Assessment 2: Data exploration and wrangling

Sujal Manandhar (23189654/1)

Part 1: Wrangling Data

In this part, we will focus on data preparation and cleaning involves several key tasks to ensure our dataset is ready for analysis.

Load Necessary Libraries

Task 1: Importing the required libraries.

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(caret)
```

```
## Loading required package: lattice
```

Load the Dataset

Task 2: Reading the dataset 'accidents.csv' and storing it in the variable 'data'.

```
data <- read.csv('accidents.csv')
```

Display Initial Rows of the Dataset

Task 3: Viewing the top 5 rows to get an overview of the data.

```
head(data, 5)
```

```
##   Number.of.Vehicles Accident.Date Time..24hr. X1st.Road.Class Road.Surface
## 1                  2    01/01/2017        2120               U     Wet/Damp
## 2                  2    04/01/2017        1500               U          Dry
## 3                  2    05/01/2017         732             A58     Wet/Damp
## 4                  2    05/01/2017         930            A646     Wet/Damp
## 5                  2    14/01/2017         909               U    Frost/Ice
##   Lighting.Conditions Daylight.Dark Weather.Conditions Local.Authority
## 1                   4          Dark                  2      Calderdale
## 2                   1      Daylight                  1      Calderdale
## 3                   4          Dark                  1      Calderdale
## 4                   1      Daylight                  1      Calderdale
## 5                   1      Daylight                  1      Calderdale
##   Type.of.Vehicle Casualty.Class Casualty.Severity Sex.of.Casualty
## 1               9              2                 2               2
## 2               9              3                 2               2
## 3               9              1                 3               1
## 4               4              1                 1               1
## 5               9              1                 3               1
##   Age.of.Casualty
## 1              16
## 2              67
## 3              56
## 4              20
## 5              46
```

List Column Names

Task 4: Retrieving and displaying the names of all columns in the dataset.

```
column_names <- names(data)
column_names
```

```
##  [1] "Number.of.Vehicles"  "Accident.Date"       "Time..24hr."
##  [4] "X1st.Road.Class"     "Road.Surface"        "Lighting.Conditions"
##  [7] "Daylight.Dark"       "Weather.Conditions"  "Local.Authority"
## [10] "Type.of.Vehicle"     "Casualty.Class"      "Casualty.Severity"
## [13] "Sex.of.Casualty"     "Age.of.Casualty"
```

Rename Columns for Clarity

Task 5: Renaming columns to improve readability and consistency.

```
colnames(data) <- c("Number_of_Vehicles", "Accident_Date", "Time_24hr", "First_Road_Class",
                    "Road_Surface", "Lighting_Conditions", "Daylight_Dark", "Weather_Conditions",
                    "Local_Authority", "Type_of_Vehicle", "Casualty_Class", "Casualty_Severity",
                    "Sex_of_Casualty", "Age_of_Casualty")

names(data)
```

```
##  [1] "Number_of_Vehicles"  "Accident_Date"       "Time_24hr"
##  [4] "First_Road_Class"    "Road_Surface"        "Lighting_Conditions"
##  [7] "Daylight_Dark"       "Weather_Conditions"  "Local_Authority"
## [10] "Type_of_Vehicle"     "Casualty_Class"      "Casualty_Severity"
## [13] "Sex_of_Casualty"     "Age_of_Casualty"
```

Data Structure Overview

Task 6: Displaying the structure of the dataset to understand the data types and initial values.

```
str(data)
```

```
## 'data.frame':    2069 obs. of  14 variables:
##  $ Number_of_Vehicles : int  2 2 2 2 2 1 1 3 1 1 ...
##  $ Accident_Date      : chr  "01/01/2017" "04/01/2017" "05/01/2017" "05/01/2017" ...
##  $ Time_24hr          : int  2120 1500 732 930 909 1659 1059 1849 1408 1325 ...
##  $ First_Road_Class   : chr  "U" "U" "A58" "A646" ...
##  $ Road_Surface       : chr  "Wet/Damp" "Dry" "Wet/Damp" "Wet/Damp" ...
##  $ Lighting_Conditions: int  4 1 4 1 1 4 1 4 1 1 ...
##  $ Daylight_Dark      : chr  "Dark" "Daylight" "Dark" "Daylight" ...
##  $ Weather_Conditions : int  2 1 1 1 1 1 1 1 1 1 ...
##  $ Local_Authority    : chr  "Calderdale" "Calderdale" "Calderdale" "Calderdale" ...
##  $ Type_of_Vehicle    : int  9 9 9 4 9 9 9 9 9 9 ...
##  $ Casualty_Class     : int  2 3 1 1 1 3 3 1 3 1 ...
##  $ Casualty_Severity  : int  2 2 3 1 3 3 3 3 3 3 ...
##  $ Sex_of_Casualty    : int  2 2 1 1 1 1 2 2 2 2 ...
##  $ Age_of_Casualty    : int  16 67 56 20 46 NA 25 50 64 22 ...
```

Statistical Summary

Task 7: Generating a concise summary of statistics for each variable in the dataset.

```
summary(data)
```

```
##  Number_of_Vehicles Accident_Date        Time_24hr    First_Road_Class
##  Min.   :1.000      Length:2069        Min.   :   0   Length:2069
##  1st Qu.:1.000      Class :character   1st Qu.:1045   Class :character
##  Median :2.000      Mode  :character   Median :1500   Mode  :character
##  Mean   :1.906                         Mean   :1405
##  3rd Qu.:2.000                         3rd Qu.:1755
##  Max.   :7.000                         Max.   :2350
##
##  Road_Surface       Lighting_Conditions Daylight_Dark      Weather_Conditions
##  Length:2069        Min.   :1.000       Length:2069        Min.   :1.000
##  Class :character   1st Qu.:1.000       Class :character   1st Qu.:1.000
##  Mode  :character   Median :1.000       Mode  :character   Median :1.000
##                     Mean   :2.015                          Mean   :1.464
##                     3rd Qu.:4.000                          3rd Qu.:1.000
##                     Max.   :7.000                          Max.   :9.000
##
##  Local_Authority    Type_of_Vehicle  Casualty_Class  Casualty_Severity
##  Length:2069        Min.   : 1.000   Min.   :1.000   Min.   :1.000
##  Class :character   1st Qu.: 9.000   1st Qu.:1.000   1st Qu.:3.000
##  Mode  :character   Median : 9.000   Median :1.000   Median :3.000
##                     Mean   : 8.917   Mean   :1.591   Mean   :2.831
##                     3rd Qu.: 9.000   3rd Qu.:2.000   3rd Qu.:3.000
##                     Max.   :97.000   Max.   :3.000   Max.   :3.000
##
##  Sex_of_Casualty Age_of_Casualty
##  Min.   :1.000   Min.   :  1.00
##  1st Qu.:1.000   1st Qu.: 21.00
##  Median :1.000   Median : 33.00
##  Mean   :1.395   Mean   : 36.21
##  3rd Qu.:2.000   3rd Qu.: 49.00
##  Max.   :2.000   Max.   :115.00
##                  NA's   :19
```

Row Count

Task 8: Counting and displaying the total number of rows in the dataset.

```
num_rows <- nrow(data)
num_rows
```

```
## [1] 2069
```

Column Count

Task 9: Counting and displaying the total number of columns in the dataset.

```
num_cols <- ncol(data)
num_cols
```

```
## [1] 14
```

Date Column Format

Task 10: Checking and converting the date format for the 'Accident_Date' column.

```
str(data$Accident_Date)
```

```
##  chr [1:2069] "01/01/2017" "04/01/2017" "05/01/2017" "05/01/2017" ...
```

```
# Convert the "Accident_Date" column to a date format
data$Accident_Date <- as.Date(data$Accident_Date, format = "%d/%m/%Y")

str(data$Accident_Date)  # After changes
```

```
##  Date[1:2069], format: "2017-01-01" "2017-01-04" "2017-01-05" "2017-01-05" "2017-01-14" ...
```

Time Column Overview

Task 11: Checking the structure and unique values of the 'Time_24hr' column.

```
str(data$Time_24hr)
```

```
##  int [1:2069] 2120 1500 732 930 909 1659 1059 1849 1408 1325 ...
```

```
unique(data$Time_24hr)
```

```
##    [1] 2120 1500  732  930  909 1659 1059 1849 1408 1325 1126 1830 1258 1133 1546
##   [16] 1335 1405 2233 1835 1430 2230 1444 1435 1840 1400 1045 2200 1806  612 1310
##   [31] 1730 1227  921 1423 1900  833  605 1338 1450 2243 1225 1854 1440 2330    5
##   [46] 1212  700 1939 1200 2227  847  823  940  743 1415 1503 1605 1315 1508 1550
##   [61] 1231 1552  800  900 1855 1353  805  444 1820  840 1913   22 1317    0   26
##  [76] 1320 2142 1130 1356 1610 1535 1614 2125 1232 1700 1755 1705  734 1720  943
##  [91] 1600 1446 1454 1949 1540 1740 1515 1701  932 1930  735 1230 1933  640 1809
## [106]  213  325 1345 1805 1453 1210 1355 1628 1341 1630  713  752 1826 1750 1656
## [121] 2023 1032 1157 2056 1432  721 1800 1215 1640 1330 1636 1732 1520 2236 2300
## [136] 1151 1207 1915 1950 2140 1642 1926 1931  907 1510 2241 1357  849  750 1114
## [151]  843  859 1758 2328   40  822 1020 1530 2206 1814 1110 1655 2043 2050 2146
## [166] 2100 1518  609  922 1433 1505 1657  929 1706 1721 1615 1058 1455 2128 2136
## [181] 1957 1240  645 2209  727  828 2130 1522 1743 1608  538 1838 1935 1214 1205
## [196] 1555 1025 1627 1625 1617  725 1943 2129 1300 1843 1715  850 1250  753  620
## [211] 1220 1418 2122  702 1745  143  611   12 1011 1607 1007   55 1531 1313 1202
## [226] 1851  744 2331 2259 1709 1713 1150 1651 1858 1644 1420 1030 1407 1308 1426
## [241] 1635 1055 1120  829  815 1521  820  521 1302 1410 1815  818 1217 1859 2053
## [256] 1132 2020 1331   52 1204  950  846 1856  831 1536 2054 2101 2040  911  915
## [271] 1533 1708 1728  830 1624 1618 1934 1402 1910 2110 1028 2004 1129 1710 1337
## [286] 1340  715  542  905 1620 1727 1725  958 1923  904 1545  651  657 1048  759
## [301]  600 1107 1902 1827 1731 1350  845  550 2148 1443 1604 1236 2104 1649 1256
## [316] 2221 1136  910 1525 2244  811  130  610 1736  718 1925 2239  756 2207  705
## [331] 2000  951 1438 1905  855 1735 2314 1209 1645 1810 2149  230  210 2108 2045
## [346] 1428 2022 1155 1632 2345 1016 2018 1603 1342 2035 1248 2132 1448   20 1734
## [361]  115 1245 2344 1211 1054 2315 1524  330 1134 2135 1548 1812  945 1328  305
## [376] 1609 2145 1316 1825  755 1839 2220 1403  131 1234  639 1015 1850  522   57
## [391]  120   45 1135 1932 1904 2250 1528 2213 1439 1131 1119 2007 1144 1108 1514
## [406] 1613 1125 1529  810 2019 2255 1945  858  458 1121 1153 1821 1216  739 1012
## [421] 2150  726 1000  540 1717  720 1123 1726  539 1629 1128 1312 1650 1511  836
## [436] 1115 1829 2058 2115 1318 2039 2238  405 1643 2304 2346  848  826 2335 1118
## [451] 1557 1112 1637  955 1145 2257 1034 1050 1005 1141 1109 2030 1239 1654  740
## [466] 1526 1741 2153 1154 1251  857 1140 1754 1343 1658  835 1352 1955   10 1425
## [481] 1813 2215 1619 1431  255  719 1527  745  920    9  936 1749  122   46 2102
## [496] 2325 1733 1233 1456 1519  200 1421 1908  150 1559 1542 1903  630 1252 2305
## [511] 2010 2245 1228 2055 1541 1958 1358 1517 1327 1803  809 1226  554 1738 1224
## [526] 1616 1907 2005  145 1914  814  209 1243  934  730 1837   50  655 1332  754
## [541] 1924 2015 2240  825 1023 1219  741 1409 1422 1716  415 1906 1558 1344 1811
## [556]  946  647 2117 2320 1911 1014 1539  731  902 1322  854 1040 1532  914 1026
## [571] 1029 1822 1857  511  625   34  638  315 1429  249 1208  348  926  816 1100
## [586]  435  158 2201 2105 2047  323 1816 1305  455  908  913  100 1255 1845 2003
## [601] 2235  824  901  430  653 1148 1027 1413 2205  712 1451  931  110 1946 1623
## [616] 2210  838 1437 2316  738  352 1942 1414 1823 1951 2208 2158 1105 1229 2052
## [631] 1537 2350  308 1940  948  927  959 2012  629 1346  707 1612  243 1804  454
## [646] 2002 1031 1309 1445 1339 1404 1024 1927 2311 1319 1653  650 1158 2116  515
## [661] 1347 2225
```

Time Format Function

Task 12: Creating a function to format time from numeric to HH string format.

```
format_time <- function(time_col) {

  # Convert to character and ensure it has 4 digits
  time_str <- sprintf("%04.0f", time_col)
  # Extract hours and minutes
  hours <- substr(time_str, 1, 2)
  minutes <- substr(time_str, 3, 4)

  # Format time into HH:MM string
  formatted_time <- paste(hours, minutes, sep = ":")

  return(formatted_time)
}
```

Apply Time Format Function

Task 13: Applying the created function to format the time and update the 'Time_24hr' column.

```
data <- data %>%
  mutate(Time_24hr_ = format_time(Time_24hr)) %>%
  select(-Time_24hr)  # Drop the original 'Time_24hr' column

# Reorder columns to place 'Time_24hr_' after 'Accident_Date'
data <- data %>%
  select(Number_of_Vehicles, Accident_Date, Time_24hr_, everything())

str(data$Time_24hr_)
```

```
##  chr [1:2069] "21:20" "15:00" "07:32" "09:30" "09:09" "16:59" "10:59" ...
```

```
unique(data$Time_24hr_)
```

```
##   [1] "21:20" "15:00" "07:32" "09:30" "09:09" "16:59" "10:59" "18:49" "14:08"
##  [10] "13:25" "11:26" "18:30" "12:58" "11:33" "15:46" "13:35" "14:05" "22:33"
##  [19] "18:35" "14:30" "22:30" "14:44" "14:35" "18:40" "14:00" "10:45" "22:00"
##  [28] "18:06" "06:12" "13:10" "17:30" "12:27" "09:21" "14:23" "19:00" "08:33"
##  [37] "06:05" "13:38" "14:50" "22:43" "12:25" "18:54" "14:40" "23:30" "00:05"
##  [46] "12:12" "07:00" "19:39" "12:00" "22:27" "08:47" "08:23" "09:40" "07:43"
##  [55] "14:15" "15:03" "16:05" "13:15" "15:08" "15:50" "12:31" "15:52" "08:00"
##  [64] "09:00" "18:55" "13:53" "08:05" "04:44" "18:20" "08:40" "19:13" "00:22"
##  [73] "13:17" "00:00" "00:26" "13:20" "21:42" "11:30" "13:56" "16:10" "15:35"
##  [82] "16:14" "21:25" "12:32" "17:00" "17:55" "17:05" "07:34" "17:20" "09:43"
##  [91] "16:00" "14:46" "14:54" "19:49" "15:40" "17:40" "15:15" "17:01" "09:32"
## [100] "19:30" "07:35" "12:30" "19:33" "06:40" "18:09" "02:13" "03:25" "13:45"
## [109] "18:05" "14:53" "12:10" "13:55" "16:28" "13:41" "16:30" "07:13" "07:52"
## [118] "18:26" "17:50" "16:56" "20:23" "10:32" "11:57" "20:56" "14:32" "07:21"
## [127] "18:00" "12:15" "16:40" "13:30" "16:36" "17:32" "15:20" "22:36" "23:00"
## [136] "11:51" "12:07" "19:15" "19:50" "21:40" "16:42" "19:26" "19:31" "09:07"
## [145] "15:10" "22:41" "13:57" "08:49" "07:50" "11:14" "08:43" "08:59" "17:58"
## [154] "23:28" "00:40" "08:22" "10:20" "15:30" "22:06" "18:14" "11:10" "16:55"
## [163] "20:43" "20:50" "21:46" "21:00" "15:18" "06:09" "09:22" "14:33" "15:05"
## [172] "16:57" "09:29" "17:06" "17:21" "16:15" "10:58" "14:55" "21:28" "21:36"
## [181] "19:57" "12:40" "06:45" "22:09" "07:27" "08:28" "21:30" "15:22" "17:43"
## [190] "16:08" "05:38" "18:38" "19:35" "12:14" "12:05" "15:55" "10:25" "16:27"
## [199] "16:25" "16:17" "07:25" "19:43" "21:29" "13:00" "18:43" "17:15" "08:50"
## [208] "12:50" "07:53" "06:20" "12:20" "14:18" "21:22" "07:02" "17:45" "01:43"
## [217] "06:11" "00:12" "10:11" "16:07" "10:07" "00:55" "15:31" "13:13" "12:02"
## [226] "18:51" "07:44" "23:31" "22:59" "17:09" "17:13" "11:50" "16:51" "18:58"
## [235] "16:44" "14:20" "10:30" "14:07" "13:08" "14:26" "16:35" "10:55" "11:20"
## [244] "08:29" "08:15" "15:21" "08:20" "05:21" "13:02" "14:10" "18:15" "08:18"
## [253] "12:17" "18:59" "20:53" "11:32" "20:20" "13:31" "00:52" "12:04" "09:50"
## [262] "08:46" "18:56" "08:31" "15:36" "20:54" "21:01" "20:40" "09:11" "09:15"
## [271] "15:33" "17:08" "17:28" "08:30" "16:24" "16:18" "19:34" "14:02" "19:10"
## [280] "21:10" "10:28" "20:04" "11:29" "17:10" "13:37" "13:40" "07:15" "05:42"
## [289] "09:05" "16:20" "17:27" "17:25" "09:58" "19:23" "09:04" "15:45" "06:51"
## [298] "06:57" "10:48" "07:59" "06:00" "11:07" "19:02" "18:27" "17:31" "13:50"
## [307] "08:45" "05:50" "21:48" "14:43" "16:04" "12:36" "21:04" "16:49" "12:56"
## [316] "22:21" "11:36" "09:10" "15:25" "22:44" "08:11" "01:30" "06:10" "17:36"
## [325] "07:18" "19:25" "22:39" "07:56" "22:07" "07:05" "20:00" "09:51" "14:38"
## [334] "19:05" "08:55" "17:35" "23:14" "12:09" "16:45" "18:10" "21:49" "02:30"
## [343] "02:10" "21:08" "20:45" "14:28" "20:22" "11:55" "16:32" "23:45" "10:16"
## [352] "20:18" "16:03" "13:42" "20:35" "12:48" "21:32" "14:48" "00:20" "17:34"
## [361] "01:15" "12:45" "23:44" "12:11" "10:54" "23:15" "15:24" "03:30" "11:34"
## [370] "21:35" "15:48" "18:12" "09:45" "13:28" "03:05" "16:09" "21:45" "13:16"
## [379] "18:25" "07:55" "18:39" "22:20" "14:03" "01:31" "12:34" "06:39" "10:15"
## [388] "18:50" "05:22" "00:57" "01:20" "00:45" "11:35" "19:32" "19:04" "22:50"
## [397] "15:28" "22:13" "14:39" "11:31" "11:19" "20:07" "11:44" "11:08" "15:14"
## [406] "16:13" "11:25" "15:29" "08:10" "20:19" "22:55" "19:45" "08:58" "04:58"
## [415] "11:21" "11:53" "18:21" "12:16" "07:39" "10:12" "21:50" "07:26" "10:00"
## [424] "05:40" "17:17" "07:20" "11:23" "17:26" "05:39" "16:29" "11:28" "13:12"
## [433] "16:50" "15:11" "08:36" "11:15" "18:29" "20:58" "21:15" "13:18" "20:39"
## [442] "22:38" "04:05" "16:43" "23:04" "23:46" "08:48" "08:26" "23:35" "11:18"
## [451] "15:57" "11:12" "16:37" "09:55" "11:45" "22:57" "10:34" "10:50" "10:05"
## [460] "11:41" "11:09" "20:30" "12:39" "16:54" "07:40" "15:26" "17:41" "21:53"
## [469] "11:54" "12:51" "08:57" "11:40" "17:54" "13:43" "16:58" "08:35" "13:52"
## [478] "19:55" "00:10" "14:25" "18:13" "22:15" "16:19" "14:31" "02:55" "07:19"
## [487] "15:27" "07:45" "09:20" "00:09" "09:36" "17:49" "01:22" "00:46" "21:02"
## [496] "23:25" "17:33" "12:33" "14:56" "15:19" "02:00" "14:21" "19:08" "01:50"
## [505] "15:59" "15:42" "19:03" "06:30" "12:52" "23:05" "20:10" "22:45" "12:28"
## [514] "20:55" "15:41" "19:58" "13:58" "15:17" "13:27" "18:03" "08:09" "12:26"
## [523] "05:54" "17:38" "12:24" "16:16" "19:07" "20:05" "01:45" "19:14" "08:14"
## [532] "02:09" "12:43" "09:34" "07:30" "18:37" "00:50" "06:55" "13:32" "07:54"
## [541] "19:24" "20:15" "22:40" "08:25" "10:23" "12:19" "07:41" "14:09" "14:22"
## [550] "17:16" "04:15" "19:06" "15:58" "13:44" "18:11" "09:46" "06:47" "21:17"
## [559] "23:20" "19:11" "10:14" "15:39" "07:31" "09:02" "13:22" "08:54" "10:40"
## [568] "15:32" "09:14" "10:26" "10:29" "18:22" "18:57" "05:11" "06:25" "00:34"
## [577] "06:38" "03:15" "14:29" "02:49" "12:08" "03:48" "09:26" "08:16" "11:00"
## [586] "04:35" "01:58" "22:01" "21:05" "20:47" "03:23" "18:16" "13:05" "04:55"
## [595] "09:08" "09:13" "01:00" "12:55" "18:45" "20:03" "22:35" "08:24" "09:01"
## [604] "04:30" "06:53" "11:48" "10:27" "14:13" "22:05" "07:12" "14:51" "09:31"
## [613] "01:10" "19:46" "16:23" "22:10" "08:38" "14:37" "23:16" "07:38" "03:52"
## [622] "19:42" "14:14" "18:23" "19:51" "22:08" "21:58" "11:05" "12:29" "20:52"
## [631] "15:37" "23:50" "03:08" "19:40" "09:48" "09:27" "09:59" "20:12" "06:29"
## [640] "13:46" "07:07" "16:12" "02:43" "18:04" "04:54" "20:02" "10:31" "13:09"
## [649] "14:45" "13:39" "14:04" "10:24" "19:27" "23:11" "13:19" "16:53" "06:50"
## [658] "11:58" "21:16" "05:15" "13:47" "22:25"
```

Display Updated Rows

Task 14: Viewing the top 5 rows to confirm the new time format.

```
head(data, 5)
```

```
##   Number_of_Vehicles Accident_Date Time_24hr_ First_Road_Class Road_Surface
## 1                  2    2017-01-01      21:20                U     Wet/Damp
## 2                  2    2017-01-04      15:00                U          Dry
## 3                  2    2017-01-05      07:32              A58     Wet/Damp
## 4                  2    2017-01-05      09:30             A646     Wet/Damp
## 5                  2    2017-01-14      09:09                U    Frost/Ice
##   Lighting_Conditions Daylight_Dark Weather_Conditions Local_Authority
## 1                   4          Dark                  2      Calderdale
## 2                   1      Daylight                  1      Calderdale
## 3                   4          Dark                  1      Calderdale
## 4                   1      Daylight                  1      Calderdale
## 5                   1      Daylight                  1      Calderdale
##   Type_of_Vehicle Casualty_Class Casualty_Severity Sex_of_Casualty
## 1               9              2                 2               2
## 2               9              3                 2               2
## 3               9              1                 3               1
## 4               4              1                 1               1
## 5               9              1                 3               1
##   Age_of_Casualty
## 1              16
## 2              67
## 3              56
## 4              20
## 5              46
```

Missing Values Check

Task 15: Checking for missing values in each column and printing the results.

```
missing_per_column <- colSums(is.na(data))
print(missing_per_column)
```

```
##  Number_of_Vehicles       Accident_Date          Time_24hr_    First_Road_Class
##                   0                   0                   0                   0
##        Road_Surface Lighting_Conditions       Daylight_Dark  Weather_Conditions
##                   0                   0                   0                   0
##     Local_Authority     Type_of_Vehicle      Casualty_Class   Casualty_Severity
##                   0                   0                   0                   0
##     Sex_of_Casualty     Age_of_Casualty
##                   0                  19
```

Task 16: Creating contingency tables for columns to assess missing data patterns.

```r
# Define the columns to check
categorical_vars <- c("First_Road_Class", "Road_Surface", "Lighting_Conditions", "Weather_Conditions",
                      "Type_of_Vehicle", "Casualty_Class", "Casualty_Severity", "Sex_of_Casualty")

# Loop through categorical variables to create contingency tables
for (var in categorical_vars) {
  contingency_table <- table(is.na(data$Age_of_Casualty), data[[var]])
  cat("\nContingency table for column:", var, "\n")
  print(contingency_table)
}
```

```
##
## Contingency table for column: First_Road_Class
##
##           1   2   3   4   6 A58 A6025 A6026 A6026(M) A6033 A6036 A6139 A62 A629
##   FALSE 147   3 768  98 708  25     3     1        1     5     5     2   2   12
##   TRUE    1   0   5   0   3   2     0     0        0     0     0     0   0    0
##
##         A629(M) A6319 A641 A643 A644 A646 A647 A649 A672 A681 B6112 B6113 B6114
##   FALSE       1     1   13    2   11   25    1    1    8    1     5     5     3
##   TRUE        0     0    0    0    0    1    0    0    0    0     0     0     0
##
##         B6138 M62   U
##   FALSE     1  31 161
##   TRUE      0   0   7
##
## Contingency table for column: Road_Surface
##
##         Wet \xa8 Damp    1    2    3    4    5  Dry Frost/Ice  Ice Snow  Wet
##   FALSE             4 1150  542   15   11    6  233         4    1    2    1
##   TRUE              0    3    6    0    0    0    6         0    0    0    0
##
##         Wet/Damp
##   FALSE       81
##   TRUE         4
##
## Contingency table for column: Lighting_Conditions
##
##            1    4    5    6    7
##   FALSE 1441  518    4   18   69
##   TRUE    10    9    0    0    0
##
## Contingency table for column: Weather_Conditions
##
##            1    2    3    4    5    6    7    8    9
##   FALSE 1666  230   16   17   64    8   18   12   19
##   TRUE    15    3    0    0    1    0    0    0    0
##
## Contingency table for column: Type_of_Vehicle
##
##            1    2    3    4    5    8    9   10   11   16   17   18   19   20
##   FALSE  142   21   65   21   58   78 1515    3   36    2    2    3   63    8
##   TRUE     1    1    0    2    1    0   14    0    0    0    0    0    0    0
##
##           21   22   23   90   97
##   FALSE   21    1    1    9    1
##   TRUE     0    0    0    0    0
##
## Contingency table for column: Casualty_Class
##
##            1    2    3
##   FALSE 1216  463  371
##   TRUE     9    3    7
##
## Contingency table for column: Casualty_Severity
##
##            1    2    3
##   FALSE   25  294 1731
##   TRUE     0    5   14
##
## Contingency table for column: Sex_of_Casualty
##
##            1    2
##   FALSE 1238  812
##   TRUE    14    5
```

Example of Missing Data Assessment

Task 17: Example of MAR and NMAR Assessment:

1. **Example of MAR:** Missing values in 'Age_of_Casualty' could be influenced by other observed factors such as 'Casualty_Severity'. This suggests the missing data leans towards MAR (Missing At Random).
2. **Example of NMAR:** Missing values in 'Age_of_Casualty' might be influenced by unobserved factors, such as the age being systematically unreported for certain groups. Since the data appears to be Missing At Random (MAR), we can use a linear model to impute the missing values. This will be addressed in "Part 3: Regression".

Duplicate Rows Check

Task 18: Identifying and counting duplicate rows in the dataset.

```
# Check for duplicate rows
duplicate_rows <- data[duplicated(data), ]
num_duplicate_rows <- nrow(duplicate_rows)
print(num_duplicate_rows)
```

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

Road Surface Anomalies

Task 19: Displaying and cleaning anomalies in the 'Road_Surface' column.

```
unique(data$Road_Surface)
```

```
##  [1] "Wet/Damp"      "Dry"           "Frost/Ice"     "Ice"
##  [5] "Snow"          "Wet"           "Wet \xa8 Damp" "2"
##  [9] "1"             "3"             "4"             "5"
```

Clean Road Surface Anomalies

Task 20: Replacing variations in 'Road_Surface' with standardized values.

```
data$Road_Surface <- recode(data$Road_Surface,
  "1" = "Dry",
  "2" = "Wet/Damp",
  "3" = "Snow",
  "4" = "Frost/Ice",
  "5" = "Flood",
  "Wet \xa8 Damp" = "Wet",
  "Wet/Damp" = "Wet",
  "Frost/Ice" = "Ice"
)

unique(data$Road_Surface)
```

```
## [1] "Wet"       "Dry"       "Ice"       "Snow"      "Wet/Damp"  "Frost/Ice"
## [7] "Flood"
```

First Road Class Anomalies

Task 21: Displaying distinct values and cleaning anomalies in the 'First_Road_Class' column.

```
unique(data$First_Road_Class)
```

```
##  [1] "U"        "A58"      "A646"     "B6138"    "A629"     "A641"
##  [7] "A672"     "A6033"    "A6139"    "A644"     "A62"      "B6114"
## [13] "A6319"    "B6112"    "M62"      "A681"     "B6113"    "A629(M)"
## [19] "A643"     "A6036"    "A6025"    "A647"     "A6026(M)" "A649"
## [25] "A6026"    "3"        "6"        "1"        "4"        "2"
```

Clean First Road Class Anomalies

Task 22: Standardizing values in 'First_Road_Class' using specific mappings and conditions.

```r
data$First_Road_Class <- recode(data$First_Road_Class,
  "6" = "U",
  "1" = "M",
  "2" = "A(M)",
  "3" = "A",
  "4" = "B",
  "5" = "C",
)

# Modify values based on additional conditions using grepl
data$First_Road_Class <- ifelse(grepl("^A.*\\(M\\)$", data$First_Road_Class),
                                "A(M)",
                                ifelse(grepl("^A.*$", data$First_Road_Class),
                                       "A",
                                       ifelse(grepl("^B.*$", data$First_Road_Class),
                                              "B",
                                              ifelse(grepl("^C.*$", data$First_Road_Class),
                                                     "C",
                                                     ifelse(grepl("^M.*$", data$First_Road_Class),
                                                            "M",
                                                            data$First_Road_Class
                                                     )
                                              )
                                       )
                                )
)

# Display unique values after recoding
unique(data$First_Road_Class)
```

```
## [1] "U"    "A"    "B"    "M"    "A(M)"
```

Identify Unnecessary Columns

Task 23: Listing all columns to identify which ones are unnecessary.

```
names(data)
```

```
##  [1] "Number_of_Vehicles"  "Accident_Date"       "Time_24hr_"
##  [4] "First_Road_Class"    "Road_Surface"        "Lighting_Conditions"
##  [7] "Daylight_Dark"       "Weather_Conditions"  "Local_Authority"
## [10] "Type_of_Vehicle"     "Casualty_Class"      "Casualty_Severity"
## [13] "Sex_of_Casualty"     "Age_of_Casualty"
```

Drop Unnecessary Column

Task 24: Removing the 'Local_Authority' column as it is non-informative for the analysis.

```
data <- data %>% select(-Local_Authority)
```

Compare Columns for Redundancy

Task 25: Comparing 'Lighting_Conditions' and 'Daylight_Dark' columns for redundancy.

```
unique(data$Lighting_Conditions)
```

```
## [1] 4 1 5 7 6
```

```
unique(data$Daylight_Dark)
```

```
## [1] "Dark"     "Daylight" ""
```

Drop Redundant Column

Task 26: Dropping 'Daylight_Dark' column due to redundancy and ease of working with integers.

```
data <- data %>% select(-Daylight_Dark)
```

Remove Duplicate Rows

Task 27: Keeping only unique rows and displaying the dimensions of the cleaned dataset.

```r
data <- unique(data)
dim(data)
```

```
## [1] 2053   12
```

Display Cleaned Data

Task 28: Showing the first 100 rows of the cleaned dataset.

```r
head(data, 100)
```

```
##     Number_of_Vehicles Accident_Date Time_24hr_ First_Road_Class Road_Surface
## 1                    2    2017-01-01      21:20                U          Wet
## 2                    2    2017-01-04      15:00                U          Dry
## 3                    2    2017-01-05      07:32                A          Wet
## 4                    2    2017-01-05      09:30                A          Wet
## 5                    2    2017-01-14      09:09                U          Ice
## 6                    1    2017-01-15      16:59                U          Wet
## 7                    1    2017-01-16      10:59                A          Wet
## 8                    3    2017-01-19      18:49                B          Dry
## 9                    1    2017-01-20      14:08                U          Dry
## 10                   1    2017-01-22      13:25                A          Wet
## 11                   2    2017-01-24      11:26                A          Wet
## 12                   2    2017-01-24      18:30                A          Dry
## 13                   2    2017-01-25      12:58                U          Dry
## 14                   3    2017-01-26      11:33                A          Wet
## 15                   1    2017-01-26      15:46                A          Dry
## 16                   1    2017-01-27      13:35                U          Ice
## 17                   2    2017-01-27      14:05                U          Dry
## 18                   2    2017-01-27      22:33                U          Wet
## 19                   2    2017-01-27      18:35                U          Wet
## 20                   1    2017-01-29      14:30                U          Wet
## 21                   2    2017-01-29      22:30                U          Wet
## 22                   1    2017-01-30      14:44                U          Dry
## 23                   2    2017-01-31      14:35                A          Wet
## 24                   2    2017-01-31      18:40                U          Wet
## 25                   2    2017-02-01      14:00                A          Wet
## 26                   2    2017-02-02      10:45                U          Wet
## 27                   2    2017-02-02      22:00                A          Dry
## 28                   1    2017-02-03      18:06                A          Dry
## 29                   3    2017-02-04      06:12                A          Wet
## 30                   2    2017-02-04      13:10                A          Dry
## 31                   1    2017-02-04      17:30                A          Dry
## 32                   2    2017-02-05      12:27                A          Dry
## 33                   1    2017-02-07      09:21                U          Dry
## 34                   2    2017-02-07      14:23                B          Dry
## 35                   2    2017-02-07      15:46                A          Dry
## 36                   1    2017-02-06      19:00                U          Wet
## 37                   1    2017-02-08      08:33                U          Dry
## 38                   1  2017-02-08     14:30             A          Wet
## 39                   1  2017-02-08     17:30             U          Dry
## 40                   2  2017-02-08     18:30             U          Dry
## 41                   4  2017-02-11     06:05             B         Snow
## 42                   3  2017-02-12     13:38             M          Wet
## 43                   3  2017-02-12     14:50             A          Wet
## 44                   2  2017-02-13     22:43             A          Dry
## 45                   2  2017-02-15     12:25             A          Dry
## 46                   2  2017-02-15     14:30             A          Dry
## 47                   1  2017-02-15     18:54             U          Wet
## 48                   2  2017-02-18     14:40             U          Dry
## 49                   1  2017-02-18     23:30             U          Wet
## 50                   1  2017-02-19     00:05             A          Wet
## 51                   3  2017-02-20     12:12             A          Wet
## 52                   2  2017-02-21     07:00             B          Dry
## 53                   1  2017-02-22     19:39             A          Wet
## 54                   1  2017-02-24     12:00             U          Dry
## 55                   1  2017-02-24     22:27             B          Dry
## 56                   4  2017-02-28     08:47             M          Wet
## 57                   2  2017-03-04     15:00             U          Dry
## 58                   2  2017-03-07     08:23             A          Dry
## 59                   2  2017-08-10     09:40             U          Wet
## 60                   3  2017-03-16     07:43             A          Dry
## 61                   2  2017-03-16     14:15             A          Dry
## 62                   1  2017-03-17     15:03             U          Wet
## 63                   1  2017-03-17     16:05             A          Dry
## 64                   2  2017-03-17     13:15             M          Dry
## 65                   2  2017-03-19     15:08             U          Dry
## 66                   2  2017-03-20     07:00             M          Wet
## 67                   2  2017-03-20     15:50             A          Dry
## 68                   1  2017-03-21     12:31             A          Dry
## 69                   1  2017-03-21     15:52             A          Wet
## 70                   1  2017-03-22     08:00             U          Wet
## 71                   2  2017-03-23     09:00             A          Dry
## 72                   2  2017-03-25     18:55             A          Dry
## 73                   2  2017-03-26     13:53             U          Dry
## 74                   2  2017-03-29     08:05             M          Dry
## 75                   2  2017-03-30     04:44             A          Wet
## 76                   1  2017-04-01     18:20             U          Dry
## 77                   3  2017-04-03     13:15             A          Dry
## 78                   1  2017-04-04     08:40             U          Dry
## 79                   2  2017-04-05     19:13             A          Dry
## 80                   1  2017-04-07     00:22             M          Dry
## 81                   1  2017-04-07     13:17             U          Dry
## 82                   2  2017-04-07     00:00             B          Dry
## 83                   1  2017-04-08     00:26             U          Dry
## 84                   1  2017-04-08     13:20             A          Dry
## 85                   2  2017-04-09     21:42             U          Dry
## 86                   2  2017-03-31     11:30             A          Dry
## 87                   2  2017-04-11     13:56             U          Dry
## 88                   1  2017-04-11     16:10             U          Dry
## 89                   3  2017-04-11     15:35             M          Dry
## 90                   1  2017-04-13     16:14             U          Dry
## 91                   2  2017-04-13     21:25             U          Dry
## 92                     1    2017-04-21      13:25               U          Dry
## 93                     1    2017-04-23      12:32               M          Dry
## 94                     2    2017-04-23      18:35               U          Dry
## 95                     2    2017-04-25      17:00               U          Wet
## 96                     3    2017-04-25      17:55               M          Dry
## 97                     2    2017-04-25      17:05               U          Wet
## 98                     2    2017-04-26      06:05               U          Dry
## 99                     2    2017-04-27      07:34            A(M)          Wet
## 100                    2    2017-04-27      17:20               U          Dry
##     Lighting_Conditions Weather_Conditions Type_of_Vehicle Casualty_Class
## 1                     4                  2               9              2
## 2                     1                  1               9              3
## 3                     4                  1               9              1
## 4                     1                  1               4              1
## 5                     1                  1               9              1
## 6                     4                  1               9              3
## 7                     1                  1               9              3
## 8                     4                  1               9              1
## 9                     1                  1               9              3
## 10                    1                  1               9              1
## 11                    1                  2               9              2
## 12                    4                  1               9              1
## 13                    1                  1               9              1
## 14                    1                  1              19              1
## 15                    1                  1               9              3
## 16                    1                  1              19              3
## 17                    1                  1               9              1
## 18                    4                  2               9              1
## 19                    4                  1               9              1
## 20                    1                  5               9              3
## 21                    4                  7               9              1
## 22                    1                  1               8              3
## 23                    1                  2               9              1
## 24                    5                  2               9              1
## 25                    1                  1               9              1
## 26                    1                  1               9              1
## 27                    4                  1               9              1
## 28                    4                  1               9              3
## 29                    4                  1               9              1
## 30                    1                  1              19              1
## 31                    4                  1               9              3
## 32                    1                  1               9              1
## 33                    1                  1               9              1
## 34                    1                  1               9              1
## 35                    1                  1               9              1
## 36                    4                  1               9              3
## 37                    1                  1               9              3
## 38                    1                  1               9              3
## 39                    7                  1               3              3
## 40                    4                  1               9              1
## 41                    4                  3               9              1
## 42                    1                  6               9              1
## 43                    1                  3               9              1
## 44                    4                  1               9              1
## 45                    1                   1                 9                1
## 46                    1                   1                 9                1
## 47                    4                   1                 9                3
## 48                    1                   1                 9                1
## 49                    6                   1                 8                3
## 50                    4                   2                 9                1
## 51                    1                   2                 9                1
## 52                    7                   1                 9                2
## 53                    6                   2                 3                1
## 54                    1                   4                 9                3
## 55                    4                   1                 9                2
## 56                    1                   1                 9                1
## 57                    1                   1                 9                1
## 58                    1                   1                 9                1
## 59                    1                   1                 9                2
## 60                    1                   1                 9                2
## 61                    1                   1                 9                1
## 62                    1                   2                 9                3
## 63                    1                   1                21                3
## 64                    1                   4                 9                1
## 65                    1                   1                 9                1
## 66                    7                   5                 9                1
## 67                    1                   1                 9                1
## 68                    1                   1                 9                3
## 69                    1                   1                 9                2
## 70                    1                   2                 9                3
## 71                    1                   1                 9                1
## 72                    4                   1                 9                2
## 73                    1                   1                 2                1
## 74                    1                   1                 9                1
## 75                    4                   1                19                1
## 76                    7                   1                 9                3
## 77                    1                   1                 9                1
## 78                    1                   1                 9                3
## 79                    7                   1                 9                1
## 80                    4                   1                10                1
## 81                    1                   1                 9                2
## 82                    4                   1                 9                1
## 83                    4                   1                 9                1
## 84                    1                   1                 5                1
## 85                    4                   1                 9                1
## 86                    1                   1                 9                2
## 87                    1                   1                 9                1
## 88                    1                   1                11                3
## 89                    1                   1                19                1
## 90                    1                   1                 9                3
## 91                    4                   1                 9                1
## 92                    1                   1                 9                3
## 93                    1                   1                 9                1
## 94                    7                   1                 9                1
## 95                    7                   1                 9                1
## 96                    7                   1                19                1
## 97                    7                   2                 8                2
## 98                    7                   1                 1                1
## 99                   1                 1               9              1
## 100                  7                 1               9              1
##     Casualty_Severity Sex_of_Casualty Age_of_Casualty
## 1                   2               2              16
## 2                   2               2              67
## 3                   3               1              56
## 4                   1               1              20
## 5                   3               1              46
## 6                   3               1              NA
## 7                   3               2              25
## 8                   3               2              50
## 9                   3               2              64
## 10                  3               2              22
## 11                  3               2              21
## 12                  3               1              29
## 13                  3               1              24
## 14                  3               1              29
## 15                  3               1               5
## 16                  3               2              14
## 17                  3               2              33
## 18                  3               1              28
## 19                  3               1              NA
## 20                  3               2              27
## 21                  3               2              60
## 22                  3               1              50
## 23                  3               2              40
## 24                  2               2             115
## 25                  3               2              48
## 26                  3               2              76
## 27                  3               1              22
## 28                  3               2              14
## 29                  2               1              26
## 30                  3               2              65
## 31                  3               1              NA
## 32                  3               1              51
## 33                  3               1              67
## 34                  3               1              28
## 35                  3               1              34
## 36                  3               2              14
## 37                  3               2              13
## 38                  2               2              76
## 39                  3               2              45
## 40                  3               2              30
## 41                  3               1              35
## 42                  3               1              57
## 43                  3               1              26
## 44                  3               1              24
## 45                  2               1              68
## 46                  3               2              53
## 47                  3               2              11
## 48                  3               1              25
## 49                  2               2              39
## 50                  3               1              19
## 51                  3               2              74
## 52                 3                1               21
## 53                 2                1               30
## 54                 3                2               59
## 55                 3                2               17
## 56                 2                1               35
## 57                 3                1               46
## 58                 3                1               18
## 59                 3                2               48
## 60                 3                1                8
## 61                 3                1               53
## 62                 3                2               50
## 63                 3                2               45
## 64                 3                2               43
## 65                 3                1               22
## 66                 3                2               21
## 67                 3                1               57
## 68                 3                2               83
## 69                 3                1               19
## 70                 3                2               13
## 71                 3                1               46
## 72                 3                1                3
## 73                 2                1               NA
## 74                 3                1               25
## 75                 3                1               29
## 76                 3                1                2
## 77                 2                2               48
## 78                 3                1                9
## 79                 3                1               34
## 80                 3                1               52
## 81                 3                2                9
## 82                 3                1               68
## 83                 2                1               26
## 84                 3                2               35
## 85                 3                1               18
## 86                 3                2               63
## 87                 3                1               45
## 88                 3                1               23
## 89                 3                1               29
## 90                 3                1               47
## 91                 3                1               42
## 92                 3                2               46
## 93                 2                2               58
## 94                 3                1                5
## 95                 3                1               29
## 96                 3                2               29
## 97                 3                2               17
## 98                 3                1               25
## 99                 3                1               33
## 100                3                1               23
```

Outlier Detection Using Different Methods

3-Sigma Rule

Task 29: Detecting outliers in 'Age_of_Casualty' using the 3-sigma rule.

```r
# Filter out rows with NA values in Age_of_Casualty column
filtered_data <- data[!is.na(data$Age_of_Casualty), ]

age_of_casualty <- filtered_data$Age_of_Casualty

# Calculate mean and standard deviation for Age_of_Casualty
mean_age <- mean(age_of_casualty)
sd_age <- sd(age_of_casualty)

# Calculate upper and lower bounds for 3-sigma rule
upper_bound <- mean_age + 3 * sd_age
lower_bound <- mean_age - 3 * sd_age

# Identify outliers using the 3-sigma rule
outliers_sigma <- age_of_casualty[age_of_casualty > upper_bound | age_of_casualty < lower_bound]

# Print the outliers
print(outliers_sigma)
```

```
## [1] 115 100  98
```

```r
# Create a dataset for outliers
outliers_dataset <- filtered_data %>%
  filter(Age_of_Casualty > upper_bound | Age_of_Casualty < lower_bound)

# Print the number of outliers
number_of_outliers_sigma <- nrow(outliers_dataset)
print(paste("Number of outliers:", number_of_outliers_sigma))
```

```
## [1] "Number of outliers: 3"
```

```r
print(outliers_dataset)
```

```
##   Number_of_Vehicles Accident_Date Time_24hr_ First_Road_Class Road_Surface
## 1                  2    2017-01-31      18:40                U          Wet
## 2                  1    2016-06-16      12:15                U          Dry
## 3                  2    2014-06-08      16:50                A          Dry
##   Lighting_Conditions Weather_Conditions Type_of_Vehicle Casualty_Class
## 1                   5                  2               9              1
## 2                   1                  1               9              3
## 3                   1                  1               9              2
##   Casualty_Severity Sex_of_Casualty Age_of_Casualty
## 1                 2               2             115
## 2                 3               1             100
## 3                 3               2              98
```

Hampel Method

Task 30: Detecting outliers using the Hampel method.

```
# Remove NA values from Age_of_Casualty
age_casualty <- filtered_data$Age_of_Casualty

# Calculate the median and MAD
median_age <- median(age_casualty)
mad_age <- mad(age_casualty)

# Calculate Hampel's upper and lower bounds
hampel_upper_bound <- median_age + 3 * mad_age
hampel_lower_bound <- median_age - 3 * mad_age

# Identify outliers using Hampel identifier
outliers_hampel <- age_casualty[age_casualty > hampel_upper_bound | age_casualty < hampel_lower_bound]

# Print the outliers
print(outliers_hampel)
```

```
## [1] 115  93  93 100  92  95  98
```

```
# Create a dataset for Hampel outliers
outliers_hampel_dataset <- filtered_data %>%
  filter(Age_of_Casualty > hampel_upper_bound | Age_of_Casualty < hampel_lower_bound)

# Print the number of Hampel outliers
number_of_outliers_hampel <- nrow(outliers_hampel_dataset)
print(paste("Number of Hampel outliers:", number_of_outliers_hampel))
```

```
## [1] "Number of Hampel outliers: 7"
```

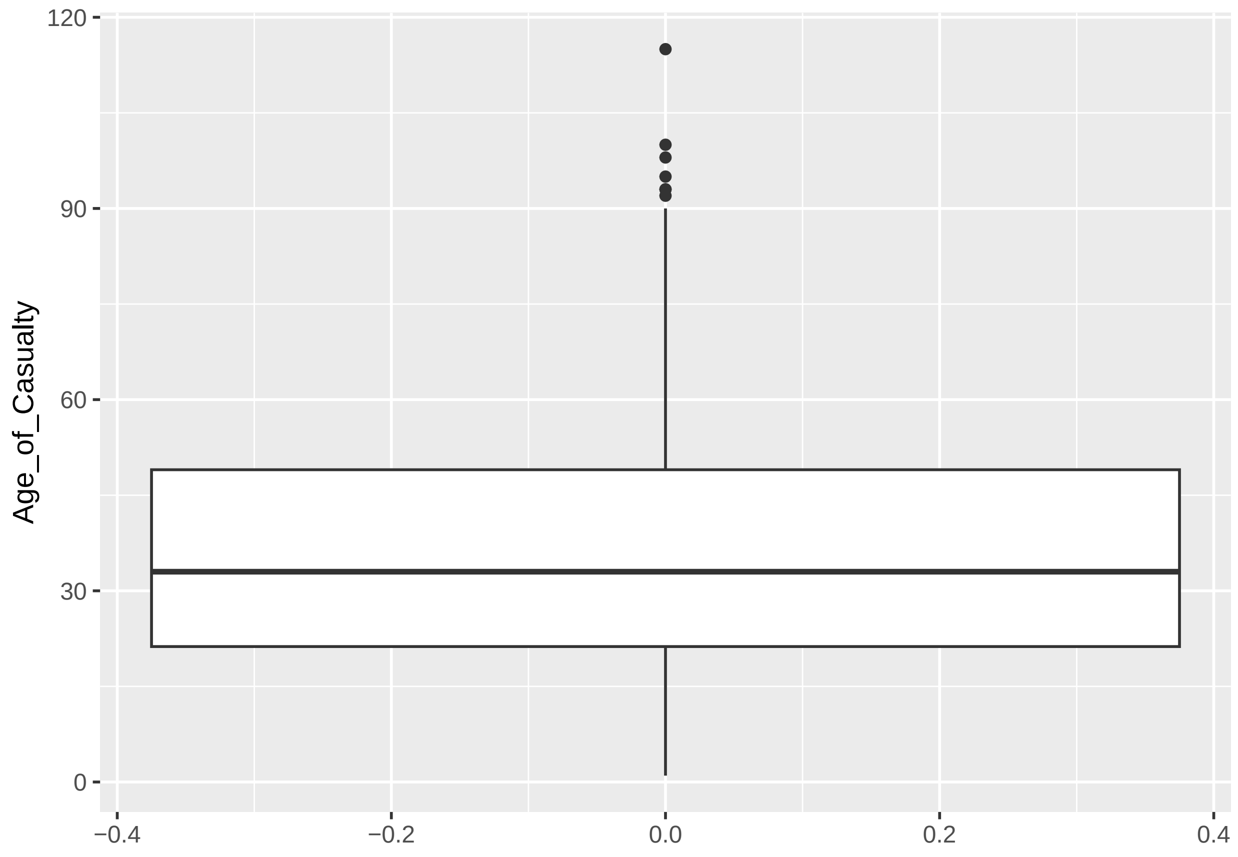
```
print(outliers_hampel_dataset)
```

```
##   Number_of_Vehicles Accident_Date Time_24hr_ First_Road_Class Road_Surface
## 1                  2    2017-01-31      18:40                U          Wet
## 2                  1    2017-06-24      12:00                A          Dry
## 3                  1    2016-03-19      19:02                U          Dry
## 4                  1    2016-06-16      12:15                U          Dry
## 5                  1    2016-06-28      16:27                A          Dry
## 6                  1    2014-02-05      17:15                U     Wet/Damp
## 7                  2    2014-06-08      16:50                A          Dry
##   Lighting_Conditions Weather_Conditions Type_of_Vehicle Casualty_Class
## 1                   5                  2               9              1
## 2                   1                  1               9              3
## 3                   1                  1               9              3
## 4                   1                  1               9              3
## 5                   1                  1               9              3
## 6                   4                  1               8              3
## 7                    1                  1               9              2
##   Casualty_Severity Sex_of_Casualty Age_of_Casualty
## 1                 2               2             115
## 2                 2               1              93
## 3                 2               1              93
## 4                 3               1             100
## 5                 2               2              92
## 6                 2               1              95
## 7                 3               2              98
```

Box Plot for Outlier Detection

Task 31: Visual inspection of outliers using a box plot.

```
# Visual inspection with boxplot
ggplot(filtered_data, aes(y = Age_of_Casualty)) +
  geom_boxplot()
```



#Question: Examine "Age of Casualty" for any outliers using the three methods discussed in the lectures. Compare the output of these methods. Which one is the best? Justify your answer. # Answer: In examining outliers in the "Age_of_Casualty" column using three distinct methods—namely the 3-sigma rule, Hampel method, and visual inspection with a box plot—a comparison reveals the strengths and applicability of each approach. The 3-sigma rule calculates outliers based on the mean and standard deviation, offering a straightforward numerical criterion that is effective for normally distributed data. It identified outliers in the “Age_of_Casualty” column using a clear statistical threshold. In contrast, the Hampel method, relying on the median and Median Absolute Deviation (MAD), proved robust against outliers and non-normally distributed data, accommodating skewed distributions better. Visual inspection with the box plot provided a graphical overview, facilitating quick identification of extreme values outside the whiskers. However, for precision and clarity in outlier detection, especially in datasets where normal distribution assumptions hold, the 3-sigma rule emerged as the preferred method. Its defined threshold based on standard deviations offers a reliable measure to pinpoint outliers in the “Age_of_Casualty” column, ensuring a systematic approach to data analysis and outlier management.

Export Cleaned Dataset

Task 32: Saving the cleaned dataset as a CSV file named ‘clean_accident.csv’.

```r
write.csv(data, "clean_accident.csv", row.names = FALSE)
```

Summary of Part 1: Wrangling Data

In Part 1 of the project, we focused on preprocessing and exploring a dataset on traffic accidents. Initially, we loaded necessary libraries and imported the dataset 'accidents.csv'. We renamed columns for clarity, adjusted date and time formats, and addressed missing data by assessing patterns across categorical variables. Outlier detection in the "Age_of_Casualty" column was conducted using the 3-sigma rule, Hampel method, and visual inspection via box plots, with the 3-sigma rule proving most suitable due to its clear statistical thresholds. We standardized column values, removed duplicates, and dropped redundant columns like 'Local_Authority' and 'Daylight_Dark'. The cleaned dataset was saved as 'clean_accident.csv', ensuring data integrity for further analysis.

Part 2: Exploration

In this part, we will do exploratory data analysis (EDA) and data visualization.

Task 1: Weather Conditions with More Male Driver Casualties than Female

Description: This analysis identifies weather conditions where male drivers have more accidents compared to female drivers, highlighting potential gender-specific risk factors in different weather scenarios.

```r
# Filter dataframe to include only drivers or riders (Casualty_Class == 1)
drivers_riders_data <- data %>%
  filter(Casualty_Class == 1)

# Calculate the number of accidents by gender (Sex_of_Casualty) and weather condition (Weather_Conditio
accidents_by_gender_weather <- drivers_riders_data %>%
  group_by(Weather_Conditions, Sex_of_Casualty) %>%
  summarise(total_accidents = n(), .groups = 'drop')

# Filter to find cases where male drivers (Sex_of_Casualty == 1) have more accidents than female driver
male_more_than_female <- accidents_by_gender_weather %>%
  filter(Sex_of_Casualty == 1 & total_accidents > lag(total_accidents)) %>%
  mutate(diff_more = total_accidents - lag(total_accidents),
         diff_more = ifelse(is.na(diff_more), total_accidents, diff_more))
```

```r
# Print the results
cat("Weather conditions where male drivers have more accidents than females:\n")
```

```
## Weather conditions where male drivers have more accidents than females:
```

```r
for (i in 1:nrow(male_more_than_female)) {
  cat(i, ". Under weather condition", male_more_than_female$Weather_Conditions[i],
      ", male drivers had", male_more_than_female$total_accidents[i],
      "more accidents than female drivers.\n")
}
```

```
## 1 . Under weather condition 4 , male drivers had 7 more accidents than female drivers.
## 2 . Under weather condition 5 , male drivers had 26 more accidents than female drivers.
## 3 . Under weather condition 7 , male drivers had 6 more accidents than female drivers.
```

```r
cat("\n")
```

Question 1: Is there any weather condition where male drivers/riders have more accidents than female drivers? Print out how many (e.g., "male drivers have —— more than females when weather is —-.

Answer: According to the analysis, there are weather conditions where male drivers/riders have more accidents than female drivers. Here are the specific findings:

a. Under weather condition 4 (Fine with high winds), male drivers (Sex_of_Casualty == 1) had 7 more accidents than female drivers (Sex_of_Casualty == 2).

b. Under weather condition 5 (Raining with high winds), male drivers had 26 more accidents than female drivers.

c. Under weather condition 7 (Fog or mist – if hazard), male drivers had 6 more accidents than female drivers.

These findings suggest that weather conditions like high winds and rain disproportionately affect male drivers in terms of accident rates compared to female drivers.

Task 2: Year with Highest Number of Casualties and Trend Over Time

Description: The graph below visualizes the trend in the number of casualties over the years, revealing insights into whether the frequency of accidents resulting in casualties has increased or decreased over time.

```r
# Extract the year from the Accident_Date
data <- data %>%
  mutate(Year = year(ymd(Accident_Date)))

# Group by year and calculate the total number of casualties
casualties_over_time <- data %>%
  group_by(Year) %>%
  summarise(total_casualties = n())
```

```r
# Find the year with the highest number of casualties
year_max_casualties <- casualties_over_time %>%
  arrange(desc(total_casualties)) %>%
  slice(1)

cat("Year with the highest number of casualties:\n")
```

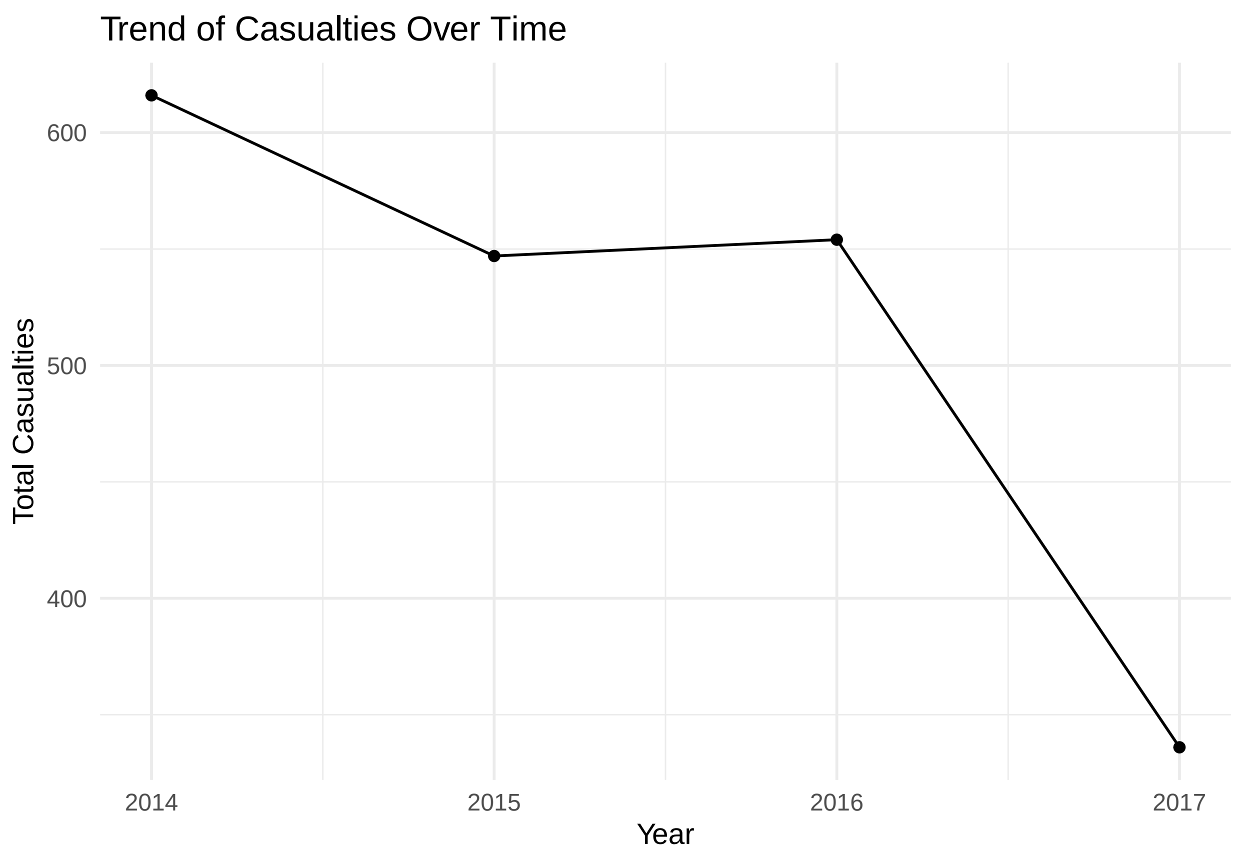
```
## Year with the highest number of casualties:
```

```r
print(year_max_casualties)
```

```
## # A tibble: 1 x 2
##    Year total_casualties
##   <dbl>            <int>
## 1  2014              616
```

```r
cat("\n")
```

```r
# Plot the trend of casualties over time
ggplot(casualties_over_time, aes(x = Year, y = total_casualties)) +
  geom_line() +
  geom_point() +
  labs(title = "Trend of Casualties Over Time", x = "Year", y = "Total Casualties") +
  theme_minimal()
```



Question 2: Is the number of casualties increased or decreased over time? Which year has the highest number of casualties?

Answer: The number of casualties has generally decreased over time. The year 2014 had the highest number of casualties according to the data. This trend indicates that efforts in road safety and possibly improvements in infrastructure or driving habits may have contributed to reducing the overall number of accidents and casualties over the years.

Task 3: Relationship Between Light Conditions and Severity

Description: The bar chart below illustrates how different light conditions affect the severity of accidents, providing insights into how visibility influences the outcomes of traffic incidents.

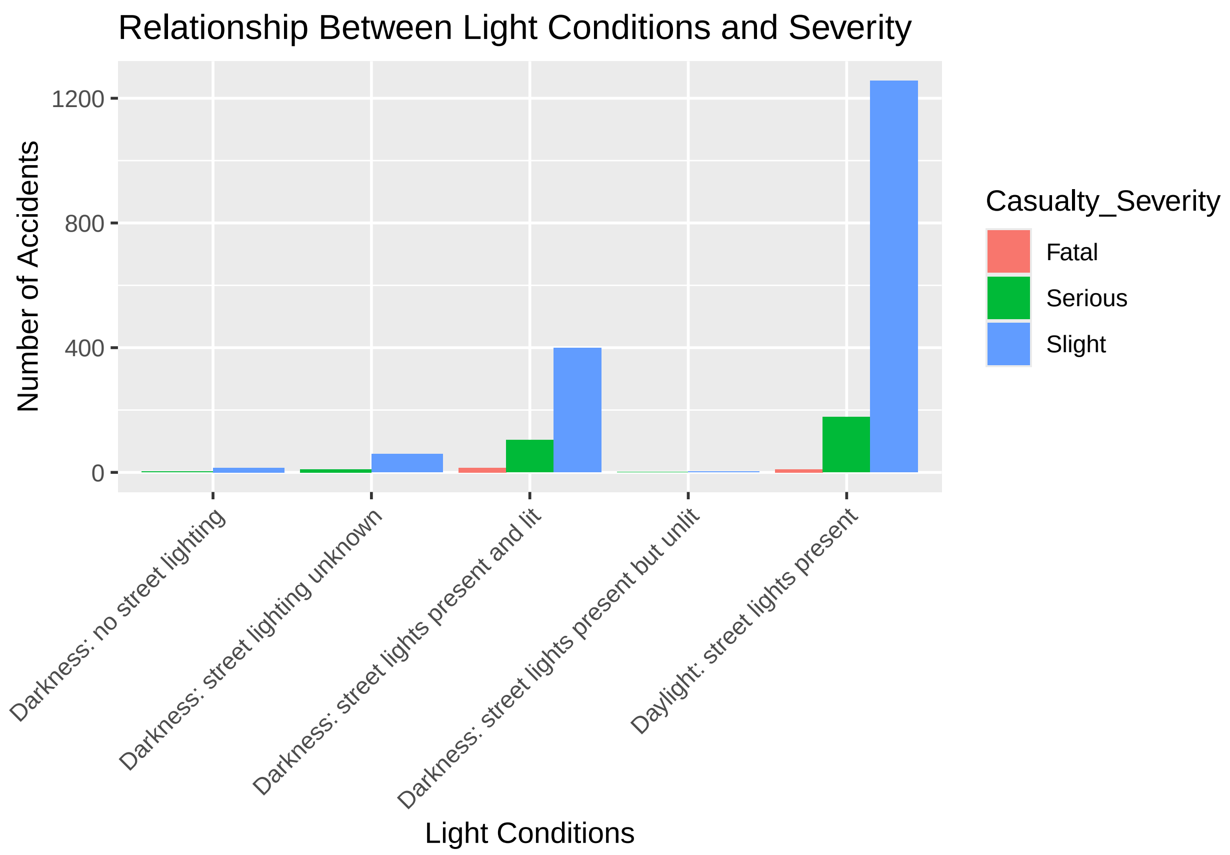
```r
unique(data$Lighting_Conditions)
```

```
## [1] 4 1 5 7 6
```

```r
unique(data$Casualty_Severity)
```

```
## [1] 2 3 1
```

```r
# Create a mapping from integers to lighting conditions characters
lighting_conditions_mapping <- c(
  "1" = "Daylight: street lights present",
  "2" = "Daylight: no street lighting",
  "3" = "Daylight: street lighting unknown",
  "4" = "Darkness: street lights present and lit",
  "5" = "Darkness: street lights present but unlit",
  "6" = "Darkness: no street lighting",
  "7" = "Darkness: street lighting unknown"
)

# Create a mapping from integers to casualty severity in string
casualty_severity_mapping <- c(
  "1" = "Fatal",
  "2" = "Serious",
  "3" = "Slight"
)

# Replace Lighting Conditions and Casualty Severity integers with corresponding character labels
data <- data %>%
  mutate(Lighting_Conditions = lighting_conditions_mapping[as.character(Lighting_Conditions)],
        Casualty_Severity = casualty_severity_mapping[as.character(Casualty_Severity)])

# Group by the original lighting conditions and severity, then count the number of accidents
light_severity_summary <- data %>%
  group_by(Lighting_Conditions, Casualty_Severity) %>%
  summarise(accident_count = n(), .groups = 'drop')

# Plot the relationship between light conditions and severity using original character labels for light
ggplot(light_severity_summary, aes(x = Lighting_Conditions, y = accident_count, fill = Casualty_Severit
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Relationship Between Light Conditions and Severity", x = "Light Conditions", y = "Numbe
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Observation

The plot illustrates how light conditions impact the severity of accidents. It shows that accidents under darkness (no street lighting) tend to have higher severity compared to daylight conditions. This insight suggests that adequate street lighting could potentially reduce the severity of accidents during darker hours.

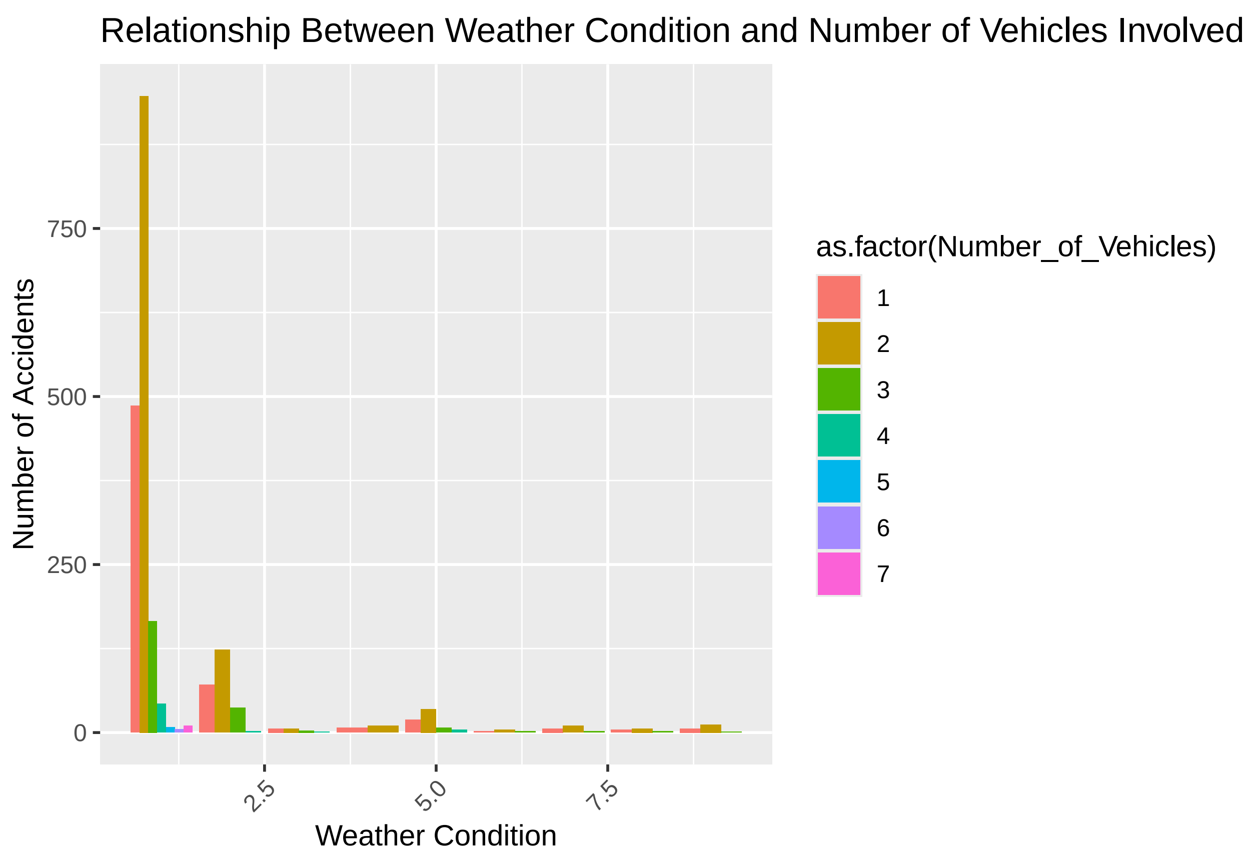
Task 4: Relationship Between Weather Condition and Number of Vehicles Involved

Description: This grouped bar chart below explores how different weather conditions correlate with the number of vehicles involved in accidents, shedding light on environmental factors influencing accident severity and complexity.

```r
# Group by weather condition and number of vehicles involved, then count the number of accidents
weather_vehicles_summary <- data %>%
  group_by(Weather_Conditions, Number_of_Vehicles) %>%
  summarise(accident_count = n())
```

```
## `summarise()` has grouped output by 'Weather_Conditions'. You can override
## using the `.groups` argument.
```

```r
# Plot the relationship between weather condition and number of vehicles involved
ggplot(weather_vehicles_summary, aes(x = Weather_Conditions, y = accident_count, fill = as.factor(Numbe
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Relationship Between Weather Condition and Number of Vehicles Involved", x = "Weather Co
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Observation

The graph depicts how weather conditions influence the number of vehicles involved in accidents. Notably, poor weather conditions such as rain and snow correlate with an increase in accidents involving multiple vehicles. This correlation implies that adverse weather conditions contribute to more complex accident scenarios, possibly due to reduced visibility and road traction.

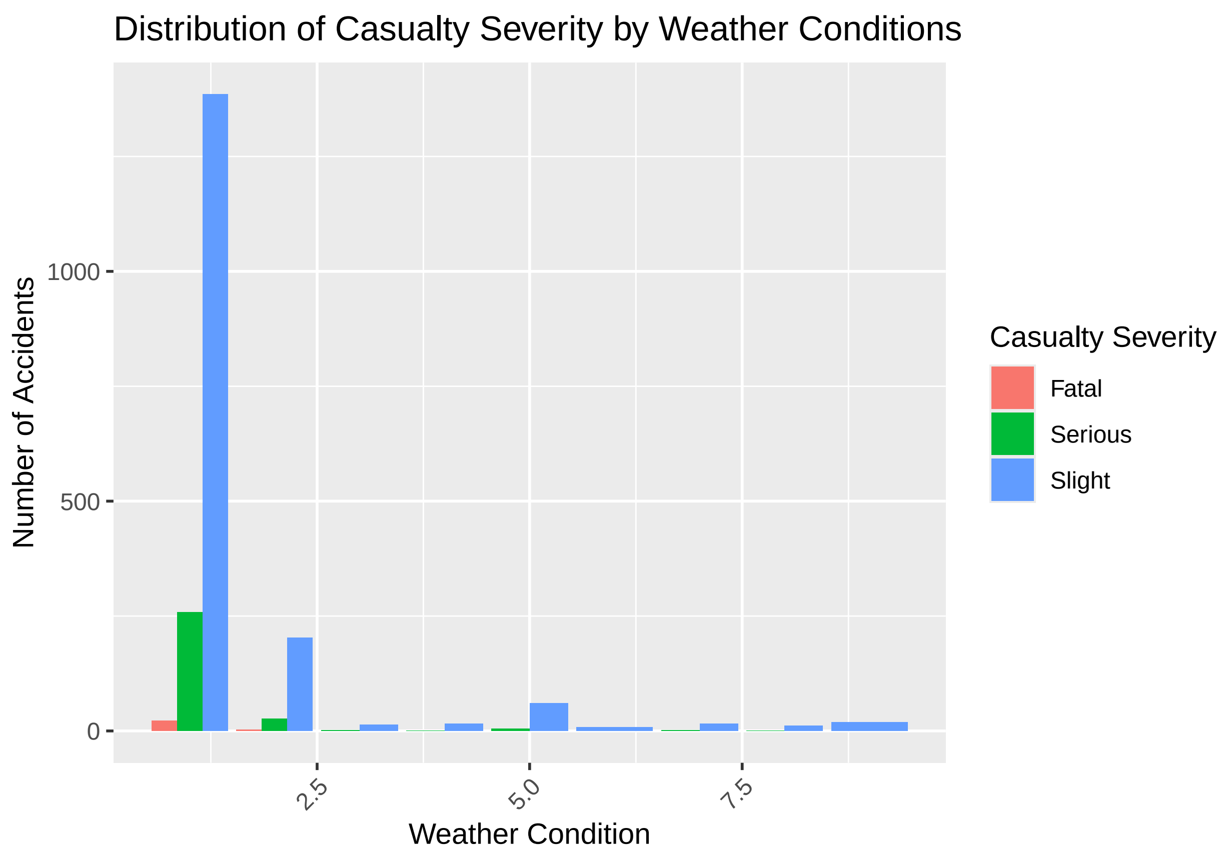
Task 5: Distribution of Casualty Severity by Weather Conditions

Description: This line plot below shows the trend in casualties over time across different lighting conditions, highlighting variations in accident frequencies during daylight versus darkness periods.

```
# Group by weather condition and casualty severity, then count the number of accidents
weather_severity_summary <- data %>%
  group_by(Weather_Conditions, Casualty_Severity) %>%
  summarise(accident_count = n())
```

```
## `summarise()` has grouped output by 'Weather_Conditions'. You can override
## using the `.groups` argument.
```

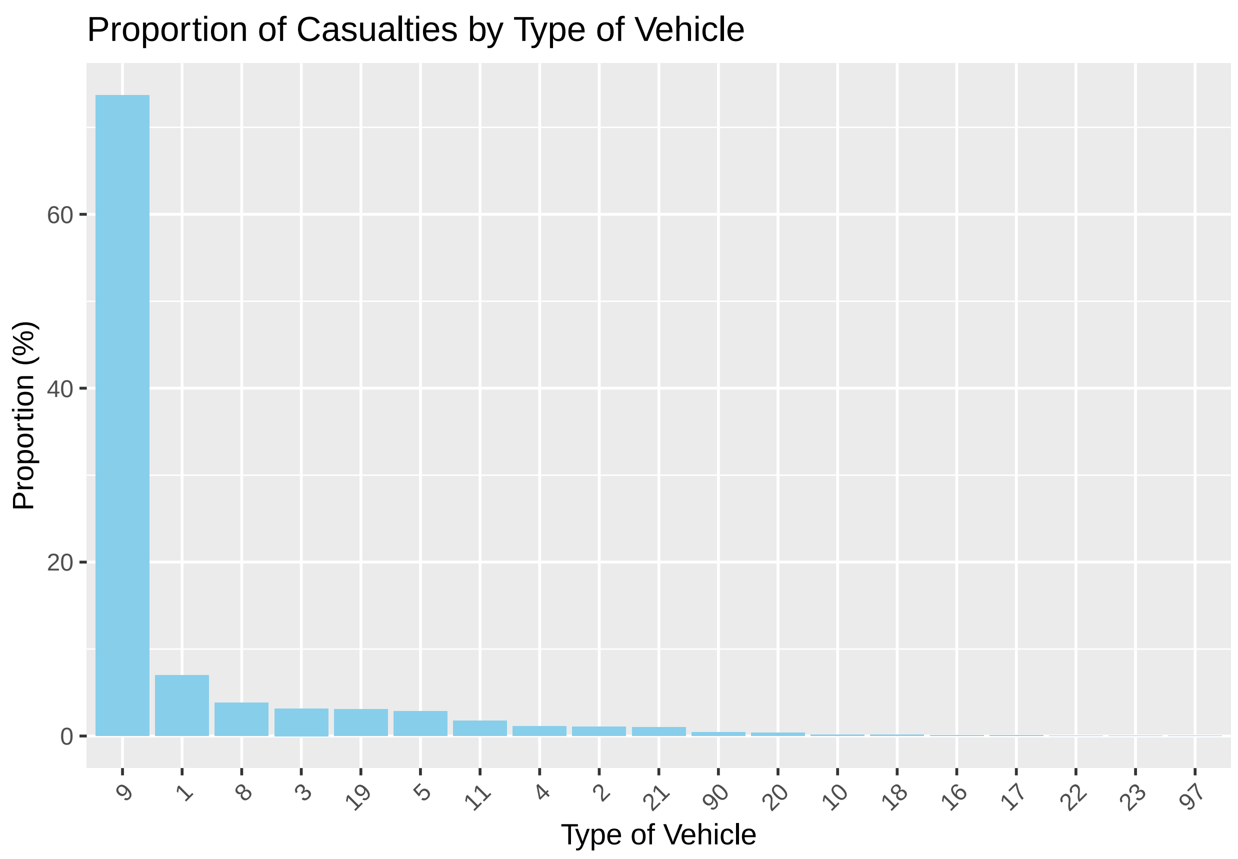
```
# Plot the distribution of casualty severity across different weather conditions
ggplot(weather_severity_summary, aes(x = Weather_Conditions, y = accident_count, fill = Casualty_Severi
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Distribution of Casualty Severity by Weather Conditions", x = "Weather Condition", y =
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Task 6: Proportion of Casualties by Type of Vehicle

Description: The histogram shown below visualizes the distribution of age among casualties, providing insights into the demographics most affected by traffic accidents.

```
# Calculate proportions of casualties by type of vehicle
vehicle_casualty_summary <- data %>%
  group_by(Type_of_Vehicle) %>%
  summarise(total_casualties = n()) %>%
  mutate(proportion = total_casualties / sum(total_casualties) * 100) %>%
  arrange(desc(proportion))

# Plot the proportion of casualties by type of vehicle
ggplot(vehicle_casualty_summary, aes(x = reorder(Type_of_Vehicle, -proportion), y = proportion)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Proportion of Casualties by Type of Vehicle", x = "Type of Vehicle", y = "Proportion (%
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Task 7: Casualties Over Time by Daylight vs. Darkness

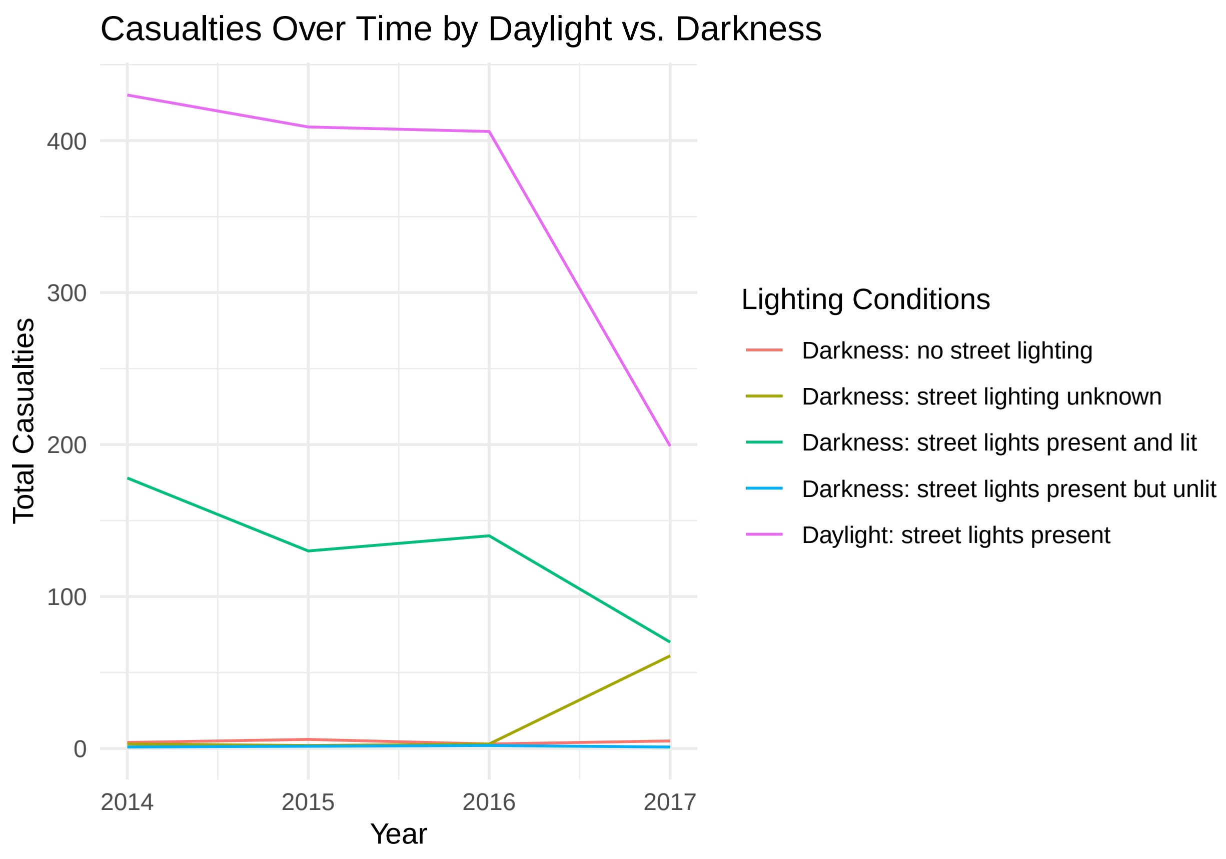
Description: The chart below illustrates the distribution of accident severity levels, offering a clear view of how severe and non-severe accidents compare within the dataset.

```
# Ensure Accident_Date is in Date format
data$Accident_Date <- as.Date(data$Accident_Date, format = "%d/%m/%Y")

# Extract the year from the Accident_Date
```

```r
data <- data %>%
  mutate(Year = lubridate::year(Accident_Date))

# Group by year, daylight/dark conditions, and calculate total casualties
daylight_casualties <- data %>%
  group_by(Year, Lighting_Conditions) %>%
  summarise(total_casualties = n())
```

```
## 'summarise()' has grouped output by 'Year'. You can override using the
## '.groups' argument.
```

```r
# Plot casualties over time by daylight vs. darkness conditions
ggplot(daylight_casualties, aes(x = Year, y = total_casualties, color = Lighting_Conditions)) +
  geom_line() +
  labs(title = "Casualties Over Time by Daylight vs. Darkness", x = "Year", y = "Total Casualties", col
  theme_minimal()
```



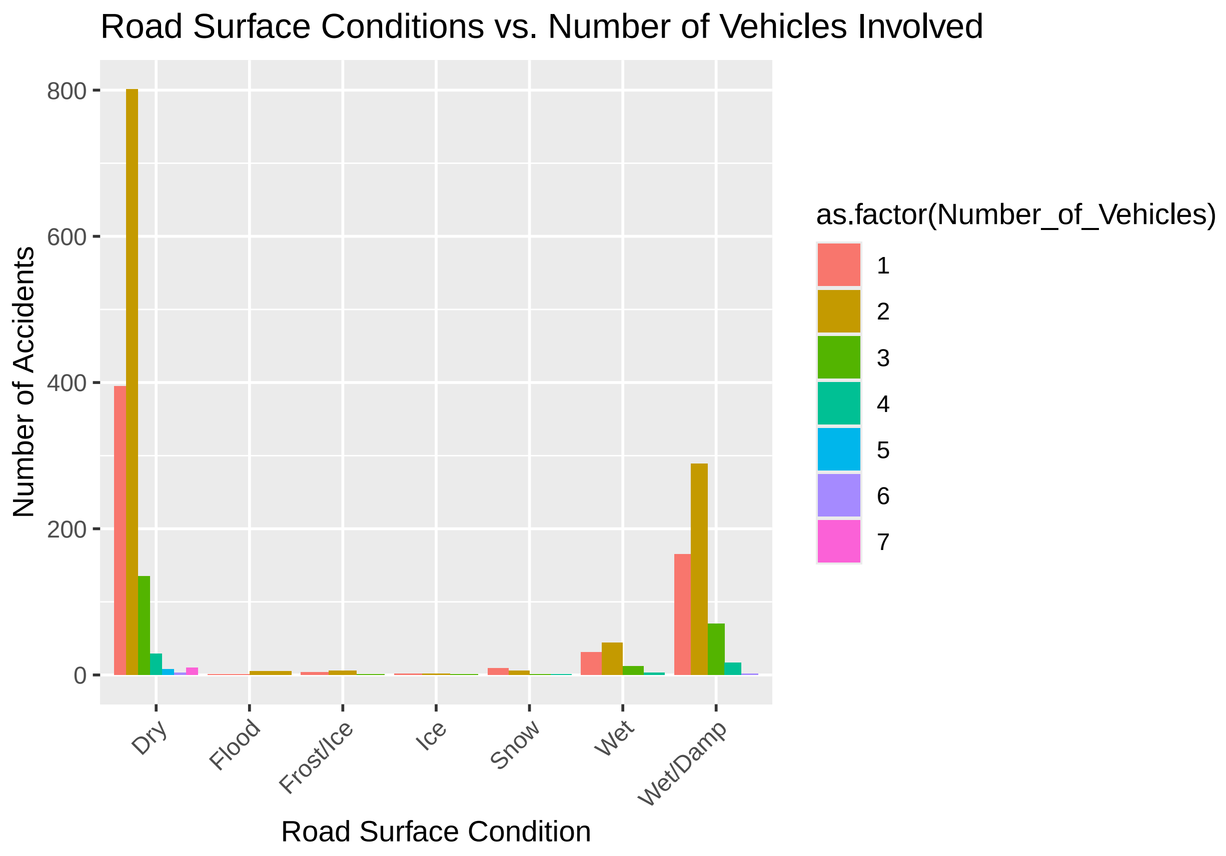
Task 8: Road Surface Conditions vs. Number of Vehicles Involved

Description: This bar chart below displays the distribution of different road surface conditions involved in accidents, offering insights into which types of surfaces are most frequently associated with traffic incidents.

```
# Group by road surface condition and number of vehicles involved, then count the number of accidents
road_surface_vehicles_summary <- data %>%
  group_by(Road_Surface, Number_of_Vehicles) %>%
  summarise(accident_count = n())
```

```
## ‘summarise()‘ has grouped output by ’Road_Surface’. You can override using the
## ‘.groups‘ argument.
```

```
# Plot the relationship between road surface condition and number of vehicles involved
ggplot(road_surface_vehicles_summary, aes(x = Road_Surface, y = accident_count, fill = as.factor(Number_
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Road Surface Conditions vs. Number of Vehicles Involved", x = "Road Surface Condition",
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Summary of Part 2: Exploration

Part 2 of the analysis explored different aspects of traffic accidents to understand what influences how severe accidents are, who gets affected the most, and how things change over time. We found that in certain weather conditions like rain and strong winds, men tend to have more accidents compared to women. The analysis also showed that accidents during bad weather often involve more vehicles. Over the years, there has been a decrease in the number of accidents causing injuries, with the year 2014 having the most injuries reported. Light conditions also play a big role, as accidents in the dark without street lights tend to be more severe. Looking at different types of vehicles involved in accidents, we found some vehicles are more commonly involved than others. These findings help us understand how different factors contribute to traffic accidents and their outcomes.

Part 3: Regression

In this part, we will use a linear regression model to impute the missing values in the "Age of Casualty" column. We will follow the tasks(steps) to prepare the data, train the model, predict the missing values, and finally save the imputed dataset.

Task 1: Convert the required columns to factors

We start by converting the relevant columns to factors, as linear regression modeling in R requires categorical predictors to be factors.

```
# Converting selected columns to factors
data$`Casualty_Class` <- as.factor(data$`Casualty_Class`)
data$`Casualty_Severity` <- as.factor(data$`Casualty_Severity`)
data$`Type_of_Vehicle` <- as.factor(data$`Type_of_Vehicle`)
data$`Weather Conditions` <- as.factor(data$`Weather_Conditions`)
```

Task 2: Create a new data frame containing rows that do not contain missing values in Age_of_Casualty

Next, we filter the dataset to include only the rows where "Age of Casualty" is not missing. This will be our training dataset for building the regression model.

```r
train_data <- data %>%
  filter(!is.na(Age_of_Casualty))
```

Task 3: Build a linear regression model using the specified predictors

We use the 'lm()' function to build a linear regression model with "Age of Casualty" as the dependent variable and the specified predictors.

```r
lm_model <- lm(`Age_of_Casualty` ~ `Casualty_Class` + `Casualty_Severity`
               + `Type_of_Vehicle` + `Weather_Conditions`, data = train_data)

print(summary(lm_model))
```

```
##
## Call:
## lm(formula = Age_of_Casualty ~ Casualty_Class + Casualty_Severity +
##     Type_of_Vehicle + Weather_Conditions, data = train_data)
##
## Residuals:
##    Min      1Q Median      3Q     Max
## -40.77 -14.53  -3.71  11.29  72.92
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 39.6040     4.1108   9.634  < 2e-16 ***
## Casualty_Class2             -9.9909     1.0987  -9.094  < 2e-16 ***
## Casualty_Class3             -5.6159     1.1764  -4.774 1.94e-06 ***
## Casualty_SeveritySerious    -0.5411     3.9633  -0.137 0.891415
## Casualty_SeveritySlight     -3.4584     3.8437  -0.900 0.368365
## Type_of_Vehicle2           -11.5422     4.4322  -2.604 0.009278 **
## Type_of_Vehicle3            -7.9884     2.8407  -2.812 0.004969 **
## Type_of_Vehicle4             1.8405     4.4458   0.414 0.678930
## Type_of_Vehicle5             6.5325     2.9596   2.207 0.027411 *
## Type_of_Vehicle8             6.2213     2.7468   2.265 0.023624 *
## Type_of_Vehicle9             4.1099     1.7257   2.382 0.017331 *
## Type_of_Vehicle10            9.1440    11.0924   0.824 0.409841
## Type_of_Vehicle11         15.8725     3.6096   4.397 1.15e-05 ***
## Type_of_Vehicle16         -4.5999    13.5000  -0.341 0.733339
## Type_of_Vehicle17        -11.5999    13.5000  -0.859 0.390304
## Type_of_Vehicle18         -6.9393    11.0864  -0.626 0.531429
## Type_of_Vehicle19          4.7142     2.9035   1.624 0.104611
## Type_of_Vehicle20         21.0015     6.9138   3.038 0.002415 **
## Type_of_Vehicle21         15.1445     4.4583   3.397 0.000695 ***
## Type_of_Vehicle22         14.4001    19.0233   0.757 0.449157
## Type_of_Vehicle23        -19.5999    19.0233  -1.030 0.302988
## Type_of_Vehicle90         -0.3492     6.5449  -0.053 0.957457
## Type_of_Vehicle97         45.0159    19.0618   2.362 0.018292 *
## Weather_Conditions        -0.5457     0.3150  -1.733 0.083313 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.95 on 2010 degrees of freedom
## Multiple R-squared:  0.0721, Adjusted R-squared:  0.06148
## F-statistic:  6.79 on 23 and 2010 DF,  p-value: < 2.2e-16
```

Task 4: Create a new data frame with rows containing missing values in Age__of__Casualty

We then create a new data frame that includes only the rows where "Age of Casualty" is missing. This dataset will be used for predicting and imputing the missing values.

```r
missing_data <- data %>%
  filter(is.na(Age_of_Casualty))
View(missing_data)
```

Task 5: Predict the missing values using the linear regression model and impute them

Using the trained regression model, we predict the missing values in "Age of Casualty" and impute these values back into the original dataset.

```r
# Ensure that Casualty_Class and Type_of_Vehicle in missing_data are converted to factors with the same
missing_data$Casualty_Class <- factor(missing_data$Casualty_Class, levels = levels(train_data$Casualty_
missing_data$Type_of_Vehicle <- factor(missing_data$Type_of_Vehicle, levels = levels(train_data$Type_of

predicted_age <- predict(lm_model, newdata = missing_data)

data$Age_of_Casualty[is.na(data$Age_of_Casualty)] <- predicted_age
```

```r
missing_data <- data %>%
  filter(is.na(Age_of_Casualty))
```

Task 6: Round the imputed values and convert the data type to integer

The predicted values might be in decimals, so we round them to the nearest integer and change the data type to integer.

```r
data$Age_of_Casualty <- round(data$Age_of_Casualty)

data$Age_of_Casualty <- as.integer(data$Age_of_Casualty)
View(data)
```

Task 7: Check if there are any missing values remaining

We verify that there are no missing values left in the dataset.

```r
missing_values <- colSums(is.na(data))
print("Columns with Missing Values:")
```

```
## [1] "Columns with Missing Values:"
```

```r
print(missing_values[missing_values > 0])
```

```
## named numeric(0)
```

Task 8: Save the imputed data to a CSV file

Finally, we save the dataset with the imputed values to a CSV file named “regression.csv”.

```r
write.csv(data, "regression.csv", row.names = FALSE)
```

Question: Discuss in your report what problems you have found while imputing the missing values, and how you deal with them.

Answer: During the imputation process of “Age of Casualty,” several issues were identified and addressed:

1. Missing Predictor Values:

a. Problem: The regression model cannot handle missing values in predictors.

b. Solution: Filtered out rows with missing values in “Casualty Class,” “Casualty Severity,” “Type of Vehicle,” and “Weather Conditions” before training.

2. Decimal Predicted Values:

a. Problem: The predicted ages were in decimals.

b. Solution: Rounded the predicted values to the nearest integer to ensure realistic age data.

3. Overfitting and Model Performance:

a. Problem: The model might overfit, performing well on training data but poorly on new data.

b. Solution: Evaluated the model’s performance using summary statistics (e.g., R-squared) to ensure a balance between model complexity and predictive power.

These steps ensured accurate and reliable imputation of missing “Age of Casualty” values.

Summary of Part 3: Regression

In this part, we successfully trained a linear regression model to impute the missing values in the “Age of Casualty” column. The model used the predictors “Casualty Class,” “Casualty Severity,” “Type of Vehicle,” and “Weather Conditions.” After predicting and imputing the missing values, we ensured the data integrity by rounding off the imputed values and converting them to integers. The final dataset, with no missing values, was saved as “regres-