Project 2: Data Mining, Classification, Prediction

SDS322E

Mining, Classification, Prediction

Kush Patel, ksp946

Introduction

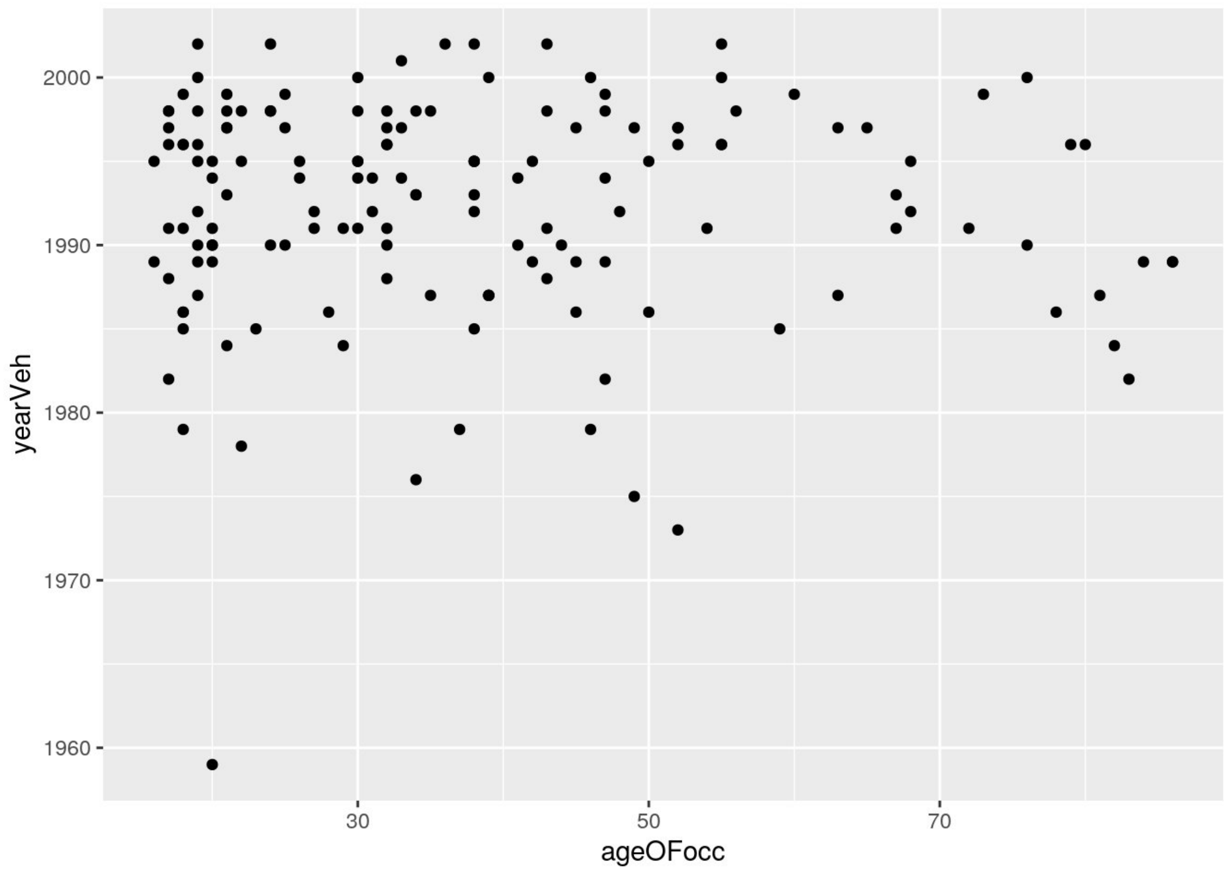
```r
library(tidyverse)
library(DAAG)
car_data <- as.data.frame(nassCDS)
tidycar <- car_data %>% select(-caseid, -abcat) %>% filter(occRole ==
    "driver") %>% na.omit()
set.seed(123)
tidycar <- sample_n(tidycar, 150)
```

This dataset is called “nassCDS” and it can be found in R under DAAG package. I found this data using following website: https://vincentarelbundock.github.io/Rdatasets/datasets.html . This is data of US police-reparted car crashed in which there is a harmful event, and from which at least one vehicle was towed. This data is collected form 1997-2002. The dataset contains 26217 observations with 15 variables. The variables are “dvcat” which is ordered factor with levels of estimated impact speeds. “weight,” observation weights, designed to account for varying sampling probabilities. “dead” factor with levels alive or dead. “airbag” is a factor with levels none and airbag. “seatbelt” with factor levels none and belted. “frontal” is a numeric vector with 0 = non-frontal impact, 1= frontal impact, “ageOFocc” tells age of occupants in years. “yearacc” year of accident. “deploy” is a numeric vector with 0 if an airbag was unavailable or did not deploy, 1 if *one or more bags deployed. “injSeverity” a numeric vector with 0-6 rating of injury*

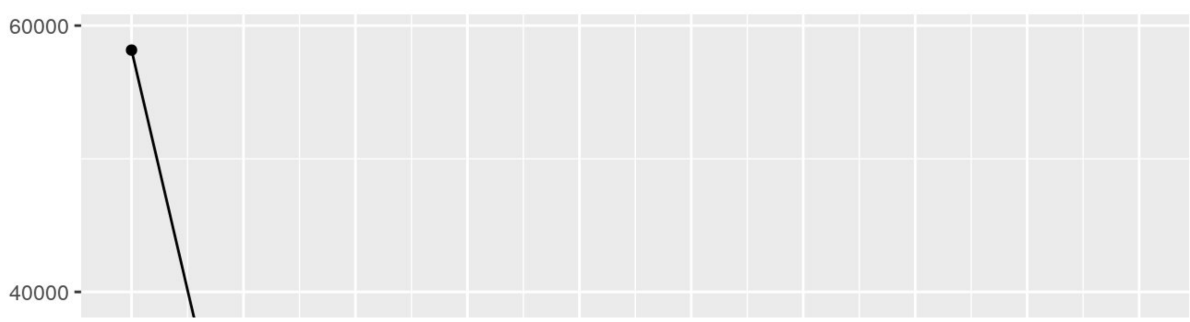
The data was mostly tidy, I just removed all NAs from dataframe and also remove ‘caseid’ and ‘abcat’ variables as they are mostly not in use for the project. I also filtered to where data include just the driver who were injured, reduring other variables. Lastly since it was a huge dataset, I have randomly selected 150 observation for the rest of the project.

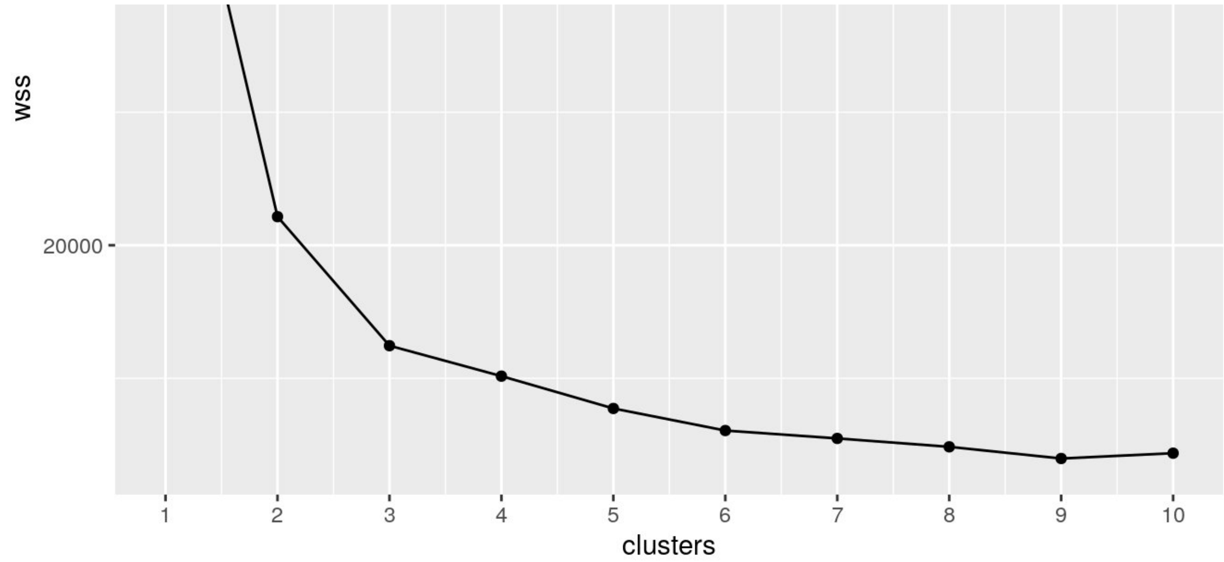
Cluster Analysis

```
library(cluster)
# selecting number of cluster
tidycar %>% ggplot() + geom_point(aes(ageOFocc, yearVeh))
```

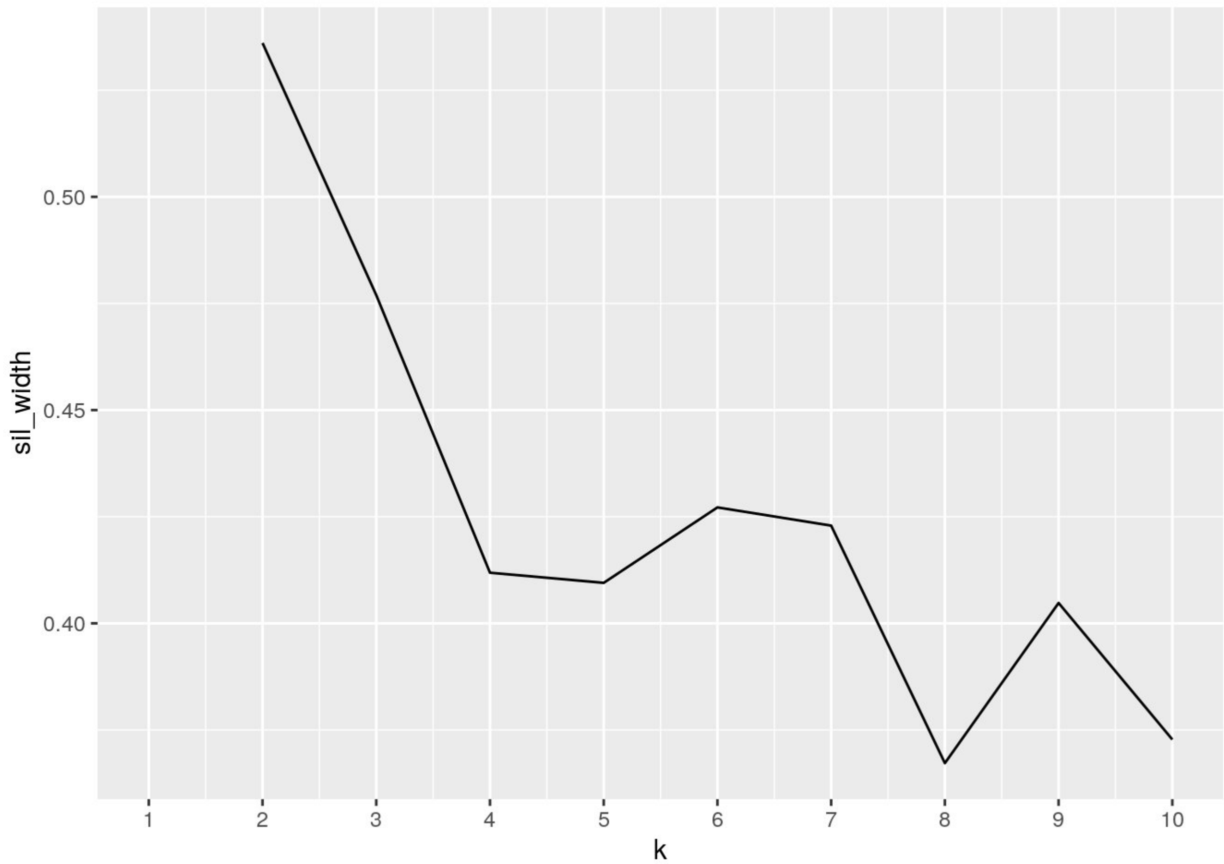



```
wss <- vector()
for (i in 1:10) {
    temp <- tidycar %>% select(yearVeh, ageOFocc) %>% kmeans(i)
    wss[i] <- temp$tot.withinss
}
ggplot() + geom_point(aes(x = 1:10, y = wss)) + geom_path(aes(x = 1:10,
    y = wss)) + xlab("clusters") + scale_x_continuous(breaks = 1:10)
```






```
# computing silhouette
clust_dat <- tidycar %>% dplyr::select(yearVeh, ageOFocc)

sil_width <- vector()
for (i in 2:10) {
    kms <- kmeans(clust_dat, centers = i)  #compute k-means solution for each k
    sil <- silhouette(kms$cluster, dist(clust_dat))  #get sil widths
    sil_width[i] <- mean(sil[, 3])  #take averages (higher is better)
}
ggplot() + geom_line(aes(x = 1:10, y = sil_width)) + scale_x_continuous(name = "k",
    breaks = 1:10)
```

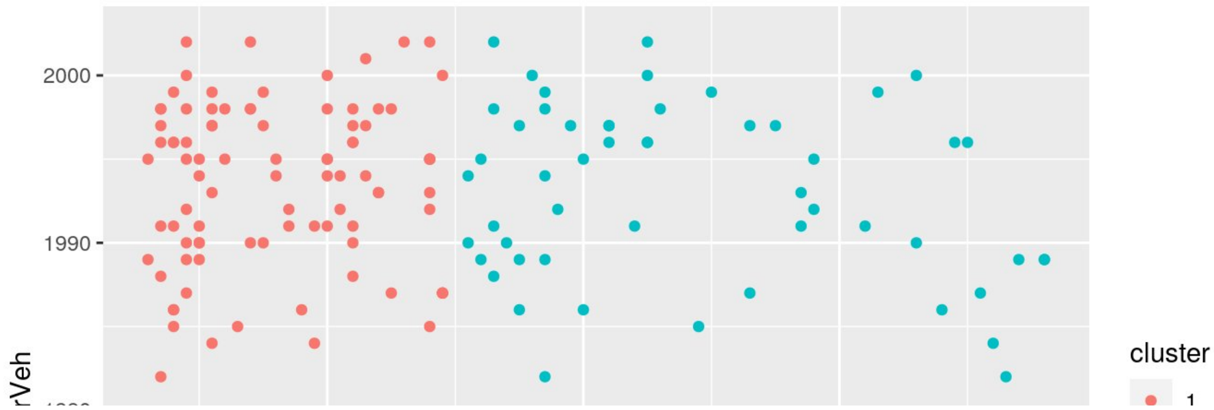


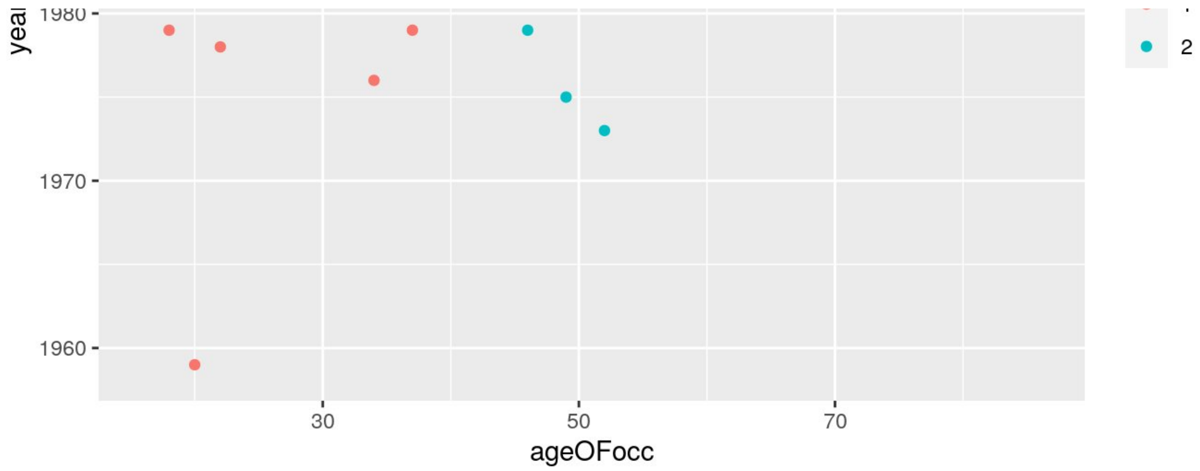
Initially by looking at scatter plot, there is no visible pattern between clusters of "ageOFocc" and *“yearVeh”, I also chose these two dataset as I wanted to know if year of vehical a person drives is connected to the age of person when the accident occurred. The wss plot shows that the number of clusters which will be appropriate for this data is 2. By computing silhouette width, it shows the same pattern of clusters = 2*

```
set.seed(562)
pam1 <- clust_dat %>% pam(k = 2)
pam1
```

```
## Medoids:
##       ID yearVeh ageOFocc
## 866   85    1994       26
## 5064 135    1996       55
## Clustering vector:
## 24037 24103  3814  2342  4318 14845  6104  8631 20590  3511 16118 11761 13036
##     1     1     1     1     2     1     2     1     1     1     1     2     2
## 17452  3687  7889 24172 12326 15942 24692  2034 21632 18097 19366 11956 12768
##     1     2     2     1     2     1     2     1     1     1     2     1     1
## 11618 19483 22249  3842 24982 10208  5120 17275 10681 21200 13829  7954 22942
##     1     2     2     1     2     1     2     2     1     1     2     2     1
## 25929 26123 24604 21296 10823 17584    50 18399  9432  9310 20383 14064 24093
##     1     2     1     1     1     1     1     2     2     1     1     1     2
##  8626 14637 10946 12821 13126 15686 23605  7839   960  8384  9110 12323 25184
##     1     2     1     1     2     2     1     1     2     1     2     1     1
##  5101 18438 11915  4136  7167 12394 25742 18369 12911  7618 11877 15879  9980
##     1     2     1     2     1     1     2     2     1     2     1     1     1
## 14465 19825  1754 13387 23928 26064   866 15371 19328 19997  6439 20563 19409
##     1     1     2     1     1     1     1     2     1     2     2     1     1
## 20623  9307  6052 18136 18228   193 12565  2819 20099
##     2     2     1     2     1     1     1     1     1
##  [ reached getOption("max.print") -- omitted 50 entries ]
## Objective function:
##    build     swap
## 11.58767 10.73916
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

```
pamclust <- clust_dat %>% mutate(cluster = as.factor(pam1$clustering))
pamclust %>% ggplot(aes(ageOFocc, yearVeh, color = cluster)) +
    geom_point()
```

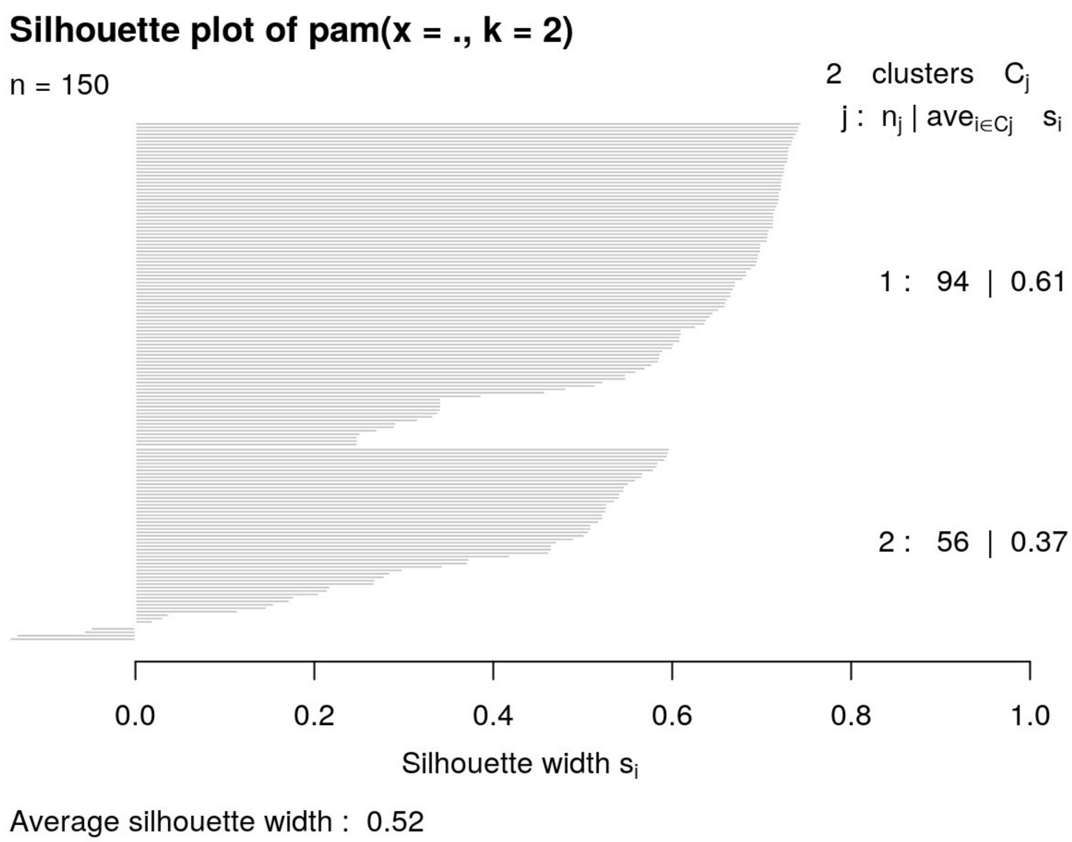





```
pam1$silinfo$avg.width
```

```
## [1] 0.5206227
```

```
plot(pam1, which = 2)
```



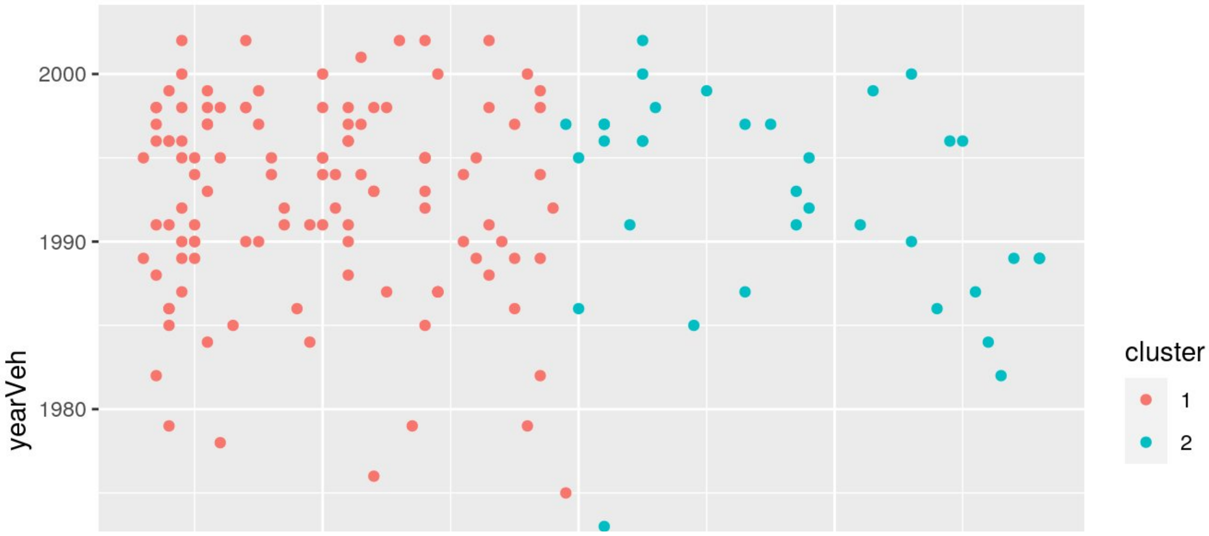
By running pam analysis, I found that the average year of vehicle for cluster 1 is 1994 and the average age of accident is 26 for cluster1. I also found that the average year of vehicle for cluster 2 is 1996 and the average age of accident is 55 for cluster2. After plotting colored graph the difference between two cluster is clearly seen. The average Silhouette width is 0.52. Silhouette width between 0.51-0.70 means that a reasonable structure has been found. Silhouette width between 0.26-0.50 means that the structure is weak and could be artificial

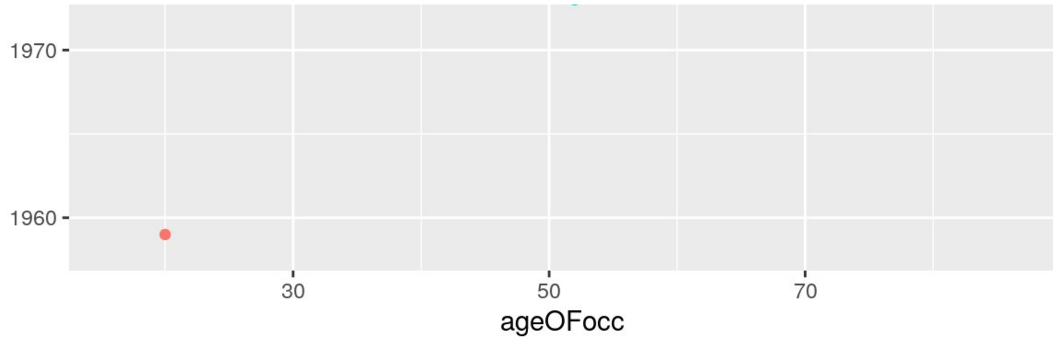
```
# clustering with 3 or more variable
```

```
final <- tidycar %>% select(ageOFocc, yearVeh, deploy, injSeverity) %>%
    as.data.frame()
pam2 <- final %>% pam(2)
pam2
```

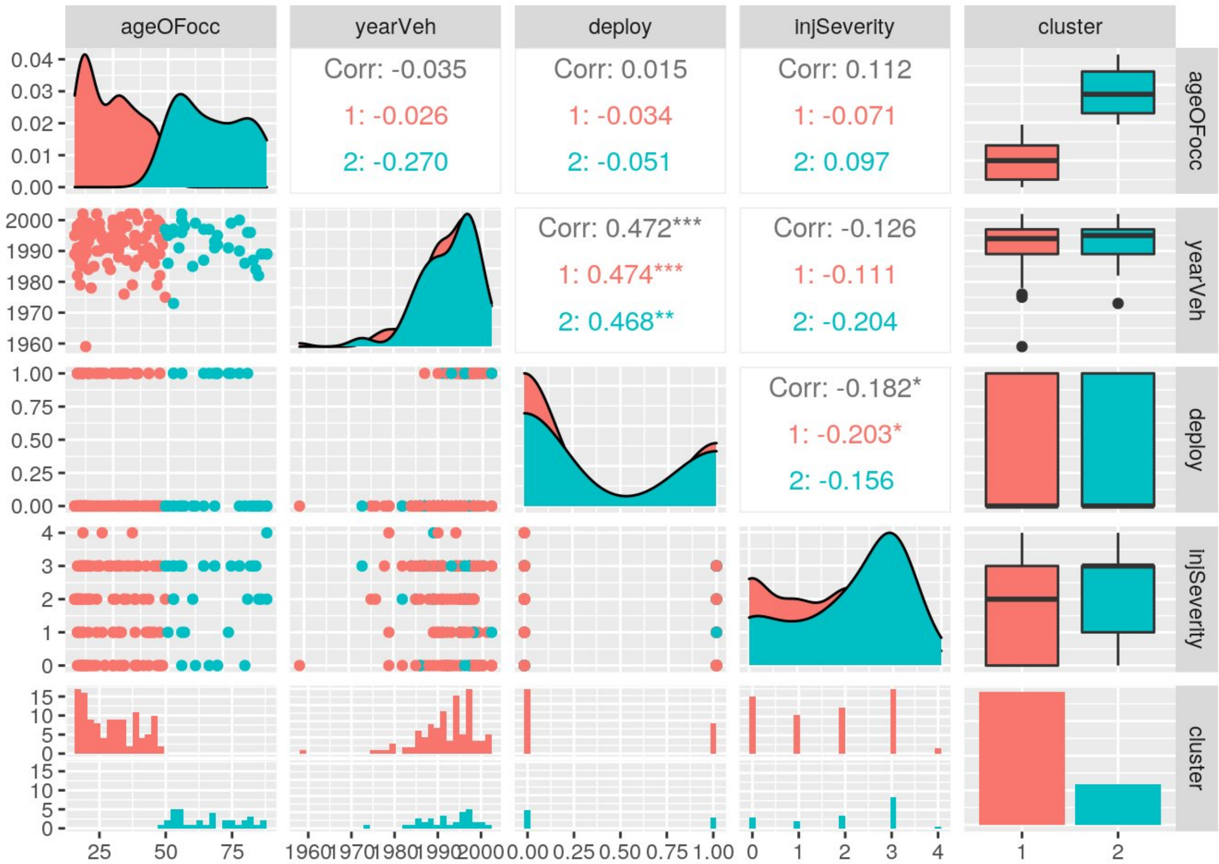
```
## Medoids:
##          ID ageOFocc yearVeh deploy injSeverity
## 15582 133        31    1992      0           2
## 4917  139        67    1993      1           3
## Clustering vector:
## 24037 24103  3814  2342  4318 14845  6104  8631 20590  3511 16118 11761 13036
##     1     1     1     1     2     1     1     1     1     1     1     2     1
## 17452  3687  7889 24172 12326 15942 24692  2034 21632 18097 19366 11956 12768
##     1     1     2     1     2     1     1     1     1     1     2     1     1
## 11618 19483 22249  3842 24982 10208  5120 17275 10681 21200 13829  7954 22942
##     1     1     2     1     2     1     1     2     1     1     2     1     1
## 25929 26123 24604 21296 10823 17584    50 18399  9432  9310 20383 14064 24093
##     1     2     1     1     1     1     1     2     1     1     1     1     2
##  8626 14637 10946 12821 13126 15686 23605  7839   960  8384  9110 12323 25184
##     1     2     1     1     1     2     1     1     1     1     1     1     1
##  5101 18438 11915  4136  7167 12394 25742 18369 12911  7618 11877 15879  9980
##     1     1     1     2     1     1     2     2     1     1     1     1     1
## 14465 19825  1754 13387 23928 26064   866 15371 19328 19997  6439 20563 19409
##     1     1     2     1     1     1     1     2     1     2     2     1     1
## 20623  9307  6052 18136 18228   193 12565  2819 20099
##     2     2     1     2     1     1     1     1     1
##  [ reached getOption("max.print") -- omitted 50 entries ]
## Objective function:
##    build     swap
## 11.73511 11.30461
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

```
final <- final %>% mutate(cluster = as.factor(pam2$clustering))
ggplot(final, aes(x = ageOFocc, y = yearVeh, color = cluster)) +
    geom_point()
```






```r
library(GGally)
ggpairs(final, aes(color = cluster))
```



I included another variable, "injSeverity" to plot a 3D graph which shows in detail the difference between these two cluster and relationship between 3 variables. Then I performed PAM clustering with 4 numeric variables. The variables that had highest correlation was "deploy" and "yearVeh", as there is a greater chance that the vehicle have airbag and it will deploy if the vehical is modern compared to older version of vehicle

Dimensionality Reduction with PCA

```r
# PCA code here
var1 <- tidycar %>% select(frontal, ageOFocc, yearVeh, deploy,
    injSeverity) %>% as.data.frame()
var1 <- data.frame(scale(var1))  #scaling data


pca1 <- prcomp(var1, center = T, scale = T)
summary(pca1)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5
## Standard deviation     1.2854 1.0327 0.9952 0.9238 0.66153
## Proportion of Variance 0.3305 0.2133 0.1981 0.1707 0.08752
## Cumulative Proportion  0.3305 0.5437 0.7418 0.9125 1.00000
```

```
eig1 <- var1 %>% cor %>% eigen()
eig1  # eigen value and eigen vectors
```

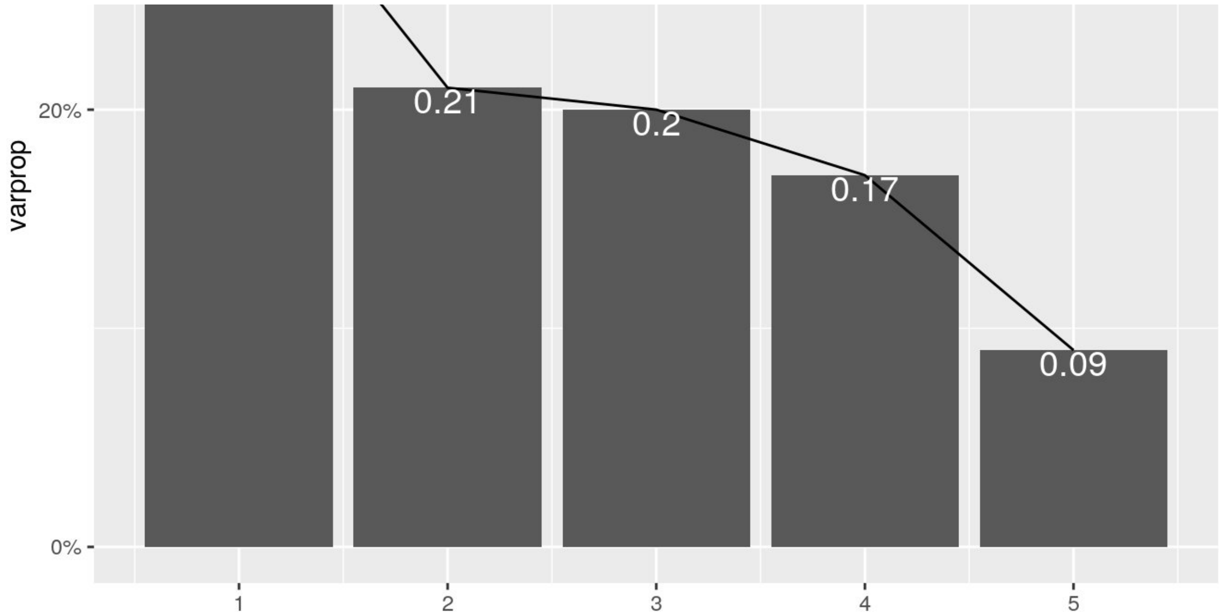
```
## eigen() decomposition
## $values
## [1] 1.6522272 1.0663907 0.9903849 0.8533773 0.4376199
##
## $vectors
##            [,1]       [,2]        [,3]       [,4]        [,5]
## [1,]  0.3549491  0.2081616  0.82675763  0.1014289 -0.36995121
## [2,] -0.1023675 -0.7896991  0.26384848 -0.5351205 -0.09962996
## [3,]  0.5513358 -0.3045061 -0.45159855  0.2307687 -0.58831068
## [4,]  0.6571904 -0.2185485  0.07780584  0.1030321  0.70969565
## [5,] -0.3572977 -0.4388127  0.19199166  0.7996802  0.05858839
```

```
var1 %>% cor  #correlation matrix
```

```
##                  frontal    ageOFocc      yearVeh      deploy injSeverity
## frontal      1.000000000 -0.04948004  0.001188138  0.29462783 -0.09001084
## ageOFocc    -0.049480036  1.00000000 -0.034557147  0.01523042  0.11240121
## yearVeh      0.001188138 -0.03455715  1.000000000  0.47239890 -0.12645175
## deploy       0.294627825  0.01523042  0.472398904  1.00000000 -0.18239220
## injSeverity -0.090010844  0.11240121 -0.126451747 -0.18239220  1.00000000
```

The sd of PC1 is 1.2348 with variance of 0.315, whereas sd of PC2 is 1.0398 with variance of 0.217. Also noticed a trend where as PCs increases, the sd and variance is decreasing. The eigenvalue of PC1 is 1.652 and eigenvalue of PC2 is 1.066.

```
# How many PCs to keep?
car_pca <- princomp(var1)
eigval <- car_pca$sdev^2
varprop = round(eigval/sum(eigval), 2)

ggplot() + geom_bar(aes(y = varprop, x = 1:5), stat = "identity") +
    xlab("") + geom_path(aes(y = varprop, x = 1:5)) + geom_text(aes(x = 1:5,
    y = varprop, label = round(varprop, 2)), vjust = 1, col = "white",
    size = 5) + scale_y_continuous(breaks = seq(0, 0.6, 0.2),
    labels = scales::percent) + scale_x_continuous(breaks = 1:10)
```



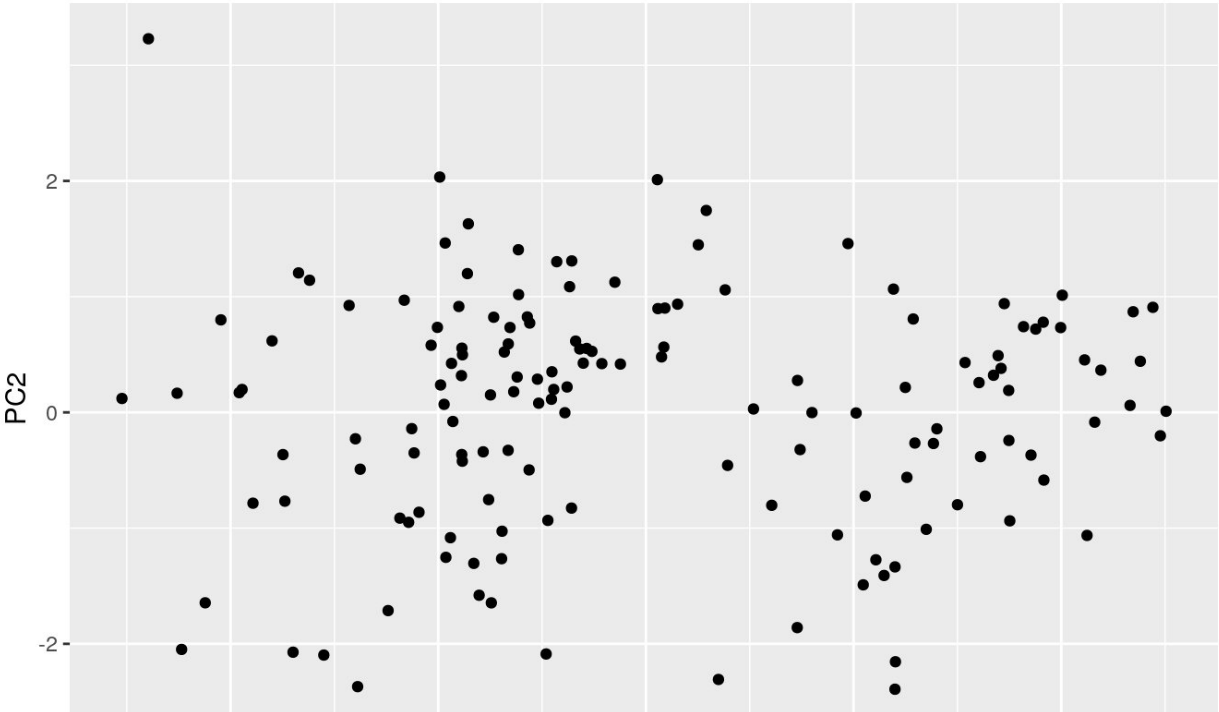

```
round(cumsum(eigval)/sum(eigval), 2)
```

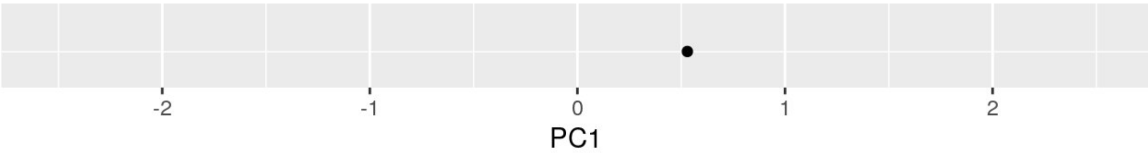
```
## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
##   0.33   0.54   0.74   0.91   1.00
```

```
eigval
```

```
##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5
## 1.6412124 1.0592814 0.9837823 0.8476881 0.4347024
```

```
cardf <- data.frame(PC1 = car_pca$scores[, 1], PC2 = car_pca$scores[,
    2])
ggplot(cardf, aes(PC1, PC2)) + geom_point()
```






```
car_pca$scores[, 1:5] %>% as.data.frame %>% top_n(-3, Comp.1)  #top 3 lowest PC1
```

```
##         Comp.1    Comp.2    Comp.3     Comp.4     Comp.5
## 3511 -2.395503 3.2283348 2.3305044 -1.7690749 -2.2207381
## 3687 -2.522863 0.1202073 0.2328742 -0.9654824 -1.4948427
## 2499 -2.257263 0.1654756 2.2837693 -0.2688596 -0.9368533
```

```
car_pca$scores[, 1:5] %>% as.data.frame %>% top_n(3, Comp.1)  #top 3 highest PC1
```

```
##         Comp.1       Comp.2     Comp.3      Comp.4      Comp.5
## 20590 2.441035  0.907843278 -0.3381565 -0.08200057 -0.05415514
## 24172 2.504534  0.009872629 -0.2066251 -0.55738825  0.22657775
## 24692 2.477065 -0.202034800 -0.1358242 -0.70098219  0.25331240
```

```
car_pca$scores[, 1:5] %>% as.data.frame %>% top_n(3, wt = desc(Comp.2))  #top 3 low
est PC2
```

```
##           Comp.1    Comp.2     Comp.3       Comp.4       Comp.5
## 12326 -1.3880394 -2.369806  1.8193594 -0.040700150  0.631003561
## 18399  1.1992932 -2.391990  0.9579743  0.259381375  0.003802955
## 19997  0.5297446 -2.997718 -0.7849620 -0.002245071 -0.658743923
```

```
car_pca$scores[, 1:3] %>% as.data.frame %>% top_n(3, wt = Comp.2)  #top 3 highest P
C2
```

```
##            Comp.1   Comp.2    Comp.3
## 3511  -2.39550289 3.228335 2.3305044
## 10681 -0.99273628 2.033342 1.0813032
## 3999   0.05527236 2.011004 0.2940965
```

Rule of thumb for picking PCs is to pick PCs until cumulative proportion of variance is > 80%. I have also summarize top annd bottom 3 PC1 and PC2. After plotting the scatter graph of PC1 vs PC2, the points looks like they average in a straight horizontal line (as expected)

Linear Classifier

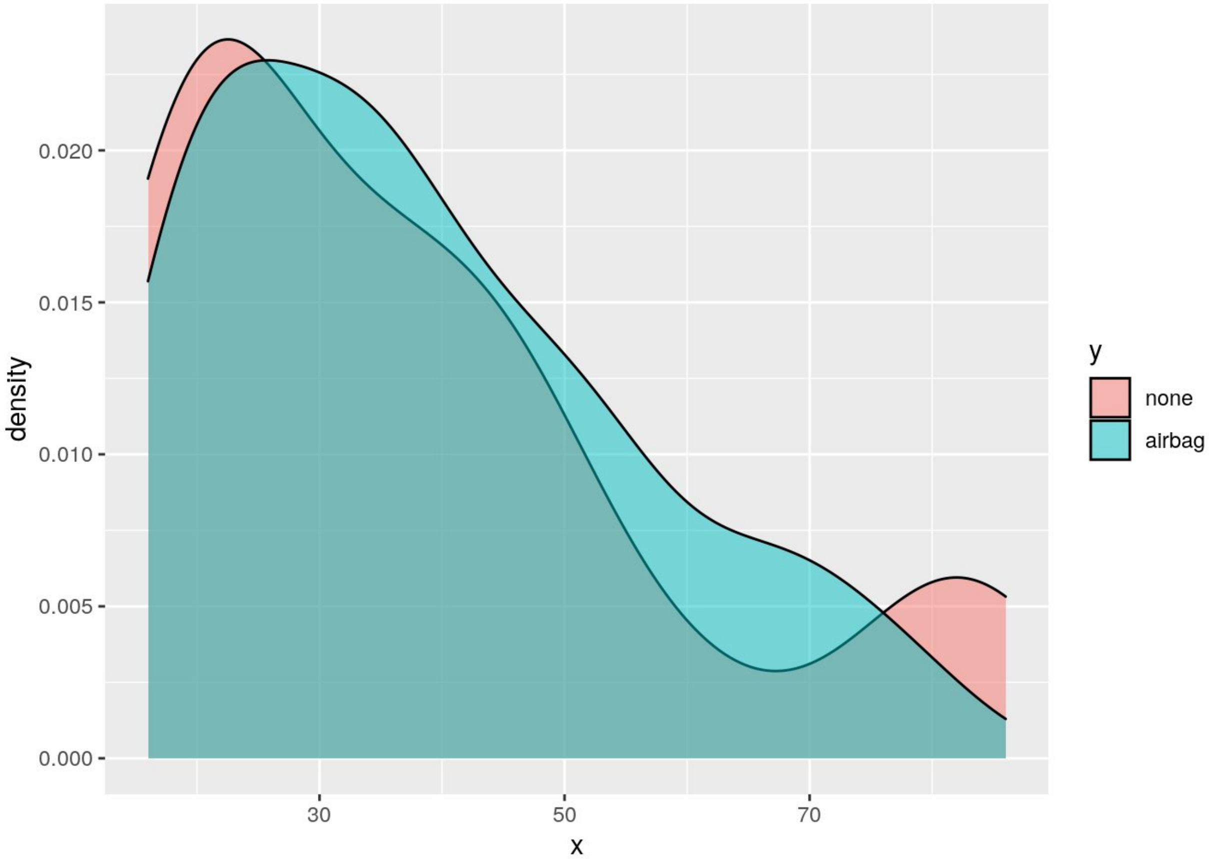
```
y <- tidycar$airbag
x <- tidycar$ageOFocc

y_hat <- sample(c("airbag", "none"), size = length(y), replace = T)
tidycar %>% select(ageOFocc, airbag) %>% mutate(predict = y_hat) %>%
    head
```

```
##   ageOFocc airbag predict
## 1       38 airbag  airbag
## 2       18 airbag    none
## 3       34 airbag  airbag
## 4       20 airbag  airbag
## 5       72 airbag  airbag
## 6       18   none    none
```

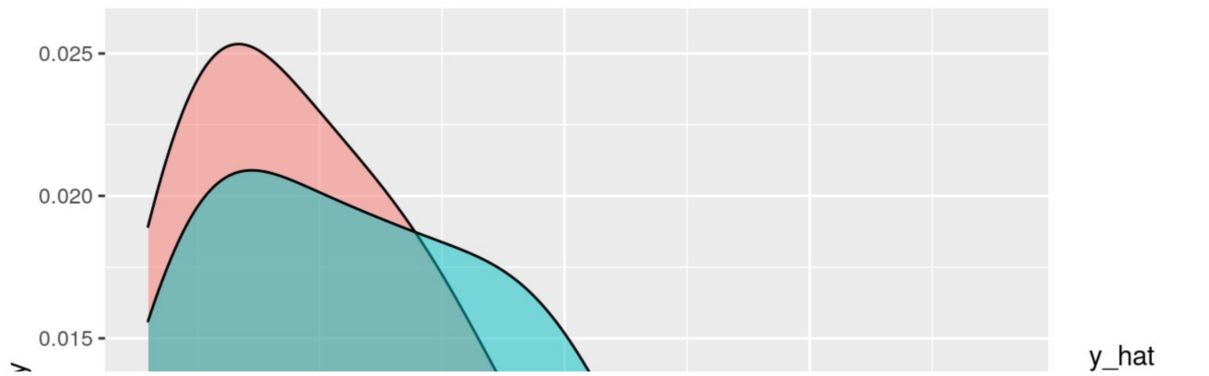
```
mean(y == y_hat)
```

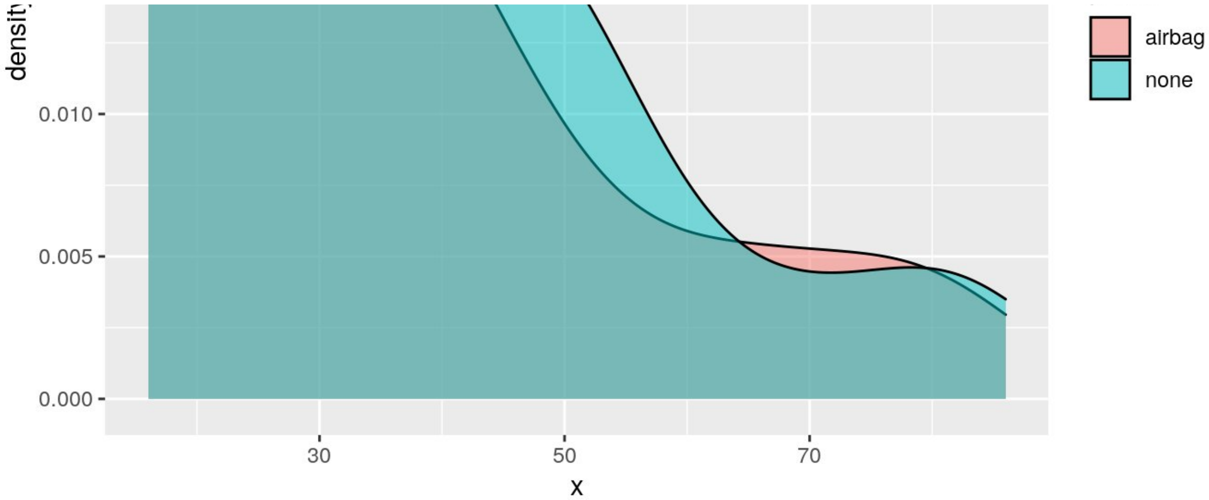
```
## [1] 0.4866667
```

```
ggplot(data.frame(x, y), aes(x)) + geom_density(aes(fill = y),
    alpha = 0.5)
```




```
ggplot(data.frame(x, y_hat), aes(x)) + geom_density(aes(fill = y_hat),
    alpha = 0.5)
```






```
# confusion matrix
table(actual = y, predicted = y_hat) %>% addmargins
```

```
##         predicted
## actual   airbag none Sum
##   none       34   31  65
##   airbag     42   43  85
##   Sum        76   74 150
```

```
actual <- c("problem", rep("no problem", 999))
predicted <- rep("no problem", 1000)
TPR <- mean(predicted[actual == "problem"] == "problem")
TNR <- mean(predicted[actual == "no problem"] == "no problem")
(TPR + TNR)/2
```

```
## [1] 0.5
```

```
# F1 score
F1 <- function(y, y_hat, positive) {
    sensitivity <- mean(y_hat[y == positive] == positive)
    precision <- mean(y[y_hat == positive] == positive)
    2 * (sensitivity * precision)/(sensitivity + precision)
}
F1(y, y_hat, "airbag")
```

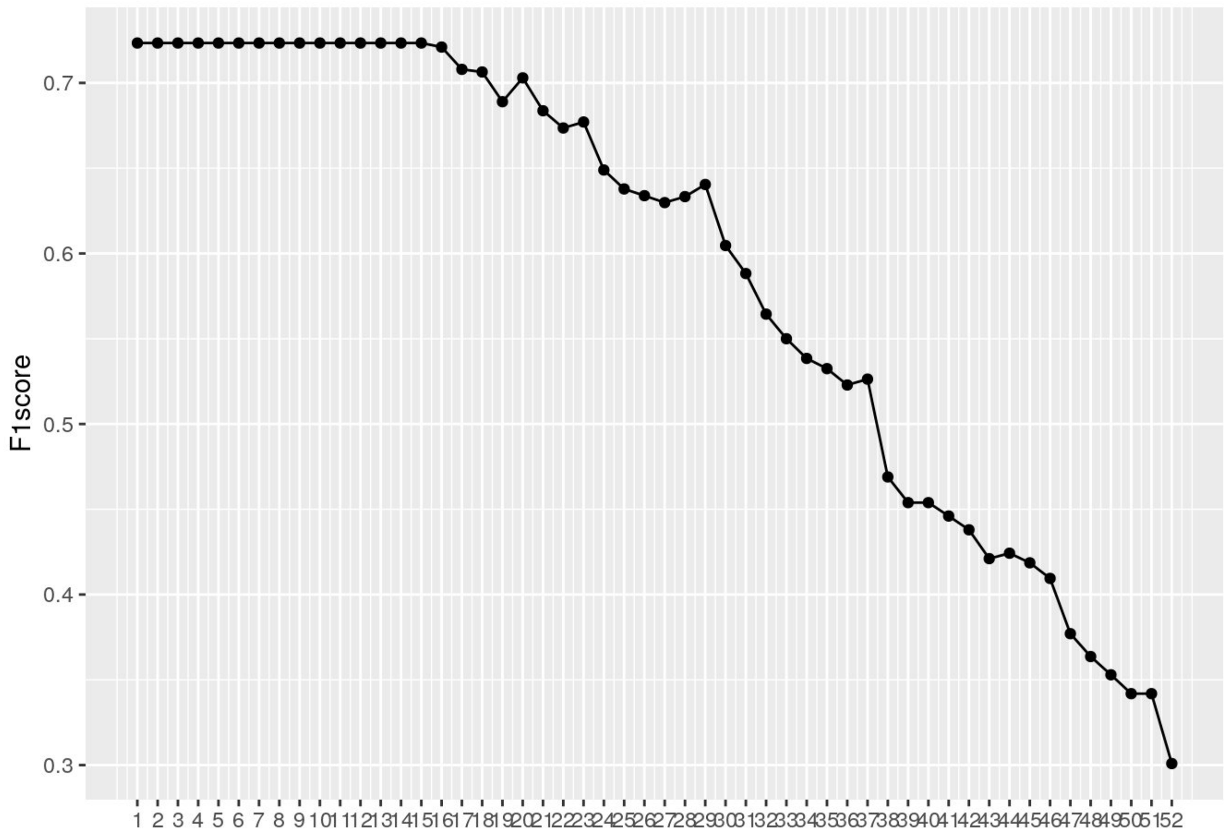
```
## [1] 0.5217391
```

```
n_distinct(tidycar$ageOFocc)
```

```
## [1] 55
```

```
F1score <- vector()
cutoff <- 1:52
for (i in cutoff) {
    y_hat <- ifelse(x > i, "airbag", "none")
```

```
    F1score[i] <- F1(y_hat, y, "airbag")
}
qplot(y = F1score) + geom_line() + scale_x_continuous(breaks = 1:52)
```



While observing the density graph of "airbag" variable, there is no visible difference between age of accident and airbag. There is jsut slight pattern towards the end where after age of 70, airbags decreases and none increases, maybe because older people are using old car which doesn't equip with airbags. Graph of y_hat is random. The confusion matrix tabulate actual vs predicted value of "airbag" variable. The F1 score is 0.52 which means there is no difference between airbag and none when compared with age. So according to the F1 plot, the cutoff should be around 16

```
# binary classification
class_diag(score = x, truth = y, positive = "airbag", cutoff = 16)
```

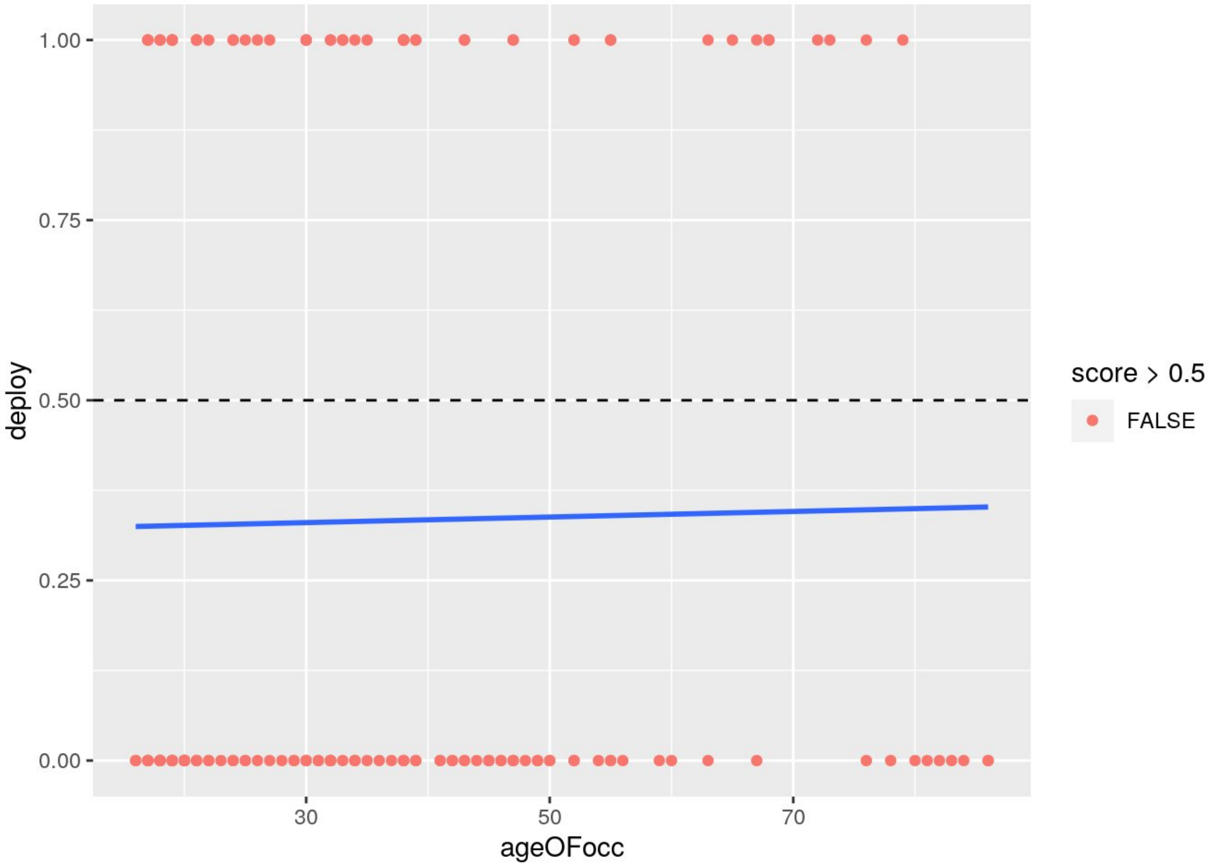
```
##            acc   sens   spec    ppv    f1     ba    auc
## Metrics 0.5667 0.9882 0.0154 0.5676 0.721 0.5018 0.5363
```

```
# linear classification
fit <- lm(deploy ~ ageOFocc, data = tidycar)
score <- predict(fit)
score %>% round(3)
```

```
## 24037 24103  3814  2342  4318 14845  6104  8631 20590  3511 16118 11761 13036
## 0.333 0.326 0.332 0.326 0.346 0.326 0.337 0.331 0.326 0.326 0.328 0.345 0.335
## 17452  3687  7889 24172 12326 15942 24692  2034 21632 18097 19366 11956 12768
## 0.326 0.338 0.348 0.333 0.352 0.332 0.335 0.325 0.331 0.326 0.352 0.326 0.326
## 11618 19483 22249  3842 24982 10208  5120 17275 10681 21200 13829  7954 22942
```

```
## 0.330 0.335 0.339 0.330 0.351 0.334 0.337 0.341 0.326 0.334 0.343 0.336 0.331
## 25929 26123 24604 21296 10823 17584    50 18399  9432  9310 20383 14064 24093
## 0.327 0.350 0.326 0.328 0.326 0.330 0.334 0.347 0.336 0.326 0.326 0.326 0.340
##  8626 14637 10946 12821 13126 15686 23605  7839   960  8384  9110 12323 25184
## 0.325 0.350 0.326 0.331 0.335 0.345 0.326 0.333 0.337 0.333 0.337 0.330 0.327
##  5101 18438 11915  4136  7167 12394 25742 18369 12911  7618 11877 15879  9980
## 0.333 0.336 0.325 0.340 0.331 0.325 0.345 0.340 0.332 0.337 0.326 0.331 0.325
## 14465 19825  1754 13387 23928 26064   866 15371 19328 19997  6439 20563 19409
## 0.330 0.330 0.338 0.331 0.329 0.327 0.329 0.350 0.331 0.348 0.349 0.328 0.329
## 20623  9307  6052 18136 18228   193 12565  2819 20099
## 0.351 0.339 0.326 0.339 0.330 0.326 0.329 0.329 0.326
##  [ reached getOption("max.print") -- omitted 50 entries ]
```

```
tidycar %>% mutate(score = score) %>% ggplot(aes(ageOFocc, deploy)) +
    geom_point(aes(color = score > 0.5)) + geom_smooth(method = "lm",
    se = F) + ylim(0, 1) + geom_hline(yintercept = 0.5, lty = 2)
```



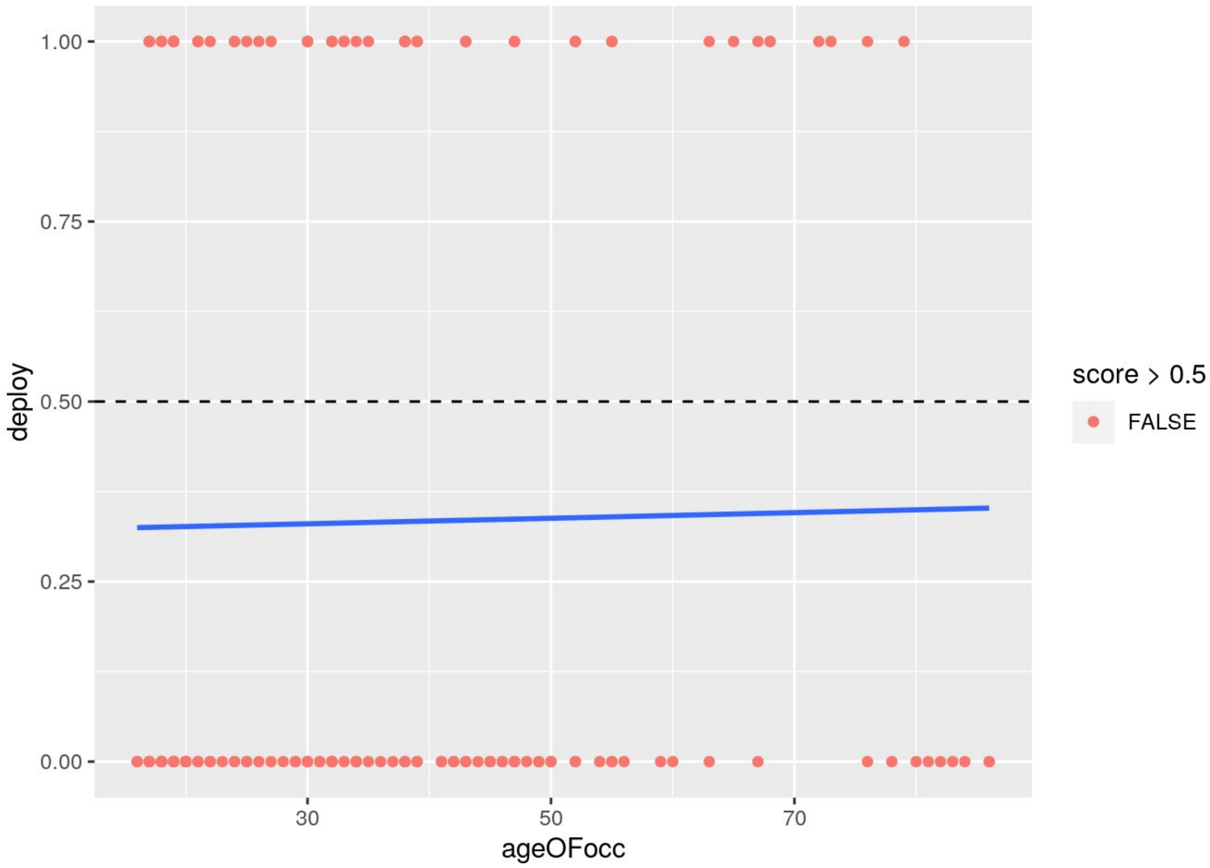

```
# logistic regression
class_diag(score, truth = tidycar$deploy, positive = 1)
```

```
##             acc sens spec ppv  f1  ba    auc
## Metrics 0.6667    0    1 NaN NaN 0.5 0.5119
```

```
fit <- glm(deploy ~ ageOFocc, data = tidycar, family = "binomial")
score <- predict(fit, type = "response")
score %>% round(3)
```

```
## 24037 24103  3814  2342  4318 14845  6104  8631 20590  3511 16118 11761 13036
## 0.333 0.326 0.332 0.326 0.347 0.326 0.337 0.331 0.326 0.326 0.328 0.345 0.335
## 17452  3687  7889 24172 12326 15942 24692  2034 21632 18097 19366 11956 12768
## 0.326 0.338 0.348 0.333 0.352 0.332 0.335 0.325 0.331 0.326 0.352 0.326 0.326
## 11618 19483 22249  3842 24982 10208  5120 17275 10681 21200 13829  7954 22942
## 0.330 0.335 0.339 0.330 0.351 0.334 0.337 0.341 0.326 0.334 0.343 0.336 0.331
## 25929 26123 24604 21296 10823 17584    50 18399  9432  9310 20383 14064 24093
## 0.327 0.350 0.326 0.328 0.326 0.330 0.334 0.347 0.336 0.326 0.326 0.326 0.340
##  8626 14637 10946 12821 13126 15686 23605  7839   960  8384  9110 12323 25184
## 0.325 0.350 0.326 0.331 0.335 0.345 0.326 0.333 0.337 0.333 0.337 0.330 0.327
##  5101 18438 11915  4136  7167 12394 25742 18369 12911  7618 11877 15879  9980
## 0.333 0.336 0.325 0.339 0.331 0.325 0.345 0.340 0.332 0.337 0.326 0.331 0.325
## 14465 19825  1754 13387 23928 26064   866 15371 19328 19997  6439 20563 19409
## 0.330 0.330 0.338 0.331 0.329 0.327 0.329 0.350 0.331 0.348 0.349 0.328 0.329
## 20623  9307  6052 18136 18228   193 12565  2819 20099
## 0.351 0.339 0.326 0.339 0.330 0.326 0.329 0.329 0.326
##  [ reached getOption("max.print") -- omitted 50 entries ]
```

```
tidycar %>% mutate(score = score) %>% ggplot(aes(ageOFocc, deploy)) +
    geom_point(aes(color = score > 0.5)) + geom_smooth(method = "glm",
    se = F, method.args = list(family = "binomial")) + ylim(0,
    1) + geom_hline(yintercept = 0.5, lty = 2)
```




```
# predicting a binary variable (response) from ALL of the
# rest of the numeric variables in your dataset
num_data <- tidycar %>% select(deploy, ageOFocc, weight, frontal,
    yearacc, yearVeh, injSeverity)
fit <- glm(deploy ~ ., data = num_data, family = "binomial")
score <- predict(fit, type = "response")
class_diag(score, tidycar$deploy, positive = 1)
```

```
##            acc sens spec    ppv    f1   ba    auc
## Metrics 0.8533 0.72 0.92 0.8182 0.766 0.82 0.8954
```

The AUC of binary classification is 0.53 which means the model is very bad. The AUC of logistic regression is not good either with 0.51. The lm and glm method graph is not proper (doesn't tell anything) as there is no visible pattern between age and having airbag or none in the car. But the AUC of glm model of all the numeric variable is 0.895 which is much much better then just "ageOFocc" and "airbag".

Non-Parametric Classifier

```
library(caret)
knn_fit <- knn3(deploy == 1 ~ ., data = num_data)
y_hat_knn <- predict(knn_fit, num_data)
y_hat_knn
```

```
##        FALSE TRUE
##   [1,]   0.6  0.4
##   [2,]   0.6  0.4
##   [3,]   0.6  0.4
##   [4,]   0.4  0.6
##   [5,]   0.6  0.4
##   [6,]   0.6  0.4
##   [7,]   0.6  0.4
##   [8,]   1.0  0.0
##   [9,]   0.2  0.8
##  [10,]   1.0  0.0
##  [11,]   0.8  0.2
##  [12,]   0.6  0.4
##  [13,]   1.0  0.0
##  [14,]   1.0  0.0
##  [15,]   0.6  0.4
##  [16,]   0.6  0.4
##  [17,]   0.0  1.0
##  [18,]   0.8  0.2
##  [19,]   0.2  0.8
##  [20,]   0.4  0.6
##  [21,]   1.0  0.0
##  [22,]   0.8  0.2
##  [23,]   0.4  0.6
##  [24,]   0.6  0.4
##  [25,]   1.0  0.0
##  [26,]   0.6  0.4
##  [27,]   0.8  0.2
##  [28,]   1.0  0.0
##  [29,]   0.8  0.2
##  [30,]   0.6  0.4
##  [31,]   1.0  0.0
##  [32,]   0.6  0.4
##  [33,]   0.8  0.2
##  [34,]   0.6  0.4
##  [35,]   1.0  0.0
```

```
## [36,]   0.2  0.8
## [37,]   0.2  0.8
## [38,]   0.8  0.2
## [39,]   0.6  0.4
## [40,]   0.4  0.6
## [41,]   0.2  0.8
## [42,]   0.4  0.6
## [43,]   0.6  0.4
## [44,]   0.8  0.2
## [45,]   0.6  0.4
## [46,]   0.8  0.2
## [47,]   0.8  0.2
## [48,]   0.8  0.2
## [49,]   1.0  0.0
## [50,]   0.4  0.6
##  [ reached getOption("max.print") -- omitted 100 rows ]
```

```
class_diag(y_hat_knn[, 2], num_data$deploy, positive = 1)
```

```
##          acc sens spec    ppv     f1    ba    auc
## Metrics 0.74 0.48 0.87 0.6486 0.5517 0.675 0.7791
```

```
# k-fold CV
set.seed(312)
k = 10  #choose number of folds
data <- num_data[sample(nrow(num_data)), ]  #randomly order rows
folds <- cut(seq(1:nrow(num_data)), breaks = k, labels = F)  #create 10 folds
diags <- NULL
for (i in 1:k) {
    ## Create training and test sets
    train <- data[folds != i, ]
    test <- data[folds == i, ]
    truth <- test$deploy
    ## Train model on training set
    fit <- glm(deploy ~ ., data = num_data, family = "binomial")
    probs <- predict(fit, newdata = test, type = "response")
    ## Test model on test set (save all k results)
    diags <- rbind(diags, class_diag(probs, truth, positive = 1))
}
summarize_all(diags, mean)
```

```
##       acc     sens     spec      ppv       f1       ba      auc
## 1 0.85335 0.71833 0.92045 0.81309 0.74763 0.81937 0.89491
```

```
# k-fold CV with kNN

k = 10  #choose number of folds
data <- num_data[sample(nrow(num_data)), ]  #randomly order rows
folds <- cut(seq(1:nrow(num_data)), breaks = k, labels = F)  #create 10 folds
diags <- NULL
for (i in 1:k) {
    ## Create training and test sets
    train <- data[folds != i, ]
```

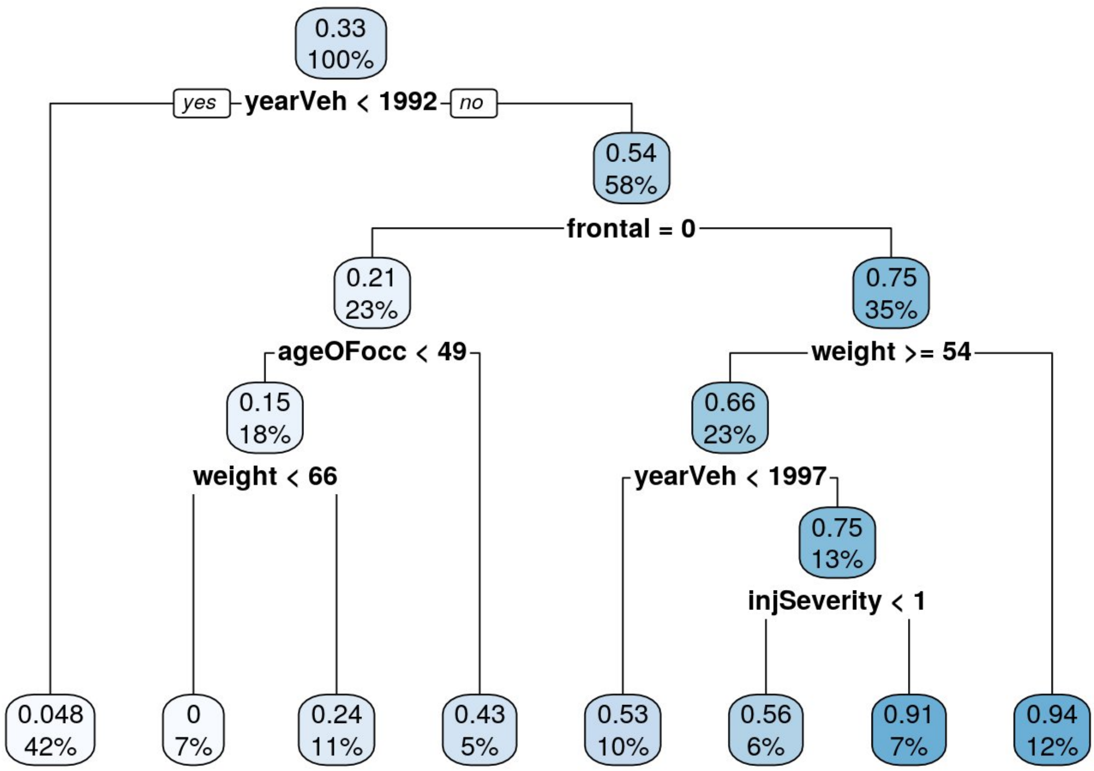
```
    test <- data[folds == i, ]
    truth <- test$deploy
    ## Train model on training set
    fit <- knn3(deploy ~ ., data = train)
    probs <- predict(fit, newdata = test)[, 2]
    ## Test model on test set (save all k results)
    diags <- rbind(diags, class_diag(probs, truth, positive = 1))
}
summarize_all(diags, mean)
```

```
##    acc    sens    spec  ppv  f1      ba    auc
## 1 0.64 0.30525 0.81155 0.45 NaN 0.55838 0.5476
```

I predicted the probability of true which means the vehical have airbags, or prob of false, which means beehical does not have airbags or all the 150 observation in the dataset using y_hat and knn fit. The knn analysis AUC is 0.779 which is not bad but it could be better. The AUC of k-fold CV is 0.895 which is very good compared to other AUCs. The AUC of k-fold CV with kNN is 0.547 which is worst then k-fold CV, we can see that doing k-fold with kNN analysis our AUC have gone down drastically.

Regression/Numeric Prediction

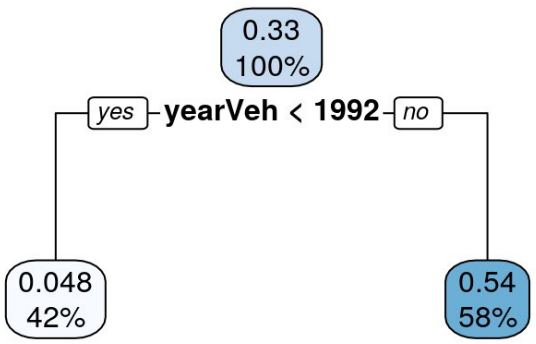
```
# classification tree
library(rpart)
library(rpart.plot)
fit <- rpart(deploy ~ ., data = num_data)
rpart.plot(fit)
```



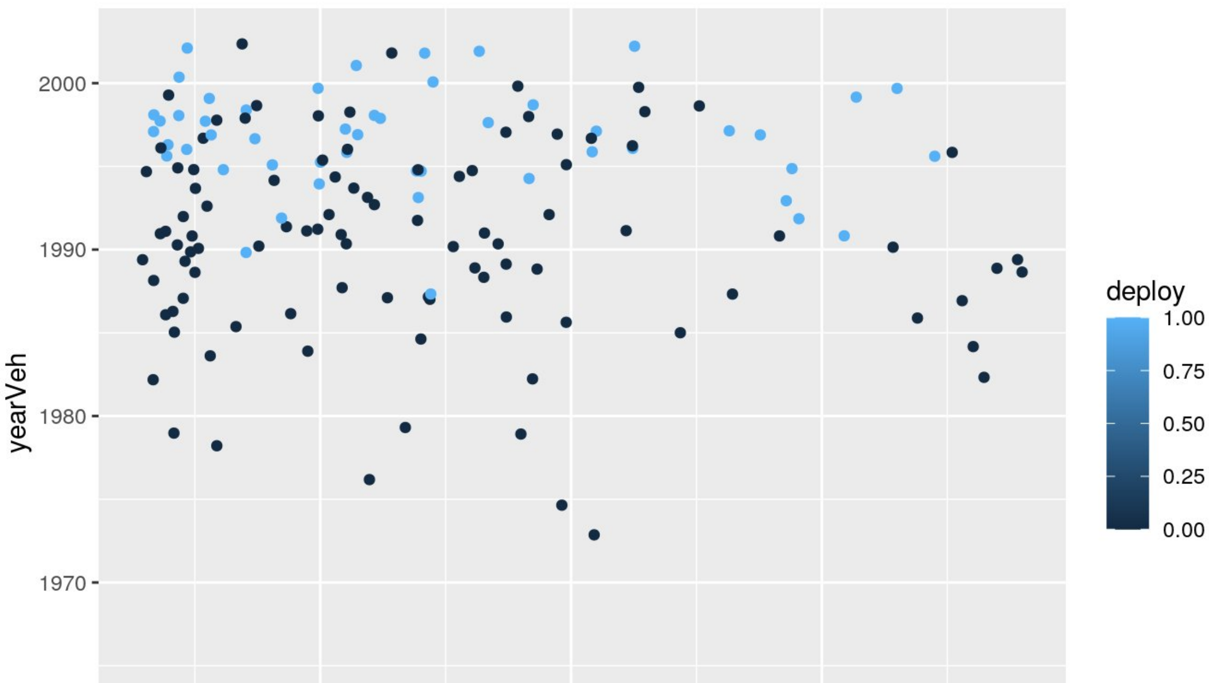
```
fit <- train(deploy ~ ., data = num_data, method = "rpart")
fit$bestTune
```

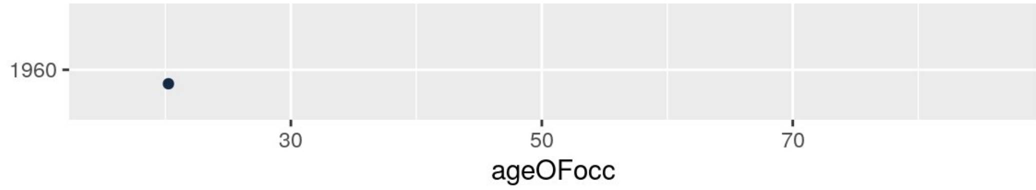
```
##          cp
## 2 0.1871715
```

```
rpart.plot(fit$finalModel)
```




```
num_data %>% ggplot(aes(ageOFocc, yearVeh)) + geom_jitter(aes(color = deploy))
```






```
fit <- lm(deploy ~ ., data = num_data)  #predict deploy from all other variables
yhat <- predict(fit)  #predicted deploy
mean((num_data$deploy - yhat)^2)  #mean squared error (MSE)
```

```
## [1] 0.1448411
```

```
# cross validation with kNN Regression

set.seed(1234)
k = 5  #choose number of folds
data <- num_data %>% sample_frac()  #randomly order rows
folds <- cut(seq(1:nrow(num_data)), breaks = k, labels = F)  #create folds
diags <- NULL
for (i in 1:k) {
    train <- data[folds != i, ]
    test <- data[folds == i, ]
    ## Fit linear regression model to training set
    fit <- knnreg(deploy ~ ., data = train)
    ## Get predictions/y-hats on test set (fold i)
    yhat <- predict(fit, newdata = test)
    ## Compute prediction error (MSE) for fold i
    diags <- mean((test$deploy - yhat)^2)
}
mean(diags)
```

```
## [1] 0.2093333
```

I produced classification tree for the numeric variable in the dataset. It gives the probability of occurrence if the value of variable is true. And the total probability all the end adds up to 1. The cp of the fit is 0.187. Next, I plotter scatter plot of ageOFacc vs. yearVeh with color of deploy. The graph tells that majority of airbags are deployed if the vehicle's year is 1995 or newer. The MSE of data is 0.144. The mean diags of cross validation with kNN regression is 0.209, which is greater than MSE which means our model is doing well.

Python

```
library(reticulate)
use_python("/usr/bin/python3", required = F)
py_install("pandas")
```

```
import pandas as pd
pd.set_option('display.max_columns', None)

ca = r.tidycar
ca.head()
```

```
##        dvcat    weight   dead  airbag seatbelt  frontal sex  ageOFocc  \
## 24037  10-24  1157.649  alive  airbag   belted      0.0   f      38.0
## 24103  25-39    41.848  alive  airbag   belted      1.0   f      18.0
## 3814   40-54    27.197  alive  airbag   belted      0.0   f      34.0
## 2342   25-39    53.770  alive  airbag   belted      0.0   f      20.0
## 4318   10-24   135.833  alive  airbag     none      1.0   m      72.0
##
##        yearacc  yearVeh occRole  deploy  injSeverity
## 24037   2002.0   1992.0  driver     0.0          1.0
## 24103   2002.0   1996.0  driver     1.0          2.0
## 3814    1997.0   1993.0  driver     0.0          2.0
## 2342    1997.0   1995.0  driver     0.0          3.0
## 4318    1998.0   1991.0  driver     1.0          1.0
```

```
filter_data = (ca.filter(['airbag', 'injSeverity'])
 .query('airbag == "airbag"').head(10))
filter_data
#filtering in python
```

```
##        airbag  injSeverity
## 24037  airbag          1.0
## 24103  airbag          2.0
## 3814   airbag          2.0
## 2342   airbag          3.0
## 4318   airbag          1.0
## 6104   airbag          0.0
## 20590  airbag          0.0
## 11761  airbag          0.0
## 24172  airbag          0.0
## 15942  airbag          2.0
```

```
# converting python dataset again to R
py$filter_data
```

```
##       airbag injSeverity
## 24037 airbag           1
## 24103 airbag           2
## 3814  airbag           2
## 2342  airbag           3
## 4318  airbag           1
## 6104  airbag           0
## 20590 airbag           0
## 11761 airbag           0
## 24172 airbag           0
## 15942 airbag           2
```

For python chuck of code in R-markdown, I install pandas and reticulate package. I converted R dataset to python using r. and filtered the data using .filter function. Then I also converted python dataset to R dataset using py$

Concluding Remarks

This was very interesting data to work to as it had many valuable variables. Overall I saw the trend that for scatter plot of “ageOFocc” vs “vearVeh” variables, there is two possible clusters. Although the linear classifier for “airbag” variable was not predictable (no trend seen), by using all numeric variable, I still got good AUC of 0.89 compared to other AUCs in this project. Lastly it was very interesting to see python and R working side by side as each of them have their own benefits and drawbacks.