Traffic Accident Analysis for Germany

Matthias Frye

2024-02-19

Contents

1 Introduction

This project was conducted as part of the HarvardX Data Science Capstone Module. The primary goal is to perform a predictive analysis of traffic accident that leverages data for Germany provided by the local government Landesbetrieb Information und Technik Nordrhein-Westfalen under the "Data License Germany – Attribution – Version 2.0".

The database comprises accidents, complete with geo location, local conditions, involved vehicles, accident type, and severity, specifically focusing on accidents involving injuries or fatalities. The overarching objective is to construct a predictive model for the severity of accidents to generate insight into common types and causes of severe accidents.

Recognizing that the accident data alone lacks strong predictive power, considerable effort is invested in integrating additional data with the accident data. Population information, geographic and historical weather data, and street information are retrieved and merged with the dataset, introducing additional variables for the predictive analysis.

To construct a comprehensive model, various modeling techniques, such as oversampling, linear modeling, singular value decomposition, random forest, and extreme gradient boosting are combined. This approach aims to enhance the predictive capabilities and provide a more nuanced understanding of accident severity.

2 Method

This chapter outlines the data preparation process, the construction of models, and the subsequent evaluation of the models, an idea inspired by Ahmed et al. (2023), who conducted a traffic accident analysis for New Zealand.

2.1 Overview

The development of the model is done in three main steps.

1. Data Load, Analysis and Preparation

The traffic accident, population, geographic, weather and street datasets are loaded from the files and combined, initiating a sequence of straightforward analyses aimed at comprehending the dataset. At last, the data is partitioned into model and final test datasets. The model dataset is further partitioned into a train and test dataset.

2. Model Development

Multiple prediction models are constructed and tested using subsets of the model data. The progression involves enhancing the approach step by step:

- Initially, a simple model based on the most frequent severity is established.
- Secondly, a simple linear model is created utilizing all available features.
- Next, the linear model is enhanced using oversampling of severe accidents.
- The linear model is further enhanced using singular value decomposition to recognize spatial patterns.
- In the next iteration, separate linear models for different vehicle types are built.
- Subsequently, a model based on the random forest approach is created.
- In the next step, a model using extreme gradient boosting is built.
- In the last step, the three latter models are combined in an ensemble approach.

3. Model Analysis

After the development of the models, an analysis of the effect sizes of the different features is conducted to gain insights into the types and causes of severe accidents. This aspect potentially stands out as the models' most significant contribution, as understanding the nature of severe accidents can offer crucial insights for preventive measures.

4. Model Testing

Model testing was executed using a distinct subset of the data, exclusively reserved for this purpose.

2.2 Data Load, Analysis and Preparation

The construction process starts with the loading, analyzing and preparing the data. Initially, required libraries are loaded:

- *readr* for reading text files.
- *tidyverse* is a collection of packages for data representation and management.
- *caret* provides a number of functions for classification and regression.
- *ggthemes* offers a number of nice looking graph themes for *ggplot*.
- *scales* provides functions for formatting axes in graphs.
- *grid* for combining several graphs into grids.
- *gridExtra* for adding title.
- *sf* provides spatial functions.
- *rnaturalearth* provides map data.
- *geodata* also provides map data.
- *rdwd* provides access to weather data from Deutscher Wetterdienst (DWD).
- *readxl* for reading Excel files.
- *randomForest* for random forest models.
- *RSpectra* for faster singular value decomposition.
- *Matrix* for matrix calculations.
- *data.table* for fast file reads *fread*.
- *xgboost* for extreme gradient boosting.
- *doParallel* enables and controls parallel processing.

```
library(readr)
library(tidyverse)
library(caret)
library(ggthemes)
library(scales)
library(gridExtra)
library(grid)
library(sf)
library(rnaturalearth)
library(geodata)
library(rdwd)
library(readxl)
library(randomForest)
library(RSpectra)
library(Matrix)
library(data.table)
library(xgboost)
library(doParallel)
```

2.2.1 Data Load

Subsequently, the traffic accident, population, geographic and weather data is loaded.

Traffic Accident Data

Subsequently, traffic accident data is retrieved from the NRW State Agency for Information and Technology Landesbetrieb Information und Technik Nordrhein-Westfalen.

Traffic accident data covering all German states is accessible for the years 2020, 2021, and 2022. A comprehensive description of the records is provided in English.

The data is acquired in three files and undergoes transformation for enhanced readability. In the final stage, the population data is merged with the accident data.

```
# define file name, URL, folder
URL <- "https://www.opengeodata.nrw.de/produkte/transport_verkehr/unfallatlas/"
zip_files <- c( "Unfallorte2022_EPSG25832_CSV.zip",
                "Unfallorte2021_EPSG25832_CSV.zip",
                "Unfallorte2020_EPSG25832_CSV.zip")

files <- c( "Unfallorte2022_LinRef.csv",
            "Unfallorte2021_LinRef.csv",
            "Unfallorte2020_LinRef.csv")

folder <- "TrafficAccidents/"

# Check if the folder exists, if it doesn't exist, create the folder
if (!file.exists(folder)) {
  dir.create(folder)
}

# loop through all 3 files
for (i in 1:3) {

  # download zip file if not exists
  if(!file.exists(paste(folder, zip_files[i], sep = ""))
    download.file(paste(URL, zip_files[i], sep = ""),
                  paste(folder, zip_files[i], sep = ""))

  # unzip file
  if(!file.exists(paste(folder, files[i], sep = "")))
    unzip(paste(folder, zip_files[i], sep = ""),
          exdir = folder, junkpaths = TRUE)
}
```

```
# load year 2022 into accidents
accidents <- read_csv2(paste(folder, files[1], sep = ""),
                       show_col_types = FALSE)

# rename variable to match with other files
names(accidents)[names(accidents) == "IstStrassenzustand"] <- "USTRZUSTAND"

# load year 2021 into tmp table
accidents_tmp <- read_csv2(paste(folder, files[2], sep = ""),
```

```
                          show_col_types = FALSE)

# combine with previous years
accidents <- rbind(accidents , accidents_tmp)

# load year 2020 into tmp table
accidents_tmp <- read_csv2(paste(folder, files[3], sep = ""),
                          show_col_types = FALSE)

# rename variable to match
names(accidents_tmp)[names(accidents_tmp) == "STRZUSTAND"] <- "USTRZUSTAND"

# combine with previous years
accidents <- rbind(accidents , accidents_tmp)

# translate to English column names
names(accidents)[names(accidents) == "OBJECTID"] <- "id"
names(accidents)[names(accidents) == "ULAND"] <- "state"
names(accidents)[names(accidents) == "UREGBEZ"] <- "region"
names(accidents)[names(accidents) == "UKREIS"] <- "district"
names(accidents)[names(accidents) == "UGEMEINDE"] <- "municipality"
names(accidents)[names(accidents) == "UJAHR"] <- "year"
names(accidents)[names(accidents) == "UMONAT"] <- "month"
names(accidents)[names(accidents) == "USTUNDE"] <- "hour"
names(accidents)[names(accidents) == "UWOCHENTAG"] <- "weekday"
names(accidents)[names(accidents) == "UKATEGORIE"] <- "severity_all"
names(accidents)[names(accidents) == "UART"] <- "collision_with"
names(accidents)[names(accidents) == "UTYP1"] <- "accident_type"
names(accidents)[names(accidents) == "ULICHTVERH"] <- "light_condition"
names(accidents)[names(accidents) == "USTRZUSTAND"] <- "road_condition"
names(accidents)[names(accidents) == "IstRad"] <- "with_bicycle"
names(accidents)[names(accidents) == "IstPKW"] <- "with_car"
names(accidents)[names(accidents) == "IstFuss"] <- "with_pedestrian"
names(accidents)[names(accidents) == "IstKrad"] <- "with_motorcycle"
names(accidents)[names(accidents) == "IstGkfz"] <- "with_truck"
names(accidents)[names(accidents) == "IstSonstige"] <- "with_other"
names(accidents)[names(accidents) == "XGCSWGS84"] <- "WGSX"
names(accidents)[names(accidents) == "YGCSWGS84"] <- "WGSY"

# convert columns into factors and calculate keys for joining other data
accidents <- accidents |>
  mutate(district_key = paste(sprintf("%02d", as.integer(state)),
                              region, district, sep = ""),
         state = factor(as.integer(state),
                        labels = c("Schleswig-Holstein",
                                   "Hamburg",
                                   "Niedersachsen",
                                   "Bremen",
                                   "Nordrhein-Westfalen",
                                   "Hessen",
                                   "Rheinland-Pfalz",
                                   "Baden-Württemberg",
                                   "Bayern",
```

```
                                "Saarland",
                                "Berlin",
                                "Brandenburg",
                                "Mecklenburg-Vorpommern",
                                "Sachsen",
                                "Sachsen-Anhalt",
                                "Thüringen")),
        region = factor(region),
        year = factor(year),
        month = factor(as.integer(month)),
        hour = factor(as.integer(hour)),
        weekday = factor(weekday,
                         labels = c("Su",
                                    "Mo",
                                    "Tu",
                                    "We",
                                    "Th",
                                    "Fr",
                                    "Sa")),
        severity = factor(ifelse(severity_all == 3, 0, 1),
                          levels = 0:1,
                          labels = c("light", "severe")),
        severity_num = as.integer(severity) - 1,
        collision_with = factor(collision_with,
                          labels= c("Other",
                                    "Vehicle starting, stopping or stationary ",
                                    "Vehicle ahead or waiting",
                                    "Vehicle in the same direction",
                                    "Oncoming vehicle",
                                    "Vehicle turning into or crossing a road",
                                    "Pedestrian",
                                    "Obstacle in the carriageway",
                                    "Off the road to the right",
                                    "Off the road to the left")),
        accident_type = factor(accident_type,
                         labels= c("Driving accident",
                                   "By turning off the road",
                                   "By turning into a road",
                                   "By crossing the road",
                                   "Involving stationary",
                                   "Vehicles moving in parallel",
                                   "Other")),
        light_condition = factor(light_condition,
                                 labels= c("Daylight",
                                           "Twilight",
                                           "Darkness")),
        road_condition = factor(road_condition,
                                labels= c("Dry",
                                          "Wet",
                                          "Slippery")))
```

Population Data

The Federal Statistical Office of Germany provides data for all politically independent municipalities with selected characteristics. Data published on 30/09/2023 (3rd quarter 2023) is used. Unfortunately, the description of the dataset is available in German only. The column names will be translated into English for better readability.

```
# define file name, URL, folder
URL <- paste("https://www.destatis.de/DE/Themen/Laender-Regionen/Regionales/",
             "Gemeindeverzeichnis/Administrativ/Archiv/GVAuszugQ/",
             "AuszugGV3QAktuell.xlsx?__blob=publicationFile",
             sep = "" )
file_name <- "AuszugGV3QAktuell.xlsx"

# download .xls file if not exists
if(!file.exists(paste(folder, file_name, sep = "")))
  download.file(URL, paste(folder, file_name, sep = ""))

# read file, ignore header and irrelevant columns
# for Germans: state = Bundesland, region = Regierungsbezirk,
# district = Kreis, municipality = Gemeinde
population <-
  read_excel(paste(folder, file_name, sep = ""),
             sheet = "Onlineprodukt_Gemeinden30092023",
             range="A6:J99999",
             col_names = c("record_type", "text_id", "state", "region", "district",
                           "i1", "i2", "i3", "area", "population"),
             col_types = c("text", "skip", "text", "text", "text",
                           "skip", "skip", "skip", "numeric","numeric"))

# we use record type 60, which holds population & area data, summarize by district
population <- population |>
  filter(record_type == "60") |>
  group_by(state, region, district) |>
  summarize(area = sum(area),
            population = sum(population),
            .groups = "keep") |>
  ungroup()

# calculate key for German districts (also used by geo_data package)
population <- population |>
  mutate(district_key = paste(sprintf("%02d", as.integer(state)),
                              region, district, sep = "")) |>
  select(district_key, population, area)
```

Geographic Data

Geographic data is acquired through the utilization of the *rnaturalearth* and *geodata* packages, enabling spatial processing and visualization of data through maps. The loading of data encompasses German states and districts. Small adjustments are necessary for district keys to facilitate joining with accident data.

```
# Load states with geometry
ger_states <- ne_states("Germany",returnclass="sf") |> select(-c(region))
```

```r
# Load districts
ger_districts <- gadm(country = "Germany", path = folder, level = 2)

# Convert into simple file
ger_districts <- sf::st_as_sf(ger_districts)

# rename district key
names(ger_districts)[names(ger_districts) == "CC_2"] <- "district_key"

# 1st adjustment: Berlin and Hamburg do not have several districts in gadm
accidents$district_key[accidents$state == "Berlin"] <- "11000"
accidents$district_key[accidents$state == "Hamburg"] <- "02000"

# 2nd adjustment: Recode Eisenach/Wartburgkreis before joining population
accidents$district_key[accidents$district_key == "16056"] <- "16063"

# join population and area data
accidents <- accidents |> left_join(population, by = "district_key")

# 3rd adjustment: Recode Göttingen after joining population
accidents$district_key[accidents$district_key == "03159"] <- "03152"

# code district_type as factor
ger_districts_tmp <- ger_districts |>
                        mutate(district_type = factor(TYPE_2,
                                                      labels= c("District",
                                                                "Large City",
                                                                "Rural District",
                                                                "Urban District",
                                                                "Water body"))) |>
                        select(district_key, district_type)

# join district_type to accidents
accidents <- accidents |> left_join(ger_districts_tmp, by = "district_key")
```

Weather Data

Deutscher Wetterdienst (DWD) provides diverse weather datasets on its ftp server, organized by data type, with separate files provided for each weather station and date.

The R package *rdwd* simplifies access to this data, and a good documentation is available. The package introduces three datasets—geoIndex, metaIndex, and fileIndex. Calculating the center point for each state, the nearest weather stations with a daily history of temperature and rainfall data for the period 2020-2022 are identified. Subsequently, the data files are downloaded and loaded into R.

Unfortunately, the accident data lacks precise dates, providing only the year, month, and weekday. As a workaround, we utilize the average rainfall and temperature for each state, year, month, and weekday to approximate the weather conditions at the time of the accident. In essence, the average rainfall and temperature of four (or five) days are merged with the accident data. This is regrettable and significantly diminishes the value of weather data for prediction. Otherwise, even hourly weather data would have been available for download on the DWD ftp server.

```r
# Find a middle point in each state
ger_points <- data.frame(st_coordinates(st_centroid(ger_states$geometry)))
```

```r
# load DWD data
data("geoIndex")
data("metaIndex")
data("fileIndex")

# which stations cover temp and rain for the period 2020 - 2022?
myIndex <- metaIndex |> filter(von_datum <= as.Date("2020-01-01") &
                                 bis_datum >= as.Date("2022-12-31") &
                                 var == "kl" &
                                 hasfile)

# which of these stations cover daily history of temperature and rain data?
fileIndex <- fileIndex |> filter(id %in% myIndex$Stations_id &
                                   res == "daily" &
                                   var == "kl" &
                                   per == "historical")

# remove redundant geo information of stations
geoIndex <- geoIndex |>
  group_by(id)  |>
  summarize(station_name = first(name),
            lon = first(lon),
            lat = first(lat)) |>
  ungroup()

# join to station/file info
fileIndex <- fileIndex |> left_join(geoIndex, by = "id")
```

A simple distance function calculates the Euclidean distance between two locations, acknowledging that this approach does not account for geodetic considerations. Nevertheless, it proves adequate for identifying the nearest station.

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

```r
distance <- function(x1, y1, x2, y2){
  return(sqrt((x2 - x1) ^ 2 + (y2 - y1) ^ 2))
}

# find the nearest station for each state
nearbyStation <- function(i){
  return( which.min(distance(fileIndex$lon, fileIndex$lat,
                             ger_points$X[i], ger_points$Y[i])))
}
finalIndex <- sapply(1:nrow(ger_points), nearbyStation)

# get list of file names
file_links <- fileIndex[finalIndex,"path"]

# download one file per state
localfiles <- dataDWD(file_links, joinbf = TRUE, sleep = 10,
                      read = FALSE, dir = folder, quiet = TRUE)

# function to load data file for one state
read2020_2022 <- function(state) {
```

```
  # get file name for state
  file <- localfiles[which(ger_states$name==state)]

  # read data
  out <- readDWD(file, quietread = TRUE)

  # filter on 2020-2022 and required variables,
  # TNK / TXK hold minimum/maximum air temperature degrees Celsius
  # RSK precipitation in mm
  # finally add state name
  out <- out |> filter(MESS_DATUM >= as.POSIXct(as.Date("2020-01-01")) &
                       MESS_DATUM <= as.POSIXct(as.Date("2022-12-31") )) |>
    select(STATIONS_ID, MESS_DATUM, TNK, TXK, RSK) |>
    mutate(state = state)
  return(out)
}

# load data files for all states
df_list <- lapply(ger_states$name, read2020_2022)

# combine several data frames into one large data frame
weather <- Reduce(function(...) merge(..., all = TRUE), df_list)

# rename columns in English
names(weather)[names(weather) == "STATIONS_ID"] <- "station_id"
names(weather)[names(weather) == "MESS_DATUM"] <- "weather_date"
names(weather)[names(weather) == "TNK"] <- "min_temp"
names(weather)[names(weather) == "TXK"] <- "max_temp"
names(weather)[names(weather) == "RSK"] <- "rain"

# calculate date variables
weather <- weather |> mutate(weather_date = date(weather_date),
                             year = factor(year(weather_date)),
                             month = factor(month(weather_date)),
                             weekday = match(weekdays(weather_date),
                                             c("Sunday", "Monday", "Tuesday", "Wednesday",
                                               "Thursday", "Friday", "Saturday")),
                             weekday = factor(weekday,
                                              labels= c("Su", "Mo", "Tu", "We",
                                                        "Th", "Fr", "Sa")))

# aggregate by state, year, month, weekday to match accident data
# this lacks precision, but is unavoidable
weather <- weather |> group_by(state, year, month, weekday) |>
                      summarize(min_temp = mean(min_temp, na.rm = TRUE),
                                max_temp = mean(max_temp, na.rm = TRUE),
                                rain = mean(rain, na.rm = TRUE),
                                weekday_count = n_distinct(weather_date),
                                .groups = "keep") |>
                      ungroup()

# join with accident data and remove unnecessary columns
accidents <- accidents  |>
```

```
  left_join(weather, by = c("state", "year", "month", "weekday")) |>
  select(-c(id, UIDENTSTLAE, region, district, municipality,
            severity_all, LINREFX, LINREFY, geometry))
```

Road Data

OpenStreetMap proves to be a valuable resource, offering comprehensive information about all roads in Germany. This includes details such as road category, the number of lanes, maximum speed, oneway restrictions, presence of sidewalks and cycleways, and whether the road is illuminated. OpenStreetMap data is provided under the Open Database License. Geofabrik provides data files for download.

The process of identifying roads where accidents occurred and retrieving associated information from OpenStreetMap proved to be computationally intensive. On the available computer, this task took more than 4 hours. Therefore, the information for each accident was pre-processed in a dedicated script to streamline the subsequent analyses. This approach ensures efficiency in handling the data and enables more expedited exploration and modeling. The script that retrieves and pre-processes the information has been made available on GitHub: *osm_match.R*. The resulting data is also available on GitHub: road_all1.csv and road_all2.csv In the current script, the data is loaded from GitHub, unzipped and joined with the accidents data.

```
URL <- "https://raw.githubusercontent.com/matthiasfrye/TrafficAccidents/main/"

file1 <- "road_all1.csv"
file2 <- "road_all2.csv"

# download file 1 if not exists
if(!file.exists(paste(folder, file1, sep = "")))
  download.file(paste(URL, file1, sep = ""), paste(folder, file1, sep = ""))

# download file 2 if not exists
if(!file.exists(paste(folder, file2, sep = "")))
  download.file(paste(URL, file2, sep = ""), paste(folder, file2, sep = ""))

# load first file
all_roads <- fread(paste(folder, file1, sep = ""))

# load second file
roads <- fread(paste(folder, file2, sep = ""))

# put both files together
all_roads <- rbind(all_roads, roads)

# rename column and remove WGSX and WGSY
names(all_roads)[names(all_roads) == "highway"] <- "road_category"

# create a key to join WGSX and WGSY
accidents$key <-
  paste(format(accidents$WGSX, digits = 13, scientific = FALSE, trim = TRUE),
        format(accidents$WGSY, digits = 13, scientific = FALSE, trim = TRUE),
        sep = "-")

# join road information and make surface a factor
accidents <- accidents  |>
  left_join(all_roads, by = "key")
```

```r
# make road variables a factor
accidents <- accidents |>
  mutate(road_category = factor(road_category),
         lit = factor(lit),
         oneway = factor(oneway),
         sidewalk = factor(sidewalk),
         cycleway = factor(cycleway)) |>
   select(-key)

rm(all_roads, roads)
```

2.2.2 Data Analysis

In the upcoming section, a comprehensive analysis of the data with a focus on accident volume and severity unfolds. To start, a few sample records will be showcased, and the structure of the data will be presented. This is followed by a series of graphs illustrating accident volume and severity by state, district type, collision with, accident type, involved vehicle types, temporal variables, road properties, and weather conditions.

Three maps of Germany are provided, highlighting the distribution of accidents, the number of accidents by population, and the distribution of accident severity.

Introduction

The following section introduces the accident data and shows the data structure and the summary statistics.

```r
# look into some records
substr(t(accidents[1:4,]),1,15)
```

```
##                 [,1]              [,2]              [,3]              [,4]
## state           "Schleswig-Holst" "Schleswig-Holst" "Schleswig-Holst" "Schleswig-Holst"
## year            "2022"            "2022"            "2022"            "2022"
## month           "2"               "5"               "5"               "5"
## hour            "19"              "11"              "12"              "8"
## weekday         "Fr"              "Su"              "Su"              "Tu"
## collision_with  "Vehicle turning" "Off the road to" "Vehicle turning" "Oncoming vehicl"
## accident_type   "By turning into" "Driving acciden" "By turning into" "Vehicles moving"
## light_condition "Darkness"        "Daylight"        "Daylight"        "Daylight"
## road_condition  "Wet"             "Dry"             "Dry"             "Dry"
## with_bicycle    "1"               "0"               "0"               "1"
## with_car        "1"               "0"               "1"               "0"
## with_pedestrian "0"               "0"               "0"               "0"
## with_motorcycle "0"               "1"               "0"               "0"
## with_truck      "0"               "0"               "0"               "0"
## with_other      "0"               "0"               "0"               "0"
## WGSX            " 9.093886"       "10.440636"       " 9.624949"       "10.672490"
## WGSY            "54.46340"        "54.26830"        "54.55599"        "53.87045"
## district_key    "01054"           "01057"           "01059"           "01003"
## severity        "light"           "severe"          "light"           "light"
## severity_num    "0"               "1"               "0"               "0"
## population      "169043"          "131266"          "206038"          "218095"
## area            "2083.53"         "1083.57"         "2071.33"         " 214.19"
## district_type   "District"        "District"        "District"        "Large City"
## min_temp        "1.375"           "5.900"           "5.900"           "7.420"
```

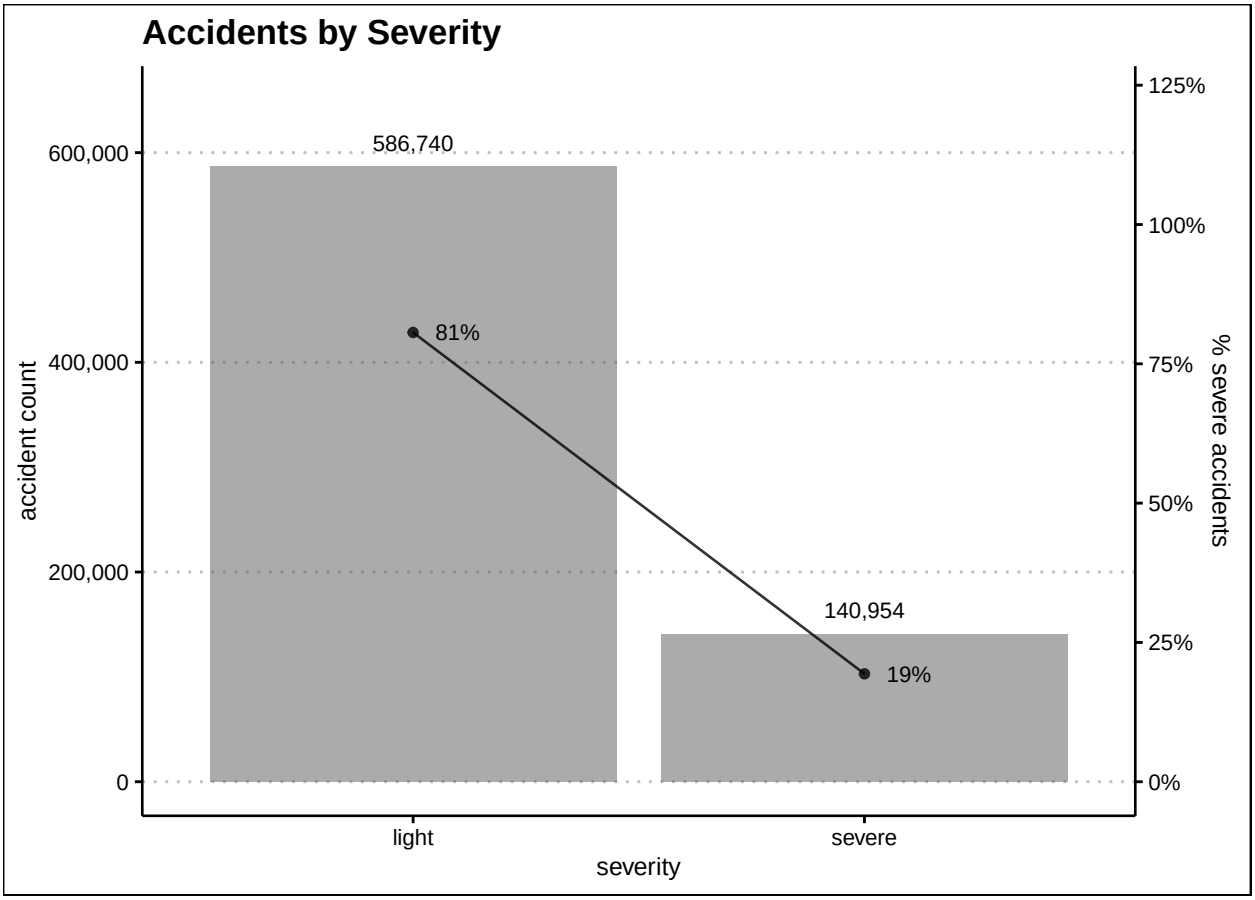
```
## max_temp        " 7.175"          "17.800"          "17.800"          "18.000"
## rain            "8.975"           "1.880"           "1.880"           "2.640"
## weekday_count   "4"               "5"               "5"               "5"
## road_category   "tertiary"        "secondary"       "primary"         "primary"
## lanes           "2"               "2"               "3"               "3"
## maxspeed        " 50"             "100"             "100"             " 50"
## lit             "unknown"         "unknown"         "unknown"         "yes"
## oneway          "unknown"         "unknown"         "unknown"         "no"
## sidewalk        "yes"             "yes"             "yes"             "yes"
## cycleway        "no"              "no"              "no"              "yes"
```

```
# show structure of the data
summary(accidents)
```

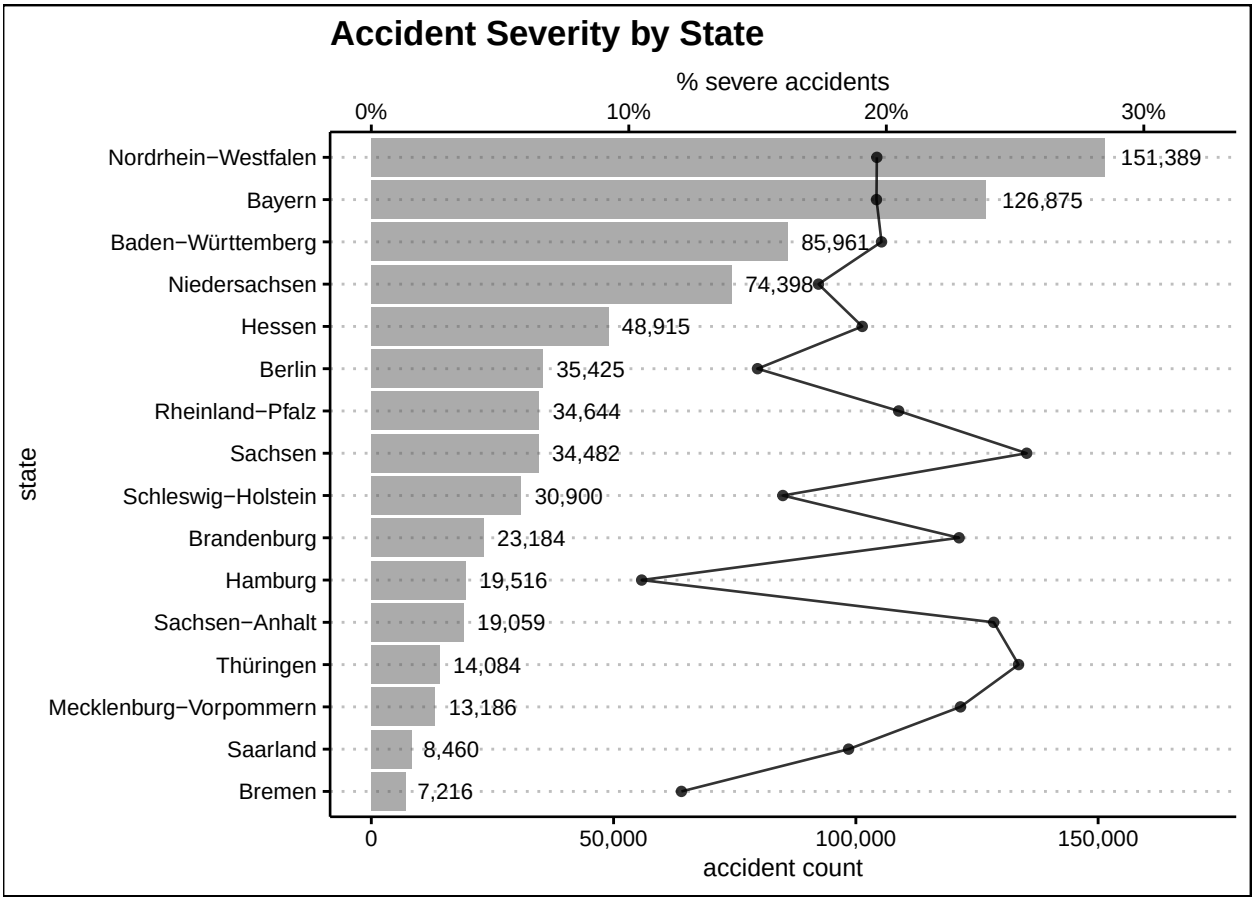
```
##     state              year            month             hour        weekday
##  Length:727694      2020:237994   9      : 77360   16     : 65544   Su: 67771   Vehicle turning into
##  Class :character   2021:233208   6      : 76189   15     : 63311   Mo:110078   Other
##  Mode  :character   2022:256492   7      : 74794   17     : 61557   Tu:114715   Vehicle ahead or wai
##                                   8      : 72156   14     : 58170   We:114038   Off the road to the
##                                   10     : 68802   13     : 54194   Th:115207   Pedestrian
##                                   5      : 64233   12     : 46811   Fr:118328   Oncoming vehicle
##                                   (Other):294160   (Other):378107   Sa: 87557   (Other)
##                      accident_type    light_condition    road_condition    with_bicycle       with_ca
##  Driving accident           :146715   Daylight:554795   Dry     :546458   Min.   :0.0000   Min.   :0
##  By turning off the road    :100263   Twilight: 39197   Wet     :166357   1st Qu.:0.0000   1st Qu.:1
##  By turning into a road     :160808   Darkness:133702   Slippery: 14879   Median :0.0000   Median :1
##  By crossing the road       : 29227                                       Mean   :0.3187   Mean   :0
##  Involving stationary       : 24853                                       3rd Qu.:1.0000   3rd Qu.:1
##  Vehicles moving in parallel:172983                                       Max.   :1.0000   Max.   :1
##  Other                      : 92845
##  with_motorcycle    with_truck        with_other         WGSX             WGSY       district_key
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.000   Min.   : 5.871   Min.   :47.32   Length:727694
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.000   1st Qu.: 7.992   1st Qu.:49.38   Class :character
##  Median :0.0000   Median :0.00000   Median :0.000   Median : 9.509   Median :51.04   Mode  :character
##  Mean   :0.1388   Mean   :0.04771   Mean   :0.113   Mean   : 9.744   Mean   :50.87
##  3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.000   3rd Qu.:11.461   3rd Qu.:52.30
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.000   Max.   :15.015   Max.   :55.03
##
##   severity_num      population           area              district_type       min_temp         max
##  Min.   :0.0000   Min.   :  34534   Min.   :  35.7   District      :112021   Min.   :-8.525   Min.
##  1st Qu.:0.0000   1st Qu.: 157443   1st Qu.: 318.2   Large City    :233988   1st Qu.: 2.000   1st Qu
##  Median :0.0000   Median : 253551   Median : 799.5   Rural District:363215   Median : 7.060   Median
##  Mean   :0.1937   Mean   : 533182   Mean   : 925.2   Urban District: 18470   Mean   : 6.819   Mean
##  3rd Qu.:0.0000   3rd Qu.: 465838   3rd Qu.:1274.6   Water body    :     0   3rd Qu.:11.525   3rd Qu
##  Max.   :1.0000   Max.   :3755251   Max.   :5495.6                           Max.   :18.325   Max.
##
##  weekday_count        road_category        lanes         maxspeed           lit             oneway
##  Min.   :4.000   secondary   :219975   Min.   :1.00   Min.   :  3.00   no     : 86911   no     : 342
##  1st Qu.:4.000   tertiary    :134593   1st Qu.:1.00   1st Qu.: 50.00   unknown:310706   unknown:5106
##  Median :4.000   residential :128710   Median :2.00   Median : 50.00   yes    :330077   yes    :1827
##  Mean   :4.409   primary     :128405   Mean   :1.97   Mean   : 63.72
##  3rd Qu.:5.000   motorway    : 44729   3rd Qu.:2.00   3rd Qu.: 70.00
##  Max.   :5.000   unclassified: 27595   Max.   :8.00   Max.   :250.00
##                  (Other)     : 43687
```

Accident Severity

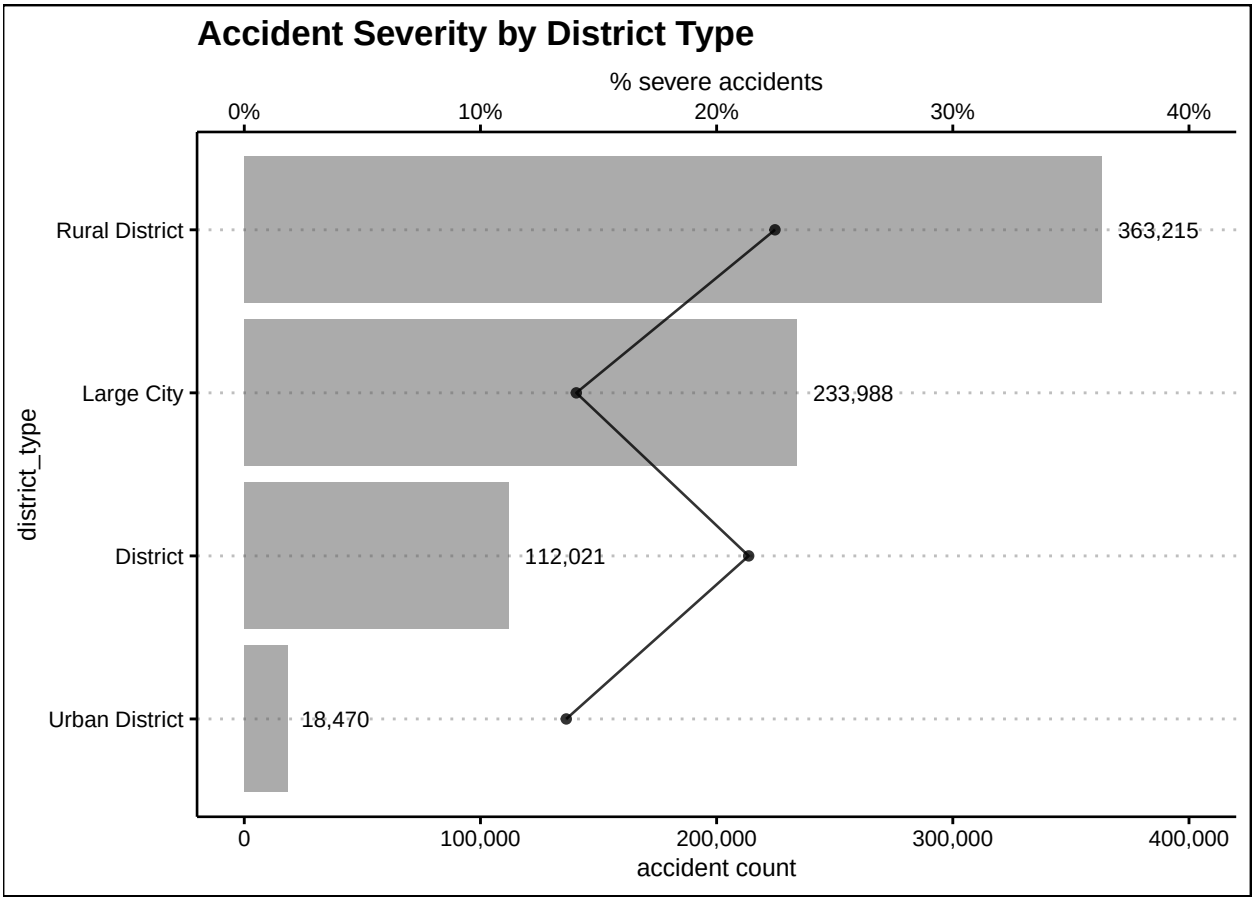
The graphs in this section illustrate accident volume and severity by state, district type, collision with, accident type, involved vehicle types,



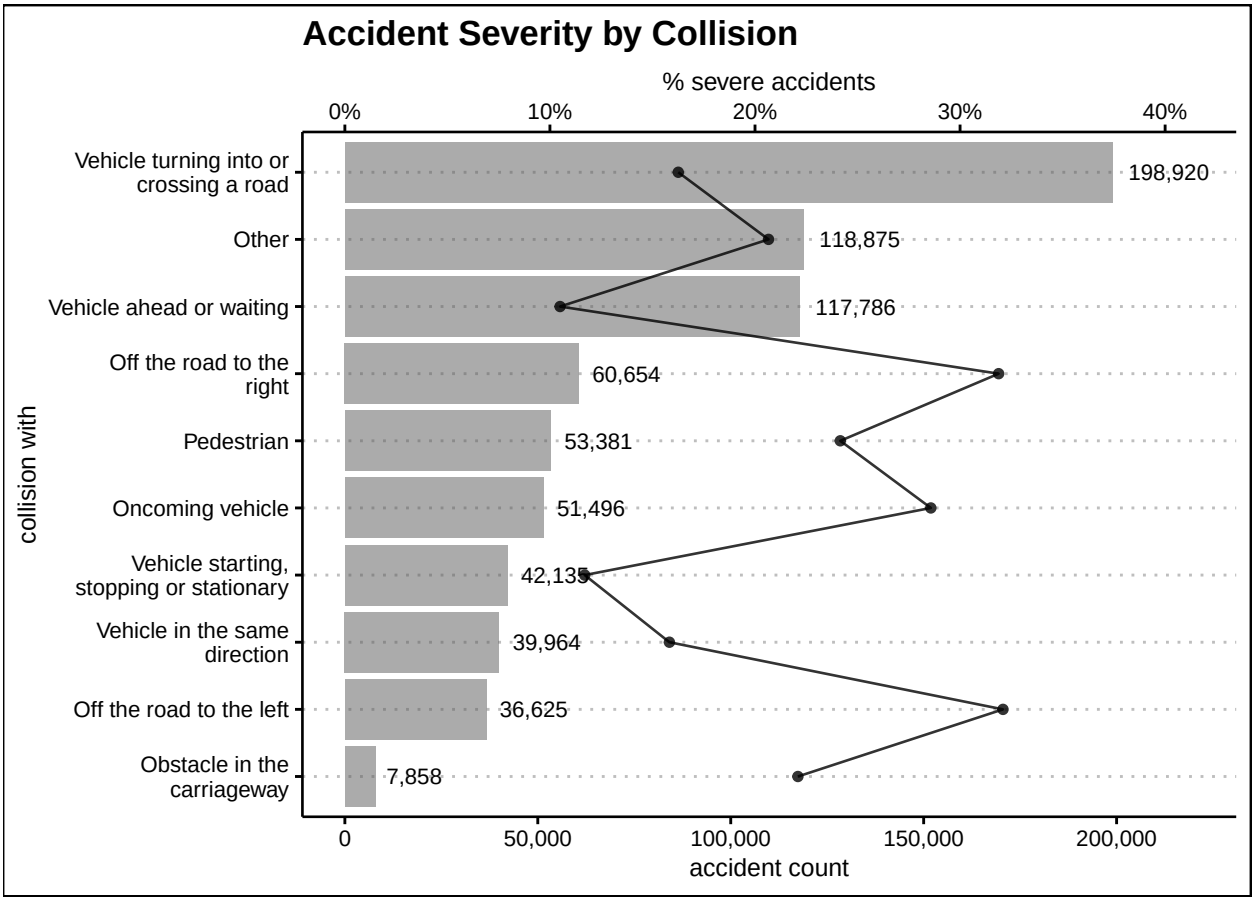
Initially, an analysis is conducted on the target variable. Even though most traffic accidents lead to minor injuries, as depicted in the graph above, the number of fatalities remains a significant and concerning aspect.



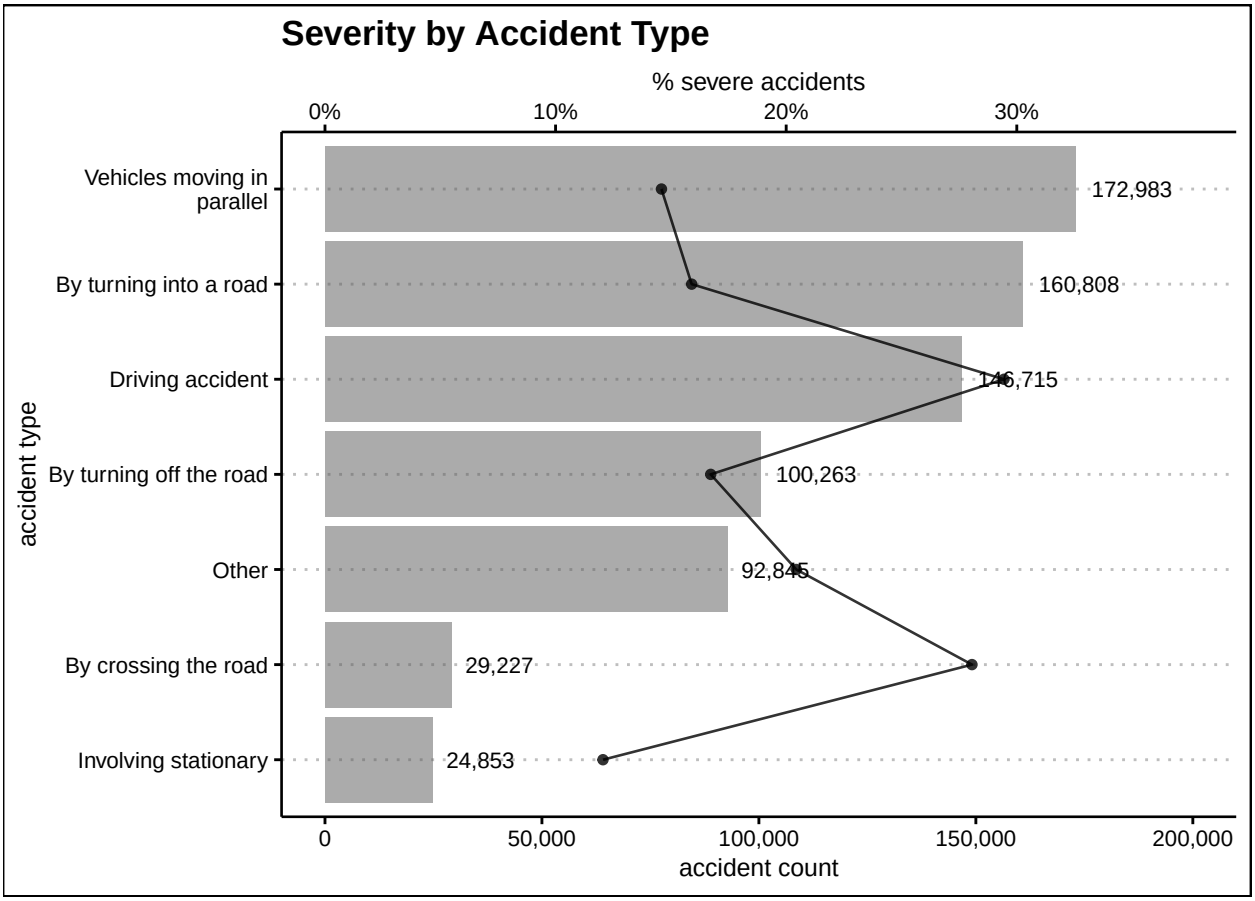
The correlation between the population size of states and the frequency of traffic accidents becomes evident in the above graph. Bremen, and Hamburg have relative few severe accidents.



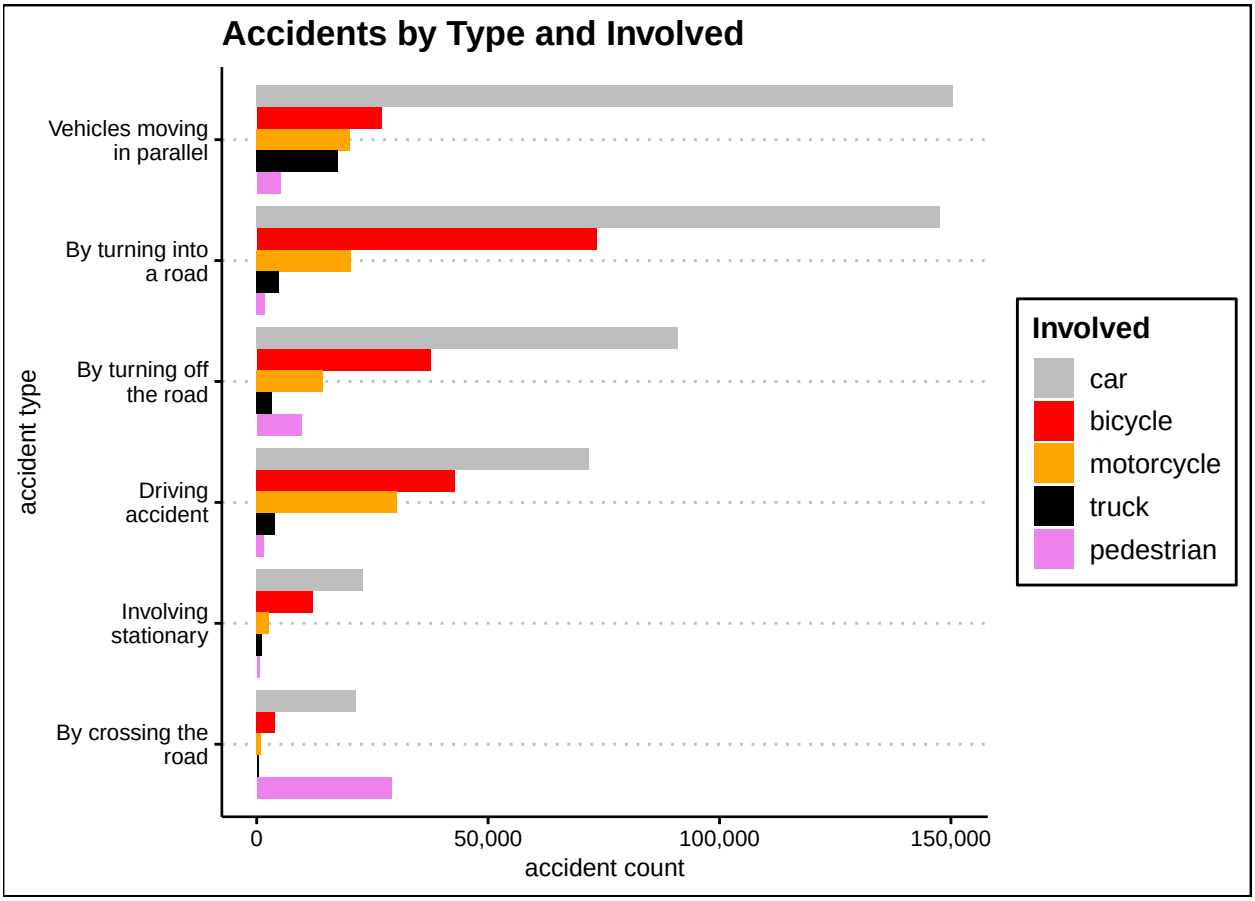
This graph shows that accidents predominantly occur in rural districts and large cities, where most of the German population resides. Less densely populated districts are associated with a higher likelihood of severe accidents.



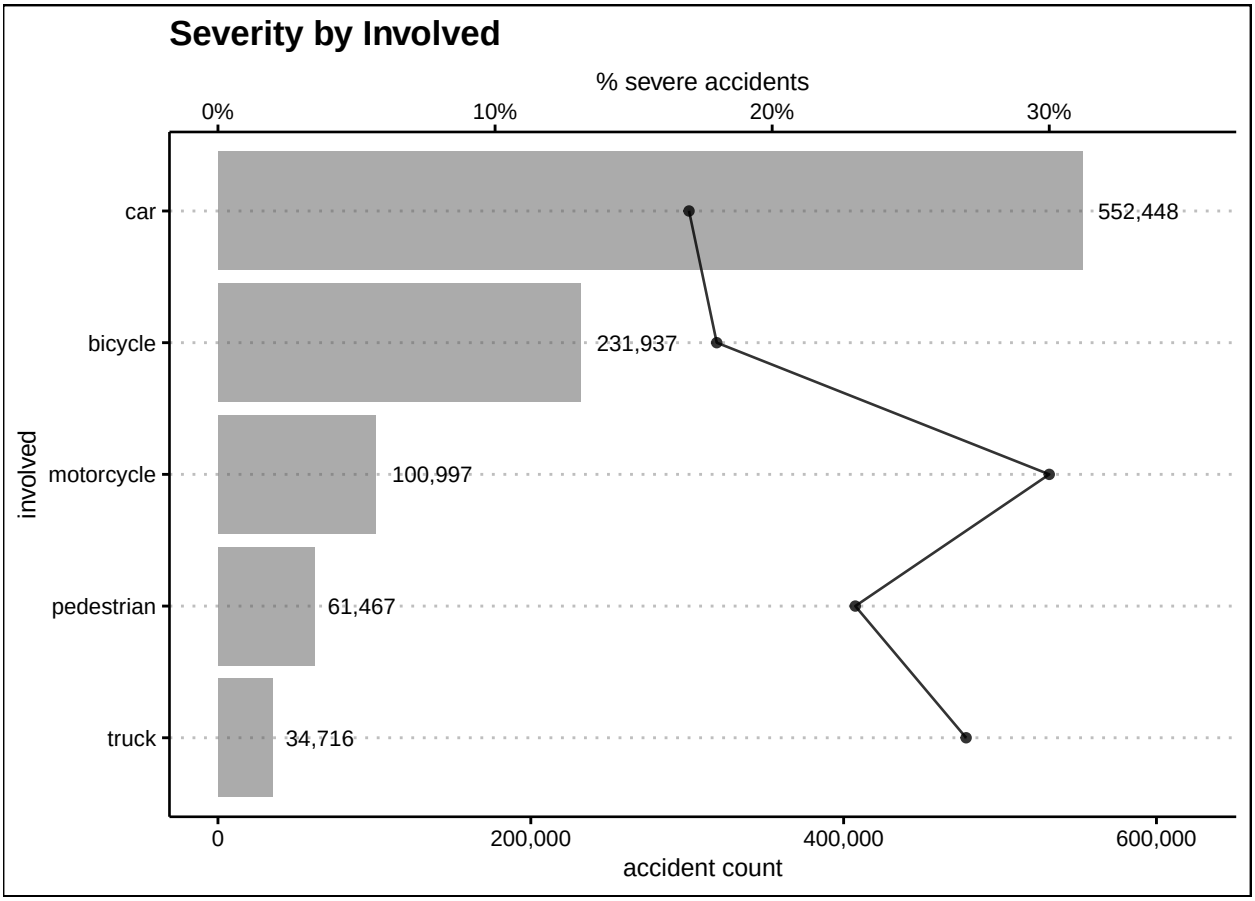
By analyzing the collision with variable in this graph, it becomes apparent that accidents predominantly occur most often with turning or crossing vehicles. However, fatal accidents are more frequent when vehicles are coming off the road or collide with oncoming vehicles.



This chart shows that the primary cause of fatal accidents is driving accidents. On the other hand, accidents resulting in lighter accidents predominantly occur between vehicles in parallel traffic.



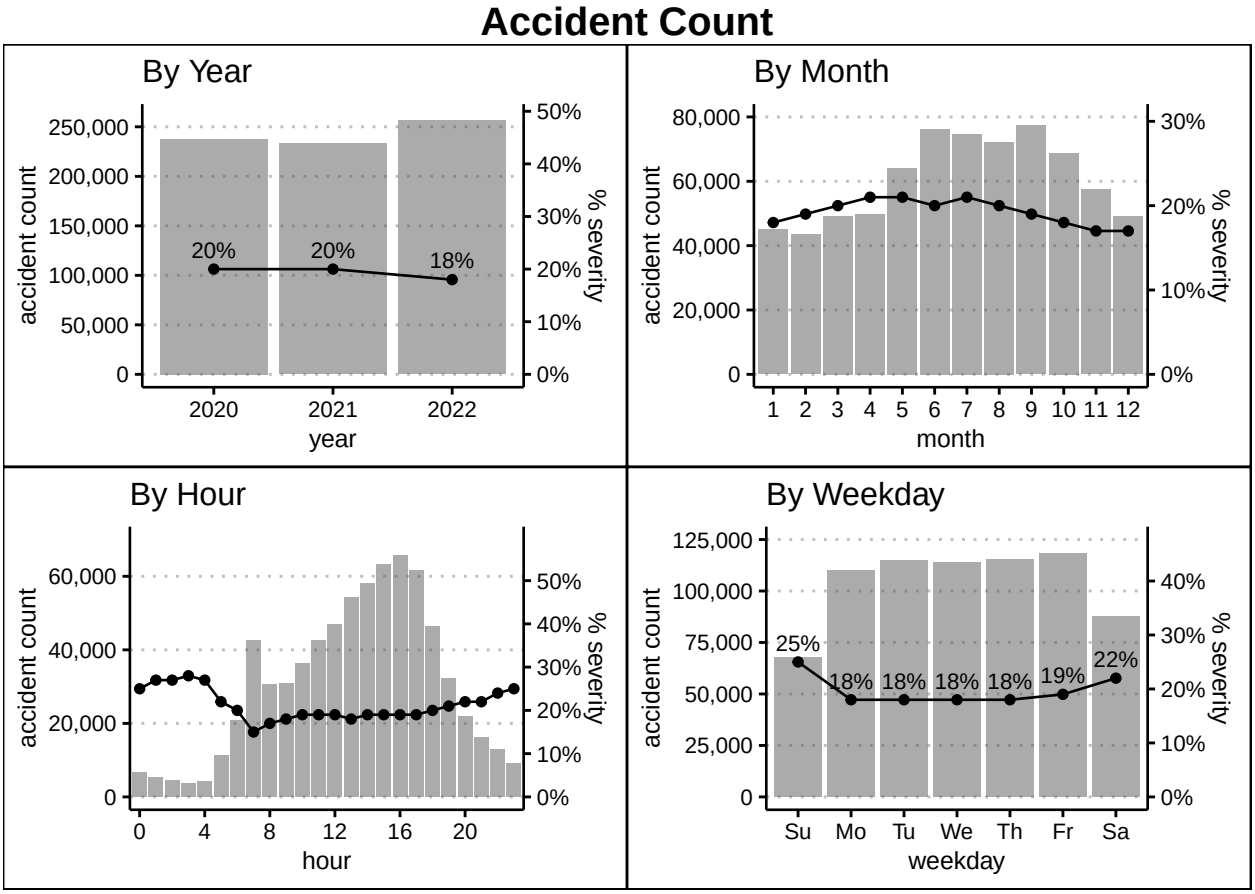
The nature of accidents varies significantly depending on the type of vehicle involved. Collisions with cars are the most prevalent across all accident types, except for accidents involving pedestrians crossing the road. Bicycle collisions frequently result from turning into a road, while accidents with trucks often occur when vehicles are moving in parallel.



The involvement of motorcycles and trucks results in more severe accidents than all other types of accidents as next the graph shows.

Temporal Effects

The following graphs reveal various temporal effects.

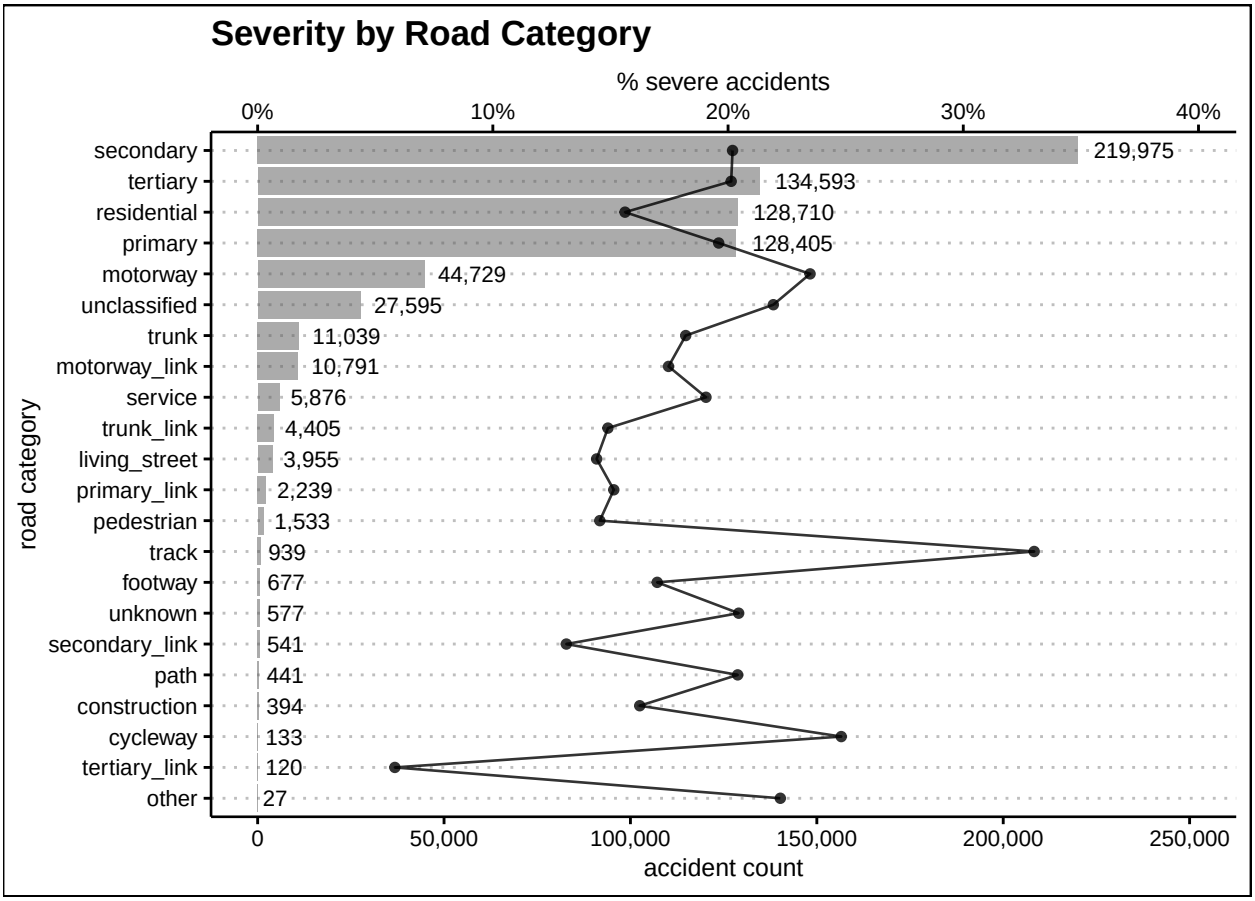


In 2020 and 2021, there's a notable reduction in accidents, likely attributable to the COVID-19 pandemic. Examining the monthly distribution exposes a discernible seasonal effect.

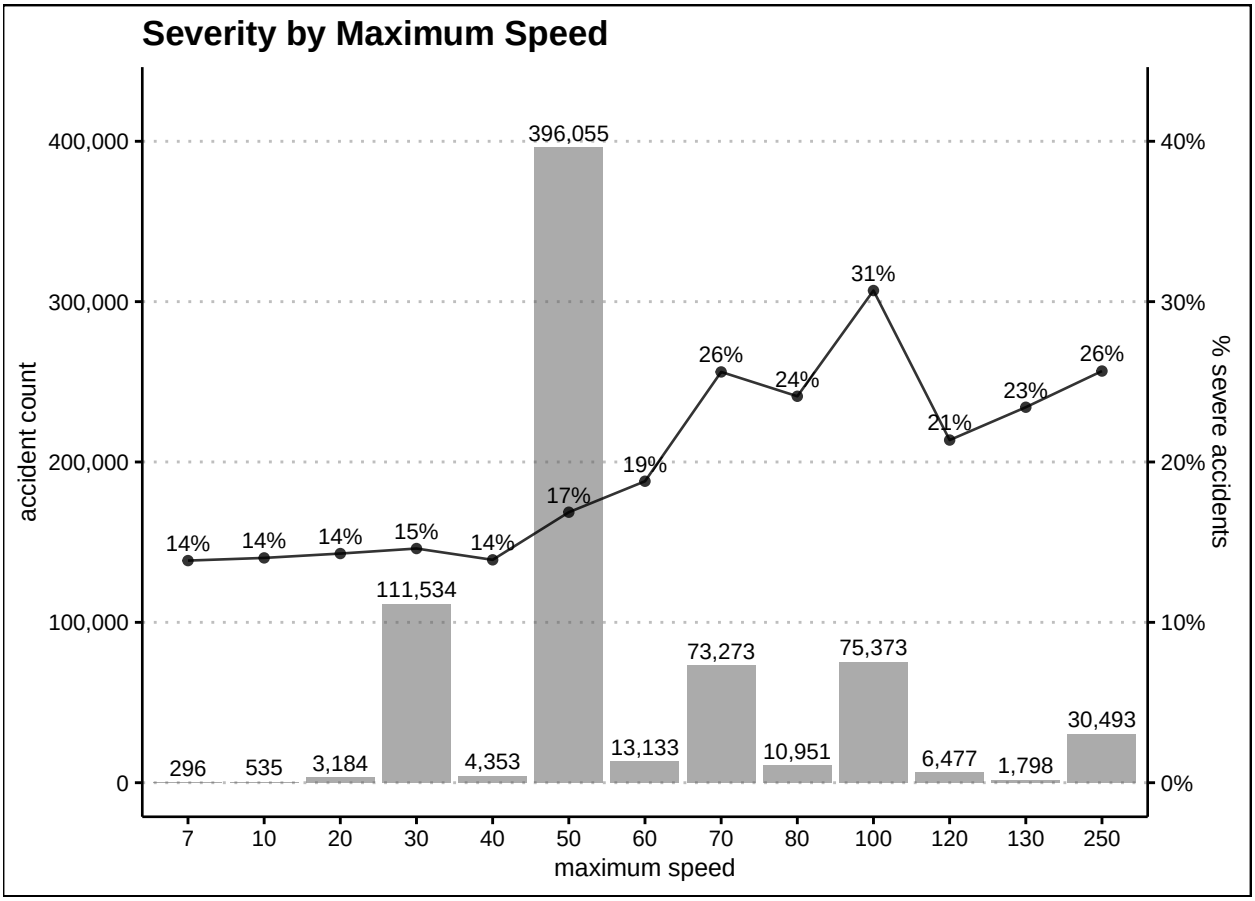
Furthermore, the distribution of accidents throughout the day exhibits a peak towards the conclusion of traditional working hours. Throughout the workweek, the number of traffic accidents steadily increases, reaching its peak on Fridays. Weekends, particularly Sundays, display a decrease in the frequency of accidents, but an increase of severity.

Road

The upcoming graphs illustrate the influence of the road on the severity of accidents.



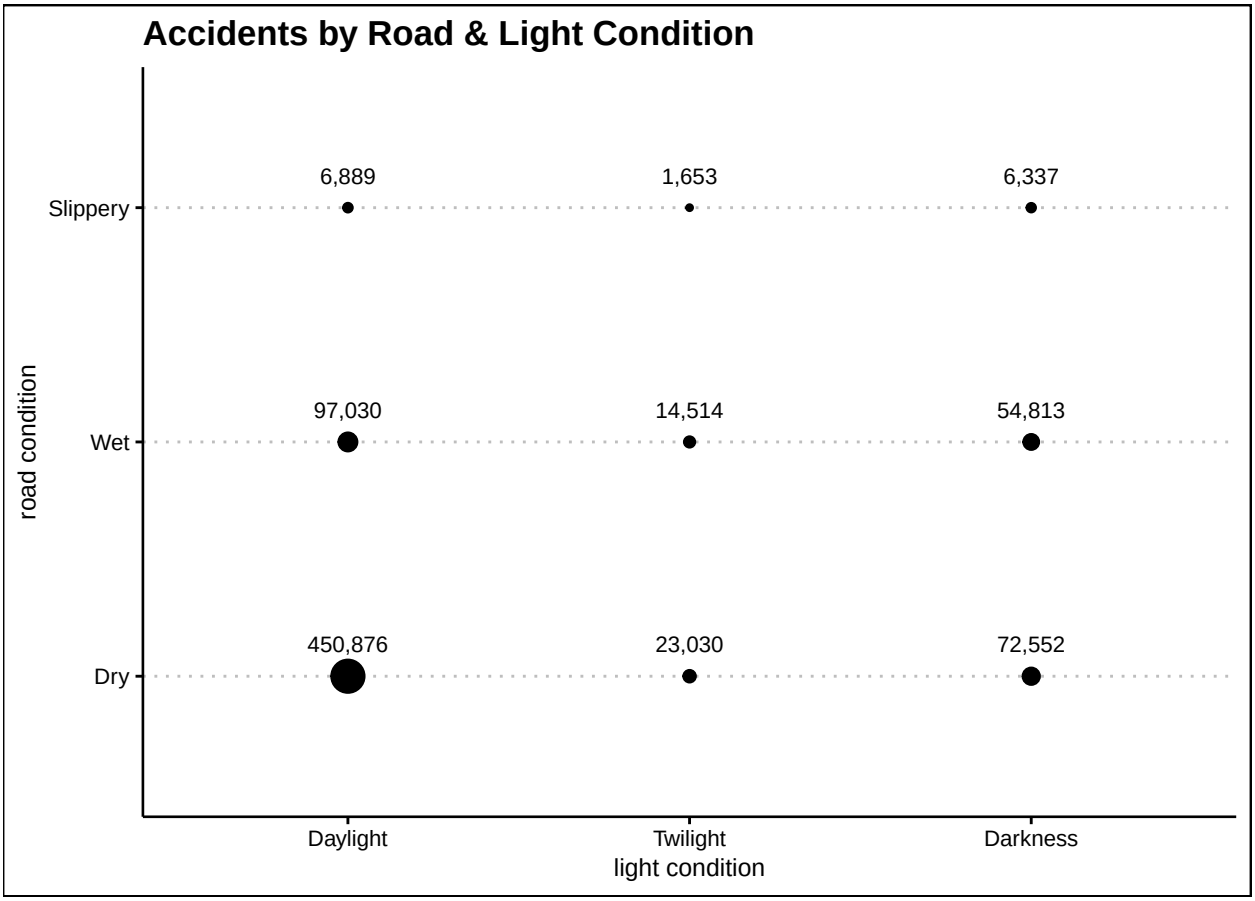
This graph shows that only a minority of accidents occur on motorways, but they have a higher severity.



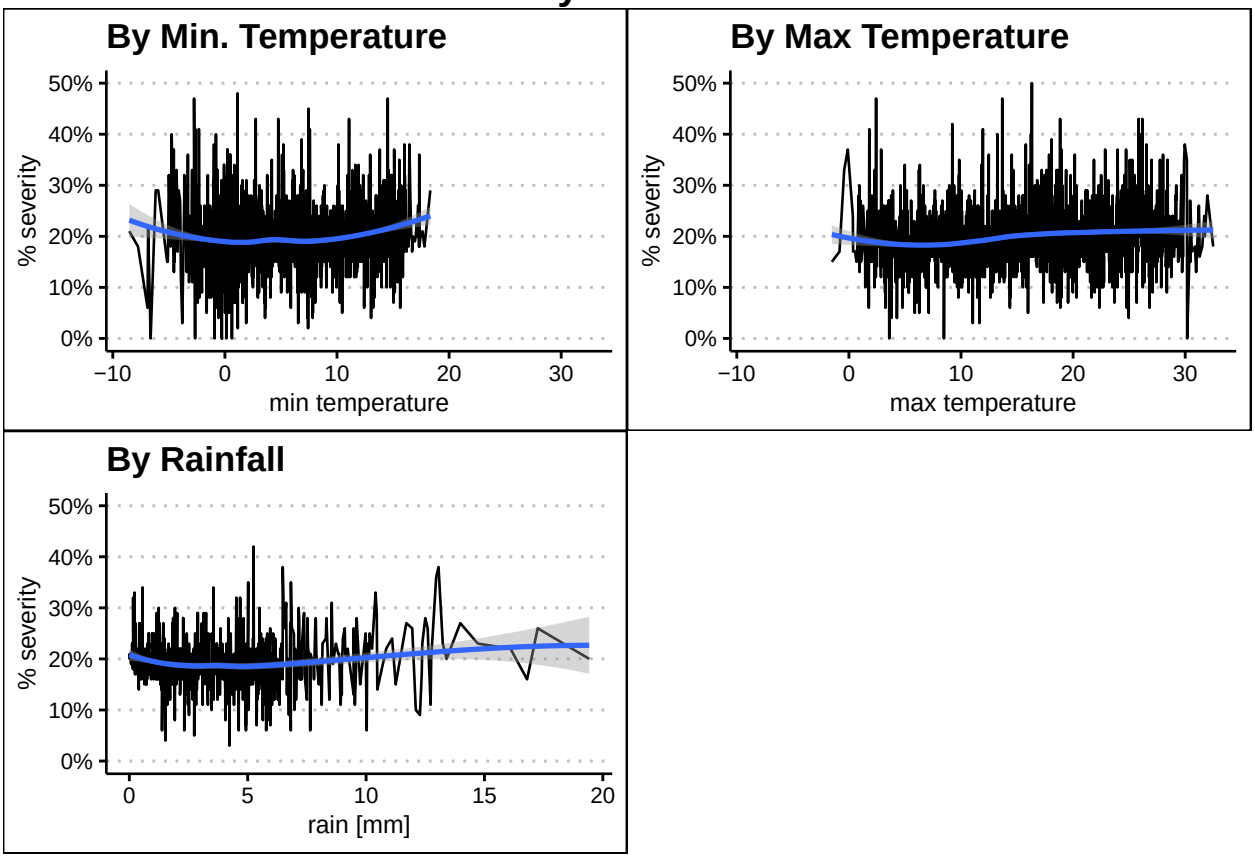
The above graph depicts relative severity by permitted maximum speed. Accidents are more severe on fast roads.

Weather Influence

The subsequent charts illustrate the weather influence on accidents.



This chart shows that the majority of accidents occur during daylight hours on dry roads.

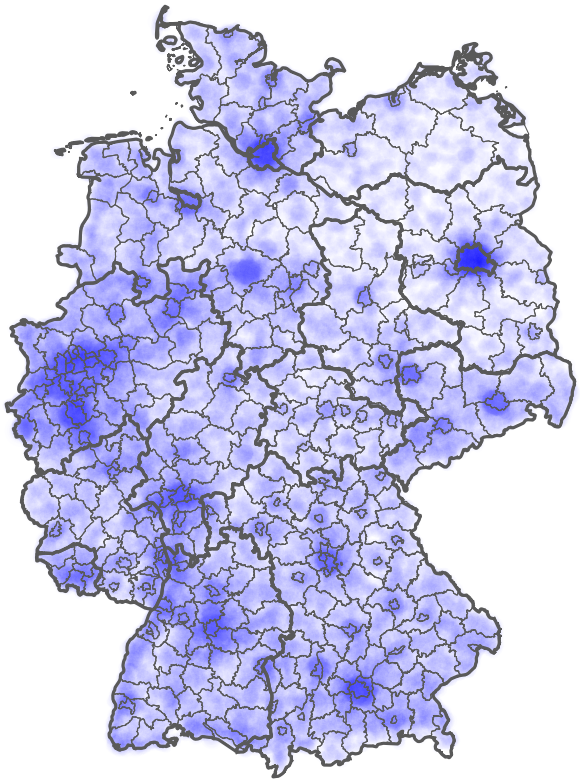


Examining the impact of weather variables on the average number of accidents per day reveals modest effects. Rainfall demonstrates a slight influence, and among the three temperature variables, the minimum temperature exhibits the biggest effect. These results should be interpreted with caution as weather data could not be mapped with high accuracy.

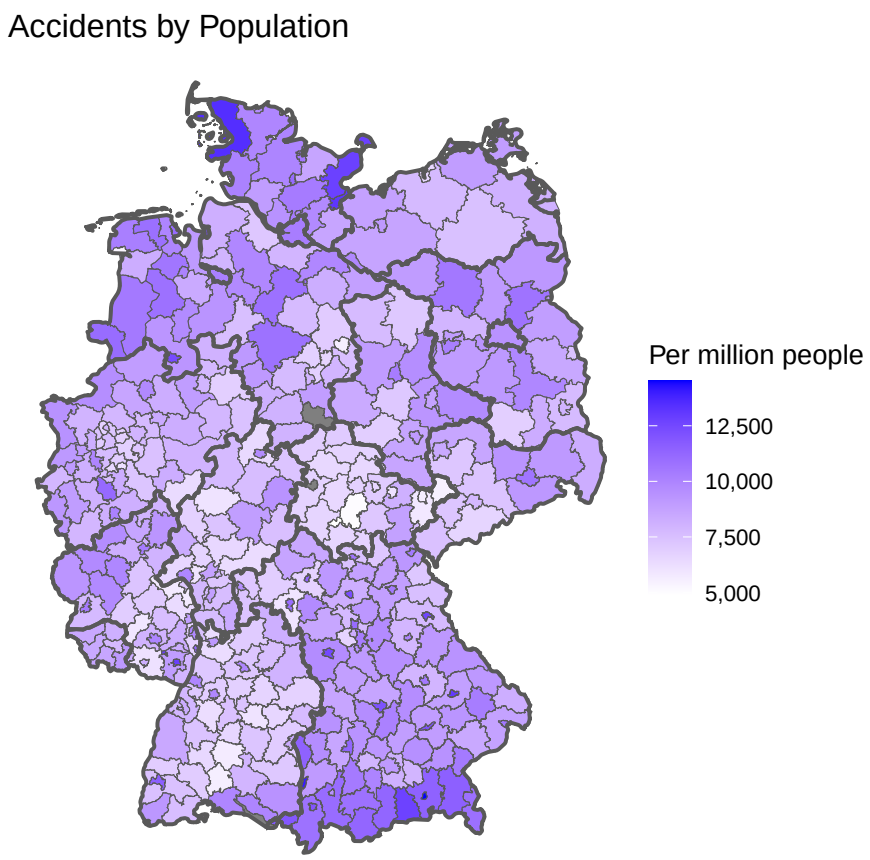
Spatial Effects

In this section, three maps of Germany are provided, highlighting the spatial distribution of accidents, the number of accidents by population, and the distribution of accident severity.

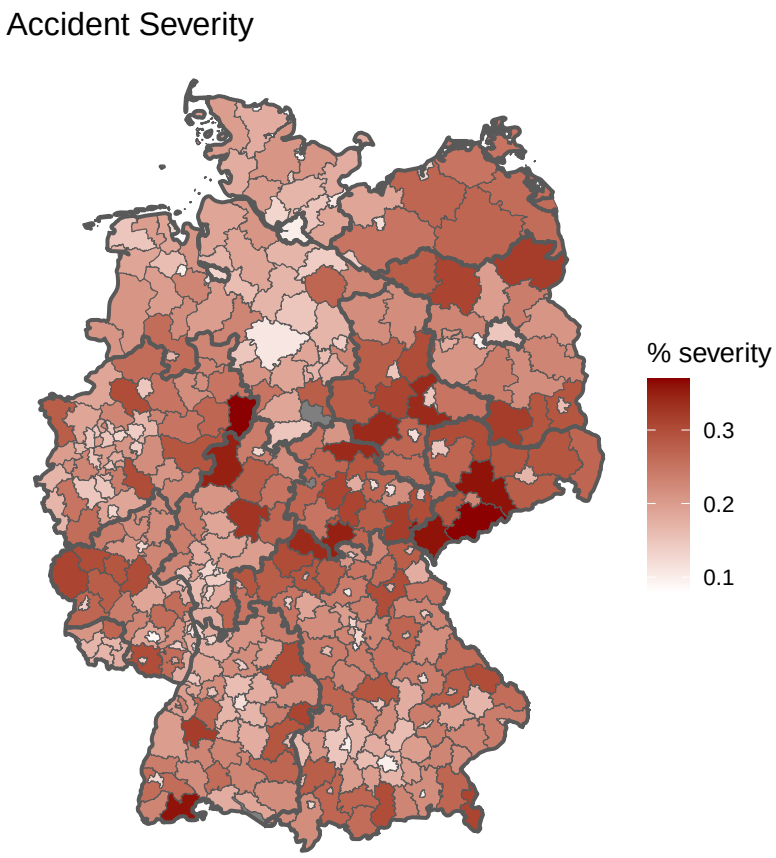
Regional Distribution of Accidents



To enable the visualization of accident counts on a map, the exact locations are converted into a grid comprising 500 x 500 elements across Germany. The graph above illustrates that the majority of traffic accidents occur in areas characterized by high population density.



This graph offers additional insights into the correlation between population and the frequency of traffic accidents. The map illustrates the number of accidents per million people residing in each district in Germany. Larger cities and tourist areas at the Northern coast and Southern mountain regions of Germany exhibit relatively more accidents, influenced by the influx of visitors in these areas.



It's also insightful to examine the severity of accidents which shows significant variance on the next map. Larger cities have fewer severe accidents.

2.2.3 Data Preparation

The last chapter provided an overview of the available data. To enable construction of the models, this data is now split into model and final test data, with 20% of the data reserved for the final test. The model data will be further split into training and test data.

```
# remove variable weekday_count
accidents <- accidents |> select(-weekday_count)

set.seed(42)

# create two partitions of accident data
test_index <- createDataPartition(y = accidents$severity, times = 1,
                                  p = 0.2, list = FALSE)
accidents_model <- accidents[-test_index,]
accidents_final_test <- accidents[test_index,]
```

Training and Test Datasets

In addition to segregating the data into model and final test datasets, we further divide the model dataset *accidents_model* into two partitions: training and test datasets. The training dataset encompasses 80%,

while the test dataset retains 20% of the data. This reserves the *accidents_final_test* data for the final validation.

```
set.seed(42)
test_index <- createDataPartition(y = accidents_model$severity,
                                  times = 1, p = 0.2, list = FALSE)
accidents_train <- accidents_model[-test_index,]
accidents_test <- accidents_model[test_index,]
```

2.3 Model Development

Initially, the *accuracy* function is defined, computing both accuracy, balanced accuracy, and F1 score. The objective is to identify a model that maximizes the balanced accuracy while maintaining good accuracy.

Next, a linear model is constructed. This model undergoes modifications through the oversampling of severe accidents, sigular value decomposition and combining separate models for different vehicle types.

Subsequently, alternative models are then developed using the random forest approach and extreme gradient boosting. Finally, an ensemble model is crafted by combining these diverse approaches.

The *accuracy* function utilizes the *confusionMatrix* function of the *caret* package and calculates accuracy, balanced accuracy, and F1 score.

```
accuracy <- function(true_severity, predicted_severity) {
  cm <- confusionMatrix(predicted_severity,
                        true_severity,
                        mode = "everything",
                        positive = "severe")
  c(cm$overall[c("Accuracy")], cm$byClass["F1"], cm$byClass["Balanced Accuracy"])
}
```

2.3.1 Simple Median Model

The initial model employs the most frequent severity as a most basic predictive measure. Given that a significant proportion of accidents have light severity, this severity level represents both the median and the most frequently occurring outcome.

This model serves as a basic benchmark and the resultant accuracy, balanced accuracy, and F1 score are recorded in the variable *evaluation*.

```
# Calculate most frequent severity
modal_severity <- as.integer(names(which.max(table(accidents_train$severity_num))))

# Compute prediction and convert back to factor
severity_hat <- rep(modal_severity, nrow(accidents_test))
severity_hat <- factor(severity_hat, levels = 0:1, labels = c("light", "severe"))
```

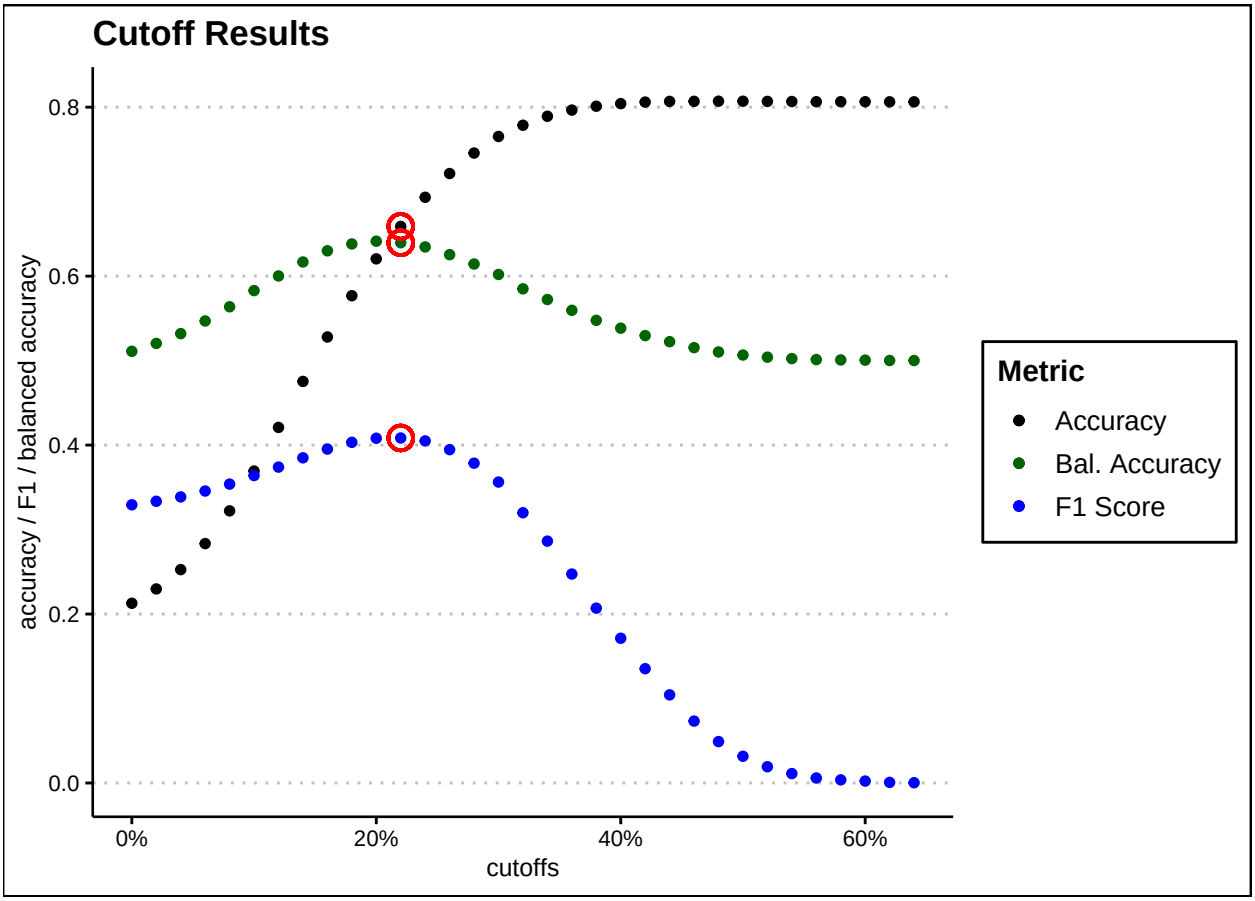
The table records the accuracy, balanced accuracy, and F1 score of the models:

```
## # A tibble: 1 x 5
##   data  model         accuracy    f1 balanced
##   <chr> <chr>            <dbl> <dbl>    <dbl>
## 1 model simple median    0.806    NA      0.5
```

2.3.2 Linear Model

In the next model, a linear approach is adopted utilizing all features. To accommodate the linear model requirements, the target variable is used in the numerical format. Once the prediction is generated, the prediction is reverted back to its original factor form. This ensures compatibility with the linear model while preserving the categorical nature of the target variable. The cutoff value for the conversion is determined using cross-validation.

```
set.seed(42)
# Train model
model_lm <- lm(severity_num ~ .,
               data = accidents_train |> select(-severity))

# Compute prediction
severity_lm <- predict(model_lm, accidents_test)

# Find the optimal cutoff to convert number predictions into factor
cutoffs <- seq(0, .65, .02)

# record accuracies
accuracies <- c()
f1_scores <- c()
balanced_accuracies <- c()

#try each cutoff
for (cutoff_lm in cutoffs) {
  # convert using cutoff
  severity_test <- ifelse((severity_lm >= cutoff_lm), 1, 0)
  severity_test <- factor(severity_test, levels = 0:1, labels = c("light", "severe"))

  # record accuracy
  a <- accuracy(accidents_test$severity, severity_test)

  #if NA set to 0
  if (is.na(a["F1"]))
     a[] <- 0
  accuracies <- c(accuracies, a["Accuracy"])
  f1_scores <- c(f1_scores, a["F1"])
  balanced_accuracies <- c(balanced_accuracies, a["Balanced Accuracy"])
}
# where is F1 score maximum
i <- which.max(f1_scores)
cutoff_lm <- cutoffs[i]
```



The graph above shows the resulting accuracy scores for each cutoff value. The highlighted point with maximum F1 score has been selected to convert the prediction from numbers to factors.

```
# use optimal cutoff for conversion
severity_lm <- ifelse((severity_lm >= cutoff_lm), 1, 0)
severity_lm <- factor(severity_lm, levels = 0:1, labels = c("light", "severe"))
```

The table records the accuracy, balanced accuracy, and F1 score of the models:

```
## # A tibble: 2 x 5
##   data  model          accuracy     f1 balanced
##   <chr> <chr>             <dbl>  <dbl>    <dbl>
## 1 model simple median    0.806 NA        0.5
## 2 model linear model     0.659  0.408    0.640
```

2.3.3 Linear Model with Oversampling

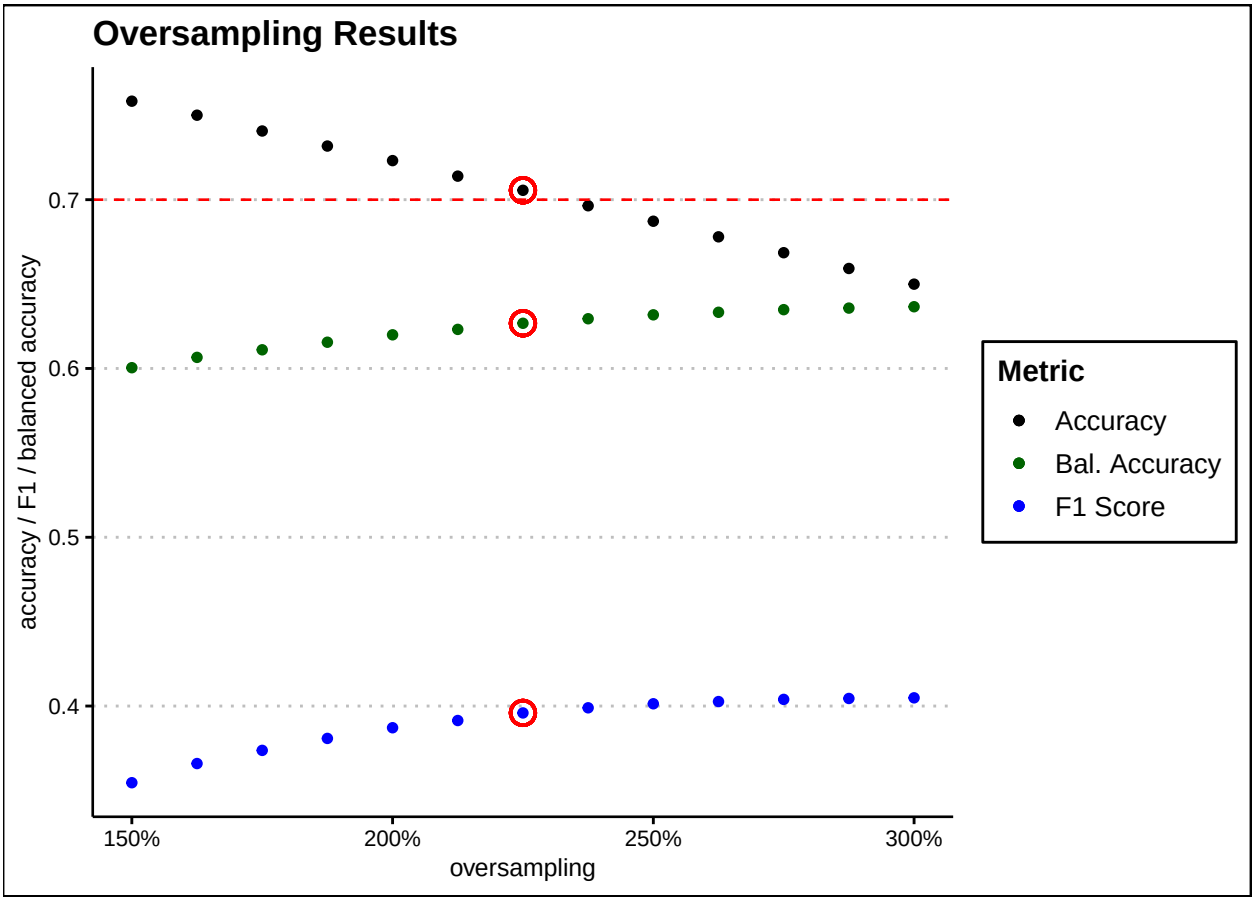
The previous model reveals an imbalance, struggling to precisely predict severe accidents. In the following model, enhancements are integrated into the linear model through the oversampling of severe accidents.

Oversampling is executed at various magnitudes to pinpoint a parameter that maximizes the F1 score without dropping accuracy below 70%. To implement this strategy, a sampling pool is established, encompassing all severe accidents. Samples of varying sizes are drawn from this pool and subsequently incorporated into the training data. The resulting models undergo evaluation.

The ultimate oversampling parameter is determined, contributing to a model that accommodates the distinctive characteristics of severe accidents.

As an alternative to oversampling, Synthetic Minority Over-sampling Technique (SMOTE) was also evaluated, but did not yield in favorable results and was no longer applied.

```
# Test oversampling with 150% - 300% of the severe accidents
over_percentages <- seq(1.5, 3, .125)

# record accuracy
accuracies <- c()
f1_scores <- c()
balanced_accuracies <- c()
for (over_percentage in over_percentages){
  set.seed(42)

  # copy of the training data
  accidents_oversample <- accidents_train

  # Filter only severe accidents
  sampling_pool <- accidents_oversample |>
    filter(severity == "severe")

  # Add a total of x% additional severe accidents
  sample_size <- round(over_percentage * nrow(sampling_pool) )

  # create extra sample
  over_sample <- sampling_pool[sample(nrow(sampling_pool), sample_size, replace = TRUE),]

  # Add extra sample to training data
  accidents_oversample <- rbind(accidents_oversample, over_sample)

  # Train linear model (without district key is much faster)
  model_olm <- lm(severity_num ~ . ,
                  data = accidents_oversample |> select(-severity, -district_key))

  # Compute prediction with rounding as simple conversion
  severity_olm <- predict(model_olm, accidents_test)
  severity_olm <- severity_olm <- round(severity_olm)
  severity_olm <- factor(severity_olm, levels = 0:1, labels = c("light", "severe"))

  # record accuracy
  a <- accuracy(accidents_test$severity, severity_olm)
  accuracies <- c(accuracies, a["Accuracy"])
  f1_scores <- c(f1_scores, a["F1"])
  balanced_accuracies <- c(balanced_accuracies, a["Balanced Accuracy"])
}

# where has accuracy dropped to less than 70%?
i <- which.max(accuracies < .70) - 1
over_percentage <- over_percentages[i]
```



The graph above shows the resulting accuracies and F1 scores for each tested magnitude of oversampling. The highlighted points have been selected to build the following models.

The linear model is subsequently trained with the selected parameter for oversampling.

```
set.seed(42)

# Model with "ideal" oversample
# copy of the training data
accidents_oversample <- accidents_train

# Filter only severe accidents
sampling_pool <- accidents_oversample |>
  filter(severity == "severe")

# Add a total of x% additional severe accidents
sample_size <- round(over_percentage * nrow(sampling_pool) )

# create extra sample
over_sample <- sampling_pool[sample(nrow(sampling_pool), sample_size, replace = TRUE),]

# Add extra sample to training data
accidents_oversample <- rbind(accidents_oversample, over_sample)
```

```r
# Train model (with district_key)
model_olm <- lm(severity_num ~ . , data = accidents_oversample |> select(-severity))

# Compute prediction and convert to factor with simple rounding
severity_olm <- predict(model_olm, accidents_test)
severity_olm <- severity_olm <- round(severity_olm)
severity_olm <- factor(severity_olm, levels = 0:1, labels = c("light", "severe"))
```

The table records the accuracy, balanced accuracy, and F1 score of the models:

```
## # A tibble: 3 x 5
##   data  model                          accuracy     f1 balanced
##   <chr> <chr>                             <dbl>  <dbl>    <dbl>
## 1 model simple median                     0.806 NA        0.5
## 2 model linear model                      0.659  0.408    0.640
## 3 model linear model with oversample      0.708  0.399    0.629
```

2.3.4 Linear Model with Singular Value Decomposition

The subsequent model once again utilizes the training data with oversampled severe accidents. Singular value decomposition (SVD) is employed to recognize patterns in the geographic distribution represented by latitude and longitude.

To accommodate the available computational resources, latitude and longitude calculations are performed on a grid of 10,000 x 10,000 elements. The count of severe accidents is aggregated for each grid element, forming matrix S. The grid approach requires an extra step to utilize the result of the SVD.

The singular value decomposition yields three matrices U, D, and V, such that:

$$S = UDV^T$$

U and V are orthogonal matrices representing rotations, while the diagonal matrix D captures the variability of the original matrix. To model the severity, a subset of $k = 100$ parameters in D is used, providing an approximation that retains most of the variability of S and is aimed at recognizing spatial patterns. The function *svds* from package *RSpectra* is used to compute the SVD with the large data volume.

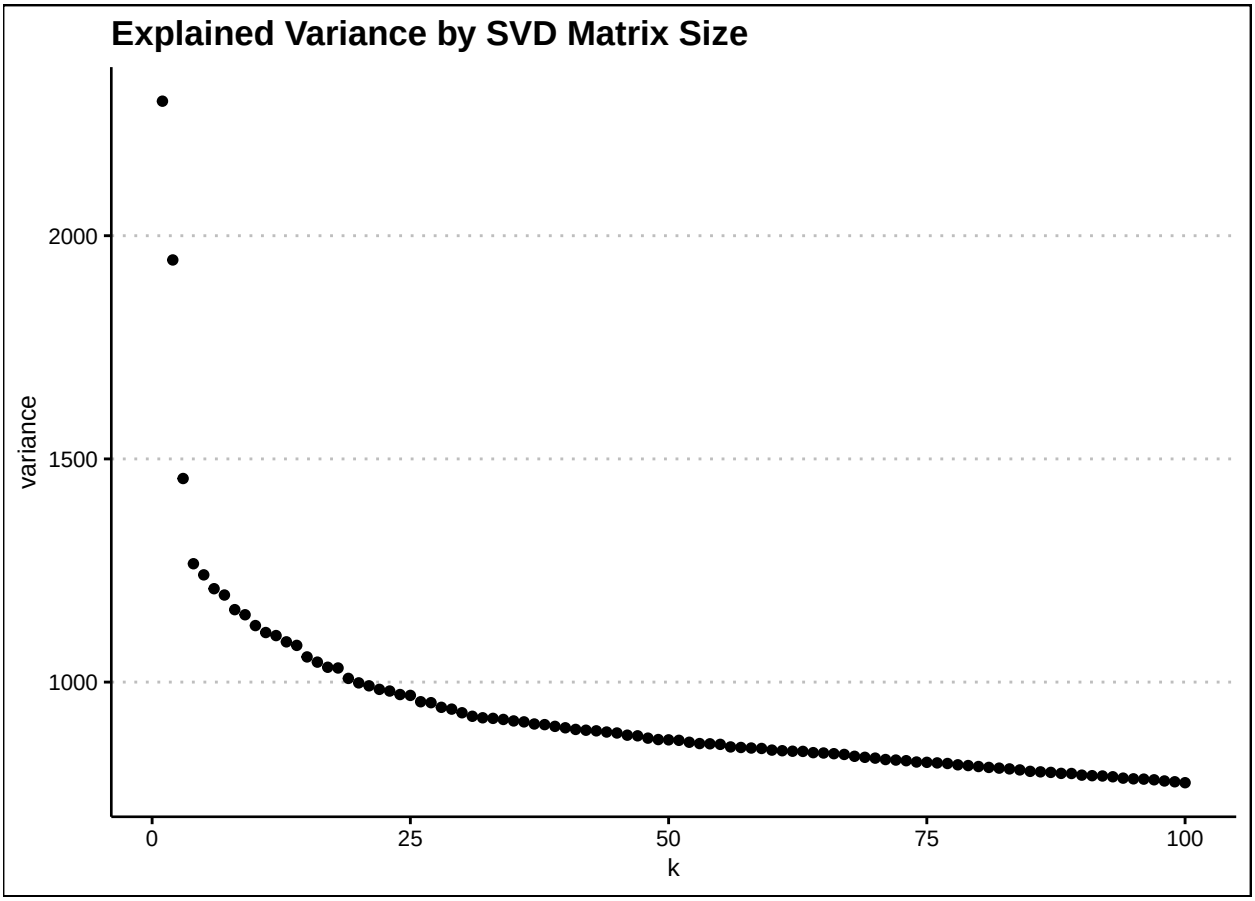
The grid approach and the complexity of the relationships make it necessary to combine SVD with another model. To facilitate this, mapping tables are generated from U and V, that map latitude and longitude to the k rotated coordinates. The training data is enriched by joining the mapping table and integrating the rotated coordinates.

Finally, a linear model is constructed, incorporating the rotated coordinates along with all other features.

```r
set.seed(42)

# Number of transformed variables
k <- 100

# Number of grid elements
n1 <- 10000
n2 <- 10000

# map on a country level
xmin <- min(accidents$WGSX)
xlen <- max(accidents$WGSX) - xmin
```

```r
ymin <- min(accidents$WGSY)
ylen <- max(accidents$WGSY) - ymin

# Compute a grid for lat, lon and summarize severity = count severe accidents in grid
accidents_tmp <- accidents_oversample|>
  mutate(severity_num,
         WGSX_grid = round((WGSX - xmin)/xlen*n1),
         WGSY_grid = round((WGSY - ymin)/ylen*n2)) |>
  group_by(WGSX_grid, WGSY_grid)|>
  summarize(severity_num = sum(severity_num),.groups = "keep")

# Convert to a lat x lon matrix with severity total
# data frame with all X coordinates in raster
accidents_tmp <- accidents_tmp |>
  pivot_wider(names_from = WGSX_grid, values_from = severity_num)

# save the coordinates
WGSY_vec <- accidents_tmp$WGSY_grid
WGSX_vec <- as.numeric(colnames(accidents_tmp[,-1]))

# convert to matrix
train_matrix <- as.matrix(accidents_tmp[,-1])
rownames(train_matrix) <- WGSY_vec

# SVD does not like NA, set NA to 0
train_matrix[is.na(train_matrix)] <- 0

# convert to a sparse matrix and calculate singular value decomposition
train_matrix_sp = as(train_matrix, "dgCMatrix")
svd_result <- svds(train_matrix_sp, k)
```

The graph below visualizes the variance by the parameter k. It quickly drops to low values and the assumption that $k = 100$ variables provide enough variability seems plausible.



The matrices U and V are now converted to data frames to enrich the training data with transformed variables.

```
# now calculate a mapping of WGSX coordinates to the k transformed coordinates
WGSX_df <- data.frame(svd_result$v)
WGSX_df$WGSX_grid <- WGSX_vec

# mapping for Y coordinates
WGSY_df <- data.frame(svd_result$u)
WGSY_df$WGSY_grid <- WGSY_vec
colnames(WGSY_df) <- str_replace(colnames(WGSY_df),"X","Y")

# join rotated X and Y coordinates to accident data
accidents_svd <- accidents_oversample |>
  mutate(WGSX_grid = round((WGSX - xmin)/xlen*n1),
         WGSY_grid = round((WGSY - ymin)/ylen*n2)) |>
  left_join(WGSX_df, by = "WGSX_grid") |>
  left_join(WGSY_df, by = "WGSY_grid") |>
  select( -c(severity, WGSX_grid, WGSY_grid))

# build linear model for enriched accident data (220+ variables!)
model_svd <- lm(severity_num ~ ., data = accidents_svd)
```

To facilitate the prediction process, the transformed variables need to be incorporated into the test data set.

```r
# to compute prediction calculate the grid for test data
accidents_tmp <- accidents_test |>
  mutate(WGSX_grid = round((WGSX - xmin)/xlen*n1),
         WGSY_grid = round((WGSY - ymin)/ylen*n2)) |>
  left_join(WGSX_df, by = "WGSX_grid") |>
  left_join(WGSY_df, by = "WGSY_grid")


# compute prediction for test data
severity_svd <- as.numeric(predict(model_svd, accidents_tmp))

# a few raster points in test with no representation in train might get NA
# guess they are all light accidents
severity_svd[is.na(severity_svd)] <- 0

# convert to factor
severity_svd <- round(severity_svd)
severity_svd <- factor(severity_svd, levels = 0:1, labels = c("light", "severe"))
```

The table records the accuracy, balanced accuracy, and F1 score of the models:

```
## # A tibble: 4 x 5
##   data  model                          accuracy     f1 balanced
##   <chr> <chr>                             <dbl>  <dbl>    <dbl>
## 1 model simple median                     0.806 NA        0.5
## 2 model linear model                      0.659  0.408    0.640
## 3 model linear model with oversample      0.708  0.399    0.629
## 4 model linear model with SVD             0.706  0.386    0.619
```

The linear model that includes singular value decomposition does not show better scores than the previously built linear model with oversampling. This outcome is potentially caused by the grid approach or the absence of meaningful spatial patterns in accident severity. As a consequence, singular value decomposition is not further employed in this model.

2.3.5 Separate Linear Models

Traffic accidents exhibit a significant diversity in the types of involved vehicles. The characteristics of accidents, whether they involve trucks, motorcycles, bicycles, or pedestrians, vary considerably. As a result, it is plausible to categorize these incidents into distinct models and then integrate them using an ensemble approach.

```r
set.seed(42)

# filter subsets by involved vehicle
accidents_truck <- accidents_oversample |> filter(with_truck == 1)
accidents_motorcycle <- accidents_oversample |> filter(with_motorcycle == 1)
accidents_bicycle <- accidents_oversample |> filter(with_bicycle == 1)
accidents_pedestrian <- accidents_oversample |> filter(with_pedestrian == 1)
accidents_car <- accidents_oversample |> filter(with_car == 1)

# Train separate models, exclude road_category, when accidents_test has
# categories that are not present in accidents_train
```

```r
model_truck <- lm(severity_num ~ .,
                  data = accidents_truck |>
                    select(-severity, -road_category))
model_motorcycle <- lm(severity_num ~ .,
                       data = accidents_motorcycle |>
                         select(-severity, -road_category))
model_bicycle <- lm(severity_num ~ .,
                    data = accidents_bicycle |>
                      select(-severity))
model_pedestrian<- lm(severity_num ~ .,
                      data = accidents_pedestrian |>
                        select(-severity))
model_car <- lm(severity_num ~ .,
                data = accidents_car |>
                  select(-severity))

# Compute prediction, ignore warning "doubtful cases"
suppressWarnings({
  severity_truck <- predict(model_truck, accidents_test)
  severity_motorcycle <- predict(model_motorcycle, accidents_test)
  severity_bicycle <- predict(model_bicycle, accidents_test)
  severity_pedestrian <- predict(model_pedestrian, accidents_test)
  severity_car <- predict(model_car, accidents_test)
})

# combine predictions: consider prediction only if vehicle type is involved
severity_sep <- (severity_truck * accidents_test$with_truck +
                   severity_motorcycle * accidents_test$with_motorcycle +
                   severity_bicycle * accidents_test$with_bicycle +
                   severity_pedestrian * accidents_test$with_pedestrian +
                   severity_car * accidents_test$with_car) /
  (accidents_test$with_truck +
     accidents_test$with_motorcycle +
     accidents_test$with_bicycle +
     accidents_test$with_pedestrian +
     accidents_test$with_car)

# Find the optimal cutoff to convert number predictions into factor
cutoffs <- seq(.35, .55, .005)

# record accuracies
accuracies <- c()
f1_scores <- c()
balanced_accuracies <- c()

#try each cutoff
for (cutoff_sep in cutoffs) {
  # convert using cutoff
  severity_test <- ifelse((severity_sep >= cutoff_sep), 1, 0)
  severity_test <- factor(severity_test, levels = 0:1, labels = c("light", "severe"))

  # record accuracy
  a <- accuracy(accidents_test$severity, severity_test)
```

```
  #if NA set to 0
  if (is.na(a["F1"]))
    a[] <- 0
  accuracies <- c(accuracies, a["Accuracy"])
  f1_scores <- c(f1_scores, a["F1"])
  balanced_accuracies <- c(balanced_accuracies, a["Balanced Accuracy"])
}
# where is F1 score maximum
i <- which.max(f1_scores)
cutoff_sep <- cutoffs[i]

# use optimal cutoff for conversion
severity_sep <- ifelse((severity_sep >= cutoff_sep), 1, 0)
severity_sep <- factor(severity_sep, levels = 0:1, labels = c("light", "severe"))

# Record result
a <- accuracy(accidents_test$severity, severity_sep)
evaluation <- rbind(evaluation, tibble(data = "model", model = "separate linear models",
                                       accuracy = a["Accuracy"], f1 = a["F1"],
                                       balanced = a["Balanced Accuracy"]))
evaluation
```

```
## # A tibble: 5 x 5
##   data  model                          accuracy     f1 balanced
##   <chr> <chr>                             <dbl>  <dbl>    <dbl>
## 1 model simple median                     0.806 NA        0.5
## 2 model linear model                      0.659  0.408    0.640
## 3 model linear model with oversample      0.708  0.399    0.629
## 4 model linear model with SVD             0.706  0.386    0.619
## 5 model separate linear models            0.690  0.418    0.647
```

```
rm(accidents_truck, accidents_motorcycle, accidents_bicycle, accidents_pedestrian,
   accidents_car, severity_truck, severity_motorcycle, severity_bicycle,
   severity_pedestrian, severity_car)
```

2.3.6 Random Forest Model

In the following model, a random forest model is constructed. The technique is highly suitable for classification and can predict the severity factor without conversion.

To optimize efficiency, a preliminary step involves conducting a correlation analysis with of each feature with accident severity to identify the most important features. Spearman correlation is applied to accomodate categorical features, which have an order (e.g. *road_condition* with levels "dry,"wet" and "slippery"). Categorical features without implicit order are ranked by the average accident severity across all accidents, facilitating the correlation analysis to pinpoint the most important features.

Different values for *mtry* and *nodesize* were tested, however, the default values for these parameters provided the best results.

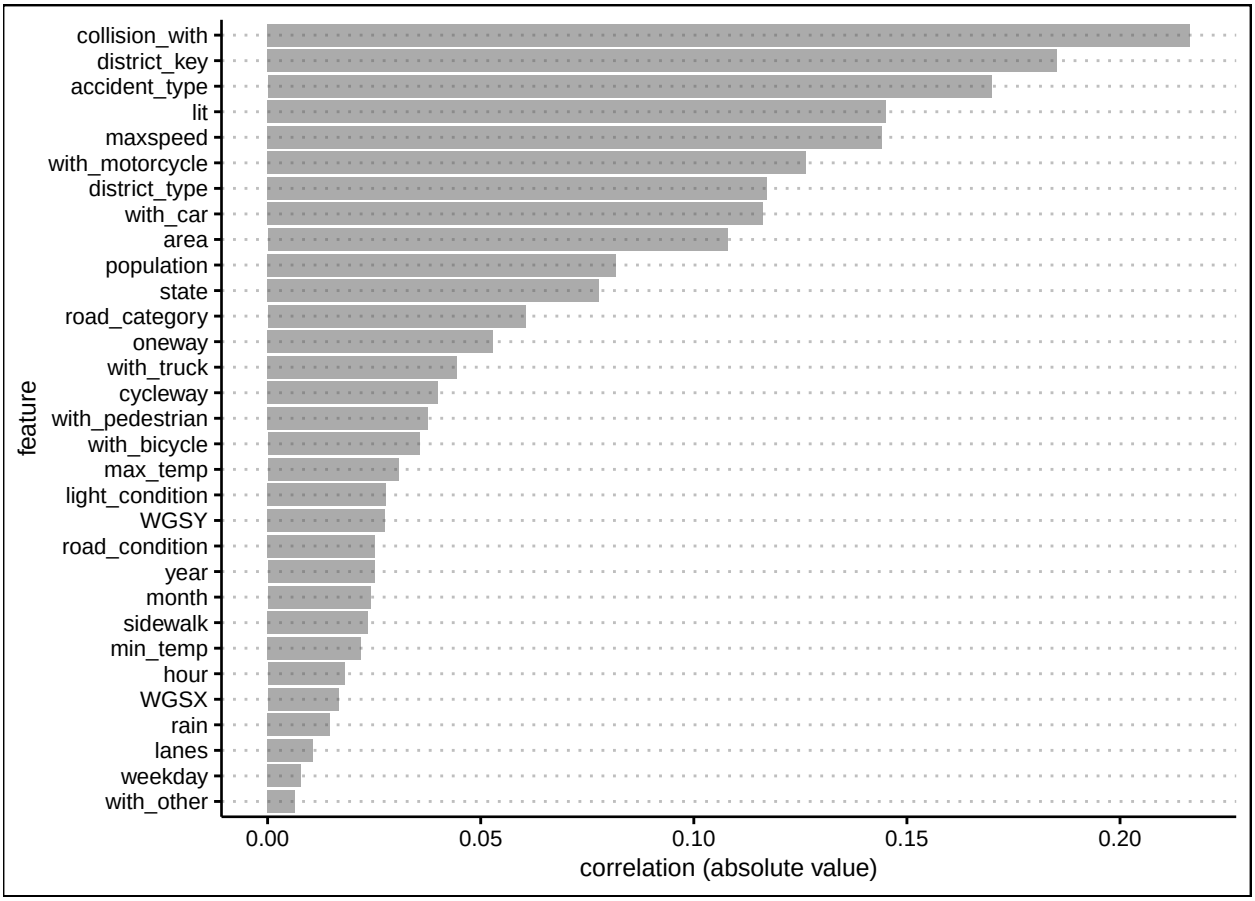
The model is trained using the training data that incorporates oversampling of severe accidents.

```
# unordered factors will be ordered by severity ratio
state_order <- accidents_oversample |>
```

```
  group_by(state, severity) |> summarize(count = n(), .groups = "keep") |> ungroup() |>
  spread(key = severity, value = count) |>
  mutate(ratio = severe / (severe + light),
         state = fct_reorder(state, ratio, .fun=sum)) |>
  arrange(ratio) |> pull(state)

collision_with_order <- accidents_oversample |>
  group_by(collision_with, severity) |> summarize(count = n(), .groups = "keep") |> ungroup() |>
  spread(key = severity, value = count) |>
  mutate(ratio = severe / (severe + light),
         collision_with = fct_reorder(collision_with, ratio, .fun=sum)) |>
  arrange(ratio) |> pull(collision_with)

accident_type_order <- accidents_oversample |>
  group_by(accident_type, severity) |> summarize(count = n(), .groups = "keep") |> ungroup() |>
  spread(key = severity, value = count) |>
  mutate(ratio = severe / (severe + light),
         accident_type = fct_reorder(accident_type, ratio, .fun=sum)) |>
  arrange(ratio) |> pull(accident_type)

district_key_order <- accidents_oversample |>
  group_by(district_key, severity) |> summarize(count = n(), .groups = "keep") |> ungroup() |>
  spread(key = severity, value = count) |>
  mutate(ratio = severe / (severe + light),
         district_key = fct_reorder(district_key, ratio, .fun=sum)) |>
  arrange(ratio) |> pull(district_key)

district_type_order <- accidents_oversample |>
  group_by(district_type, severity) |> summarize(count = n(), .groups = "keep") |> ungroup() |>
  spread(key = severity, value = count) |>
  mutate(ratio = severe / (severe + light),
         district_type = fct_reorder(district_type, ratio, .fun=sum)) |>
  arrange(ratio) |> pull(district_type)

road_category_order <- accidents_oversample |>
  group_by(road_category, severity) |> summarize(count = n(), .groups = "keep") |> ungroup() |>
  spread(key = severity, value = count) |>
  mutate(ratio = severe / (severe + light),
         road_category = fct_reorder(road_category, ratio, .fun=sum)) |>
  arrange(ratio) |> pull(road_category)

# convert all unordered factors to factors in new order
features <- accidents_oversample  |>
  select( severity, year, month, hour, weekday, light_condition, road_condition,
          WGSX, WGSY, population, district_type, area, min_temp, max_temp, rain,
          collision_with, accident_type, with_bicycle, with_car, with_pedestrian,
          with_motorcycle, with_truck, with_other, district_key, state,
          road_category, lanes, maxspeed, lit, oneway, sidewalk, cycleway) |>
  mutate(district_type = factor(as.character(district_type),
                                levels = district_type_order),
         collision_with = factor(as.character(collision_with),
                                 levels = collision_with_order),
         accident_type = factor(as.character(accident_type),
```

```
                                levels = accident_type_order),
         district_key = factor(district_key, levels = district_key_order),
         state = factor(state, levels = state_order),
         road_category = factor(as.character(road_category),
                                levels = road_category_order))

# convert all features to numerical valuesfor correlation
features <- features  |>
  mutate(severity = as.integer(severity), year = as.integer(year),
         month = as.integer(month), hour = as.integer(hour),
         weekday = as.integer(weekday), light_condition = as.integer(light_condition),
         road_condition = as.integer(road_condition),
         district_type = as.integer(district_type),
         collision_with = as.integer(collision_with),
         accident_type = as.integer(accident_type),
         with_bicycle = as.integer(with_bicycle), with_car = as.integer(with_car),
         with_pedestrian = as.integer(with_pedestrian),
         with_motorcycle = as.integer(with_motorcycle),
         with_truck = as.integer(with_truck), with_other = as.integer(with_other),
         district_key = as.integer(district_key), state = as.integer(state),
         road_category = as.integer(road_category), lit = as.integer(lit),
         oneway = as.integer(oneway), sidewalk = as.integer(sidewalk),
         cycleway = as.integer(cycleway))

# calculate all correlations with severity (absolute value)
corr <- abs(cor(features, method = "spearman")[-1,1])
```

The variables collision_with, accident_type, maxspeed, lit, with_motorcycle, and with_car exhibit the highest correlation with accident severity. The variables area, population, district_key, and district_type also appear in the top ten correlated variables; however, underlying interdependencies are not visualized in this analysis.



The random forest model is now trained using the features exhibiting the highest correlation with the severity variable. The training utilizes the dataset with oversampling.

```
# train model with select features
model_orf <- randomForest(severity ~ collision_with + accident_type + lit + maxspeed +
                                     with_motorcycle + district_type + with_car +
                                     area + population + road_category + oneway +
                                     with_truck + cycleway + with_pedestrian + with_bicycle,
                          data = accidents_oversample)

# compute prediction
severity_orf <- predict(model_orf, accidents_test)
```

The table records the accuracy, balanced accuracy, and F1 score of the models:

```
## # A tibble: 6 x 5
##   data  model                          accuracy     f1 balanced
##   <chr> <chr>                             <dbl>  <dbl>    <dbl>
## 1 model simple median                     0.806 NA        0.5
## 2 model linear model                      0.659  0.408    0.640
## 3 model linear model with oversample      0.708  0.399    0.629
## 4 model linear model with SVD             0.706  0.386    0.619
## 5 model separate linear models            0.690  0.418    0.647
## 6 model random forest                     0.714  0.405    0.633
```

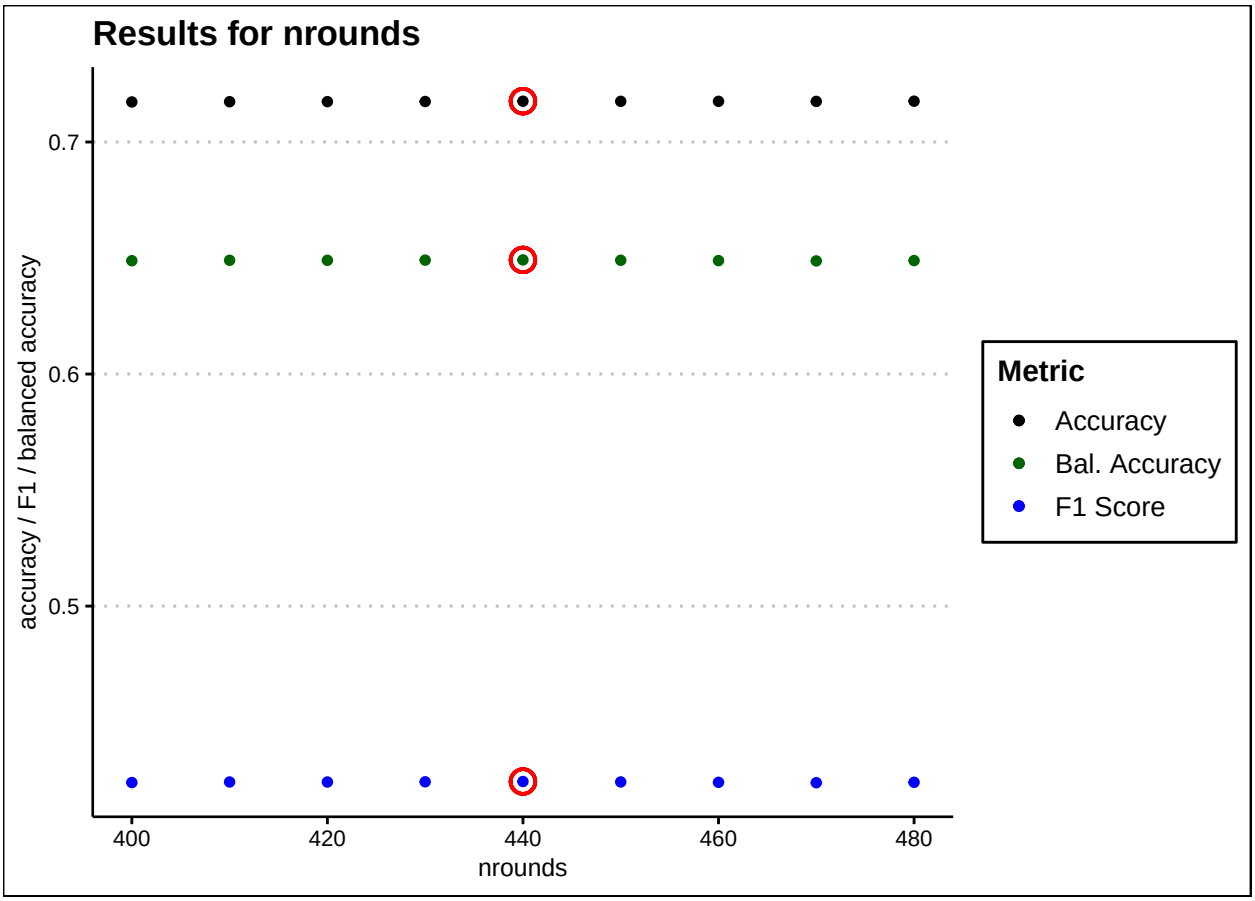
2.3.7 Extreme Gradient Boosting

Utilizing the extreme gradient boosting package, a gradient boosting model with a tree model has been implemented. The training of the model involves oversampling severe accidents within the training data. In order to comply with the requirements of the model, all factor variables are transformed into binary features {0, 1} for each factor level.

Cross-validation is employed with parallel processing to determine the optimal value for *nrounds* with a very small *eta* of 0.025. While different values for *max_depth* and *eta* were tested during model design, the script refrains from performing cross-validation for these parameters in the interest of reasonable computing time.

```
# convert training data into matrix, factor levels transformed into binary features
train_matrix <- sparse.model.matrix(severity_num ~ .,
                                    data = accidents_oversample |> select(-severity))

# convert test data into matrix with binary features, too
test_matrix <- sparse.model.matrix(severity_num ~ .,
                                   data = accidents_test |> select(-severity))

# enable parallel procesing with all available cores
registerDoParallel(cores=detectCores())

# Test different numbers for nrounds
eta <- .025
nrounds_values <- seq(400, 480, 10)

# Parallel processing with doParallel
a <- foreach (i = 1:length(nrounds_values), .combine = c) %dopar% {
  set.seed(42)

  # train model
  model_xgb <- xgboost(data = train_matrix,
                       label = accidents_oversample$severity_num,
                       eta = eta,
                       max_depth = 7,
                       nrounds = nrounds_values[i],
                       objective = "binary:logistic",
                       verbose = 0)

  # Compute prediction with rounding as simple conversion
  severity_xgb <- predict(model_xgb, test_matrix)
  severity_xgb <- round(severity_xgb)
  severity_xgb <- factor(severity_xgb, levels = 0:1, labels = c("light", "severe"))

  # record accuracy
  accuracy(accidents_test$severity, severity_xgb)
}

# extract scores from result
accuracies <- a[names(a) == "Accuracy"]
f1_scores <- a[names(a) == "F1"]
balanced_accuracies <-  a[names(a) == "Balanced Accuracy"]

# where is the maximum f1 score?
```

```r
i <- which.max(f1_scores)
nrounds <- nrounds_values[i]
```



The graph above shows the resulting accuracies and F1 scores for each tested *nrounds*. The highlighted points have been selected to build the following models.

The gradient boosting model is subsequently trained with the selected parameter.

```r
set.seed(42)

# train model
model_xgb <- xgboost(data = train_matrix,
                     label = accidents_oversample$severity_num,
                     eta = eta,
                     max_depth = 7,
                     nrounds = nrounds_values[i],
                     objective = "binary:logistic",
                     verbose = 0)

# Compute prediction
severity_xgb <- predict(model_xgb, test_matrix)
```

```r
# Find the optimal cutoff to convert number predictions into factor
cutoffs <- seq(.35, .55, .005)

# record accuracies
accuracies <- c()
f1_scores <- c()
balanced_accuracies <- c()

#try each cutoff
for (cutoff_xgb in cutoffs) {
  # convert using cutoff
  severity_test <- ifelse((severity_xgb >= cutoff_xgb), 1, 0)
  severity_test <- factor(severity_test, levels = 0:1, labels = c("light", "severe"))

  # record accuracy
  a <- accuracy(accidents_test$severity, severity_test)

  #if NA set to 0
  if (is.na(a["F1"]))
    a[] <- 0
  accuracies <- c(accuracies, a["Accuracy"])
  f1_scores <- c(f1_scores, a["F1"])
  balanced_accuracies <- c(balanced_accuracies, a["Balanced Accuracy"])
}
# where is F1 score maximum?
i <- which.max(f1_scores)
cutoff_xgb <- cutoffs[i]

# use optimal cutoff for conversion
severity_xgb <- ifelse((severity_xgb >= cutoff_xgb), 1, 0)
severity_xgb <- factor(severity_xgb, levels = 0:1, labels = c("light", "severe"))
```

The table records the accuracy, balanced accuracy, and F1 score of the models:

```
## # A tibble: 7 x 5
##   data  model                          accuracy     f1 balanced
##   <chr> <chr>                             <dbl>  <dbl>    <dbl>
## 1 model simple median                     0.806 NA        0.5
## 2 model linear model                      0.659  0.408    0.640
## 3 model linear model with oversample      0.708  0.399    0.629
## 4 model linear model with SVD             0.706  0.386    0.619
## 5 model separate linear models            0.690  0.418    0.647
## 6 model random forest                     0.714  0.405    0.633
## 7 model extreme gradient boosting         0.694  0.428    0.655
```

2.3.8 Final Ensemble Model

The final model combines the previous three models in an ensemble approach. All three models show similar accuracy, balanced accuracy, and F1 scores. Consequently, all have the same vote, and a majority approach is used to predict a severe accident.

```r
# Compute prediction
severity_ens <- factor(ifelse((severity_sep == "severe") +
                              (severity_orf == "severe") +
                              (severity_xgb == "severe") >= 2,
                              1,
                              0),
                       levels = 0:1,
                       labels = c("light", "severe"))
```

The table records the accuracy, balanced accuracy, and F1 score of the models:

```
## # A tibble: 8 x 5
##   data  model                          accuracy     f1 balanced
##   <chr> <chr>                             <dbl>  <dbl>    <dbl>
## 1 model simple median                     0.806 NA        0.5
## 2 model linear model                      0.659  0.408    0.640
## 3 model linear model with oversample      0.708  0.399    0.629
## 4 model linear model with SVD             0.706  0.386    0.619
## 5 model separate linear models            0.690  0.418    0.647
## 6 model random forest                     0.714  0.405    0.633
## 7 model extreme gradient boosting         0.694  0.428    0.655
## 8 model ensemble                          0.704  0.427    0.653
```

This comprehensive approach, combining oversampling, linear modeling, random forest, extreme gradient boosting with the ensemble approach, provided a favorable balance of accuracy, balanced accuracy, and F1-score.

2.4 Model Analysis

This chapter dwelves into the additional analysis of models constructed in the preceding section. The application of linear models yields insights into the effect size of the features and potentially represents the models' most significant contribution. Understanding the nature of severe accidents can offer crucial insights for prevention.

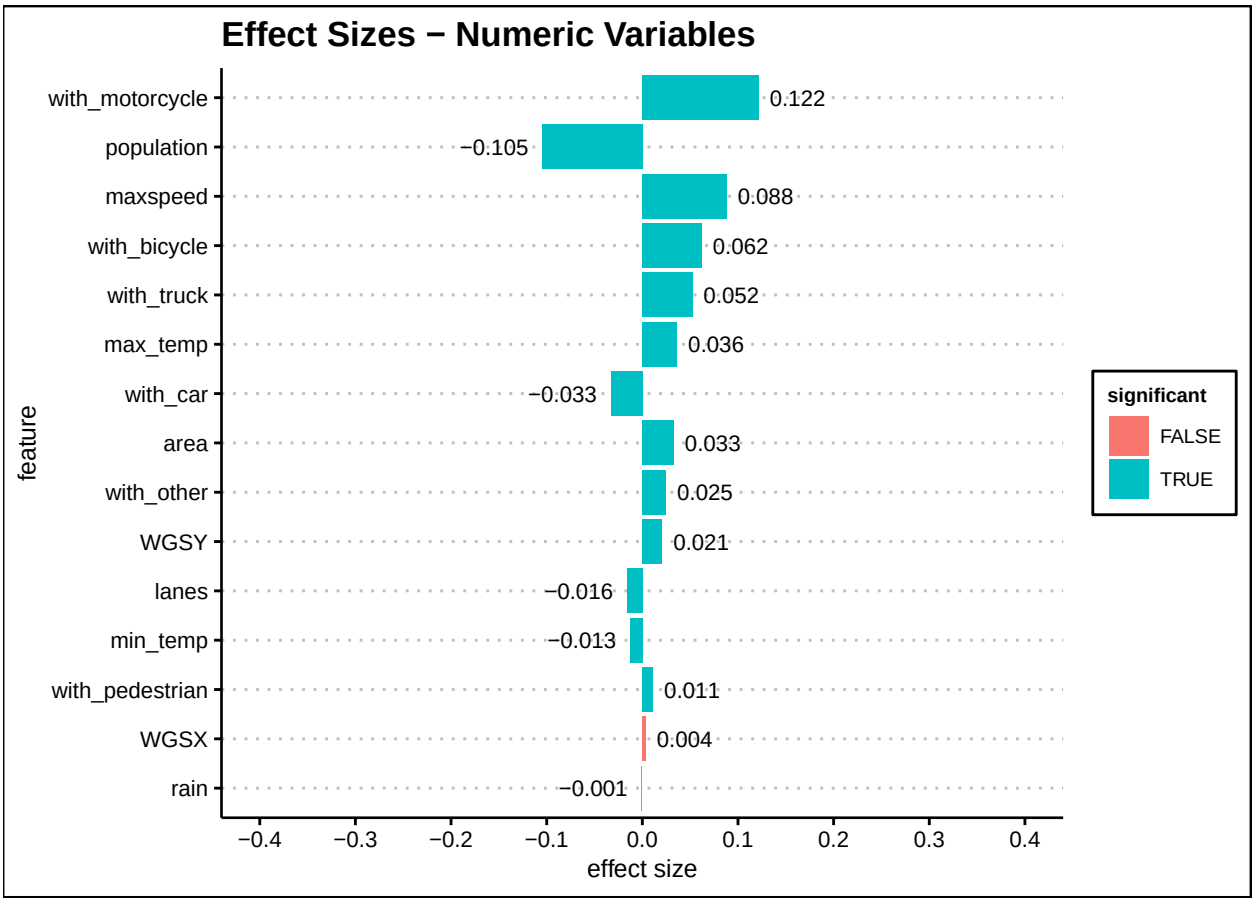
An analysis of the estimates for each feature in the linear model with oversampling is presented. To ensure comparability of effect sizes, all numerical variables are normalized. Direct comparisons are made for numerical features, while factors are analyzed individually. The confidence level is set to 5% for evaluating the significance of the model estimates.

Furthermore, insights into the importance of features for the severity of accidents are obtained by studying the extreme gradient boosting model. The model provides a dedicated function that allows the extraction of accuracy gains associated with each feature.

This detailed analysis enhances our understanding of the impact and relevance of individual features in predicting accident severity.

```r
# Norm all numerical variables so that effect sizes become comparable
preprocessParams <- preProcess(accidents_oversample, method = c("center", "scale"))
accidents_oversample_standardized <- predict(preprocessParams, accidents_oversample)

# Train model (without district_key as this has lots of factor levels)
model_olm_std <- lm(severity_num ~ . , data = accidents_oversample_standardized |>
                                              select(-severity, -district_key))
```

```r
# red effect sizes and p values into data frame
effect_sizes <- summary(model_olm_std)$coefficients[,"Estimate"]
p_values <- summary(model_olm_std)$coefficients[,"Pr(>|t|)"]
```

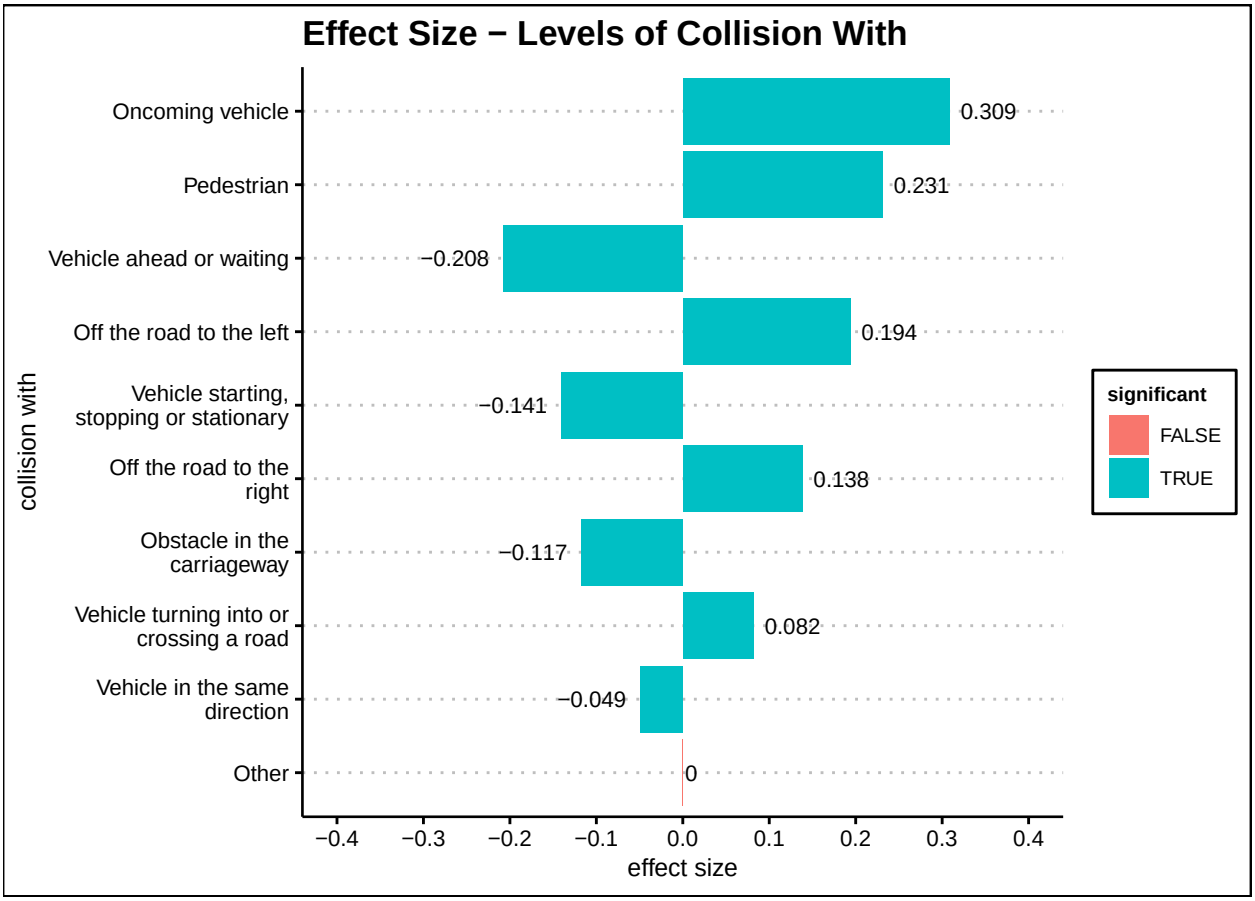


The above graph shows that involved vehicle types have a significant impact on the severity of accidents, with trucks playing a different role than motorcycles and bicycles, as they pose a danger to other drivers.

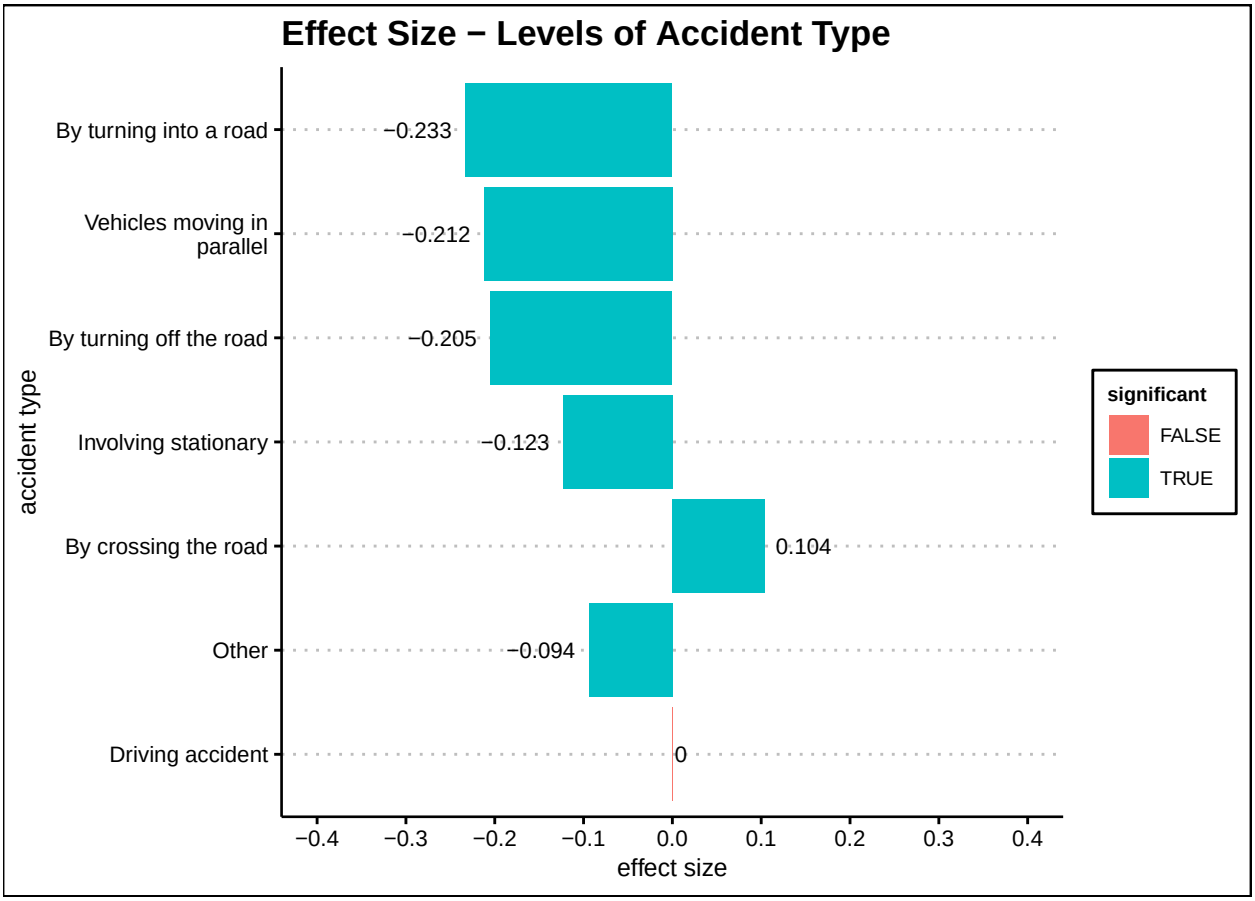
Population size in administrative areas has also a big impact on the likelihood of severe accidents. Traffic speeds are lower in cities, leading to less severe accidents. The maximum speed is another factor leading to more severe accidents.

It is interesting to note that the maximum daily temperature has a more pronounced impact on severity than the minimum temperature. However, these results should be treated with some caution due to the limited accuracy of weather information caused by the missing date information in the original data.

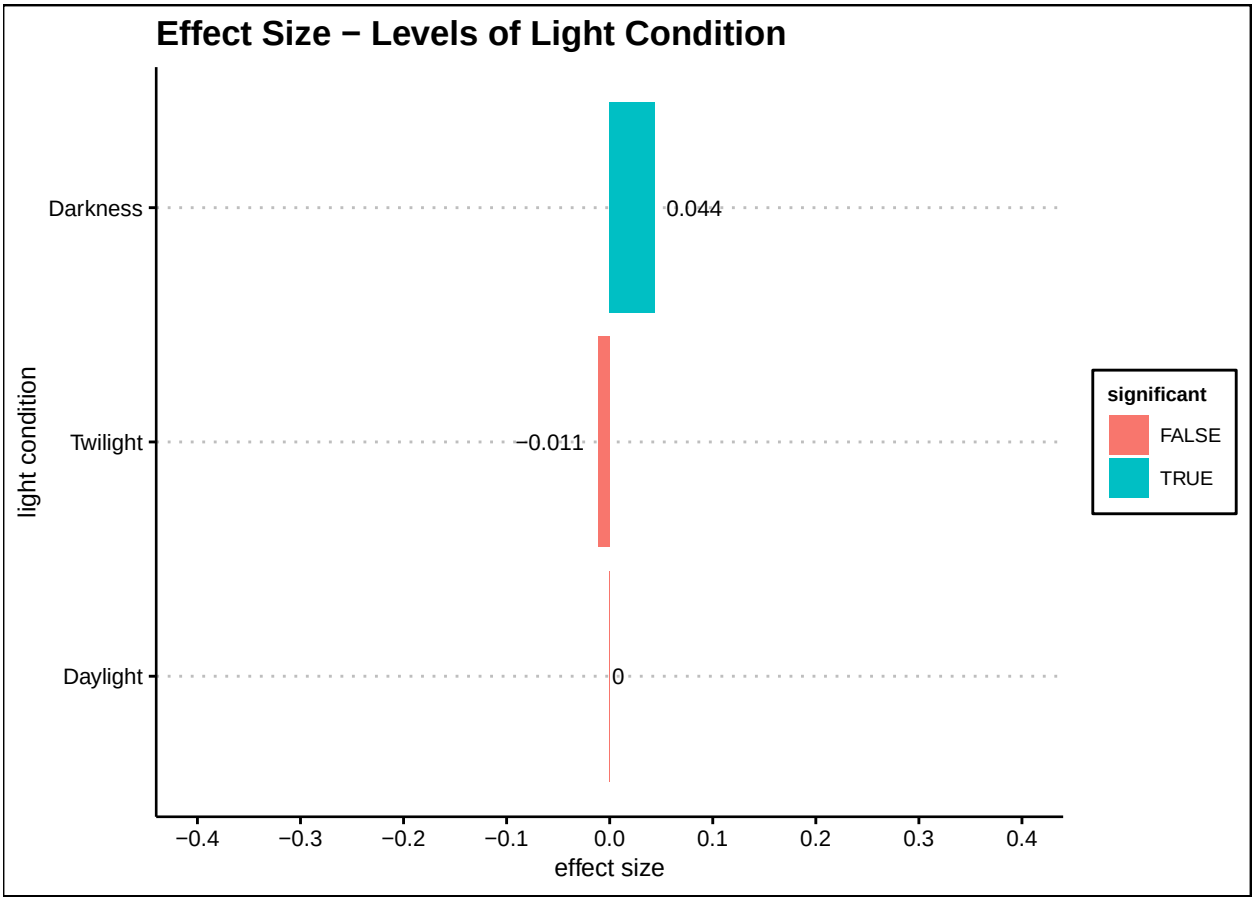
When examining the factor variables in the following section, the effect sizes are relative effects compared to the first factor level.



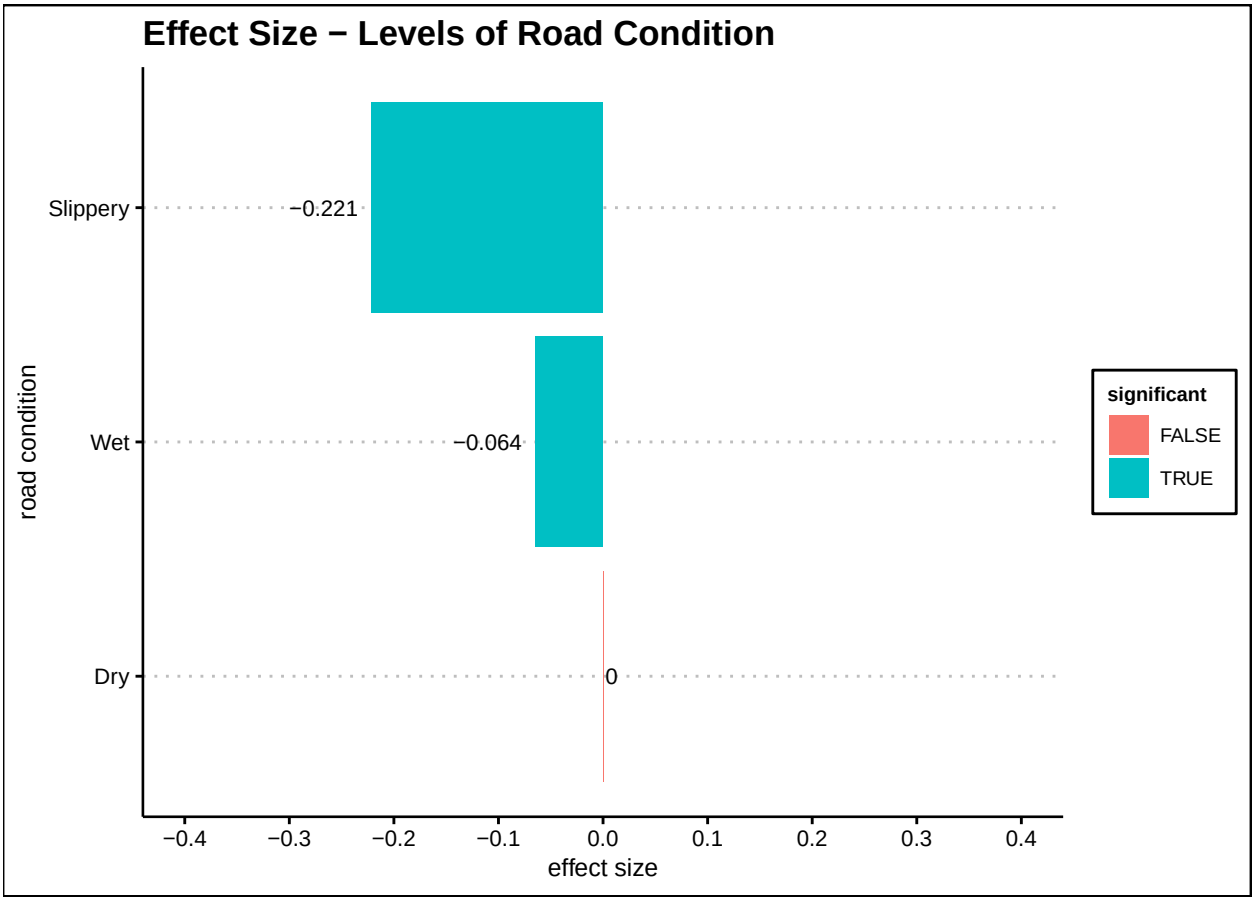
The first graph shows that most severe accidents result from oncoming vehicles, pedestrians, and getting off the road.



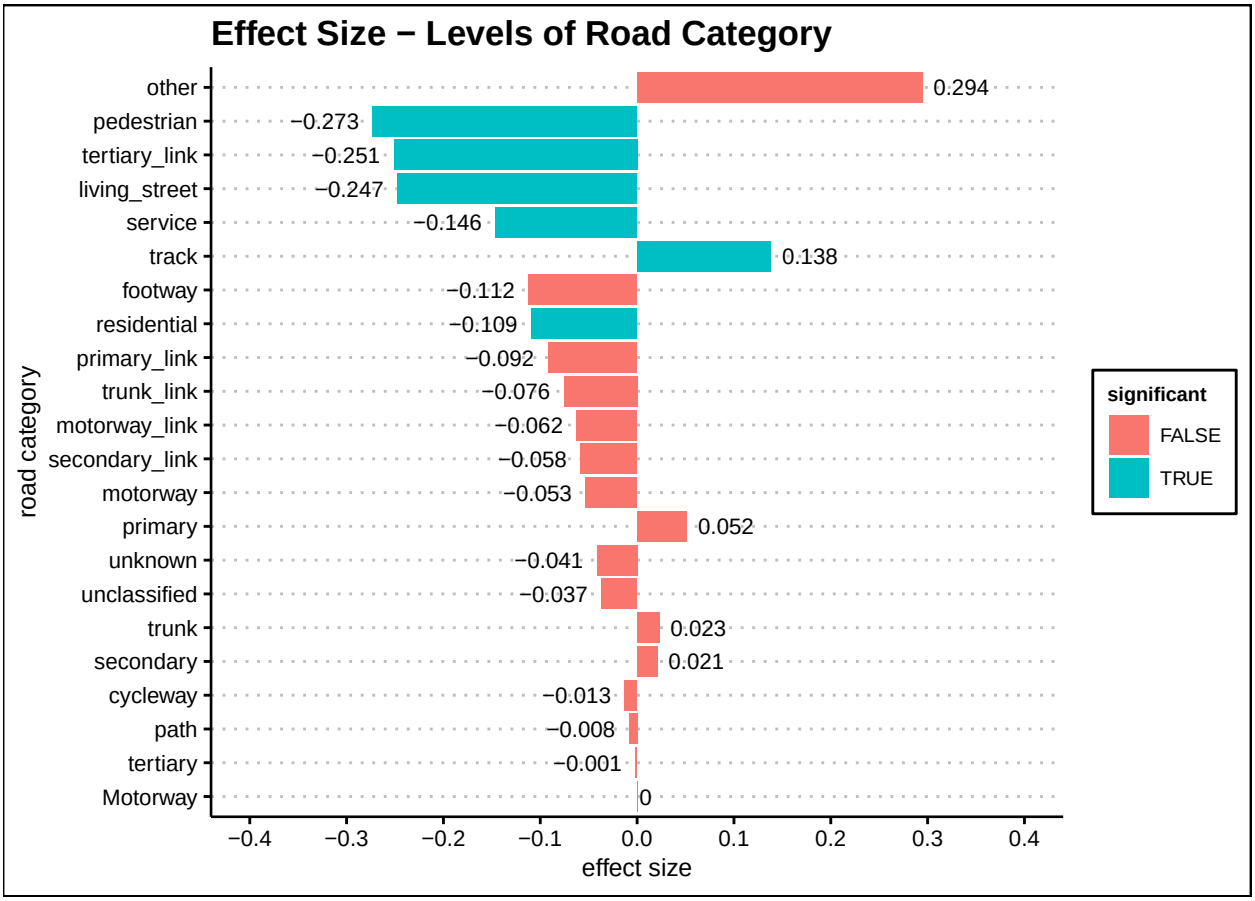
The effect of accident type in the next graph is relative to the level "driving accidents" which seems to serve as a generic type. The other accident types generally lead to less severe accidents, except for road crossings.



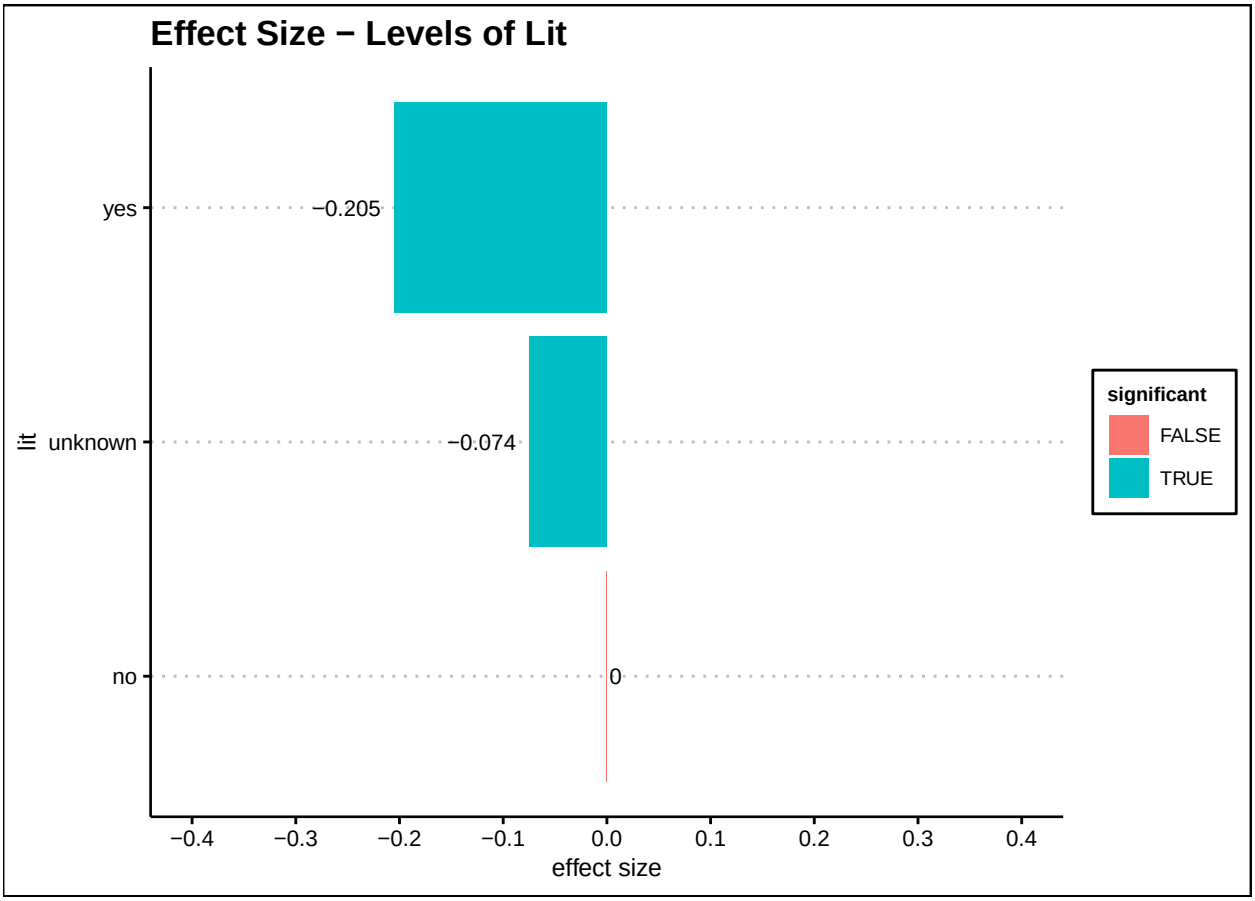
Accidents in darkness have a small tendency to be more severe as the following graph shows.



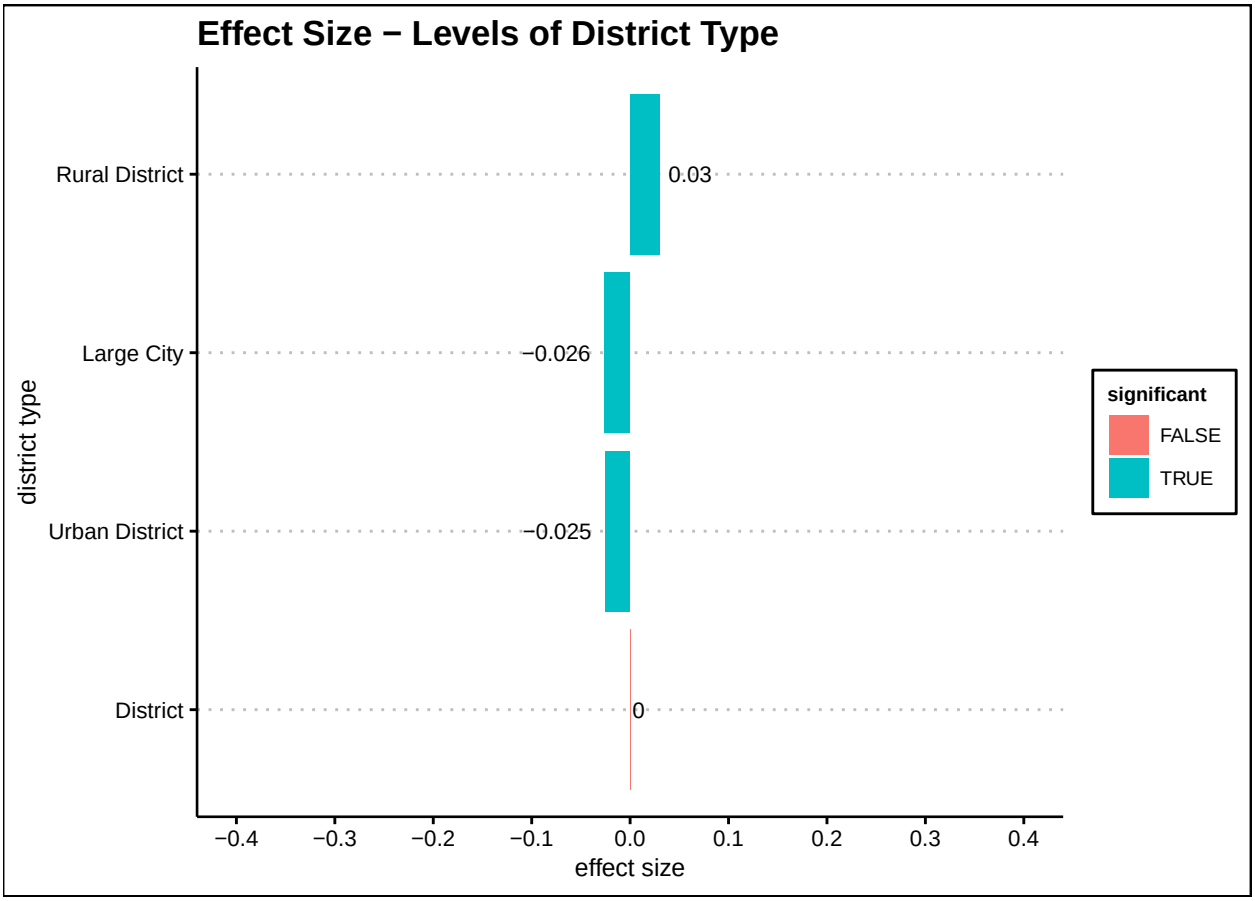
The results for road condition in the above graph are a bit surprising as dry roads show highest severity.



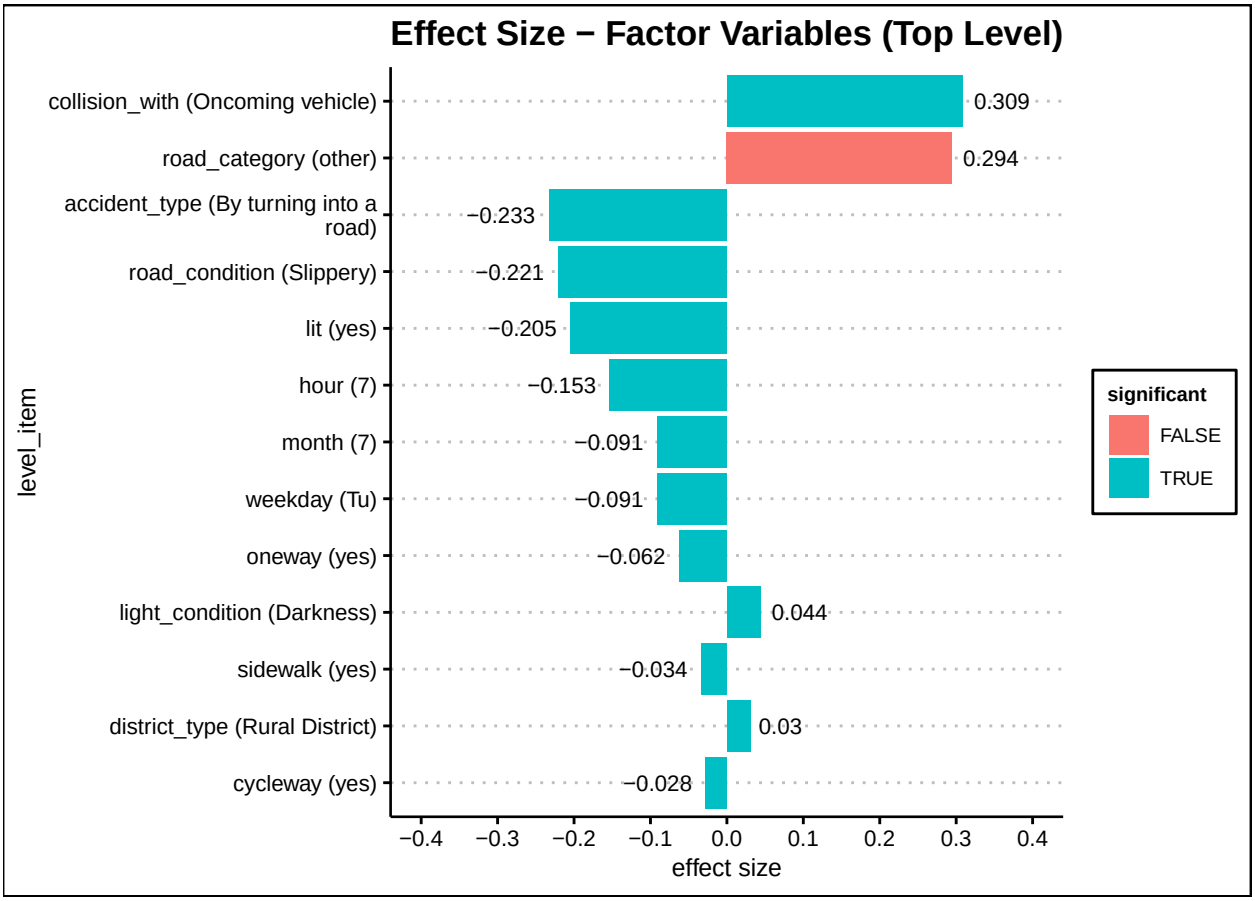
Road category shows few significant effects on accident severity. For those categories, that show a significant effect, the number of accidents must be considered, too (see section 2.2.2). residential roads stand out with over 10,000 accidents and a significant effect size. When compared to accidents on motorways, incidents on residential streets tend to be less severe.



The presence of streetlights is associated with a reduction in the severity of accidents. This effect could stem directly from the illumination provided by streetlights or indirectly result from their typical placement in areas characterized by slower traffic.



Large cities and urban areas have slower traffic. The lower effect sizes shown in the graph below are plausible.

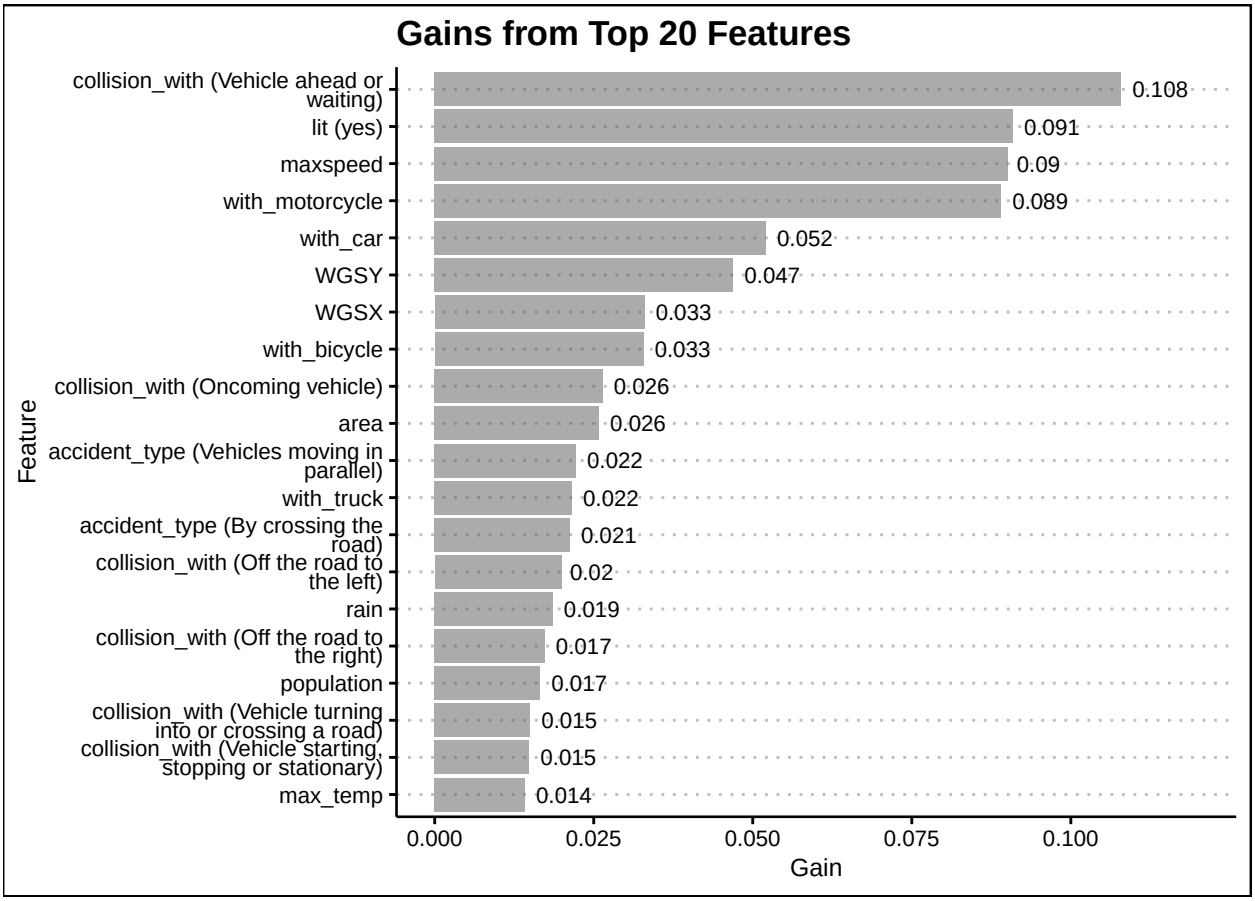


The summary graph above highlights the effect sizes of the factor levels with the most substantial impact for each factor. A significant portion of these effects is likely attributed to the slower speed associated with these factor levels, such as reduced speed when turning into a road or encountering slippery conditions.

Additionally, collisions with oncoming vehicles emerge as a factor contributing to more severe accidents, aligning with intuitive expectations.

Following this, an exploration into the feature importance within the extreme gradient boosting model is conducted. It's important to note that direct comparisons with the linear model results are not feasible, as factor variables have been recoded into binary features specifically for the extreme gradient boosting model.

```
# get importance data from the model
importance_df <- xgb.importance(feature_names = colnames(train_matrix), model = model_xgb)
```



The graph above shows the accuracy gains from the top 20 features in the extreme gradient boosting model. Overall, the results are consistent with the effect sizes in the linear model: lit roads, maximum speed and accidents with motorcycles are important factors for the severity of accidents.

The feature importance analysis, unfortunately, does not offer clarity on whether the effect on accident severity is positive or negative. Consequently, it remains unclear, whether the results for collisions with vehicles ahead or waiting, in comparison to oncoming vehicles, align with the linear model results.

3 Results

This chapter delves into the testing of the previously developed models, presenting the final test results. The models undergo testing using the *final_holdout_test* dataset specifically set aside for this purpose. This testing process provides insights into the performance and predictive capabilities of the models on previously unseen data.

The test results are summarized at the end of this chapter.

3.1 Model Testing

The process begins the combination of separate models for each vehicle type. Subsequently, the random forest model is tested. Next, the extreme gradient boosting model undergoes testing. Finally, the ensemble model is subjected to testing.

For simplicity, the models are not retrained with the complete *accidents_model* dataset; instead, the existing models that were trained with the subset *accident_train* are reused.

Separate Linear Models

The next model tested utilizes separate models for each involved vehicle.

```
# Compute prediction, ignore warning "doubtful cases"
suppressWarnings({
  severity_truck <- predict(model_truck, accidents_final_test)
  severity_motorcycle <- predict(model_motorcycle, accidents_final_test)
  severity_bicycle <- predict(model_bicycle, accidents_final_test)
  severity_pedestrian <- predict(model_pedestrian, accidents_final_test)
  severity_car <- predict(model_car, accidents_final_test)
})

# combine predictions: consider prediction only if vehicle type is involved
severity_sep <- (severity_truck * accidents_final_test$with_truck +
                   severity_motorcycle * accidents_final_test$with_motorcycle +
                   severity_bicycle * accidents_final_test$with_bicycle +
                   severity_pedestrian * accidents_final_test$with_pedestrian +
                   severity_car * accidents_final_test$with_car) /
  (accidents_final_test$with_truck +
     accidents_final_test$with_motorcycle +
     accidents_final_test$with_bicycle +
     accidents_final_test$with_pedestrian +
     accidents_final_test$with_car)

# convert to factor using cutoff
severity_sep <- ifelse((severity_sep >= cutoff_sep), 1, 0)
severity_sep <- factor(severity_sep, levels = 0:1, labels = c("light", "severe"))
```

Random Forest Model

The following test evaluates the random forest model.

```
# compute prediction
severity_orf <- predict(model_orf, accidents_final_test)
```

Extreme Gradient Boosting

The following test evaluates the extreme gradient boosting model.

```
# convert test data into matrix with all factor levels transformed into binary features
test_matrix <- sparse.model.matrix(severity_num ~ .,
                                   data = accidents_final_test |> select(-severity ) )

# Compute prediction
severity_xgb <- predict(model_xgb, test_matrix)

# use optimal cutoff for conversion
severity_xgb <- ifelse((severity_xgb >= cutoff_xgb), 1, 0)
severity_xgb <- factor(severity_xgb, levels = 0:1, labels = c("light", "severe"))
```

Final Ensemble Model

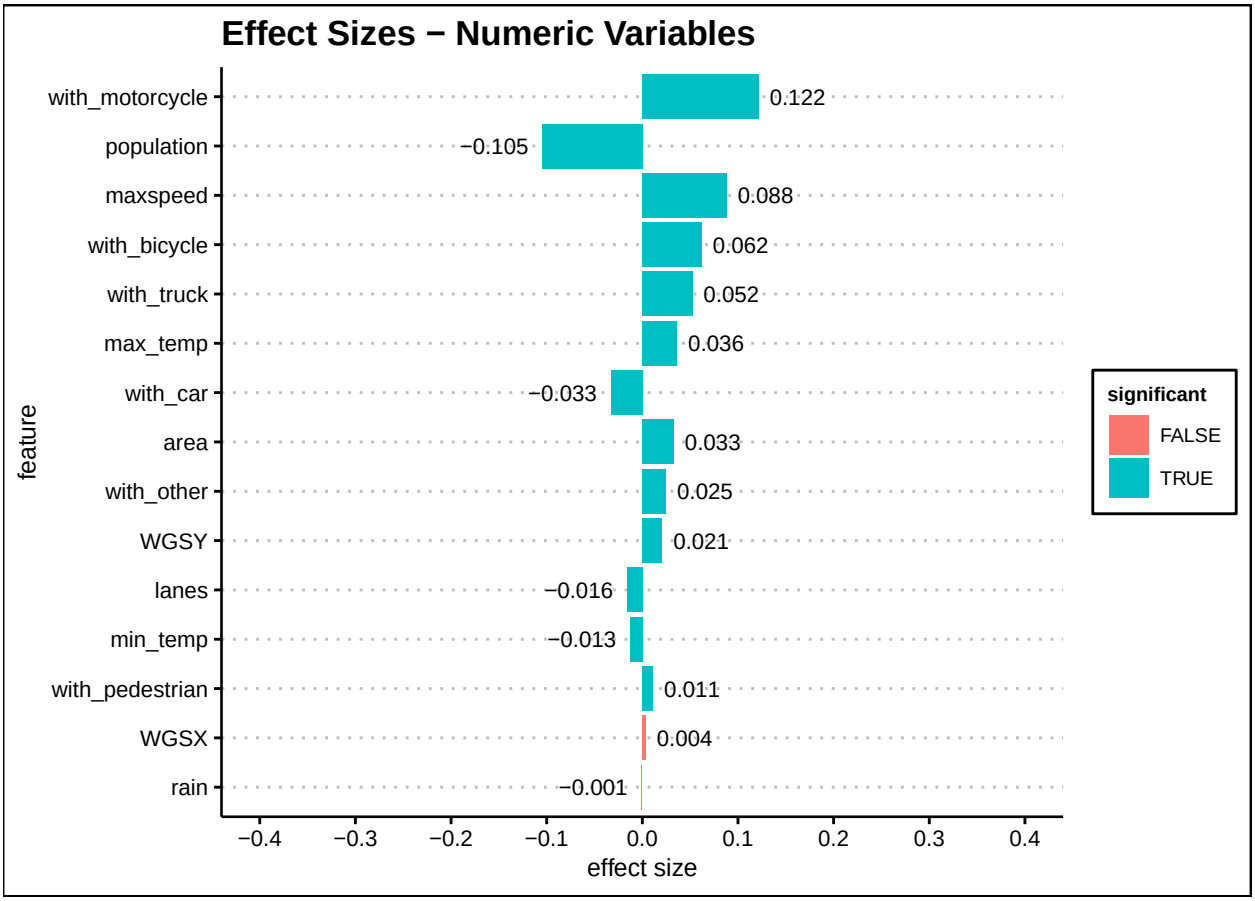
The last tested model combines the previous three models.

```r
# Compute prediction
severity_ens <- factor(ifelse((severity_sep == "severe") +
                                (severity_orf == "severe") +
                                (severity_xgb == "severe") >= 2,
                              1,
                              0),
                       levels = 0:1,
                       labels = c("light", "severe"))
```

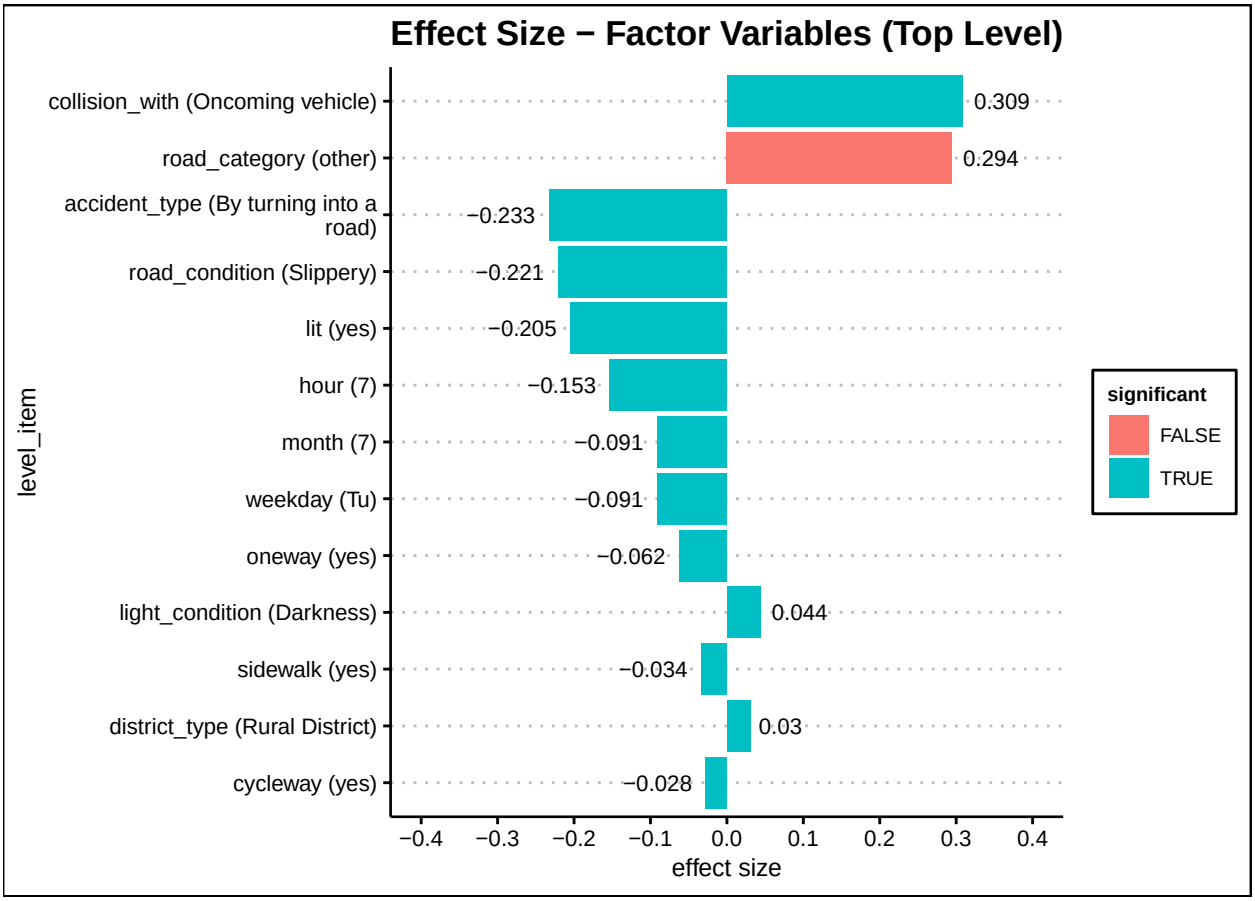
3.2 Result Summary

The table below provides an overview of accuracy, balanced accuracy, and F1 for the various models during both the model development and test phases. It's worth noting that enhancements to one parameter frequently resulted in a negative impact on another. The extreme gradient boosting model shows the highest F1 score, while the ensemble model provides a favorable balance between accuracy, F1 score, and balanced accuracy.

```
## # A tibble: 12 x 5
##    data       model                          accuracy     f1 balanced
##    <chr>      <chr>                             <dbl>  <dbl>    <dbl>
##  1 model      simple median                     0.806 NA        0.5
##  2 model      linear model                      0.659  0.408    0.640
##  3 model      linear model with oversample      0.708  0.399    0.629
##  4 model      linear model with SVD             0.706  0.386    0.619
##  5 model      separate linear models            0.690  0.418    0.647
##  6 model      random forest                     0.714  0.405    0.633
##  7 model      extreme gradient boosting         0.694  0.428    0.655
##  8 model      ensemble                          0.704  0.427    0.653
##  9 final test separate linear models            0.691  0.419    0.648
## 10 final test random forest                     0.715  0.405    0.634
## 11 final test extreme gradient boosting         0.694  0.428    0.655
## 12 final test ensemble                          0.704  0.426    0.653
```



The above graph illustrates the effect sizes of numerical features in the linear model, offering valuable insights for further analysis and accident prevention. Accidents with motorcycles, population size and maximium speed emerge as the most influential factors on accident severity.



The above graph depicts the effect sizes of factor levels with the most substantial impact in the linear model. Accidents involving collisions with oncoming vehicles tend to result in more severe accidents, whereas turning into a road, slippery road conditions, and lit roads are associated with less severe accidents.

4 Discussion

Various data sources have been successfully combined for this comprehensive analysis. The descriptive analysis and predictive models have yielded valuable insights into traffic accidents in Germany and highlight some causes of severe accidents. Various modeling techniques, including oversampling, linear modeling, random forest, and extreme gradient boosting, have been effectively combined to construct a comprehensive ensemble model.

However, the accuracy, F1 score, and balanced accuracy of the models are constrained by two primary limitations. The first limitation is associated with the available data. The accident data exclusively covers incidents involving injuries or fatalities, lacking information on lighter accidents. This limitation restricts the contrast in accident severity within the dataset. Additionally, the absence of the exact date in the accident data hinders precise mapping of weather information. As revealed in the model analysis, weather variables demonstrated minimal predictive value as a consequence of this limitation.

The second limitation is the available computing power. The random forest model could not be computed with all feature variables. Incorporating all variables was not feasible due to limited computing resources. Furthermore, the computation of singular value decomposition was only viable by calculating a grid for longitude and latitude. It was not feasible with all individual geo coordinates.

The following areas offer interesting potential for further extension of the work:

- **Detailed accident data**: The Geo Portal NRW offers accident data for the state NRW, including exact date and time. This data could be leveraged to train an auxiliary model for better assessing the impact of weather.
- **Road geometry**: OpenStreetMap data could be utilized to calculate road geometries such as curvatures, grades, intersections, and junction types. This additional information can contribute to a more nuanced analysis and understanding of the road infrastructure, aiding in the assessment of its impact on accident severity.
- **Advanced model evaluation**: Exploring more sophisticated model evaluation techniques like Shapley Additive Explanations (SHAP) could provide further insights from the models.

In summary, the project provides valuable insights into the causes of serious road accidents in Germany and has served as a great opportunity to explore some interesting machine learning approaches. The outlined areas for extension offer avenues for enhancing the depth and breadth of the analysis.

5 References

Ahmed, S., Hossain, M. A., Ray, S. K., Bhuiyan, M. M. I., & Sabuj, S. R. (2023). *A study on road accident prediction and contributing factors using explainable machine learning models: analysis and performance.* Transportation research interdisciplinary perspectives, 19, 100814. https://www.sciencedirect.com/science/article/pii/S2590198223000611

Boessenkool (2023). *rdwd.* https://bookdown.org/brry/rdwd/

Deutscher Wetterdienst (2023). *CDC-OpenData area.* https://opendata.dwd.de/climate_environment/CDC/Readme_intro_CDC_ftp.pdf

Landesbetrieb Information und Technik Nordrhein-Westfalen (2023). *Verkehrsunfälle mit Personenschaden in Deutschland (Unfallatlas).* https://www.opengeodata.nrw.de/produkte/transport_verkehr/unfallatlas/

Landesbetrieb Information und Technik Nordrhein-Westfalen (2023b). *Data record description.* https://www.opengeodata.nrw.de/produkte/transport_verkehr/unfallatlas/DSB_Unfallatlas_EN.pdf

Statistisches Bundesamt (2023). *Alle politisch selbständigen Gemeinden mit ausgewählten Merkmalen am 30.09.2023 (3. Quartal 2023).* https://www.destatis.de/DE/Themen/Laender-Regionen/Regionales/Gemeindeverzeichnis/Administrativ/Archiv/GVAuszugQ/AuszugGV3QAktuell.html