Capstone Project - Traffic Analysis

Ray Z

1/17/2022

Capstone Project - Traffic Analysis

Introduction

Consistent traffic flow is critical for commuters, travelers, businesses, and accurate metrics of this can be employed by government agencies to improve safety and efficiency in transportation networks. Traffic congestion incurs an opportunity cost among individuals who would otherwise be working; not to mention, it also increases vehicular emissions and degrades the environment. Traffic flow is subject to a variety of factors, including inclement weather, time of day, and the particular day of the year. Using machine learning techniques, we will predict traffic flow using a variety of factors.

The data used in this report is the Metro Interstate Traffic Volume Data Set from the UCI Machine Learning Repository. It contains 48,204 data points on hourly traffic from Minneapolis to St. Paul, MN on westbound I-94, including weather and holiday features, from 2012-2018. Factors/variables in the dataset include:

1. Holiday

- categorical variable for US holidays

2. Temperature

- numerical variable, measured in Kelvin

3. Rain

- numerical variable, measured in millimeter

4. Snow

- numerical variable, measured in millimeters

5. Clouds

- numerical variable, percentage of cloud cover

6. Weather

- categorical variable, with a few categories

7. Weather Description

- categorical variable, short descriptor of weather

8. Date/time

- numerical, ordinal in this context

9. Traffic Volume

- numerical variable, number of cars

The goal of this project is apply several classification models to the traffic data and evaluate their effectiveness using k-fold cross validation given a training and testing set. We will first perform exploratory data analysis, and clean the data. We will normalize the data, and then conduct machine learning algorithms, determining which technique is most accurate. Supervised machine learning algorithms that will be used for this classification problem include:

1. K Nearest Neighbors
2. Decision Tree
3. Multinomial Logistic Regression
4. Neural Networks

Methods/Analysis

We will begin by loading and cleaning the data.

Data Loading

```
# Load packages
if (!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## Warning: package 'tidyverse' was built under R version 4.0.5
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.0     v forcats 0.5.1
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
## Warning: package 'tibble' was built under R version 4.0.5
```

```
## Warning: package 'tidyr' was built under R version 4.0.5
```

```
## Warning: package 'readr' was built under R version 4.0.5

## Warning: package 'purrr' was built under R version 4.0.5

## Warning: package 'dplyr' was built under R version 4.0.5

## Warning: package 'stringr' was built under R version 4.0.5

## Warning: package 'forcats' was built under R version 4.0.5

## -- Conflicts --------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if (!require(R.utils)) install.packages("R.utils", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: R.utils

## Warning: package 'R.utils' was built under R version 4.0.5

## Loading required package: R.oo

## Warning: package 'R.oo' was built under R version 4.0.3

## Loading required package: R.methodsS3

## Warning: package 'R.methodsS3' was built under R version 4.0.3

## R.methodsS3 v1.8.1 (2020-08-26 16:20:06 UTC) successfully loaded. See ?R.methodsS3 for help.

## R.oo v1.24.0 (2020-08-26 16:11:58 UTC) successfully loaded. See ?R.oo for help.

##
## Attaching package: 'R.oo'

## The following object is masked from 'package:R.methodsS3':
##
##     throw

## The following objects are masked from 'package:methods':
##
##     getClasses, getMethods

## The following objects are masked from 'package:base':
##
##     attach, detach, load, save

## R.utils v2.11.0 (2021-09-26 08:30:02 UTC) successfully loaded. See ?R.utils for help.
```

```
##
## Attaching package: 'R.utils'

## The following object is masked from 'package:tidyr':
##
##     extract

## The following object is masked from 'package:utils':
##
##     timestamp

## The following objects are masked from 'package:base':
##
##     cat, commandArgs, getOption, inherits, isOpen, nullfile, parse,
##     warnings
```

```
if (!require(e1071)) install.packages("e1071", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: e1071

## Warning: package 'e1071' was built under R version 4.0.5
```

```
if (!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Warning: package 'caret' was built under R version 4.0.5

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
if (!require(class)) install.packages("class", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: class
```

```
if (!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: lubridate

## Warning: package 'lubridate' was built under R version 4.0.5

##
## Attaching package: 'lubridate'

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

```
library(R.utils)
library(e1071)
library(caret)
library(class)
library(lubridate)

if (!file.exists("traffic.csv")) {
    download.file("https://archive.ics.uci.edu/ml/machine-learning-databases/00492/Metro_Interstate_Tra
        "traffic.csv.gz")
    gunzip("traffic.csv.gz")
}
df <- read.csv("traffic.csv")
```

Let us first examine a few key aspects of the dataset.

```
nrow(df)
```

```
## [1] 48204
```

```
head(df)
```

```
##   holiday   temp rain_1h snow_1h clouds_all weather_main weather_description
## 1    None 288.28       0       0         40       Clouds    scattered clouds
## 2    None 289.36       0       0         75       Clouds       broken clouds
## 3    None 289.58       0       0         90       Clouds     overcast clouds
## 4    None 290.13       0       0         90       Clouds     overcast clouds
## 5    None 291.14       0       0         75       Clouds       broken clouds
## 6    None 291.72       0       0          1        Clear        sky is clear
##             date_time traffic_volume
## 1 2012-10-02 09:00:00           5545
## 2 2012-10-02 10:00:00           4516
## 3 2012-10-02 11:00:00           4767
## 4 2012-10-02 12:00:00           5026
## 5 2012-10-02 13:00:00           4918
## 6 2012-10-02 14:00:00           5181
```

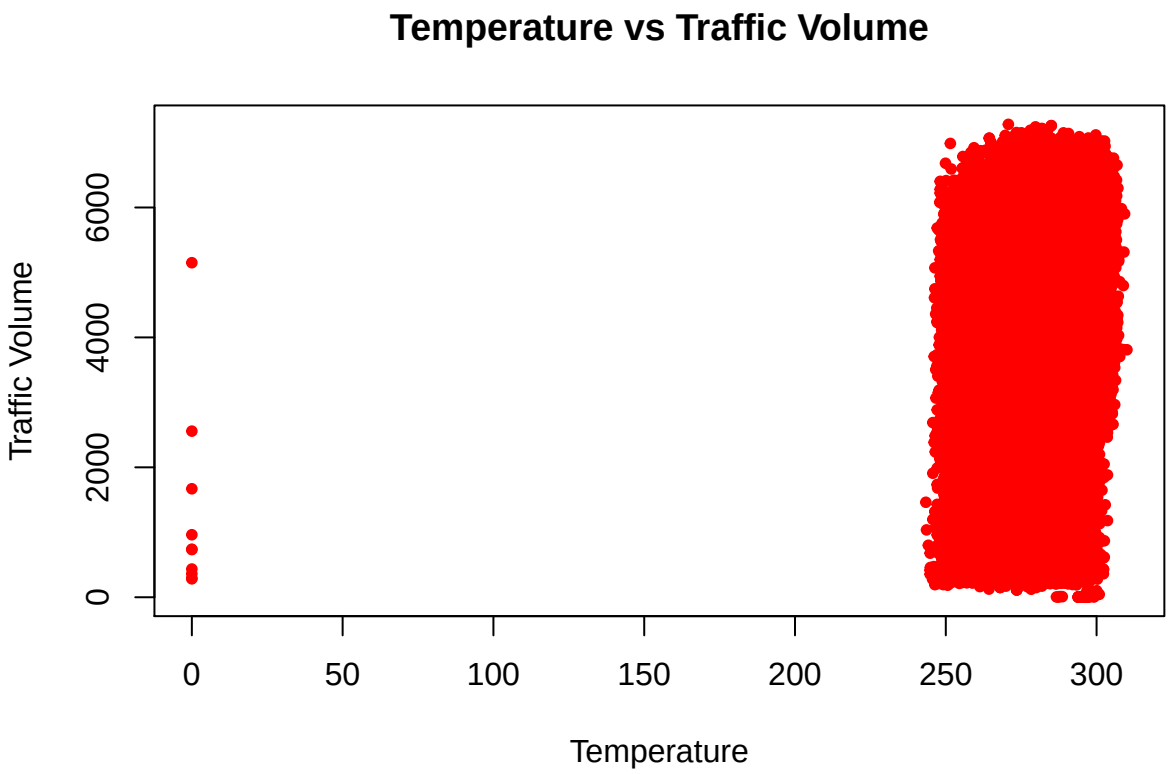
We can see the 9 columns, and 48204 total entries.

Exploratory Data Analysis and Data Cleaning

Let us first generate some simple scatter plots between the numerical predictor variables and the dependent variable which we will be trying to predict (traffic_volume). We can also generate a box and whisker plot for the categorical variable weather_main.

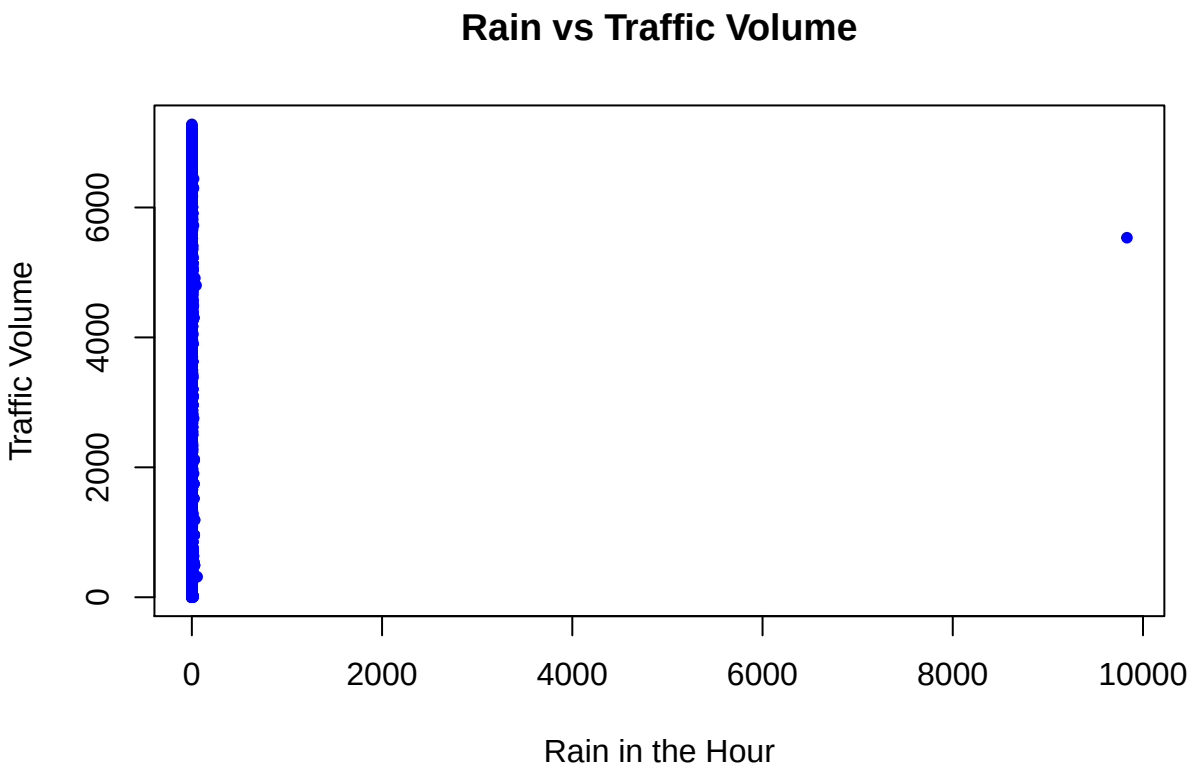
Temperature versus Traffic Volume

```
plot(df$temp, df$traffic_volume, main = "Temperature vs Traffic Volume",
    xlab = "Temperature", ylab = "Traffic Volume", col = "red",
    pch = 20)
```



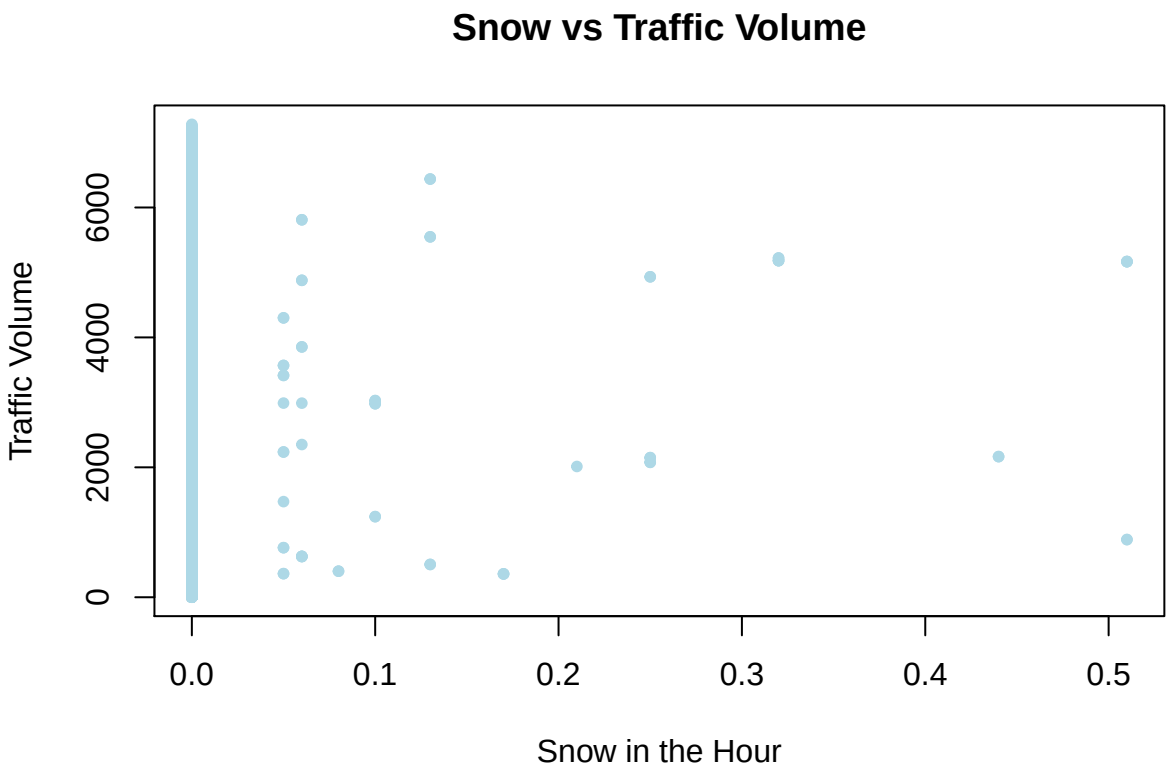
Rain versus Traffic Volume

```
plot(df$rain_1h, df$traffic_volume, main = "Rain vs Traffic Volume",
    xlab = "Rain in the Hour", ylab = "Traffic Volume", col = "blue",
    pch = 20)
```



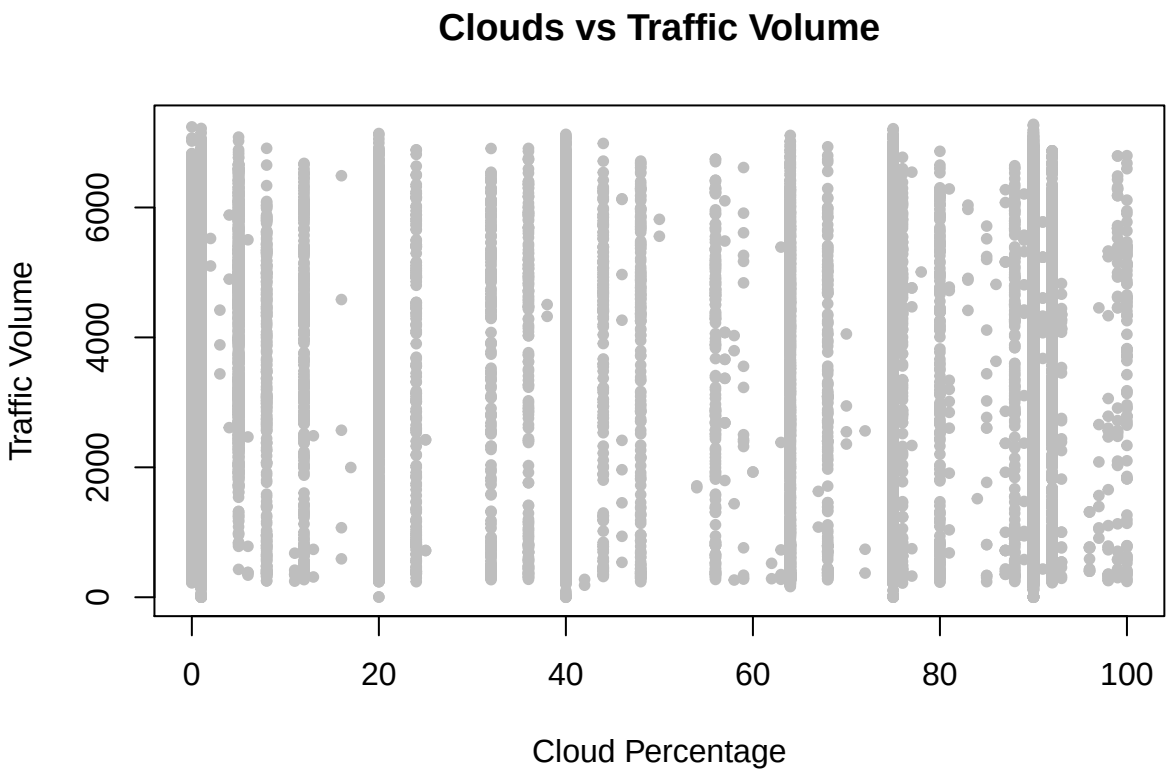
Snow versus Traffic Volume

```
plot(df$snow_1h, df$traffic_volume, main = "Snow vs Traffic Volume",
    xlab = "Snow in the Hour", ylab = "Traffic Volume", col = "#ADD8E6",
    pch = 20)
```



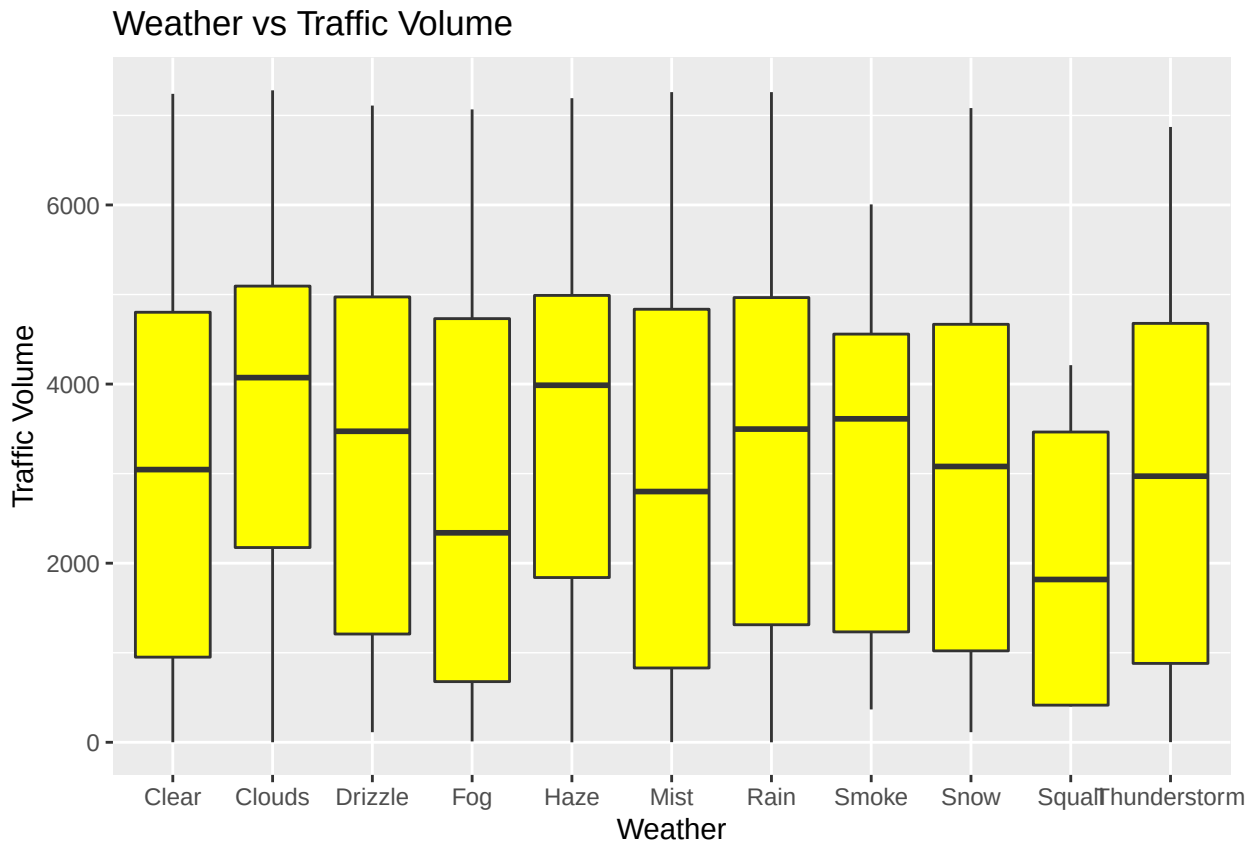
Clouds versus Traffic Volume

```
plot(df$clouds_all, df$traffic_volume, main = "Clouds vs Traffic Volume",
    xlab = "Cloud Percentage", ylab = "Traffic Volume", col = "grey",
    pch = 20)
```



Weather versus Traffic Volume

```
ggplot(df, aes(x = weather_main, y = traffic_volume)) + labs(title = "Weather vs Traffic Volume",
    x = "Weather", y = "Traffic Volume") + geom_boxplot(fill = "yellow")
```



It appears that our scatterplots for temperature and rain are skewed by 11 outliers. These appear to be at temperature 0 kelvin, and one extremely high rain event ~10000mm (10 meters!). We can create a subset without these outliers, and examine them to ensure they are measurement errors and we are not discarding important data.

```
df2 <- subset(df, temp > 200 & rain_1h < 1000)  # remove outliers (kelvin <200 and rain >1000)
outliers <- subset(df, temp < 200 | rain_1h > 1000)  # examine outliers
print(outliers)
```

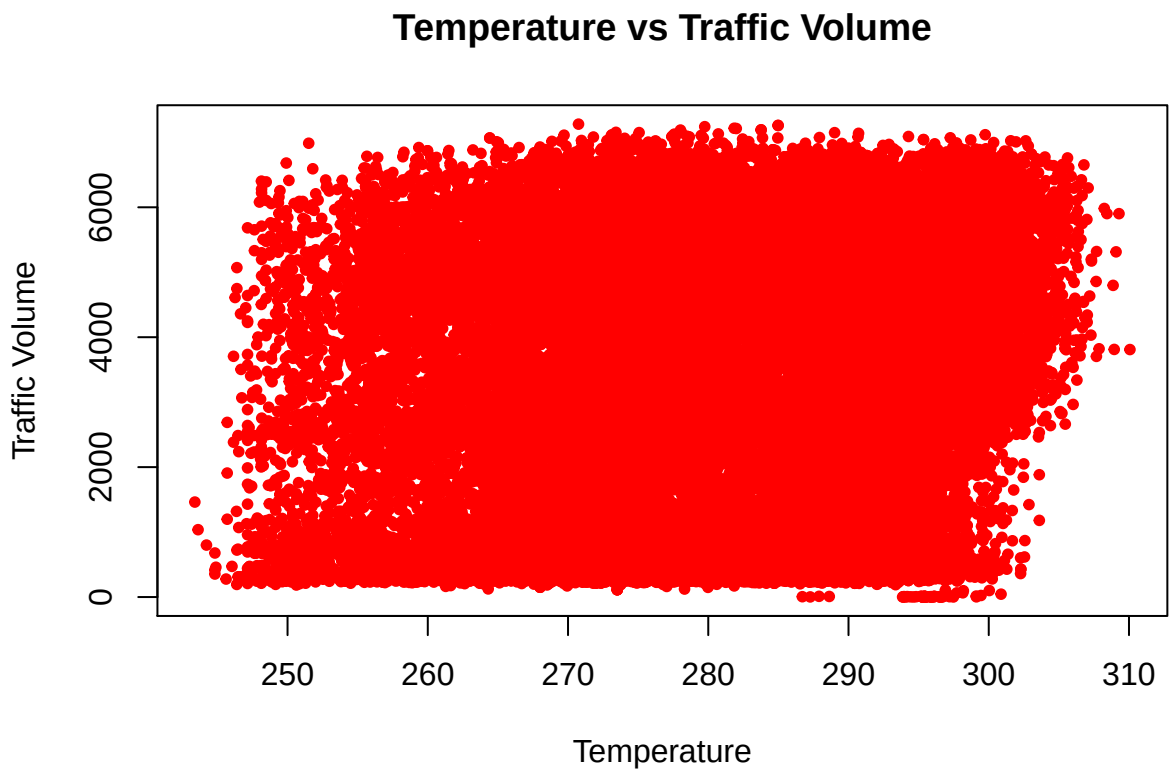
```
##       holiday   temp rain_1h snow_1h clouds_all weather_main
## 11899    None   0.00     0.0       0          0        Clear
## 11900    None   0.00     0.0       0          0        Clear
## 11901    None   0.00     0.0       0          0        Clear
## 11902    None   0.00     0.0       0          0        Clear
## 11947    None   0.00     0.0       0          0        Clear
## 11948    None   0.00     0.0       0          0        Clear
## 11949    None   0.00     0.0       0          0        Clear
## 11950    None   0.00     0.0       0          0        Clear
## 11951    None   0.00     0.0       0          0        Clear
## 11952    None   0.00     0.0       0          0        Clear
## 24873    None 302.11  9831.3       0         75         Rain
##       weather_description           date_time traffic_volume
## 11899        sky is clear 2014-01-31 03:00:00            361
## 11900        sky is clear 2014-01-31 04:00:00            734
## 11901        sky is clear 2014-01-31 05:00:00           2557
## 11902        sky is clear 2014-01-31 06:00:00           5150
```

```
## 11947         sky is clear 2014-02-02 03:00:00            291
## 11948         sky is clear 2014-02-02 04:00:00            284
## 11949         sky is clear 2014-02-02 05:00:00            434
## 11950         sky is clear 2014-02-02 06:00:00            739
## 11951         sky is clear 2014-02-02 07:00:00            962
## 11952         sky is clear 2014-02-02 08:00:00           1670
## 24873     very heavy rain 2016-07-11 17:00:00           5535
```

As we can see, there are 11 outliers, 10 of which are at temperature 0 kelvin (which is unlikely to occur from natural conditions), while one has a rain measurement of 9831.3 mm, or 9.8 meters of rain (also unlikely to occur from natural conditions). Thus, we can use the dataframe with these outliers removed. Graphing the same plots with the outliers removed gives us the following graphs.

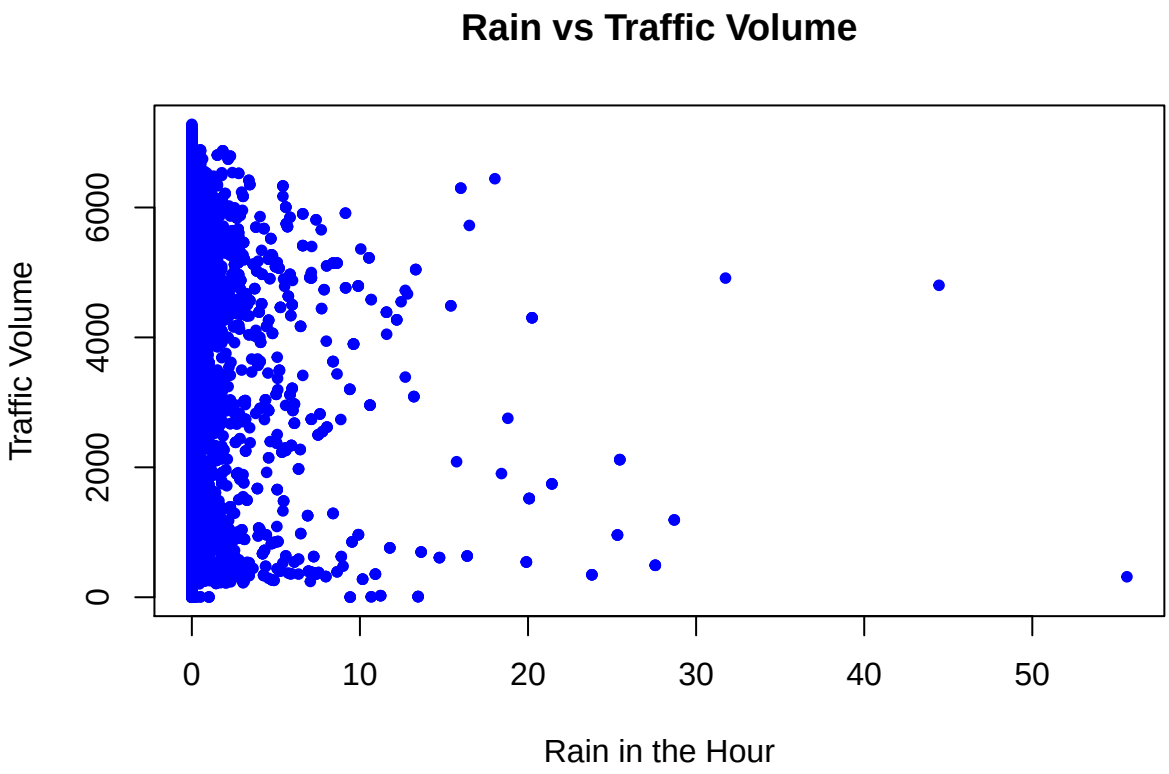
Temperature versus Traffic Volume

```
plot(df2$temp, df2$traffic_volume, main = "Temperature vs Traffic Volume",
    xlab = "Temperature", ylab = "Traffic Volume", col = "red",
    pch = 20)
```



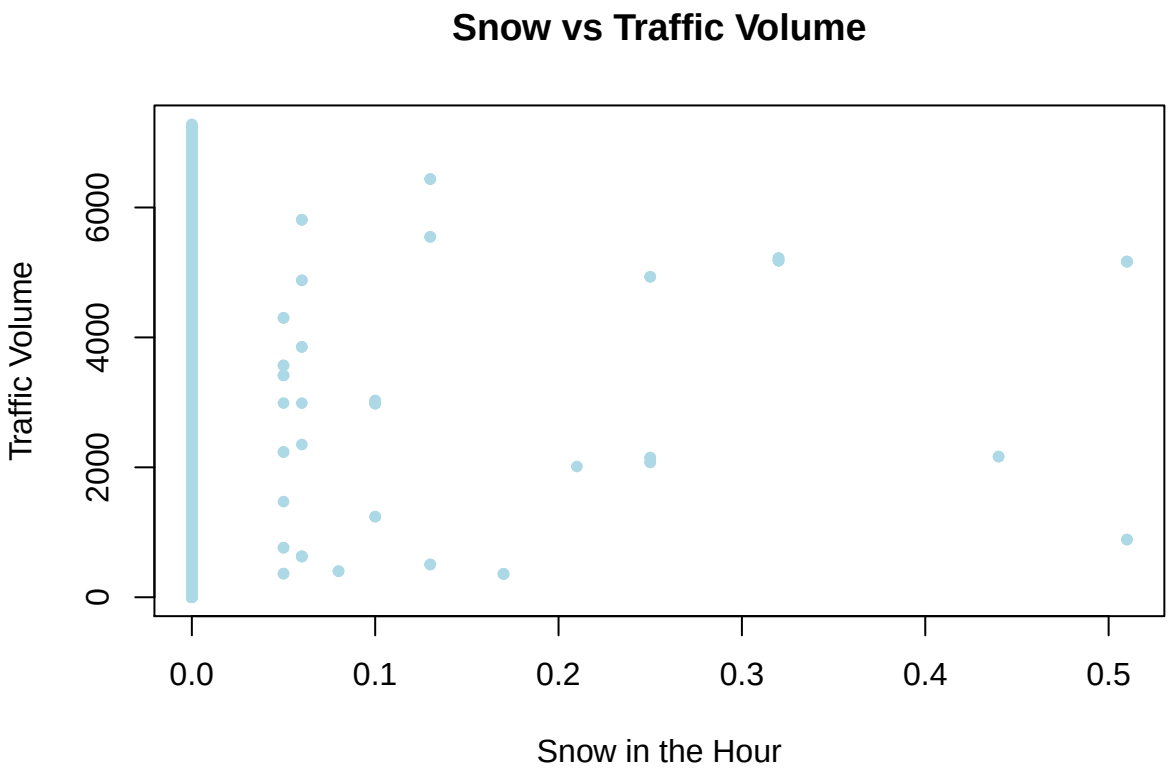
Rain versus Traffic Volume

```
plot(df2$rain_1h, df2$traffic_volume, main = "Rain vs Traffic Volume",
    xlab = "Rain in the Hour", ylab = "Traffic Volume", col = "blue",
    pch = 20)
```



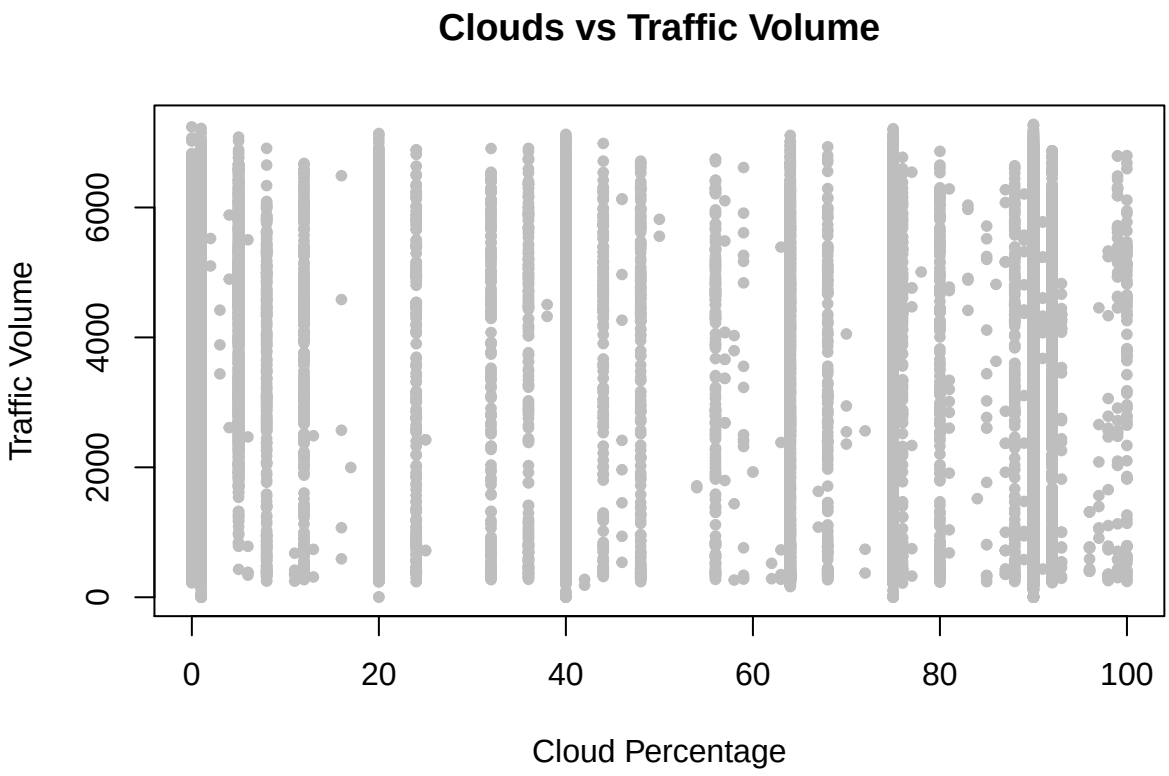
Snow versus Traffic Volume

```
plot(df2$snow_1h, df2$traffic_volume, main = "Snow vs Traffic Volume",
    xlab = "Snow in the Hour", ylab = "Traffic Volume", col = "#ADD8E6",
    pch = 20)
```



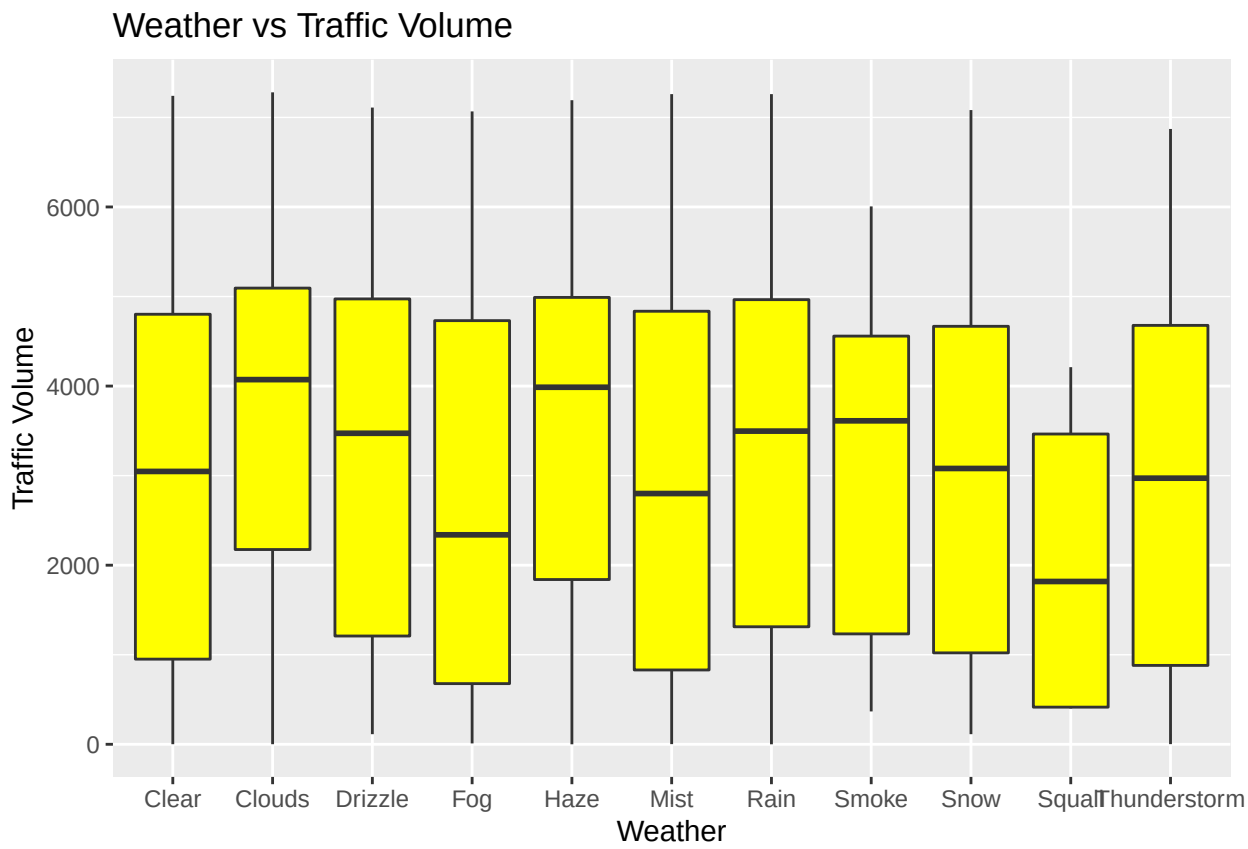
Clouds versus Traffic Volume

```
plot(df2$clouds_all, df2$traffic_volume, main = "Clouds vs Traffic Volume",
    xlab = "Cloud Percentage", ylab = "Traffic Volume", col = "grey",
    pch = 20)
```



Weather versus Traffic Volume

```
ggplot(df2, aes(x = weather_main, y = traffic_volume)) + labs(title = "Weather vs Traffic Volume",
    x = "Weather", y = "Traffic Volume") + geom_boxplot(fill = "yellow")
```

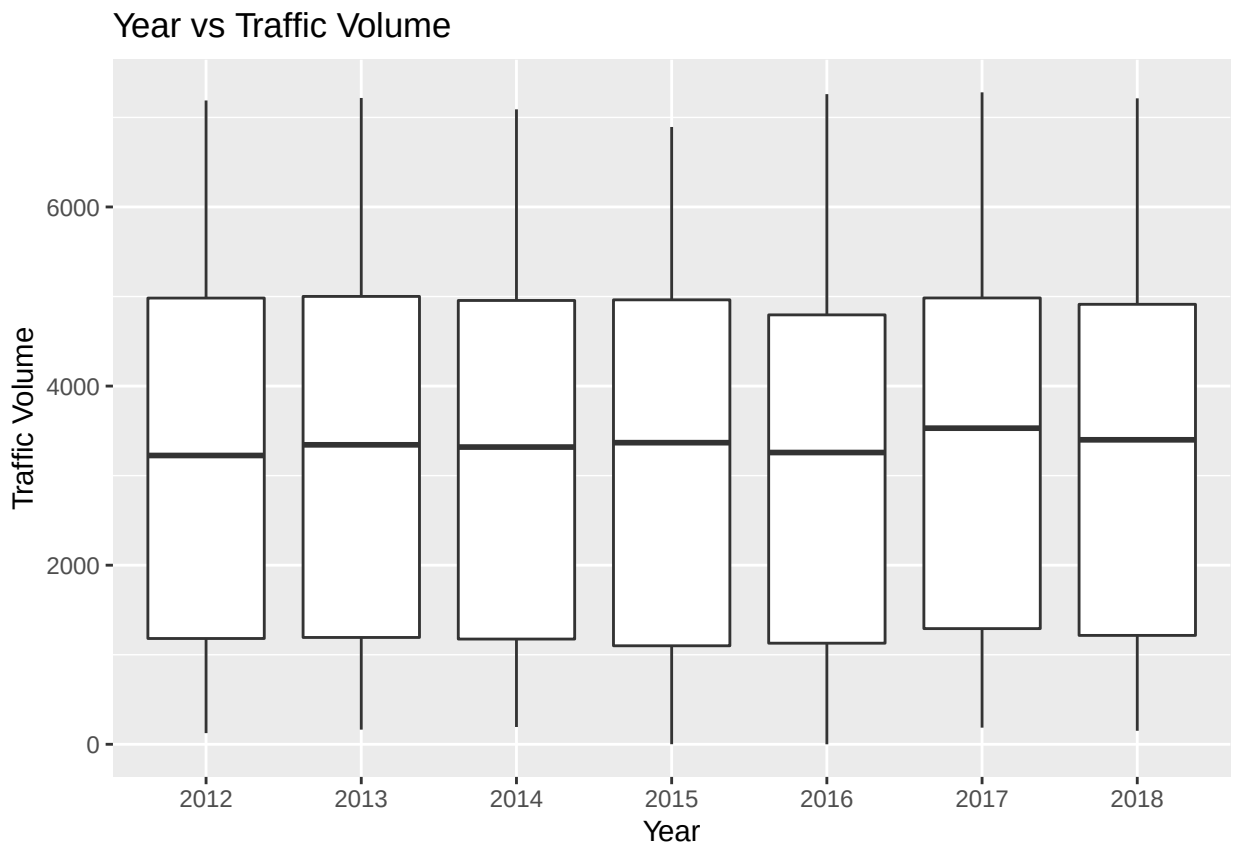


There does not appear to be any significant correlation of features so far. Let us visualize the data with respect for time. We must first parse the date_time column in the df2 data frame.

```
df2$time <- parse_date_time(df2$date_time, "ymd HMS")
df2$year <- df2$time %>%
    year()
df2$month <- df2$time %>%
    month()
df2$day <- df2$time %>%
    day()
df2$hour <- df2$time %>%
    hour()
```

We can first examine the year to year variation.

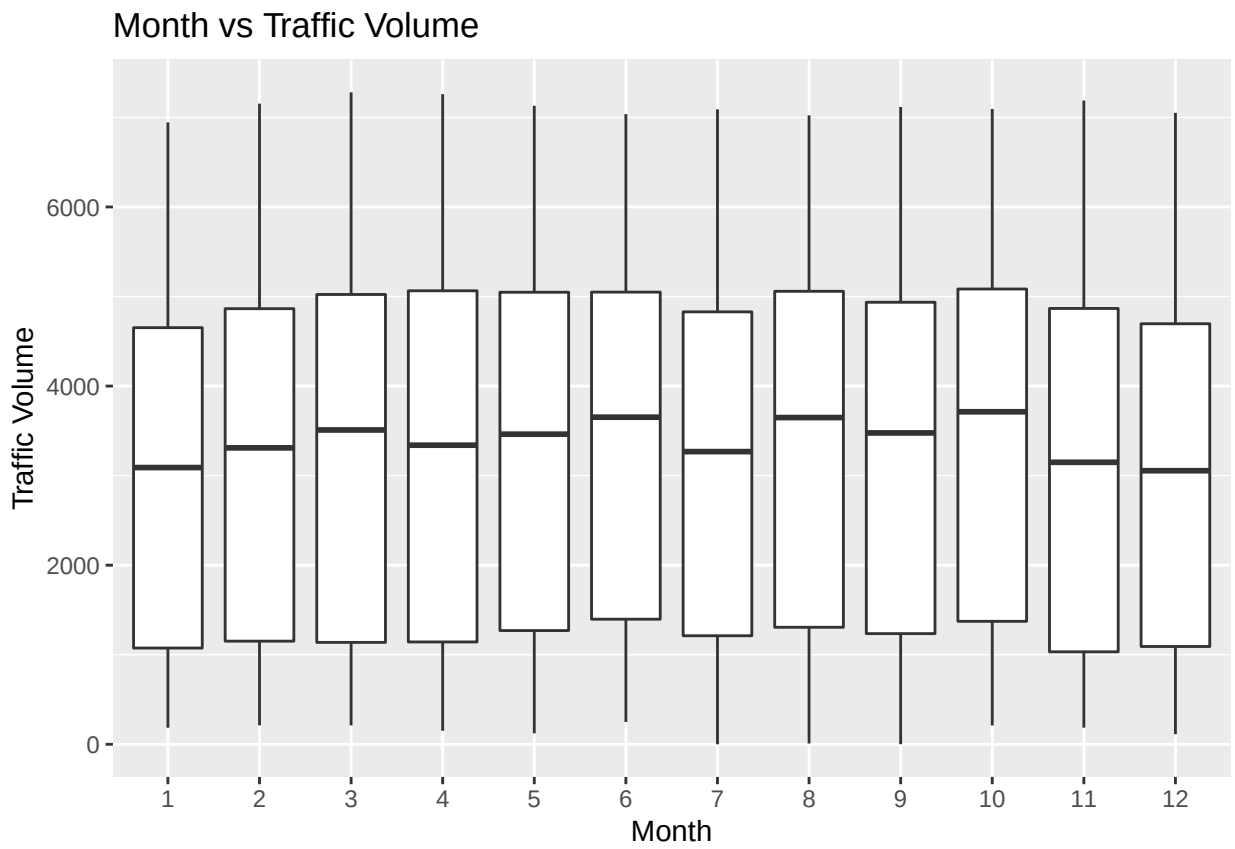
```
ggplot(df2, aes(x = as.factor(year), y = traffic_volume)) + labs(title = "Year vs Traffic Volume",
    x = "Year", y = "Traffic Volume") + geom_boxplot()
```



There appears to be no significant variation in the distribution between years. We will run the algorithm over all the years to account for the whole dataset.

We can also examine the month to month variation.

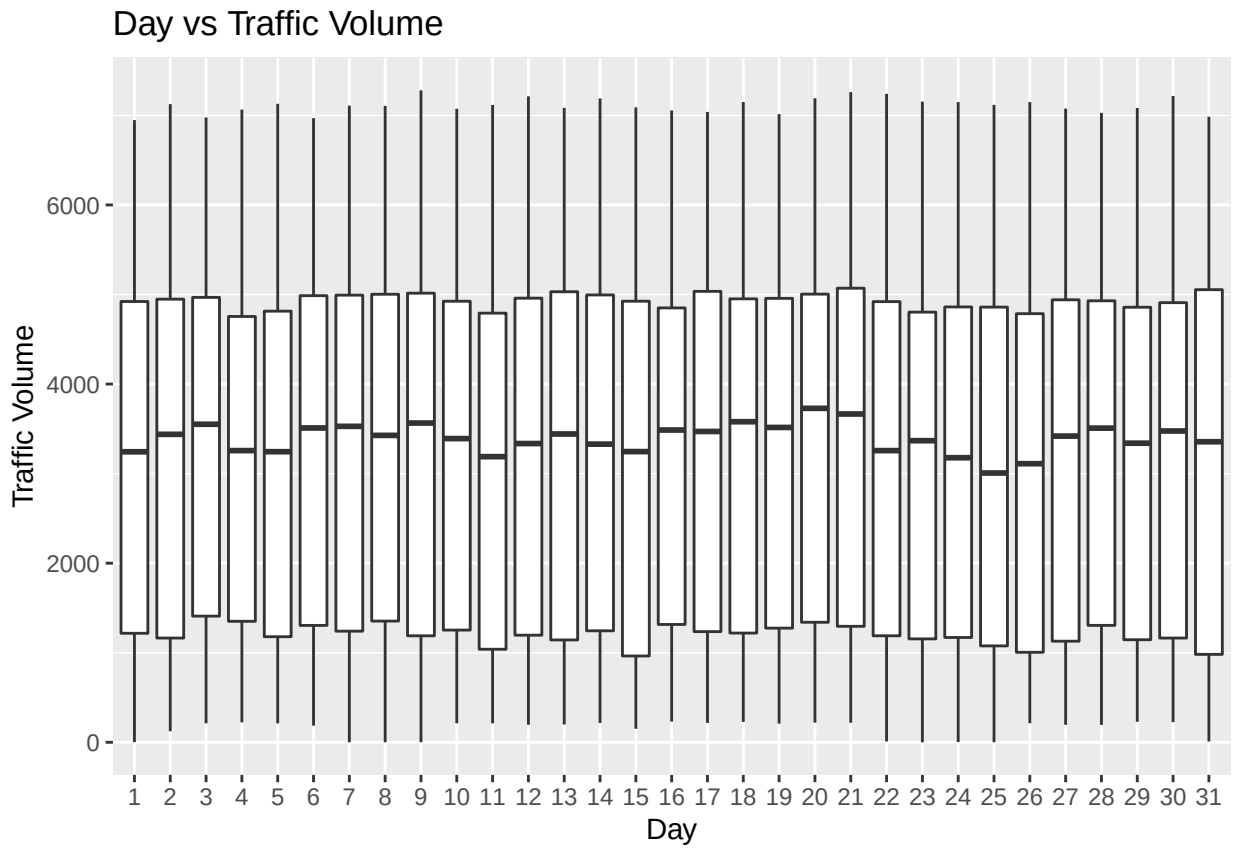
```
ggplot(df2, aes(x = as.factor(month), y = traffic_volume)) +
    labs(title = "Month vs Traffic Volume", x = "Month", y = "Traffic Volume") +
    geom_boxplot()
```



While there is some variation in traffic volume, the means and IQRs are too similar to suggest significant variation of traffic flow based on month. Let's leave this as is for now.

We can also analyze the traffic flow with respect to the day of the month.

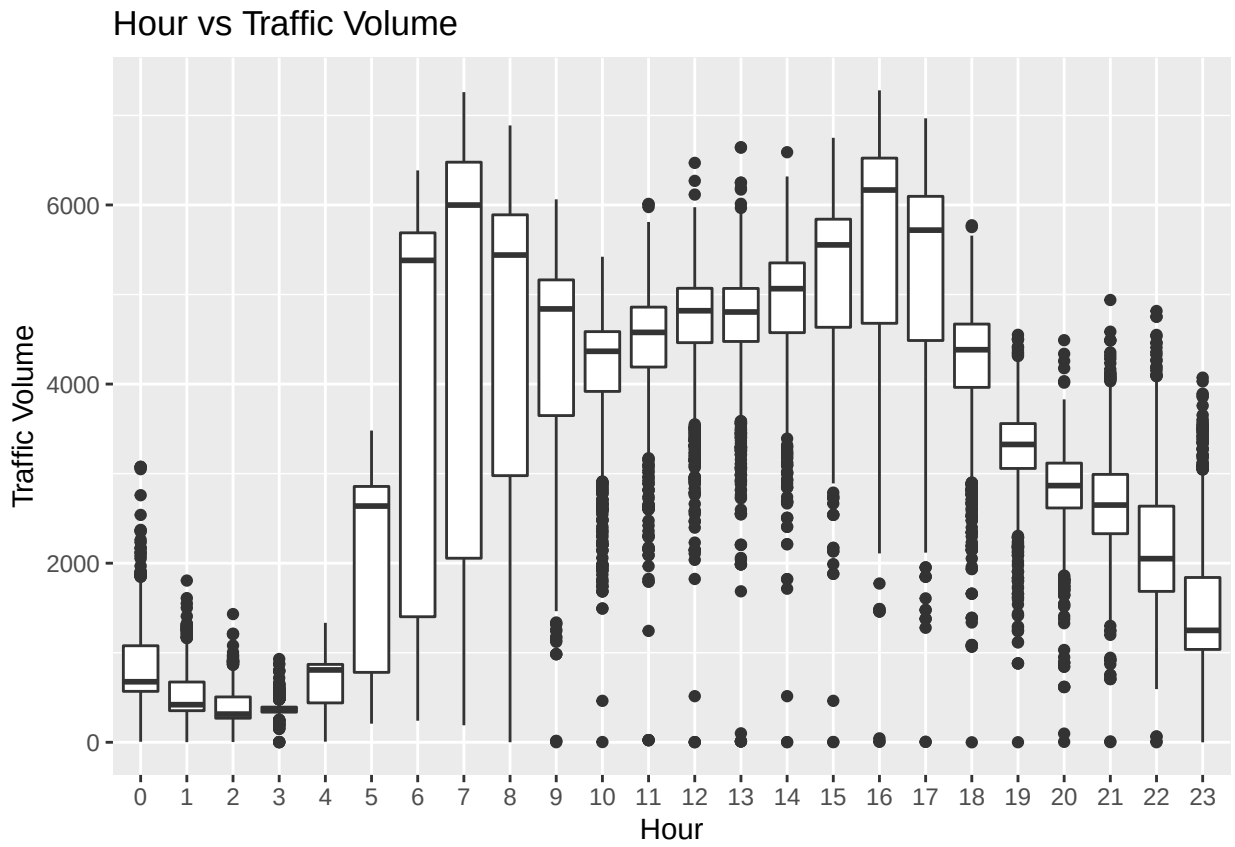
```
ggplot(df2, aes(x = as.factor(day), y = traffic_volume)) + labs(title = "Day vs Traffic Volume",
    x = "Day", y = "Traffic Volume") + geom_boxplot()
```



Again we see a little variation. It could be a possibility that certain days are associated with less traffic volume (for instance, the weekend). However, like the previous graph of month versus traffic volume, the mean and IQRs seem to fall close to one another, making this likely not a critical variable to classify traffic flow.

It seems most likely that the traffic flow will change most with the hour of the day. Let's analyze traffic flow with respect to the hour.

```
ggplot(df2, aes(x = as.factor(hour), y = traffic_volume)) + labs(title = "Hour vs Traffic Volume",
    x = "Hour", y = "Traffic Volume") + geom_boxplot()
```

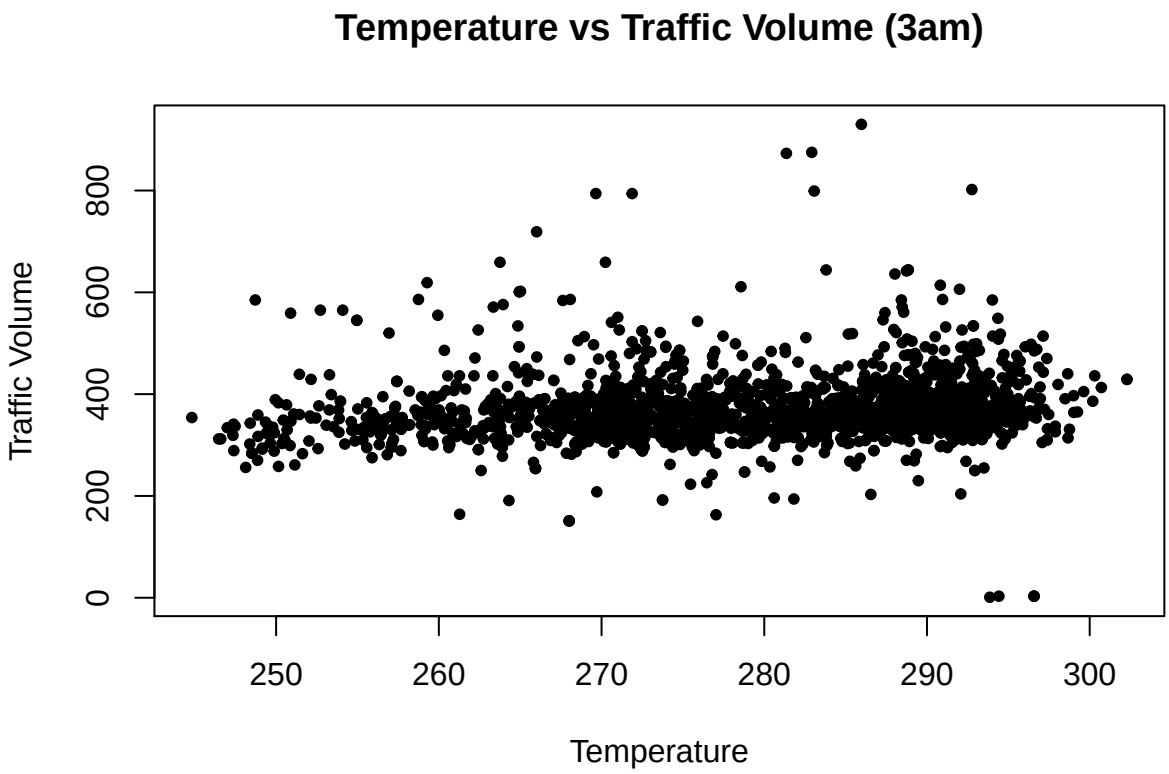


It appears that there is a trend with respect to the hour. During hours in the middle of the day, consistently higher traffic volume values can be seen, while hours at night have lower traffic volume values. The lack of overlap between the distributions of traffic flow between hours, as well as the small span of some distributions suggest that this is a variable to consider when training our models.

We can also analyze the relationship between other variables (temperature, etc) with respect to the traffic volume while holding the hour constant.

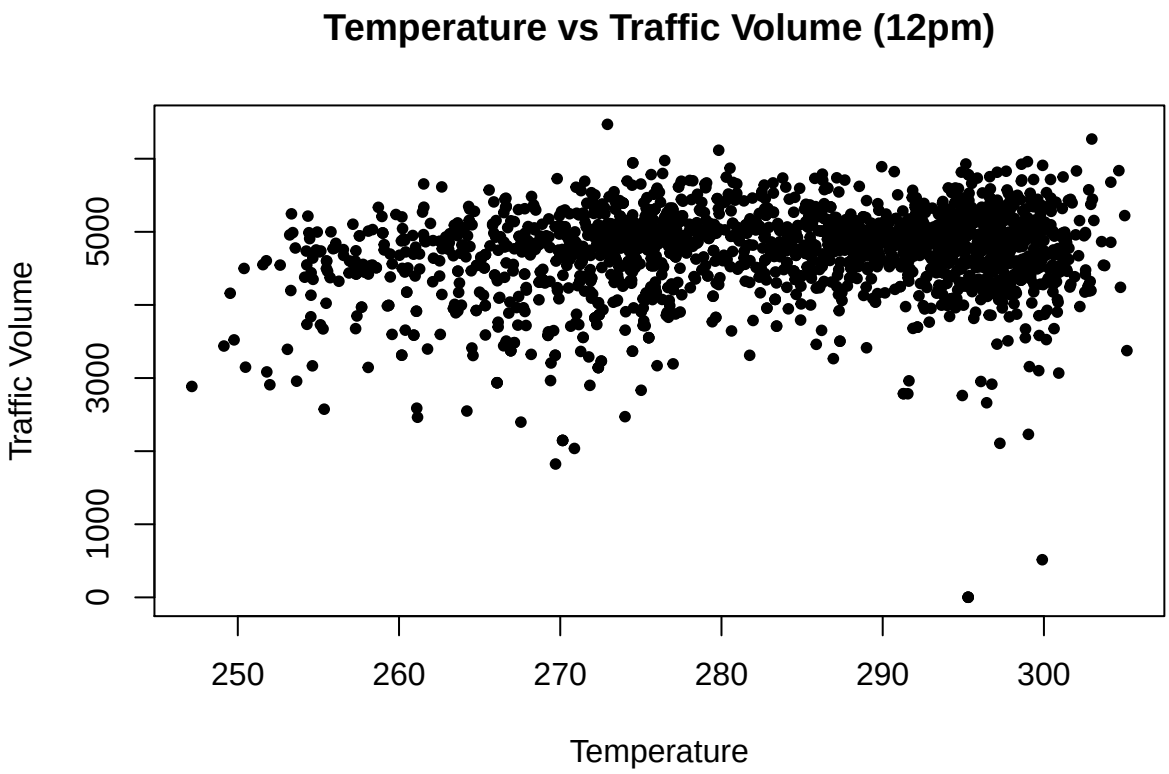
Temperature versus Traffic Volume at 3am

```
df2_hour3 <- subset(df2, hour == 3)
plot(df2_hour3$temp, df2_hour3$traffic_volume, main = "Temperature vs Traffic Volume (3am)",
    xlab = "Temperature", ylab = "Traffic Volume", pch = 20)
```



Temperature versus traffic volume at 12pm

```
df2_hour12 <- subset(df2, hour == 12)
plot(df2_hour12$temp, df2_hour12$traffic_volume, main = "Temperature vs Traffic Volume (12pm)",
    xlab = "Temperature", ylab = "Traffic Volume", pch = 20)
```



Variables such as temperature, can be seen to be positively strongly correlated when holding the hour constant. Thus, a supervised learning algorithm trained on multiple predictor variables should be improved with the inclusion of the hour variable.

Modeling Approach

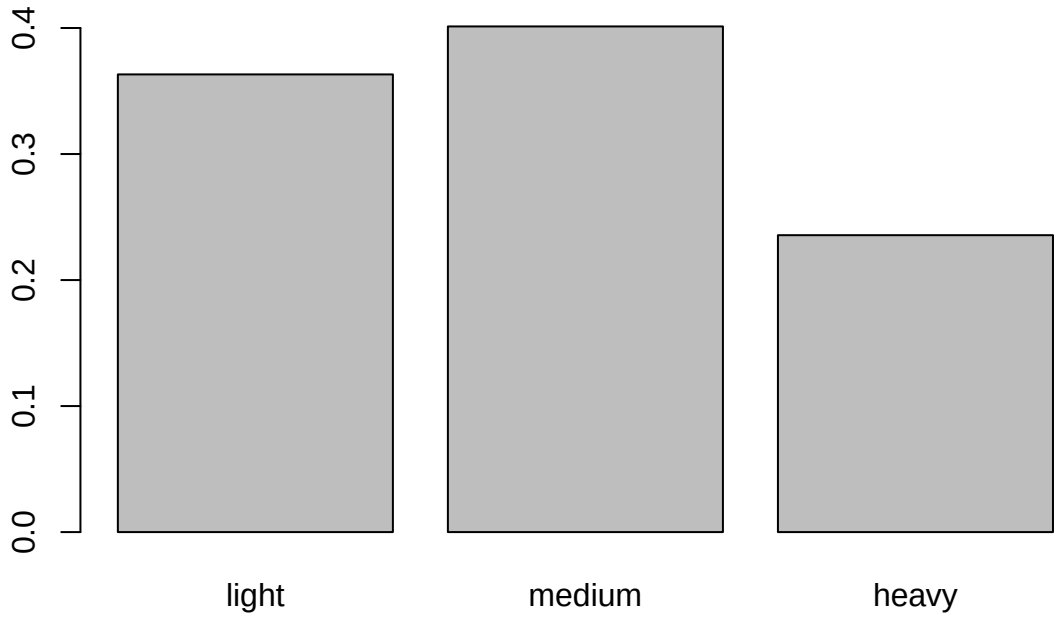
We will treat this as a classification problem, and categorize the traffic_volume column into a few distinct categories representative of the traffic strength. As the current traffic_flow values range from 0 to 7280, we can split this into three bins of width 2500. Our models will classify the traffic flow as one of these three categories, for light, medium, and heavy traffic.

```
range(df2$traffic_volume)
```

```
## [1]    0 7280
```

We see that the traffic_volume ranges from 0 to 7280.

```
df2$traffic <- cut(df2$traffic_volume, c(0, 2500, 5000, 7500),
    labels = c("light", "medium", "heavy"), right = FALSE)
barplot(prop.table(table(df2$traffic)))
```



The data appears roughly even between light and medium traffic, with fewer heavy traffic instances than light and medium.

Preprocess the data (normalize) Before we create a classifier model such as K Nearest Neighbors, it is important that we normalize the data. Normalizing the data for KNN is imperative because KNN relies on Euclidean distance to determine the closest points; without normalization, variables with lower variance would be interpreted as closer, resulting in a biased estimation. Let"s create a normalize() function that will normalize a variable using min-max scaling. We can use lapply to apply this function to each varaible in the data frame that we need normalized. Let"s create a new data frame with this processed data that we will use for the machine learning algorithms.

```
df3 <- subset(df2, select = -c(holiday, weather_main, weather_description,
    year, month, day, time, date_time, traffic_volume, traffic))

normalize <- function(x) {
    # normalize using min-max scaling
    ((x - min(x))/(max(x) - min(x)))
}

processed_df <- as.data.frame(lapply(df3, normalize))

processed_df$traffic <- df2$traffic
```

Subset the data Using the set seed 123 for reproducibility, we can create a train test split. We can use a 90% training set and a 10% testing set as we have many data points and thus the testing set should be

representative of the target we are trying to predict and not be affected by variation. This will allow us to later perform cross validation to analyze the effectiveness of our machine learning algorithms.

```
set.seed(123, sample.kind = "Rounding")
```

```
## Warning in set.seed(123, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
train_index <- sample(1:nrow(processed_df), 0.9 * nrow(processed_df))
training_data <- processed_df[train_index, ]  # training data
testing_data <- processed_df[-train_index, ]  # testing data
```

Model 1 - KNN The K Nearest Neighbors algorithm classifies data given the level of the k nearest neighbors. Let us run this algorithm with all the predictors. Rather than using a single train/test split, we can use repeated k-fold cross validation through trainControl with 10 fold cross validation repeated 5 times.

```
model_1 <- train(traffic ~ ., method = "knn", trControl = trainControl(method = "repeatedcv",
    number = 5, repeats = 10), data = training_data)
```

```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used
```

```
model_1
```

```
## k-Nearest Neighbors
##
## 43373 samples
##     5 predictor
##     3 classes: 'light', 'medium', 'heavy'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 10 times)
## Summary of sample sizes: 34698, 34699, 34698, 34699, 34698, 34697, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.7593157  0.6314181
##   7  0.7632398  0.6378913
##   9  0.7668388  0.6437314
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

Now our K Nearest Neighbors model is trained with k=9, with the relatively high accuracy of 0.7874. Now we can apply this model to the test set to see the model accuracy.

```
knn_pred <- predict(model_1, newdata = testing_data)
cm_1 <- confusionMatrix(knn_pred, testing_data$traffic)
cm_1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction light medium heavy
##     light   1393    167    69
##     medium   154   1530   283
##     heavy    116    302   806
##
## Overall Statistics
##
##                Accuracy : 0.7737
##                  95% CI : (0.7616, 0.7854)
##     No Information Rate : 0.4147
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6534
##
##  Mcnemar's Test P-Value : 0.004458
##
## Statistics by Class:
##
##                      Class: light Class: medium Class: heavy
## Sensitivity                0.8376        0.7654       0.6960
## Specificity                0.9252        0.8451       0.8859
## Pos Pred Value             0.8551        0.7778       0.6585
## Neg Pred Value             0.9154        0.8356       0.9021
## Prevalence                 0.3450        0.4147       0.2402
## Detection Rate             0.2890        0.3174       0.1672
## Detection Prevalence       0.3380        0.4081       0.2539
## Balanced Accuracy          0.8814        0.8052       0.7909
```

```
model_1_accuracy <- cm_1$overall["Accuracy"]
```

The confusion matrix shows that our model accuracy is relatively high.

Model 2 - Decision Tree Our second model will be a decision tree based on recursive partitioning. We will use rpart2, which allows for the optimization of the tree depth. We will pass in a tunegrid parameter with a few values to optmize the tree depth.

```
tunegrid <- expand.grid(maxdepth = c(1, 3, 5, 7, 9, 11))

model_2 <- train(traffic ~ ., method = "rpart2", trControl = trainControl(method = "repeatedcv",
    number = 5, repeats = 10), tuneGrid = tunegrid, data = training_data)
```

```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used
```

```
model_2
```

```
## CART
##
```

```
## 43373 samples
##     5 predictor
##     3 classes: 'light', 'medium', 'heavy'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 10 times)
## Summary of sample sizes: 34698, 34698, 34699, 34698, 34699, 34698, ...
## Resampling results across tuning parameters:
##
##   maxdepth  Accuracy   Kappa
##    1        0.6116708  0.3608714
##    3        0.7522191  0.6277354
##    5        0.7522191  0.6277354
##    7        0.7781108  0.6630361
##    9        0.7781108  0.6630361
##   11        0.7781108  0.6630361
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was maxdepth = 7.
```

```
dt_pred <- predict(model_2, newdata = testing_data)
cm_2 <- confusionMatrix(dt_pred, testing_data$traffic)
cm_2
```

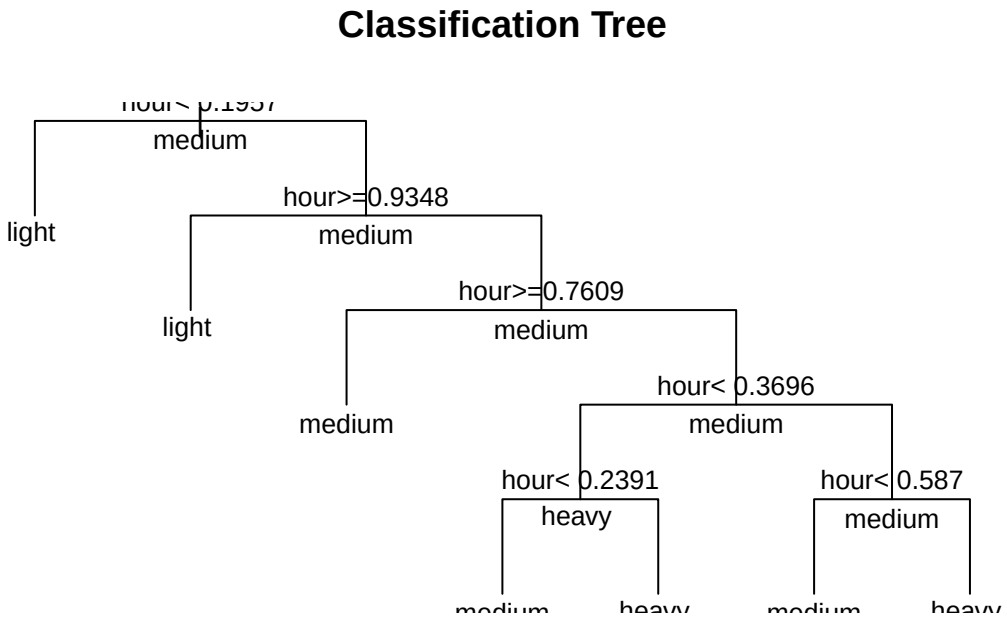
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction light medium heavy
##     light   1308     87     0
##     medium   200   1554   247
##     heavy    155    358   911
##
## Overall Statistics
##
##                Accuracy : 0.7828
##                  95% CI : (0.7709, 0.7944)
##     No Information Rate : 0.4147
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6694
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: light Class: medium Class: heavy
## Sensitivity                0.7865        0.7774       0.7867
## Specificity                0.9724        0.8415       0.8599
## Pos Pred Value             0.9376        0.7766       0.6397
## Neg Pred Value             0.8964        0.8421       0.9273
## Prevalence                 0.3450        0.4147       0.2402
## Detection Rate             0.2714        0.3224       0.1890
## Detection Prevalence       0.2894        0.4151       0.2954
## Balanced Accuracy          0.8795        0.8095       0.8233
```

```
model_2_accuracy <- cm_2$overall["Accuracy"]
```

This accuracy is higher than that of the K Nearest Neighbor algorithm.

We can plot out the splits that the decision tree made, noting which variables were split at what point.

```
plot(model_2$finalModel, uniform = TRUE, main = "Classification Tree")
text(model_2$finalModel, all = TRUE, cex = 0.8)
```



It seems like the hour variable is very significant in classifying the traffic flow.

Model 3 - Multinomial Regression This question is at its core a logistic regression problem. As the dependent variable in this case has multiple levels and is not binary, we can fit a multinomial logistic regression.

```
model_3 <- train(traffic ~ ., method = "multinom", trControl = trainControl(method = "repeatedcv",
    number = 5, repeats = 10), trace = FALSE, data = training_data)
```

```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used
```

```
model_3
```

```
## Penalized Multinomial Regression
##
## 43373 samples
##     5 predictor
##     3 classes: 'light', 'medium', 'heavy'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 10 times)
## Summary of sample sizes: 34699, 34697, 34698, 34699, 34699, 34698, ...
## Resampling results across tuning parameters:
##
##   decay  Accuracy   Kappa
##   0e+00  0.5696885  0.3017817
##   1e-04  0.5696908  0.3017854
##   1e-01  0.5694694  0.3014286
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was decay = 1e-04.
```

```
mn_pred <- predict(model_3, newdata = testing_data)
cm_3 <- confusionMatrix(mn_pred, testing_data$traffic)
cm_3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction light medium heavy
##     light   1233    501   489
##     medium   430   1498   669
##     heavy      0      0     0
##
## Overall Statistics
##
##                Accuracy : 0.5666
##                  95% CI : (0.5525, 0.5806)
##     No Information Rate : 0.4147
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.298
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: light Class: medium Class: heavy
## Sensitivity                0.7414        0.7494       0.0000
## Specificity                0.6864        0.6104       1.0000
## Pos Pred Value             0.5547        0.5768          NaN
## Neg Pred Value             0.8344        0.7746       0.7598
## Prevalence                 0.3450        0.4147       0.2402
## Detection Rate             0.2558        0.3108       0.0000
## Detection Prevalence       0.4612        0.5388       0.0000
## Balanced Accuracy          0.7139        0.6799       0.5000
```

```
model_3_accuracy <- cm_3$overall["Accuracy"]
```

The multinomial logarithmic regression performed not as well as KNN and DT.

Model 4 - Neural Network We can use a simple feed-forward neural network with one hidden layer using the nnet method. This is less computationally advantageous than the previous algorithms, however, we will see how effectively it can make predictions on the test set.

```
model_4 <- train(traffic ~ ., method = "nnet", trControl = trainControl(method = "repeatedcv",
    number = 5, repeats = 10), trace = FALSE, data = training_data)
```

```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used
```

```
model_4
```

```
## Neural Network
##
## 43373 samples
##     5 predictor
##     3 classes: 'light', 'medium', 'heavy'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 10 times)
## Summary of sample sizes: 34699, 34697, 34699, 34698, 34699, 34698, ...
## Resampling results across tuning parameters:
##
##   size  decay  Accuracy   Kappa
##   1     0e+00  0.6124525  0.3623734
##   1     1e-04  0.6124041  0.3622920
##   1     1e-01  0.6047426  0.3508192
##   3     0e+00  0.6664144  0.4747829
##   3     1e-04  0.6699855  0.4776714
##   3     1e-01  0.6874675  0.5100654
##   5     0e+00  0.7088334  0.5495677
##   5     1e-04  0.7236640  0.5703087
##   5     1e-01  0.7416736  0.6004821
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 5 and decay = 0.1.
```

```
nn_pred <- predict(model_4, newdata = testing_data)
cm_4 <- confusionMatrix(nn_pred, testing_data$traffic)
cm_4
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction light medium heavy
##     light   1389    165     0
```

```
##     medium   164   1580   527
##     heavy    110    254   631
##
## Overall Statistics
##
##                Accuracy : 0.7469
##                  95% CI : (0.7344, 0.7591)
##     No Information Rate : 0.4147
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6068
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: light Class: medium Class: heavy
## Sensitivity                0.8352        0.7904       0.5449
## Specificity                0.9477        0.7551       0.9006
## Pos Pred Value             0.8938        0.6957       0.6342
## Neg Pred Value             0.9161        0.8356       0.8622
## Prevalence                 0.3450        0.4147       0.2402
## Detection Rate             0.2882        0.3278       0.1309
## Detection Prevalence       0.3224        0.4712       0.2064
## Balanced Accuracy          0.8915        0.7727       0.7228
```

```
model_4_accuracy <- cm_4$overall["Accuracy"]
```

The Neural Network also performed well, similar to the K Nearest Neighbors algorithm and the Decision Tree. However, the computational complexity and runtime are significantly higher than the other algorithms.

Results

The following are our model results.

```
tab <- matrix(c(model_1_accuracy, model_2_accuracy, model_3_accuracy,
    model_4_accuracy), ncol = 1, byrow = TRUE)
colnames(tab) <- "Accuracy"
rownames(tab) <- c("K Nearest Neighbors", "Decision Tree", "Multinomial Regression",
    "Neural Network")
tab <- as.table(tab)
tab
```

```
##                          Accuracy
## K Nearest Neighbors     0.7736515
## Decision Tree           0.7827801
## Multinomial Regression 0.5665975
## Neural Network          0.7468880
```

A Decision Tree with 5 levels performed the highest, with K Nearest Neighbors and Neural Network close behind. Multinomial regression performed significantly worse. Accuracy is a simple metric of model performance, calculated as (true positives + true negatives)/number of elements. When one target classification

outweighs the others, such as a disease with 99% negative population cases, a test that only returns negative would have a very high accuracy but not be useful. In our case, with almost equal categories of low, medium, and high traffic, accuracy is a useful metric of how well the model performs. A 5 level Decision Tree performed the highest by this metric.

Conclusions

This project was a classification problem in which we classified traffic flow (a numerical value from 0 to 7280) into three categories (light, medium, and heavy) and predicted this using K Nearest Neighbors, Decision Tree, Multinomial Regression, and Neural Network algorithms. Out of the four supervised learning methods we selected, Decision Tree with 5 levels performed the best on the testing set. K Nearest Neighbors and Neural Network performed similarly well, with K Nearest Neighbors performing slightly better than the Neural Network. Multinomial regression performed significantly worse than these two. Overall, we were able to generate fairly accurate predictions of traffic volume based on five variables: temperature, rainfall, snowfall, cloud coverage, and hour of day.

As aforementioned, predicting traffic volume is vital for all who use public roadways, including commuters, travelers, and businesses. Having this metric allows traffic to be better regulated for safety, efficiency, and environmental impact. This project could possibly have been limited by its scope. The Minneapolis-St. Paul, Minnesota area might be very distinct from other locations in the United States and the world due to its weather, vehicle options, driver characteristics, industries, and much more; an application of this model could overfit when applied to other geographical areas or simply due to random variance as the model follows the data too closely. Future work could examine this problem as a regression problem, predicting the exact traffic flow based on the variables and quantifying the error using MAE or RMSE. We also didn't delve much into the time aspect of the issue, in particular the holidays. This could be integer encoded or one hot encoded as a predictor variable to determine the traffic flow.