

Different Models

Zhuocheng Lin

3/22/2020

1. Packages

```
library(tidyverse)
library(modelr)
library(caret)
library(ROSE)
library(randomForest)
library(glmnet)
library(rpart)
library(BMA)
```

2. Data pre-processing

(1) Read data

```
df <- read_csv("./tidy.csv", col_types = cols(.default = col_character())) %>%
  type_convert()
```

(2) Specify factors

```
df_format <- df %>%
  mutate(TMC = factor(TMC), Severity = factor(Severity), Year = factor(Year), Wday = factor(Wday)) %>%
  mutate_if(is.logical, factor) %>%
  mutate_if(is.character, factor)
```

(3) Narrow down to one State

```
df_format %>% count(State) %>% arrange(desc(n))
```

```
## # A tibble: 49 x 2
##   State      n
##   <fct> <int>
## 1 CA      650285
## 2 TX      291281
## 3 FL      221148
## 4 SC      143606
## 5 NC      141397
## 6 NY      136288
## 7 PA       89120
```

```
## 8 MI      88488
## 9 IL      86105
## 10 GA     82547
## # ... with 39 more rows

# choose TX as the target State
df_TX <- df_format %>% filter(State == "TX") %>% select(-State)
```

(4) Remove unuseful variables

```
# remove variables with only 1 distinct value
df_TX %>% summarise_all(~ n_distinct(.)) %>%
  pivot_longer(everything(), names_to = "variable", values_to = "n") %>% filter(n == 1)

## # A tibble: 1 x 2
##   variable      n
##   <chr>      <int>
## 1 Turning_Loop    1

df_TX <- df_TX %>% select(-Turning_Loop)
```

(5) Drop weather condition/TMC levels

```
# some Weather_Condition levels only have a few observations
# which can be a problem when we try to build a model
df_TX %>% count(Weather_Condition) %>% filter(n < 20) %>% select(Weather_Condition)

## # A tibble: 18 x 1
##   Weather_Condition
##   <fct>
## 1 Blowing Dust
## 2 Drizzle and Fog
## 3 Haze / Windy
## 4 Heavy Drizzle
## 5 Heavy T-Storm / Windy
## 6 Light Drizzle / Windy
## 7 Light Freezing Fog
## 8 Light Haze
## 9 Light Ice Pellets
## 10 Light Rain Showers
## 11 Light Snow / Windy
## 12 N/A Precipitation
## 13 Rain Showers
## 14 Sand
## 15 Showers in the Vicinity
## 16 Smoke
## 17 Thunder / Windy
## 18 Wintry Mix

drop_weather <- df_TX %>% count(Weather_Condition) %>% filter(n < 20) %>% select(Weather_Condition)
drop_weather <- drop_weather$Weather_Condition %>% unlist()
df_TX <- df_TX %>% filter(!(Weather_Condition %in% drop_weather))
df_TX <- df_TX %>% mutate(Weather_Condition = factor(Weather_Condition))

# it's the same to TMC
```

```
df_TX %>% count(TMC) %>% filter(n < 10)

## # A tibble: 5 x 2
##   TMC      n
##   <fct> <int>
## 1 200      1
## 2 239      3
## 3 248      4
## 4 336      1
## 5 339      9

drop_TMC <- df_TX %>% count(TMC) %>% filter(n < 10) %>% select(TMC)
drop_TMC <- drop_TMC$TMC %>% unlist()
df_TX <- df_TX %>% filter(!TMC %in% drop_TMC) %>% mutate(TMC = factor(TMC))
```

(6) Add new labels

```
# group level 3 and 4 together, as "Severe"
# group level 1 and 2 together, as "Not Severe"
df_label <- df_TX %>%
  mutate("Status" = factor(ifelse(Severity == "3" | Severity == "4", "Severe", "Not Severe"),
                             levels = c("Not Severe", "Severe")))
df_label %>% select(Severity, Status)

## # A tibble: 291,138 x 2
##   Severity Status
##   <fct>    <fct>
## 1 2      Not Severe
## 2 2      Not Severe
## 3 2      Not Severe
## 4 2      Not Severe
## 5 3      Severe
## 6 2      Not Severe
## 7 2      Not Severe
## 8 3      Severe
## 9 3      Severe
## 10 2     Not Severe
## # ... with 291,128 more rows
```

(7) Near Zero-Variance Predictors

```
# these variable may become zero-variance when the data are split into subsets
# remove them
nzv <- nearZeroVar(df_label, saveMetrics = T)
nzv[nzv$nzv,]

##           freqRatio percentUnique zeroVar  nzv
## Visibility      21.90824   0.0151131079 FALSE TRUE
## Amenity         52.04026   0.0006869594 FALSE TRUE
## Bump          5597.80769   0.0006869594 FALSE TRUE
## Give_Way       176.09124   0.0006869594 FALSE TRUE
## No_Exit        969.46000   0.0006869594 FALSE TRUE
## Railway        106.70921   0.0006869594 FALSE TRUE
## Roundabout     26466.09091   0.0006869594 FALSE TRUE
```

```
## Station          62.63672  0.0006869594  FALSE TRUE
## Stop             66.12889  0.0006869594  FALSE TRUE
## Traffic_Calming 2598.44643  0.0006869594  FALSE TRUE

nzv_cols <- rownames(nzv[nzv$nzv,])
df_label <- df_label %>%
  select(-nzv_cols)
```

(8) Partition

```
set.seed(1)
df_parts <- resample_partition(df_label, c(train = 0.6, valid = 0.2, test = 0.2))
train <- as_tibble(df_parts$train)
valid <- as_tibble(df_parts$valid)
test <- as_tibble(df_parts$test)
# check Weather_Condition levels / TMC levels
# train should have more levels than valid and test
tr <- train %>% select(Weather_Condition) %>% distinct()
va <- valid %>% select(Weather_Condition) %>% distinct()
te <- test %>% select(Weather_Condition) %>% distinct()
setdiff(va, tr)

## # A tibble: 0 x 1
## # ... with 1 variable: Weather_Condition <fct>
setdiff(te, tr)

## # A tibble: 0 x 1
## # ... with 1 variable: Weather_Condition <fct>

tr <- train %>% select(TMC) %>% distinct()
va <- valid %>% select(TMC) %>% distinct()
te <- test %>% select(TMC) %>% distinct()
setdiff(va, tr)

## # A tibble: 0 x 1
## # ... with 1 variable: TMC <fct>
setdiff(te, tr)

## # A tibble: 0 x 1
## # ... with 1 variable: TMC <fct>
```

(9) Sampling

```
new_train <- ovun.sample(Status ~ .,
  data = train %>% select(-Severity),
  method = "both", p = 0.5, N = 90000)$data %>% as_tibble()
table(new_train$Status)

##
## Not Severe      Severe
##      44981      45019
```

3. Use different models to fit the data

(1) Linear Regression

```
new_train_linear <- new_train %>%
  mutate(Status = as.numeric(recode(Status, "Not Severe" = 0, "Severe" = 1)))

# lm_total <- lm(Status ~ ., data = new_train_linear)
# step(lm_total)
# based on the result of step() function, build the models below
lm_ <- list()
lm_[[1]] <- lm(Status ~ TMC, data = new_train_linear)
lm_[[2]] <- lm(Status ~ TMC + Side, data = new_train_linear)
lm_[[3]] <- lm(Status ~ TMC + Side + Traffic_Signal, data = new_train_linear)
lm_[[4]] <- lm(Status ~ TMC + Side + Traffic_Signal + Start_Lat, data = new_train_linear)
lm_[[5]] <- lm(Status ~ TMC + Side + Traffic_Signal + Start_Lat + Hour, data = new_train_linear)
lm_[[6]] <- lm(Status ~ TMC + Side + Traffic_Signal + Start_Lat + Hour +
  Junction, data = new_train_linear)
lm_[[7]] <- lm(Status ~ TMC + Side + Traffic_Signal + Start_Lat + Hour +
  Junction + Wday, data = new_train_linear)
lm_[[8]] <- lm(Status ~ TMC + Side + Traffic_Signal + Start_Lat + Hour +
  Junction + Wday + Distance, data = new_train_linear)
lm_[[9]] <- lm(Status ~ TMC + Side + Traffic_Signal + Start_Lat + Hour +
  Junction + Wday + Distance + Crossing, data = new_train_linear)
lm_[[10]] <- lm(Status ~ TMC + Side + Traffic_Signal + Start_Lat + Hour +
  Junction + Wday + Distance + Crossing + Duration, data = new_train_linear)
lm_[[11]] <- lm(Status ~ TMC + Side + Traffic_Signal + Start_Lat + Hour +
  Junction + Wday + Distance + Crossing + Duration + Year, data = new_train_linear)
lm_[[12]] <- lm(Status ~ TMC + Side + Traffic_Signal + Start_Lat + Hour +
  Junction + Wday + Distance + Crossing + Duration + Year +
  Start_Lng, data = new_train_linear)
lm_[[13]] <- lm(Status ~ TMC + Side + Traffic_Signal + Start_Lat + Hour +
  Junction + Wday + Distance + Crossing + Duration + Year +
  Start_Lng + Pressure, data = new_train_linear)
lm_[[14]] <- lm(Status ~ TMC + Side + Traffic_Signal + Start_Lat + Hour +
  Junction + Wday + Distance + Crossing + Duration + Year +
  Start_Lng + Pressure + Weather_Condition, data = new_train_linear)
lm_[[15]] <- lm(Status ~ TMC + Side + Traffic_Signal + Start_Lat + Hour +
  Junction + Wday + Distance + Crossing + Duration + Year +
  Start_Lng + Pressure + Weather_Condition + Month, data = new_train_linear)
lm_[[16]] <- lm(Status ~ TMC + Side + Traffic_Signal + Start_Lat + Hour +
  Junction + Wday + Distance + Crossing + Duration + Year +
  Start_Lng + Pressure + Weather_Condition + Month + Wind_Speed, data = new_train_linear)
lm_[[17]] <- lm(Status ~ TMC + Side + Traffic_Signal + Start_Lat + Hour +
  Junction + Wday + Distance + Crossing + Duration + Year +
  Start_Lng + Pressure + Weather_Condition + Month + Wind_Speed + Civil_Twilight, data = new_train_linear)
lm_[[18]] <- lm(Status ~ TMC + Side + Traffic_Signal + Start_Lat + Hour +
  Junction + Wday + Distance + Crossing + Duration + Year +
  Start_Lng + Pressure + Weather_Condition + Month + Wind_Speed + Civil_Twilight + Humidity, data = new_train_linear)
lm_[[19]] <- lm(Status ~ TMC + Side + Traffic_Signal + Start_Lat + Hour +
  Junction + Wday + Distance + Crossing + Duration + Year +
  Start_Lng + Pressure + Weather_Condition + Month + Wind_Speed + Civil_Twilight + Humidity, data = new_train_linear)

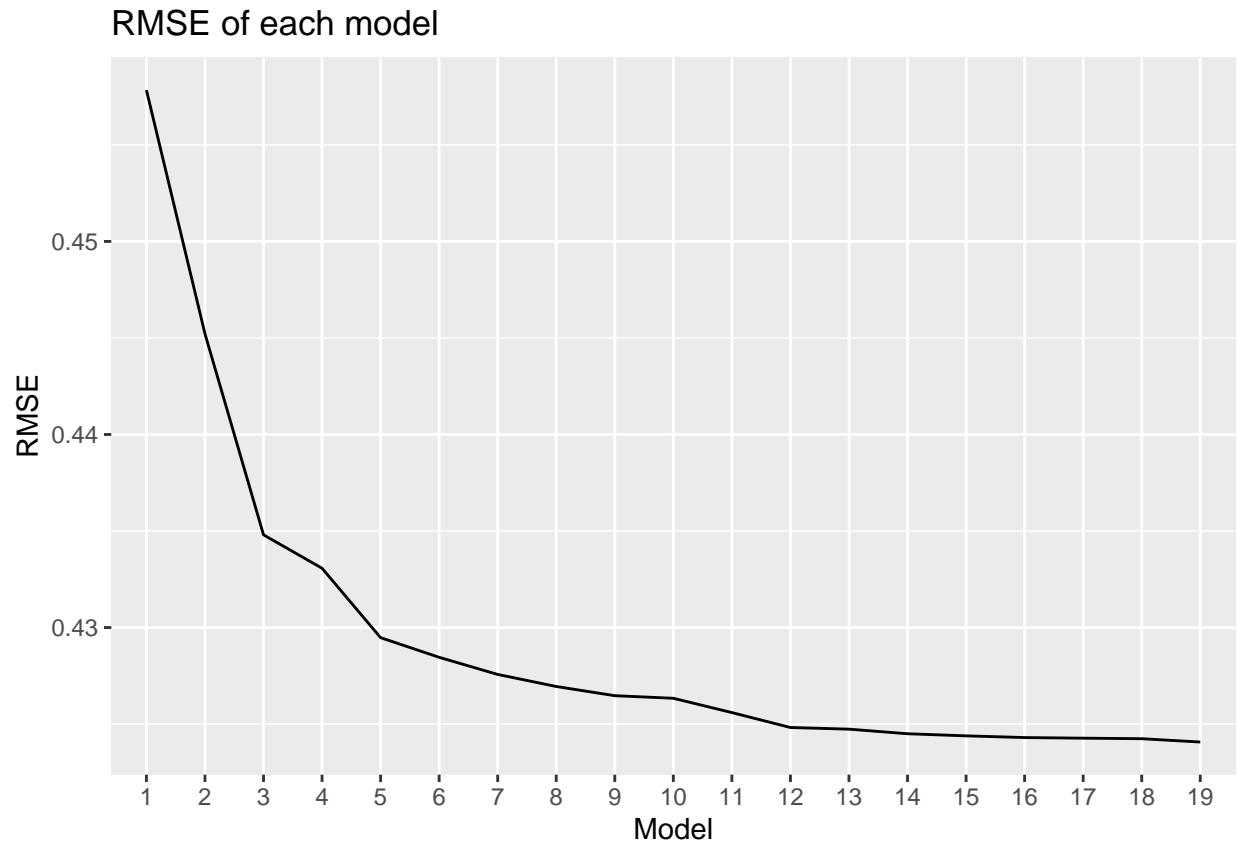
all_rmse = vector(length = 19)
```

```

for (i in 1:19) {
  all_rmse[i] <- rmse(lm_[[i]], data = new_train_linear)
}

# choose 15th model as the best
ggplot(tibble(rmse = all_rmse)) +
  geom_line(aes(factor(1:19), rmse, group = 1)) +
  labs(x = "Model",
       y = "RMSE",
       title = "RMSE of each model")

```

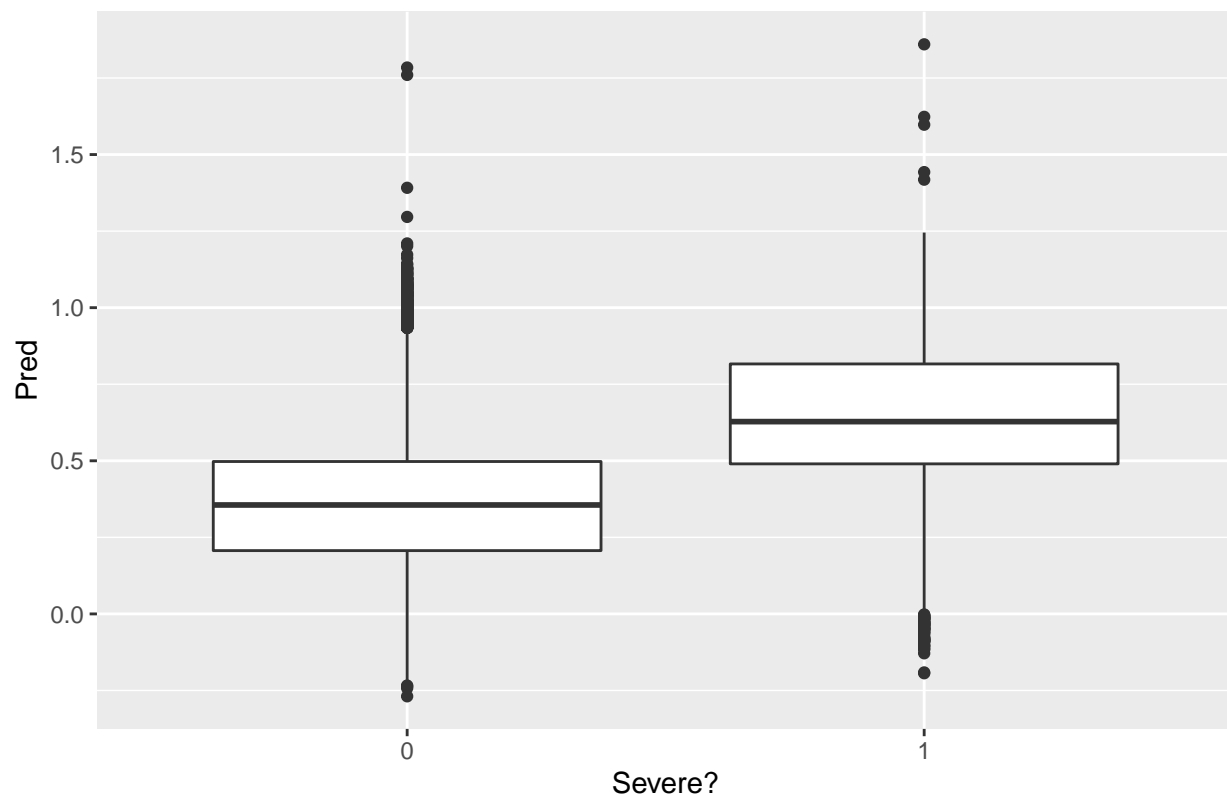


```

# get predictions on valid dataset
valid_pred_lm <- valid %>%
  mutate(Status = as.numeric(recode(Status, "Not Severe" = 0, "Severe" = 1))) %>%
  add_predictions(lm_[[15]])
# see the predicted value distribution
ggplot(valid_pred_lm) +
  geom_boxplot(aes(x = factor(Status), y = pred)) +
  labs(x = "Severe?",
       y = "Pred",
       title = "The predicted value distribution on each status")

```

The predicted value distribution on each status



```
# choose 0.5 as the boundary

valid_pred_lm <- valid_pred_lm %>%
  mutate(pred_status = ifelse(pred > 0.5, 1, 0))
# accuracy
confusionMatrix(table(valid_pred_lm$pred_status, valid_pred_lm$Status))
```

```
## Confusion Matrix and Statistics
##
##
##      0      1
## 0 31614 4369
## 1 10308 11937
##
##              Accuracy : 0.7479
##              95% CI : (0.7444, 0.7515)
##      No Information Rate : 0.72
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.4375
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##              Sensitivity : 0.7541
##              Specificity : 0.7321
##              Pos Pred Value : 0.8786
```

```
##           Neg Pred Value : 0.5366
##           Prevalence : 0.7200
##           Detection Rate : 0.5429
##   Detection Prevalence : 0.6180
##           Balanced Accuracy : 0.7431
##
##           'Positive' Class : 0
##
```

(5) Logistic Regression

```
# BIC best model
# model_bic <- bic.glm(Status ~ ., data = new_train, glm.family = "binomial")
# saveRDS(model_bic, "./model_bic.rds")
model_bic <- readRDS("./model_bic.rds")
variables <- model_bic$label[1] %>% str_split(pattern = ",") %>% unlist()
bic_best <- glm(Status ~ ., data = new_train %>% select(variables, Status), family = "binomial")
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(variables)` instead of `variables` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

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

```
valid_pred <- valid %>%
  mutate("lg" = predict(bic_best, newdata = valid, type = "response")) %>%
  mutate("lg" = ifelse(lg > 0.5, "Severe", "Not Severe"))
valid_pred %>% select(Status, lg)
```

```
## # A tibble: 58,228 x 2
##   Status    lg
##   <fct>    <chr>
## 1 Not Severe Not Severe
## 2 Not Severe Not Severe
## 3 Not Severe Not Severe
## 4 Severe     Severe
## 5 Severe     Severe
## 6 Not Severe Not Severe
## 7 Not Severe Not Severe
## 8 Not Severe Not Severe
## 9 Not Severe Not Severe
## 10 Not Severe Not Severe
## # ... with 58,218 more rows
```

```
table(valid$Status)
```

```
##
## Not Severe    Severe
##      41922      16306
```

```
confusionMatrix(table(valid_pred$lg, valid_pred$Status))
```

```
## Confusion Matrix and Statistics
##
##
##           Not Severe Severe
```



```

## Not Severe      31266   4227
## Severe          10656  12079
##
## Accuracy : 0.7444
## 95% CI : (0.7408, 0.7479)
## No Information Rate : 0.72
## P-Value [Acc > NIR] : < 2.2e-16
##
## Kappa : 0.4343
##
## McNemar's Test P-Value : < 2.2e-16
##
## Sensitivity : 0.7458
## Specificity : 0.7408
## Pos Pred Value : 0.8809
## Neg Pred Value : 0.5313
## Prevalence : 0.7200
## Detection Rate : 0.5370
## Detection Prevalence : 0.6096
## Balanced Accuracy : 0.7433
##
## 'Positive' Class : Not Severe
##

# AIC best model
# model_aic <- glm(Status ~ ., data = new_train, family = "binomial")
# model_aic <- step(model_aic)
# saveRDS(model_aic, "./model_aic.rds")
model_aic <- readRDS("./model_aic.rds")
# these are the dropped variables
model_aic$anova

##           Step Df  Deviance Resid. Df Resid. Dev      AIC
## 1              NA      NA      89856   95311.56 95599.56
## 2 - Nautical_Twilight  1 0.5852068    89857   95312.15 95598.15
## 3   - Civil_Twilight  1 0.6463587    89858   95312.80 95596.80

aic_best <- glm(Status ~ TMC + Side + Traffic_Signal + Hour +
  Start_Lat + Distance + Wday + Junction +
  Crossing + Duration + Year + Start_Lng +
  Wind_Speed + Pressure + Month + Day +
  Sunrise_Sunset + Astronomical_Twilight +
  Temperature + Weather_Condition + Humidity,
  data = new_train,
  family = "binomial")

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

valid_pred <- valid %>%
  mutate("lg" = predict(aic_best, newdata = valid, type = "response")) %>%
  mutate("lg" = ifelse(lg > 0.5, "Severe", "Not Severe"))
valid_pred %>% select(Status, lg)

## # A tibble: 58,228 x 2
##   Status    lg
##   <fct>    <chr>

```

```
## 1 Not Severe Not Severe
## 2 Not Severe Not Severe
## 3 Not Severe Not Severe
## 4 Severe      Severe
## 5 Severe      Severe
## 6 Not Severe Not Severe
## 7 Not Severe Not Severe
## 8 Not Severe Not Severe
## 9 Not Severe Not Severe
## 10 Not Severe Not Severe
## # ... with 58,218 more rows
```

```
table(valid$Status)
```

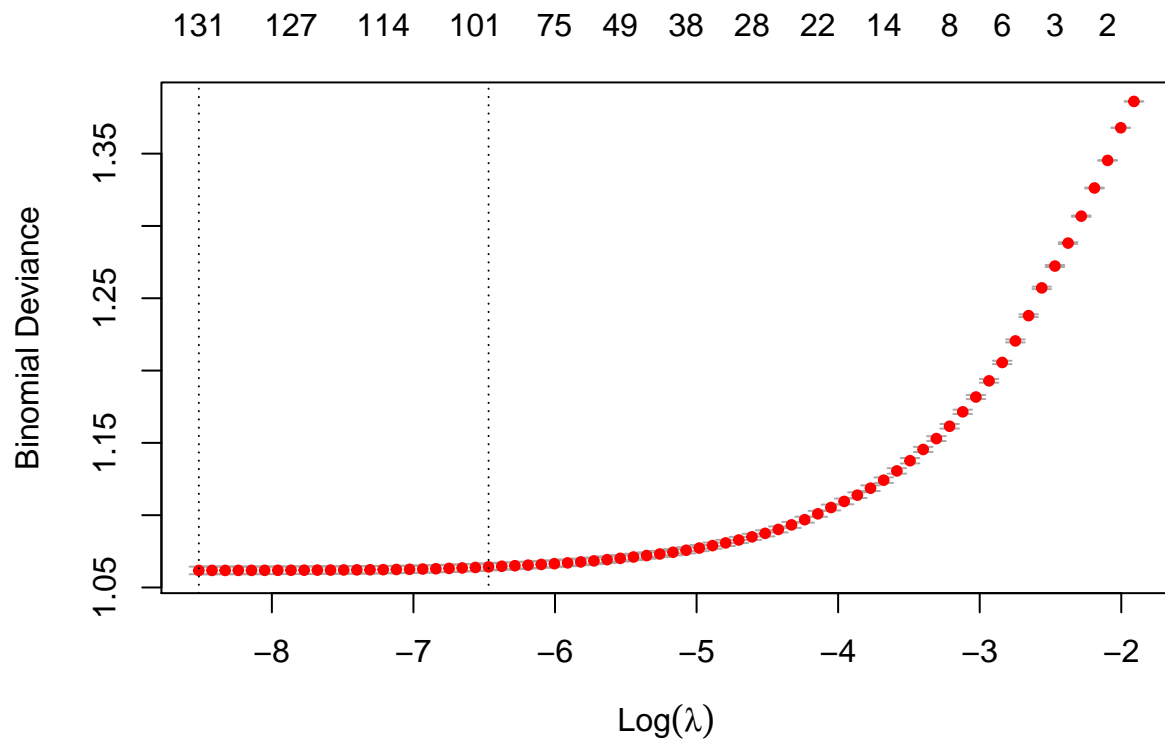
```
##
## Not Severe      Severe
##      41922      16306
```

```
confusionMatrix(table(valid_pred$lg, valid_pred$Status))
```

```
## Confusion Matrix and Statistics
##
##
##              Not Severe Severe
## Not Severe      31302  4216
## Severe          10620 12090
##
##              Accuracy : 0.7452
##              95% CI : (0.7416, 0.7487)
##              No Information Rate : 0.72
##              P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.4358
##
## Mcnemar's Test P-Value : < 2.2e-16
##
##              Sensitivity : 0.7467
##              Specificity : 0.7414
##              Pos Pred Value : 0.8813
##              Neg Pred Value : 0.5324
##              Prevalence : 0.7200
##              Detection Rate : 0.5376
##              Detection Prevalence : 0.6100
##              Balanced Accuracy : 0.7441
##
##              'Positive' Class : Not Severe
##
```

(3) Sparse Logistic regression

```
x <- model.matrix(Status ~ ., data = new_train)
model_total <- glmnet(x, new_train$Status, family = "binomial")
plot(model_total, xvar = "lambda", label = T)
```

```
# use the best lambda
valid_pred <- valid %>%
  mutate("slg" = predict(model_lambda,
    newx = model.matrix(Status ~ ., data = valid %>% select(-Severity)),
    s = "lambda.min", type = "response")[,1]) %>%
  mutate("slg" = ifelse(slg > 0.5, "Severe", "Not Severe"))
valid_pred %>% select(Status, slg)
```

```
## # A tibble: 58,228 x 2
##   Status    slg
##   <fct>    <chr>
## 1 Not Severe Not Severe
## 2 Not Severe Not Severe
## 3 Not Severe Not Severe
## 4 Severe     Severe
## 5 Severe     Severe
## 6 Not Severe Not Severe
## 7 Not Severe Not Severe
## 8 Not Severe Not Severe
## 9 Not Severe Not Severe
## 10 Not Severe Not Severe
## # ... with 58,218 more rows
```

```
table(valid$Status)
```

```
##
## Not Severe    Severe
```

```
##          41922          16306
confusionMatrix(table(valid_pred$slg, valid_pred$Status))

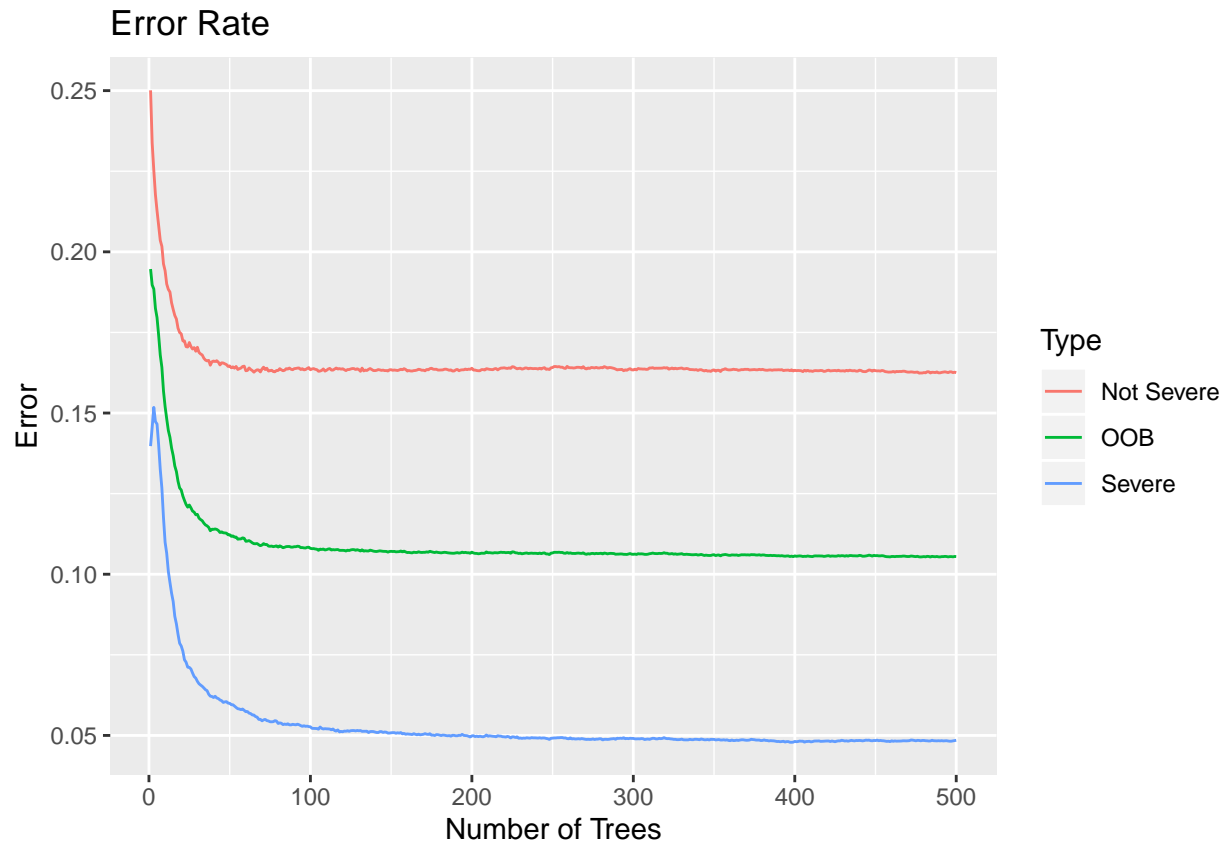
## Confusion Matrix and Statistics
##
##
##           Not Severe Severe
## Not Severe      31305  4212
## Severe          10617 12094
##
##           Accuracy : 0.7453
##           95% CI : (0.7418, 0.7489)
## No Information Rate : 0.72
## P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.4361
##
## Mcnemar's Test P-Value : < 2.2e-16
##
##           Sensitivity : 0.7467
##           Specificity : 0.7417
##           Pos Pred Value : 0.8814
##           Neg Pred Value : 0.5325
##           Prevalence : 0.7200
##           Detection Rate : 0.5376
## Detection Prevalence : 0.6100
##           Balanced Accuracy : 0.7442
##
##           'Positive' Class : Not Severe
##
# write_csv(valid_pred, "./results/TX_valid.csv")
```

(4) Random forest

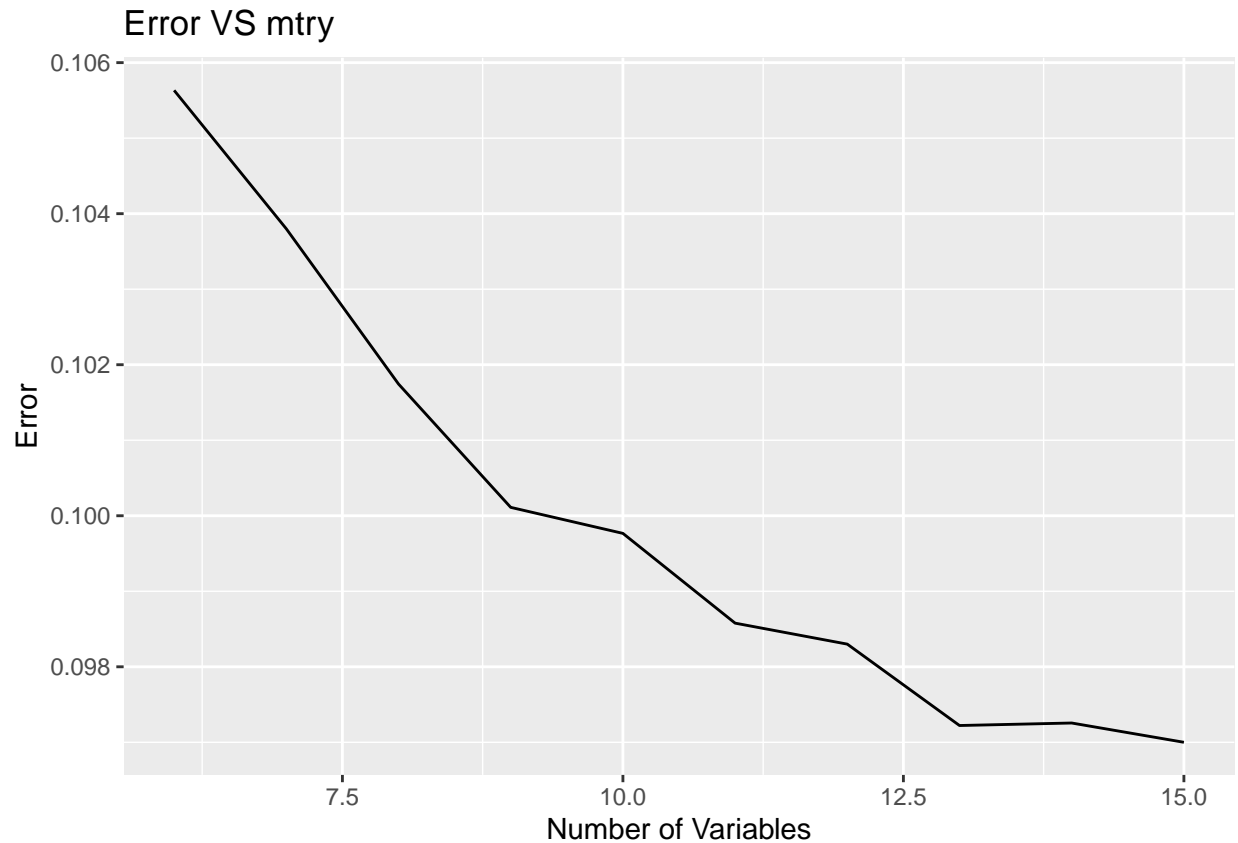
```
model <- randomForest(Status ~ ., data = new_train, mtry = 6, ntree = 500)

# see if ntree = 500 is enough
error_data <- model$err.rate %>%
  as_tibble() %>%
  mutate("Trees" = seq_along(OOB)) %>%
  pivot_longer(cols = 1:3, names_to = "Type", values_to = "Error")

ggplot(error_data, aes(Trees, Error, color = Type)) +
  geom_line() +
  labs(x = "Number of Trees",
       title = "Error Rate")
```



```
# try different mtry
oob_values <- vector(length = 10)
for (i in 1:10) {
  temp_model <- randomForest(Status ~ ., data = new_train, mtry = (i + 5))
  oob_values[i] <- temp_model$err.rate[nrow(temp_model$err.rate), 1]
}
ggplot(tibble("Error" = oob_values), aes(x = 6:15, y = Error)) +
  geom_line(aes(group = 1)) +
  labs(x = "Number of Variables",
       title = "Error VS mtry")
```



```
# choose mtry = 13 as the best model
best_model <- randomForest(Status ~ ., data = new_train, mtry = 13, ntree = 500)

valid_pred_rf <- valid %>%
  add_predictions(best_model)
table(valid_pred_rf$Status)
```

```
##
## Not Severe      Severe
##      41922      16306

confusionMatrix(valid_pred_rf$pred, valid_pred_rf$Status)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction  Not Severe Severe
## Not Severe    33167   2175
## Severe        8755  14131
##
##              Accuracy : 0.8123
##              95% CI : (0.8091, 0.8155)
## No Information Rate : 0.72
## P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.5856
##
```

```
## McNemar's Test P-Value : < 2.2e-16
##
##      Sensitivity : 0.7912
##      Specificity : 0.8666
##      Pos Pred Value : 0.9385
##      Neg Pred Value : 0.6175
##      Prevalence : 0.7200
##      Detection Rate : 0.5696
##      Detection Prevalence : 0.6070
##      Balanced Accuracy : 0.8289
##
##      'Positive' Class : Not Severe
##
```

(5) Decision tree

```
model_decision <- rpart(Status ~ ., data = new_train, method = "class")

valid_pred_dc <- valid %>%
  mutate(pred = predict(model_decision, valid, type = "class"))
confusionMatrix(table(valid_pred_dc$pred, valid_pred_dc$Status))
```

```
## Confusion Matrix and Statistics
##
##
##      Not Severe Severe
## Not Severe    30990  4044
## Severe        10932 12262
##
##      Accuracy : 0.7428
##      95% CI : (0.7392, 0.7464)
##      No Information Rate : 0.72
##      P-Value [Acc > NIR] : < 2.2e-16
##
##      Kappa : 0.4351
##
## McNemar's Test P-Value : < 2.2e-16
##
##      Sensitivity : 0.7392
##      Specificity : 0.7520
##      Pos Pred Value : 0.8846
##      Neg Pred Value : 0.5287
##      Prevalence : 0.7200
##      Detection Rate : 0.5322
##      Detection Prevalence : 0.6017
##      Balanced Accuracy : 0.7456
##
##      'Positive' Class : Not Severe
##
```