KNOWLEDGE DISCOVERY BY ANALYZING TRAFFIC, WEATHER CONDITIONS AND DRIVING OFFENSES DATA IN A PARTICULAR SPOT IN A SPECIFIC ROAD

Arman Yousef Zadeh Shooshtari

Computer Science Department, Faculty of Computer Sciences, University of Victoria, Canada.
armanyousefzade95@gmail.com

Abstract

In this project, I analyzed traffic, weather and driving offenses data to discover knowledge in the subject of traffic and its reasons in Tehran, Iran. I narrowed my concentration on the north bound of Tehran-Karaj freeway. It is necessary to mention that I mainly used C# programming language, Python, and R language for my data collecting and statistical and data mining related tasks. In order to achieve the desired results, I took advantage of the following algorithms and techniques: Basic Statistical Methods, Diagram Drawing Methods, chi-squared and correlation Methods, data difference formula, Apriori algorithm, Decision Tree using C4.5 methods, logistic regression, and linear regression. The ultimate data file after cleaning and adding augmented data consisted of 31 columns and about 8000 rows. Of course in some phases, I had to increase the number of columns in order to achieve better and more accurate results.

I gathered my traffic and driving offenses data from Iran's ministry of roads and urban development (Iran's Road Managing Center) [1] and my weather data from WorldWeatherOnline API [2]. Iran's Road Managing Center provides daily and hourly traffic count data for each month of the year in excel format. I merged all the excel files into one csv file which contains data rows for each and every hour of the year (1394 in Shamsi). For collecting weather information for each day, I used a web service (API) from WorldWeatherOnline. It is necessary to mention that I wrote a program in c# to retrieve the weather data. Moreover, I calculated some calendar properties for my data which has done by using the c# PersianCalendar library.

The weather hourly data were not available for all hours of the day, which led me to use some kind of central tendency to calculate a unique set of data for each day. For example, I used MOD for the weather code and Mean for numerical attributes like temperature. The data contains 3.9% missing in column Time/Date, which I omitted those rows completely. After that, I generated some basic statistical data which are available in the following sections. Then, I produced some diagrams in order to show some basic rules and then I focused on finding the correlation between the attributes. As far as our data is mostly numerical, I used Covariance and Correlation Coefficient. You can see the result in the "Correlation.xlsx" and "Correlation2.xlsx" file. The file contains the correlation coefficient value for every two columns. After that, I calculated data differences and generated the difference matrix. Then, I used the Apriori algorithm to find appropriate association rules and by removing undesired rules which had low confidence, support or lift (not strong rules) I generated 500 rules. In the previous step, I changed some parts of the database and I added some columns and also discretized some of them in order to facilitate the process of finding association rules. You can find the rules in the "Rules.txt" and "Rules.xlsx" files. The next step was generating decision trees which was done by using the ctree method in R. you can find the trees in the Decision Tree Folder. In the next step, I used python for implementing logistic regression and linear regression. I used regression to predict different attributes based on other attributes. The implementation of logistic regression and linear regression is available in the Jupyter Notebooks folder.

The desired knowledge to be discovered by this project was finding the causes of congestion and traffic and also finding the reasons for the different driving offenses with respect to some parameters. I should say that I might have reached some of the desired knowledge which you can see at the end of this report.

Table of Contents

1 Introduction

In this project, I analyzed traffic, weather and driving offenses data to discover knowledge in the subject of traffic and its reasons in Tehran, Iran. I narrowed my concentration on the north bound of Tehran-Karaj freeway. I gathered my traffic and driving offenses data from Iran's ministry of roads and urban development (Iran's Road Managing Center) [1] and my weather data from WorldWeatherOnline API [2]. I described my complete methodology in details below. It is necessary to mention that I mainly used C# programming language, Python, and R language for my data collecting and statistical and data mining related tasks.

2 Data Collection

As I mentioned above, the main data for this project is the traffic and driving offenses data which is from Iran's Road Managing Center. The weather data is from WorldWeatherOnline.

2.1 Traffic and Driving Offenses Data

Iran's Road Managing Center provides daily and hourly traffic count data for each month of the year in excel format. The properties which are included in those excel files are discussed below, in the next section. I merged all the excel files into one csv file which contains data rows for each and every hour of the year (1394 in Shamsi). It contains 8419 data rows.

2.2 Weather Data

For collecting weather information for each day, I used a web service (API) from WorldWeatherOnline. It has a Past Local Weather Service which provides a comprehensive weather archive. With each request, I was able to receive data for a period of 30 days. By sending requests and receiving the responses for all month of the year 1394, I reached a csv file containing the whole weather data for all 364 days. It is necessary to mention that I wrote a program in c# to achieve all above.

Moreover, I calculated some calendar properties for my data which has done using the c# PersianCalendar library. As I mentioned earlier, I am going to discuss data properties in the following sections below.

3 Data Cleaning

As far as the traffic data didn't need any special 'Data Cleaning' process, I only focus on cleaning the weather and calendar data which were augmented later.

The weather API accepts requests in HTTP GET messages and it will response in either JSON or XML formats. I used the JSON format in my application for receiving weather data for a specific period of time. The response message consists of the following fields:

- Astronomy Data (sunrise, sunset, moonrise, and moonset) which the last 2 parameters don't have any special influence in traffic patterns.
- Min and Max of the Temperature in a day
- Hourly data which this itself consists of following parameters for some hours in a day:
    - Humidity
    - Wind speed

- Temperature
- Cloud covering percentage
- Pressure
- Weather code
- Etc.

As I underlined the word ‘some’ in the last page, in the last line, the hourly data are not available for all hours of the day, which led me to use some kind of central tendency to calculate a unique set of data for each day. For example, I used MOD for the weather code as it doesn’t make any sense to use other central tendency parameters like mean.

4 Data Properties

The ultimate data file after cleaning and adding augmented data consisted of 31 columns and about 8000 rows. The properties/columns are:

Property	Type	Description
Month	Numerical	Date
Day	Numerical	Date
From(H)	Numerical	Time
From(M)	Numerical	Time
From(S)	Numerical	Time
To(H)	Numerical	Time
To(M)	Numerical	Time
To(S)	Numerical	Time
Working Duration	Numerical	-
Total Vehicles	Numerical	-
Total Vehicles (Class 1)	Numerical	-
Total Vehicles (Class 2)	Numerical	-
Total Vehicles (Class 3)	Numerical	-
Total Vehicles (Class 4)	Numerical	-
Total Vehicles (Class 5)	Numerical	-
AVG Velocity	Numerical	Total Average Velocity of vehicles
No of Unauthorized Overtaking’s	Numerical	-
No of Unauthorized Distance	Numerical	-
No of Unauthorized Speed	Numerical	-
Is Holiday	Boolean(Asymmetric)	About 30% Yes
Is a Day Before Holiday	Boolean(Asymmetric)	About 30% Yes
Sun Rise (Hour)	Numerical	-
Sun Rise (Minute)	Numerical	-
Sun Rise(AM/PM)	Boolean(Symmetric)	-
Sun Set (Hour)	Numerical	-
Sun Set (Minute)	Numerical	-
Sun Set(AM/PM)	Boolean(Symmetric)	-
AVG Temperature	Numerical	-
Weather Mode	Numerical	A code determining the weather mode
Visibility (KM)	Numerical	-

Cloud Covering (%)	Numerical	-

The columns are self-explanatory. Further columns may be added during the project different phases. The weather mode values are described in the table below.

Weather Code	Condition
395	Moderate or heavy snow in area with thunder
392	Patchy light snow in area with thunder
389	Moderate or heavy rain in area with thunder
386	Patchy light rain in area with thunder
377	Moderate or heavy showers of ice pellets
374	Light showers of ice pellets
371	Moderate or heavy snow showers
368	Light snow showers
365	Moderate or heavy sleet showers
362	Light sleet showers
359	Torrential rain shower
356	Moderate or heavy rain shower
353	Light rain shower
350	Ice pellets
338	Heavy snow
335	Patchy heavy snow
332	Moderate snow
329	Patchy moderate snow
326	Light snow
323	Patchy light snow
320	Moderate or heavy sleet
317	Light sleet
314	Moderate or Heavy freezing rain
311	Light freezing rain
308	Heavy rain
305	Heavy rain at times
302	Moderate rain
299	Moderate rain at times
296	Light rain
293	Patchy light rain
284	Heavy freezing drizzle
281	Freezing drizzle
266	Light drizzle
263	Patchy light drizzle
260	Freezing fog

248	Fog
230	Blizzard
227	Blowing snow
200	Thundery outbreaks in nearby
185	Patchy freezing drizzle nearby
182	Patchy sleet nearby
179	Patchy snow nearby
176	Patchy rain nearby
143	Mist
122	Overcast
119	Cloudy
116	Partly Cloudy
113	Clear/Sunny

The data contains 3.9% missing in column Time/Date, which I omitted those rows completely. The CSV and XLSX files are available and named “Total.csv” and “Total.xlsx”. Also, I provided the source codes of the programs which were designed in order to clear the data, preprocessing them and so on and so forth. It is available and named “Codes.zip”. The codes are written in C#.Net and using Visual Studio 2013.

5 Basic Statistical parameters

In the following table you can see the basic statistical parameters for each column of data:

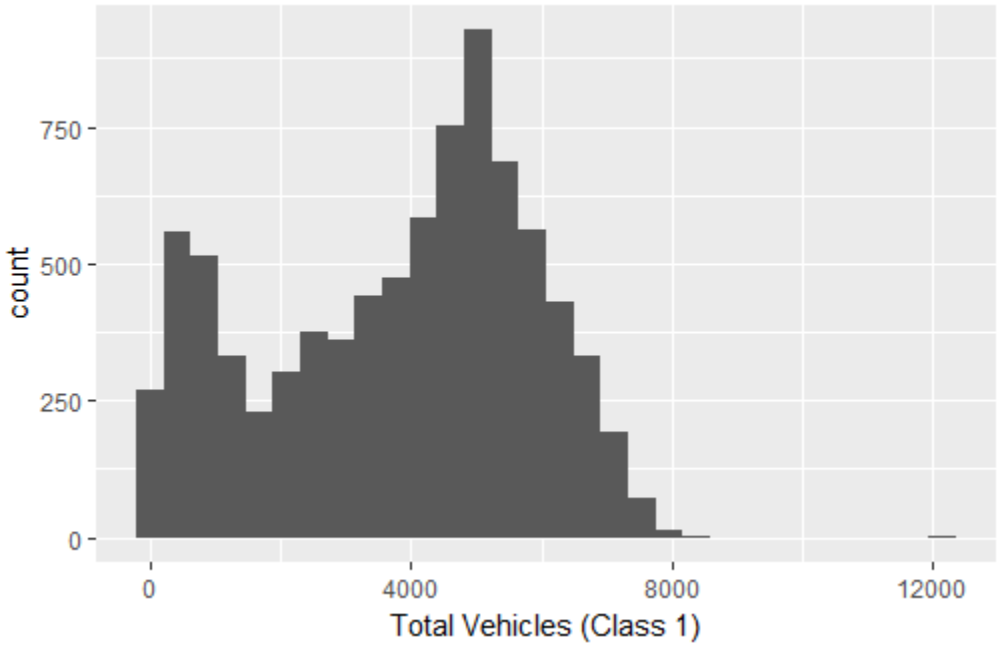
Column Name	Mean	Median	Mod	Variance	Standard Deviation	Range
Working Duration	59.6	60	60	9.8	3.1	45
Total Vehicles	4157.1	4827	0	4814533.8	2194.2	12563
Total Vehicles (C1)	3795.6	4220	0	4140876.9	2034.9	12135
Total Vehicles (C2)	190.7	85	0	103945.9	322.4	3287
Total Vehicles (C3)	51.4	16	0	5454.6	73.9	779
Total Vehicles (C4)	56.2	35	0	4314.8	65.7	350
Total Vehicles (C5)	63.2	9	0	11598.7	107.7	693
AVG Velocity	90.5	104.98	0	1214.5	34.8	125.42
No. Unau… Overtaking’s	594.1	173	0	598657.6	773.7	4459
No. Unau… Distance	827.2	721	0	563247.1	750.5	4037
No. Unau…Speed	197.8	0	0	181736.0	426.3	2453
Is Holiday	0.3	FALSE	FALSE	0.2	0.4	1
Is a Day Before Holiday	0.3	FALSE	FALSE	0.2	0.4	1
AVG Temperature	21.2	21.75	35.3	108.4	10.4	36
Weather Mode	-	113	113	1092.2	33.0	-
Visibility (KM)	9.9	10	10	0.2	0.4	3.125
Cloud Covering (%)	14.8	7.75	0	334.7	18.3	98.125

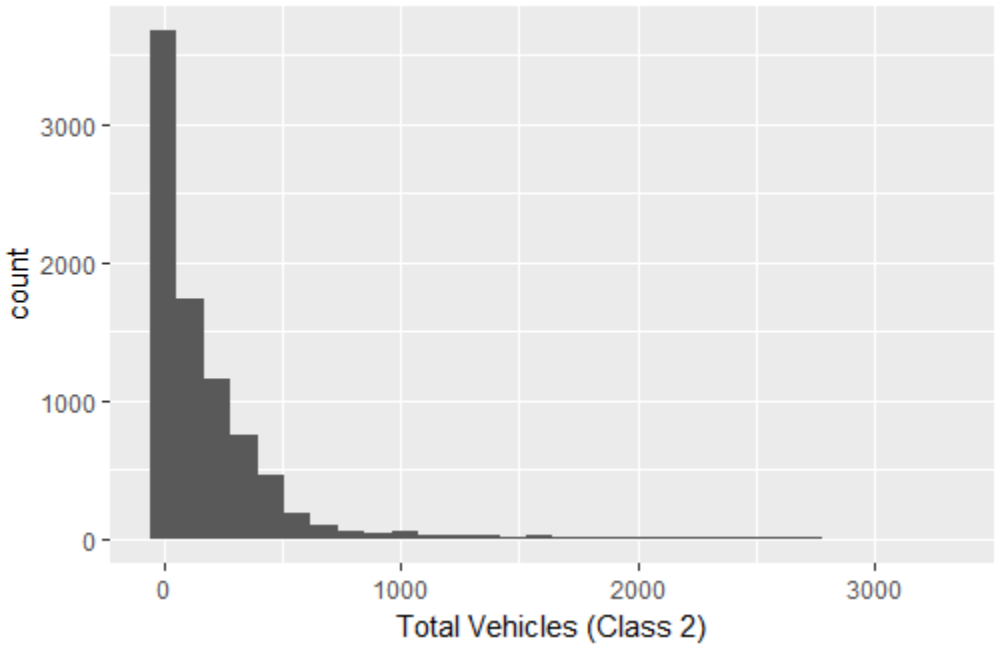
6 Diagrams

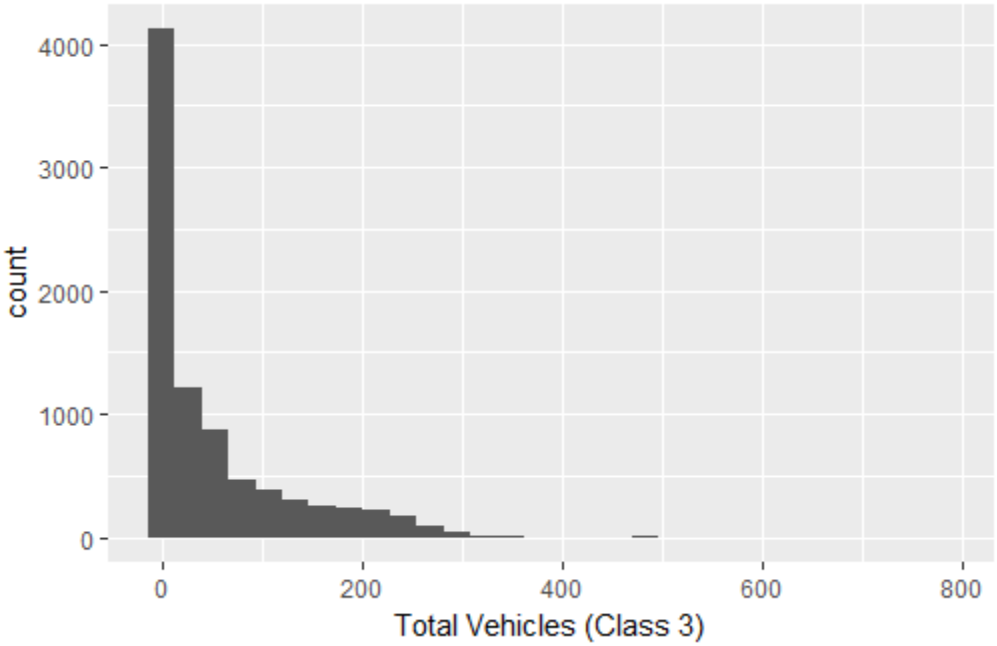
In this section we are going to draw some important diagrams in two parts:

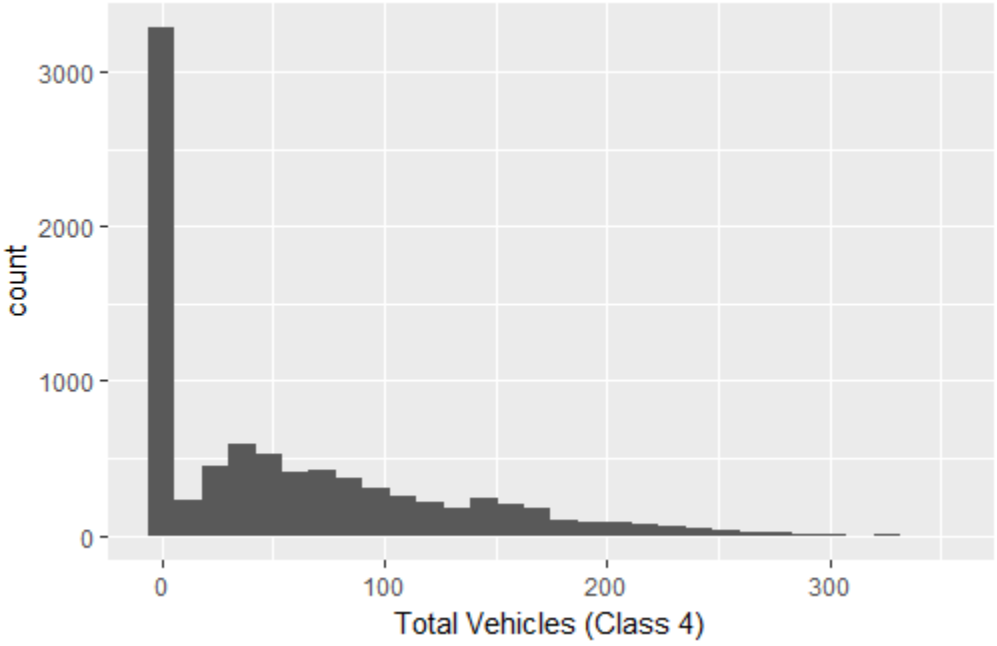
6.1 Diagrams for individual properties

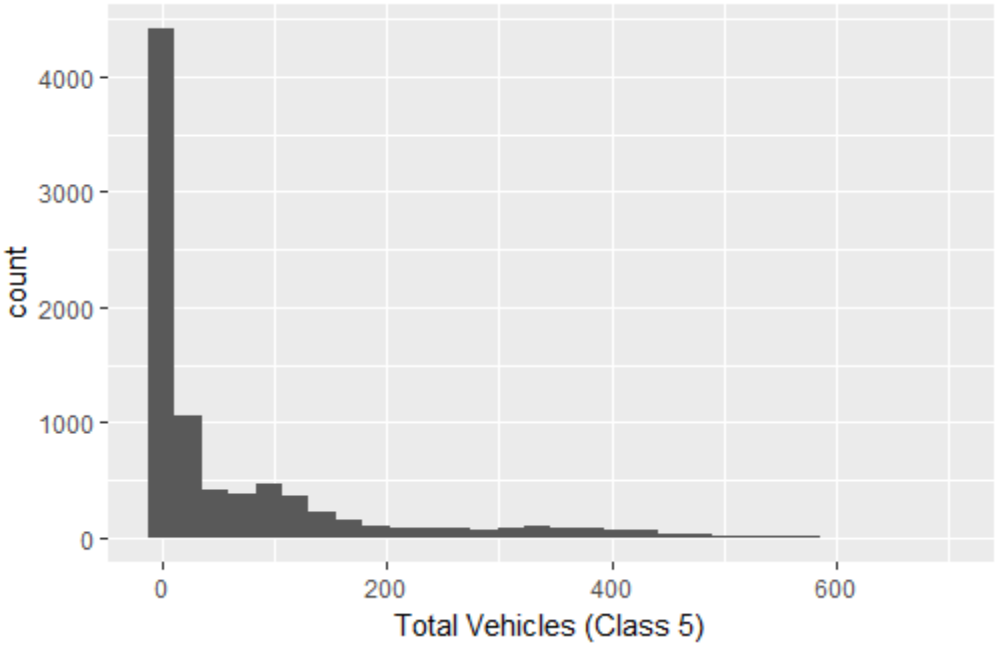
Here you can see diagrams for some important properties. The diagrams are self-explanatory.

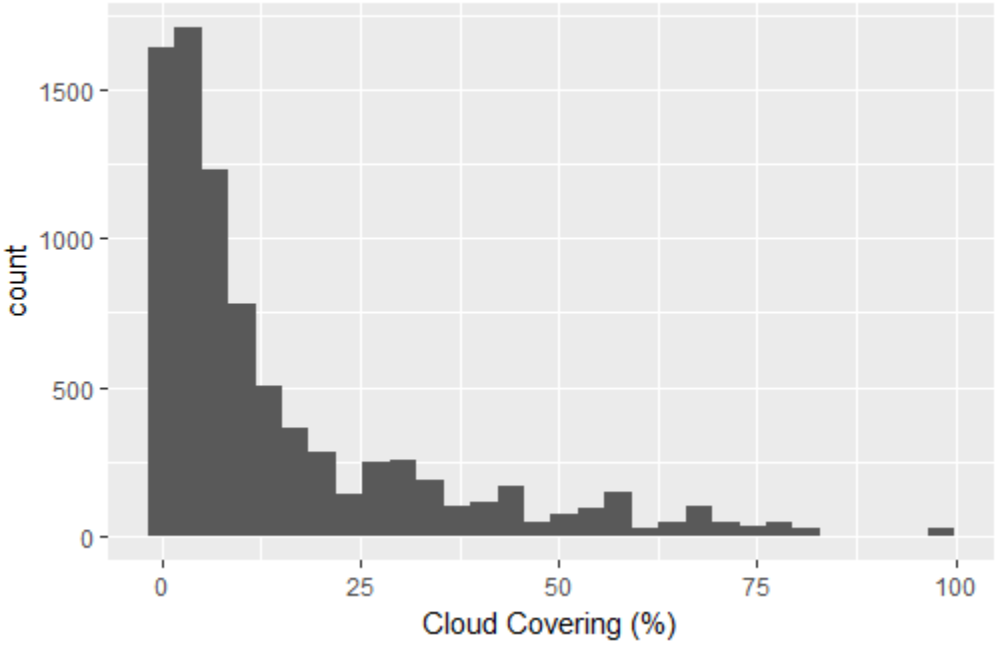


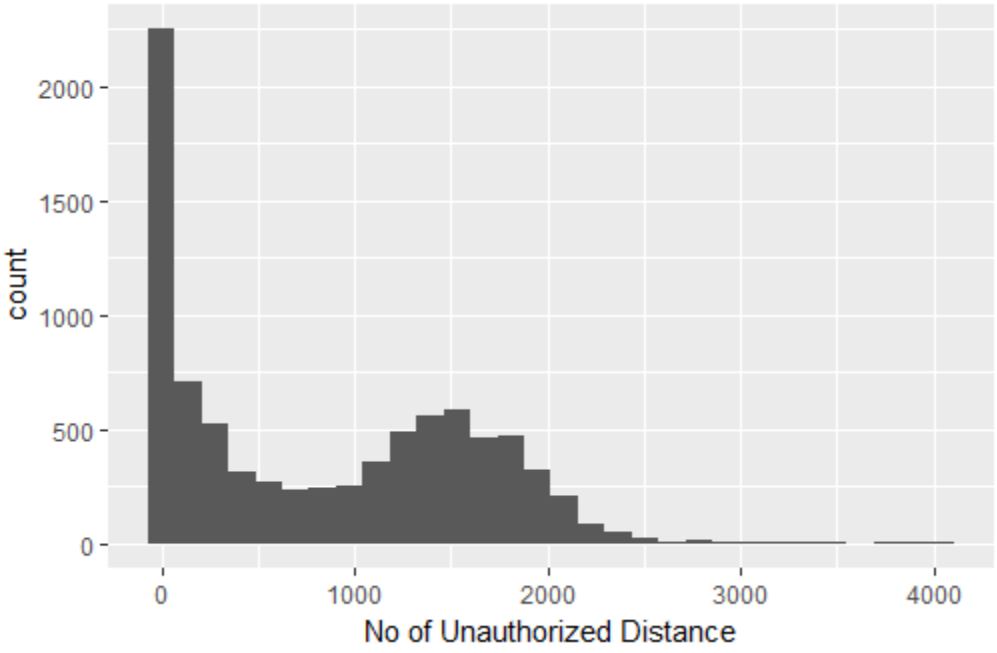


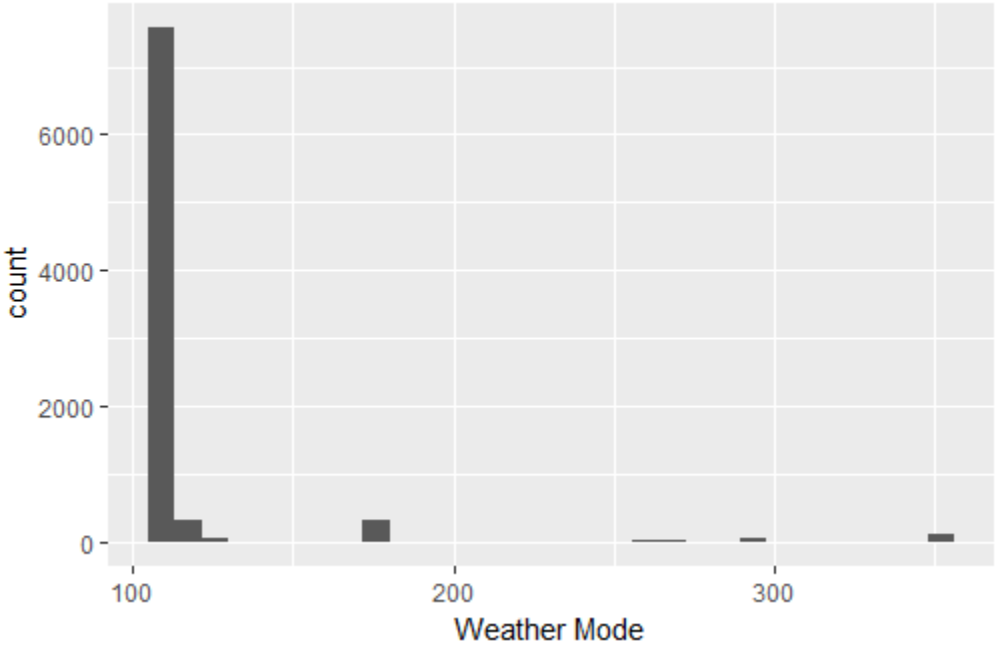

4000
3000
2000
1000
0
count
0
200
400
600
800
Total Vehicles (Class 3)

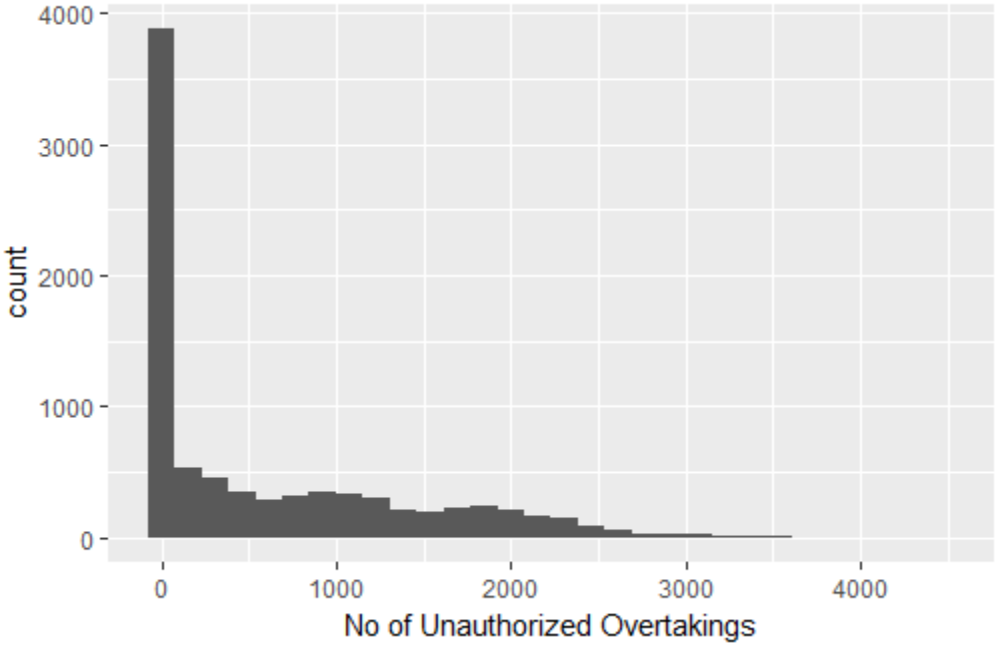

3000
2000
1000
0
count
0
100
200
300
Total Vehicles (Class 4)

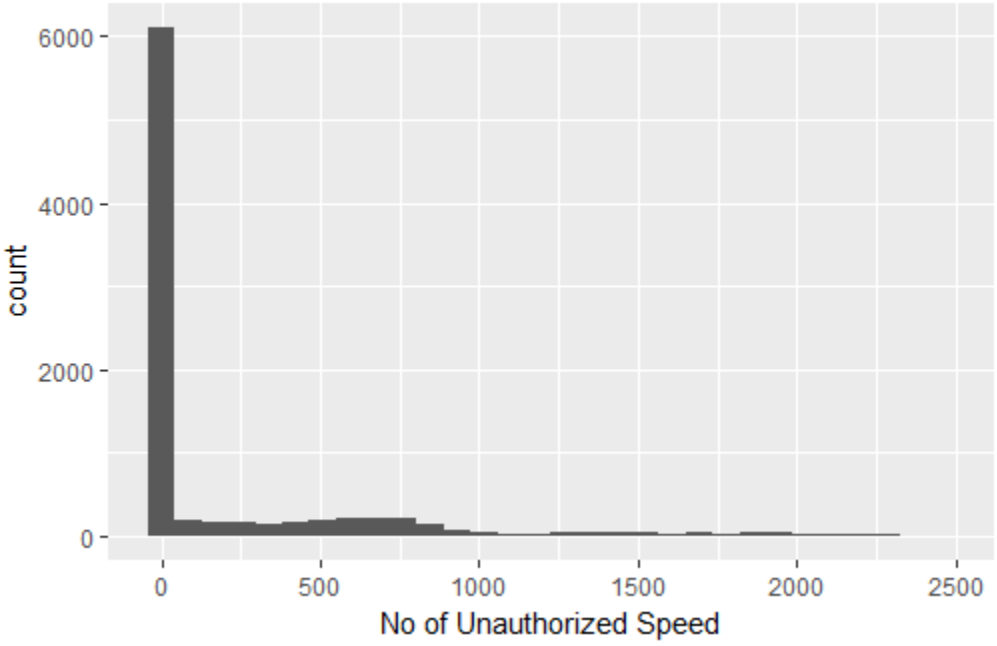

4000
3000
2000
1000
0
count
0
200
400
600
Total Vehicles (Class 5)

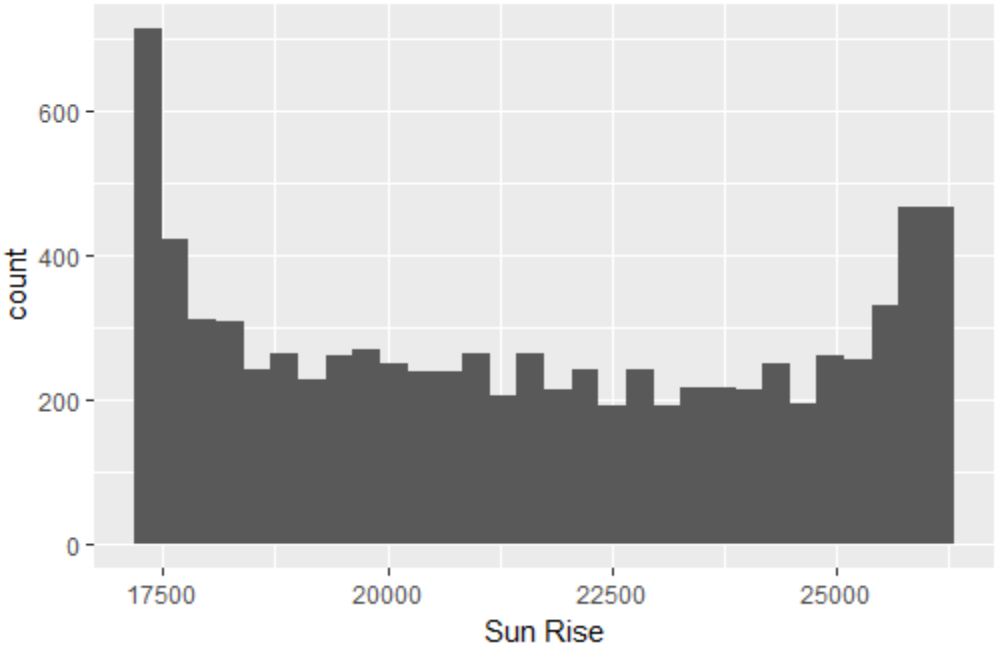

1500
1000
500
0
count
0
25
50
75
100
Cloud Covering (%)

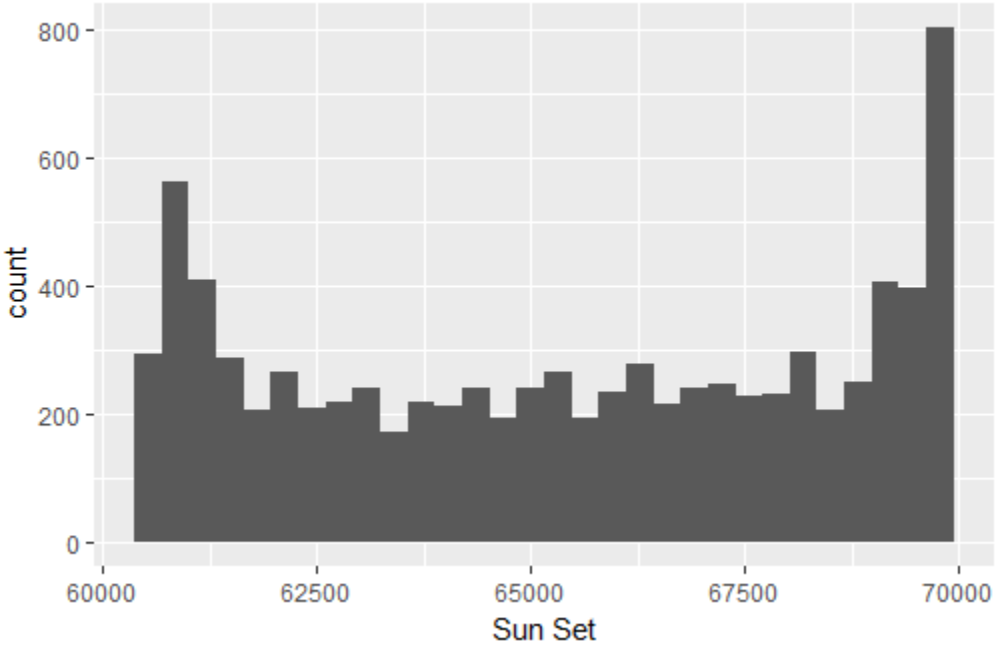

2000
1500
1000
500
0
count
0
1000
2000
3000
4000
No of Unauthorized Distance

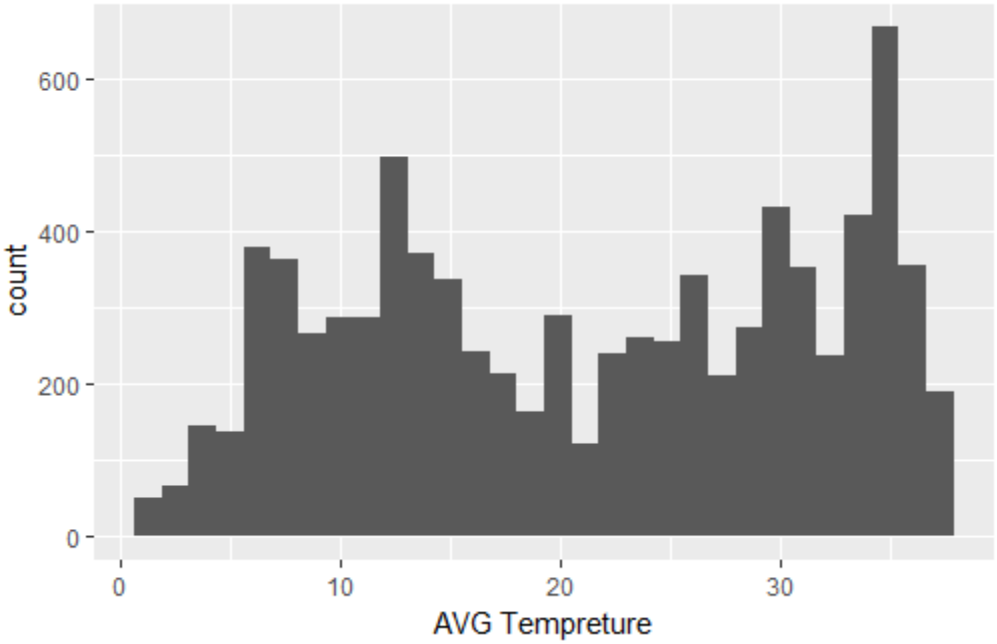

6000
4000
2000
0
count
100
200
300
Weather Mode

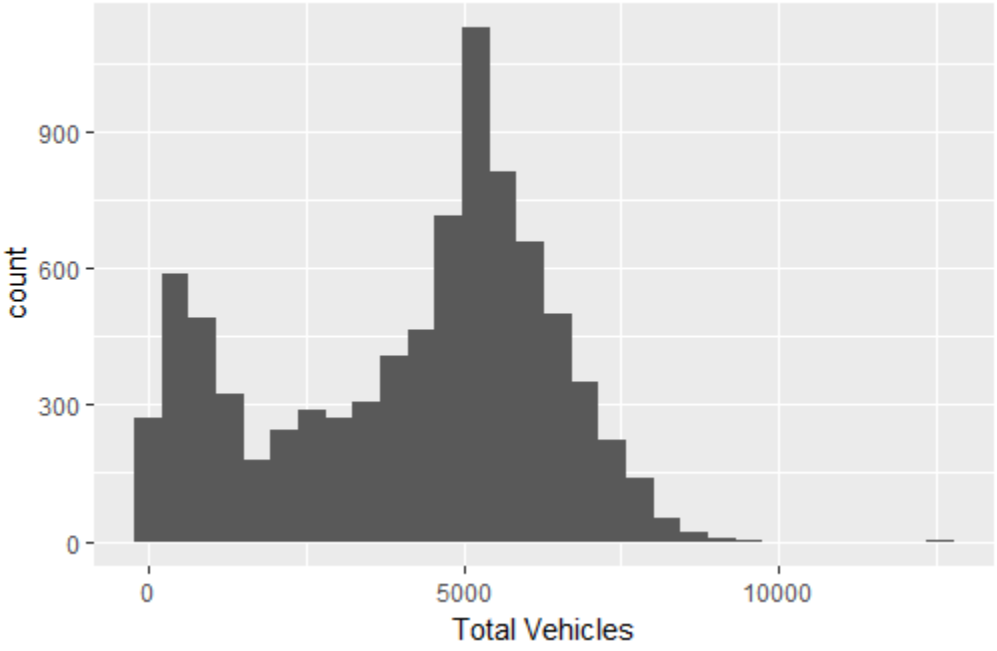

4000
3000
2000
1000
0
count
0
1000
2000
3000
4000
No of Unauthorized Overtakings

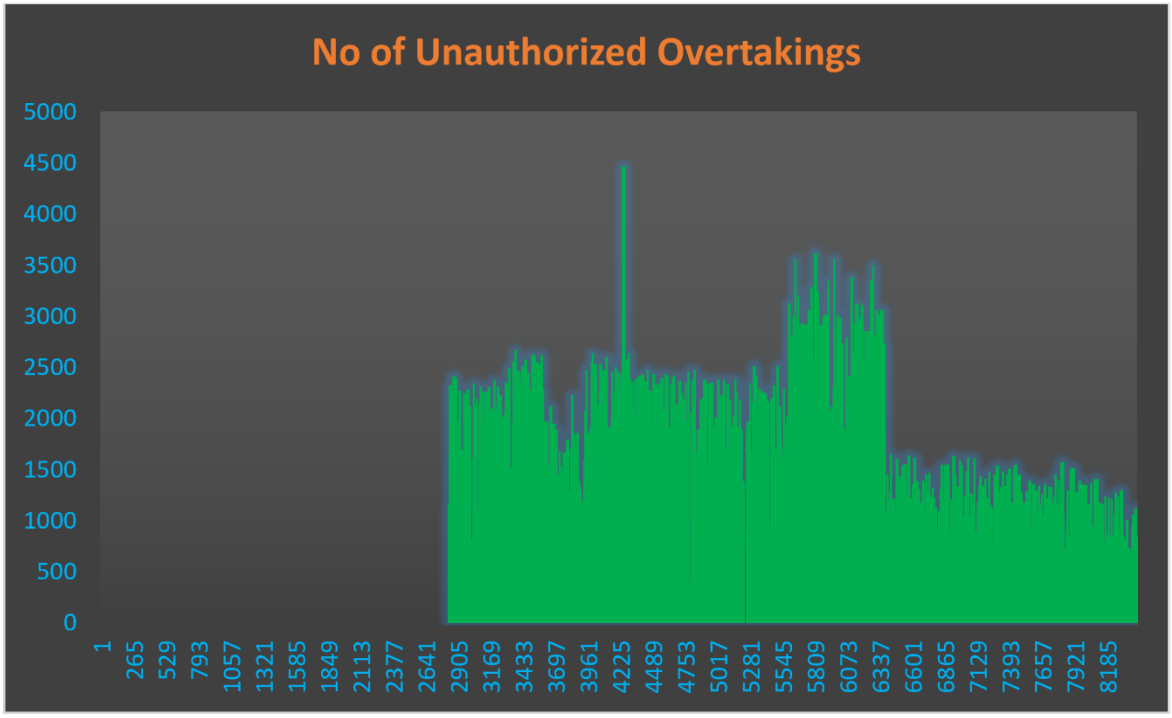

6000
4000
2000
0
count
0
500
1000
1500
2000
2500
No of Unauthorized Speed

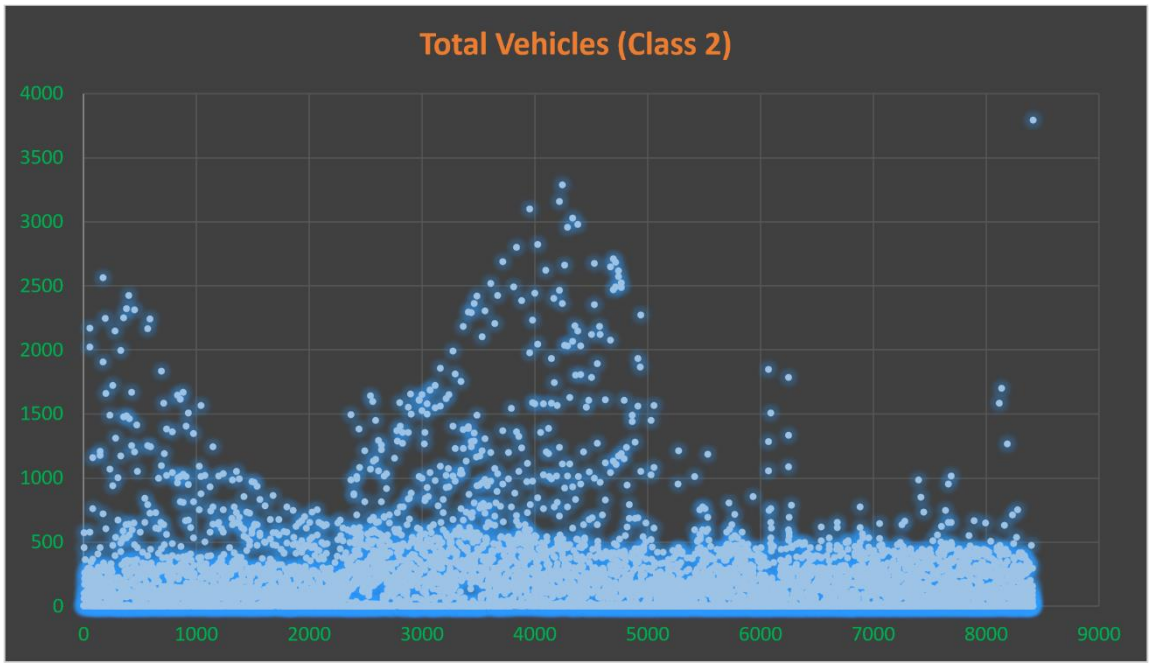

600
400
200
0
count
17500
20000
22500
25000
Sun Rise

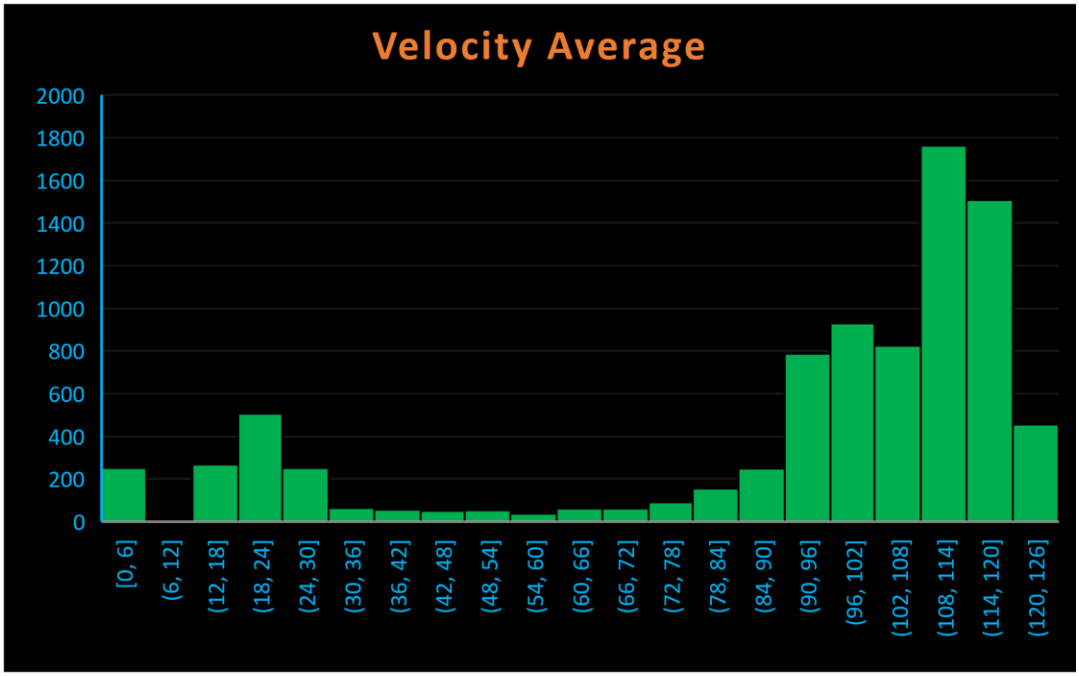

800
600
400
200
0
count
60000
62500
65000
67500
70000
Sun Set

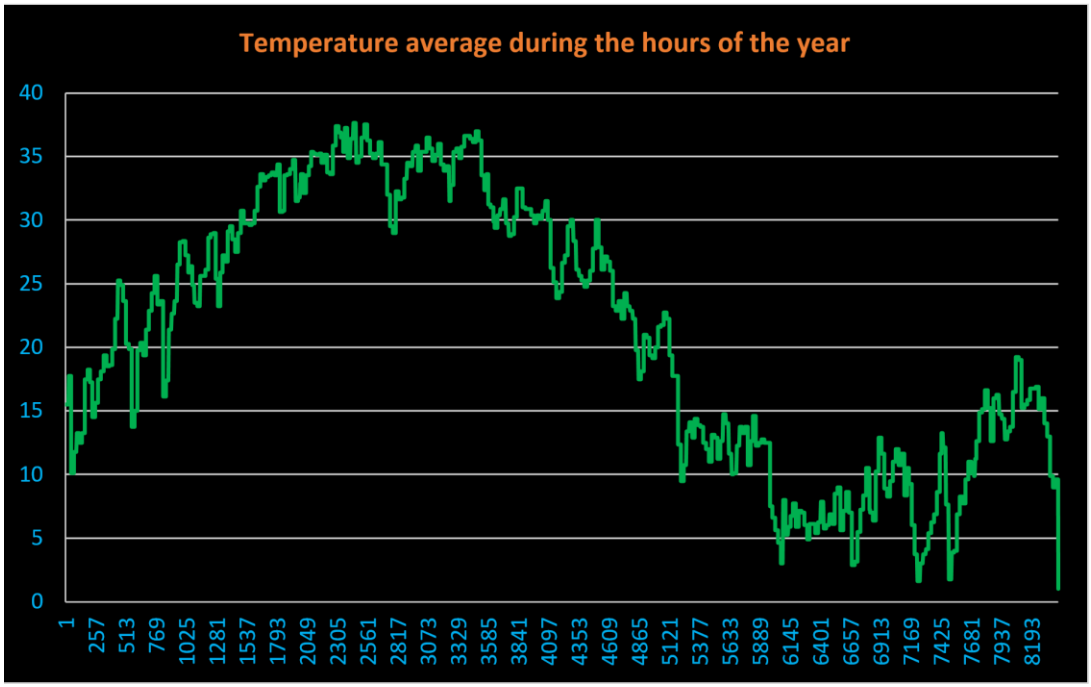

600
400
200
0
count
0
10
20
30
AVG Tempreture

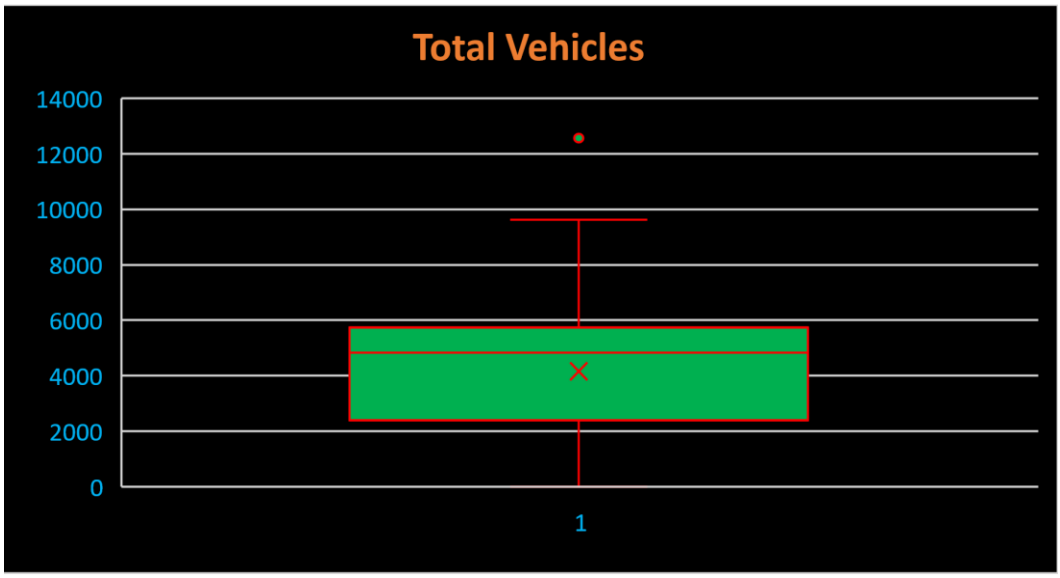

900
600
300
0
count
0
5000
10000
Total Vehicles

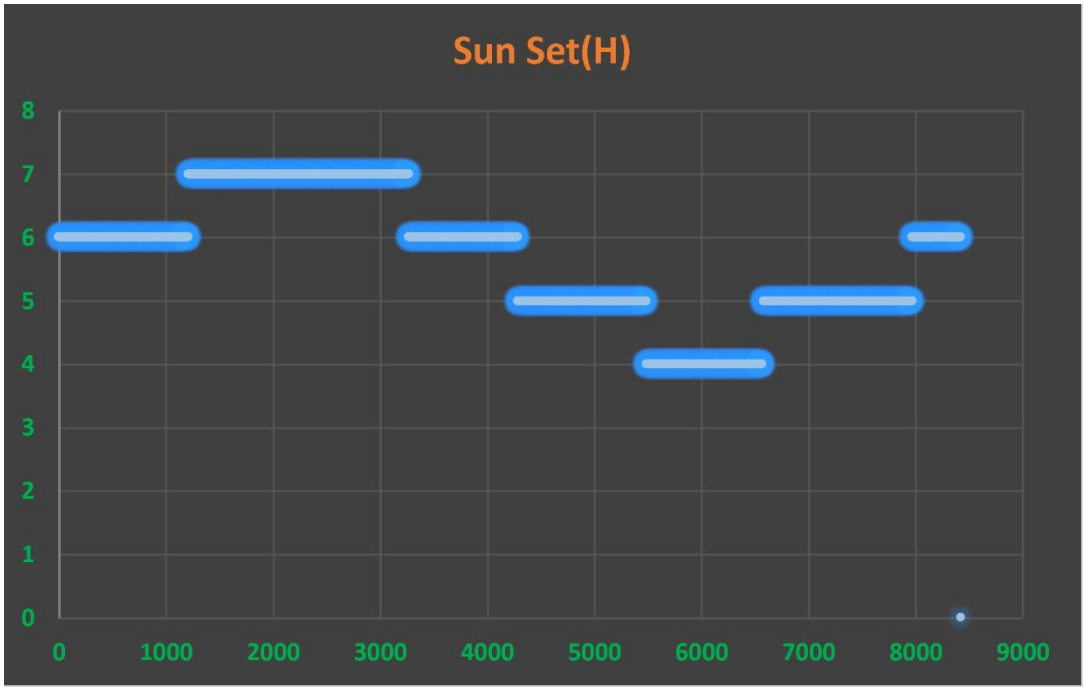

No of Unauthorized Overtakings
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
1
265
529
793
1057
1321
1585
1849
2113
2377
2641
2905
3169
3433
3697
3961
4225
4489
4753
5017
5281
5545
5809
6073
6337
6601
6865
7129
7393
7657
7921
8185

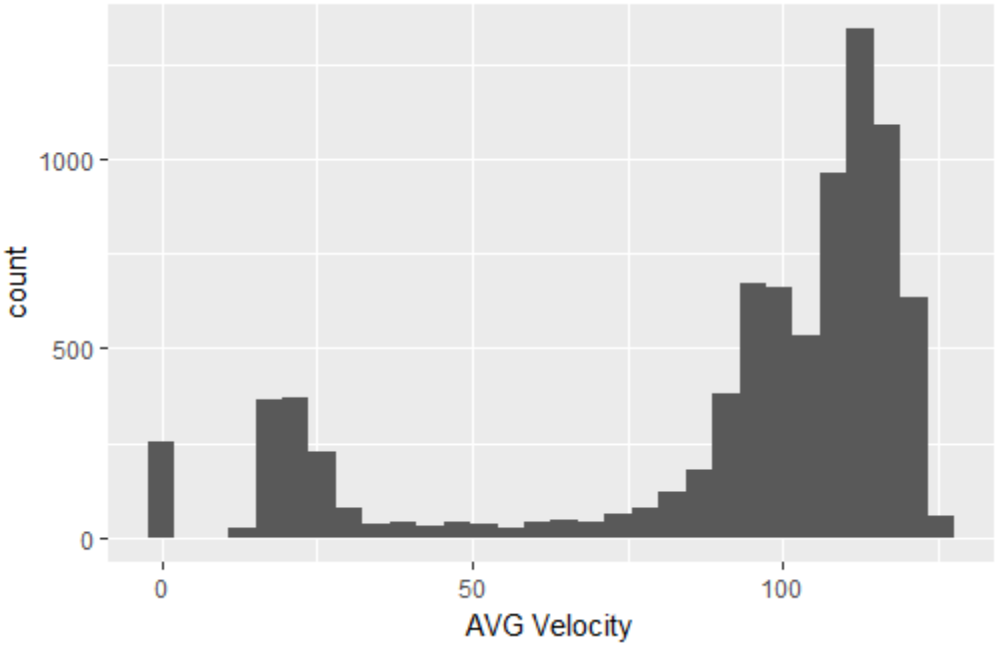

Total Vehicles (Class 2)
4000
3500
3000
2500
2000
1500
1000
500
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000

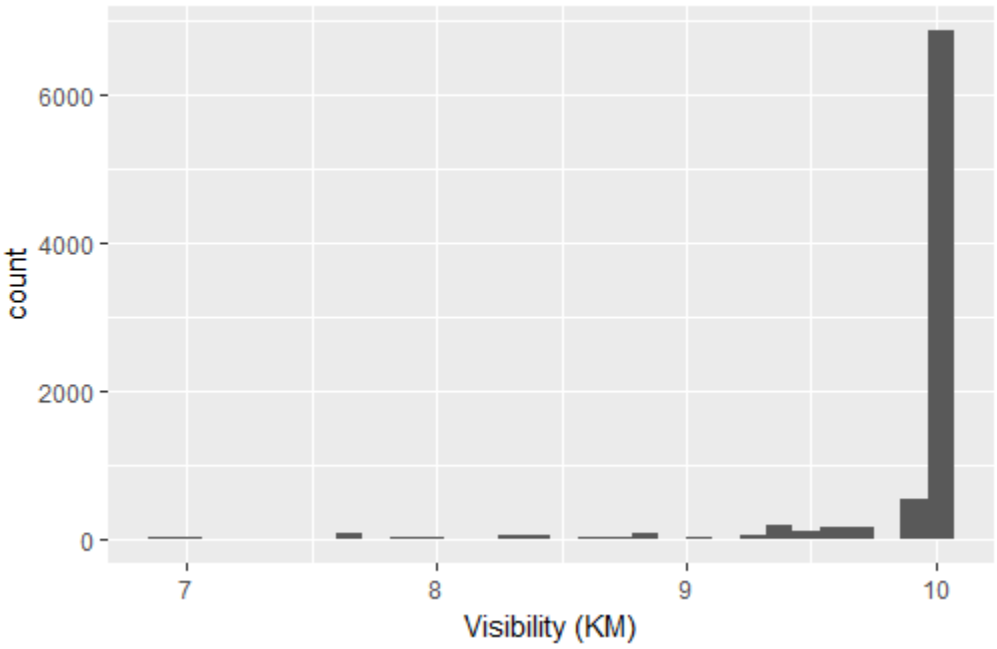

Velocity Average
2000
1800
1600
1400
1200
1000
800
600
400
200
0
[0, 6]
(6, 12]
(12, 18]
(18, 24]
(24, 30]
(30, 36]
(36, 42]
(42, 48]
(48, 54]
(54, 60]
(60, 66]
(66, 72]
(72, 78]
(78, 84]
(84, 90]
(90, 96]
(96, 102]
(102, 108]
(108, 114]
(114, 120]
(120, 126]

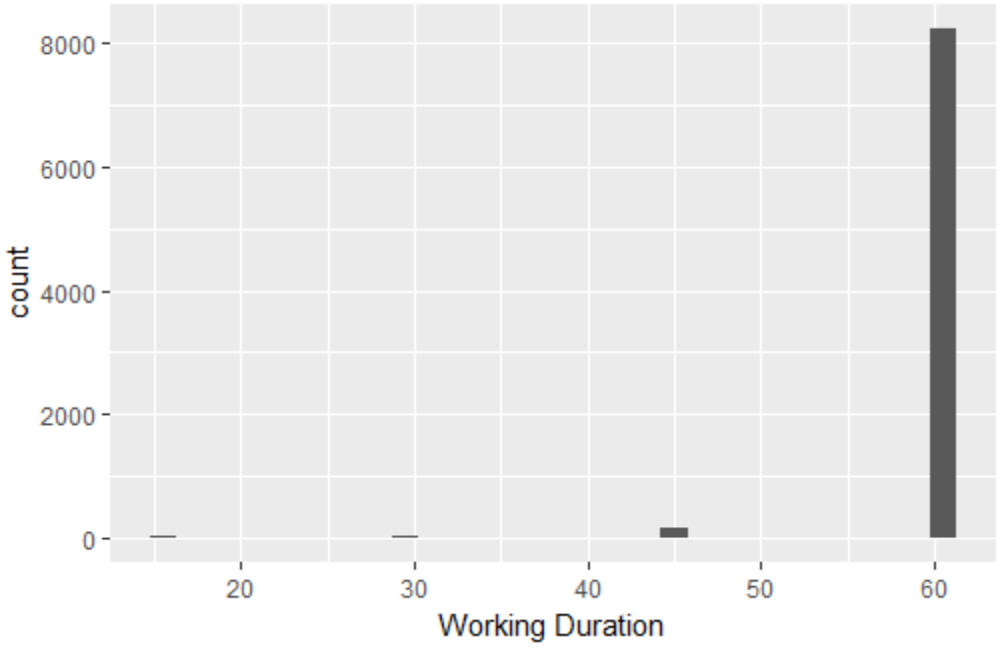

Temperature average during the hours of the year
40
35
30
25
20
15
10
5
0
1
257
513
769
1025
1281
1537
1793
2049
2305
2561
2817
3073
3329
3585
3841
4097
4353
4609
4865
5121
5377
5633
5889
6145
6401
6657
6913
7169
7425
7681
7937
8193


Total Vehicles
14000
12000
10000
8000
6000
4000
2000
0
1

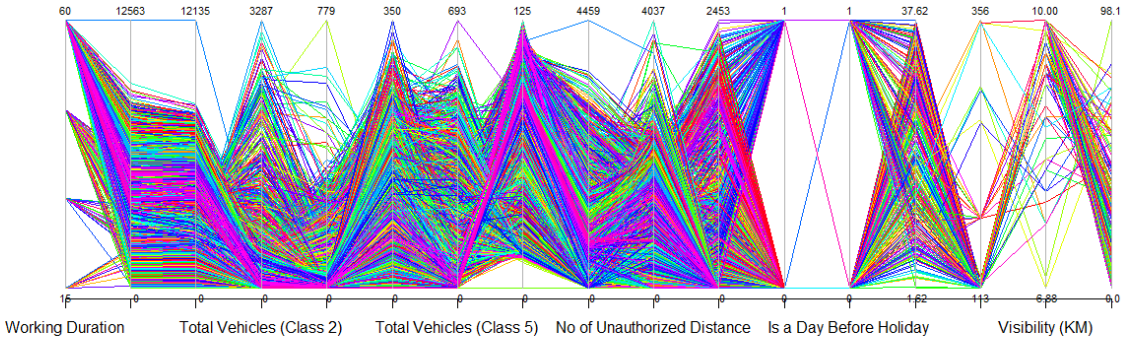

Sun Set(H)
8
7
6
5
4
3
2
1
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000


count
1000
500
0
0
50
100
AVG Velocity


count
6000
4000
2000
0
7
8
9
10
Visibility (KM)



6.2 Parallel Coordinates Diagram



7 Correlations

In this section, I am going to show the amount of relativity or correlation between the properties. As far as my data is mostly numerical, I used Covariance and Correlation Coefficient. You can see the result in the “Correlation.xlsx” and “Correlation2.xlsx” file. The file contains the correlation coefficient value for every two columns. I used coloring in order to understand the positive or negative correlations.

The pseudo code used in order to achieve this result is:

```
correlationmat = matrix(NA,32,32)

for(i in 1:31)
{
  correlationmat[i,1] = names(TotalData)[i]
  correlationmat[1,i] = names(TotalData)[i]
}

for(i in c(1,2,3,4,6,7,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,25,26,28,29,30,31))
{
  for(j in c(1,2,3,4,6,7,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,25,26,28,29,30,31))
  {
    correlationmat[i+1,j+1] = round(cor(TotalData[,c(i)],TotalData[,c(j)]),2)
  }
}

```

8 Data Differences

As far as I have 8419 rows of data, computing the complete Data Difference matrix is not feasible. Although I have written the necessary codes in R in order to do so which you can see the codes below.

```
diffmatrix = matrix(0,8419,8419)

minvector = matrix(0,1,31)
maxvector = matrix(0,1,31)

for(i in 1:31)
{
  minvector[i] = min(TotalData[,i])
  maxvector[i] = max(TotalData[,i])
}

for(i in 1:8419)
{
  for(j in 1:8419)
  {
    sumsigma = 0
    sumsigmadiff = 0

    for(k in 1:31)
    {
      sig = 0
      dif = 0

      if(TotalData[i,k] != 0 | TotalData[j,k] != 0)
      {
        sig = 1
      }

      dif = abs(TotalData[i,k] - TotalData[j,k])

      if(!(k == 20 |k == 21 |k == 24 |k == 27 ) & sig != 0)
      {
        dif = dif/(maxvector[k]-minvector[k])
      }

```

```
37          sumsigmadiff = sumsigmadiff + (dif*sig)
38
39          sumsigma = sumsigma + sig
40
41        }
42
43        diffmatrix[i,j] = round(sumsigmadiff/sumsigma,2)
44        diffmatrix[i,j]
45      }
46    }
```

However, I calculated this matrix for the first 24 rows as you can see below.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	0	0.01	0.03	0.02	0.03	0.03	0.04	0.04	0.05	0.06	0.08	0.1	0.12	0.14	0.1	0.11	0.14	0.18	0.16	0.17	0.16	0.09	0.1	0.08
2	0.01	0	0.02	0.03	0.03	0.03	0.04	0.04	0.05	0.06	0.08	0.1	0.12	0.14	0.1	0.11	0.14	0.18	0.16	0.17	0.16	0.1	0.1	0.08
3	0.03	0.02	0	0.03	0.04	0.03	0.05	0.05	0.06	0.07	0.09	0.11	0.12	0.14	0.11	0.12	0.14	0.18	0.16	0.17	0.17	0.1	0.11	0.09
4	0.02	0.03	0.03	0	0.01	0.03	0.03	0.03	0.04	0.04	0.06	0.08	0.1	0.12	0.08	0.09	0.12	0.16	0.14	0.14	0.14	0.07	0.08	0.07
5	0.03	0.03	0.04	0.01	0	0.02	0.03	0.03	0.04	0.05	0.06	0.07	0.09	0.11	0.08	0.09	0.1	0.15	0.12	0.13	0.13	0.07	0.07	0.06
6	0.03	0.03	0.03	0.03	0.02	0	0.02	0.03	0.04	0.05	0.07	0.08	0.09	0.11	0.09	0.1	0.11	0.15	0.13	0.14	0.14	0.08	0.09	0.07
7	0.04	0.04	0.05	0.03	0.03	0.02	0	0.01	0.02	0.03	0.05	0.07	0.09	0.1	0.07	0.08	0.1	0.14	0.12	0.14	0.14	0.08	0.09	0.09
8	0.04	0.04	0.05	0.03	0.03	0.03	0.01	0	0.02	0.02	0.04	0.06	0.08	0.1	0.06	0.07	0.1	0.14	0.12	0.13	0.13	0.07	0.08	0.08
9	0.05	0.05	0.06	0.04	0.04	0.04	0.02	0.02	0	0.02	0.04	0.06	0.09	0.1	0.07	0.08	0.1	0.14	0.12	0.13	0.13	0.08	0.09	0.09
10	0.06	0.06	0.07	0.04	0.05	0.05	0.03	0.02	0.02	0	0.03	0.05	0.08	0.09	0.05	0.07	0.09	0.13	0.11	0.13	0.13	0.07	0.08	0.09
11	0.08	0.08	0.09	0.06	0.06	0.07	0.05	0.04	0.04	0.03	0	0.03	0.05	0.07	0.03	0.04	0.07	0.12	0.1	0.1	0.1	0.07	0.07	0.07
12	0.1	0.1	0.11	0.08	0.07	0.08	0.07	0.06	0.06	0.05	0.03	0	0.03	0.05	0.03	0.02	0.04	0.13	0.11	0.08	0.08	0.07	0.08	0.08
13	0.12	0.12	0.12	0.1	0.09	0.09	0.09	0.08	0.09	0.08	0.05	0.03	0	0.03	0.04	0.04	0.02	0.11	0.09	0.06	0.06	0.09	0.09	0.09
14	0.14	0.14	0.14	0.12	0.11	0.11	0.1	0.1	0.1	0.09	0.07	0.05	0.03	0	0.06	0.06	0.03	0.08	0.05	0.04	0.04	0.11	0.1	0.1
15	0.1	0.1	0.11	0.08	0.08	0.09	0.07	0.06	0.07	0.05	0.03	0.03	0.04	0.06	0	0.01	0.04	0.12	0.09	0.07	0.07	0.06	0.06	0.08
16	0.11	0.11	0.12	0.09	0.09	0.1	0.08	0.07	0.08	0.07	0.04	0.02	0.04	0.06	0.01	0	0.03	0.12	0.1	0.07	0.06	0.06	0.06	0.08
17	0.14	0.14	0.14	0.12	0.1	0.11	0.1	0.1	0.1	0.09	0.07	0.04	0.02	0.03	0.04	0.03	0	0.09	0.07	0.05	0.04	0.08	0.07	0.09
18	0.18	0.18	0.18	0.16	0.15	0.15	0.14	0.14	0.14	0.13	0.12	0.13	0.11	0.08	0.12	0.12	0.09	0	0.04	0.07	0.08	0.12	0.12	0.14
19	0.16	0.16	0.16	0.14	0.12	0.13	0.12	0.12	0.12	0.11	0.1	0.11	0.09	0.05	0.09	0.1	0.07	0.04	0	0.04	0.05	0.09	0.09	0.11
20	0.17	0.17	0.17	0.14	0.13	0.14	0.14	0.13	0.13	0.13	0.1	0.08	0.06	0.04	0.07	0.07	0.05	0.07	0.04	0	0.02	0.09	0.08	0.1
21	0.16	0.16	0.17	0.14	0.13	0.14	0.14	0.13	0.13	0.13	0.1	0.08	0.06	0.04	0.07	0.06	0.04	0.08	0.05	0.02	0	0.08	0.07	0.1
22	0.09	0.1	0.1	0.07	0.07	0.08	0.08	0.07	0.08	0.07	0.07	0.07	0.09	0.11	0.06	0.06	0.08	0.12	0.09	0.09	0.08	0	0.01	0.06
23	0.1	0.1	0.11	0.08	0.07	0.09	0.09	0.08	0.09	0.08	0.07	0.08	0.09	0.1	0.06	0.06	0.07	0.12	0.09	0.08	0.07	0.01	0	0.05
24	0.08	0.08	0.09	0.07	0.06	0.07	0.09	0.08	0.09	0.09	0.07	0.08	0.09	0.1	0.08	0.08	0.09	0.14	0.11	0.1	0.1	0.06	0.05	0

9 Expecting Knowledge Discovery

I expect to discover some kind of correlation between the factors of the traffic, weather, and calendar. I may discover the reasons and the factors of the traffic, the reasons for different driving offenses and so on. Also, I may discover some kind of knowledge in other fields which are unexpected.

It is necessary to mention that the quality of my data is high enough to rely on the results but if I could use other traffic data like google traffic, I could have been more certain on my results.

10 Association Rule Mining

In this section, I am going to explain the methods and techniques used to mine Association Rules. I have to say that, as you might have perceived earlier, my Database is some kind of Data Table, which I need to convert it to Transaction Set in order to mine Association Rules. I mined Association Rules using Apriori Algorithm in R. I used “arules” and “arulesviz” libraries to mine and visualize the Rules:

```
install.packages("arules");
install.packages("arulesViz");
```

Before stepping into the Mining Association Rules, I need to first select the useful columns (attributes) and omit the unnecessary ones and also I need to discretize the attributes in order to facilitate converting the Data Table to Transaction Set. I did steps along with converting the Data Table to Transaction Set as follows:

```
TotalData$Month <- sub("^", "Month ", TotalData$Month )
```

Numeric column: `discretize (TotalData$`Total Vehicles`,categories = 10);`

Nominal column: `as.factor(TotalData$Month)`

Change to the transaction (useful columns):

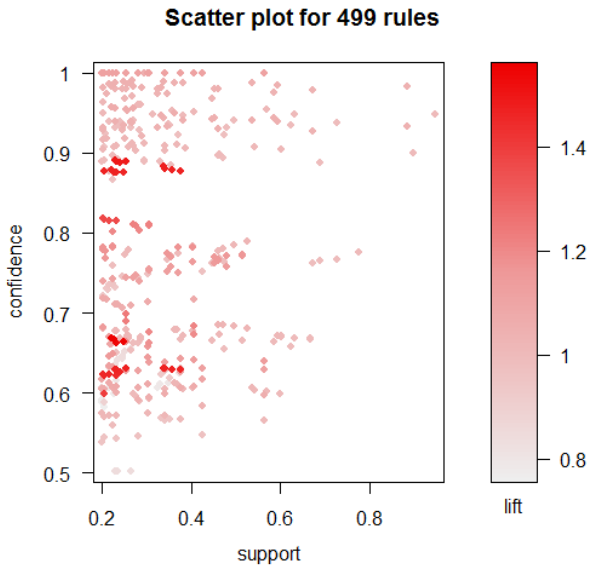
```
TRData= as(TotalData[,c(1,3,6,10,16,17,18,19,20,21,28,29,30,31)],"transactions")
```

I then mined Association Rules using Apriori algorithm and plotted them in order to find the desired Rules as follows:

```
Rules = apriori(TRData, parameter=list(support=0.2, confidence=0.5, maxlen=31));

plot(rules)
```

I mined 499 Rules which the plot for them with respect to “support”, “confidence” and “lift” will look like this:



I then wrote the rules into a text file which you can find it as “Rules.txt”. In that file, a typical rule will look like this:

"84" "{No of Unauthorized Speed=[0, 245),Is Holiday=Holiday} => {Weather Mode=Clear/Sunny}" 0.22 0.88 0.98

Which it represents the following data:

Rule No. – Rule (LHS=>RHS) – support – confidence – lift

You can find all the rules in the “Rules.txt” file, as mentioned earlier. Here you can see interesting knowledge:

{Is a Day Before Holiday=Not_a_Day_Before_Holiday,Weather Mode=Clear/Sunny,Visibility (KM)=[9.38,10.00]} => {No of Unauthorized Overtakings=[0, 446)} 0.33 0.57 0.95

It shows that on a sunny day, which is not a day before a holiday and the visibility is high, the number of unauthorized overtaking is low. You can find these kinds of interesting rules implying further filters on them in the excel file, “Rules.xlsx”. [3]

Before moving on to the next section I should mention the point that the method and the parameters which I used for discretizing, has a high influence on the results, so for extracting further knowledge, I may want to consider adopting other methods like Decision Trees, which I did and I am going to describe these methods in the following sections.

11 Decision Tree Mining

In order to find further knowledge, I adopted decision tree on different attributes such as “Total Vehicles” or “No Of unauthorized speed” to find the premises that resulted in an increase in “No Of unauthorized speed” for example. I used “ctree” function to generate the trees. The following code shows the steps:

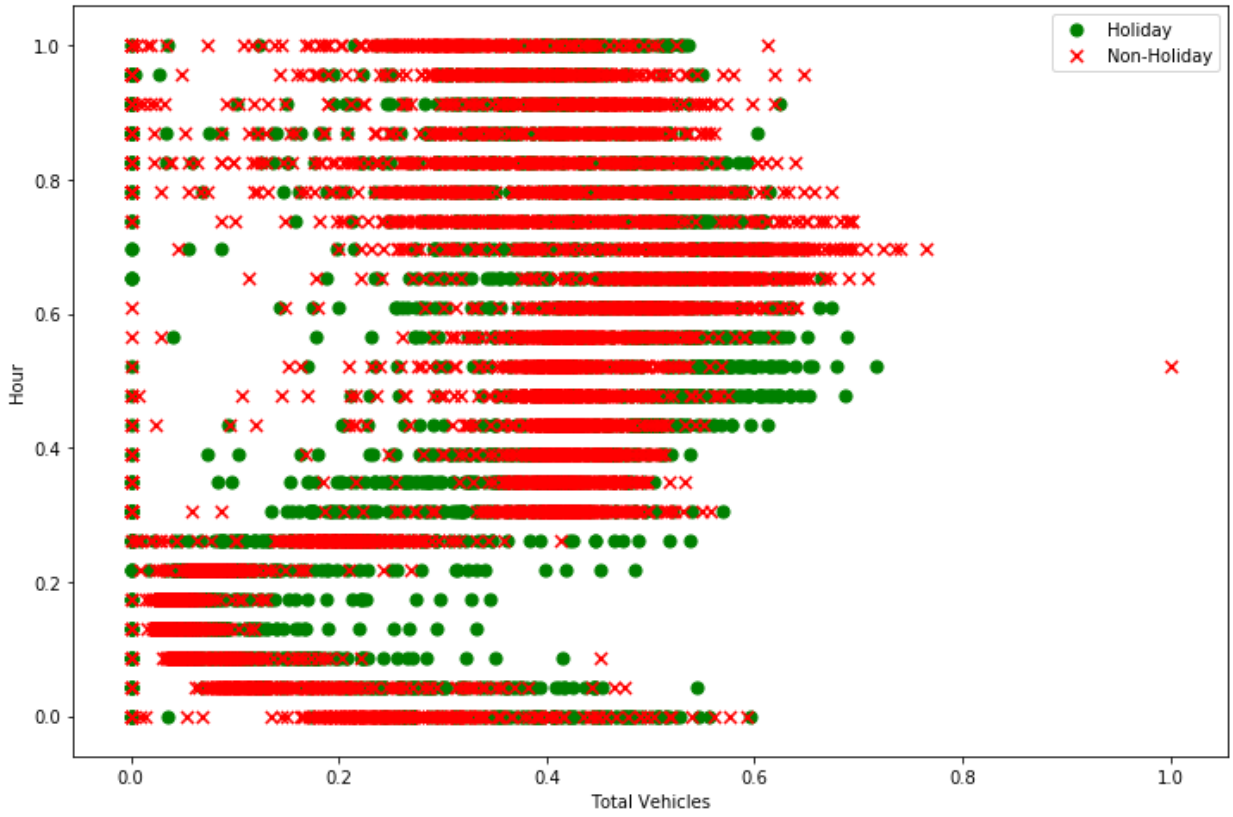
```
> form <- as.formula(`Total Vehicles` ~ `Is a Day Before Holiday`+`AVG Tempreture`+`From(H)`+`Weather Mode`+`Cloud Covering (%)`)
> a = ctree(form,TotalData)
> png("airct.png", res=100, height=7500, width=5000)
> plot(a, type="simple")
> dev.off()
```

The first line generates the desired formula which in this case is to predict the parameter before ‘~’ with respect to the other parameters. The second line generates the decision tree. The 3rd-5th lines draw the tree into an enormous png file for further analyzing by the user. You can find the decision tree images in the folder “Decision Trees”.

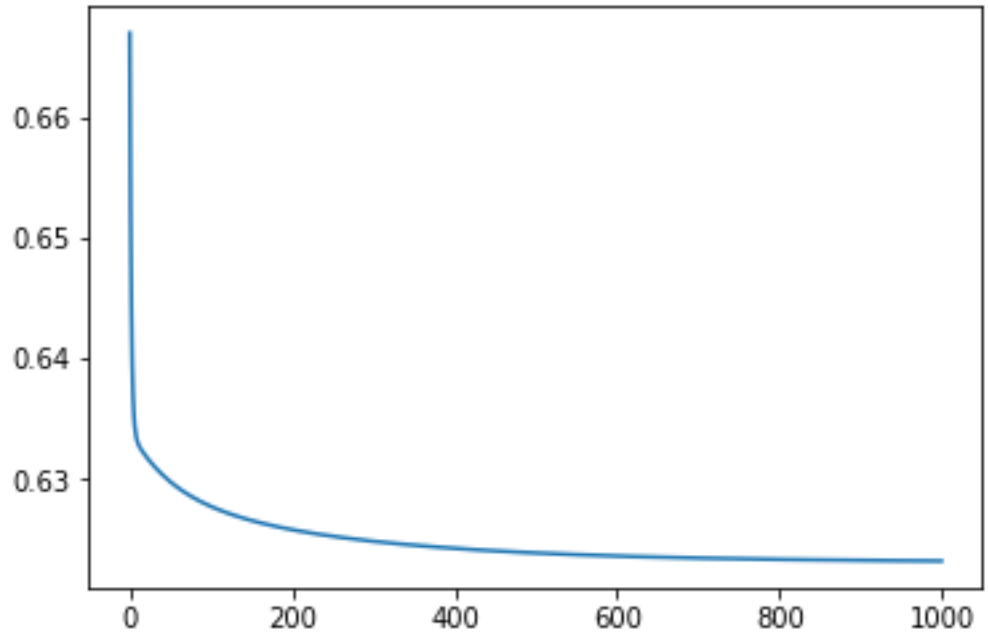
12 Logistic Regression

I used Logistic Regression for three applications in this project. I used Python for implementing that. I trained the classification and then I tested the accuracy of the classifier. All the results, diagrams, and implementations are available in Jupyter Notebook folder. Here, I just explain them shortly, you can see that folder to see everything in details. Firstly, I utilized logistic regression in order to predict whether a

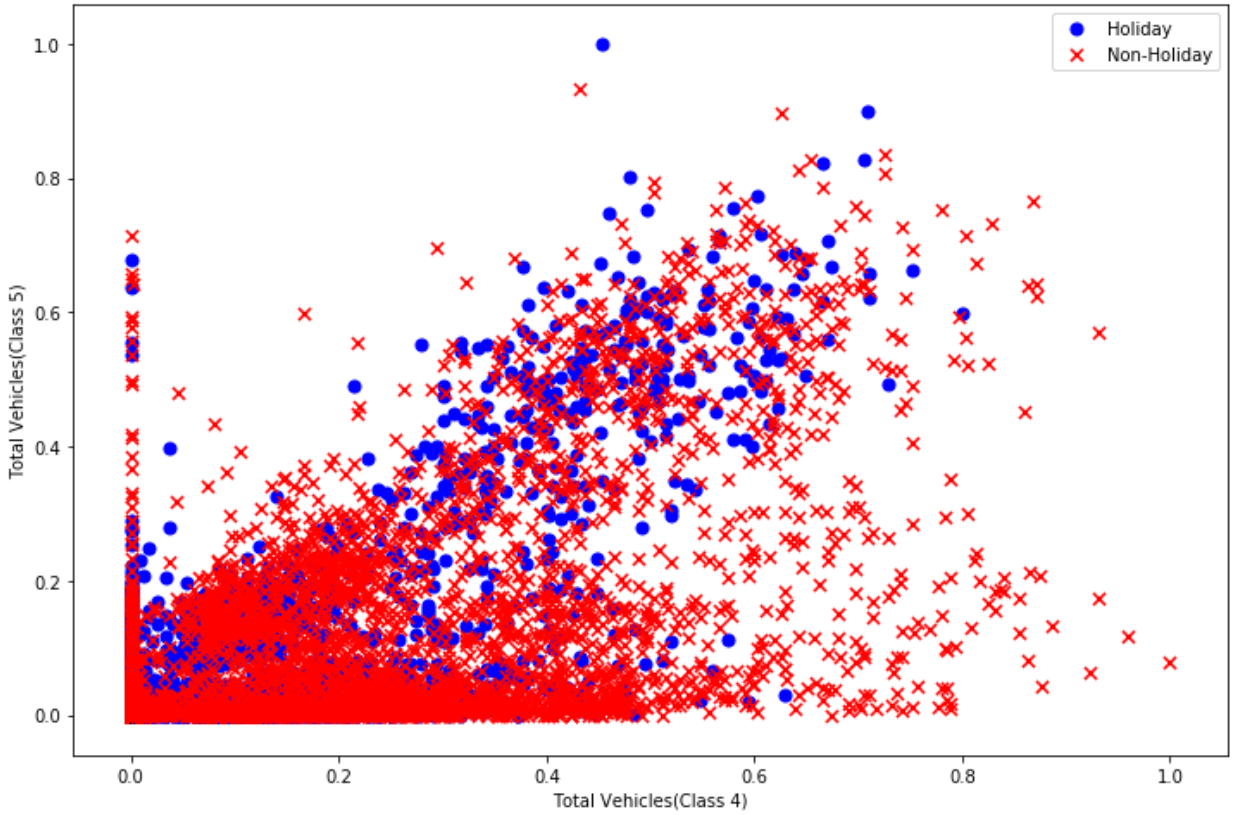
day is a holiday based on the number of vehicles and hour. The diagram which you can see below, shows the distribution of data based on these two attributes:



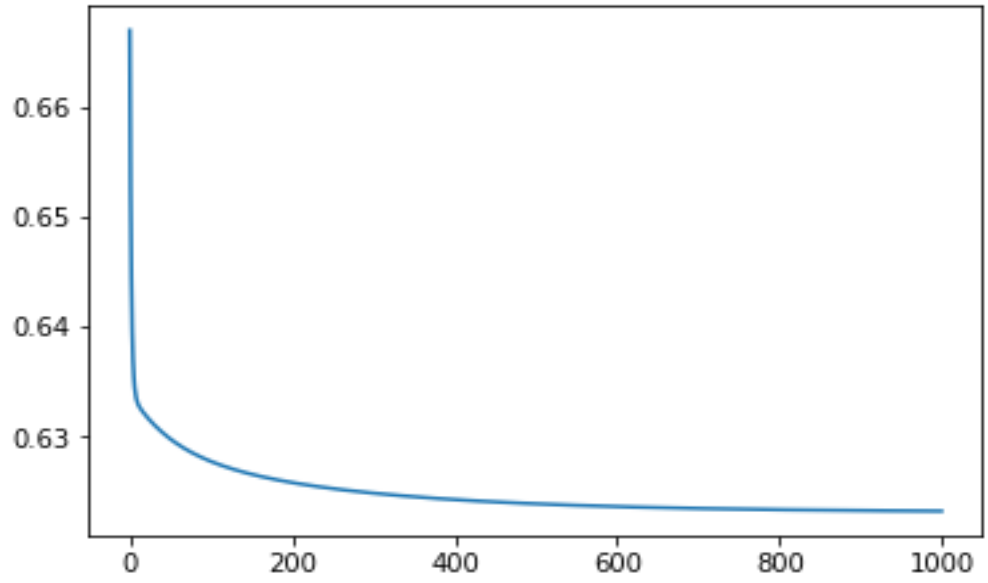
Below, you can see the Error curve which is decreasing and shows us that training the classifier was done correctly:



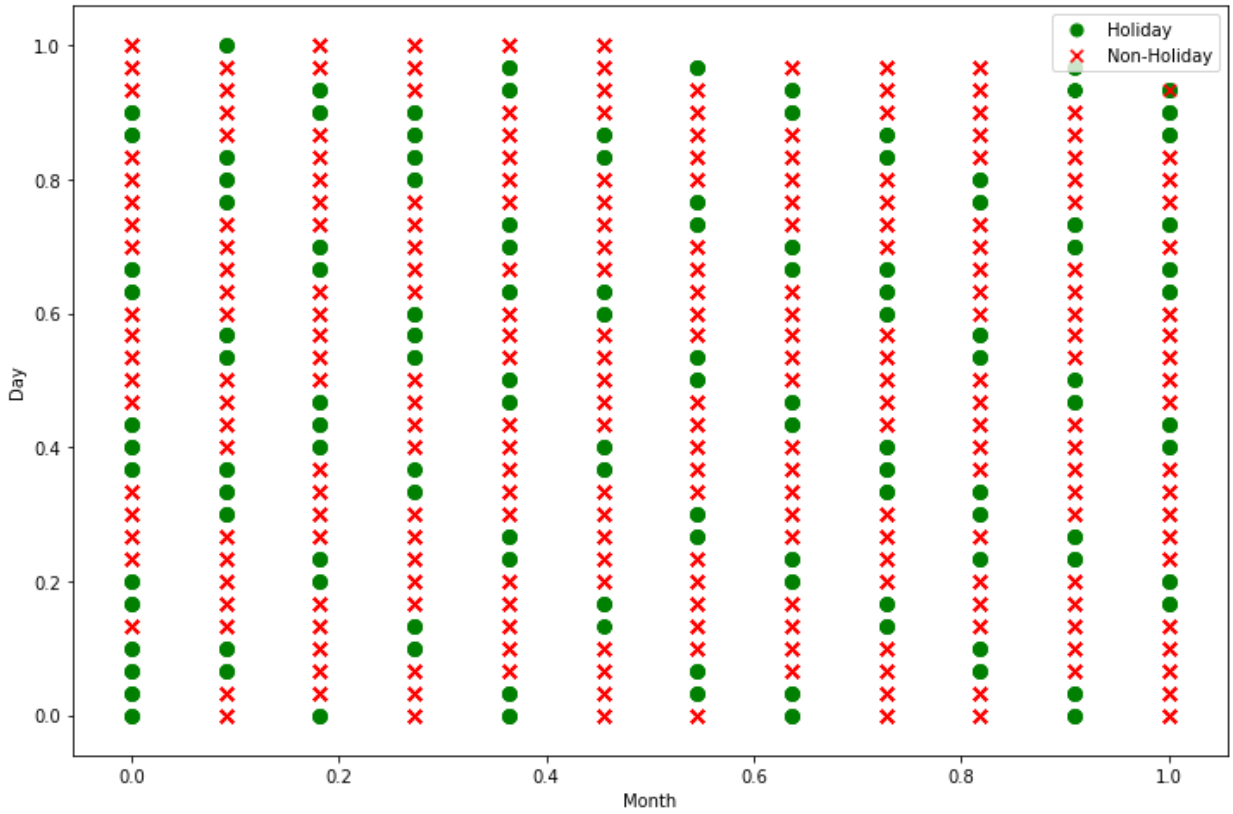
After the training, I tested the classifier on the data and I got 67% accuracy. The second application of logistic regression was for predicting whether a day is a holiday based on Number of vehicles of class 4 and vehicles of class 5. The diagram that is below depicts the distribution of data based on these two attributes:



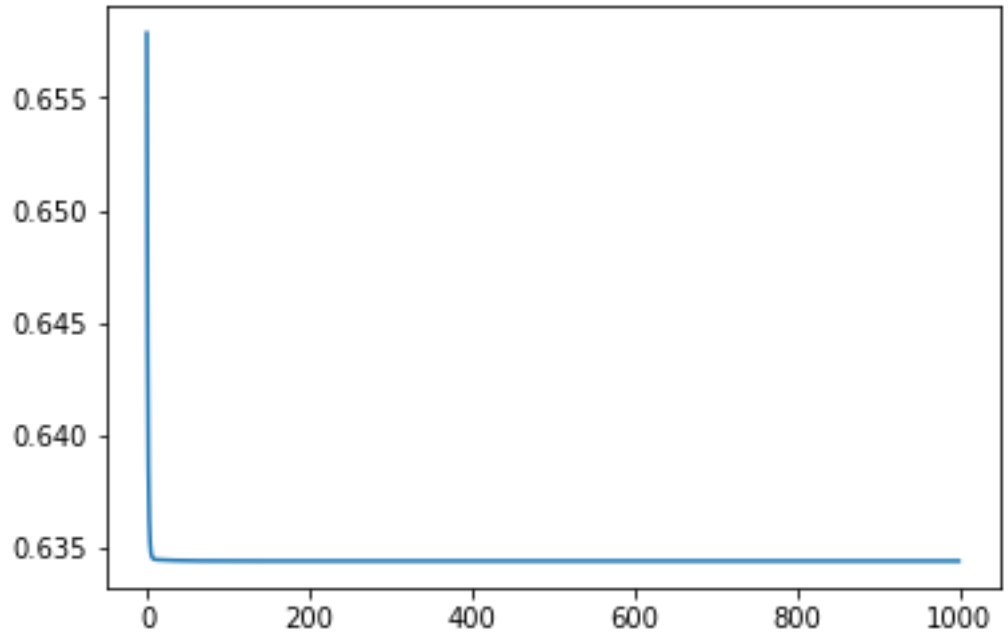
Below, you can observe the Error curve that is decreasing and proves us that training the classifier phase was done correctly:



In the testing phase, the model classified the data with 67% accuracy. The last application of logistic regression classifier in my project is for predicting whether a day is a holiday or not based on Month and Day. You can see the distribution of holidays based on month and day below:



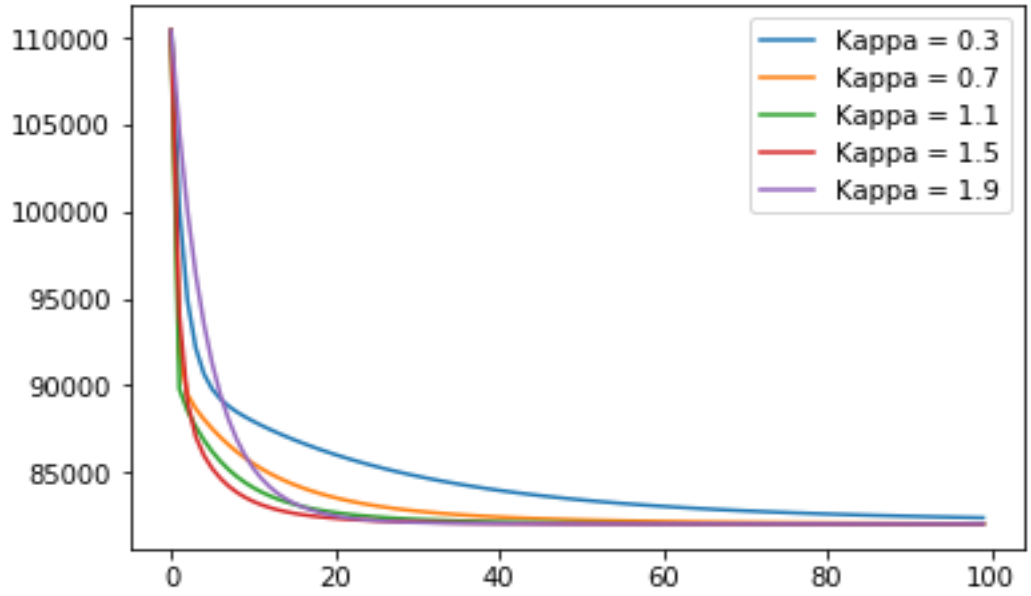
And below, you can see the error curve which is decreasing:



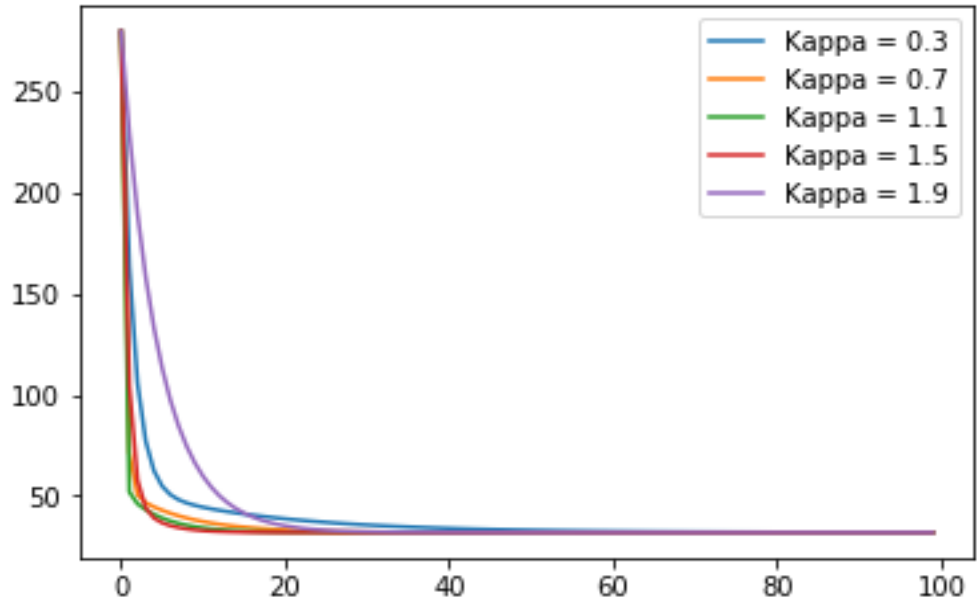
The accuracy of this classifier was 66.9 % on the data.

13 Linear Regression

I used Linear Regression for two applications in this project. I used Python for implementation. All the results, diagrams, and implementations are available in Jupyter Notebook folder. Here, I just explain them shortly, you can see that folder to see everything in details. Firstly, I utilized linear regression in order to predict Number of unauthorized speed based on Number of total vehicles and average of velocity. Below, you can see the error curves for different Kappa values.



The second application of Linear Regression was for predicting temperature average based on Month and day. The error curves correspond to different Kappa values is below:



14 Conclusion & Discussion

By examining the Association Rules, Decision Trees and Correlation Matrix, I can find some causes of congestion in the Tehran Karaj Highway and also I can find the situations which lead to increase in a number of different driving offenses. As this project was a small student one, it needs more endeavor and work in order to reach satisfying results. In the end, I should confess that after this project, I concluded that the most important phase in Data Mining is the Data Cleaning phase because it has a big effect in the output and the results of the project.

15 Acknowledgments

I want to give my deepest appreciations to Professor Alex Thomo, who provided us with this great opportunity to meet this great field of computer science, Data Mining.

16 References

[1] http://www.141.ir/SitePages/index.aspx

[2] https://developer.worldweatheronline.com

[3] https://www.r-bloggers.com/association-rule-learning-and-the-apriori-algorithm/