Telematics/GPS data

Amos Natido

10/14/2020

```
pacman::p_load(data.table,
               tidyverse,
               mlr, # for Model building
               lubridate,
               ggmap, # Plotting spatial data
               geosphere, # for Haversine distance
               kableExtra,
               factoextra
               )
```

```
#' Setting Working directory path and loading data
path <- getwd() %>%
        setwd()

#' GPS data
gps <- fread("data/sample_trips.csv") %>%
            as.data.frame()


#' Coercing local_dtm to POSIXct time
gps <- gps %>%
              mutate(date.time = dmy_hms(local_dtm))

#' Checking for duplicates
# gps[duplicated.data.frame(gps)]

# Removing duplicates
gps <- gps %>%
distinct(longitude, latitude, .keep_all = TRUE)
#' Structure of the data
glimpse(gps)
```

```
## Rows: 8,483
## Columns: 5
## $ trip_nb   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ local_dtm <chr> "19MAY17:07:33:59", "19MAY17:07:34:01", "19MAY17:07:34:02...
## $ latitude  <dbl> 40.03795, 40.03818, 40.03835, 40.03845, 40.03855, 40.0388...
## $ longitude <dbl> -83.07134, -83.07127, -83.07130, -83.07127, -83.07124, -8...
## $ date.time <dttm> 2017-05-19 07:33:59, 2017-05-19 07:34:01, 2017-05-19 07:...
```

Part 1 (GPS data)

The steps used to clean the GPS data was based on the six dimensions of data quality.

(a) **Accuracy**: The accuracy of measurements of the variables were checked and it appears the decimal places of the instances of the 'longitude' and 'latitude' variables are not consistent. The decimal places vary from 2 to 6 and this may be an indication measurement errors with the GPS device. This was remedied via linear interpolation.

(b) **Completeness**: The data was checked for missing values. There were not cases of empty cells however, there were some gaps in the rows. These gaps may be ascribed to loss in connectivity of the GPS tracking device. These created time series issues since the instances of data is assumed to be recorded at regular time interval of 1 secs. There cases which had a record time of 2 secs. This was remedied using linear interpolation to obtain a complete row of data

(c) **Consistency**: As discussed above under the accuracy. The decimal places of both longitude and latitude variables were not consistent.

(d) **Timeliness**: The gaps in the data created due to connectivity problem also leads to the issue of timeliness. Thus the data is somehow not up to date.

(e) **Uniqness**: There were some duplicate instances based on longitude and latitude variable measurements and these were removed.

(f) **Validity** There appeared to outliers or anomalies due to invalid records in some of the fields. These were detected using the density based spatial clustering with noise technique (DBSCAN).

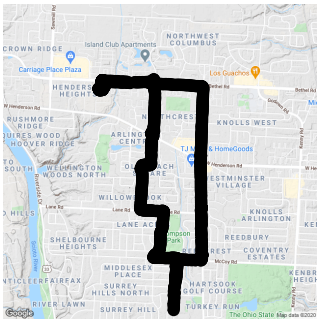
Spatial data visualization

Below we plot and visualize the trajectory of the various trips on a google map to see the details and excat locations of the moving vehicle. There appear to gaps in the path of some of the trips.

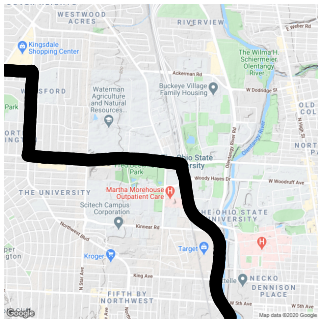
```
# Registering for a Google API to fetch map
register_google(key = "AIzaSyCl2syYwf7SBAx2rHuiVg1DMTDNikAe95E")

map_plot_func = function(t_n, df, x = "longitude", y = "latitude"){
                      dat <- df %>% filter(trip_nb == t_n)
                      lon <- median(as.matrix(dat[[x]]))
                      lat <- median(as.matrix(dat[[y]]))
                      map_g <- get_map(location = c(lon, lat), maptype = "terrain",
                      source = "google",
                      crop = TRUE,
                      scale = 1,
                      zoom = 14)

ggmap(map_g, base_layer = ggplot(dat, aes(x = .data[[x]], y = .data[[y]]))) +
                          geom_point() +
                          ggtitle(paste("Trip", t_n)) +
                          theme(axis.text.x = element_blank(),
                          axis.ticks.x = element_blank(),
                          axis.text.y = element_blank(),
                          axis.ticks.y = element_blank())
}
```

```
cowplot::plot_grid(map_plot_func(t_n = 1, gps), map_plot_func(t_n = 2, gps),
                    map_plot_func(t_n = 3, gps), map_plot_func(t_n = 4, gps))
```

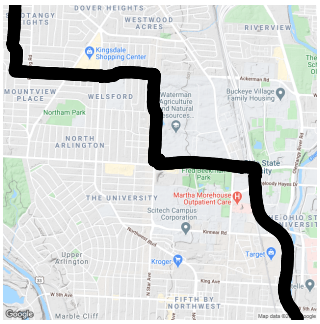
Trip 1



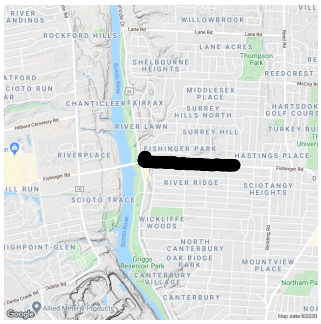
Trip 2



Trip 3

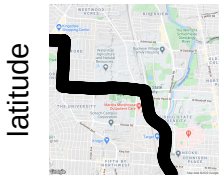


Trip 4

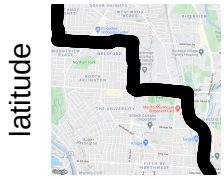


Trip 1

Trip 2



Trip 3



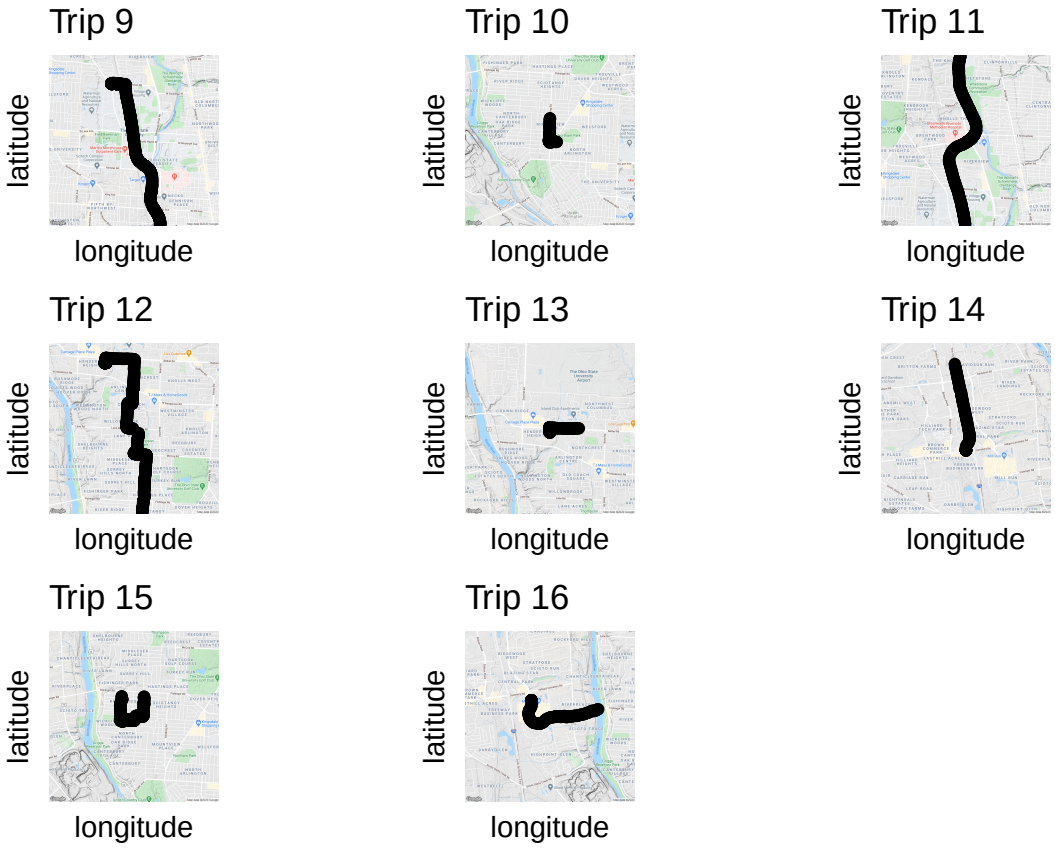
Trip 4

Trip 5

Trip 6

Trip 7

Trip 8

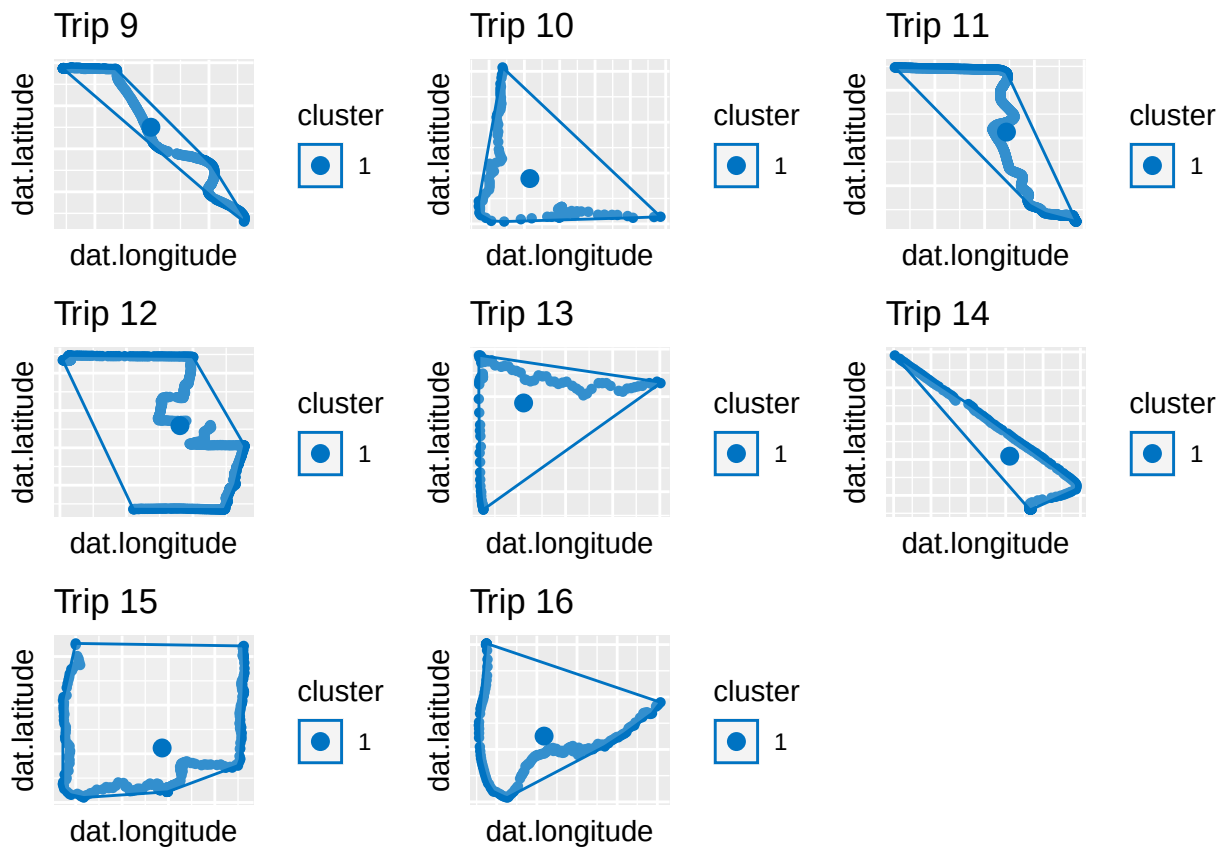


Outlier/Anomaly Detection

The observed gaps in the path in some of the trips calls for the need to perform some further exploratory analysis to detect some outliers. This was done using the Density-based spatial clustering of applications with noise (DBSCAN). The bivariate spatial analysis of the longitude and latitude measurements indicates that trip number 1 and 8 have outliers that need to considered. These observations appeared to be inconsistent, invalid and cannot be explained thus were removed. Missing examples were then imputed using linear interpolation.

```
# Anomaly detection with DBSCAN
Dbscan_func = function(d.f, tn){
                dat <- d.f %>% filter(trip_nb == tn)
                db <- fpc::dbscan(as.matrix(dat$longitude, dat$latitude),
                           eps = 0.15, MinPts = 5)

fviz_cluster(db, data.frame(dat$longitude, dat$latitude),
             stand = FALSE, elipse = FALSE, geom = "point" ,palette = "jco") +
ggtitle(paste("Trip", tn)) +
theme(axis.text.x = element_blank(),
axis.ticks.x=element_blank(),
axis.text.y = element_blank(),
axis.ticks.y = element_blank())
}
```


Trip 1
Trip 2
Trip 3
Trip 4
Trip 5
Trip 6
Trip 7
Trip 8
dat.latitude
dat.longitude
cluster
1




```
## Removing anomalous points from trip 1 and 8 and replacing with NA
#' Adding a row_index as column
gps <- gps %>%
       mutate(row_index = as.numeric(row.names(gps)), .before = trip_nb)

#' Function to obtain row indices of outliers
anomalous_indices_func <- function(tri.no, df){
                                trip_df <- df %>% filter(trip_nb == tri.no)
                                trip_dbscan <- fpc::dbscan(as.matrix(trip_df$longitude,
                                 trip_df$latitude), eps = 0.01, MinPts = 10)

                           ind <- which(trip_dbscan$cluster == 0) # indices of outliers
                           row.indices <- trip_df[ind, ]$row_index
                           return(row.indices)
                            }

anomalous_points <- c(anomalous_indices_func(1, gps), anomalous_indices_func(8,gps))

#' Replacing anomalous points with NA
gps_df <- gps %>%
mutate(longitude = replace(longitude, anomalous_points , NA),
latitude = replace(latitude, anomalous_points, NA)
)
```

```
# Interpolating missing instances to fill gaps
gps_1 <- gps_df %>%
         select(trip_nb, longitude, latitude, date.time)
```

```
interpo_func <- function(var,dt) approx(dt,var,xout=seq(min(dt),max(dt), by = 1))$y
                gps_interpo <- setDT(gps_1)[,lapply(.SD,interpo_func,
                                dt = date.time), by = list(trip_nb),
                               .SDcols = c("latitude","longitude","date.time")]
```

Distance, Speed and Acceleration

Using the cleaned data set, we calculate distance using the Haversine formula given as follows:

$$a = \sin^2(\Delta\phi/2) + \cos(\phi_1)\cos(\phi_2)\sin^2\Delta\lambda/2$$

$$c = 2a\tan 2(\sqrt{a}, \sqrt{(1-a)})$$

$$d = Rc$$

where d is in meters, ϕ is latitude in radians, λ is longitude in radians,$R = 6,371$ km is the earth's radius. The Haversine formula calculates the great-circle distance between two points – that is, the shortest distance over the earth's surface. Speed, s $= \frac{d}{t}$ and acceleration $= \frac{\Delta s}{t}$. Here, $t = 1$ sec.

```
get.dist <- function(lon, lat) distHaversine(tail(cbind(lon,lat),-1), p1 =
                    head(cbind(lon,lat),-1))

# Calculating distances
gps_interpo_df <- gps_interpo %>%
                  group_by(trip_nb) %>%
                  mutate(date.time = as.POSIXct(date.time,
                        origin = "1970-01-0100:00:00", tz = "UTC"),
                        time.diff = as.numeric(difftime(date.time,
                                    lag(date.time, default = first(date.time)),
                                    units = "secs")))

# Calculating distance, velocity and acceleration
gps_interpo_df_vel <- gps_interpo_df %>%
                      group_by(trip_nb) %>%
                      mutate(distance = c(0, get.dist(longitude,latitude)), velocity = 3.6*distance,
                      acceleration = c(0,diff(velocity,1)))
```

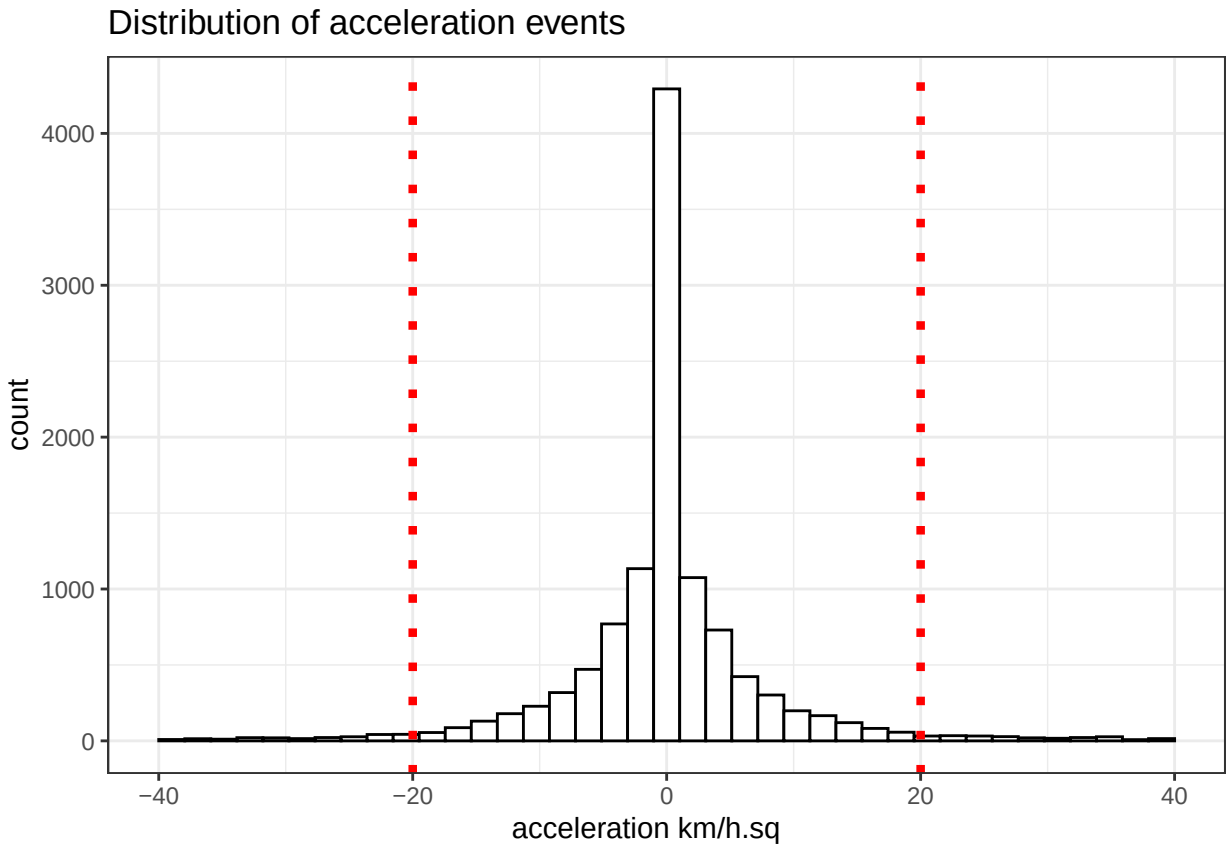
2. Thresholds for hard events

Thresholds for the hard events were set using the criteria for determining outliers. That is, Threshold for hard acceleration $= Q_4 + 1.5 \times \text{IQR} = 5.592681\, m/s^2$, Threshold for hard acceleration $= Q_4 + 1.5 \times \text{IQR} = -5.755827\, m/s^2$ and Threshold for idling $= 0\, m/s^2$. Where, Q_2 and Q_3 represent the second and third quantile of the distribution of acceleration. The distribution of acceleration event is shown below

```
#' Determining Thresholds of Hard Events

#' Threshold for hard brakes
thresh_brake <- fivenum(gps_interpo_df_vel$acceleration)[2] -
                1.5*IQR(gps_interpo_df_vel$acceleration)
```

```r
#' Threshold for hard acceleration
thresh_acc <- fivenum(gps_interpo_df_vel$acceleration)[4] +
              1.5*IQR(gps_interpo_df_vel$acceleration)



ggplot(gps_interpo_df_vel, aes(x = acceleration)) +
  geom_histogram(bins = 40, colour="black", fill="white") +
  labs(title = "Distribution of acceleration events", x  = " acceleration km/h.sq      ")+
  theme_bw()+
  xlim(-40,40) +
   geom_vline(xintercept = c(-20,20), linetype="dotted",
                color = "red", size=1.5)
```

```
## Warning: Removed 147 rows containing non-finite values (stat_bin).
```



3. Summary of Trip events

The table below summarizes the events per trip. The events include the distance travelled, number of hard accelerations, hard brakes and idling.

```r
Events_summary <- gps_interpo_df_vel %>%
                  group_by(trip_nb) %>%
                  summarise(Distance = round(sum(distance),0),
                            HardAccelerations = sum(acceleration > 20),
                            HardBrakes = sum(acceleration < -20),
```

Table 1: Summary of Events Per Trip

Trip No.	Distance (meters)	Hard Accelerations	Hard Brakes	Idlings
1	11006	83	73	30
2	10005	15	13	4
3	12175	16	12	25
4	1459	4	5	6
5	9586	16	10	29
6	9144	24	23	5
7	1806	2	1	47
8	21868	30	23	36
9	4792	12	12	17
10	1032	12	11	1
11	16431	22	19	5
12	7200	25	22	30
13	1012	4	4	1
14	2492	2	0	2
15	2293	28	30	2
16	2712	16	19	7

```
                          Idling = sum(acceleration == 0))
Events_summary %>%
                kable(caption = 'Summary of Events Per Trip',
                      col.names = c("Trip No.","Distance (meters)","Hard Accelerations", "Hard Brakes"
                booktabs = TRUE) %>%
                kable_classic_2(full_width = T)
```

Part 2 (Modeling)

```
#' Splitting the simulated Telematics data into Training and Test sets

telem <- fread("data/simulated_summary_total.csv") %>%
         as.data.frame()


train_index <- sample(1:nrow(telem), 0.8 * nrow(telem))
test_index <- setdiff(1:nrow(telem), train_index)

telem_train <- telem[train_index,-c(1,2)]
telem_test <- telem[test_index, -c(1,2)]

head(telem)
```

```
##   Vehicle Days Distance HardBrakes HardAccelerations NightTime_Pct VehicleType
## 1       1  365    13114        152               56         0.005         SUV
## 2       2  365    18707        147                1         0.010         SUV
## 3       3  365    16659        151              127         0.019       Truck
## 4       4  365    13330        126              147         0.000         SUV
## 5       5  365    22533         10               11         0.001       Truck
## 6       6  365    16212         25              227         0.006       Truck
```

Table 2: Distribution of vehicle types

VehicleType	Frequency	proportion
Car	9085	0.3028
Minivan	1520	0.0507
SUV	7463	0.2488
Truck	11932	0.3977

Chi-Square Statistics	df	P-value
7722.2	3	2.2×10^{-16}

Table 3: Chi-square test for proportions of vehicle types

```
##   Loss
## 1    0
## 2    0
## 3    0
## 4    1
## 5    0
## 6    0
```

Analysis

1. Is there a statistically difference between Vehicle types

Here, we test the hypothesis problem using chi-square test for multiple comparison.

H_o : The proportions of vehicle types are equal.

H_1 : The proportions of the vehicle types are not equal.

The p-value of the chi-square test is less than 0.05, so we reject the null hypothesis H_o and accept the alternate hypothesis H_1. Thus, we conclude there is a statistically difference between vehicle types.

```
 vehicle_prop <- telem %>%
                 group_by(VehicleType) %>%
                  summarise(Frequency = n()) %>%
                  mutate(proportion = round(Frequency / sum(Frequency),4))

kable(vehicle_prop,
      caption = "Distribution of vehicle types",
      booktabs = TRUE) %>%
         kable_classic_2(full_width = T)
```

```
#' Chisquare test
chi_test <- chisq.test(table(telem$VehicleType))
```

2. Are hard brakes and hard accelerations equally important in predicting risk?

Hard accelerations variable is more important in predicting risk compared to hard brakes. Refer to the feature importance figure under model building below.

Model Building

Here were are furnished with a data with some telematics variables and our task is to build a model that segments the drivers based on the likelihood of occurrence of collision. As a first step in our workflow, we first checked the quality of the data and performed exploratory data analysis to identify pattern in the covariates and the target variables. We proceeded with model specification, then with evaluation and selection and finally model explanation.

Data Description

The telematics data is a hypothetical data set on instances of 30,000 vehicles and it essentially consists of the following features: `Vehicle ID`, `Days data was collected`, `yearly distance travelled`, `hard acceleration`, `hard brake`, `percentage of miles travelled at night`, `vehicle type` and response variable is `Loss` indicating occurrence of collision or no collision

Exploratory Data Analysis (EDA)

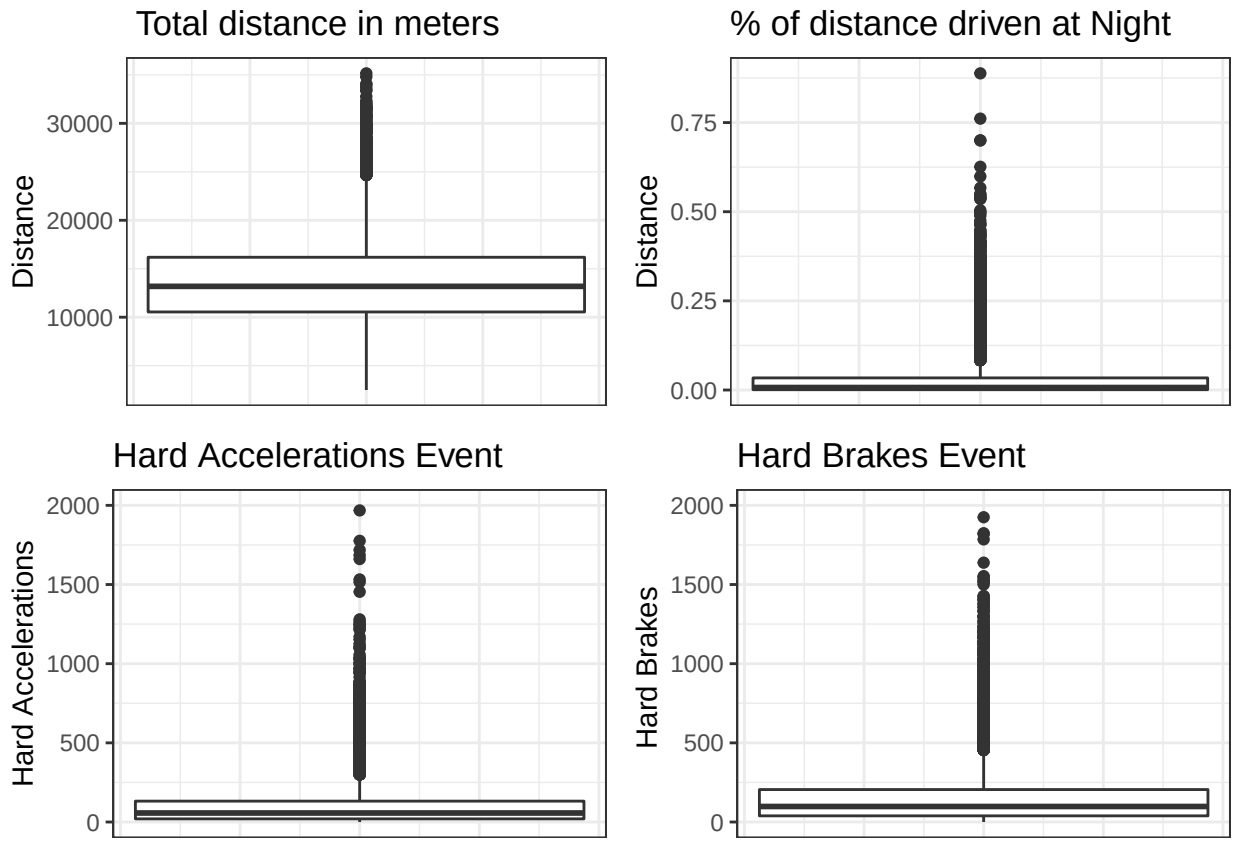
The following plots show exploratory analysis of the predictor variables and the target variable. The main insights derived from the plots include the issue of class imbalance with the target variable, the predictors having skewed distribution and contain some outliers. Among the vehicle types, trucks have higher collision rate and vans have least collision rate.

```
#' Exploratory data Analysis


plt_1 <- ggplot() +
  geom_boxplot(aes(y = telem_train$Distance)) +
  labs(title = " Total distance in meters ", y = "Distance") +
  theme_bw()+
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank())

plt_2 <- ggplot() +
  geom_boxplot(aes(y = telem_train$NightTime_Pct)) +
  labs(title = "% of distance driven at Night", y = " Distance ") +
  theme_bw()+
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank())

plt_3 <-ggplot() +
  geom_boxplot(aes(y = telem_train$HardAccelerations)) +
  labs(title = "Hard Accelerations Event", y = "Hard Accelerations") +
  theme_bw()+
  ylim(0,2000)+
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank())

plt_4 <- ggplot() +
  geom_boxplot(aes(y = telem_train$HardBrakes)) +
  labs(title = "Hard Brakes Event", y = "Hard Brakes ") +
  theme_bw()+
  ylim(0,2000)+
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
```

Table 4: Contigency Table of Vehicle Type and Class

Vehicle Type	No collision	Collision
Car	7955	1130
Minivan	1365	155
SUV	6368	1095
Truck	10281	1651

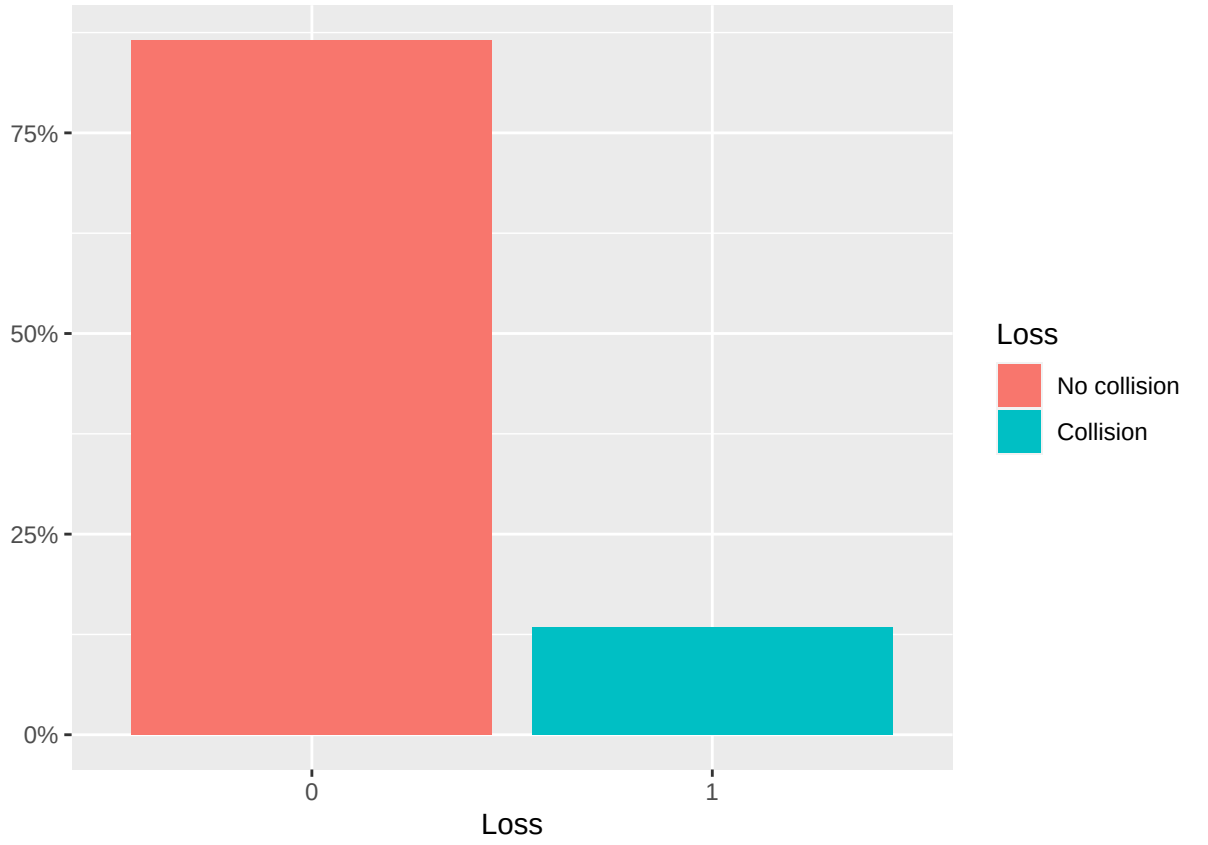
```
cowplot:: plot_grid(plt_1, plt_2, plt_3, plt_4)
```




```
#Contigency Table
tab <- telem%>%
  group_by(VehicleType, Loss) %>%
  summarise(n = n()) %>%
spread(Loss, n)


tab %>%
  kable(caption = 'Contigency Table of Vehicle Type and Class',
      col.names = c("Vehicle Type","No collision", "Collision"),
      booktabs = TRUE) %>%
      kable_classic_2(full_width = T)
```

```
# Target Variable
# Categorical Variables
```

```
ggplot(data = telem_train, aes(as.factor(Loss), fill = as.factor(Loss)))+
 geom_bar(aes(y = (..count..)/sum(..count..))) +
  labs(x = "Loss", y = "") +
          scale_y_continuous(labels=scales::percent) +
  guides(fill = guide_legend(title="Loss")) +
  scale_fill_discrete(
    labels = c("No collision", "Collision"))
```



Transformation and Feature Engineering

The `Hard acceleration`, `Hard Brakes` and `percentage of miles travelled at night` have skewed distribution concentrated around smaller values, structural zeros and relatively few outliers. Using domain knowledge and potential goals of business we could transform these variables from the continuous scale to a categorical scale. The transformation (binning) was done using chosen quantile values. For instance, the 25^{th} percentile of Hard acceleration and Hard brakes was used as the threshold to categorize the vehicles into low risk and high risk. Where vehicles with hard events below 25^{th} percentile is considered low risk and high risk otherwise. The 25^{th} percentile for hard acceleration is around 20 per year and for hard brake is around 40 per year. For the `percentage of miles travelled at night`, the 75^{th} percentile was used as a threshold for segmentation. The `Distance` variable had few natural outliers so no categorization was done. However, these few outliers were capped. The distance was then transformed using min-max normalization.

```
#'

#' quantile distribution of numeric variables
fivenum_sum_acc <- fivenum(telem_train$HardAccelerations)
fivenum_sum_Brake <- fivenum(telem_train$HardBrakes)
```

```r
fivenum_sum_night_drive <- fivenum(telem_train$NightTime_Pct)

telem_train <- telem_train %>%
          mutate(Hard_Acc_category = cut(HardAccelerations,
               breaks = c(-Inf,fivenum_sum_acc[2],Inf),
               labels = c("lowRisk","HighRisk")),
               .after= HardAccelerations) %>%

          mutate(Hard_Brake_category = cut(HardBrakes,
              breaks = c(-Inf,fivenum_sum_Brake[2],Inf),
              labels = c("lowRisk","HighRisk")),
              .after= HardBrakes) %>%

          mutate( Night_Time_category = cut(NightTime_Pct,
                breaks =   c(-Inf,fivenum_sum_night_drive[4],Inf),
                labels = c("lowRisk","HighRisk")),.after= NightTime_Pct)


#'
telem_test <- telem_test %>%
          mutate(Hard_Acc_category = cut(HardAccelerations,
                breaks = c(-Inf,fivenum_sum_acc[2], Inf),
                labels = c("lowRisk","HighRisk")),
                .after= HardAccelerations) %>%

         mutate(Hard_Brake_category = cut(HardBrakes,
                breaks = c(-Inf,fivenum_sum_Brake[2], Inf),
                labels = c("lowRisk","HighRisk")),.after= HardBrakes) %>%

         mutate(Night_Time_category = cut(NightTime_Pct,
                breaks = c(-Inf,fivenum_sum_night_drive[4],Inf),
                labels = c("lowRisk","HighRisk")),.after= NightTime_Pct)


train_df <- telem_train %>%
      select(!c("HardAccelerations", "HardBrakes", "NightTime_Pct"))%>%
      mutate(VehicleType = as.factor(VehicleType), Loss = as.factor(Loss))


test_df <- telem_test %>%
      select(!c("HardAccelerations", "HardBrakes", "NightTime_Pct")) %>%
      mutate(VehicleType = as.factor(VehicleType), Loss = as.factor(Loss))


#' Capping and normalizing Distance

cap <- fivenum(telem_train$Distance)[4] + 1.5*IQR(telem_train$Distance)

train_df <- capLargeValues(train_df, target = "Loss",
                           cols = c("Distance"),
                           threshold = cap)

test_df <- capLargeValues(test_df, target = "Loss",
```

```
                          cols = c("Distance"),
                          threshold = cap)

# Normalize Train set
 train_df <- normalizeFeatures(train_df,
              target = "Loss",
              method = "scale",
              cols = NULL,
              on.constant = "quiet"
            )

# Normalize Test set
 test_df <- normalizeFeatures(test_df,
              target = "Loss",
              method = "scale",
              cols = NULL,
              on.constant = "quiet"
            )
```

```
#' Creating Learning Task
#' Unbalanced class
trainTask <- makeClassifTask(data = train_df,
                             target = "Loss",
                             positive = "1")

testTask <- makeClassifTask(data = test_df,
                            target = "Loss",
                            positive = "1")

#' Balancing class using different sampling technique
train_over <-  oversample(trainTask, rate = 6)
test_over  <-  oversample(testTask, rate = 6)

train_under <- undersample(trainTask, rate = 1/6)
test_under <- undersample(testTask, rate = 1/6)

train_smote <- smote(trainTask, rate = 6, nn = 8)
test_smote <- smote(testTask, rate = 6, nn = 8)
```

Feature Importance

Feature importance was carried using the information gain criteria. The plot below reveals `Distance` variable is the most important and the least significant is the `Hard Brakes`.

```
fv = generateFilterValuesData(train_smote, method = "FSelectorRcpp_information.gain")

plotFilterValues(fv) +
  labs(title = "Feature importance")
```

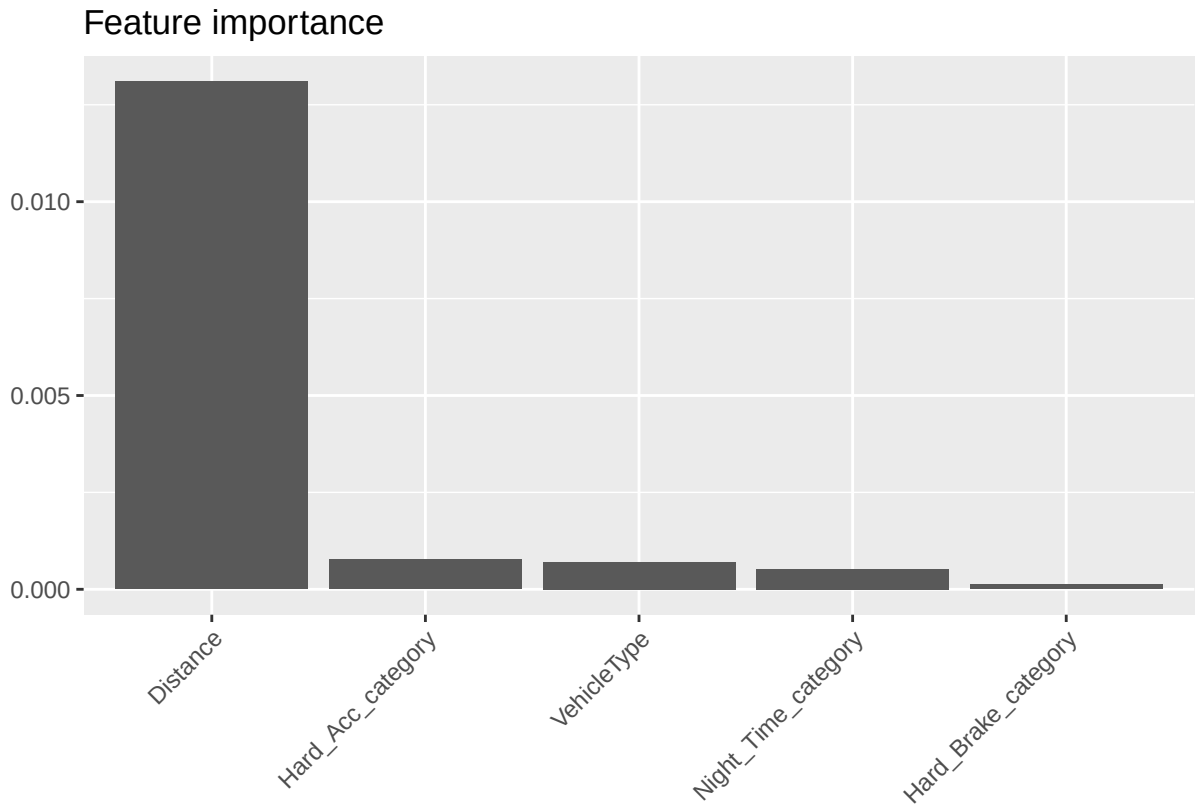


Figure 1: Feature importance of variables in predicting Loss

Model Specification

The response type or the outcome of this data is binary so we specify various binary classification models of differnt level of complexities. We fitted logistic regression ,random Forest, gradient boosting and the XGboost models using the training data sets. Since the response is highly imbalanced, we used different sampling techniques including undersampling the majority class, oversampling the majority class and the SMOTE algorithm. The specified models were trained across the different samples but there was no significant difference among them.

Model Performance and Evaluation

In training the model on the different samples, we ensured model performance by using some validation techniques especially cross-validation. We also tuned the hyperparameters such as number of trees, tree depth using random search and grid search. The optimal parameters parameter values are then used to fit the final best model for prediction. We used the test_set data for prediction and evaluated our prediction using chosen metrics including recall, precision, AUC. Threshold value for prediction was tuned slightly from 0.5 to $0.4 : 0.6$ towards to postive class. Using recall and precision metrics, Logistic regression model tend to be the best model followed by the gradient boosting method. Thus, with this model, we segment these vehicles by calculating the posterior probabilities of each vehicle belonging to a collision class or no collision class.

```
# Making a logistic Learner

logistic_learner <- makeLearner("classif.logreg",
                                predict.type = "prob",
                                fix.factors.predict = TRUE,
                                predict.threshold = 0.40)

# cross validation (cv) accuracy
cv.logistic <- crossval(learner = logistic_learner,
                        task = train_over,iters = 15L,
                        stratify = TRUE,
                        measures = list(acc ,tpr),
                        show.info = F)

#cv.logistic$measures.test

#training model
log_fit <- train(logistic_learner, train_over)
#getLearnerModel(log_model)

#predicting on test data
log_pred <- predict(log_fit, test_over)

perf_metrics <- performance(log_pred, measures = list(tpr,ppv, acc, auc,f1))
```

```
# Random Forest
param_set_ranf <- getParamSet("classif.randomForest")

# create a learner
ranf_learner <- makeLearner("classif.randomForest",
                            predict.type = "prob",
                            par.vals = list(ntree = 100, mtry = 2),
                            predict.threshold = 0.4)

ranf_learner$par.vals <- list(importance = TRUE)
```

```
#' setting  tunable parameters
#' grid search to find hyperparameters

ranf_param <- makeParamSet(
                        makeIntegerParam("ntree",lower = 50, upper = 100),
                        makeIntegerParam("mtry", lower = 2, upper = 10),
                        makeIntegerParam("nodesize", lower = 10, upper = 50)
                        )

#' random search for 50 iterations
rand_search <- makeTuneControlRandom(maxit = 40L )

#' Cross-validation
cv_rand <- makeResampleDesc("CV",iter =3L , predict = "both")

#hypertuning
ranf_tune <- tuneParams(learner = ranf_learner,
                        task = train_over,
                        par.set = ranf_param,
                        resampling = cv_rand,
                        control = rand_search,
                        measures = list(tpr, ppv, acc))


#cv accuracy
ranf_tune$y
ranf_tune$x #optimal parameters

#using hyperparameters for modeling
ranf_tree <- setHyperPars(ranf_learner, par.vals = ranf_tune$x)


rand_forest_fit <- train(ranf_tree, train_over)

rand_pred <- predict(rand_forest_fit, test_over)

#' Performance metrics
rand_perf <- performance(rand_pred, list(tpr,ppv, acc, auc,f1))
```

```
gb_param_set <- getParamSet("classif.gbm")

gb_learner <- makeLearner("classif.gbm",
                          predict.type = "prob",
                          predict.threshold = 0.4)

#specify tuning method
gb_rand_search <- makeTuneControlRandom(maxit = 50L)

#3 fold cross validation
cv_gb <- makeResampleDesc("CV",iters = 3L)

#parameters
gb_param <- makeParamSet(
```

```
              makeDiscreteParam("distribution", values = "bernoulli"),
              makeIntegerParam("n.trees", lower = 10, upper = 100),
              makeIntegerParam("interaction.depth", lower = 2, upper = 5),
              makeIntegerParam("n.minobsinnode", lower = 2, upper = 8),
              makeNumericParam("shrinkage",lower = 0.01, upper = 1)
)

#tune parameters
gb_tune <- tuneParams(learner = gb_learner,
                      task = train_smote,
                      resampling = cv_gb,
                      measures = list(acc ,tpr),
                      par.set = gb_param,
                      control = gb_rand_search)



#check CV accuracy
gb_tune$y

#set parameters
gb_optim <- setHyperPars(learner = gb_learner, par.vals = gb_tune$x)

#train
gb_fit <- train(gb_optim, train_smote)

#test
gb_pred <- predict(gb_fit, test_smote)

performance(gb_pred, list(tpr,ppv, acc, auc,f1))
```

```
# Hot encoding
train_xg <- createDummyFeatures(train_smote)
test_xg <- createDummyFeatures(test_smote)

xgb_learner <- makeLearner("classif.xgboost",
                           predict.type = "prob",
                           predict.threshold = 0.4)

xgb_learner$par.vals <- list(
                        objective = "binary:logistic",
                        eval_metric = "error",
                        nrounds = 250 )

#define parameters for tuning
xgb_param <- makeParamSet(
              makeIntegerParam("nrounds",lower=200,upper=600),
              makeIntegerParam("max_depth",lower=3,upper=20),
              makeNumericParam("lambda",lower=0.55,upper=0.60),
              makeNumericParam("eta", lower = 0.001, upper = 0.5),
              makeNumericParam("subsample", lower = 0.10, upper = 0.80),
              makeNumericParam("min_child_weight",lower=1,upper=5),
              makeNumericParam("colsample_bytree",lower = 0.2,upper = 0.8)
              )
```

	Recall	Precision	F1	AUC	Accuracy
Logistic	0.8450	0.5083	0.6340	0.5823	0.5290
Gradient Boosting	0.7638	0.5082	0.6103	0.5516	0.5276
XGboost	0.5134	0.4979	0.5231	0.5254	0.5134
Random Forest	0.4855	0.5078	0.4964	0.5304	0.5230

Table 5: Model Evaluation using SMOTE sampling technique and tuned threshold

```
#define search function
xgb_rand_search <- makeTuneControlRandom(maxit = 50L) #do 100 iterations

#3 fold cross validation
cv_xgb <- makeResampleDesc("CV",iters = 3L)

#tune parameters
xgb_tune <- tuneParams(learner = xgb_learner,
                       task = train_xg,
                       resampling = cv_xgb,
                       measures = list(acc ,tpr),
                       par.set = xgb_param,
                       control = xgb_rand_search)

#set parameters
xgb_optim <- setHyperPars(learner = xgb_learner, par.vals = xgb_tune$x)

#train model
xgb_fit <- train(xgb_optim, train_xg)

#test model
xgb_pred <- predict(xgb_fit, test_xg)
performance(xgb_pred, list(ppv, tpr, acc, auc, f1, gmean))
xgb_Thresh <- setThreshold(xgb_pred, threshold = c("1" = .4, "0" = .60 ))
performance(xgb_Thresh, measures = list(ppv, tpr, acc, auc, f1))
```

Data Set Enhancement

1. Research questions

(a) Does a policyholder who drives more at night have higher claim frequency compared to a policyholder who drives in the morning?

A person may drive in the night habitually for a purpose. For instance, he may be driving to work at night on routine basis which would help him develop a good night driving behavior over time. Higher percentage of night driving may reduce the expected time until a first accident. Thus, a person who drives more at night may not necessarily experience a collision and be at risk compared to a person who drives in the morning.

(b) Is road type a good preditor of losses due to high collision rate?

Insured who drives irresponsibly on interstate highways may result in higher aggregate loss due to collision than an insured who drives on local roads.This may impact insurer books of business.

(c) What are the key performing telematics variables and the modeling framework that required for

classifying automobile policy holders?

Availability and knowledge these key variables would help in faster modeling and effectively price data products. These variable may not be exclusive nor exhustive. They can be fine tuned, redefined, updated as the business scenarios evolve.

2. Additional Attributes

(a) Road type: Division of the distance into 4 road types. Highways, arterials, collectors and locals

(b) Week/Weekend: Division of distance into week days (Monday to Friday) and weekend (Saturday, Sunday)

(c) Seasons type: Dividing distance into fall, spring, summer and winter seasons

(d) Altitude: The height gained by Vehicle above sea level

(e) Age of vehicle: The difference between the year the vehicle was acquired and the year the telematics data were collected

3. Sample Size

The sample size for the quantitative research questions stated above in general can be estimated using the formula, $n \geq \left(\frac{Z_\alpha \sigma}{E}\right)^2$. Where,

- $Z\alpha$ is the $Z-$score value at the desired level of significance, α
- E is the specified margin of error
- σ is the standard deviation of the variable of interest

Since distance variable is an integral component in addressing the research questions, we would use the distance data to estimate σ. From the distance data, $\sigma = 3,985$ excluding outliers. Thus for a margin of error, $E = 50$, $\alpha = 0.05$ and $Z_{\alpha/2} = 1.96$, the estimated sample size is given $n \geq \left(\frac{1.96(3985)}{50}\right)^2 = 24,000$.