Car Insurance Claims

The miners

2022-12-11

Contents


```
##
## Attachement du package : 'dplyr'
```

```
## Les objets suivants sont masqués depuis 'package:stats':
##
##     filter, lag
```

```
## Les objets suivants sont masqués depuis 'package:base':
```

```
##
##     intersect, setdiff, setequal, union

## randomForest 4.7-1.1

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

##
## Attachement du package : 'randomForest'

## L'objet suivant est masqué depuis 'package:dplyr':
##
##     combine

## L'objet suivant est masqué depuis 'package:ggplot2':
##
##     margin

## corrplot 0.92 loaded

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

## Le chargement a nécessité le package : lattice

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

##
## Attachement du package : 'neuralnet'

## L'objet suivant est masqué depuis 'package:dplyr':
##
##     compute

##
## Attachement du package : 'data.table'

## Les objets suivants sont masqués depuis 'package:dplyr':
##
##     between, first, last

## Le chargement a nécessité le package : foreach

## Le chargement a nécessité le package : iterators

## Le chargement a nécessité le package : parallel
```

Introduction

about that industry

Being able to predict the risk of a policyholder's complaint is fundamental for an insurance company. It could impact its profitability. Our team decided to dive in that industry. We decided to conduct an analysis 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 predictive model for such a situation.

classification

A dataset containing information about insurance holders and details about them is going to be used to achieve progress towards 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" This is a boolean indicator showing if a for the respective 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 once our classification models have been trained.

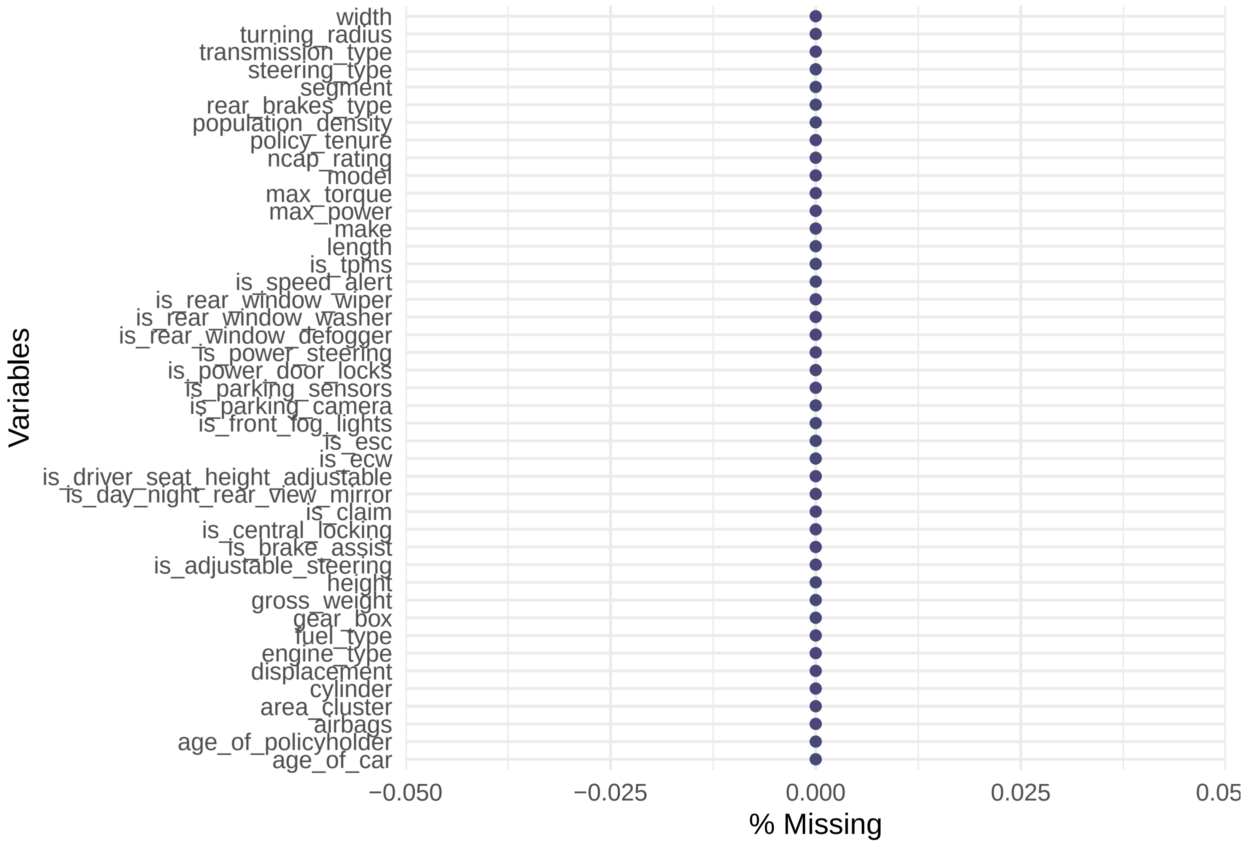
To conduct our research, we plan to undertake the following steps. First, we will have a first look at the data and drop some variables, because our data set is very large and requires a lot of computation power. Then, We will continue with a regular exploratory data analysis to understand our data. After this step, 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. Finally we will compare these method to identify the most suitable for our case.

We will finish by comparing

Let's start by filtering our dataset before starting our EDA as said.

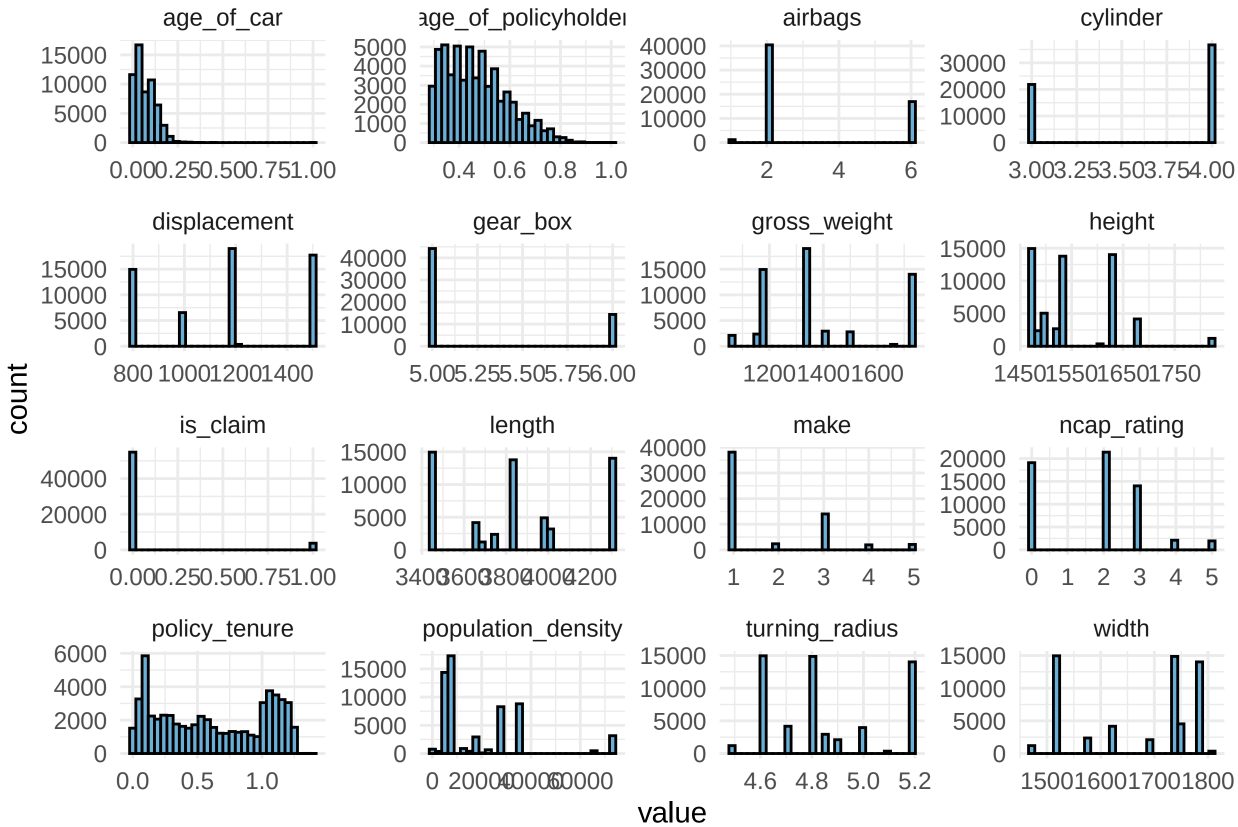
```
## 'data.frame':    58592 obs. of  44 variables:
##  $ policy_id                       : chr  "ID00001" "ID00002" "ID00003" "ID00004" ...
##  $ policy_tenure                   : num  0.516 0.673 0.841 0.9 0.596 ...
##  $ age_of_car                      : num  0.05 0.02 0.02 0.11 0.11 0.07 0.16 0.14 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@4400rp
##  $ max_power                       : chr  "40.36bhp@6000rpm" "40.36bhp@6000rpm" "40.36bhp@6000rpm" "
##  $ engine_type                     : chr  "F8D Petrol Engine" "F8D Petrol Engine" "F8D Petrol Engine
##  $ 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" ...
```

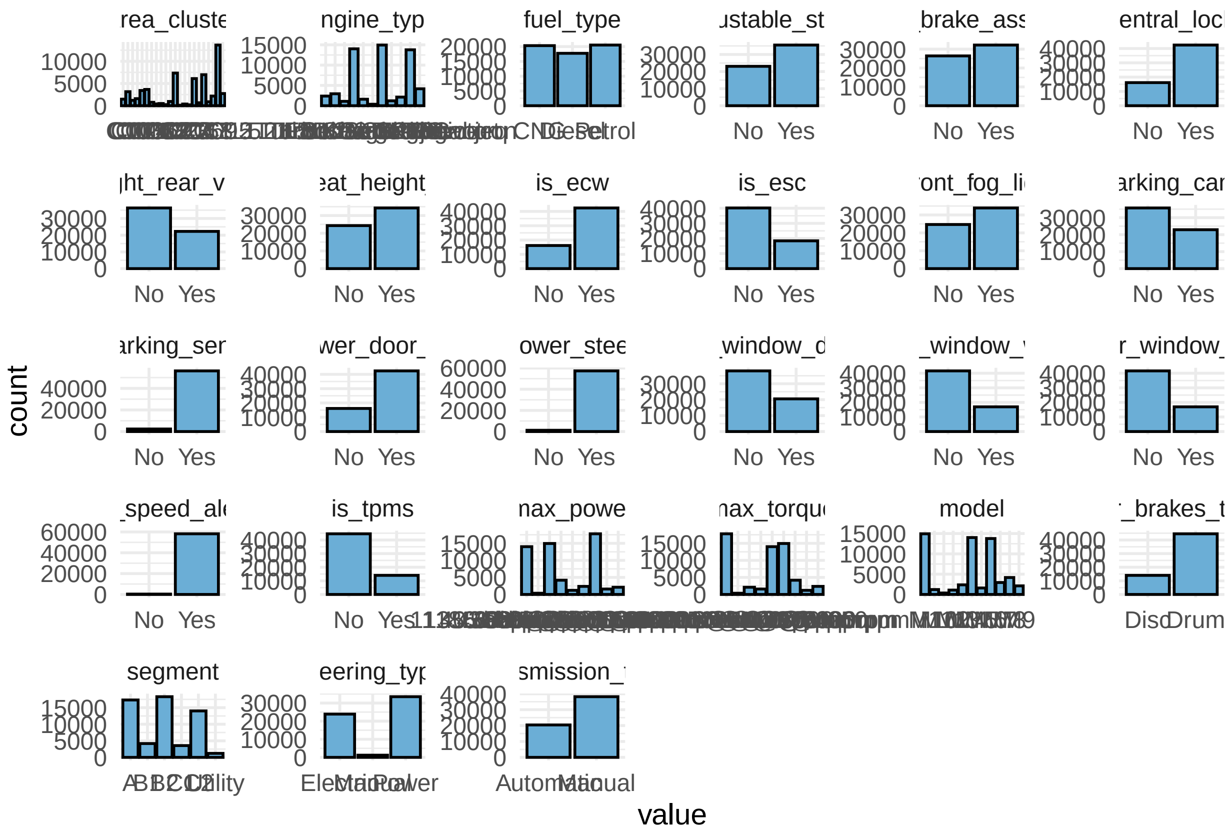
```
##  $ ncap_rating                     : int  0 0 0 2 2 3 5 2 3 0 ...
##  $ is_claim                        : int  0 0 0 0 0 0 0 0 0 0 ...

##                    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
```



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.

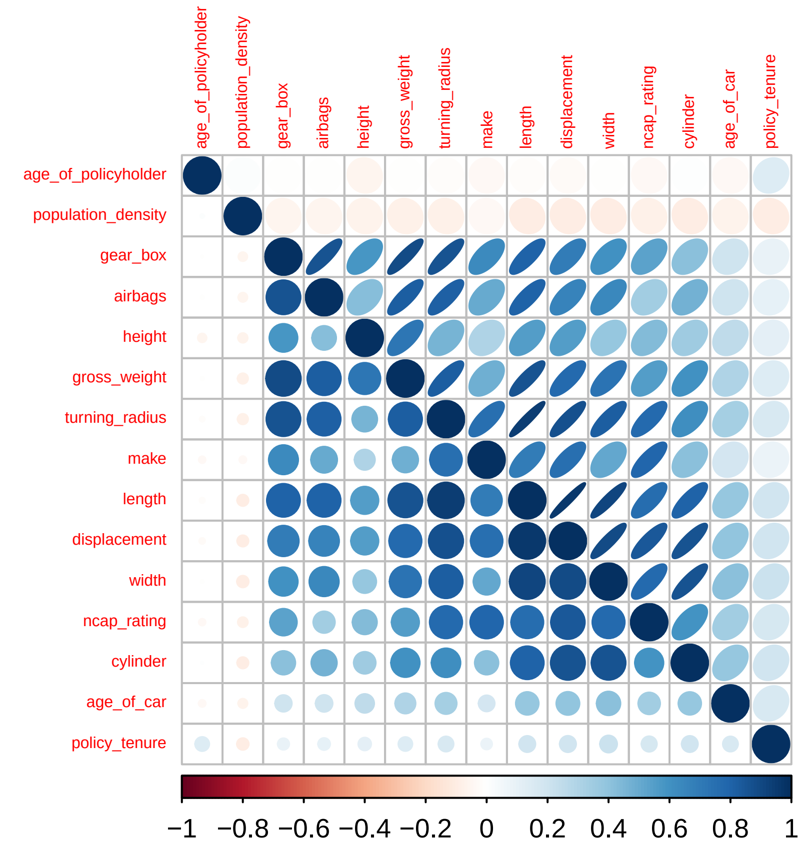
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


age_of_car
age_of_policyholder
airbags
cylinder
displacement
gear_box
gross_weight
height
is_claim
length
make
ncap_rating
policy_tenure
population_density
turning_radius
width
count
value




```
##                      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
```



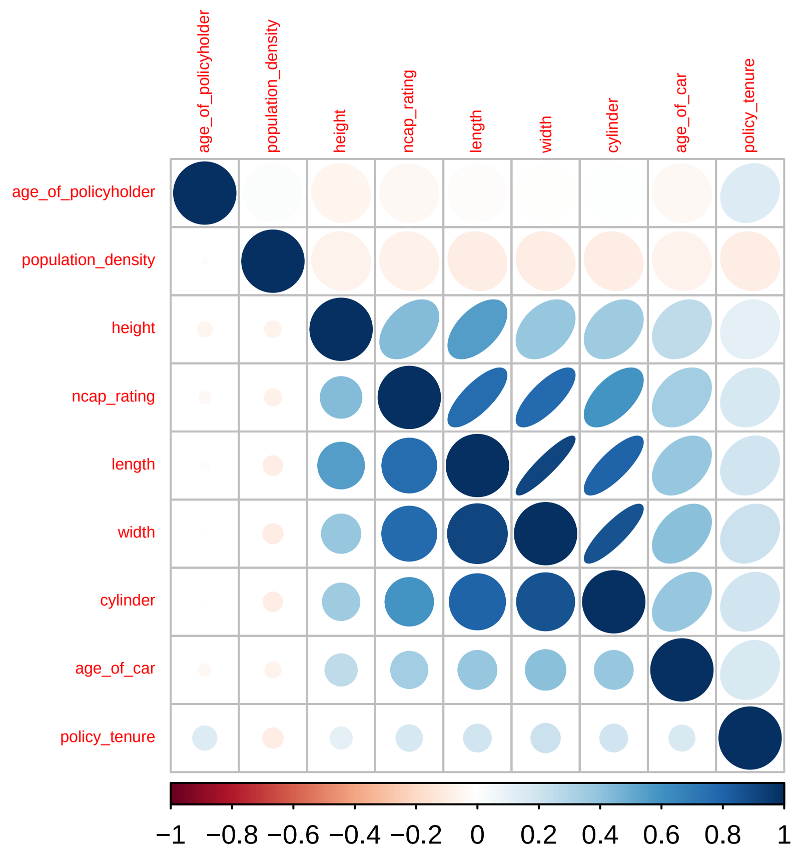
Looking at correlations we note a high correlation for the variable displacement such as cylinder (0.87), turning_radius with length (0.95), gross_weight with length (0.86), width with length (0.92).

displacement, airbags, gear_box, turning_radius, gross_weight

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.

```
##                     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
```



We drop the variables width, segment, engine_type, max_torque, max_power, ncap_rating and rating.

CHECK code

```
##
## 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.

```
##
## 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
```

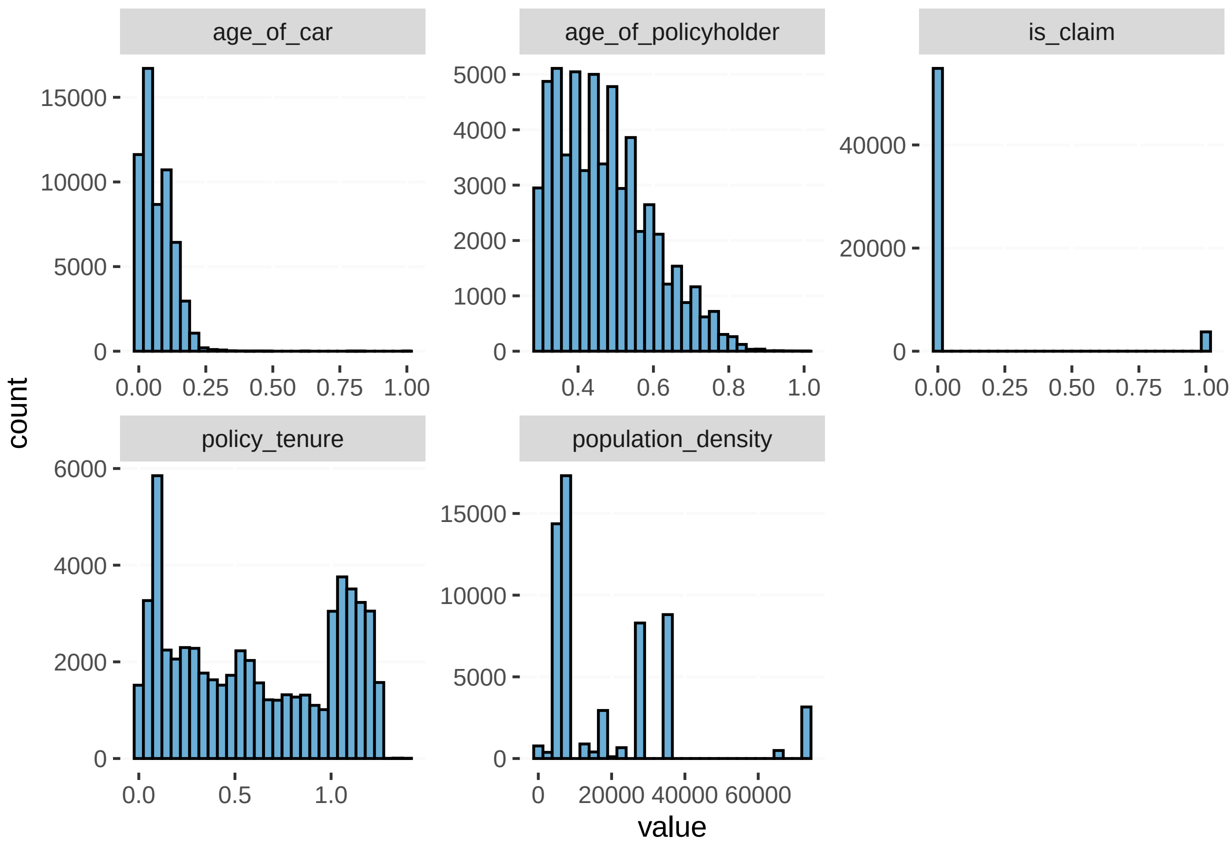
policy_tenure	age_of_car	age_of_policyholder	population_density	model	is_claim
0.5158736	0.05	0.6442308	4990	M1	0
0.6726185	0.02	0.3750000	27003	M1	0
0.8411103	0.02	0.3846154	4076	M1	0

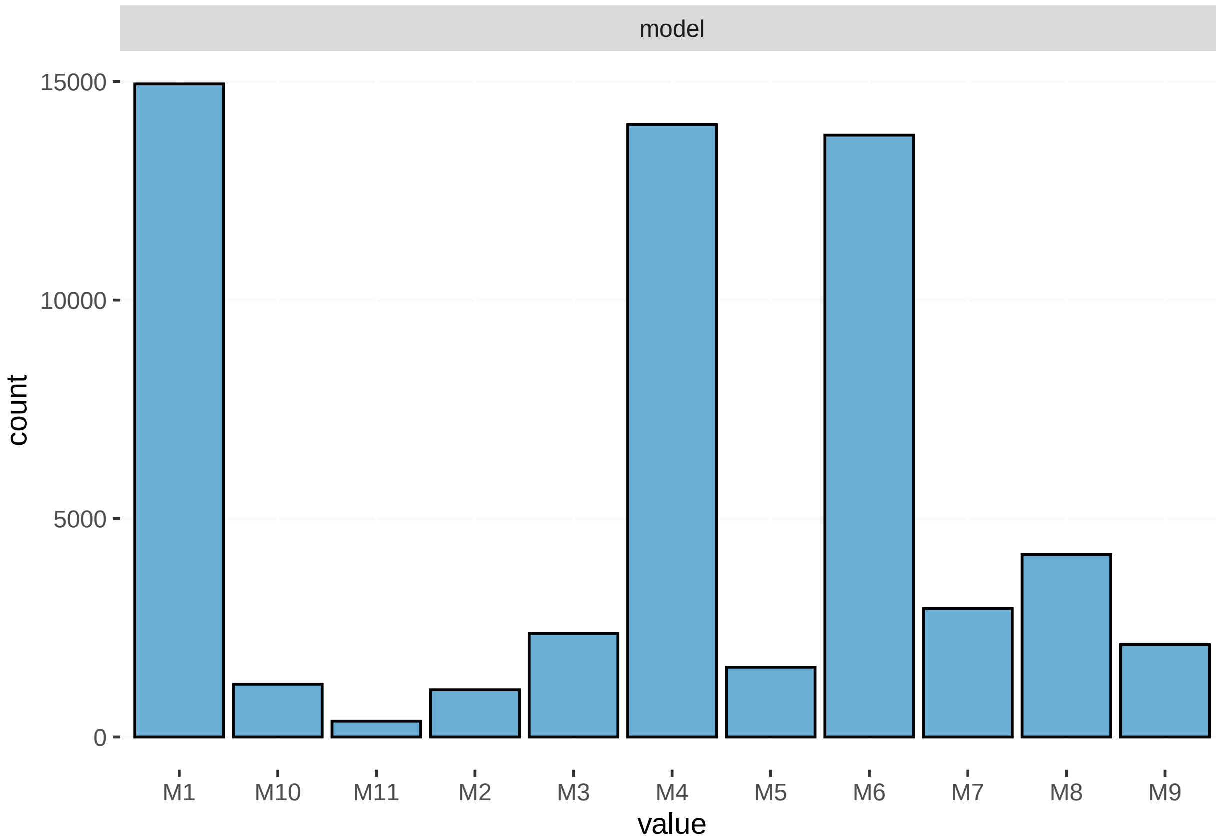
policy_tenure	age_of_car	age_of_policyholder	population_density	model	is_claim
0.9002766	0.11	0.4326923	21622	M2	0
0.5964028	0.11	0.6346154	34738	M3	0
1.0187085	0.07	0.5192308	13051	M4	0

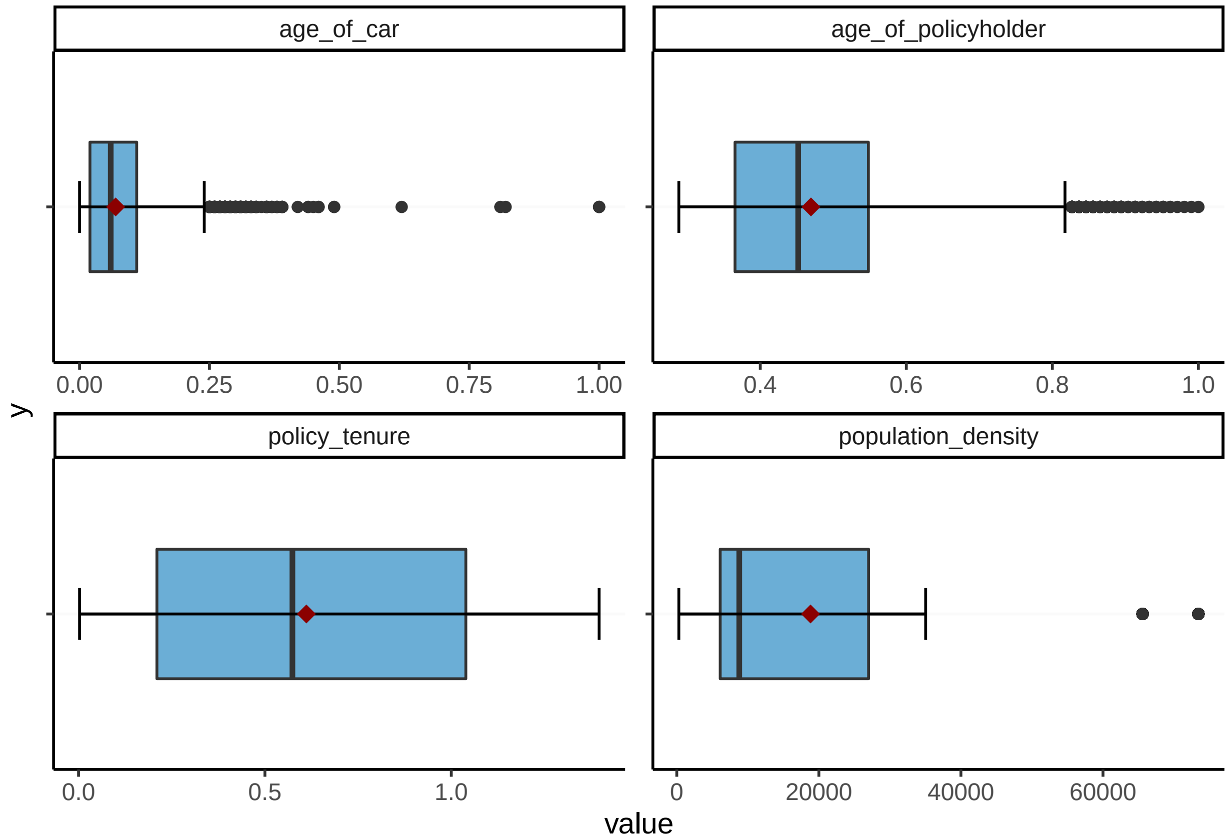
Based on our previous analysis, our finals explanatory variables would be: policy_tenure, age_of_car, age_of_policyholder, model and population_density.

EDA

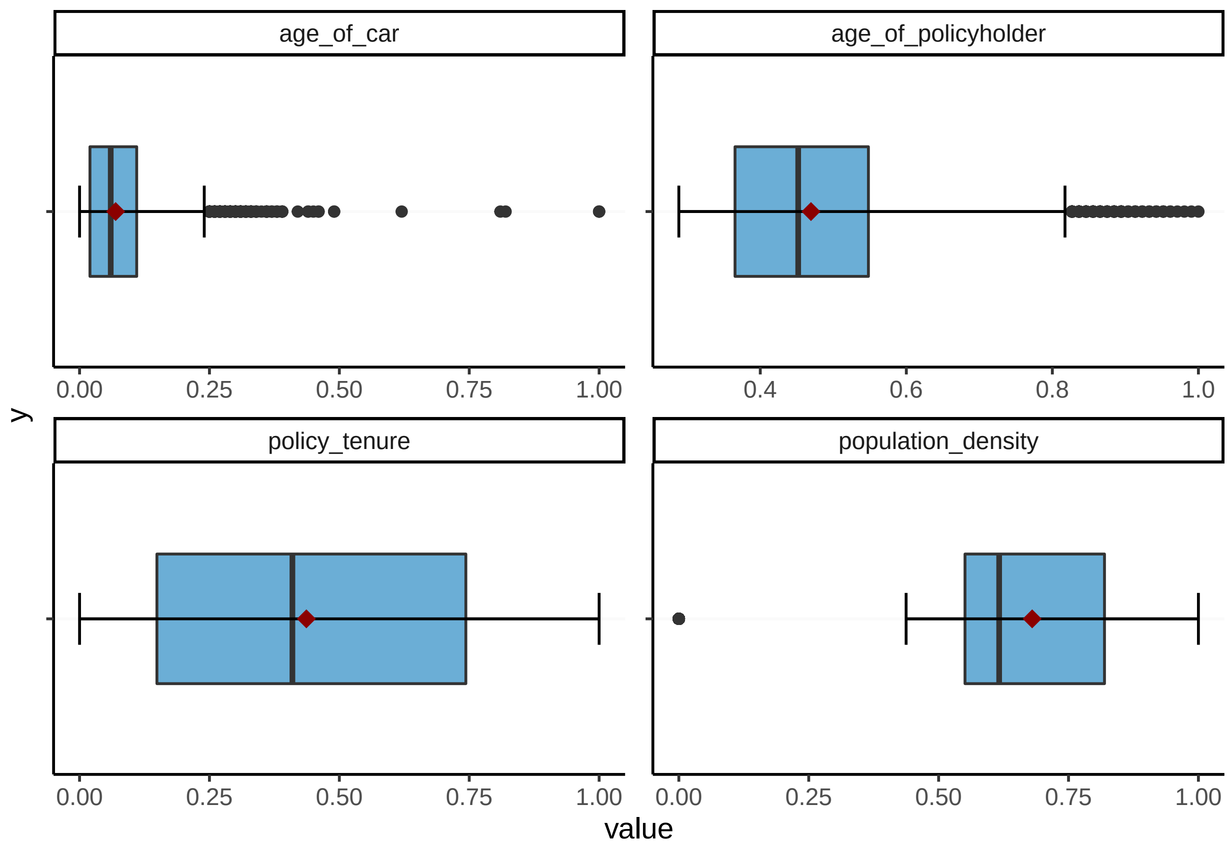
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




model
15000
10000
5000
0
count
M1
M10
M11
M2
M3
M4
M5
M6
M7
M8
M9
value



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.



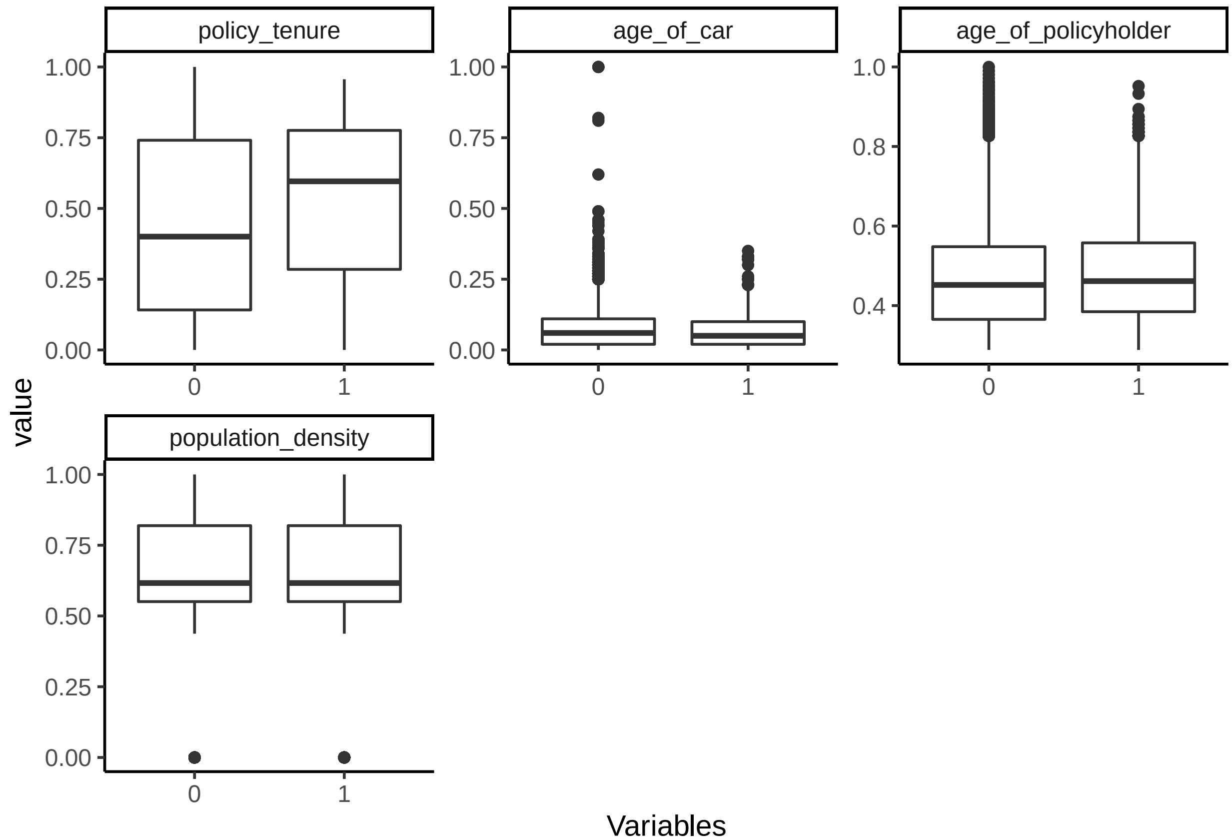
With the log transformation on population_density, the distribution has become a little bit more centered and symmetrical.

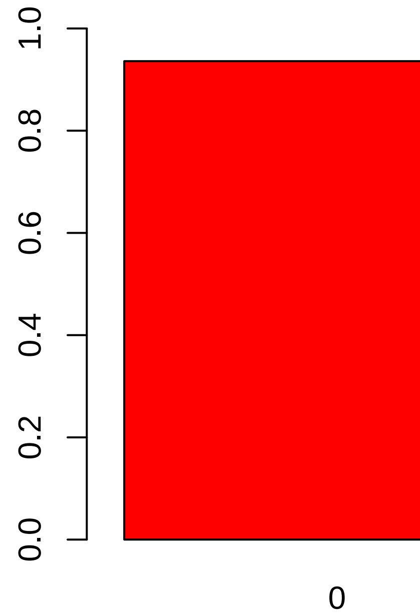
	policy_tenure	age_of_car	age_of_policyholder	population_density	model	is_claim
	Min. :0.0000 1st Qu.:0.1489 Median :0.4097 Mean :0.4366 3rd Qu.:0.7435 Max. :1.0000	Min. :0.00000 1st Qu.:0.02000 Median :0.06000 Mean :0.06942 3rd Qu.:0.11000 Max. :1.00000	Min. :0.2885 1st Qu.:0.3654 Median :0.4519 Mean :0.4694 3rd Qu.:0.5481 Max. :1.0000	Min. :0.0000 1st Qu.:0.5508 Median :0.6165 Mean :0.6801 3rd Qu.:0.8192 Max. :1.0000	Length:58592 Class :character Mode :character NA NA NA	Min. :0.00000 1st Qu.:0.00000 Median :0.00000 Mean :0.06397 3rd Qu.:0.00000 Max. :1.00000

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

```
##
## Attachement du package : 'reshape2'

## Les objets suivants sont masqués depuis 'package:data.table':
##
##     dcast, melt

## L'objet suivant est masqué depuis 'package:tidyr':
```

```
##
##     smiths
## Using is_claim as id variables
```





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

```
##
##          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.

```
##
##          0          1
## 0.93628218 0.06371782
```

```
##
##          0          1
## 0.93565729 0.06434271
```

```
##  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

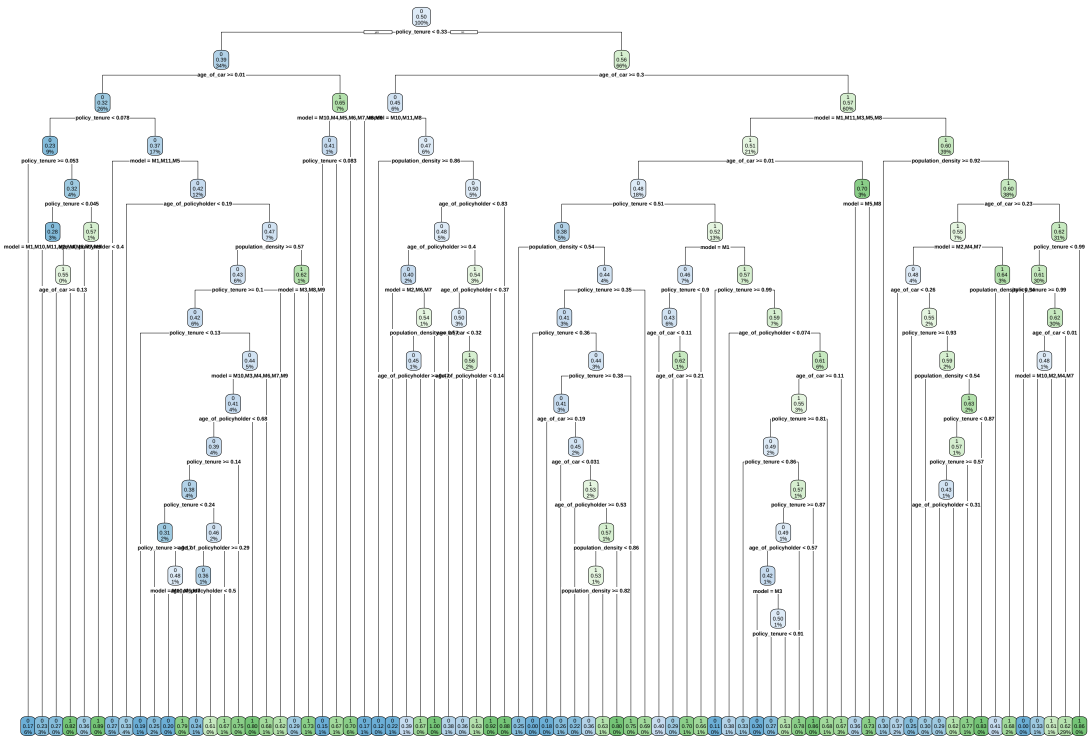
can be placed in rectangular areas

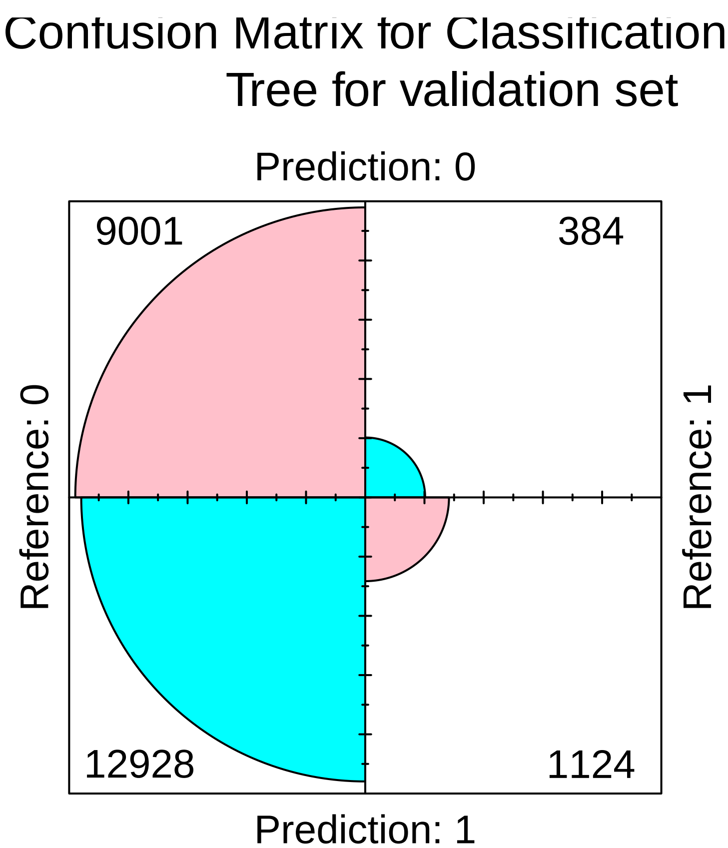
A classification tree is a method of supervised algorithm used for classification. It consists in continuously splitting the data into sub-parts based on data features (sub-parts form rectangles on the graph) until getting only one class in the splitted area. At that point, data's class is the most homogeneous 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 ranges 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





We start by fitting a full grown tree. It allows us to get a first look at the performances. We see that tree is better at classifying negative outcomes and struggles with False positives.

We see that using a balanced data set solved our issue and display a full grown tree.

We want to identify the
somewhat

We try to look at the lowest cp value to prune the tree. The found value equals 0.001. The value is similar to the default value that we chosen to compute to fit the classification tree.

chosed
ful

However It is known that we should not choose a tree based on it's cp, but we should choose rather a tree size. Ordinarily, we would like to have a small tree with a small cp. Such a tree would avoid overfitting.

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

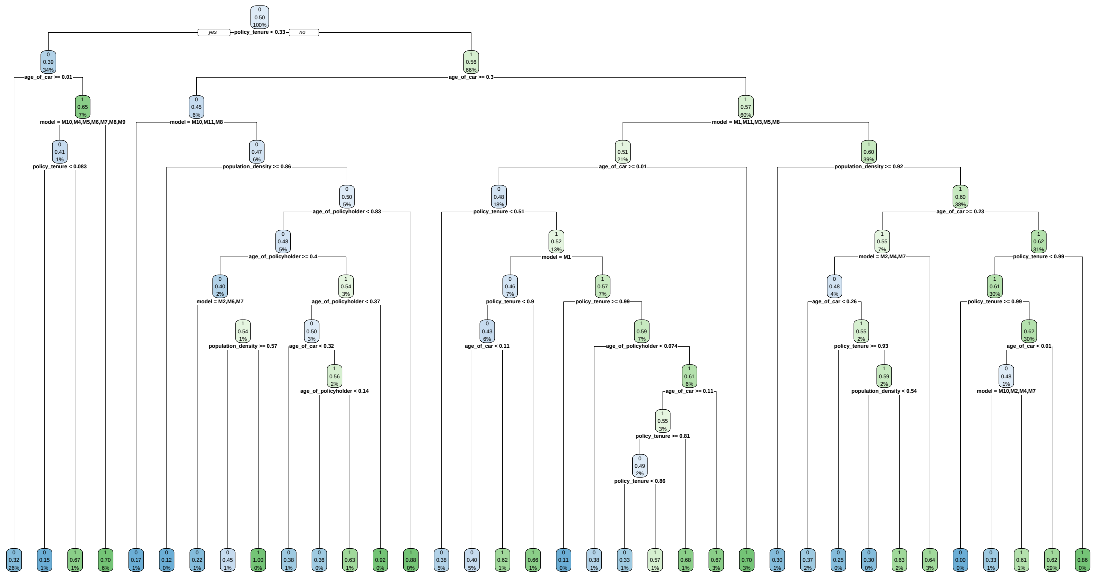
```
##        16        17        10        11        12        18
## 0.8065778 0.8065778 0.8074735 0.8083691 0.8088170 0.8106083
```

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

find

To choose a tree size we should look for the smallest minimum error within one std. error and a small CP value.

We see that the fourth observation where the nsplit = 13 and CP = 0.003 gives the best error.



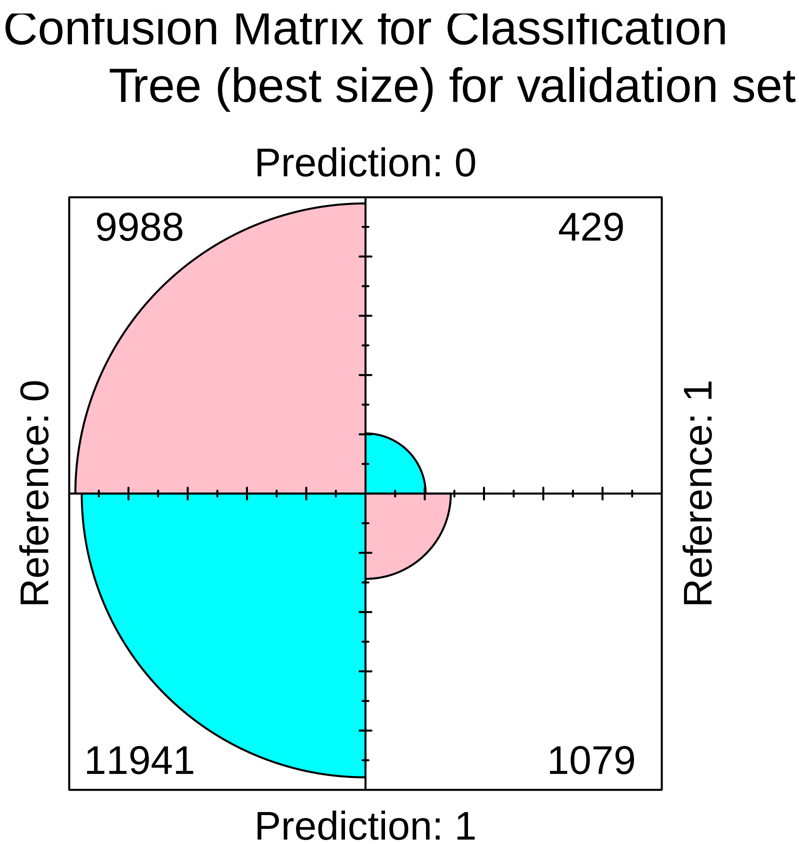
```
## [1] 36
```

The resulting pruned tree with at its best size has 15 leaves. presents this shape.

Performance evaluation

```
## 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
##
```



Performance evaluation: we first fitted a full grown tree that showed mitigated performances. The full tree predicted better the negative outcomes than the positive and had issues with False positives. We then look at the resulting final tree.

Looking at the performance metrics:

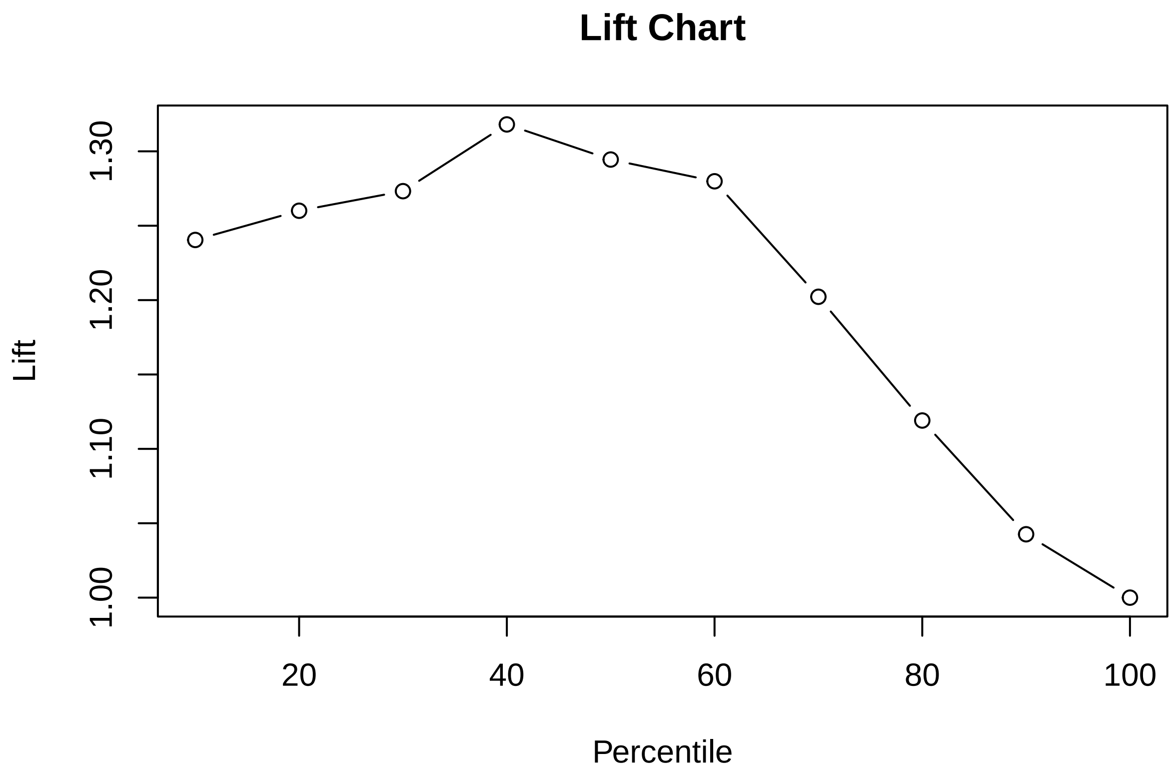
- Accuracy highs at 53.7 %
- Specificity highs at 52.6 %
- Sensitivity is 68.6%

From all positive outcome predicted, only 9 % where right whereas 96% of negative outcomes where correctly predicted.

Conclusion on method's performance

We conclude that predicting positive outcomes is difficult for the model since not even 10% of its positive predictions were right. The model is better at predicting negative outcomes than positive ones. Therefore, we may want to rely on an other model with respect to positive prediction.

Lift Chart




```
## [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

```
##   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
```

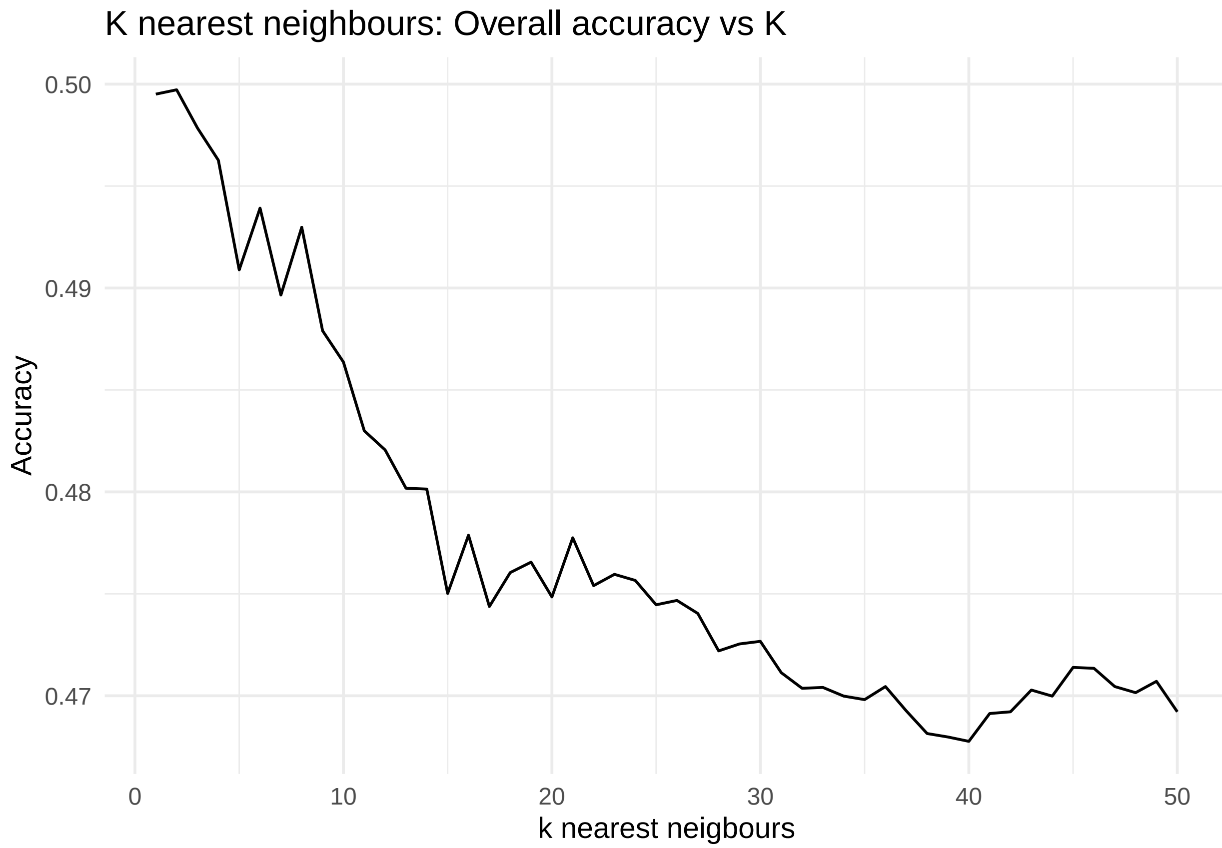
```
## 'data.frame':    58592 obs. of  16 variables:
##  $ policy_tenure      : num  0.368 0.481 0.601 0.644 0.426 ...
##  $ age_of_car         : num  0.05 0.02 0.02 0.11 0.11 0.07 0.16 0.14 0.07 0.04 ...
##  $ age_of_policyholder: num  0.644 0.375 0.385 0.433 0.635 ...
##  $ population_density : num  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 ...
```

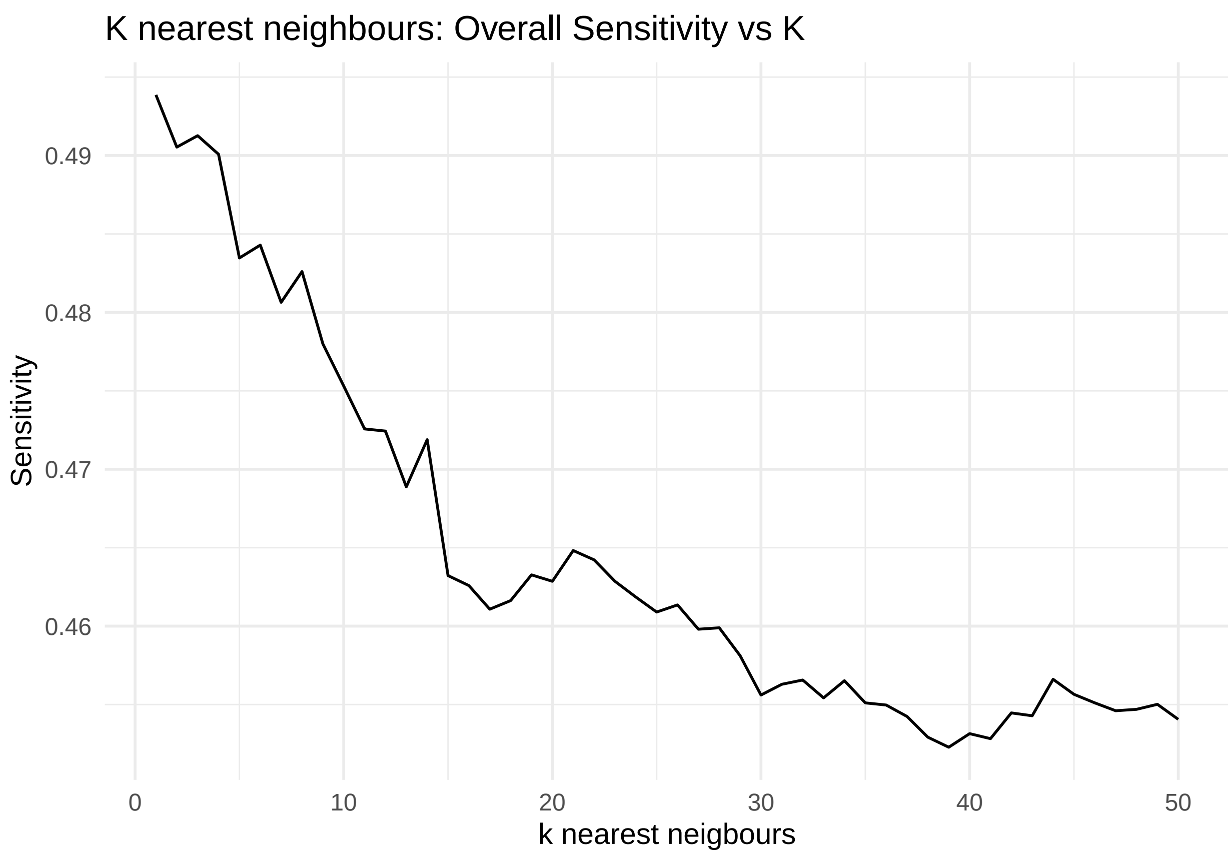
Method application

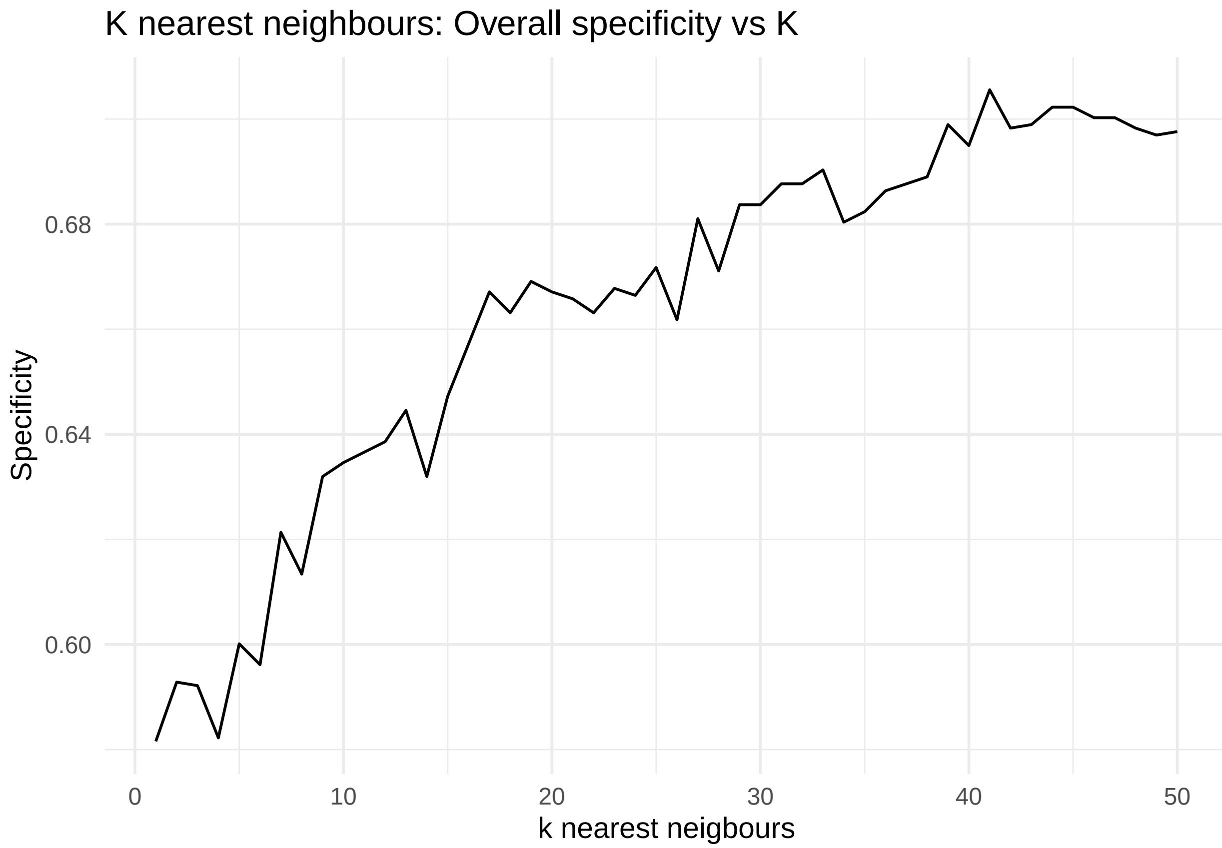
```
## [1] 2
```

```
## [1] 41
```

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

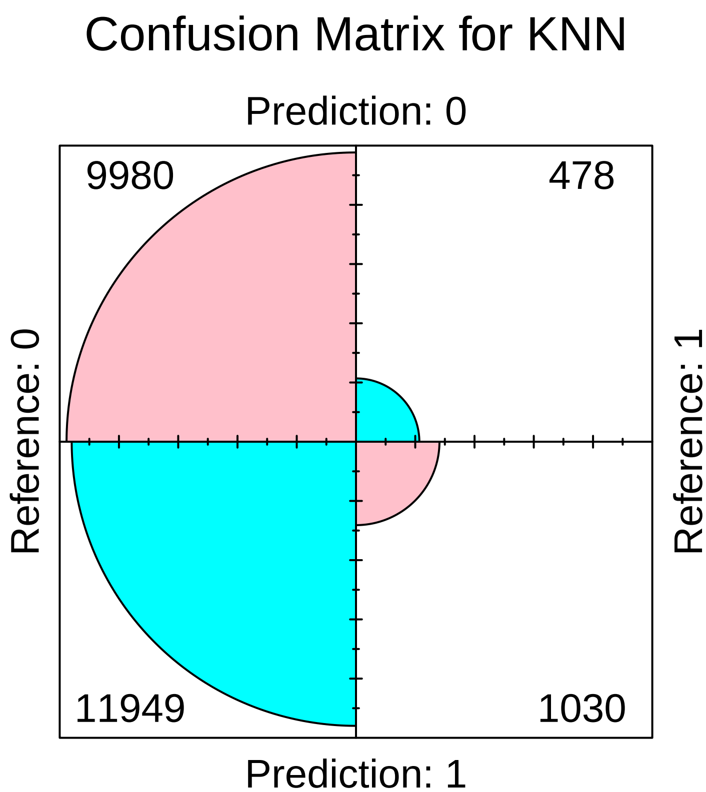

K nearest neighbours: Overall accuracy vs K
0.50
0.49
0.48
0.47
Accuracy
0
10
20
30
40
50
k nearest neigbours


K nearest neighbours: Overall Sensitivity vs K
Sensitivity
0.49
0.48
0.47
0.46
0
10
20
30
40
50
k nearest neigbours



Since we are looking for k which give us max optimum value of Specificity we chose k = 31

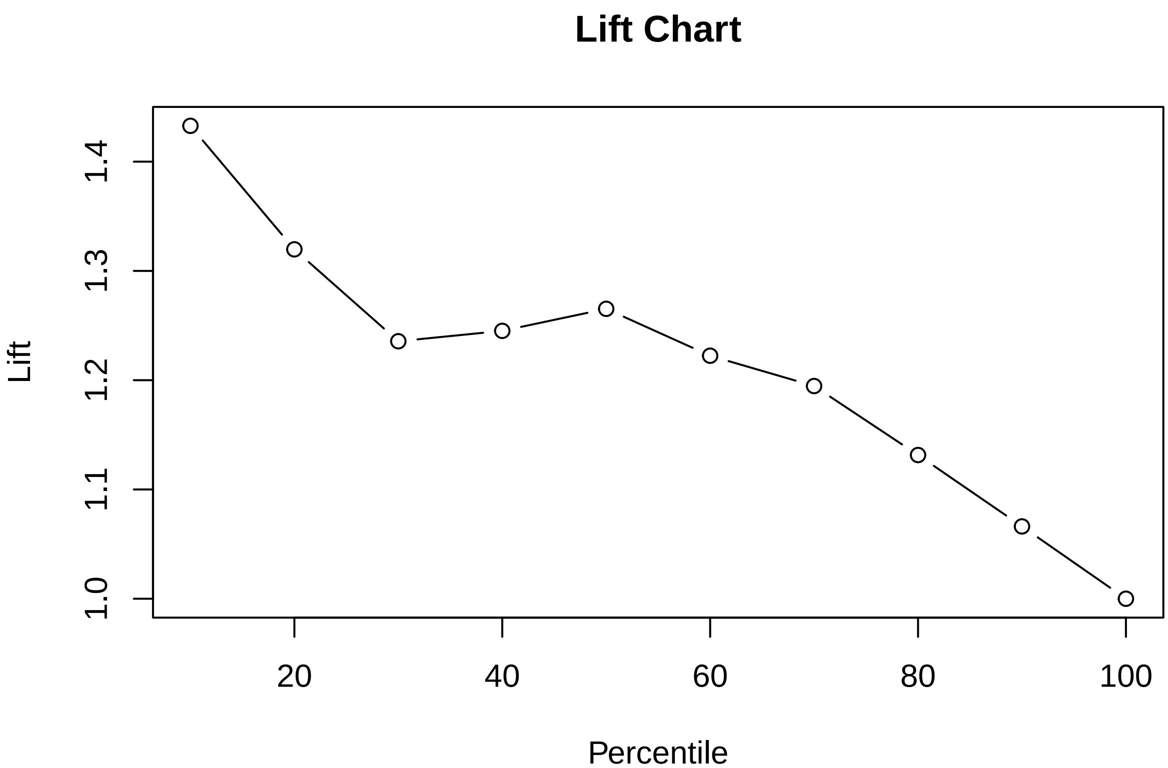
```
## Specificity
##   0.6830239
```




```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0  9980   478
##          1 11949  1030
##
##                Accuracy : 0.4698
##                  95% CI : (0.4634, 0.4762)
##     No Information Rate : 0.9357
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0304
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.45511
##             Specificity : 0.68302
##          Pos Pred Value : 0.95429
##          Neg Pred Value : 0.07936
##              Prevalence : 0.93566
##          Detection Rate : 0.42582
##    Detection Prevalence : 0.44622
##       Balanced Accuracy : 0.56906
##
##        'Positive' Class : 0
```

##

Lift Chart




```
## [1] 1.432789
```

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.

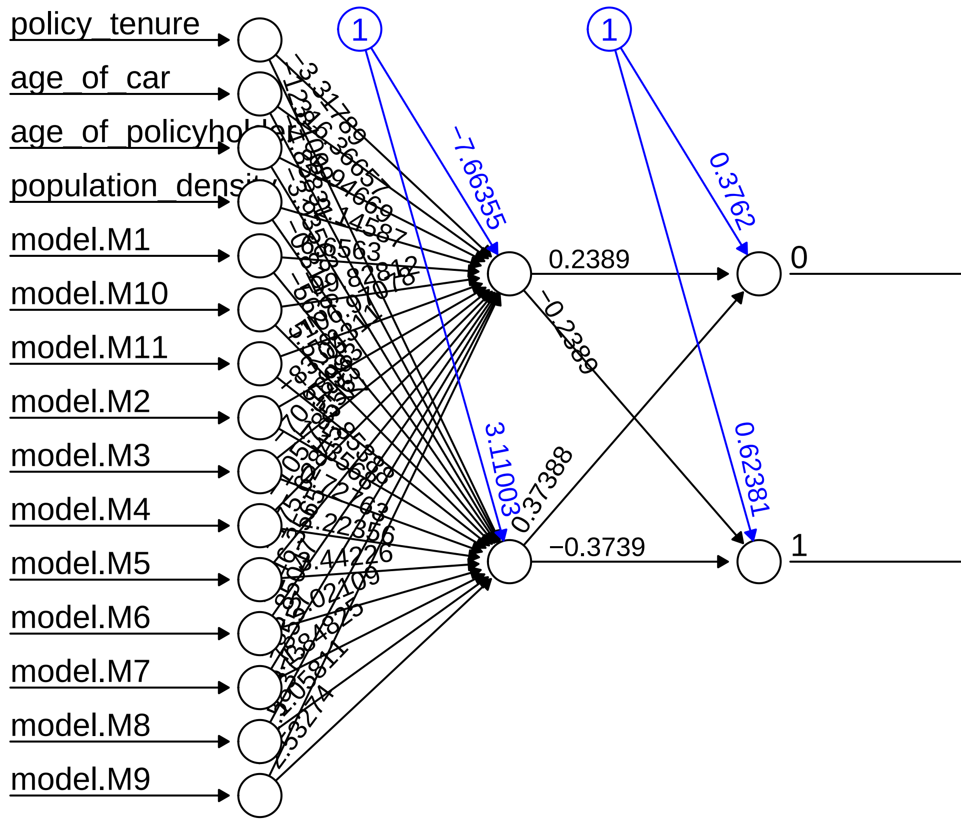
Values from variables go into the input layer. Then we initialize a very small and random weight in relation with our imputed values to start training the model. The resulting value is a node in the hidden layer. Weights are then updated again for many times in order to find an optimum value which minimize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

Data transformation if needed

```
##  policy_tenure      age_of_car      age_of_policyholder population_density
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.0000      Min.   :0.0000
##  1st Qu.:0.2093   1st Qu.:0.05714   1st Qu.:0.1304      1st Qu.:0.5508
##  Median :0.5319   Median :0.17143   Median :0.2609      Median :0.6165
##  Mean   :0.5234   Mean   :0.18959   Mean   :0.2796      Mean   :0.6760
##  3rd Qu.:0.8441   3rd Qu.:0.31429   3rd Qu.:0.3913      3rd Qu.:0.8192
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.0000      Max.   :1.0000
##     model.M1         model.M10          model.M11           model.M2
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.000000   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.00000
##  Median :0.0000   Median :0.00000   Median :0.000000   Median :0.00000
##  Mean   :0.2585   Mean   :0.02009   Mean   :0.004687   Mean   :0.02187
##  3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:0.00000
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.000000   Max.   :1.00000
##     model.M3          model.M4          model.M5          model.M6
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.00000   Median :0.0000   Median :0.00000   Median :0.0000
##  Mean   :0.03527   Mean   :0.2467   Mean   :0.02813   Mean   :0.2333
##  3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.0000
##  Max.   :1.00000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000
##     model.M7          model.M8           model.M9       is_claim
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   0:2240
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1:2240
##  Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.05201   Mean   :0.06161   Mean   :0.03795
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
```

Method application

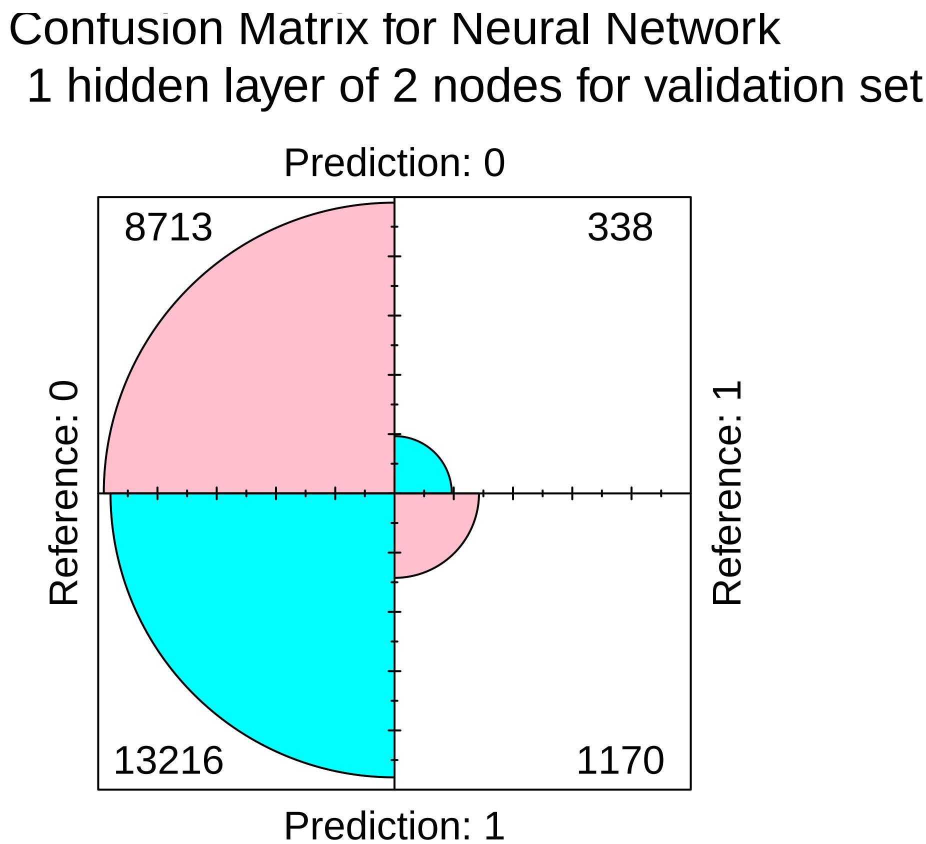
Common practice is to use 1 hidden layer for fitting neural networks. We are first going to try the algorithm with two nodes. Then we are going to add a third node in order to compare both models.



1 hidden layer and 2 nodes

1 hidden layer and 3 nodes

Performance evaluation



Conclusion on method performance

Logistic Regression

Method Description

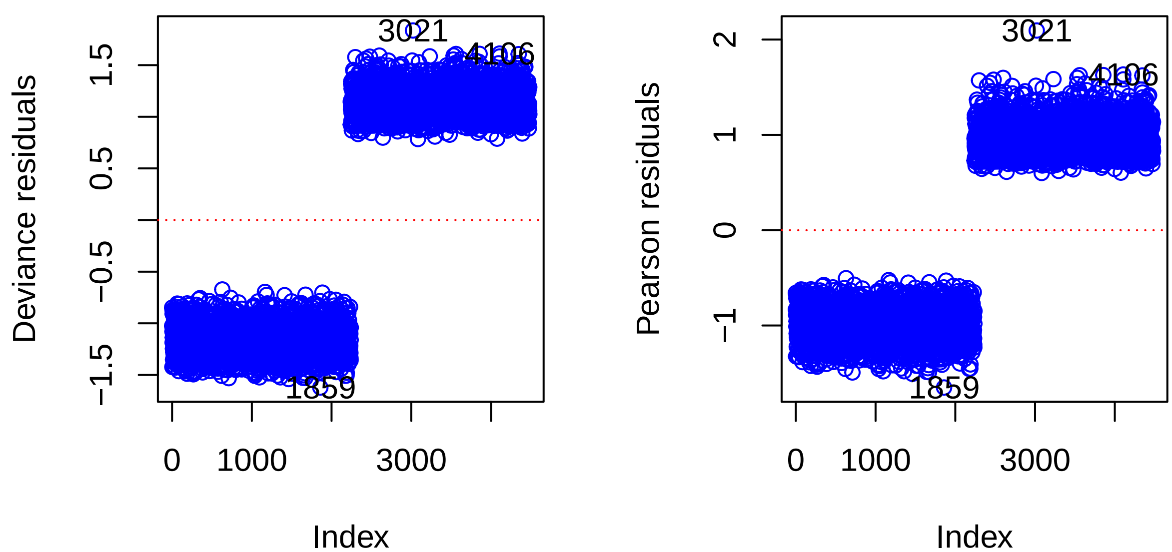
The logistic regression is a model that is also suited for binary response variables. It is a modeling technique used in statistics to predict the probability of an event occurring based on different independent variables. Building the model with a logit link function as it is usually the best suited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

Method Application

```
##
## 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.



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.

```
## 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.

```
## 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
```

```
##
## 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.

```
## Start:  AIC=6087.65
## is_claim ~ policy_tenure + age_of_car + age_of_policyholder +
##     population_density + model
```

```
##
## 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.

```
## 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
```

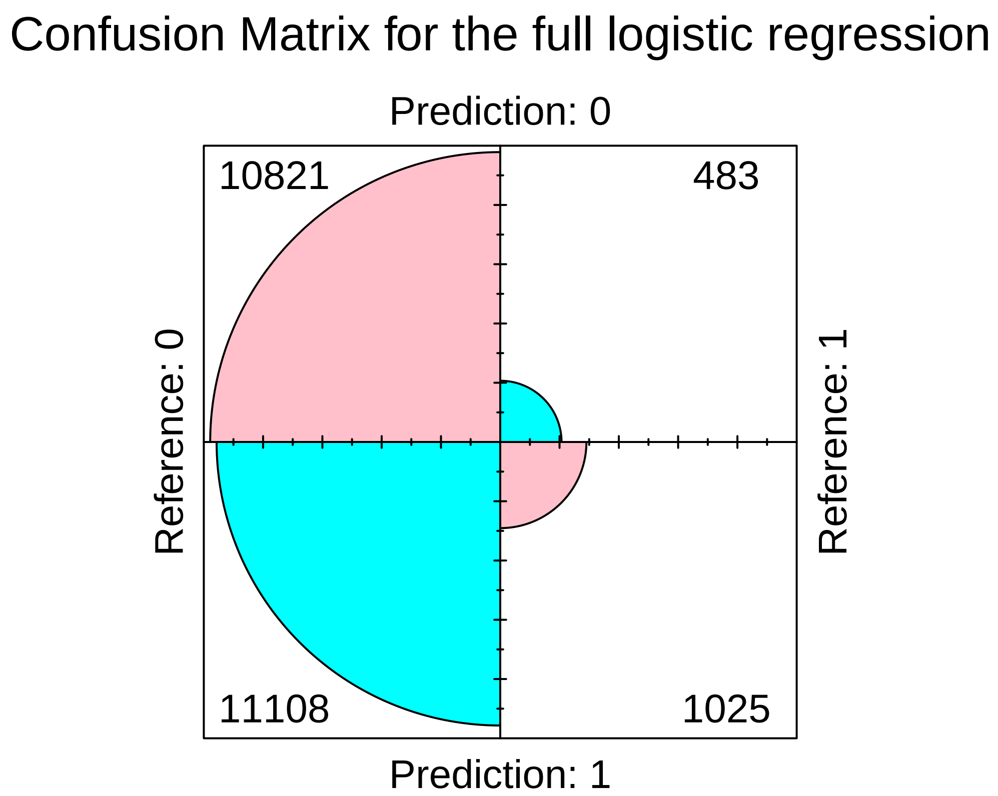
```
##
## Step:  AIC=6079.87
## is_claim ~ policy_tenure + age_of_car + population_density
##
##                      Df Deviance    AIC
## - population_density   1   6073.2 6079.2
## <none>                     6071.9 6079.9
## + age_of_policyholder  1   6070.8 6080.8
## + model               10   6058.4 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
```

```
##
## 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
```

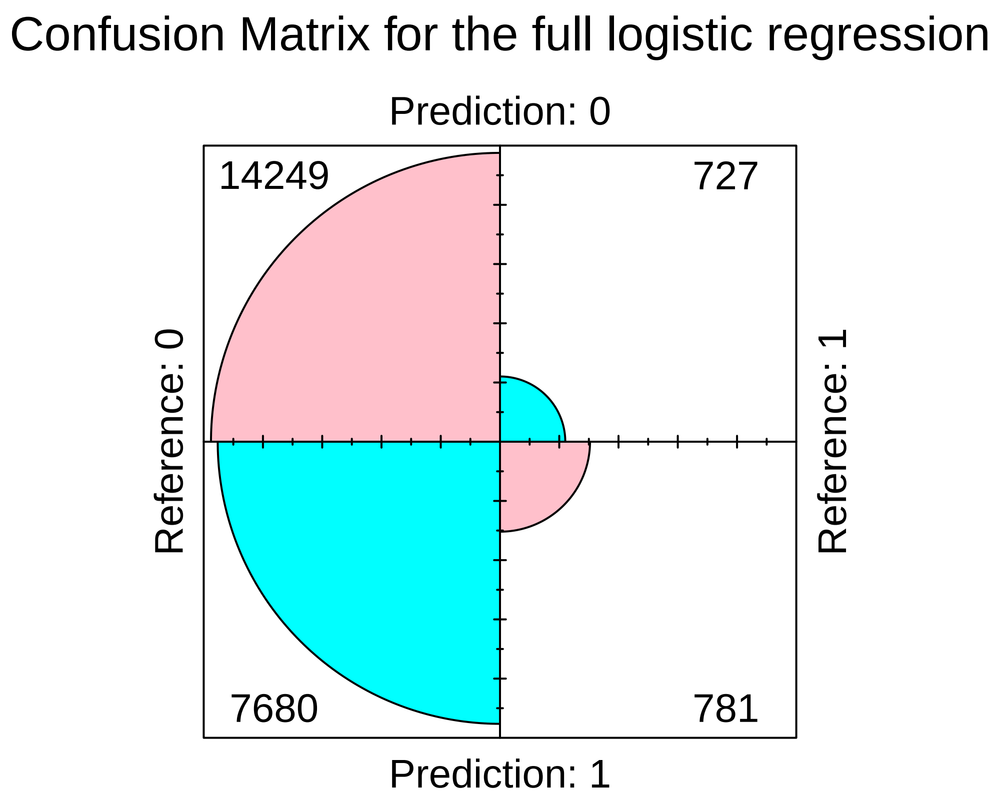
The selection with forward and backward at the same time results in the same variables as the backward selection. Every coefficient is also significant. The AIC of the reduced model is marginally better.

Performance Evaluation

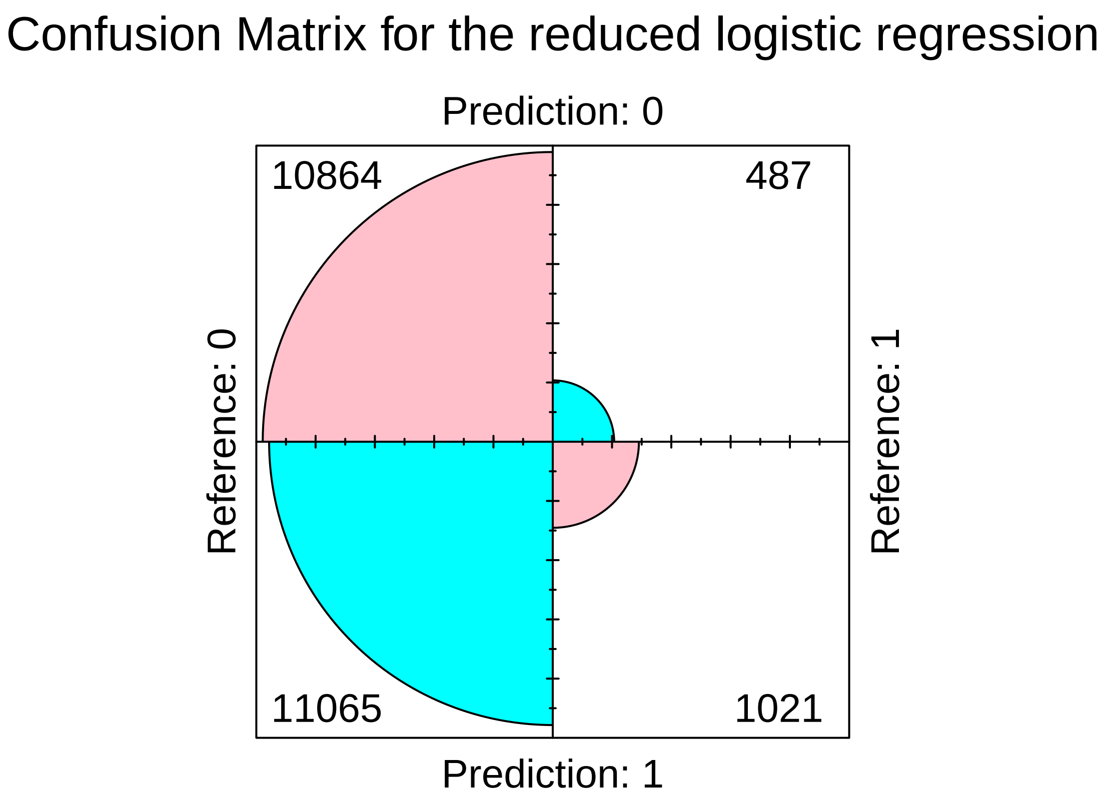
Confusion Matrix



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.



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.



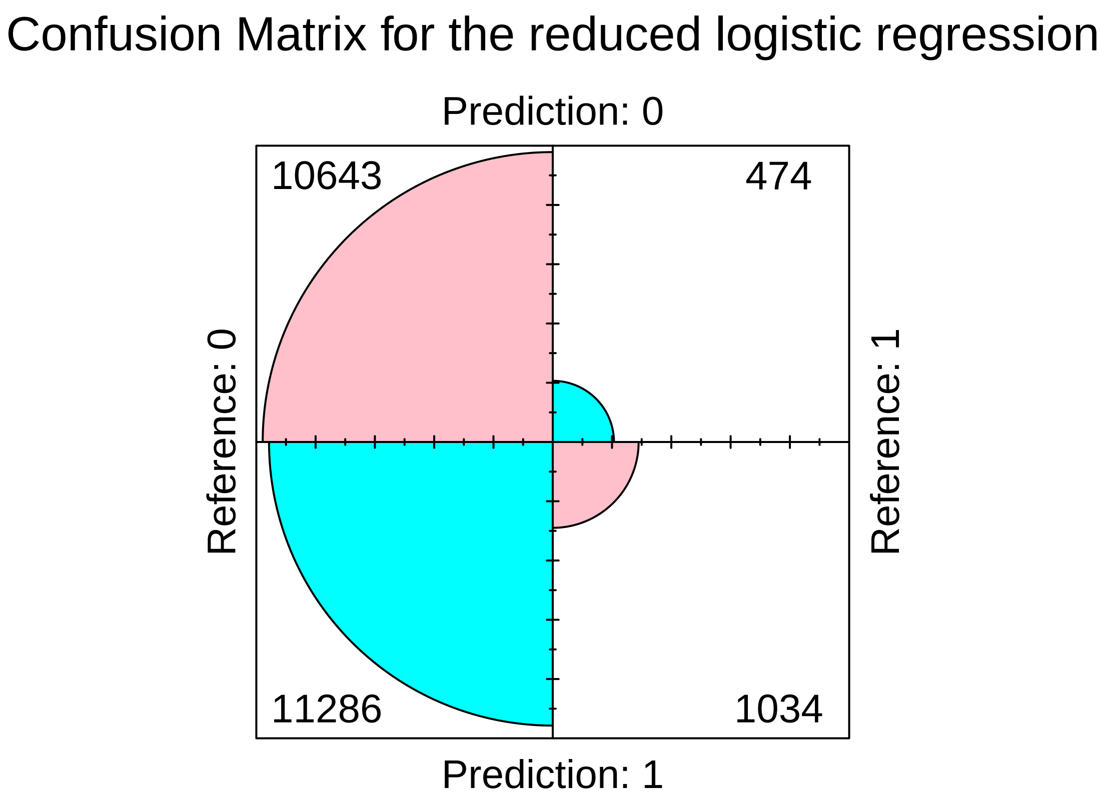
With the model with a reduced number of variables, we seem to get a little bit more false positives true positive: more is_claim= 1 predictions. The prediction performance is very similar, but one could argue that this reduced model performs marginally better since it reduces the false positives more 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
## 0.5071042
```

```
## Sensitivity
##   0.6770557
```

```
## Specificity
##    0.495417
```

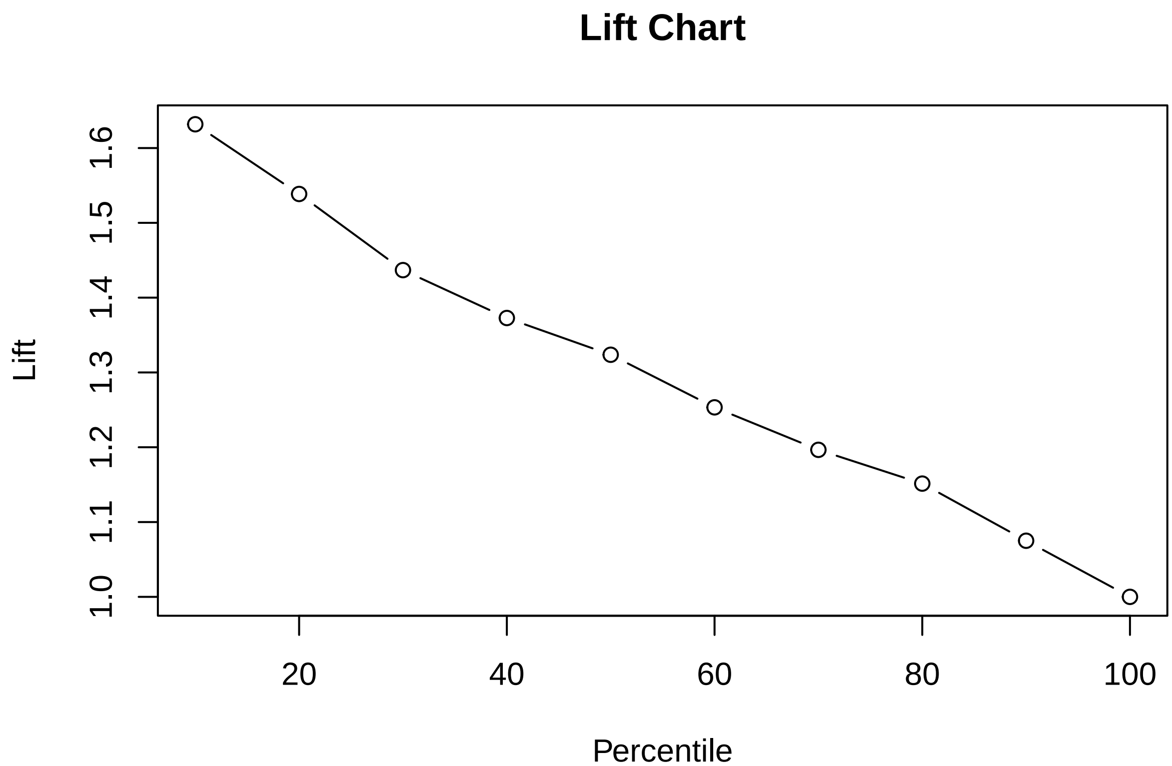
The accuracy of 51.8% reflects the high number of false positives and negatives.




```
##  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




```
## [1] 1.631787
```

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

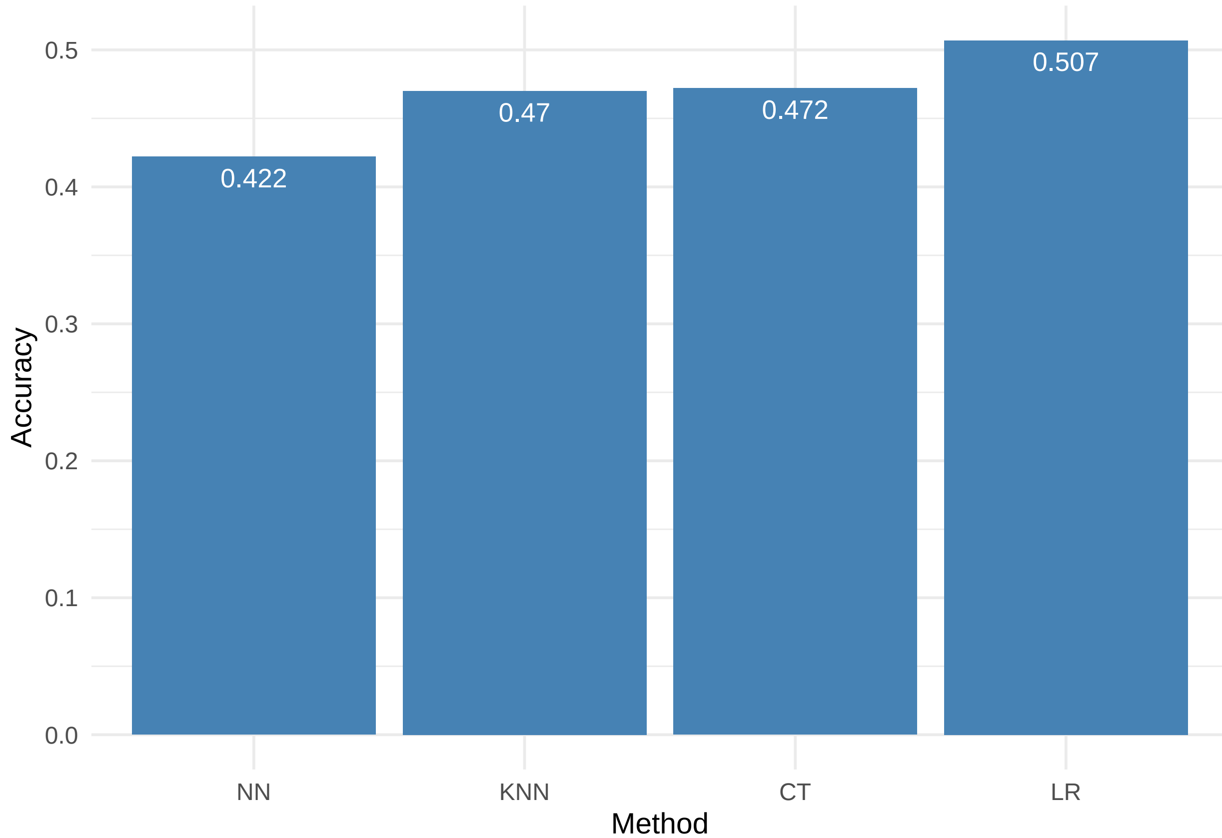
Conclusion on method performance

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%.

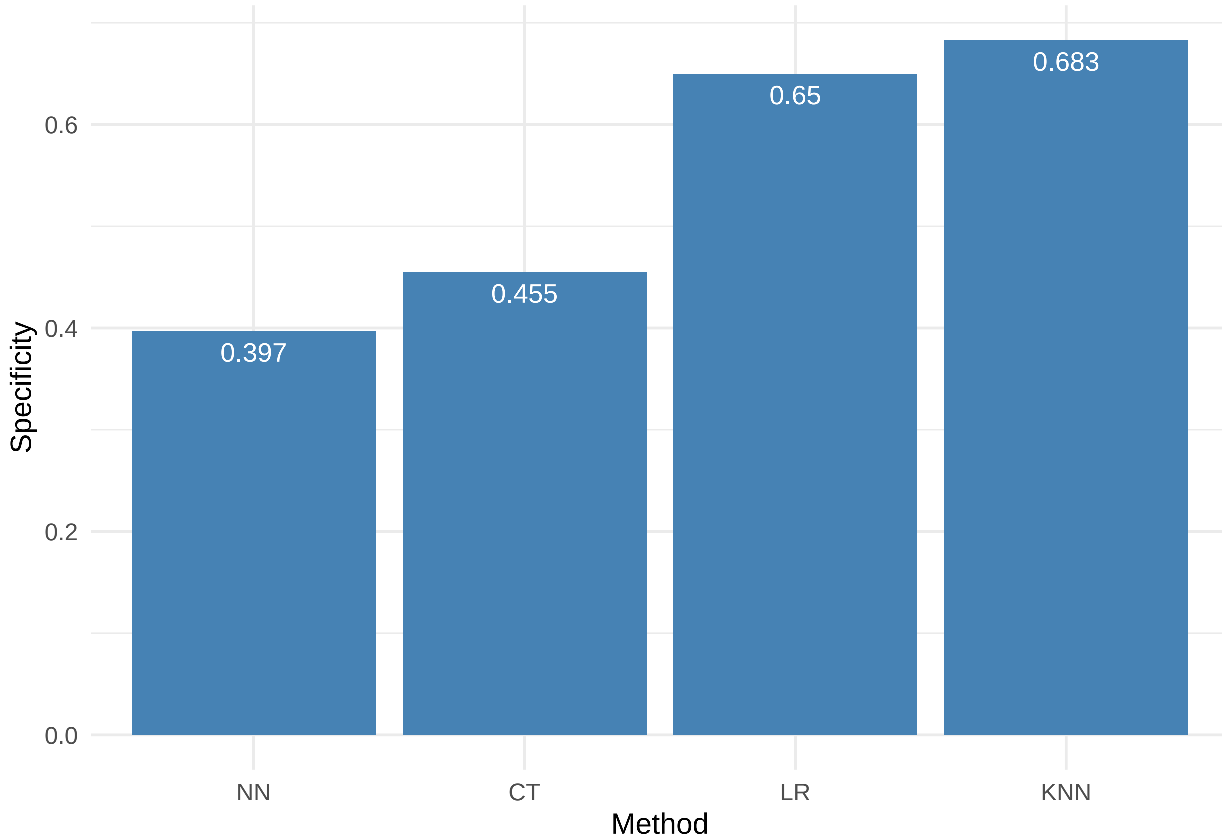
Ensemble

All methods comparison

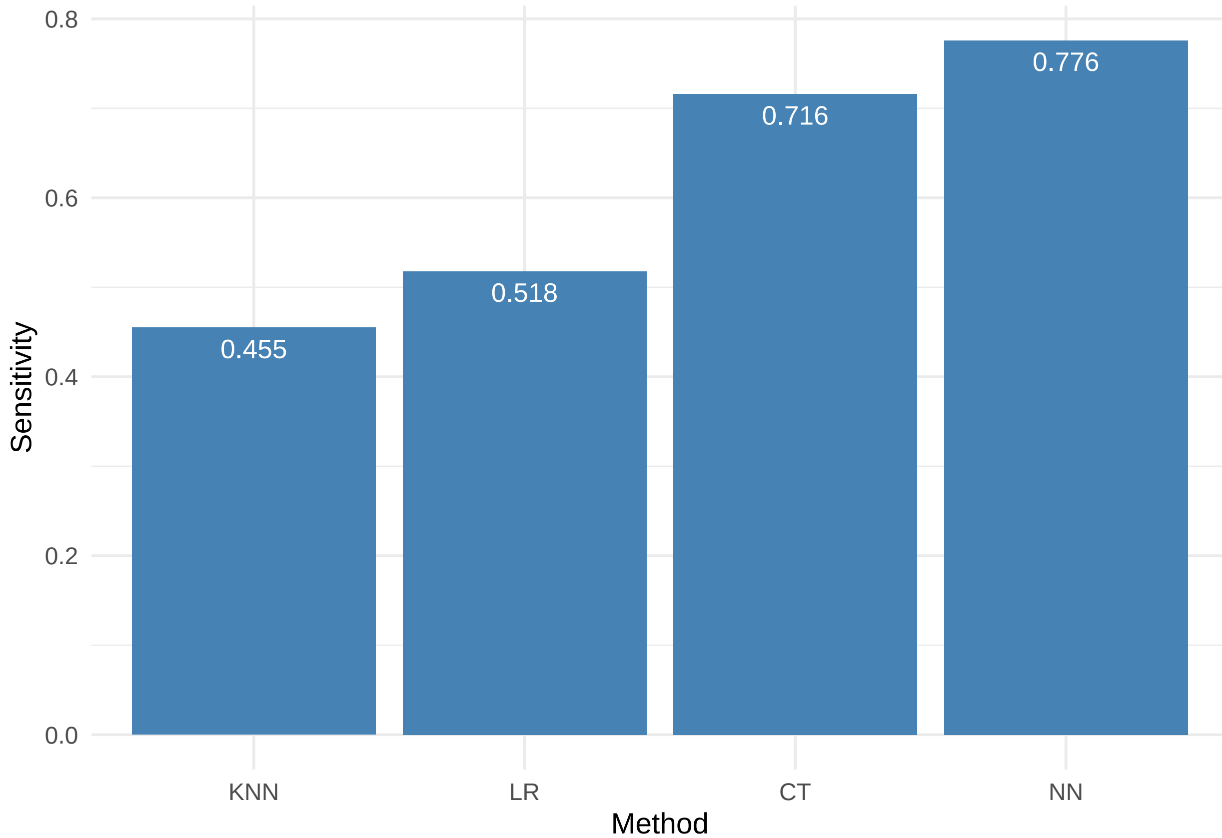
```
## No id variables; using all as measure variables
```




```
## No id variables; using all as measure variables
```




```
## No id variables; using all as measure variables
```



Conclusions