Car Insurance Claims

The miners

2022-12-11

# Libraries  
library(naniar) # Missing values  
library(ggplot2) # Graphics  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

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

library(tidyr)  
library(randomForest) # Random forest

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(corrplot) # Plot correlations

## corrplot 0.92 loaded

library(knitr) #Table presentation  
library(GGally) # Correlation and scatter plots

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

library(class) # knn  
library(rpart) # trees  
library(caret) # preprocess data

## Loading required package: lattice

library(rpart.plot) # plot trees  
library(forecast) # accuracy

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

library(neuralnet) # Neural netwok

##   
## Attaching package: 'neuralnet'

## The following object is masked from 'package:dplyr':  
##   
## compute

library(CustomerScoringMetrics) #Lift and gain  
library(data.table)

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

source("PlotResidLogist.R") # Deviance Residuals  
  
library(doParallel)#Accelerating computations by setting CPU to work in parallel

## Loading required package: foreach

## Loading required package: iterators

## Loading required package: parallel

cores = detectCores()  
cl <- makeCluster(cores-1)  
registerDoParallel(cl)

# Introduction

Being able to predict the risk of a policyholder’s complaint is fundamental for an insurance company. It could impact its profitability. We decided to conduct an analysis about that industry to better understand and identify the complaint phenomenon with regards to a car insurance business. Specifically, we want to predict whether an insured is going to claim a file. We will try to achieve that through different algorithmic classification methods and by trying to identify the best classification model for such a situation.

A dataset containing information about insurance holders and details about them is going to be used to achive our goal. The dataset is large and comes from the Kaggle web platform. It contains 58’500 observations and 44 variables. Among this information, we find attributes such as, the population density of the insured’s city, the age of the policyholder, the age of the insured car, the car model, etc. However, we are particularly interested in the explained variable “Is\_claim” It is a boolean indicator showing if a policyholder filed a claim. A positive case will be denoted by 1 and 0 in the opposite case. Therefore we will try to predict it.

To conduct our research, we plan to undertake the following steps. First, we will have a first look at the data and drop some variables, because our data set is very large and requires a lot of computation power. We will continue with a regular exploratory data analysis to understand our data. Then, we will explore different methods of classification. After this step, We are going to explore classification tree, k-nearest neighbors, neural network, logistic regression and ensembles. We will finish by comparing these method to identify the most suitable for our case.

### Dataset preparation

We need to manipulate our data before starting the Exploratory Data Analysis part in order to reduce the amount of data and making our computers ables to handle the dataset.

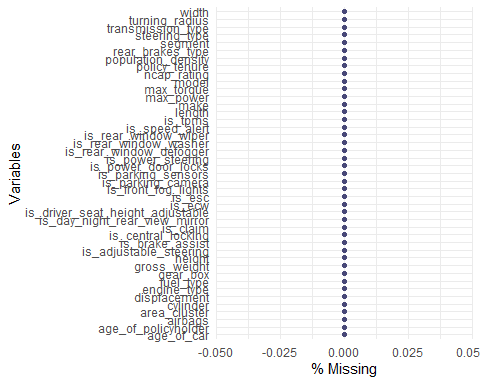
cars <- read.csv("train.csv", sep=",", header = TRUE)# Original data set  
  
str(cars) # types of variable and dimensions

## 'data.frame': 58592 obs. of 44 variables:  
## $ policy\_id : chr "ID00001" "ID00002" "ID00003" "ID00004" ...  
## $ policy\_tenure : num 0.516 0.673 0.841 0.9 0.596 ...  
## $ age\_of\_car : num 0.05 0.02 0.02 0.11 0.11 0.07 0.16 0.14 0.07 0.04 ...  
## $ age\_of\_policyholder : num 0.644 0.375 0.385 0.433 0.635 ...  
## $ area\_cluster : chr "C1" "C2" "C3" "C4" ...  
## $ population\_density : int 4990 27003 4076 21622 34738 13051 6112 8794 6112 17804 ...  
## $ make : int 1 1 1 1 2 3 4 1 3 1 ...  
## $ segment : chr "A" "A" "A" "C1" ...  
## $ model : chr "M1" "M1" "M1" "M2" ...  
## $ fuel\_type : chr "CNG" "CNG" "CNG" "Petrol" ...  
## $ max\_torque : chr "60Nm@3500rpm" "60Nm@3500rpm" "60Nm@3500rpm" "113Nm@4400rpm" ...  
## $ max\_power : chr "40.36bhp@6000rpm" "40.36bhp@6000rpm" "40.36bhp@6000rpm" "88.50bhp@6000rpm" ...  
## $ engine\_type : chr "F8D Petrol Engine" "F8D Petrol Engine" "F8D Petrol Engine" "1.2 L K12N Dualjet" ...  
## $ airbags : int 2 2 2 2 2 6 2 2 6 6 ...  
## $ is\_esc : chr "No" "No" "No" "Yes" ...  
## $ is\_adjustable\_steering : chr "No" "No" "No" "Yes" ...  
## $ is\_tpms : chr "No" "No" "No" "No" ...  
## $ is\_parking\_sensors : chr "Yes" "Yes" "Yes" "Yes" ...  
## $ is\_parking\_camera : chr "No" "No" "No" "Yes" ...  
## $ rear\_brakes\_type : chr "Drum" "Drum" "Drum" "Drum" ...  
## $ displacement : int 796 796 796 1197 999 1493 1497 1197 1493 1197 ...  
## $ cylinder : int 3 3 3 4 3 4 4 4 4 4 ...  
## $ transmission\_type : chr "Manual" "Manual" "Manual" "Automatic" ...  
## $ gear\_box : int 5 5 5 5 5 6 5 5 6 5 ...  
## $ steering\_type : chr "Power" "Power" "Power" "Electric" ...  
## $ turning\_radius : num 4.6 4.6 4.6 4.8 5 5.2 5 4.8 5.2 4.85 ...  
## $ length : int 3445 3445 3445 3995 3731 4300 3990 3845 4300 3990 ...  
## $ width : int 1515 1515 1515 1735 1579 1790 1755 1735 1790 1745 ...  
## $ height : int 1475 1475 1475 1515 1490 1635 1523 1530 1635 1500 ...  
## $ gross\_weight : int 1185 1185 1185 1335 1155 1720 1490 1335 1720 1410 ...  
## $ is\_front\_fog\_lights : chr "No" "No" "No" "Yes" ...  
## $ is\_rear\_window\_wiper : chr "No" "No" "No" "No" ...  
## $ is\_rear\_window\_washer : chr "No" "No" "No" "No" ...  
## $ is\_rear\_window\_defogger : chr "No" "No" "No" "Yes" ...  
## $ is\_brake\_assist : chr "No" "No" "No" "Yes" ...  
## $ is\_power\_door\_locks : chr "No" "No" "No" "Yes" ...  
## $ is\_central\_locking : chr "No" "No" "No" "Yes" ...  
## $ is\_power\_steering : chr "Yes" "Yes" "Yes" "Yes" ...  
## $ is\_driver\_seat\_height\_adjustable: chr "No" "No" "No" "Yes" ...  
## $ is\_day\_night\_rear\_view\_mirror : chr "No" "No" "No" "Yes" ...  
## $ is\_ecw : chr "No" "No" "No" "Yes" ...  
## $ is\_speed\_alert : chr "Yes" "Yes" "Yes" "Yes" ...  
## $ ncap\_rating : int 0 0 0 2 2 3 5 2 3 0 ...  
## $ is\_claim : int 0 0 0 0 0 0 0 0 0 0 ...

cars <- cars[,c(-1)] # Drop ID  
  
sapply(cars,function(x) length(unique(x))) # Check unique values

## policy\_tenure age\_of\_car   
## 58592 49   
## age\_of\_policyholder area\_cluster   
## 75 22   
## population\_density make   
## 22 5   
## segment model   
## 6 11   
## fuel\_type max\_torque   
## 3 9   
## max\_power engine\_type   
## 9 11   
## airbags is\_esc   
## 3 2   
## is\_adjustable\_steering is\_tpms   
## 2 2   
## is\_parking\_sensors is\_parking\_camera   
## 2 2   
## rear\_brakes\_type displacement   
## 2 9   
## cylinder transmission\_type   
## 2 2   
## gear\_box steering\_type   
## 2 3   
## turning\_radius length   
## 9 9   
## width height   
## 10 11   
## gross\_weight is\_front\_fog\_lights   
## 10 2   
## is\_rear\_window\_wiper is\_rear\_window\_washer   
## 2 2   
## is\_rear\_window\_defogger is\_brake\_assist   
## 2 2   
## is\_power\_door\_locks is\_central\_locking   
## 2 2   
## is\_power\_steering is\_driver\_seat\_height\_adjustable   
## 2 2   
## is\_day\_night\_rear\_view\_mirror is\_ecw   
## 2 2   
## is\_speed\_alert ncap\_rating   
## 2 5   
## is\_claim   
## 2

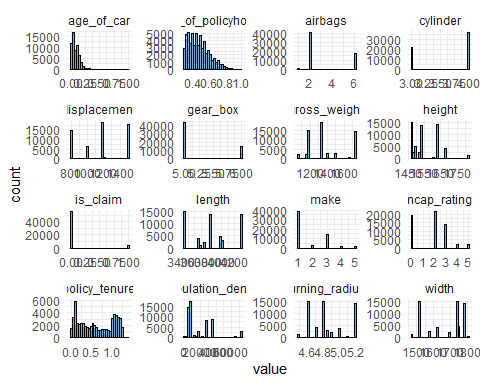
gg\_miss\_var(cars, show\_pct = TRUE) # See if any missing value



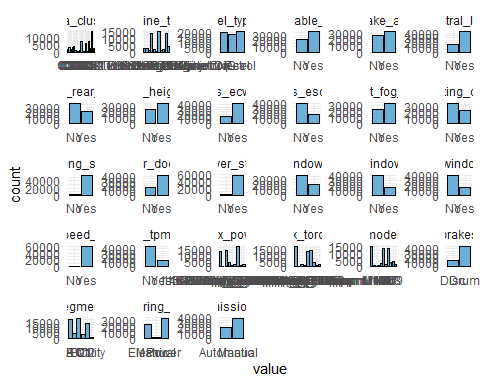
Our original data set is precisely composed of 58592 observations and 44 variables. Variables are numerical and categorical where some of them are booleans. We notice the presence of variables that might be less relevant. We could drop them in order to alleviate computation resources. Let’s start with the variable “ID” which is of no help.

#Histograms  
cars %>% select(1:3, 5:6,13,20:21,23,25:29,42:43) %>% gather() %>%   
 ggplot(aes(value)) +   
 facet\_wrap(~ key, scales = "free") + # Display figures in many facets  
 geom\_histogram(color = "black", fill = "#6baed6") +   
 theme\_minimal()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



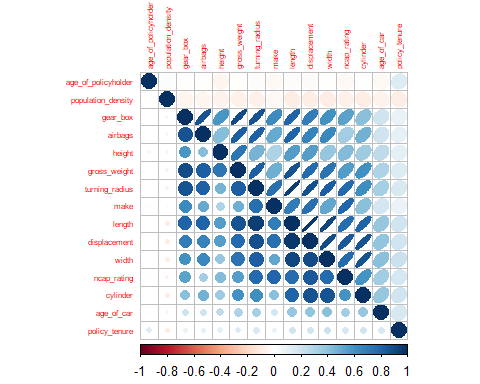
# Barcharts  
cars %>% select(4,7:12,14:19,22,24,30:41) %>% gather() %>%  
 ggplot(aes(x = value)) +  
 facet\_wrap(~ key, scales = "free") + # Display figures in many facets  
 geom\_bar(color = "black", fill = "#6baed6") +  
 theme\_minimal()



cars\_numeric <- cars[,c(1:3,5:6,13,20:21,23,25:29,42)] # Only numerical values  
  
# Correlations  
cor2 <- data.frame(round(cor(cars\_numeric),3)) # Table with correlations  
cor2

## policy\_tenure age\_of\_car age\_of\_policyholder  
## policy\_tenure 1.000 0.166 0.144  
## age\_of\_car 0.166 1.000 -0.035  
## age\_of\_policyholder 0.144 -0.035 1.000  
## population\_density -0.100 -0.062 0.010  
## make 0.086 0.188 -0.032  
## airbags 0.104 0.209 -0.008  
## displacement 0.194 0.393 -0.024  
## cylinder 0.191 0.380 0.004  
## gear\_box 0.095 0.202 -0.003  
## turning\_radius 0.166 0.333 -0.017  
## length 0.191 0.383 -0.020  
## width 0.213 0.414 -0.006  
## height 0.119 0.259 -0.054  
## gross\_weight 0.141 0.302 -0.008  
## ncap\_rating 0.173 0.349 -0.032  
## population\_density make airbags displacement cylinder  
## policy\_tenure -0.100 0.086 0.104 0.194 0.191  
## age\_of\_car -0.062 0.188 0.209 0.393 0.380  
## age\_of\_policyholder 0.010 -0.032 -0.008 -0.024 0.004  
## population\_density 1.000 -0.035 -0.060 -0.091 -0.092  
## make -0.035 1.000 0.502 0.753 0.411  
## airbags -0.060 0.502 1.000 0.661 0.479  
## displacement -0.091 0.753 0.661 1.000 0.866  
## cylinder -0.092 0.411 0.479 0.866 1.000  
## gear\_box -0.057 0.633 0.860 0.692 0.410  
## turning\_radius -0.078 0.754 0.811 0.875 0.616  
## length -0.092 0.692 0.809 0.962 0.805  
## width -0.098 0.512 0.640 0.899 0.862  
## height -0.066 0.303 0.424 0.555 0.352  
## gross\_weight -0.078 0.481 0.829 0.776 0.603  
## ncap\_rating -0.071 0.792 0.342 0.847 0.598  
## gear\_box turning\_radius length width height gross\_weight  
## policy\_tenure 0.095 0.166 0.191 0.213 0.119 0.141  
## age\_of\_car 0.202 0.333 0.383 0.414 0.259 0.302  
## age\_of\_policyholder -0.003 -0.017 -0.020 -0.006 -0.054 -0.008  
## population\_density -0.057 -0.078 -0.092 -0.098 -0.066 -0.078  
## make 0.633 0.754 0.692 0.512 0.303 0.481  
## airbags 0.860 0.811 0.809 0.640 0.424 0.829  
## displacement 0.692 0.875 0.962 0.899 0.555 0.776  
## cylinder 0.410 0.616 0.805 0.862 0.352 0.603  
## gear\_box 1.000 0.862 0.809 0.602 0.580 0.895  
## turning\_radius 0.862 1.000 0.945 0.826 0.460 0.823  
## length 0.809 0.945 1.000 0.916 0.554 0.862  
## width 0.602 0.826 0.916 1.000 0.389 0.734  
## height 0.580 0.460 0.554 0.389 1.000 0.728  
## gross\_weight 0.895 0.823 0.862 0.734 0.728 1.000  
## ncap\_rating 0.530 0.779 0.768 0.772 0.437 0.556  
## ncap\_rating  
## policy\_tenure 0.173  
## age\_of\_car 0.349  
## age\_of\_policyholder -0.032  
## population\_density -0.071  
## make 0.792  
## airbags 0.342  
## displacement 0.847  
## cylinder 0.598  
## gear\_box 0.530  
## turning\_radius 0.779  
## length 0.768  
## width 0.772  
## height 0.437  
## gross\_weight 0.556  
## ncap\_rating 1.000

# Plot correlations  
corrplot(as.matrix(cor2), # Plot of upper right part  
 order = 'AOE',  
 type = 'lower',  
 tl.pos = "lt",  
 tl.cex = 0.5,)  
  
corrplot(as.matrix(cor2), # Plot of lower left part  
 add = TRUE, type = 'upper',  
 method = 'ellipse',  
 order = 'AOE',  
 diag = FALSE,  
 tl.pos = 'n',  
 cl.pos = 'n',  
 tl.cex = 0.5)



# Dropping variables based on correlation  
cars\_reduced <- cars[,-c(20,13,23,25,29)]

Looking at correlations we note a high correlation for the variable displacement such as cylinder (0.87), turning\_radius with length (0.95), gross\_weight with length (0.86), width with length (0.92).

displacement, airbags, gear\_box, turning\_radius, gross\_weight

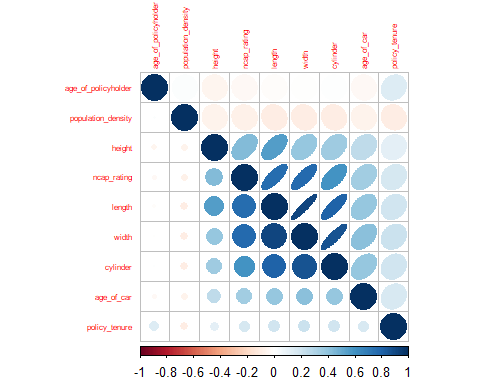
#Random forest  
# rf\_fit <- randomForest(as.factor(cars$is\_claim) ~ .,#Fit  
# data = cars\_reduced,  
# ntree = 500,   
# mtry = 4, # variables randomly sampled as candidates at each # split.   
# importance = TRUE)  
#   
# varImpPlot(rf\_fit, type = 1) # Variable importance  
# sort(round(rf\_fit$importance[,4],3), decreasing = TRUE)[1:19]  
  
# Variables keeped by the random forest  
cars\_reduced\_rf <- cars\_reduced[,c("policy\_tenure","age\_of\_car",  
 "age\_of\_policyholder","population\_density",  
 "area\_cluster","height","width","segment",  
 "model","length","engine\_type","max\_torque",  
 "max\_power","ncap\_rating","cylinder")]

To select important variables we also use a random forest method which sort variables by decrease in Gini score. The ranking showing the important variables recommends at least these 15 variables: policy\_tenure, age\_of\_car, age\_of\_policyholder, population\_density, area\_cluster, height, width, segment, model, length, engine\_type, max\_torque, max\_power, ncap\_rating and cylinder.

# Correlations  
cor3 <- data.frame(round(cor(cars\_reduced\_rf[,c(1:4,6,7,10,14,15)]),3))  
cor3

## policy\_tenure age\_of\_car age\_of\_policyholder  
## policy\_tenure 1.000 0.166 0.144  
## age\_of\_car 0.166 1.000 -0.035  
## age\_of\_policyholder 0.144 -0.035 1.000  
## population\_density -0.100 -0.062 0.010  
## height 0.119 0.259 -0.054  
## width 0.213 0.414 -0.006  
## length 0.191 0.383 -0.020  
## ncap\_rating 0.173 0.349 -0.032  
## cylinder 0.191 0.380 0.004  
## population\_density height width length ncap\_rating  
## policy\_tenure -0.100 0.119 0.213 0.191 0.173  
## age\_of\_car -0.062 0.259 0.414 0.383 0.349  
## age\_of\_policyholder 0.010 -0.054 -0.006 -0.020 -0.032  
## population\_density 1.000 -0.066 -0.098 -0.092 -0.071  
## height -0.066 1.000 0.389 0.554 0.437  
## width -0.098 0.389 1.000 0.916 0.772  
## length -0.092 0.554 0.916 1.000 0.768  
## ncap\_rating -0.071 0.437 0.772 0.768 1.000  
## cylinder -0.092 0.352 0.862 0.805 0.598  
## cylinder  
## policy\_tenure 0.191  
## age\_of\_car 0.380  
## age\_of\_policyholder 0.004  
## population\_density -0.092  
## height 0.352  
## width 0.862  
## length 0.805  
## ncap\_rating 0.598  
## cylinder 1.000

corrplot(as.matrix(cor3), # Plot of lower left part  
 order = 'AOE',  
 type = 'lower',  
 tl.pos = "lt",  
 tl.cex = 0.5,)  
  
corrplot(as.matrix(cor3), add = TRUE, # Plot of upper right part  
 type = 'upper',  
 method = 'ellipse',  
 order = 'AOE',  
 diag = FALSE,  
 tl.pos = 'n',  
 cl.pos = 'n',  
 tl.cex = 0.5)



# Filtering variables based on domain knowledge and correlations  
cars\_reduced <- cars\_reduced\_rf[,-c( 7,8,11,12,13,14,15)]# Drop variables  
  
cars\_final <- cbind(cars\_reduced, cars$is\_claim) # Bind explanatory variable  
  
colnames(cars\_final)[9] <- "is\_claim" # Rename column

We drop the variables width, segment, engine\_type, max\_torque, max\_power, ncap\_rating and rating.

## CHECK code

# We drop height and weight since model defines these features.  
  
logistic\_regression\_final\_variables <- glm(is\_claim~.,  
 family=binomial(link='logit'),  
 data= cars\_final[,-c(5,7)])  
  
summary(logistic\_regression\_final\_variables)

##   
## Call:  
## glm(formula = is\_claim ~ ., family = binomial(link = "logit"),   
## data = cars\_final[, -c(5, 7)])  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.5451 -0.4048 -0.3417 -0.2963 2.9057   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.977e+00 3.578e-01 -8.319 < 2e-16 \*\*\*  
## policy\_tenure 8.416e-01 4.417e-02 19.056 < 2e-16 \*\*\*  
## age\_of\_car -3.526e+00 3.510e-01 -10.044 < 2e-16 \*\*\*  
## age\_of\_policyholder 2.827e-01 1.363e-01 2.074 0.03808 \*   
## population\_density -2.979e-06 1.030e-06 -2.892 0.00383 \*\*   
## height -3.805e-04 2.666e-04 -1.427 0.15363   
## length 1.232e-04 7.025e-05 1.754 0.07945 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 27860 on 58591 degrees of freedom  
## Residual deviance: 27366 on 58585 degrees of freedom  
## AIC: 27380  
##   
## Number of Fisher Scoring iterations: 5

We can see that the coefficient NA for cluster9 is symptomatic of a multicolinearity issue. Therefore, we remove area cluster and keep population instead.

logistic\_regression\_final\_variables <- glm(is\_claim~.,  
 family=binomial(link='logit')  
 , data= cars\_final[,-c(5,7,4)])  
  
summary(logistic\_regression\_final\_variables)

##   
## Call:  
## glm(formula = is\_claim ~ ., family = binomial(link = "logit"),   
## data = cars\_final[, -c(5, 7, 4)])  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.5498 -0.4048 -0.3421 -0.2974 2.8968   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -3.091e+00 3.557e-01 -8.690 <2e-16 \*\*\*  
## policy\_tenure 8.525e-01 4.404e-02 19.359 <2e-16 \*\*\*  
## age\_of\_car -3.504e+00 3.509e-01 -9.986 <2e-16 \*\*\*  
## age\_of\_policyholder 2.754e-01 1.363e-01 2.021 0.0433 \*   
## height -3.709e-04 2.666e-04 -1.391 0.1642   
## length 1.336e-04 7.015e-05 1.904 0.0569 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 27860 on 58591 degrees of freedom  
## Residual deviance: 27374 on 58586 degrees of freedom  
## AIC: 27386  
##   
## Number of Fisher Scoring iterations: 5

cars\_final <- cars\_final[,-c(5,6,8)]  
kable(head(cars\_final), format="markdown")

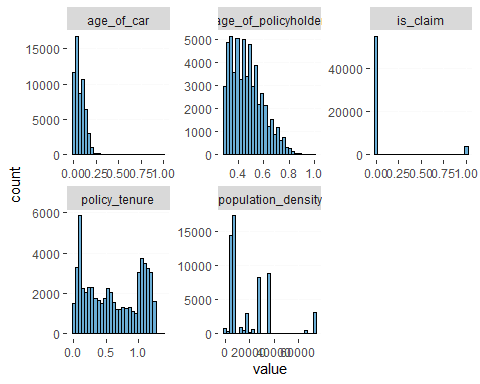
| policy\_tenure | age\_of\_car | age\_of\_policyholder | population\_density | model | is\_claim |
| --- | --- | --- | --- | --- | --- |
| 0.5158736 | 0.05 | 0.6442308 | 4990 | M1 | 0 |
| 0.6726185 | 0.02 | 0.3750000 | 27003 | M1 | 0 |
| 0.8411103 | 0.02 | 0.3846154 | 4076 | M1 | 0 |
| 0.9002766 | 0.11 | 0.4326923 | 21622 | M2 | 0 |
| 0.5964028 | 0.11 | 0.6346154 | 34738 | M3 | 0 |
| 1.0187085 | 0.07 | 0.5192308 | 13051 | M4 | 0 |

Based on our previous analysis, our finals explanatory variables would be: policy\_tenure, age\_of\_car, age\_of\_policyholder, model and population\_density.

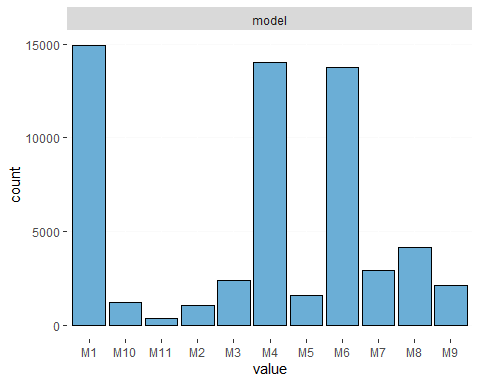
# EDA

#Histograms  
cars\_final %>% select(1:4,6) %>% gather() %>%   
 ggplot(aes(value)) +   
 facet\_wrap(~ key, scales = "free") +   
 geom\_histogram(color = "black", fill = "#6baed6") +   
 theme(plot.title=element\_text(hjust=0.5),  
 panel.background = element\_rect(fill = "white"),  
 panel.grid.major.y = element\_line(color = "grey98"))

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



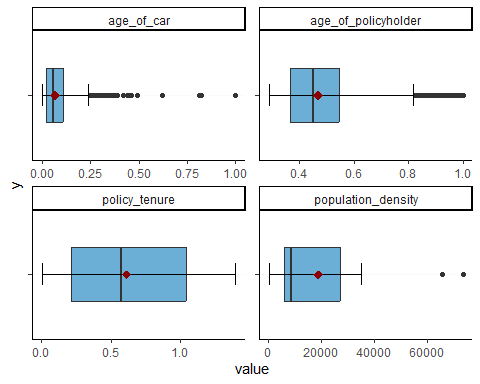
# Barcharts  
cars\_final %>% select(5) %>% gather() %>%  
 ggplot(aes(x = value)) +  
 facet\_wrap(~ key, scales = "free") +  
 geom\_bar(color = "black", fill = "#6baed6") +  
 theme(plot.title=element\_text(hjust=0.5),  
 panel.background = element\_rect(fill = "white"),  
 panel.grid.major.y = element\_line(color = "grey98"))



# boxplot  
cars\_final %>% select(1:4) %>% gather() %>%  
 ggplot(aes(x = value,y="")) +  
   
 facet\_wrap(~ key, scales = "free")+  
 # add horizontal line to "whiskers" of boxplot  
 geom\_boxplot(fill = "#6baed6", width = 0.5) +   
 stat\_boxplot(geom = "errorbar", width = 0.2) +# plot boxplot  
 stat\_summary(fun.y=mean, colour="darkred", geom="point", shape=18, size=3,show\_guide = FALSE)+  
 theme\_classic() +  
 theme(plot.title=element\_text(hjust=0.5),  
 panel.background = element\_rect(fill = "white"),  
 panel.grid.major.y = element\_line(color = "grey98"))

## Warning: The `fun.y` argument of `stat\_summary()` is deprecated as of ggplot2 3.3.0.  
## ℹ Please use the `fun` argument instead.

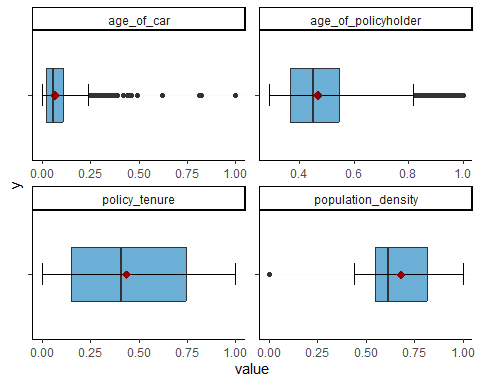
## Warning: The `show\_guide` argument of `layer()` is deprecated as of ggplot2 2.0.0.  
## ℹ Please use the `show.legend` argument instead.



We can see that the distribution of the data is not symmetrical. However, since the data is already normalized, we are not able to do a log transform on the data except for population density.

# Adjust scale with log for population density  
cars\_final$population\_density <- log(cars\_final$population\_density)  
  
# Normalizing population\_density  
normalized.pop <- (cars\_final$population\_density - min (cars\_final$population\_density)) / (max(cars\_final$population\_density)-min(cars\_final$population\_density))  
  
cars\_final$population\_density <- as.vector(normalized.pop)  
  
  
# Normalizing policy tenure  
normalized.policy <- (cars\_final$policy\_tenure - min (cars\_final$policy\_tenure)) / (max(cars\_final$policy\_tenure)-min(cars\_final$policy\_tenure))  
  
cars\_final$policy\_tenure <- as.vector(normalized.policy)

# boxplot  
cars\_final %>% select(1:4) %>% gather() %>%  
 ggplot(aes(x = value,y="")) +  
   
 facet\_wrap(~ key, scales = "free")+  
 # add horizontal line to "whiskers" of boxplot  
 geom\_boxplot(fill = "#6baed6", width = 0.5) +   
 stat\_boxplot(geom = "errorbar", width = 0.2) +# plot boxplot  
 stat\_summary(fun.y=mean, colour="darkred", geom="point", shape=18, size=3,show\_guide = FALSE)+  
 theme\_classic() +  
 theme(plot.title=element\_text(hjust=0.5),  
 panel.background = element\_rect(fill = "white"),  
 panel.grid.major.y = element\_line(color = "grey98"))



With the log transformation on population\_density, the distribution has become a little bit more centered and symmetrical.

kable(summary(cars\_final), format="markdown")

|  | policy\_tenure | age\_of\_car | age\_of\_policyholder | population\_density | model | is\_claim |
| --- | --- | --- | --- | --- | --- | --- |
|  | Min. :0.0000 | Min. :0.00000 | Min. :0.2885 | Min. :0.0000 | Length:58592 | Min. :0.00000 |
|  | 1st Qu.:0.1489 | 1st Qu.:0.02000 | 1st Qu.:0.3654 | 1st Qu.:0.5508 | Class :character | 1st Qu.:0.00000 |
|  | Median :0.4097 | Median :0.06000 | Median :0.4519 | Median :0.6165 | Mode :character | Median :0.00000 |
|  | Mean :0.4366 | Mean :0.06942 | Mean :0.4694 | Mean :0.6801 | NA | Mean :0.06397 |
|  | 3rd Qu.:0.7435 | 3rd Qu.:0.11000 | 3rd Qu.:0.5481 | 3rd Qu.:0.8192 | NA | 3rd Qu.:0.00000 |
|  | Max. :1.0000 | Max. :1.00000 | Max. :1.0000 | Max. :1.0000 | NA | Max. :1.00000 |

# Commented for time computing  
# ggpairs(cars\_final[,c(1:4,6)])+  
# theme\_bw()

We can see from the plots above that our remaining variables are not highly correlated between them.

#boxplot for the distribution of 0 1   
df = cars\_final[, -5]   
df$is\_claim <- as.factor(df$is\_claim)  
library(reshape2)

##   
## Attaching package: 'reshape2'

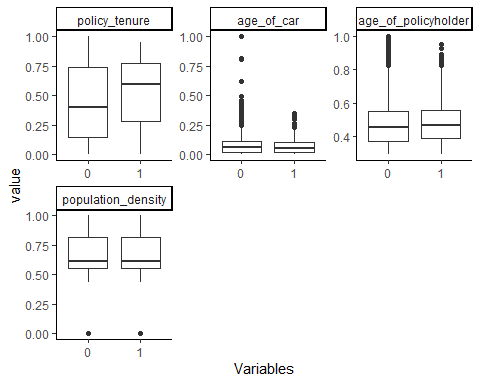
## The following objects are masked from 'package:data.table':  
##   
## dcast, melt

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

df.melt = melt(df)

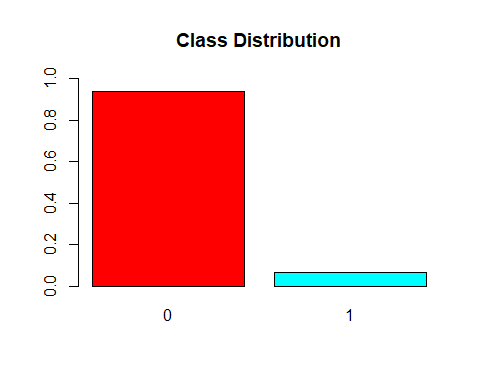
## Using is\_claim as id variables

# HEAD  
  
df.boxplot = ggplot(df.melt, aes(y=value,is\_claim)) +   
 geom\_boxplot() +   
 facet\_wrap(~variable, scales = "free", ncol=3) + theme\_classic() + xlab("Variables")  
  
df.boxplot



We can can the distribution is normal for is\_claim = 0 and is\_claim = 1 with outliers

barplot(prop.table(table(cars$is\_claim)),  
 col = rainbow(2),  
 ylim = c(0, 1),  
 main = "Class Distribution") # outcome variable distribution



prop.table(table(cars$is\_claim))

##   
## 0 1   
## 0.93603222 0.06396778

When the proportions of the different classes in a classification problem are imbalanced (in our case 1 present about 6% from is\_claim), it means that one class (the majority class) is much more frequent than the other class(es) (the minority class(es)). This can cause the model to have a high performance (e.g. high F1-score) on the majority class and low performance on the minority class.

To address this issue, we can re-sample the data to create a more balanced distribution of the classes. This can be done using techniques such as undersampling (removing observations from the majority class) or oversampling . These techniques can help the model to learn better from the minority class and improve its performance on it.

set.seed(1)  
index <- sample(nrow(cars\_final),nrow(cars\_final)\*0.60)  
cars\_train = cars\_final[index,]  
cars\_validation = cars\_final[-index,]  
proportions(table(cars\_train$is\_claim))

##   
## 0 1   
## 0.93628218 0.06371782

proportions(table(cars\_validation$is\_claim))

##   
## 0 1   
## 0.93565729 0.06434271

# Downsample   
cars\_train[,6] <- as.factor(cars\_train$is\_claim)  
train\_downsampling <- downSample(cars\_train[,-6],cars\_train$is\_claim,yname = "is\_claim")  
  
# Normalizing function  
normalizing <- function(x) {  
 (x - min(x)) / (max(x) - min(x))  
}  
  
# Normalizing variables again on downsample's basis:  
  
# Normalizing train data - downsampled  
train\_downsampling[,c(1:4)] <- lapply(train\_downsampling[,c(1:4)], normalizing)  
  
# Normalizing validation data  
cars\_validation[,c(1:4)] <- lapply(cars\_validation[,c(1:4)], normalizing)  
  
  
summary(cars\_validation)

## policy\_tenure age\_of\_car age\_of\_policyholder population\_density  
## Min. :0.0000 Min. :0.00000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.1507 1st Qu.:0.02000 1st Qu.:0.1096 1st Qu.:0.5508   
## Median :0.4266 Median :0.06000 Median :0.2329 Median :0.6165   
## Mean :0.4540 Mean :0.06931 Mean :0.2574 Mean :0.6799   
## 3rd Qu.:0.7746 3rd Qu.:0.11000 3rd Qu.:0.3699 3rd Qu.:0.8192   
## Max. :1.0000 Max. :1.00000 Max. :1.0000 Max. :1.0000   
## model is\_claim   
## Length:23437 Min. :0.00000   
## Class :character 1st Qu.:0.00000   
## Mode :character Median :0.00000   
## Mean :0.06434   
## 3rd Qu.:0.00000   
## Max. :1.00000

# Predictive analytics

## Classification tree

### Method’s description

A classification tree is a method of supervised algorithm used for classification. It consists in continuously splitting the data into sub-parts based on data features (sub-parts could be segmented in rectangular areas on the graph) until getting only one class in the splitted area. At that point, data’s class is the most homogeneous as said previously. The splits are done in such a way that it minimizes the impurity of the new area. The algorithm may chose one of the following measures of impurity: Gini Index or Entropy measure. These measures range between 0 and 1 and allow to identify to which class the data belongs.

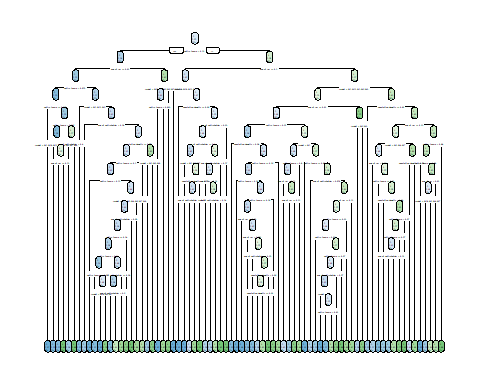
*Intuition*: A higher number of terminal nodes is expected to decrease the overall error until reaching the point of overfitting.

Therefore, it is wise to stop a tree’s growth. In that sense, we need to prune the tree and use cross-validation to get a best tree size.

### Method application

tree\_fit <- rpart(train\_downsampling$is\_claim ~.,  
 data = train\_downsampling, # Data set  
 cp = 0.001,  
 method = "class") # Method: Classification  
  
  
rpart.plot(tree\_fit) # Plot the resulting tree

## Warning: labs do not fit even at cex 0.15, there may be some overplotting



# Predict values for validation set  
pred\_tree\_v\_tr <- predict(tree\_fit,  
 cars\_validation[,c(-6)],  
 type ="class")

We start by fitting a full grown tree. It allows us to get a first look.

# Find the best cp with minimal xerror || /!\ code may need to be improved  
cp\_minimal <- tree\_fit$cptable[which.min(  
 tree\_fit$cptable[,"CP"]), # Take lowest cp and Take lowest xerror  
 1]# CP column   
  
pruned.tree <- prune(tree\_fit, cp= cp\_minimal) # Prune the train model with lowest cp

We want to identify the lowest cp value to prune the tree. That value equals 0.001 and is similar to the default value that we chosed to compute the full grown classification tree.

Also, it is known that we should not choose only a tree based on it’s cp, but we should also consider the tree size. Ordinarily, we would like to have a small tree with a small cp. Such a tree would avoid overfitting. To choose a tree size we should find the smallest minimum error within one standard error and a small CP value.

# Cross-validation  
  
table\_cp <- pruned.tree$cptable # Save cps  
  
# DOES SOMEONE HAVE A BETTER METHOD ?  
rows\_small\_errors\_std <- head(order(table\_cp[,5]),6) # Know the rows (folds) having the smallest standard errors of the estimates  
  
table\_error <- matrix(table\_cp[,4] + table\_cp[,5]) # Sum the std error and the error  
  
table\_error <- cbind(table\_cp[,1], # Include CP  
 table\_cp[,2], # Include nsplit  
 table\_error, # Error + std error  
 table\_cp[,5]) # Std error  
  
rows\_small\_errors\_std <- head(sort(table\_error[,4]),6) # Know the rows (folds) having the smallest std error  
rows\_small\_errors <- head(sort(table\_error[,3]),6) # Display the smallest standard errors  
  
rows\_small\_errors\_std # Display and chose the best size

## 16 17 10 11 12 18   
## 0.01461348 0.01461348 0.01461631 0.01461913 0.01462053 0.01462612

rows\_small\_errors # Display

## 16 17 10 11 12 18   
## 0.8065778 0.8065778 0.8074735 0.8083691 0.8088170 0.8106083

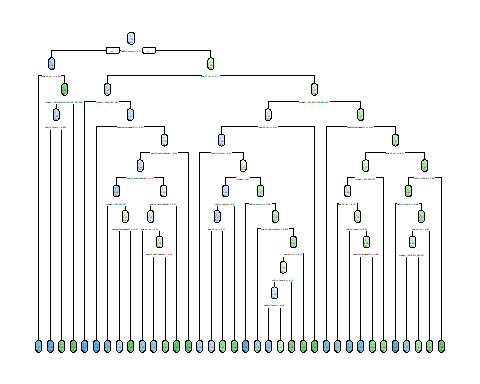
kable(table\_cp)

| CP | nsplit | rel error | xerror | xstd |
| --- | --- | --- | --- | --- |
| 0.1464286 | 0 | 1.0000000 | 1.0392857 | 0.0149288 |
| 0.0441964 | 1 | 0.8535714 | 0.8625000 | 0.0147985 |
| 0.0133929 | 2 | 0.8093750 | 0.8147321 | 0.0146817 |
| 0.0074405 | 3 | 0.7959821 | 0.8183036 | 0.0146917 |
| 0.0049107 | 8 | 0.7535714 | 0.8071429 | 0.0146599 |
| 0.0044643 | 9 | 0.7486607 | 0.8053571 | 0.0146546 |
| 0.0040179 | 10 | 0.7441964 | 0.8017857 | 0.0146439 |
| 0.0035714 | 11 | 0.7401786 | 0.8058036 | 0.0146559 |
| 0.0031250 | 12 | 0.7366071 | 0.8004464 | 0.0146399 |
| 0.0028274 | 14 | 0.7303571 | 0.7928571 | 0.0146163 |
| 0.0026786 | 17 | 0.7218750 | 0.7937500 | 0.0146191 |
| 0.0018973 | 18 | 0.7191964 | 0.7941964 | 0.0146205 |
| 0.0017857 | 25 | 0.7040179 | 0.8049107 | 0.0146533 |
| 0.0015625 | 28 | 0.6986607 | 0.8008929 | 0.0146412 |
| 0.0014881 | 32 | 0.6915179 | 0.7991071 | 0.0146358 |
| 0.0014509 | 35 | 0.6870536 | 0.7919643 | 0.0146135 |
| 0.0013393 | 50 | 0.6584821 | 0.7919643 | 0.0146135 |
| 0.0012500 | 56 | 0.6504464 | 0.7959821 | 0.0146261 |
| 0.0011161 | 61 | 0.6441964 | 0.8107143 | 0.0146703 |
| 0.0010000 | 75 | 0.6263393 | 0.8107143 | 0.0146703 |

best\_cp <- table\_cp[as.integer(names(rows\_small\_errors)[1]),1]

We see that the 16th observation where nsplit = 35 and CP = 0.0014509 gives the best error choice.

pruned.tree <- prune(tree\_fit, cp = best\_cp) # Prune the train model with best cp  
  
rpart.plot(pruned.tree) # Plot the tree with best size



sum(pruned.tree$frame$ncompete == 0) # Number of leaves

## [1] 36

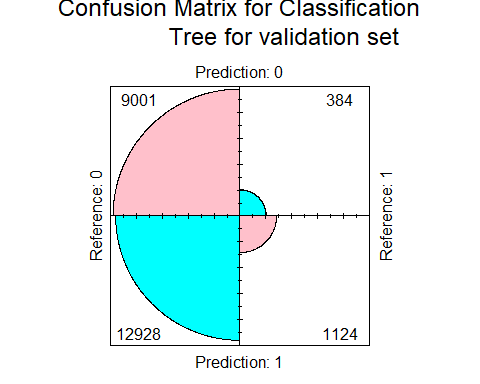
The resulting pruned tree at its best size has 36 leaves.

### Performance evaluation

# Confusion matrix for classification tree with validation set  
conf\_matrix\_tree\_v <- confusionMatrix(  
 pred\_tree\_v\_tr,  
 factor(cars\_validation$is\_claim),  
 positive = "1")  
conf\_matrix\_tree\_v

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 9001 384  
## 1 12928 1124  
##   
## Accuracy : 0.432   
## 95% CI : (0.4257, 0.4384)  
## No Information Rate : 0.9357   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.032   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.74536   
## Specificity : 0.41046   
## Pos Pred Value : 0.07999   
## Neg Pred Value : 0.95908   
## Prevalence : 0.06434   
## Detection Rate : 0.04796   
## Detection Prevalence : 0.59956   
## Balanced Accuracy : 0.57791   
##   
## 'Positive' Class : 1   
##

# plot confusion matrix  
fourfoldplot(conf\_matrix\_tree\_v$table,  
 color = c("cyan", "pink"),  
 conf.level = 0,  
 margin = 1,  
 main = "Confusion Matrix for Classification  
 Tree for validation set")



Looking at the performance metrics for the full grown tree:

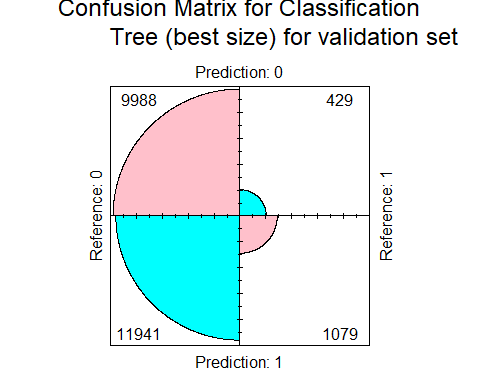
* Accuracy highs at 43.2%
* Specificity highs at 41%
* Sensitivity is 74.5%

We see that the tree is better at classifying negative than positives outcome and struggles with false positive values that represents the majority of classifieds values.

# Prediction on Validation data  
pred\_tree\_v\_pruned <- predict(pruned.tree, # Pruned tree  
 cars\_validation[,c(-16)],# Validation set  
 type ="class")  
  
# Confusion matrix for classification tree with validation set  
conf\_matrix\_tree\_v <- confusionMatrix(  
 pred\_tree\_v\_pruned,  
 factor(cars\_validation$is\_claim),  
 positive = "1")  
conf\_matrix\_tree\_v

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 9988 429  
## 1 11941 1079  
##   
## Accuracy : 0.4722   
## 95% CI : (0.4658, 0.4786)  
## No Information Rate : 0.9357   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0375   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.71552   
## Specificity : 0.45547   
## Pos Pred Value : 0.08287   
## Neg Pred Value : 0.95882   
## Prevalence : 0.06434   
## Detection Rate : 0.04604   
## Detection Prevalence : 0.55553   
## Balanced Accuracy : 0.58549   
##   
## 'Positive' Class : 1   
##

# plot confusion matrix  
fourfoldplot(conf\_matrix\_tree\_v$table,  
 color = c("cyan", "pink"),  
 conf.level = 0,  
 margin = 1,  
 main = "Confusion Matrix for Classification  
 Tree (best size) for validation set")



Looking at the performance metrics for the best size tree:

* Accuracy highs at 47.2 %
* Specificity highs at 45.6 %
* Sensitivity is 71.6%

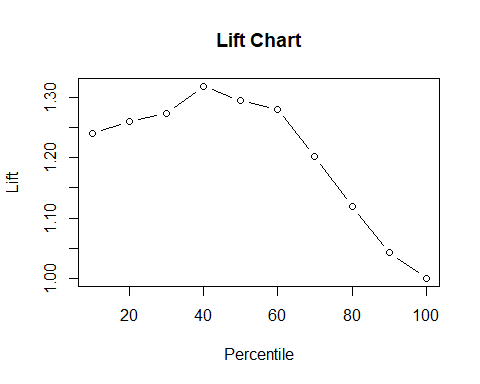
From all positive outcome predicted, almost 8.3% where right while almost 95.9% of negative outcomes where correctly predicted. Also, the performances for accuracy improved of almost 4% and the one for specificity improved of almost 5%, but the performance was lower for sensitivity of almost 3%.

### Conclusion on method’s performance

We conclude that predicting positive outcomes is difficult for the classification tree model since not even 10% of its positive predictions are right, even if cross-validation has been used. The method is better at predicting negative outcomes than positive ones. Therefore, we may want to analyse an other model with hopping to find better results with respect to positive prediction.

#### Lift Chart

#Getting the probabilities from the model  
tree.outcome <- as.data.frame(predict(pruned.tree, # Pruned tree  
 cars\_validation[,c(-16)],# Validation set  
 type ="prob"))[,2]  
  
  
#Lift chart plot   
liftChart(tree.outcome, factor(cars\_validation$is\_claim))



topDecileLift(tree.outcome, factor(cars\_validation$is\_claim))

## [1] 1.240424

## KNN

### Method’s description

The k-nearest-neighbors algorithm that can be used for classification. To classify or predict a new record, the method relies on finding “similar” records in the training data. These “neighbors” are then used to derive a classification for the new record by voting (for classification) .

### Data transformation

# Add dummy variables   
cars\_final$model <- as.factor(cars\_final$model) # change variable format to factor   
  
dummies <- dummyVars(~ ., data = cars\_final) # create object for dummy variables  
cars\_final\_dummy <- as.data.frame(predict(dummies, newdata = cars\_final)) # apply dummies to data  
  
cars\_final$is\_claim <- as.factor(cars\_final$is\_claim) # change variable format to factor  
cars\_final\_dummy$is\_claim <- as.factor(cars\_final\_dummy$is\_claim)   
  
head(cars\_final\_dummy) # resulting in 11

## policy\_tenure age\_of\_car age\_of\_policyholder population\_density model.M1  
## 1 0.3681298 0.05 0.6442308 0.5141315 1  
## 2 0.4805800 0.02 0.3750000 0.8192361 1  
## 3 0.6014574 0.02 0.3846154 0.4775735 1  
## 4 0.6439038 0.11 0.4326923 0.7790792 0  
## 5 0.4259022 0.11 0.6346154 0.8647506 0  
## 6 0.7288679 0.07 0.5192308 0.6878563 0  
## model.M10 model.M11 model.M2 model.M3 model.M4 model.M5 model.M6 model.M7  
## 1 0 0 0 0 0 0 0 0  
## 2 0 0 0 0 0 0 0 0  
## 3 0 0 0 0 0 0 0 0  
## 4 0 0 1 0 0 0 0 0  
## 5 0 0 0 1 0 0 0 0  
## 6 0 0 0 0 1 0 0 0  
## model.M8 model.M9 is\_claim  
## 1 0 0 0  
## 2 0 0 0  
## 3 0 0 0  
## 4 0 0 0  
## 5 0 0 0  
## 6 0 0 0

str(cars\_final\_dummy)

## 'data.frame': 58592 obs. of 16 variables:  
## $ policy\_tenure : num 0.368 0.481 0.601 0.644 0.426 ...  
## $ age\_of\_car : num 0.05 0.02 0.02 0.11 0.11 0.07 0.16 0.14 0.07 0.04 ...  
## $ age\_of\_policyholder: num 0.644 0.375 0.385 0.433 0.635 ...  
## $ population\_density : num 0.514 0.819 0.478 0.779 0.865 ...  
## $ model.M1 : num 1 1 1 0 0 0 0 0 0 0 ...  
## $ model.M10 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ model.M11 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ model.M2 : num 0 0 0 1 0 0 0 0 0 0 ...  
## $ model.M3 : num 0 0 0 0 1 0 0 0 0 0 ...  
## $ model.M4 : num 0 0 0 0 0 1 0 0 1 0 ...  
## $ model.M5 : num 0 0 0 0 0 0 1 0 0 0 ...  
## $ model.M6 : num 0 0 0 0 0 0 0 1 0 0 ...  
## $ model.M7 : num 0 0 0 0 0 0 0 0 0 1 ...  
## $ model.M8 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ model.M9 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ is\_claim : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

train\_dummy <- cars\_final\_dummy[index,] # training set with dummies   
valid\_dummy <- cars\_final\_dummy[-index,] #validation set with dummies  
train\_dummy\_downsampled <- downSample(train\_dummy[,-16],train\_dummy$is\_claim,yname = "is\_claim") # undersampling for having same probation   
  
  
# Normalizing data  
train\_dummy\_downsampled[,c(1:4)] <- lapply(train\_dummy\_downsampled[,c(1:4)], normalizing)  
  
valid\_dummy[,c(1:4)] <- lapply(valid\_dummy[,c(1:4)], normalizing)

### Method application

set.seed(1)  
# Find optimal k  
  
# Data frame for k from 1 to 50 and respective accuracy  
accuracy <- data.frame(k = seq(1, 50, 1), overallaccuracy = rep(0, 50))   
  
# Use for loop to find k with highest accuracy  
for (i in 1:50) {  
 knn.pred<-knn(train\_dummy\_downsampled[,-16],valid\_dummy[,-16],  
 cl=train\_dummy\_downsampled[,16],k=i)  
 accuracy[i,2]<-confusionMatrix(knn.pred,valid\_dummy[,16])$overall[1]  
}   
# Data frame for k from 1 to 50 and respective specificity  
Specificity <- data.frame(k = seq(1, 50, 1), overallspecificity = rep(0, 50))  
  
# Use for loop to find k with highest specificity  
for (i in 1:50) {  
 knn.pred<-knn(train\_dummy\_downsampled[,-16],valid\_dummy[,-16],  
 cl=train\_dummy\_downsampled[,16],k=i)  
 Specificity[i,2]<-confusionMatrix(knn.pred,valid\_dummy[,16])$byClass[2]  
}   
# Data frame for k from 1 to 50 and respective Sensitivity   
Sensitivity <- data.frame(k = seq(1, 50, 1), overallSensitivity = rep(0, 50))  
  
# Use for loop to find k with highest Sensitivity   
for (i in 1:50) {  
 knn.pred<-knn(train\_dummy\_downsampled[,-16],valid\_dummy[,-16],  
 cl=train\_dummy\_downsampled[,16],k=i)  
 Sensitivity[i,2]<-confusionMatrix(knn.pred,valid\_dummy[,16])$byClass[1]  
}

which(accuracy[,2] == max(accuracy[,2])) # max accuracy

## [1] 2

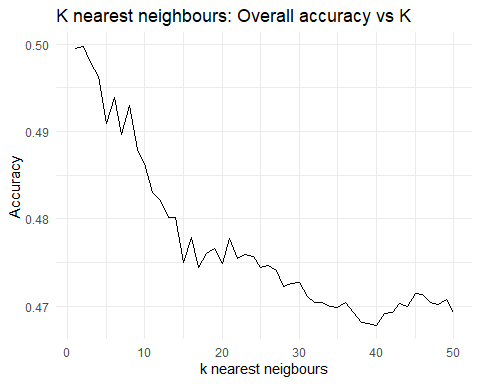
which(Specificity[,2] == max(Specificity[,2])) # max specificity

## [1] 41

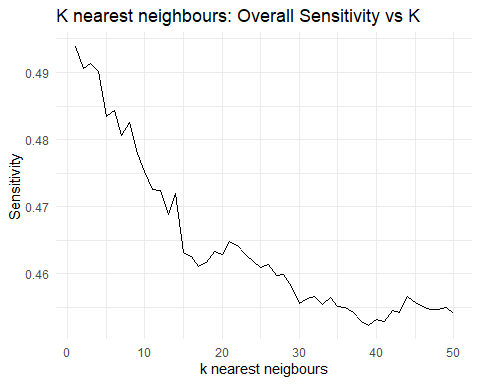
which(Sensitivity[,2] == max(Sensitivity[,2])) # max specificity

## [1] 1

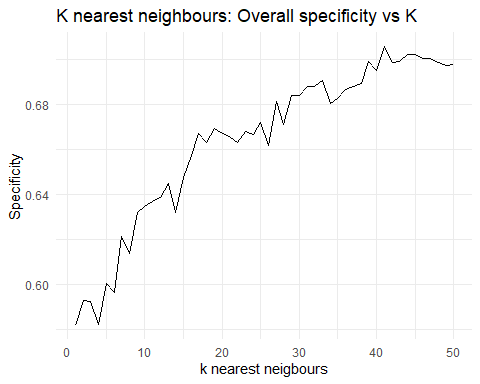
# Plot accuracy for different k  
ggplot(accuracy, aes(k, overallaccuracy)) +   
 geom\_line() +   
 theme\_minimal() +  
 labs (title = "K nearest neighbours: Overall accuracy vs K",   
 y = "Accuracy",   
 x = "k nearest neigbours" )



# Plot Sensitivity for different k  
ggplot(Sensitivity, aes(k, overallSensitivity)) +   
 geom\_line() +   
 theme\_minimal() +  
 labs (title = "K nearest neighbours: Overall Sensitivity vs K",   
 y = "Sensitivity",   
 x = "k nearest neigbours" )



# Plot specificity for different k  
ggplot(Specificity, aes(k, overallspecificity)) +   
 geom\_line() +   
 theme\_minimal() +  
 labs (title = "K nearest neighbours: Overall specificity vs K",   
 y = "Specificity",   
 x = "k nearest neigbours" )

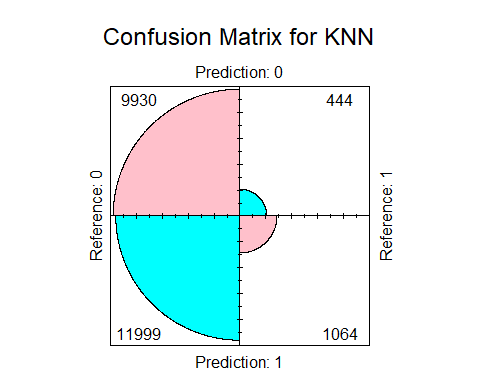


Since we are looking for k which give us max optimum value of Specificity we chose k = 41

knn.pred.valid <- knn(train\_dummy\_downsampled[, -16], valid\_dummy[, -16], cl = train\_dummy\_downsampled[, 16], k = 41 ,prob = T) #prediction on validation data using best k=6  
cmk <- confusionMatrix(knn.pred.valid,valid\_dummy[,16]) #confusion matrix   
cmk$byClass[2]

## Specificity   
## 0.7055703

fourfoldplot(cmk$table, color = c("cyan", "pink"),  
 conf.level = 0, margin = 1, main = "Confusion Matrix for KNN") # plot confusion ma

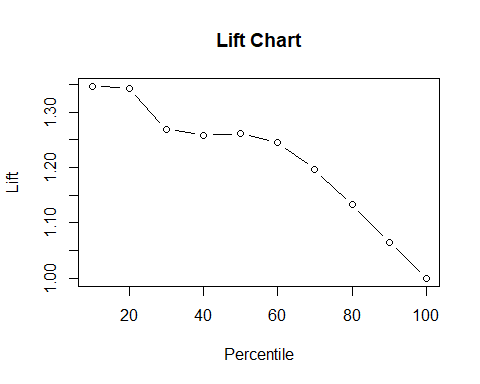


cmk

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 9930 444  
## 1 11999 1064  
##   
## Accuracy : 0.4691   
## 95% CI : (0.4627, 0.4755)  
## No Information Rate : 0.9357   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0347   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.45283   
## Specificity : 0.70557   
## Pos Pred Value : 0.95720   
## Neg Pred Value : 0.08145   
## Prevalence : 0.93566   
## Detection Rate : 0.42369   
## Detection Prevalence : 0.44263   
## Balanced Accuracy : 0.57920   
##   
## 'Positive' Class : 0   
##

#### Lift Chart

#Extracting the probabilities  
knn\_outcome <- as.data.table(cbind(as.data.table(knn.pred.valid), knn\_prob = attr(knn.pred.valid, "prob")))  
  
# Adjusting probabilities  
knn\_outcome[knn.pred.valid != 1, knn\_prob := 1 - knn\_prob]  
  
# plot  
liftChart(knn\_outcome$knn\_prob, factor(valid\_dummy$is\_claim))



topDecileLift(knn\_outcome$knn\_prob, factor(valid\_dummy$is\_claim))

## [1] 1.406256

With a top decile lift of 1.40, the model’s lift drastically decreases at the second decile then slowly declines with each decile.

## Neural Network

### Method description:

Neural network tries to identify complex relationships between variables. It can be represented by edges (weights) interconnecting nodes (values) and organized in different layers. There are three levels of layer:

1. *Input layer:* concerned with the predictors
2. *Hidden layer:* concerned with weighted relations
3. *Output layer:* concerned with the final output (class)

*Intuition*: Transform input values and identify relations (translated by weights) to identify output values.

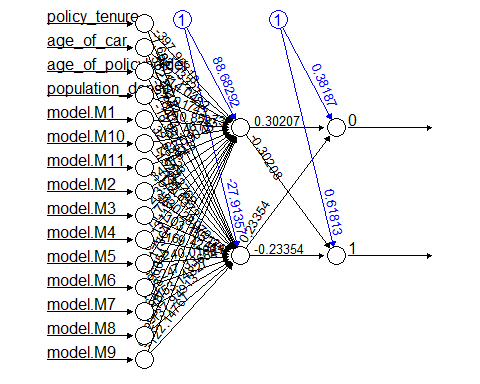
Values from variables go into the input layer. Then we initialize a very small and random weight in relation with our imputed values to start training the model. The resulting value is a node in the hidden layer. Weights are then updated again for many times in order to find an optimum value which minimize the output error (predicted output with respect to the true output). These weights depend on a function called “transfer function”.

### Method application

Common practice is to use 1 hidden layer for fitting neural networks. We are first going to try the algorithm with two nodes. Next, we are going to continue the analysis with a third node in order to compare performances.

#### 1 hidden layer and 2 nodes

# Run Neural Network (N.N.) with 1 hidden layer and 2 nodes  
nn\_1H\_2N <- neuralnet(is\_claim ~ .,  
 data = train\_dummy\_downsampled,  
 hidden = 2) # 1 hidden layer of 2 nodes  
  
plot(nn\_1H\_2N, rep="best")



# Predict the output on validation set  
validation\_prediction\_1H\_2N <- predict(nn\_1H\_2N,  
 valid\_dummy[,-16],  
 type = "class")  
  
validation\_prediction\_1H\_2N\_binary <-  
 ifelse(validation\_prediction\_1H\_2N[,1]  
 >= 0.5, 1, 0) # Transform probabilities as binary outcome

We use an algorithm of neural network with one layer of two nodes. The variables that are used as input need to be numerical and normalized (from 0 to 1). These steps has already been done previously and this is why we input the variable model under its “dummified” form. Then, we train the algorithm and use it to predict data from our validation set.

#### 1 hidden layer and 4 nodes

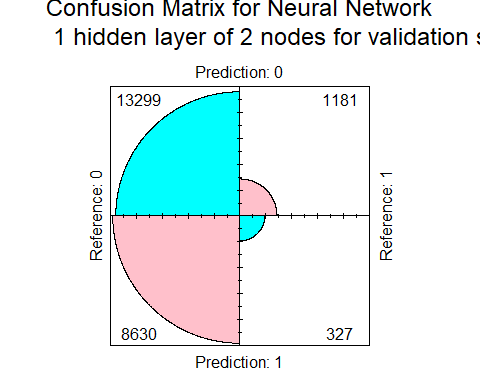
# run Neural Network (N.N.) with 1 hidden layer and 4 nodes  
# nn\_1H\_4N <- neuralnet(is\_claim ~ .,  
# data = train\_dummy\_downsampled,  
# hidden = c(4)) # 1 hidden layer of 4 nodes  
#   
#   
#   
# plot(nn\_1H\_4N, rep="best") # plots the neural net with 4 nodes  
#   
#   
#   
# # Predict the output on validation set  
# validation\_prediction\_1H\_4N <- predict(nn\_1H\_4N,  
# valid\_dummy[,-16],  
# type = "class")  
#   
# validation\_prediction\_1H\_4N\_binary <-  
# ifelse(validation\_prediction\_1H\_4N[,1]  
# >= 0.5, 1, 0) # Transform probabilities as binary outcome

### Performance evaluation

# Confusion matrix for classification tree with validation set  
conf\_matrix\_1H2N\_valid <- confusionMatrix(  
 factor(validation\_prediction\_1H\_2N\_binary),  
 factor(valid\_dummy$is\_claim), # change dataset  
 positive = "1")  
  
conf\_matrix\_1H2N\_valid

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 13299 1181  
## 1 8630 327  
##   
## Accuracy : 0.5814   
## 95% CI : (0.575, 0.5877)  
## No Information Rate : 0.9357   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : -0.0535   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.21684   
## Specificity : 0.60646   
## Pos Pred Value : 0.03651   
## Neg Pred Value : 0.91844   
## Prevalence : 0.06434   
## Detection Rate : 0.01395   
## Detection Prevalence : 0.38217   
## Balanced Accuracy : 0.41165   
##   
## 'Positive' Class : 1   
##

# plot confusion matrix  
fourfoldplot(conf\_matrix\_1H2N\_valid$table,  
 color = c("cyan", "pink"),  
 conf.level = 0,  
 margin = 1,  
 main = "Confusion Matrix for Neural Network  
 1 hidden layer of 2 nodes for validation set")



## Confusion matrix for classification tree with validation set  
#conf\_matrix\_1H4N\_valid <- confusionMatrix(  
#factor(validation\_prediction\_1H\_4N\_binary),  
#factor(valid\_dummy$is\_claim), # change dataset  
#positive = "1")  
#  
#conf\_matrix\_1H4N\_valid  
#  
## plot confusion matrix  
#fourfoldplot(conf\_matrix\_1H4N\_valid$table,  
# color = c("cyan", "pink"),  
# conf.level = 0,  
# margin = 1,  
# main = "Confusion Matrix for Neural Network  
# 1 hidden layer of 4 nodes for validation set")

Looking at the performance metrics for the neural network with 2 nodes:

* Accuracy is 59.4 %
* Specificity is 62.0 %
* Sensitivity is 20.6%

Looking at the performance metrics for the neural network with 4 nodes:

* Accuracy highs at 56.7 %
* Specificity highs at 59.4 %
* Sensitivity is 19.0%

Comparing 4 nodes to 2 nodes we see that almost 3.1% (versus 3.5% for 2 nodes) positive outcomes predicted where correctly predicted while almost 91.4% (versus 91.9% for 2 nodes) of negative outcomes where correctly predicted for the higher nodes neural network. Also, the performances for accuracy decreased of almost 3% and the one for specificity decreased of almost 3% and for sensitivity it decreased of almost 1%.

### Conclusion on method performance

We conclude that predicting positive outcomes is still difficult since not even 5% of positive predictions are correctly predicted. The Neural Network method with one hidden layer and two nodes provides better results than four nodes. Nonetheless, the performances for our business case are very poor.

## Logistic Regression

### Method Description

The logistic regression is a model that is also suited for binary response variables. It is a modeling technique used in statistics to predict the probability of an event occurring based on different independent variables. Building the model with a logit link function as it is usually the best suited one for the logistic regression.

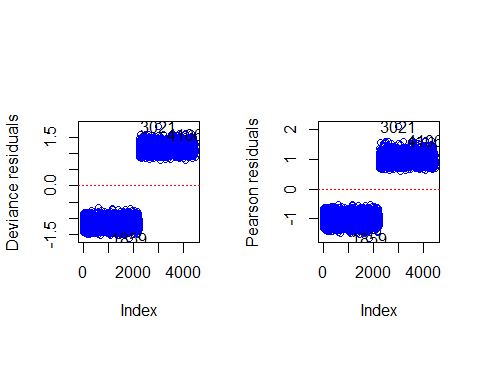
### Method Application

cars.lg <- glm(is\_claim ~., data= train\_downsampling, family=binomial(link="logit"))  
  
summary(cars.lg) # Displaying the resulting model

##   
## Call:  
## glm(formula = is\_claim ~ ., family = binomial(link = "logit"),   
## data = train\_downsampling)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.6217 -1.1485 0.0561 1.1390 1.8354   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.26571 0.15201 -1.748 0.0805 .   
## policy\_tenure 0.99056 0.09941 9.965 < 2e-16 \*\*\*  
## age\_of\_car -1.85521 0.31159 -5.954 2.62e-09 \*\*\*  
## age\_of\_policyholder 0.12697 0.14709 0.863 0.3880   
## population\_density -0.19614 0.18370 -1.068 0.2856   
## modelM10 -0.06149 0.23780 -0.259 0.7959   
## modelM11 -0.38564 0.41248 -0.935 0.3498   
## modelM2 0.50452 0.23030 2.191 0.0285 \*   
## modelM3 -0.13328 0.17131 -0.778 0.4366   
## modelM4 0.14300 0.09739 1.468 0.1420   
## modelM5 0.11491 0.19127 0.601 0.5480   
## modelM6 0.17503 0.09725 1.800 0.0719 .   
## modelM7 0.12728 0.15240 0.835 0.4036   
## modelM8 -0.06633 0.13543 -0.490 0.6243   
## modelM9 0.09928 0.17630 0.563 0.5734   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 6210.6 on 4479 degrees of freedom  
## Residual deviance: 6057.7 on 4465 degrees of freedom  
## AIC: 6087.7  
##   
## Number of Fisher Scoring iterations: 4

From the dummies born from “model” variable, only model =M2 is statistically significant. H0 cannot be rejected for the beta estimates of population\_density and age\_of\_policeholder either. The other explanatory variables are significant, but the intercept is not.

PlotResidLogist(cars.lg) # Deviance residuals



If we plot the deviance residuals, we notice that the positive residuals and negative ones each form a clear group. The residuals appearing only on one half of the observations is due to downsampling. The way every observation is tightly grouped may result from the artificially generated data. It is also the case for the pearson residuals. We can see an extreme residual that stands out (observation 440). Considering the amount of observations, this single observation should not affect the model too much. A robust logisitc regression is not required.

anova(cars.lg, test="Chisq")

## Analysis of Deviance Table  
##   
## Model: binomial, link: logit  
##   
## Response: is\_claim  
##   
## Terms added sequentially (first to last)  
##   
##   
## Df Deviance Resid. Df Resid. Dev Pr(>Chi)   
## NULL 4479 6210.6   
## policy\_tenure 1 103.251 4478 6107.3 < 2.2e-16 \*\*\*  
## age\_of\_car 1 34.170 4477 6073.2 5.049e-09 \*\*\*  
## age\_of\_policyholder 1 1.042 4476 6072.1 0.3074   
## population\_density 1 1.341 4475 6070.8 0.2469   
## model 10 13.142 4465 6057.7 0.2158   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

According to the analysis of the deviance table, “population\_density”,“age\_of\_policyholder”, and “model” could be unworth to be part of the model with p-value>0.05. To further investigate the importance of the variables, we can run a stepwise selection.

cars\_glm\_back <- step(cars.lg, direction = "backward") # Model with backward selection

## Start: AIC=6087.65  
## is\_claim ~ policy\_tenure + age\_of\_car + age\_of\_policyholder +   
## population\_density + model  
##   
## Df Deviance AIC  
## - model 10 6070.8 6080.8  
## - age\_of\_policyholder 1 6058.4 6086.4  
## - population\_density 1 6058.8 6086.8  
## <none> 6057.7 6087.7  
## - age\_of\_car 1 6093.8 6121.8  
## - policy\_tenure 1 6158.9 6186.9  
##   
## Step: AIC=6080.79  
## is\_claim ~ policy\_tenure + age\_of\_car + age\_of\_policyholder +   
## population\_density  
##   
## Df Deviance AIC  
## - age\_of\_policyholder 1 6071.9 6079.9  
## - population\_density 1 6072.1 6080.1  
## <none> 6070.8 6080.8  
## - age\_of\_car 1 6105.1 6113.1  
## - policy\_tenure 1 6185.6 6193.6  
##   
## Step: AIC=6079.87  
## is\_claim ~ policy\_tenure + age\_of\_car + population\_density  
##   
## Df Deviance AIC  
## - population\_density 1 6073.2 6079.2  
## <none> 6071.9 6079.9  
## - age\_of\_car 1 6106.4 6112.4  
## - policy\_tenure 1 6191.9 6197.9  
##   
## Step: AIC=6079.18  
## is\_claim ~ policy\_tenure + age\_of\_car  
##   
## Df Deviance AIC  
## <none> 6073.2 6079.2  
## - age\_of\_car 1 6107.3 6111.3  
## - policy\_tenure 1 6194.7 6198.7

summary(cars\_glm\_back)

##   
## Call:  
## glm(formula = is\_claim ~ policy\_tenure + age\_of\_car, family = binomial(link = "logit"),   
## data = train\_downsampling)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.4797 -1.1518 0.1053 1.1345 1.7990   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.33365 0.06430 -5.189 2.11e-07 \*\*\*  
## policy\_tenure 1.04187 0.09563 10.895 < 2e-16 \*\*\*  
## age\_of\_car -1.59893 0.27582 -5.797 6.75e-09 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 6210.6 on 4479 degrees of freedom  
## Residual deviance: 6073.2 on 4477 degrees of freedom  
## AIC: 6079.2  
##   
## Number of Fisher Scoring iterations: 4

The backward selection results in a model with 3 explanatory variables: policy\_tenure and age\_of\_car. Every beta is very stastically significant.

cars\_glm\_forward <- step(cars.lg, direction = "forward") # Model with forward selection

## Start: AIC=6087.65  
## is\_claim ~ policy\_tenure + age\_of\_car + age\_of\_policyholder +   
## population\_density + model

summary(cars\_glm\_forward)

##   
## Call:  
## glm(formula = is\_claim ~ policy\_tenure + age\_of\_car + age\_of\_policyholder +   
## population\_density + model, family = binomial(link = "logit"),   
## data = train\_downsampling)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.6217 -1.1485 0.0561 1.1390 1.8354   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.26571 0.15201 -1.748 0.0805 .   
## policy\_tenure 0.99056 0.09941 9.965 < 2e-16 \*\*\*  
## age\_of\_car -1.85521 0.31159 -5.954 2.62e-09 \*\*\*  
## age\_of\_policyholder 0.12697 0.14709 0.863 0.3880   
## population\_density -0.19614 0.18370 -1.068 0.2856   
## modelM10 -0.06149 0.23780 -0.259 0.7959   
## modelM11 -0.38564 0.41248 -0.935 0.3498   
## modelM2 0.50452 0.23030 2.191 0.0285 \*   
## modelM3 -0.13328 0.17131 -0.778 0.4366   
## modelM4 0.14300 0.09739 1.468 0.1420   
## modelM5 0.11491 0.19127 0.601 0.5480   
## modelM6 0.17503 0.09725 1.800 0.0719 .   
## modelM7 0.12728 0.15240 0.835 0.4036   
## modelM8 -0.06633 0.13543 -0.490 0.6243   
## modelM9 0.09928 0.17630 0.563 0.5734   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 6210.6 on 4479 degrees of freedom  
## Residual deviance: 6057.7 on 4465 degrees of freedom  
## AIC: 6087.7  
##   
## Number of Fisher Scoring iterations: 4

The forward selection ends up with different variables: the model is the same as the full one. This selection results in a minority of statistically significant beta estimates.

cars\_glm\_both <- step(cars.lg, direction = "both") # Forward and backward

## Start: AIC=6087.65  
## is\_claim ~ policy\_tenure + age\_of\_car + age\_of\_policyholder +   
## population\_density + model  
##   
## Df Deviance AIC  
## - model 10 6070.8 6080.8  
## - age\_of\_policyholder 1 6058.4 6086.4  
## - population\_density 1 6058.8 6086.8  
## <none> 6057.7 6087.7  
## - age\_of\_car 1 6093.8 6121.8  
## - policy\_tenure 1 6158.9 6186.9  
##   
## Step: AIC=6080.79  
## is\_claim ~ policy\_tenure + age\_of\_car + age\_of\_policyholder +   
## population\_density  
##   
## Df Deviance AIC  
## - age\_of\_policyholder 1 6071.9 6079.9  
## - population\_density 1 6072.1 6080.1  
## <none> 6070.8 6080.8  
## + model 10 6057.7 6087.7  
## - age\_of\_car 1 6105.1 6113.1  
## - policy\_tenure 1 6185.6 6193.6  
##   
## Step: AIC=6079.87  
## is\_claim ~ policy\_tenure + age\_of\_car + population\_density  
##   
## Df Deviance AIC  
## - population\_density 1 6073.2 6079.2  
## <none> 6071.9 6079.9  
## + age\_of\_policyholder 1 6070.8 6080.8  
## + model 10 6058.4 6086.4  
## - age\_of\_car 1 6106.4 6112.4  
## - policy\_tenure 1 6191.9 6197.9  
##   
## Step: AIC=6079.18  
## is\_claim ~ policy\_tenure + age\_of\_car  
##   
## Df Deviance AIC  
## <none> 6073.2 6079.2  
## + population\_density 1 6071.9 6079.9  
## + age\_of\_policyholder 1 6072.1 6080.1  
## + model 10 6059.5 6085.5  
## - age\_of\_car 1 6107.3 6111.3  
## - policy\_tenure 1 6194.7 6198.7

summary(cars\_glm\_both)

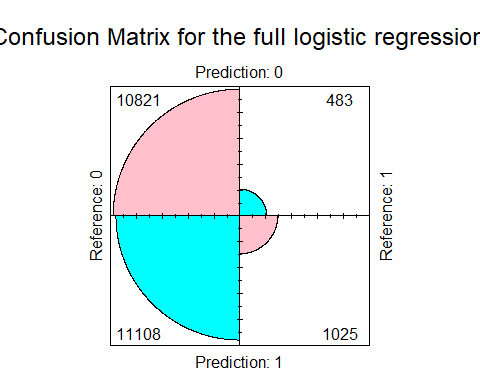
##   
## Call:  
## glm(formula = is\_claim ~ policy\_tenure + age\_of\_car, family = binomial(link = "logit"),   
## data = train\_downsampling)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.4797 -1.1518 0.1053 1.1345 1.7990   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.33365 0.06430 -5.189 2.11e-07 \*\*\*  
## policy\_tenure 1.04187 0.09563 10.895 < 2e-16 \*\*\*  
## age\_of\_car -1.59893 0.27582 -5.797 6.75e-09 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 6210.6 on 4479 degrees of freedom  
## Residual deviance: 6073.2 on 4477 degrees of freedom  
## AIC: 6079.2  
##   
## Number of Fisher Scoring iterations: 4

The selection with forward and backward at the same time results in the same variables as the backward selection. Every coefficient is also significant. The AIC of the reduced model is marginally better.

### Performance Evaluation

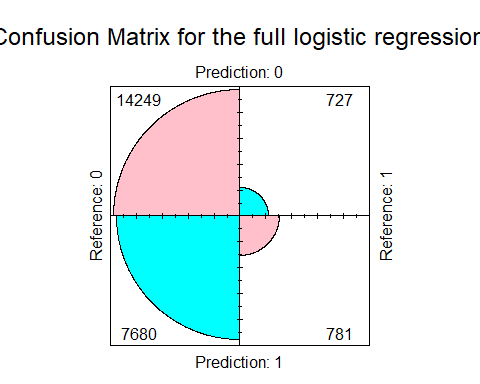
Confusion Matrix

# Full model  
pred\_logi <- predict(cars.lg, newdata = cars\_validation[,-16], type = "response")  
  
# 50% cutoff  
logi\_outcome <- as.data.table(pred\_logi)  
logi\_outcome <- ifelse(logi\_outcome > 0.5, 1, 0)  
  
  
#Confusion Matrix  
cm1 <- confusionMatrix(  
 factor(logi\_outcome),  
 factor(cars\_validation$is\_claim),   
 positive = "1")  
#Plot  
fourfoldplot(cm1$table,  
 color = c("cyan", "pink"),  
 conf.level = 0,  
 margin = 1,  
 main = "Confusion Matrix for the full logistic regression")



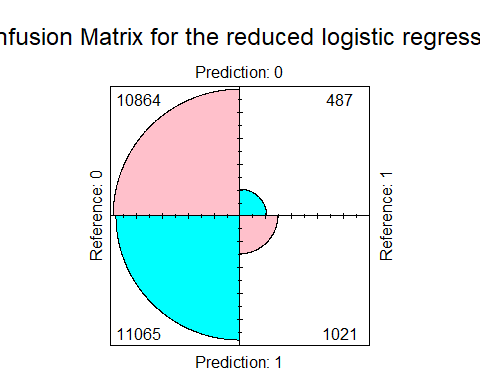
The downsampling allows us to better detect when a claim occurs, but it is at the cost of increasing the false positives which are numerous as we can see. We can try to choose another cutoff to see if we can improve the model.

# cutoff at probability of 0.55  
logi\_outcome <- as.data.table(pred\_logi)  
logi\_outcome <- ifelse(logi\_outcome > 0.55, 1, 0)  
  
#Confusion Matrix  
cm2 <- confusionMatrix(  
 factor(logi\_outcome),  
 factor(cars\_validation$is\_claim),   
 positive = "1")  
#Plot  
fourfoldplot(cm2$table,  
 color = c("cyan", "pink"),  
 conf.level = 0,  
 margin = 1,  
 main = "Confusion Matrix for the full logistic regression")



We can see the tradeoff right away. The 5% increased on the cutoff value reduced the predictions of is\_claim by about 30% for both true positives and false positives.

# Backward/Both   
cars.lg2 <- glm(is\_claim ~ policy\_tenure+age\_of\_car , data= train\_downsampling, family=binomial(link="logit"))  
pred\_logi\_back <- predict(cars.lg2, newdata = cars\_validation[,-16], type = "response")  
  
# 50% cutoff  
logi\_outcome\_back <- as.data.table(pred\_logi\_back)  
logi\_outcome\_back <- ifelse(logi\_outcome\_back > 0.5, 1, 0)  
  
#Confusion Matrix  
cm3 <- confusionMatrix(  
 factor(logi\_outcome\_back),  
 factor(cars\_validation$is\_claim),   
 positive = "1")  
#Plot  
fourfoldplot(cm3$table,  
 color = c("cyan", "pink"),  
 conf.level = 0,  
 margin = 1,  
 main = "Confusion Matrix for the reduced logistic regression")



With the model with a reduced number of variables, we seem to get a little bit more false positives true positive: more is\_claim= 1 predictions. The prediction performance is very similar, but one could argue that this reduced model performs marginally better since it reduces the false positives more than the true positives. Thus, to reduce computational resources and have a slightly better performing model, we can choose the model with the reduced number of variables. We can try to see whether the model can perform better with the whole training set combined with a very low cutoff value.

# Accuracy  
cm3$overall["Accuracy"]

## Accuracy   
## 0.5071042

#Sensitivity  
cm3$byClass["Sensitivity"]

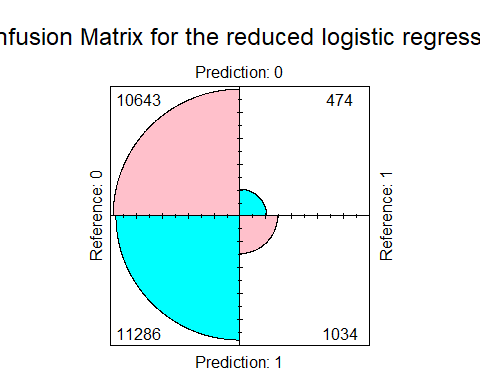
## Sensitivity   
## 0.6770557

#Specificity  
cm3$byClass["Specificity"]

## Specificity   
## 0.495417

The accuracy of 51.8% reflects the high number of false positives and negatives.

# Backward/Both   
cars.lg3 <- glm(is\_claim ~ policy\_tenure+age\_of\_car , data= cars\_train, family=binomial(link="logit"))  
pred\_logi\_back2 <- predict(cars.lg3, newdata = cars\_validation[,-16], type = "response")  
  
# 6% cutoff  
logi\_outcome\_back2 <- as.data.table(pred\_logi\_back2)  
logi\_outcome\_back2 <- ifelse(logi\_outcome\_back2 > 0.06, 1, 0)  
  
#Confusion Matrix  
cm4 <- confusionMatrix(  
 factor(logi\_outcome\_back2),  
 factor(cars\_validation$is\_claim),   
 positive = "1")  
#Plot  
fourfoldplot(cm4$table,  
 color = c("cyan", "pink"),  
 conf.level = 0,  
 margin = 1,  
 main = "Confusion Matrix for the reduced logistic regression")



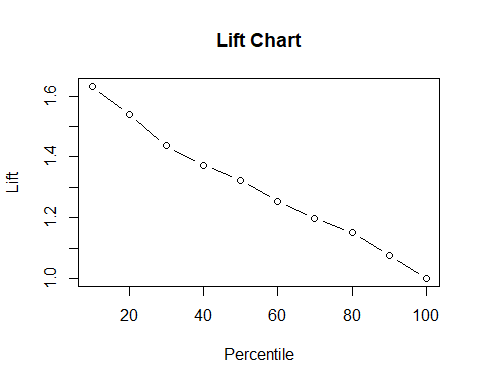
# Accuracy  
cm4$overall["Accuracy"]

## Accuracy   
## 0.4982293

With a cutoff of 6%, the model performs similarly to the model constructed on the downsample. Decreasing of increasing the cutoff results in the same aforementioned tradeoff. We get similar performance with this model, but at very low cutoff values. At a 0.5 and even 0.2, the sensitivity/specificity tradeoff is maxed out: specificity is at 1 and sensitivity 0.

Lift Chart

logi\_outcome <- as.data.table(pred\_logi\_back)  
  
liftChart(factor(logi\_outcome$pred\_logi\_back) , factor(cars\_validation$is\_claim))



topDecileLift(factor(logi\_outcome$pred\_logi) , factor(cars\_validation$is\_claim))

## [1] 1.631787

The top decile lift is at 1.63, and it gradually decreases in a linear fashion.

### Conclusion on method performance

The logistic regression’s performance is not great. The accuracy is only at 51.8% for the chosen model (2 explanatory variables with downsample) and the sensitivity of 64.4% lets room for a lot of undetected claims. The false positives are also numerous with a specificity of 50.8%.

## Ensemble

### Methods’s description

Machine learning ensemble approaches combine the results of various models to provide a single prediction. By avoiding overfitting and enhancing prediction robustness, ensemble approaches aim to enhance the generalization performance of the model.

### Methods’s application

#Actual outcome  
actual\_outcome <- cars\_validation[,6]  
  
# Classification Tree  
tree.outcome2 <- cbind(as.data.frame(tree.outcome),tree\_prob=tree.outcome)  
tree.outcome2$tree.outcome <- ifelse(tree.outcome2$tree.outcome>0.5,1,0)  
colnames(tree.outcome2) <- c("tree\_pred","tree\_prob")  
  
#KNN  
knn\_outcome2 <- as.data.frame(knn\_outcome)  
colnames(knn\_outcome2) <- c("knn\_pred", "knn\_prob")  
  
#Neural Network  
nn\_outcome2 <- data.frame(validation\_prediction\_1H\_2N[,1])  
nn\_outcome2 <- cbind(ifelse(nn\_outcome2>=0.5, 1, 0), nn\_outcome2)  
colnames(nn\_outcome2) <- c("nn\_pred","nn\_prob")  
  
#Logistic Regression  
logi\_outcome2 <- cbind( logi\_outcome, logi\_outcome)  
colnames(logi\_outcome2) <- c("logi\_pred","logi\_prob")  
logi\_outcome2$logi\_pred <- ifelse(logi\_outcome2$logi\_pred>0.5,1,0)  
  
ensemble <- data.table(actual\_outcome,tree.outcome2,knn\_outcome2,nn\_outcome2,logi\_outcome2)

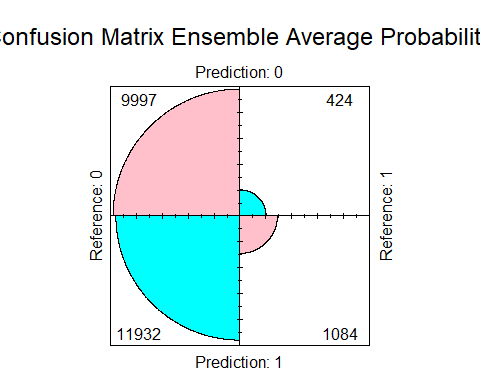
# Average probabilities column  
ensemble[, avg\_prob := (logi\_prob + knn\_prob + tree\_prob + nn\_prob) / 4]  
  
# # Majority vote  
ensemble[, maj\_vote := as.numeric(as.character(logi\_pred))+ as.numeric(as.character(knn\_pred)) + as.numeric(as.character(tree\_pred))+ as.numeric(as.character(nn\_pred))]  
  
ensemble[, maj\_vote := ifelse(maj\_vote >= 3, 1, 0)]  
  
# Display first 10 rows  
kable(ensemble[1:10, ], format = "markdown")

| actual\_outcome | tree\_pred | tree\_prob | knn\_pred | knn\_prob | nn\_pred | nn\_prob | logi\_pred | logi\_prob | avg\_prob | maj\_vote |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0.4017857 | 1 | 0.5365854 | 1 | 0.6152165 | 1 | 0.5708271 | 0.5311037 | 1 |
| 0 | 0 | 0.3828829 | 0 | 0.4390244 | 0 | 0.3818694 | 0 | 0.4878397 | 0.4229041 | 0 |
| 0 | 1 | 0.6216216 | 1 | 0.5609756 | 0 | 0.3818694 | 1 | 0.5849568 | 0.5373559 | 1 |
| 0 | 0 | 0.3182979 | 0 | 0.4146341 | 1 | 0.6839425 | 0 | 0.4431299 | 0.4650011 | 0 |
| 0 | 1 | 0.7042254 | 0 | 0.4878049 | 0 | 0.3821588 | 1 | 0.5955241 | 0.5424283 | 0 |
| 0 | 0 | 0.3043478 | 0 | 0.4634146 | 0 | 0.3818694 | 0 | 0.4770876 | 0.4066799 | 0 |
| 0 | 0 | 0.3182979 | 0 | 0.4634146 | 1 | 0.6839425 | 0 | 0.4260763 | 0.4729328 | 0 |
| 0 | 0 | 0.3182979 | 0 | 0.3658537 | 1 | 0.6839425 | 0 | 0.4547489 | 0.4557107 | 0 |
| 1 | 1 | 0.7042254 | 1 | 0.5365854 | 0 | 0.3818740 | 1 | 0.5183462 | 0.5352577 | 1 |
| 0 | 1 | 0.6216216 | 0 | 0.3902439 | 1 | 0.6839425 | 1 | 0.5064656 | 0.5505684 | 1 |

### Performance Evaluation

Using the average probability with a 0.5 cutoff

#Confusion Matrix   
cm\_ens\_prob <- confusionMatrix(  
 factor(ifelse(ensemble$avg\_prob > 0.5, 1 ,0)),  
 factor(cars\_validation$is\_claim),   
 positive = "1")  
#Plot  
fourfoldplot(cm\_ens\_prob$table,  
 color = c("cyan", "pink"),  
 conf.level = 0,  
 margin = 1,  
 main = "Confusion Matrix Ensemble Average Probability")



# Accuracy  
cm\_ens\_prob$overall["Accuracy"]

## Accuracy   
## 0.4727994

#Sensitivity  
cm\_ens\_prob$byClass["Sensitivity"]

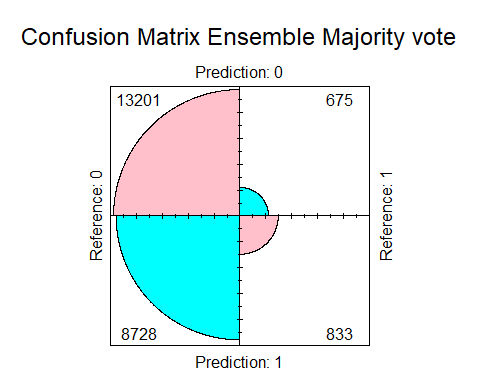
## Sensitivity   
## 0.7188329

#Specificity  
cm\_ens\_prob$byClass["Specificity"]

## Specificity   
## 0.4558803

Using the majority vote

cm\_ens\_maj <- confusionMatrix(  
 factor(ensemble$maj\_vote),  
 factor(cars\_validation$is\_claim),   
 positive = "1")  
#Plot  
fourfoldplot(cm\_ens\_maj$table,  
 color = c("cyan", "pink"),  
 conf.level = 0,  
 margin = 1,  
 main = "Confusion Matrix Ensemble Majority vote")



# Accuracy  
cm\_ens\_maj$overall["Accuracy"]

## Accuracy   
## 0.5987968

#Sensitivity  
cm\_ens\_maj$byClass["Sensitivity"]

## Sensitivity   
## 0.5523873

#Specificity  
cm\_ens\_maj$byClass["Specificity"]

## Specificity   
## 0.6019882

The majority vote approach’s lesser sensitivity may be due in part to the fact that its projections are more “moderate” in terms of is\_claim=1 predictions. Making a choice based on the majority vote effectively requires that the majority of the models agree on the forecast, which can reduce the likelihood of false positives (i.e., predicting the outcome variable as 1 when it is actually 0). Because there will be fewer overall true positive predictions, the sensitivity may be reduced as a result. On the other hand, because it is essentially averaging the probabilities of all the models, the average probability technique may be less conservative in its forecasts (more is\_claim =1 predictions). Although there may be more false positives as a result, there may also be more actual positive predictions, increasing sensitivity.

### Conclusion

Overall, given that the majority vote strategy offers greater precision and specificity than the average probability approach, it would be a preferable option for this particular case. When selecting an ensemble approach, it’s crucial to take the trade-offs between the various performance measures into account because different applications could give particular metrics a higher priority (e.g., sensitivity versus specificity).

# All methods comparison

Overall, we are interested in a balance between Sensitivity and Specificity as we need to identify correctly claimed files, but also to avoid mistakes in negatives predictions. The situation is tricky since that situation corresponds to a trade-off. Accepting a little decrease in specificity would provide a lot of misclassified negative outcomes given the commonness of that class. On the other hand, accepting a little decrease in sensitivity would goe against or goal.

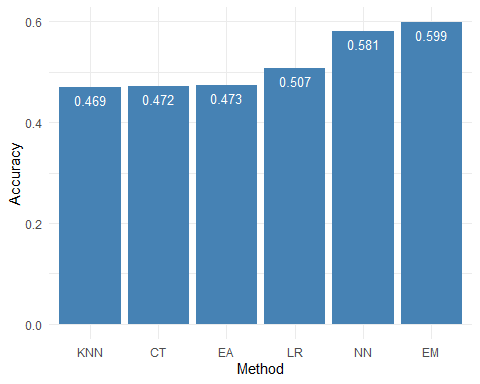
#### Accuracy

Accuracy measures the quantity of all correct predictions. Formula: TP+TN / TP + TN +FP + FN

Accuracy <- data.frame(LR = round(cm3$overall[1],3)) #predict accuracy by using logistic regression   
Accuracy <- cbind(Accuracy, KNN = round(cmk$overall[1],3)) #predict accuracy by using KNN  
Accuracy <- cbind(Accuracy,CT = round(conf\_matrix\_tree\_v$overall[1],3)) #predict accuracy by using tree  
Accuracy <- cbind(Accuracy,NN=round(conf\_matrix\_1H2N\_valid$overall[1],3)) #predict accuracy by using neural network  
Accuracy <- cbind(Accuracy,EM=round(cm\_ens\_maj$overall[1],3)) #predict accuracy ensemble majority   
Accuracy <- cbind(Accuracy,EA=round(cm\_ens\_prob$overall[1],3)) #predict accuracy ensemble average   
  
Accuracy.melt = melt(Accuracy)

## No id variables; using all as measure variables

ggplot(data = Accuracy.melt,aes(x=reorder(variable, value),y=value))+ylab("Accuracy") + xlab("Method") +  
 geom\_bar(stat="identity", fill="steelblue")+  
 geom\_text(aes(label=value), vjust=1.6, color="white", size=3.5)+  
 theme\_minimal()



When comparing all methods we see that two models outperform the others. The ensemble with votes model provides an accuracy of 59.9% and is followed by Neural net with a score of 58.1%. The difference is not so big. We retain that Ensemble model with votes is the model providing the most correct positives and negatives predictions.

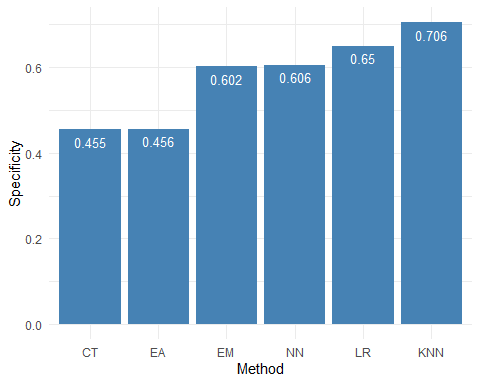
#### Specificity

Specificity measures the quantity of correct negative predictions. Formula: Specificity = TN / (TN + FP)

Specificity\_P <- data.frame(LR = round(cm2$byClass[2],3)) #predict specificity by using logistic regression   
Specificity\_P <- cbind(Specificity\_P, KNN = round(cmk$byClass[2],3)) #predict specificity by using KNN  
Specificity\_P <- cbind(Specificity\_P,CT = round(conf\_matrix\_tree\_v$byClass[2],3)) #predicts specificity tree  
Specificity\_P <- cbind(Specificity\_P,NN=round(conf\_matrix\_1H2N\_valid$byClass[2],3)) #predict specificity neural network  
Specificity\_P <- cbind(Specificity\_P,EM=round(cm\_ens\_maj$byClass[2],3)) #predict specificity majority voting   
Specificity\_P <- cbind(Specificity\_P,EA=round(cm\_ens\_prob$byClass[2],3)) #predict specificity ensemble average   
  
Specificity\_P.melt = melt(Specificity\_P)

## No id variables; using all as measure variables

ggplot(data = Specificity\_P.melt,aes(x=reorder(variable, value),y=value))+ylab("Specificity") + xlab("Method") +  
 geom\_bar(stat="identity", fill="steelblue")+  
 geom\_text(aes(label=value), vjust=1.6, color="white", size=3.5)+  
 theme\_minimal()



KNN model is the best in class in this category. It outperforms the others with a ratio of 70.6%, while the classification tree and ensemble average have poor performances. This score is almost 5% better than the second best model (Logistic Regression). A median performance could be expressed by the Ensemble vote and the Neural network. We retain that KNN is very good at classifying negatives outcomes.

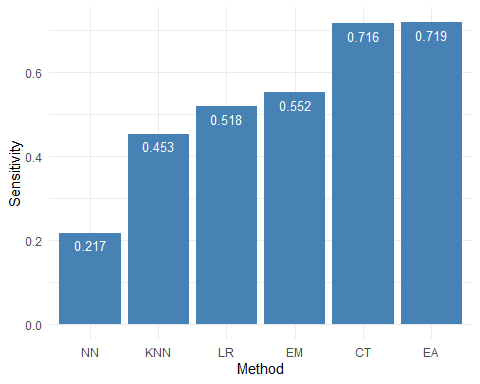
#### Sensitivity

Sensitivity measures the quantity of correct positive predictions. Formula: Sensitivity = TP / (TP + FN)

Sensitivity <- data.frame(LR = round(cm2$byClass[1],3)) #predict Sensitivity by using logistic regression   
Sensitivity <- cbind(Sensitivity, KNN = round(cmk$byClass[1],3)) #predict Sensitivity by using KNN  
Sensitivity <- cbind(Sensitivity,CT = round(conf\_matrix\_tree\_v$byClass[1],3)) #predicts Sensitivity tree  
Sensitivity <- cbind(Sensitivity,NN=round(conf\_matrix\_1H2N\_valid$byClass[1],3)) #predict Sensitivity neural network  
Sensitivity <- cbind(Sensitivity,EM=round(cm\_ens\_maj$byClass[1],3)) #predict Sensitivity majority voting   
Sensitivity <- cbind(Sensitivity,EA=round(cm\_ens\_prob$byClass[1],3)) #predict Sensitivity ensemble average   
  
Sensitivity.melt = melt(Sensitivity)

## No id variables; using all as measure variables

ggplot(data = Sensitivity.melt,aes(x=reorder(variable, value),y=value))+ylab("Sensitivity") + xlab("Method") +  
 geom\_bar(stat="identity", fill="steelblue")+  
 geom\_text(aes(label=value), vjust=1.6, color="white", size=3.5)+  
 theme\_minimal()



Two models are distinct in this category: Classification tree with a score of almost 71.6% and Ensemble average with a score of almost 71.9 %. The difference is extremely small and denote a similar performance. The worst performance is provided by the Neural net with 21.7% of sensitivity. The median performance could be expressed with the Logistic regression and Ensemble vote models. We retain that Ensemble Average(1st) and Classification tree(2nd) are the models providing the most correct positives predictions.

# Conclusions

Our team completed an analysis related to a car insurance business. We wanted to identify if a policyholder was going to claim a file or not by using different methods of classification and to find the best model predicting such a situation. We did it through few methods providing different performances. They were assessed through a range of metrics such as accuracy, specificity and sensitivity. However, the relevant criterion for determining the best model is found in a trade-off between sensitivity and specificity. This criterion determined the voting ensemble as the best method for our business case. Overall, we saw that the best accuracy was given by the ensemble with votes and represented our needed trade-off, but it was also expressed in the median values with respect to specificity and sensitivity. This situation contrasts with the other methods of classification where rather placed at boundaries which doesn’t express our willingness of good performance in both correct positive and correct negative predictions. The solution for the car insurance case lies then in the combination of many models and can be expressed through the use of the ensemble voting method.