# ANNEX

## Contents

# 1. Data Cleaning

## 1.1. Accidents Datasets

Required packages:

```
require(tidyr)
require(dplyr)
require(data.table)
require(skimr)
require(psych)
require(purrr)
require(Hmisc)
require(lsr)
require(DescTools)
require(arsenal)
require(sjmisc)
require(zoo)
require(ggplot2)
require(ggExtra)
require(ggcorrplot)
require(ggpubr)
require(hrbrthemes)
require(RColorBrewer)
require(viridis)
require(geojsonio)
require(broom)
require(mapproj)
require(kableExtra)
require(stringr)
require(zoo)
require(tidyverse)
require(tidymodels)
require(themis)
require(vip)
require(randomForest)
require(rpart)
require(rpart.plot)

theme_set(theme_minimal())
```

Import datasets from 2018 and 2019:

```
dfc_18 = read.csv("datasets/accidents_causes_gu_bcn/2018_accidents_causes_gu_bcn_.csv",
    fileEncoding = "UTF-8")
dfg_18 = read.csv("datasets/accidents_gu_bcn/2018_accidents_gu_bcn.csv",
    fileEncoding = "UTF-8")
dfp_18 = read.csv("datasets/accidents_persones_gu_bcn/2018_accidents_persones_gu_bcn_.csv",
    fileEncoding = "UTF-8")
dft_18 = read.csv("datasets/accidents_tipus_gu_bcn/2018_accidents_tipus_gu_bcn_.csv",
    fileEncoding = "UTF-8")
dfv_18 = read.csv("datasets/accients_vehicles_gu_bcn/2018_accidents_vehicles_gu_bcn_.csv",
    fileEncoding = "UTF-8")

dfc_19 = read.csv("datasets/accidents_causes_gu_bcn/2019_accidents_causes_gu_bcn_.csv",
    fileEncoding = "UTF-8")
```

```
dfg_19 = read.csv("datasets/accidents_gu_bcn/2019_accidents_gu_bcn.csv",
    fileEncoding = "UTF-8")
dfp_19 = read.csv("datasets/accidents_persones_gu_bcn/2019_accidents_persones_gu_bcn_.csv",
    fileEncoding = "UTF-8")
dft_19 = read.csv("datasets/accidents_tipus_gu_bcn/2019_accidents_tipus_gu_bcn_.csv",
    fileEncoding = "UTF-8")
dfv_19 = read.csv("datasets/accients_vehicles_gu_bcn/2019_accidents_vehicles_gu_bcn_.csv",
    fileEncoding = "UTF-8")
```

Fix headers:

```
names_c = c("COD_EXPD", "COD_DIST", "NOM_DIST", "COD_BARR", "NOM_BARR",
    "COD_CARR", "NOM_CARR", "NUM_POST", "NOM_DIA", "DIA_SETM",
    "DIA_TIPUS", "ANY", "MES", "NOM_MES", "DIA", "HORA", "TORN",
    "CAUSA", "X", "Y", "LONG", "LAT")
colnames(dfc_18) = names_c
colnames(dfc_19) = names_c
dfc = rbind(dfc_18, dfc_19)
rm(dfc_18, dfc_19)

names_g = c("COD_EXPD", "COD_DIST", "NOM_DIST", "COD_BARR", "NOM_BARR",
    "COD_CARR", "NOM_CARR", "NUM_POST", "NOM_DIA", "DIA_SETM",
    "DIA_TIPUS", "ANY", "MES", "NOM_MES", "DIA", "HORA", "TORN",
    "CAUSA_V", "NUM_MS", "NUM_LL", "NUM_LG", "NUM_VICT", "NUM_VEHS",
    "X", "Y", "LONG", "LAT")
colnames(dfg_18) = names_g
colnames(dfg_19) = names_g
dfg = rbind(dfg_18, dfg_19)
rm(dfg_18, dfg_19)

names_p1 = c("COD_EXPD", "COD_DIST", "NOM_DIST", "COD_BARR",
    "NOM_BARR", "COD_CARR", "NOM_CARR", "NUM_POST", "NOM_DIA",
    "DIA_SETM", "DIA_TIPUS", "ANY", "MES", "NOM_MES", "DIA",
    "TORN", "HORA", "CAUSA_V", "TIP_VEH", "SEXE", "EDAT", "TIP_PER",
    "DESC_SIT", "DESC_VICT", "X", "Y", "LONG", "LAT")
names_p2 = c("COD_EXPD", "COD_DIST", "NOM_DIST", "COD_BARR",
    "NOM_BARR", "COD_CARR", "NOM_CARR", "NUM_POST", "NOM_DIA",
    "DIA_SETM", "DIA_TIPUS", "ANY", "MES", "NOM_MES", "DIA",
    "TORN", "HORA", "CAUSA_V", "TIP_VEH", "SEXE", "EDAT", "TIP_PER",
    "LLOC", "MOTIU_C", "MOTIU_V", "DESC_VICT", "X", "Y", "LONG",
    "LAT")
colnames(dfp_18) = names_p1
colnames(dfp_19) = names_p2
dfp_18 = dfp_18 %>% select(-setdiff(names_p1, names_p2))
dfp_19 = dfp_19 %>% select(-setdiff(names_p2, names_p1))
dfp = rbind(dfp_18, dfp_19)
rm(dfp_18, dfp_19)

names_t = c("COD_EXPD", "COD_DIST", "NOM_DIST", "COD_BARR", "NOM_BARR",
    "COD_CARR", "NOM_CARR", "NUM_POST", "NOM_DIA", "DIA_SETM",
    "DIA_TIPUS", "ANY", "MES", "NOM_MES", "DIA", "HORA", "TORN",
    "TIP_ACC", "X", "Y", "LONG", "LAT")
colnames(dft_18) = names_t
colnames(dft_19) = names_t
```

```
dft = rbind(dft_18, dft_19)
rm(dft_18, dft_19)

names_v = c("COD_EXPD", "COD_DIST", "NOM_DIST", "COD_BARR", "NOM_BARR",
    "COD_CARR", "NOM_CARR", "NUM_POST", "NOM_DIA", "DIA_SETM",
    "DIA_TIPUS", "ANY", "MES", "NOM_MES", "DIA", "HORA", "TORN",
    "CAUSA_V", "TIP_VEH", "MODEL_VEH", "MARCA_VEH", "COL_VEH",
    "TIP_CARNET", "T_CARNET", "X", "Y", "LONG", "LAT")
colnames(dfv_18) = names_v
colnames(dfv_19) = names_v
dfv = rbind(dfv_18, dfv_19)
rm(dfv_18, dfv_19)

rm(names_c, names_g, names_p1, names_p2, names_t, names_v)
```

Get common dataset and drop common columns:

```
common = c("COD_DIST", "NOM_DIST", "COD_BARR", "NOM_BARR", "COD_CARR",
    "NOM_CARR", "NUM_POST", "NOM_DIA", "DIA_SETM", "DIA_TIPUS",
    "ANY", "MES", "NOM_MES", "DIA", "HORA", "TORN", "X", "Y",
    "LONG", "LAT")

dfm = select(dfp, c("COD_EXPD", all_of(common)))
dfc = select(dfc, -all_of(common))
dfg = select(dfg, -c(all_of(common), "CAUSA_V"))
dfp = select(dfp, -all_of(common))
dft = select(dft, -all_of(common))
dfv = select(dfv, -c(all_of(common), "CAUSA_V"))

rm(common)
```

Trim COD_EXPD:

```
dfm$COD_EXPD = as.character(lapply(dfm$COD_EXPD, str_trim))
dfc$COD_EXPD = as.character(lapply(dfc$COD_EXPD, str_trim))
dfg$COD_EXPD = as.character(lapply(dfg$COD_EXPD, str_trim))
dfp$COD_EXPD = as.character(lapply(dfp$COD_EXPD, str_trim))
dft$COD_EXPD = as.character(lapply(dft$COD_EXPD, str_trim))
dfv$COD_EXPD = as.character(lapply(dfv$COD_EXPD, str_trim))
```

#### 1.1.1. Injury Severity And Victims Characteristics

Link worst injury severity to each COD_EXPD:

```
data = dfp

data = data.frame(lapply(data, as.character), stringsAsFactors = FALSE)
data = data %>% mutate(DESC_VICT = replace(DESC_VICT, grepl("Ferit lleu: Rebutja assistència sanitària"
    DESC_VICT, fixed = TRUE), "1. FL_RS")) %>% mutate(DESC_VICT = replace(DESC_VICT,
    grepl("Ferit lleu: Amb assistència", DESC_VICT, fixed = TRUE),
    "2. FL_AS")) %>% mutate(DESC_VICT = replace(DESC_VICT, grepl("Ferit lleu: Hospitalització",
    DESC_VICT, fixed = TRUE), "3. FL_HH")) %>% mutate(DESC_VICT = replace(DESC_VICT,
    grepl("Ferit greu", DESC_VICT, fixed = TRUE), "4. FG")) %>%
    mutate(DESC_VICT = replace(DESC_VICT, grepl("Mort", DESC_VICT,
        fixed = TRUE), "5. M"))
data = data.frame(lapply(data, as.factor))
```

```
data$EDAT = as.numeric(data$EDAT)

aux = select(data, c("COD_EXPD", "DESC_VICT"))
aux = aux[order(aux$COD_EXPD, aux$DESC_VICT, decreasing = c(FALSE,
    TRUE)), ]
aux = aux[!duplicated(aux$COD_EXPD), ]

data = data.frame(lapply(data, as.character), stringsAsFactors = FALSE)
aux = data.frame(lapply(aux, as.character), stringsAsFactors = FALSE)
for (i in 1:nrow(data)) {
    data$DESC_VICT[i] = aux$DESC_VICT[aux$COD_EXPD == data$COD_EXPD[i]]
}
rm(i)
```

Filter drivers:

```
data = data.frame(lapply(data, as.character), stringsAsFactors = FALSE)
data = data[data$TIP_PER == "Conductor", ]
data = data.frame(lapply(data, as.factor))
data$EDAT = as.numeric(data$EDAT)
table(data$TIP_PER, data$DESC_VICT)

set.seed(123)
data = data[sample(1:nrow(data)), ]
data = data[!duplicated(data$COD_EXP), ]

dfp = data
rm(aux, data)
```

### 1.1.2. Type Of Accident

Add label "Multiple" to accidents with more than one type of accident:

```
data = dft

data = data.frame(lapply(data, as.character), stringsAsFactors = FALSE)
data_dup = data[duplicated(data$COD_EXPD) | duplicated(data$COD_EXPD,
    fromLast = T), ] %>% filter(TIP_ACC != "Desconegut") %>%
    group_by(COD_EXPD) %>% summarise(TIP_ACC = ifelse(n() > 1 &
    TIP_ACC == "Altres", "Elimina", TIP_ACC)) %>% filter(TIP_ACC !=
    "Elimina") %>% group_by(COD_EXPD) %>% summarise(TIP_ACC = ifelse(n() >
    1, "Múltiple", TIP_ACC))

data = rbind(data[!duplicated(data$COD_EXPD) & !duplicated(data$COD_EXPD,
    fromLast = T), ], data_dup)
dft = data
rm(data, data_dup)
```

### 1.1.3. Number Of Victims And Vehicles

Some rows are fully duplicated, drop them:

```
data = dfg
data = unique(data)
dfg = data
rm(data)
```

Store cleaned datasets:

```
# write.csv(dfc, 'FinalDatasets/df_causes.csv', row.names=F)
# write.csv(dfg, 'FinalDatasets/df_implicats.csv',
# row.names=F) write.csv(dfp,
# 'FinalDatasets/df_gravetat.csv', row.names=F)
# write.csv(dft, 'FinalDatasets/df_tipus.csv', row.names=F)
# write.csv(dfv, 'FinalDatasets/df_vehicles.csv',
# row.names=F) write.csv(dfm, 'FinalDatasets/df_main.csv',
# row.names=F)
```

## 1.2. Traffic Density

Compute moving average for both 2018 and 2019:

```
all_trams_2018 = list.files(path = "datasets/trams/2018", full.names = TRUE,
    recursive = TRUE)
dftram = ldply(all_trams_2018, read.csv, header = TRUE, .progress = "text")

dftram = dftram %>% filter(estatActual != 0 & estatActual !=
    6)

dftram = dftram %>% mutate(data = as.character(data)) %>% mutate(data = as.POSIXct(data,
    format = "%Y%m%d%H%M%S"))

dftram_wrapped = dftram %>% group_by(data = cut(data, "1 hour")) %>%
    dplyr::summarise(avg_estat = mean(estatActual), num_obs = n())

# write.csv(dftram_wrapped,'datasets/trams/dftram_grouped_dates.csv',
# row.names=F)

all_trams_2019 = list.files(path = "datasets/trams/2019", full.names = TRUE,
    recursive = TRUE)
dftram = ldply(all_trams_2019, read.csv, header = TRUE, .progress = "text")

dftram = dftram %>% filter(estatActual != 0 & estatActual !=
    6)

dftram = dftram %>% mutate(data = as.character(data)) %>% mutate(data = as.POSIXct(data,
    format = "%Y%m%d%H%M%S"))

dftram_wrapped = dftram %>% group_by(data = cut(data, "1 hour")) %>%
    dplyr::summarise(avg_estat = mean(estatActual), num_obs = n())

# write.csv(dftram_wrapped,'datasets/trams/dftram_grouped_dates2.csv',
# row.names=F)
```

## 1.3. Meteorological Measures

Grop observations of each station and summarise by mean:

```
fabra = read.csv("datasets/meteo/2019_d5_observatori_fabra.csv")
raval = read.csv("datasets/meteo/2019_x4_barcelona_el_raval.csv")
uni = read.csv("datasets/meteo/2019_x8_barcelona_zona_universitaria.csv")

meteo = rbind(fabra, raval, uni) %>% mutate(DATA_LECTURA = as.Date(DATA_LECTURA,
    format = "%d/%m/%Y")) %>% filter(DATA_LECTURA >= as.Date("2018-01-01") &
    DATA_LECTURA < as.Date("2020-01-01")) %>% group_by(DATA_LECTURA) %>%
    summarise(across(.cols = where(is.numeric), .fns = mean)) %>%
    mutate(across(where(is.numeric), round, 3))

# write.csv(meteo, 'datasets/meteo/meteo_wrapped.csv',
# row.names=F)
```

## 1.4. Combined Dataset

Merge traffic density to common dataset:

```
dftram1 = read.csv("datasets/trams/dftram_grouped_dates.csv",
    stringsAsFactors = F)
dftram2 = read.csv("datasets/trams/dftram_grouped_dates2.csv",
    stringsAsFactors = F)
dftram = rbind(dftram1, dftram2)

dftram$data = as.POSIXct(dftram$data)
dfm = dfm[!duplicated(dfm$COD_EXPD), ] %>% mutate(data = ISOdatetime(ANY,
    MES, DIA, HORA, 0, 0))

dfm = merge(dfm, dftram, by = "data", all.x = T)
```

Merge meteorological measures to common dataset:

```
dfmeteo = read.csv("datasets/meteo/meteo_wrapped.csv", stringsAsFactors = F)

dfmeteo$DATA_LECTURA = as.POSIXct(dfmeteo$DATA_LECTURA)
colnames(dfmeteo)[colnames(dfmeteo) == "DATA_LECTURA"] = "data"

dfm = dfm %>% mutate(data = as.POSIXct(format(data, format = "%Y-%m-%d")))

dfm = merge(dfm, dfmeteo, by = "data", all.x = T)
```

Merge every dataset and store it:

```
dataset = merge(dfp, dfm, by = "COD_EXPD")
dataset = merge(dataset, dfc, by = "COD_EXPD")
dataset = merge(dataset, dft, by = "COD_EXPD")
dataset = merge(dataset, dfg, by = "COD_EXPD")

colnames(dataset)[colnames(dataset) == "data"] = "DATA"
colnames(dataset)[colnames(dataset) == "avg_estat"] = "AVG_ESTAT"
colnames(dataset)[colnames(dataset) == "num_obs"] = "NUM_OBS"

# write.csv(dataset, 'FinalDatasets/df_final.csv',
# row.names=F)
```

## 1.5. Mutate Combined Dataset

Group hours into intervals:

```
df = read.csv("FinalDatasets/df_final.csv")

df = df %>% mutate(INT_H = ifelse(HORA %in% c(22, 23, 0, 1),
    "22:00-01:59", ifelse(HORA %in% 2:5, "02:00-05:59", ifelse(HORA %in%
        6:9, "06:00-09:59", ifelse(HORA %in% 10:13, "10:00-13:59",
        ifelse(HORA %in% 14:17, "14:00-17:59", "18:00-21:59"))))))
```

Split day between weekday, weekend and holiday:

```
df = df %>% mutate(DIA_TIPUS = ifelse(DIA_SETM == "Dl" & HORA %in%
    1:2, "Festiu", ifelse(DIA_SETM == "Dv" & HORA %in% 22:23,
    "Festiu", ifelse(DIA_SETM %in% c("Ds", "Dg"), "Festiu", "Laboral"))))
```

```
df_festius = read.csv("datasets/festius.csv", stringsAsFactors = F)
colnames(df_festius) = "DATA"

df = df %>% mutate(DIA_TIPUS = if_else(DIA_TIPUS == "Festiu",
    "Finde", DIA_TIPUS))
table(df$DIA_TIPUS)

festius = function(df, df_festius) {
    for (i in 1:nrow(df_festius)) {
        data = df_festius$DATA[i]
        df = df %>% mutate(DIA_TIPUS = ifelse(DATA == data, "Festiu",
            DIA_TIPUS))
    }
    return(df)
}

df = festius(df, df_festius)
# write.csv(df, 'FinalDatasets/dataset.csv', row.names=F)
```

Change number of victims and vehicles to categorical variables:

```
df = read.csv("FinalDatasets/dataset.csv", stringsAsFactors = F)

df = df %>% mutate(NUM_VICT = case_when(NUM_VICT < 2 ~ "1", NUM_VICT <
    4 ~ "2-3", TRUE ~ ">3")) %>% mutate(NUM_VEHS = case_when(NUM_VEHS <
    2 ~ "1", NUM_VEHS < 3 ~ "2", NUM_VEHS < 4 ~ "3", TRUE ~ ">3"))

# write.csv(df, 'FinalDatasets/dataset.csv', row.names=F)
```

Translate each categorical variable for the exploratory analysis:

```
df = read.csv("FinalDatasets/dataset.csv", stringsAsFactors = F)

df = df %>% mutate(CAUSA_V = case_when(CAUSA_V == "Altres" ~
    "Others", CAUSA_V == "Creuar per fora pas de vianants" ~
    "Cross outside pedestrian crossing", CAUSA_V == "Desobeir altres senyals" ~
    "Disobey other signs", CAUSA_V == "Desobeir el senyal del semàfor" ~
    "Disobey traffic light", CAUSA_V == "No és causa del  vianant" ~
    "Not the pedestrian fault", CAUSA_V == "Transitar a peu per la calçada" ~
    "Walk along the road")) %>%
mutate(TIP_VEH = case_when(TIP_VEH == "Altres vehicles amb motor" ~
    "Other motor vehicles", TIP_VEH == "Altres vehicles sense motor" ~
    "Other non-motor vehicles", TIP_VEH == "Autobús" ~ "Bus",
    TIP_VEH == "Autobús articulat" ~ "Articulated bus", TIP_VEH ==
        "Autocar" ~ "Coach", TIP_VEH == "Camió rígid <= 3,5 tones" ~
        "Rigid truck <= 3.5 tons", TIP_VEH == "Camió rígid > 3,5 tones" ~
        "Rigid truck > 3.5 tons", TIP_VEH == "Ciclomotor" ~ "Moped",
    TIP_VEH == "Desconegut" ~ "Unknown", TIP_VEH == "Furgoneta" ~
        "Van", TIP_VEH == "Maquinària d'obres i serveis" ~ "Construction machinery",
    TIP_VEH == "Motocicleta" ~ "Motorcycle", TIP_VEH == "Quadricicle < 75 cc" ~
        "Quadricycle < 75 cc", TIP_VEH == "Quadricicle > 75 cc" ~
        "Quadricycle > 75 cc", TIP_VEH == "Taxi" ~ "Taxi", TIP_VEH ==
        "Tot terreny" ~ "Off-road", TIP_VEH == "Tractor camió" ~
        "Truck tractor", TIP_VEH == "Turisme" ~ "Car", TIP_VEH ==
        "Veh. mobilitat personal amb motor" ~ "Personal mobility device (motor)",
```

```
    TIP_VEH == "Veh. mobilitat personal sense motor" ~ "Personal mobility device (no motor)",
    TIP_VEH == "Tren o tramvia" ~ "Train / Tram", TIP_VEH ==
        "Bicicleta" ~ "Bike")) %>%
mutate(SEXE = case_when(SEXE == "Home" ~ "Male", SEXE == "Dona" ~
    "Female", TRUE ~ "Unknown")) %>%
mutate(DESC_VICT = case_when(DESC_VICT == "1. FL_RS" ~ "No Injury",
    DESC_VICT == "2. FL_AS" ~ "Mild", DESC_VICT == "3. FL_HH" ~
        "Moderate", DESC_VICT == "4. FG" ~ "Severe", DESC_VICT ==
        "5. M" ~ "Fatal")) %>%
mutate(NOM_DIA = case_when(NOM_DIA == "Dilluns" ~ "Monday", NOM_DIA ==
    "Dimarts" ~ "Tuesday", NOM_DIA == "Dimecres" ~ "Wednesday",
    NOM_DIA == "Dijous" ~ "Thursday", NOM_DIA == "Divendres" ~
        "Friday", NOM_DIA == "Dissabte" ~ "Saturday", NOM_DIA ==
        "Diumenge" ~ "Sunday")) %>%
mutate(DIA_TIPUS = case_when(DIA_TIPUS == "Festiu" ~ "Holiday",
    DIA_TIPUS == "Laboral" ~ "Weekday", DIA_TIPUS == "Finde" ~
        "Weekend")) %>%
mutate(NOM_MES = case_when(NOM_MES == "Gener" ~ "January", NOM_MES ==
    "Febrer" ~ "February", NOM_MES == "Març" ~ "March", NOM_MES ==
    "Abril" ~ "April", NOM_MES == "Maig" ~ "May", NOM_MES ==
    "Juny" ~ "June", NOM_MES == "Juliol" ~ "July", NOM_MES ==
    "Agost" ~ "August", NOM_MES == "Setembre" ~ "September",
    NOM_MES == "Octubre" ~ "October", NOM_MES == "Novembre" ~
        "November", NOM_MES == "Desembre" ~ "December")) %>%
mutate(TORN = case_when(TORN == "Matí" ~ "Morning", TORN == "Tarda" ~
    "Afternoon", TORN == "Nit" ~ "Night")) %>%
mutate(CAUSA = case_when(CAUSA == "Alcoholèmia" ~ "Alcohol",
    CAUSA == "Calçada en mal estat" ~ "Road conditions", CAUSA ==
        "Drogues o medicaments" ~ "Drugs", CAUSA == "Estat de la senyalització" ~
        "Bad signaling", CAUSA == "Excés de velocitat o inadequada" ~
        "Speeding", CAUSA == "Factors meteorològics" ~ "Meteorological factors",
    CAUSA == "No hi ha causa mediata" ~ "Unknown", CAUSA == "Objectes o animals a la calçada" ~
        "Road obstacles")) %>%
mutate(TIP_ACC = case_when(TIP_ACC == "Abast" ~ "Rear-end", TIP_ACC ==
    "Abast multiple" ~ "Chain reaction", TIP_ACC == "Altres" ~
    "Others", TIP_ACC == "Atropellament" ~ "Rollover", TIP_ACC ==
    "Bolcada (més de dues rodes)" ~ "Dump (>2 wheels)", TIP_ACC ==
    "Caiguda (dues rodes)" ~ "Fall (2 wheels)", TIP_ACC == "Caiguda interior vehicle" ~
    "Inside vehicle fall", TIP_ACC == "Col.lisió frontal" ~ "Head-on",
    TIP_ACC == "Col.lisió lateral" ~ "Sideswipe", TIP_ACC ==
        "Col.lisió fronto-lateral" ~ "Side-impact", TIP_ACC ==
        "Desconegut" ~ "Unknown", TIP_ACC == "Encalç" ~ "Rear-end",
    TIP_ACC == "Múltiple" ~ "Multiple", TIP_ACC == "Resta sortides de via" ~
        "Derailment", TIP_ACC == "Sortida de via amb xoc o col.lisió" ~
        "Derailment", TIP_ACC == "Xoc amb animal a la calçada" ~
        "Animal collision", TIP_ACC == "Xoc contra element estàtic" ~
        "Obstacle collision"))
```

Drop unused columns:

```
df = df %>% select(-c("TIP_PER", "COD_DIST", "NOM_BARR", "COD_BARR",
    "NOM_CARR", "COD_CARR", "NUM_POST", "DIA_SETM", "MES", "DIA",
    "HORA", "X", "Y", "NUM_OBS", "NUM_MS", "NUM_LL", "NUM_LG"))
```

```
df = df %>% filter(NOM_DIST != "Desconegut") %>% filter(SEXE !=
    "Unknown")
```

Impute age:

```
vehicle_imp = c("Bike", "Othor motor vehicles", "Other non-motor vehicles",
    "Personal mobility device (motor)", "Personal mobility device (no motor)")
aux = df %>% filter(EDAT > 15)
mean_age = round(mean(aux$EDAT), 0)
df = df %>% mutate(EDAT = ifelse(EDAT < 16 & !(TIP_VEH %in% vehicle_imp),
    mean_age, EDAT))
```

Store the combined dataset:

```
df = df[complete.cases(df), ]
# write.csv(df, 'FinalDatasets/dataset.csv', row.names=F)
rm(list = ls())
```

# 2. Exploratory Analysis

Load combined dataset:

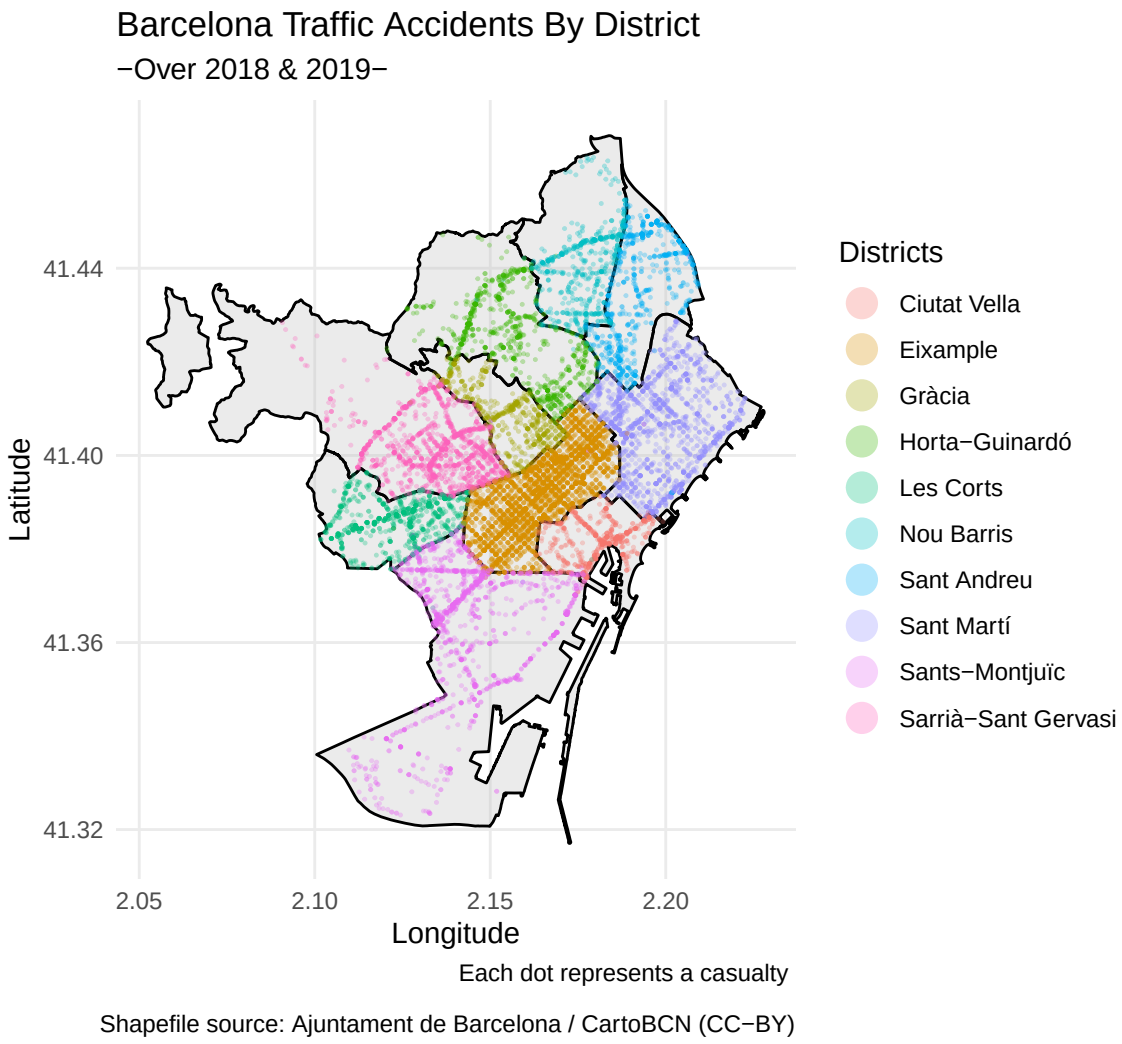
```
df = read.csv("FinalDatasets/dataset.csv")

order = c("No Injury", "Mild", "Moderate", "Severe", "Fatal")
df = df %>% mutate(DESC_VICT = ordered(DESC_VICT, levels = order)) %>%
    mutate(ANY = as.factor(ANY)) %>% mutate(DATA = as.POSIXct(DATA)) %>%
    select(-c("COD_EXPD")) %>% mutate_at(c("LAT", "LONG"), as.numeric) %>%
    mutate(LONG = ifelse(LONG > 1000, LONG/1000, LONG)) %>% mutate(LAT = ifelse(LAT >
    1000, LAT/1000, LAT))

df_categorical = df[, sapply(df, is.factor)]
df_numerical = df[, sapply(df, is.numeric) | names(df) %in% c("DESC_VICT")]
rm(order)
```

## 2.1. Explore Data

### 2.1.1. Mapping Of The Accidents In Barcelona

Plot each accident from 2018 and 2019:

```
map <- geojson_read("datasets/bcn-geodata-master/districtes/districtes.geojson", what = "sp")
goodmap = broom::tidy(map)

ggplot() +
  geom_polygon(data = goodmap,
               aes(x = long, y = lat, group = group), fill="grey", color = "black", alpha=0.3) +
  geom_point(data = df, aes(x = LONG, y = LAT, color=NOM_DIST), alpha=0.3, size = 0.2) +
  coord_map() +

  labs(title = "Barcelona Traffic Accidents By District",
       subtitle = "-Over 2018 & 2019-",

       caption = "Each dot represents a casualty \n
       Shapefile source: Ajuntament de Barcelona / CartoBCN (CC-BY)",
       x = "Longitude", y = "Latitude",
       color = "Districts") +
  guides(colour = guide_legend(override.aes = list(size=5, alpha = 0.3)))
```

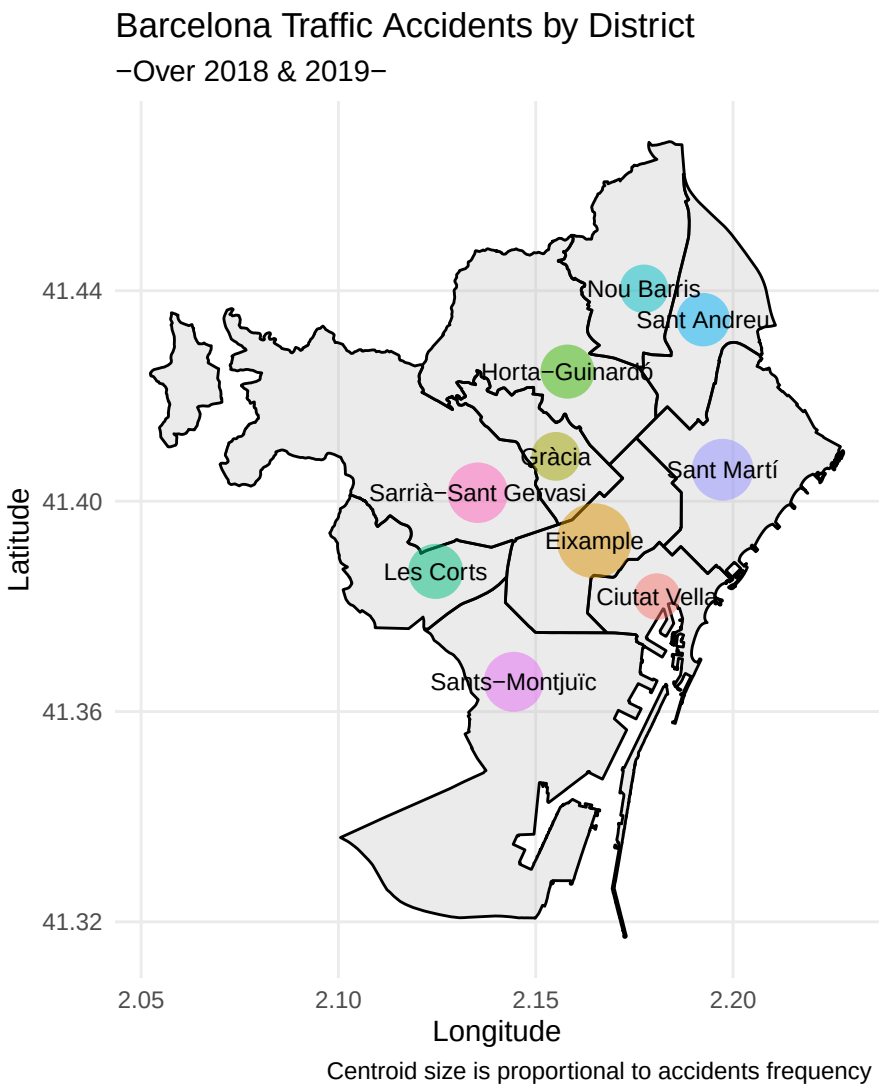



### 2.1.2. Mapping Of The Accidents By District

Plot frequency centroids of all accidents and severe only:

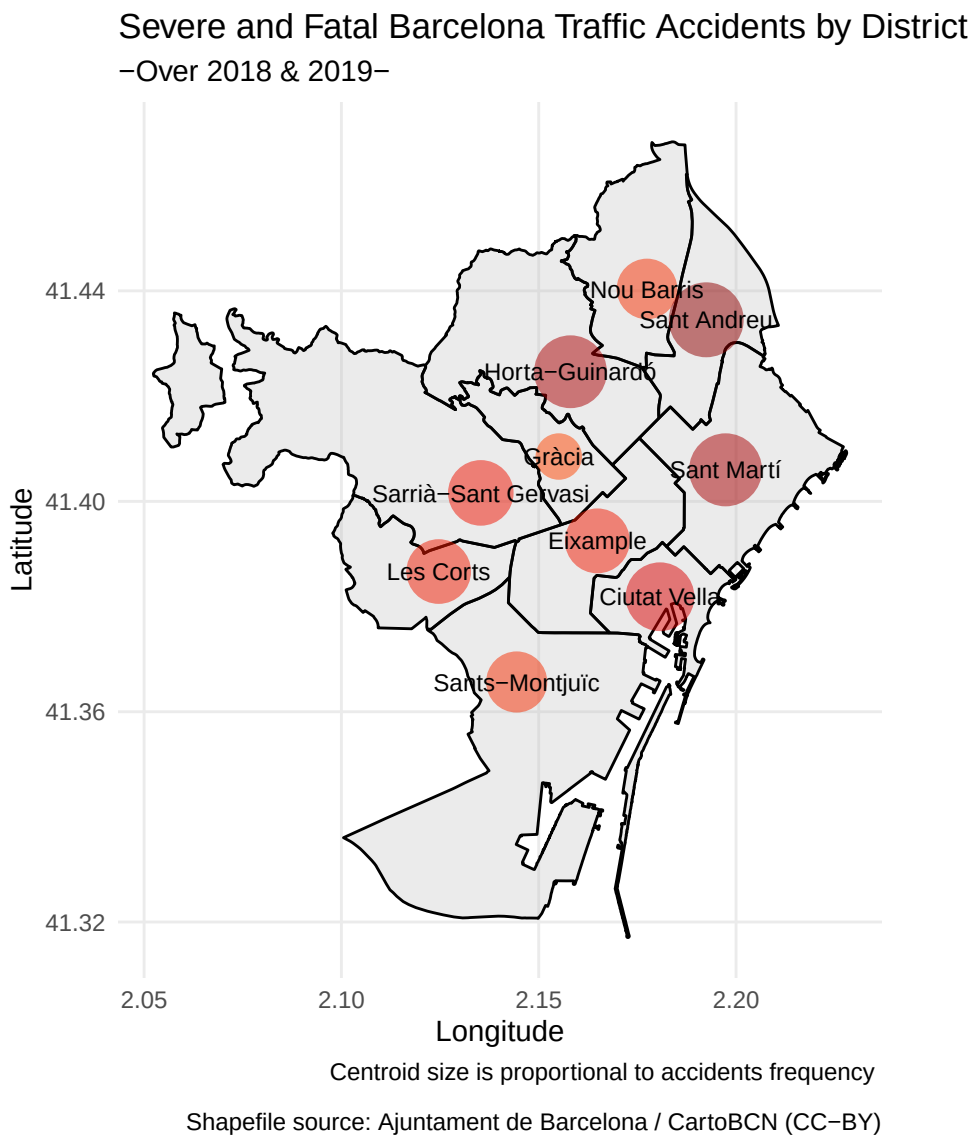
```
centroids_list = list()
centroids = df %>%
  add_count(NOM_DIST) %>%
  group_by(NOM_DIST, n) %>%
  summarise_at(vars(LONG, LAT), ~ format(round(mean(.), 20))) %>%
  mutate_at(vars(LONG, LAT), as.numeric) %>%
  as.data.frame()

severe_count = df %>%
  filter(DESC_VICT %in% c("Severe", "Fatal")) %>%
  add_count(NOM_DIST) %>%
  group_by(NOM_DIST, n) %>%
  summarise() %>%
```

```
  rename(v = n) %>%
  as.data.frame()

centroids = merge(centroids, severe_count, by = "NOM_DIST") %>%
  mutate(freq = v/n * 100)


ggplot() +
  geom_polygon(data = goodmap,
               aes(x = long, y = lat, group = group), fill="grey", color = "black", alpha=0.3) +
  geom_point(data = centroids, aes(x = LONG, y = LAT, color=NOM_DIST, size=n), alpha=0.5) +
  scale_size_continuous(range = c(7.5,12.5)) +
  geom_text(data = centroids, aes(x = LONG, y = LAT, label = NOM_DIST), size=3) +
  coord_map() +

  labs(title = "Barcelona Traffic Accidents by District",
       subtitle = "-Over 2018 & 2019-",

       caption = "Centroid size is proportional to accidents frequency \n
       Shapefile source: Ajuntament de Barcelona / CartoBCN (CC-BY)",
       x = "Longitude", y = "Latitude") +
  theme(legend.position = "none")
```




```
ggplot() +
  geom_polygon(data = goodmap,
               aes(x = long, y = lat, group = group), fill="grey", color = "black", alpha=0.3) +
  geom_point(data = centroids, aes(x = LONG, y = LAT, colour=freq, size=freq), alpha=0.5) +
  scale_colour_gradient2(
    low = "orangered",
    mid = "red2",
    high = "red4",
    midpoint = mean(centroids$freq)
  ) +
  scale_size_continuous(range = c(7.5,12.5)) +
  geom_text(data = centroids, aes(x = LONG, y = LAT, label = NOM_DIST), size=3) +
  coord_map() +

   labs(title = "Severe and Fatal Barcelona Traffic Accidents by District",
       subtitle = "-Over 2018 & 2019-",
```

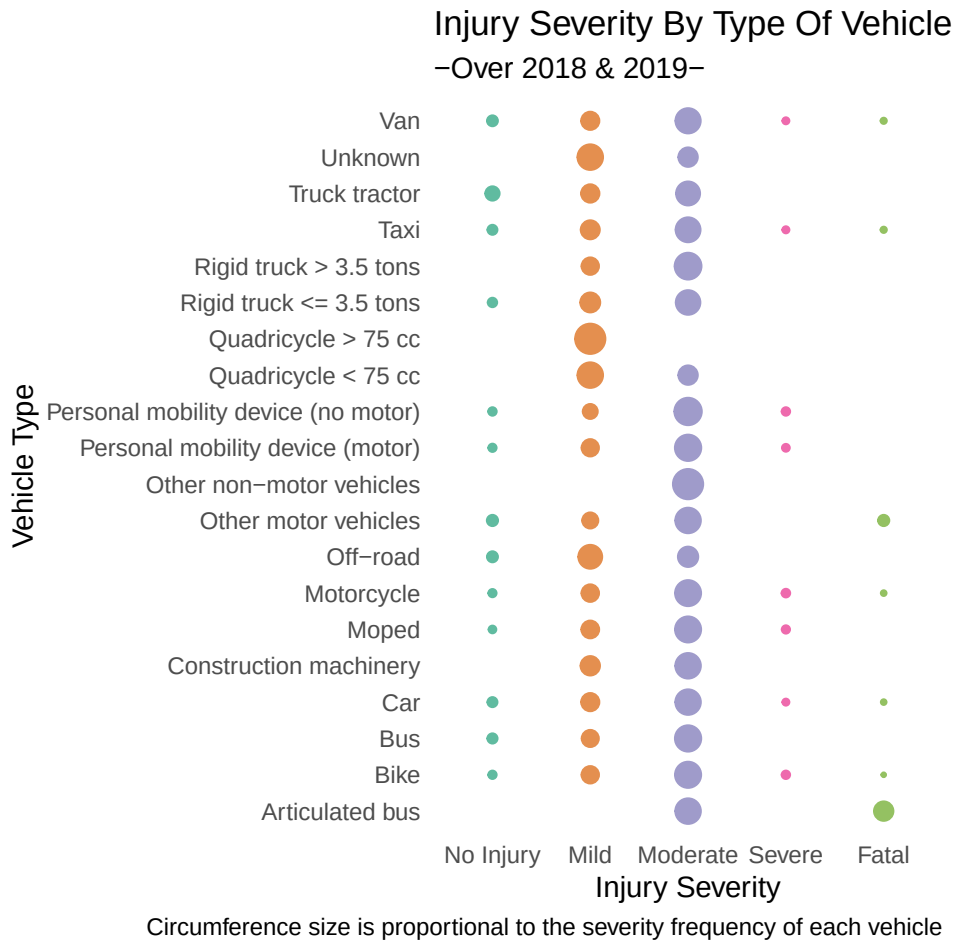
```
      caption = "Centroid size is proportional to accidents frequency \n
      Shapefile source: Ajuntament de Barcelona / CartoBCN (CC-BY)",
      x = "Longitude", y = "Latitude") +
theme(legend.position = "none")
```



### 2.1.3. Bubble Charts Of Categorical Variables Vs. Response Variable

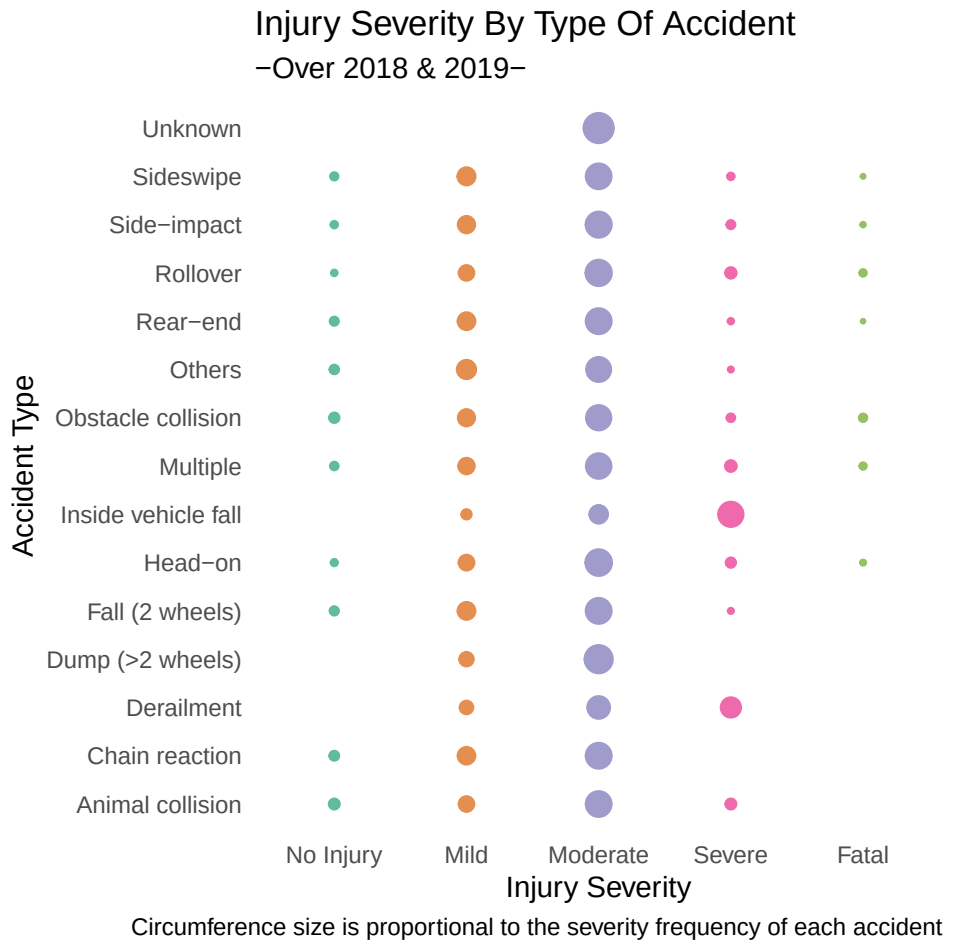
Visualize the contents of the contingency table. Type of vehicle vs. injury severity:

```
df %>%
  group_by(TIP_VEH, DESC_VICT) %>%
  summarise(n = n()) %>%
  mutate(freq = n / sum(n)) %>%
  ggplot(aes(x=TIP_VEH, y=DESC_VICT, color=DESC_VICT)) +
  geom_point(aes(size=freq), alpha=0.7) +
  scale_color_brewer(palette = "Dark2") +
  scale_size_continuous(range = c(0.5,5)) +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
  coord_flip() +

  labs(title = "Injury Severity By Type Of Vehicle",
       subtitle = "-Over 2018 & 2019-",

       caption = "Circumference size is proportional to the severity frequency of each vehicle",
       x = "Vehicle Type", y = "Injury Severity") +
  theme(legend.position = "none")
```




Type of accident vs. injury severity:

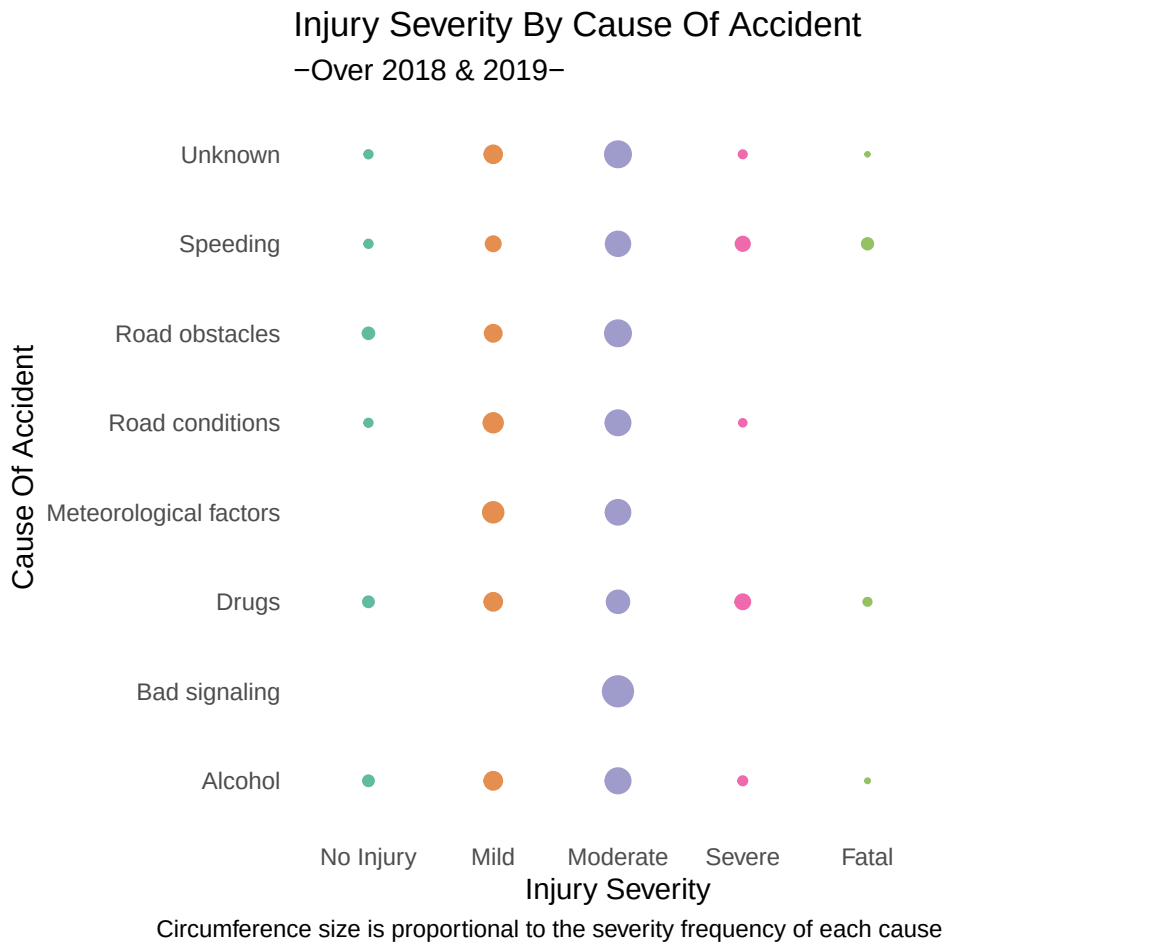
```r
df %>%
  group_by(TIP_ACC, DESC_VICT) %>%
  summarise(n = n()) %>%
  mutate(freq = n / sum(n)) %>%
  ggplot(aes(x=TIP_ACC, y=DESC_VICT, color=DESC_VICT)) +
  geom_point(aes(size=freq), alpha=0.7) +
  scale_color_brewer(palette = "Dark2") +
  scale_size_continuous(range = c(0.5,5)) +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
  coord_flip() +

  labs(title = "Injury Severity By Type Of Accident",
       subtitle = "-Over 2018 & 2019-",

       caption = "Circumference size is proportional to the severity frequency of each accident",
       x = "Accident Type", y = "Injury Severity") +
  theme(legend.position = "none")
```




Cause vs. injury severity:

```r
df %>%
  group_by(CAUSA, DESC_VICT) %>%
```

```r
summarise(n = n()) %>%
mutate(freq = n / sum(n)) %>%
ggplot(aes(x=CAUSA, y=DESC_VICT, color=DESC_VICT)) +
geom_point(aes(size=freq), alpha=0.7) +
scale_color_brewer(palette = "Dark2") +
scale_size_continuous(range = c(0.5,5)) +
theme(panel.grid.major = element_blank(),
      panel.grid.minor = element_blank()) +
coord_flip() +

labs(title = "Injury Severity By Cause Of Accident",
     subtitle = "-Over 2018 & 2019-",

     caption = "Circumference size is proportional to the severity frequency of each cause",
     x = "Cause Of Accident", y = "Injury Severity") +
theme(legend.position = "none")
```



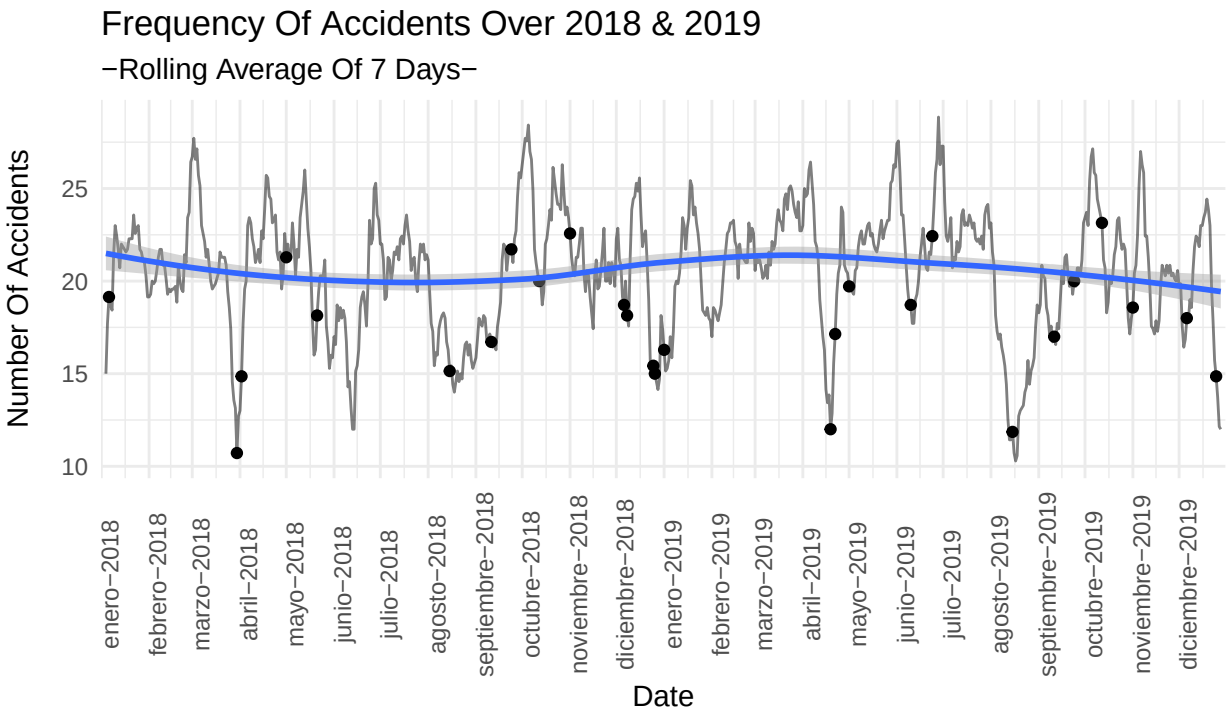
### 2.1.4. Frequency Of Accidents Over Time

Rolling average of 1 week through 2018 and 2019:

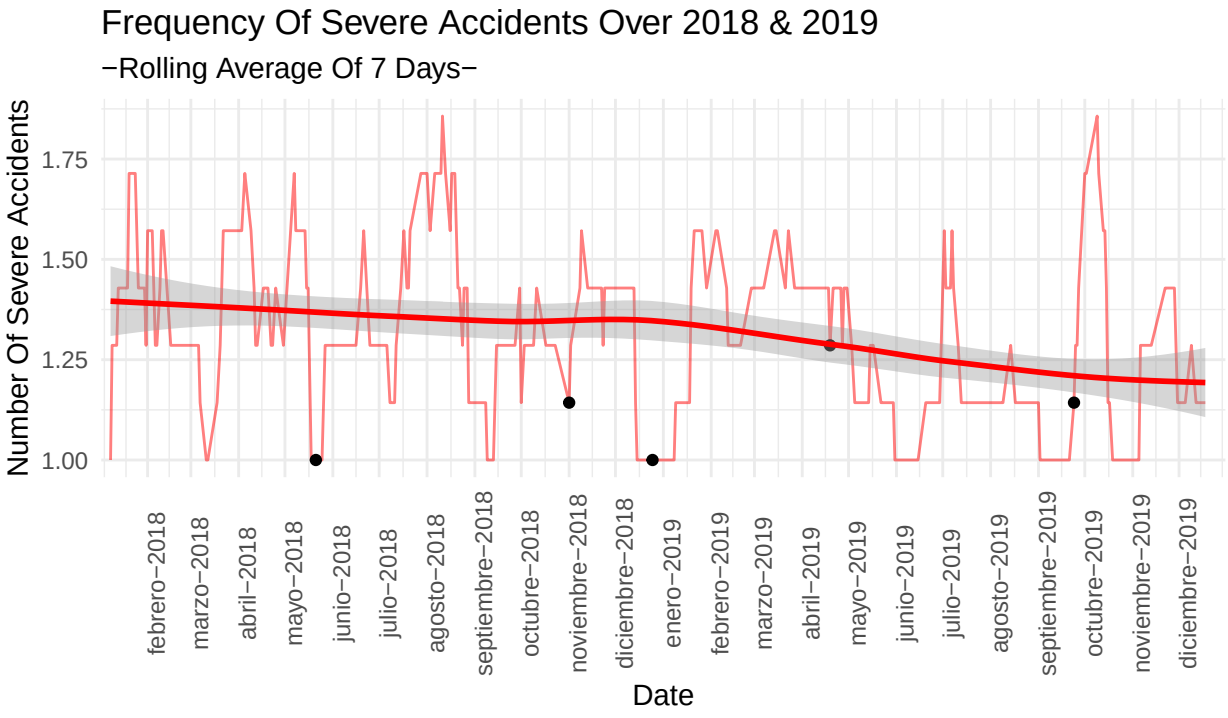
```
dfreq_07 = df %>% select(c("DATA", "ANY", "DIA_TIPUS", "NOM_MES")) %>%
    mutate(DIA_TIPUS = ifelse(DIA_TIPUS == "Holiday", "Holiday",
        "Normal")) %>% mutate(DATA = as.POSIXct(DATA)) %>% count(DATA,
    ANY, DIA_TIPUS, NOM_MES) %>% mutate(acc_07 = rollmean(n,
    k = 7, fill = NA))

dfreq_07_severe = df %>% filter(DESC_VICT %in% c("Severe", "Fatal")) %>%
    select(c("DATA", "ANY", "DIA_TIPUS", "NOM_MES")) %>% mutate(DIA_TIPUS = ifelse(DIA_TIPUS ==
    "Holiday", "Holiday", "Normal")) %>% mutate(DATA = as.POSIXct(DATA)) %>%
    count(DATA, ANY, DIA_TIPUS, NOM_MES) %>% mutate(acc_07 = rollmean(n,
    k = 7, fill = NA))
```

Plot temporal series:

```
freq_07_list = list()

freq_07_list[[1]] = ggplot(data = dfreq_07, aes(x = DATA, y = acc_07)) +
    geom_line(alpha = 0.5) + geom_point(data = filter(dfreq_07,
    DIA_TIPUS == "Holiday")) + geom_smooth(method = "loess") +
    scale_x_datetime(breaks = "1 month", date_labels = "%B-%Y",
        expand = c(0, 0)) + theme(axis.text.x = element_text(angle = 90)) +

labs(title = "Frequency Of Accidents Over 2018 & 2019", subtitle = "-Rolling Average Of 7 Days-",
    x = "Date", y = "Number Of Accidents", caption = "Holidays are marked as dots")

freq_07_list[[2]] = ggplot(data = dfreq_07_severe, aes(x = DATA,
    y = acc_07)) + geom_line(color = "red", alpha = 0.5) + geom_point(data = filter(dfreq_07_severe,
    DIA_TIPUS == "Holiday")) + geom_smooth(method = "loess",
    color = "red") + scale_x_datetime(breaks = "1 month", date_labels = "%B-%Y",
    expand = c(0, 0)) + theme(axis.text.x = element_text(angle = 90)) +

labs(title = "Frequency Of Severe Accidents Over 2018 & 2019",
    subtitle = "-Rolling Average Of 7 Days-", x = "Date", y = "Number Of Severe Accidents",
    caption = "Holidays are marked as dots")

ggarrange(plotlist = freq_07_list, ncol = 1, nrow = 2, align = "hv")
```




Holidays are marked as dots



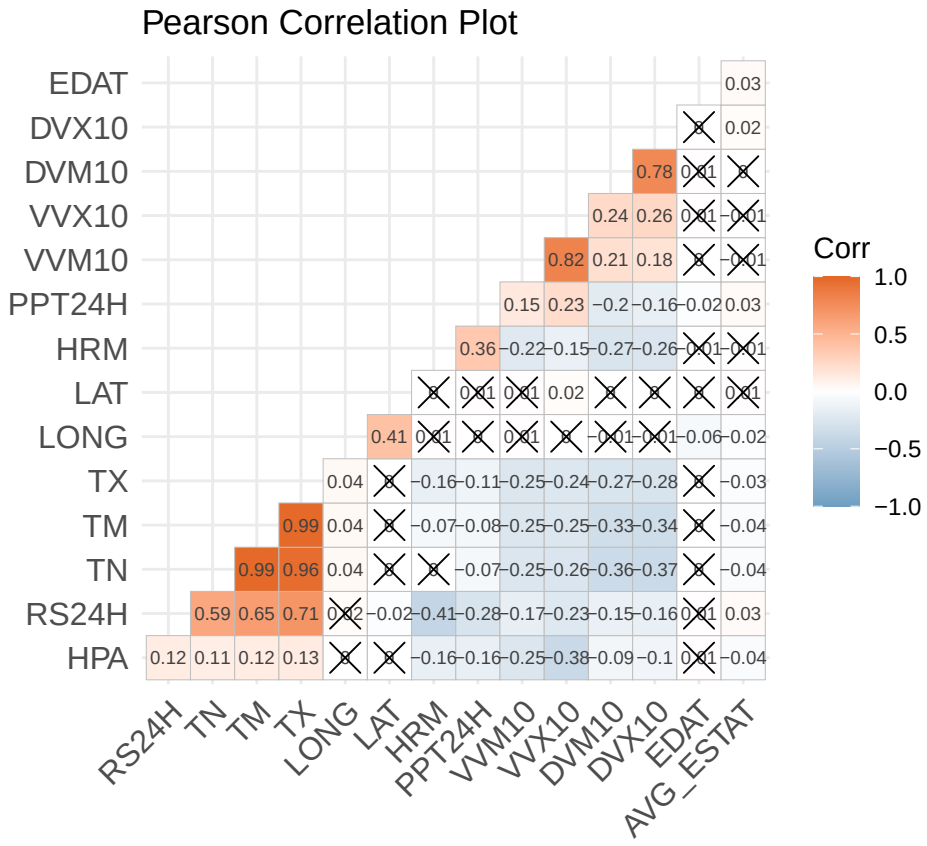

Holidays are marked as dots

## 2.2. Continuous Variables

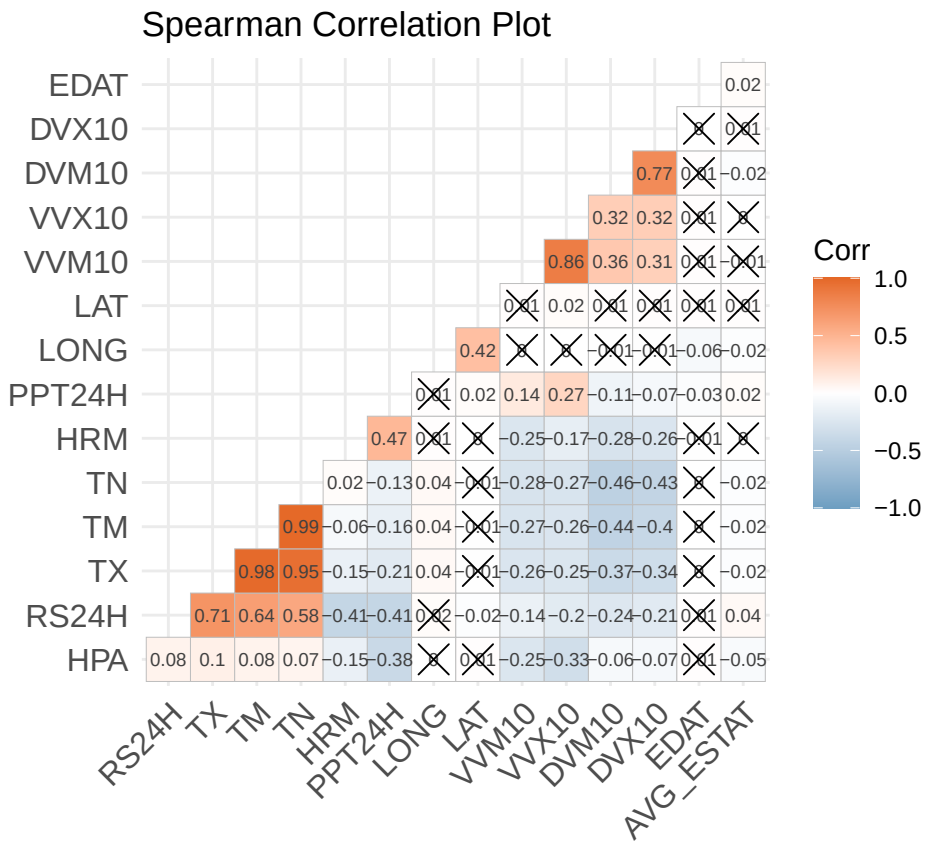
### 2.2.2. Correlation Plots

Pearson and Spearman correlation plots:

```
df_cor_num = df_numerical %>% keep(is.numeric)

pearson = rcorr(as.matrix(df_cor_num), type = "pearson")
spearman = rcorr(as.matrix(df_cor_num), type = "spearman")

ggcorrplot(pearson$r, hc.order = TRUE, type = "lower", colors = c("#6D9EC1",
    "white", "#E46726"), lab = T, lab_size = 2.5, lab_col = "grey25",
    p.mat = pearson$P) + labs(title = "Pearson Correlation Plot")
```



```
ggcorrplot(spearman$r, hc.order = TRUE, type = "lower", colors = c("#6D9EC1",
    "white", "#E46726"), lab = T, lab_size = 2.5, lab_col = "grey25",
    p.mat = spearman$P) + labs(title = "Spearman Correlation Plot")
```

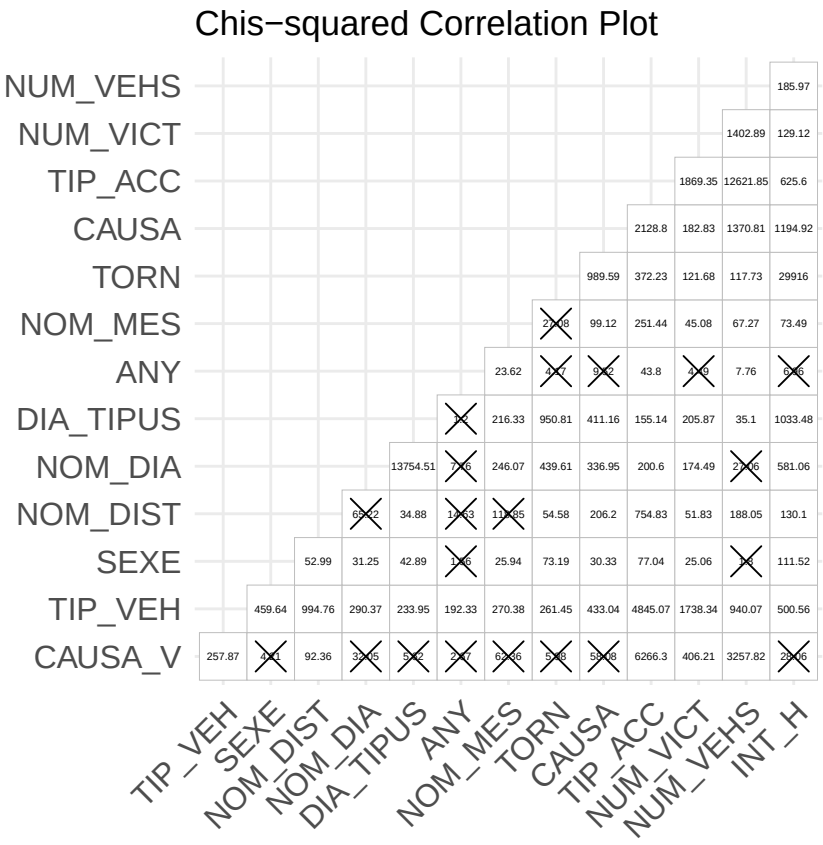


## 2.3. Categorical Variables

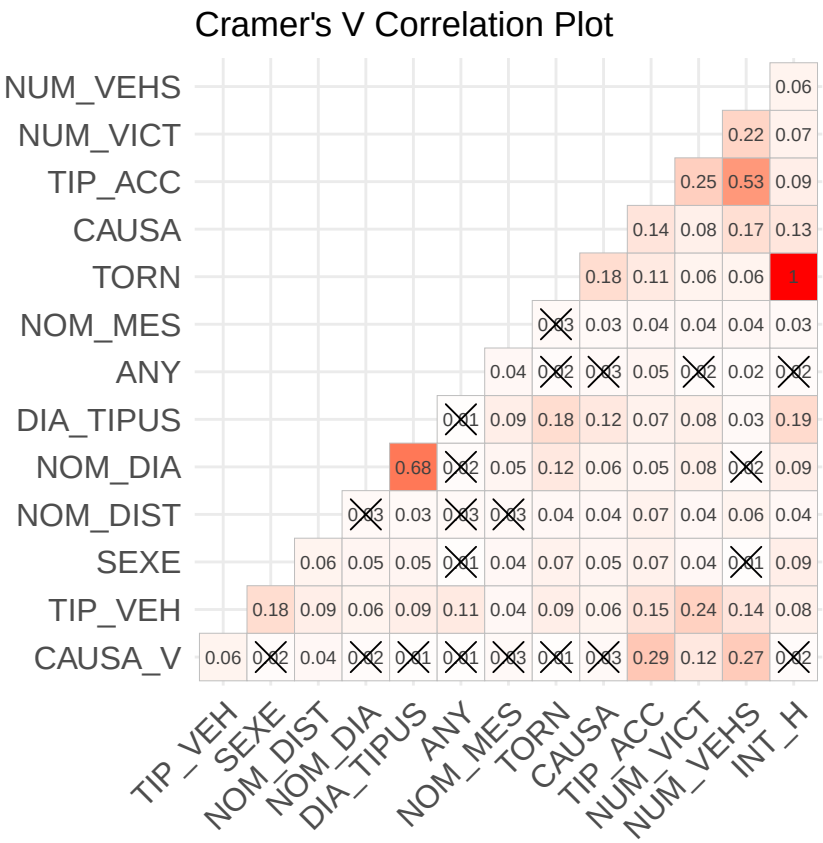
### 2.3.2. Correlation Plots

P-value with chi-squared test and cramer's V test:

```
chisqt_matrix = PairApply(select(df_categorical, -c("DESC_VICT")),
    function(x, y) chisq.test(x, y, simulate.p.value = T)$statistic,
    symmetric = T)

chisqt_p = PairApply(select(df_categorical, -c("DESC_VICT")),
    function(x, y) chisq.test(x, y, simulate.p.value = T)$p.value,
    symmetric = T)

cramersV_matrix = PairApply(select(df_categorical, -c("DESC_VICT")),
    CramerV, symmetric = TRUE)

ggcorrplot(chisqt_matrix, hc.order = F, type = "lower", colors = c("white",
    "white", "white"), lab = T, lab_size = 1.5, lab_col = "black",
    p.mat = chisqt_p, show.legend = F) + labs(title = "Chis-squared Correlation Plot") +
    scale_fill_gradient2(limit = c(0, 1e+100))
```

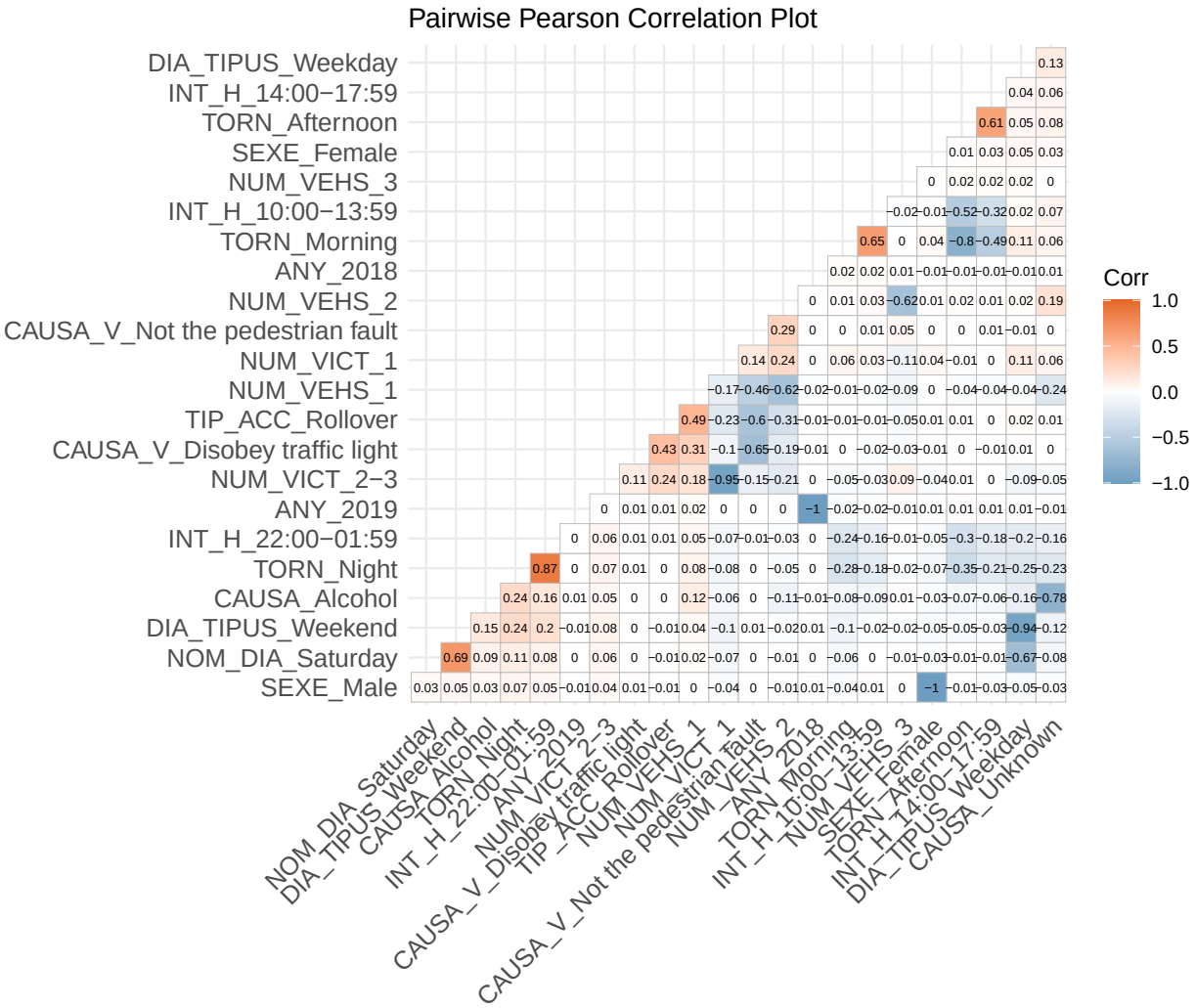


```
ggcorrplot(cramersV_matrix, hc.order = F, type = "lower", colors = c("white",
    "white", "red"), lab = T, lab_size = 2.5, lab_col = "grey25",
    show.legend = F, p.mat = chisqt_p) + labs(title = "Cramer's V Correlation Plot")
```



Pairwise for more detail:

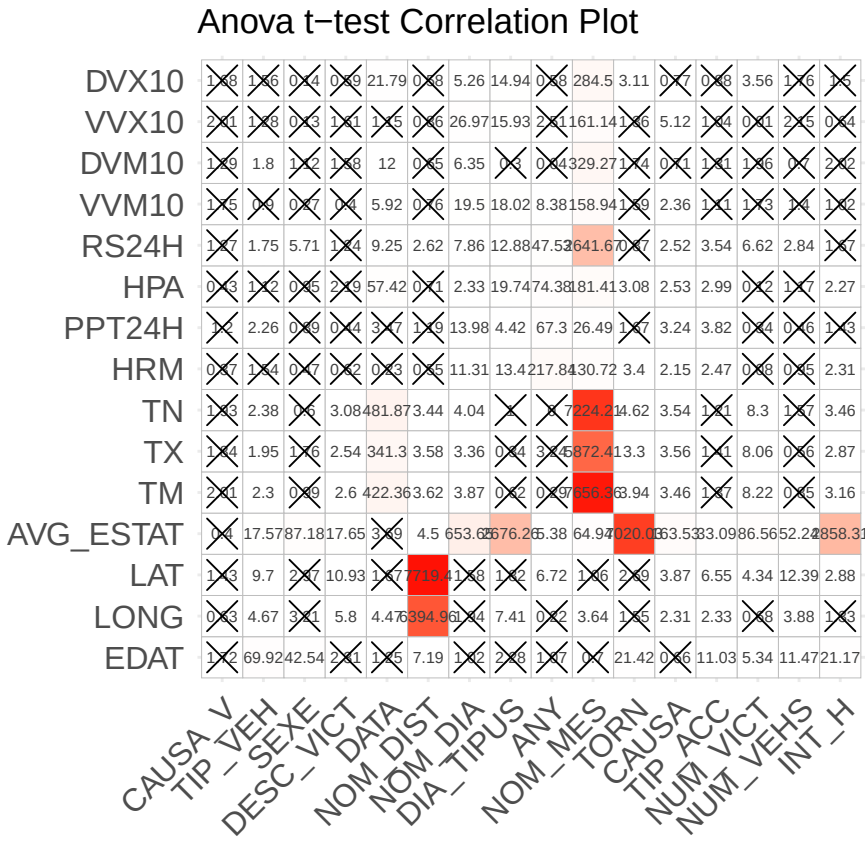
```
pairwise = df_categorical %>% select(-c("DESC_VICT")) %>% to_dummy(suffix = "label") %>%
    cor()
diag(pairwise) = 0

keep = colSums(abs(pairwise) > 0.6) > 0

ggcorrplot(pairwise[keep, keep], hc.order = TRUE, type = "lower",
    colors = c("#6D9EC1", "white", "#E46726"), lab = T, lab_size = 2) +
    labs(title = "Pairwise Pearson Correlation Plot")
```



## 2.4. Continuous And Categorical Variables

P-value with anova t-test:

```
aov_matrix_sc = PairApply(df, function(x, y) if (is.numeric(x)) {
    summary(aov(x ~ y))[[1]][["F value"]][[1]]
} else {
    if (is.numeric(y)) {
        summary(aov(y ~ x))[[1]][["F value"]][[1]]
    } else {
        1337
    }
}, symmetric = T)

aov_matrix_pv = PairApply(df, function(x, y) if (is.numeric(x)) {
    summary(aov(x ~ y))[[1]][["Pr(>F)"]][[1]]
} else {
    if (is.numeric(y)) {
        summary(aov(y ~ x))[[1]][["Pr(>F)"]][[1]]
    } else {
        1337
    }
}, symmetric = T)

remove = names(df %>% keep(is.numeric))
aov_matrix_sc = aov_matrix_sc[!rownames(aov_matrix_sc) %in% remove,
    ]
aov_matrix_sc = aov_matrix_sc[, remove]
aov_matrix_pv = aov_matrix_pv[!rownames(aov_matrix_pv) %in% remove,
    ]
aov_matrix_pv = aov_matrix_pv[, remove]

ggcorrplot(aov_matrix_sc, hc.order = F, colors = c("white", "white",
    "white"), lab = T, lab_size = 2, lab_col = "grey25", p.mat = aov_matrix_pv,
    show.legend = F) + labs(title = "Anova t-test Correlation Plot") +
    scale_fill_gradient2(limit = c(0, max(aov_matrix_sc) + 100),
        low = "white", high = "red", mid = "white")
```




Drop highly correlated variables and store dataset:

```
df = df %>% select(-c("TX", "TN", "VVX10", "DVX10", "TORN", "NOM_DIA",
    "LAT", "LONG", "DATA"))

df_categorical = df[, sapply(df, is.factor)]
df_numerical = df[, sapply(df, is.numeric) | names(df) %in% c("DESC_VICT")]

# write.csv(df, 'FinalDatasets/dataset_models.csv',
# row.names=F)
```

## 2.5. Contingency Tables

Final continuous variables:

```
matrix = describeBy(df_numerical ~ DESC_VICT, mat = T, digits = 3)
remove = c("DESC_VICT*1", "DESC_VICT*2", "DESC_VICT*3", "DESC_VICT*4",
    "DESC_VICT*5")
matrix[!rownames(matrix) %in% remove, ] %>% kable() %>% kable_styling(full_width = T,
    font_size = 15)
```

Final categorical variables:

```
table.control1 = tableby.control(test = T, total = F, cat.stats = c("countrowpct"),
    cat.test = "chisq")

cross.tab1 = tableby(DESC_VICT ~ ., data = df_categorical, control = table.control1)

cross.tab2 = tableby(~., data = select(df_categorical, -DESC_VICT))

s1 = summary(cross.tab1, text = T)
s2 = summary(cross.tab2, text = T)

d1 = as.data.frame(s1)
d2 = as.data.frame(s2)

dj = cbind(d1, select(d2, colnames(d2)[2]))
colnames(dj)[colnames(dj) == "Var.1"] <- "Variable"

kable(dj) %>% kable_styling(full_width = T, font_size = 15)
```

## 2.6. Group Levels From Categorical Features

Group low frequency categories:

```
df = read.csv("FinalDatasets/dataset_models.csv", stringsAsFactors = F)

order = c("No Injury", "Mild", "Moderate", "Severe", "Fatal")
df = df %>% mutate(DESC_VICT = ordered(DESC_VICT, levels = order))

Heavy = c("Rigid truck <= 3.5 tons", "Rigid truck > 3.5 tons",
    "Truck tractor", "Bus", "Articulated bus")

Other = c("Other non-motor vehicles", "Construction machinery",
    "Quadricycle < 75 cc", "Unknown", "Quadricycle > 75 cc")

RoadC = c("Road conditions", "Road obstacles", "Bad signaling")

df = df %>% mutate(TIP_VEH = case_when(TIP_VEH %in% Heavy ~ "Heavy",
    TIP_VEH %in% Other ~ "Other", TRUE ~ TIP_VEH)) %>% mutate(TIP_ACC = case_when(TIP_ACC %in%
    c("Unknown", "Others") ~ "Other", TRUE ~ TIP_ACC)) %>% mutate(CAUSA = case_when(CAUSA %in%
    c("Alcohol", "Drugs") ~ "Drugs", CAUSA %in% RoadC ~ "Road condition",
    TRUE ~ CAUSA))

# write.csv(df, 'FinalDatasets/dataset_models.csv',
# row.names=F)
```

# 3. Random Forests Workflow

Start counting time:

```
total_ti = Sys.time()
```

Import dataframe:

```
df = read.csv("FinalDatasets/dataset_models.csv")

order = c("No Injury", "Mild", "Moderate", "Severe", "Fatal")
df = df %>% mutate(DESC_VICT = ordered(DESC_VICT, levels = order))
```

## 3.1. 5-class Classifier

Split data:

```
set.seed(1998)

split = initial_split(df, strata = DESC_VICT, prob = 0.7)
train = training(split)
test = testing(split)

table(train$DESC_VICT)
table(test$DESC_VICT)
```

### 3.1.1. Model With Downsample

Recipe:

```
rec = recipe(DESC_VICT ~ ., data = train) %>% themis::step_downsample(DESC_VICT)

prep = prep(rec)
juiced = juice(prep)
```

Cross-validation folds:

```
set.seed(2000)

cv_folds = vfold_cv(train, v = 10, strata = DESC_VICT)
```

Tune hyperparameters:

```
rf_tune = rand_forest(mtry = tune(), trees = 1024, min_n = tune()) %>%
    set_engine("ranger") %>% set_mode("classification")

rf_tune_wf = workflow() %>% add_recipe(rec) %>% add_model(rf_tune)

ti = Sys.time()

doParallel::registerDoParallel()
set.seed(2004)

rf_tune_res = rf_tune_wf %>% tune_grid(resamples = cv_folds,
    grid = 20)

tf = Sys.time()
```

```
model_tune_t = tf - ti
model_tune_t
```

Pick best tune:

```
best_auc = select_best(rf_tune_res, "roc_auc")
```

Final model:

```
final_rf = finalize_model(rf_tune, best_auc)

final_rf = final_rf %>% set_engine("ranger", importance = "impurity")

rf_best = workflow() %>% add_recipe(rec) %>% add_model(final_rf) %>%
    last_fit(split)
```

Store model:

```
# save.image('models_nodum/RF_5_down.RData')
```

### 3.1.2. Model With Upsample And Downsample

Recipe:

```
rec = recipe(DESC_VICT ~ ., data = train) %>% themis::step_upsample(DESC_VICT,
    over_ratio = 0.1) %>% themis::step_downsample(DESC_VICT)

prep = prep(rec)
juiced = juice(prep)
```

Cross-validation folds:

```
set.seed(2000)

cv_folds = vfold_cv(train, v = 10, strata = DESC_VICT)
```

Tune hyperparameters:

```
rf_tune = rand_forest(mtry = tune(), trees = 1024, min_n = tune()) %>%
    set_engine("ranger") %>% set_mode("classification")

rf_tune_wf = workflow() %>% add_recipe(rec) %>% add_model(rf_tune)

ti = Sys.time()

doParallel::registerDoParallel()
set.seed(2004)

rf_tune_res = rf_tune_wf %>% tune_grid(resamples = cv_folds,
    grid = 20)

tf = Sys.time()
model_tune_t = tf - ti
model_tune_t
```

Best tune:

```
best_auc = select_best(rf_tune_res, "roc_auc")
```

Final model:

```
final_rf = finalize_model(rf_tune, best_auc)

final_rf = final_rf %>% set_engine("ranger", importance = "impurity")

rf_best = workflow() %>% add_recipe(rec) %>% add_model(final_rf) %>%
    last_fit(split)
```

Store model:

```
# save.image('models_nodum/RF_5_updown.RData')
```

## 3.2. 3-class Classifier 1

Group into 3 categories:

```
df = read.csv("FinalDatasets/dataset_models.csv")

order = c("No Injury", "Mild", "Moderate", "Severe", "Fatal")
df = df %>%
  mutate(DESC_VICT = ordered(DESC_VICT, levels = order))

df = df %>%
  mutate(DESC_VICT = as.character(DESC_VICT)) %>%
  mutate(DESC_VICT = case_when(

    DESC_VICT %in% c("No Injury", "Mild") ~ "Mild-",
                                DESC_VICT %in% c("Severe", "Fatal") ~ "Severe+",
                                TRUE ~ "Moderate")) %>%
  mutate(DESC_VICT = ordered(DESC_VICT, levels = c("Mild-", "Moderate", "Severe+")))
```

Split data:

```
set.seed(1998)

split = initial_split(df, strata = DESC_VICT, prob = 0.7)
train = training(split)
test = testing(split)

table(train$DESC_VICT)
table(test$DESC_VICT)
```

### 3.2.1. Model With Downsample

Recipe:

```
rec = recipe(DESC_VICT ~ ., data = train) %>% themis::step_downsample(DESC_VICT)

prep = prep(rec)
juiced = juice(prep)
```

Cross-validation:

```
set.seed(2000)

cv_folds = vfold_cv(train, v = 10, strata = DESC_VICT)
```

Tune hyperparameters:

```
rf_tune = rand_forest(mtry = tune(), trees = 1024, min_n = tune()) %>%
    set_engine("ranger") %>% set_mode("classification")

rf_tune_wf = workflow() %>% add_recipe(rec) %>% add_model(rf_tune)

ti = Sys.time()

doParallel::registerDoParallel()
set.seed(2004)

rf_tune_res = rf_tune_wf %>% tune_grid(resamples = cv_folds,
    grid = 20)

tf = Sys.time()
model_tune_t = tf - ti
model_tune_t
```

Best tune:

```
best_auc = select_best(rf_tune_res, "roc_auc")
```

Final model:

```
final_rf = finalize_model(rf_tune, best_auc)

final_rf = final_rf %>% set_engine("ranger", importance = "impurity")

rf_best = workflow() %>% add_recipe(rec) %>% add_model(final_rf) %>%
    last_fit(split)
```

Store model:

```
# save.image('models_nodum/RF_3_down.RData')
```

### 3.2.2. Model With Downsample And Upsample

Recipe:

```
rec = recipe(DESC_VICT ~ ., data = train) %>% themis::step_upsample(DESC_VICT,
    over_ratio = 0.1) %>% themis::step_downsample(DESC_VICT)

prep = prep(rec)
juiced = juice(prep)
```

Cross-validation:

```
set.seed(2000)

cv_folds = vfold_cv(train, v = 10, strata = DESC_VICT)
```

Tune hyperparameters:

```
rf_tune = rand_forest(mtry = tune(), trees = 1024, min_n = tune()) %>%
    set_engine("ranger") %>% set_mode("classification")

rf_tune_wf = workflow() %>% add_recipe(rec) %>% add_model(rf_tune)
```

```
ti = Sys.time()

doParallel::registerDoParallel()
set.seed(2004)

rf_tune_res = rf_tune_wf %>% tune_grid(resamples = cv_folds,
    grid = 20)

tf = Sys.time()
model_tune_t = tf - ti
model_tune_t
```

Best tune:

```
best_auc = select_best(rf_tune_res, "roc_auc")
```

Final model:

```
final_rf = finalize_model(rf_tune, best_auc)

final_rf = final_rf %>% set_engine("ranger", importance = "impurity")

rf_best = workflow() %>% add_recipe(rec) %>% add_model(final_rf) %>%
    last_fit(split)
```

Store model:

```
# save.image('models_nodum/RF_3_updown.RData')
```

### 3.3. 3-class Classifier 2

Group into 3 categories:

```
df = read.csv("FinalDatasets/dataset_models.csv")

order = c("No Injury", "Mild", "Moderate", "Severe", "Fatal")
df = df %>%
  mutate(DESC_VICT = ordered(DESC_VICT, levels = order))

df = df %>%
  mutate(DESC_VICT = as.character(DESC_VICT)) %>%
  mutate(DESC_VICT = case_when(

    DESC_VICT %in% c("No Injury", "Mild", "Moderate") ~ "Moderate-",
                                TRUE ~ DESC_VICT)) %>%
  mutate(DESC_VICT = ordered(DESC_VICT, levels = c("Moderate-", "Severe", "Fatal")))
```

Split data:

```
set.seed(1998)

split = initial_split(df, strata = DESC_VICT, prob = 0.7)
train = training(split)
test = testing(split)

table(train$DESC_VICT)
table(test$DESC_VICT)
```

### 3.3.1. Model With Downsample

Recipe:

```
rec = recipe(DESC_VICT ~ ., data = train) %>% themis::step_downsample(DESC_VICT)

prep = prep(rec)
juiced = juice(prep)
```

Cross-validation:

```
set.seed(2000)

cv_folds = vfold_cv(train, v = 10, strata = DESC_VICT)
```

Tune hyperparameters:

```
rf_tune = rand_forest(mtry = tune(), trees = 1024, min_n = tune()) %>%
    set_engine("ranger") %>% set_mode("classification")

rf_tune_wf = workflow() %>% add_recipe(rec) %>% add_model(rf_tune)

ti = Sys.time()

doParallel::registerDoParallel()
set.seed(2004)

rf_tune_res = rf_tune_wf %>% tune_grid(resamples = cv_folds,
    grid = 20)

tf = Sys.time()
model_tune_t = tf - ti
model_tune_t
```

Best tune:

```
best_auc = select_best(rf_tune_res, "roc_auc")
```

Final model:

```
final_rf = finalize_model(rf_tune, best_auc)

final_rf = final_rf %>% set_engine("ranger", importance = "impurity")

rf_best = workflow() %>% add_recipe(rec) %>% add_model(final_rf) %>%
    last_fit(split)
```

Store model:

```
# save.image('models_nodum/RF_3_down2.RData')
```

### 3.3.2. Model With Downsample And Upsample

Recipe:

```
rec = recipe(DESC_VICT ~ ., data = train) %>% themis::step_upsample(DESC_VICT,
    over_ratio = 0.1) %>% themis::step_downsample(DESC_VICT)

prep = prep(rec)
```

```
juiced = juice(prep)
```

Recipe:

```
set.seed(2000)

cv_folds = vfold_cv(train, v = 10, strata = DESC_VICT)
```

Tune hyperparameters:

```
rf_tune = rand_forest(mtry = tune(), trees = 1024, min_n = tune()) %>%
    set_engine("ranger") %>% set_mode("classification")

rf_tune_wf = workflow() %>% add_recipe(rec) %>% add_model(rf_tune)

ti = Sys.time()

doParallel::registerDoParallel()
set.seed(2004)

rf_tune_res = rf_tune_wf %>% tune_grid(resamples = cv_folds,
    grid = 20)

tf = Sys.time()
model_tune_t = tf - ti
model_tune_t
```

Best tune:

```
best_auc = select_best(rf_tune_res, "roc_auc")
```

Final model:

```
final_rf = finalize_model(rf_tune, best_auc)

final_rf = final_rf %>% set_engine("ranger", importance = "impurity")

rf_best = workflow() %>% add_recipe(rec) %>% add_model(final_rf) %>%
    last_fit(split)
```

Store model:

```
# save.image('models_nodum/RF_3_updown2.RData')
```

### 3.4. Binary Classifier 1

Group into 2 categories:

```
df = read.csv("FinalDatasets/dataset_models.csv")

order = c("No Injury", "Mild", "Moderate", "Severe", "Fatal")
df = df %>%
  mutate(DESC_VICT = ordered(DESC_VICT, levels = order))

df = df %>%
  mutate(DESC_VICT = as.character(DESC_VICT)) %>%
  mutate(DESC_VICT = case_when(
```

```
    DESC_VICT %in% c("No Injury", "Mild", "Moderate") ~ "Moderate-",
                                  TRUE ~ "Severe+")) %>%
  mutate(DESC_VICT = ordered(DESC_VICT, levels = c("Moderate-", "Severe+")))
```

Split data:

```
set.seed(1998)

split = initial_split(df, strata = DESC_VICT, prob = 0.7)
train = training(split)
test = testing(split)

table(train$DESC_VICT)
table(test$DESC_VICT)
```

### 3.4.1. Model With Downsample

Recipe:

```
rec = recipe(DESC_VICT ~ ., data = train) %>% step_dummy(contains("CAUSA")) %>%
    themis::step_downsample(DESC_VICT)

prep = prep(rec)
juiced = juice(prep)
colnames(juiced)
```

Cross-validation:

```
set.seed(2000)

cv_folds = vfold_cv(train, v = 10, strata = DESC_VICT)
```

Tune hyperparameters:

```
rf_tune = rand_forest(mtry = tune(), trees = 1024, min_n = tune()) %>%
    set_engine("ranger") %>% set_mode("classification")

rf_tune_wf = workflow() %>% add_recipe(rec) %>% add_model(rf_tune)

ti = Sys.time()

doParallel::registerDoParallel()
set.seed(2004)

rf_tune_res = rf_tune_wf %>% tune_grid(resamples = cv_folds,
    grid = 20)

tf = Sys.time()
model_tune_t = tf - ti
model_tune_t
```

Best tune:

```
best_auc = select_best(rf_tune_res, "roc_auc")
```

Final model:

```
final_rf = finalize_model(rf_tune, best_auc)

final_rf = final_rf %>% set_engine("ranger", importance = "impurity")

rf_best = workflow() %>% add_recipe(rec) %>% add_model(final_rf) %>%
    last_fit(split)
```

Store model:

```
# save.image('models_nodum/RF_2_down.RData')
```

### 3.4.2. Model With Downsample And Upsample

Recipe:

```
rec = recipe(DESC_VICT ~ ., data = train) %>% themis::step_upsample(DESC_VICT,
    over_ratio = 0.1) %>% themis::step_downsample(DESC_VICT)

prep = prep(rec)
juiced = juice(prep)
```

Cross-validation:

```
set.seed(2000)

cv_folds = vfold_cv(train, v = 10, strata = DESC_VICT)
```

Tune hyperparameters:

```
rf_tune = rand_forest(mtry = tune(), trees = 1024, min_n = tune()) %>%
    set_engine("ranger") %>% set_mode("classification")

rf_tune_wf = workflow() %>% add_recipe(rec) %>% add_model(rf_tune)

ti = Sys.time()

doParallel::registerDoParallel()
set.seed(2004)

rf_tune_res = rf_tune_wf %>% tune_grid(resamples = cv_folds,
    grid = 20)

tf = Sys.time()
model_tune_t = tf - ti
model_tune_t
```

Best tune:

```
best_auc = select_best(rf_tune_res, "roc_auc")
```

Final model:

```
final_rf = finalize_model(rf_tune, best_auc)

final_rf = final_rf %>% set_engine("ranger", importance = "impurity")

rf_best = workflow() %>% add_recipe(rec) %>% add_model(final_rf) %>%
    last_fit(split)
```

Store model:

```
# save.image('models_nodum/RF_2_updown.RData')
```

## 3.5. Binary Classifier 2

Group into 2 categories:

```
df = read.csv("FinalDatasets/dataset_models.csv")

order = c("No Injury", "Mild", "Moderate", "Severe", "Fatal")
df = df %>%
  mutate(DESC_VICT = ordered(DESC_VICT, levels = order))

df = df %>%
  mutate(DESC_VICT = as.character(DESC_VICT)) %>%
  mutate(DESC_VICT = case_when(

    DESC_VICT %in% c("No Injury", "Mild") ~ "Mild-",
                                TRUE ~ "Moderate+")) %>%
  mutate(DESC_VICT = ordered(DESC_VICT, levels = c("Mild-", "Moderate+")))
```

Split data:

```
set.seed(1998)

split = initial_split(df, strata = DESC_VICT, prob = 0.7)
train = training(split)
test = testing(split)

table(train$DESC_VICT)
table(test$DESC_VICT)
```

### 3.5.1. Model With Downsample

Recipe:

```
rec = recipe(DESC_VICT ~ ., data = train) %>% themis::step_downsample(DESC_VICT)

prep = prep(rec)
juiced = juice(prep)
```

Cross-validation:

```
set.seed(2000)

cv_folds = vfold_cv(train, v = 10, strata = DESC_VICT)
```

Tune hyperparameters:

```
rf_tune = rand_forest(mtry = tune(), trees = 1024, min_n = tune()) %>%
    set_engine("ranger") %>% set_mode("classification")

rf_tune_wf = workflow() %>% add_recipe(rec) %>% add_model(rf_tune)

ti = Sys.time()

doParallel::registerDoParallel()
```

```
set.seed(2004)

rf_tune_res = rf_tune_wf %>% tune_grid(resamples = cv_folds,
    grid = 20)

tf = Sys.time()
model_tune_t = tf - ti
model_tune_t
```

Best tune:

```
best_auc = select_best(rf_tune_res, "roc_auc")
```

Final model:

```
final_rf = finalize_model(rf_tune, best_auc)

final_rf = final_rf %>% set_engine("ranger", importance = "impurity")

rf_best = workflow() %>% add_recipe(rec) %>% add_model(final_rf) %>%
    last_fit(split)
```

Store model:

```
# save.image('models_nodum/RF_2_down2.RData')
```

### 3.5.2. Model With Downsample And Upsample

Recipe:

```
rec = recipe(DESC_VICT ~ ., data = train) %>% themis::step_upsample(DESC_VICT,
    over_ratio = 0.1) %>% themis::step_downsample(DESC_VICT)

prep = prep(rec)
juiced = juice(prep)
```

Cross-validation:

```
set.seed(2000)

cv_folds = vfold_cv(train, v = 10, strata = DESC_VICT)
```

Tune hyperparameters:

```
rf_tune = rand_forest(mtry = tune(), trees = 1024, min_n = tune()) %>%
    set_engine("ranger") %>% set_mode("classification")

rf_tune_wf = workflow() %>% add_recipe(rec) %>% add_model(rf_tune)

ti = Sys.time()

doParallel::registerDoParallel()
set.seed(2004)

rf_tune_res = rf_tune_wf %>% tune_grid(resamples = cv_folds,
    grid = 20)

tf = Sys.time()
```

```
model_tune_t = tf - ti
model_tune_t
```

Best tune:

```
best_auc = select_best(rf_tune_res, "roc_auc")
```

Final model:

```
final_rf = finalize_model(rf_tune, best_auc)

final_rf = final_rf %>% set_engine("ranger", importance = "impurity")

rf_best = workflow() %>% add_recipe(rec) %>% add_model(final_rf) %>%
    last_fit(split)
```

Store model:

```
# save.image('models_nodum/RF_2_updown2.RData')
```

Store time:

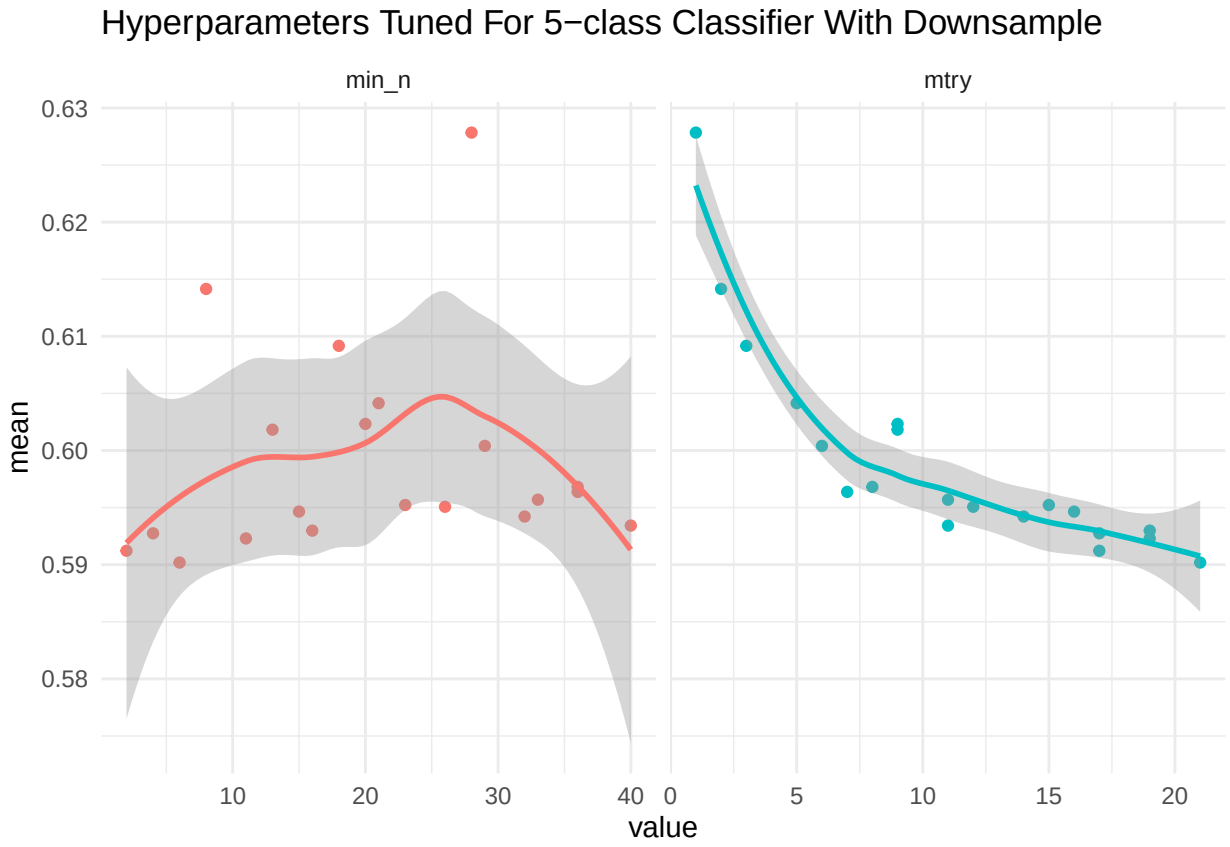
```
total_tf = Sys.time()
total_t = total_tf - total_ti
total_t
```

# 4. Random Forests Results

## 4.1. 5-class Classi

### 4.1.1. Downsample

```
load("models_nodum/RF_5_down.RData")
title = "Hyperparameters Tuned For 5-class Classifier With Downsample"

rf_tune_res %>% collect_metrics() %>% filter(.metric == "roc_auc") %>%
    select(mean, min_n, mtry) %>% pivot_longer(min_n:mtry, values_to = "value",
    names_to = "parameter") %>% ggplot(aes(value, mean, color = parameter)) +
    geom_point(show.legend = F) + geom_smooth(method = "loess",
    show.legend = F) + facet_wrap(~parameter, scales = "free_x") +
    labs(title = title)
```




```
conf = rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf$table %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | No Injury | Mild | Moderate | Severe | Fatal |
|---|---|---|---|---|---|
| No Injury | 22 | 198 | 502 | 5 | 0 |
| Mild | 14 | 237 | 629 | 22 | 2 |
| Moderate | 29 | 335 | 894 | 18 | 0 |
| Severe | 10 | 71 | 226 | 15 | 0 |
| Fatal | 13 | 132 | 327 | 29 | 8 |

```
rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT, .pred_class) %>%
    summary() %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | multiclass | 0.3146067 |
| kap | multiclass | 0.0132956 |
| sens | macro | 0.3617793 |
| spec | macro | 0.8035577 |
| ppv | macro | 0.2110715 |
| npv | macro | 0.8024196 |
| mcc | multiclass | 0.0156849 |
| j_index | macro | 0.1653369 |
| bal_accuracy | macro | 0.5826685 |
| detection_prevalence | macro | 0.2000000 |
| precision | macro | 0.2110715 |
| recall | macro | 0.3617793 |
| f_meas | macro | 0.1748546 |

```
rf_best %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

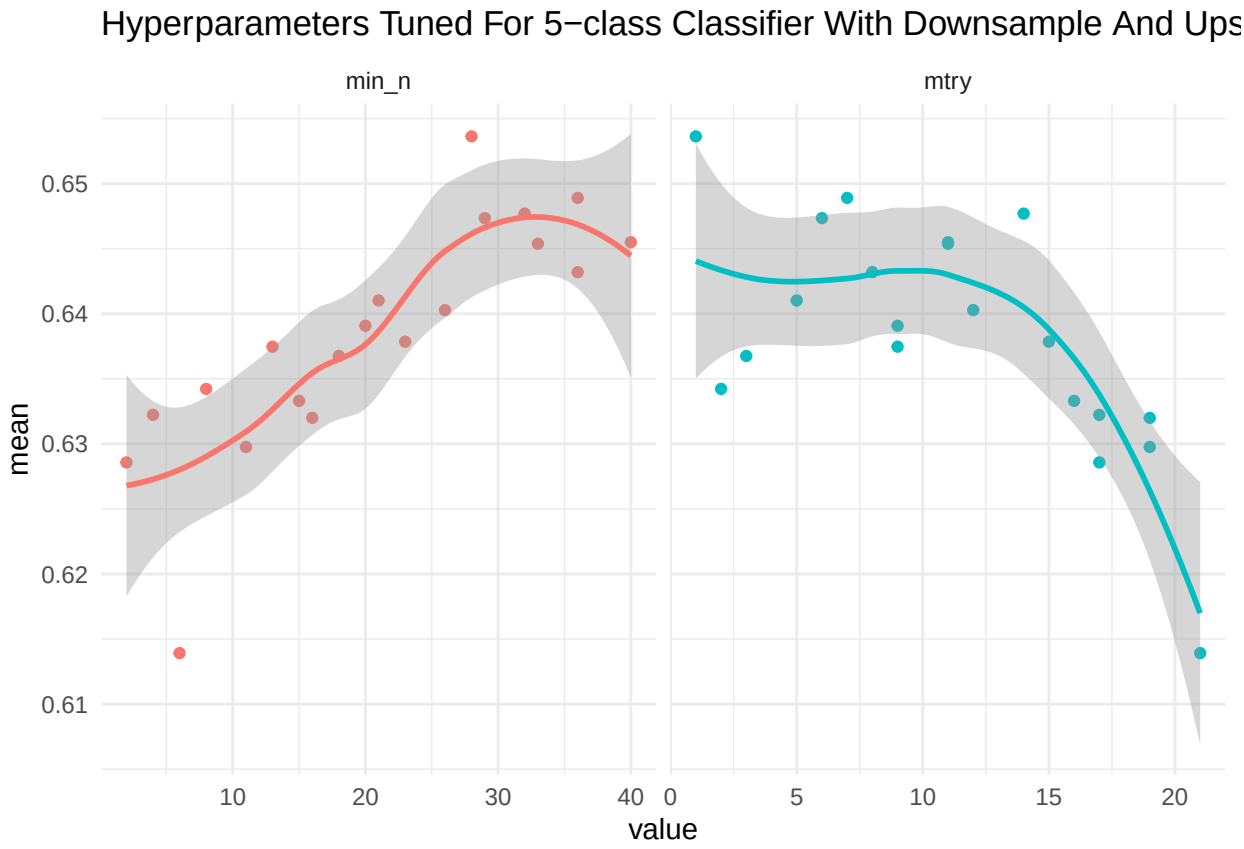
| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | multiclass | 0.3146067 | Preprocessor1_Model1 |
| roc_auc | hand_till | 0.6396666 | Preprocessor1_Model1 |

```
model = pull_workflow_fit(rf_best$.workflow[[1]])
paste(model$fit$prediction.error) %>% kable(col.names = "OOB Error") %>%
    kable_styling(full_width = F, font_size = 15)
```

| OOB Error |
|---|
| 0.620820652285388 |

#### 4.1.2. Updownsample

```
load("models_nodum/RF_5_updown.RData")
title = "Hyperparameters Tuned For 5-class Classifier With Downsample And Upsample"

rf_tune_res %>% collect_metrics() %>% filter(.metric == "roc_auc") %>%
    select(mean, min_n, mtry) %>% pivot_longer(min_n:mtry, values_to = "value",
    names_to = "parameter") %>% ggplot(aes(value, mean, color = parameter)) +
    geom_point(show.legend = F) + geom_smooth(method = "loess",
    show.legend = F) + facet_wrap(~parameter, scales = "free_x") +
    labs(title = title)
```




```
conf = rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf$table %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | No Injury | Mild | Moderate | Severe | Fatal |
|---|---|---|---|---|---|
| No Injury | 17 | 118 | 279 | 4 | 1 |
| Mild | 39 | 379 | 959 | 22 | 2 |
| Moderate | 22 | 359 | 999 | 25 | 2 |
| Severe | 9 | 113 | 320 | 35 | 3 |
| Fatal | 1 | 4 | 21 | 3 | 2 |

```
rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT, .pred_class) %>%
    summary() %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | multiclass | 0.3830926 |
| kap | multiclass | 0.0317327 |
| sens | macro | 0.3126934 |
| spec | macro | 0.8077577 |
| ppv | macro | 0.2317096 |
| npv | macro | 0.8056623 |
| mcc | multiclass | 0.0360979 |
| j_index | macro | 0.1204511 |
| bal_accuracy | macro | 0.5602255 |
| detection_prevalence | macro | 0.2000000 |
| precision | macro | 0.2317096 |
| recall | macro | 0.3126934 |
| f_meas | macro | 0.2216635 |

```
rf_best %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

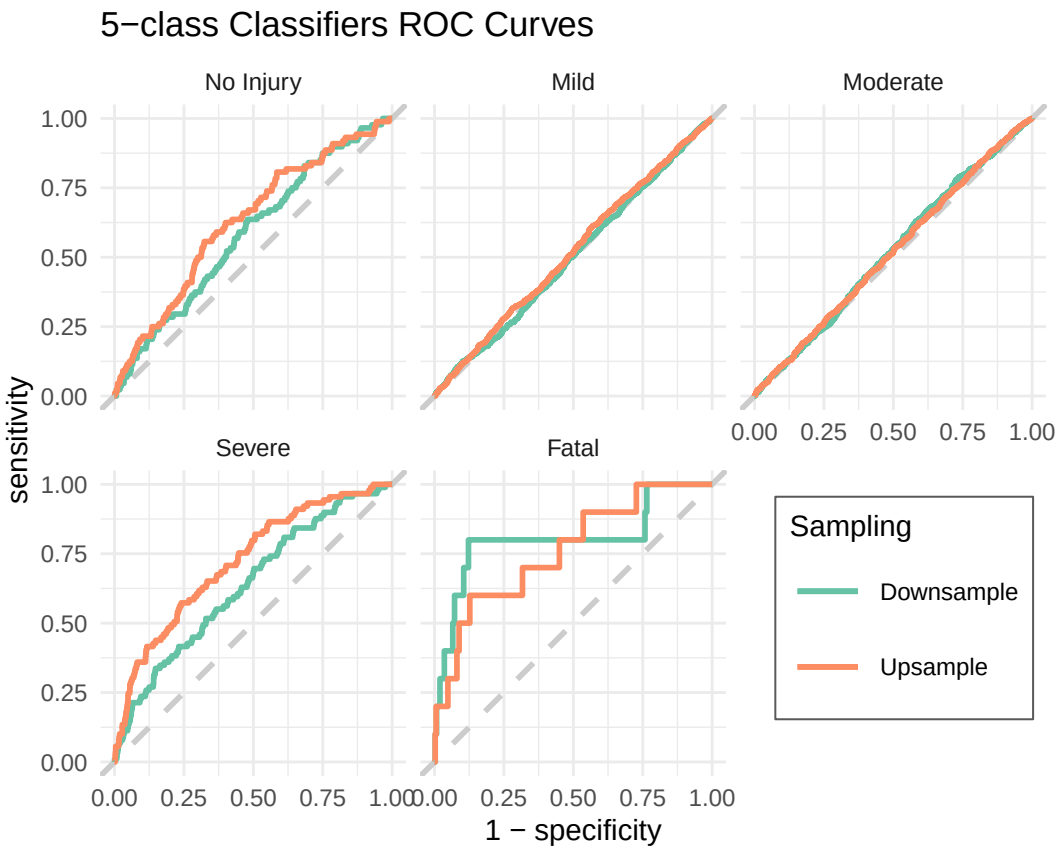
| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | multiclass | 0.3830926 | Preprocessor1_Model1 |
| roc_auc | hand_till | 0.6665326 | Preprocessor1_Model1 |

```
model = pull_workflow_fit(rf_best$.workflow[[1]])
paste(model$fit$prediction.error) %>% kable(col.names = "OOB Error") %>%
    kable_styling(full_width = F, font_size = 15)
```

| OOB Error |
|---|
| 0.381463384647363 |

#### 4.1.3. ROC For 5 Class Models

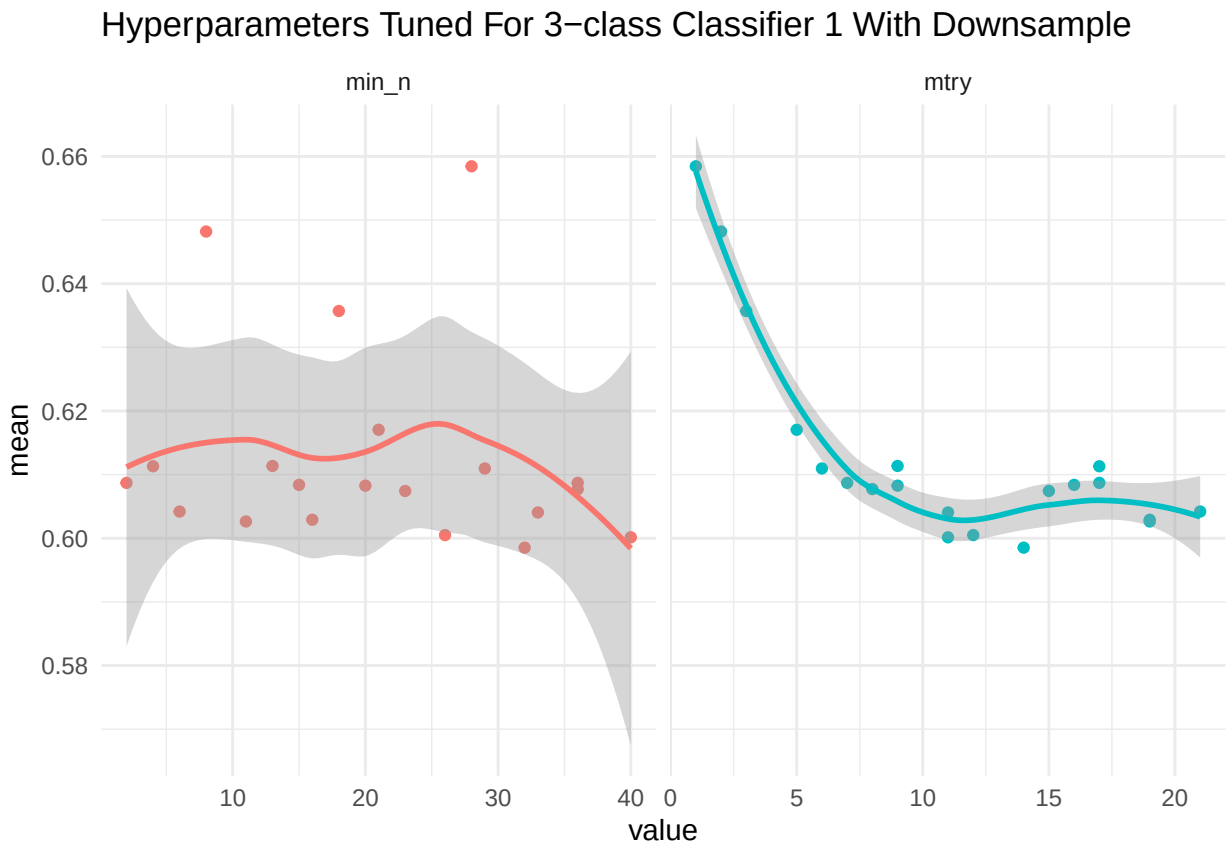
```
roc_5 = ggplot() + geom_abline(lty = 2, color = "gray80", size = 1)

load("models_nodum/RF_5_down.RData")
sens_data = rf_best %>% collect_predictions() %>% roc_curve(DESC_VICT,
    `.pred_No Injury`, .pred_Mild, .pred_Moderate, .pred_Severe,
    .pred_Fatal) %>% mutate(.level = ordered(.level, levels = c("No Injury",
    "Mild", "Moderate", "Severe", "Fatal")))
roc_5 = roc_5 + geom_path(sens_data, mapping = aes(1 - specificity,
    sensitivity, color = "Downsample"), size = 1)

load("models_nodum/RF_5_updown.RData")
sens_data = rf_best %>% collect_predictions() %>% roc_curve(DESC_VICT,
    `.pred_No Injury`, .pred_Mild, .pred_Moderate, .pred_Severe,
    .pred_Fatal) %>% mutate(.level = ordered(.level, levels = c("No Injury",
    "Mild", "Moderate", "Severe", "Fatal")))
roc_5 = roc_5 + geom_path(sens_data, mapping = aes(1 - specificity,
    sensitivity, color = "Upsample"), size = 1)

colors <- c(Downsample = "#66C2A5", Upsample = "#FC8D62", Smote = "#8DA0CB")
roc_5 + facet_wrap(~.level) + coord_equal() + scale_color_manual(values = colors) +
    labs(title = "5-class Classifiers ROC Curves", color = "Sampling") +
    theme(legend.position = c(0.85, 0.25), legend.key.size = unit(1,
        "cm"), legend.box.background = element_rect(color = "grey35"))
```



## 4.2. 3-class Classifier 1

### 4.2.1. Downsample

```
load("models_nodum/RF_3_down2.RData")
title = "Hyperparameters Tuned For 3-class Classifier 1 With Downsample"

rf_tune_res %>% collect_metrics() %>% filter(.metric == "roc_auc") %>%
    select(mean, min_n, mtry) %>% pivot_longer(min_n:mtry, values_to = "value",
    names_to = "parameter") %>% ggplot(aes(value, mean, color = parameter)) +
    geom_point(show.legend = F) + geom_smooth(method = "loess",
    show.legend = F) + facet_wrap(~parameter, scales = "free_x") +
    labs(title = title)
```




```
conf = rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf$table %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | Moderate- | Severe | Fatal |
|---|---|---|---|
| Moderate- | 1876 | 29 | 1 |
| Severe | 1073 | 25 | 1 |
| Fatal | 701 | 22 | 11 |

```
rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT, .pred_class) %>%
    summary() %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | multiclass | 0.5113667 |
| kap | multiclass | 0.0142857 |
| sens | macro | 0.5630246 |
| spec | macro | 0.7252257 |
| ppv | macro | 0.3406649 |
| npv | macro | 0.6707346 |
| mcc | multiclass | 0.0418177 |
| j_index | macro | 0.2882503 |
| bal_accuracy | macro | 0.6441252 |
| detection_prevalence | macro | 0.3333333 |
| precision | macro | 0.3406649 |
| recall | macro | 0.5630246 |
| f_meas | macro | 0.2491034 |

```
rf_best %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

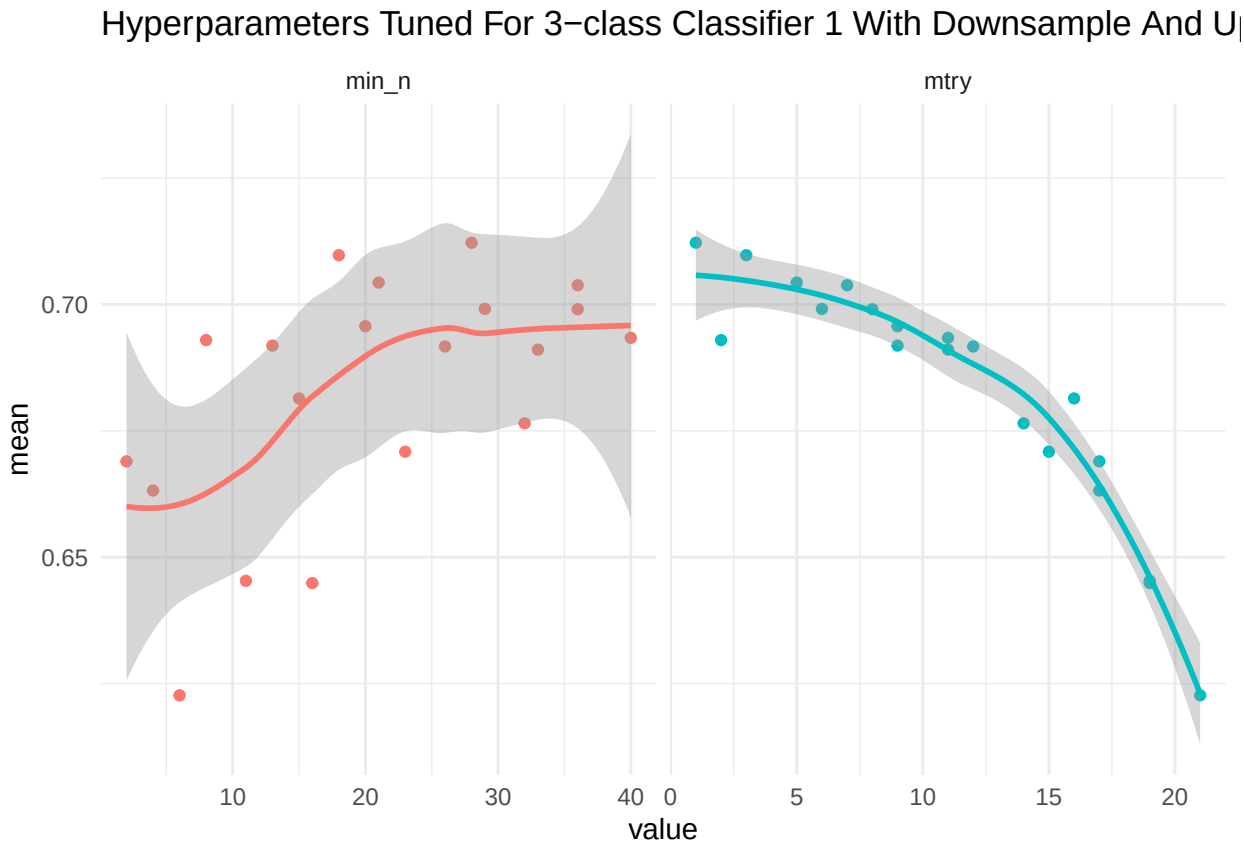
| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | multiclass | 0.5113667 | Preprocessor1_Model1 |
| roc_auc | hand_till | 0.7133752 | Preprocessor1_Model1 |

```
model = pull_workflow_fit(rf_best$.workflow[[1]])
paste(model$fit$prediction.error) %>% kable(col.names = "OOB Error") %>%
    kable_styling(full_width = F, font_size = 15)
```

| OOB Error |
|---|
| 0.441629309709482 |

#### 4.2.2. Updownsample

```
load("models_nodum/RF_3_updown2.RData")
title = "Hyperparameters Tuned For 3-class Classifier 1 With Downsample And Upsample"

rf_tune_res %>% collect_metrics() %>% filter(.metric == "roc_auc") %>%
    select(mean, min_n, mtry) %>% pivot_longer(min_n:mtry, values_to = "value",
    names_to = "parameter") %>% ggplot(aes(value, mean, color = parameter)) +
    geom_point(show.legend = F) + geom_smooth(method = "loess",
    show.legend = F) + facet_wrap(~parameter, scales = "free_x") +
    labs(title = title)
```




```
conf = rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf$table %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | Moderate- | Severe | Fatal |
|---|---|---|---|
| Moderate- | 3315 | 55 | 9 |
| Severe | 328 | 21 | 4 |
| Fatal | 7 | 0 | 0 |

```
rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT, .pred_class) %>%
    summary() %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | multiclass | 0.8922172 |
| kap | multiclass | 0.0697811 |
| sens | macro | 0.3948450 |
| spec | macro | 0.7294614 |
| ppv | macro | 0.3468499 |
| npv | macro | 0.6832392 |
| mcc | multiclass | 0.0896801 |
| j_index | macro | 0.1243064 |
| bal_accuracy | macro | 0.5621532 |
| detection_prevalence | macro | 0.3333333 |
| precision | macro | 0.3468499 |
| recall | macro | 0.3948450 |
| f_meas | macro | 0.3470458 |

```
rf_best %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

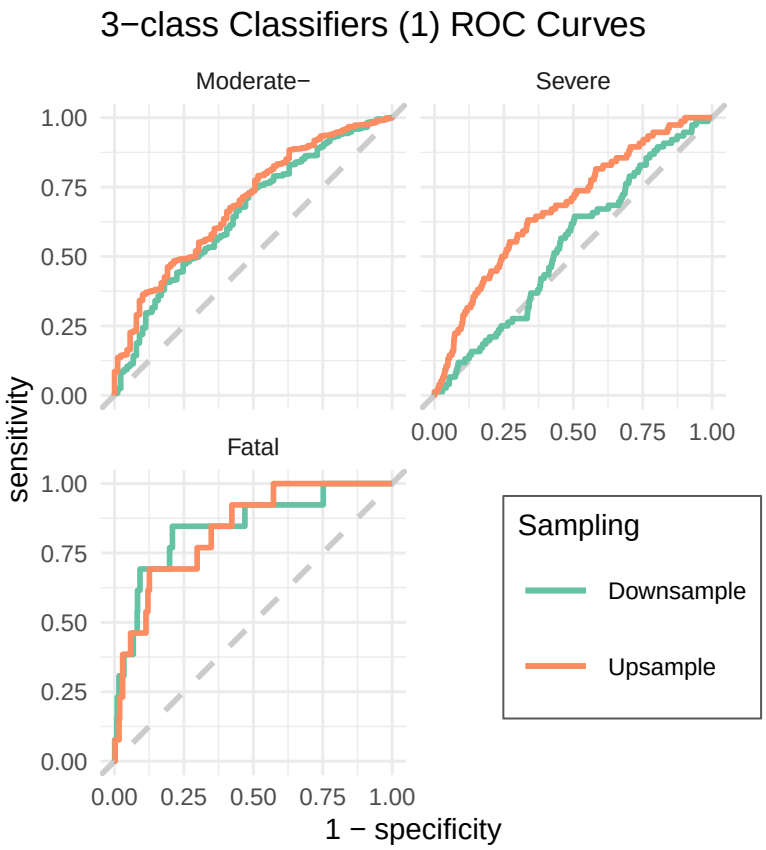
| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | multiclass | 0.8922172 | Preprocessor1_Model1 |
| roc_auc | hand_till | 0.7195081 | Preprocessor1_Model1 |

```
model = pull_workflow_fit(rf_best$.workflow[[1]])
paste(model$fit$prediction.error) %>% kable(col.names = "OOB Error") %>%
    kable_styling(full_width = F, font_size = 15)
```

| OOB Error |
|---|
| 0.134835215363717 |

#### 4.2.3. ROC For 3 Class Models

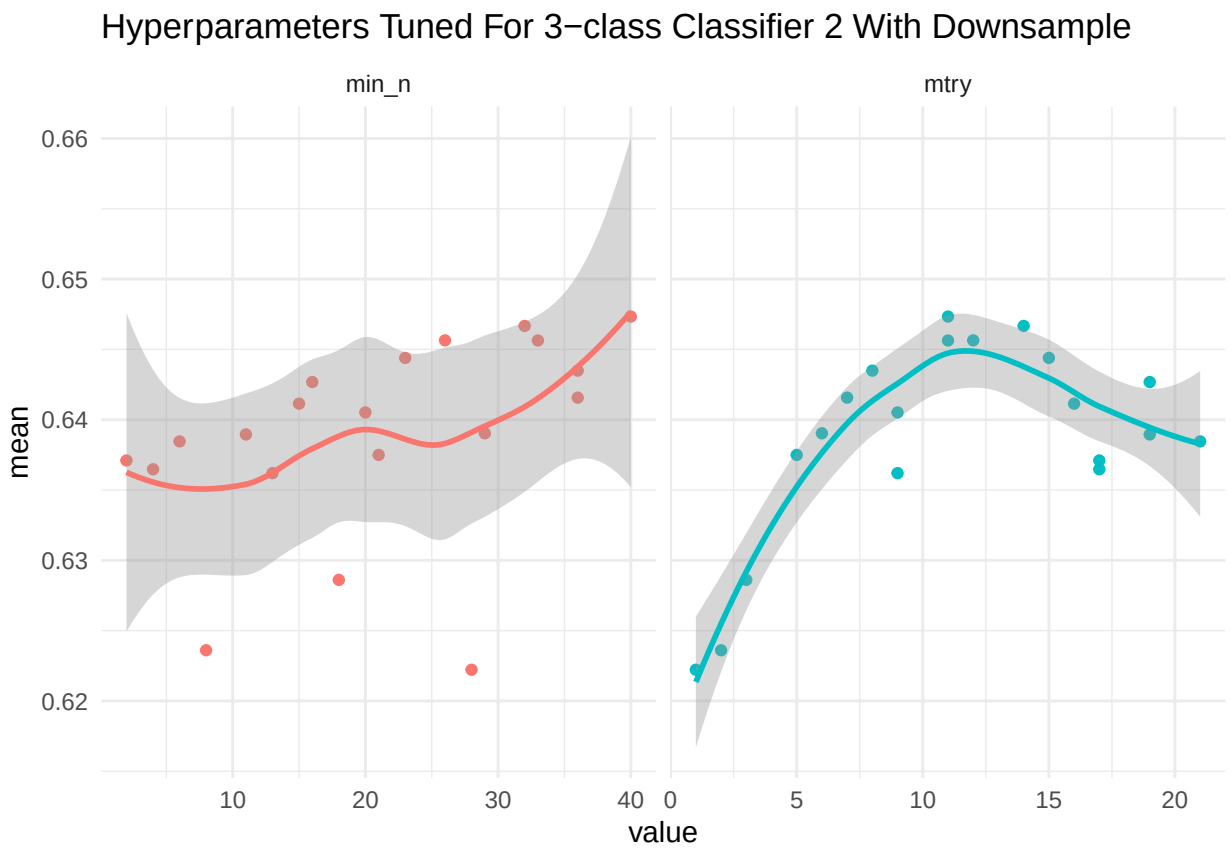
```
roc_5 = ggplot() + geom_abline(lty = 2, color = "gray80", size = 1)

load("models_nodum/RF_3_down2.RData")
sens_data = rf_best %>% collect_predictions() %>% roc_curve(DESC_VICT,
    `.pred_Moderate-`, .pred_Severe, .pred_Fatal) %>% mutate(.level = ordered(.level,
    levels = c("Moderate-", "Severe", "Fatal")))
roc_5 = roc_5 + geom_path(sens_data, mapping = aes(1 - specificity,
    sensitivity, color = "Downsample"), size = 1)

load("models_nodum/RF_3_updown2.RData")
sens_data = rf_best %>% collect_predictions() %>% roc_curve(DESC_VICT,
    `.pred_Moderate-`, .pred_Severe, .pred_Fatal) %>% mutate(.level = ordered(.level,
    levels = c("Moderate-", "Severe", "Fatal")))
roc_5 = roc_5 + geom_path(sens_data, mapping = aes(1 - specificity,
    sensitivity, color = "Upsample"), size = 1)

colors <- c(Downsample = "#66C2A5", Upsample = "#FC8D62", Smote = "#8DA0CB")
roc_5 + facet_wrap(~.level, ncol = 2) + coord_equal() + scale_color_manual(values = colors) +
    labs(title = "3-class Classifiers (1) ROC Curves", color = "Sampling") +
    theme(legend.position = c(0.85, 0.25), legend.key.size = unit(1,
        "cm"), legend.box.background = element_rect(color = "grey35"))
```



## 4.3. 3-class Classifier 2

### 4.3.1. Downsample

```
load("models_nodum/RF_3_down.RData")
title = "Hyperparameters Tuned For 3-class Classifier 2 With Downsample"

rf_tune_res %>% collect_metrics() %>% filter(.metric == "roc_auc") %>%
    select(mean, min_n, mtry) %>% pivot_longer(min_n:mtry, values_to = "value",
    names_to = "parameter") %>% ggplot(aes(value, mean, color = parameter)) +
    geom_point(show.legend = F) + geom_smooth(method = "loess",
    show.legend = F) + facet_wrap(~parameter, scales = "free_x") +
    labs(title = title)
```




```
conf = rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf$table %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | Mild- | Moderate | Severe+ |
|---|---|---|---|
| Mild- | 442 | 1040 | 14 |
| Moderate | 339 | 784 | 13 |
| Severe+ | 281 | 765 | 61 |

```
rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT, .pred_class) %>%
    summary() %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | multiclass | 0.3442097 |
| kap | multiclass | 0.0197609 |
| sens | macro | 0.4707324 |
| spec | macro | 0.6712306 |
| ppv | macro | 0.3468998 |
| npv | macro | 0.6732985 |
| mcc | multiclass | 0.0245527 |
| j_index | macro | 0.1419631 |
| bal_accuracy | macro | 0.5709815 |
| detection_prevalence | macro | 0.3333333 |
| precision | macro | 0.3468998 |
| recall | macro | 0.4707324 |
| f_meas | macro | 0.2895380 |

```
rf_best %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

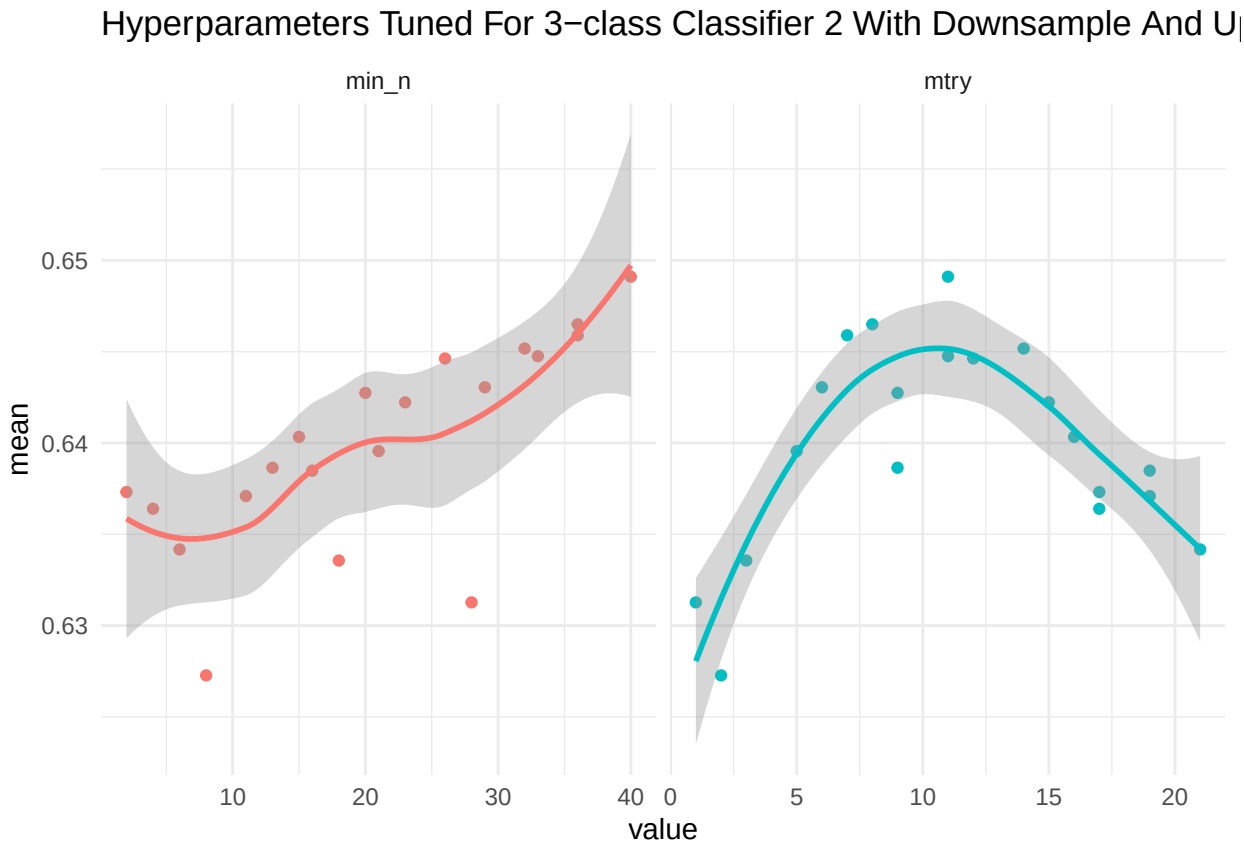
| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | multiclass | 0.3442097 | Preprocessor1_Model1 |
| roc_auc | hand_till | 0.6570811 | Preprocessor1_Model1 |

```
model = pull_workflow_fit(rf_best$.workflow[[1]])
paste(model$fit$prediction.error) %>% kable(col.names = "OOB Error") %>%
    kable_styling(full_width = F, font_size = 15)
```

| OOB Error |
|---|
| 0.412167456437651 |

#### 4.3.2. Updownsample

```
load("models_nodum/RF_3_updown.RData")
title = "Hyperparameters Tuned For 3-class Classifier 2 With Downsample And Upsample"

rf_tune_res %>% collect_metrics() %>% filter(.metric == "roc_auc") %>%
    select(mean, min_n, mtry) %>% pivot_longer(min_n:mtry, values_to = "value",
    names_to = "parameter") %>% ggplot(aes(value, mean, color = parameter)) +
    geom_point(show.legend = F) + geom_smooth(method = "loess",
    show.legend = F) + facet_wrap(~parameter, scales = "free_x") +
    labs(title = title)
```




```
conf = rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf$table %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | Mild- | Moderate | Severe+ |
|---|---|---|---|
| Mild- | 437 | 1049 | 14 |
| Moderate | 495 | 1137 | 22 |
| Severe+ | 130 | 403 | 52 |

```
rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT, .pred_class) %>%
    summary() %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | multiclass | 0.4348756 |
| kap | multiclass | 0.0189892 |
| sens | macro | 0.4805209 |
| spec | macro | 0.6691204 |
| ppv | macro | 0.3558822 |
| npv | macro | 0.6710135 |
| mcc | multiclass | 0.0209789 |
| j_index | macro | 0.1496412 |
| bal_accuracy | macro | 0.5748206 |
| detection_prevalence | macro | 0.3333333 |
| precision | macro | 0.3558822 |
| recall | macro | 0.4805209 |
| f_meas | macro | 0.3438711 |

```
rf_best %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

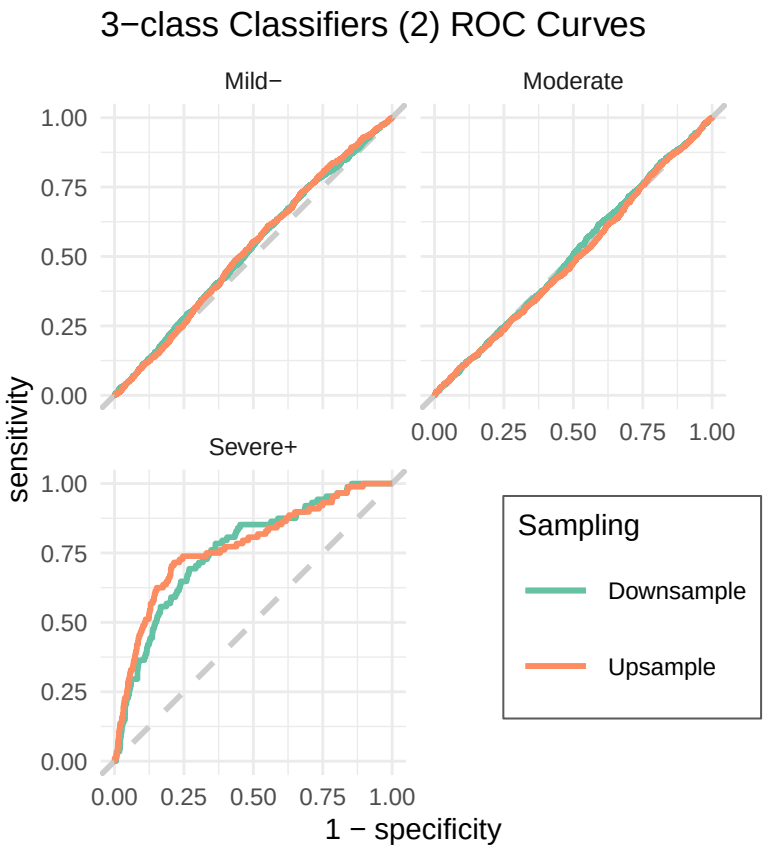
| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | multiclass | 0.4348756 | Preprocessor1_Model1 |
| roc_auc | hand_till | 0.6675586 | Preprocessor1_Model1 |

```
model = pull_workflow_fit(rf_best$.workflow[[1]])
paste(model$fit$prediction.error) %>% kable(col.names = "OOB Error") %>%
    kable_styling(full_width = F, font_size = 15)
```

| OOB Error |
|---|
| 0.29559309638775 |

#### 4.3.3. ROC For 3 Class Models

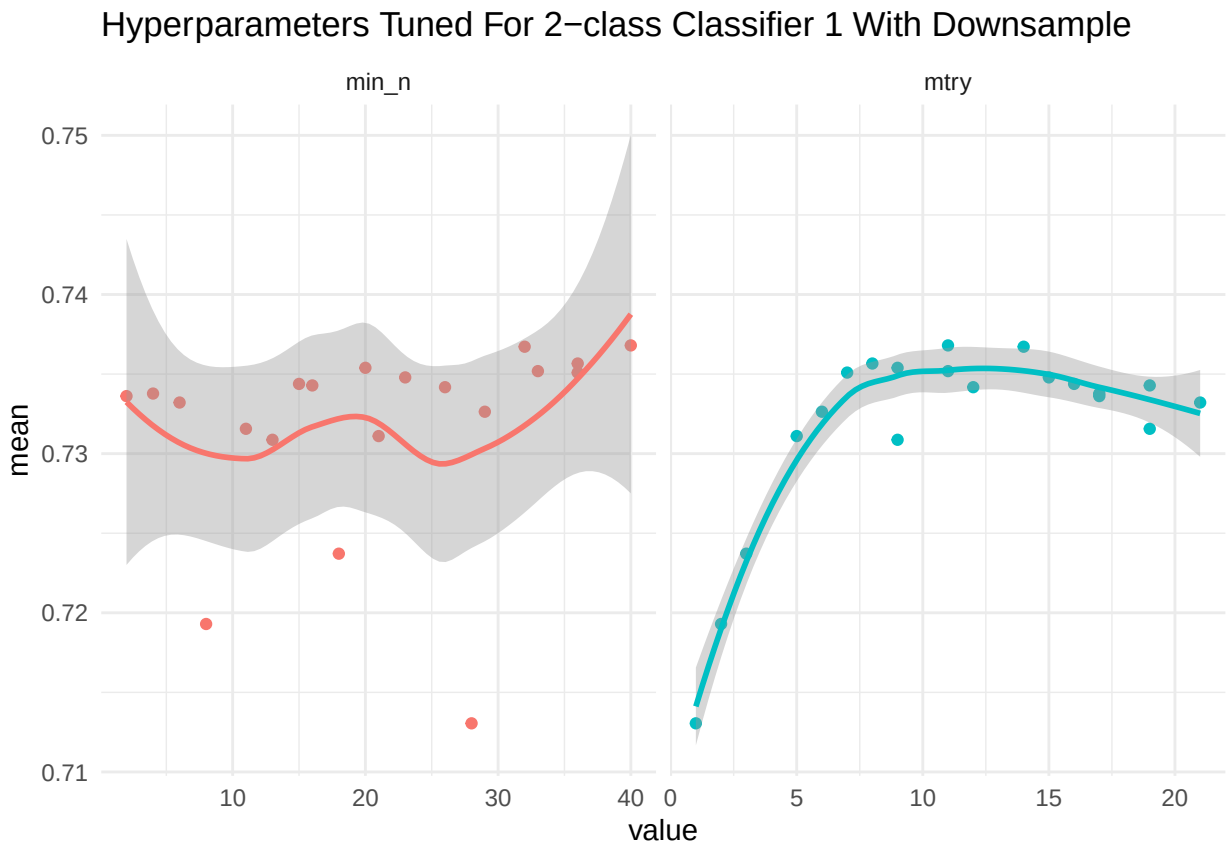
```
roc_5 = ggplot() + geom_abline(lty = 2, color = "gray80", size = 1)

load("models_nodum/RF_3_down.RData")
sens_data = rf_best %>% collect_predictions() %>% roc_curve(DESC_VICT,
    `.pred_Mild-`, .pred_Moderate, `.pred_Severe+`) %>% mutate(.level = ordered(.level,
    levels = c("Mild-", "Moderate", "Severe+")))
roc_5 = roc_5 + geom_path(sens_data, mapping = aes(1 - specificity,
    sensitivity, color = "Downsample"), size = 1)

load("models_nodum/RF_3_updown.RData")
sens_data = rf_best %>% collect_predictions() %>% roc_curve(DESC_VICT,
    `.pred_Mild-`, .pred_Moderate, `.pred_Severe+`) %>% mutate(.level = ordered(.level,
    levels = c("Mild-", "Moderate", "Severe+")))
roc_5 = roc_5 + geom_path(sens_data, mapping = aes(1 - specificity,
    sensitivity, color = "Upsample"), size = 1)

colors <- c(Downsample = "#66C2A5", Upsample = "#FC8D62", Smote = "#8DA0CB")
roc_5 + facet_wrap(~.level, ncol = 2) + coord_equal() + scale_color_manual(values = colors) +
    labs(title = "3-class Classifiers (2) ROC Curves", color = "Sampling") +
    theme(legend.position = c(0.85, 0.25), legend.key.size = unit(1,
        "cm"), legend.box.background = element_rect(color = "grey35"))
```



## 4.4. Binary Classifier 1

### 4.4.1. Downsample

```
load("models_nodum/RF_2_down.RData")
title = "Hyperparameters Tuned For 2-class Classifier 1 With Downsample"

rf_tune_res %>% collect_metrics() %>% filter(.metric == "roc_auc") %>%
    select(mean, min_n, mtry) %>% pivot_longer(min_n:mtry, values_to = "value",
    names_to = "parameter") %>% ggplot(aes(value, mean, color = parameter)) +
    geom_point(show.legend = F) + geom_smooth(method = "loess",
    show.legend = F) + facet_wrap(~parameter, scales = "free_x") +
    labs(title = title)
```




```
conf = rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf$table %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | Moderate- | Severe+ |
|---|---|---|
| Moderate- | 2519 | 33 |
| Severe+ | 1131 | 56 |

```
rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT, .pred_class) %>%
    summary() %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | binary | 0.6886868 |
| kap | binary | 0.0455037 |
| sens | binary | 0.6901370 |
| spec | binary | 0.6292135 |
| ppv | binary | 0.9870690 |
| npv | binary | 0.0471778 |
| mcc | binary | 0.1045787 |
| j_index | binary | 0.3193505 |
| bal_accuracy | binary | 0.6596752 |
| detection_prevalence | binary | 0.6825354 |
| precision | binary | 0.9870690 |
| recall | binary | 0.6901370 |
| f_meas | binary | 0.8123186 |

```
rf_best %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

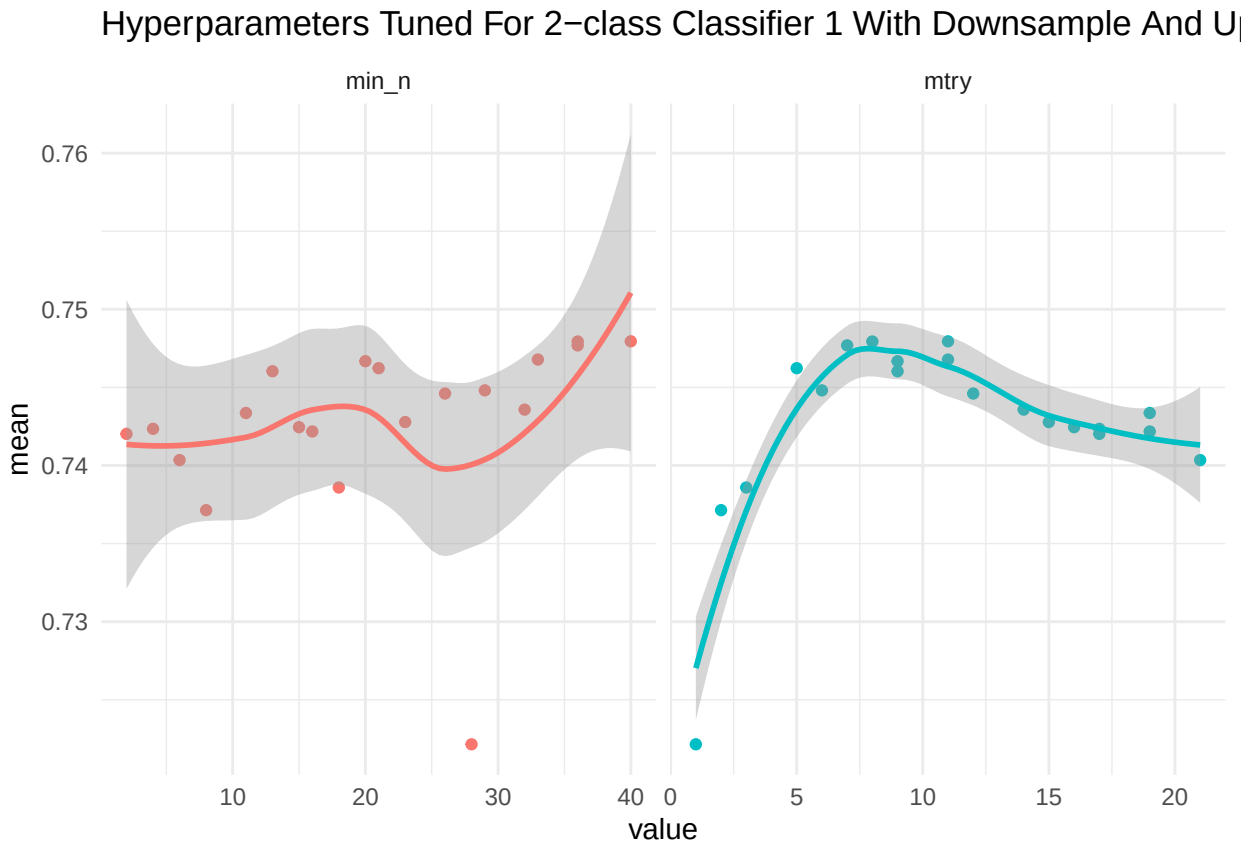
| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | binary | 0.6886868 | Preprocessor1_Model1 |
| roc_auc | binary | 0.7245898 | Preprocessor1_Model1 |

```
model = pull_workflow_fit(rf_best$.workflow[[1]])
paste(model$fit$prediction.error) %>% kable(col.names = "OOB Error") %>%
    kable_styling(full_width = F, font_size = 15)
```

| OOB Error |
|---|
| 0.206737902295291 |

### 4.4.2. Updownsample

```
load("models_nodum/RF_2_updown.RData")
title = "Hyperparameters Tuned For 2-class Classifier 1 With Downsample And Upsample"

rf_tune_res %>% collect_metrics() %>% filter(.metric == "roc_auc") %>%
    select(mean, min_n, mtry) %>% pivot_longer(min_n:mtry, values_to = "value",
    names_to = "parameter") %>% ggplot(aes(value, mean, color = parameter)) +
    geom_point(show.legend = F) + geom_smooth(method = "loess",
    show.legend = F) + facet_wrap(~parameter, scales = "free_x") +
    labs(title = title)
```




```
conf = rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf$table %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | Moderate- | Severe+ |
|---|---|---|
| Moderate- | 3228 | 56 |
| Severe+ | 422 | 33 |

```
rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT, .pred_class) %>%
    summary() %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | binary | 0.8721583 |
| kap | binary | 0.0848857 |
| sens | binary | 0.8843836 |
| spec | binary | 0.3707865 |
| ppv | binary | 0.9829476 |
| npv | binary | 0.0725275 |
| mcc | binary | 0.1189772 |
| j_index | binary | 0.2551701 |
| bal_accuracy | binary | 0.6275850 |
| detection_prevalence | binary | 0.8783097 |
| precision | binary | 0.9829476 |
| recall | binary | 0.8843836 |
| f_meas | binary | 0.9310643 |

```
rf_best %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

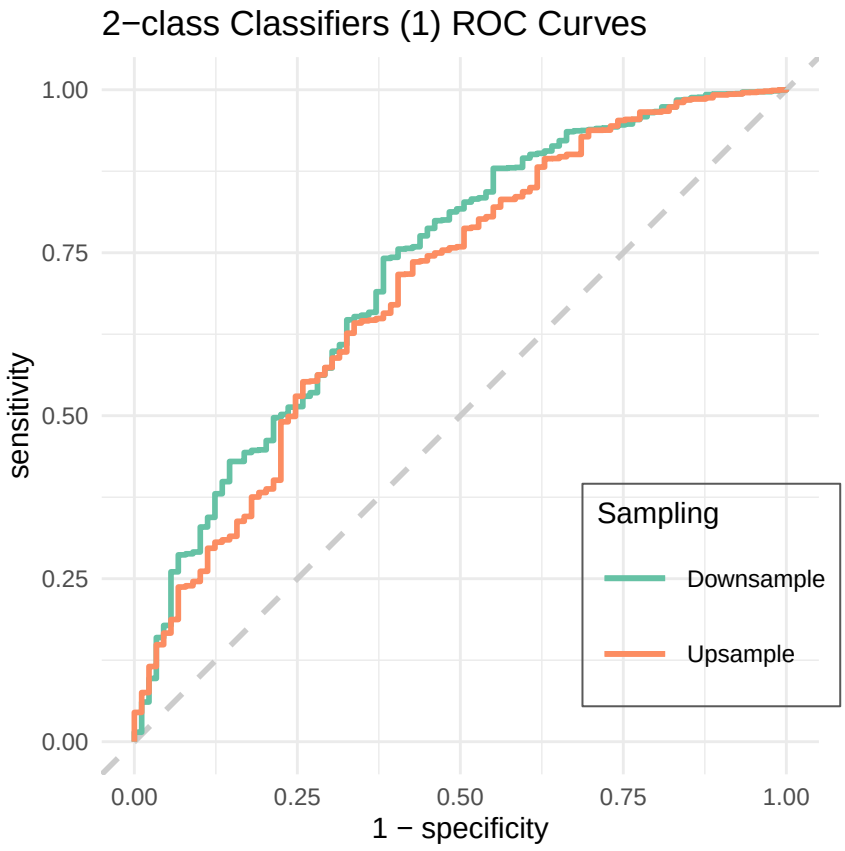
| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | binary | 0.8721583 | Preprocessor1_Model1 |
| roc_auc | binary | 0.6985162 | Preprocessor1_Model1 |

```
model = pull_workflow_fit(rf_best$.workflow[[1]])
paste(model$fit$prediction.error) %>% kable(col.names = "OOB Error") %>%
    kable_styling(full_width = F, font_size = 15)
```

| OOB Error |
|---|
| 0.0882170299244528 |

#### 4.4.3. ROC For 2 Class Models 1

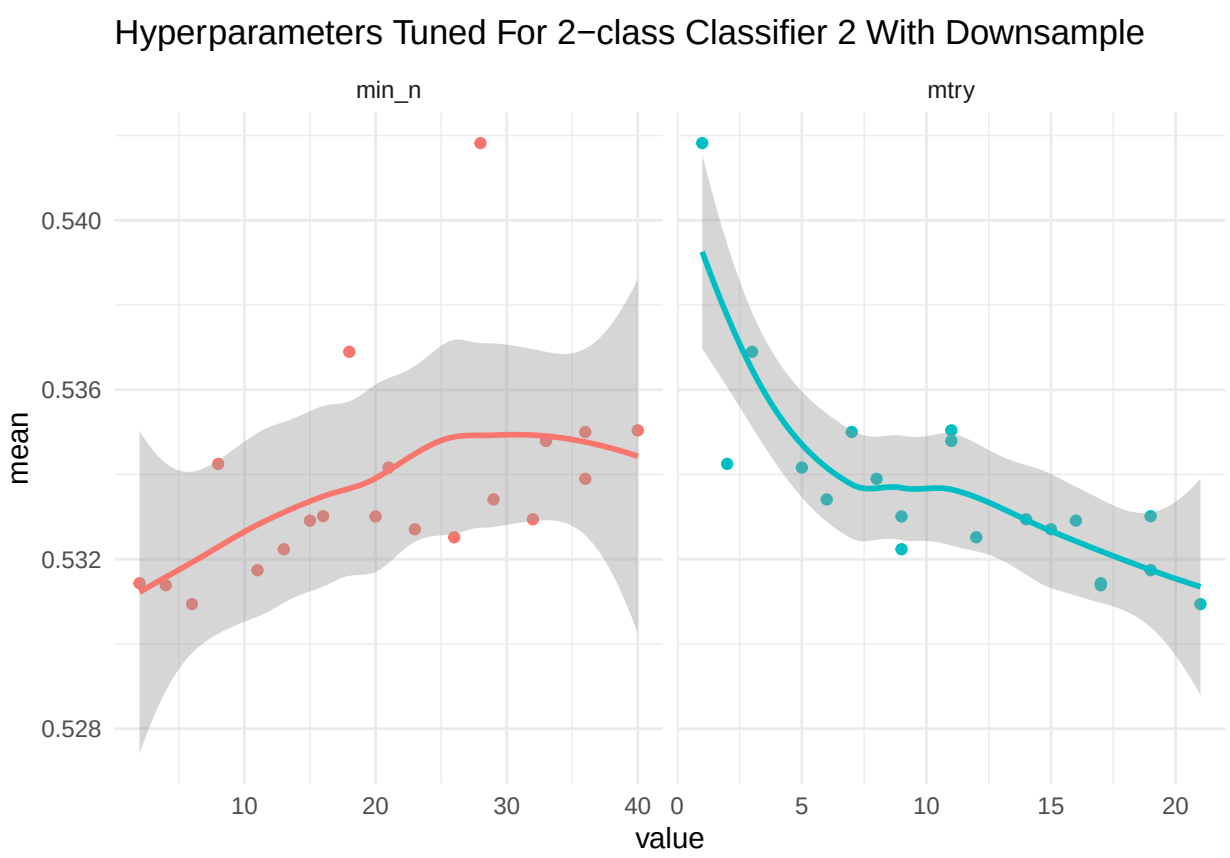
```
roc_5 = ggplot() + geom_abline(lty = 2, color = "gray80", size = 1)

load("models_nodum/RF_2_down.RData")
sens_data = rf_best %>% collect_predictions() %>% roc_curve(DESC_VICT,
    `.pred_Moderate-`)
roc_5 = roc_5 + geom_path(sens_data, mapping = aes(1 - specificity,
    sensitivity, color = "Downsample"), size = 1)

load("models_nodum/RF_2_updown.RData")
sens_data = rf_best %>% collect_predictions() %>% roc_curve(DESC_VICT,
    `.pred_Moderate-`)
roc_5 = roc_5 + geom_path(sens_data, mapping = aes(1 - specificity,
    sensitivity, color = "Upsample"), size = 1)

colors <- c(Downsample = "#66C2A5", Upsample = "#FC8D62", Smote = "#8DA0CB")
roc_5 + coord_equal() + scale_color_manual(values = colors) +
    labs(title = "2-class Classifiers (1) ROC Curves", color = "Sampling") +
    theme(legend.position = c(0.85, 0.25), legend.key.size = unit(1,
        "cm"), legend.box.background = element_rect(color = "grey35"))
```



## 4.5. Binary Classifier 2

### 4.5.1. Downsample

```
load("models_nodum/RF_2_down2.RData")
title = "Hyperparameters Tuned For 2-class Classifier 2 With Downsample"

rf_tune_res %>% collect_metrics() %>% filter(.metric == "roc_auc") %>%
    select(mean, min_n, mtry) %>% pivot_longer(min_n:mtry, values_to = "value",
    names_to = "parameter") %>% ggplot(aes(value, mean, color = parameter)) +
    geom_point(show.legend = F) + geom_smooth(method = "loess",
    show.legend = F) + facet_wrap(~parameter, scales = "free_x") +
    labs(title = title)
```




```
conf = rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf$table %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | Mild- | Moderate+ |
|---|---|---|
| Mild- | 548 | 1241 |
| Moderate+ | 515 | 1435 |

```
rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT, .pred_class) %>%
    summary() %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | binary | 0.5303557 |
| kap | binary | 0.0429331 |
| sens | binary | 0.5155221 |
| spec | binary | 0.5362481 |
| ppv | binary | 0.3063164 |
| npv | binary | 0.7358974 |
| mcc | binary | 0.0467485 |
| j_index | binary | 0.0517702 |
| bal_accuracy | binary | 0.5258851 |
| detection_prevalence | binary | 0.4784702 |
| precision | binary | 0.3063164 |
| recall | binary | 0.5155221 |
| f_meas | binary | 0.3842917 |

```
rf_best %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

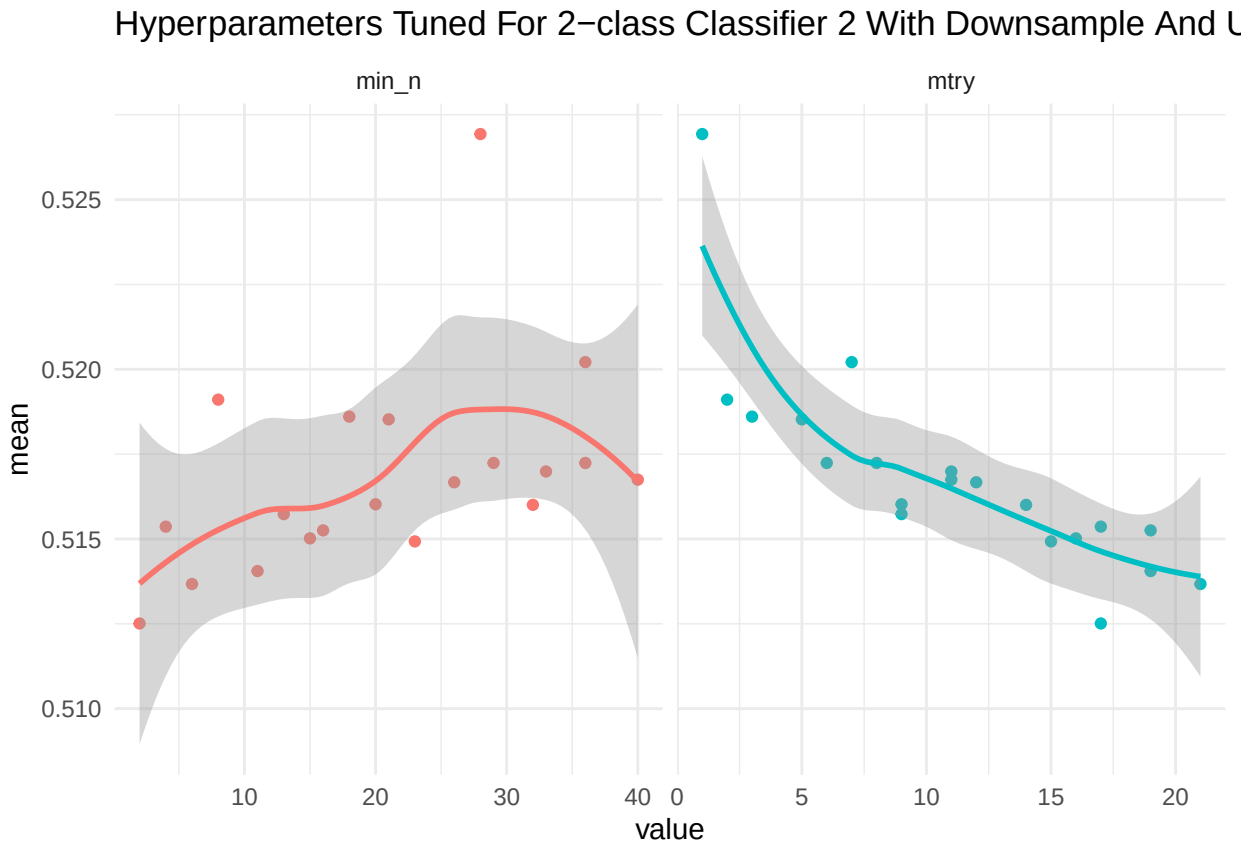
| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | binary | 0.5303557 | Preprocessor1_Model1 |
| roc_auc | binary | 0.5348578 | Preprocessor1_Model1 |

```
model = pull_workflow_fit(rf_best$.workflow[[1]])
paste(model$fit$prediction.error) %>% kable(col.names = "OOB Error") %>%
    kable_styling(full_width = F, font_size = 15)
```

| OOB Error |
|---|
| 0.248185437225306 |

### 4.5.2. Updownsample

```
load("models_nodum/RF_2_updown2.RData")
title = "Hyperparameters Tuned For 2-class Classifier 2 With Downsample And Upsample"

rf_tune_res %>% collect_metrics() %>% filter(.metric == "roc_auc") %>%
    select(mean, min_n, mtry) %>% pivot_longer(min_n:mtry, values_to = "value",
    names_to = "parameter") %>% ggplot(aes(value, mean, color = parameter)) +
    geom_point(show.legend = F) + geom_smooth(method = "loess",
    show.legend = F) + facet_wrap(~parameter, scales = "free_x") +
    labs(title = title)
```




```
conf = rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf$table %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | Mild- | Moderate+ |
|---|---|---|
| Mild- | 448 | 981 |
| Moderate+ | 615 | 1695 |

```
rf_best %>% collect_predictions() %>% conf_mat(DESC_VICT, .pred_class) %>%
    summary() %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | binary | 0.5731479 |
| kap | binary | 0.0496995 |
| sens | binary | 0.4214487 |
| spec | binary | 0.6334081 |
| ppv | binary | 0.3135059 |
| npv | binary | 0.7337662 |
| mcc | binary | 0.0509235 |
| j_index | binary | 0.0548568 |
| bal_accuracy | binary | 0.5274284 |
| detection_prevalence | binary | 0.3821878 |
| precision | binary | 0.3135059 |
| recall | binary | 0.4214487 |
| f_meas | binary | 0.3595506 |

```
rf_best %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

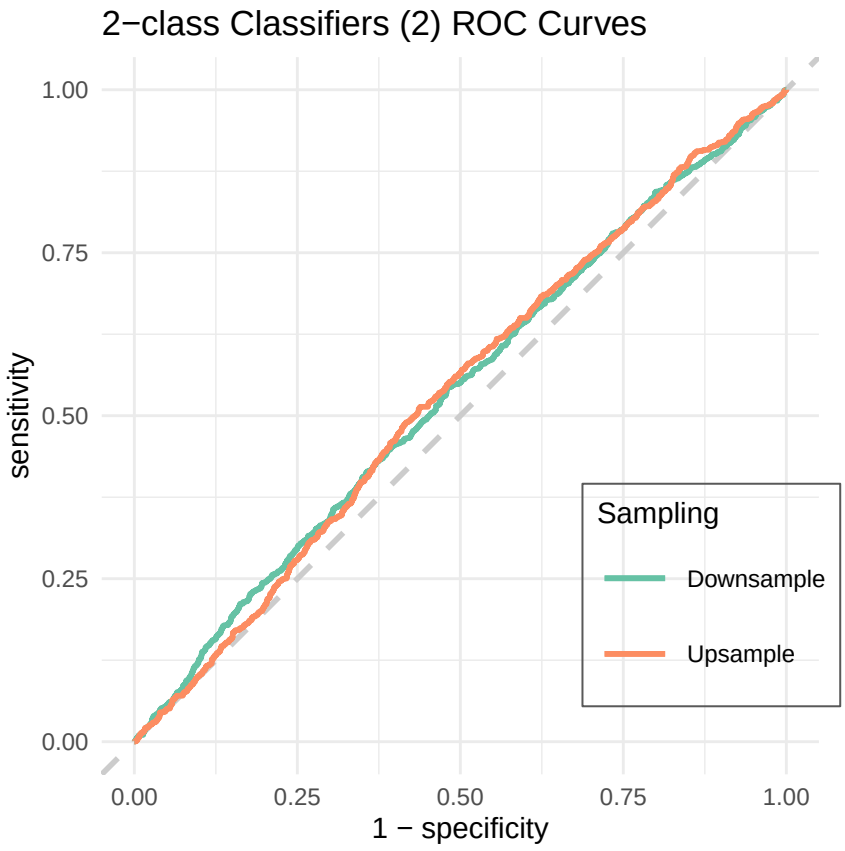
| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | binary | 0.5731479 | Preprocessor1_Model1 |
| roc_auc | binary | 0.5344507 | Preprocessor1_Model1 |

```
model = pull_workflow_fit(rf_best$.workflow[[1]])
paste(model$fit$prediction.error) %>% kable(col.names = "OOB Error") %>%
    kable_styling(full_width = F, font_size = 15)
```

| OOB Error |
|---|
| 0.224306452459627 |

#### 4.5.3. ROC For 2 Class Models 2

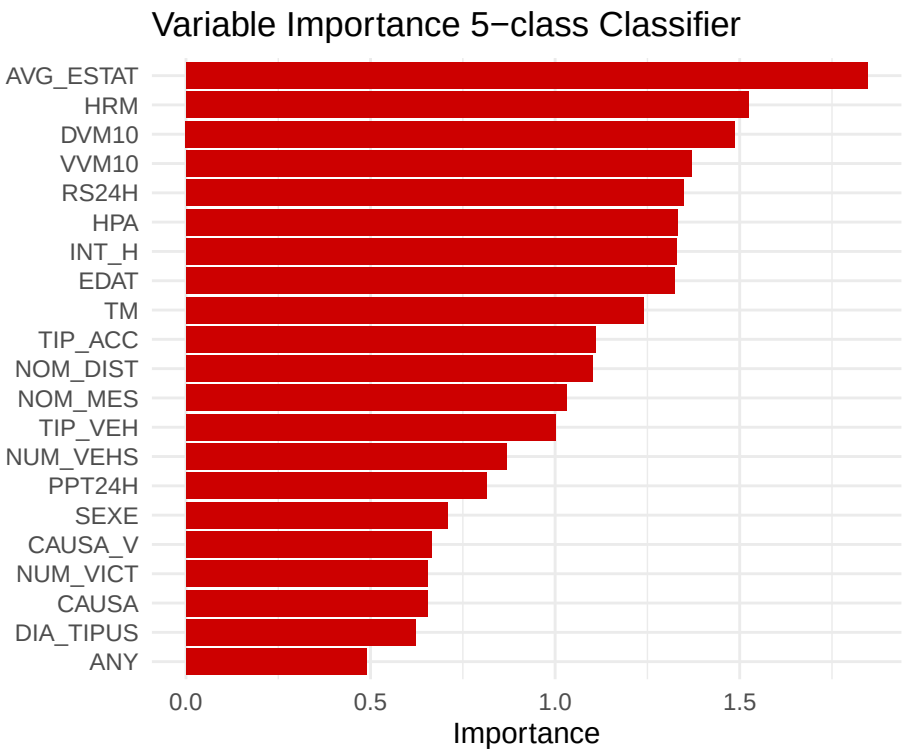
```
roc_5 = ggplot() + geom_abline(lty = 2, color = "gray80", size = 1)

load("models_nodum/RF_2_down2.RData")
sens_data = rf_best %>% collect_predictions() %>% roc_curve(DESC_VICT,
    `.pred_Mild-`)
roc_5 = roc_5 + geom_path(sens_data, mapping = aes(1 - specificity,
    sensitivity, color = "Downsample"), size = 1)

load("models_nodum/RF_2_updown2.RData")
sens_data2 = rf_best %>% collect_predictions() %>% roc_curve(DESC_VICT,
    `.pred_Mild-`)
roc_5 = roc_5 + geom_path(sens_data2, mapping = aes(1 - specificity,
    sensitivity, color = "Upsample"), size = 1)

colors <- c(Downsample = "#66C2A5", Upsample = "#FC8D62", Smote = "#8DA0CB")
roc_5 + coord_equal() + scale_color_manual(values = colors) +
    labs(title = "2-class Classifiers (2) ROC Curves", color = "Sampling") +
    theme(legend.position = c(0.85, 0.25), legend.key.size = unit(1,
        "cm"), legend.box.background = element_rect(color = "grey35"))
```



## 5. Variable Importance Selection

5-class Classifier

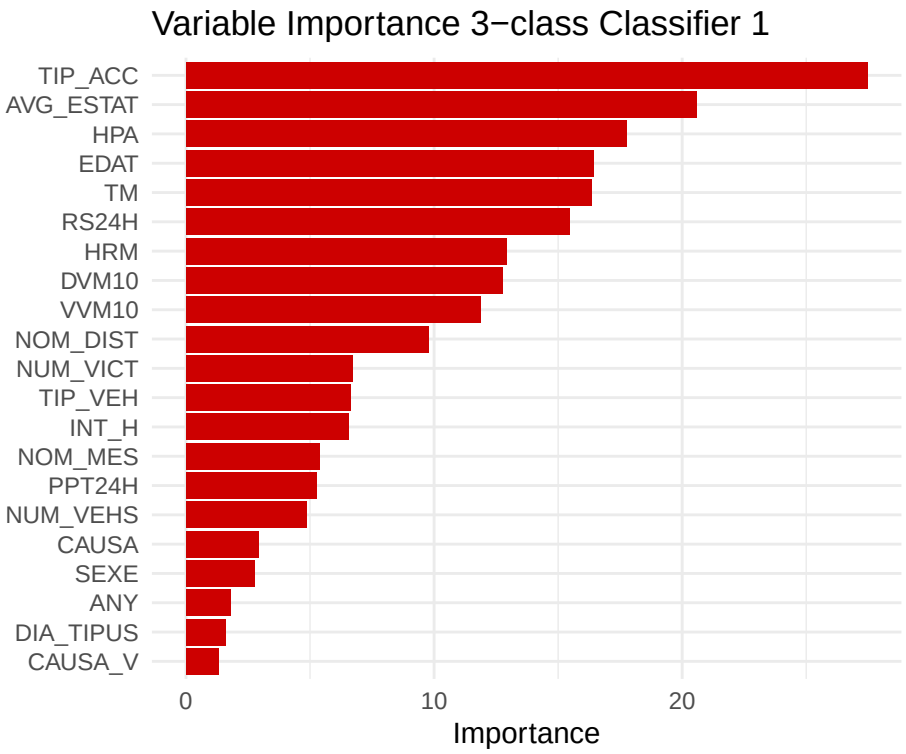
```
load("models_nodum/RF_5_down.RData")
title = "Variable Importance 5-class Classifier"

model = pull_workflow_fit(rf_best$.workflow[[1]])
vip(model, num_features = 25, geom = "col", aesthetics = list(fill = "red3")) +
    labs(title = title)
```




```
vi_5_RF = vi(model, method = "model") %>% slice(tail(row_number(),
    5))
```

3-class Classifier 1

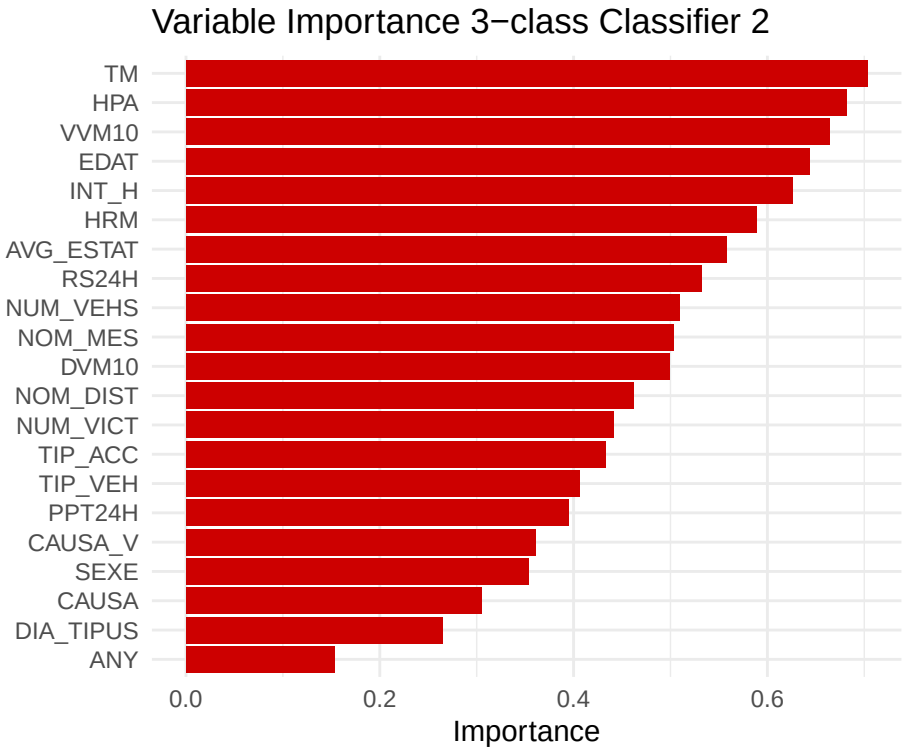
```
load("models_nodum/RF_3_down.RData")
title = "Variable Importance 3-class Classifier 1"


model = pull_workflow_fit(rf_best$.workflow[[1]])
vip(model, num_features = 25, geom = "col", aesthetics = list(fill = "red3")) +
    labs(title = title)
```




```
vi_3_RF = vi(model, method = "model") %>% slice(tail(row_number(),
    5))
```

3-class Classifier 2

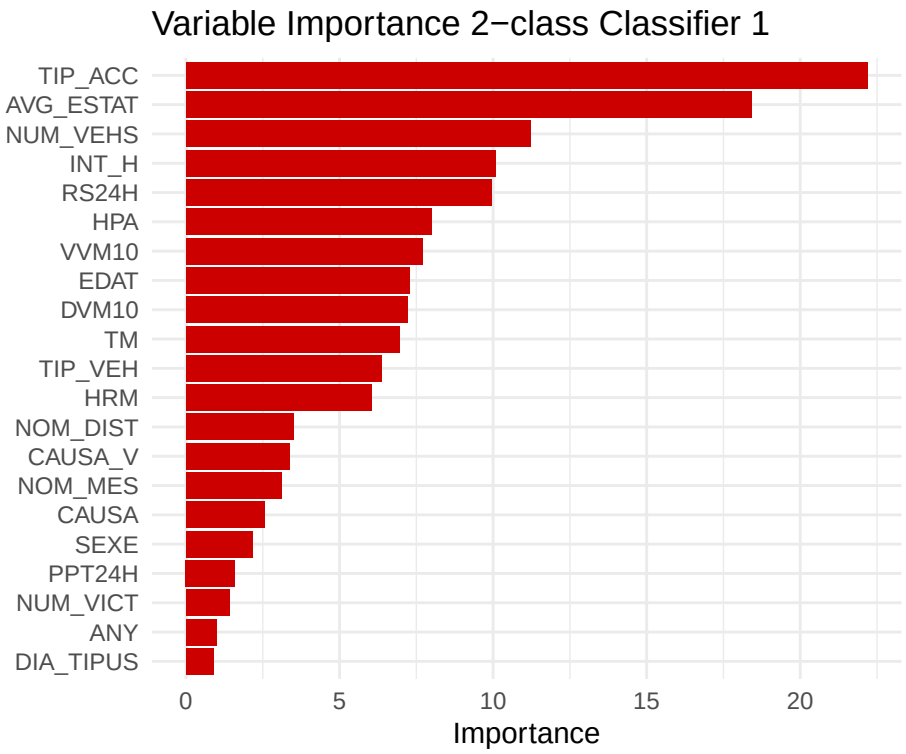
```
load("models_nodum/RF_3_down2.RData")
title = "Variable Importance 3-class Classifier 2"


model = pull_workflow_fit(rf_best$.workflow[[1]])
vip(model, num_features = 25, geom = "col", aesthetics = list(fill = "red3")) +
    labs(title = title)
```




```
vi_3_RF2 = vi(model, method = "model") %>% slice(tail(row_number(),
    5))
```

Binary Classifier 1

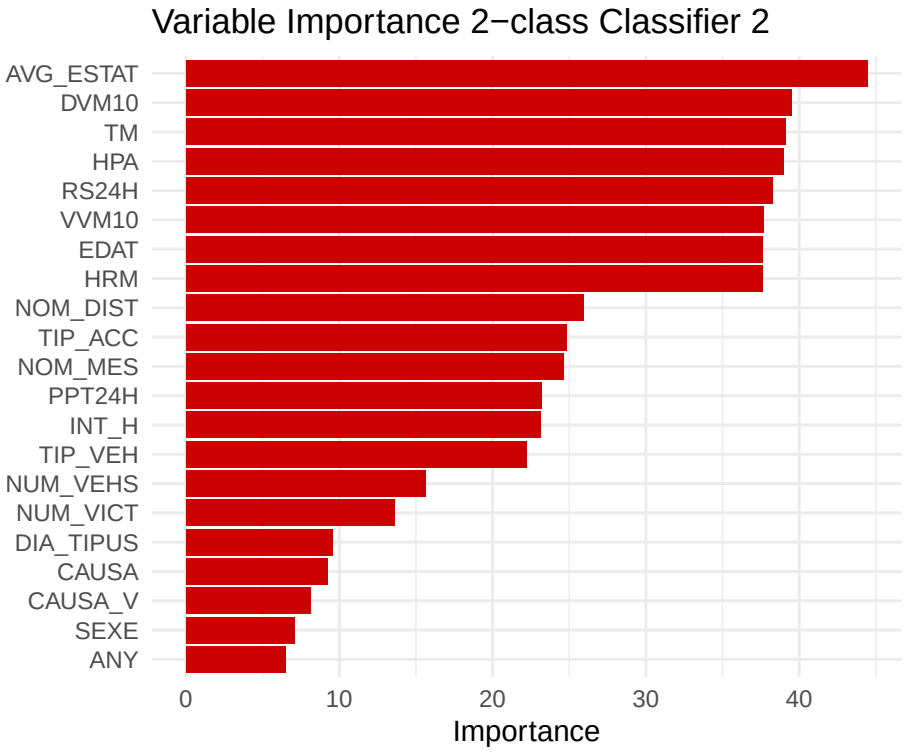
```
load("models_nodum/RF_2_down.RData")
title = "Variable Importance 2-class Classifier 1"


model = pull_workflow_fit(rf_best$.workflow[[1]])
vip(model, num_features = 25, geom = "col", aesthetics = list(fill = "red3")) +
    labs(title = title)
```




```
vi_2_RF = vi(model, method = "model") %>% slice(tail(row_number(),
    5))
```

Binary Classifier 2

```
load("models_nodum/RF_2_down2.RData")
title = "Variable Importance 2-class Classifier 2"


model = pull_workflow_fit(rf_best$.workflow[[1]])
vip(model, num_features = 25, geom = "col", aesthetics = list(fill = "red3")) +
    labs(title = title)
```




```
vi_2_RF2 = vi(model, method = "model") %>% slice(tail(row_number(),
    5))
```

Store variable importance selection:

```
rm(list = setdiff(ls(), c("vi_5_RF", "vi_3_RF", "vi_3_RF2", "vi_2_RF",
    "vi_2_RF2")))
# save.image('models/VI_RF.Rdata')
```

# 6. Classification Trees Workflow

Start counting time:

```
total_ti = Sys.time()
```

Import dataframe:

```
df = read.csv("FinalDatasets/dataset_models.csv")

order = c("No Injury", "Mild", "Moderate", "Severe", "Fatal")
df = df %>% mutate(DESC_VICT = ordered(DESC_VICT, levels = order))

load("models/VI_RF.Rdata")
```

## 6.1. 5-class Classifier

Split data:

```
set.seed(123)

split = initial_split(df, strata = DESC_VICT, prob = 0.7)
train = training(split)
test = testing(split)

table(train$DESC_VICT)
table(test$DESC_VICT)
```

Sampling:

```
rec = recipe(DESC_VICT ~ ., data = train) %>% step_rm(!!!syms(c(vi_5_RF$Variable))) %>%
    themis::step_downsample(DESC_VICT, under_ratio = 5) %>% themis::step_upsample(DESC_VICT)

prep = prep(rec)
juiced = juice(prep)
```

Cross-validation:

```
set.seed(234)

cv_folds = vfold_cv(train, v = 10, strata = DESC_VICT)
```

Tune hyperparameters:

```
dt_tune = decision_tree(cost_complexity = tune(), min_n = tune()) %>%
    set_engine("rpart") %>% set_mode("classification")

dt_tune_wf = workflow() %>% add_recipe(rec) %>% add_model(dt_tune)

ti = Sys.time()

doParallel::registerDoParallel()
set.seed(345)

tree_grid = grid_regular(cost_complexity(c(-1, -1)), min_n(),
    levels = 32)
tree_grid[1] = -1
dt_tune_res = dt_tune_wf %>% tune_grid(resamples = cv_folds,
```

```
    grid = tree_grid)

tf = Sys.time()
model_tune_t = tf - ti
model_tune_t
```

Best tune:

```
best_auc = select_best(dt_tune_res, "roc_auc")
```

Final model:

```
final_dt = finalize_model(dt_tune, best_auc)

dt_best = workflow() %>% add_recipe(rec) %>% add_model(final_dt) %>%
    last_fit(split)
```

Prune tree:

```
model = pull_workflow_fit(dt_best$.workflow[[1]])
# plotcp(model$fit)
i = which.min(model$fit$cptable[, 4])[[1]]
xerror_best = model$fit$cptable[i, 4]
std_best = model$fit$cptable[i, 5]
table_best = as.data.frame(model$fit$cptable) %>% filter(CP >
    0) %>% filter(xerror <= xerror_best + std_best) %>% arrange(nsplit)

cp_prune = table_best[1, 1]
```

Fit pruned tree:

```
best_auc[1, 1] = cp_prune

final_dt_prune = finalize_model(dt_tune, best_auc)

dt_best_prune = workflow() %>% add_recipe(rec) %>% add_model(final_dt_prune) %>%
    last_fit(split)
```

Store model:

```
# save.image('models_dt_moredata/DT_5_down_vi.RData')
```

## 6.2. 3-class Classifier 1

Group into 3 categories:

```
df = read.csv("FinalDatasets/dataset_models.csv")

order = c("No Injury", "Mild", "Moderate", "Severe", "Fatal")
df = df %>%
  mutate(DESC_VICT = ordered(DESC_VICT, levels = order))

df = df %>%
  mutate(DESC_VICT = as.character(DESC_VICT)) %>%
  mutate(DESC_VICT = case_when(

    DESC_VICT %in% c("No Injury", "Mild") ~ "Mild-",
```

```
                                DESC_VICT %in% c("Severe", "Fatal") ~ "Severe+",
                                TRUE ~ "Moderate")) %>%
  mutate(DESC_VICT = ordered(DESC_VICT, levels = c("Mild-", "Moderate", "Severe+")))
```

Split data:

```
set.seed(123)

split = initial_split(df, strata = DESC_VICT, prob = 0.7)
train = training(split)
test = testing(split)

table(train$DESC_VICT)
table(test$DESC_VICT)
```

Sampling:

```
rec = recipe(DESC_VICT ~ ., data = train) %>% step_rm(!!!syms(c(vi_3_RF$Variable))) %>%
    themis::step_downsample(DESC_VICT, under_ratio = 1.6) %>%
    themis::step_upsample(DESC_VICT)

prep = prep(rec)
juiced = juice(prep)
```

Cross-validation:

```
set.seed(234)

cv_folds = vfold_cv(train, v = 10, strata = DESC_VICT)
```

Tune hyperparameters:

```
dt_tune = decision_tree(cost_complexity = tune(), min_n = tune()) %>%
    set_engine("rpart") %>% set_mode("classification")

dt_tune_wf = workflow() %>% add_recipe(rec) %>% add_model(dt_tune)

ti = Sys.time()

doParallel::registerDoParallel()
set.seed(345)

tree_grid = grid_regular(cost_complexity(c(-1, -1)), min_n(),
    levels = 32)
tree_grid[1] = -1

dt_tune_res = dt_tune_wf %>% tune_grid(resamples = cv_folds,
    grid = tree_grid)

tf = Sys.time()
model_tune_t = tf - ti
model_tune_t
```

Best tune:

```
best_auc = select_best(dt_tune_res, "roc_auc")
```

Final model:

```
final_dt = finalize_model(dt_tune, best_auc)

dt_best = workflow() %>% add_recipe(rec) %>% add_model(final_dt) %>%
    last_fit(split)
```

Prune tree:

```
model = pull_workflow_fit(dt_best$.workflow[[1]])
# plotcp(model$fit)
i = which.min(model$fit$cptable[, 4])[[1]]
xerror_best = model$fit$cptable[i, 4]
std_best = model$fit$cptable[i, 5]
table_best = as.data.frame(model$fit$cptable) %>% filter(CP >
    0) %>% filter(xerror <= xerror_best + std_best) %>% arrange(nsplit)

cp_prune = table_best[1, 1]
```

Fit pruned tree:

```
best_auc[1, 1] = cp_prune

final_dt_prune = finalize_model(dt_tune, best_auc)

dt_best_prune = workflow() %>% add_recipe(rec) %>% add_model(final_dt_prune) %>%
    last_fit(split)
```

Store model:

```
# save.image('models_dt_moredata/DT_3_down_vi.RData')
```

### 6.3. 3-class Classifier 2

Group into 3 categories:

```
df = read.csv("FinalDatasets/dataset_models.csv")

order = c("No Injury", "Mild", "Moderate", "Severe", "Fatal")
df = df %>%
  mutate(DESC_VICT = ordered(DESC_VICT, levels = order))

df = df %>%
  mutate(DESC_VICT = as.character(DESC_VICT)) %>%
  mutate(DESC_VICT = case_when(

    DESC_VICT %in% c("No Injury", "Mild", "Moderate") ~ "Moderate-",
                                TRUE ~ DESC_VICT)) %>%
  mutate(DESC_VICT = ordered(DESC_VICT, levels = c("Moderate-", "Severe", "Fatal")))
```

Split data:

```
set.seed(123)

split = initial_split(df, strata = DESC_VICT, prob = 0.7)
train = training(split)
test = testing(split)
```

```
table(train$DESC_VICT)
table(test$DESC_VICT)
```

Sampling:

```
rec = recipe(DESC_VICT ~ ., data = train) %>% step_rm(!!!syms(c(vi_3_RF2$Variable))) %>%
    themis::step_downsample(DESC_VICT, under_ratio = 5) %>% themis::step_upsample(DESC_VICT)

prep = prep(rec)
juiced = juice(prep)
```

Cross-validation:

```
set.seed(234)

cv_folds = vfold_cv(train, v = 10, strata = DESC_VICT)
```

Tune hyperparameters:

```
dt_tune = decision_tree(cost_complexity = tune(), min_n = tune()) %>%
    set_engine("rpart") %>% set_mode("classification")

dt_tune_wf = workflow() %>% add_recipe(rec) %>% add_model(dt_tune)

ti = Sys.time()

doParallel::registerDoParallel()
set.seed(345)

tree_grid = grid_regular(cost_complexity(c(-1, -1)), min_n(),
    levels = 32)
tree_grid[1] = -1
dt_tune_res = dt_tune_wf %>% tune_grid(resamples = cv_folds,
    grid = tree_grid)

tf = Sys.time()
model_tune_t = tf - ti
model_tune_t
```

Best tune:

```
best_auc = select_best(dt_tune_res, "roc_auc")
```

Final model:

```
final_dt = finalize_model(dt_tune, best_auc)

dt_best = workflow() %>% add_recipe(rec) %>% add_model(final_dt) %>%
    last_fit(split)
```

Prune tree:

```
model = pull_workflow_fit(dt_best$.workflow[[1]])
# plotcp(model$fit)
i = which.min(model$fit$cptable[, 4])[[1]]
xerror_best = model$fit$cptable[i, 4]
std_best = model$fit$cptable[i, 5]
table_best = as.data.frame(model$fit$cptable) %>% filter(CP >
```

```
    0) %>% filter(xerror <= xerror_best + std_best) %>% arrange(nsplit)

cp_prune = table_best[1, 1]
```

Fit pruned tree:

```
best_auc[1, 1] = cp_prune

final_dt_prune = finalize_model(dt_tune, best_auc)

dt_best_prune = workflow() %>% add_recipe(rec) %>% add_model(final_dt_prune) %>%
    last_fit(split)
```

Store model:

```
# save.image('models_dt_moredata/DT_3_down2_vi.RData')
```

## 6.4. Binary Classifier 1

Group into 2 categories:

```
df = read.csv("FinalDatasets/dataset_models.csv")

order = c("No Injury", "Mild", "Moderate", "Severe", "Fatal")
df = df %>%
  mutate(DESC_VICT = ordered(DESC_VICT, levels = order))

df = df %>%
  mutate(DESC_VICT = as.character(DESC_VICT)) %>%
  mutate(DESC_VICT = case_when(

    DESC_VICT %in% c("No Injury", "Mild", "Moderate") ~ "Moderate-",
                                TRUE ~ "Severe+")) %>%
  mutate(DESC_VICT = ordered(DESC_VICT, levels = c("Moderate-", "Severe+")))
```

Split data:

```
set.seed(123)

split = initial_split(df, strata = DESC_VICT, prob = 0.7)
train = training(split)
test = testing(split)

table(train$DESC_VICT)
table(test$DESC_VICT)
```

Sampling:

```
rec = recipe(DESC_VICT ~ ., data = train) %>% step_rm(!!!syms(c(vi_2_RF$Variable))) %>%
    themis::step_downsample(DESC_VICT, under_ratio = 3) %>% themis::step_upsample(DESC_VICT)

prep = prep(rec)
juiced = juice(prep)
```

Cross-validation:

```
set.seed(234)

cv_folds = vfold_cv(train, v = 10, strata = DESC_VICT)
```

Tune hyperparameters:

```
dt_tune = decision_tree(cost_complexity = tune(), min_n = tune()) %>%
    set_engine("rpart") %>% set_mode("classification")

dt_tune_wf = workflow() %>% add_recipe(rec) %>% add_model(dt_tune)

ti = Sys.time()

doParallel::registerDoParallel()
set.seed(345)

tree_grid = grid_regular(cost_complexity(c(-1, -1)), min_n(),
    levels = 32)
tree_grid[1] = -1
dt_tune_res = dt_tune_wf %>% tune_grid(resamples = cv_folds,
    grid = tree_grid)

tf = Sys.time()
model_tune_t = tf - ti
model_tune_t
```

Best tune:

```
best_auc = select_best(dt_tune_res, "roc_auc")
```

Final model:

```
final_dt = finalize_model(dt_tune, best_auc)

dt_best = workflow() %>% add_recipe(rec) %>% add_model(final_dt) %>%
    last_fit(split)
```

Prune tree:

```
model = pull_workflow_fit(dt_best$.workflow[[1]])
# plotcp(model$fit)
i = which.min(model$fit$cptable[, 4])[[1]]
xerror_best = model$fit$cptable[i, 4]
std_best = model$fit$cptable[i, 5]
table_best = as.data.frame(model$fit$cptable) %>% filter(CP >
    0) %>% filter(xerror <= xerror_best + std_best) %>% arrange(nsplit)

cp_prune = table_best[1, 1]
```

Fit pruned tree:

```
best_auc[1, 1] = cp_prune

final_dt_prune = finalize_model(dt_tune, best_auc)

dt_best_prune = workflow() %>% add_recipe(rec) %>% add_model(final_dt_prune) %>%
    last_fit(split)
```

Store model:

```
# save.image('models_dt_moredata/DT_2_down_vi.RData')
```

## 6.5. Binary Classifier 2

Group into 2 categories:

```
df = read.csv("FinalDatasets/dataset_models.csv")

order = c("No Injury", "Mild", "Moderate", "Severe", "Fatal")
df = df %>%
  mutate(DESC_VICT = ordered(DESC_VICT, levels = order))

df = df %>%
  mutate(DESC_VICT = as.character(DESC_VICT)) %>%
  mutate(DESC_VICT = case_when(

    DESC_VICT %in% c("No Injury", "Mild") ~ "Mild-",
                                TRUE ~ "Moderate+")) %>%
  mutate(DESC_VICT = ordered(DESC_VICT, levels = c("Mild-", "Moderate+")))
```

Split data:

```
set.seed(123)

split = initial_split(df, strata = DESC_VICT, prob = 0.7)
train = training(split)
test = testing(split)

table(train$DESC_VICT)
table(test$DESC_VICT)
```

Sampling:

```
rec = recipe(DESC_VICT ~ ., data = train) %>% step_rm(!!!syms(c(vi_2_RF2$Variable))) %>%
    themis::step_downsample(DESC_VICT, under_ratio = 1) %>% themis::step_upsample(DESC_VICT)

prep = prep(rec)
juiced = juice(prep)
```

Cross-validation:

```
set.seed(234)

cv_folds = vfold_cv(train, v = 10, strata = DESC_VICT)
```

Tune hyperparameters:

```
dt_tune = decision_tree(cost_complexity = tune(), min_n = tune()) %>%
    set_engine("rpart") %>% set_mode("classification")

dt_tune_wf = workflow() %>% add_recipe(rec) %>% add_model(dt_tune)

ti = Sys.time()

doParallel::registerDoParallel()
set.seed(345)
```

```
tree_grid = grid_regular(cost_complexity(c(-1, -1)), min_n(),
    levels = 32)
tree_grid[1] = -1
dt_tune_res = dt_tune_wf %>% tune_grid(resamples = cv_folds,
    grid = tree_grid)

tf = Sys.time()
model_tune_t = tf - ti
model_tune_t
```

Best tune:

```
best_auc = select_best(dt_tune_res, "roc_auc")
```

Final model:

```
final_dt = finalize_model(dt_tune, best_auc)

dt_best = workflow() %>% add_recipe(rec) %>% add_model(final_dt) %>%
    last_fit(split)
```

Prune tree:

```
model = pull_workflow_fit(dt_best$.workflow[[1]])
# plotcp(model$fit)
i = which.min(model$fit$cptable[, 4])[[1]]
xerror_best = model$fit$cptable[i, 4]
std_best = model$fit$cptable[i, 5]
table_best = as.data.frame(model$fit$cptable) %>% filter(CP >
    0) %>% filter(xerror <= xerror_best + std_best) %>% arrange(nsplit)

cp_prune = table_best[1, 1]
```

Fit pruned tree:

```
best_auc[1, 1] = cp_prune

final_dt_prune = finalize_model(dt_tune, best_auc)

dt_best_prune = workflow() %>% add_recipe(rec) %>% add_model(final_dt_prune) %>%
    last_fit(split)
```

Store model:

```
# save.image('models_dt_moredata/DT_2_down2_vi.RData')
```

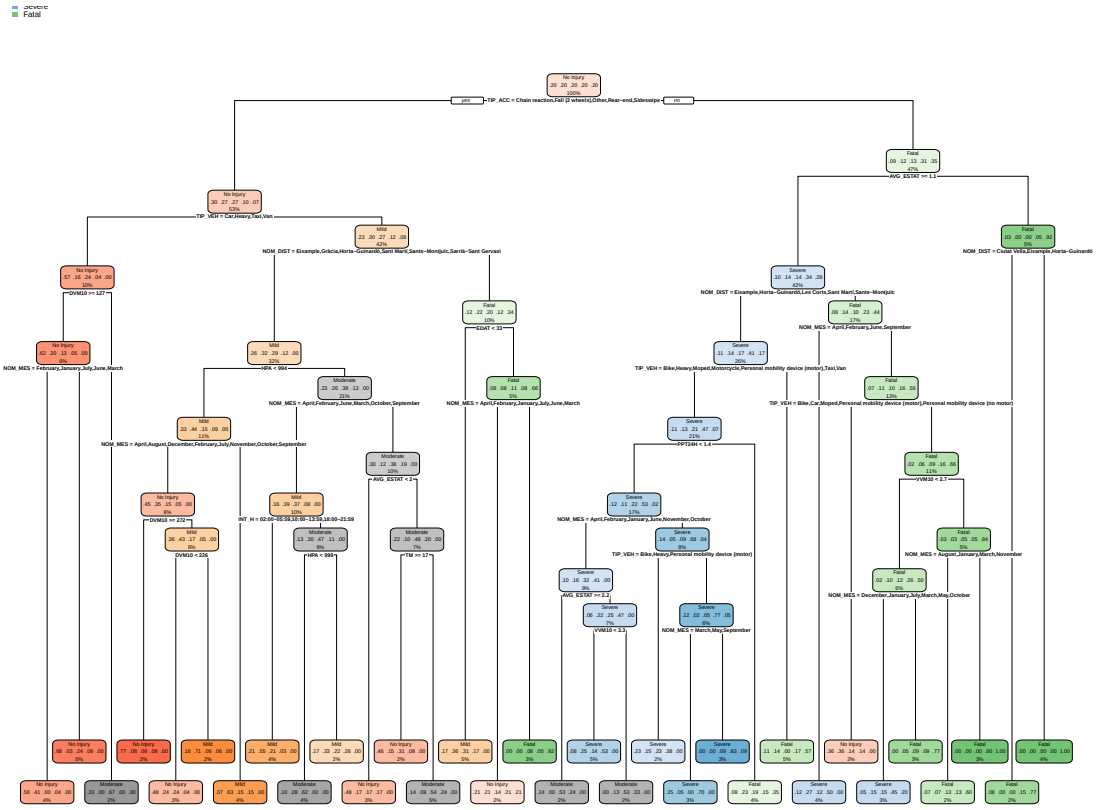
# 7. Classification Trees Results

## 7.1. 5-class Classifier

### 7.1.1. Grown tree

Tree:

```
load("models_dt_moredata/DT_5_down_vi.RData")

model = pull_workflow_fit(dt_best$.workflow[[1]])
rpart.plot(model$fit)
```




Performance:

```
conf = dt_best %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf[[1]] %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | No Injury | Mild | Moderate | Severe | Fatal |
|---|---|---|---|---|---|
| No Injury | 49 | 279 | 767 | 13 | 0 |
| Mild | 18 | 289 | 692 | 9 | 0 |
| Moderate | 18 | 155 | 410 | 7 | 0 |
| Severe | 12 | 151 | 456 | 33 | 4 |
| Fatal | 9 | 91 | 258 | 17 | 2 |

```
dt_best %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | multiclass | 0.2094143 | Preprocessor1_Model1 |
| roc_auc | hand_till | 0.6293038 | Preprocessor1_Model1 |

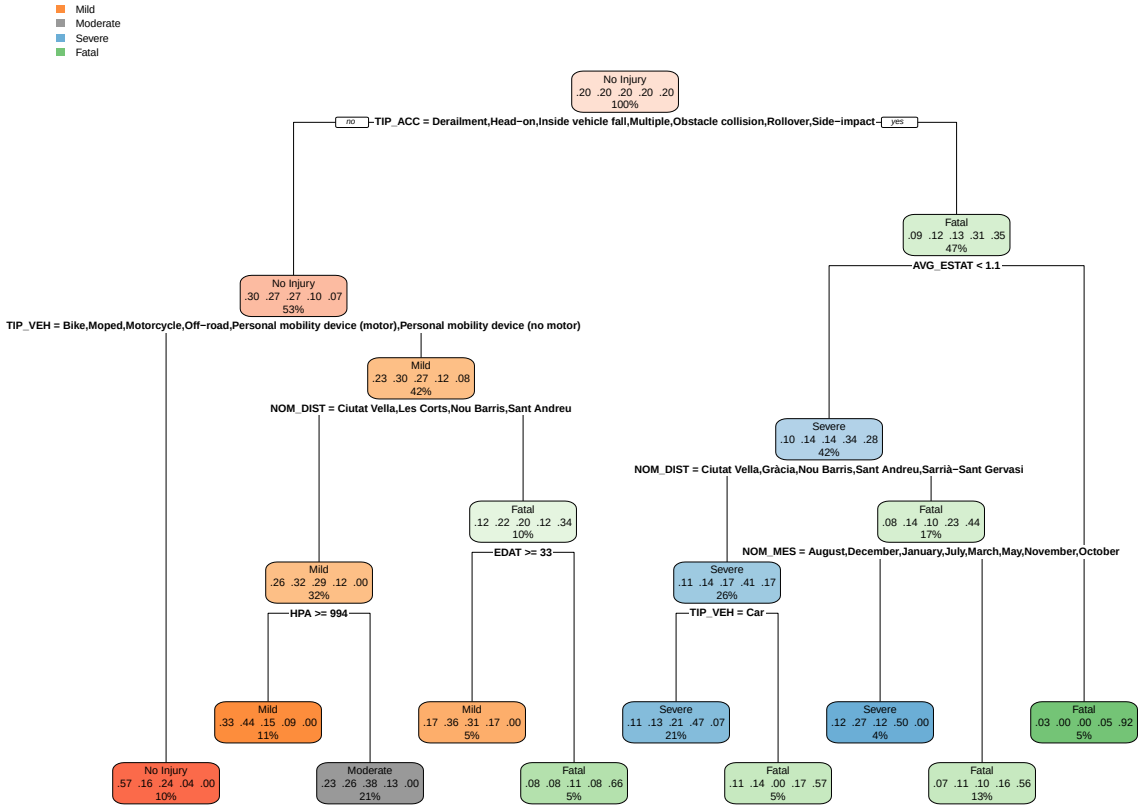
```
dt_best %>% collect_predictions() %>% conf_mat(DESC_VICT, .pred_class) %>%
    summary() %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | multiclass | 0.2094143 |
| kap | multiclass | 0.0229328 |
| sens | macro | 0.3343062 |
| spec | macro | 0.8045859 |
| ppv | macro | 0.2162911 |
| npv | macro | 0.8049272 |
| mcc | multiclass | 0.0312788 |
| j_index | macro | 0.1388921 |
| bal_accuracy | macro | 0.5694461 |
| detection_prevalence | macro | 0.2000000 |
| precision | macro | 0.2162911 |
| recall | macro | 0.3343062 |
| f_meas | macro | 0.1464700 |

### 7.1.2. Pruned tree

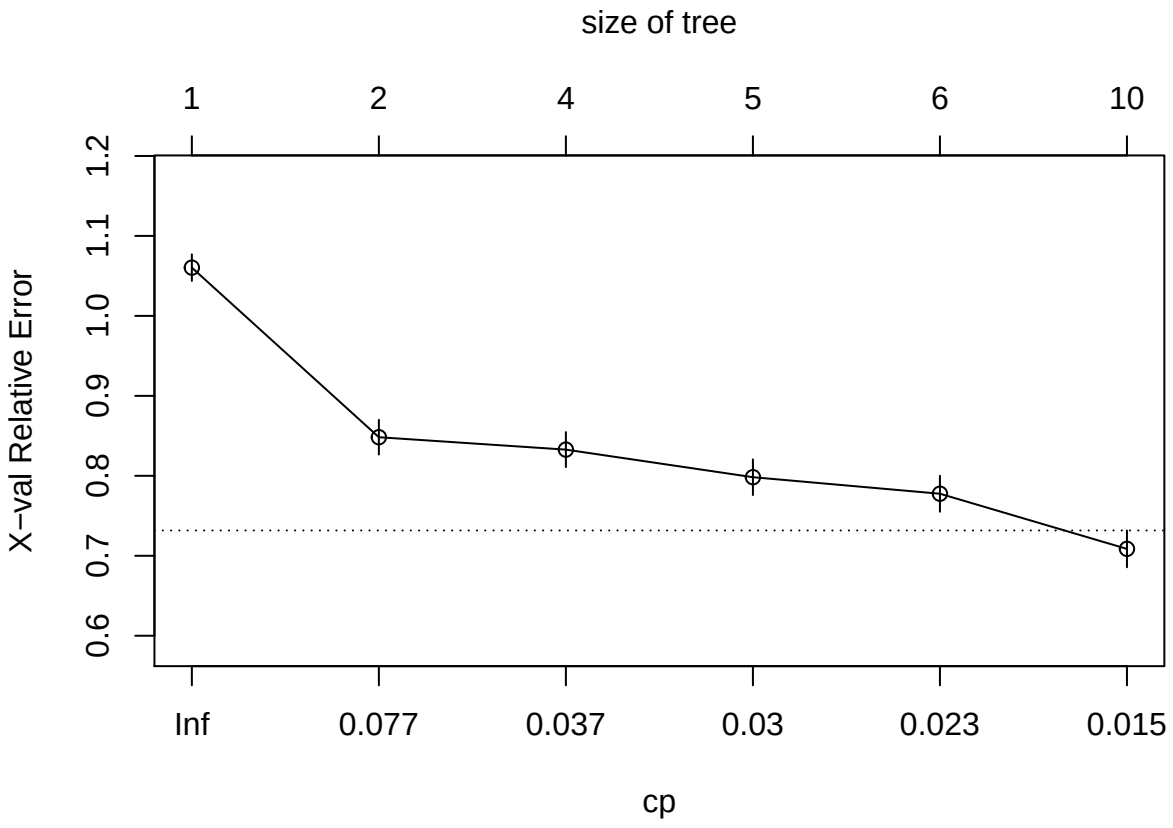
Tree:

```
model = pull_workflow_fit(dt_best_prune$.workflow[[1]])
rpart.plot(model$fit, left = F)
```




Performance:

```
plotcp(model$fit)
```




```
conf = dt_best_prune %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf[[1]] %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | No Injury | Mild | Moderate | Severe | Fatal |
|---|---|---|---|---|---|
| No Injury | 27 | 121 | 266 | 0 | 0 |
| Mild | 19 | 250 | 653 | 12 | 0 |
| Moderate | 32 | 281 | 731 | 9 | 0 |
| Severe | 16 | 194 | 586 | 41 | 4 |
| Fatal | 12 | 119 | 347 | 17 | 2 |

```
dt_best_prune %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | multiclass | 0.2810912 | Preprocessor1_Model1 |
| roc_auc | hand_till | 0.6545723 | Preprocessor1_Model1 |

```
dt_best_prune %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class) %>% summary() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

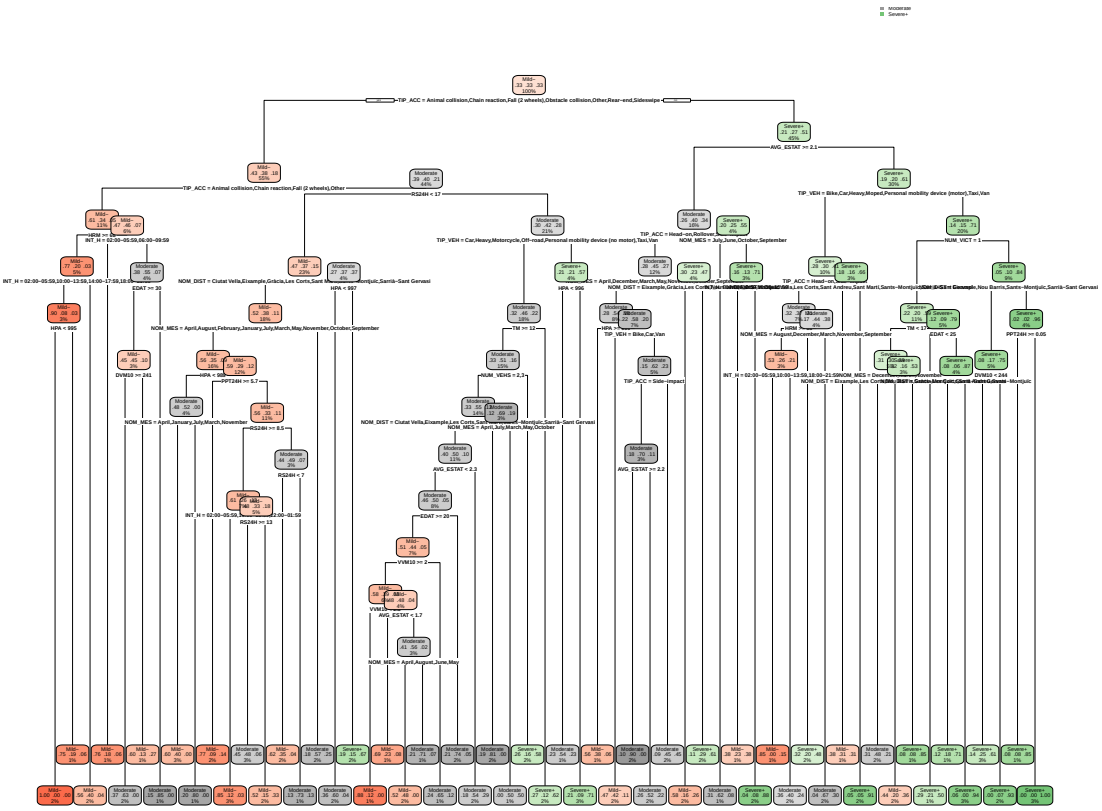
| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | multiclass | 0.2810912 |
| kap | multiclass | 0.0190496 |
| sens | macro | 0.3298219 |
| spec | macro | 0.8034348 |
| ppv | macro | 0.2159732 |
| npv | macro | 0.8034982 |
| mcc | multiclass | 0.0234711 |
| j_index | macro | 0.1332567 |
| bal_accuracy | macro | 0.5666283 |
| detection_prevalence | macro | 0.2000000 |
| precision | macro | 0.2159732 |
| recall | macro | 0.3298219 |
| f_meas | macro | 0.1732631 |

## 7.2. 3-class Classifier 1

### 7.2.1. Grown tree

Tree:

```
load("models_dt_moredata/DT_3_down_vi.RData")

model = pull_workflow_fit(dt_best$.workflow[[1]])
rpart.plot(model$fit)
```



Performance:

```
conf = dt_best %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf[[1]] %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | Mild- | Moderate | Severe+ |
|---|---|---|---|
| Mild- | 448 | 1038 | 36 |
| Moderate | 484 | 1121 | 20 |
| Severe+ | 134 | 423 | 35 |

```
dt_best %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | multiclass | 0.4289917 | Preprocessor1_Model1 |
| roc_auc | hand_till | 0.5922418 | Preprocessor1_Model1 |

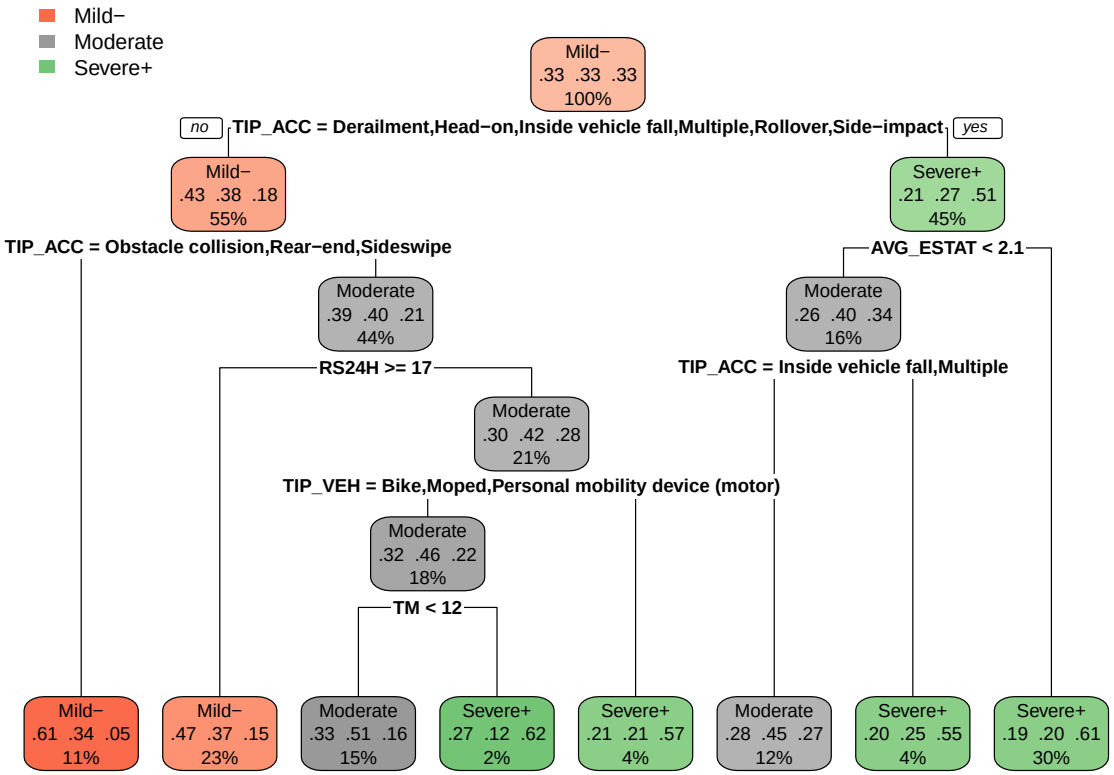
```
dt_best %>% collect_predictions() %>% conf_mat(DESC_VICT, .pred_class) %>%
    summary() %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | multiclass | 0.4289917 |
| kap | multiclass | 0.0154516 |
| sens | macro | 0.4130125 |
| spec | macro | 0.6699695 |
| ppv | macro | 0.3477724 |
| npv | macro | 0.6707811 |
| mcc | multiclass | 0.0171284 |
| j_index | macro | 0.0829820 |
| bal_accuracy | macro | 0.5414910 |
| detection_prevalence | macro | 0.3333333 |
| precision | macro | 0.3477724 |
| recall | macro | 0.4130125 |
| f_meas | macro | 0.3272079 |

### 7.2.2. Pruned tree

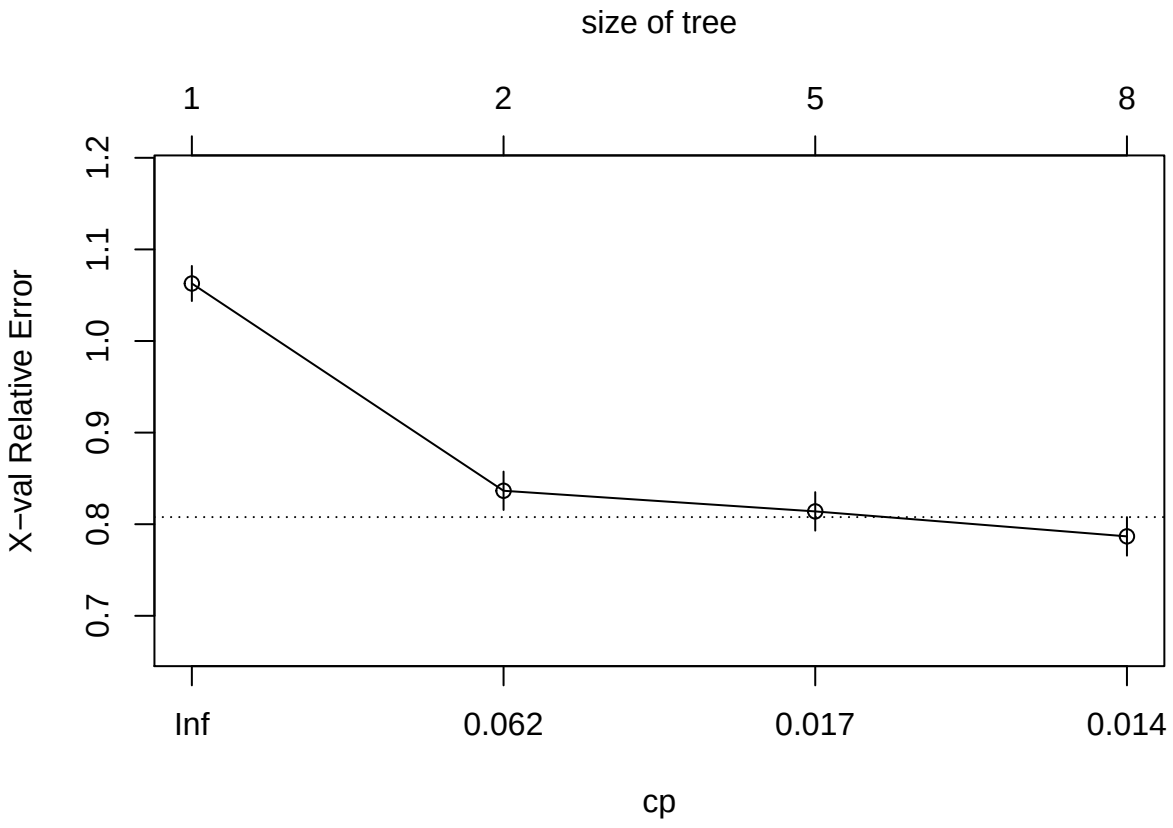
Tree:

```
model = pull_workflow_fit(dt_best_prune$.workflow[[1]])
rpart.plot(model$fit, left = F)
```




Performance:

```
plotcp(model$fit)
```




```
conf = dt_best_prune %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf[[1]] %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | Mild- | Moderate | Severe+ |
|---|---|---|---|
| Mild- | 491 | 1129 | 20 |
| Moderate | 349 | 780 | 23 |
| Severe+ | 226 | 673 | 48 |

```
dt_best_prune %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | multiclass | 0.3527681 | Preprocessor1_Model1 |
| roc_auc | hand_till | 0.6209309 | Preprocessor1_Model1 |

```
dt_best_prune %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class) %>% summary() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

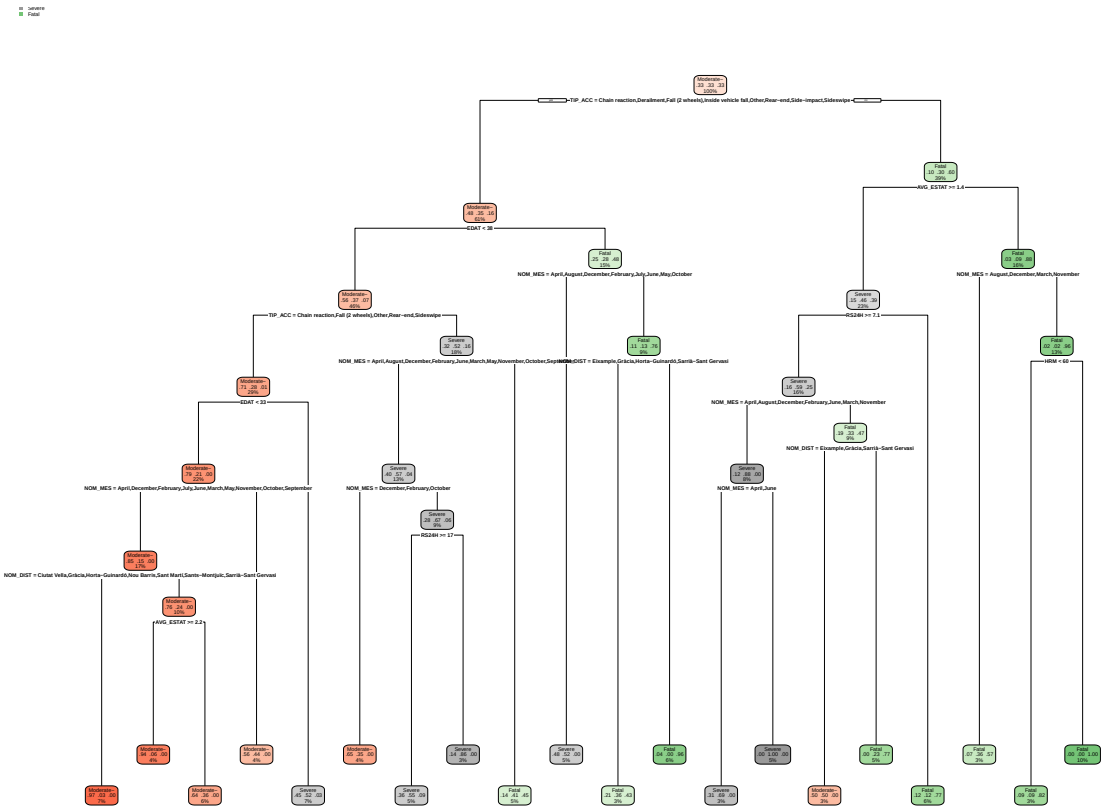
| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | multiclass | 0.3527681 |
| kap | multiclass | 0.0133962 |
| sens | macro | 0.4300548 |
| spec | macro | 0.6673961 |
| ppv | macro | 0.3423867 |
| npv | macro | 0.6713664 |
| mcc | multiclass | 0.0164282 |
| j_index | macro | 0.0974509 |
| bal_accuracy | macro | 0.5487254 |
| detection_prevalence | macro | 0.3333333 |
| precision | macro | 0.3423867 |
| recall | macro | 0.4300548 |
| f_meas | macro | 0.2910551 |

## 7.3. 3-class Classifier 2

### 7.3.1. Grown tree

Tree:

```
load("models_dt_moredata/DT_3_down2_vi.RData")

model = pull_workflow_fit(dt_best$.workflow[[1]])
rpart.plot(model$fit)
```




Performance:

```
conf = dt_best %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf[[1]] %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | Moderate- | Severe | Fatal |
|---|---|---|---|
| Moderate- | 1894 | 17 | 1 |
| Severe | 1265 | 28 | 2 |
| Fatal | 503 | 25 | 4 |

```
dt_best %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | multiclass | 0.5151110 | Preprocessor1_Model1 |
| roc_auc | hand_till | 0.6690622 | Preprocessor1_Model1 |

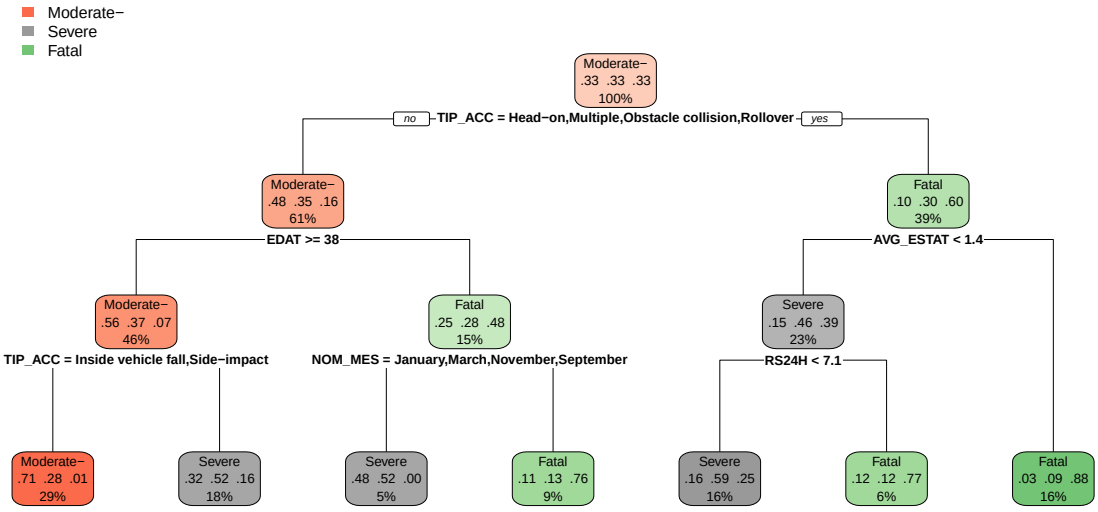
```
dt_best %>% collect_predictions() %>% conf_mat(DESC_VICT, .pred_class) %>%
    summary() %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | multiclass | 0.5151110 |
| kap | multiclass | 0.0152812 |
| sens | macro | 0.4962108 |
| spec | macro | 0.7598097 |
| ppv | macro | 0.3399087 |
| npv | macro | 0.6713910 |
| mcc | multiclass | 0.0483935 |
| j_index | macro | 0.2560204 |
| bal_accuracy | macro | 0.6280102 |
| detection_prevalence | macro | 0.3333333 |
| precision | macro | 0.3399087 |
| recall | macro | 0.4962108 |
| f_meas | macro | 0.2451506 |

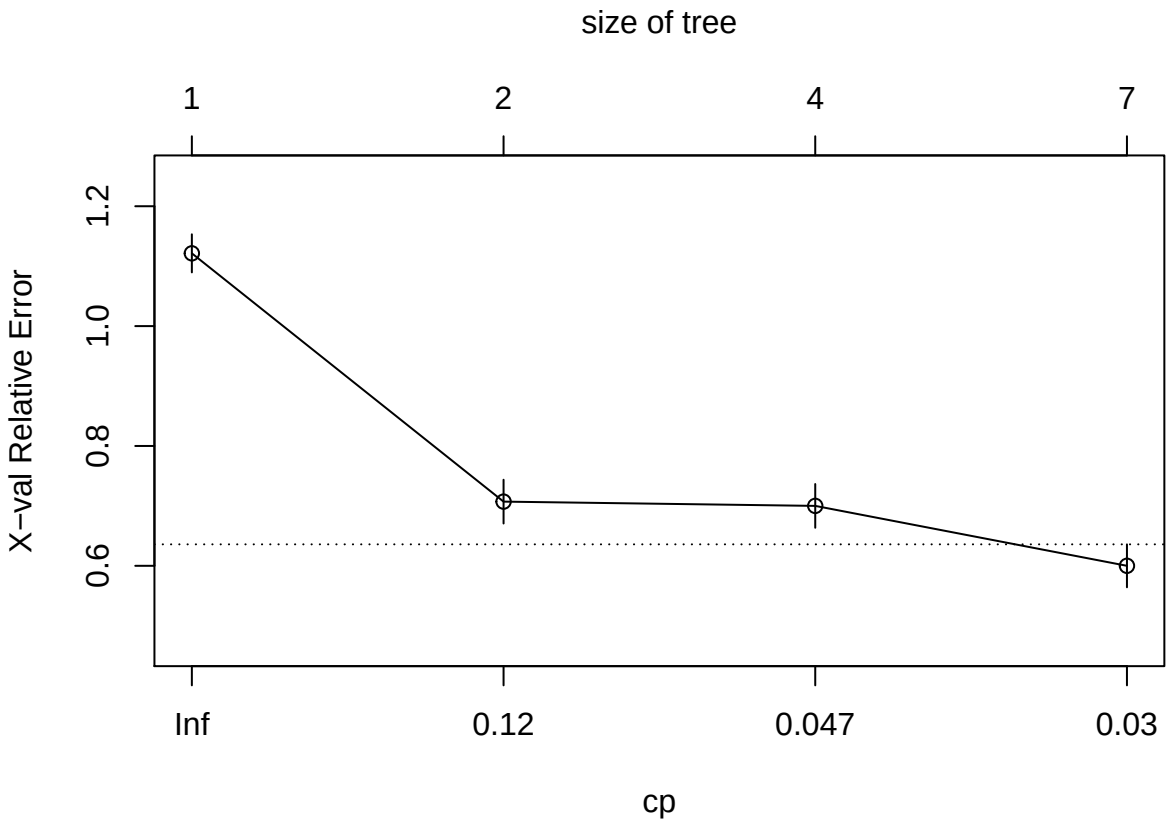
### 7.3.2. Pruned tree

Tree:

```
model = pull_workflow_fit(dt_best_prune$.workflow[[1]])
rpart.plot(model$fit, left = F)
```




```
plotcp(model$fit)
```




```
conf = dt_best_prune %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf[[1]] %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | Moderate- | Severe | Fatal |
|---|---|---|---|
| Moderate- | 1934 | 10 | 1 |
| Severe | 1385 | 46 | 2 |
| Fatal | 343 | 14 | 4 |

```
dt_best_prune %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | multiclass | 0.5306232 | Preprocessor1_Model1 |
| roc_auc | hand_till | 0.7426524 | Preprocessor1_Model1 |

```
dt_best_prune %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class) %>% summary() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

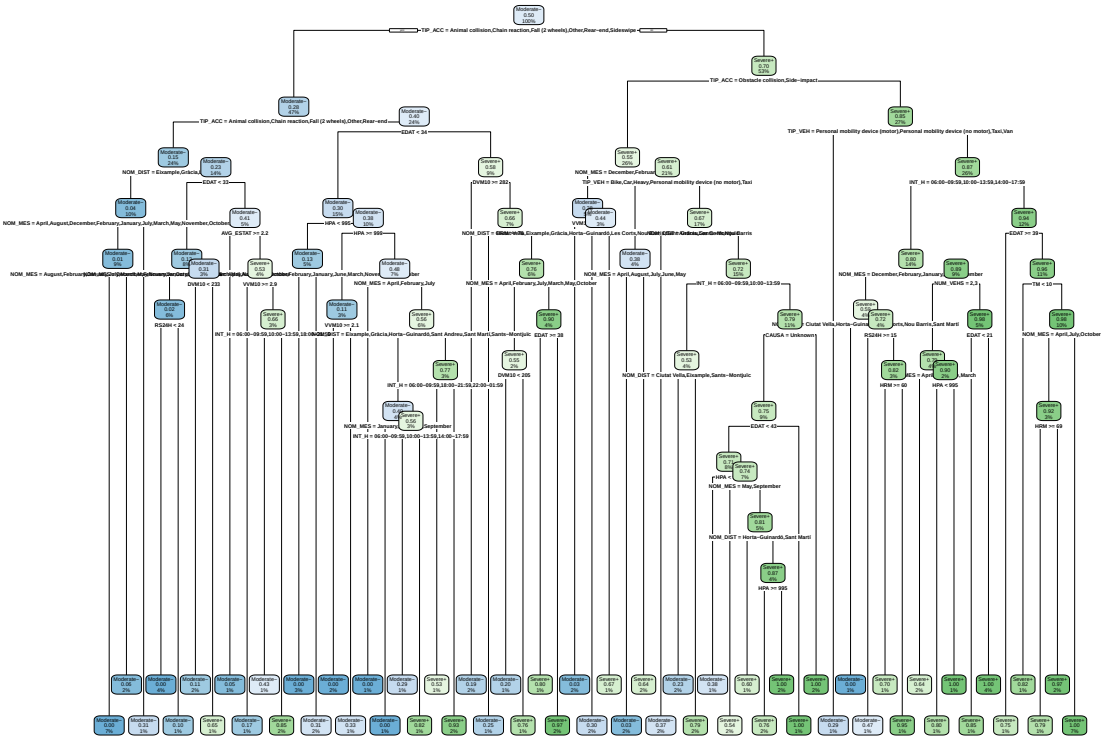
| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | multiclass | 0.5306232 |
| kap | multiclass | 0.0285356 |
| sens | macro | 0.5855660 |
| spec | macro | 0.7944838 |
| ppv | macro | 0.3458418 |
| npv | macro | 0.6751645 |
| mcc | multiclass | 0.0905921 |
| j_index | macro | 0.3800499 |
| bal_accuracy | macro | 0.6900249 |
| detection_prevalence | macro | 0.3333333 |
| precision | macro | 0.3458418 |
| recall | macro | 0.5855660 |
| f_meas | macro | 0.2576007 |

## 7.4. Binary Classifier 1

### 7.4.1. Grown tree

Tree:

```
load("models_dt_moredata/DT_2_down_vi.RData")

model = pull_workflow_fit(dt_best$.workflow[[1]])
rpart.plot(model$fit)
```



Performance:

```
conf = dt_best %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf[[1]] %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | Moderate- | Severe+ |
|---|---|---|
| Moderate- | 2697 | 30 |
| Severe+ | 965 | 47 |

```
dt_best %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | binary | 0.7338861 | Preprocessor1_Model1 |
| roc_auc | binary | 0.6929522 | Preprocessor1_Model1 |

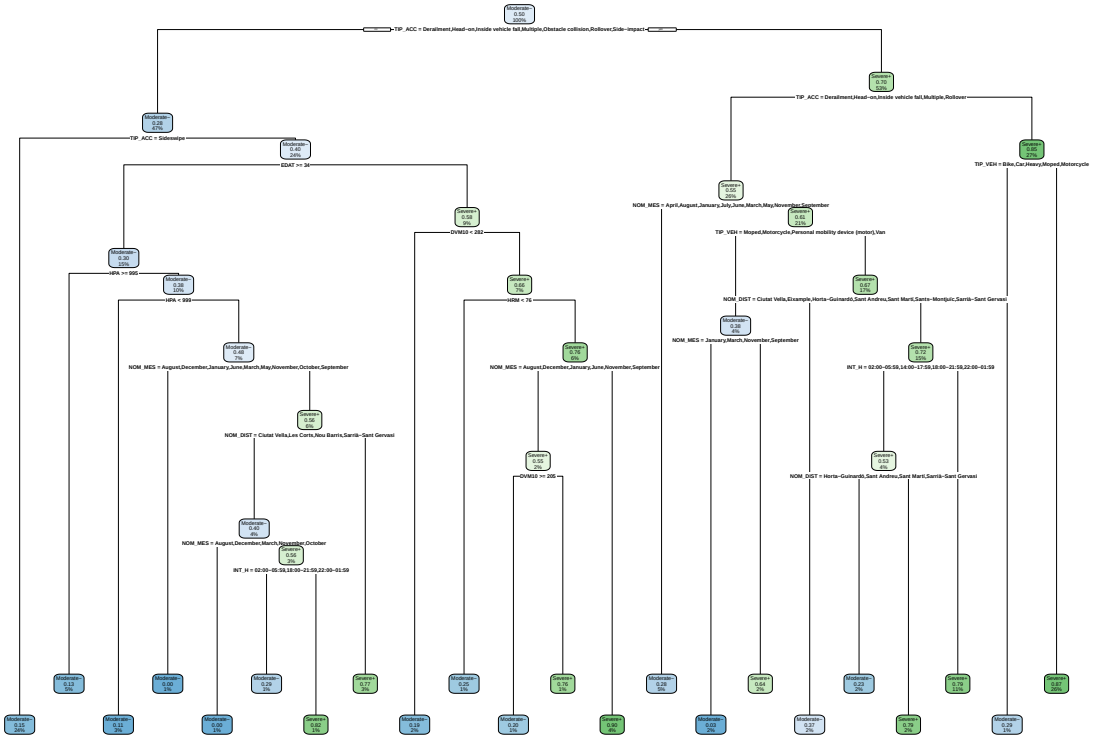
```
dt_best %>% collect_predictions() %>% conf_mat(DESC_VICT, .pred_class) %>%
    summary() %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | binary | 0.7338861 |
| kap | binary | 0.0499545 |
| sens | binary | 0.7364828 |
| spec | binary | 0.6103896 |
| ppv | binary | 0.9889989 |
| npv | binary | 0.0464427 |
| mcc | binary | 0.1108770 |
| j_index | binary | 0.3468724 |
| bal_accuracy | binary | 0.6734362 |
| detection_prevalence | binary | 0.7293394 |
| precision | binary | 0.9889989 |
| recall | binary | 0.7364828 |
| f_meas | binary | 0.8442636 |

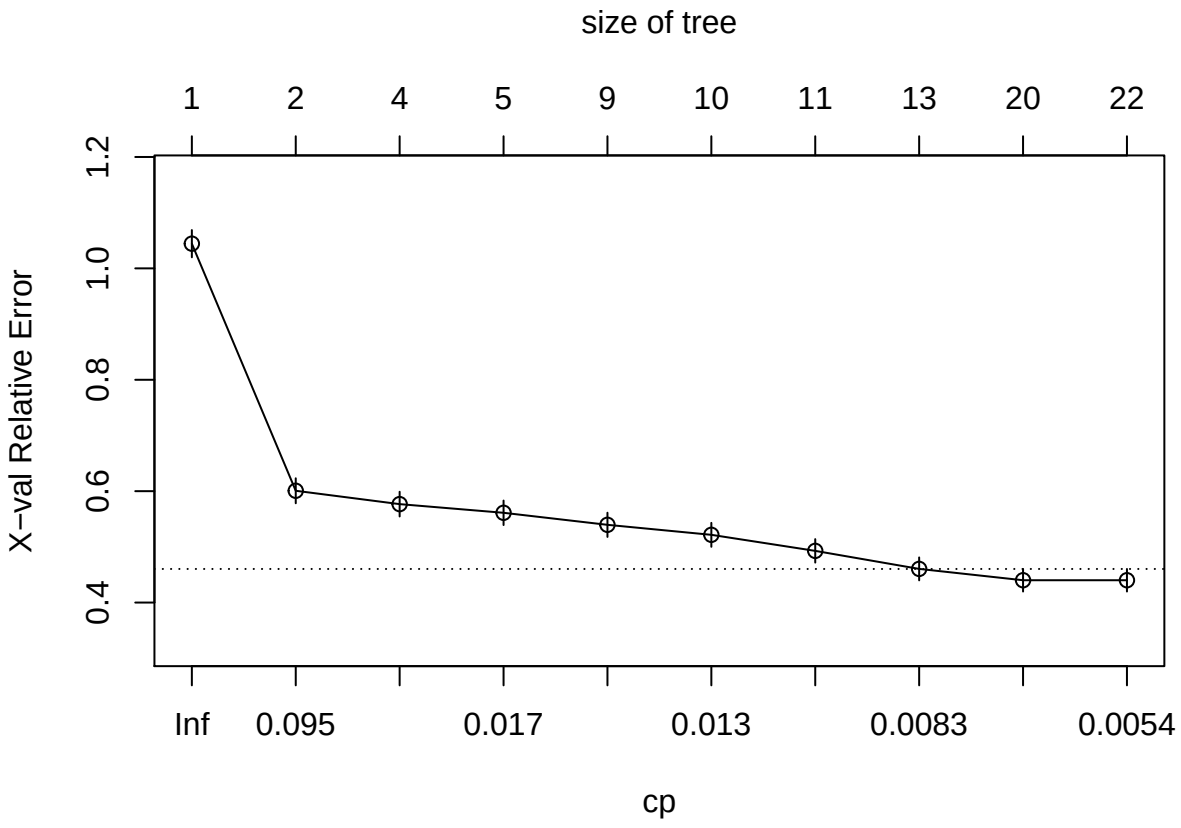
### 7.4.2. Pruned tree

Tree:

```
model = pull_workflow_fit(dt_best_prune$.workflow[[1]])
rpart.plot(model$fit, left = F)
```



```
plotcp(model$fit)
```




```
conf = dt_best_prune %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf[[1]] %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | Moderate- | Severe+ |
|---|---|---|
| Moderate- | 2750 | 29 |
| Severe+ | 912 | 48 |

```
dt_best_prune %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | binary | 0.7483284 | Preprocessor1_Model1 |
| roc_auc | binary | 0.7232706 | Preprocessor1_Model1 |

```
dt_best_prune %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class) %>% summary() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

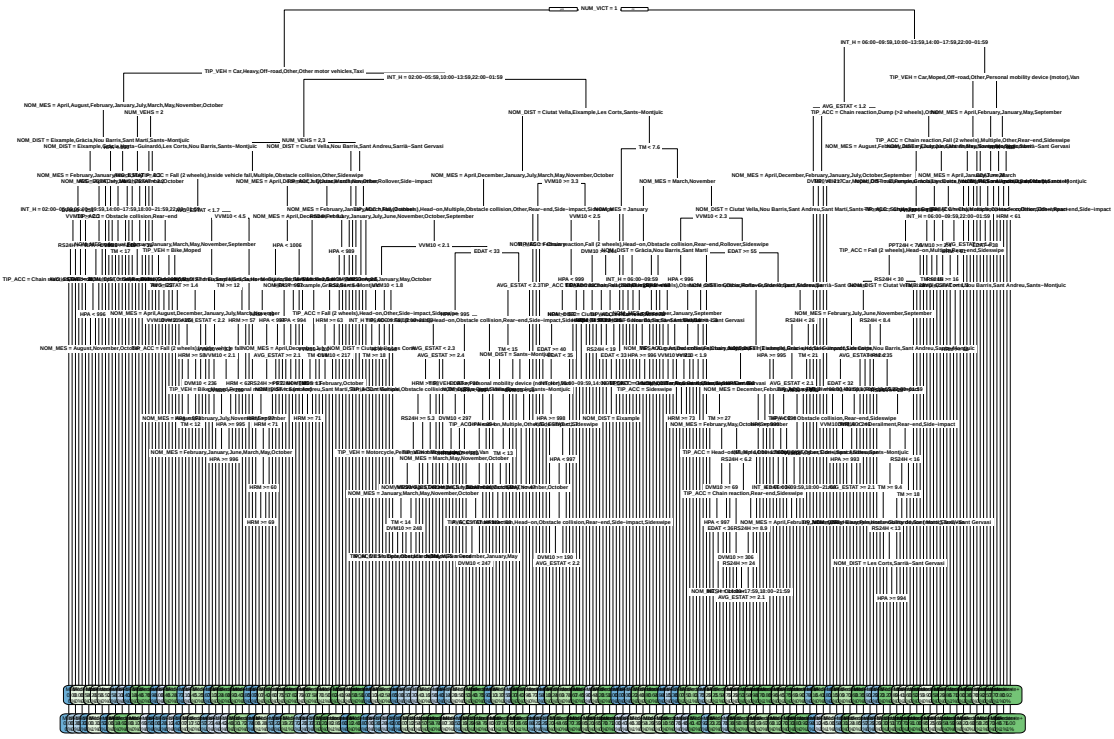
| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | binary | 0.7483284 |
| kap | binary | 0.0566038 |
| sens | binary | 0.7509558 |
| spec | binary | 0.6233766 |
| ppv | binary | 0.9895646 |
| npv | binary | 0.0500000 |
| mcc | binary | 0.1216976 |
| j_index | binary | 0.3743324 |
| bal_accuracy | binary | 0.6871662 |
| detection_prevalence | binary | 0.7432469 |
| precision | binary | 0.9895646 |
| recall | binary | 0.7509558 |
| f_meas | binary | 0.8539047 |

## 7.5. Binary Classifier 2

### 7.5.1. Grown tree

Tree:

```
load("models_dt_moredata/DT_2_down2_vi.RData")

model = pull_workflow_fit(dt_best$.workflow[[1]])
rpart.plot(model$fit, type = 0)
```



Performance:

```
conf = dt_best %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf[[1]] %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | Mild- | Moderate+ |
|---|---|---|
| Mild- | 512 | 1343 |
| Moderate+ | 551 | 1333 |

```
dt_best %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | binary | 0.4934474 | Preprocessor1_Model1 |
| roc_auc | binary | 0.4908120 | Preprocessor1_Model1 |

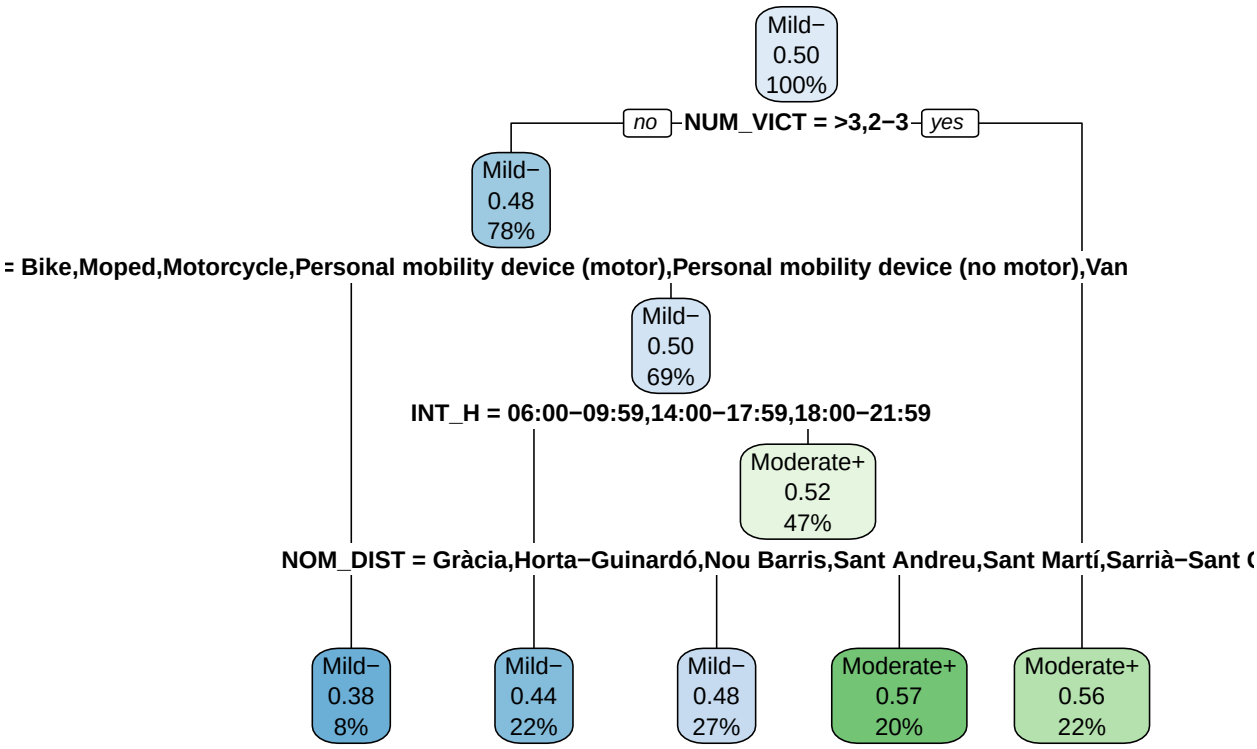
```
dt_best %>% collect_predictions() %>% conf_mat(DESC_VICT, .pred_class) %>%
    summary() %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | binary | 0.4934474 |
| kap | binary | -0.0165063 |
| sens | binary | 0.4816557 |
| spec | binary | 0.4981315 |
| ppv | binary | 0.2760108 |
| npv | binary | 0.7075372 |
| mcc | binary | -0.0182357 |
| j_index | binary | -0.0202128 |
| bal_accuracy | binary | 0.4898936 |
| detection_prevalence | binary | 0.4961220 |
| precision | binary | 0.2760108 |
| recall | binary | 0.4816557 |
| f_meas | binary | 0.3509253 |

### 7.5.2. Pruned tree

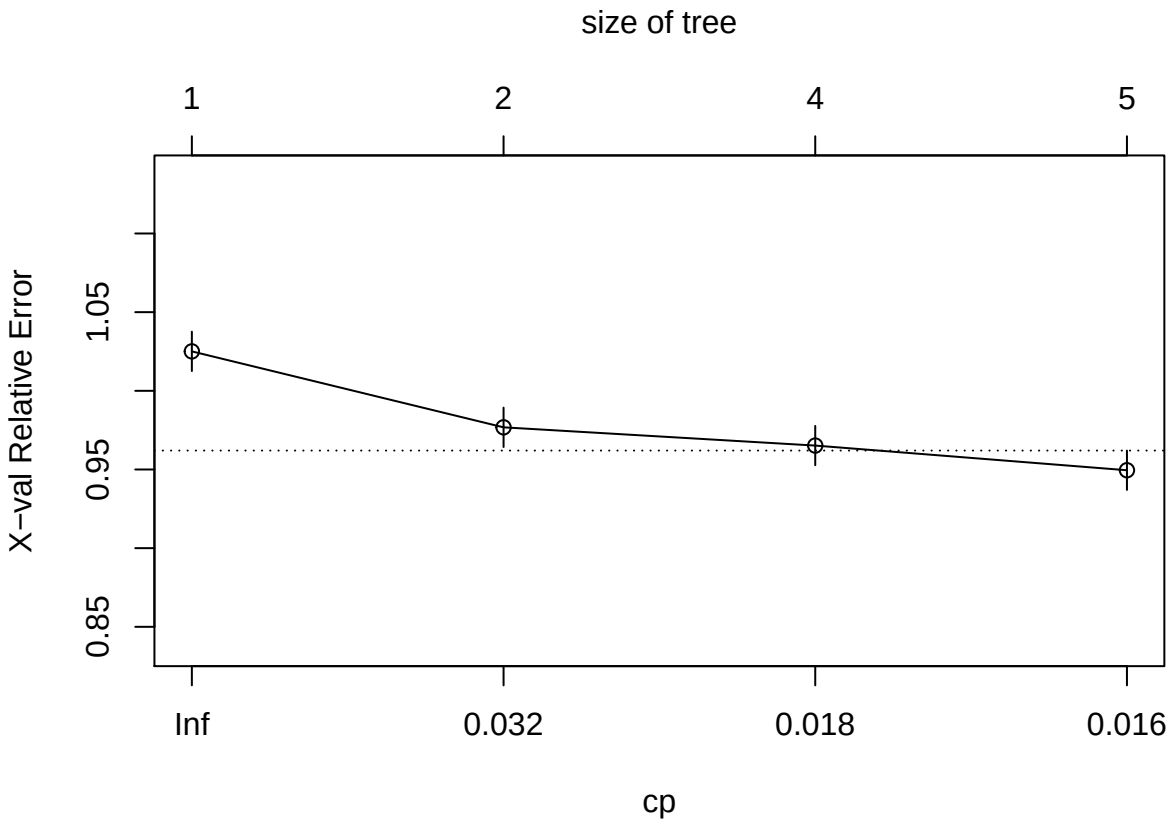
Tree:

```
model = pull_workflow_fit(dt_best_prune$.workflow[[1]])
rpart.plot(model$fit, left = F)
```




Performance:

```
plotcp(model$fit)
```




```
conf = dt_best_prune %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class)

conf[[1]] %>% kable() %>% kable_styling(full_width = F, font_size = 15)
```

| | Mild- | Moderate+ |
|---|---|---|
| Mild- | 659 | 1438 |
| Moderate+ | 404 | 1238 |

```
dt_best_prune %>% collect_metrics() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | binary | 0.5073549 | Preprocessor1_Model1 |
| roc_auc | binary | 0.5534633 | Preprocessor1_Model1 |

```
dt_best_prune %>% collect_predictions() %>% conf_mat(DESC_VICT,
    .pred_class) %>% summary() %>% kable() %>% kable_styling(full_width = F,
    font_size = 15)
```

| .metric | .estimator | .estimate |
|---|---|---|
| accuracy | binary | 0.5073549 |
| kap | binary | 0.0638547 |
| sens | binary | 0.6199436 |
| spec | binary | 0.4626308 |
| ppv | binary | 0.3142585 |
| npv | binary | 0.7539586 |
| mcc | binary | 0.0750532 |
| j_index | binary | 0.0825743 |
| bal_accuracy | binary | 0.5412872 |
| detection_prevalence | binary | 0.5608451 |
| precision | binary | 0.3142585 |
| recall | binary | 0.6199436 |
| f_meas | binary | 0.4170886 |