Final Project

Code

Edward Ho

2022-06-01

Introduction

My project will outline predictor variables of interest that could potentially predict the severity of a car accident.

What's in this dataset? What is it?

To begin, the dataset is a countrywide record of car accidents from 49 different states. The data set covers a wide time frame where it tracks monthly and daily data spanning from 2016 to 2021. It's important to note that the data was collected from various sources—such as the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors, etc. There's a wide variety of data, with over 1.5 accidents, or observations, that have been recorded with the highest frequencies of recorded accidents being from the state of California. This model contain various variables, but most notably, one variable contained is the severity attribute which spans from 1-4, where 1 is the lowest severity and 4 is the highest. This severity is determined by it's impact on traffic, or the shortest delay on oncoming traffic as a result of the accident.

Why even create the model?

I'll list further details on the variables below, but essentially, the original data set contained nearly 47 different variables. All these variables levels in importance based on what one might be interested in predicting, but as I noted previously, I want to see what factors may be especially important in predicting severity of a certain accident. Many times we are forced to drive in various different weather conditions and usually we bat a blind eye towards the dangers that may result from these conditions. Weather conditions such as wind speed, weather conditions (rain, snow, etc.), night/day, temperature, and maybe even the

presence of certain road factors (roundabouts, cross ways, etc.). It may be important to know the relationship of these accidents and how they might vary in severity to potentially raise awareness to future drivers.

Loading Data and Packages

The project was found through Kaggle, and as I noted before, there were nearly 1.5 million observations in the data. Therefore, I wasn't able to import the raw data as a whole into R, therefore I must note that I did some light preliminary cleaning of the data set by filtering by only what I will be using as the subject of my analysis, 2021 California data. Therefore, after filtering out blank inputs, and grouping by 2021 and California exclusive cases, we now have a new data set that I can do further cleaning on. Let's clean the data a bit before displaying the full table and splitting the data for further analysis.

Data Cleaning

Below, we will be removing some unneeded variables, renaming some columns, and other techniques for needed clarity in reading the data. Note, there is no need to remove missing/blank observations since I have already made sure they were not included through excel and before importing the csv.

- Check for any remaining na's

Hide

```
accidents <- na.omit(accidents_data)
sum(is.na(accidents))
```

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

- Remove the start_time and state column since I had already filtered that completely in excel. variables

Hide

```
accidents <- accidents %>%
  select(-Start_Time,-State,-Wind_Direction)
```

- Cleaning the variable names for clarity

Hide

```
accidents <- janitor::clean_names(accidents)
```

- Now I must factor encode all the variables that are boolean values.

Hide

```
accidents$severity <- factor(accidents$severity)
accidents$side <- factor(accidents$side)
accidents$weather_condition <- factor(accidents$weather_condition)
accidents$bump <- factor(accidents$bump)
accidents$crossing <- factor(accidents$crossing)
accidents$railway <- factor(accidents$railway)
accidents$stop <- factor(accidents$stop)
accidents$traffic_signal <- factor(accidents$traffic_signal)
accidents$sunrise_sunset <- factor(accidents$sunrise_sunset)
```

Now the data is cleaned, and we can proceed with the analysis

Variables

Below are noteworthy variables that will be used in our analysis (note that there are 17 variables, but I chose only select few important ones to display here. A full, detailed explanation of each variable can be found in the codebook in the zip file.)

Table 1: Variable Details

Variable Name	Description	Type
severity	Shows the severity of the accident, a number between 1 and 4, where 1 indicates the least impact on traffic (i.e., short delay as a result of the accident) and 4 indicates a significant impact on traffic (i.e., long delay).	Numeric
side	Shows the relative side of the street (Right/Left)	Factor
visbility_mi	Shows visibility (in miles).	Numeric
wind_speed_mph	Shows wind speed (in miles per hour).	Numeric
precipitation_in	Shows precipitation amount in inches, if there is any.	Numeric
weather_condition	Shows the weather condition (rain, snow, thunderstorm, fog, etc.)	Factor
crossing	Presence of crossing in a nearby location.	Factor
stop	Presence of stop in a nearby location.	Factor

Variable Name	Description	Type
traffic_signal	Presence of traffic signal in a nearby location.	Factor
sunrise_sunset	hows the period of day (i.e. day or night) based on sunrise/sunset.	Factor

Data Split

I split my data using a proportion of 80/20, where the training data is 80% of my total data, and the testing data covers the other 20%. Because I had a large imbalance in the response variable, severity, then I must use stratified sampling (which you can see in the EDA portion).

Hide

```
accidents_split <- initial_split(accidents, prop = 0.80,strata = severity)
accidents_train <- training(accidents_split)
accidents_test <- testing(accidents_split)
```

Now I run a sanity check on the dimensions of the testing and training set to see if they match

Hide

```
dim(accidents)
```

```
## [1] 122416     17
```

Hide

```
dim(accidents_train) + dim(accidents_test)
```

```
## [1] 122416     34
```

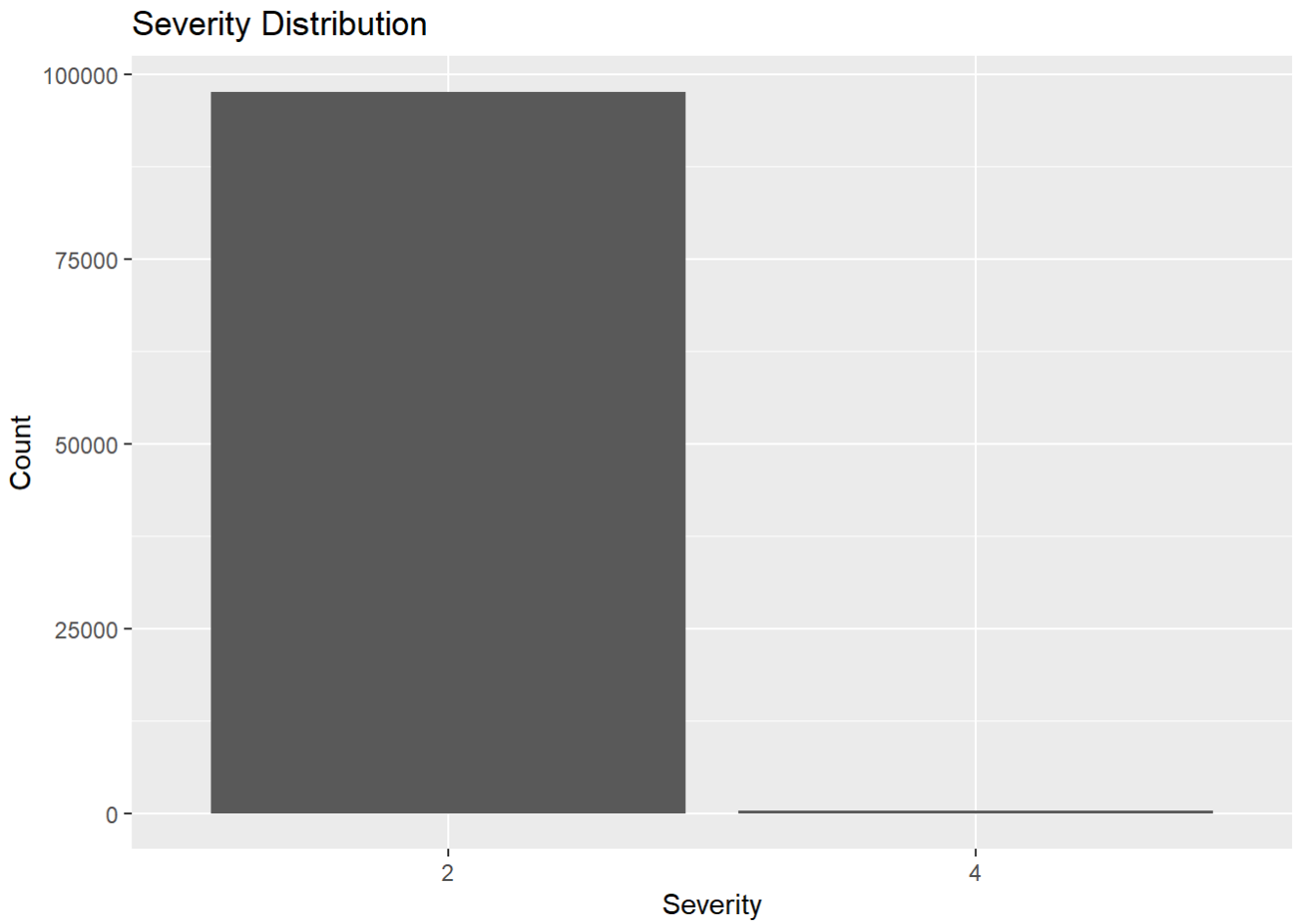
Exploratory Data Analysis

The proceeding section will be exploratory data analysis on the training set to find any potential patterns and notions we might have before we proceed with fitting the models.

Severity Distribution

Hide

```
ggplot(accidents_train, aes(severity)) +
  geom_bar() +
  labs(
    title = "Severity Distribution",
    x = "Severity",
    y = "Count"
  )
```



Here is the proportion of the response variable with a severity of 4.

Hide

```
sum(accidents_train$severity=="4")/nrow(accidents_train)
```

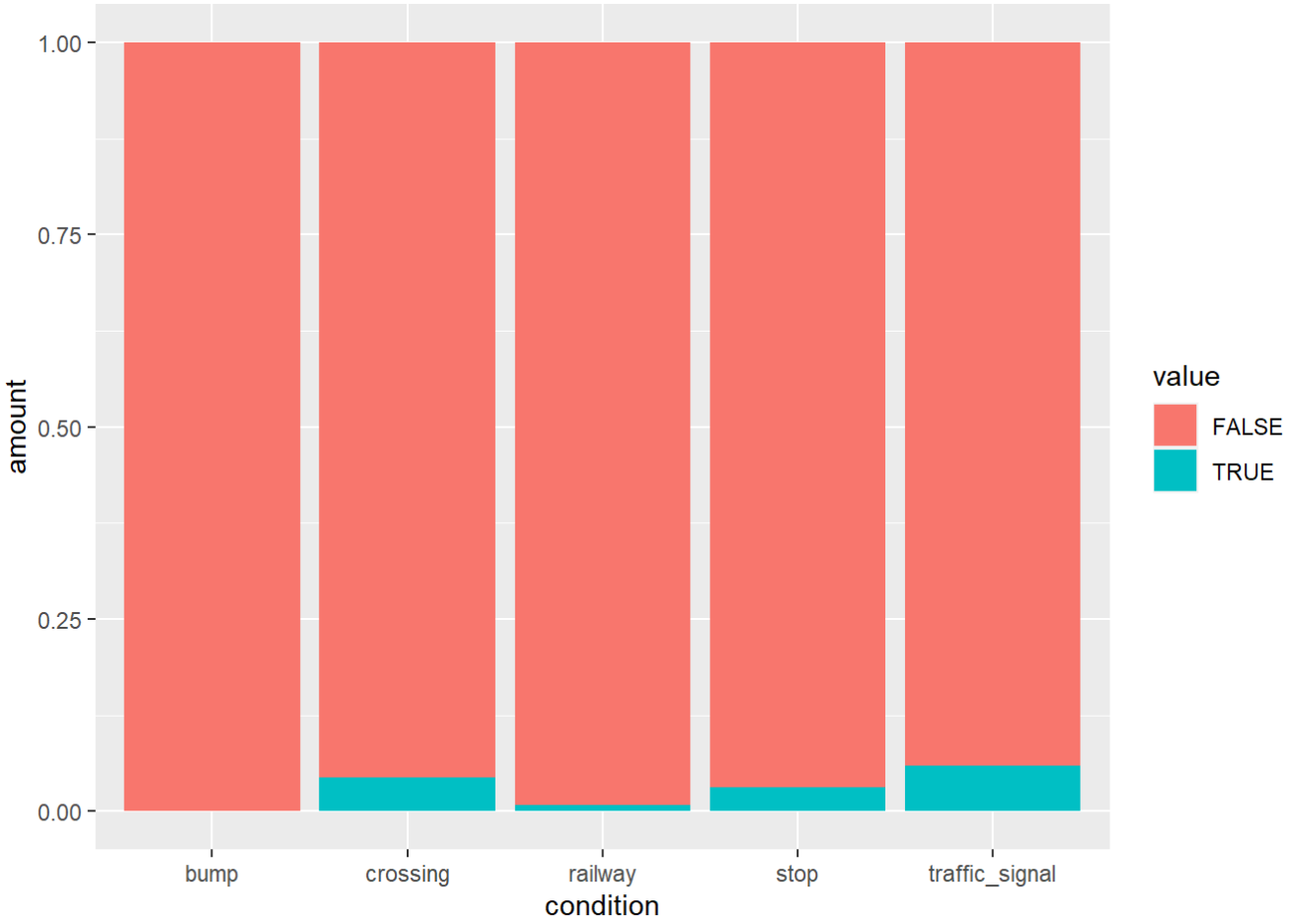
```
## [1] 0.002961238
```

Notice that we find ourselves an extremely prevalent class unbalance in our response observations, therefore, something such as down sampling will be extremely valuable later on when creating the recipe. Also, notice how the response variables only contain 2 and 4 despite the range being from 1-4, therefore it appears all the accidents were either mild or severe in terms of severity.

Exploring the Location landmarks Factor Variables

Hide

```
value <- c("TRUE" , "FALSE")
amount <- c(sum(accidents_train[12]=="TRUE"),sum(accidents_train[12]=="FALSE"),sum(acciden
ts_train[13]=="TRUE"),sum(accidents_train[13]=="FALSE"),sum(accidents_train[14]=="TRUE"),s
um(accidents_train[14]=="FALSE"),sum(accidents_train[15]=="TRUE"),sum(accidents_train[15]=
="FALSE"),sum(accidents_train[16]=="TRUE"),sum(accidents_train[16]=="FALSE"))
condition <-c("bump","bump","crossing","crossing","railway","railway","stop","stop","traff
ic_signal","traffic_signal")
data <- data.frame(value,condition,amount)
ggplot(data, aes(x = condition, y = amount, fill = value)) +
  geom_col(position = "fill")
```

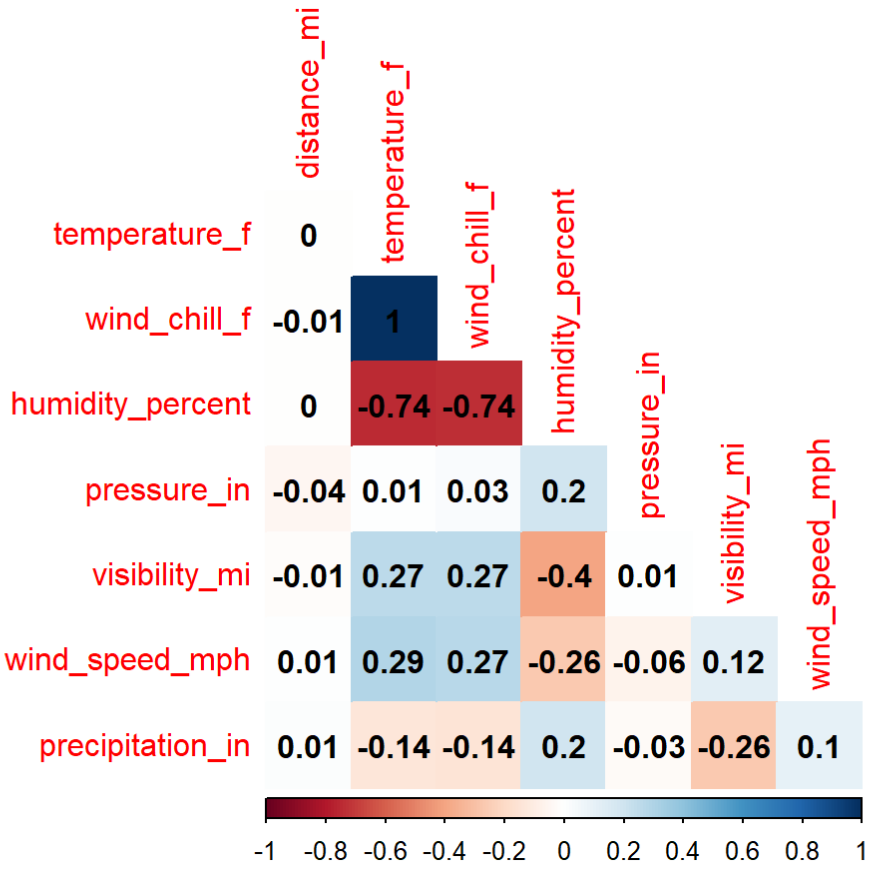


The first thing we can notice already is the that crossing, traffic signals, and stop related traffic incidents are much more frequent than bumps and railway accidents.

Potential Predictor Correlation

Hide

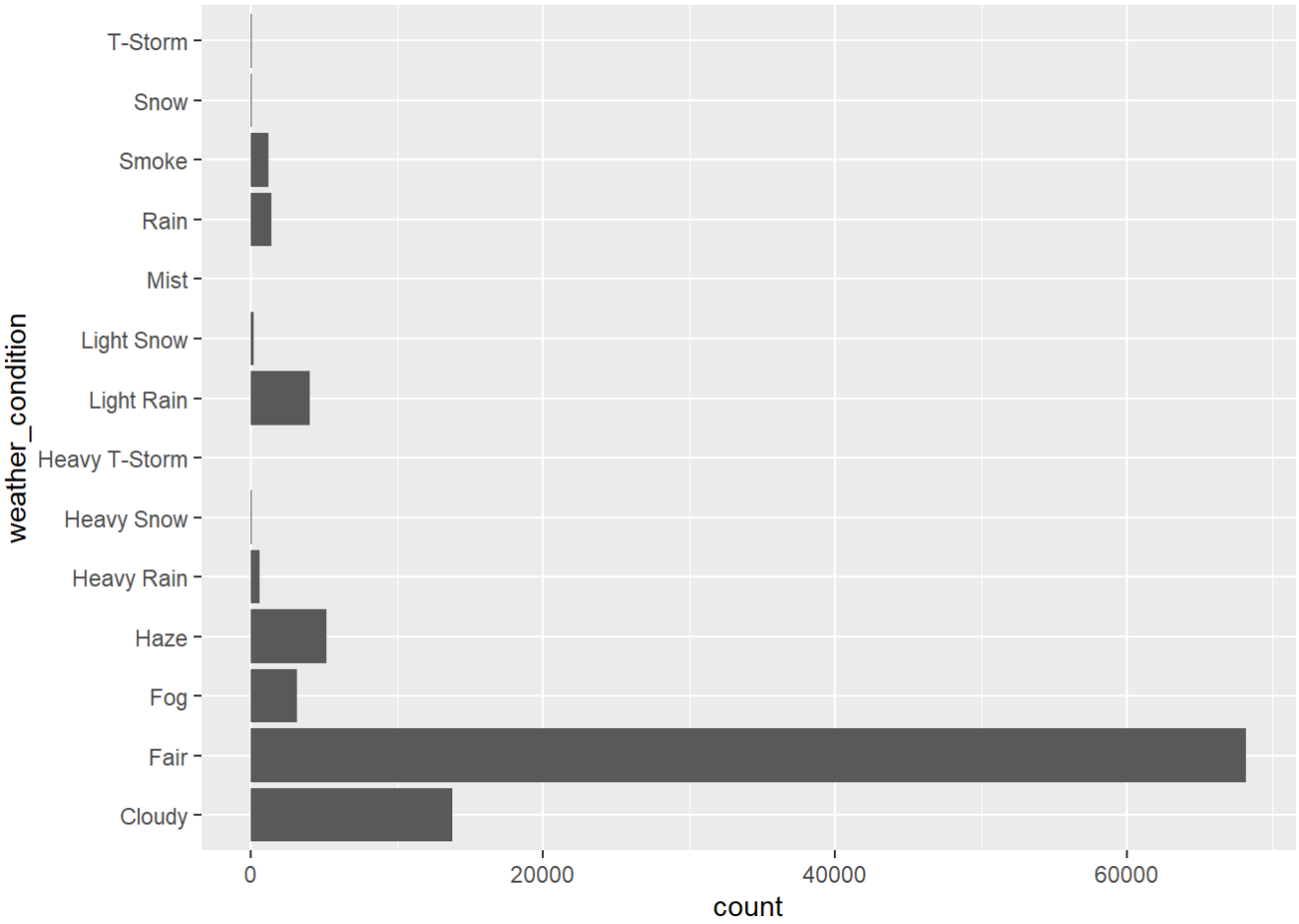
```
accidents_train %>%
  select(where(is.numeric)) %>%
  cor() %>%
  corrplot(method = 'color',type = "lower",diag = FALSE,addCoef.col="Black")
```



Weather conditions

Hide

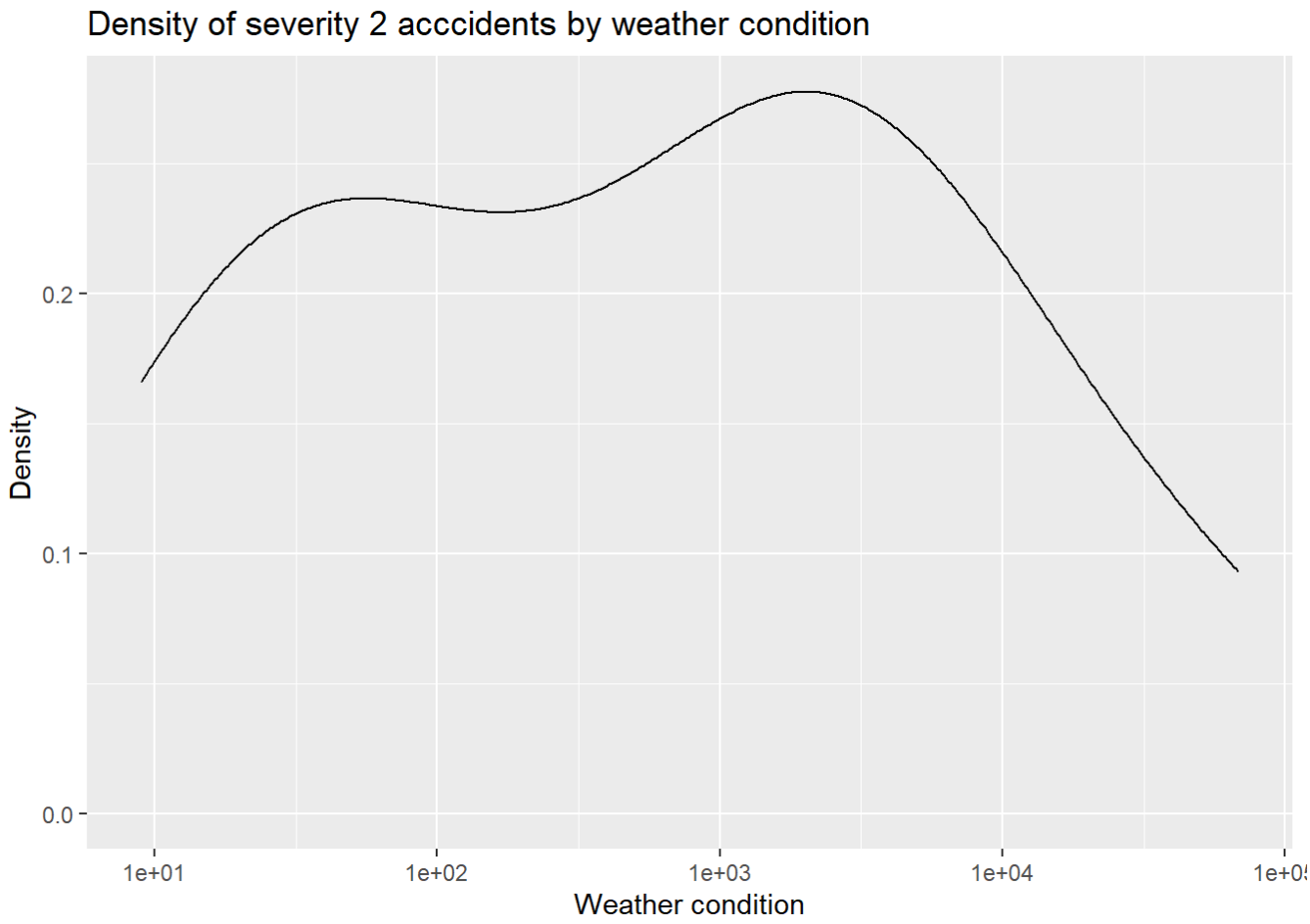
```
ggplot(data = accidents_train) + geom_bar(mapping = aes(x = weather_condition)) + coord_fl
ip()
```



Severity and Weather Condition

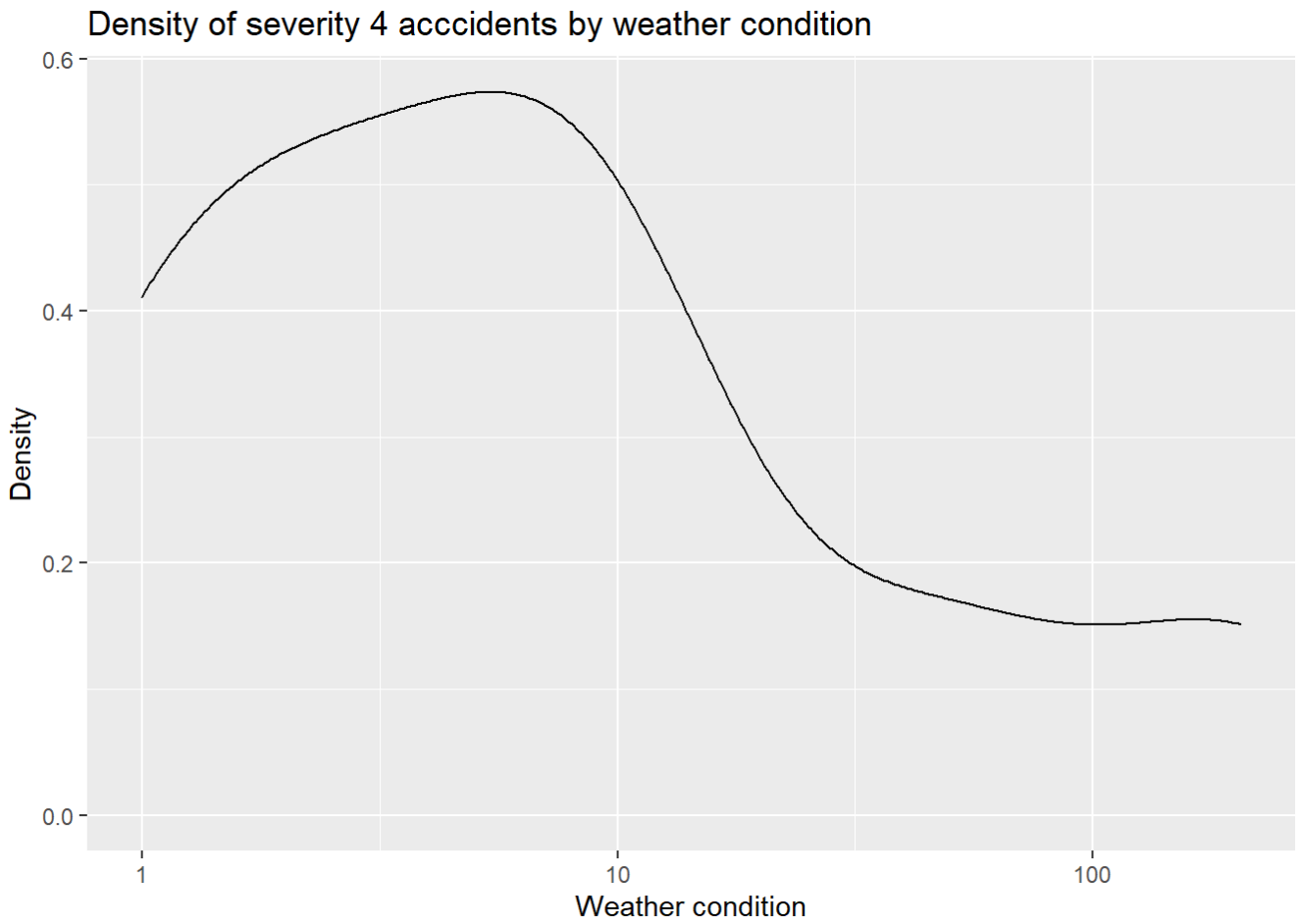
Hide

```
severity_2_dist <- accidents_train %>%
    group_by(weather_condition) %>%
    filter(severity==2) %>%
    summarise(count = n()) %>%
    ggplot() +
    geom_density(mapping = aes(x=count)) +
    scale_x_log10() +
    labs(x = "Weather condition", y="Density", title ="Density of severity 2 acccidents by
weather condition")

severity_4_dist <- accidents_train %>%
    group_by(weather_condition) %>%
    filter(severity==4) %>%
    summarise(count = n()) %>%
    ggplot() +
    geom_density(mapping = aes(x=count))+
    scale_x_log10() +
    labs(x = "Weather condition", y="Density", title ="Density of severity 4 acccidents by
weather condition")

print(severity_2_dist)
```



Hide

```
print(severity_4_dist)
```



Essentially, what this graph is telling is that as the amount of different weather conditions increase in our data set, there are fewer and fewer severity 2 accidents. But vice versa, there are few weather conditions that have a high number of severity 2 accidents.

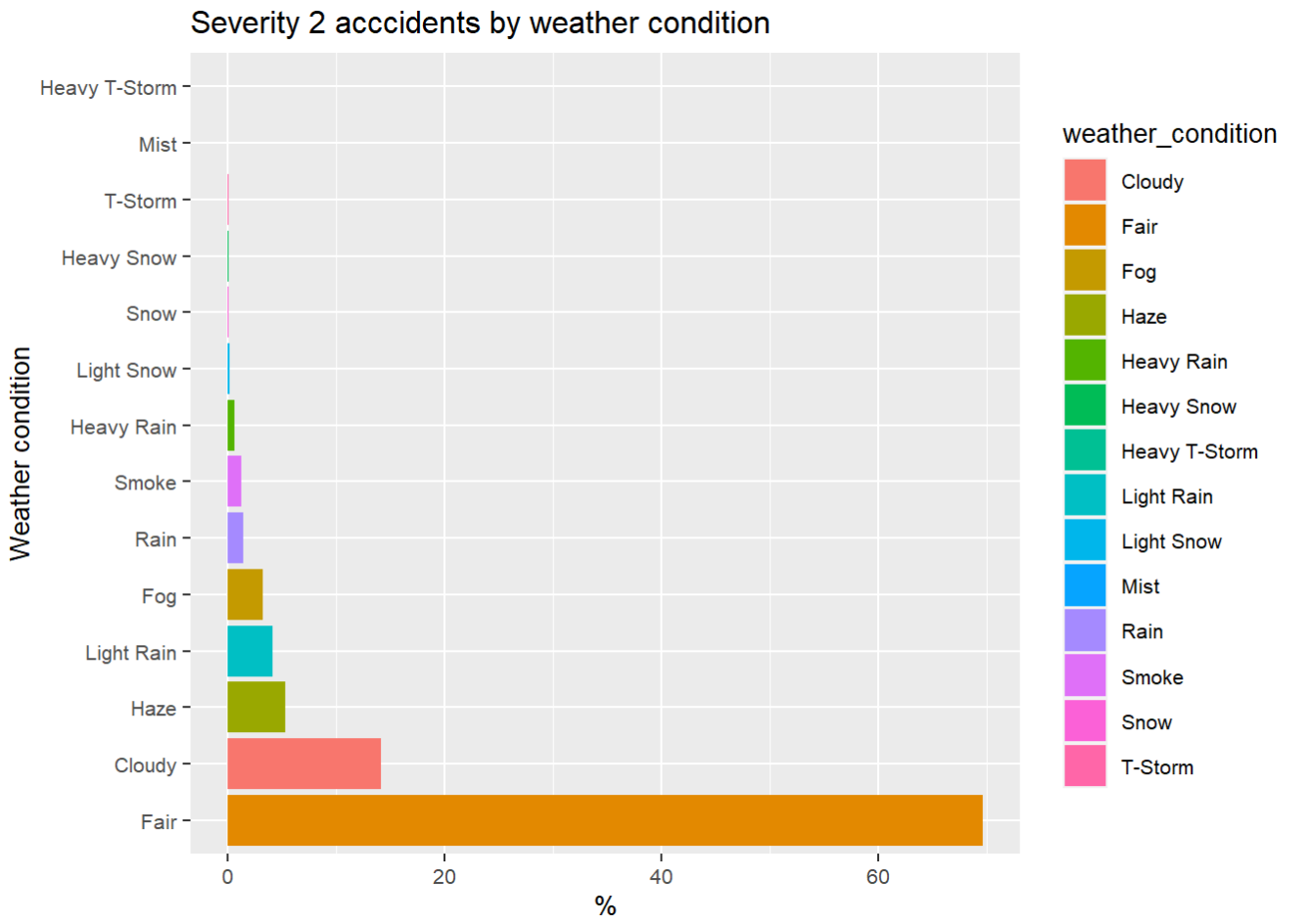
Hide

```
severity_2 <- accidents_train %>%
    group_by(weather_condition) %>%
    filter(severity==2) %>%
    summarise(count = n()/nrow(.)*100) %>%
    arrange(-count) -> severity2_weather
severity_2
```

```
## # A tibble: 14 × 2
##    weather_condition    count
##    <fct>                <dbl>
##  1 Fair              69.6
##  2 Cloudy            14.1
##  3 Haze               5.31
##  4 Light Rain         4.12
##  5 Fog                3.24
##  6 Rain               1.44
##  7 Smoke              1.20
##  8 Heavy Rain         0.611
##  9 Light Snow         0.198
## 10 Snow               0.0809
## 11 Heavy Snow         0.0379
## 12 T-Storm            0.0328
## 13 Mist               0.0174
## 14 Heavy T-Storm      0.00922
```

Hide

```
severity_2_plot <- severity2_weather %>%
    ggplot() +
    geom_col(mapping=aes(x=reorder(weather_condition, -count), y=count, fill=weather_condi
tion))+
    theme(text = element_text(size=10)) +
    labs(x = "Weather condition", y="%", title ="Severity 2 acccidents by weather conditio
n") +
    coord_flip()
severity_2_plot
```



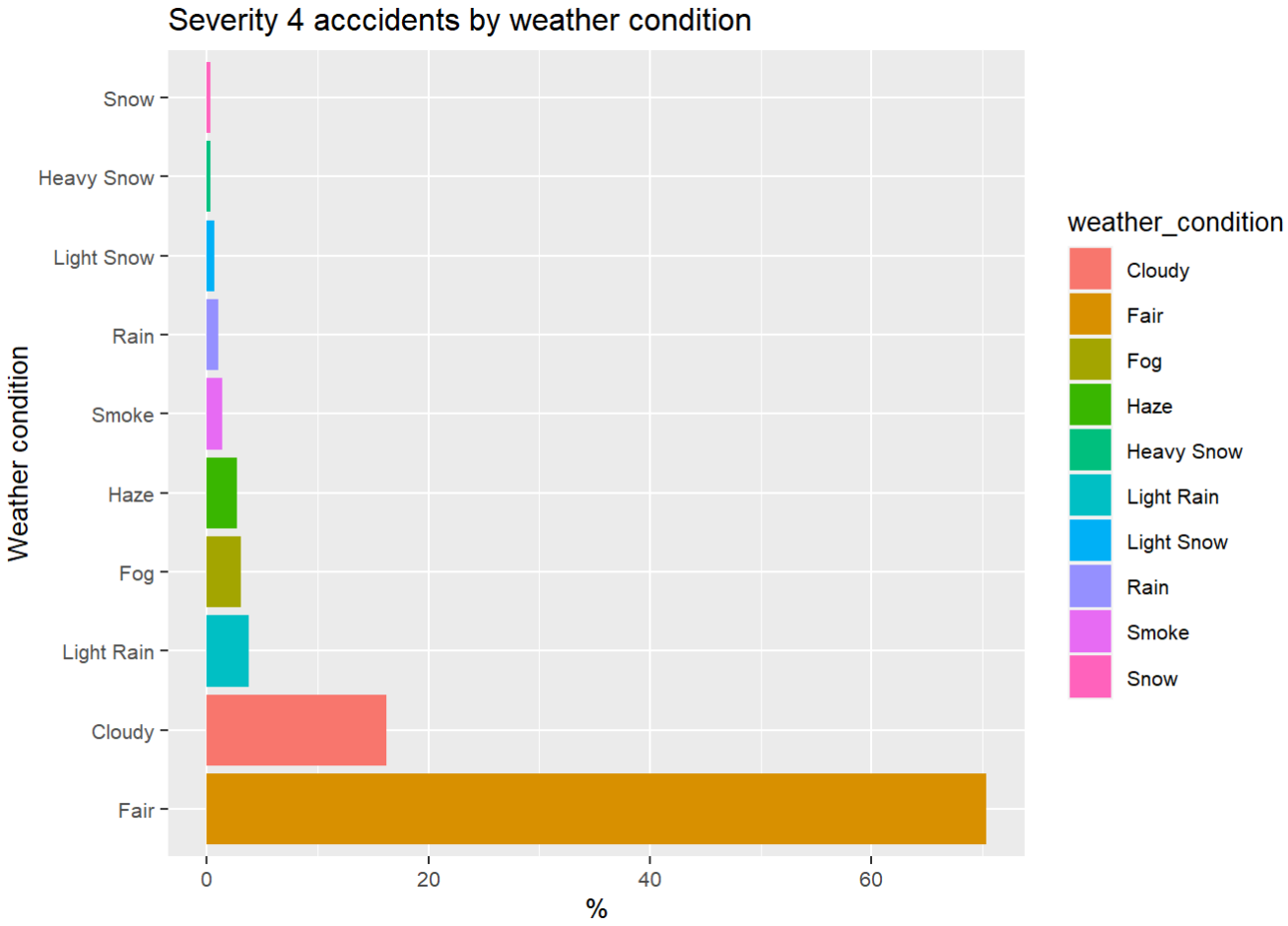
Hide

```
severity_4 <- accidents_train %>%
    group_by(weather_condition) %>%
    filter(severity==4) %>%
    summarise(count = n()/nrow(.)*100) %>%
    arrange(-count)
severity_4
```

```
## # A tibble: 10 × 2
##    weather_condition  count
##    <fct>              <dbl>
##  1 Fair              70.3
##  2 Cloudy            16.2
##  3 Light Rain         3.79
##  4 Fog                3.10
##  5 Haze               2.76
##  6 Smoke              1.38
##  7 Rain               1.03
##  8 Light Snow         0.690
##  9 Heavy Snow         0.345
## 10 Snow               0.345
```

Hide

```
severity_4_plot <- severity_4 %>%
    ggplot() +
    geom_col(mapping=aes(x=reorder(weather_condition, -count), y=count, fill=weather_condi
tion))+
    theme(text = element_text(size=10)) +
    labs(x = "Weather condition", y="%", title ="Severity 4 acccidents by weather conditio
n") +
    coord_flip()
severity_4_plot
```



Model Building

Recipe, Folds, and Saving the Data

Hide

```
accidents_folds <- vfold_cv(accidents_train, v=5, strata=severity)

accidents_recipe <- recipe(severity ~ ., data = accidents_train) %>%
  step_dummy(side) %>%
  step_dummy(weather_condition) %>%
  step_dummy(bump) %>%
  step_dummy(crossing) %>%
  step_dummy(railway) %>%
  step_dummy(stop) %>%
  step_dummy(traffic_signal) %>%
  step_dummy(sunrise_sunset) %>%
  step_normalize(all_predictors()) %>%
  step_upsample(severity,over_ratio = 1)
```

Preparing and running the Models

The models I'll be using are:

- logistic regression
- single decision tree
- boosted tree
- random forests
- k-nearest neighbors

logistic regression

Hide

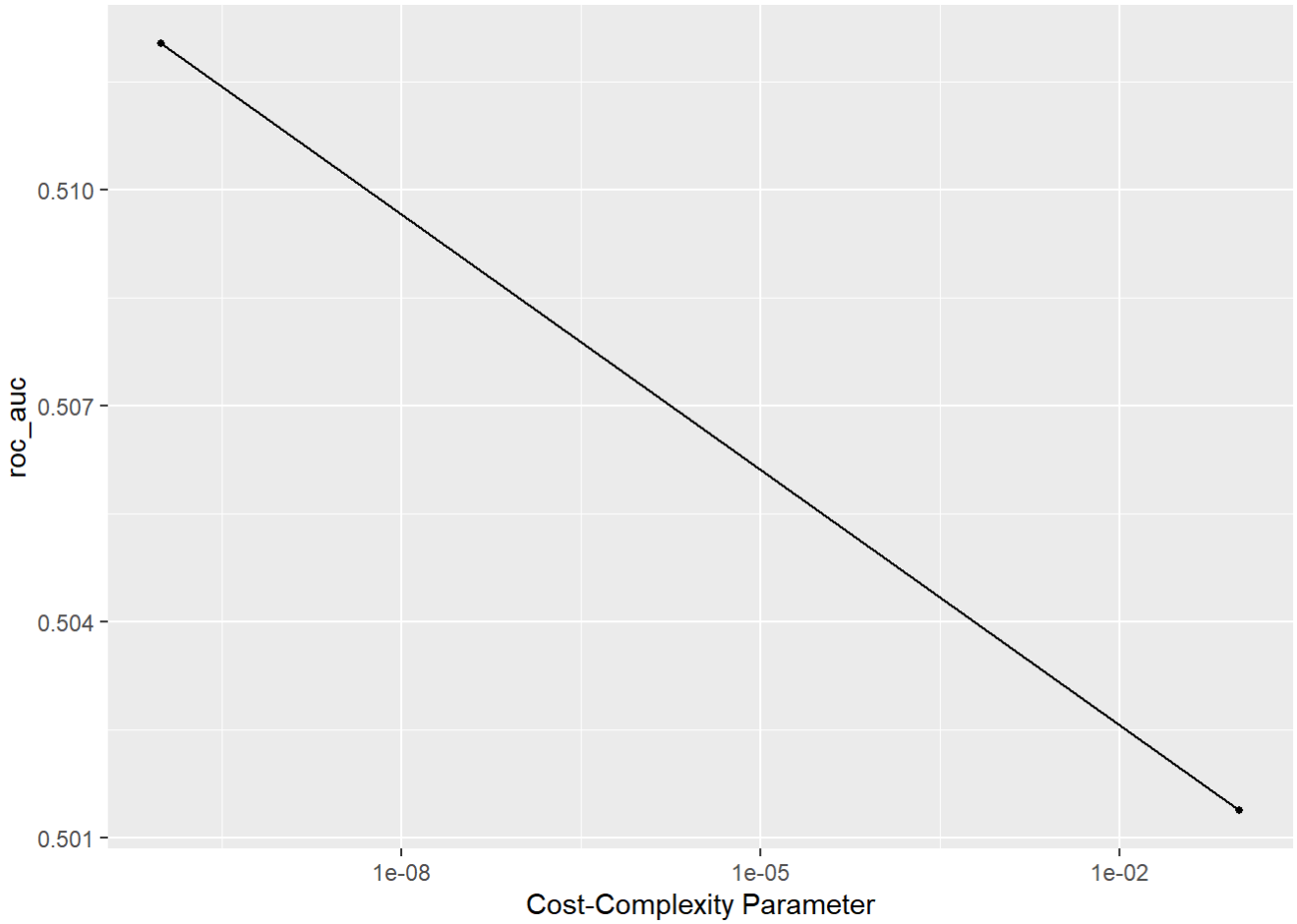
```
log_reg <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

log_wkflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(accidents_recipe)

log_fit <- fit(log_wkflow,accidents_train)
log_fit %>%
  tidy()
```

```
## # A tibble: 29 × 5
##    term             estimate std.error statistic   p.value
##    <chr>               <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)       -0.182    0.226      -0.804 4.21e-  1
##  2 distance_mi       -0.0573   0.00484   -11.8   2.78e- 32
##  3 temperature_f      0.750    0.0709     10.6   4.27e- 26
##  4 wind_chill_f      -0.795    0.0702    -11.3   9.71e- 30
##  5 humidity_percent   0.0271   0.00820     3.31  9.33e-  4
##  6 pressure_in       -0.0972   0.00504   -19.3   7.78e- 83
##  7 visibility_mi      0.0616   0.00430    14.3   1.46e- 46
##  8 wind_speed_mph    -0.116    0.00530   -21.9   2.34e-106
##  9 precipitation_in  -0.135    0.00957   -14.1   5.09e- 45
## 10 side_R            -0.0753   0.00467   -16.1   2.25e- 58
## # … with 19 more rows
```

single decision tree

Hide

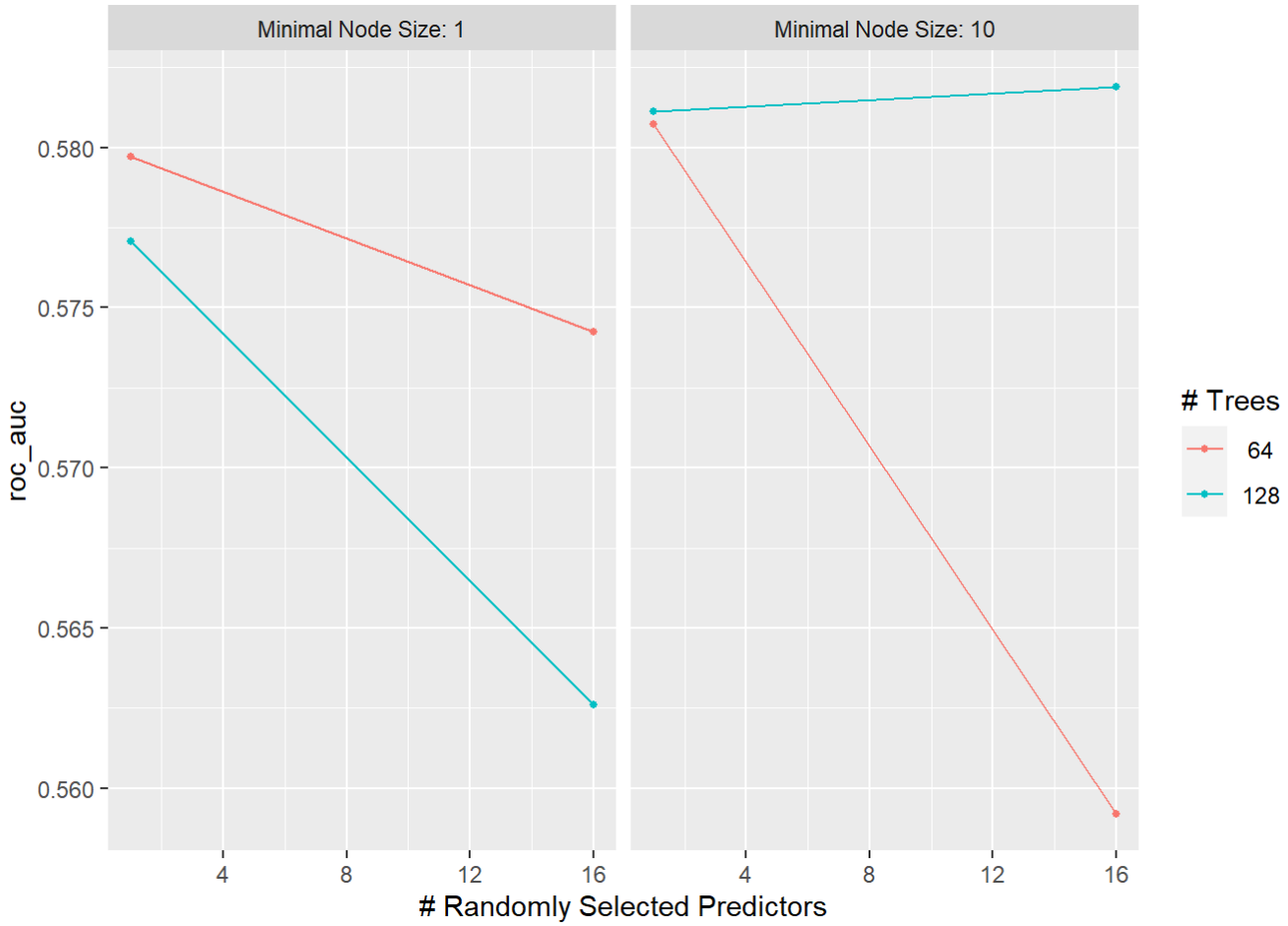
```
tree_spec <- decision_tree() %>%
  set_engine("rpart")

class_tree_spec <- tree_spec %>%
  set_mode("classification")

class_tree_wf <- workflow() %>%
  add_model(class_tree_spec %>%
  set_args(cost_complexity = tune())) %>%
  add_recipe(accidents_recipe)

class_tree_param_grid <- grid_regular(cost_complexity(range = c(-10, -1)), levels = 2)

class_tree_tune_res <- tune_grid(
  class_tree_wf,
  resamples = accidents_folds,
  grid = class_tree_param_grid,
  metrics = metric_set(roc_auc)
)
autoplot(class_tree_tune_res)
```



random forests

Hide

```
spec <- rand_forest() %>%
  set_engine("ranger", importance = "impurity")
rf_spec <- spec %>%
  set_mode("classification")
rf_spec_wf <- workflow() %>%
  add_model(rf_spec %>% set_args(mtry = tune(),trees = tune(), min_n = tune())) %>%
  add_recipe(accidents_recipe)
rf_param_grid <- grid_regular(mtry(range = c(1,16)),trees(range = c(64,128)),min_n(range =
c(1,10)),levels = c(2,2,2))
rf_tune_res <- tune_grid(
  rf_spec_wf,
  resamples = accidents_folds,
  grid = rf_param_grid,
  metrics = metric_set(roc_auc)
)
autoplot(rf_tune_res)
```



boosted tree

k-nearest neighbors