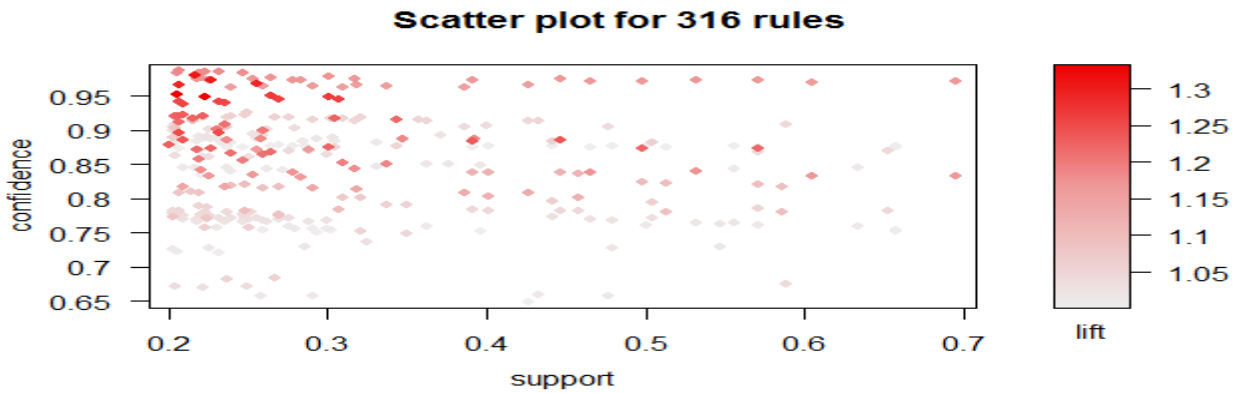
Arman Yousefzade Shoushtari 92222131 armanyousefzade95@gmail.com

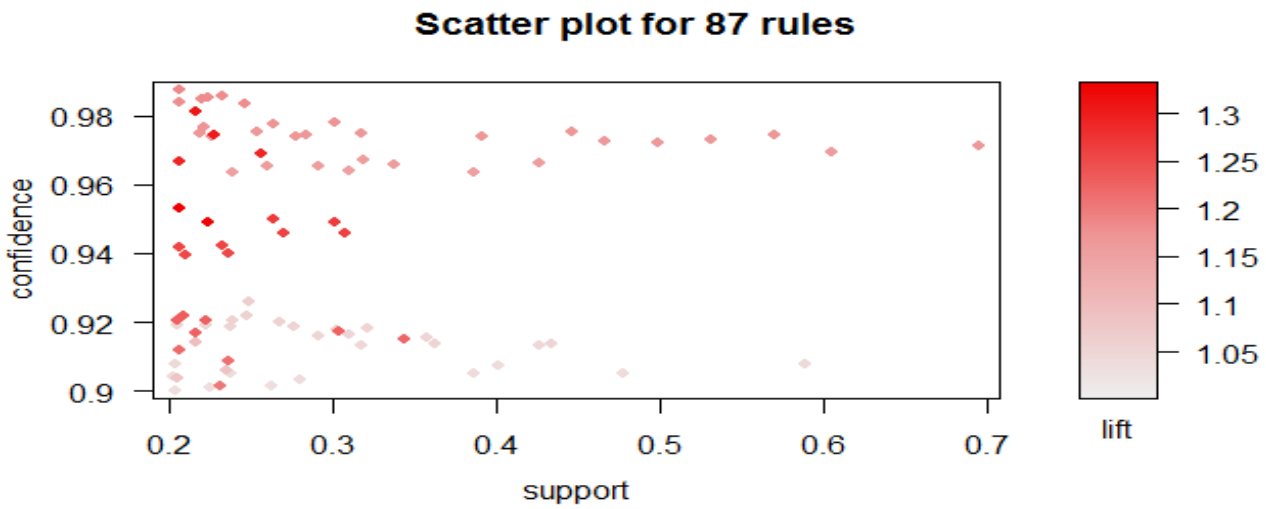
England car accidents (2015)

Abstract

The project took me about **60** hours. The main tools I used for this project includes R and java programming languages. The diagrams were drawn by R and also some measures were computed by R like mean, median and variance and etc. The diagrams include histogram, box plot, pie chart, scatter plot. Data table has 20 columns and 2664 rows. The data is collected from: https://data.gov.uk/dataset/road-traffic-accidents since the primary data was not complete enough for data mining and it needed some further columns, so a java program was written to add five further columns such as:
Day of week column: this column was added according to date of every accident.
Holiday or weekend column: was added according to date and calendar of year 2015.
Season column: was added according to date of every accident.
Time (descriptive): was added according to time of every accident, since there was just numeric time of every accident in primary data so I found is useful to add this column for example If the time of accident is 14:30 program will put afternoon in this column.
Age of casualty (nominal): was added to data according to ages of casualties.
The java program reads every row and according to date of the accidents determines the day of week that accident occurred in and season of accident and also it determines that accident occurred on a holiday or a business day. The java program also produce a time column that show nominal time of every accident. I also add a column that show nominal age of casualties. For getting useful information and knowledge from the data I used apriori algorithms in R, this algorithm first produce frequent sets and according to these frequent sets produces association rules with suitable support, confidence and lift values. I run apriori algorithm on my data with support greater than 0.1 and confidence greater than 0.6 and lift greater than 1 the diagram shows the results in below:



I also run the apriori algorithm with support greater than 0.2 and confidence greater than 0.9 and lift greater than 1 the diagram shows the result facts:



I will send both files of association rules for you with title of associationrules1.txt and associationrules2.txt. For example some of the rules are shown in below:

{Road Surface=Dry, Age of casualty (nominal) =young} **=>** {Casualty Severity=Slight}"

Support: 0.259 **confidence: 0.892** **lift: 1.022**

{Lighting Conditions=Daylight: street lights present, Weather Conditions=Fine without high winds, Type of Vehicle=Car} => {Casualty Severity=Slight}"

Support: 0.362 **confidence: 0.913** **lift: 1.046**

{Weather Conditions=Fine without high wind, Season=summer} => {RoadSurface=Dry}"

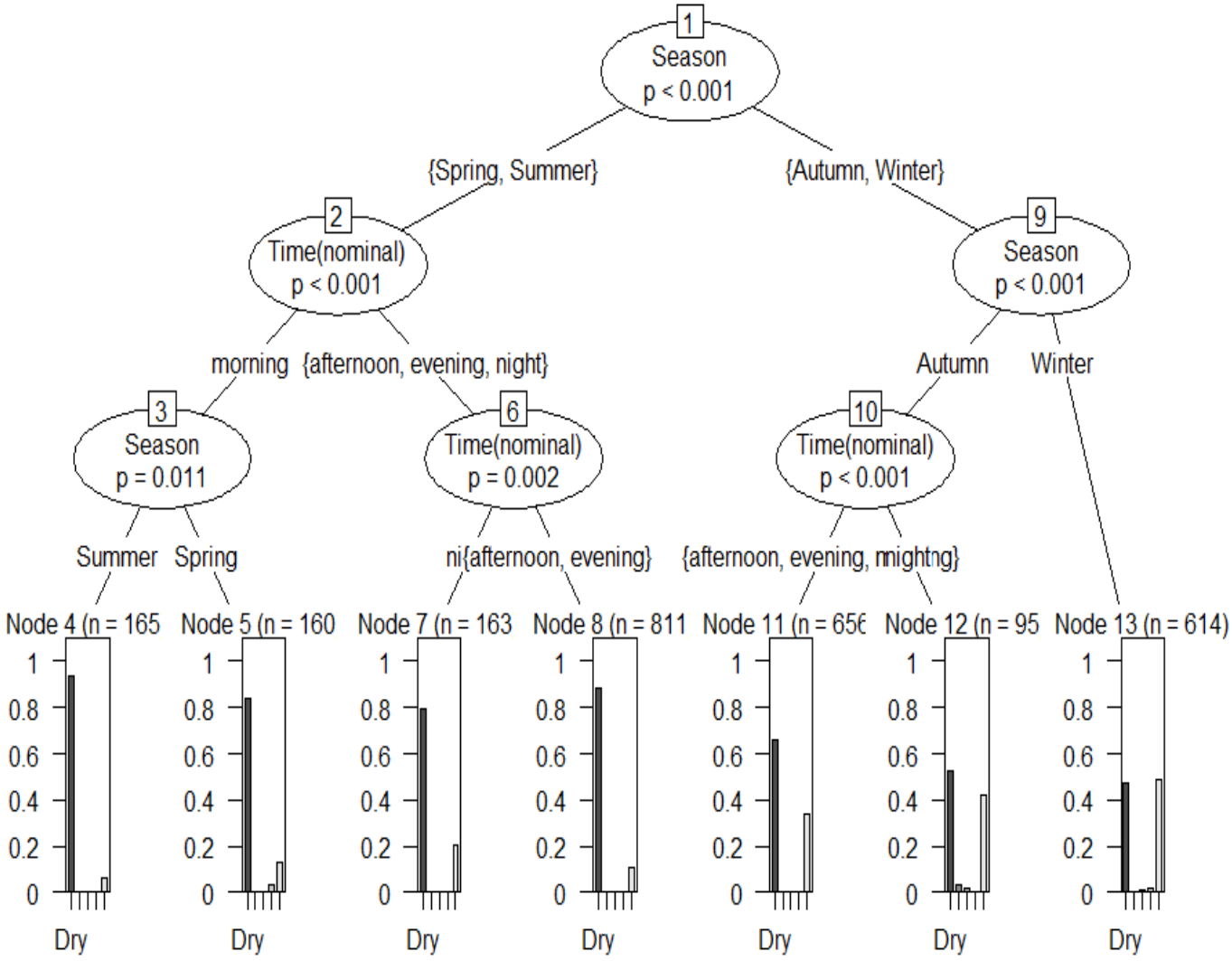
Support: 0.222 **confidence: 0.948** **lift: 1.324**

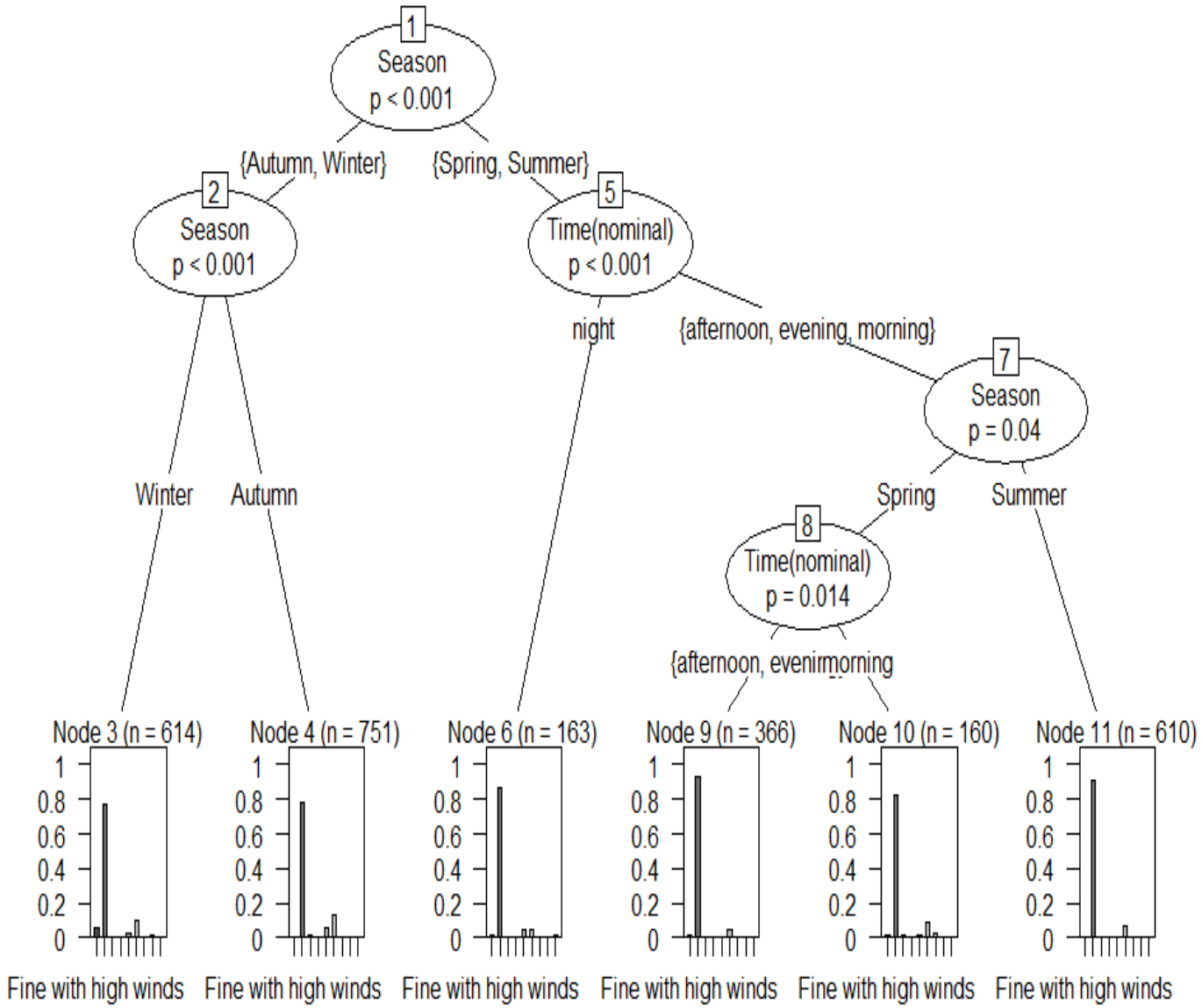
{Road Surface=Dry, Weather Conditions=Fine without high wind, Time (nominal) = afternoon} => {Lighting Conditions=Daylight: street lights present}"

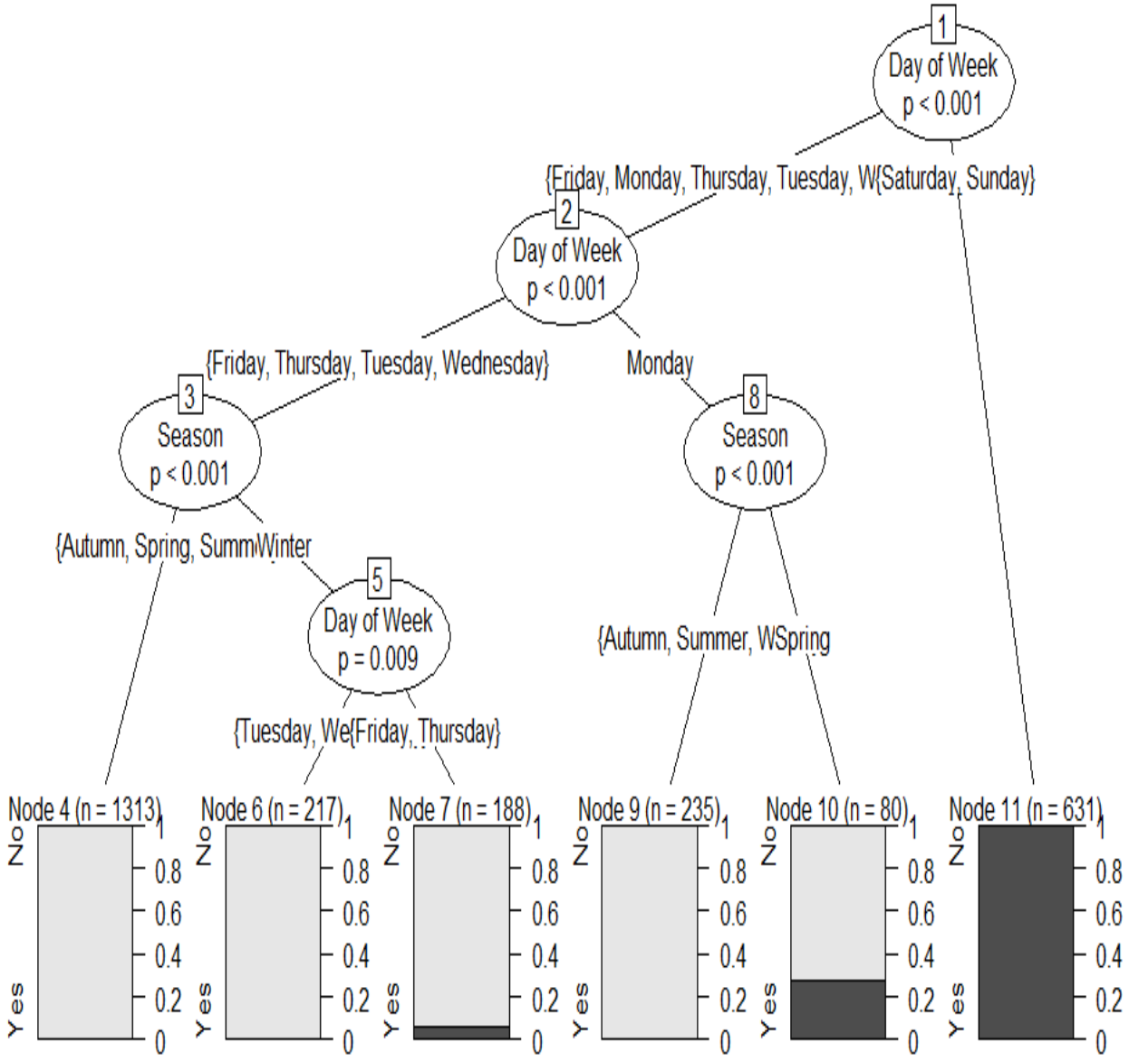
Support: 0.300 **confidence: 0.949** **lift: 1.264**

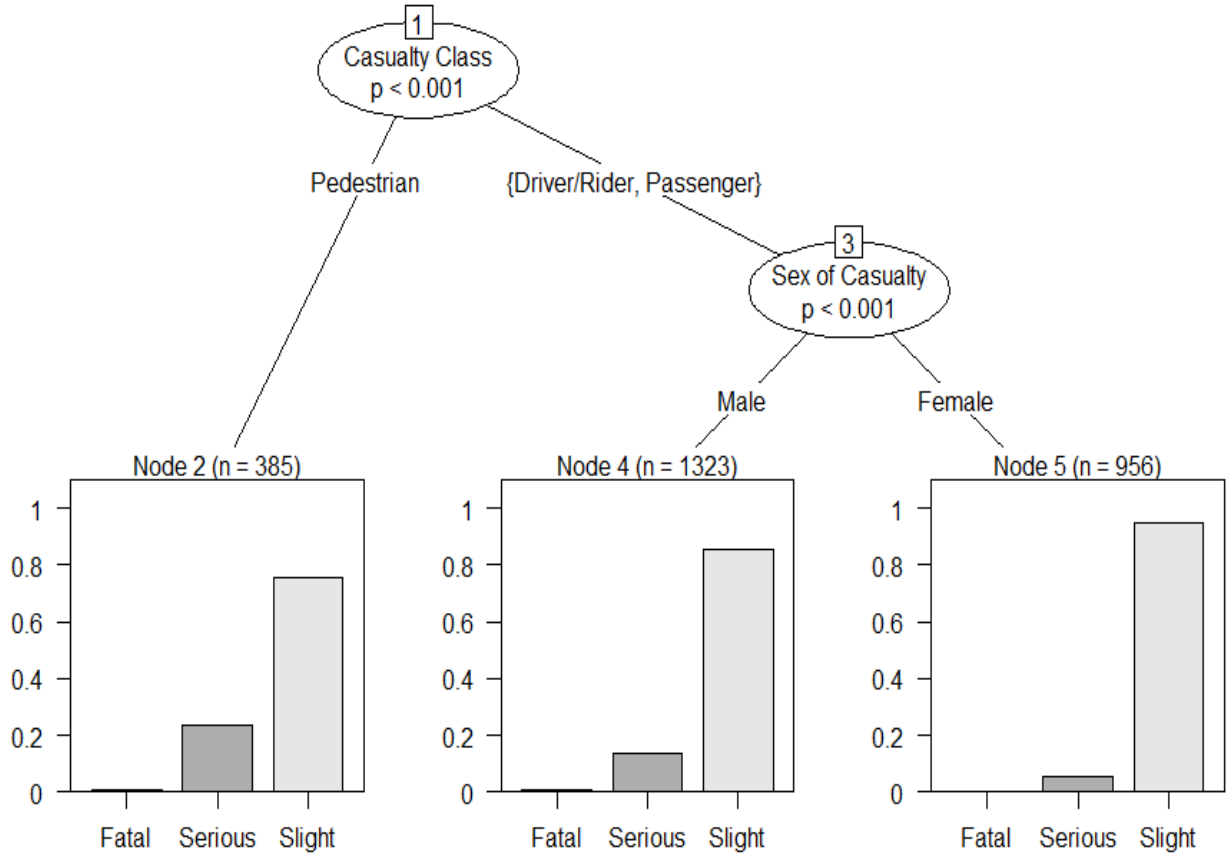
Decision tree

By **R** some decision trees were drawn:




1
Season
p < 0.001
{Autumn, Winter}
{Spring, Summer}
2
Season
p < 0.001
5
Time(nominal)
p < 0.001
Winter
Autumn
night
{afternoon, evening, morning}
7
Season
p = 0.04
Spring
Summer
8
Time(nominal)
p = 0.014
{afternoon, evenirmorning
Node 3 (n = 614)
Node 4 (n = 751)
Node 6 (n = 163)
Node 9 (n = 366)
Node 10 (n = 160)
Node 11 (n = 610)
1
0.8
0.6
0.4
0.2
0
Fine with high winds
Fine with high winds
Fine with high winds
Fine with high winds
Fine with high winds
Fine with high winds


1
Day of Week
p < 0.001
{Friday, Monday, Thursday, Tuesday, W
{Saturday, Sunday}
2
Day of Week
p < 0.001
{Friday, Thursday, Tuesday, Wednesday}
Monday
3
Season
p < 0.001
8
Season
p < 0.001
{Autumn, Spring, Summ
Winter
5
Day of Week
p = 0.009
{Autumn, Summer, W
Spring
{Tuesday, We
{Friday, Thursday}
Node 4 (n = 1313)
Node 6 (n = 217)
Node 7 (n = 188)
Node 9 (n = 235)
Node 10 (n = 80)
Node 11 (n = 631)
No
Yes
1
0.8
0.6
0.4
0.2
0



Facts and knowledge

A lot of useful facts and knowledge can be extracted from the data of England car accidents for example: Effects of age on accidents in order to know whether young drivers have more accidents than older ones or not. Effect of vehicles number on casualties' severity. Percept of accidents which include more than 2 vehicles. We can answer below questions according to knowledge we will extract from the data:

What time of day has more accidents?
Does lighting condition have any effects on accidents?

How we can reduce number of accidents?

What kinds of roads have more accidents?

Can motorways reduce the accidents?

Which weather condition has more accident?

Which weather condition is safer for driving?

Which day of weak is safer for driving?

Is driving during night safer or during daylight?

Is driving on holidays more dangerous than driving on business days?

Are women drivers more careful than men?

Is winter the most dangerous season for driving or not?

Which kinds of vehicles are safer for driving?

Does road surface have any effects on accidents?

Does old people drive more carefully than young people?

Should we increase the allowed age for getting driver license?

And etc.

The knowledge that we extract from the data is very useful for reducing the accidents that happen all over the world every year therefore, we can save a lot of money and we can save a lot of people that every year are killed or injured in accidents. For example according the data a lot of accidents happen in rainy weather so police can make more speed limitations or other strict regulations for driving in these special and dangerous conditions. In future I can collect more data about accidents from all over the word. Also I can use more algorithms of classification, these algorithms will be very helpful for getting more valuable information and knowledge.

Meta data:

Reference Number: nominal (missing percent=0)
Grid Ref: Easting: nominal (missing percent=0)
Grid Ref: Northing: nominal (missing percent=0)
Number of Vehicles in accident: discrete numeric(missing percent=0)
Accident Date: nominal (missing percent=0)
Accident Time (24hr): nominal (missing percent=0)
Accident Time (descriptive): nominal (missing percent=0)
Accident Road Class: nominal (missing percent=51)
Accident Road Surface: nominal (missing percent=0)
Lighting Conditions: nominal (missing percent=0)
Weather Conditions: nominal (missing percent=0)

Casualty Class: nominal (missing percent=0)
Casualty Severity: nominal (missing percent=0)
Sex of Casualty: symmetric Boolean (missing percent=0)
Age of Casualty: discrete numeric(missing percent=0)
Type of Vehicle: nominal (missing percent=0)
Day of week: nominal (missing percent=0)
Holiday or weekend: Asymmetric Boolean(missing percent=0)
Season of accident: nominal (missing percent=0)
Age of casualty(nominal): nominal (missing percent=0)

Accident Road Class column has 51% missing data and I put unclassified for these missing rows and in data mining process I call these missing values unclassified.

The **pseudo** code of java program that was written for adding 5 further columns: (the program is attached with file)

```
Void Addfurhercolumns (String Readingpath, String WritingPath) {
String line;
FileWriter fw = new FileWriter (WritingPath);
BufferedWriter BW = new BufferedWriter (fw);
File Reader Fr = new FileReader (Readingpath);
BufferedReader br = new BufferedReader (Fr);
Line = br.readLine ();
line = line + ", Day of Week" + ", HolidayOrWeekend"+", Time (nominal)"+", Season";
bw.write (line);
bw.newLine ();
String dateInput = "";
String date;
String time;
String season="summer";
String [] lineArray;
String [] dateArray;
For (int i = 2; i <= 2665; i++) {
```

```
Line = br.readLine ();
Line Array = line.split (",");
Date = lineArray [4];
Time=lineArray [5];
DateArray = date.split ("-");
Switch (dateArray [1]) {
   Case "Jan":
      Season="winter";
      DateInput = dateArray [0] + "/" + "1" + "/" + "2015";
      Break;
   Case "Feb":
       Season="winter";
      DateInput = dateArray [0] + "/" + "2" + "/" + "2015";
      Break;
   Case "Mar":
       Season="spring";
      DateInput = dateArray [0] + "/" + "3" + "/" + "2015";
      Break;
   Case "Apr":
      Season="spring";
      DateInput = dateArray [0] + "/" + "4" + "/" + "2015";
      Break;
   Case "May":
      Season="spring";
      DateInput = dateArray [0] + "/" + "5" + "/" + "2015";
      Break;
   Case "Jun":
      Season="summer";
      DateInput = dateArray [0] + "/" + "6" + "/" + "2015";
      Break;
   Case "Jul":
```

```
        Season="summer";
        DateInput = dateArray [0] + "/" + "7" + "/" + "2015";
        Break;
    Case "Aug":
        Season="summer";
        DateInput = dateArray [0] + "/" + "8" + "/" + "2015";
        Break;
    Case "Sep":
        Season="autumn";
        DateInput = dateArray [0] + "/" + "9" + "/" + "2015";
        Break;
    Case "Oct":
        Season="autumn";
        DateInput = dateArray [0] + "/" + "10" + "/" + "2015";
        Break;
    Case "Nov":
        Season="autumn";
        DateInput = dateArray [0] + "/" + "11" + "/" + "2015";
        Break;
    Case "Dec":
         Season="winter";
        DateInput = dateArray [0] + "/" + "12" + "/" + "2015";
        Break;
}
SimpleDateFormat format1 = new SimpleDateFormat ("dd/MM/yyyy");
Date dt1 = format1.parse (dateInput);
DateFormat format2 = new SimpleDateFormat ("EEEE");
String finalDay = format2.format (dt1);
If (IsHolidayOrWeekend (finalDay, dateInput)) {
    line = line + "," + finalDay + ", Yes";
} else {
```

```
            line = line + "," + finalDay + ", No";
        }
        Line=line+ConvertNumericTimeToNominalTime (time) +","+season;
        bw.write (line);
        bw.newLine ();
    }
    BW. Close ();
    br.close ();
    fr.close ();
    fw.close ();
}
```

Boolean IsHolidayOrWeekend (String dayOfWeek, String date) {

```
    If (dayOfWeek.equals ("Saturday") || dayOfWeek.equals ("Sunday")) {
        Return true;
    }
    Switch (date) {
        Case "1/1/2015":
            Return true;
        Case "3/4/2015":
            Return true;
        Case "6/4/2015":
            Return true;
        Case "4/5/2015":
            Return true;
        Case "25/5/2015":
            Return true;
        Case "31/8/2015":
            Return true;
        Case "25/12/2015":
```

```
            Return true;
        Case "26/12/2015":
            Return true;
    }
    Return false;
  }
  Public static String ConvertNumericTimeToNominalTime (String time) {
    Int time2=Integer.valueOf (time);
    If (600<=time2&&time2<1200) {
      Return ", morning";
    }
    If (1200<=time2&&time2<1800) {
      Return ", afternoon";
    }
    If (1800<=time2&&time2<2100) {
      Return ", evening";
    }
    If (2100<=time2&&time2<=2359) {
      Return ", night";
    }
    If (0<=time2&&time2<=600) {
      Return ", night";
    }
    Return ", night";
  }
}
Public static String getNominalAge (String Age) {
    Int age = Integer.valueOf (Age);
    If (age < 11) {
      Return "child";
```

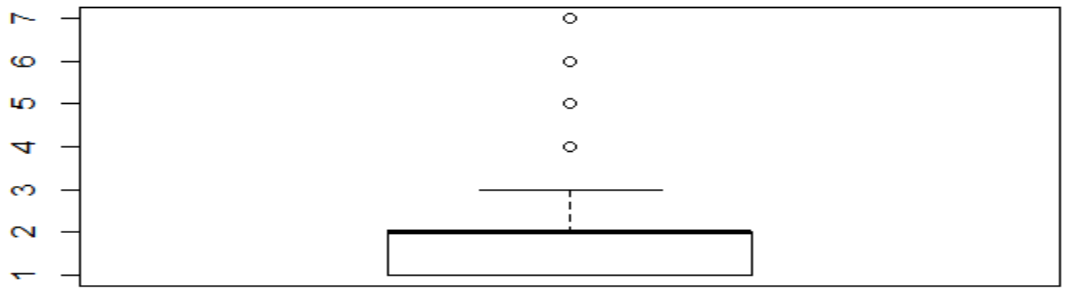
```
    }
    If (11 <= age && age <= 18) {
       Return "teenager";
    }
    If (18 < age && age <= 35) {
       Return "young";
    }
    If (35 < age && age < 60) {
       Return "middle age";
    }
    If (age >= 60) {
       Return "old";
    }
    Return "young";
}
```

Statistics and diagrams:

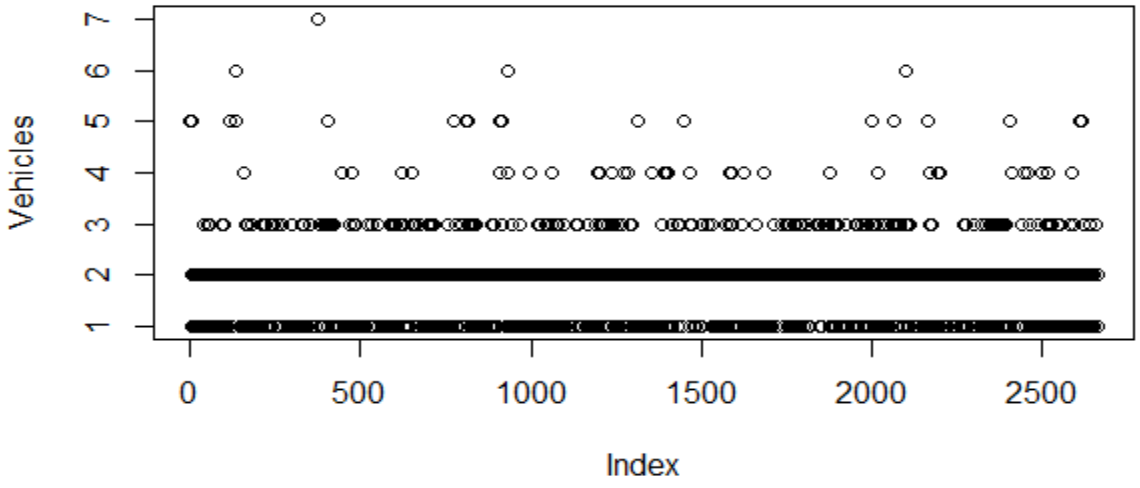
Number of Vehicles

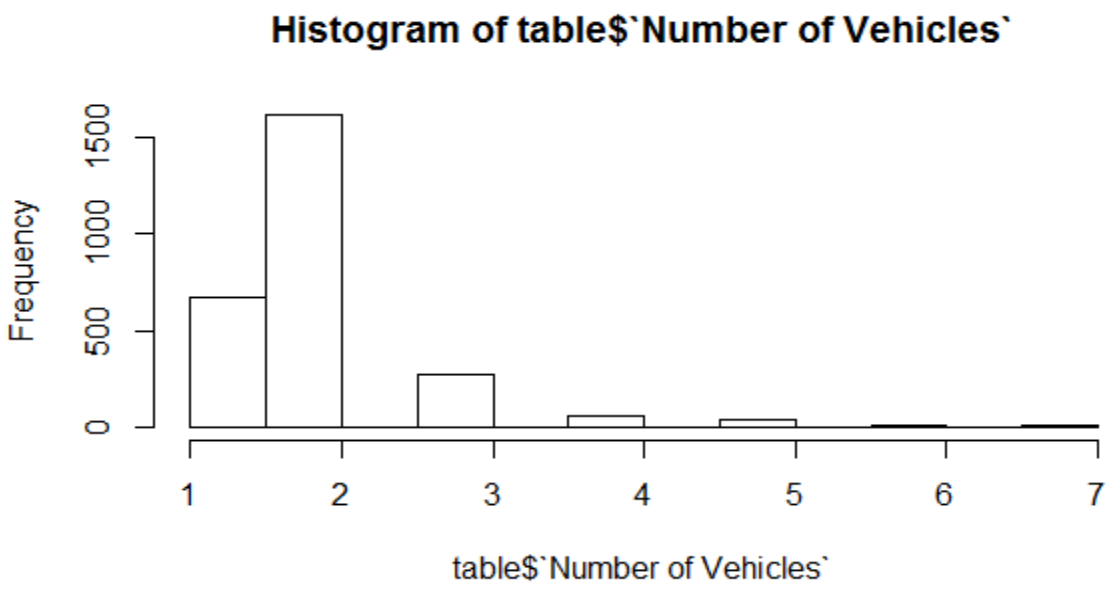
Mean of number of vehicles: 1.952703
Median of number of vehicles: 2
Variance of number of vehicles: 0.6451523
Standard deviation of number of vehicles: 0.8032137
Five numbers of number of vehicles:
minimum=1 First Quartile=1 SecondQuartile=2 ThirdQuartile=2 maximum=7

Box plot (Number of Vehicles):



Scatter plot (Number of Vehicles):

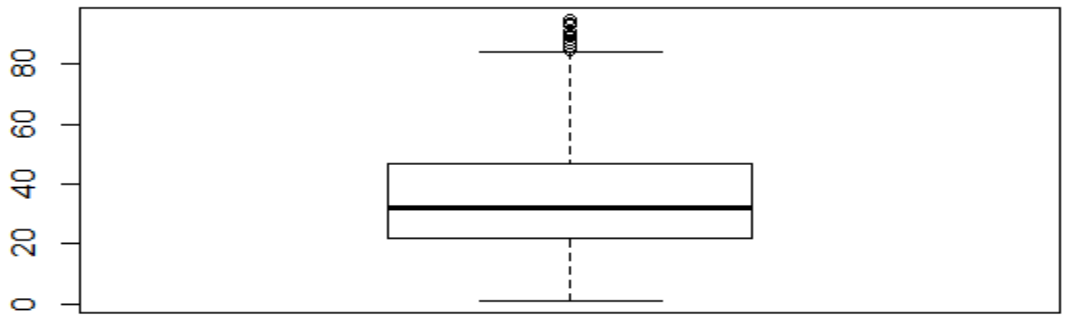




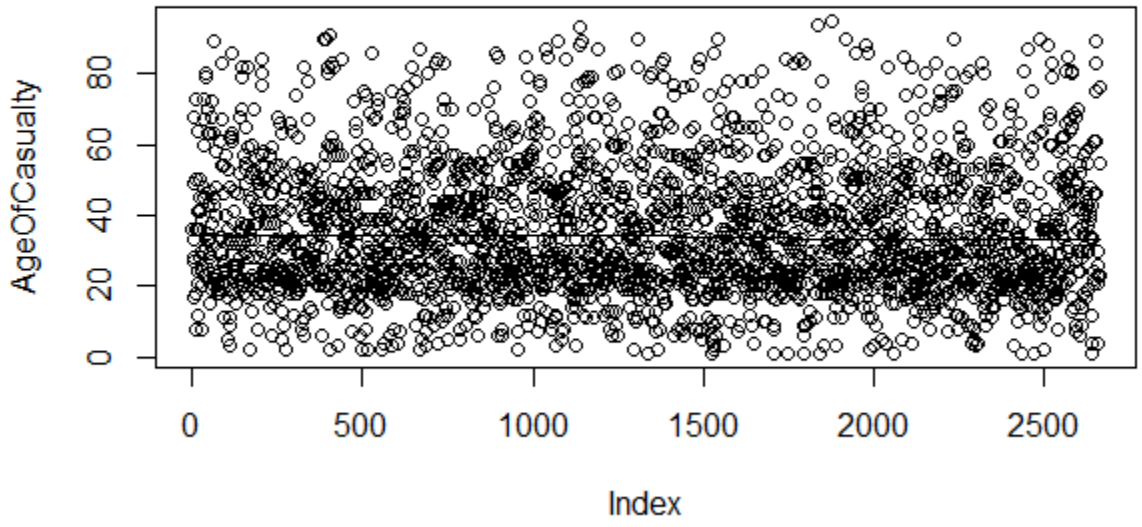
Age of Casualty

Mean of Age of Casualty: 35.73311
Median of Age of Casualty: 32
Variance of Age of Casualty: 347.9937
Standard deviation of Age of Casualty: 18.65459
Five numbers of Age of Casualty:
minimum=1 First Quartile=22 SecondQuartile=32 ThirdQuartile=47 maximum=95

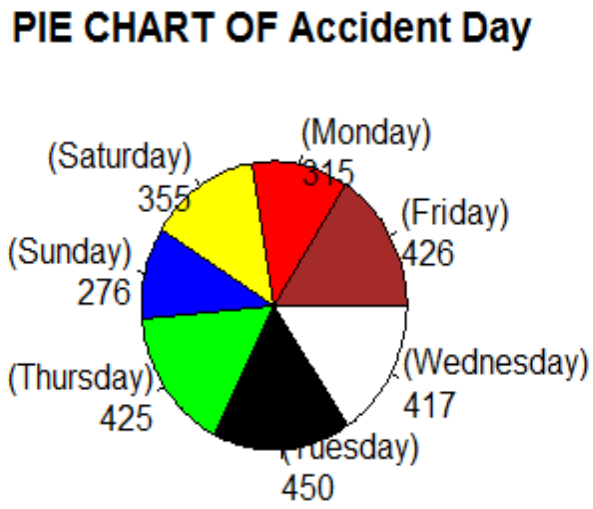
Box plot (Age of Casualty):

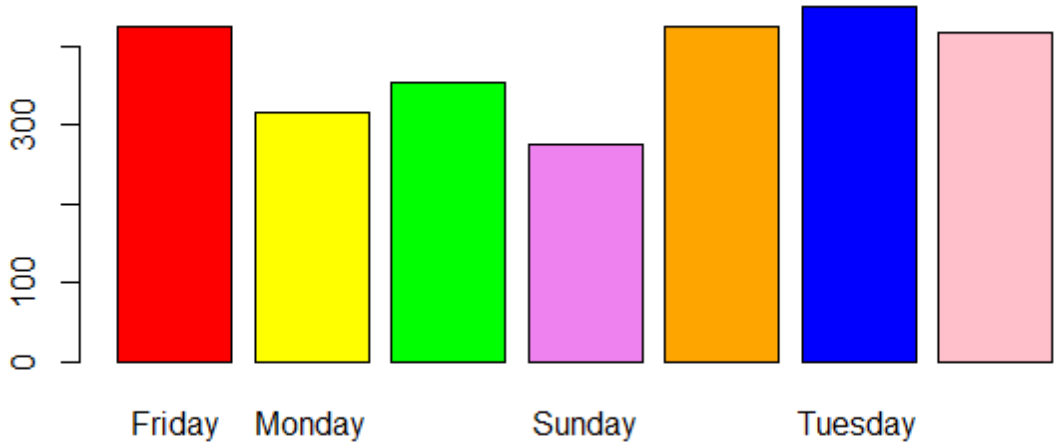


Scatter plot (Number of Vehicles):

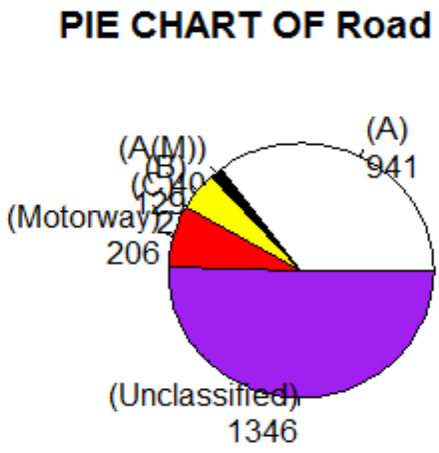


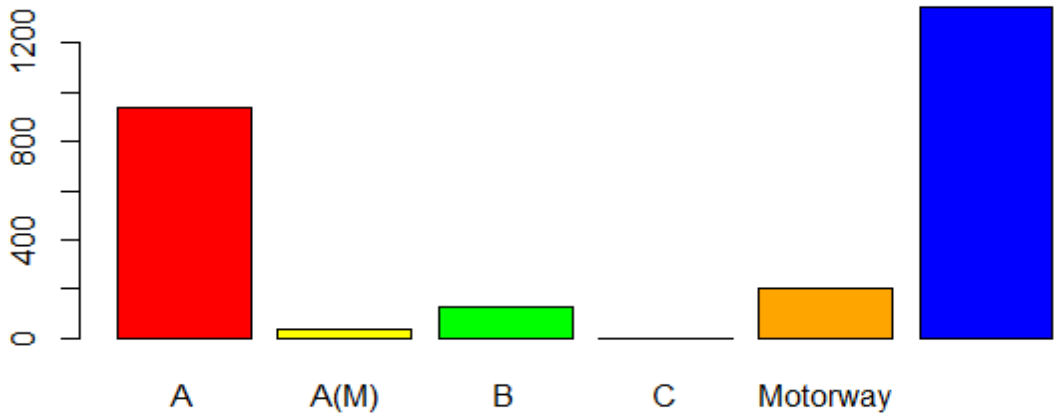
Accident Day(mode=Tuesday)



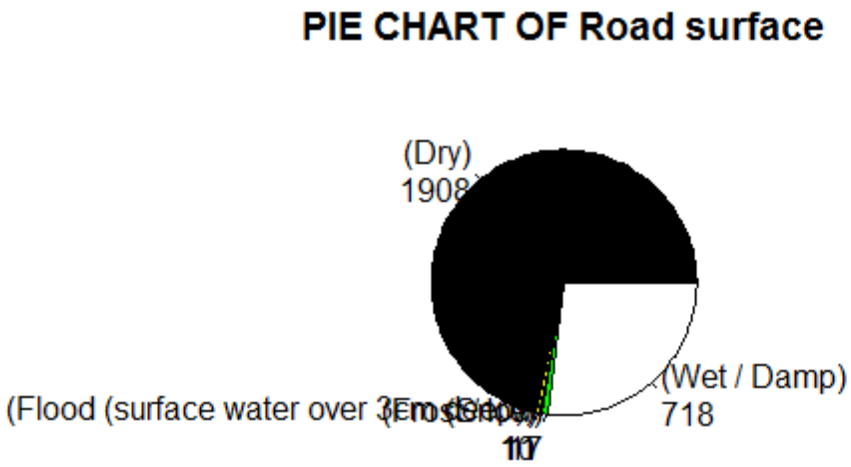


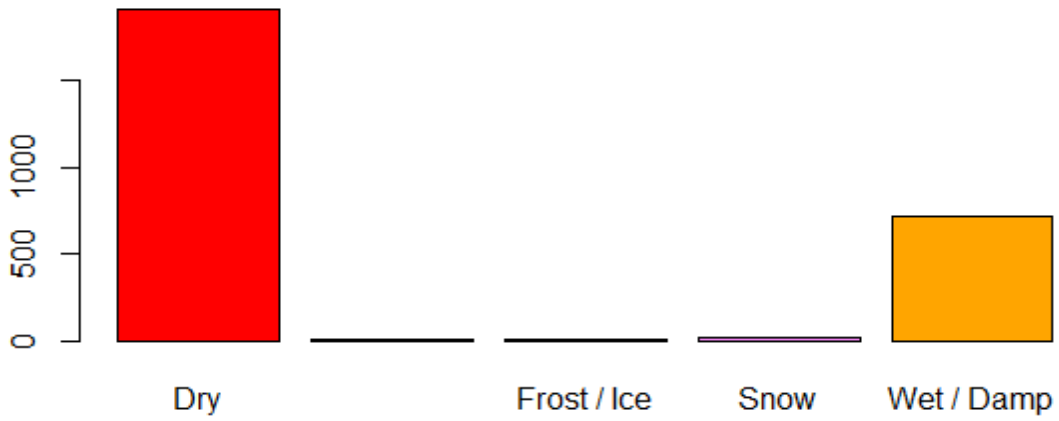
Road class (mode=Unclassified)



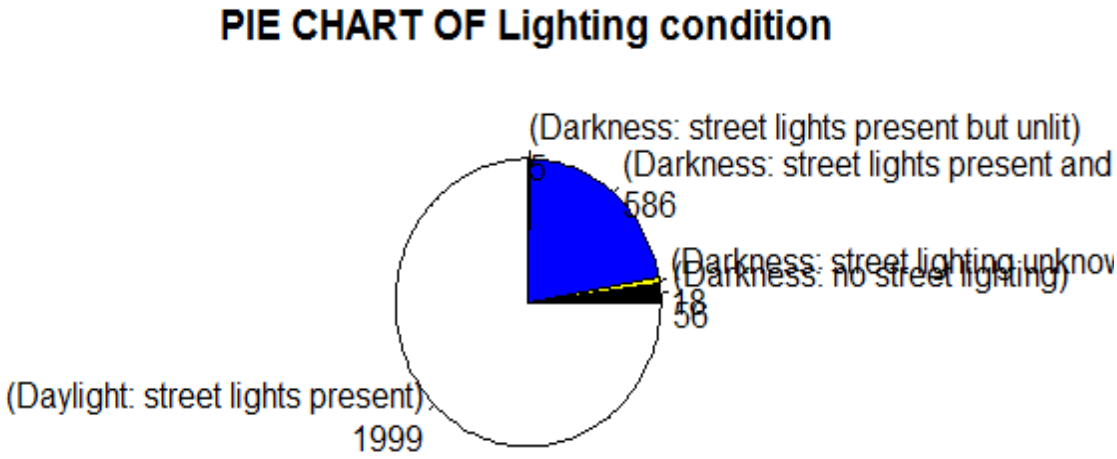


Road surface (mode=Dry)

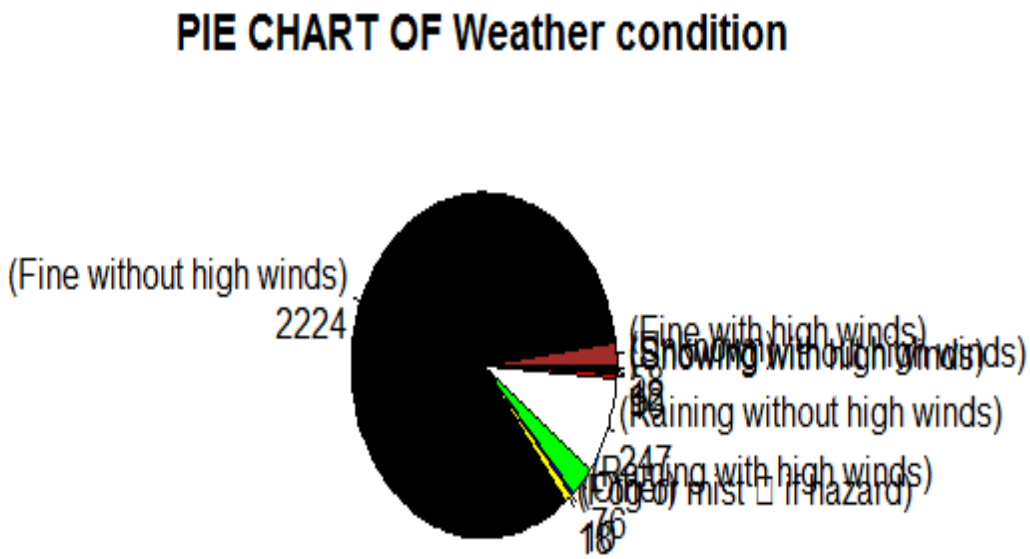


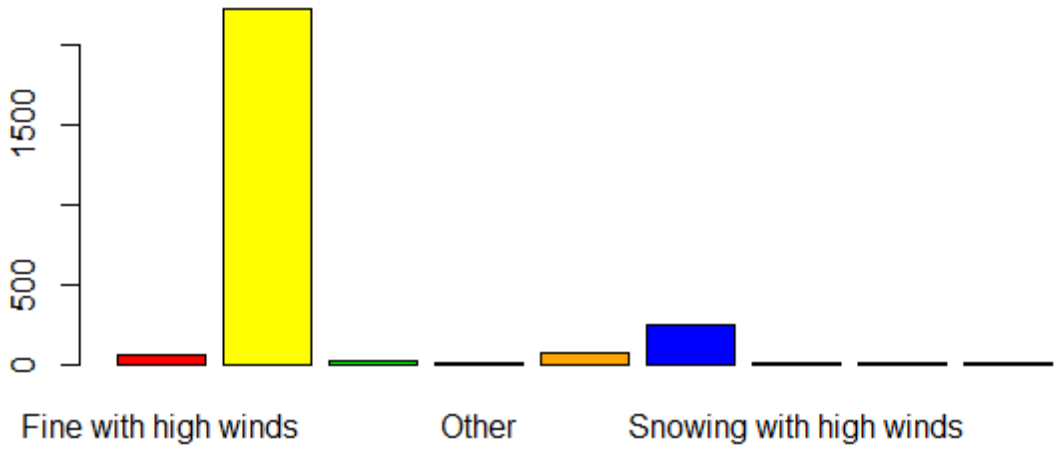


Lighting condition (mode=Daylight)

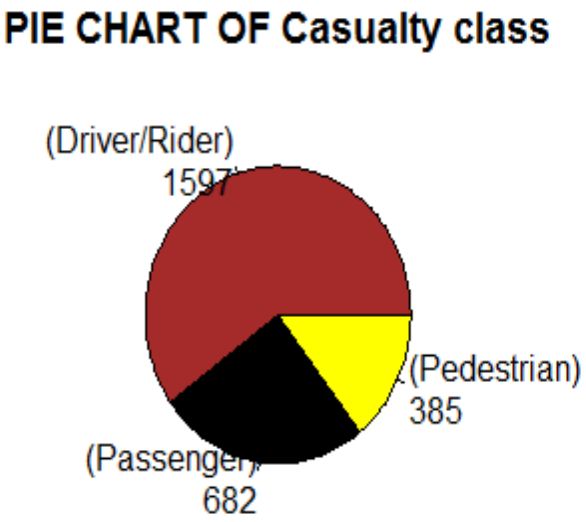


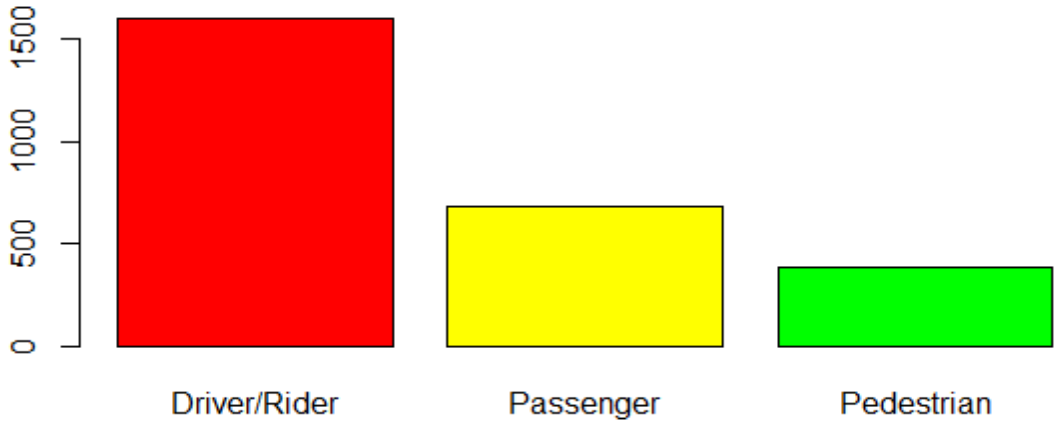

1500
500
0
Darkness: no street lighting
Darkness: street lights present but unlit


PIE CHART OF Weather condition
(Fine without high winds)
2224
(Fine with high winds)
(Raining without high winds)
247

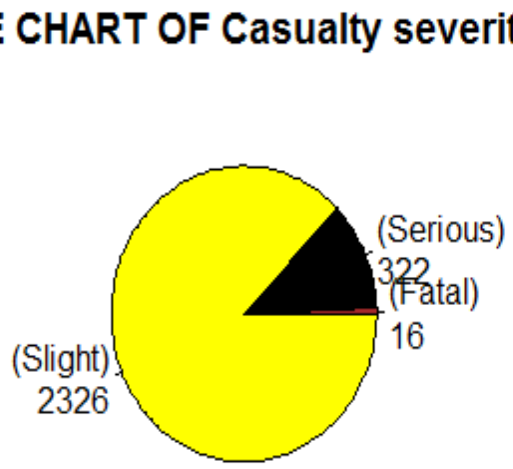


Casualty class(mode=Driver/Rider)

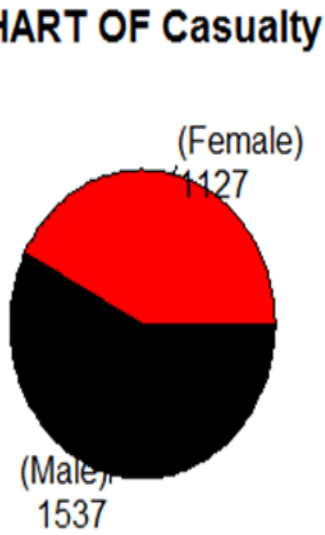


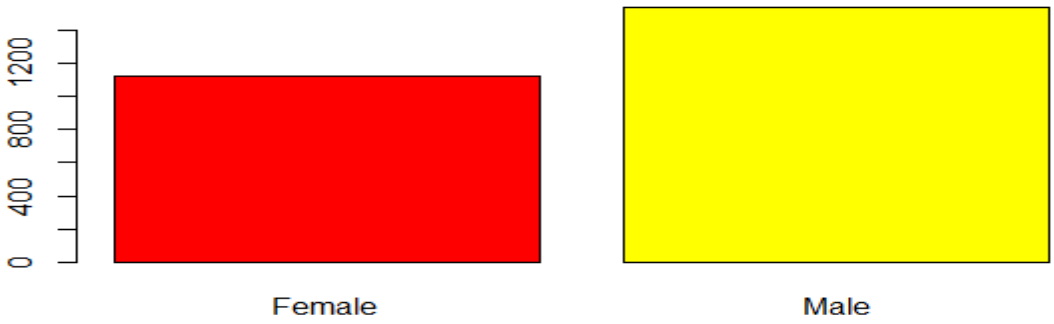


Casualty severity (mode=slight)

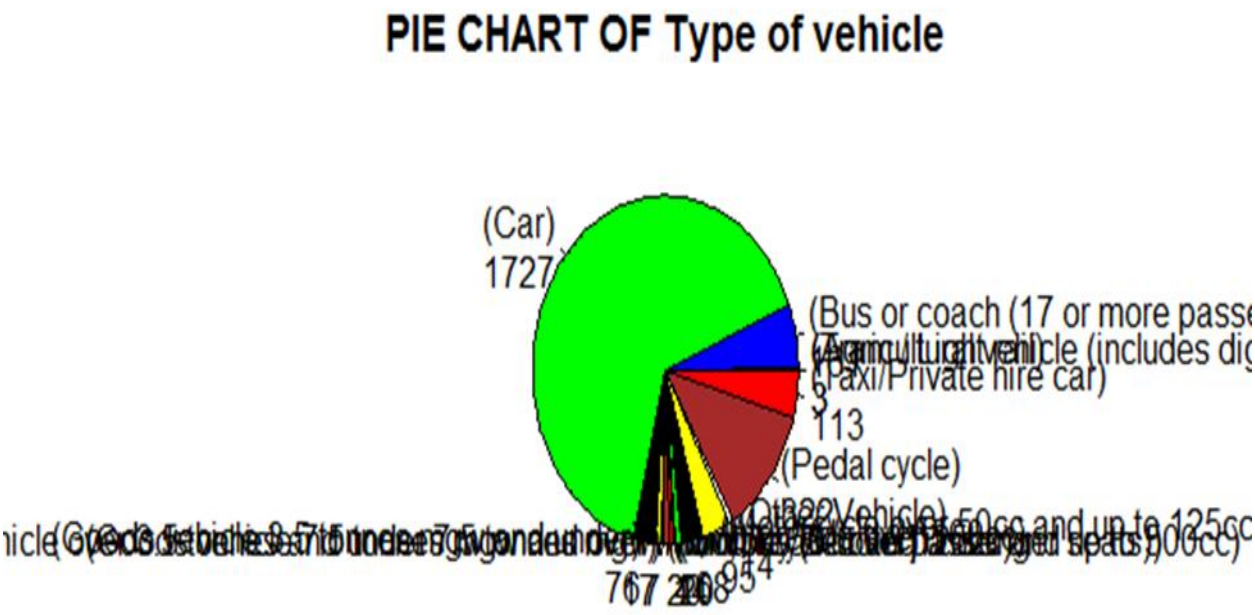


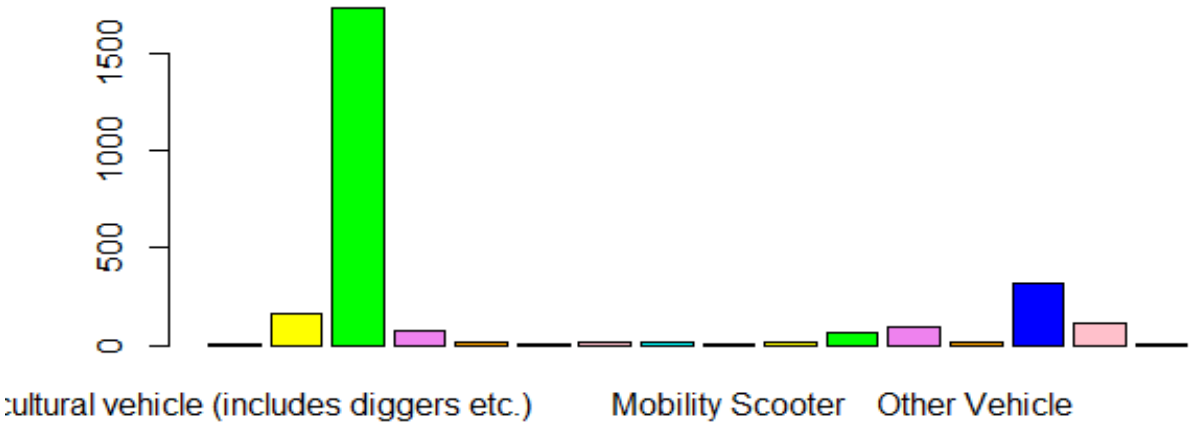
Casualty sex (mode=Female)



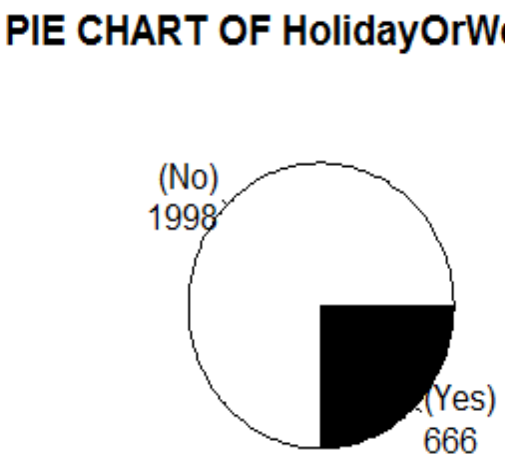


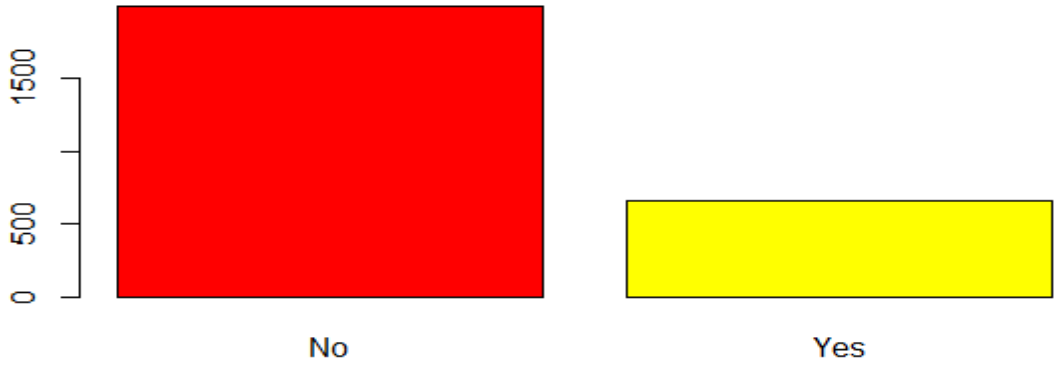
Type of vehicles (mode=car)



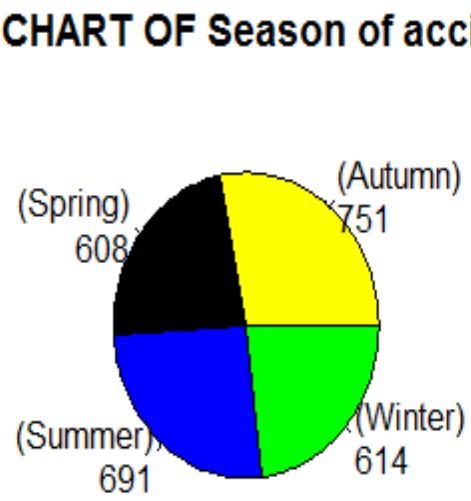


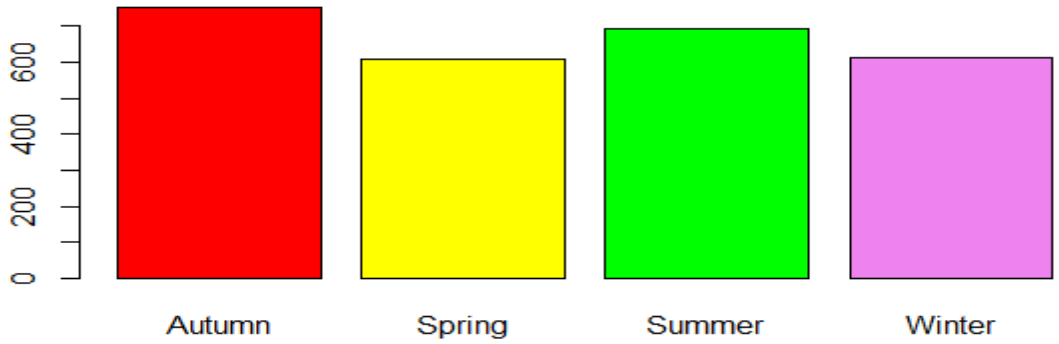
HolidayOrWeekend (mode=no)



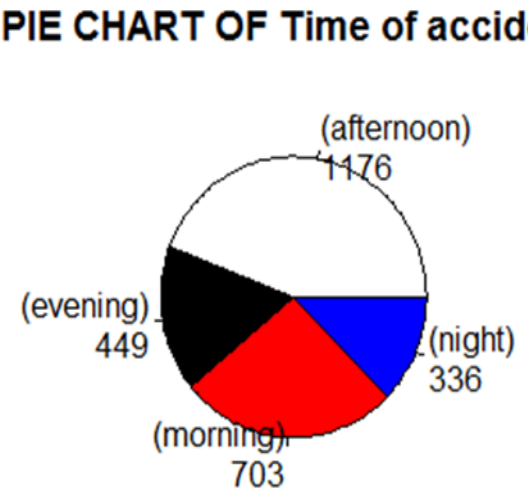


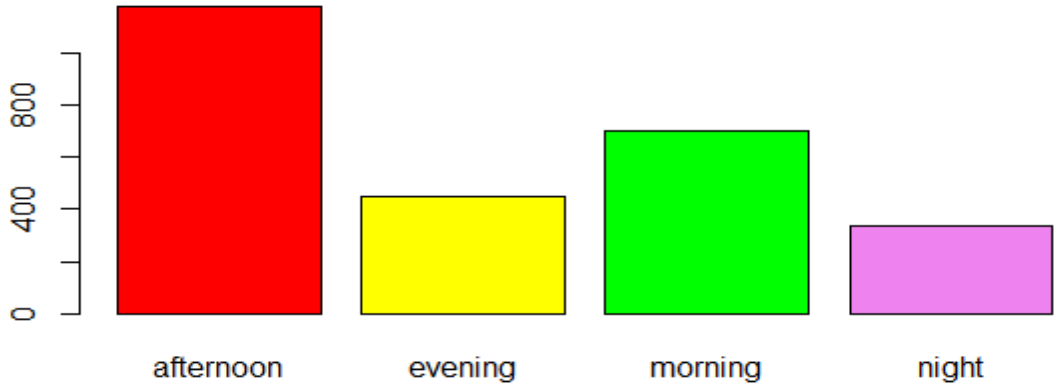
Season of accident(mode=Autumn)



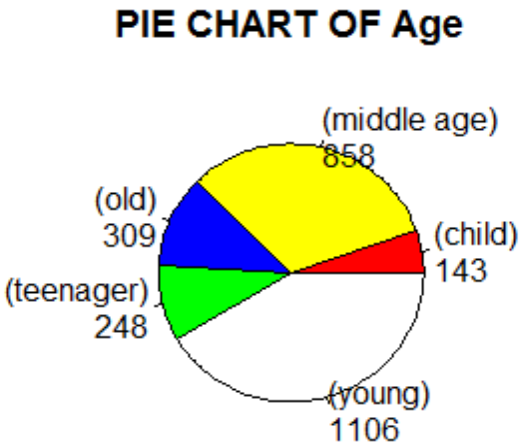


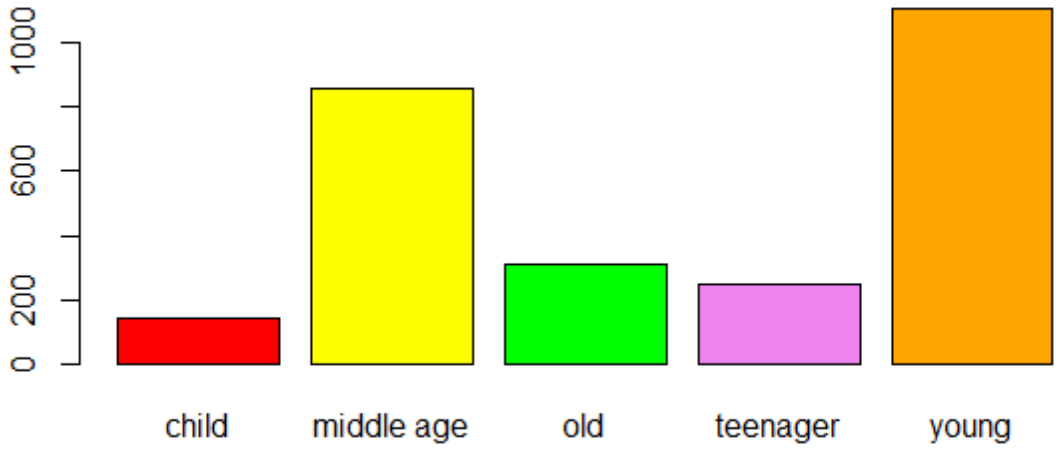
Time of accident (mode=afternoon)





Age of casualty (nominal, mode=young)





Distance matrix

A java program was written for calculating distance matrix, this program produce distance matrix in a csv file, the csv file is attached with this file.

The **pseudo** code of java program that was written for calculating distance matrix: (the program is attached with file)

```
Public static float calculate Distance (Accident a, Accident b) {
    Int Makhraj = 17;
    Float soorat = 0;
    Soorat += (abs (a.NumberofVehicles - b.NumberofVehicles)) / 6;
    Soorat += (abs (a.AgeofCasualty - b.AgeofCasualty)) / 94;
    If (a.HolidayOrWeekend == 0 && b.HolidayOrWeekend == 0) {
      Makhraj--;
    } else {
      If (a.HolidayOrWeekend! = b.HolidayOrWeekend) {
        Soorat += 1;
```

```
    }
}
If (a.RoadClass.equals ("Unclassified") || b.RoadClass.equals ("Unclassified")) {
    Makhraj--;
} else {
    If (! a.RoadClass.equals (b.RoadClass)) {
        Soorat += 1;
    }
}
If (! a.AccidentDate.equals (b.AccidentDate)) {
    Soorat += 1;
}
If (! a.Time24hr.equals (b.Time24hr)) {
    Soorat += 1;
}
If (! a.RoadSurface.equals (b.RoadSurface)) {
    Soorat += 1;
}
If (! a.LightingConditions.equals (b.LightingConditions)) {
    Soorat += 1;
}
If (! a.WeatherConditions.equals (b.WeatherConditions)) {
    Soorat += 1;
}
If (! a.CasualtyClass.equals (b.CasualtyClass)) {
    Soorat += 1;
}
If (! a.CasualtySeverity.equals (b.CasualtySeverity)) {
    Soorat += 1;
```

```
    }
    If (! a.SexofCasualty.equals (b.SexofCasualty)) {
        Soorat += 1;
    }
    If (! a.TypeofVehicle.equals (b.TypeofVehicle)) {
        Soorat += 1;
    }
    If (! a.DayofWeek.equals (b.DayofWeek)) {
        Soorat += 1;
    }
    If (! a.Timenominal.equals (b.Timenominal)) {
        Soorat += 1;
    }
    If (! a.Season.equals (b.Season)) {
        Soorat += 1;
    }
    If (! a.ageOfCasualtyNominal.equals (b.ageOfCasualtyNominal)) {
        Soorat += 1;
    }
    Return soorat / makhraj;
}
Public static float [] [] CreateDistanceMatrix (Accident [] accidents) {
    Float [] [] distanceMatrix = new float [2665] [2665];
    For (int i = 1; i <= 2664; i++) {
        For (int j = 1; j <= i; j++) {
            DistanceMatrix[i] [j] = calculateDistance (accidents[i], accidents[j]);
        }
    }
    For (int i = 1; i <= 2664; i++) {
```

```
        For (int j = i + 1; j <= 2664; j++) {
            DistanceMatrix[i] [j] = distanceMatrix[j] [i];
        }
    }
    Return distanceMatrix;
}
```

Columns correlation analysis

Correlation analysis was done by **R** for some columns of data, the results are in below:

Road Surface and Weather Conditions

X-squared = 3000.2 DF = 32 p-value < 2.2e-16

Dependent because **3000.2>1.492**

Lighting Conditions and Time (nominal)

X-squared = 1165.2 DF = 12 p-value < 2.2e-16

Dependent because **1165.2>0.0147**

Road Surface and casualty severity

X-squared = 5.2885 DF = 8 p-value=0.7263

Independent because **5.2885<9.877**

Day of weak and HolidayOrWeekend

X-squared = 2487.2 DF = 6 p-value < 2.2e-16

Dependent because **2487.2>2.1e-05**

Season and Weather Conditions

X-squared = 198.23 DF = 24 p-value < 2.2e-16

Dependent because **198.23>0.535**

Road class and type of vehicle

X-squared = 206 DF = 75 p-value = 3.707e-14

Dependent because **206>16.005**

Day of weak and type of vehicle

X-squared = 200.13 DF = 90 p-value = 2.379e-10

Dependent because **200.13>29.71**

Time (nominal) and type of vehicle

X-squared = 188.4 DF = 45 p-value = 2.2e-16

Dependent because **188.4>4.057**

Quality of data

I think the data that I collected relatively has high quality because there is little missing data in it, little noise and few **outliers.** Because of these factors the data is appropriate for data mining process and is potential for extracting very helpful and valuable knowledge and wisdom.