Predict Injury from Crowdsourced Bike Incidents

2023-12-14

Contents

```r
#install.packages("tidyverse")a
#install.packages("tidymodels")
#install.packages("kknn")
#install.packages("yardstick")
#install.packages("ggplot2")
#install.packages("ggcorrplot")
library(tidyverse)
library(tidymodels)
tidymodels_prefer()
```

Introduction

The aim of this project is to predict whether a bicyclist will injure from a bike incident or not. I use crowdsourced near-miss and collision data from bikemaps.org. Bikemaps.org is crowsourcing platform where bicyclists can map their incidents and record relevant information, including collision/near-miss details, environmental conditions, personal info and injury level. Four machine learning models are applied to yield the most accurate model for this binary classification problem.

Why predict injury from bike incidents

Biking is sustainable transportation mode growing more and more popular these days. However, bicyclists are one of the most vulnerable road user group in daily travel. I want to predict whether a bicyclists will injure from an incident and understand the factors that is important to the prediction of injury risks. In that way, we can adopt preventive measures to reduce the risk of injury and thus protect bicyclists.

Data description

The data collection happens at this website: https://bikemaps.org. Cyclists can drop a point on the interactive map on the website and they will be invited to fill out a survey. The data is available at this link: https://bikemaps.org/incidents-only.json. As most of the questions are multiple choice, most variables we can leverage are categorical variables. Here are the list of the variables:

`i_type`:incident type
`incident_with`: the object with which the incident happened with
`date`: date of the incident
`p_type`:incident type with more information
`personal_involvement`: whether the person who report the incident is involved in the incident (otherwise as witness) `details`: text description of what happen
`impact`: whether the incident has any impact on the reporter's decision of future biking behavior
`injury`: degree of injury caused by the incident `trip_purpose`: whether the trip is for commute, recreation, social, personal business or during work
`regular_cyclist`: whether the cyclist who involved in the incident is a regular cyclist or not `helmet`: whether the cyclist who involved in the incident wear a helmet or not
`road_conditions`: the road condition is dry, wet, icy or snowy `sightlines`: the condition of sightlines while the incident happened `cars_on_roadside`: whether there are cars on the road side or not while the incident happened `bike_lights`: whether the cyclist used bike light `terrain`: the terrain where the incident happened
`aggressive`: removed from the questionnaire `intersection`: whether the incident happened at an intersection
`witness_vehicle`: removed from the questionnaire `bicycle_type`: type of bike involved in the incident `ebike`: whether the bike involved is an ebike
`ebike_class`: if it is an ebike what is the ebike class
`ebike_speed`: if it is an ebike what is the ebike speed
`direction` : the direction cyclist was heading
`turning`: how the cyclist was moving, heading straight, turning left or right
`age`: age of the reporter
`gender`: gender of the reporter
`birthmonth`: birth month of the reporter
`pk`: index of the incident report
`longitude`: longitude of the place where the incident took place `latitude`: latitude of the place where the incident took place

Project Roadmap

I will first clean and manipulate the data before performing exploratory data analysis to get a better sense of the variables. Our goal is to predict a binary class of whether a bicylist is injured or not from the incident details provided. We will perform a train/test split on the data, set folds for 5-folds cross validation and make a recipe. K-nearest neighbor, Logistic regression, Elastic net and Random forest will be applied to model the training data. I will then augment the best performing model on our testing data and see how good my model is able to predict injury.

Data Cleaning and manipulation

Loading raw data

First let's look at what we have in our data.

```
data <- read_csv("world.xlsx - Incident.csv")
colnames(data)
```

```
##  [1] "i_type"               "incident_with"        "date"
##  [4] "p_type"               "personal_involvement" "details"
##  [7] "impact"               "injury"               "trip_purpose"
## [10] "regular_cyclist"      "helmet"               "road_conditions"
## [13] "sightlines"           "cars_on_roadside"     "bike_lights"
## [16] "terrain"              "aggressive"           "intersection"
## [19] "witness_vehicle"      "bicycle_type"         "ebike"
## [22] "ebike_class"          "ebike_speed"          "direction"
## [25] "turning"              "age"                  "gender"
## [28] "birthmonth"           "pk"                   "longitude"
## [31] "latitude"
```

There's a lot of variables! However, not all of them are relevant in predicting injury. Let's remove the temporal and spatial information, details (text information), impact of the injury and pk (the incident index) and direction.

Bikemaps.org started to collect bike incidents from the public from 2014. In this study, we only excerpt the data after December 2016 because that's when users can identify whether they are on an e-bike in the report survey. I think it will be interesting to see if factoring in e-bike can help predict injury level from bike incidents. So before dropping the temporal information, we will filter the data first.

```
library(dplyr)
data <- data %>% dplyr::filter(date > as.Date('2016-11-30'))

data <- data %>%
  #drop column that are not related to injury prediction
  dplyr::select(-c(date,details,birthmonth,impact,pk,longitude,latitude,direction))
```

In addition, if the reporter is not involved in the incident, the age and gender information is not relevant for predict the injury. So we will see how many reports are generated by persons that are not directly involved in the incident but witnesses themselves. If not too many we will drop them as well.

```
summary(as.factor(data$personal_involvement))
```

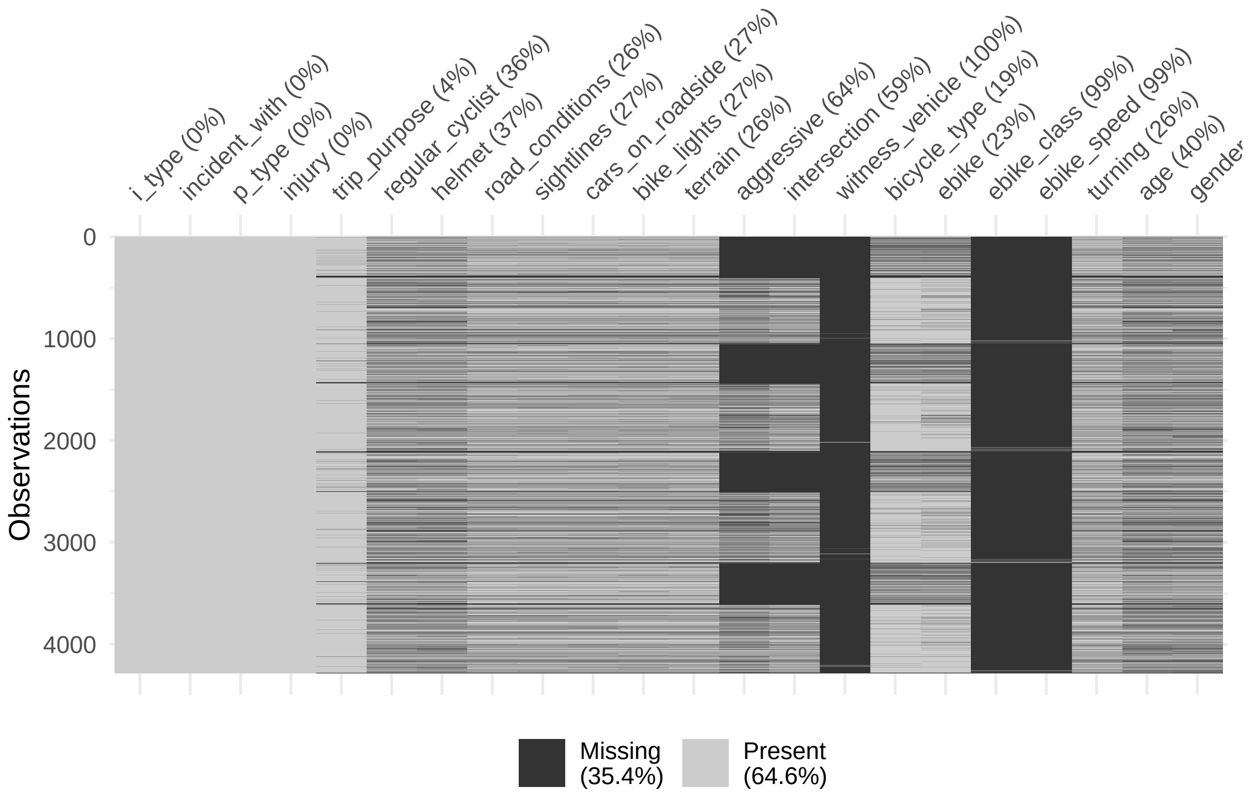
```
##   No  Yes NA's
##  796 3850  434
```

It seems around 15% of the reports are added by witness. We will drop those and assume the NA's are self-reported (I know, this is a big assumption.. But the NA' also consisted of more than 8% of all data, so we will keep them with this assumption in mind when we draw conclusion from the analysis.)

```
data <- data %>%
  filter(is.na(personal_involvement)|personal_involvement!="No") %>%
  select(-personal_involvement)
```

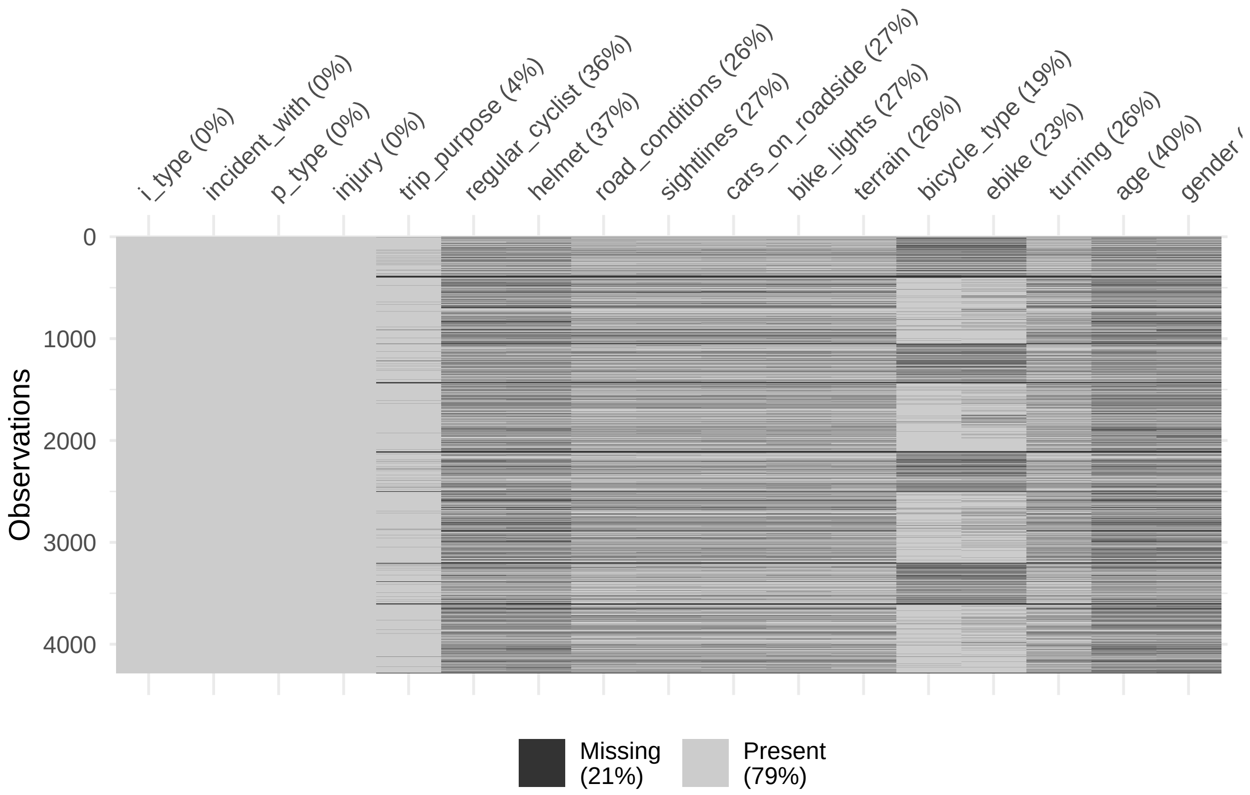
Missing Data Visualization

Now let's see how many missing data we have and decide how we can deal with them.



That's a lot of missing data... It doesn't make sense to impute for those column that has more than 50% missing. Let's drop those column.

```
#drop all column with more than 50% missing
data <- data %>%
  purrr::discard(~sum(is.na(.x))/length(.x)* 100 >=50)
vis_miss(data)
```



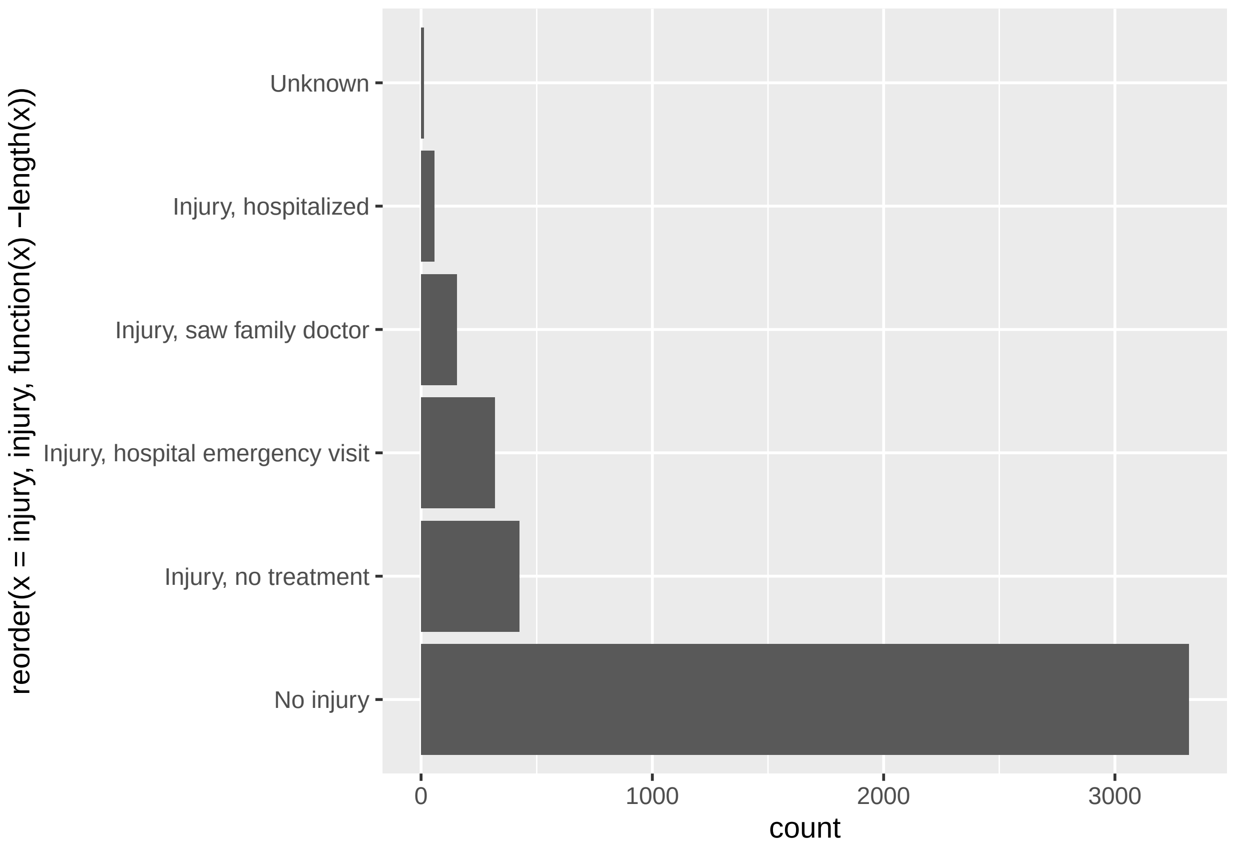
That looks better! Let's start our data exploration!

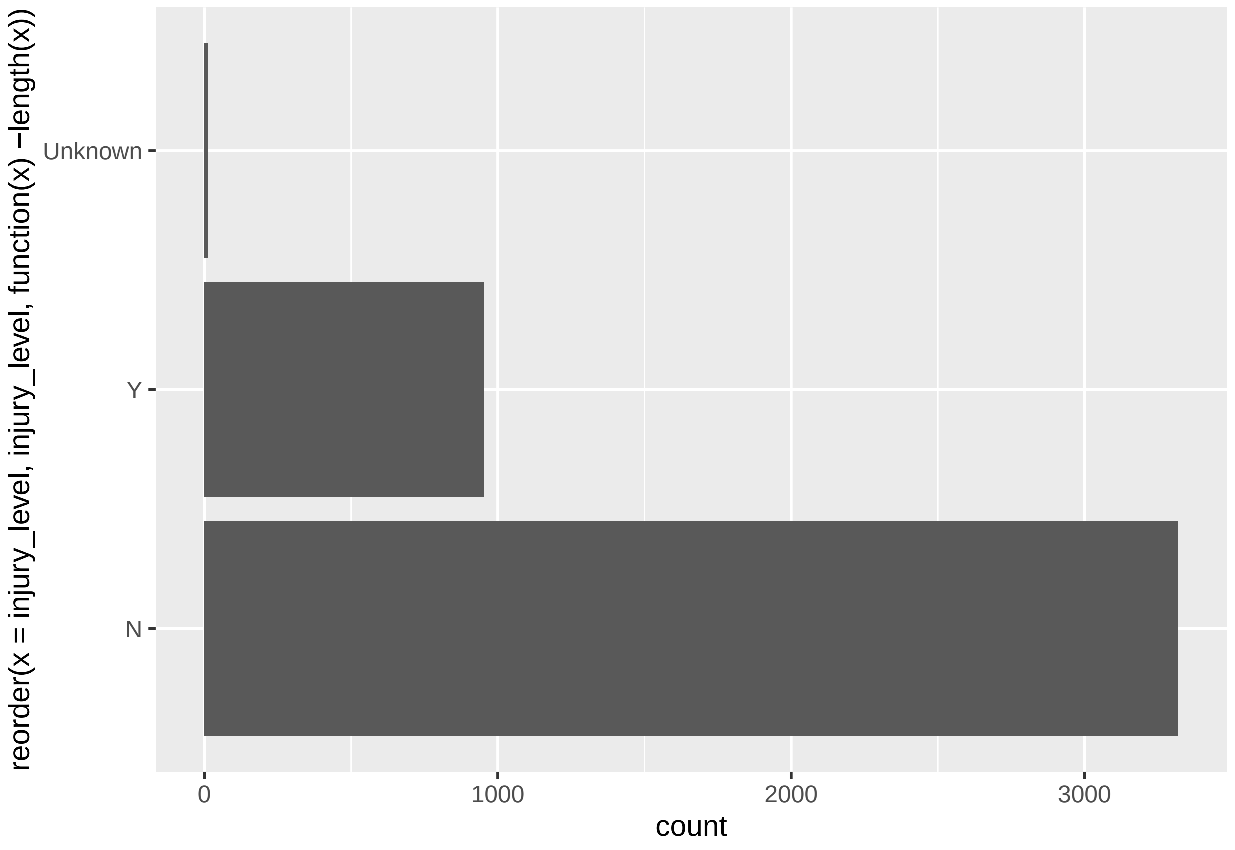
EDA

Since it is all categorical data that we are using, so we will not make a correlation plot. But we will make some separate plot for some variables to get a sense of whether they will be suitable predictors to include in our final recipe.

Outcome

There's six levels in the raw data and we see a significant imbalance of levels: many No jury's! Therefore, we will collapse the rest of the levels into injury.

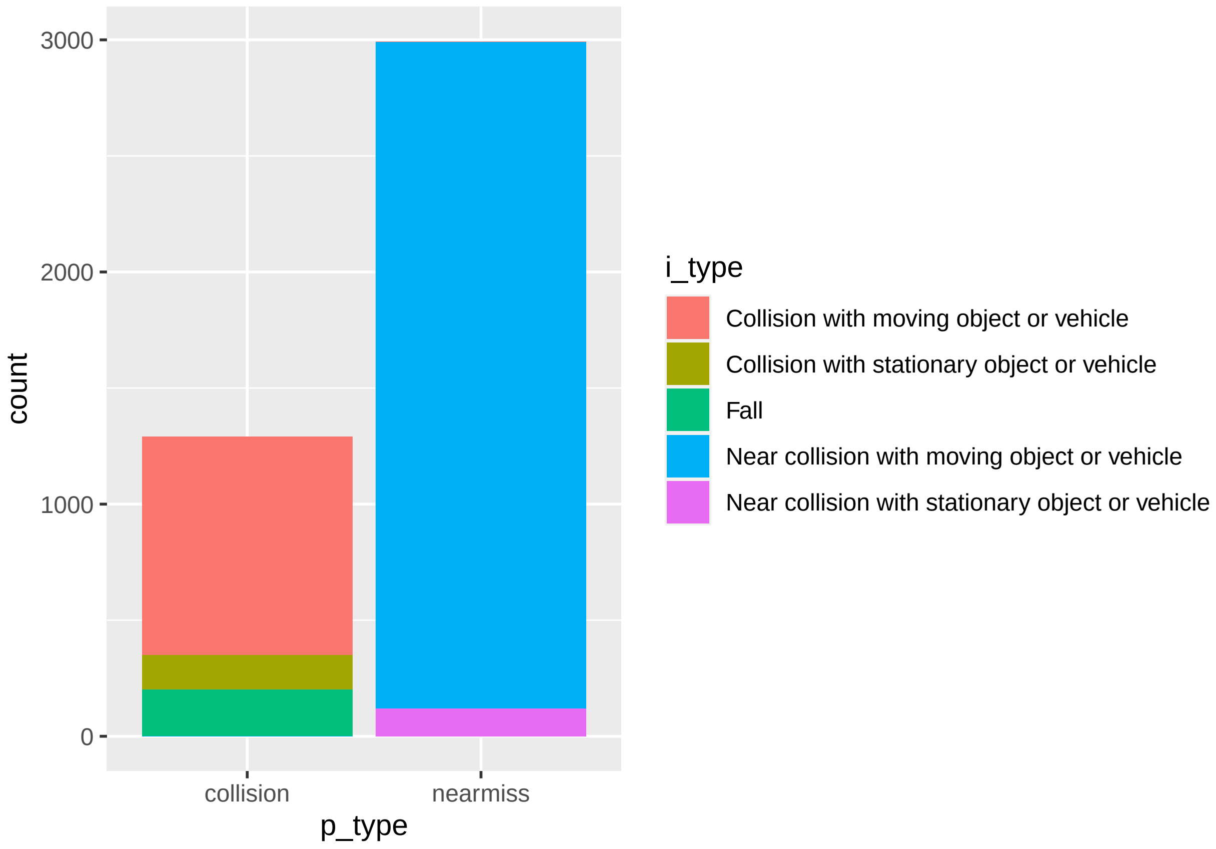


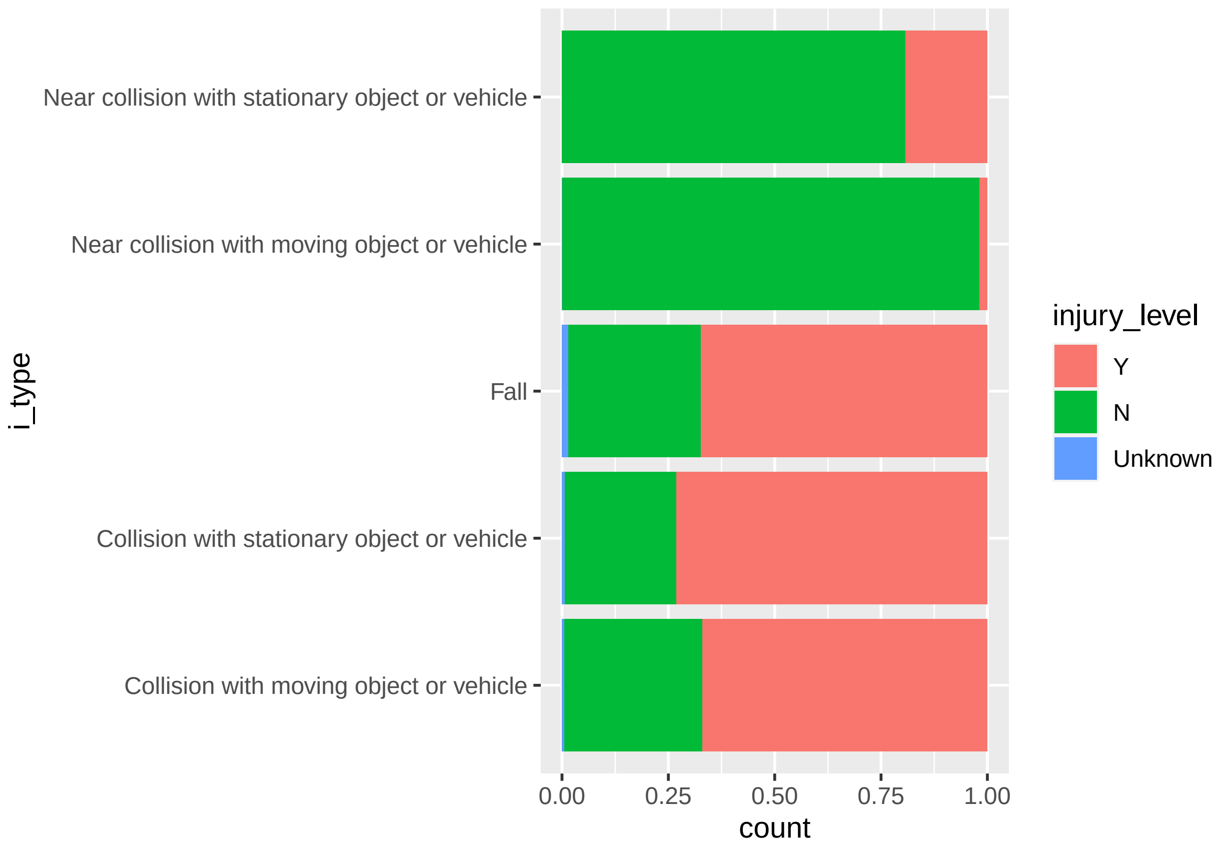


That looks much better! We will deal with the unknowns later.

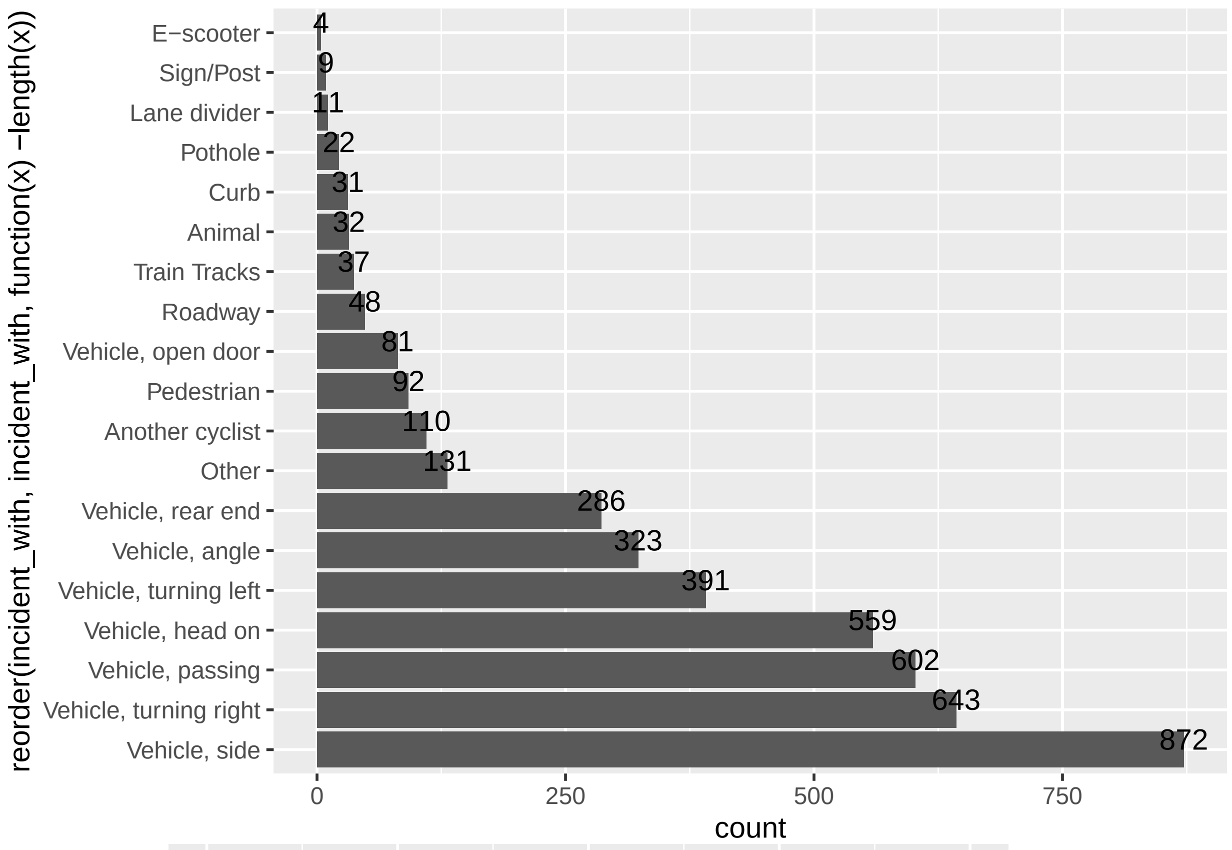
Predictors

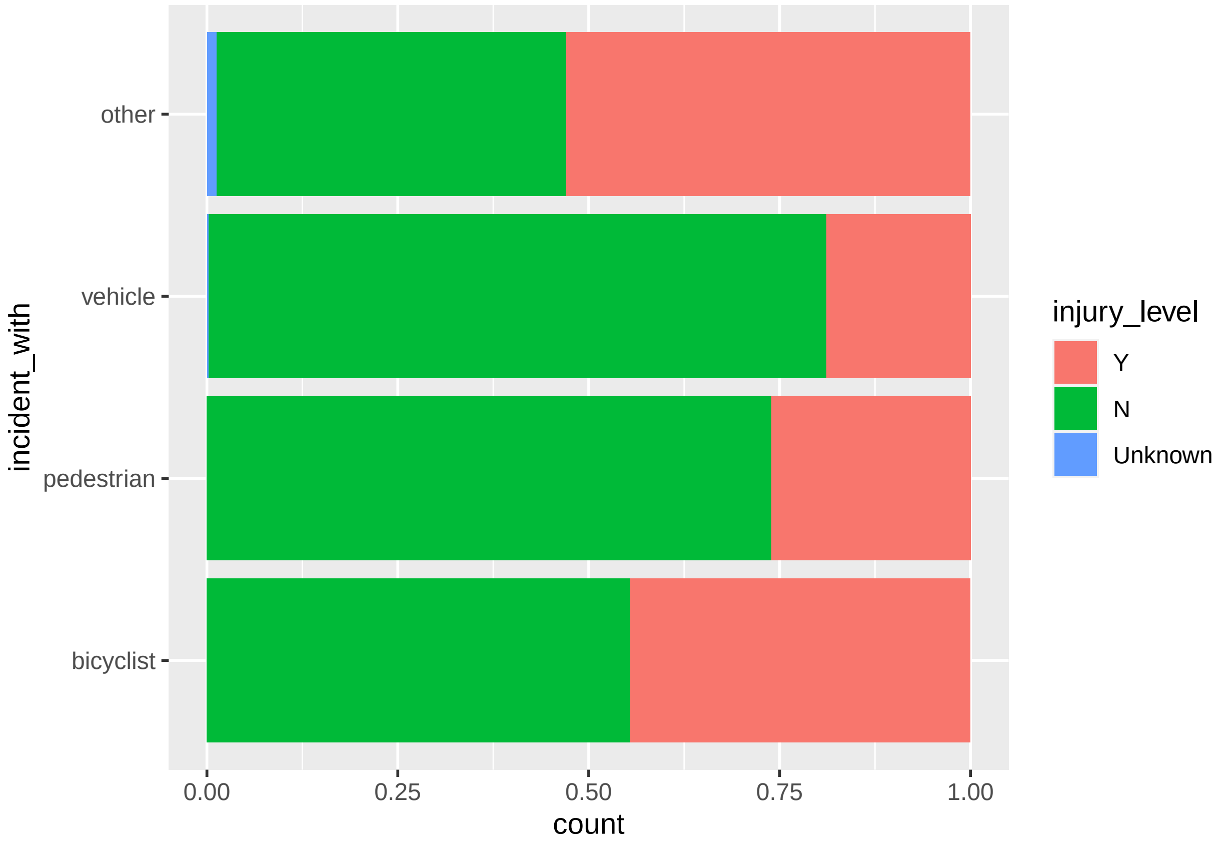
Report type (p_type) and incident type (i_type) From the following graph, we can see that the i_type is perfectly separable by p_type, so we will drop p_type. And by looking at the percentage stack bar chart, we are able to tell that people that were involved in a near-miss would have a much lower chance of getting injury than an actual collision. Therefore, i_type will be a very strong predictor of injury level.



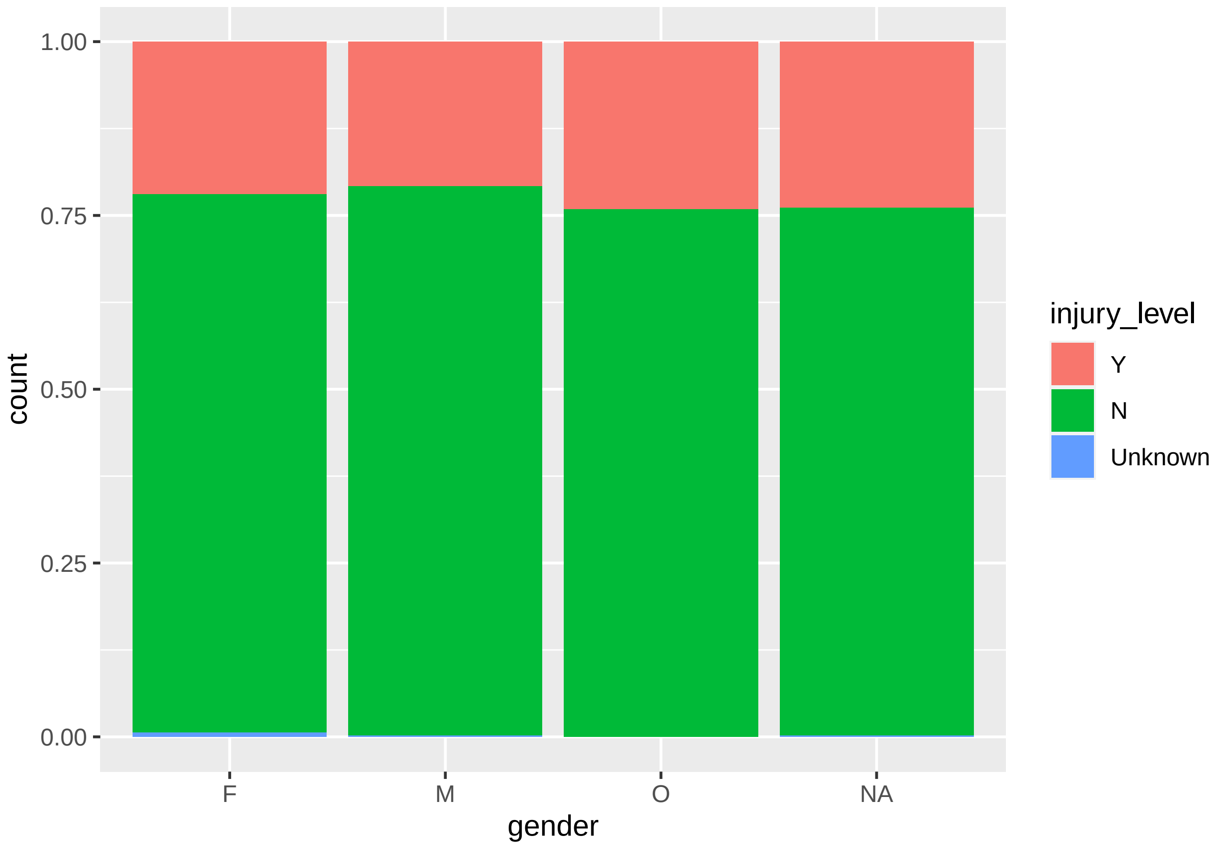


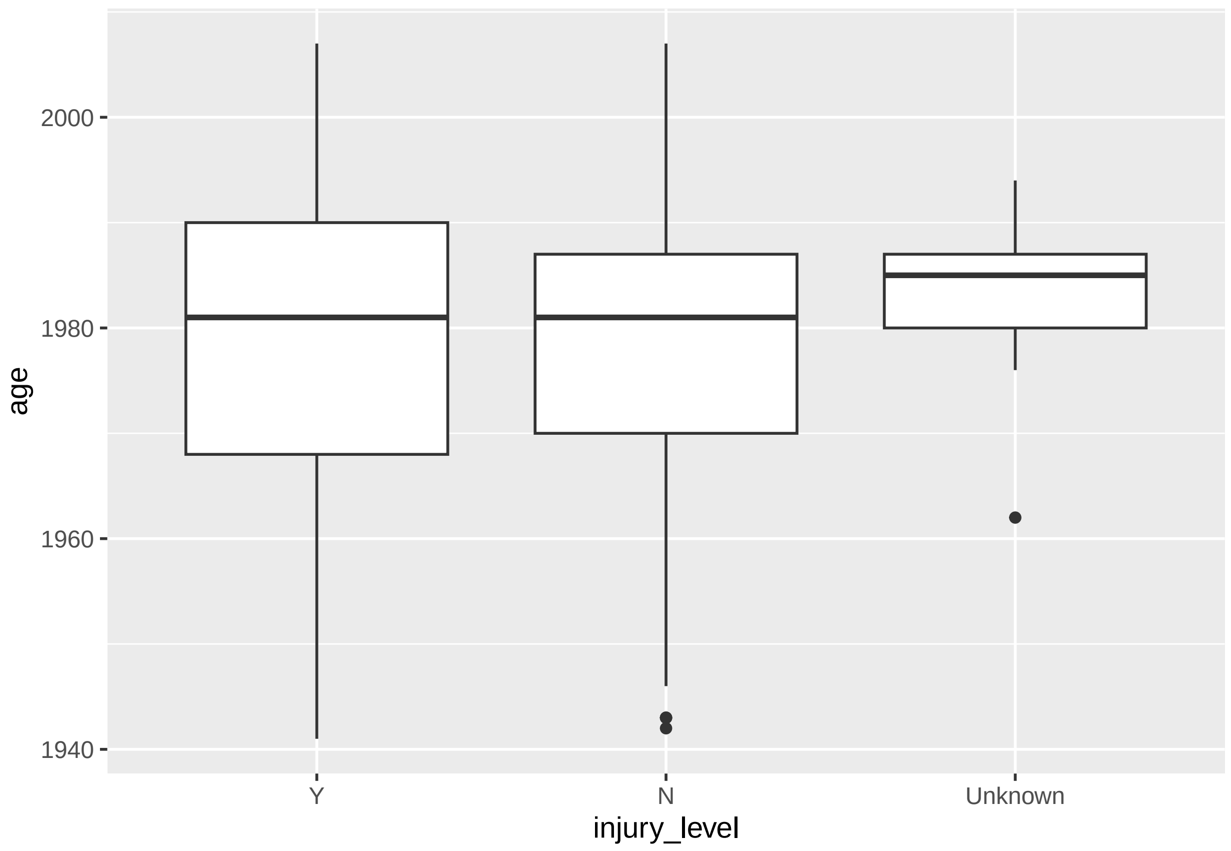
Incident with The object that cyclists had incident with can be important for predicting the injury level. There are too many levels in this variable. We will collapse all vehicle related levels and lump those less than 80 (5%) occurances into Others. One might guess that a collision with a pedestrian might be less serious than with a vehicle. However, we see from the stack bar chart that it seems vehicles has a lower injury rate that the rest. I guess that's because there are far more incidents with vehicle than with the others, which causes the pattern we saw here. Nonetheless, this is an important piece of information to help our model to correctly predict injury.


reorder(incident_with, incident_with, function(x) −length(x))
E−scooter
Sign/Post
Lane divider
Pothole
Curb
Animal
Train Tracks
Roadway
Vehicle, open door
Pedestrian
Another cyclist
Other
Vehicle, rear end
Vehicle, angle
Vehicle, turning left
Vehicle, head on
Vehicle, passing
Vehicle, turning right
Vehicle, side
4
9
11
22
31
32
37
48
81
92
110
131
286
323
391
559
602
643
872
0
250
500
750
count


incident_with
other
vehicle
pedestrian
bicyclist
0.00
0.25
0.50
0.75
1.00
count
injury_level
Y
N
Unknown

Demography Literature seems to agree that older female are more likely to get injured in incidents than other people. Therefore, we include the two variables: gender and age to predict injury. However, from our graph here, we don't see a clear trend. We





Handle Unknowns

You might have seen that, in this dataset we have "I don't know", "Don't remember", "Don't Remember","I don't remember","Unknown" as part of the reponses. Therefore, we will turn them all into NA's. Once we do that, we will drop the empty levels and order the levels by frequency. Lastly, since it is not appropriate to impute for those entries a NA outcome, we will simply drop those doesn't specify injury_level.

```
#convert data into NA
values_to_convert <- c("I don't know", "Don't remember",  "Don't Remember","I don't remember","Unknown"
# Use dplyr to convert values to NA in the entire dataset

data3 <- data2 %>%
  mutate_if(is.factor, ~replace(., . %in% values_to_convert, NA)) %>%
     mutate(across(where(is.factor), ~fct_drop(.))) %>%
      #drop the changed factors
      mutate_if(is.factor,~fct_infreq(.))
      #reorder factor level by frequency (start with most frequent)

#drop NAs in the outcome variable
data3 <- data3 %>%
  drop_na(injury_level)
summary(data3$injury_level)
```

```
##    N    Y
## 3319  954
```

##Setting up Models ### Split data We will split our data into train/testing set. To account for the class imbalance in our outcome variable, we will use stratifed sampling. The same technique is applied when we creat the data folds for cross validation.

```
##split data
set.seed(3435)
data_split <- initial_split(data3, strata = "injury_level", prop = 0.75)

data_train <- training(data_split)
data_test <- testing(data_split)

data_fold <- vfold_cv(data_train, v = 5,
                      strata = "injury_level")
```

Dimension for training and testing set

```
## [1] 3204   16
```

```
## [1] 1069   16
```

Build recipe

With all the ingredients ready, now we are ready to build a recipe! After data exploration, we end up with 15 predictors and 1 binary outcome variable.

Remember we still have a lot of NA's in our dataset: we will use bootstrap aggregation (bag) tree technique to impute for NA's in the 15 predictors. Here's how it works: "for each variable requiring imputation, a bagged tree is created where the outcome is the variable of interest and the predictors are any other variables listed in the impute_with formula. One advantage to the bagged tree is that is can accept predictors that have missing values themselves." In this recipe, I didn't sepcify "impute with" so it will use all the predictor variables in my dataset to impute the NA's.

The last step of my recipe is to upsample my outcome variable with a ratio 0.5. It means it will upsample the "Yes" category to half of the "No". Package `themis` helps to do the work.

```
#install.packages("themis")
library(themis)

data_recipe <- recipe(injury_level ~ . , data = data_train) %>%
  step_impute_bag(i_type , incident_with,
                  trip_purpose, regular_cyclist , helmet ,
                  road_conditions , sightlines , cars_on_roadside ,
                  bike_lights , terrain , bicycle_type , ebike ,
                  turning , age , gender,impute_with = imp_vars(all_predictors())) %>%
    step_dummy(all_nominal_predictors()) %>%
    step_upsample(injury_level, over_ratio = 0.5, skip = TRUE)
```

Building Prediction Models

In this study, I selected four models to predict injury from bike incidents: K Nearest Neighbors, Logistic regression, Elastic Net and Random Forest. We will set up workflow, tune hyperparameters, finalize our workflow by incorporating the best performing hyperparameter selected based on the metric area under the ROC curve.

Workflow

```r
#install.packages("ranger")
library(ranger)
library(yardstick)
```

Set up workflow for K Nearest Neighbors

```r
#knn
mod_knn <- nearest_neighbor(neighbors = tune()) %>%
  set_mode("classification") %>%
  set_engine("kknn")

wf_knn <- workflow() %>%
  add_model(mod_knn) %>%
  add_recipe(data_recipe)
```

Set up workflow for Logistic Regression

```r
#logistic
mod_lr <- logistic_reg() %>%
  set_mode("classification") %>%
  set_engine("glm")

wf_lr <- workflow() %>%
  add_model(mod_lr) %>%
  add_recipe(data_recipe)
```

Set up workflow for Elastic Net

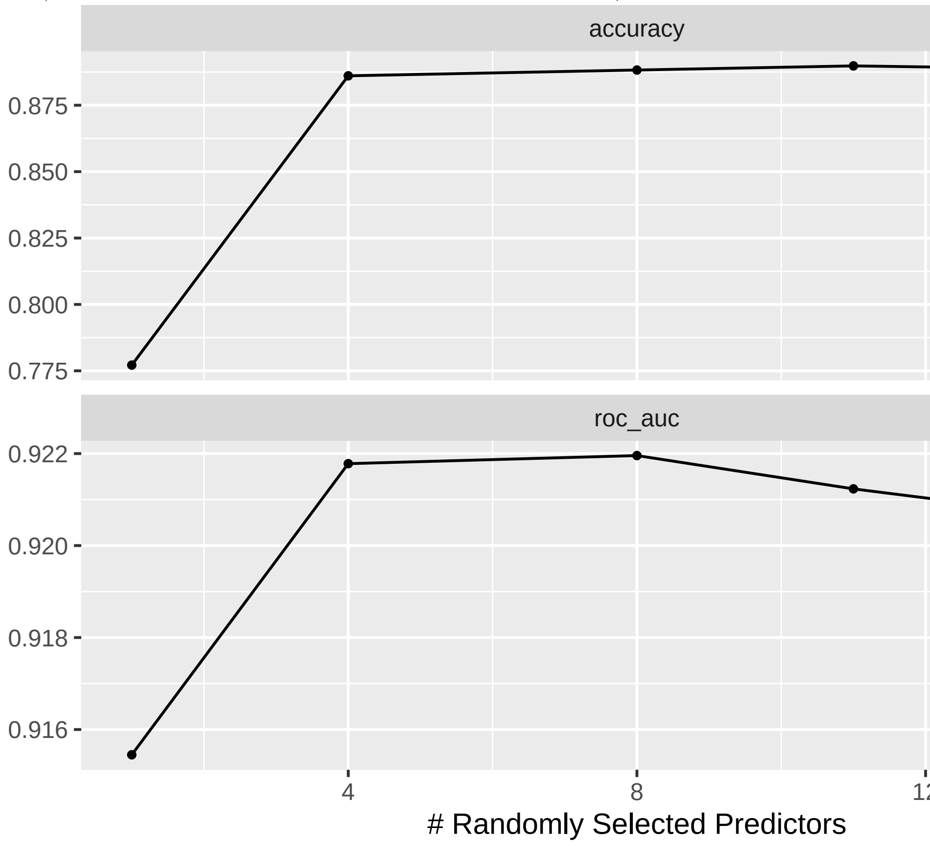
```r
#logistic elastic net
mod_en <- logistic_reg(penalty = tune(),
                           mixture = tune()) %>%
  set_mode("classification") %>%
  set_engine("glmnet")

wf_en <- workflow() %>%
  add_model(mod_en) %>%
  add_recipe(data_recipe)
```

For the random forest model, since we have 15 variables in total, so the meaningful m_try will be from 1 to 15. Since our dataset is relatively big so to save time, we will fix number of trees and minimal nodes to focus on tuning m_try. From the first few trials I figured that number of trees and minimal nodes did not impact the outcome too much. So I chose 600 and 25 to reduce run time. I add `importance = "impurity"` so that at the end we can make an importance plot to see what are the important variables in the prediction process.

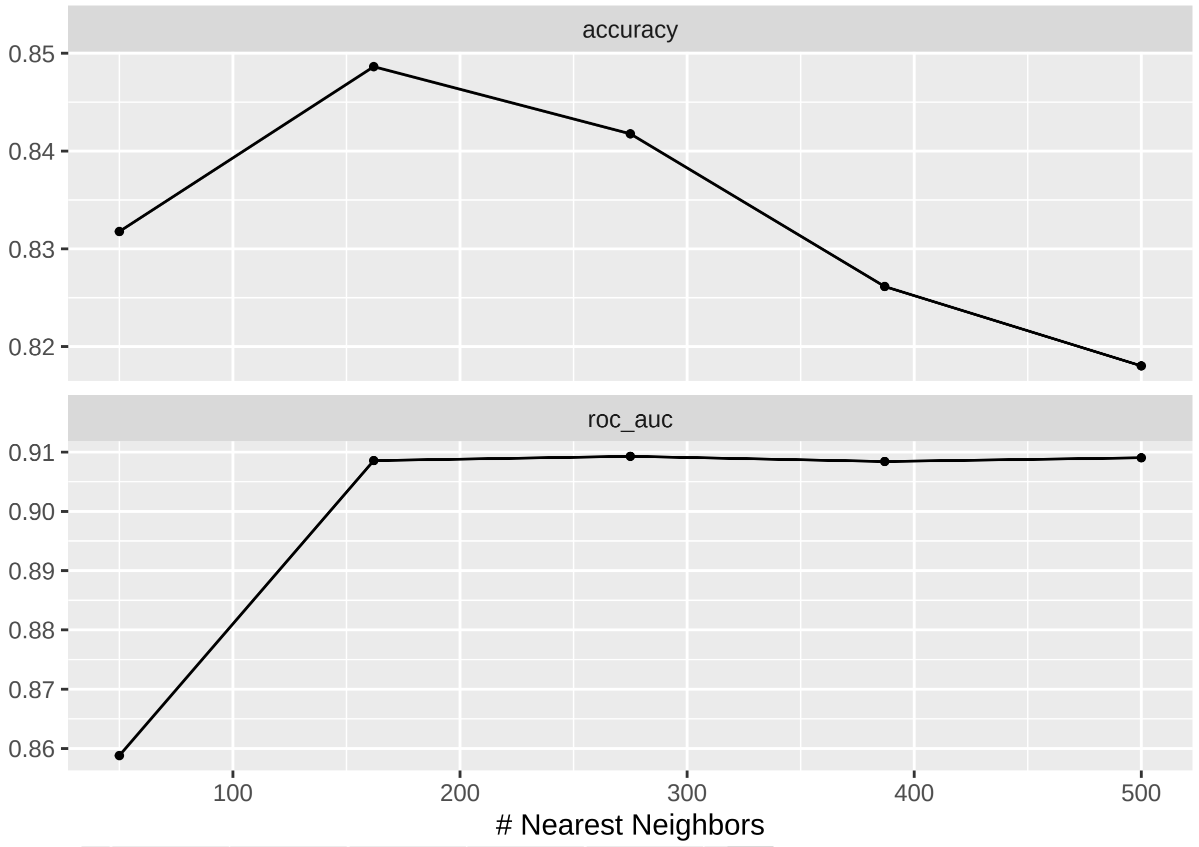
```r
#random forest
rf_mod <- rand_forest(
  mtry = tune(),
  trees = 600,
  min_n = 25
) %>%
  set_mode("classification") %>%
  set_engine("ranger",importance = "impurity")

rf_wkflow <- workflow() %>%
  add_model(rf_mod) %>%
  add_recipe(data_recipe)
```

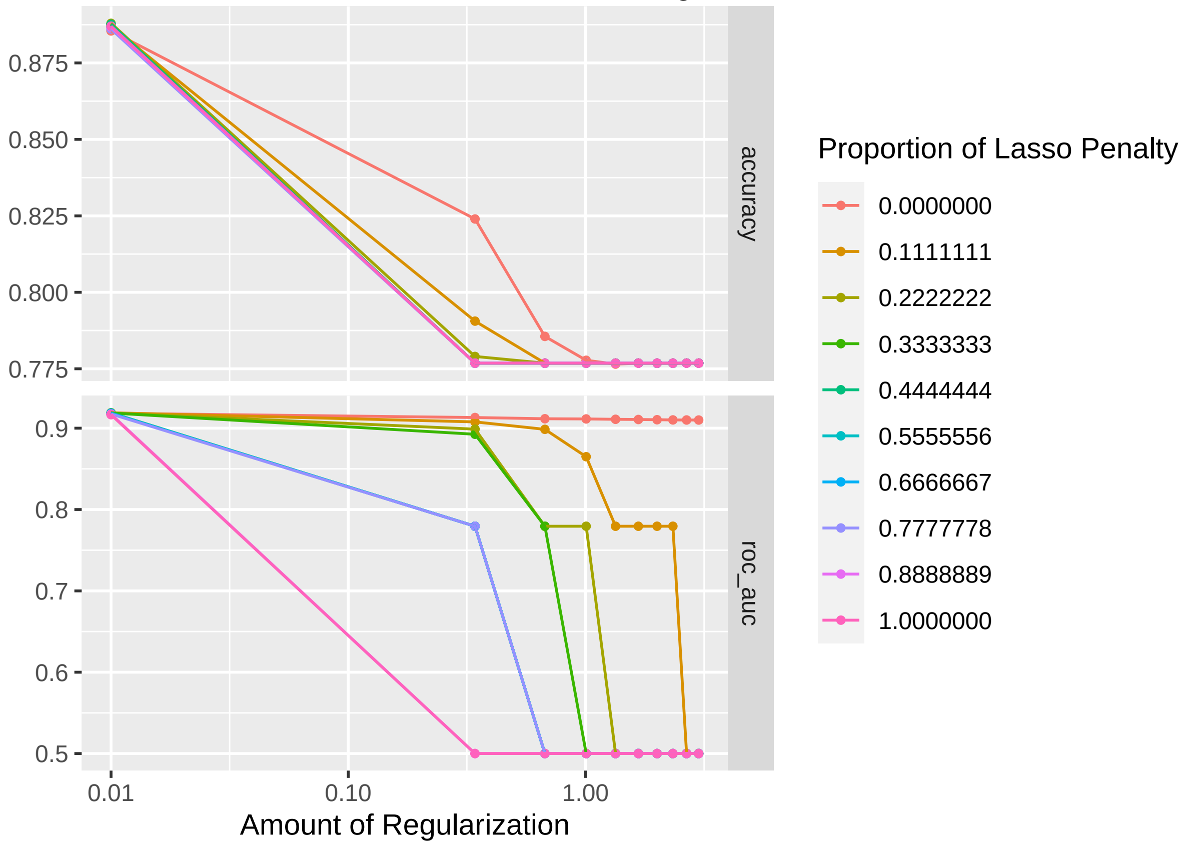
Tune model

Let's see the tuning results of the three models. We will select the best performing hyperparameters to finalize our workflow. In both random forest and knn, we see a bell curve. It is less clear for elatistic net, so we will



leave it for `select_best()` to help us decide.


accuracy
0.85
0.84
0.83
0.82
roc_auc
0.91
0.90
0.89
0.88
0.87
0.86
100
200
300
400
500
# Nearest Neighbors


0.875
0.850
0.825
0.800
0.775
accuracy
0.9
0.8
0.7
0.6
0.5
roc_auc
0.01
0.10
1.00
Amount of Regularization
Proportion of Lasso Penalty
0.0000000
0.1111111
0.2222222
0.3333333
0.4444444
0.5555556
0.6666667
0.7777778
0.8888889
1.0000000

```
## # A tibble: 1 x 2
##    mtry .config
##   <int> <chr>
## 1     8 Preprocessor1_Model3
```

```
## # A tibble: 1 x 2
##   neighbors .config
##       <int> <chr>
## 1       162 Preprocessor1_Model2
```

```
## # A tibble: 1 x 3
##   penalty mixture .config
##     <dbl>   <dbl> <chr>
## 1    0.01   0.111 Preprocessor1_Model011
```

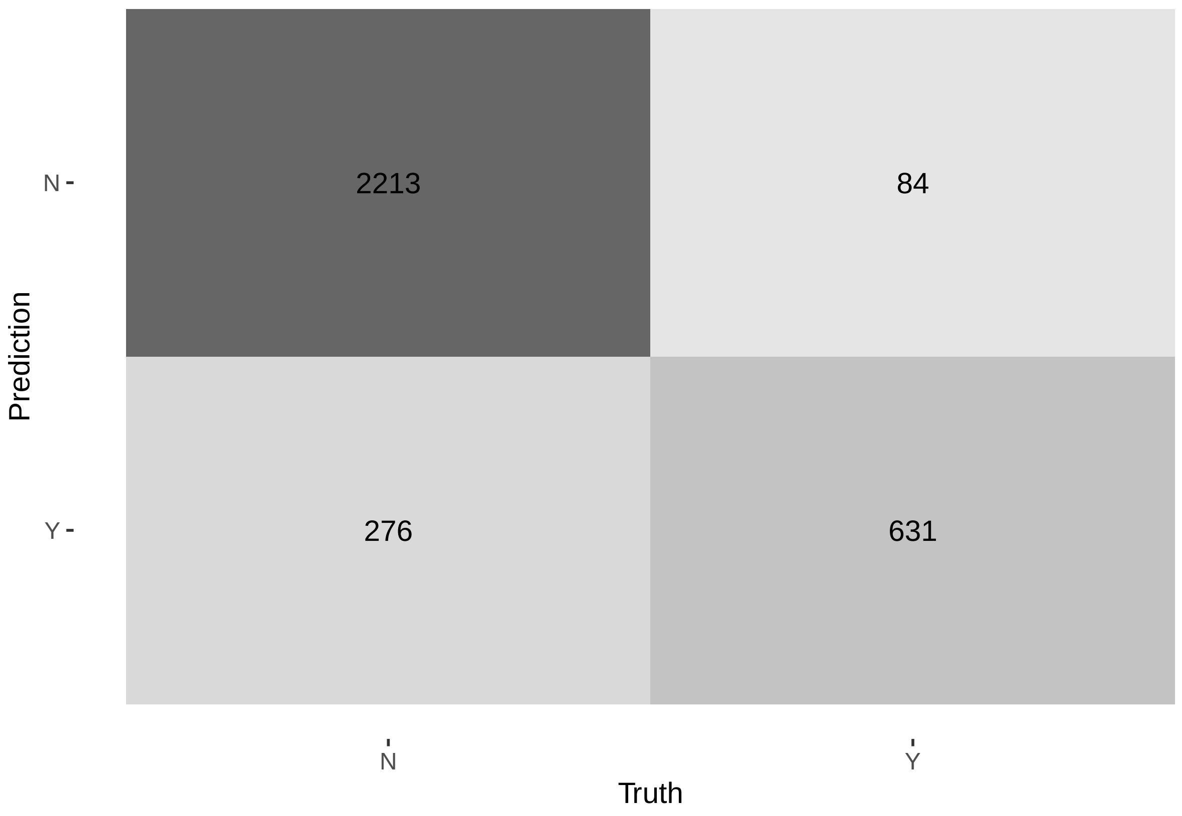
Training data fit

Now we have the finalized workflow, we can use cross validation to fit the model and get a averaged value of training auc value.

```r
#fit to train set
rf_fit <- fit_resamples(
  rf_final,
  resamples = data_fold,
  control = control_resamples(save_pred = TRUE))
save(rf_fit,file="20231214rffit.rda")
```

```r
#knn
knn_fit <-fit_resamples(
  knn_final,
  resamples = data_fold,
  control = control_resamples(save_pred = TRUE))
save(knn_fit,file="20231214knnfit.rda")

#en
en_fit <-fit_resamples(
  en_final,
  resamples = data_fold,
  control = control_resamples(save_pred = TRUE))
save(en_fit,file="20231214enfit.rda")

#lr
lr_fit <- fit_resamples(
  wf_lr,
  resamples = data_fold,
  control = control_resamples(save_pred = TRUE))
save(lr_fit,file="20231214lrfit.rda")
```

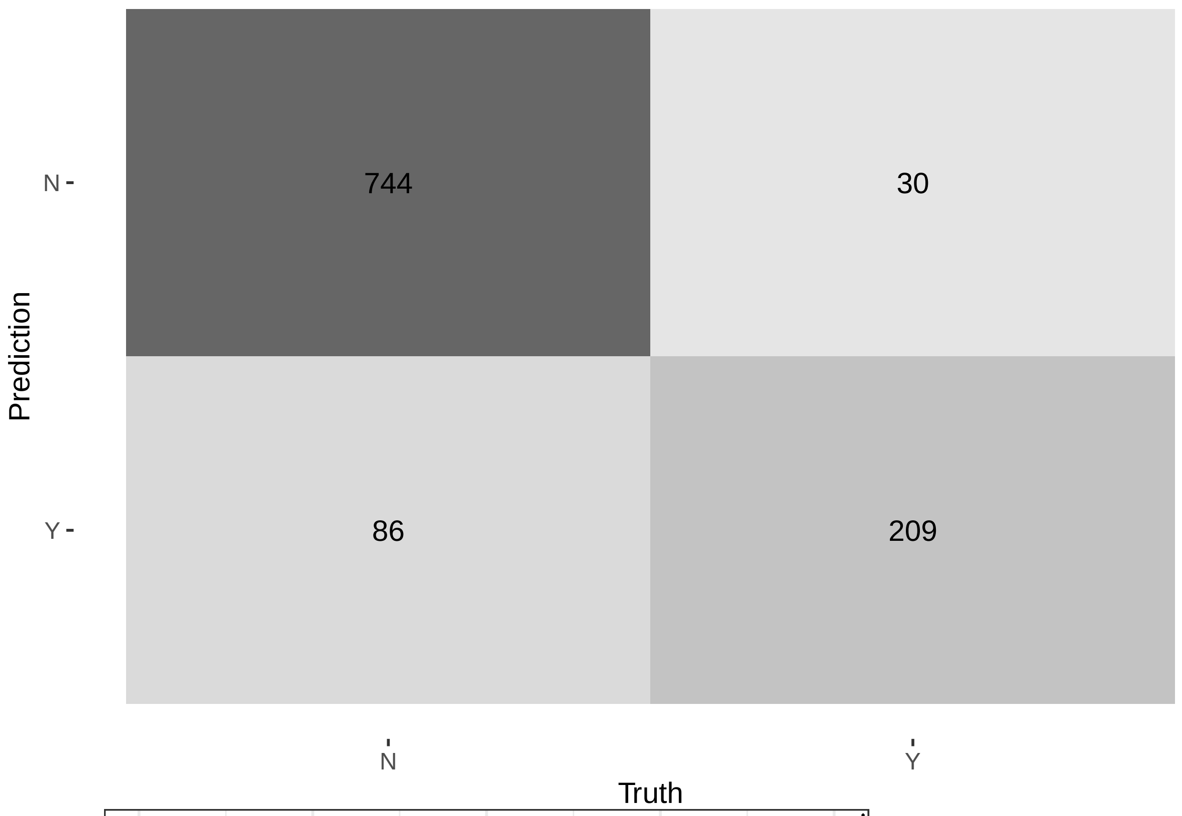
```
##                models .metric .estimator      mean n     std_err
## 1       random forest roc_auc     binary 0.9214909 5 0.006304038
## 2                 knn roc_auc     binary 0.9085612 5 0.006036424
## 3         elastic net roc_auc     binary 0.9186756 5 0.003511297
## 4 logistic regression roc_auc     binary 0.9170674 5 0.004625017
##               .config
## 1 Preprocessor1_Model1
## 2 Preprocessor1_Model1
## 3 Preprocessor1_Model1
## 4 Preprocessor1_Model1
```

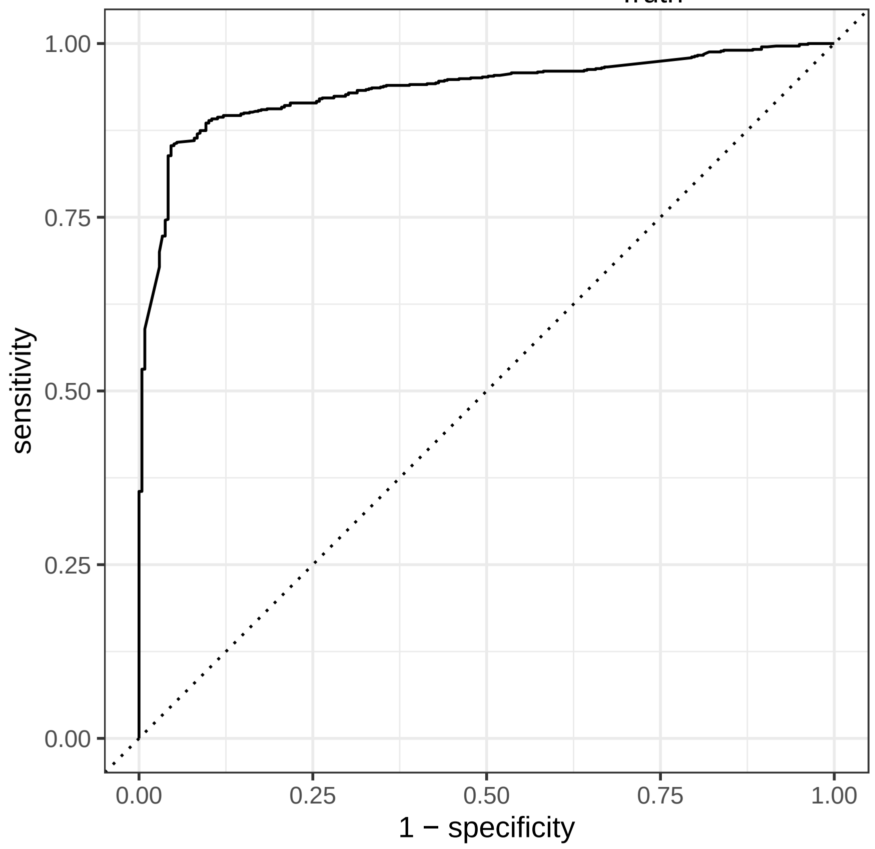


All the models seem to perform very well! They all have an auc value higher than 90%. Elastic Net and logistic regression end up with very similar results. I assume that's because the penalty applied is very limited (0.1) therefore it doesn't restrict the coefficient too much.

Best Model Test Results

Finally, we will see how they perform on the testing data! This will be a fairer metric than training auc because the model never sees this part of the dataset at all. I selected random forest, which seems to perform best according to the performance on the training dataset.



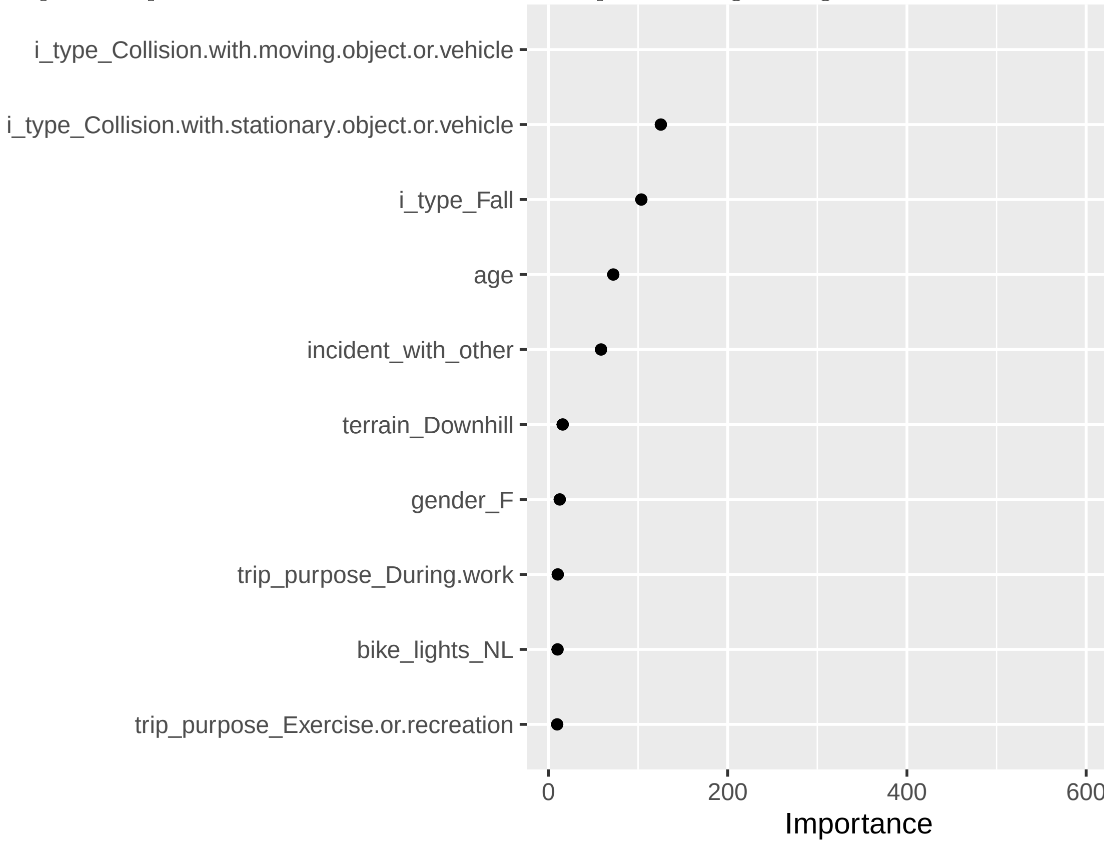


```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.934
```

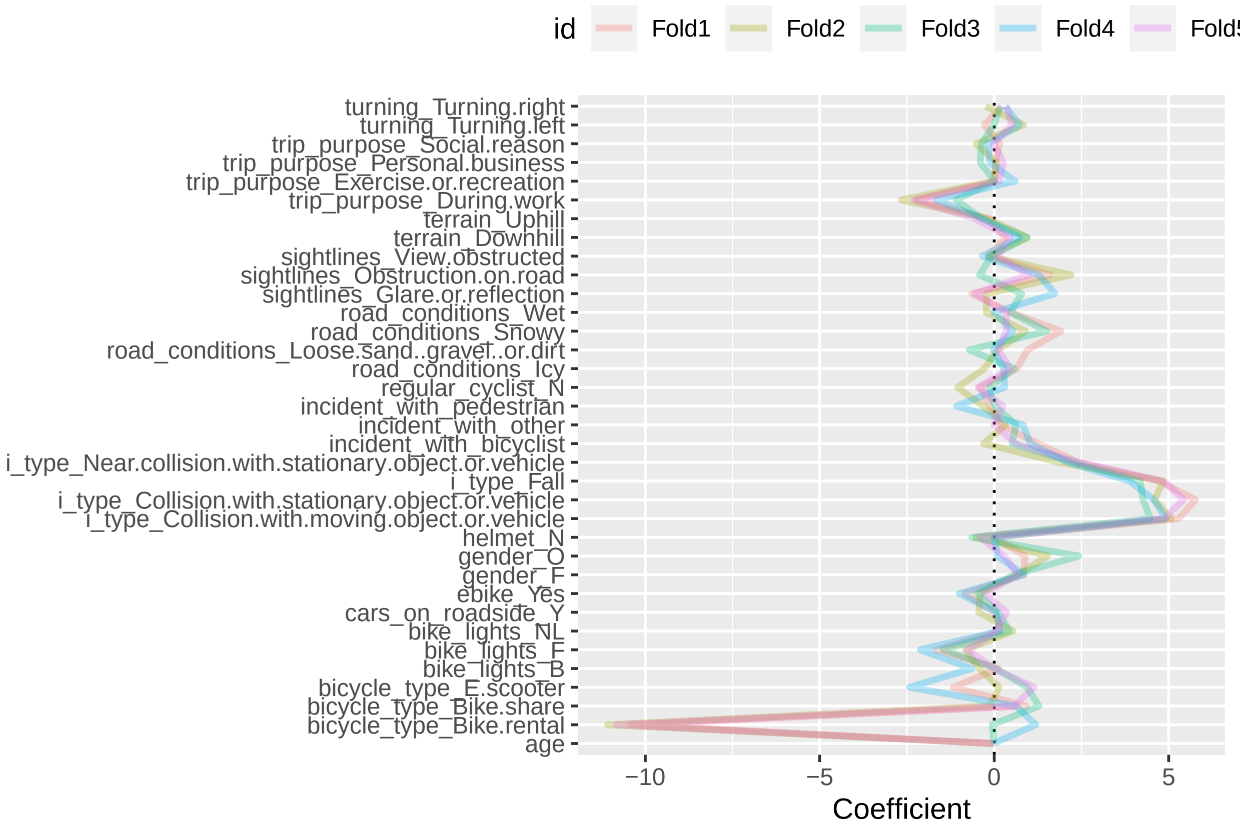
According to the heatmap and ROC, we can see the model perform pretty good on the testing data. Woohoo!

Inference

Finally, like I said in the introduction, I am also interested in seeing what are the important factors in predicting injury. We will look at the importance plot from random forest and coefficient plot from logistic regression



to infer causual relationships.



Both graphs show that i_type is the most important/significant predictor of injury. It turns out that age does play a small role in the prediction as well. The two models seem to pick up similar pattern in the dataset as both shows trip purpose (during work), terrain (downhill) and gender (female) are important predictors.

Conclusion

In this study, I explore the crowdsourced bike incident dataset and set up four machine learning models to predict the binary outcome of injured or not in an incident. It turns out that all four models were able to reach a training AUC above 0.9. Among them, random forest model performed the best (AUC = 0.93).

This study has a couple of limitations. First, the dataset is heavily biased towards no injury outcome (70%). That's why I used upsampling to account for the imbalance. Future studies can consider algorithms or models to handle imbalanced classes, such as Random Forest With Class Weighting and naive classifier. As we can see in the importance plot, incident type is the strongest predictor. That's because most of the reports are near-miss (72%), which can simply imply no injury most of the time. Therefore, it might be interesting to see how the model will change if we take out this highly predictive vairable if we want to figure out what are the other variables that are contributing to the injury level. Lastly, since I imputed for 21% of the data in this study, I could have brought in significant assumption into the study.

In conclusion, I found that random forest is the best performing model for predicting injury in bike incidents (AUC = 0.93) and incident type is most useful predictor.