Tina Rezvanian

June 3, 2019

1. Question 1:

Can you create the summary statistics and histogram of Days Open on a federal case? The number of days open can be calculated as Days Open = TERMINATION_DATE – FILING_DATE.

First, I start by attaching the required libraries. I read the dataset. The data contains 457669 observations and 25 columns. I check if the data set contains any NA, NaN, or infinite value. It turns out that the only column with NA variables is the TRANSFER_DATE column, which correspond to cases that are not transfered. I check each variables range and format and then, I divide data columns into factor variables, continuous variables, and date variables in var_factor, var_cont, and var_factor, respectively, based on data type and definitions in the codebook. I create OPEN_DAYS as TERMINATION_DATE – FILING_DATE. OPEN_DAYS is a dependent variable and TERMINATION_DATE and FILING_DATE are independent variables in this question. The filing date of all cases is known, however, the termination date of pending cases is not known, as they are not terminated yet. In the data, 1900-01-01 is the termination date listed for pending cases. Therefore, in the OPEN_DAYS column, their respective number of open days has a negative value where number of days a case is open cannot be negative. Also, the value of Open Days for pending cases is not meaningful since their termination date cannot be 1900-01-01. We can assume that the OPEN_DAYS is NULL for pending cases. As a result, looking at the statistics of Open Days for all cases would not be useful because OPEN_DAYS values of pending cases is not meaningful and would deviate the results.

Looking at the summary statistics, the number of open days for closed cases lies between 0 and 11555 days. On average, cases have been open for 248.5 days. 50% of the cases have been open for less than 359.2 days. It also shows that most cases (75% of cases) have been open between 92 and 512 days. In the plot titled “Frequency Distribution: Open Days for All Cases” the histogram of Open Days clearly separates pending cases and closed cases. The plot “Frequency Distribution: Open Days for Closed Cases” is a better way of looking at the histogram of Open Days. Looking at the histogram bins and the number of cases in each bin, most cases has been open less than 5000 days, and very few cases have been open for more than 6000 days. I am also plotting the box plot to show where the range of open days are for 75% of cases and also to show the outliers.

```
df <- readRDS('./data_exercise_short.rds')
names(df)
```

```
##  [1] "CIRCUIT"                 "DISTRICT"
##  [3] "ORIGIN"                  "FILING_DATE"
##  [5] "JURISDICTION"            "DIVERSITY_RESIDENCE"
##  [7] "JURY_DEMAND"             "MONETARY_AMOUNT_DEMANDED"
##  [9] "COUNTY_OF_RESIDENCE"     "PLAINTIFF"
## [11] "DEFENDANT"               "TRANSFER_DATE"
## [13] "TERMINATION_DATE"        "PROCEDURAL_PROGRESS"
## [15] "DISPOSITION"             "NATURE_OF_JUDGMENT"
## [17] "AMOUNT_RECEIVED"         "JUDGEMENT"
## [19] "PRO_SE"                  "FEE_STATUS"
## [21] "STATUS_CODE"             "YEAR_OF_TAPE"
## [23] "id"
```

```
indx <- apply(df, 2, function(x) any(is.nan(x)| is.na(x) | is.infinite(x)))
names(df)[indx]   # "TRANSFER_DATE"
```

```
## [1] "TRANSFER_DATE"
```

```
#check missing values
# only TRANSFER_DATE has na' s: the rest are shown with -8
# no need to impute, I will generate a new variable as to show ehather uis was transfere
d (binary variable)
var_factor <- c("CIRCUIT","DISTRICT","ORIGIN","JURISDICTION","DIVERSITY_RESIDENCE" ,"JUR
Y_DEMAND",
                "COUNTY_OF_RESIDENCE","PLAINTIFF","DEFENDANT","PROCEDURAL_PROGRESS","DIS
POSITION","NATURE_OF_JUDGMENT" ,
               "JUDGEMENT","PRO_SE","FEE_STATUS","STATUS_CODE","YEAR_OF_TAPE","id")

for (i in var_factor){
  df[[i]] <- as.factor(df[[i]])
}

var_date <- c("FILING_DATE", "TRANSFER_DATE" , "TERMINATION_DATE")

for (i in var_date){
  df[[i]] <- as.Date(df[[i]])
}

var_cont <- c("MONETARY_AMOUNT_DEMANDED", "AMOUNT_RECEIVED" )

for (i in var_cont){
  df[[i]] <- as.numeric(df[[i]])
}


df$OPEN_DAYS <- df$TERMINATION_DATE - df$FILING_DATE
df$OPEN_DAYS <- as.numeric(df$OPEN_DAYS)

ggplot(df, aes(x=OPEN_DAYS)) +
  geom_histogram(binwidth = 500, fill ="cyan3") +
  xlab('Open Days') + ylab('Frequency')+ ggtitle('Frequency Distribution: Open Days for
 All Cases')+
  theme(plot.title = element_text(hjust = 0.5))

closed_df <- subset(df, STATUS_CODE=='L')  # the rest are not terminated yet, 1900-01-01

ggplot(closed_df, aes(x=OPEN_DAYS)) +
  geom_histogram(binwidth = 500, fill ="cyan3", bins = 100) +
  xlab('Open Days') + ylab('Frequency')+ ggtitle('Frequency Distribution: Open Days for
 Closed Cases')+
  theme(plot.title = element_text(hjust = 0.5))

ggplot(closed_df, aes(y=OPEN_DAYS, x = 0)) +
  geom_boxplot(fill ="cyan3", color = 'black', width=0.2) +
   ylab('Open Days')+xlim(-0.5,0.5) + ggtitle('Box Plot: Open Days for Closed Cases')+
  theme(plot.title = element_text(hjust = 0.5))

ggplot(closed_df, aes(y=OPEN_DAYS, x = 0)) +
  geom_boxplot(fill ="cyan3", color = 'black', width=0.2) +
   ylab('Open Days')+xlim(-0.5,0.5) + ggtitle('Box Plot: Open Days for Closed Cases')+
  theme(plot.title = element_text(hjust = 0.5)) + ylim(0,2500)
```

```
y_hist <- hist(closed_df$OPEN_DAYS,breaks=100)
print(y_hist)
```

```
## $breaks
##  [1]     0   200   400   600   800  1000  1200  1400  1600  1800  2000
## [12]  2200  2400  2600  2800  3000  3200  3400  3600  3800  4000  4200
## [23]  4400  4600  4800  5000  5200  5400  5600  5800  6000  6200  6400
## [34]  6600  6800  7000  7200  7400  7600  7800  8000  8200  8400  8600
## [45]  8800  9000  9200  9400  9600  9800 10000 10200 10400 10600 10800
## [56] 11000 11200 11400 11600 11800 12000 12200 12400 12600 12800 13000
## [67] 13200 13400 13600 13800 14000 14200 14400 14600 14800 15000 15200
## [78] 15400 15600 15800 16000 16200 16400 16600 16800 17000 17200 17400
## [89] 17600 17800 18000 18200 18400 18600 18800
##
## $counts
##  [1] 162708  84758  48908  23673  16053  12558   8561   5643   3449   1973
## [11]    982    760   1198    368    299    258    183    106     63     60
## [21]     32     21     17      9      8      5      9     12      3      3
## [31]      2      2      3      3      2      1      0      2      1      1
## [41]      2      0      2      0      1      0      0      0      3      0
## [51]      0      0      0      0      1      0      0      2      0      0
## [61]      0      0      0      0      0      0      0      0      0      0
## [71]      0      0      0      0      0      0      0      0      0      0
## [81]      0      0      0      0      0      0      0      0      0      0
## [91]      0      2      0      1
##
## $density
##  [1] 2.182764e-03 1.137047e-03 6.561116e-04 3.175785e-04 2.153545e-04
##  [6] 1.684683e-04 1.148477e-04 7.570209e-05 4.626909e-05 2.646823e-05
## [11] 1.317375e-05 1.019557e-05 1.607143e-05 4.936801e-06 4.011151e-06
## [16] 3.461127e-06 2.454985e-06 1.422013e-06 8.451588e-07 8.049132e-07
## [21] 4.292870e-07 2.817196e-07 2.280587e-07 1.207370e-07 1.073218e-07
## [26] 6.707610e-08 1.207370e-07 1.609826e-07 4.024566e-08 4.024566e-08
## [31] 2.683044e-08 2.683044e-08 4.024566e-08 4.024566e-08 2.683044e-08
## [36] 1.341522e-08 0.000000e+00 2.683044e-08 1.341522e-08 1.341522e-08
## [41] 2.683044e-08 0.000000e+00 2.683044e-08 0.000000e+00 1.341522e-08
## [46] 0.000000e+00 0.000000e+00 0.000000e+00 4.024566e-08 0.000000e+00
## [51] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 1.341522e-08
## [56] 0.000000e+00 0.000000e+00 2.683044e-08 0.000000e+00 0.000000e+00
## [61] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [66] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [71] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [76] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [81] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [86] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [91] 0.000000e+00 2.683044e-08 0.000000e+00 1.341522e-08
##
## $mids
##  [1]   100   300   500   700   900  1100  1300  1500  1700  1900  2100
## [12]  2300  2500  2700  2900  3100  3300  3500  3700  3900  4100  4300
## [23]  4500  4700  4900  5100  5300  5500  5700  5900  6100  6300  6500
## [34]  6700  6900  7100  7300  7500  7700  7900  8100  8300  8500  8700
## [45]  8900  9100  9300  9500  9700  9900 10100 10300 10500 10700 10900
## [56] 11100 11300 11500 11700 11900 12100 12300 12500 12700 12900 13100
## [67] 13300 13500 13700 13900 14100 14300 14500 14700 14900 15100 15300
## [78] 15500 15700 15900 16100 16300 16500 16700 16900 17100 17300 17500
```

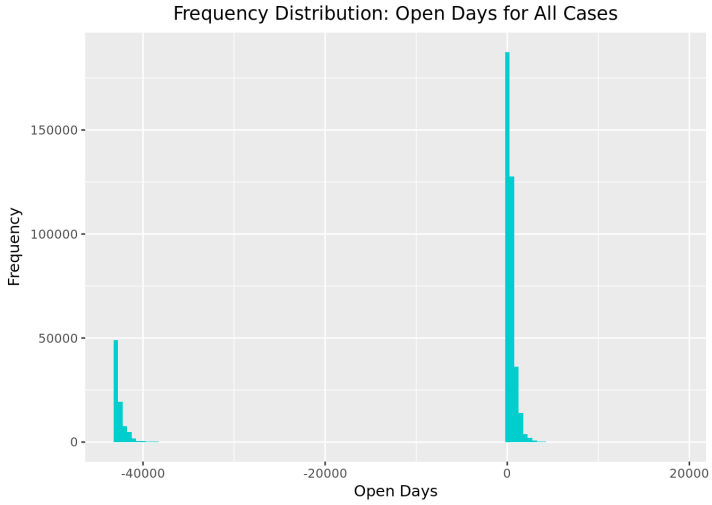
```
## [89] 17700 17900 18100 18300 18500 18700
##
## $xname
## [1] "closed_df$OPEN_DAYS"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

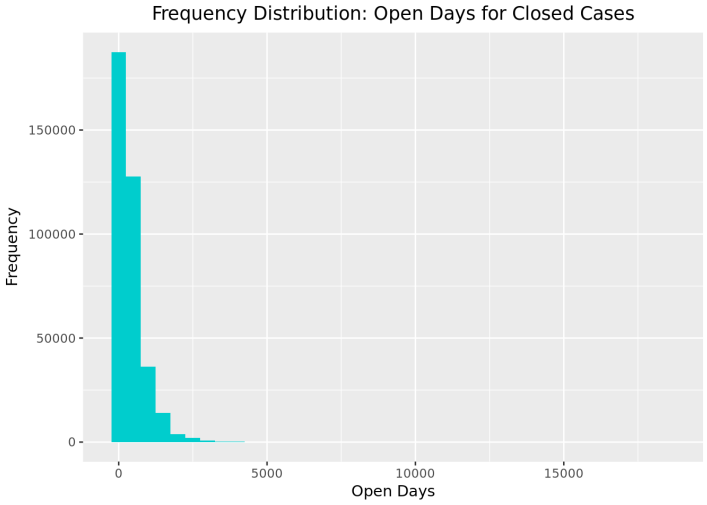
```
summary(closed_df$OPEN_DAYS)
```

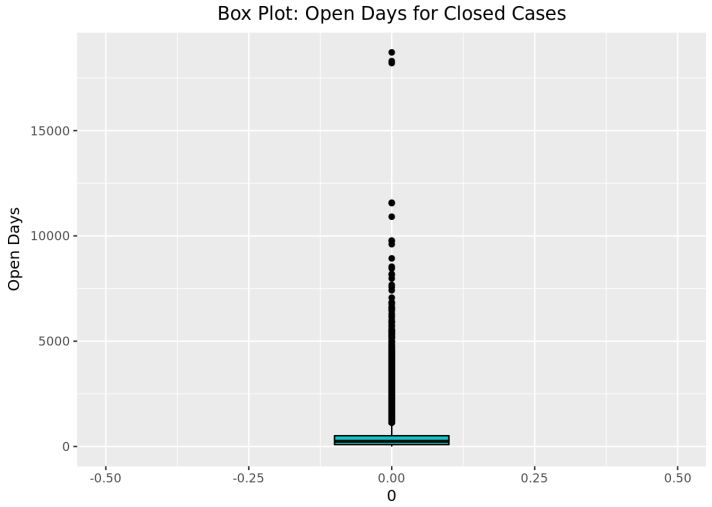
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    92.0   248.0   396.2   512.0 18713.0
```

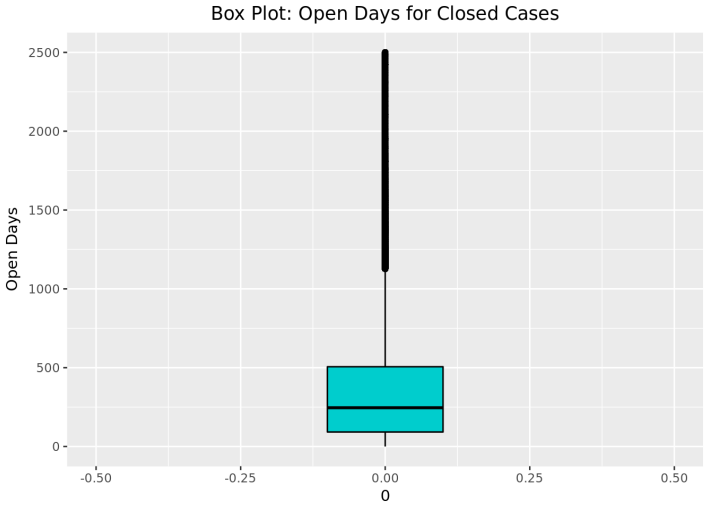
```
describe(closed_df$OPEN_DAYS)
```

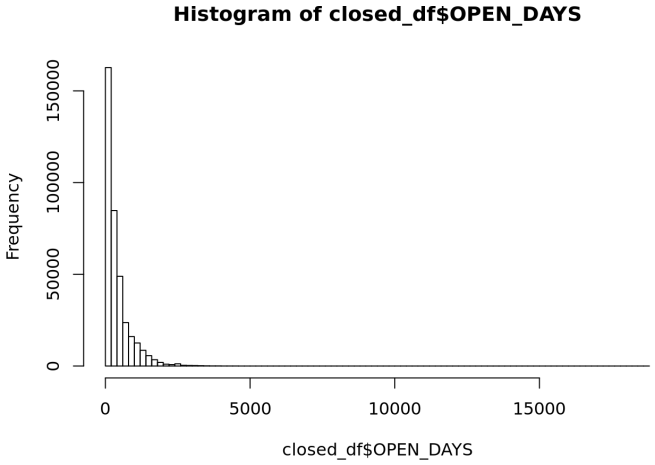
```
## closed_df$OPEN_DAYS
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##   372711        0     3425        1    396.2    433.3       11       29
##      .25      .50      .75      .90      .95
##       92      248      512      994     1314
##
## lowest :     0     1     2     3     4, highest: 11555 11576 18205 18298 18713
```











2. Question 2:

How is “Days Open” related to a case’s Circuit (CIRCUIT) and the year the case was closed (YEAR OF TAPE)?

Open Days is a dependent variable, referred to a (Y) and the remaining variables are independent variables (Xs). CIRCUIT is a categorical variable with a total of 12 levels. YEAR OF TAPE is the year at which cases are terminated. Year of tape is listed as 2099 for the pending cases. It is better to consider YEAR OF TAPE as a categorical variable, since YEAR OF TAPE for pending cases is not meaningful, and its magnitude does not

represent meaningful values. YEAR OF TAPE has seven levels with level 2099 associated to pending cases. I am using three techniques to identify how Days Open is related to Circuit and Year Of Tape: 1) Visualization, 2) correlation Analysis, and 3) hypothesis testing.

1. Visualization:

Plotting Open Days with respect to YEAR OF TAPE for all data in plot “Open Days for All Cases per Year of Tape”, clearly separates the pending cases in year label ‘2099’ and negative values. Since we do not have data on the number of open days for pending cases, I use only closed cases to find how Open Days is related to CIRCUIT and YEAR OF TAPE. Plots “Open Days for Closed Cases per Year of Tape” and “Density Distribution of Open Days per Year of Tape” show how number of Open Days changes per category of Year of Tape. Both plots show similar Open Days per years. Plots “Open Days for Closed Cases per Circuit” and “Density Distribution of Open Days per Circuit” show how number of Open Days changes per category of Circuit. Circuit 4 has greater number of Open days.the distribution of open days in Circuit 4 is more disperesed than in the other circuits. Circuits with greater Open Days include cirsuit 4, followed by Circuit 1 and 7. Plot “Mean of Open Days for each Circuit and Year of Tape” looks at the mean number of open days where both year of tape and circuit change. For example circuit 4 and years of 2016, 2017, and 2018 in an increasing order, have higher mean. Plots of “‘Median of Open Days for each Circuit and Year of Tape’” similarly, plots the median of open days when year of tape and circuit interact. Circuit 4 in years of 2016, 2017, and 2018 have positive relationship with Open Days, in an increasing order. They Also have greater median than other levels in Year of Tape and Circuit. Same is true for circuit 8 in years 2013 and 2014, Circuit 7 in year 2015,Circuit 1 in year 2015, and Circuit 10 and year 2018. Plots “‘Box Plot: Open Days for each Circuit and Year of Tape” and “Box Plot: Open Days for each Circuit and Year of Tape” shows boxplot from the number of open days of cases in when year of tape and circuit change. Although we can see the median here to, it is a good way to captures the distribution of cases’ open days.

2. Correlation Analysis:

I look at the correlation between Open Days with Year of Tape and Year Of Tape indivitually and in combination. In order to find the correlation I use Biserial correlation as I need the correlation between a continuous and a categorical variable.

Individual: I look at the correlation between Open Days with Year of Tape. I also look at the correlation between Open Days with Circuit levels.The results are in plot “Biserial Correlation with Circuit and Year of Tape”. Circuits, 1, 4, 6, 7, 8 have positive relationship with number of open days. Therefore cases in circuit 1, 4, 6, 7, 8 have greater number of open days. Also, years 17 and 18 have positive relationship with number of open days. Cases Closed on years 17 and 18 have greater number of open days.

Interaction: I calculate the Biserial correlation for every combination of Year of Tape and Circuit. The results confirm the boxplots from the visualization section, where cases that are in the Circuit 4 and years of 2016, 2017, and 2018 have the highest positive relationship with Open Days. It can be seen that Circuit 9, 10, and 11 have negative relation with the number of open days in all circuit levels. Also Circuit 2 have negative relation with the number of open days in all circuit levels. Circuit 8 in years 2013 and 2014, Circuit 6 and 7 in year 2015, Circuit 1 in year 2015. Circuit 10 in year 2018 follow with the highest positive relation.

3. Hypothesis Testing:

Based on the histogram in question 1 and also the density plots in the visualization section, we know that the distribution of Open Days is not normal, and using standard one-way anova test is not justified. Kruskal-Wallist test is a non-parametric test that investigates if the parameters of distribution of open days across different levels of Circuit is equal. The null hypothesis is rejected as the resulting p-value < 2.2e-16. Now let’s test if the parameters of the distribution of yearly open days is equal. Again, the p-value < 2.2e-16 rejects this hypothesis. CIRCUIT_YEAR_OF_TAPE is new categorical variable that shows all combinations of Year of Tape and Circuit. It has 72 levels (12*6=72). In this section I test if the parameters of distribution of open days across all these 72

levels is equal.
Therefore, the parameters of at least one of the distribution of open days across all these 72 levels is not equal. The p-value < 2.2e-16 rejects this hypothesis. The value of the parameter of the open days distribution differs at least one of the categories.

```
yot <- as.numeric(as.numeric_version(df$YEAR_OF_TAPE))
print (c("Min of Year of Tape: ", min(yot), "Max of Year of Tape: ", max(yot)))
```

```
## [1] "Min of Year of Tape: " "2013"                 "Max of Year of Tape: "
## [4] "2099"
```

```
#  all cases:
ggplot(df, aes(y=OPEN_DAYS, x = YEAR_OF_TAPE, fill = YEAR_OF_TAPE)) +  ggtitle('Open Day
s for All Cases per Year of Tape')+
  theme(plot.title = element_text(hjust = 0.5))+
  geom_boxplot()+xlab('Year of Tape')+ylab('Open Days')


ggplot(closed_df, aes(y=OPEN_DAYS, x = YEAR_OF_TAPE, fill = YEAR_OF_TAPE)) +  ggtitle('O
pen Days for Closed Cases per Year of Tape')+
  theme(plot.title = element_text(hjust = 0.5))+
  geom_boxplot()+xlab('Year of Tape')+ylab('Open Days')

ggplot(closed_df, aes(y=OPEN_DAYS, x = YEAR_OF_TAPE, fill = YEAR_OF_TAPE)) +  ggtitle('O
pen Days for Closed Cases per Year of Tape (Zoomed)')+
  theme(plot.title = element_text(hjust = 0.5))+
  geom_boxplot()+xlab('Year of Tape')+ylab('Open Days') + ylim(0,2500)

ggplot(df, aes(y=OPEN_DAYS, x = CIRCUIT, fill = CIRCUIT)) +  ggtitle('Open Days for All
 Cases per Circuit')+
  theme(plot.title = element_text(hjust = 0.5))+
  geom_boxplot()+xlab('Circuit')+ylab('Open Days')


ggplot(closed_df, aes(y=OPEN_DAYS, x = CIRCUIT, fill = CIRCUIT)) +  ggtitle('Open Days f
or Closed Cases per Circuit')+
  theme(plot.title = element_text(hjust = 0.5))+
  geom_boxplot()+xlab('Circuit')+ylab('Open Days')

ggplot(closed_df, aes(y=OPEN_DAYS, x = CIRCUIT, fill = CIRCUIT)) +
  geom_boxplot() + ylim(0,2500) +  ggtitle('Open Days for Closed Cases per Circuit (Zoom
ed)')+
  theme(plot.title = element_text(hjust = 0.5))+
  geom_boxplot()+xlab('Circuit')+ylab('Open Days')


ggplot(closed_df, aes(closed_df$OPEN_DAYS)) +
  geom_density(aes(data = closed_df$OPEN_DAYS, fill = closed_df$CIRCUIT), position = 'id
entity', alpha = 0.5) +
  labs(x = 'Circuit', y = 'Density') + scale_fill_discrete(name = 'CIRCUIT')+  ggtitle(
'Density Distribution of Open Days per Circuit')+
  theme(plot.title = element_text(hjust = 0.5))

ggplot(closed_df, aes(closed_df$OPEN_DAYS)) +
   geom_density(aes(data = closed_df$OPEN_DAYS, fill = closed_df$YEAR_OF_TAPE), position
 = 'identity', alpha = 0.5) +
  labs(x = 'Year of Tape', y = 'Density') + scale_fill_discrete(name = 'YEAR_OF_TAPE')+
ggtitle('Density Distribution of Open Days per Year of Tape')+
  theme(plot.title = element_text(hjust = 0.5))

# grouped <- closed_df %>%
#   dplyr::group_by(CIRCUIT, YEAR_OF_TAPE) %>%
#   dplyr::summarise(
#     Mean=mean(OPEN_DAYS),
```

```
#     Median = median(OPEN_DAYS), SD = sd(OPEN_DAYS))

grouped <- read.csv('./grouped.csv')


ggplot(grouped, aes(x=CIRCUIT, y=YEAR_OF_TAPE)) + geom_point(aes(size = Mean), color='cy
an3')+
  xlab('Circuit') + ylab('Year of Tape') + ggtitle('Mean of Open Days for each Circuit a
nd Year of Tape')+
  theme(plot.title = element_text(hjust = 0.5))


ggplot(grouped, aes(x=CIRCUIT, y=YEAR_OF_TAPE)) + geom_point(aes(size = Median), color=
'orange')+
  xlab('Circuit') + ylab('Year of Tape') + ggtitle('Median of Open Days for each Circuit
and Year of Tape')+
  theme(plot.title = element_text(hjust = 0.5))


ggplot(closed_df, aes(x=CIRCUIT, y=OPEN_DAYS, color=YEAR_OF_TAPE)) +
  geom_boxplot(position=position_dodge(0.9), outlier.size = 0.5)+
  ylim(0,6000) + xlab('Circuit')+ylab('Open Days') +  ggtitle('Box Plot: Open Days for e
ach Circuit and Year of Tape')+
  theme(plot.title = element_text(hjust = 0.5))


ggplot(closed_df, aes(x=YEAR_OF_TAPE, y=OPEN_DAYS, color=CIRCUIT)) +
  geom_boxplot(position=position_dodge(0.9), outlier.size = 0.5)+
  ylim(0,6000) + xlab('Year of Tape')+ylab('Open Days') +  ggtitle('Box Plot: Open Days
 for each Circuit and Year of Tape')+
  theme(plot.title = element_text(hjust = 0.5))

# 2)  Correlation Analysis

# q2_data <- closed_df[,colnames(closed_df) %in% c("CIRCUIT","YEAR_OF_TAPE", "OPEN_DAY
S")]
# q2_data_bm <- createDummyFeatures(q2_data, cols = c("CIRCUIT","YEAR_OF_TAPE"))
#
# names<-names(q2_data_bm)[2:19]
#
#
# cor_df <- data.frame(cor = 1:18)
# row.names(cor_df) <- names
#
# for (i in names){
#   cor_df[i, 'cor'] <- biserial.cor(as.numeric(q2_data_bm$OPEN_DAYS), q2_data_bm[[i]])
# }

cor_df <- read.csv('./cor_df_single.csv')
cor_df$cor <- cor_df$cor*(-1)
cor_df$sign <- ifelse(cor_df$cor > 0, 'Positive', 'Negative')

ggplot(cor_df, aes(x=X, y=cor)) +
  geom_point(aes( color=sign)) + theme(axis.text.x = element_text(size=8, angle=90))+
```

```
  xlab(' ')+ ylab('Correlation')+ ggtitle('Point-Biserial Correlation with Circuit and Y
ear of Tape')+
  theme(plot.title = element_text(hjust = 0.5))



closed_df$CIRCUIT_YEAR_OF_TAPE <- with(closed_df, interaction(CIRCUIT,  YEAR_OF_TAPE), d
rop = TRUE )
# q2_data <- closed_df[,colnames(closed_df) %in% c("CIRCUIT_YEAR_OF_TAPE", "OPEN_DAYS")]
# q2_data_bm <- createDummyFeatures(q2_data, cols = c("CIRCUIT_YEAR_OF_TAPE"))
#
# names<-names(q2_data_bm)[2:73]
#
#
# cor_df <- data.frame(cor = 1:72)
# row.names(cor_df) <- names
#
# for (i in names){
#   cor_df[i, 'cor'] <- biserial.cor(as.numeric(q2_data_bm$OPEN_DAYS), q2_data_bm[[i]])
# }
#
# cor_df$CIRCUIT_YEAR_OF_TAPE <- names
# cor_df$CIRCUIT <- substr(cor_df$CIRCUIT_YEAR_OF_TAPE, nchar(cor_df$CIRCUIT_YEAR_OF_TAP
E)-7+1, nchar(cor_df$CIRCUIT_YEAR_OF_TAPE)-5)
#
# cor_df$YEAR_OF_TAPE <- substr(cor_df$CIRCUIT_YEAR_OF_TAPE, nchar(cor_df$CIRCUIT_YEAR_O
F_TAPE)-4+1, nchar(cor_df$CIRCUIT_YEAR_OF_TAPE))
#
#
# cor_df$Correlation <- abs(cor_df$cor)
# cor_df$sign <- ifelse(cor_df$cor > 0, 'Negative','Positive')

cor_df <- read.csv('./cor_df_interaction.csv')
cor_df$CIRCUIT <- as.character(cor_df$CIRCUIT)

ggplot(cor_df, aes(x=CIRCUIT, y=YEAR_OF_TAPE)) + geom_point(aes(size = Correlation, colo
r = sign))+
  xlab('Circuit') + ylab('Year of Tape') + ggtitle('Point-Biserial Correlation for each
 Circuit and Year of Tape')+
  theme(plot.title = element_text(hjust = 0.5))


# 3)  Hypothesis Testing:

kruskal.test(OPEN_DAYS ~  CIRCUIT , data = closed_df)
```

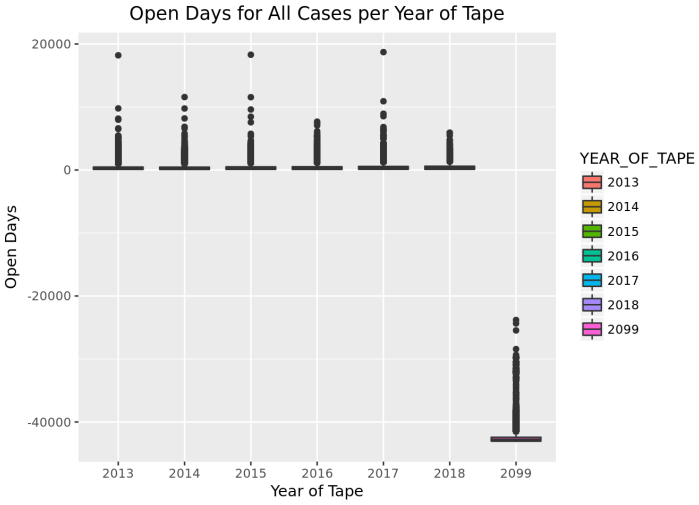
```
##
##  Kruskal-Wallis rank sum test
##
## data:  OPEN_DAYS by CIRCUIT
## Kruskal-Wallis chi-squared = 15815, df = 11, p-value < 2.2e-16
```

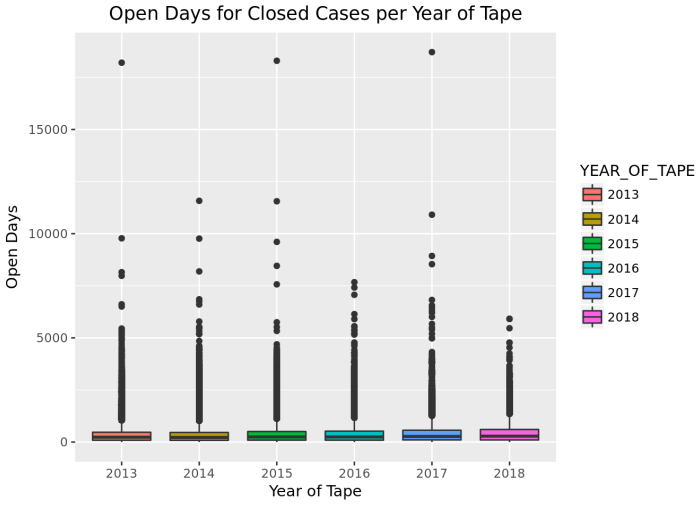
```
kruskal.test(OPEN_DAYS ~  YEAR_OF_TAPE , data = closed_df)
```

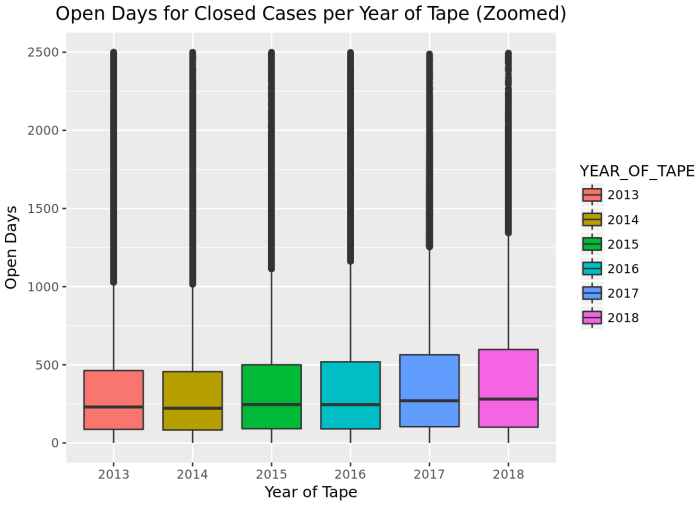
```
##
##  Kruskal-Wallis rank sum test
##
## data:  OPEN_DAYS by YEAR_OF_TAPE
## Kruskal-Wallis chi-squared = 1470.8, df = 5, p-value < 2.2e-16
```

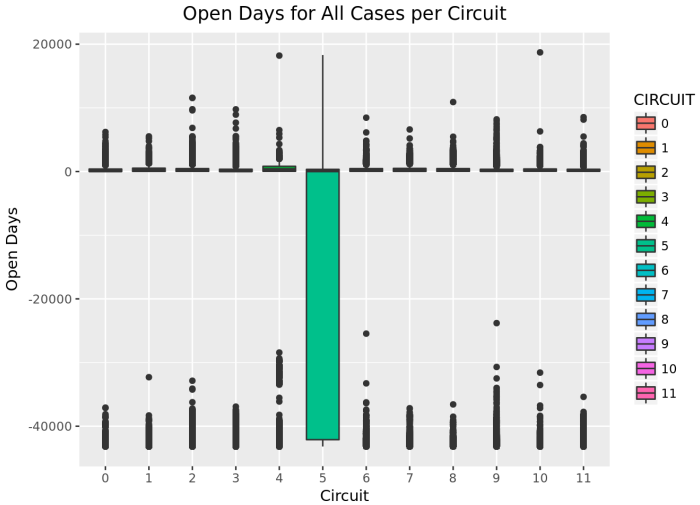
```
kruskal.test(OPEN_DAYS ~  CIRCUIT_YEAR_OF_TAPE , data = closed_df)
```

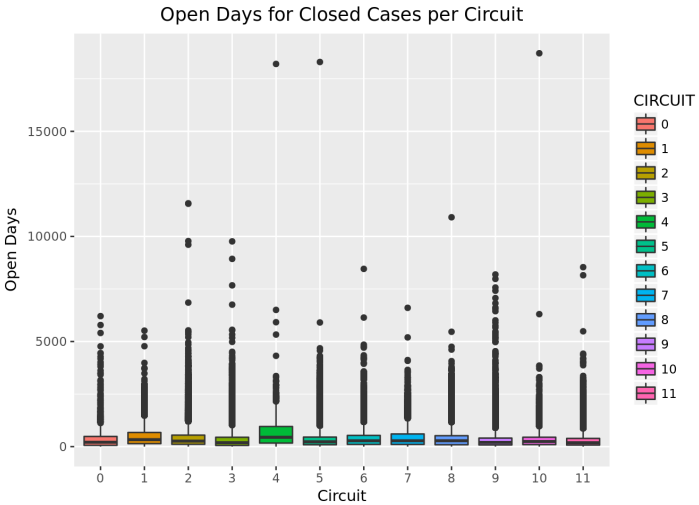
```
##
##  Kruskal-Wallis rank sum test
##
## data:  OPEN_DAYS by CIRCUIT_YEAR_OF_TAPE
## Kruskal-Wallis chi-squared = 25528, df = 71, p-value < 2.2e-16
```

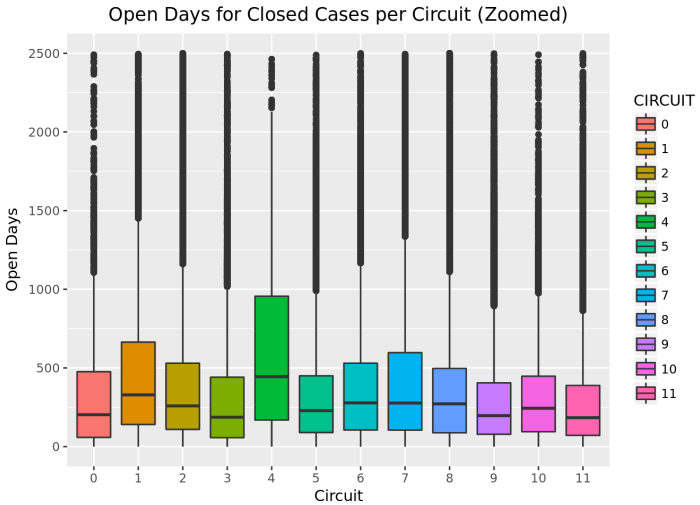

Open Days for All Cases per Year of Tape
Open Days
Year of Tape
YEAR_OF_TAPE
2013
2014
2015
2016
2017
2018
2099

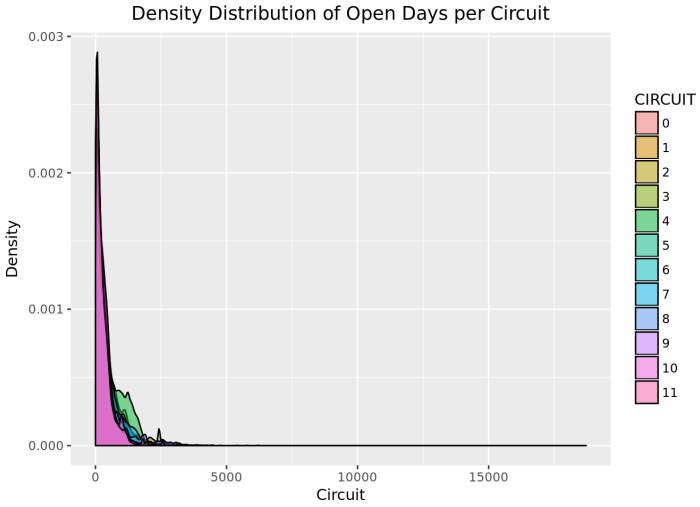

Open Days for Closed Cases per Year of Tape
Open Days
Year of Tape
YEAR_OF_TAPE
2013
2014
2015
2016
2017
2018

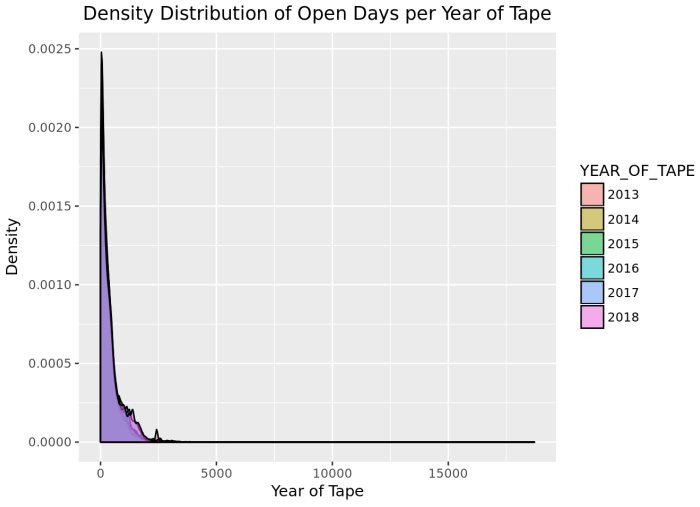

Open Days for Closed Cases per Year of Tape (Zoomed)
Open Days
Year of Tape
YEAR_OF_TAPE
2013
2014
2015
2016
2017
2018

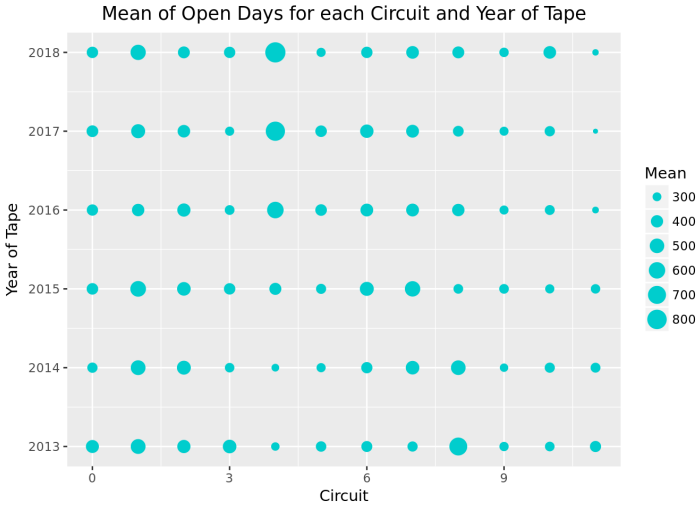

Open Days for All Cases per Circuit
Open Days
Circuit
CIRCUIT
0
1
2
3
4
5
6
7
8
9
10
11

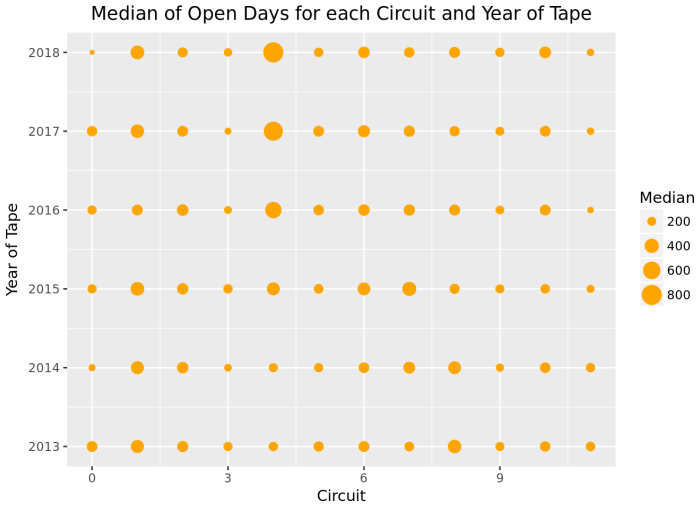

Open Days for Closed Cases per Circuit
Open Days
Circuit
CIRCUIT
0
1
2
3
4
5
6
7
8
9
10
11

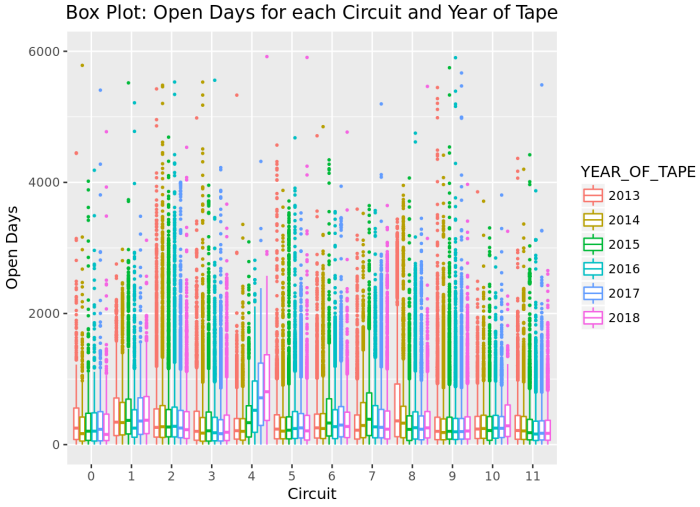

Open Days for Closed Cases per Circuit (Zoomed)
Open Days
Circuit
CIRCUIT
0
1
2
3
4
5
6
7
8
9
10
11

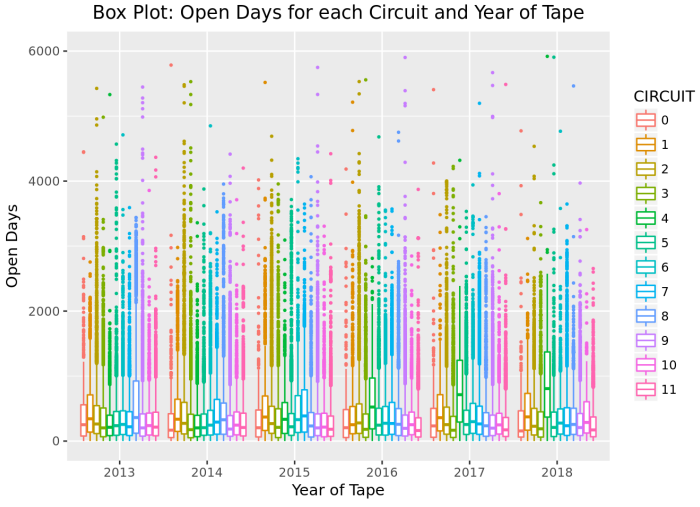

Density Distribution of Open Days per Circuit
Density
Circuit
CIRCUIT
0
1
2
3
4
5
6
7
8
9
10
11

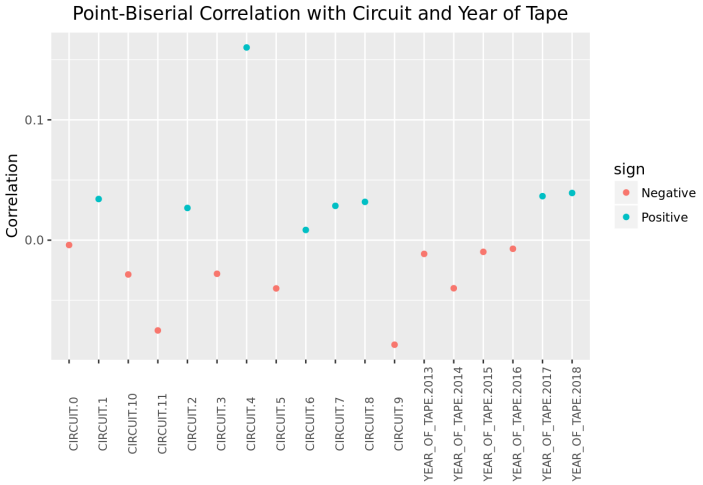

Density Distribution of Open Days per Year of Tape
Density
Year of Tape
YEAR_OF_TAPE
2013
2014
2015
2016
2017
2018

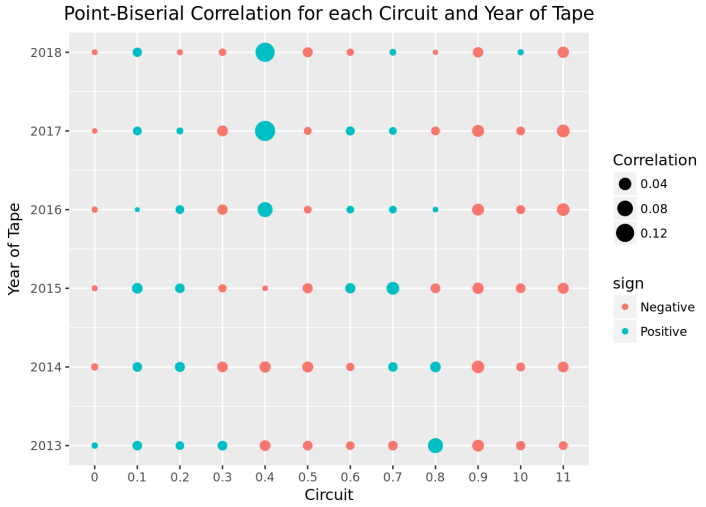

Mean of Open Days for each Circuit and Year of Tape
Year of Tape
Circuit
Mean
300
400
500
600
700
800


Median of Open Days for each Circuit and Year of Tape
Year of Tape
Circuit
Median
200
400
600
800


Box Plot: Open Days for each Circuit and Year of Tape
Open Days
Circuit
YEAR_OF_TAPE
2013
2014
2015
2016
2017
2018


Box Plot: Open Days for each Circuit and Year of Tape
Open Days
Year of Tape
CIRCUIT
0
1
2
3
4
5
6
7
8
9
10
11





Question 3:

Knowing the expected Days Open for a case when it is filed could be very helpful for litigation management. Could you build a model using closed cases that predicts the Days Open for pending federal cases? How did your model perform on a holdout of closed cases? Why did you choose this model over other approaches? Tip: In the dataset, the variable STATUS_CODE indicates whether the case is closed (L) or pending (S). Think about how the status of the case might affect the choice of predictors in your model.

Data Preperation and Feature Engineering: After Checking missing values, value ranges, and outliers, I prepare my data by keeping the useful variables in keep_var. Although, The only variable with NA values is TRANSFER_DATE, based on the codebook, “-8” stands for missing values. “DIVERSITY_RESIDENCE”,“PLAINTIFF”, “DEFENDANT”,“FEE_STATUS” are columns in data that include “-8”. Since all of these four variables are factor variables, I get “-8” as another separate category. I will look into each of the four columns later in the section to reduce their category levels are elliminate them.

Column id, includes unique values and does not provide any information for predicting Open Days. Columns STATUS_CODE, DISPOSITION, PROCEDURAL_PROGRESS, AMOUNT_RECEIVED, JUDGMENT, NATURE_OF_JUDGMENT, and TERMINATION_DATE are eliminated as they not useful in prediction model for Open Days. For example procedural process of pending cases are associated to “-8” while its levels in closed cases includes 13 non-missing valu categories. As a result,it is not useful for the prediction and is excluded from the model. Also AMOUNT_RECEIVED is information that will be known after a case is closed. other columns, either represent future information, or are not useful, or leak information. The column DIVERSITY_RESIDENCE is eliminated as it is defined for only 104601 observations.

Eliminating Dates: Since we are only looking into the closed cases in training models, therefore, using filing dates as a predictor biases the results. for example, considering we only use closed cases for training, the later filing dates would associate to least open days. , YEAR_OF_TAPE is also elliminated with the same logic. TRANSFER_DATE is a date that is dependent to the filing date. I generate a binary variable with value 1 if a case has been transferred, and 0 otherwise.

Reducing Categories: COUNTY_OF_RESIDENCE and MONETARY_AMOUNT_DEMANDED are two predictors that their levels are merged based on the definitions in the codebook. For example, for COUNTY_OF_RESIDENCE, values of 88888 are renamed as category “3” and 99999 as category “4”. Those whose plaintiff is US Government are found from the plaintiff column and renamed as “1” and the rest of the counties are category “2”. Now the number of categories is reduced to 4 from 3052. For the MONETARY_AMOUNT_DEMANDED, the codebook states that values above 10,000 are listed as 9999. Dollar figure is rounded to the nearest thousand and that data may not be accurate. Plot “Histogram: Monetary Amount

Demanded" shows that most of the data is in the first bin. Plot "Open Days vs. Monetary Amount Demanded" shows that the average number of Open Days for greater Monetary Amount Demanded is slightly decreasing. Therefore, due to non-accuracy of data and distribution of the variable, I categories this continuous variable into 3 categories as "1" for less than or equal to 2000, "2" for greater than 2000 and less than or equal to 8000, and "3" for greater than 8000. I look at the histogram of a variable and how number of open days vary across different levels of the variable, and reduce the number categories. For example: DISTRICT and PRO SE are two variables that their greatest 'p' categories are kept while the remaining are merged using 'mergelevels' function. ORIGIN is merged based on the fact that '8', '9,'10','11','12' are all reopened cases in codebook description. After plotting the histogram and extracting number of cases in each category, categories "3","7",'13' have few observations and are merged.

Final Set of Predictors, Model Development, Performance Metrics : Vector keep_var has all the variables that are used for a prediction model. All predictors are categorical. Data frame "data" represents the total training set including closed cases and useful variables. As explained in question 1, there are few cases with number of open days greater than 6000, which are eliminated as outliers. Training and Testing Data: 75% of data is randomly selected for training prediction models and the rest will be unseen by models to test the performance of models. Model Based on Response Variable and Data characters, and Model Assumptions: GLM: The number of Open Days is of count type and is bounded to be equal or greater than zero. The linear model is not a good option as it treats the response variable as a continous variable, and in prediction produces negative values. Alternatively, after fitting a linear model, and plotting the error terms, it is clear that the errors do not have equal variance across fitted values. The Q-Q plot is also far from the diagnal line and therefore, the distribution of errors is not normal. The data does not have the linear regression model's assumption. The performance of the model is shown both on the training set to see the fit measure, and on the unseen test data to capture the test error. RMSE of 388.8285, shows the average of squared differences between prediction and actual open days. A R-squared of 26% shows that using this model, we have 26% improvemnet in our predictions than simply using the average of open days of trainig data. AIC also is a metric useful for comparing multiple models, particularly models with different number of variables. AIC of the linear regression is 4,145,711. A generalized Linear Model (GLM) with Poisson link function does not have the restrictive assumptions of the linear regression model. However, after fitting the Poisson GLM model and Chi-square test shows that the Poisson model does not fit the data. In order to keep the record of model metrics, the R-squares of 24.5%, RMSE of 392.94, and AIC of 86,851,400 are improved in the Poisson model compared to the linear regression model. Investigating the reason for why Poisson model does not fit the data. The result of the dispersion test favors the alternative hypothesis that the true dispersion is greater than 1. Therefore, a negative binomial or zero-inflated model are more appropriate. Following the same logic as the Poisson GLM, a Negative Binomial Model is fitted as tested through chi-squared test. The test again suggests that the negative binomial model does not fit the data. The dispersion may not be fully captured by negative binomial model. ZNB: Zero-Inflated Negative Binomial Model (ZNB) fits the data and resolves the dispersion and zero-inflation. ZNB is a two phase method where part of cases with zero Open Days are used in a binary classification model to identify the excess zeros. This part of model is named zero model. The part that trains and predicts the number of Open Days is the count model. The ZNB model performance metrics are as follows. The model AIC of 3,817,377 is improved compared to previous models, however, its prediction performance on test data is worsened, shown by RMSE of 401.98 and R-squares 21%. The Vuong test compares the zero-inflated model with an ordinary Negative Binomial regression model. Here, the test statistic is significant, indicating that the zero-inflated model is superior to the standard Negative Binomial model.

Now moving to machine learning methods, tree-based methods are another way to develop regression models. Trees segment the predictor space into a number of regions. Each time, the tree is split into two, in a way that returns the greatest improvement is a loss function such as residual sum of squares for a regression model. Tree based models do not have any assumptions and can capture interactions between variables. They tend to perform well with categorical variables but can become computationally expensive as more categories per predictor results in more branches. Using rpart library, I grow a tree on trainig data as large as possible. Then I

prune it through tuning the cp (complexity parameter) parameter. Tree: The model “Tree” is a simple tree trained on the train data. The Mean Squared Error of the Tree Model is 129,890.68. This is the MSE of the fitted values versus actual values in the training data. In terms of its performance on the unseen portion of data, the Tree model reduces the test error metric of RMSE to 355.03, and improves R2-squared to %38. Now using a tree model, the predictions would be 38% better than simply using the average of the open days. Using a tree can overfit the data in the training phase, it is often useful to prune the tree and re-evaluate the performance of model on the unseen test data. I plot how relative errors are changing for different levels of cp values. Then I extract the value of cp at associated to the minimum error. Since error curve becomes saturated, I choose the minimum level of cp that saturates the relative error to prune the tree. As expected, the MSE of the model is increases to 130,258.06 as it is on the training portion. If the Tree model is overfitting, we expect that the Pruned Tree Model reduces the test error. In terms of its performance on the unseen portion of data, the test error of RMSE for the Pruned Tree Model remains approximately the same 355.51. The same is true for the R2-squared of %38. So pruning has reduced the complexity of the model without sacrificing from its predictive performance and test error. By comparing the actual versus predictions plot of the Pruned Tree model with Zero-Inflated Negative Binomial Model, many points at the top left corner of the ZNB plot are now vanished. The dispersion of points are more directed towards the diagonal line that represents a perfect prediction. Following the same logic, I use Random Forest Model (RF) to strategically reduce overfitting and add a little bit of noise to the model. RF is an ensemble learning method that captures different subsets of variables and observations from the training data. It trains a tree on each subset and outputs the mean of the Open Days at each ending node (leaf) of the subset tree. Since the number of cases are 372,680 in the closed cases data, I use H2O library to train a random forest. Initially, I run a RF model with default parameters, to use as a bench mark for my best resulting model from RF. This benchmark model has a model MSE of 218725.18., and testing error RMSE of 366.31. RF: RF has a number pf parameters that can be tuned. I create a grid search that captures all combinations of tuning parameters. I use ntrees, max_depth, mtries, and sample_rate as my tuning parameters. The best model has a training MSE of 126,118.6, which is improved compared to the benchmark model. In terms of the performance on the unseen portion of data, the testing error of RMSE is 361.31. The R-squared of the predictions is 36%. Both the RMSE and R2-squared are better than the benchmark RF model but not as good as the Pruned Tree Model. Comparing the Actual versus predicted plot for the Best RF and Pruned Tree, we see that points in the Pruned Tree Model are placed closer to the diagonal line compared to the best Random Forest. GBM: Next Model to investigate is the Gradient Boosting Machines (GBM). Unlike random forest, Gradient Boosting trains several models in a sequential way and adds them gradually. The parameters used for the grid search for my GMB model are learning rate which is the step size by which models are added. Similar to RF it also uses max_depth and sample_rate. Here, instead of mtries in RF,we can use column_sample_size as another way to sample from variables.

While the training MSE is less than that of RFm the testing Error RMSE and prediction R-squared are almost equal.

XGBoost At last, I use the eXtreme Gradient Boosting (XGBoost) model from caret library to fit on training data. caret automates and optimizes required steps for developing a model through arguments of the ‘train’ function and ‘trainControl’ functions. XGBoost also uses the gradient boosting concept. The difference, however, is that it uses a more regularized model formalization to deal with overfitting. I use ‘trainControl’ function to determine the training step. I specify using repeated cross-validation with grid search for 5 times. As you can see in the results, the testing error has remained almost equivalent to that of RF and GBM model. Comparing the actual versus predicted plots, we can find groups of points where a GBM is a better prediction. For instance when Open days is around 3000 days. Also, when a RF provides a better prediction. For instance when Open days is around 1800 days. However as the performance metrics of models indicate, the spread of points in these plot are similar for RF, GBM, and XGBoost too. After trying multiple models, I plot the performance measures for all models to compare their performance. We can see that Tree Model has the lowest test error with a slight difference with the Pruned Tree Model. They both take short amount of time compared to other RF, GBM, and XGboost models. The Pruned Tree Model is the least complex model that provides the best prediction performance. It also can be easily plotted to better understand the branches.

```
closed_df <- subset(df, STATUS_CODE=='L')  # the rest are not terminated yet, 1900-01-01


for (i in names(closed_df)){
  if ("-8" %in% unique(closed_df[[i]])){
    print(i)}

}
```

```
## [1] "DIVERSITY_RESIDENCE"
## [1] "PLAINTIFF"
## [1] "DEFENDANT"
## [1] "FEE_STATUS"
```

```
indx <- apply(closed_df, 2, function(x) any(is.nan(x)| is.na(x) | is.infinite(x)))
names(closed_df)[indx]   # "TRANSFER_DATE"
```

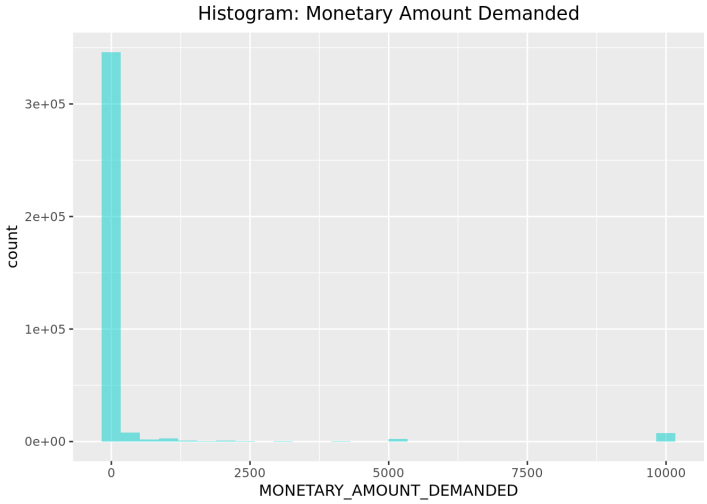
```
## [1] "TRANSFER_DATE"
```

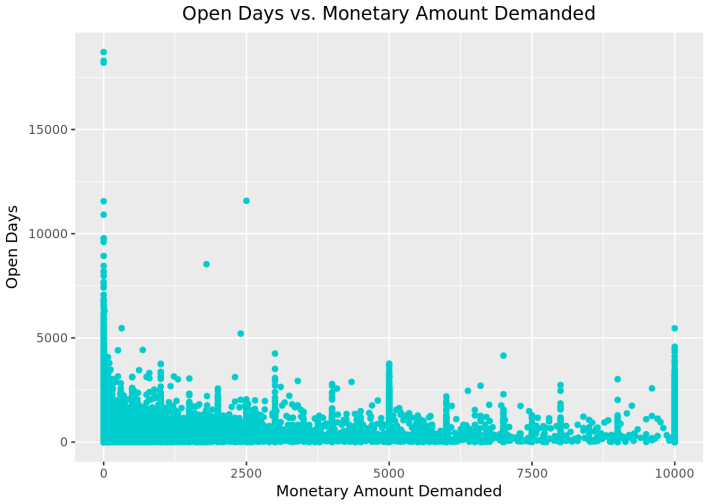
```
closed_df$TRANSFER_DATE_CAT <- ifelse(closed_df$TRANSFER_DATE == as.Date("1931-01-01") |
is.na(closed_df$TRANSFER_DATE), 0, 1)
closed_df$TRANSFER_DATE_CAT <- as.factor(closed_df$TRANSFER_DATE_CAT)


ggplot(closed_df, aes(x=MONETARY_AMOUNT_DEMANDED)) +
  geom_histogram(alpha=0.5, position="identity", fill = 'cyan3') +  ggtitle('Histogram:
 Monetary Amount Demanded')+
  theme(plot.title = element_text(hjust = 0.5))
```

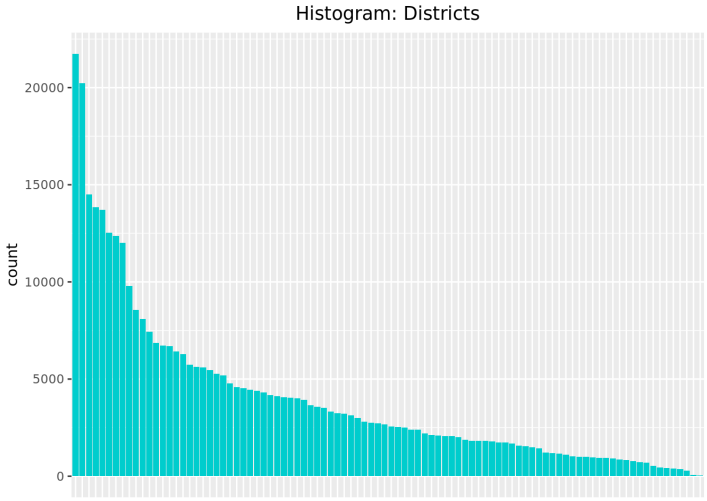
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

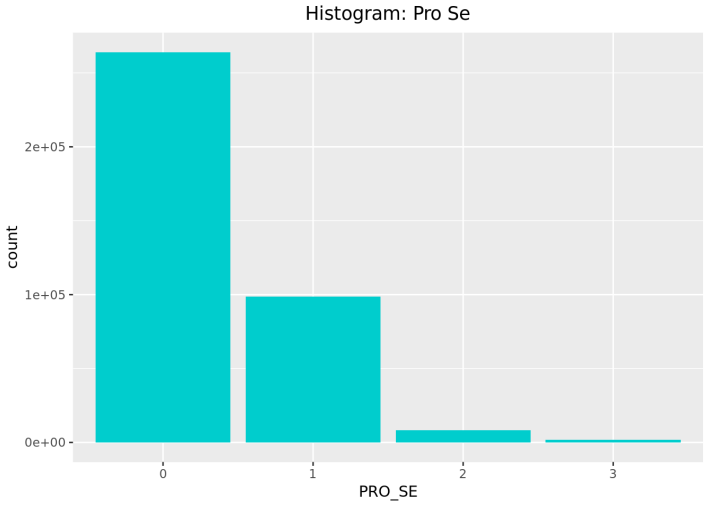
```
ggplot(closed_df, aes(x=MONETARY_AMOUNT_DEMANDED, y=OPEN_DAYS)) + geom_point(color = 'cy
an3')+
  xlab('Monetary Amount Demanded')+ ylab('Open Days') + ggtitle('Open Days vs. Monetary
 Amount Demanded')+
  theme(plot.title = element_text(hjust = 0.5))


closed_df$MONETARY_AMOUNT_DEMANDED_CAT <- ifelse(closed_df$MONETARY_AMOUNT_DEMANDED <= 2
000, 1,
                                                 ifelse(closed_df$MONETARY_AMOUNT_DEMAND
ED <= 8000 , 2,3))

closed_df$MONETARY_AMOUNT_DEMANDED_CAT <- as.factor(closed_df$MONETARY_AMOUNT_DEMANDED_C
AT)


closed_df$DIVERSITY_RESIDENCE <- as.character(closed_df$DIVERSITY_RESIDENCE)
closed_df$DIVERSITY_RESIDENCE_P <- ifelse(closed_df$DIVERSITY_RESIDENCE != "-8", substri
ng(closed_df$DIVERSITY_RESIDENCE, 1,1), "-8")
closed_df$DIVERSITY_RESIDENCE_D <- ifelse(closed_df$DIVERSITY_RESIDENCE != "-8", substri
ng(closed_df$DIVERSITY_RESIDENCE, 2), "-8")
closed_df$DIVERSITY_RESIDENCE_P <- as.factor(closed_df$DIVERSITY_RESIDENCE_P)
closed_df$DIVERSITY_RESIDENCE_D <- as.factor(closed_df$DIVERSITY_RESIDENCE_D)



closed_df$ORIGIN_CAT <- ifelse(closed_df$ORIGIN %in% c("8","9",'10','11','12'), "7",
                               ifelse(closed_df$ORIGIN%in% c("3","7",'13'), '3',closed_d
f$ORIGIN))   # 7: reopened////////// 3: 3,7,13

closed_df$ORIGIN_CAT <- as.factor(closed_df$ORIGIN_CAT)

theTable <- within(closed_df,
                   DISTRICT <- factor(DISTRICT,
                                      levels=names(sort(table(DISTRICT),
                                                        decreasing=TRUE))))

ggplot(theTable, aes(x=DISTRICT)) +
  geom_histogram(fill ="cyan3", stat = 'count') + theme(axis.title.x=element_blank(),
                                                        axis.text.x=element_blank(),
                                                        axis.ticks.x=element_blank()) +
  xlab('District')  + ggtitle('Histogram: Districts')+
  theme(plot.title = element_text(hjust = 0.5))

mergelevels <- function(vec, p=16, newname='other', levs){
  t <- vec
  if (!is.character(t)){
    t <- as.character(t)
  }
  t[t == "-8" | is.na(t) | t == ""] <- "missing"
  t <- gsub("[^[:alnum:] ]", "", t)
  t_levs <- names(sort(table(t), decreasing=T))
  dims <- length(t_levs)
```

```
  if (!missing(levs)) {
    newlevs <- setdiff(t_levs, c(levs, NA_character_))
    if (length(newlevs) > 0)
      ord <- unique(c(levs, newname))
    else
      ord <- levs

  } else if (dims >= p){
    keeplevs <- t_levs[1:p]
    newlevs <- setdiff(t_levs, c(keeplevs, NA_character_))
    ord <- c(sort(keeplevs), newname)
    # make other category as the last level
  } else {
    vec <- factor(t, levels = t_levs)
    return(vec)
  }

  t[t %in% newlevs] <- newname
  vec <- factor(t, levels = ord)
  return(vec)
}

closed_df$DISTRICT_CAT <- mergelevels(closed_df$DISTRICT, p=20, newname='other')
closed_df$DISTRICT_CAT <- as.factor(closed_df$DISTRICT_CAT)

ggplot(theTable, aes(x=PRO_SE)) +
  geom_histogram(fill ="cyan3", stat = 'count') +  ggtitle('Histogram: Pro Se')+
  theme(plot.title = element_text(hjust = 0.5))


closed_df$PRO_SE_CAT <- mergelevels(closed_df$PRO_SE, p=1, newname='1')
closed_df$PRO_SE_CAT <- as.factor(closed_df$PRO_SE_CAT)


closed_df$COUNTY_OF_RESIDENCE_CAT <- ifelse(closed_df$COUNTY_OF_RESIDENCE == "88888", 3,
                                            ifelse(closed_df$COUNTY_OF_RESIDENCE == '999
99' , 4,
                                                   ifelse(closed_df$PLAINTIFF == 'UNITED
STATES OF AMERICA',1,2)))

closed_df$COUNTY_OF_RESIDENCE_CAT <- as.factor(closed_df$COUNTY_OF_RESIDENCE_CAT)
```





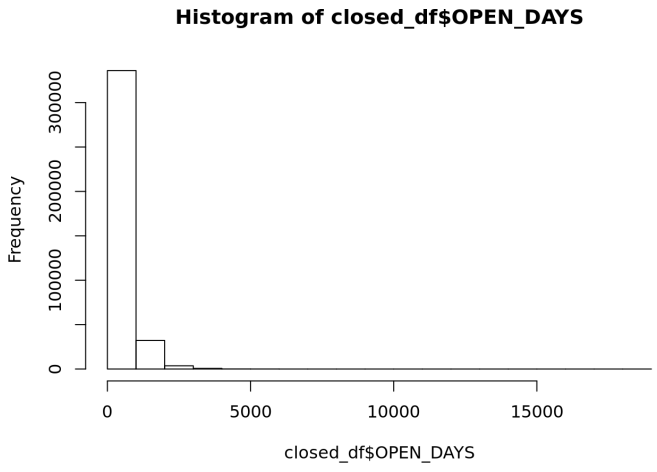





```
# data prepration

keep_var <- c("OPEN_DAYS", "FEE_STATUS", "PRO_SE_CAT", "TRANSFER_DATE_CAT", "MONETARY_AM
OUNT_DEMANDED_CAT",
              "COUNTY_OF_RESIDENCE_CAT",
              "JURY_DEMAND",  "ORIGIN_CAT", "DISTRICT_CAT", "CIRCUIT")

data <- closed_df[, names(closed_df) %in% keep_var]

y_hist <- hist(closed_df$OPEN_DAYS)
print(y_hist)
```

```
## $breaks
##  [1]     0  1000  2000  3000  4000  5000  6000  7000  8000  9000 10000
## [12] 11000 12000 13000 14000 15000 16000 17000 18000 19000
##
## $counts
##  [1] 336100  32184   3607    670     87     32     12      5      5      3
## [11]      1      2      0      0      0      0      0      0      3
##
## $density
##  [1] 9.017711e-04 8.635109e-05 9.677740e-06 1.797639e-06 2.334248e-07
##  [6] 8.585741e-08 3.219653e-08 1.341522e-08 1.341522e-08 8.049132e-09
## [11] 2.683044e-09 5.366088e-09 0.000000e+00 0.000000e+00 0.000000e+00
## [16] 0.000000e+00 0.000000e+00 0.000000e+00 8.049132e-09
##
## $mids
##  [1]   500  1500  2500  3500  4500  5500  6500  7500  8500  9500 10500
## [12] 11500 12500 13500 14500 15500 16500 17500 18500
##
## $xname
## [1] "closed_df$OPEN_DAYS"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```
data <- subset(data, OPEN_DAYS <= 6000)


samp_size <- floor(0.75*nrow(data))
set.seed(1000)

train_ind <- sample(seq_len(nrow(data)), size = samp_size)
train <- data[train_ind,]
test <- data[-train_ind,]
```



```
# linear regression
lm.fit <- lm(OPEN_DAYS ~ ., data = train)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = OPEN_DAYS ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1833.5  -225.5   -92.7   123.3  5632.6
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     527.9329    12.5723  41.992  < 2e-16 ***
## CIRCUIT1                         47.6350     9.4714   5.029 4.92e-07 ***
## CIRCUIT2                         88.7009     9.5944   9.245  < 2e-16 ***
## CIRCUIT3                        -14.1947     9.3800  -1.513 0.130205
## CIRCUIT4                        -45.6821     8.8333  -5.172 2.32e-07 ***
## CIRCUIT5                         24.2462     9.0775   2.671 0.007562 **
## CIRCUIT6                          0.2254     8.9230   0.025 0.979850
## CIRCUIT7                         59.3986     8.8732   6.694 2.17e-11 ***
## CIRCUIT8                          3.4421     8.7703   0.392 0.694711
## CIRCUIT9                        -30.2565     8.7956  -3.440 0.000582 ***
## CIRCUIT10                       -40.9676     9.0001  -4.552 5.32e-06 ***
## CIRCUIT11                        -2.5948     9.1433  -0.284 0.776572
## JURY_DEMANDD                    -33.0434     3.8626  -8.555  < 2e-16 ***
## JURY_DEMANDN                   -238.1198     2.5928 -91.837  < 2e-16 ***
## JURY_DEMANDP                   -200.7343     2.5864 -77.612  < 2e-16 ***
## FEE_STATUSFP                     34.3795     2.2377  15.364  < 2e-16 ***
## TRANSFER_DATE_CAT1              117.1754     7.0325  16.662  < 2e-16 ***
## MONETARY_AMOUNT_DEMANDED_CAT2   161.1423     6.7466  23.885  < 2e-16 ***
## MONETARY_AMOUNT_DEMANDED_CAT3    22.9099     5.4179   4.229 2.35e-05 ***
## ORIGIN_CAT2                    -113.2129     2.4287 -46.615  < 2e-16 ***
## ORIGIN_CAT3                    -373.8604     8.9235 -41.896  < 2e-16 ***
## ORIGIN_CAT4                     -89.2813     4.0877 -21.841  < 2e-16 ***
## ORIGIN_CAT5                     -43.1803     5.8069  -7.436 1.04e-13 ***
## ORIGIN_CAT6                     727.3772     5.0861 143.014  < 2e-16 ***
## ORIGIN_CAT7                    -109.1421    13.1742  -8.285  < 2e-16 ***
## DISTRICT_CAT08                  -57.1725     5.9823  -9.557  < 2e-16 ***
## DISTRICT_CAT12                  -15.1919     9.1759  -1.656 0.097798 .
## DISTRICT_CAT13                  -33.7222     9.1177  -3.699 0.000217 ***
## DISTRICT_CAT20                   -8.0952     9.7035  -0.834 0.404134
## DISTRICT_CAT25                  487.9391     8.4534  57.721  < 2e-16 ***
## DISTRICT_CAT39                  -72.3769     9.5863  -7.550 4.36e-14 ***
## DISTRICT_CAT3A                   35.2869     8.8886   3.970 7.19e-05 ***
## DISTRICT_CAT3C                 -129.6252     8.8779 -14.601  < 2e-16 ***
## DISTRICT_CAT3E                  -58.7376     9.5057  -6.179 6.45e-10 ***
## DISTRICT_CAT3L                  -22.8096     9.9098  -2.302 0.021352 *
## DISTRICT_CAT40                  -59.8133     9.8510  -6.072 1.27e-09 ***
## DISTRICT_CAT41                  -39.7253     9.3305  -4.258 2.07e-05 ***
## DISTRICT_CAT45                    6.3411     9.4186   0.673 0.500784
## DISTRICT_CAT47                  165.9899     9.4536  17.558  < 2e-16 ***
## DISTRICT_CAT52                 -130.0934     8.5306 -15.250  < 2e-16 ***
## DISTRICT_CAT70                  -35.4606     9.5048  -3.731 0.000191 ***
## DISTRICT_CAT71                  -11.6045     9.0276  -1.285 0.198638
## DISTRICT_CAT72                  152.3835     9.3818  16.242  < 2e-16 ***
```

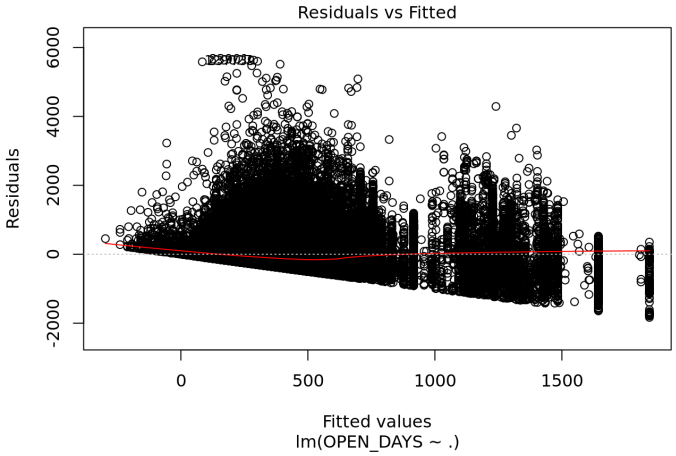
```
## DISTRICT_CAT73                 -59.6494     8.1616  -7.309 2.71e-13 ***
## DISTRICT_CATother               22.0424     6.8012   3.241 0.001191 **
## PRO_SE_CAT1                    -80.9427     1.8748 -43.173  < 2e-16 ***
## COUNTY_OF_RESIDENCE_CAT2        20.2286     6.0841   3.325 0.000885 ***
## COUNTY_OF_RESIDENCE_CAT3       146.9014     6.4705  22.703  < 2e-16 ***
## COUNTY_OF_RESIDENCE_CAT4        60.7758    11.6617   5.212 1.87e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 393 on 279461 degrees of freedom
## Multiple R-squared:  0.2561, Adjusted R-squared:  0.256
## F-statistic:  2005 on 48 and 279461 DF,  p-value: < 2.2e-16
```

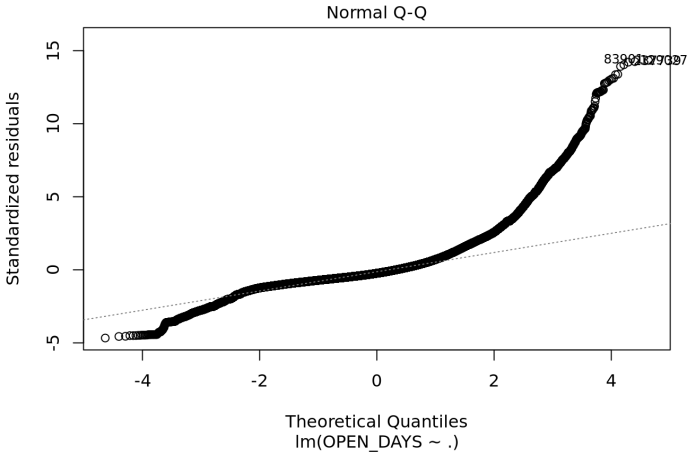
```
plot(lm.fit, 1)
plot(lm.fit, 2)

p <- predict(lm.fit, train[, !(colnames(train) %in% c("OPEN_DAYS"))])
print(c("MSE: ", MSE(p, train$OPEN_DAYS)))
```

```
## [1] "MSE: "           "154389.95116933"
```

```
p <- predict(lm.fit, test[, !(colnames(test) %in% c("OPEN_DAYS"))])


model_performance<-function(name, mod,  pred,actual, aic = TRUE){
  rmse <- RMSE(pred, actual)[[1]]
  r2 <- R2_Score(pred, actual)

  if (aic == FALSE){aic <- NA}
  else{aic <- AIC(mod)}

  out <- data.frame(name, aic, r2, rmse)
  names(out) <- c("Model", "AIC",  "R2" , "RMSE" )
  return(out)
}

lm_perf <- model_performance('lm', lm.fit, p, test$OPEN_DAYS)
print(model_performance('lm', lm.fit, p, test$OPEN_DAYS))
```

```
##   Model     AIC        R2     RMSE
## 1    lm 4132687 0.2610596 388.8285
```






```
# glm: poisson

glm.pois <- glm(OPEN_DAYS ~ ., data = train, family = 'poisson')
summary(glm.pois)
```

```
##
## Call:
## glm(formula = OPEN_DAYS ~ ., family = "poisson", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -76.239  -15.088   -5.831    6.153  174.047
##
## Coefficients:
##                                 Estimate Std. Error   z value Pr(>|z|)
## (Intercept)                    6.2273019  0.0016642  3741.875  < 2e-16 ***
## CIRCUIT1                       0.1115152  0.0011984    93.056  < 2e-16 ***
## CIRCUIT2                       0.2057229  0.0012221   168.333  < 2e-16 ***
## CIRCUIT3                      -0.0474509  0.0012323   -38.506  < 2e-16 ***
## CIRCUIT4                      -0.1509393  0.0011741  -128.552  < 2e-16 ***
## CIRCUIT5                       0.0431849  0.0011769    36.693  < 2e-16 ***
## CIRCUIT6                      -0.0052324  0.0011657    -4.489 7.17e-06 ***
## CIRCUIT7                       0.0893232  0.0011363    78.608  < 2e-16 ***
## CIRCUIT8                       0.0098467  0.0011327     8.693  < 2e-16 ***
## CIRCUIT9                      -0.0967077  0.0011579   -83.522  < 2e-16 ***
## CIRCUIT10                     -0.1278862  0.0011878  -107.668  < 2e-16 ***
## CIRCUIT11                     -0.0121824  0.0011978   -10.171  < 2e-16 ***
## JURY_DEMANDD                  -0.0066091  0.0004349   -15.198  < 2e-16 ***
## JURY_DEMANDN                  -0.5430735  0.0003071 -1768.343  < 2e-16 ***
## JURY_DEMANDP                  -0.4079980  0.0002907 -1403.712  < 2e-16 ***
## FEE_STATUSFP                   0.1184060  0.0003189   371.343  < 2e-16 ***
## TRANSFER_DATE_CAT1             0.2955872  0.0008408   351.560  < 2e-16 ***
## MONETARY_AMOUNT_DEMANDED_CAT2  0.3541641  0.0007354   481.576  < 2e-16 ***
## MONETARY_AMOUNT_DEMANDED_CAT3  0.0582572  0.0006994    83.297  < 2e-16 ***
## ORIGIN_CAT2                   -0.3141905  0.0003486  -901.401  < 2e-16 ***
## ORIGIN_CAT3                   -0.7752510  0.0012660  -612.359  < 2e-16 ***
## ORIGIN_CAT4                   -0.2750189  0.0006039  -455.385  < 2e-16 ***
## ORIGIN_CAT5                   -0.0859245  0.0007298  -117.734  < 2e-16 ***
## ORIGIN_CAT6                    0.7762412  0.0003932  1974.087  < 2e-16 ***
## ORIGIN_CAT7                   -0.3947097  0.0021551  -183.148  < 2e-16 ***
## DISTRICT_CAT08                -0.1431856  0.0007400  -193.499  < 2e-16 ***
## DISTRICT_CAT12                -0.0676247  0.0011941   -56.633  < 2e-16 ***
## DISTRICT_CAT13                -0.0443233  0.0011547   -38.384  < 2e-16 ***
## DISTRICT_CAT20                -0.0123623  0.0012682    -9.748  < 2e-16 ***
## DISTRICT_CAT25                 0.7762046  0.0010281   754.962  < 2e-16 ***
## DISTRICT_CAT39                -0.2494662  0.0013037  -191.353  < 2e-16 ***
## DISTRICT_CAT3A                 0.0819000  0.0011279    72.613  < 2e-16 ***
## DISTRICT_CAT3C                -0.5345758  0.0012338  -433.273  < 2e-16 ***
## DISTRICT_CAT3E                -0.2346392  0.0012884  -182.121  < 2e-16 ***
## DISTRICT_CAT3L                -0.0608813  0.0012465   -48.843  < 2e-16 ***
## DISTRICT_CAT40                -0.1703987  0.0012843  -132.676  < 2e-16 ***
## DISTRICT_CAT41                -0.1234680  0.0012171  -101.443  < 2e-16 ***
## DISTRICT_CAT45                -0.0003209  0.0012168    -0.264    0.792
## DISTRICT_CAT47                 0.3474434  0.0011168   311.105  < 2e-16 ***
## DISTRICT_CAT52                -0.2780820  0.0010542  -263.792  < 2e-16 ***
## DISTRICT_CAT70                -0.1690539  0.0013285  -127.247  < 2e-16 ***
## DISTRICT_CAT71                -0.0587537  0.0011854   -49.564  < 2e-16 ***
## DISTRICT_CAT72                 0.4062868  0.0011722   346.616  < 2e-16 ***
```

```
## DISTRICT_CAT73                -0.2352883  0.0010727  -219.347  < 2e-16 ***
## DISTRICT_CATother              0.0482715  0.0008301    58.149  < 2e-16 ***
## PRO_SE_CAT1                   -0.2672218  0.0002759  -968.721  < 2e-16 ***
## COUNTY_OF_RESIDENCE_CAT2       0.0687185  0.0009166    74.974  < 2e-16 ***
## COUNTY_OF_RESIDENCE_CAT3       0.3704747  0.0009473   391.084  < 2e-16 ***
## COUNTY_OF_RESIDENCE_CAT4       0.1938629  0.0015457   125.420  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 114484618  on 279509  degrees of freedom
## Residual deviance:  84876076  on 279461  degrees of freedom
## AIC: 86851400
##
## Number of Fisher Scoring iterations: 6
```

```
p_train <- predict(glm.pois, train[, !(colnames(train) %in% c("OPEN_DAYS"))], type = 're
sponse')
R2_Score(p_train, train$OPEN_DAYS)
```

```
## [1] 0.2415282
```

```
p <- predict(glm.pois, train[, !(colnames(train) %in% c("OPEN_DAYS"))])
print(c("MSE: ", MSE(p, train$OPEN_DAYS)))
```

```
## [1] "MSE: "            "359308.134361602"
```

```
p_test <- predict(glm.pois, test[, !(colnames(test) %in% c("OPEN_DAYS"))], type = 'respo
nse')

glm.pois_perf <- model_performance('glm.pois', glm.pois,  p_test, test$OPEN_DAYS)
glm.pois_perf
```

```
##      Model      AIC       R2    RMSE
## 1 glm.pois 86851400 0.245319 392.948
```

```
1 - pchisq(summary(glm.pois)$deviance, summary(glm.pois)$df.residual)
```

```
## [1] 0
```

```
dispersiontest(glm.pois)
```

```
##
##  Overdispersion test
##
## data:  glm.pois
## z = 119.46, p-value < 2.2e-16
## alternative hypothesis: true dispersion is greater than 1
## sample estimates:
## dispersion
##   384.5029
```

```
# glm.nbinom <- glm.nb(OPEN_DAYS ~ ., data = train)
# saveRDS(glm.nbinom, 'glm.nbinom.rds')
glm.nbinom <- readRDS('glm.nbinom.rds')
summary(glm.nbinom)
```

```
##
## Call:
## glm.nb(formula = OPEN_DAYS ~ ., data = train, init.theta = 0.9437813015,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.7072  -0.9843  -0.3197   0.3014   7.5905
##
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     6.293749   0.032978 190.848  < 2e-16 ***
## CIRCUIT1                        0.060148   0.024842   2.421 0.015466 *
## CIRCUIT2                        0.174200   0.025164   6.923 4.43e-12 ***
## CIRCUIT3                       -0.068625   0.024603  -2.789 0.005283 **
## CIRCUIT4                       -0.154274   0.023170  -6.658 2.77e-11 ***
## CIRCUIT5                       -0.004491   0.023809  -0.189 0.850396
## CIRCUIT6                       -0.045415   0.023405  -1.940 0.052328 .
## CIRCUIT7                        0.002056   0.023273   0.088 0.929602
## CIRCUIT8                       -0.122733   0.023004  -5.335 9.54e-08 ***
## CIRCUIT9                       -0.121374   0.023071  -5.261 1.43e-07 ***
## CIRCUIT10                      -0.101011   0.023607  -4.279 1.88e-05 ***
## CIRCUIT11                      -0.027335   0.023983  -1.140 0.254368
## JURY_DEMANDD                   -0.028365   0.010128  -2.801 0.005101 **
## JURY_DEMANDN                   -0.570636   0.006800 -83.923  < 2e-16 ***
## JURY_DEMANDP                   -0.468642   0.006782 -69.099  < 2e-16 ***
## FEE_STATUSFP                    0.106533   0.005871  18.145  < 2e-16 ***
## TRANSFER_DATE_CAT1              0.328582   0.018442  17.817  < 2e-16 ***
## MONETARY_AMOUNT_DEMANDED_CAT2   0.335834   0.017690  18.984  < 2e-16 ***
## MONETARY_AMOUNT_DEMANDED_CAT3 -0.012014   0.014214  -0.845 0.397988
## ORIGIN_CAT2                    -0.271052   0.006373 -42.533  < 2e-16 ***
## ORIGIN_CAT3                    -0.563858   0.023411 -24.085  < 2e-16 ***
## ORIGIN_CAT4                    -0.202322   0.010727 -18.862  < 2e-16 ***
## ORIGIN_CAT5                    -0.008582   0.015231  -0.563 0.573119
## ORIGIN_CAT6                     1.019424   0.013329  76.481  < 2e-16 ***
## ORIGIN_CAT7                    -0.304942   0.034583  -8.818  < 2e-16 ***
## DISTRICT_CAT08                 -0.113189   0.015689  -7.214 5.41e-13 ***
## DISTRICT_CAT12                 -0.119225   0.024068  -4.954 7.28e-07 ***
## DISTRICT_CAT13                 -0.244641   0.023915 -10.229  < 2e-16 ***
## DISTRICT_CAT20                 -0.021235   0.025453  -0.834 0.404109
## DISTRICT_CAT25                  0.818626   0.022169  36.927  < 2e-16 ***
## DISTRICT_CAT39                 -0.231987   0.025147  -9.225  < 2e-16 ***
## DISTRICT_CAT3A                 -0.013817   0.023313  -0.593 0.553401
## DISTRICT_CAT3C                 -0.551892   0.023290 -23.696  < 2e-16 ***
## DISTRICT_CAT3E                 -0.270706   0.024936 -10.856  < 2e-16 ***
## DISTRICT_CAT3L                 -0.094242   0.025992  -3.626 0.000288 ***
## DISTRICT_CAT40                 -0.115204   0.025839  -4.459 8.25e-06 ***
## DISTRICT_CAT41                 -0.101049   0.024473  -4.129 3.64e-05 ***
## DISTRICT_CAT45                 -0.012992   0.024704  -0.526 0.598948
## DISTRICT_CAT47                  0.318737   0.024791  12.857  < 2e-16 ***
## DISTRICT_CAT52                 -0.229336   0.022373 -10.250  < 2e-16 ***
## DISTRICT_CAT70                 -0.186487   0.024936  -7.479 7.50e-14 ***
## DISTRICT_CAT71                 -0.085570   0.023680  -3.614 0.000302 ***
```

```
## DISTRICT_CAT72                0.373771   0.024605  15.191  < 2e-16 ***
## DISTRICT_CAT73               -0.240017   0.021408 -11.212  < 2e-16 ***
## DISTRICT_CATother             0.034296   0.017836   1.923 0.054502 .
## PRO_SE_CAT1                  -0.269614   0.004920 -54.804  < 2e-16 ***
## COUNTY_OF_RESIDENCE_CAT2      0.085929   0.015966   5.382 7.36e-08 ***
## COUNTY_OF_RESIDENCE_CAT3      0.316963   0.016978  18.669  < 2e-16 ***
## COUNTY_OF_RESIDENCE_CAT4      0.187268   0.030590   6.122 9.25e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(0.9438) family taken to be 1)
##
##     Null deviance: 389306  on 279509  degrees of freedom
## Residual deviance: 326571  on 279461  degrees of freedom
## AIC: 3836056
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  0.94378
##           Std. Err.:  0.00227
##
##  2 x log-likelihood:  -3835955.56200
```

```
p_train <- predict(glm.nbinom, train[, !(colnames(train) %in% c("OPEN_DAYS"))], type =
'response')
R2_Score(p_train, train$OPEN_DAYS)
```

```
## [1] 0.1996526
```

```
p <- predict(glm.nbinom, train[, !(colnames(train) %in% c("OPEN_DAYS"))])
print(c("MSE: ", MSE(p, train$OPEN_DAYS)))
```

```
## [1] "MSE: "           "359305.16109886"
```

```
p_test <- predict(glm.nbinom, test[, !(colnames(test) %in% c("OPEN_DAYS"))], type = 'res
ponse')

glm.nbinom_perf <- model_performance('glm.nbinom', glm.nbinom, p_test, test$OPEN_DAYS )

print(glm.nbinom_perf)
```

```
##        Model     AIC        R2     RMSE
## 1 glm.nbinom 3836056 0.2012613 404.2553
```

```
1 - pchisq(summary(glm.nbinom)$deviance, summary(glm.nbinom)$df.residual)
```

```
## [1] 0
```

```
# zero-inflated neg binomial

# glm.nb.inf <- zeroinfl(OPEN_DAYS ~ ., data = train, dist="negbin")
# saveRDS(glm.nb.inf, "glm.nb.inf.rds")
glm.nb.inf <- readRDS("glm.nb.inf.rds")
summary(glm.nb.inf)   # log(theta) is significant, an indication for the presence of dis
persion.
```

```
##
## Call:
## zeroinfl(formula = OPEN_DAYS ~ ., data = train, dist = "negbin")
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -1.0095 -0.6994 -0.2952  0.3444 27.9674
##
## Count model coefficients (negbin with log link):
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     6.390152   0.032430 197.042  < 2e-16 ***
## CIRCUIT1                       -0.039055   0.024965  -1.564 0.117731
## CIRCUIT2                        0.070679   0.025308   2.793 0.005227 **
## CIRCUIT3                       -0.170143   0.024754  -6.873 6.27e-12 ***
## CIRCUIT4                       -0.254545   0.023454 -10.853  < 2e-16 ***
## CIRCUIT5                       -0.104799   0.024012  -4.364 1.27e-05 ***
## CIRCUIT6                       -0.134210   0.023669  -5.670 1.43e-08 ***
## CIRCUIT7                       -0.100063   0.023548  -4.249 2.14e-05 ***
## CIRCUIT8                       -0.220890   0.023287  -9.485  < 2e-16 ***
## CIRCUIT9                       -0.226264   0.023350  -9.690  < 2e-16 ***
## CIRCUIT10                      -0.200267   0.023840  -8.400  < 2e-16 ***
## CIRCUIT11                      -0.126494   0.024153  -5.237 1.63e-07 ***
## JURY_DEMANDD                   -0.031693   0.009838  -3.222 0.001275 **
## JURY_DEMANDN                   -0.567085   0.006511 -87.103  < 2e-16 ***
## JURY_DEMANDP                   -0.460042   0.006573 -69.988  < 2e-16 ***
## FEE_STATUSFP                    0.107904   0.005609  19.237  < 2e-16 ***
## MONETARY_AMOUNT_DEMANDED_CAT2   0.333429   0.017267  19.311  < 2e-16 ***
## MONETARY_AMOUNT_DEMANDED_CAT3  -0.008749   0.013635  -0.642 0.521106
## TRANSFER_DATE_CAT1              0.326279   0.017964  18.163  < 2e-16 ***
## ORIGIN_CAT2                    -0.272649   0.006159 -44.268  < 2e-16 ***
## ORIGIN_CAT3                    -0.565834   0.022757 -24.864  < 2e-16 ***
## ORIGIN_CAT4                     0.009220   0.011392   0.809 0.418329
## ORIGIN_CAT5                    -0.017086   0.014678  -1.164 0.244389
## ORIGIN_CAT6                     0.984936   0.013091  75.235  < 2e-16 ***
## ORIGIN_CAT7                     0.012554   0.039420   0.318 0.750124
## DISTRICT_CAT08                 -0.117656   0.015105  -7.789 6.73e-15 ***
## DISTRICT_CAT12                 -0.080453   0.023355  -3.445 0.000572 ***
## DISTRICT_CAT13                 -0.136855   0.023406  -5.847 5.01e-09 ***
## DISTRICT_CAT20                 -0.020801   0.024524  -0.848 0.396334
## DISTRICT_CAT25                  0.807384   0.021525  37.508  < 2e-16 ***
## DISTRICT_CAT39                 -0.224521   0.024304  -9.238  < 2e-16 ***
## DISTRICT_CAT3A                 -0.013653   0.022618  -0.604 0.546076
## DISTRICT_CAT3C                 -0.555141   0.022478 -24.697  < 2e-16 ***
## DISTRICT_CAT3E                 -0.275398   0.024016 -11.467  < 2e-16 ***
## DISTRICT_CAT3L                 -0.104326   0.025071  -4.161 3.16e-05 ***
## DISTRICT_CAT40                 -0.113631   0.024957  -4.553 5.28e-06 ***
## DISTRICT_CAT41                 -0.099907   0.023629  -4.228 2.36e-05 ***
## DISTRICT_CAT45                 -0.028714   0.023851  -1.204 0.228628
## DISTRICT_CAT47                  0.289260   0.023910  12.098  < 2e-16 ***
## DISTRICT_CAT52                 -0.231442   0.021608 -10.711  < 2e-16 ***
## DISTRICT_CAT70                 -0.184786   0.023924  -7.724 1.13e-14 ***
## DISTRICT_CAT71                 -0.088776   0.022885  -3.879 0.000105 ***
## DISTRICT_CAT72                  0.374417   0.023735  15.775  < 2e-16 ***
```

```
## DISTRICT_CAT73                -0.240913   0.020626 -11.680  < 2e-16 ***
## DISTRICT_CATother              0.032317   0.017200   1.879 0.060259 .
## PRO_SE_CAT1                   -0.269895   0.004652 -58.014  < 2e-16 ***
## COUNTY_OF_RESIDENCE_CAT2       0.084428   0.015375   5.491 3.99e-08 ***
## COUNTY_OF_RESIDENCE_CAT3       0.325229   0.016470  19.747  < 2e-16 ***
## COUNTY_OF_RESIDENCE_CAT4       0.183698   0.029616   6.203 5.55e-10 ***
## Log(theta)                     0.019633   0.002497   7.863 3.74e-15 ***
##
## Zero-inflation model coefficients (binomial with logit link):
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    -3.65319    0.48322  -7.560 4.03e-14 ***
## CIRCUIT1                       -6.04843    0.67807  -8.920  < 2e-16 ***
## CIRCUIT2                       -4.35430    0.25614 -17.000  < 2e-16 ***
## CIRCUIT3                       -3.81878    0.17641 -21.648  < 2e-16 ***
## CIRCUIT4                       -4.48474    0.24177 -18.549  < 2e-16 ***
## CIRCUIT5                       -4.05912    0.22156 -18.321  < 2e-16 ***
## CIRCUIT6                       -2.44024    0.14564 -16.755  < 2e-16 ***
## CIRCUIT7                       -4.80683    0.28315 -16.976  < 2e-16 ***
## CIRCUIT8                       -4.99530    0.31462 -15.877  < 2e-16 ***
## CIRCUIT9                       -4.47246    0.20048 -22.309  < 2e-16 ***
## CIRCUIT10                      -4.37838    0.26323 -16.633  < 2e-16 ***
## CIRCUIT11                      -3.98308    0.20960 -19.004  < 2e-16 ***
## JURY_DEMANDD                   -0.18159    0.12716  -1.428 0.153289
## JURY_DEMANDN                    0.70433    0.09196   7.659 1.87e-14 ***
## JURY_DEMANDP                    0.95816    0.09245  10.364  < 2e-16 ***
## FEE_STATUSFP                   -0.01252    0.07032  -0.178 0.858660
## MONETARY_AMOUNT_DEMANDED_CAT2  -1.39224    0.38804  -3.588 0.000333 ***
## MONETARY_AMOUNT_DEMANDED_CAT3   0.40810    0.16314   2.502 0.012364 *
## TRANSFER_DATE_CAT1             -0.50579    0.33861  -1.494 0.135252
## ORIGIN_CAT2                    -8.01644   26.97777  -0.297 0.766352
## ORIGIN_CAT3                    -3.04146   48.45340  -0.063 0.949949
## ORIGIN_CAT4                     4.37075    0.07738  56.481  < 2e-16 ***
## ORIGIN_CAT5                    -1.73436    1.65518  -1.048 0.294713
## ORIGIN_CAT6                    -9.75567   31.66954  -0.308 0.758047
## ORIGIN_CAT7                     4.34380    0.11226  38.694  < 2e-16 ***
## DISTRICT_CAT08                 -0.24621    0.38343  -0.642 0.520782
## DISTRICT_CAT12                  1.94413    0.41645   4.668 3.04e-06 ***
## DISTRICT_CAT13                  2.44488    0.41450   5.898 3.67e-09 ***
## DISTRICT_CAT20                 -0.57185    0.89169  -0.641 0.521318
## DISTRICT_CAT25                 -0.86079    0.62320  -1.381 0.167210
## DISTRICT_CAT39                  1.56747    0.46628   3.362 0.000775 ***
## DISTRICT_CAT3A                  0.47056    0.45594   1.032 0.302040
## DISTRICT_CAT3C                  0.12416    0.46939   0.265 0.791384
## DISTRICT_CAT3E                 -0.07229    0.53205  -0.136 0.891919
## DISTRICT_CAT3L                 -0.66925    0.67691  -0.989 0.322819
## DISTRICT_CAT40                  1.12874    0.50173   2.250 0.024468 *
## DISTRICT_CAT41                  0.91490    0.50091   1.826 0.067777 .
## DISTRICT_CAT45                 -1.17495    0.45720  -2.570 0.010173 *
## DISTRICT_CAT47                -10.05728   64.16302  -0.157 0.875445
## DISTRICT_CAT52                  0.03742    0.55617   0.067 0.946363
## DISTRICT_CAT70                 -0.70990    0.70181  -1.012 0.311765
## DISTRICT_CAT71                 -0.62929    0.60052  -1.048 0.294686
## DISTRICT_CAT72                 -0.82768    0.68827  -1.203 0.229147
## DISTRICT_CAT73                  0.11431    0.45410   0.252 0.801255
```

```
## DISTRICT_CATother               0.30108    0.38790   0.776 0.437639
## PRO_SE_CAT1                     0.31693    0.06212   5.102 3.36e-07 ***
## COUNTY_OF_RESIDENCE_CAT2       -0.02069    0.24603  -0.084 0.932979
## COUNTY_OF_RESIDENCE_CAT3        0.22848    0.25133   0.909 0.363301
## COUNTY_OF_RESIDENCE_CAT4        0.27126    0.39165   0.693 0.488553
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Theta = 1.0198
## Number of iterations in BFGS optimization: 109
## Log-likelihood: -1.909e+06 on 99 Df
```

```
p_train <- predict(glm.nb.inf, train[, !(colnames(train) %in% c("OPEN_DAYS"))], type =
'response')
R2_Score(p_train, train$OPEN_DAYS)
```

```
## [1] 0.2087775
```

```
print(c("MSE: ", MSE(p_train, train$OPEN_DAYS)))
```

```
## [1] "MSE: "            "164216.738160355"
```

```
p_test <- predict(glm.nb.inf, test[, !(colnames(test) %in% c("OPEN_DAYS"))], type = 'res
ponse')
glm.nb.inf_perf <- model_performance('glm.nb.inf', glm.nb.inf, p_test, test$OPEN_DAYS)
glm.nb.inf_perf
```

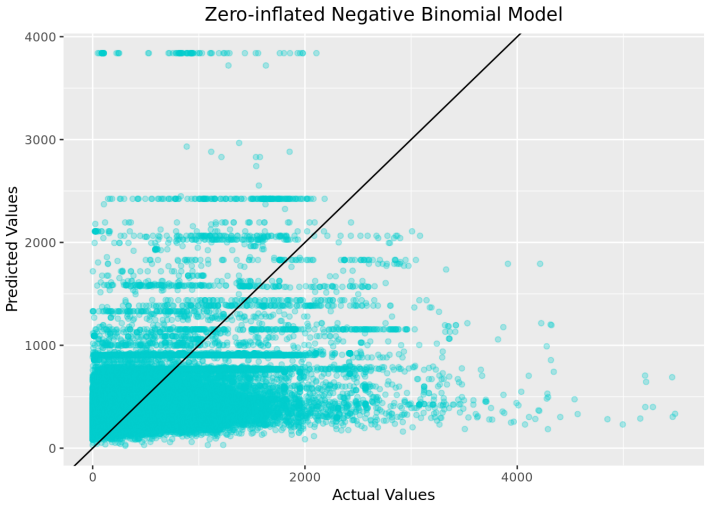
```
##        Model     AIC        R2     RMSE
## 1 glm.nb.inf 3817377 0.2102241 401.9808
```

```
glm.nb.inf0 <- update(glm.nb.inf, . ~ 1)
pchisq(2 * (logLik(glm.nb.inf) - logLik(glm.nb.inf0)), df = 73, lower.tail=FALSE)
```

```
## 'log Lik.' 0 (df=99)
```

```
# comparing neg binomial with zero inflated neg binomial

# vuong(glm.nb.inf, glm.nbinom)

# Vuong Non-Nested Hypothesis Test-Statistic:
#   (test-statistic is asymptotically distributed N(0,1) under the
#    null that the models are indistinguishible)
# -------------------------------------------------------------
#   Vuong z-statistic             H_A    p-value
# Raw                    38.51545 model1 > model2 < 2.22e-16
# AIC-corrected          38.18810 model1 > model2 < 2.22e-16
# BIC-corrected          36.51017 model1 > model2 < 2.22e-16
```

```
res_df <- as.data.frame(cbind(test$OPEN_DAYS, p_test))

names(res_df) <- c('Actual', 'Predicted')

p <- ggplot(res_df, aes(Actual, Predicted))
p <- p + geom_point(color = 'cyan3', alpha=0.3) + ggtitle('Zero-inflated Negative Binomi
al Model') + xlab('Actual Values')+ylab('Predicted Values') +
  theme(plot.title = element_text(hjust = 0.5))
p <- p + geom_abline(intercept = 0, slope = 1)
p
```




```
# tree <- rpart(OPEN_DAYS~.,data[train_ind,], control = rpart.control(cp = 0))
# saveRDS(tree, "tree.rds")
tree <- readRDS("tree.rds")
# summary(tree)
tree.pred <- predict(tree, test)

tree_perf <- model_performance("tree", tree, tree.pred, test$OPEN_DAYS, aic = FALSE)
tree_perf
```

```
##   Model AIC       R2    RMSE
## 1  tree  NA 0.383936 355.031
```

```
print(c("MSE: ", MSE(predict(tree, train), train$OPEN_DAYS)))
```

```
## [1] "MSE: "            "129890.684037165"
```

```
plotcp(tree)


# pruned_tree <- prune(tree, cp = tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"])
# saveRDS(pruned_tree, "pruned_tree.rds")
pruned_tree <- readRDS("pruned_tree.rds")
rpart.plot(pruned_tree, main="Pruned Decision Tree", fallen.leaves=FALSE,  box.palette="GnBu", roundint=FALSE)
pruned_tree.pred <- predict(pruned_tree,test)#,type="class")
R2_Score(pruned_tree.pred, test$OPEN_DAYS)
```

```
## [1] 0.3822592
```

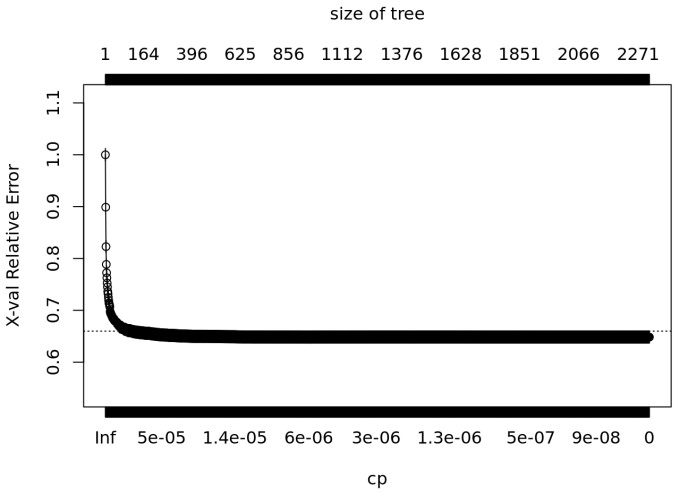
```
pruned_tree_perf <- model_performance("pruned_tree", pruned_tree, pruned_tree.pred, test$OPEN_DAYS, aic = FALSE)
print(pruned_tree_perf)
```

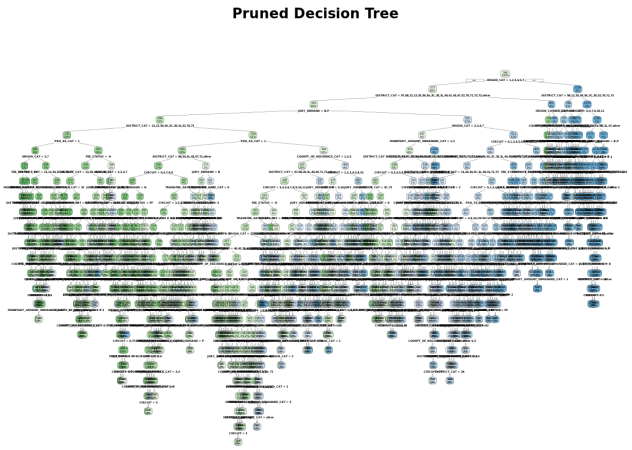
```
##         Model AIC        R2     RMSE
## 1 pruned_tree  NA 0.3822592 355.5138
```

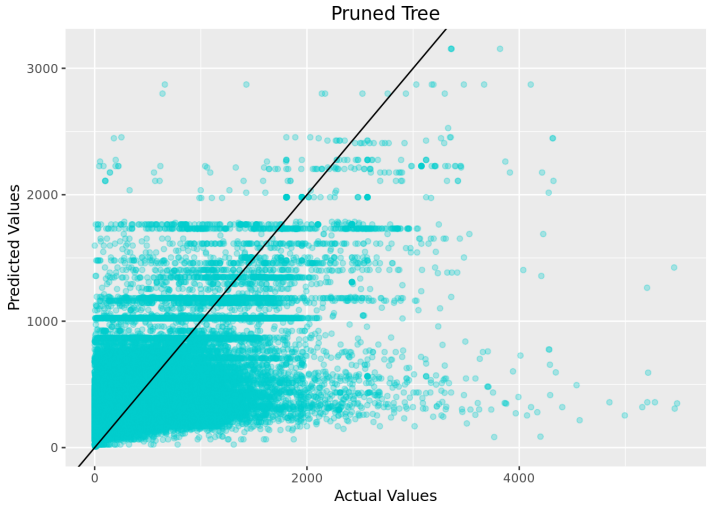
```
print(c("MSE: ", MSE(predict(pruned_tree, train), train$OPEN_DAYS)))
```

```
## [1] "MSE: "            "130258.066689209"
```

```
# -----------------prediction vs Actual
# tree
predict_tree <-as.data.frame(predict(pruned_tree, test))
response <- as.data.frame(test$OPEN_DAYS)
res_df <- as.data.frame(cbind(response$`test$OPEN_DAYS`, predict_tree$`predict(pruned_tree, test)`))
colnames(res_df) <- c('Actual', 'Predicted')
#colnames(res_df)[colnames(res_df)=="as.numeric(as.vector(splits[[2]]$OPEN_DAYS))"]    <- "response"
# lm.model <- lm(Actual ~ Predicted, data = res_df)
# res_df <- cbind(res_df, predict(lm.model, interval = 'confidence'))
p <- ggplot(res_df, aes(Actual, Predicted))
p <- p + geom_point(color = 'cyan3', alpha=0.3) + ggtitle('Pruned Tree') + xlab('Actual Values')+ylab('Predicted Values') +
  theme(plot.title = element_text(hjust = 0.5))
p <- p + geom_abline(intercept = 0, slope = 1)
p
```








```
# ################################################################ Ensemble Learning
# initialize h2o session
h2o.no_progress()
h2o.init()
```

```
##
## H2O is not running yet, starting it now...
##
## Note:  In case of errors look at the following log files:
##     /tmp/RtmpQ3WsKz/h2o_rstudio_user_started_from_r.out
##     /tmp/RtmpQ3WsKz/h2o_rstudio_user_started_from_r.err
##
##
## Starting H2O JVM and connecting: . Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         1 seconds 717 milliseconds
##     H2O cluster timezone:       Etc/UTC
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.22.1.1
##     H2O cluster version age:    5 months and 6 days !!!
##     H2O cluster name:           H2O_started_from_R_rstudio-user_mjk343
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   26.67 GB
##     H2O cluster total cores:    64
##     H2O cluster allowed cores:  64
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         XGBoost, Algos, AutoML, Core V3, Core V4
##     R Version:                  R version 3.4.3 (2017-11-30)
```

```
data.h2o <- as.h2o(data)
splits <- h2o.splitFrame(data.h2o,c(0.75),seed=1000)

htrain <- h2o.assign(splits[[1]], "train.hex")
hvalid <- h2o.assign(splits[[2]], "valid.hex")
# ------------------------------------
y <- "OPEN_DAYS"
x <- setdiff(names(data), y)

rf1 <- h2o.randomForest(training_frame = htrain, validation_frame = hvalid, x=x, y=y, nt
rees = 1000,
                        stopping_rounds = 10,stopping_metric = "RMSE", seed = 1000000, k
eep_cross_validation_models= TRUE,
                        keep_cross_validation_predictions=TRUE, score_each_iteration = T
RUE)
# summary(rf1)
perf_rf1 <- h2o.performance(rf1, hvalid)
print(perf_rf1)
```

```
## H2ORegressionMetrics: drf
##
## MSE:  131019.4
## RMSE:  361.966
## MAE:  243.4492
## RMSLE:  1.393966
## Mean Residual Deviance :  131019.4
```

```
# ########################################## grid search RF
# rf_params1 <- list(ntrees = c(50, 100, 300, 500, 1000), max_depth = c(10,20,30,40,50),
mtries= seq(2, 10, by = 2),
#                    sample_rate = c(.55, .70, .80))
# search_criteria <- list(  strategy = "RandomDiscrete",  stopping_metric = "rmse",  sto
pping_tolerance = 0.005,
#   stopping_rounds = 2,  max_runtime_secs = 30*60)
# h2o.grid(algorithm ="randomForest", grid_id = "rf_grid_id", x = x, y = y, training_fra
me = htrain,
#          validation_frame = hvalid, seed =1,search_criteria = search_criteria, hyper_p
arams =rf_params1)
#
# rf_gridperf <- h2o.getGrid(grid_id = "rf_grid_id",sort_by = "rmse", decreasing = FALS
E)
#
# best_rf <- h2o.getModel(rf_gridperf@model_ids[[1]])
#
# h2o.saveModel(best_rf, 'best_rf_final')

best_rf <- h2o.loadModel("best_rf_final/DRF_model_R_1559420409492_4")
response <- as.data.frame(as.numeric(as.vector(splits[[2]]$OPEN_DAYS)))



best_rf@parameters$mtries
```

```
## [1] 4
```

```
print(best_rf@model[["model_summary"]])
```

```
## Model Summary:
##   number_of_trees number_of_internal_trees model_size_in_bytes min_depth
## 1             100                      100             3340348        20
##   max_depth mean_depth min_leaves max_leaves mean_leaves
## 1        20   20.00000       2436       2801  2656.41000
```

```
predict_rf <-as.data.frame(h2o.predict(best_rf, htrain))

print(c("MSE: ", MSE(predict_rf$predict, as.numeric(as.vector(splits[[1]]$OPEN_DAYS)))))
```

```
## [1] "MSE: "            "126118.627492842"
```

```
predict_rf <-as.data.frame(h2o.predict(best_rf, hvalid))
best_rf_perf <- model_performance("best.rf", best_rf, predict_rf, response$`as.numeric(a
s.vector(splits[[2]]$OPEN_DAYS))`, aic=FALSE)
print(best_rf_perf)
```

```
##     Model AIC        R2     RMSE
## 1 best.rf  NA 0.3675912 361.3155
```

```
# ------------------------------prediction vs. actual
res_df <- as.data.frame(cbind(response$`as.numeric(as.vector(splits[[2]]$OPEN_DAYS))`, p
redict_rf$predict))
colnames(res_df) <- c('Actual', 'Predicted')
#colnames(res_df)[colnames(res_df)=="as.numeric(as.vector(splits[[2]]$OPEN_DAYS))"]    <-
 "response"
# lm.model <- lm(Actual ~ Predicted, data = res_df)
# res_df <- cbind(res_df, predict(lm.model, interval = 'confidence'))
p <- ggplot(res_df, aes(Actual, Predicted))
p <- p + geom_point(color = 'cyan3', alpha=0.3) + ggtitle('Random Forest') + xlab('Actua
l Values')+ylab('Predicted Values') +
  theme(plot.title = element_text(hjust = 0.5))
p <- p + geom_abline(intercept = 0, slope = 1)
p
```

```
# ##############################################################################
# ############################## GBM
# gbm <- h2o.gbm(training_frame = htrain, validation_frame = hvalid, x=x, y=y,
#                ntrees=500, learn_rate=0.05, score_each_iteration = TRUE)
#
# h2o.saveModel(gbm, 'gbm')
gbm <- h2o.loadModel('gbm/GBM_model_R_1559512812192_2')

response <- as.data.frame(as.numeric(as.vector(splits[[2]]$OPEN_DAYS)))


plot(gbm, timestep = "number_of_trees", metric = "rmse")

gbm_pred <- h2o.predict(gbm, hvalid)

gbm_perf <- h2o.performance(model = gbm,
                            newdata = hvalid)

print(gbm_perf)
```

```
## H2ORegressionMetrics: gbm
##
## MSE:  133350.6
## RMSE:  365.172
## MAE:  246.7905
## RMSLE:  NaN
## Mean Residual Deviance :  133350.6
```

```
# ############################## GBM grid

# gbm_params1 <- list(learn_rate = c(0.01, 0.05, 0.1),
#                     max_depth = c(3, 5, 9),
#                     sample_rate = c(0.8, 1.0),
#                     col_sample_rate = c(0.2, 0.5, 1.0))

# Train and validate a cartesian grid of GBMs
# gbm_grid1 <- h2o.grid("gbm", x = x, y = y,
#                       grid_id = "gbm_grid1",
#                       training_frame = htrain,
#                       validation_frame = hvalid,
#                       ntrees = 200,
#                       seed = 1,
#                       hyper_params = gbm_params1)
#
# # Get the grid results, sorted by validation rmse
# gbm_gridperf1 <- h2o.getGrid(grid_id = "gbm_grid1",
#                              sort_by = "rmse",
#                              decreasing = FALSE)
# print(gbm_gridperf1)
#
# best_gbm <- h2o.getModel(gbm_gridperf1@model_ids[[1]])

best_gbm <- h2o.loadModel("best_gbm_final/GBM_model_R_1559420409492_5")

gbm_pred <- as.data.frame(h2o.predict(best_gbm, htrain))

print(c("MSE: ", MSE(gbm_pred$predict, as.numeric(as.vector(splits[[1]]$OPEN_DAYS)))))
```

```
## [1] "MSE: "            "128541.952540905"
```

```
best_gbm@model[["model_summary"]]
```

```
## Model Summary:
##   number_of_trees number_of_internal_trees model_size_in_bytes min_depth
## 1             200                      200              257307         0
##   max_depth mean_depth min_leaves max_leaves mean_leaves
## 1         9    4.32000          1        302    97.68000
```

```
best_gbm@parameters$mtries
```

```
## NULL
```

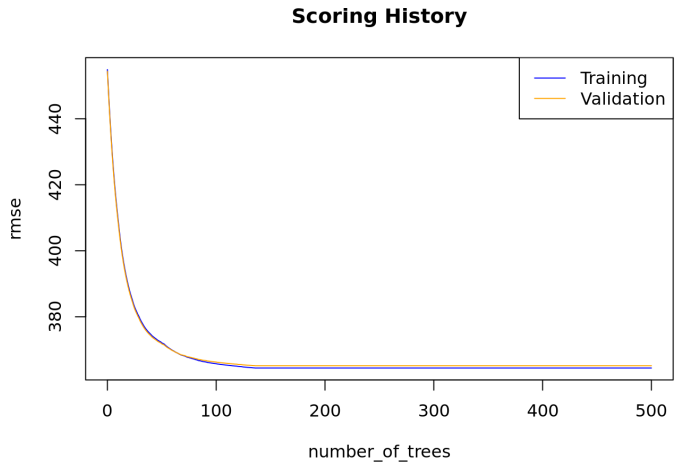
```
best_gbm_perf <- h2o.performance(model = best_gbm,
                                 newdata = hvalid)

print(best_gbm_perf)
```

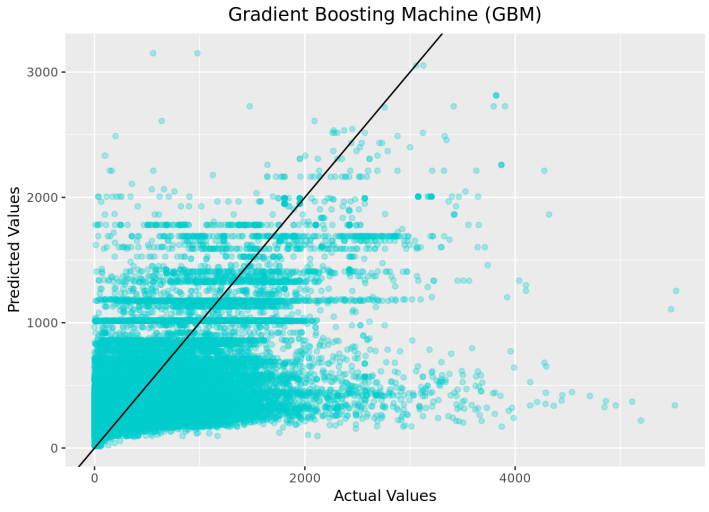
```
## H2ORegressionMetrics: gbm
##
## MSE:  130558.4
## RMSE:  361.3286
## MAE:  242.758
## RMSLE:  1.392857
## Mean Residual Deviance :  130558.4
```

```
predict_gbm <-as.data.frame(h2o.predict(best_gbm, hvalid))
best_gbm_perf <- model_performance( "best.gbm",best_gbm, predict_gbm, response$`as.numer
ic(as.vector(splits[[2]]$OPEN_DAYS))`, aic = FALSE)

best_gbm_perf
```

```
##      Model AIC        R2     RMSE
## 1 best.gbm  NA 0.3675452 361.3286
```

```
# ----------------------- Prediction vs. actual
predict_gbm <-as.data.frame(h2o.predict(best_gbm, hvalid))
response <- as.data.frame(as.numeric(as.vector(splits[[2]]$OPEN_DAYS)))
res_df <- as.data.frame(cbind(response$`as.numeric(as.vector(splits[[2]]$OPEN_DAYS))`, p
redict_gbm$predict))
colnames(res_df) <- c('Actual', 'Predicted')
#colnames(res_df)[colnames(res_df)=="as.numeric(as.vector(splits[[2]]$OPEN_DAYS))"]    <-
 "response"
# lm.model <- lm(Actual ~ Predicted, data = res_df)
# res_df <- cbind(res_df, predict(lm.model, interval = 'confidence'))
p <- ggplot(res_df, aes(Actual, Predicted))
p <- p + geom_point(color = 'cyan3', alpha=0.3) + ggtitle('Gradient Boosting Machine (GB
M)') + xlab('Actual Values')+ylab('Predicted Values') +
  theme(plot.title = element_text(hjust = 0.5))
p <- p + geom_abline(intercept = 0, slope = 1)
p
```





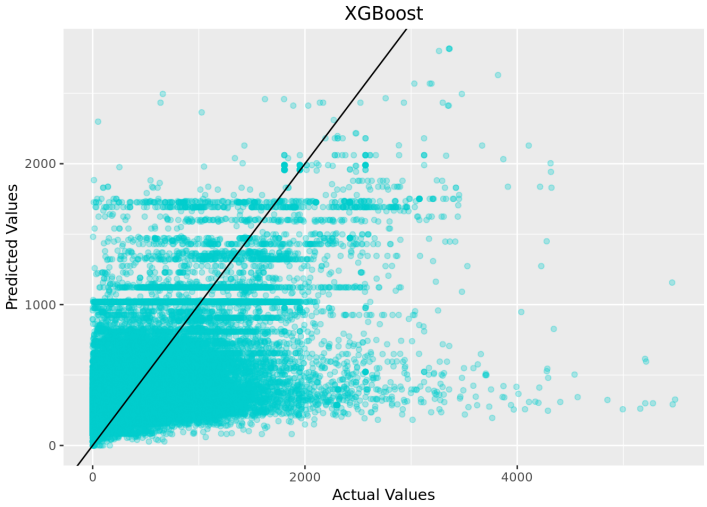
```
# ctrl <- trainControl(method = "repeatedcv",
                       # number = 5,
                       # repeats = 3,
                       # search = "grid",
                       # selectionFunction = "best",
                       # allowParallel = TRUE)
set.seed(849)
# caret.xgbt.model <- caret::train(OPEN_DAYS~., data=train,method = 'xgbTree',trControl
 = ctrl, metric = "RMSE")

# saveRDS(caret.xgbt.model,'caret.xgbt.model.rds')
caret.xgbt.model <- readRDS('caret.xgbt.model.rds')

predict_xgbt <- as.data.frame(predict(caret.xgbt.model, newdata = train))#,type='raw'
response <- as.data.frame(train$OPEN_DAYS)
print(c("MSE: ", MSE(response$`train$OPEN_DAYS`, predict_xgbt)))
```

```
## [1] "MSE: "           "134366.57411218"
```

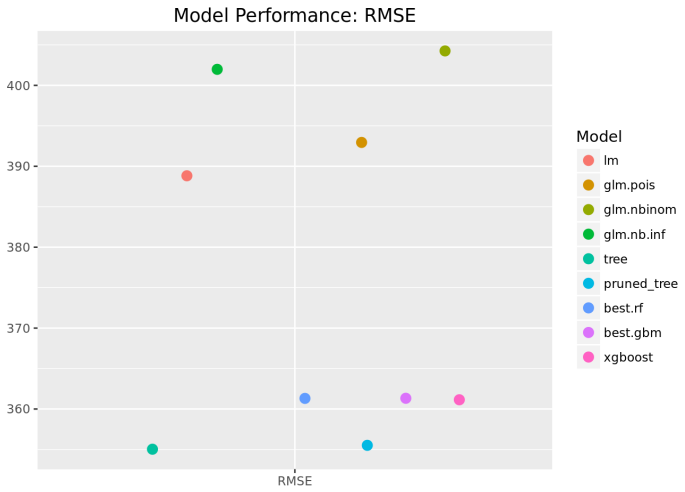
```
# summary(caret.xgbt.model)
#------------------------------ prediction vs. actual
predict_xgbt <- as.data.frame(predict(caret.xgbt.model, newdata = test))#,type='raw'
response <- as.data.frame(test$OPEN_DAYS)
res_df <- as.data.frame(cbind(response$`test$OPEN_DAYS`, predict_xgbt$`predict(caret.xgb
t.model, newdata = test)` ))
colnames(res_df) <- c('Actual', 'Predicted')

res_df$Predicted0 <- ifelse(res_df$Predicted < 0, 0, res_df$Predicted)
xgb_perf <- model_performance("xgboost", caret.xgbt.model, res_df$Predicted0, test$OPEN_
DAYS, aic = FALSE)
print(xgb_perf)
```

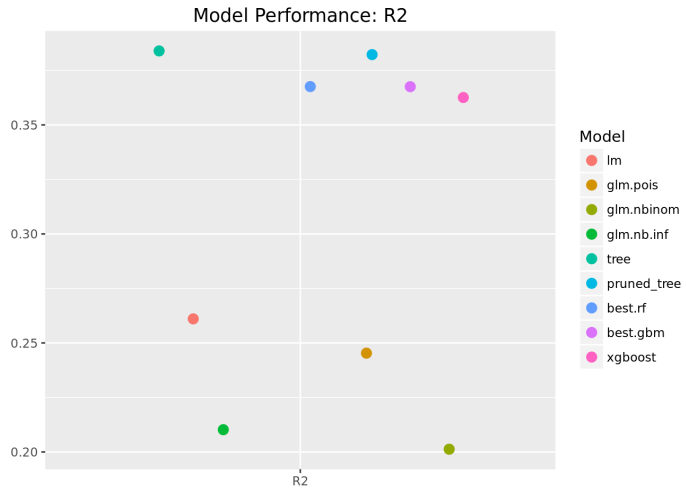
```
##     Model AIC        R2     RMSE
## 1 xgboost  NA 0.3625544 361.1394
```

```
p <- ggplot(res_df, aes(Actual, Predicted0))
p <- p + geom_point(color = 'cyan3', alpha=0.3) + ggtitle('XGBoost') + xlab('Actual Valu
es')+ylab('Predicted Values') +
  theme(plot.title = element_text(hjust = 0.5))
p <- p + geom_abline(intercept = 0, slope = 1)
p
```




```
models_performance <- data.frame()
models_performance <- rbind(models_performance, as.data.frame(lm_perf))
models_performance <- rbind(models_performance, as.data.frame(glm.pois_perf) )
models_performance <- rbind(models_performance, as.data.frame(glm.nbinom_perf) )
models_performance <- rbind(models_performance, as.data.frame(glm.nb.inf_perf) )
models_performance <- rbind(models_performance, as.data.frame(tree_perf) )
models_performance <- rbind(models_performance, as.data.frame(pruned_tree_perf) )
models_performance <- rbind(models_performance, as.data.frame(best_rf_perf) )
models_performance <- rbind(models_performance, as.data.frame(best_gbm_perf) )
models_performance <- rbind(models_performance, as.data.frame(xgb_perf) )


set.seed(5)
ggplot(models_performance, aes('RMSE', RMSE)) + geom_jitter(width = 0.35, aes(colour = M
odel), size=3) +
  xlab(' ') + ylab(' ')  + ggtitle('Model Performance: RMSE') +
  theme(plot.title = element_text(hjust = 0.5)) + ylab(' ')

set.seed(5)
ggplot(models_performance, aes('R2', R2)) + geom_jitter(width = 0.35, aes(colour = Mode
l), size=3) +
  xlab(' ') + ylab(' ')  + ggtitle('Model Performance: R2') +
  theme(plot.title = element_text(hjust = 0.5)) + ylab(' ')
```





Question 4:

In your model, what are the top 3 key predictors? Could you show in visualization how these variables are related to Days Open?

Generalized Linear Models provide interpretable estimated effect sizes (coefficients) that tell us how a change in a variable can change the link of the Open Days. It is very straightforward, as it provides a direction of relationship and a magnitude. For example, if coefficient of JURY_DEMANDN is -0.56, it would simply mean that, if the case is a JURY_DEMAND of type N, results in the decrease in the link value of Open Days by the amount of -0.56. These measures can help us in identifying the main drivers or factors that mostly impact number of Open Days. They can be used to find recommendations for improving the process. we can find on which conditions the Open days are high to find ways to resolve that problem. Or we can find the characteristics of factors, or conditions that have low Open cases. in this way we may be able to use those information on other categories.

Comparing Feature Importance: In GLM Zero-Inflated Negative Binomial Models, zero models predict non-occurrence of the outcome. Therefore, in the right plot shows that ORIGIN_CAT6 and ORIGIN_CAT25 are the most important predictors in getting the excess zeros. In contrast, JURY_DEMANDN, and ORIGIN_CAT3 are the most important predictors of the ZNB zero model that result in having positive number of Open Days. The plot on the left tells us that DISTRICT_CAT47, ORIGIN_CAT6, ORIGIN_CAT2 and CIRCUIT1 have the most effect on the number of Open days. They also all positively impact the expected number of Open Days. I am interpreting model results through Variable Importance (VI) plots, Partial Dependence Plot (PDP) and Individual Conditional Expectation (ICE) boxplot.

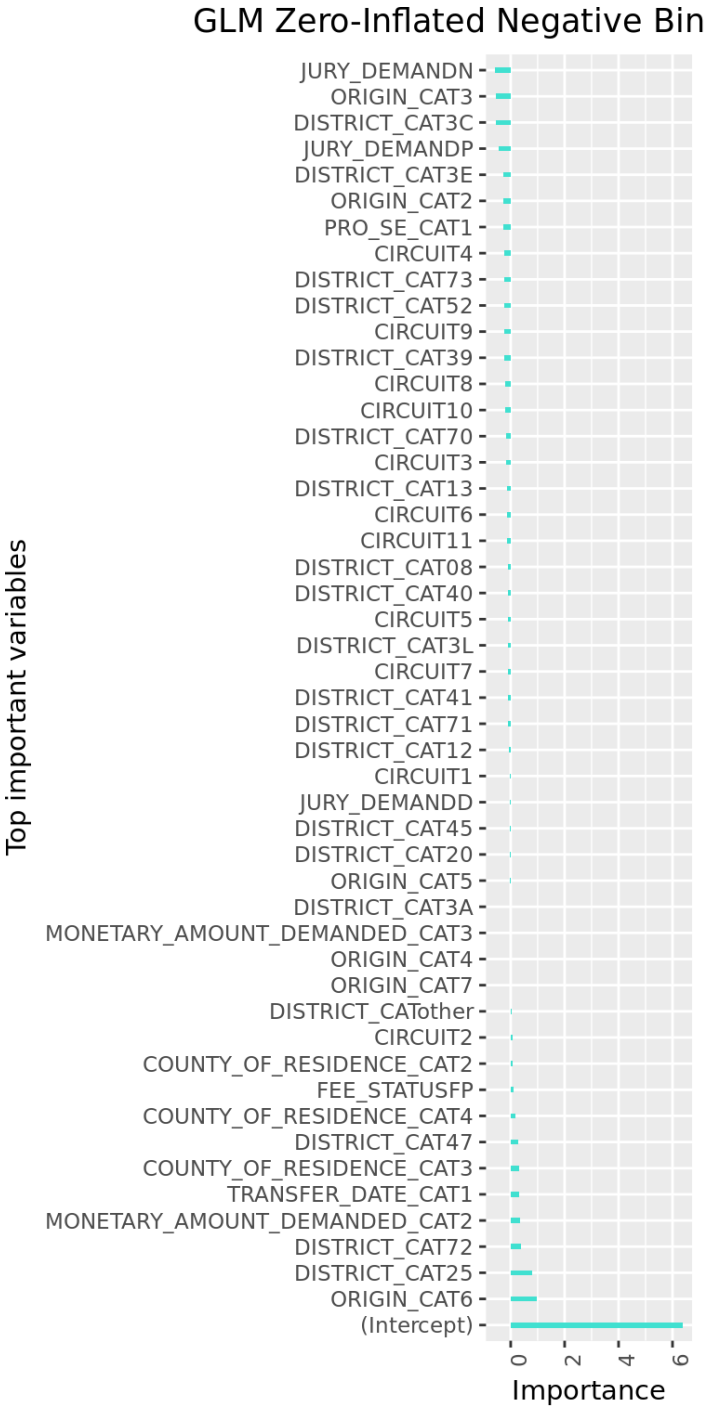
VI: I am using model libraries and objects to extract VI, I use library iml (Interpretable Machine Learning), and pdp. Variable importance plots tells us that what is the overall effect of a variable in predicting the dependent variable (Open days) and is captured through adding the total amount of loss that is decreased due to splits over a given predictor. But it does not tell us the direction of impact when using most machine learning methods such as Trees, RF, GBM, and XGBoost. Plots showing ‘Top important variables’ on the y axis for Tree Model and Pruned Tree, also plots titled “Variable Importance :DRF” and “Variable Importance :GBM” for the RF and GBM models. The plot ‘Importance’ at the end is for the XGBoost. These plots provide an importance measure for the averall impact of a veriable with its all levels. Tree, Pruned Tree, RF and GBM tell us that ORIGIN_CAT, DISTRICT_CAT, JURY_DEMAND, followed by CIRCUIT are the most important predictors.

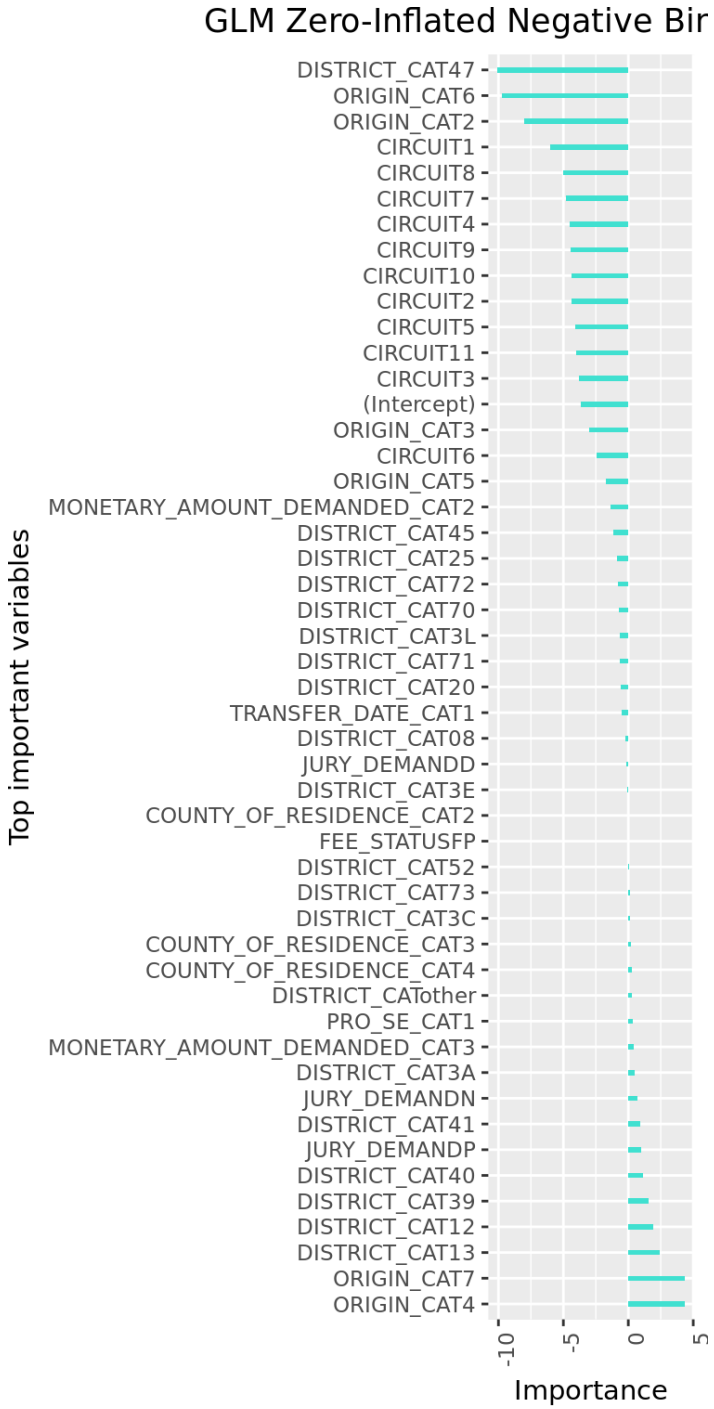
PDP: The Partial Dependence Plot shows how mean Open Days changes with changes in a given predictor. It shows how each predictor is related to Days Open. The predictor that has the most amount in change in the mean of Open Days is more important. The mean of Open days at ORIGIN_CAT6 is significantly larger that its other levels, indicating high relationship. DISTRICT_CAT has a lot of variation in the mean response across different levels. The amount of change in each category is not as large as changes in ORIGIN_CAT6,however, DISTRICT_CAT 25 is the variable factor with the largest impact in mean Open Days.
JURY_DEMAND B and N seems to be the next variable factor with largest change in Open Days. For COUNTY_OF_RESIDENCE_CAT the category 3 have greater median.We can see that although ORIGIN_CAT is the feature with the highest importance, only category 25 among all other categories of ORIGIN_CAT have high effect of the mean response.

ICE: The ICE box plots tell us how each observation or case’s Open Days has changed, when a factor is changing. So the same logic holds for identifing variables with larger effect. For JURY_DEMAND the category N has the narrowest spread and the median is significantly smaller from other JURY_DEMAND categories. this indicates significant difference in the number of open days per individual compared to other levels in JURY_DEMAND. Therefore in JURY_DEMAND category N has high effect on open days.

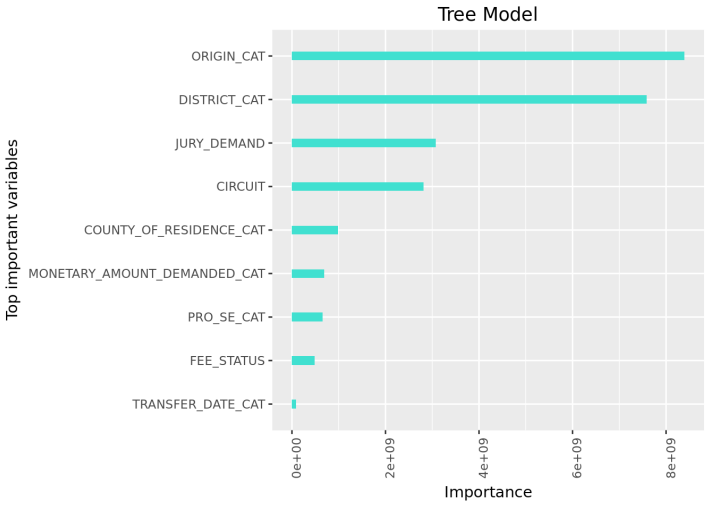
The Top three predictors: ORIGIN_ CAT6, DISTRICT_CAT25, JURY_DEMAND B. The Top three variables: ORIGIN_CAT, DISTRICT_CAT, JURY_DEMAND Among ORIGIN_CAT, ORIGIN_CAT3 is the most important factor. The number of open days in each observation and also the mean number of Open days are highly affected by ORIGIN_CAT3. the same logic is true for aforementioned variables and their categories.

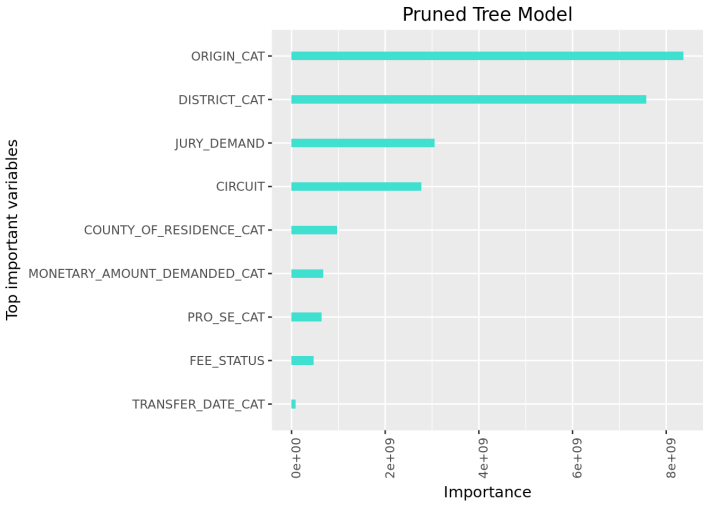
```
# glm zero inflated neg binomial GLM
glm.nb.inf <- readRDS("glm.nb.inf.rds")

temp <- data.frame(glm.nb.inf$coefficients)
temp <- temp[order(temp$count, decreasing = TRUE), , drop = FALSE]
temp[['X']] <- rownames(temp)
rownames(temp) <- NULL
temp$X <- factor(temp$X, levels = temp$X)
p<-ggplot(data=temp, aes(x=X, y=count)) +
  geom_bar(stat="identity",  width = 0.2 , fill = "turquoise") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1), legend.position = 'none')+
  labs(x = "Top important variables", y = 'Importance',
       title="GLM Zero-Inflated Negative Binomial: Count Part")  + coord_flip() +
  theme(plot.title = element_text(hjust = 0.5))
p

# zero model
temp <- data.frame(glm.nb.inf$coefficients)
temp <- temp[order(temp$zero, decreasing = TRUE), , drop = FALSE]
temp[['X']] <- rownames(temp)
rownames(temp) <- NULL
temp$X <- factor(temp$X, levels = temp$X)
p<-ggplot(data=temp, aes(x=X, y=zero)) +
  geom_bar(stat="identity",  width = 0.2 , fill = "turquoise") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1), legend.position = 'none')+
  labs(x = "Top important variables", y = 'Importance',
       title="GLM Zero-Inflated Negative Binomial: Logit Part")  + coord_flip() +
  theme(plot.title = element_text(hjust = 0.5))
p
```


GLM Zero-Inflated Negative Bin
Top important variables
JURY_DEMANDN
ORIGIN_CAT3
DISTRICT_CAT3C
JURY_DEMANDP
DISTRICT_CAT3E
ORIGIN_CAT2
PRO_SE_CAT1
CIRCUIT4
DISTRICT_CAT73
DISTRICT_CAT52
CIRCUIT9
DISTRICT_CAT39
CIRCUIT8
CIRCUIT10
DISTRICT_CAT70
CIRCUIT3
DISTRICT_CAT13
CIRCUIT6
CIRCUIT11
DISTRICT_CAT08
DISTRICT_CAT40
CIRCUIT5
DISTRICT_CAT3L
CIRCUIT7
DISTRICT_CAT41
DISTRICT_CAT71
DISTRICT_CAT12
CIRCUIT1
JURY_DEMANDD
DISTRICT_CAT45
DISTRICT_CAT20
ORIGIN_CAT5
DISTRICT_CAT3A
MONETARY_AMOUNT_DEMANDED_CAT3
ORIGIN_CAT4
ORIGIN_CAT7
DISTRICT_CAOther
CIRCUIT2
COUNTY_OF_RESIDENCE_CAT2
FEE_STATUSFP
COUNTY_OF_RESIDENCE_CAT4
DISTRICT_CAT47
COUNTY_OF_RESIDENCE_CAT3
TRANSFER_DATE_CAT1
MONETARY_AMOUNT_DEMANDED_CAT2
DISTRICT_CAT72
DISTRICT_CAT25
ORIGIN_CAT6
(Intercept)
0
2
4
6
Importance

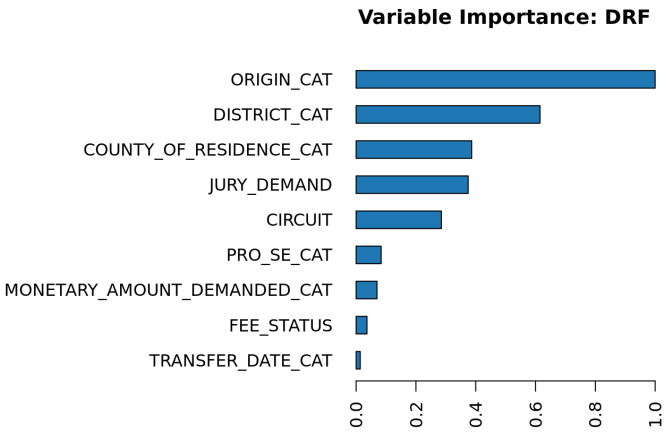

GLM Zero-Inflated Negative Bir
Top important variables
DISTRICT_CAT47
ORIGIN_CAT6
ORIGIN_CAT2
CIRCUIT1
CIRCUIT8
CIRCUIT7
CIRCUIT4
CIRCUIT9
CIRCUIT10
CIRCUIT2
CIRCUIT5
CIRCUIT11
CIRCUIT3
(Intercept)
ORIGIN_CAT3
CIRCUIT6
ORIGIN_CAT5
MONETARY_AMOUNT_DEMANDED_CAT2
DISTRICT_CAT45
DISTRICT_CAT25
DISTRICT_CAT72
DISTRICT_CAT70
DISTRICT_CAT3L
DISTRICT_CAT71
DISTRICT_CAT20
TRANSFER_DATE_CAT1
DISTRICT_CAT08
JURY_DEMANDD
DISTRICT_CAT3E
COUNTY_OF_RESIDENCE_CAT2
FEE_STATUSFP
DISTRICT_CAT52
DISTRICT_CAT73
DISTRICT_CAT3C
COUNTY_OF_RESIDENCE_CAT3
COUNTY_OF_RESIDENCE_CAT4
DISTRICT_CAOther
PRO_SE_CAT1
MONETARY_AMOUNT_DEMANDED_CAT3
DISTRICT_CAT3A
JURY_DEMANDN
DISTRICT_CAT41
JURY_DEMANDP
DISTRICT_CAT40
DISTRICT_CAT39
DISTRICT_CAT12
DISTRICT_CAT13
ORIGIN_CAT7
ORIGIN_CAT4
-10
-5
0
5
Importance

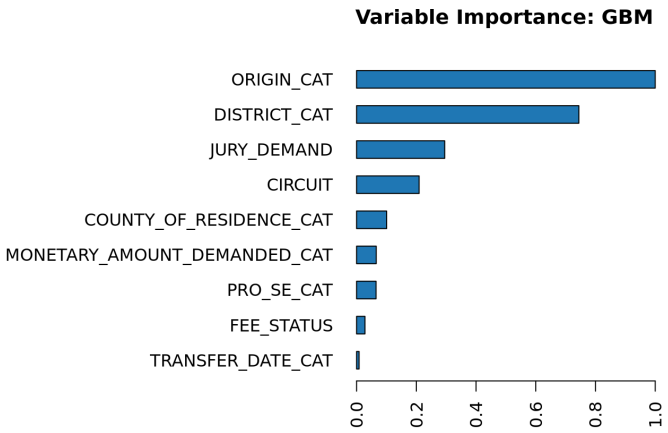
```
# Tree
tree <- readRDS("tree.rds")
temp <- data.frame(tree$variable.importance)
temp <- temp[order(temp$tree.variable.importance, decreasing = FALSE), , drop = FALSE]
temp[['X']] <- rownames(temp)
rownames(temp) <- NULL
temp$X <- factor(temp$X, levels = temp$X)
p<-ggplot(data=temp, aes(x=X, y=tree.variable.importance)) +
  geom_bar(stat="identity",  width = 0.2 , fill = "turquoise") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1), legend.position = 'none')+
  labs(x = "Top important variables", y = 'Importance', title="Tree Model")  +
  coord_flip() + theme(plot.title = element_text(hjust = 0.5))
p

# Pruned Tree
pruned_tree <- readRDS("pruned_tree.rds")
temp <- data.frame(pruned_tree$variable.importance)
temp <- temp[order(temp$pruned_tree.variable.importance, decreasing = FALSE), , drop = F
ALSE]
temp[['X']] <- rownames(temp)
rownames(temp) <- NULL
temp$X <- factor(temp$X, levels = temp$X)
p<-ggplot(data=temp, aes(x=X, y=pruned_tree.variable.importance)) +
  geom_bar(stat="identity",  width = 0.2 , fill = "turquoise") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1), legend.position = 'none')+
  labs(x = "Top important variables", y = 'Importance', title="Pruned Tree Model")  +
  coord_flip() + theme(plot.title = element_text(hjust = 0.5))
p
```

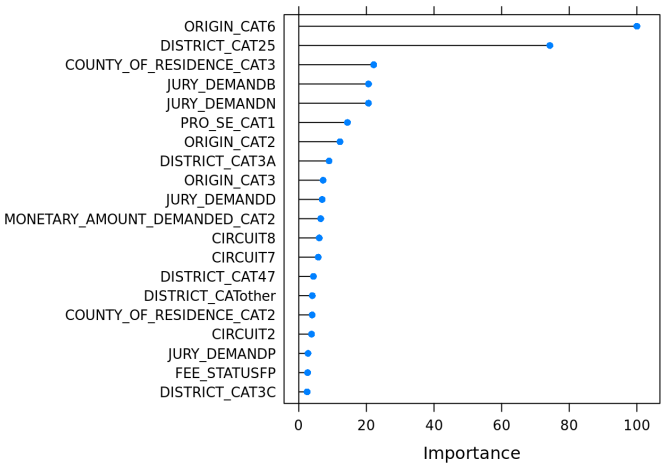





```
# rf
h2o.varimp_plot(best_rf)
# gbm
h2o.varimp_plot(best_gbm)

#xgboost
xgbt_varImp <- varImp(caret.xgbt.model)
plot(xgbt_varImp, top = 20)
```







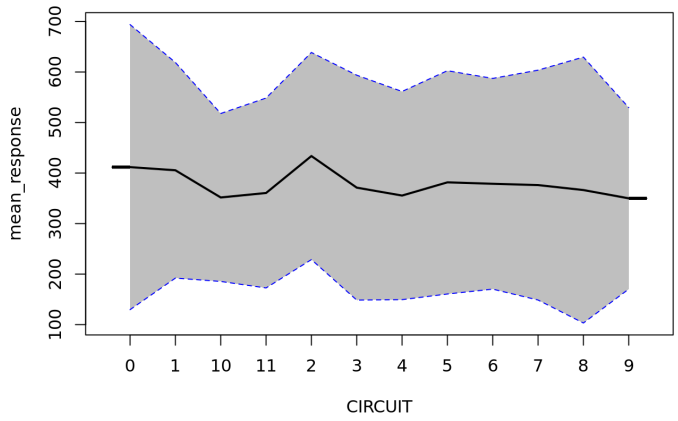
Partial Dependence Plots: Random Forest

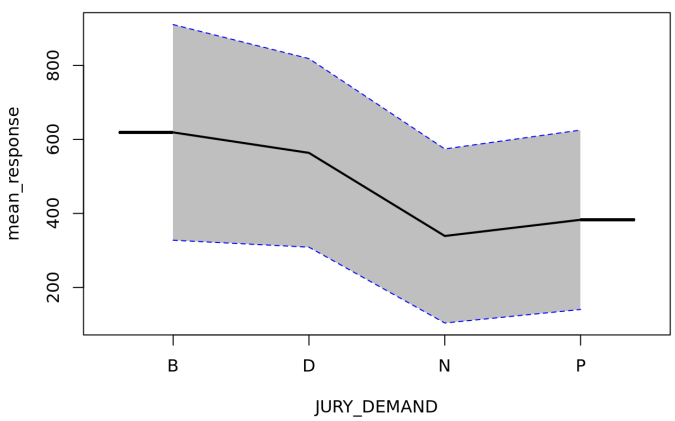
```
#---------------------------- Partioal Dependency Plot and Ice plot
h2o.partialPlot(best_rf, data = htrain, nbins = 30)
```

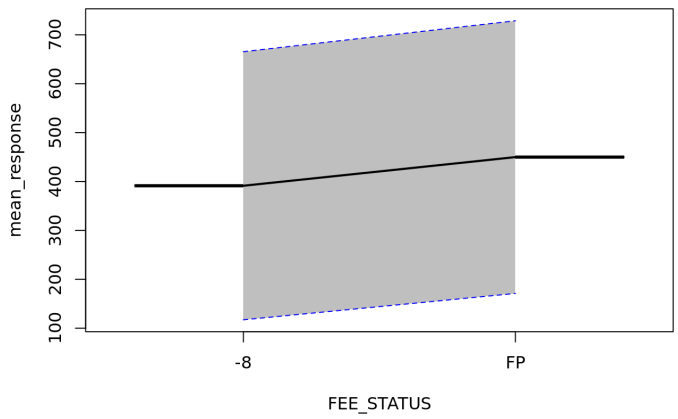
```
## [[1]]
## PartialDependence: Partial Dependence Plot of model DRF_model_R_1559420409492_4 on co
lumn 'CIRCUIT'
##    CIRCUIT mean_response stddev_response std_error_mean_response
## 1        0    411.765181      282.164932                0.533565
## 2        1    405.512677      213.439185                0.403607
## 3       10    351.692017      166.223673                0.314324
## 4       11    360.548002      187.971095                0.355447
## 5        2    433.710955      204.998176                0.387645
## 6        3    371.150114      222.588091                0.420907
## 7        4    355.467866      206.049833                0.389634
## 8        5    381.629006      221.017292                0.417936
## 9        6    378.790595      208.558164                0.394377
## 10       7    376.217005      227.530547                0.430253
## 11       8    366.377687      263.214118                0.497729
## 12       9    349.977269      179.673624                0.339757
##
## [[2]]
## PartialDependence: Partial Dependence Plot of model DRF_model_R_1559420409492_4 on co
lumn 'JURY_DEMAND'
##   JURY_DEMAND mean_response stddev_response std_error_mean_response
## 1           B    618.912336      291.316303                0.550870
## 2           D    563.558370      254.744576                0.481714
## 3           N    339.089188      235.178391                0.444715
## 4           P    382.832415      242.373928                0.458321
##
## [[3]]
## PartialDependence: Partial Dependence Plot of model DRF_model_R_1559420409492_4 on co
lumn 'FEE_STATUS'
##   FEE_STATUS mean_response stddev_response std_error_mean_response
## 1         -8    391.349568      274.194206                0.518492
## 2         FP    449.931188      278.865554                0.527326
##
## [[4]]
## PartialDependence: Partial Dependence Plot of model DRF_model_R_1559420409492_4 on co
lumn 'MONETARY_AMOUNT_DEMANDED_CAT'
##   MONETARY_AMOUNT_DEMANDED_CAT mean_response stddev_response
## 1                            1    393.473272      268.708328
## 2                            2    441.586912      294.916131
## 3                            3    417.801820      282.609081
##   std_error_mean_response
## 1                0.508119
## 2                0.557677
## 3                0.534405
##
## [[5]]
## PartialDependence: Partial Dependence Plot of model DRF_model_R_1559420409492_4 on co
lumn 'TRANSFER_DATE_CAT'
##   TRANSFER_DATE_CAT mean_response stddev_response std_error_mean_response
## 1                 0    394.137522      274.855292                0.519742
## 2                 1    455.344808      266.643884                0.504215
##
## [[6]]
```

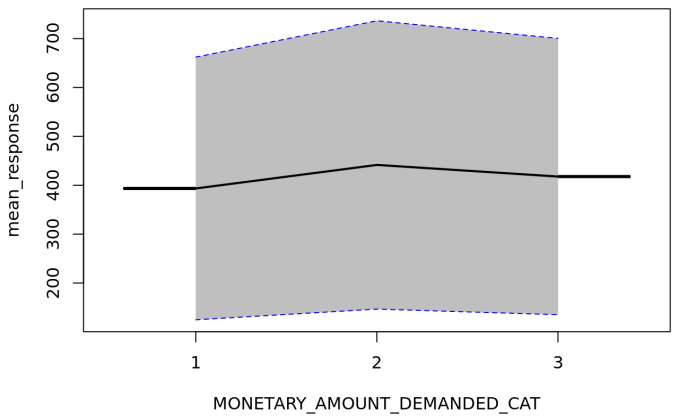
```
## PartialDependence: Partial Dependence Plot of model DRF_model_R_1559420409492_4 on co
lumn 'ORIGIN_CAT'
##   ORIGIN_CAT mean_response stddev_response std_error_mean_response
## 1          1    392.609294      223.846003                0.423285
## 2          2    268.463308      132.950633                0.251405
## 3          3    323.703265      138.598987                0.262086
## 4          4    315.672158      138.041027                0.261031
## 5          5    478.682748      305.122314                0.576976
## 6          6   1163.042787      431.207325                0.815399
## 7          7    310.647898      166.633793                0.315099
##
## [[7]]
## PartialDependence: Partial Dependence Plot of model DRF_model_R_1559420409492_4 on co
lumn 'DISTRICT_CAT'
##   DISTRICT_CAT mean_response stddev_response std_error_mean_response
## 1           07    401.531840      209.425367                0.396017
## 2           08    355.108302      163.076510                0.308372
## 3           12    321.791648      149.316741                0.282353
## 4           13    292.792882      185.528069                0.350828
## 5           20    357.144770      167.291121                0.316342
##
## ---
##    DISTRICT_CAT mean_response stddev_response std_error_mean_response
## 16           52    310.073369      146.584328                0.277186
## 17           70    313.438465      138.742835                0.262358
## 18           71    319.396225      142.209705                0.268914
## 19           72    411.495995      171.088648                0.323523
## 20           73    292.031518      150.423787                0.284446
## 21        other    380.390466      219.131217                0.414370
##
## [[8]]
## PartialDependence: Partial Dependence Plot of model DRF_model_R_1559420409492_4 on co
lumn 'PRO_SE_CAT'
##   PRO_SE_CAT mean_response stddev_response std_error_mean_response
## 1          0    425.753181      267.018326                0.504923
## 2          1    379.886853      309.191958                0.584672
##
## [[9]]
## PartialDependence: Partial Dependence Plot of model DRF_model_R_1559420409492_4 on co
lumn 'COUNTY_OF_RESIDENCE_CAT'
##   COUNTY_OF_RESIDENCE_CAT mean_response stddev_response
## 1                       1    389.906042      237.308031
## 2                       2    368.661374      224.520372
## 3                       3    402.209877      290.807596
## 4                       4    403.538254      280.635422
##   std_error_mean_response
## 1                0.448742
## 2                0.424561
## 3                0.549908
## 4                0.530672
```

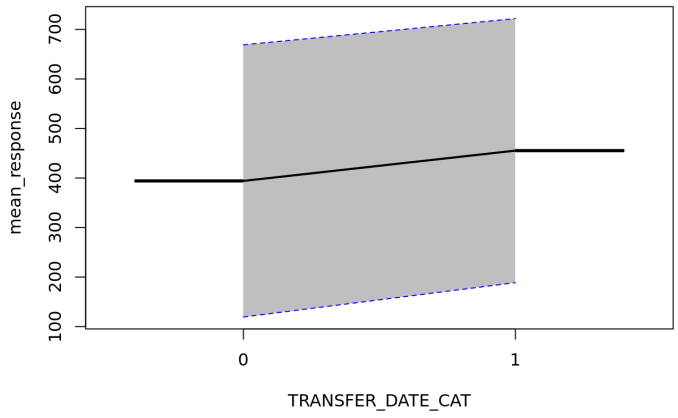
```
#-----------
```

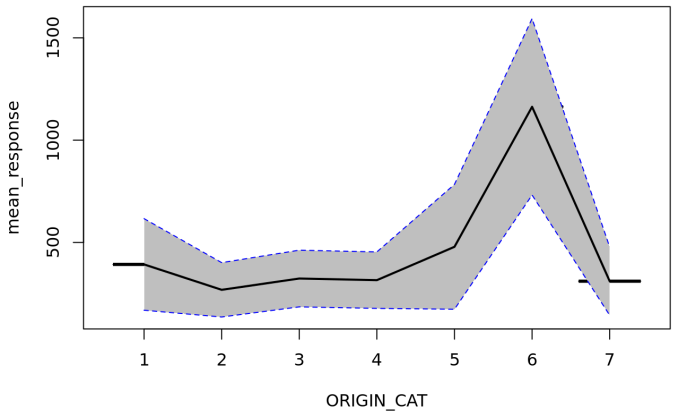

mean_response
100
200
300
400
500
600
700
0
1
10
11
2
3
4
5
6
7
8
9
CIRCUIT

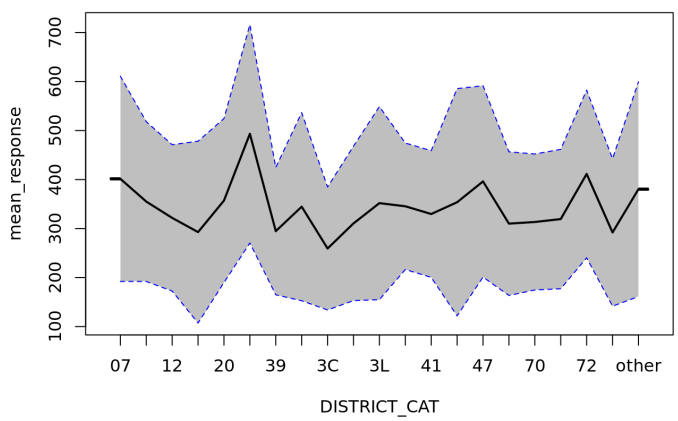

mean_response
200
400
600
800
B
D
N
P
JURY_DEMAND

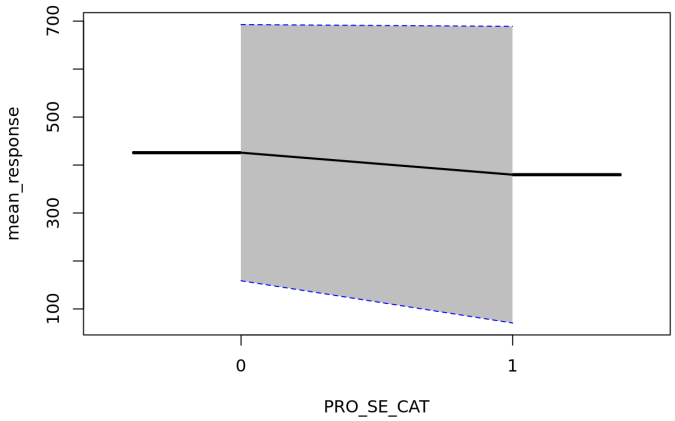

mean_response
100
200
300
400
500
600
700
-8
FP
FEE_STATUS

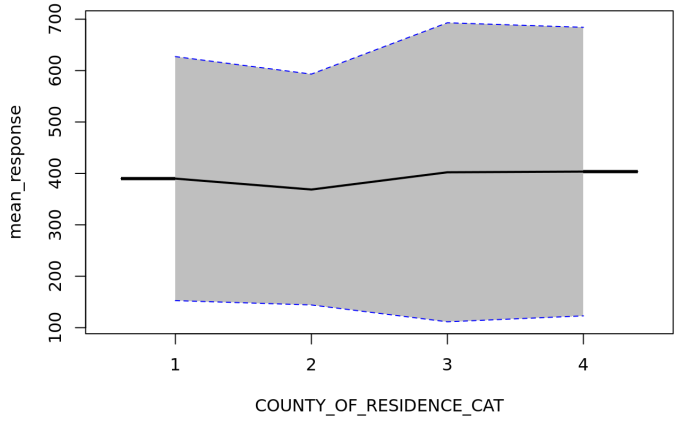

mean_response
200
300
400
500
600
700
1
2
3
MONETARY_AMOUNT_DEMANDED_CAT


mean_response
100
200
300
400
500
600
700
0
1
TRANSFER_DATE_CAT


mean_response
500
1000
1500
1
2
3
4
5
6
7
ORIGIN_CAT





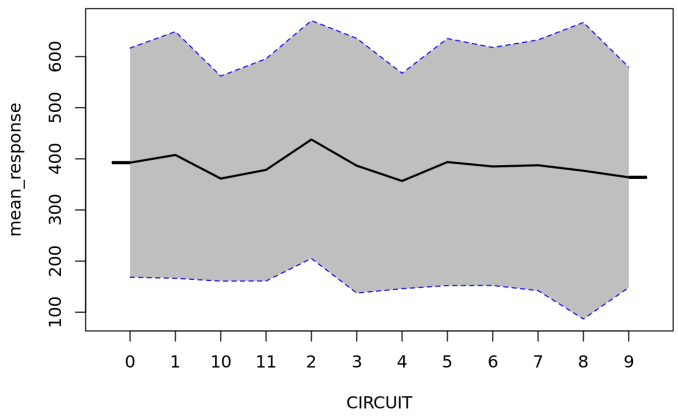


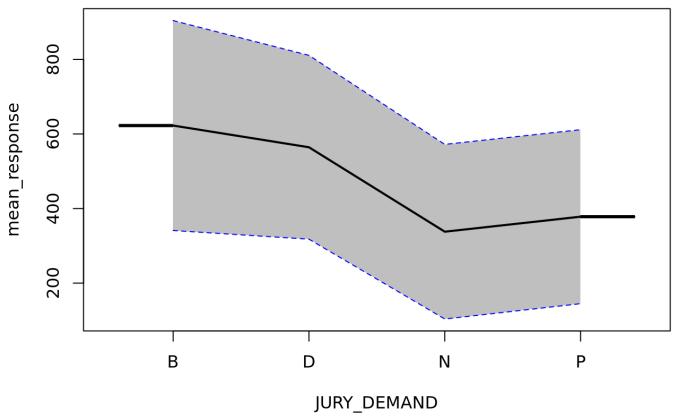
Partial Dependence Plots: GBM

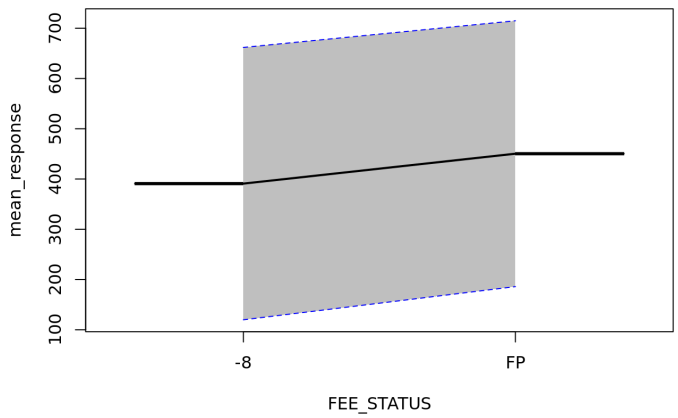
```
h2o.partialPlot(best_gbm, data = htrain, nbins = 30)
```

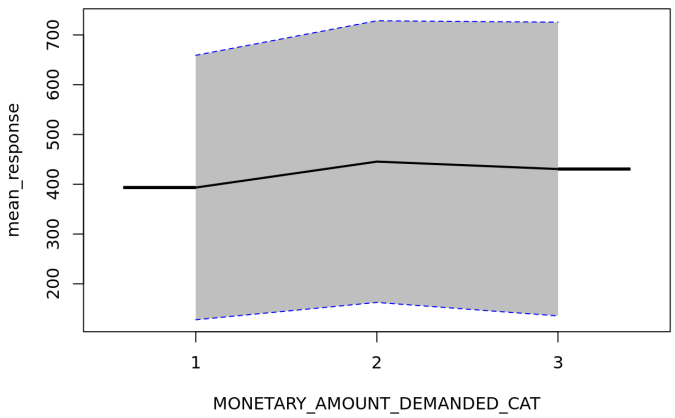
```
## [[1]]
## PartialDependence: Partial Dependence Plot of model GBM_model_R_1559420409492_5 on co
lumn 'CIRCUIT'
##    CIRCUIT mean_response stddev_response std_error_mean_response
## 1        0    392.606456      224.113445                0.423791
## 2        1    407.629827      241.315999                0.456321
## 3       10    361.401225      200.556265                0.379245
## 4       11    378.473431      217.379527                0.411058
## 5        2    437.776757      232.708363                0.440044
## 6        3    386.598010      249.143501                0.471122
## 7        4    356.719771      210.735282                0.398494
## 8        5    393.718851      241.611277                0.456879
## 9        6    385.083819      232.794568                0.440207
## 10       7    387.452872      245.231948                0.463726
## 11       8    376.751955      290.075657                0.548524
## 12       9    363.876741      215.669156                0.407823
##
## [[2]]
## PartialDependence: Partial Dependence Plot of model GBM_model_R_1559420409492_5 on co
lumn 'JURY_DEMAND'
##   JURY_DEMAND mean_response stddev_response std_error_mean_response
## 1           B    622.570017      281.541488                0.532386
## 2           D    564.340493      246.483339                0.466092
## 3           N    337.897099      234.193100                0.442852
## 4           P    378.189362      233.285074                0.441134
##
## [[3]]
## PartialDependence: Partial Dependence Plot of model GBM_model_R_1559420409492_5 on co
lumn 'FEE_STATUS'
##   FEE_STATUS mean_response stddev_response std_error_mean_response
## 1         -8    390.867649      270.955219                0.512367
## 2         FP    450.399152      264.476529                0.500116
##
## [[4]]
## PartialDependence: Partial Dependence Plot of model GBM_model_R_1559420409492_5 on co
lumn 'MONETARY_AMOUNT_DEMANDED_CAT'
##   MONETARY_AMOUNT_DEMANDED_CAT mean_response stddev_response
## 1                            1    393.448788      265.687464
## 2                            2    445.531455      282.928090
## 3                            3    430.592280      295.041475
##   std_error_mean_response
## 1                0.502406
## 2                0.535008
## 3                0.557914
##
## [[5]]
## PartialDependence: Partial Dependence Plot of model GBM_model_R_1559420409492_5 on co
lumn 'TRANSFER_DATE_CAT'
##   TRANSFER_DATE_CAT mean_response stddev_response std_error_mean_response
## 1                 0    394.212670      271.828527                0.514019
## 2                 1    463.611292      270.836602                0.512143
##
## [[6]]
```

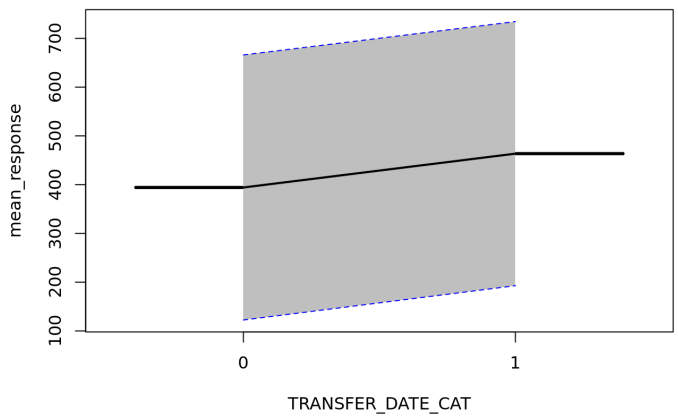
```
## PartialDependence: Partial Dependence Plot of model GBM_model_R_1559420409492_5 on co
lumn 'ORIGIN_CAT'
##   ORIGIN_CAT mean_response stddev_response std_error_mean_response
## 1          1    393.269918      220.160539                0.416316
## 2          2    265.734018      128.225648                0.242471
## 3          3    317.775846       95.690864                0.180948
## 4          4    314.572080      111.539617                0.210918
## 5          5    492.052404      279.933818                0.529346
## 6          6   1204.195367      521.201247                0.985575
## 7          7    293.958406      115.845365                0.219060
##
## [[7]]
## PartialDependence: Partial Dependence Plot of model GBM_model_R_1559420409492_5 on co
lumn 'DISTRICT_CAT'
##   DISTRICT_CAT mean_response stddev_response std_error_mean_response
## 1           07    414.827260      218.734418                0.413620
## 2           08    356.251980      174.526961                0.330025
## 3           12    324.990454      157.762247                0.298323
## 4           13    286.293073      197.488897                0.373445
## 5           20    367.804687      151.479263                0.286442
##
## ---
##    DISTRICT_CAT mean_response stddev_response std_error_mean_response
## 16           52    295.282843      130.364052                0.246514
## 17           70    307.712754      128.649699                0.243272
## 18           71    315.357638      137.138910                0.259325
## 19           72    433.588039      175.529770                0.331921
## 20           73    287.283549      146.901586                0.277786
## 21        other    382.583899      215.011958                0.406581
##
## [[8]]
## PartialDependence: Partial Dependence Plot of model GBM_model_R_1559420409492_5 on co
lumn 'PRO_SE_CAT'
##   PRO_SE_CAT mean_response stddev_response std_error_mean_response
## 1          0    423.342800      262.320345                0.496039
## 2          1    374.760496      306.705353                0.579970
##
## [[9]]
## PartialDependence: Partial Dependence Plot of model GBM_model_R_1559420409492_5 on co
lumn 'COUNTY_OF_RESIDENCE_CAT'
##   COUNTY_OF_RESIDENCE_CAT mean_response stddev_response
## 1                       1    399.305271      231.349161
## 2                       2    369.592687      223.223806
## 3                       3    407.441013      285.283551
## 4                       4    402.710149      265.816779
##   std_error_mean_response
## 1                0.437474
## 2                0.422109
## 3                0.539462
## 4                0.502651
```

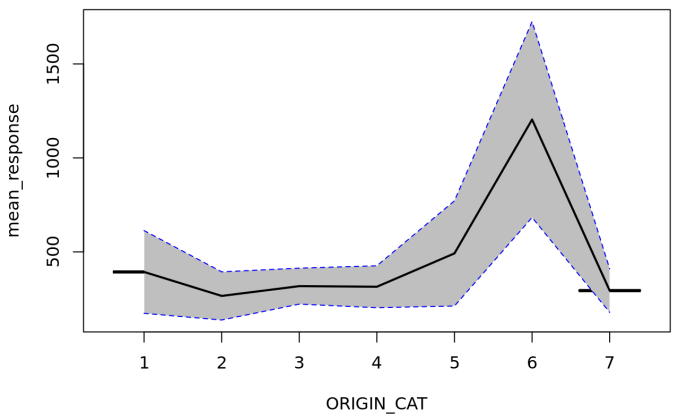

mean_response
100
200
300
400
500
600
0
1
10
11
2
3
4
5
6
7
8
9
CIRCUIT

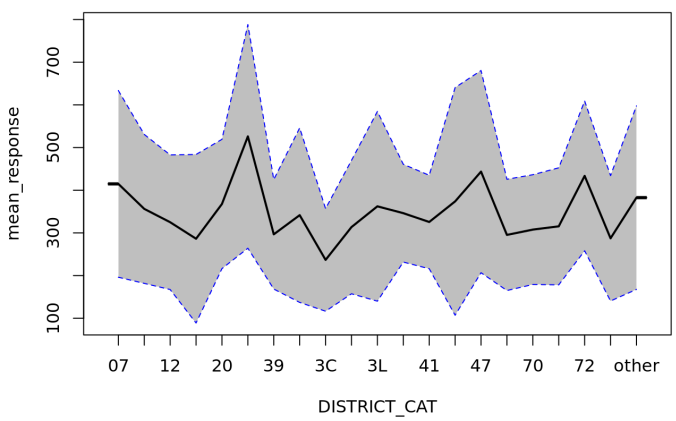

mean_response
200
400
600
800
B
D
N
P
JURY_DEMAND

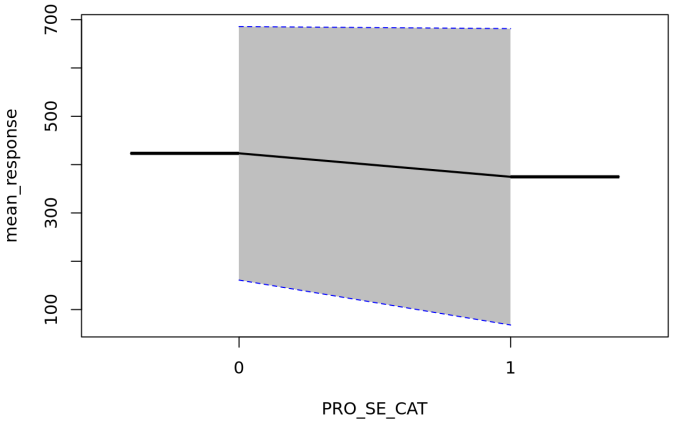

mean_response
100
200
300
400
500
600
700
-8
FP
FEE_STATUS

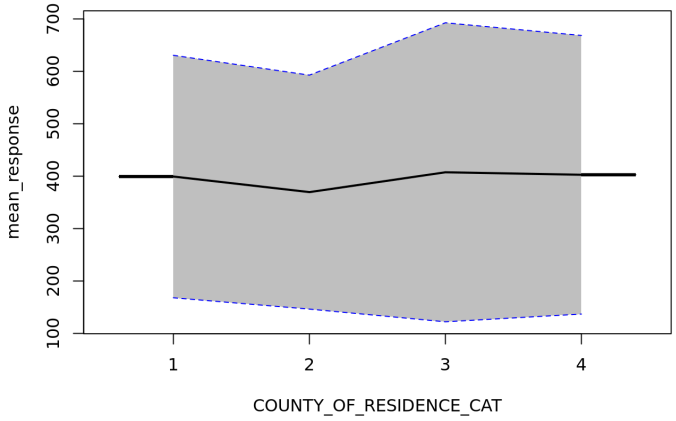

mean_response
200
300
400
500
600
700
1
2
3
MONETARY_AMOUNT_DEMANDED_CAT


mean_response
100
200
300
400
500
600
700
0
1
TRANSFER_DATE_CAT


mean_response
500
1000
1500
1
2
3
4
5
6
7
ORIGIN_CAT

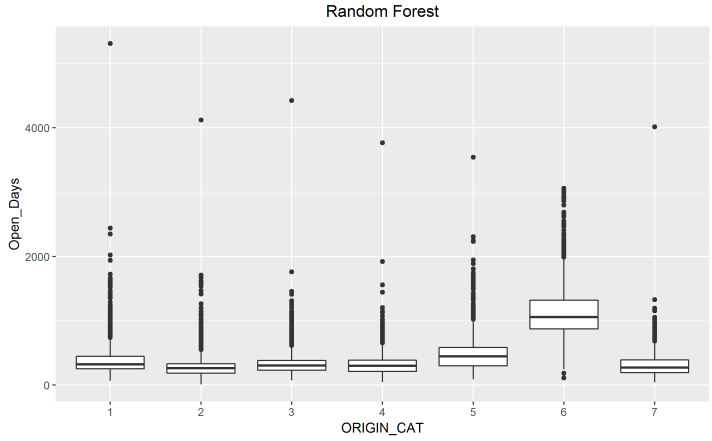

mean_response
100
300
500
700
07
12
20
39
3C
3L
41
47
70
72
other
DISTRICT_CAT

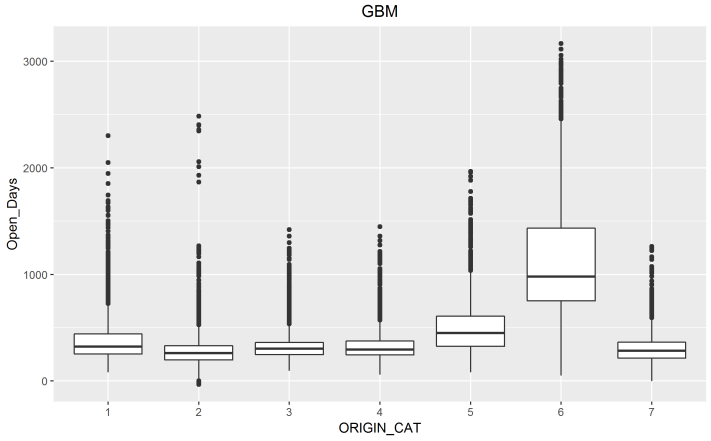

mean_response
100
300
500
700
0
1
PRO_SE_CAT

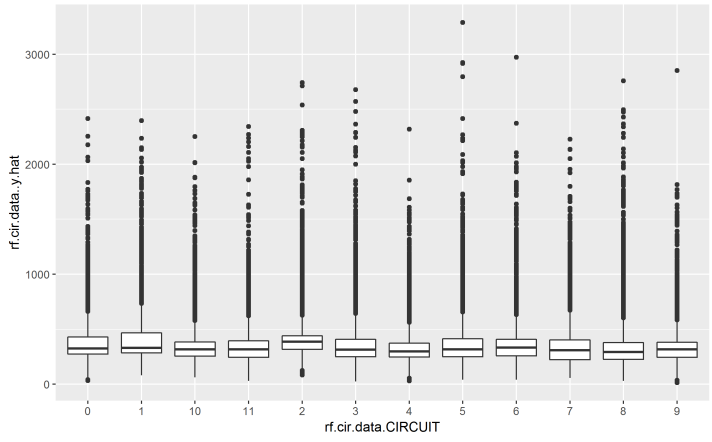

mean_response
100
200
300
400
500
600
700
1
2
3
4
COUNTY_OF_RESIDENCE_CAT

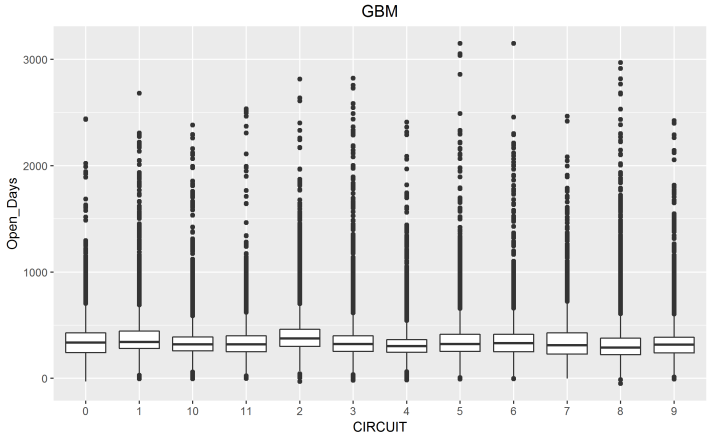
```
# ----------------- ICE boxplot for the most important features
# features <- as.data.frame(splits[[2]][, !(colnames(hvalid) %in% c("OPEN_DAYS"))])
# response <- as.data.frame(as.numeric(as.vector(splits[[2]]$OPEN_DAYS)))
#
# predfunc <- function(model, newdata)  {
#   results <- as.data.frame(h2o.predict(model, as.h2o(newdata)))
#   return(results)
# }
#
# predictor.rf <- Predictor$new(model = best_rf,   data = features,   y = response, pred
ict.fun = predfunc)
# predictor.gbm<- Predictor$new(model = best_gbm,   data = features,   y = response, pre
dict.fun = predfunc)
#
#
# rf.org <- Partial$new(predictor.rf, "ORIGIN_CAT") %>% plot() + ggtitle("RF")
# gbm.org <- Partial$new(predictor.gbm, "ORIGIN_CAT") %>% plot() + ggtitle("GBM")
#
# temp <- data.frame(rf.org$data$ORIGIN_CAT, rf.org$data$.y.hat)
# g <- ggplot(temp, aes(y=rf.org.data..y.hat, x = rf.org.data.ORIGIN_CAT)) +
#   geom_boxplot()  + labs(x = "ORIGIN_CAT", y = 'Open_Days', title="Random Forest")  +
#   theme(plot.title = element_text(hjust = 0.5))
# ggsave(file="ice_rf_boxplot_ORIGIN_CAT.png", g)
#
# temp <- data.frame( gbm.org$data$ORIGIN_CAT, gbm.org$data$.y.hat)
# g <- ggplot(temp, aes(y=gbm.org.data..y.hat, x = gbm.org.data.ORIGIN_CAT)) +
#   geom_boxplot() +labs(x = "ORIGIN_CAT", y = 'Open_Days', title="GBM")  +
#   theme(plot.title = element_text(hjust = 0.5))
# ggsave(file="ice_gbm_boxplot_ORIGIN_CAT.png", g)
#
# rf.cir <- Partial$new(predictor.rf, "CIRCUIT") %>% plot() + ggtitle("RF")
# gbm.cir <- Partial$new(predictor.gbm, "CIRCUIT") %>% plot() + ggtitle("GBM")
#
# temp <- data.frame( rf.cir$data$CIRCUIT, rf.cir$data$.y.hat)
# g <- ggplot(temp, aes(y=rf.cir.data..y.hat, x = rf.cir.data.CIRCUIT)) +
#   geom_boxplot()  + labs(x = "CIRCUIT", y = 'Open_Days', title="Random Forest")  +
#   theme(plot.title = element_text(hjust = 0.5))
# ggsave(file="ice_rf_boxplot_CIRCUIT.png", g)
#
# temp <- data.frame(gbm.cir$data$CIRCUIT, gbm.cir$data$.y.hat)
# g <- ggplot(temp, aes(y=gbm.cir.data..y.hat, x = gbm.cir.data.CIRCUIT)) +
#   geom_boxplot() +labs(x = "CIRCUIT", y = 'Open_Days', title="GBM")  +
#   theme(plot.title = element_text(hjust = 0.5))
# ggsave(file="ice_gbm_boxplot_CIRCUIT.png", g)
#
#
# rf.dist <- Partial$new(predictor.rf, "DISTRICT_CAT") %>% plot() + ggtitle("RF")
# gbm.dist <- Partial$new(predictor.gbm, "DISTRICT_CAT") %>% plot() + ggtitle("GBM")
#
# temp <- data.frame( rf.dist$data$DISTRICT_CAT, rf.dist$data$.y.hat)
# g <- ggplot(temp, aes(y=rf.dist.data..y.hat, x = rf.dist.data.DISTRICT_CAT)) +
#    geom_boxplot()  + labs(x = "DISTRICT_CAT", y = 'Open_Days', title="Random Forest")
 +
```

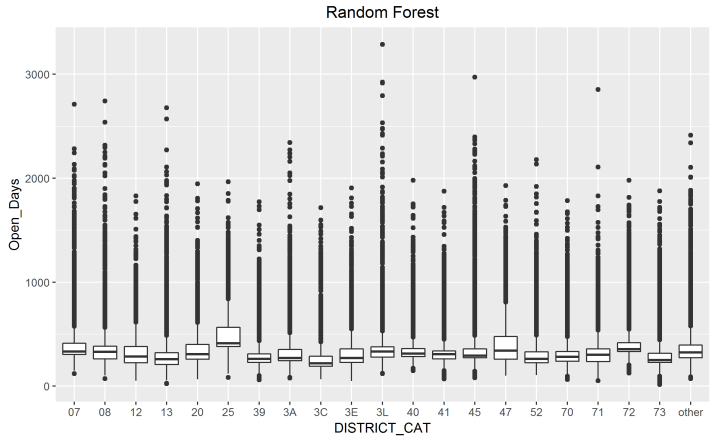
```
#   theme(plot.title = element_text(hjust = 0.5))
# ggsave(file="ice_rf_boxplot_DISTRICT_CAT.png", g)
#
# temp <- data.frame(gbm.dist$data$DISTRICT_CAT, gbm.dist$data$.y.hat)
# g <- ggplot(temp, aes(y=gbm.dist.data..y.hat, x = gbm.dist.data.DISTRICT_CAT)) +
#   geom_boxplot() +labs(x = "DISTRICT_CAT", y = 'Open_Days', title="GBM")  +
#   theme(plot.title = element_text(hjust = 0.5))
# ggsave(file="ice_gbm_boxplot_DISTRICT_CAT.png", g)
#
#
# rf.jur <- Partial$new(predictor.rf, "JURY_DEMAND") %>% plot() + ggtitle("RF")
# gbm.jur <- Partial$new(predictor.gbm, "JURY_DEMAND") %>% plot() + ggtitle("GBM")
#
#
# temp <- data.frame( rf.jur$data$JURY_DEMAND, rf.jur$data$.y.hat)
# g <- ggplot(temp, aes(y=rf.jur.data..y.hat, x = rf.jur.data.JURY_DEMAND)) +
#   geom_boxplot()  + labs(x = "JURY_DEMAND", y = 'Open_Days', title="Random Forest")  +
#   theme(plot.title = element_text(hjust = 0.5))
# ggsave(file="ice_rf_boxplot_JURY_DEMAND.png", g)
#
# temp <- data.frame(gbm.jur$data$JURY_DEMAND, gbm.jur$data$.y.hat)
# g <- ggplot(temp, aes(y=gbm.jur.data..y.hat, x = gbm.jur.data.JURY_DEMAND)) +
#   geom_boxplot() +labs(x = "JURY_DEMAND", y = 'Open_Days', title="GBM")  +
#   theme(plot.title = element_text(hjust = 0.5))
# ggsave(file="ice_gbm_boxplot_JURY_DEMAND.png", g)

knitr::include_graphics("ice_rf_boxplot_ORIGIN_CAT.png")
knitr::include_graphics("ice_gbm_boxplot_ORIGIN_CAT.png")
knitr::include_graphics("ice_rf_boxplot_CIRCUIT.png")
knitr::include_graphics("ice_gbm_boxplot_CIRCUIT.png")
knitr::include_graphics("ice_rf_boxplot_DISTRICT_CAT.png")
knitr::include_graphics("ice_gbm_boxplot_DISTRICT_CAT.png")
knitr::include_graphics("ice_rf_boxplot_JURY_DEMAND.png")
knitr::include_graphics("ice_gbm_boxplot_JURY_DEMAND.png")
```

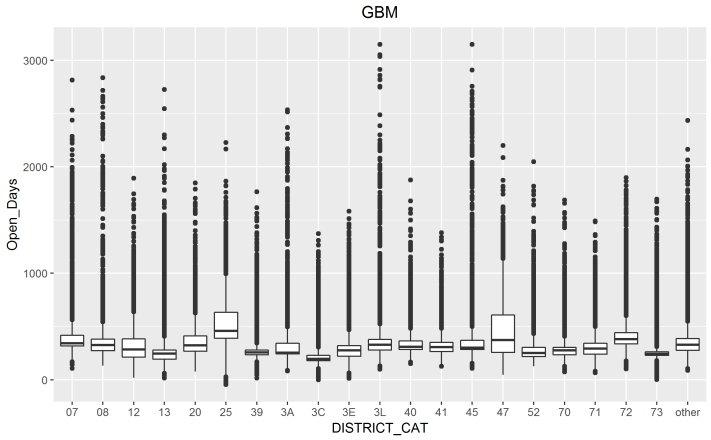

Random Forest
Open_Days
ORIGIN_CAT

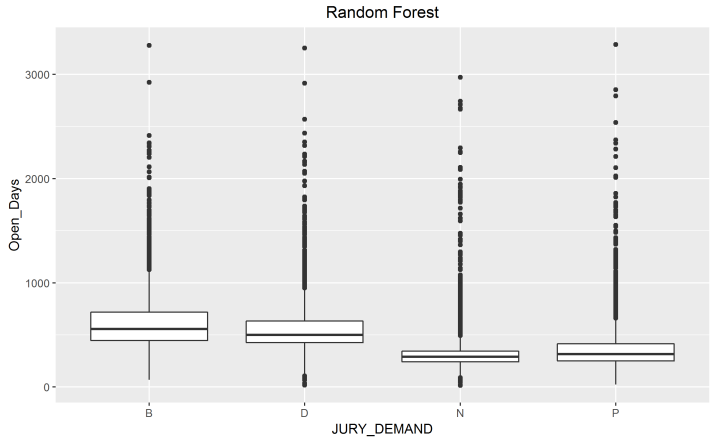

GBM
Open_Days
ORIGIN_CAT

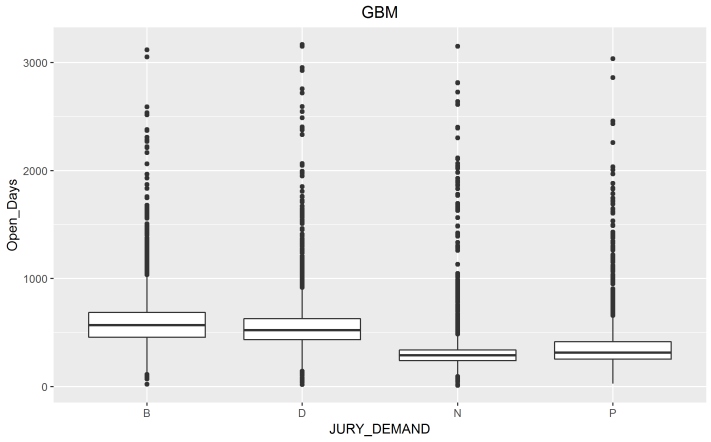

rf.cir.data..y.hat
rf.cir.data.CIRCUIT


GBM
Open_Days
CIRCUIT


Random Forest
Open_Days
DISTRICT_CAT


GBM
Open_Days
DISTRICT_CAT


Random Forest
Open_Days
JURY_DEMAND


GBM
Open_Days
JURY_DEMAND

Question 5:

How could we use this data to improve our legal operations in Liberty Mutual? There is no need to write down anything specific in your reports. In your interview, you will be asked to share your thought with us.

As suggested, I am not writing down an answer for this question, and will be talking about this question in the interview.