",
       Hour %in% c(4, 5, 6, 7) ~ "Morning",
       Hour %in% c(8, 9, 10, 11) ~ "Late Morning",
       Hour %in% c(12, 13, 14, 15) ~ "Afternoon",
      Hour %in% c(16, 17, 18, 19) ~ "Evening",
       Hour %in% c(20, 21, 22, 23) ~ "Late Evening"))

lafd_filtered %>% ggplot(aes(x=incident_period)) + geom_bar()
```

##Mean Variables

```
emsgroup <- lafd_filtered %>% filter(ppelevel == "EMS")
mean(emsgroup$route_response, na.rm = TRUE)

nonemsgroup <- lafd_filtered %>% filter(ppelevel == "Non-EMS")
mean(nonemsgroup$route_response, na.rm = TRUE)

enginegroup <- lafd_filtered %>% filter(unittype == "E - ENGINE")
mean(enginegroup$route_response, na.rm = TRUE)

truckgroup <- lafd_filtered %>% filter(unittype == "T - TRUCK")
mean(truckgroup$route_response, na.rm = TRUE)
```

#Forest | Find important variables for notify time

```
set.seed(123)

lafd_filtered$firstindistrict <- as.numeric(lafd_filtered$firstindistrict)
lafd_filtered$emergencycode <- as.factor(lafd_filtered$emergencycode)
lafd_filtered$dispatchsequence <- as.numeric(lafd_filtered$dispatchsequence)
lafd_filtered$dispatchstatus <- as.factor(lafd_filtered$dispatchstatus)
lafd_filtered$unittype <- as.factor(lafd_filtered$unittype)
lafd_filtered$ppelevel <- as.factor(lafd_filtered$ppelevel)


n <- nrow(lafd_filtered)
i.train <- sample(1:n, round(1/20*n), replace = FALSE)
lafd_filtered.train <- lafd_filtered[i.train,]
lafd_filtered.test <- lafd_filtered[-i.train,]


lafd_filtered.train$notify_time_numeric <- as.numeric(lafd_filtered.train$notify_time, units = "secs")


lafd_filtered.train <- lafd_filtered.train %>%
 drop_na(notify_time_numeric)


lafd_filtered.train <- lafd_filtered.train %>%
 mutate(across(where(is.character), as.factor))

lafd_filtered.train <- lafd_filtered.train %>%
  mutate(across(everything(), ~ifelse(. %in% c("", "NA", "N/A"), NA, .))) %>%
  na.omit()

levels(lafd_filtered$notify_time_numeric)
levels(lafd_filtered$notify_time_numeric)
forestnotify <- randomForest(factor(notify_time_numeric) ~ firstindistrict + emergencycode +
                             dispatchsequence + dispatchstatus + unittype + ppelevel,
                             data = lafd_filtered.train,
                             ntree = 100,
                             mtry = 3,
                             importance = TRUE, na.action = na.omit)
```

```
varImpPlot(forestnotify)

print(forestnotify)
```

##Hypothesis Testing and Bootstrapping

```
##Hypothesis Testing for PPE LEVEL using t test

#T-TEST METHOD:

ttestppelevel <- t.test(responsetime ~ ppelevel, data = lafd_filtered, var.equal = TRUE)
print(ttestppelevel)

#P = 2.2 * 10^-16: Null Hypothesis Rejected

#t = -22.032, df = 1221354, p-value < 2.2e-16

##Hypothesis Testing for dispatch status using T Test

#T-TEST METHOD:

lafd_filtered <- lafd_filtered %>%
  filter(dispatchstatus != "OTHER")

ttestdispatchstatus <- t.test(responsetime ~ dispatchstatus, data = lafd_filtered, var.equal = TRUE)
print(ttestdispatchstatus)

#t = -61.208, df = 6974552, p-value < 2.2e-16
```

```
##Bootstrap + sampling distribution for PPE level

##BOOTSTRAP: HYPOTHESIS TEST FOR PPE LEVEL AND RESPONSE TIME

set.seed(123)
n_resamples <- 1000
simulatedppelevelresponse <- lafd_filtered %>%
  group_by(ppelevel) %>%
  summarise(
    sampling_distribution = list(
      replicate(
        n_resamples,
        mean(sample(route_response, size = n(), replace = TRUE), na.rm = TRUE))))

##Calculating mean response time EMS
meanresponsetimeems <- lafd_filtered %>%
  filter(ppelevel == "EMS") %>%
  summarise(mean_response = mean(route_response, na.rm = TRUE))

print(meanresponsetimeems)

# EMS mean response time (seconds) = 122.0729

##Calculating mean response time Non-EMS
```

```r
meanresponsetimenonems <- lafd_filtered %>%
  filter(ppelevel == "Non-EMS") %>%
  summarise(mean_response = mean(route_response, na.rm = TRUE))

print(meanresponsetimenonems)

# EMS mean response time (seconds) = 143.9319

observeddiff <- 143.9319 - 122.0729

set.seed(123)
n_resamples <- 1000

nullhypothesis <- replicate(n_resamples, {
  shuffled_ppe <- sample(lafd_filtered$ppelevel)
  lafd_filtered %>%
    mutate(shuffled_ppe = shuffled_ppe) %>%
    group_by(shuffled_ppe) %>%
    summarise(mean_response = mean(route_response, na.rm = TRUE)) %>%
    summarise(diff = diff(mean_response)) %>%
    pull(diff)})

pvalue <- mean(abs(nullhypothesis) >= abs(observeddiff))
print(pvalue)

ggplot(data.frame(nullhypothesis), aes(x = nullhypothesis)) +
  geom_histogram(fill = "blue", alpha = 0.5) +
  geom_vline(xintercept = observeddiff, color = "red", linetype = "dashed") +
  geom_vline(xintercept = -observeddiff, color = "red", linetype = "dashed") +
  labs(
    title = "Null Distribution of Mean Differences",
    x = "Mean Difference (EMS - Non-EMS)",
    y = "Density"
  ) +
  theme_minimal()
```

##Fastest Scenario

```r
#fastest district

fastest_district <- lafd_filtered |> group_by(firstindistrict) |> summarize(mean_route_response = mean(

fastest_district

#emergency code
fastest_emergencycode <- lafd_filtered |> group_by(emergencycode) |> summarize(mean_route_response = mea

fastest_emergencycode

# dispatchsequence

fastest_dispatchsequence <- lafd_filtered |> group_by(dispatchsequence) |> summarize(mean_route_respons
```

```r
fastest_dispatchsequence


# dispatchstatus

fastest_dispatchstatus <- lafd_filtered |> group_by(dispatchstatus) |> summarize(mean_route_response = m

fastest_dispatchstatus

# unittype

fastest_unittype <- lafd_filtered |> group_by(unittype) |> summarize(mean_route_response = mean(route_re

fastest_unittype

# ppelevel

fastest_ppelevel <- lafd_filtered |> group_by(ppelevel) |> summarize(mean_route_response = mean(route_re

fastest_ppelevel


#fastest scenario

fast_scenerios <- lafd_filtered |>
  filter(firstindistrict == 102, emergencycode == "Emergency",
    dispatchsequence == 1,
    dispatchstatus == "QTR",
    unittype == "E - ENGINE",
    ppelevel == "EMS"
  )

fast_scenerios

fastestscenerio <- lafd_filtered |>
  filter(firstindistrict == 102, emergencycode == "Emergency",
    dispatchsequence == 1,
    dispatchstatus == "QTR",
    unittype == "E - ENGINE",
    ppelevel == "EMS"
  ) |> summarize(mean(route_response))


fastestscenerio


#regular scenario

lafd_filtered |> summarize(mean(route_response))

lafd_filtered |> arrange(route_response)
```