Appendix B. Exploratory Data Analysis

```r
library(tidyverse) # data manipulation
library(here) # allocate file
library(dplyr) # data manipulation
library(ggplot2) # data visualization
library(lubridate)
library(patchwork) # merge visual plots to one
library(VIM) # tools for the visualization of missing or imputed values
library(naniar)
library(fastDummies) # automatically create dummy variables columns
library(zoo) # assign mean to NaN values
library(sqldf) # using SQL
library(viridis) # best. color. palette. evar.
library(reshape2)
library(ggrepel)
library(forcats)
library(scales)
library(treemapify) #plot treemap visualization
library(janitor)

#---------------------------------------------------
# Load the dataset
#---------------------------------------------------
df <- read_csv(here('data', 'car_accident.csv'))

## filter necessary columns for analysis
df <- df[-c(1:2,6,8:10,12,18:21,32,34:35,40)]
glimpse(df)
#-----------------------------------------------------
# Data Wrangling
#-----------------------------------------------------
# get column names
colnms <- colnames(df)

# Create a variable that only contain the columns with missing values
dfNA <- df[ , colSums(is.na(df))!=0]

# Check for NA values
missing_data <- summary(aggr(dfNA,prop=TRUE,combined=TRUE,
                             cex.axis=0.4, sortVars=TRUE))

#-------------------------------------------------------------
# Deal with missing data
#-------------------------------------------------------------
# categorical variables
df[c(6,17:18,21:22,35:36)] <- df[c(6,17:18,21:22,35:36)]%>%
  replace(is.na(.), "Unknown")

df[c(15,16,19,20)] <- df[c(15,16,19,20)] %>%
  replace(is.na(.), 0)

# numeric variables
```

```r
df[c(25,28:34)] <-
  lapply(df[c(25,28:34)], as.numeric) # convert columns to numeric

df[c(25,28:34)] <-
  na.aggregate(df[c(25,28:34)]) # replace NA values with mean

# filter driver seat and injury level
fatality_accidents <- df %>% filter(SEATING_POSITION=="D",
                                    Inj_Level_Desc=="Fatality")

#------------------------------------------------------------
# Data Visualization
#------------------------------------------------------------

#----------------------------------------------------------
# 1. The "When" - Time
#----------------------------------------------------------

####################################################################
## Total Road Fatalities by Year (2006-2020)
####################################################################
# Fatal accident proportion by year
accident_summary_year <- fatality_accidents %>%
  mutate(year = year(ACCIDENTDATE)) %>%
  group_by(year) %>%
  tally()

ggplot(accident_summary_year) +
  aes(x = year, y = n) +
  geom_line(size = 0.5, colour = "#B22222") +
  geom_point(color = "#B22222", size = 2) +
  geom_label(
    aes(label=n),
    nudge_x = 0.5,
    nudge_y = 6,
    check_overlap = TRUE,
    size = 3.5)+
  labs(x = "year",
       y = "total fatalities",
       title = "Total Road Fatalities by Year (2006-2020)") +
  scale_x_continuous(breaks = c(2006, 2008, 2010, 2012,
                                2014, 2016, 2018, 2020)) +
  scale_y_continuous(expand = c(0, 0), limits = c(100, 280),
                     breaks = c(0, 50, 100, 150, 200, 250, 300, 350)) +
  ggthemes::theme_tufte() +
  theme(plot.title = element_text(size = 15L, hjust = 0.5))


####################################################################
## Fatal accident proportion by Month (2015-2019)
####################################################################

year_15_20 <- fatality_accidents %>%
  filter(ACCIDENTDATE > as.Date("2014-12-31"))

accident_summary_month <-  year_15_20 %>%
  mutate(month = months(as.Date(year_15_20$ACCIDENTDATE))) %>%
  group_by(month) %>%
  tally()

accident_summary_month$month <- ordered( # order month chronically
  accident_summary_month$month, levels=c("January","February","March",
                                         "April","May","June","July",
                                         "August","September","October",
                                         "November","December"))
## bar plot
accident_summary_month %>%
  group_by(month)  %>%
  ggplot(aes(x = month, y = n)) +
  geom_col(fill = "#B22222") +
  ggtitle("Fatalities by Month (2015-2019)") +
  scale_y_continuous(expand = c(0, 0), limits = c(0, 120,140),
                     breaks = c(0,20,40,60,80,100,120,140)) +
  geom_label(aes(x = month, y = n, label = n)) +
  labs(x = "month", y = "fatalities") +
  coord_flip() +
  ggthemes::theme_tufte() +
  theme(plot.title = element_text(size = 15L, hjust = 0.5))

####################################################################
## Fatalities by Hour and Weekdays (2015-2019)
####################################################################

## add columns for accident hours
fatality_accidents$ACCIDENT_HOUR <- as.character(format(
  strptime(fatality_accidents$ACCIDENTTIME, "%H:%M"), "%H"))

# order Day Of Week chronically
fatality_accidents$Day_Week_Description <- ordered(
  fatality_accidents$Day_Week_Description, levels=c("Monday", "Tuesday", "Wednesday",
                                                    "Thursday", "Friday", "Saturday",
                                                    "Sunday"))
# create new column to specify weekend and weekday
fatality_accidents$ACCIDENTDATE <- as.Date(fatality_accidents$ACCIDENTDATE)
weekdays1 <- c('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday')

fatality_accidents$wDay <- factor(
  (weekdays(fatality_accidents$ACCIDENTDATE) %in% weekdays1),
  levels=c(FALSE, TRUE), labels=c('weekend', 'weekday'))

fatal_week_hour <- sqldf(
  "
  SELECT
      Day_Week_Description,
      ACCIDENT_HOUR,
      wDay,
      COUNT(*) as total
      FROM fatality_accidents
    GROUP BY
      Day_Week_Description,
      ACCIDENT_HOUR,
      wDay"
)

#heatmap
ggplot(fatal_week_hour) +
  aes(
    x = ACCIDENT_HOUR,
    y = Day_Week_Description,
    fill = `total`
  ) +
  geom_tile(size = 1.2) +
  scale_fill_distiller(palette = "Reds", direction = 1) +
  labs(
    x = "hours",
    y = "weekdays",
    title = "Weekday vs. Hourly Road Fatalities (2006-2020)",
    fill = "fatalities"
  ) +
  ggthemes::theme_tufte() +
  theme(plot.title = element_text(size = 15L, hjust = 0.5))

####################################################################
## Fatalities by Weekend Vs. Weekdays Distribution (2006-2020)
####################################################################
#barplot
ggplot(fatal_week_hour) +
  aes(x = ACCIDENT_HOUR, y = total, fill = wDay) +
  geom_boxplot(shape = "circle") +
  scale_fill_brewer(palette = "Reds", direction = -1) +
  labs(
    x = "hour",
    y = "fatalities",
    title = "Fatalities by Time (2006-2020)"
  ) +
  ggthemes::theme_tufte() +
  theme(
    legend.position = "none",
    plot.title = element_text(size = 15L,
                              hjust = 0.5),
    plot.subtitle = element_text(size = 13L,
                                 hjust = 0.5)
  ) +
  facet_wrap(vars(wDay))


####################################################################
## Fatalities by Day and Night (General) (2006-2020)
####################################################################
```

```
morning_hour <- c('00','01','02','03','04','05',
                  '06','07','08','09','10','11','12')
afternoon_hour <- c('13','14','15','16','17')
evening_hour <- c('18','19','20')
night_hour <- c('21','22','23')

# create new column specify Day vs Night time
fatality_accidents$day_night<-
  ifelse(fatality_accidents$ACCIDENT_HOUR %in% morning_hour, "Morning",
         ifelse(fatality_accidents$ACCIDENT_HOUR %in% afternoon_hour, "Afternoon",
                ifelse(fatality_accidents$ACCIDENT_HOUR %in% evening_hour, "Evening",
                       ifelse(fatality_accidents$ACCIDENT_HOUR %in% night_hour, "Night",NA))))


#create new variable
day_night <- sqldf(
  "
  SELECT
      day_night,
      COUNT(*) as value
      FROM fatality_accidents
    GROUP BY day_night
      "
)

# calculate percentage
day_night %>%
  arrange(desc(value)) %>%
  mutate(prop = percent(value / sum(value))) -> day_night

# pie chart
ggplot(day_night, aes(x = "", y = value, fill = fct_inorder(day_night))) +
  geom_bar(stat = "identity", width = 1) +
  geom_col(color = "black", width = 1) +
  coord_polar("y", start = 0) +
  geom_label_repel(aes(label = prop), size=5,
                   show.legend = F, nudge_x =1, nudge_y = 1) +
  labs(
    title = "Fatalities by Daytime (2006-2020)"
  ) +
  scale_fill_brewer(palette = "Reds") +
  theme_classic() +
  guides(fill = guide_legend(title = "Daytime")) +
  ggthemes::theme_tufte() +
  theme(plot.title = element_text(size = 15L, hjust = 0.5))

####################################################################
## Fatalities by Day and Night (Weekend Vs. Weekdays) (2006-2020)
####################################################################
#create new variable
day_night_wDay <- sqldf(
  "
  SELECT
```

```
      day_night,
      SUM(CASE WHEN wDay = 'weekend' THEN 1 ELSE 0 END) AS weekend,
      SUM(CASE WHEN wDay = 'weekday' THEN 1 ELSE 0 END) AS weekday
      FROM fatality_accidents
    GROUP BY day_night
      "
)

# Transform the data into the long format
day_night_wDay <- melt(day_night_wDay)

# double pie charts
ggplot(day_night_wDay, aes(x = "", y = value, fill = day_night)) +
  geom_bar(stat = "identity", width = 1, position = position_fill()) +
  coord_polar(theta = "y") +
  facet_wrap( ~ variable) +
  scale_fill_brewer(palette = "Reds") +
  theme_classic() +
  theme(axis.line = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        plot.title = element_text(hjust = 0.5, color = "#666666"))

#-----------------------------------------------------------
# 1. The "Where" - Location
#-----------------------------------------------------------

####################################################################
## Top 10 LGA with highest road fatalities (2006-2020)
####################################################################

fatality_accidents %>%
  group_by(LGA_NAME) %>%
  dplyr::summarise(Total = n()) %>%
  top_n(10, Total)  %>%
  ggplot(aes(area = Total, fill = Total, label = LGA_NAME)) +
  geom_treemap() +
  labs(
    title = "Top 10 LGA with highest road fatalities (2006-2020)"
  ) +
  geom_treemap_text(fontface = "italic", colour = "white", place = "topleft",
                    reflow = T,grow = TRUE) +
  theme(plot.title = element_text(size = 15L, hjust = 0.5))

####################################################################
## Fatalities by Road Geometry (2006-2020)
####################################################################
fatal_geo <- sqldf(
  "
  SELECT
      Road_Geometry_Desc,
      COUNT(*) AS value
      FROM fatality_accidents
      where Road_Geometry_Desc != 'Unknown'
    GROUP BY
      Road_Geometry_Desc
    ORDER BY COUNT(*) DESC
      "
)

# bar plot
ggplot(fatal_geo) +
  aes(x = Road_Geometry_Desc, y = value) +
  geom_bar(stat='identity', fill = "#FF8C00") +
  geom_label(
    aes(x = Road_Geometry_Desc, y = value, label=value))+
  labs(
    x = "road geometry",
    y = "fatalities",
    title = "Fatalities by Road Geometry (2006-2020)"
  ) +
  coord_flip() +
  ggthemes::theme_tufte() +
  theme(plot.title = element_text(size = 15L, hjust = 0.5))


#-----------------------------------------------------------
# 1. The "Why" - Other factors
#-----------------------------------------------------------
####################################################################
## Fatalities by Speed (2006-2020)
####################################################################
#create new variable
fatal_speed <- sqldf(
  "
  SELECT
      SPEED_ZONE,
      Age_Group,
      COUNT(*) AS value
      FROM fatality_accidents
      WHERE SPEED_ZONE NOT IN ('030','075','888','999')
    GROUP BY
      SPEED_ZONE,
      Age_Group
      "
)

# bar plot
ggplot(fatal_speed) +
  aes(x = SPEED_ZONE, y = value) +
  geom_bar(stat='identity',fill="#B22222") +
  labs(
    x = "speed",
    y = "fatalities",
    title = "Fatalities by Speed (2006-2020)"
  ) +
  ggthemes::theme_tufte() +
  theme(plot.title = element_text(size = 15L, hjust = 0.5))

####################################################################
# Pie chart Road User Type
####################################################################
#create new variable
arranged <- sqldf(
  "
  SELECT
      Road_User_Type_Desc,
      COUNT(*) as value
    FROM fatality_accidents
    WHERE Road_User_Type_Desc != 'Unknown'
    GROUP BY Road_User_Type_Desc
      "
)

# calculate percentage
arranged %>%
  arrange(desc(value)) %>%
  mutate(prop = percent(value / sum(value))) -> arranged

# pie chart
ggplot(arranged, aes(x = "", y = value,
                     fill = fct_inorder(Road_User_Type_Desc))) +
  geom_bar(stat = "identity", width = 1) +
  geom_col(color = "black", width = 1) +
  coord_polar("y", start = 0) +
  geom_label_repel(aes(label = prop),
                   size=5,
                   show.legend = F,
                   nudge_x =1,
                   nudge_y = 1) +
  labs(
    title = "Fatalities by Road Users (2006-2020)"
  ) +
  scale_fill_brewer(palette = "Reds") +
  theme_classic() +
  guides(fill = guide_legend(title = "Road Users")) +
  ggthemes::theme_tufte() +
  theme(plot.title = element_text(size = 15L, hjust = 0.5))

####################################################################
# Pie chart Accident Type
####################################################################
#create new variable
accident_type <- sqldf(
  "
  SELECT
      Accident_Type_Desc,
      COUNT(*) as value
    FROM fatality_accidents
    WHERE Accident_Type_Desc != 'Unknown'
    GROUP BY Accident_Type_Desc
      "
)

# calculate percentage
accident_type %>%
  arrange(desc(value)) %>%
  mutate(prop = percent(value / sum(value))) -> accident_type

# pie chart
ggplot(accident_type, aes(x = "", y = value,
                          fill = fct_inorder(Accident_Type_Desc))) +
  geom_bar(stat = "identity", width = 1) +
  geom_col(color = "black", width = 1) +
  coord_polar("y", start = 0) +
  labs(
    title = "Fatalities by Accident Types (2006-2020)"
  ) +
  scale_fill_brewer(palette = "Reds") +
  theme_classic() +
  guides(fill = guide_legend(title = "Accident Types")) +
  ggthemes::theme_tufte() +
  theme(plot.title = element_text(size = 15L, hjust = 0.5))

####################################################################
# Number of Fatal accidents by Road Surface Condition
####################################################################
car_accidents <- read_csv(here("data/car_accident.csv"))
car_accidents <- clean_names(car_accidents)

df2 <- car_accidents %>%
  filter(seating_position == 'D') %>%
  group_by(surface_cond_desc) %>%
  summarise(total_fatalities = n_distinct(accident_no),
            number_of_sameples = n(),
            fatalities_ratio = total_fatalities / number_of_sameples)

df2 %>%
  filter(surface_cond_desc != 'Unknown') %>%
  ggplot(aes(x = surface_cond_desc, y = fatalities_ratio)) +
  geom_bar(stat = 'identity', fill = "#B22222") +
  xlab("Surface Condition Description") +
  ylab("Total Accidents with Fatalities / Total Accidents") +
  labs(title = "Number of Fatal Accidents by Road Surface Conditions") +
  ggthemes::theme_tufte() +
  theme(
    plot.title = element_text(size = 16L,
                              face = "bold",
                              hjust = 0.5)
  )

####################################################################
```

```r
# Number of Fatal accidents by Weather Conditions
####################################################################
df3 <- car_accidents %>%
  filter(seating_position == 'D') %>%
  group_by(conditions) %>%
  summarise(total_fatalities = n_distinct(accident_no),
            number_of_sameples = n(),
            fatalities_ratio = total_fatalities / number_of_sameples)


df3 %>%
  filter(conditions != 'Unknown') %>%
  ggplot(aes(x = conditions, y = fatalities_ratio)) +
  geom_bar(stat = 'identity', fill = "#B22222") +
  xlab("Weather Condition Description") +
  ylab("Total Accidents with Fatalities / Total Accidents") +
  labs(title = "Number of Fatal Accidents by Weather Conditions") +
  coord_flip() +
  ggthemes::theme_tufte() +
  theme(
    plot.title = element_text(size = 16L,
                              face = "bold",
                              hjust = 0.5)
  )

####################################################################
# Number of Fatal accidents by Light Conditions
####################################################################
df4 <- car_accidents %>%
  filter(seating_position == 'D') %>%
  group_by(light_condition_desc) %>%
  summarise(total_fatalities = n_distinct(accident_no),
            number_of_sameples = n(),
            fatalities_ratio = total_fatalities / number_of_sameples)


df4 %>%
  filter(light_condition_desc != 'Unknown') %>%
  ggplot(aes(x = light_condition_desc, y = fatalities_ratio)) +
  geom_bar(stat = 'identity', fill = "#B22222") +
  xlab("Light Conditions Description") +
  ylab("Total Accidents with Fatalities / Total Accidents") +
  labs(title = "Number of Fatal Accidents by Light Conditions") +
  coord_flip() +
  ggthemes::theme_tufte() +
  theme(
    plot.title = element_text(size = 16L,
                              face = "bold",
                              hjust = 0.5)
  )
```

Appendix C. Data Pre-processing

```r
library(tidyverse)
library(here)
library(caret)
library(dplyr)
library(mltools)
library(VIM)
library(summarytools)
library(moments)
library(outliers)
library(DataExplorer)

df <- read_csv(here('data', 'car_accident.csv'))

################################################################################
# Filter and mutate data for basic analysis ----
################################################################################
# Convert fatality to numerical
df <- df %>%
  mutate(FATAL_ACCIDENT =  case_when(
    FATAL_ACCIDENT == 'Y' ~ 1,
    FATAL_ACCIDENT == 'N' ~ 0)
  )

# convert speed zone to numeric
df <- transform(df, SPEED_ZONE = as.numeric(SPEED_ZONE))
# Filter df - filter to driver based data and remove outliers
# drop values where speed zone has incorrect data e.g. > 200km/hr
df <- df[df$SPEED_ZONE < 200, ]
df <- df[df$NO_OF_CYLINDERS < 25, ]

# Convert no_of_cylinders to factor
df$NO_OF_CYLINDERS <- as.factor(df$NO_OF_CYLINDERS)

target <- c('Drivers', 'Motorcyclists')
df <- df %>%
  filter(Road_User_Type_Desc %in% target)

dfSummary(df)


################################################################################
# Define numerical and categorical columns ----
################################################################################

numerical_cols <- c('NO_OF_VEHICLES', 'NO_PERSONS','SPEED_ZONE',
                    'VEHICLE_YEAR_MANUF', 'TOTAL_NO_OCCUPANTS',
                    'LIGHT_CONDITION', 'CloudCover', 'WindSpeed',
                    'Temperature', 'DewPoint', 'RelativeHumidity',
                    'Precipitation')
```

```
cat_cols <- c('SEX', 'SEATING_POSITION', 'Accident_Type_Desc',
              'Road_Surface_Type_Desc', 'Surface_Cond_Desc',
              'Atmosph_Cond_Desc', 'Light_Condition_Desc','Conditions',
              'Age_Group', 'Day_Week_Description', 'NO_OF_CYLINDERS')

indexing_cols <- c('ACCIDENT_NO', 'FATAL_ACCIDENT', 'ACCIDENTDATE',
                   'ACCIDENTTIME' , 'Road_User_Type_Desc')

log_cols <- c('Precipitation', 'CloudCover', 'RelativeHumidity', 'NO_PERSONS',
              'TOTAL_NO_OCCUPANTS', 'NO_OF_VEHICLES', 'SPEED_ZONE',
              'LIGHT_CONDITION')

all_features_cols <- c(cat_cols)
```

Appendix C-1. Random Imputation - Handle Missing Values Technique

```
################################################################################
# Handle missing values-----
################################################################################
################################################################################
# Replace missing values at same frequency they appear in column
################################################################################
# which(myV>7)[1]
idx_greater_than <- function(value, list){
#Find the first index of vector 'list' that has a corresponding value greater
#than 'value'

  for(i in 1:length(list)){
    if(list[i] > value){
      return(i)
    }
  }
}

# MAIN LOGIC STARTS HERE ----
replace_nan_df <- df[all_features_cols]

for(name in names(replace_nan_df)){
  column_vector <- pull(replace_nan_df, name)

  # Get index of nans
  nan_idxs <- which(is.na(column_vector))

  # If no nans, don't worry
  if(length(nan_idxs)==0){
    next
  }

  srs_notnull <- column_vector[!is.na(column_vector)]

  # Get unique labels and counts for the non-nan features
  unique_frequency_df <- as.data.frame(table(srs_notnull))
  labels <- as.character(unique_frequency_df$srs_notnull)
  counts <- unique_frequency_df$Freq
  cum_counts <- cumsum(counts)

  # Generate random numbers of size len(nan_idxs)
  set.seed(1)
  rand_vals <- floor(runif(length(nan_idxs), min=0, max=length(srs_notnull)))

  new_vals <- c()
  for(x in rand_vals){
    #Find out the largest number in cum_counts that each rand_val is less than
    larger_value_index <- idx_greater_than(x, cum_counts)
    # Get values corresponding to above index
    new_vals <- append(new_vals, labels[larger_value_index])
  }
  # Update the df with the new vals
  df[nan_idxs, name] = new_vals
}

sum(is.na(df))
```

Appendix C-2. Log Transformation

```
################################################################################
# NEED TO PERFORM LOG FUNCTION ON SKEWED NUMERICAL DATA HERE
################################################################################
log_df <- df[log_cols]
glimpse(df)
for(name in names(log_df)){
  column_vector <- pull(log_df, name)
  column_vector <- as.numeric(column_vector)
  skew <- skewness(column_vector)
  if (skew < - 0.5){
    constant <- max(column_vector) + 1
    new_vals <- constant - column_vector
    new_vals <- log(new_vals)
    df[name] = as.numeric(new_vals)
    hist(new_vals)
  } else if (skew > 0.5){
    constant <- 1
    new_vals <- log((column_vector + constant))
    new_vals <- as.numeric(new_vals)
    df[name] = new_vals
  }
}
```

Appendix C-3. Transform Categorical Data Based on EDA

```
################################################################################
# Transform Categorical Data Based on EDA ----
```

```
################################################################################
#Use dummyVars function to create binary variables.
category_df <- df[cat_cols]
variables <- dummyVars("~.", data = category_df, sep = "_")


category_df <- data.frame(predict(variables, newdata = category_df))

category_df <- category_df %>%
     mutate_if(is.double, as.factor)

base_df <- df[indexing_cols]
numerical_df <- df[numerical_cols]
```

Appendix C-4. Outlier

```
################################################################################
# Outlier
################################################################################
# calculate z-score
mean(df$AGE)
#calculate z score
z.scores <- df$AGE %>% na.omit %>% scores(type = "z")
z.scores %>% summary()

# Finds the total number of outliers according to the z-score
length (which( abs(z.scores) >3 ))
```

Appendix C-5. Outlier

```
################################################################################
# Outlier
################################################################################
# calculate z-score
mean(df$AGE)
#calculate z score
z.scores <- df$AGE %>% na.omit %>% scores(type = "z")
z.scores %>% summary()

# Finds the total number of outliers according to the z-score
length (which( abs(z.scores) >3 ))
```

Appendix C-6. Standardization

```
################################################################################
# Standardize/ scale numeric values
################################################################################
```

```r
final_df <- cbind(base_df, category_df)
final_df <- cbind(final_df, numerical_df)
```

Appendix C-7. Save Elements For Balancing Data

```r
################################################################################
# Create features to keep in df based on EDA and domain knowledge
################################################################################
final_cols <- c("ACCIDENT_NO","FATAL_ACCIDENT",
                "ACCIDENTDATE","ACCIDENTTIME",
                "Road_User_Type_Desc","SEXF",
                "SEXM",
                "Accident_Type_DescCollision.with.a.fixed.object",
                "Accident_Type_DescStruck.animal",
                "Accident_Type_DescStruck.Pedestrian",
                "Accident_Type_DescVehicle.overturned..no.collision.",
                "Road_Surface_Type_DescUnpaved",
                "Surface_Cond_DescDry",
                "Surface_Cond_DescIcy",
                "Surface_Cond_DescMuddy",
                "Surface_Cond_DescSnowy",
                "Surface_Cond_DescWet",
                "Atmosph_Cond_DescClear",
                "Atmosph_Cond_DescFog",
                "Atmosph_Cond_DescRaining",
                "Atmosph_Cond_DescSmoke",
                "Atmosph_Cond_DescStrong.winds",
                "Light_Condition_DescDark.No.street.lights",
                "Light_Condition_DescDark.Street.lights.off",
                "ConditionsOvercast",
                "ConditionsRain",
                "ConditionsRain..Overcast",
                "Age_Group16.17",
                "Age_Group17.21",
                "Age_Group70.",
                "Day_Week_DescriptionSaturday",
                "Day_Week_DescriptionSunday",
                "NO_OF_CYLINDERS_4",
                "NO_OF_CYLINDERS_6",
                "NO_OF_CYLINDERS_8",
                "NO_OF_CYLINDERS_12",
                "NO_OF_VEHICLES",
                "NO_PERSONS",
                "SPEED_ZONE",
                "VEHICLE_YEAR_MANUF",
                "TOTAL_NO_OCCUPANTS",
                "LIGHT_CONDITION",
                "CloudCover",
                "WindSpeed",
                "Temperature",
                "DewPoint",
                "RelativeHumidity",
                "Precipitation")


final_df <- final_df[final_cols]
dfSummary(final_df)
write_csv(final_df, here("data", "Car_Accident_Data_No_Na.csv"))
```

Appendix C-8. Balancing Data

```
library(tidyverse)
library(here)
library(caret)
library(dplyr)
library(mltools)
library(VIM)
library(summarytools)
library(DMwR)

################################################################################
# Read in data
df <- read.csv(here("data", "Car_Accident_Data_No_Na.csv"))

################################################################################
# Create train, test and cross validation df function
################################################################################
create_train_test_cross <- function(df){
  train_df <- df[df$ACCIDENTDATE <  "2017-01-01",]
  test_df <- df[(df$ACCIDENTDATE > "2017-01-01" & df$ACCIDENTDATE < "2020-01-01"),]
  cross_valid_df <- df[(df$ACCIDENTDATE > "2020-01-01" ),]
  return (list(train_df, test_df, cross_valid_df))
}
################################################################################
################################################################################
# Create different balanced data sets
################################################################################
################################################################################
# Create train and test for no transformation techniques
################################################################################
no_change_list <- create_train_test_cross(df)
no_change_train <- no_change_list[[1]]
no_change_test <- no_change_list[[2]]
no_change_cross <- no_change_list[[3]]

drops <- c("Road_User_Type_Desc","ACCIDENTTIME", "ACCIDENT_NO", "ACCIDENTDATE")
no_change_train <- no_change_train[ , !(names(no_change_train) %in% drops)]
no_change_test <- no_change_test[ , !(names(no_change_test) %in% drops)]
no_change_cross <- no_change_cross[ , !(names(no_change_cross) %in% drops)]

write_csv(no_change_train, here("data", "no_change_train.csv"))
write_csv(no_change_test, here("data", "test_set_generic.csv"))
write_csv(no_change_cross, here("data", "cross_set_generic.csv"))
```

Appendix D. Logistic Regression Model and Evaluation

```
library(MASS)
library(tidyverse)
library(caret)
library(here)
library(pROC)
```

```
knitr::opts_chunk$set(warning = TRUE, message = TRUE)
no_change_train <-read_csv(here("data", "no_change_train.csv"))
test_generic <- read_csv(here("data", "test_set_generic.csv"))
smote_train <- read_csv(here("data", "smote_train.csv"))
under_sample_train <- read_csv(here("data", "under_sample_train.csv"))
cross_validation <- read_csv(here("data", "no_change_cross.csv"))
```

Appendix D-1. Baseline Model

```
base_final_model <- glm(formula = FATAL_ACCIDENT ~ SPEED_ZONE +
                    Accident_Type_DescStruck.Pedestrian +
                    Accident_Type_DescCollision.with.a.fixed.object + SEXM +
                    NO_PERSONS + Light_Condition_DescDark.No.street.lights +
                    Atmosph_Cond_DescClear + Age_Group70. + CloudCover +
                    NO_OF_CYLINDERS_4 +  LIGHT_CONDITION +
                    Accident_Type_DescStruck.animal + ConditionsRain..Overcast +
                    NO_OF_CYLINDERS_12 + Surface_Cond_DescIcy + SEXF +
                    Age_Group17.21 + Atmosph_Cond_DescRaining +
                    Surface_Cond_DescDry + Surface_Cond_DescWet +
                    VEHICLE_YEAR_MANUF +
                    Accident_Type_DescVehicle.overturned..no.collision. +
                    ConditionsRain + Atmosph_Cond_DescFog +
                    Atmosph_Cond_DescStrong.winds + NO_OF_VEHICLES +
                    NO_OF_CYLINDERS_6 + Surface_Cond_DescSnowy +
                    NO_OF_CYLINDERS_8 + Age_Group16.17,
                    family = binomial, data = no_change_train)
```

Appendix D-2. Testing Model

```
model_test <- test_generic[,-1]

pred <- predict(base_final_model, model_test, type = "response")
pred <- as.data.frame(pred)
lift_threshold <- 0.5
pred <- mutate(pred, pred = ifelse(pred >= lift_threshold, 1,
                                   ifelse(pred < lift_threshold, 0, NA)))

pred_y <- as.numeric(pred > 0)
true_y <- as.numeric(test_generic$FATAL_ACCIDENT)
pred_y_factor <- as.factor(pred_y)
```

```
true_y_factor <- as.factor(true_y)
confusion_matrix_1 <- confusionMatrix(pred_y_factor, true_y_factor, positive = "1")
```

Appendix D-3. Confusion Matrix

```
print(confusion_matrix_1)
print(" ")
```

```
print(confusion_matrix_1$byClass)
```

```
#Create ROC curve
idx <- order(-pred)
recall <- cumsum(true_y[idx] == 1) / sum(true_y == 1)
specificity <- (sum(true_y == 0) - cumsum(true_y[idx] == 0)) / sum(true_y == 0)
roc_df <- data.frame(recall = recall, specificity = specificity)
roc <- ggplot(roc_df, aes(x=specificity, y=recall)) +
  geom_line(color='blue') +
  scale_x_reverse(expand=c(0, 0)) +
  scale_y_continuous(expand=c(0, 0)) +
  geom_line(data=data.frame(x=(0:100) / 100), aes(x=x, y=1-x),
            linetype='dotted', color='red')

print(roc)

auc <- sum(roc_df$recall[-1] * diff(1 - roc_df$specificity))
print(paste0("AUC: ", auc))
```

Appendix D-4. Under Sample Final Model

```
under_sample_final_model <- glm(formula = FATAL_ACCIDENT ~ SPEED_ZONE +
                                  Accident_Type_DescStruck.Pedestrian +
                                  Accident_Type_DescCollision.with.a.fixed.object +
                                  SEXM +  NO_PERSONS + Atmosph_Cond_DescClear +
                                  LIGHT_CONDITION + Age_Group70. + CloudCover +
                                  Atmosph_Cond_DescRaining +
                                  Atmosph_Cond_DescStrong.winds +
                                  Atmosph_Cond_DescFog + NO_OF_CYLINDERS_4 +
                                  Surface_Cond_DescDry + Surface_Cond_DescWet +
                                  Accident_Type_DescStruck.animal +
                                  VEHICLE_YEAR_MANUF + Age_Group17.21 +
                                  ConditionsRain..Overcast +
                                  Surface_Cond_DescSnowy + NO_OF_CYLINDERS_6 +
                                  Road_Surface_Type_DescUnpaved +
                                  Light_Condition_DescDark.No.street.lights,
                                  family = binomial,
                                  data = under_sample_train)
```

Appendix D-5. ROC Curves

```r
model_test <- test_generic[,-1]
pred <- predict(under_sample_final_model, model_test, type = "response")
pred <- as.data.frame(pred)
pred <- mutate(pred, pred = ifelse(pred >= lift_threshold, 1,
                                   ifelse(pred < lift_threshold, 0, NA)))

print("Compiling confusion matrix")
pred_y <- as.numeric(pred > 0)
true_y <- as.numeric(test_generic$FATAL_ACCIDENT)
pred_y_factor <- as.factor(pred_y)
true_y_factor <- as.factor(true_y)
confusion_matrix_2 <- confusionMatrix(pred_y_factor, true_y_factor, positive = "1")
print(confusion_matrix_2)
print(confusion_matrix_2$byClass)

#Create ROC curve
idx <- order(-pred)
recall <- cumsum(true_y[idx] == 1) / sum(true_y == 1)
specificity <- (sum(true_y == 0) - cumsum(true_y[idx] == 0)) / sum(true_y == 0)
roc_df <- data.frame(recall = recall, specificity = specificity)
roc <- ggplot(roc_df, aes(x=specificity, y=recall)) +
  geom_line(color='blue') +
  scale_x_reverse(expand=c(0, 0)) +
  scale_y_continuous(expand=c(0, 0)) +
  geom_line(data=data.frame(x=(0:100) / 100), aes(x=x, y=1-x),
            linetype='dotted', color='red')

print(roc)


auc <- sum(roc_df$recall[-1] * diff(1 - roc_df$specificity))
print(paste0("AUC: ", auc))


print(under_sample_final_model)
```

```r
model_test <- cross_validation[,-1]
pred <- predict(under_sample_final_model, model_test, type = "response")
pred <- as.data.frame(pred)
pred <- mutate(pred, pred = ifelse(pred >= lift_threshold, 1,
                                   ifelse(pred < lift_threshold, 0, NA)))

print("Compiling confusion matrix")
pred_y <- as.numeric(pred > 0)
true_y <- as.numeric(cross_validation$FATAL_ACCIDENT)
pred_y_factor <- as.factor(pred_y)
true_y_factor <- as.factor(true_y)
confusion_matrix_3 <- confusionMatrix(pred_y_factor, true_y_factor, positive = "1")
print(confusion_matrix_3)
print(confusion_matrix_3$byClass)
```

```r
#Create ROC curve
idx <- order(-pred)
recall <- cumsum(true_y[idx] == 1) / sum(true_y == 1)
specificity <- (sum(true_y == 0) - cumsum(true_y[idx] == 0)) / sum(true_y == 0)
roc_df <- data.frame(recall = recall, specificity = specificity)
roc <- ggplot(roc_df, aes(x=specificity, y=recall)) +
  geom_line(color='blue') +
  scale_x_reverse(expand=c(0, 0)) +
  scale_y_continuous(expand=c(0, 0)) +
  geom_line(data=data.frame(x=(0:100) / 100), aes(x=x, y=1-x),
            linetype='dotted', color='red')

print(roc)


auc <- sum(roc_df$recall[-1] * diff(1 - roc_df$specificity))
print(paste0("AUC: ", auc))


print(under_sample_final_model)
```