Project

Creating Value Through Data Mining

Malou-Tinette Kouango & Veronica Rowe

2021-12-13

NOTE:

- Use LDA on categorical variables only?
- ...

Project- Stroke Prediction Dataset # Clearing the console

```
#clear all the console
list=ls()
rm(list=ls())
cat("\014")
```

```
ls()
```

```
## character(0)
```

#Reading the Stroke dataset

```
Stroke <- read.csv(file.choose())
```

Loading the relevant packages

```
library(ggplot2)
library(corrplot)
library(ggcorrplot)
library(imputeTS)
library(FNN)
library(dummies)
library(e1071)
library (caret)
library(reshape2)
library(dplyr)
library(class)
library(naniar)
require(ellipse)
library(cowplot)
library(rpart)
library(rpart.plot)
library(MASS)
library(DiscriMiner)
library(gains)
library(pROC)
```

```


# Types of variables:
str(Stroke)
```

EDA- Explortory Data Analysis

```
## 'data.frame':    5110 obs. of  12 variables:
##  $ id               : int  9046 51676 31112 60182 1665 56669 53882 10434 27419 60491 ...
##  $ gender           : chr  "Male" "Female" "Male" "Female" ...
##  $ age              : num  67 61 80 49 79 81 74 69 59 78 ...
##  $ hypertension     : int  0 0 0 0 1 0 1 0 0 0 ...
##  $ heart_disease    : int  1 0 1 0 0 0 1 0 0 0 ...
##  $ ever_married     : chr  "Yes" "Yes" "Yes" "Yes" ...
##  $ work_type        : chr  "Private" "Self-employed" "Private" "Private" ...
##  $ Residence_type   : chr  "Urban" "Rural" "Rural" "Urban" ...
##  $ avg_glucose_level: num  229 202 106 171 174 ...
##  $ bmi              : chr  "36.6" "N/A" "32.5" "34.4" ...
##  $ smoking_status   : chr  "formerly smoked" "never smoked" "never smoked" "smokes" ...
##  $ stroke           : int  1 1 1 1 1 1 1 1 1 1 ...
```

We have 12 variables in the Stroke dataset, each having different types. Our outcome variable "Stroke" is a binary outcome, taking value 1 if the person had a stroke and 0 if they haven't had a stroke. => change it as factor - Bmi is seen as a character because of the N/A => we must change N/A strings to real NA in R, and set the variable as numeric to be able to visualize it in our exploratory graphs. - The variables gender, ever_married, work_type, Residence_type and smoking status are seen as character, when they are categorical => we must change them to factor for better data visualization. - The variable id (= id of the patient in the hospital) is irrelevant to our analysis => we remove it

```
# Changing N/A characters in bmi to real NA:
Stroke = replace_with_na(Stroke, replace = list(bmi = "N/A")) # bmi has NA now
# Change bmi from character to numerical variable:
Stroke$bmi <- as.numeric(Stroke$bmi)
Stroke$stroke <- as.factor(Stroke$stroke)

# Change character variables as categorical (factor)
Stroke[,c(2,6:8,11)] <- lapply(Stroke[,c(2,6:8,11)], as.factor)

# Remove id row
Stroke = Stroke[,-1]
str(Stroke)
```

```
## 'data.frame':    5110 obs. of  11 variables:
##  $ gender           : Factor w/ 3 levels "Female","Male",..: 2 1 2 1 1 2 2 1 1 1 ...
##  $ age              : num  67 61 80 49 79 81 74 69 59 78 ...
##  $ hypertension     : int  0 0 0 0 1 0 1 0 0 0 ...
##  $ heart_disease    : int  1 0 1 0 0 0 1 0 0 0 ...
##  $ ever_married     : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 1 2 2 ...
##  $ work_type        : Factor w/ 5 levels "children","Govt_job",..: 4 5 4 4 5 4 4 4 4 4 ...
##  $ Residence_type   : Factor w/ 2 levels "Rural","Urban": 2 1 1 2 1 2 1 2 1 2 ...
##  $ avg_glucose_level: num  229 202 106 171 174 ...
##  $ bmi              : num  36.6 NA 32.5 34.4 24 29 27.4 22.8 NA 24.2 ...
##  $ smoking_status   : Factor w/ 4 levels "formerly smoked",..: 1 2 2 3 2 1 2 2 4 4 ...
##  $ stroke           : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

```
# Show how many different values there is for each variable
sapply(Stroke, function(x) length(unique(x)))
```

```
##            gender               age      hypertension     heart_disease
##                 3               104                 2                 2
##      ever_married         work_type    Residence_type avg_glucose_level
##                 2                 5                 2              3979
##               bmi    smoking_status            stroke
##               419                 4                 2
```

```
summary(Stroke)
```

```
##     gender          age         hypertension     heart_disease     ever_married
##  Female:2994   Min.   : 0.08   Min.   :0.00000   Min.   :0.00000   No :1757
##  Male  :2115   1st Qu.:25.00   1st Qu.:0.00000   1st Qu.:0.00000   Yes:3353
##  Other :   1   Median :45.00   Median :0.00000   Median :0.00000
##                Mean   :43.23   Mean   :0.09746   Mean   :0.05401
##                3rd Qu.:61.00   3rd Qu.:0.00000   3rd Qu.:0.00000
##                Max.   :82.00   Max.   :1.00000   Max.   :1.00000
##
##          work_type    Residence_type avg_glucose_level      bmi
##  children     : 687   Rural:2514     Min.   : 55.12    Min.   :10.30
##  Govt_job     : 657   Urban:2596     1st Qu.: 77.25    1st Qu.:23.50
##  Never_worked :  22                  Median : 91.89    Median :28.10
##  Private      :2925                  Mean   :106.15    Mean   :28.89
##  Self-employed: 819                  3rd Qu.:114.09    3rd Qu.:33.10
##                                      Max.   :271.74    Max.   :97.60
##                                                        NA's   :201
##          smoking_status stroke
##  formerly smoked: 885   0:4861
##  never smoked   :1892   1: 249
##  smokes         : 789
##  Unknown        :1544
##
##
##
```

Using sapply function from TP1 to find out more about the variables. We realise that in gender, there is only one observation as categorised as "Other".

To simplify our analysis, we will later disregard the row with that observation, as it not significant enough to be taken into consideration (1/5110 = 0.01956947%) (row/observation number id = 56'156/ row = 3117). We will also disregard the ID column as it only represent the patient id in the hospital and thus has no impact on our outcome variable of interest Stroke.

Finding out information on our outcome variable

```
table(Stroke$stroke)
```

```
##
##    0    1
## 4861  249
```

```
ratio_0 = 4861/(4861+249)
ratio_1 = 1 - ratio_0
ratio_0
```

```
## [1] 0.951272
```

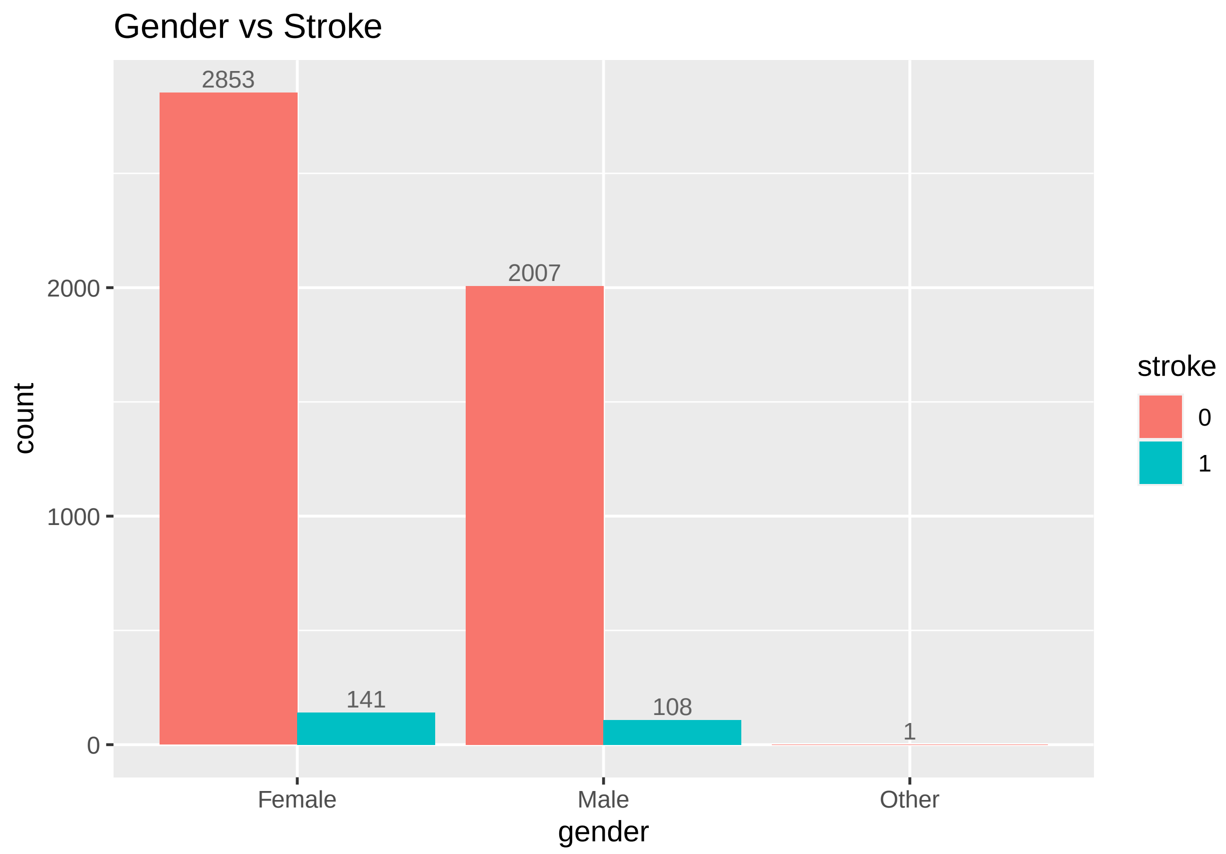
```
ratio_1
```

```
## [1] 0.04872798
```

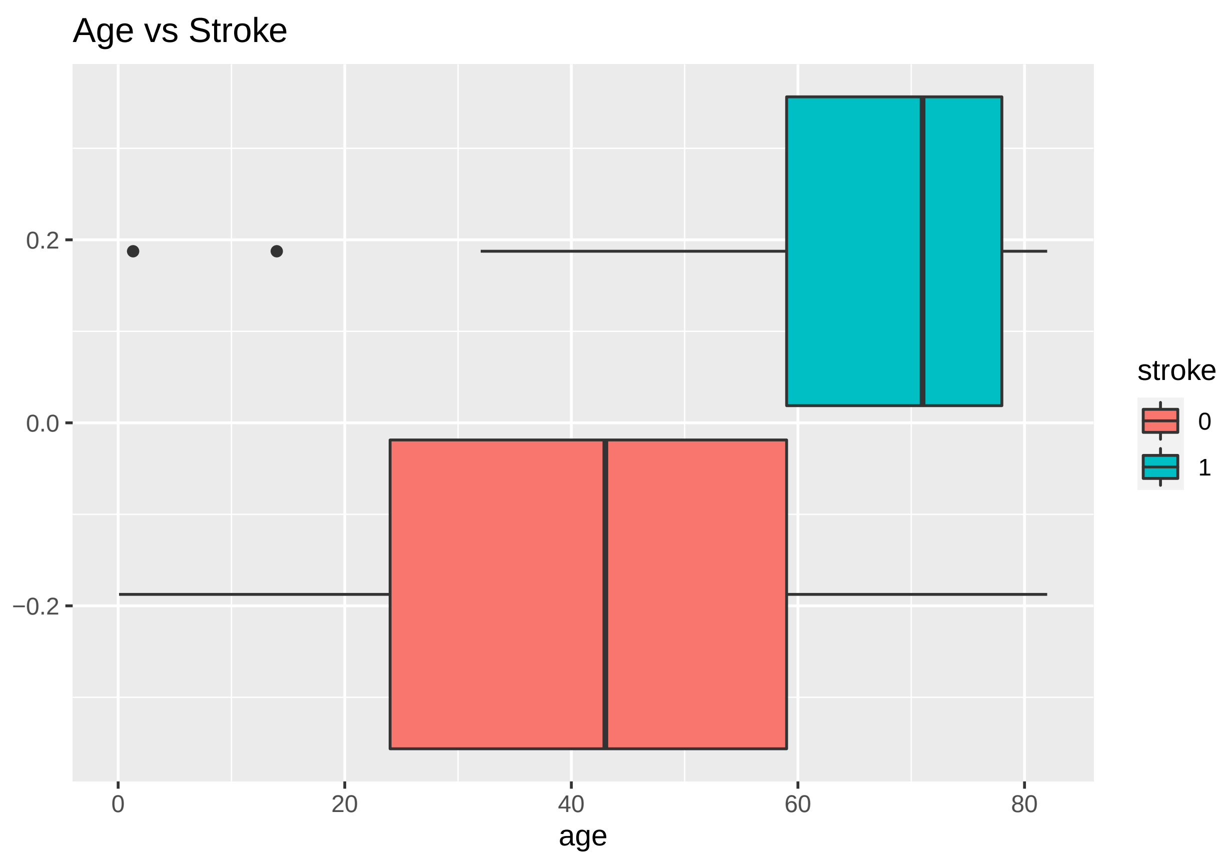
We see that 249 people had a stroke and 4861 did not. We have an unbalanced binary outcome, with 4861/(4861+249) = 95.1272% non-stroke and only 4.8728% of stroke occurrences.

Data Visualization

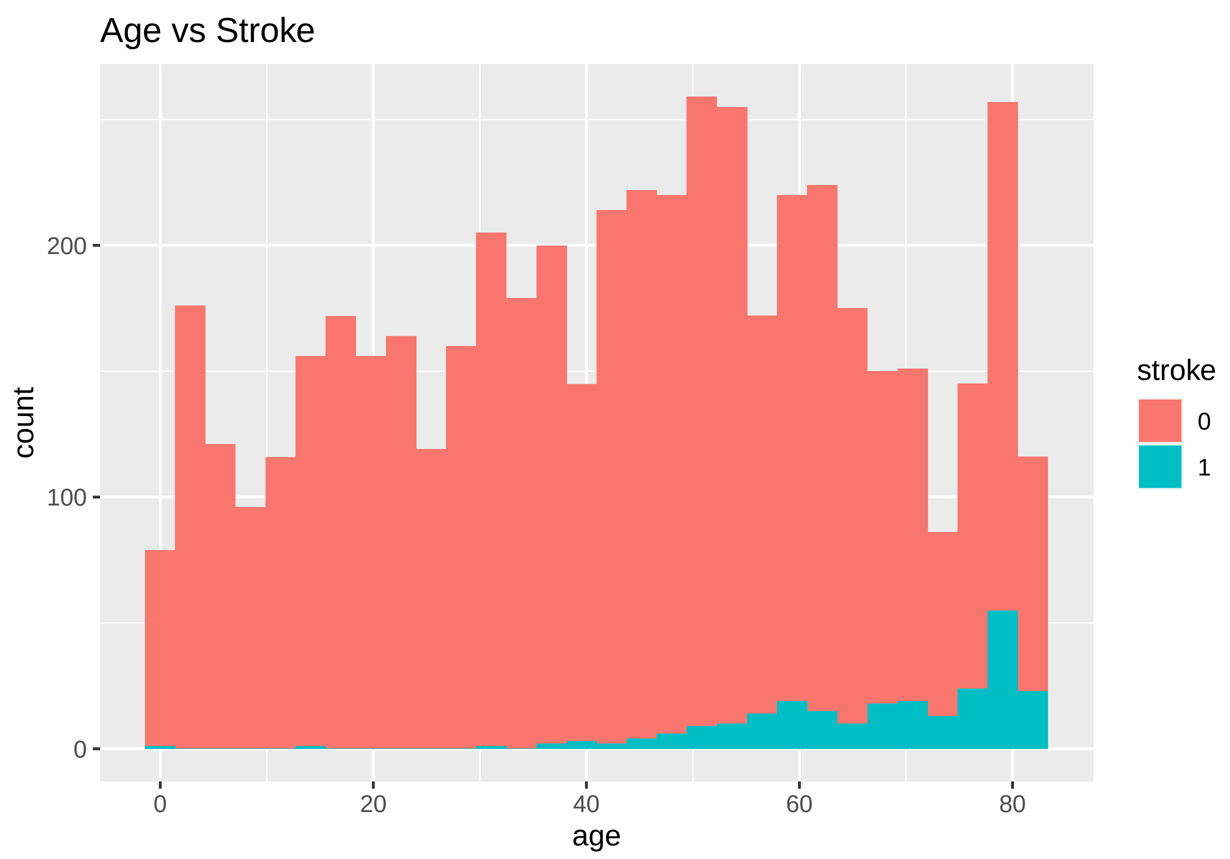
```r
# Visualize the data:
# gender:
ggplot(data = Stroke, aes(x = gender, fill = stroke)) + geom_bar(position="dodge") + ggtitle("Gender vs
```



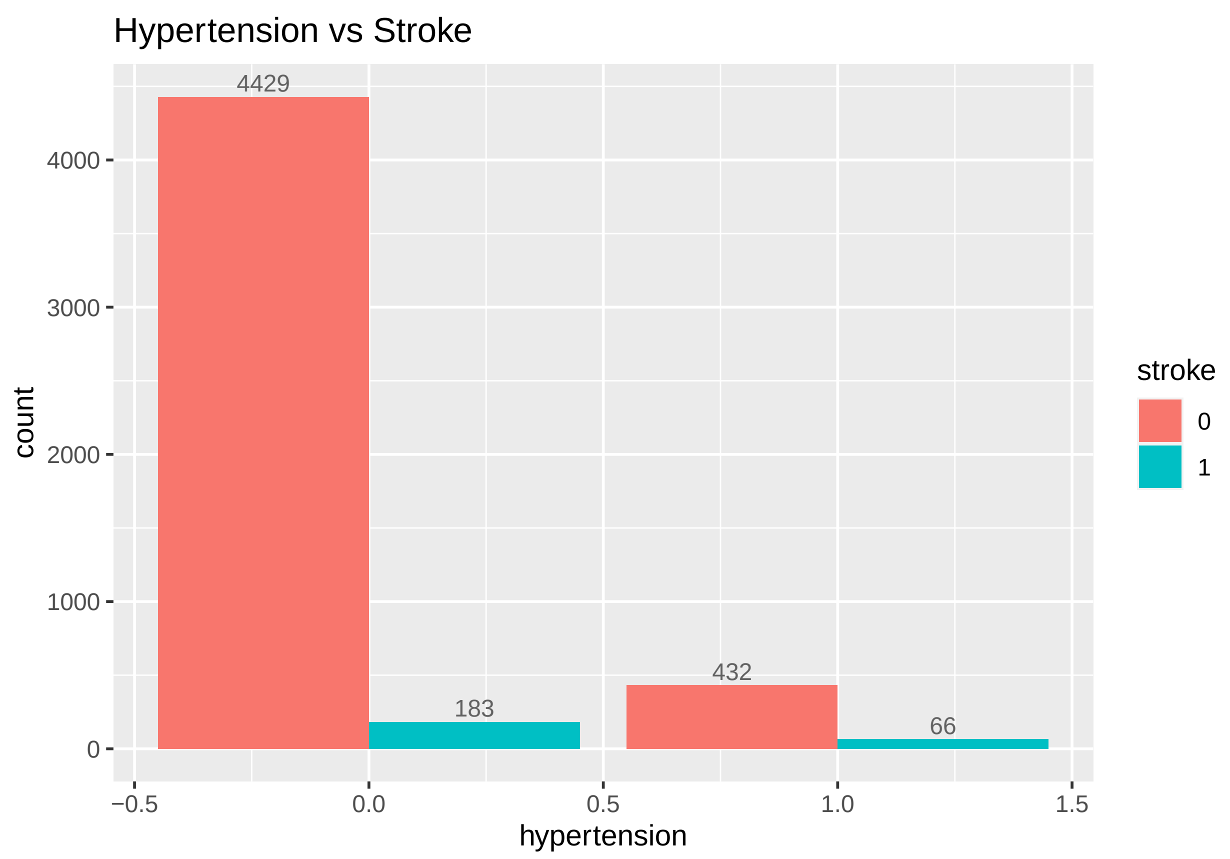
```r
# age:
ggplot(data = Stroke, aes(x = age, fill = stroke)) + geom_boxplot() + ggtitle("Age vs Stroke")
```



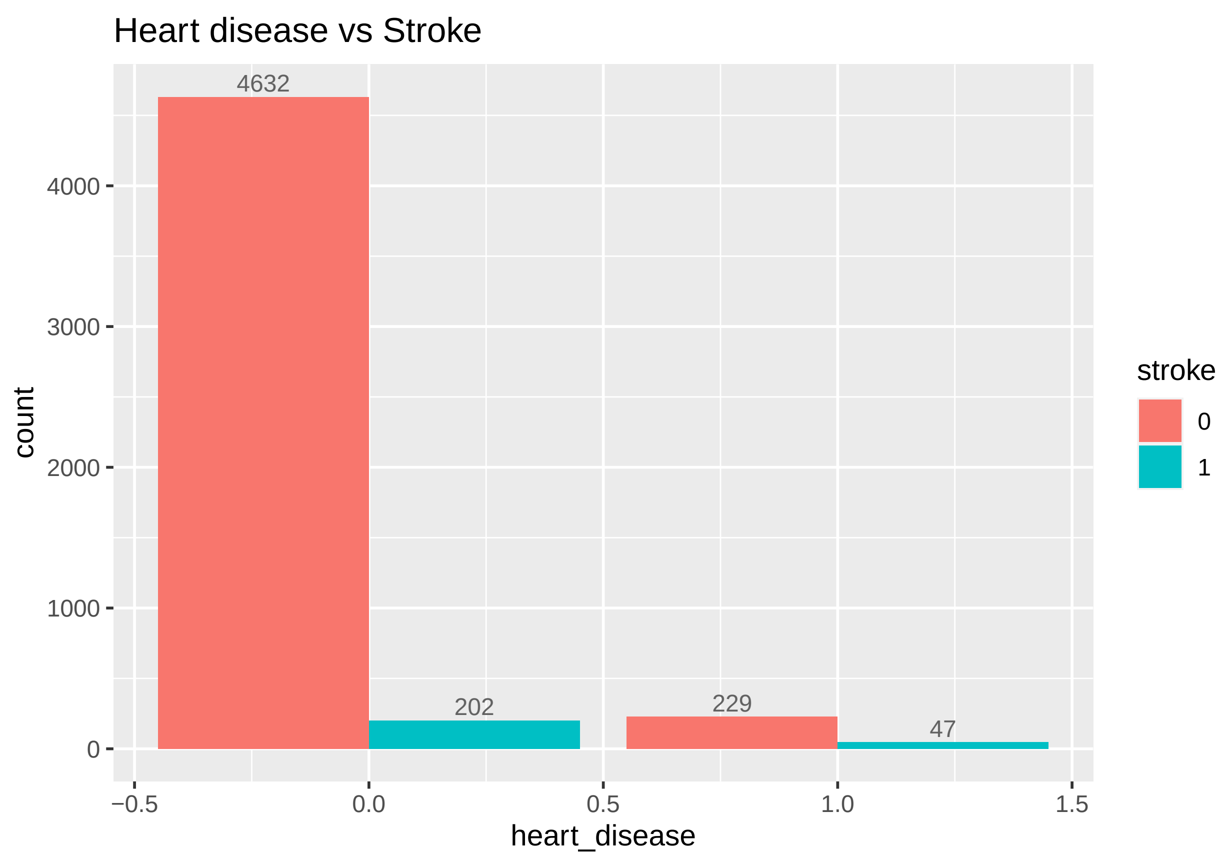
```
ggplot(data = Stroke, aes(x = age, fill = stroke)) + geom_histogram() + ggtitle("Age vs Stroke")
```



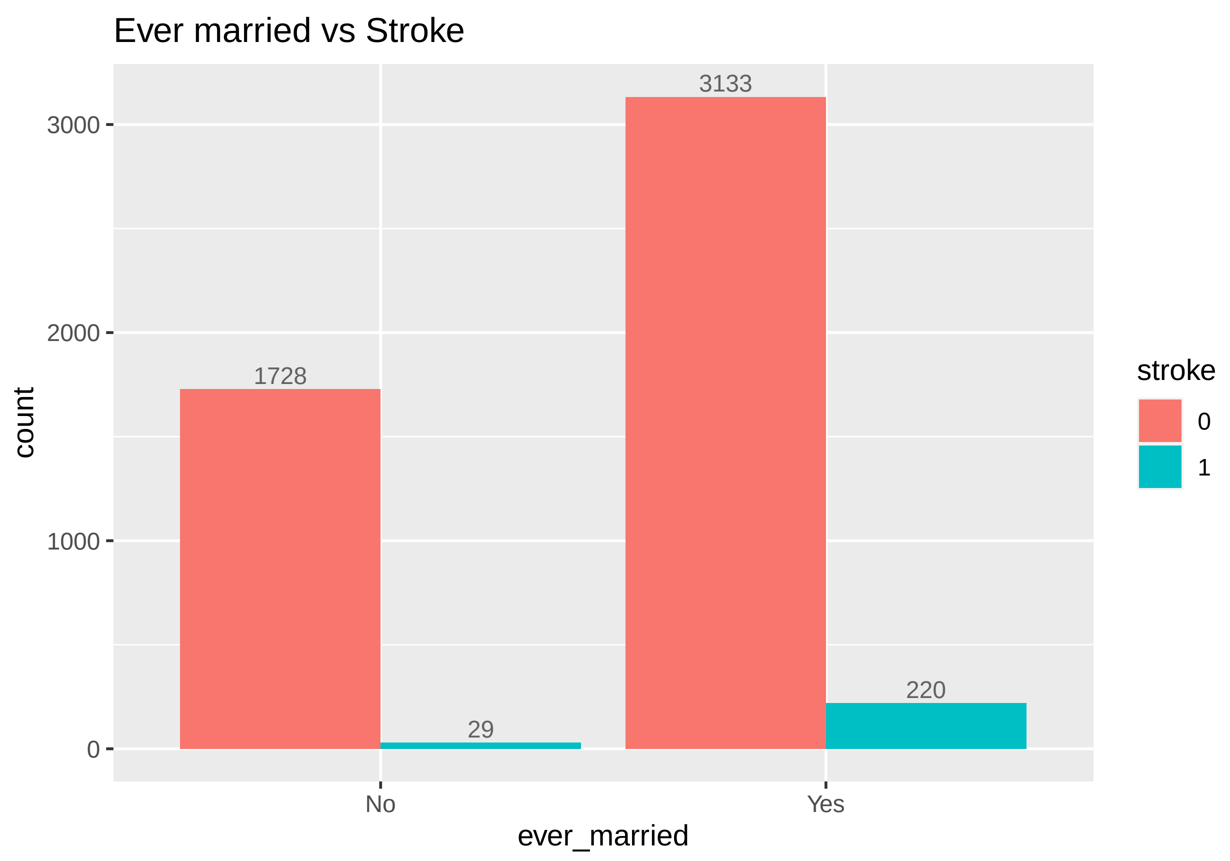
```
# hypertension:
ggplot(data = Stroke, aes(x = hypertension, fill = stroke)) + geom_bar(position="dodge") + ggtitle("Hyp
```



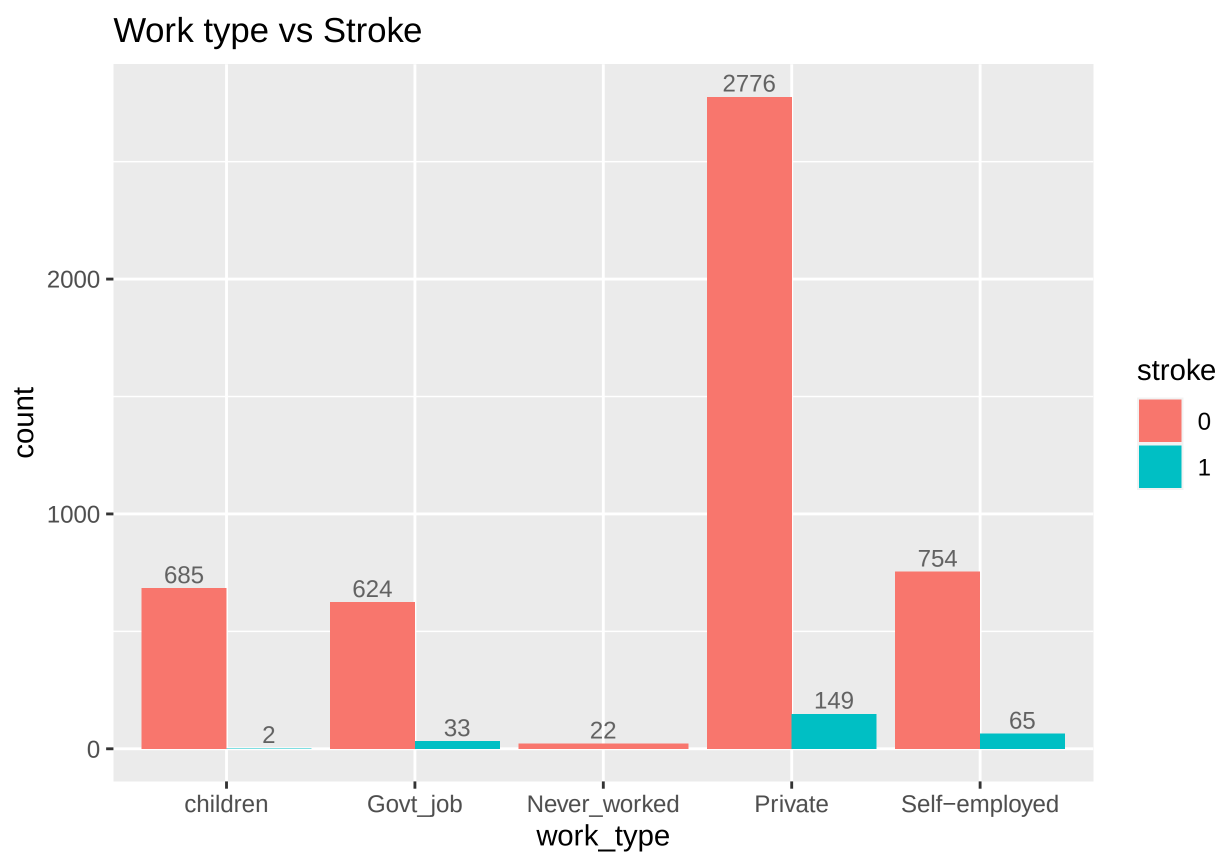
```
# heart disease:
ggplot(data = Stroke, aes(x = heart_disease, fill = stroke)) + geom_bar(position="dodge") + ggtitle("He
```



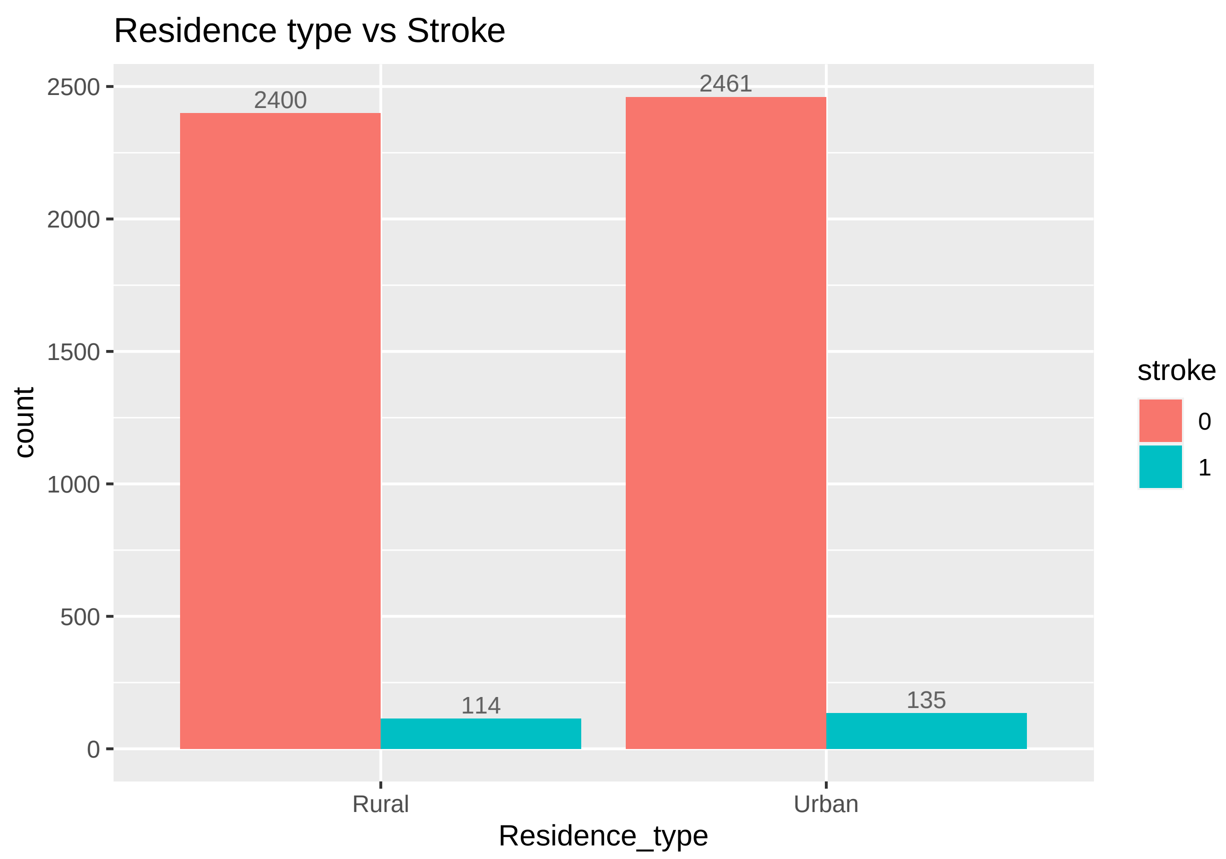
```
# ever married:
ggplot(data = Stroke, aes(x = ever_married, fill = stroke)) + geom_bar(position="dodge") + ggtitle("Eve
```



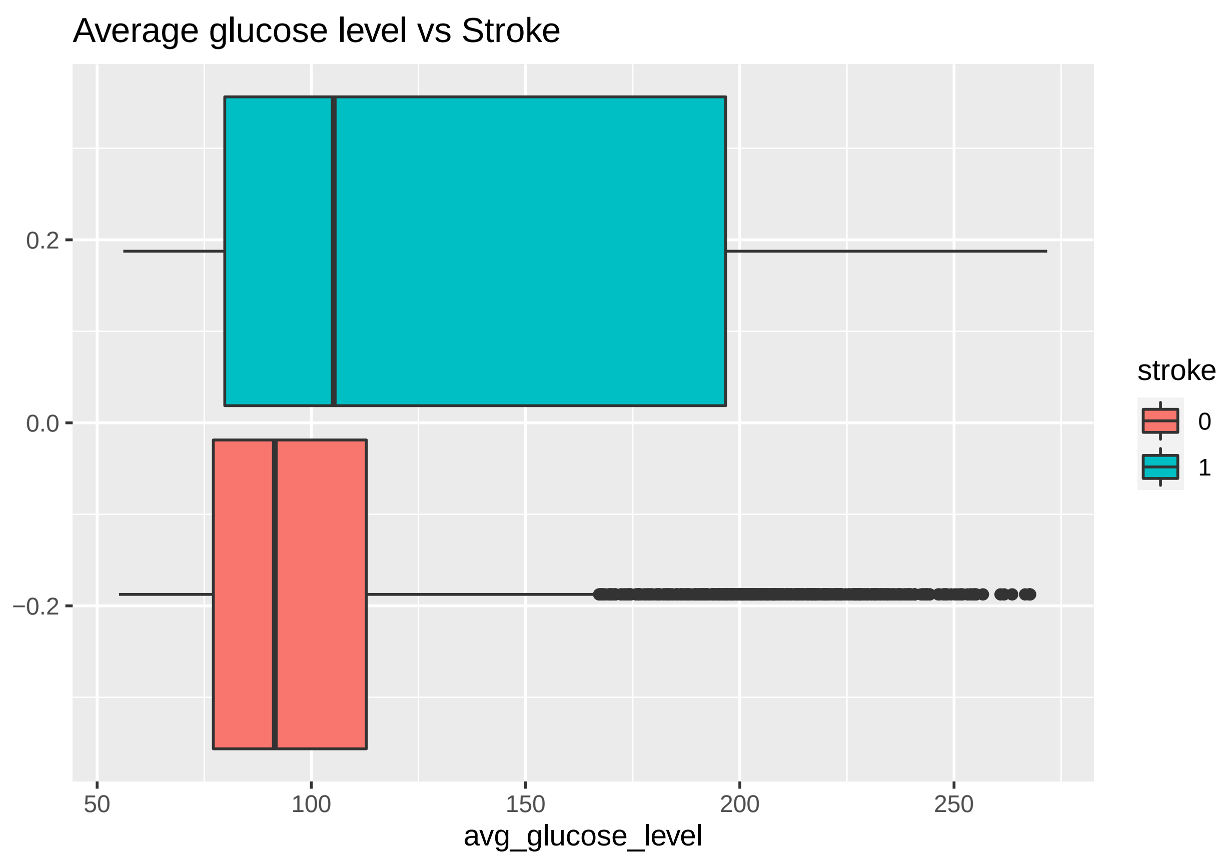
```
# work type:
ggplot(data = Stroke, aes(x = work_type, fill = stroke)) + geom_bar(position="dodge") + ggtitle("Work t
```



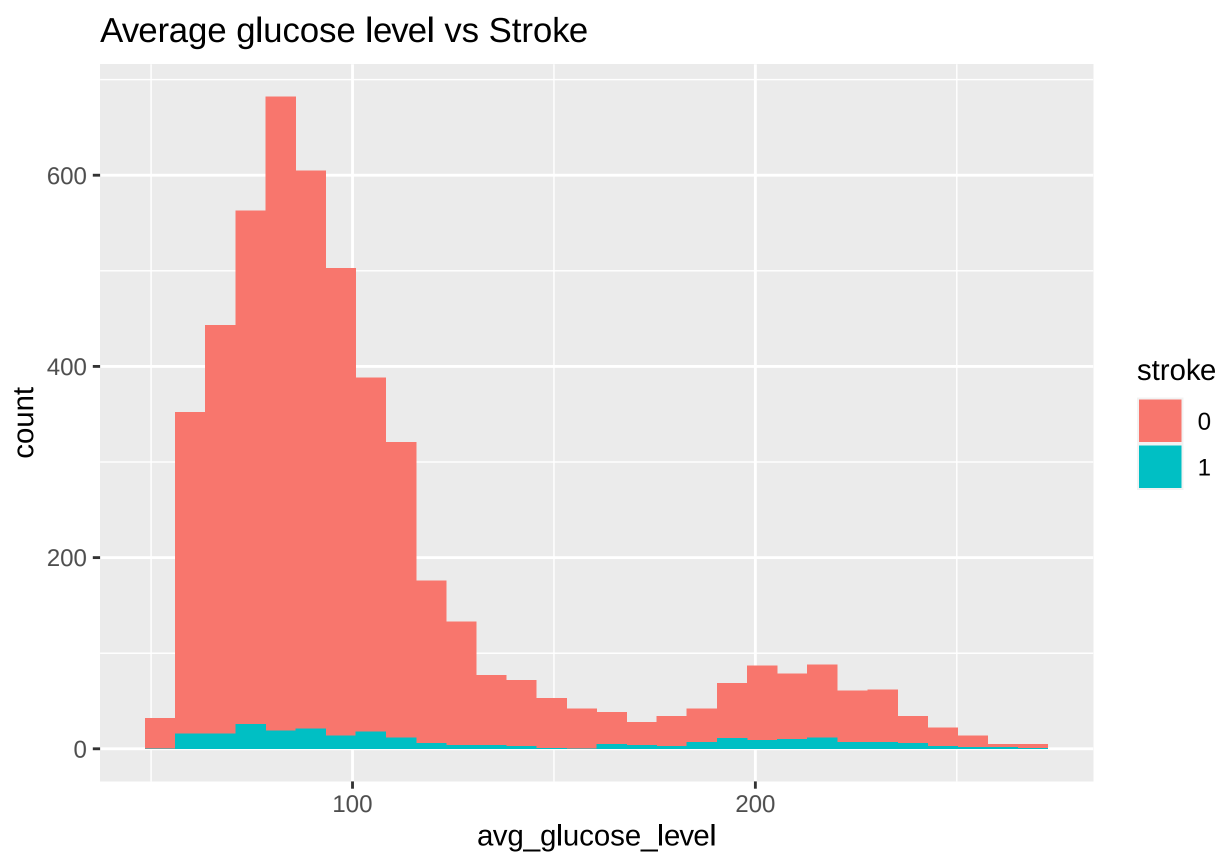
```
# Residence type:
ggplot(data = Stroke, aes(x = Residence_type, fill = stroke)) + geom_bar(position="dodge") + ggtitle("R
```



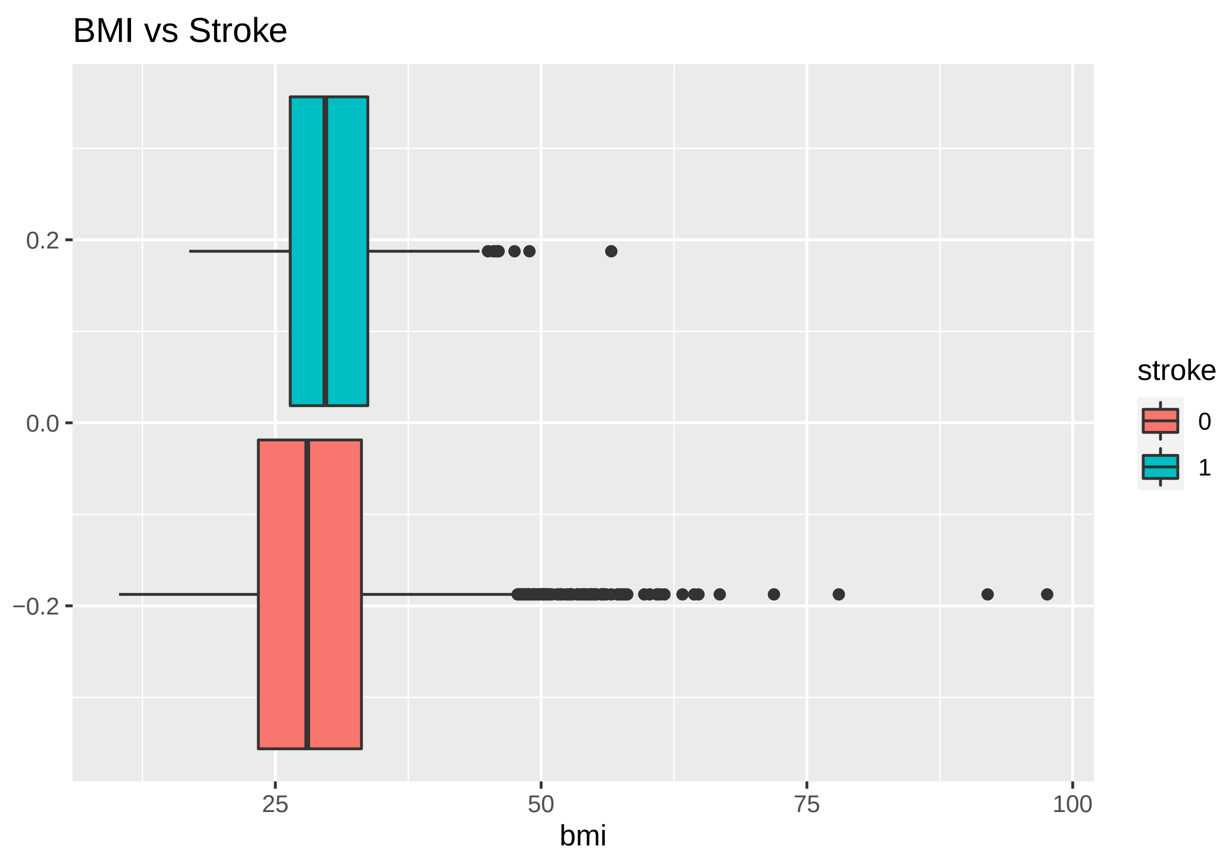
```r
# Average glucose level:
ggplot(data = Stroke, aes(x = avg_glucose_level, fill = stroke)) + geom_boxplot() + ggtitle("Average glu
```



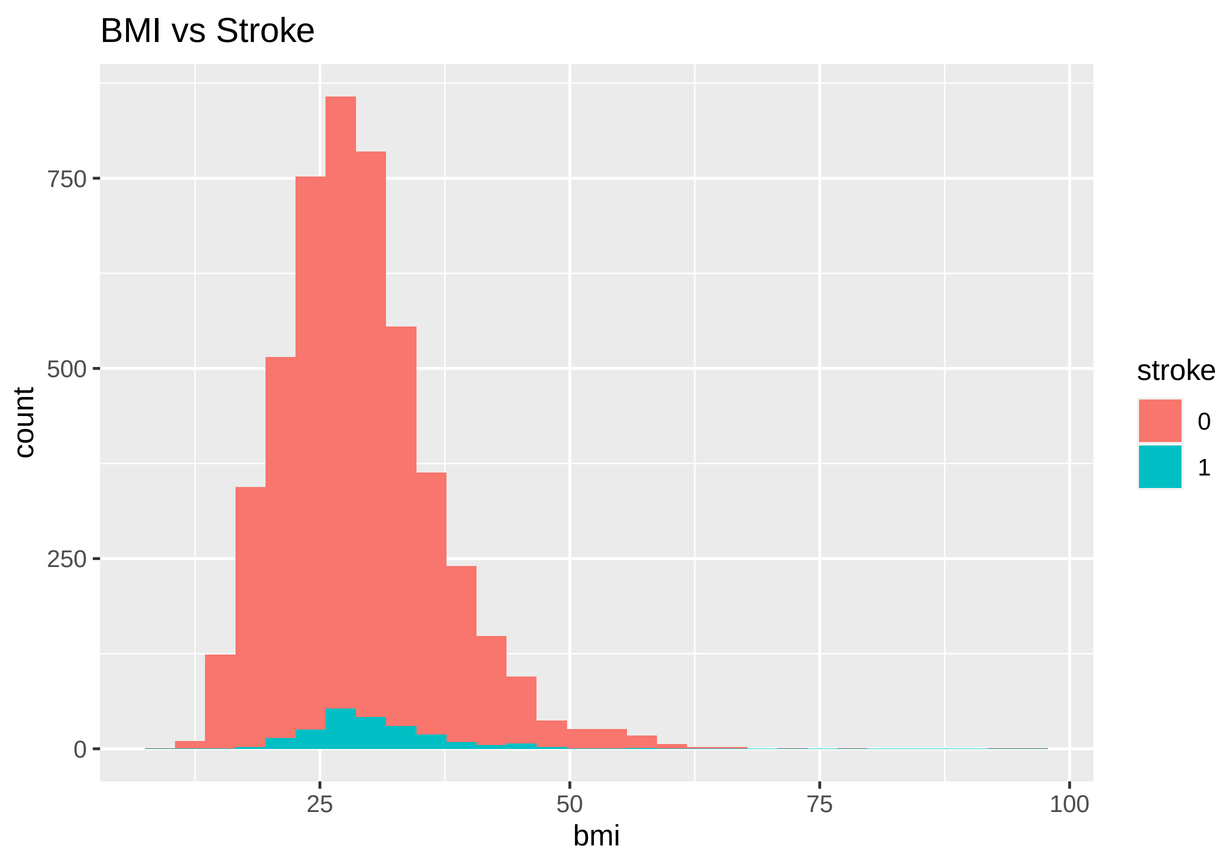
```
ggplot(data = Stroke, aes(x = avg_glucose_level, fill = stroke)) + geom_histogram() + ggtitle("Average g
```



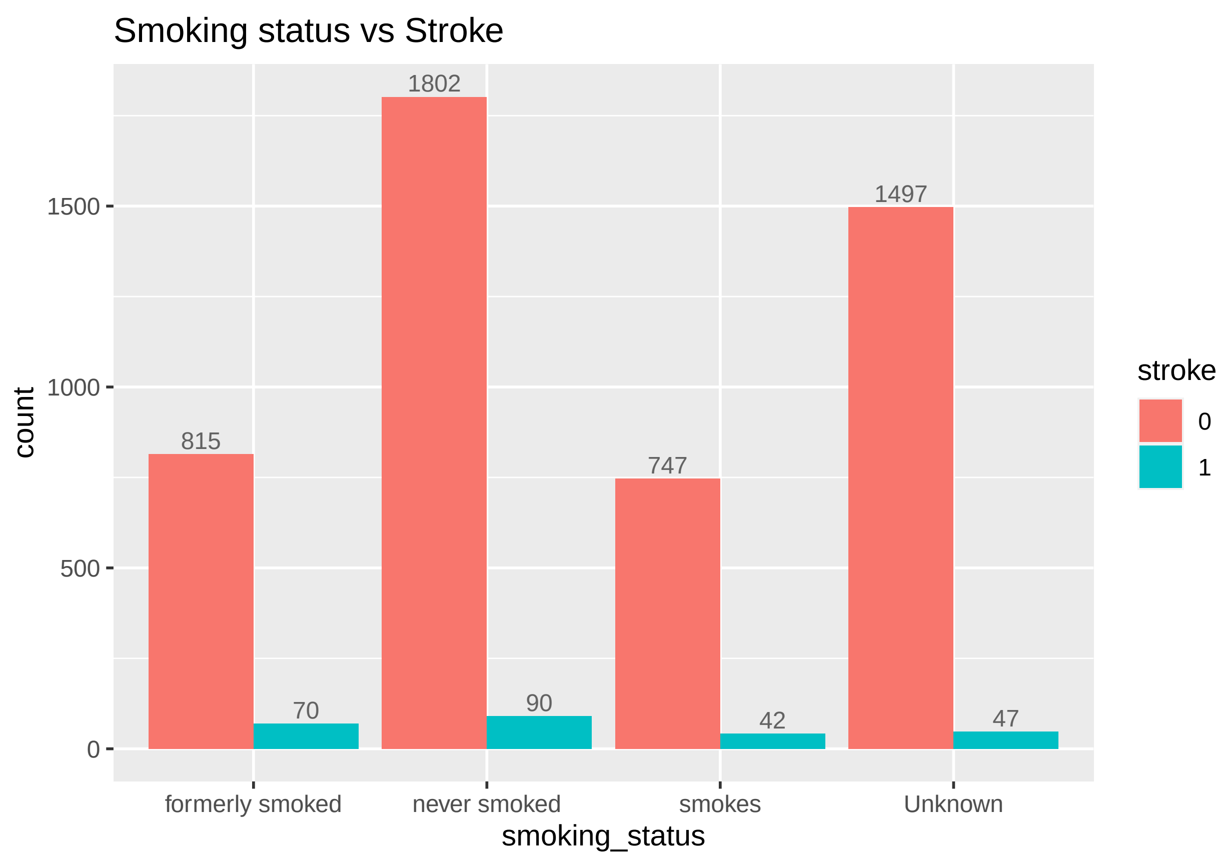
```
# BMI:
ggplot(data = Stroke, aes(x = bmi, fill = stroke)) + geom_boxplot() + ggtitle("BMI vs Stroke")
```



```r
ggplot(data = Stroke, aes(x = bmi, fill = stroke)) + geom_histogram() + ggtitle("BMI vs Stroke")
```



```
# Smoking status:
ggplot(data = Stroke, aes(x = smoking_status, fill = stroke)) + geom_bar(position="dodge") + ggtitle("Sm
```



In the histograms, we see that the average age of the patient is in his 40ties and the average glucose and bmi are right-skewed.

First of all, our data visualisation are our assumptions without having explored or analysed the significance of the explanatory variables. Moreover, the outcome variable stoke is unbalanced so we should be wary of these assumptions. We are visualising them & identifying potential explanations in order to have an idea before going further.

We see that the levels of males/females having strokes are about the same. However, this needs to be scaled as we also see that there are more women in the sample than men. It is safe to say that men have a higher chance of having a stroke (yet to be determined the effect/in which significance level).

From the boxplot of Age variable. It is undeniable that the older the person, the higher are their chances of having a stroke (blue boxplot more to the right) and an important difference between both means of both groups. This can be proved also on our histogram.

Hypertension seems to have an effect, again the scaling is problematic but in relative terms, a lot of patients who suffered a stroke had hypertension. The same for the heart disease but in a way lower importance (blue line in 1 column is less prevalent than in hypertension).

Marriage status, strokes seem to be more common in the people that were/are married. This is due to underlying assumptions, usually married people are older (so maybe it is correlated with age, hypertension, heart disease, that have a higher probability to be the case, denoted by a result of 1, for older people).

As for the working status, we see that private work type seems to accentuate the likelihood of a strike, however, taking into consideration, the number of observations of that category is way higher so it is logical that stroke probability is also higher. Drop o pas drop ?

Residence type is pretty evenly distributed and there seems to be a little difference between both categories, therefore, we have decided to drop this variable.

For the average glucose, We see that a higher level indicates a higher chance of getting a stroke. It is fair to assume that excessive sugar levels are not healthy. This comment is also valid for our dataset. This same idea is also in the BMI variable, where the difference is not so flagrant but again, medically BMI (even if it is contested) does indicate something about the patient's health status.

As for the smoking variable, the scaling is again problematic as our outcome variable is unbalanced, we will have to further investigate in order to see the actual impact it has on the likelihood of having a stroke.

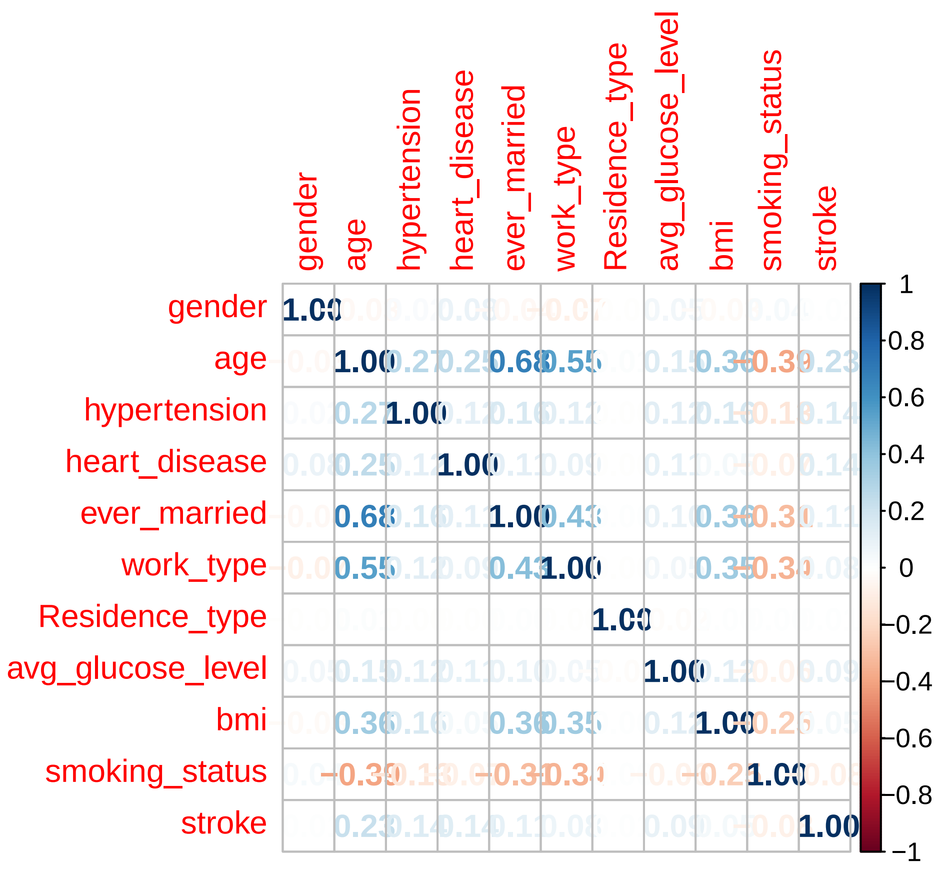
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Summary We see that all "medical terms" (hypertension, heart disease, glucose level, BMI, smoking, age) most of the time have an impact. It is commonly known that the general health status of a person will have an effect on the likelihood of having a stroke/heart attack and other illnesses. Usually, the weaker a person is the higher the chance of them getting sick. This assumption is pretty much satisfied in our data.

We see in the more "social terms" (gender, age, marriage status, work type, residence type). These variables have an impact on the age of the person (that is how they are directly linked to the medical terms & why they can be significant and should be taken into consideration before being potentially taken out by the model). Marriage status is the most clear example as older people usually have a higher chance of being married, the same goes for work type with the "children/never_worked categories connected to a young age" . It must be estimated and analysed in order to not assume spurious correlations. We need the data to back up any assumptions.

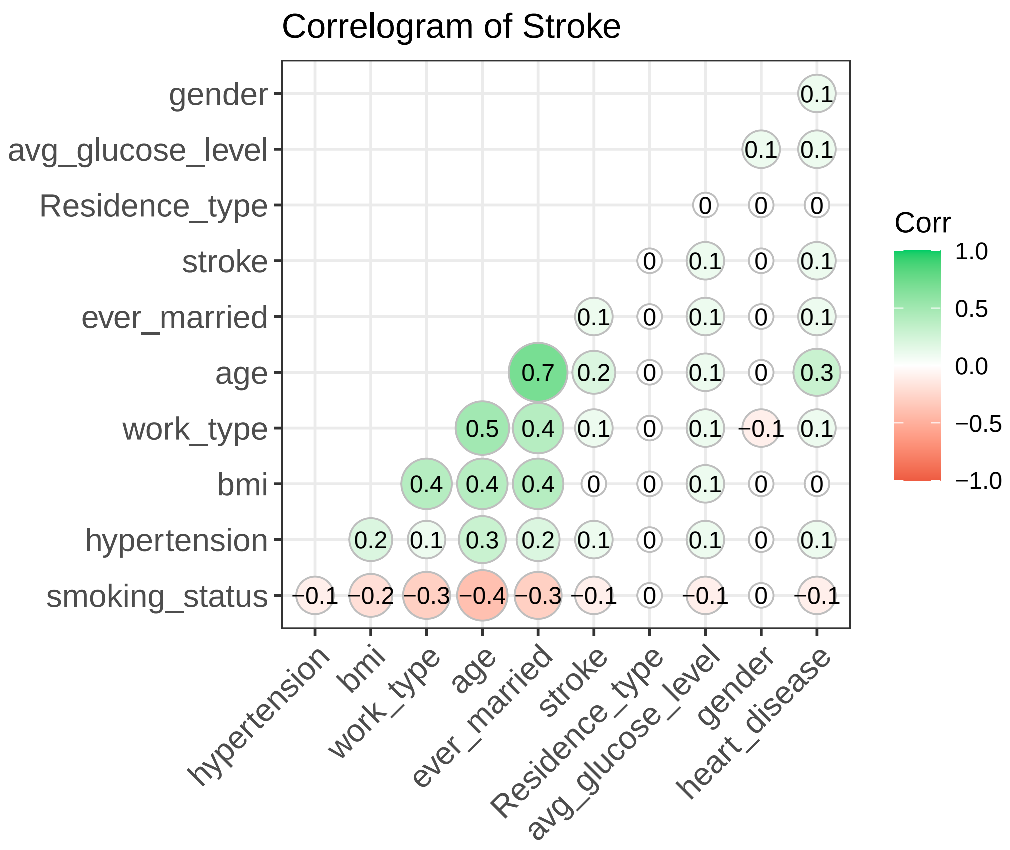
Correlation plots

```r
Stroke.corr <- Stroke
Stroke.corr[,1] <- as.numeric(factor(Stroke.corr[,1]))
Stroke.corr[,2] <- as.numeric(factor(Stroke.corr[,2]))
Stroke.corr[,3] <- as.numeric(factor(Stroke.corr[,3]))
Stroke.corr[,4] <- as.numeric(factor(Stroke.corr[,4]))
Stroke.corr[,5] <- as.numeric(factor(Stroke.corr[,5]))
Stroke.corr[,6] <- as.numeric(factor(Stroke.corr[,6]))
Stroke.corr[,7] <- as.numeric(factor(Stroke.corr[,7]))
Stroke.corr[,8] <- as.numeric(factor(Stroke.corr[,8]))
Stroke.corr[,9] <- as.numeric(factor(Stroke.corr[,9]))
Stroke.corr[,10] <- as.numeric(factor(Stroke.corr[,10]))
Stroke.corr[,11] <- as.numeric(factor(Stroke.corr[,11]))

corrplot(cor(na.omit(Stroke.corr)), method = "number")
```

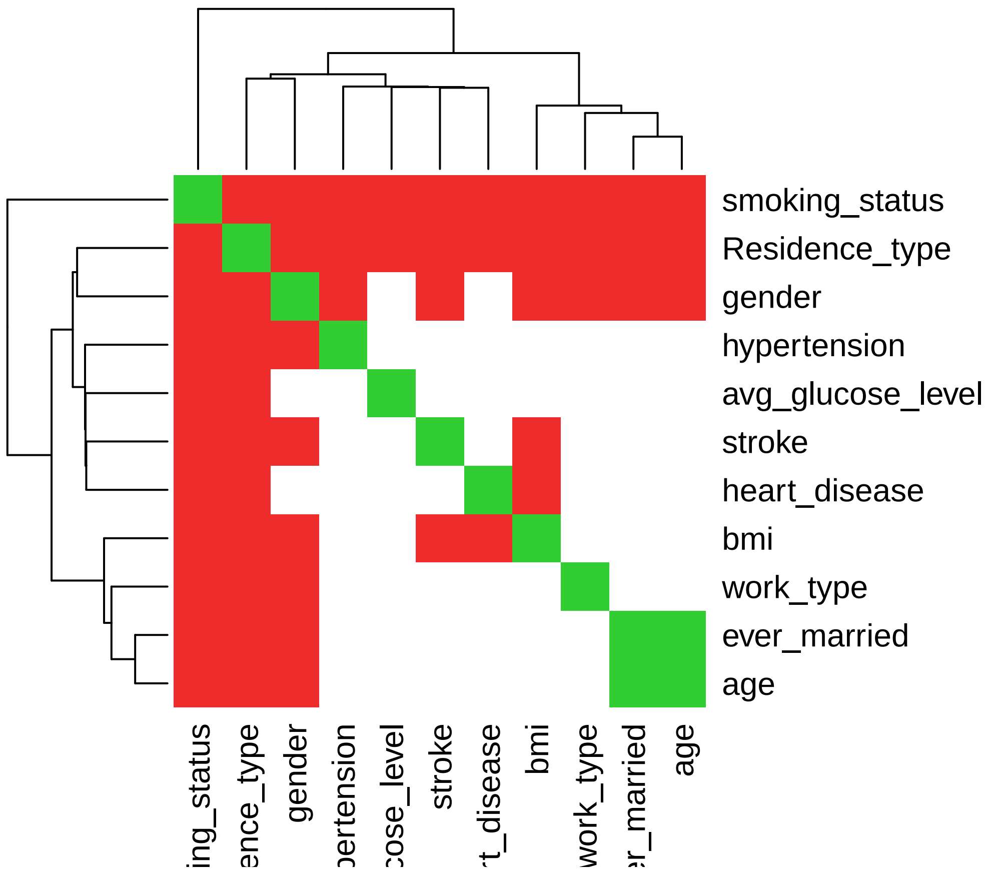


```
corr <- round(cor(na.omit(Stroke.corr)), 1)
ggcorrplot(corr, hc.order = TRUE,
           type = "lower",
           lab = TRUE,
           lab_size = 3,
           method="circle",
           colors = c("tomato2", "white", "springgreen3"),
           title="Correlogram of Stroke",
           ggtheme=theme_bw)
```

```
## Warning: ‘guides(<scale> = FALSE)‘ is deprecated. Please use ‘guides(<scale> =
## "none")‘ instead.
```



```
heatmap(x = corr, col = c("firebrick2", "white", "limegreen"), symm = TRUE)
```



Preparing the data for analysis :

```
# Common data standardization
# Remove only row with gender = "Other"
Stroke = Stroke[-3117,]
# Set 2 levels for gender (and not 3):
Stroke$gender <- factor(Stroke$gender, levels = c("Female", "Male"))

# Get rid of correlated variables: ever_married and work_type_children:
Stroke = Stroke[,-5]

# Replace NAs with mean value (column bmi):
Stroke <- na_mean(Stroke)
```

1) Logistic Regression

Partition for LR:

We split 60%-Training, 40%-Validation and normalize the data

```r
set.seed(1)
train.index.lr = sample(c(1:NROW(Stroke)), round(NROW(Stroke)*0.5))
train.lr = Stroke[train.index.lr,]  # Train set
valid.test.lr = Stroke[-train.index.lr,]
valid.index.lr = sample(c(1:NROW(valid.test.lr)), round(NROW(valid.test.lr)*0.6))
valid.lr = valid.test.lr[valid.index.lr,] # Valid set
test.lr = valid.test.lr[-valid.index.lr,] # Test set

dim(train.lr)
```

```
## [1] 2554   10
```

```r
dim(valid.lr)
```

```
## [1] 1533   10
```

```r
dim(test.lr)
```

```
## [1] 1022   10
```

```r
# The data has been correctly partitioned into 60% (3'065 obs) in T and 40% in V (2'044 obs)
```

Doing LR

```r
set.seed(1)
# 1) Logistic Regression
logit.reg <- glm(stroke ~., data = train.lr, family = "binomial")
options(scipen = 999)
summary(logit.reg)
```

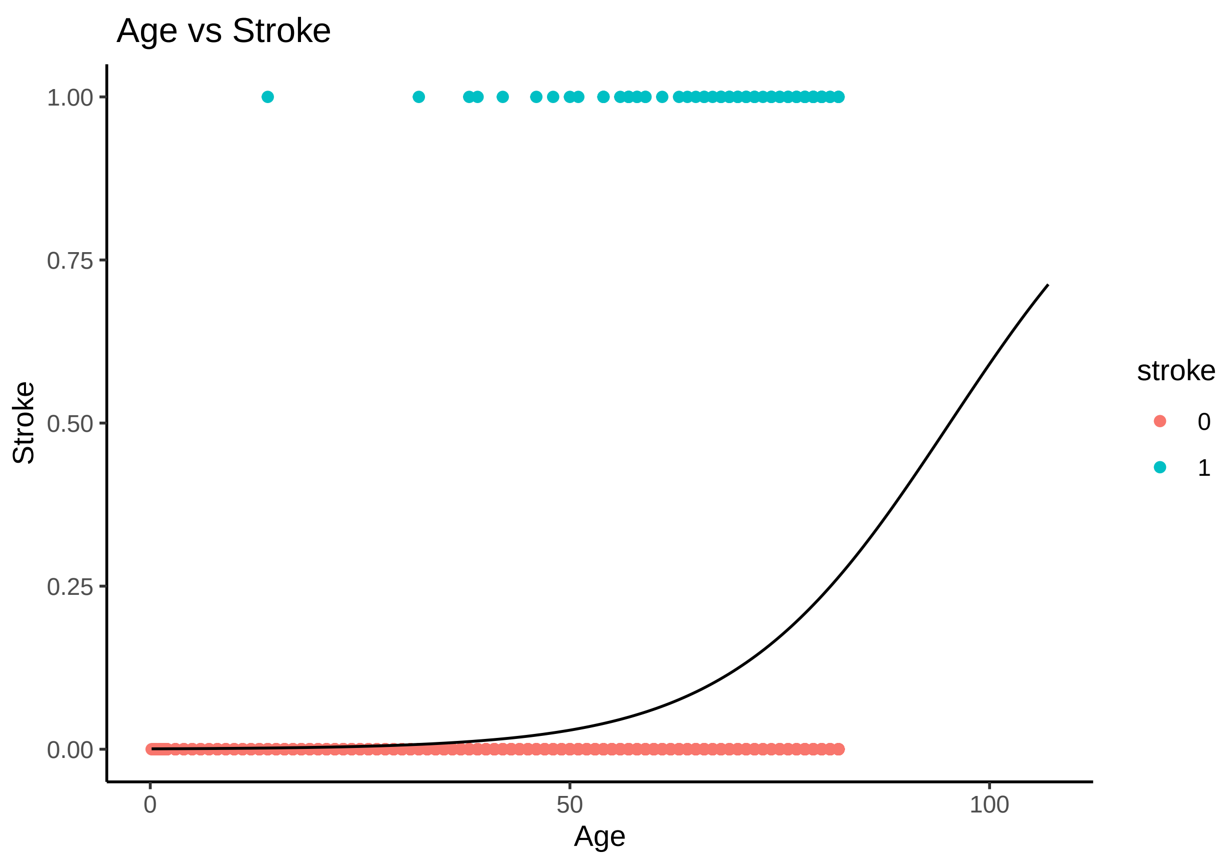
```
##
## Call:
## glm(formula = stroke ~ ., family = "binomial", data = train.lr)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1918  -0.3189  -0.1624  -0.0824   3.5431
##
## Coefficients:
##                             Estimate Std. Error z value             Pr(>|z|)
## (Intercept)                -6.871647   1.109622  -6.193       0.000000000591
## genderMale                 -0.079608   0.198349  -0.401              0.68816
## age                         0.075445   0.008411   8.970 < 0.0000000000000002
## hypertension                0.714353   0.224014   3.189              0.00143
## heart_disease               0.573700   0.251742   2.279              0.02267
## work_typeGovt_job          -0.891500   1.165161  -0.765              0.44419
## work_typeNever_worked     -11.217690 638.089433  -0.018              0.98597
## work_typePrivate           -0.887875   1.143017  -0.777              0.43729
## work_typeSelf-employed      -1.263710   1.178934  -1.072             0.28376
## Residence_typeUrban          0.165124   0.193104   0.855             0.39249
## avg_glucose_level            0.003068   0.001704   1.801             0.07174
## bmi                         -0.003033   0.015891  -0.191             0.84865
## smoking_statusnever smoked  -0.112022   0.248067  -0.452             0.65157
## smoking_statussmokes         0.139040   0.314279   0.442             0.65819
## smoking_statusUnknown        0.203656   0.284873   0.715             0.47467
##
## (Intercept)                 ***
## genderMale
## age                         ***
## hypertension                **
## heart_disease               *
## work_typeGovt_job
## work_typeNever_worked
## work_typePrivate
## work_typeSelf-employed
## Residence_typeUrban
## avg_glucose_level           .
## bmi
## smoking_statusnever smoked
## smoking_statussmokes
## smoking_statusUnknown
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1033.36  on 2553  degrees of freedom
## Residual deviance:  803.86  on 2539  degrees of freedom
## AIC: 833.86
##
## Number of Fisher Scoring iterations: 15
```

For variables with NA (self-employed and unknown), they are the defaults dummy values; just like genderFemale is the default dummy of the regression and ever_married_No is the default dummy for ever_married:

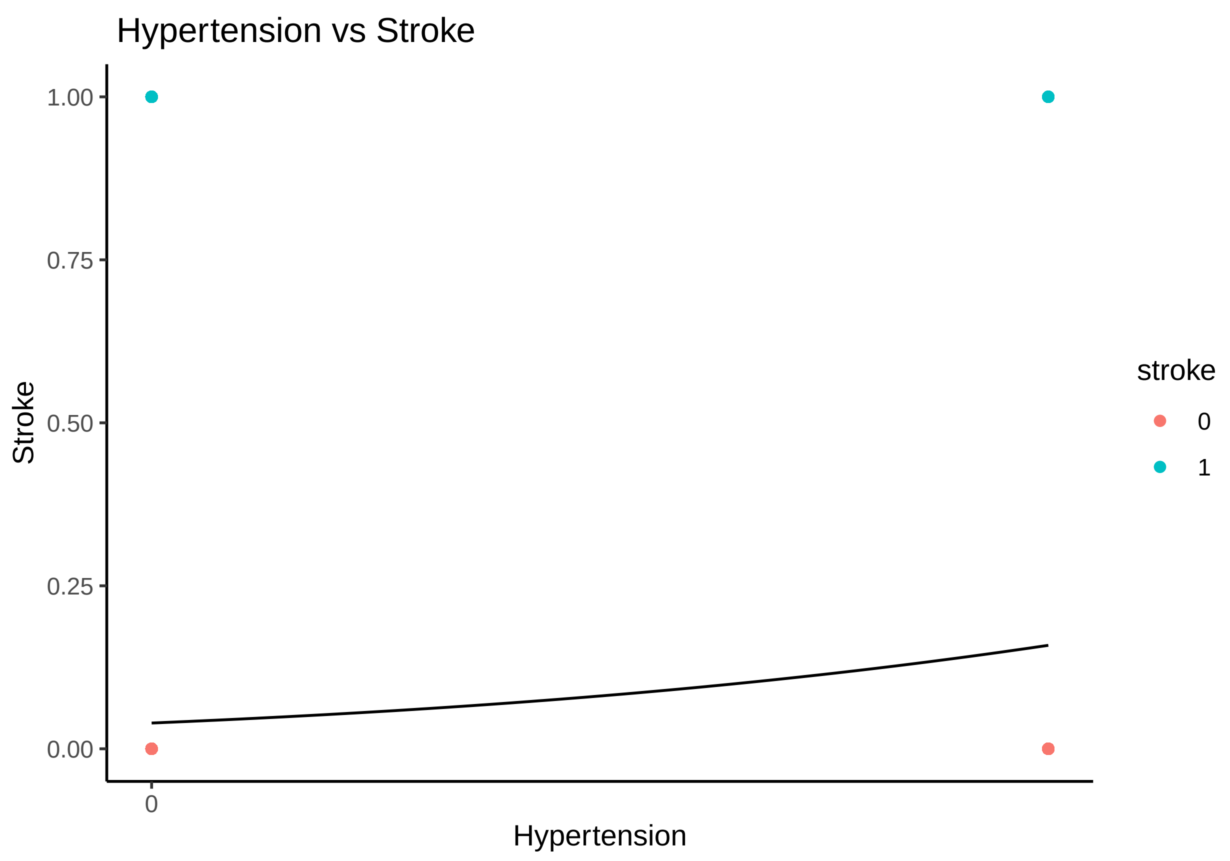
- The effect of being male is positive => male have higher propensity of stroke
- The coefficients effect of the work_type dummies are in comparison to the default value unknown (which is why it has NA).
- The coefficients effect of the smoking_status dummies are in comparison to the default value self-employed (with NA).

The significant predictors are in order (from most influencer to least) age, hypertension and avg_glucose_level.

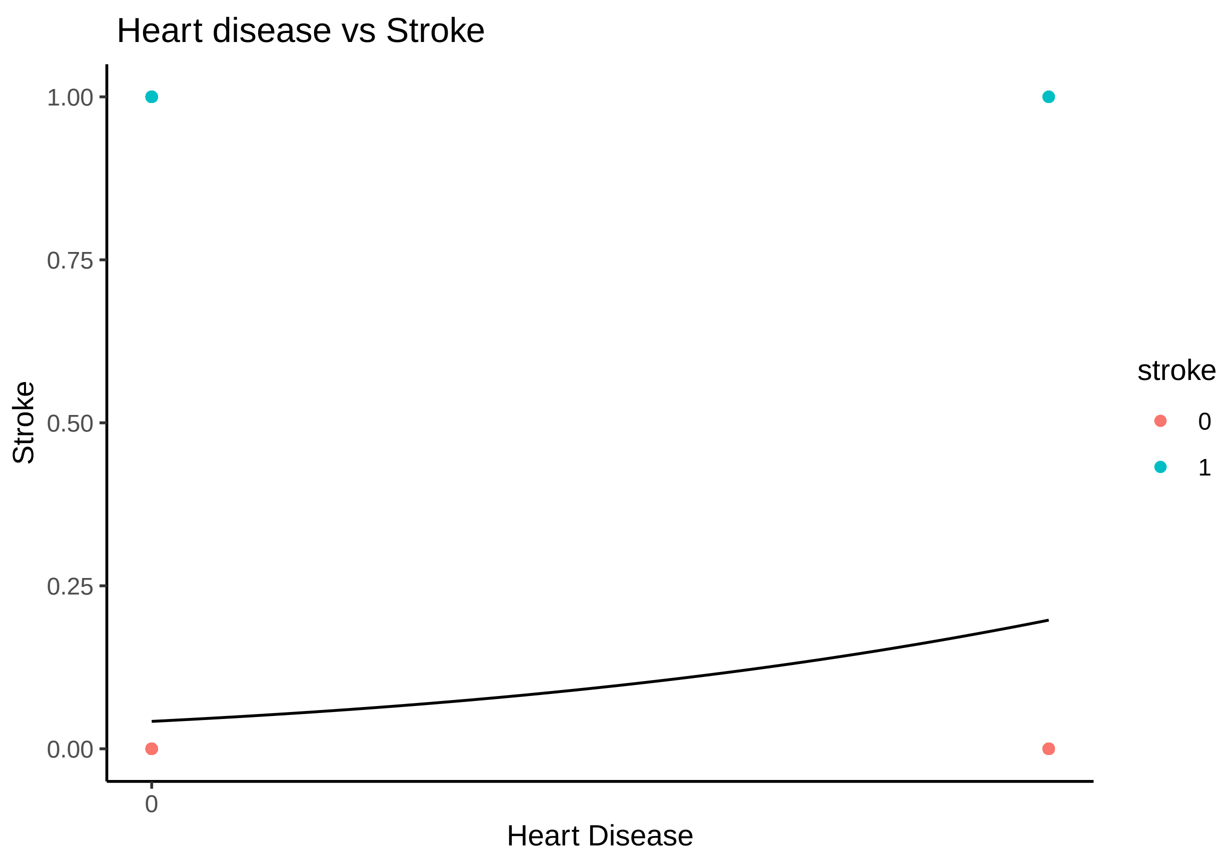
```r
# Graphs for the significant variables
# For age:
mod <- glm(stroke ~ age, data = train.lr, binomial("logit"))
coefs <- coef(mod)
x_plot <- with(train.lr, seq(min(age), max(age)+25, length.out = 200))
y_plot <- plogis(coefs[1] + coefs[2] * x_plot)

ggplot(valid.lr) + geom_point(aes(x=age,y=as.integer(stroke)-1, color=stroke)) + geom_line(aes(x= x_plo
```



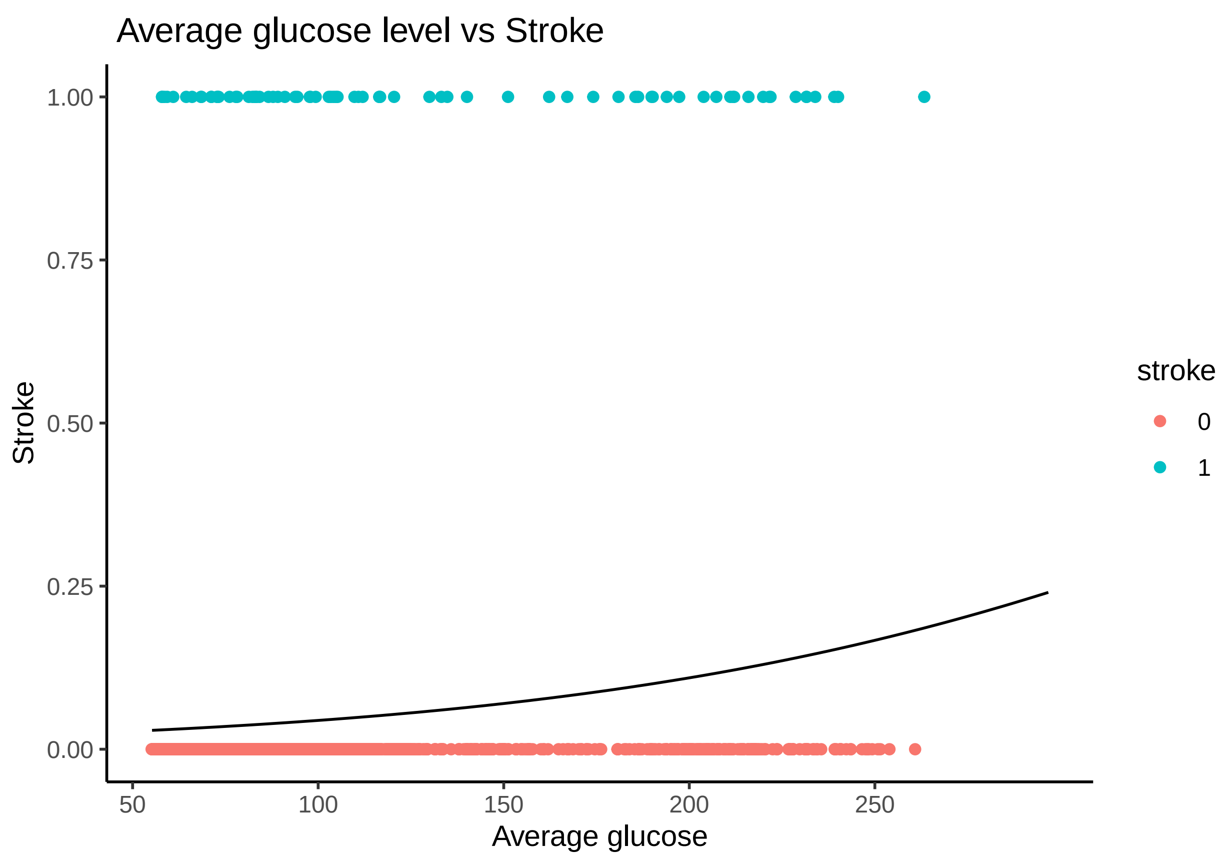
```r
# For hypertension:
mod2 <- glm(stroke ~ hypertension, data = train.lr, binomial("logit"))
coefs2 <- coef(mod2)
x_plot2 <- with(train.lr, seq(min(hypertension), max(hypertension), length.out = 200))
y_plot2 <- plogis(coefs2[1] + coefs2[2] * x_plot2)

ggplot(valid.lr) + geom_point(aes(x=hypertension,y=as.integer(stroke)-1, color=stroke)) + geom_line(aes
```



```
# For heart_disease:
mod3 <- glm(stroke ~ heart_disease, data = train.lr, binomial("logit"))
coefs3 <- coef(mod3)
x_plot3 <- with(train.lr, seq(min(heart_disease), max(heart_disease), length.out = 200))
y_plot3 <- plogis(coefs3[1] + coefs3[2] * x_plot3)

ggplot(valid.lr) + geom_point(aes(x=heart_disease,y=as.integer(stroke)-1, color=stroke)) + geom_line(ae
```



```
# For average glucose level:
mod4 <- glm(stroke ~ avg_glucose_level, data = train.lr, binomial("logit"))
coefs4 <- coef(mod4)
x_plot4 <- with(train.lr, seq(min(avg_glucose_level), max(avg_glucose_level)+25, length.out = 200))
y_plot4 <- plogis(coefs4[1] + coefs4[2] * x_plot4)

ggplot(valid.lr) + geom_point(aes(x=avg_glucose_level,y=as.integer(stroke)-1, color=stroke)) + geom_lin
```



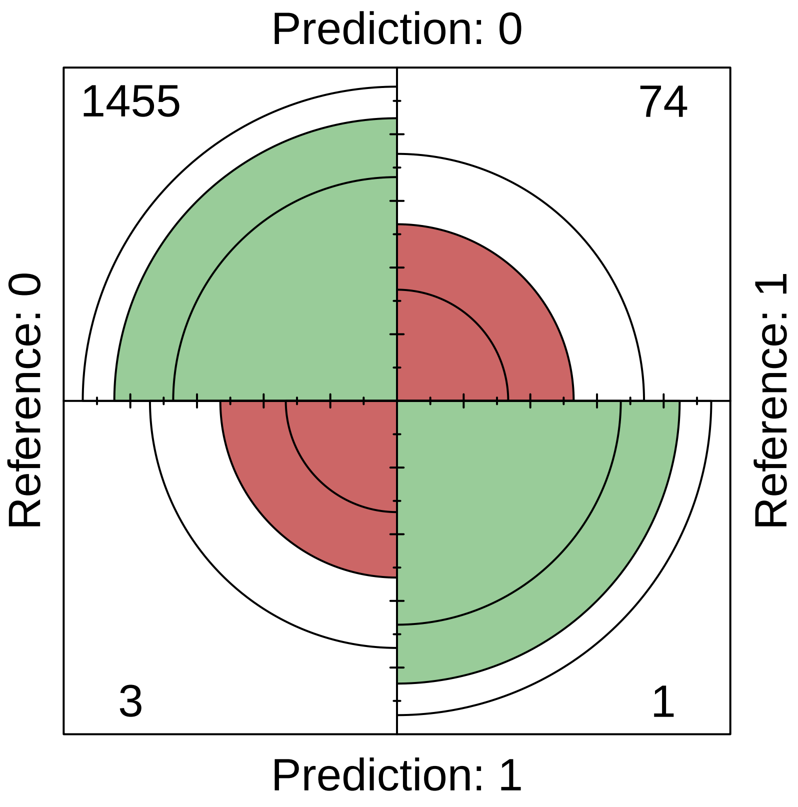
- Age: The older you get, the more probable it is you get a stroke. The first instances of stroke arises around 30 year of age according to the graph; and it start rising the older you get.
- Average glucose: the progression of the slope is slower and way less impactful than age; this is to be expected as the variable avg_glucose_level has lower significance in the model.

```
# Confusion matrix with cut-off 0.5:
set.seed(1)
pred.lr = predict(logit.reg, valid.lr[,-10], type = "response")
confusionMatrix(as.factor(ifelse(pred.lr > 0.5, 1, 0)), valid.lr$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1455   74
##          1    3    1
##
##                Accuracy : 0.9498
##                  95% CI : (0.9376, 0.9602)
##     No Information Rate : 0.9511
##     P-Value [Acc > NIR] : 0.6226
##
##                   Kappa : 0.0205
##
##  Mcnemar's Test P-Value : 0.000000000000001496
```

```
##
##             Sensitivity : 0.0133333
##             Specificity : 0.9979424
##          Pos Pred Value : 0.2500000
##          Neg Pred Value : 0.9516024
##              Prevalence : 0.0489237
##          Detection Rate : 0.0006523
##    Detection Prevalence : 0.0026093
##       Balanced Accuracy : 0.5056379
##
##        'Positive' Class : 1
##
```

```
fourfoldplot(confusionMatrix(as.factor(ifelse(pred.lr > 0.5, 1, 0)), valid.lr$stroke, positive = "1")$ta
```

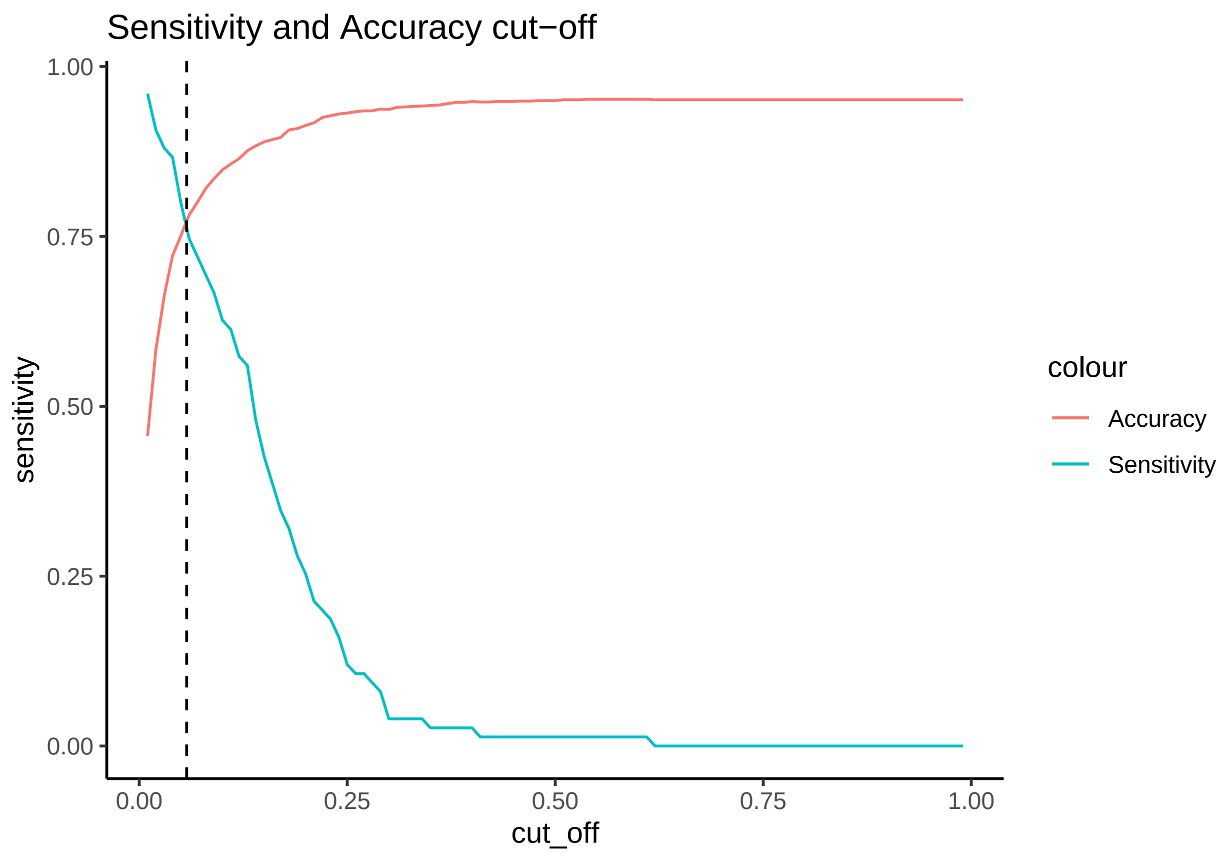


This model is not able to predict any stroke; it is not efficient for the goal of our analysis. It is most probable that we have too many unnecessary variables in our model.

The model's goal is to predict stroke. As such, a false positive (saying a patient is likely to get a stroke when they are not) is better than a false negative (saying a patient will not get a stroke when they are actually likely). Thus, sensitivity (rate of true positive) is more important than both specificity (rate of true negative); because detecting a false stroke have less negative consequences than not detecting it (preventive medical monitoring cannot hurt, but not providing them when it could have been useful does).

Find the cutoff that maximizes both sensitivity and overall accuracy:

```r
# Find the best cut-off:
cut_off = seq(0.01,0.99,0.01)
metrics.lr <- data.frame(cut_off, sensitivity = rep(0, length(cut_off)), accuracy = rep(0, length(cut_o

for (i in 1:length(cut_off)){
  x <- as.factor(ifelse(pred.lr >= cut_off[i],1,0))
  metrics.lr[i, 2] <- confusionMatrix(x, valid.lr$stroke, positive = "1")$byClass[1]
  metrics.lr[i, 3] <- confusionMatrix(x, valid.lr$stroke, positive = "1")$overall[1]
}

cutoff.plot = ggplot(data= metrics.lr) + geom_line(aes(x=cut_off,y=sensitivity, color="Sensitivity")) +

cutoff.plot
```



```r
metrics.lr$difference = sqrt((metrics.lr$accuracy - metrics.lr$sensitivity)^2)
cut_off.best = cut_off[which.min(metrics.lr$difference)]
cut_off.best
```

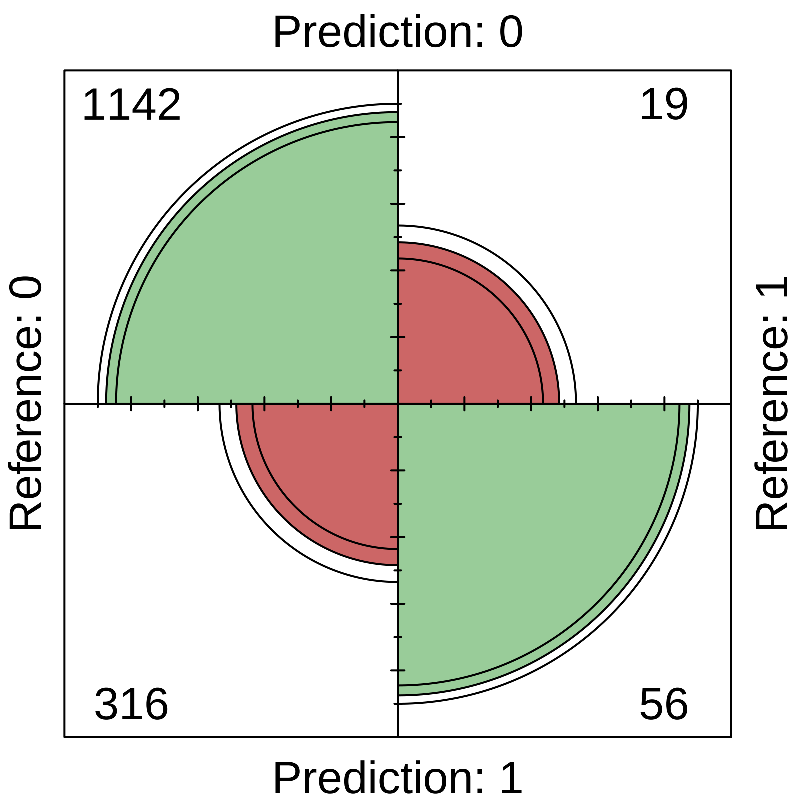
```
## [1] 0.06
```

The best cut-off is 0.06; which is closer to the ratio of stroke/non-stroke in presence in this hospital's dataset: 4.8728%.

```
# Confusion matrix with best cut-off (0.4):
conf.lr = confusionMatrix(as.factor(ifelse(pred.lr > cut_off.best, 1, 0)), valid.lr$stroke, positive =
conf.lr
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1142   19
##          1  316   56
##
##                Accuracy : 0.7815
##                  95% CI : (0.7599, 0.8019)
##     No Information Rate : 0.9511
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1841
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.74667
##             Specificity : 0.78326
##          Pos Pred Value : 0.15054
##          Neg Pred Value : 0.98363
##              Prevalence : 0.04892
##          Detection Rate : 0.03653
##    Detection Prevalence : 0.24266
##       Balanced Accuracy : 0.76497
##
##        'Positive' Class : 1
##
```

```
fourfoldplot(confusionMatrix(as.factor(ifelse(pred.lr > cut_off.best, 1, 0)), valid.lr$stroke, positive
```



#### 2) Classification Tree

Partition for Classification Tree

We partition into 60/40 and put categorical variables with more than 2 levels as dummies:

```
set.seed(1)
Stroke.dummy = Stroke   # Stroke.withNA

# Put work_type as dummy:
work_type_Dummies <- cbind(Stroke.dummy[1:4], dummy(Stroke.dummy$work_type, sep = "_"), Stroke.dummy[6:
names(work_type_Dummies)[5:9] <- c("children","Govt_job", "Never_Worked", "Private", "Self-employed")
#work_type_Dummies
Stroke.dummy <- work_type_Dummies
```

```
# Put smoking_status as dummy:
smoking_status_Dummies <- cbind( Stroke.dummy[1:12], dummy(Stroke.dummy$smoking_status, sep = "_"), Str
names(smoking_status_Dummies)[13:16] <- c("formerly_smoked", "never_smoked", "smokes", "unknown")
#smoking_status_Dummies
Stroke.dummy <- smoking_status_Dummies

train.index.ct = sample(c(1:NROW(Stroke.dummy)), round(NROW(Stroke.dummy)*0.5))
train.ct = Stroke.dummy[train.index.ct,]   # Train set
valid.test.ct = Stroke.dummy[-train.index.ct,]
valid.index.ct = sample(c(1:NROW(valid.test.ct)), round(NROW(valid.test.ct)*0.6))
valid.ct = valid.test.ct[valid.index.ct,] # Valid set
test.ct = valid.test.ct[-valid.index.ct,] # Test set

dim(train.ct)
```

```
## [1] 2554   17
```

```
dim(valid.ct)
```

```
## [1] 1533   17
```

```
dim(test.ct)
```

```
## [1] 1022   17
```

Doing CT

```
# Full-grown tree:
set.seed(1)
c.tree = rpart(stroke ~., data = train.ct, method = "class", cp=0, minbucket=1, minsplit = 1, xval =5, 
length(c.tree$frame$var[c.tree$frame$var == "<leaf>"])
```

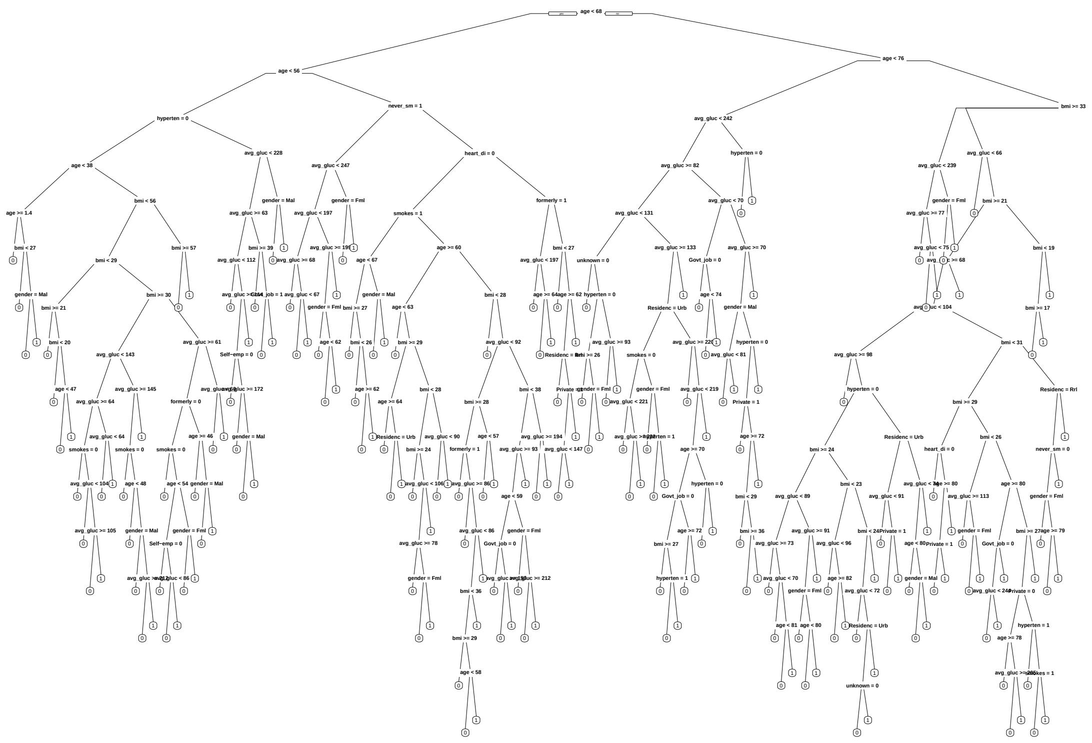
```
## [1] 190
```

```
c.tree$variable.importance
```

```
## avg_glucose_level               age               bmi            gender
##         86.407429         61.162516         57.178997         20.349436
##           Private      hypertension    Residence_type     Self-employed
##         13.923963         13.647692         11.560232          7.947806
##          Govt_job     heart_disease   formerly_smoked            smokes
##          7.440944          6.714577          6.174170          5.864389
##      never_smoked           unknown
##          4.544718          3.054517
```

```
prp(c.tree)
```

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



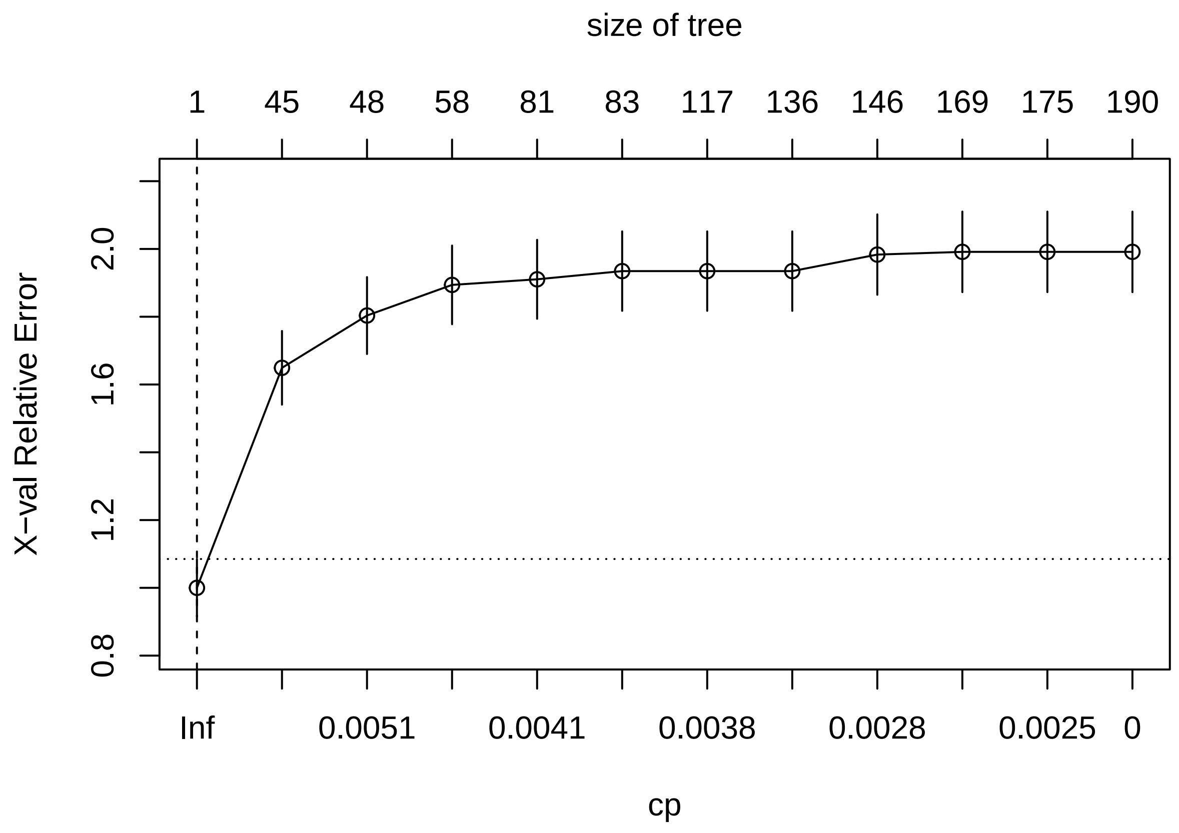
```r
# Find the best pruned tree:
set.seed(1)
printcp(c.tree)
```

```
##
## Classification tree:
## rpart(formula = stroke ~ ., data = train.ct, method = "class",
##     parms = list(prior = c(ratio_0, ratio_1)), cp = 0, minbucket = 1,
##     minsplit = 1, xval = 5)
##
## Variables actually used in tree construction:
##  [1] age               avg_glucose_level bmi               formerly_smoked
##  [5] gender            Govt_job          heart_disease     hypertension
##  [9] never_smoked      Private           Residence_type    Self-employed
## [13] smokes            unknown
##
## Root node error: 124.45/2554 = 0.048728
##
## n= 2554
##
##           CP nsplit rel error xerror    xstd
## 1  0.0064826      0  1.000000 1.0000 0.08510
## 2  0.0050891     44  0.618321 1.6492 0.10879
## 3  0.0050891     47  0.603053 1.8036 0.11349
## 4  0.0042414     57  0.526718 1.8939 0.11614
## 5  0.0040285     80  0.389736 1.9104 0.11662
## 6  0.0038168     82  0.381679 1.9346 0.11730
## 7  0.0038168    116  0.251908 1.9346 0.11730
## 8  0.0030534    135  0.167939 1.9346 0.11730
## 9  0.0025445    145  0.137405 1.9834 0.11866
## 10 0.0025445    168  0.061069 1.9914 0.11888
## 11 0.0025445    174  0.038168 1.9914 0.11888
## 12 0.0000000    189  0.000000 1.9914 0.11888
```

```
prune_cp.index = which.min(c.tree$cptable[,"xerror"])
prune_cp = c.tree$cptable[prune_cp.index,"CP"]
prune_cp
```

```
## [1] 0.006482589
```

```
p <- plotcp(c.tree)
p + abline(v = prune_cp.index, lty = "dashed")
```



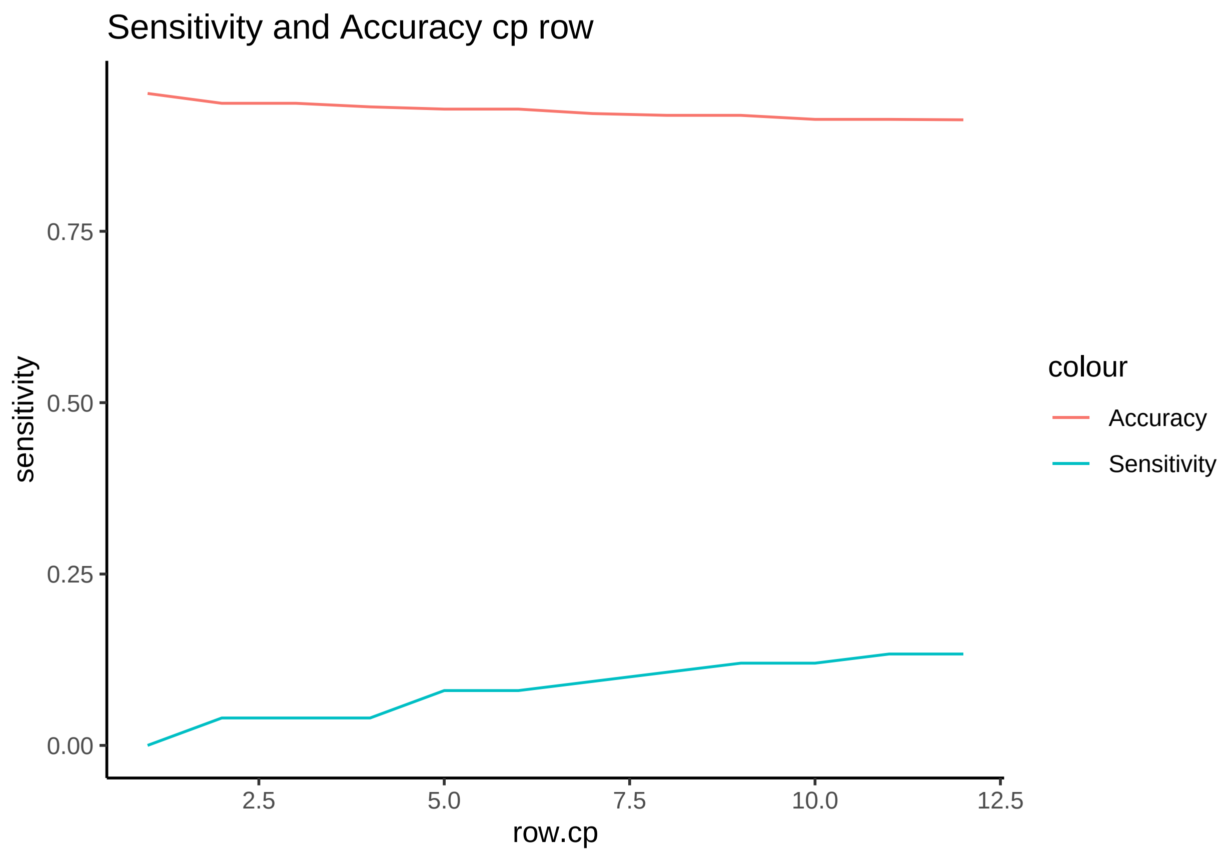
```
## integer(0)
```

```
# Pruned tree:
set.seed(1)
pruned.ct <- prune(c.tree, cp = prune_cp, parms = list(prior= c(ratio_0, ratio_1)))
length(pruned.ct$frame$var[pruned.ct$frame$var == "<leaf>"])
```

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

Again, the model is not able to accurately predict strokes due to a highly unbalanced dataset. We must find the best cp, such that it maximizes both sensitivity and accuracy for the model to be able to predict stroke (according to our goal).

```
# Find the best cp:
pred.ct = predict(c.tree, valid.ct, type = "class")

row.cp = seq(1,nrow(c.tree$cptable),1)
metrics.ct <- data.frame(row.cp, sensitivity = rep(0, length(row.cp)), accuracy = rep(0, length(row.cp)

for (i in 1:length(row.cp)){
  set.seed(1)
  pruned <- prune(c.tree, cp = c.tree$cptable[,"CP"][row.cp][i])
  x <- predict(pruned, valid.ct, type = "class", parms = list(prior= c(ratio_0, ratio_1)))
  metrics.ct[i, 2] <- confusionMatrix(x, valid.ct$stroke, positive = "1")$byClass[1]
  metrics.ct[i, 3] <- confusionMatrix(x, valid.ct$stroke, positive = "1")$overall[1]
}

cp.plot = ggplot(data= metrics.ct) + geom_line(aes(x=row.cp,y=sensitivity, color="Sensitivity")) + geom_

cp.plot
```

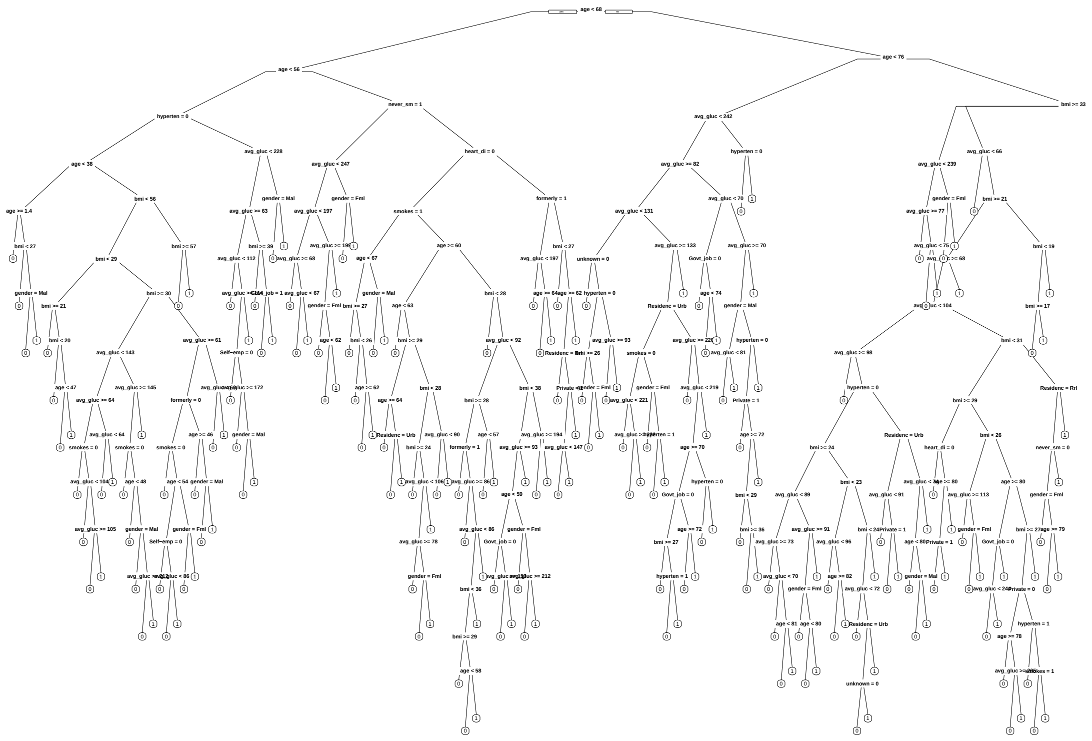


```
metrics.ct$difference = sqrt((metrics.ct$accuracy - metrics.ct$sensitivity)^2)
row.cp.best = row.cp[which.min(metrics.ct$difference)]
cp.best = c.tree$cptable[,"CP"][row.cp][row.cp.best]
cp.best
```

```
## 12
##  0
```

```
set.seed(1)
best.pruned <- prune(c.tree, cp = cp.best)
pred.best.pruned <- predict(best.pruned, valid.ct, type = "class", parms = list(prior= c(ratio_0, ratio_
prp(best.pruned)
```

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

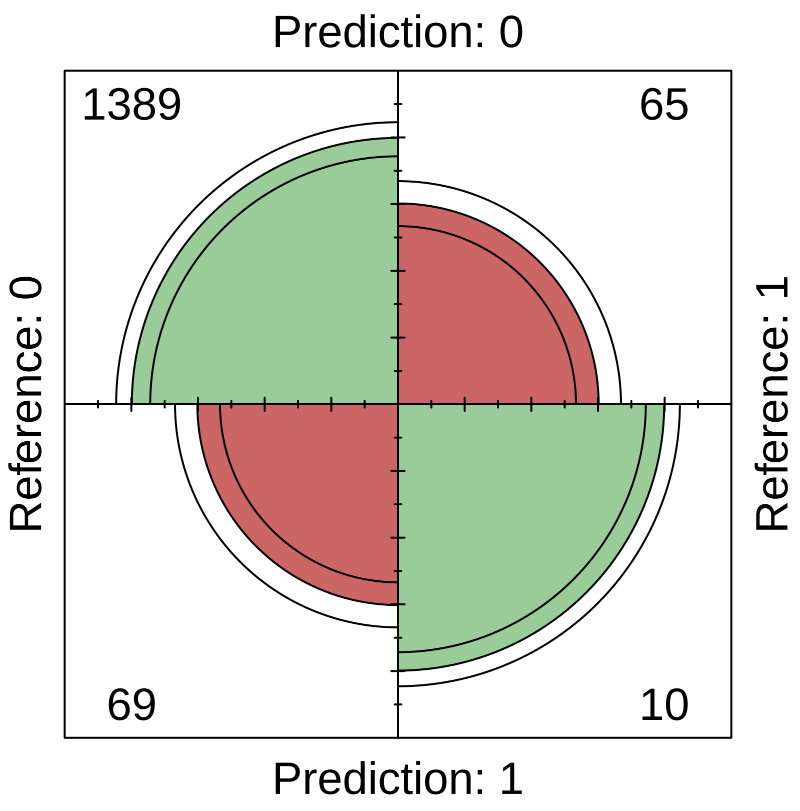


The best pruned tree for predictive stroke is (almost) the full classification tree.

```
# Confusion matrix for CT:
conf.ct = confusionMatrix(pred.best.pruned, valid.ct$stroke, positive = "1")
conf.ct
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    0    1
##          0 1389   65
##          1   69   10
##
##                Accuracy : 0.9126
##                  95% CI : (0.8973, 0.9263)
##     No Information Rate : 0.9511
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : 0.0839
##
##  Mcnemar's Test P-Value : 0.7955
##
##             Sensitivity : 0.133333
##             Specificity : 0.952675
##          Pos Pred Value : 0.126582
##          Neg Pred Value : 0.955296
##              Prevalence : 0.048924
##          Detection Rate : 0.006523
##    Detection Prevalence : 0.051533
##       Balanced Accuracy : 0.543004
##
##        'Positive' Class : 1
##
```

```
fourfoldplot(confusionMatrix(pred.best.pruned, valid.ct$stroke, positive = "1")$table, color = c("#CC66
```



3) KNN

Partition for KNN

We partition into 60/40, create dummy variables, change all predictors to numerical and normalize them:

```
# Dummify:
Stroke$stroke <- as.integer(ifelse(Stroke$stroke == "0", 0,1))  # change stroke to int during dummifica
dummies = dummyVars(~ ., data=Stroke)
Stroke.fltr = as.data.frame(predict(dummies, newdata = Stroke)[,-10])
Stroke.fltr$stroke = as.factor(Stroke.fltr$stroke)  # change stroke back to factor
str(Stroke.fltr)
```

```
## 'data.frame':    5109 obs. of  18 variables:
##  $ gender.Female                 : num  0 1 0 1 1 0 0 1 1 1 ...
##  $ gender.Male                   : num  1 0 1 0 0 1 1 0 0 0 ...
##  $ age                           : num  67 61 80 49 79 81 74 69 59 78 ...
##  $ hypertension                  : num  0 0 0 0 1 0 1 0 0 0 ...
##  $ heart_disease                 : num  1 0 1 0 0 0 1 0 0 0 ...
##  $ work_type.children            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ work_type.Govt_job            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ work_type.Never_worked        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ work_type.Private             : num  1 0 1 1 0 1 1 1 1 1 ...
##  $ Residence_type.Rural          : num  0 1 1 0 1 0 1 0 1 0 ...
##  $ Residence_type.Urban          : num  1 0 0 1 0 1 0 1 0 1 ...
##  $ avg_glucose_level             : num  229 202 106 171 174 ...
##  $ bmi                           : num  36.6 28.9 32.5 34.4 24 ...
##  $ smoking_status.formerly smoked: num  1 0 0 0 0 1 0 0 0 0 ...
##  $ smoking_status.never smoked   : num  0 1 1 0 1 0 1 1 0 0 ...
##  $ smoking_status.smokes         : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ smoking_status.Unknown        : num  0 0 0 0 0 0 0 0 1 1 ...
##  $ stroke                        : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

```
prop.table(table(Stroke.fltr$stroke))
```

```
##
##          0          1
## 0.95126248 0.04873752
```

```
set.seed(1)
train.index.knn = sample(NROW(Stroke.fltr), round(NROW(Stroke.fltr)*0.6))
train.knn = Stroke.fltr[ train.index.knn,]
valid.knn  = Stroke.fltr[-train.index.knn,]



train.index.knn = sample(c(1:NROW(Stroke.fltr)), round(NROW(Stroke.fltr)*0.5))
train.knn = Stroke.fltr[train.index.knn,]  # Train set
valid.test.knn = Stroke.fltr[-train.index.knn,]
valid.index.knn = sample(c(1:NROW(valid.test.knn)), round(NROW(valid.test.knn)*0.6))
valid.knn = valid.test.knn[valid.index.knn,] # Valid set
test.knn = valid.test.knn[-valid.index.knn,] # Test set
```

```
dim(train.knn)
```

```
## [1] 2554   18
```

```
dim(valid.knn)
```

```
## [1] 1533   18
```

```
dim(test.knn)
```

```
## [1] 1022   18
```

```
print(table(train.knn$stroke) / table(Stroke.fltr$stroke)) #Training set partition
```

```
##
##         0         1
## 0.5006173 0.4859438
```

```
print(table(valid.knn$stroke) / table(Stroke.fltr$stroke)) #Validation set partition
```

```
##
##         0         1
## 0.3004115 0.2931727
```

```
print(table(test.knn$stroke) / table(Stroke.fltr$stroke)) #Validation set partition
```

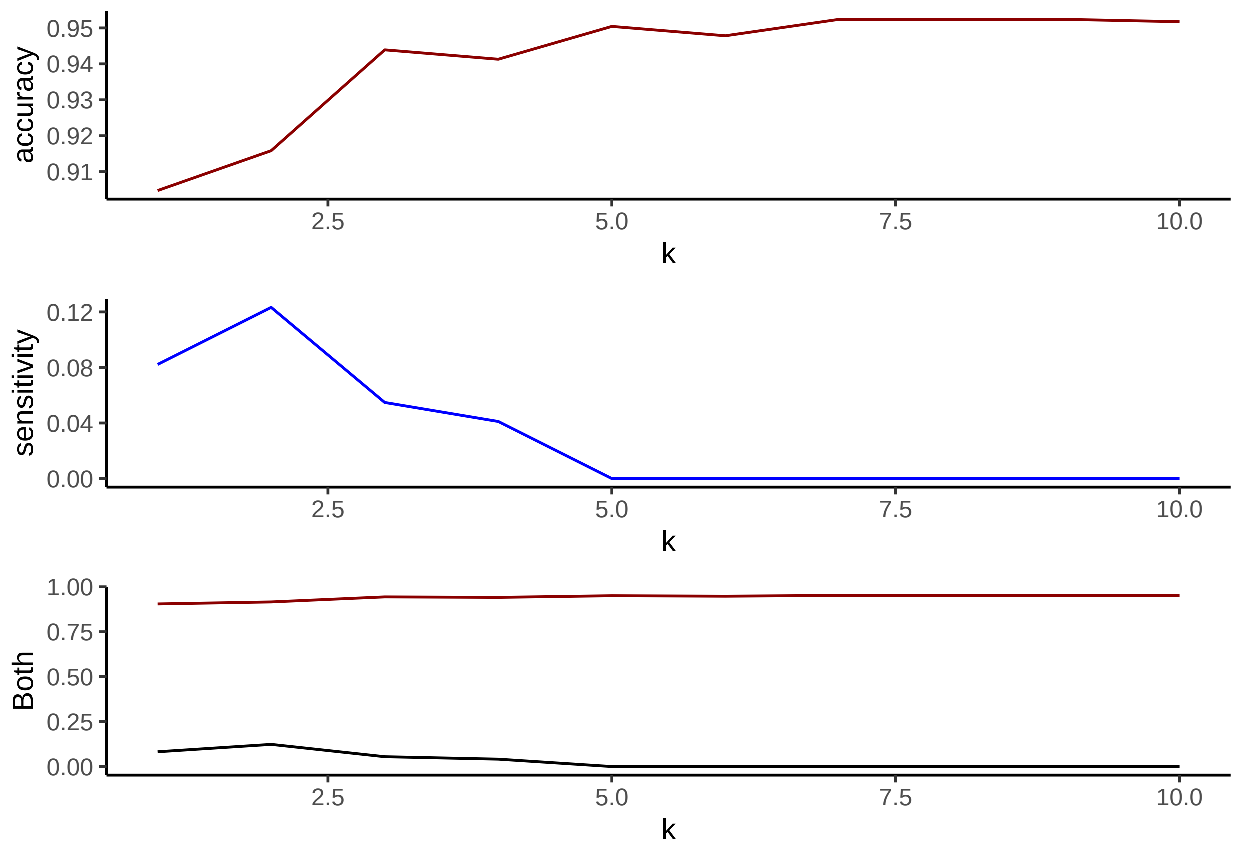
```
##
##         0         1
## 0.1989712 0.2208835
```

```
# Normalize the dummified sets:
train.norm.knn = train.knn
valid.norm.knn = valid.knn
test.norm.knn = test.knn
# Normalization of numerical variables:
norm.values = preProcess(train.knn[, c(3,12,13)], method=c("range"))
train.norm.knn[, c(3,12,13)] = predict(norm.values, train.knn[, c(3,12,13)])
valid.norm.knn[, c(3,12,13)] = predict(norm.values, valid.knn[, c(3,12,13)])
test.norm.knn[, c(3,12,13)] = predict(norm.values, test.knn[, c(3,12,13)])
```

Doing KNN

```
### Creating the graph for best K. (column 21/22 = stroke0/stroke1)

set.seed(1)
nk = 10
```

```
metrics.knn <- data.frame(k = seq(1, nk, 1), accuracy = rep(0, nk), sensitivity = rep(0, nk), F1_score =

for(i in 1:nk) {
  knn.pred = knn(train.norm.knn[, -18], valid.norm.knn[, -18], cl = train.norm.knn[, 18], k = i)
  metrics.knn[i, 2] = confusionMatrix(knn.pred, valid.norm.knn[, 18], positive = "1")$overall[1]
  metrics.knn[i, 3] = confusionMatrix(knn.pred, valid.norm.knn[, 18], positive = "1")$byClass[1]
}

accuracy.plot = ggplot(data= metrics.knn) + geom_line(aes(x=k,y=accuracy), color="darkred") + theme_cla
sensitivity.plot = ggplot(data= metrics.knn) + geom_line(aes(x=k,y=sensitivity), color="blue") + theme_
both.plot = ggplot(data= metrics.knn) + geom_line(aes(x=k,y=accuracy), color="darkred") + geom_line(aes

plot_grid(accuracy.plot, sensitivity.plot, both.plot, ncol = 1, align = "v")
```



```
# Compute best trade-off sensitivity-accuracy
metrics.knn$difference = sqrt((metrics.knn$accuracy - metrics.knn$sensitivity)^2)
k.best = metrics.knn$k[which.min(metrics.knn$difference)]
k.best
```

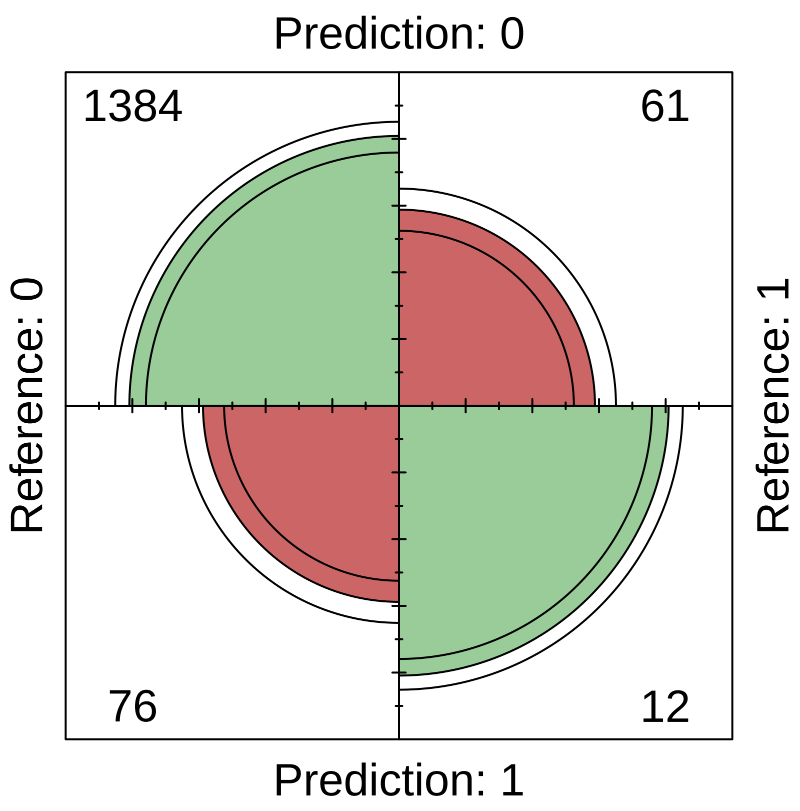
```
## [1] 2
```

The best k is 2, which is logical for such an unbalanced dataset. Indeed, the higher the value of k (= number of reference neighbors), the higher the chance of belonging to the majority group, especially in our case where non-stroke represents 95% of observations. So in case of unbalanced dataset, instead of faulting the results (with a bias due to a lack of diversity), a lower k helps being more precise.

```
### Creating the confusion Matrix --> need just to change the K = 1 & the 21 corresponds to the stroke

pred.knn = knn(train.norm.knn[, -18], valid.norm.knn[, -18], cl = train.norm.knn[, 18], k = k.best, prol
## k.best = 1:
conf.knn = confusionMatrix(pred.knn, valid.norm.knn[, 18], positive = "1")
conf.knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1384   61
##          1   76   12
##
##                Accuracy : 0.9106
##                  95% CI : (0.8952, 0.9244)
##     No Information Rate : 0.9524
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : 0.1023
##
##  Mcnemar's Test P-Value : 0.2317
##
##             Sensitivity : 0.164384
##             Specificity : 0.947945
##          Pos Pred Value : 0.136364
##          Neg Pred Value : 0.957785
##              Prevalence : 0.047619
##          Detection Rate : 0.007828
##    Detection Prevalence : 0.057404
##       Balanced Accuracy : 0.556164
##
##        'Positive' Class : 1
##
```

```
fourfoldplot(confusionMatrix(pred.knn, valid.norm.knn[, 18], positive = "1")$table, color = c("#CC6666"
```

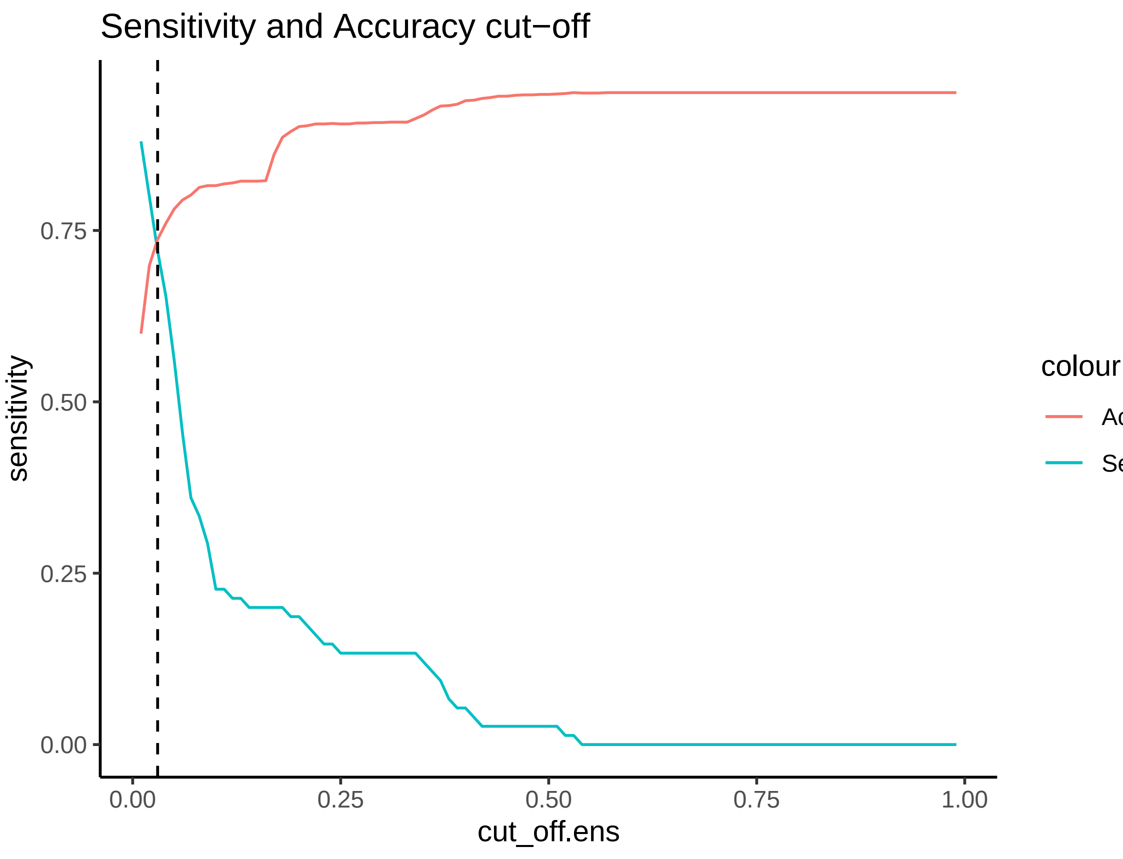


```
pred.ct.prob = predict(best.pruned, valid.ct, type = "prob", parms = list(prior= c(ratio_0, ratio_1)))[

pred.knn.prob = attr(pred.knn, "prob")

# LR:
results.df = data.frame(actual = valid.lr$stroke, lr.binary = as.factor(ifelse(pred.lr > cut_off.best, 

for(i in 1:NROW(results.df)){
  fix_prob = if(results.df[i,"knn.binary"]==1) results.df[i,"knn.prob"] else 1 - results.df[i,"knn.prob
  results.df[i,"knn.prob"] = fix_prob
}

accuracy.df = data.frame("LR.Accuracy" = conf.lr$overall[1], "CT.Accuracy" = conf.ct$overall[1], "KNN.A

sensitivity.df = data.frame("LR.Sensitivity" = conf.lr$byClass[1], "CT.Sensitivity" = conf.ct$byClass[1
```

```
# Ensembles:
results.df$average.prob = apply(results.df[,c(3,5,7)],FUN = mean, MARGIN = 1)
results.df[,c(2,4,6)] = sapply(results.df[,c(2,4,6)], as.integer) - 1
results.df$major.voting = as.factor(apply(results.df[,c(2,4,6)],FUN = function(x)
names(table(melt(x)$value))[which.max(table(melt(x)$value))], MARGIN = 1))
```

```
# Majority vote (ensemble):
accuracy.df = cbind(accuracy.df, ensemble = confusionMatrix(results.df$major.voting, results.df$actual,
sensitivity.df = cbind(sensitivity.df, ensemble = confusionMatrix(results.df$major.voting, results.df$ac
```

```
# Average (ensemble):
# Find the cutoff for average (ensemble): that maximizes both sensitivity and overall accuracy

# Find the best ensemble cut-off:
cut_off.ens = seq(0.01,0.99,0.01)
metrics.ens <- data.frame(cut_off.ens, sensitivity = rep(0, length(cut_off.ens)), accuracy = rep(0, leng

for (i in 1:length(cut_off.ens)){
  x <- as.factor(ifelse(results.df$average.prob >= cut_off.ens[i],1,0))
  metrics.ens[i, 2] <- confusionMatrix(x, results.df$actual, positive = "1")$byClass[1]
  metrics.ens[i, 3] <- confusionMatrix(x, results.df$actual, positive = "1")$overall[1]
}

cutoff.plot.ens = ggplot(data= metrics.ens) + geom_line(aes(x=cut_off.ens,y=sensitivity, color="Sensiti

cutoff.plot.ens
```



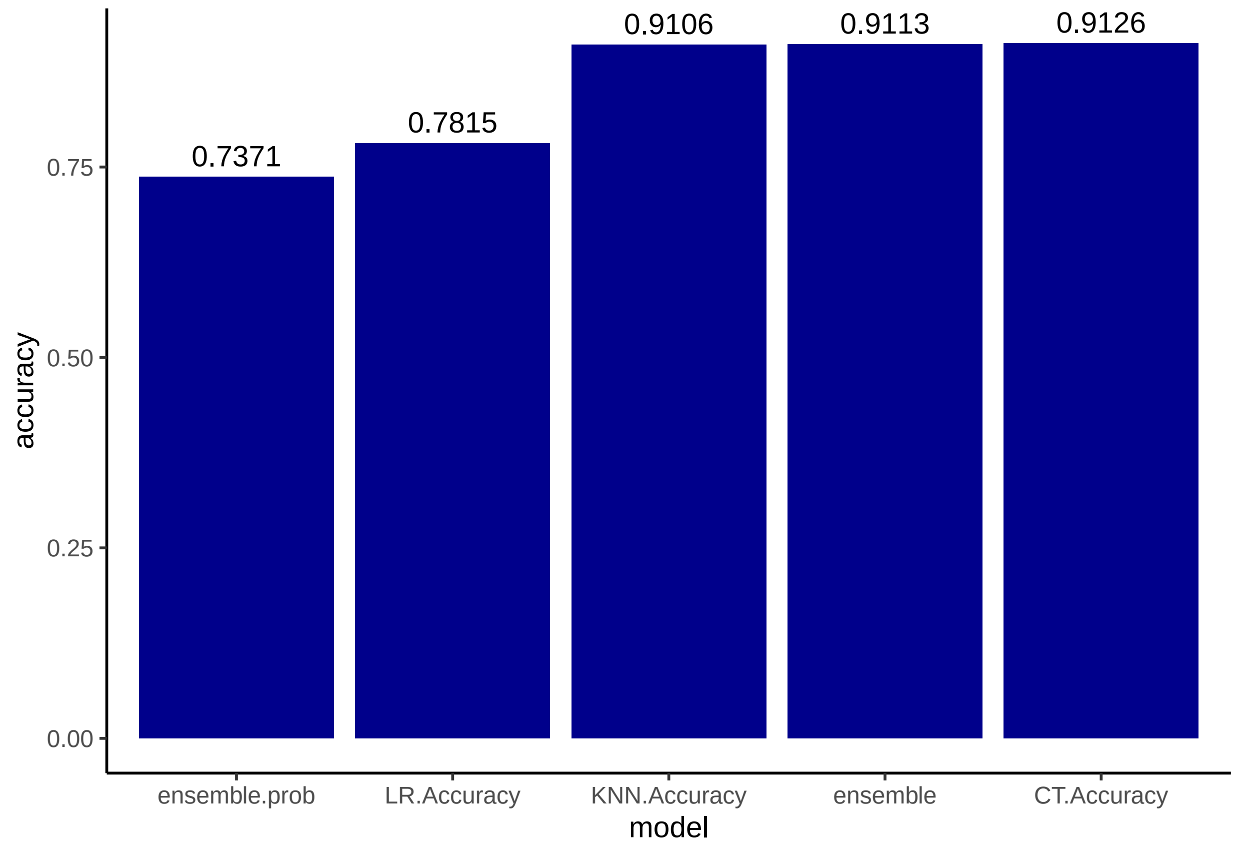
4) Ensemble methods

```
metrics.ens$difference = sqrt((metrics.ens$accuracy - metrics.ens$sensitivity)^2)
cut_off.best.ens = cut_off.ens[which.min(metrics.ens$difference)]
cut_off.best.ens
```

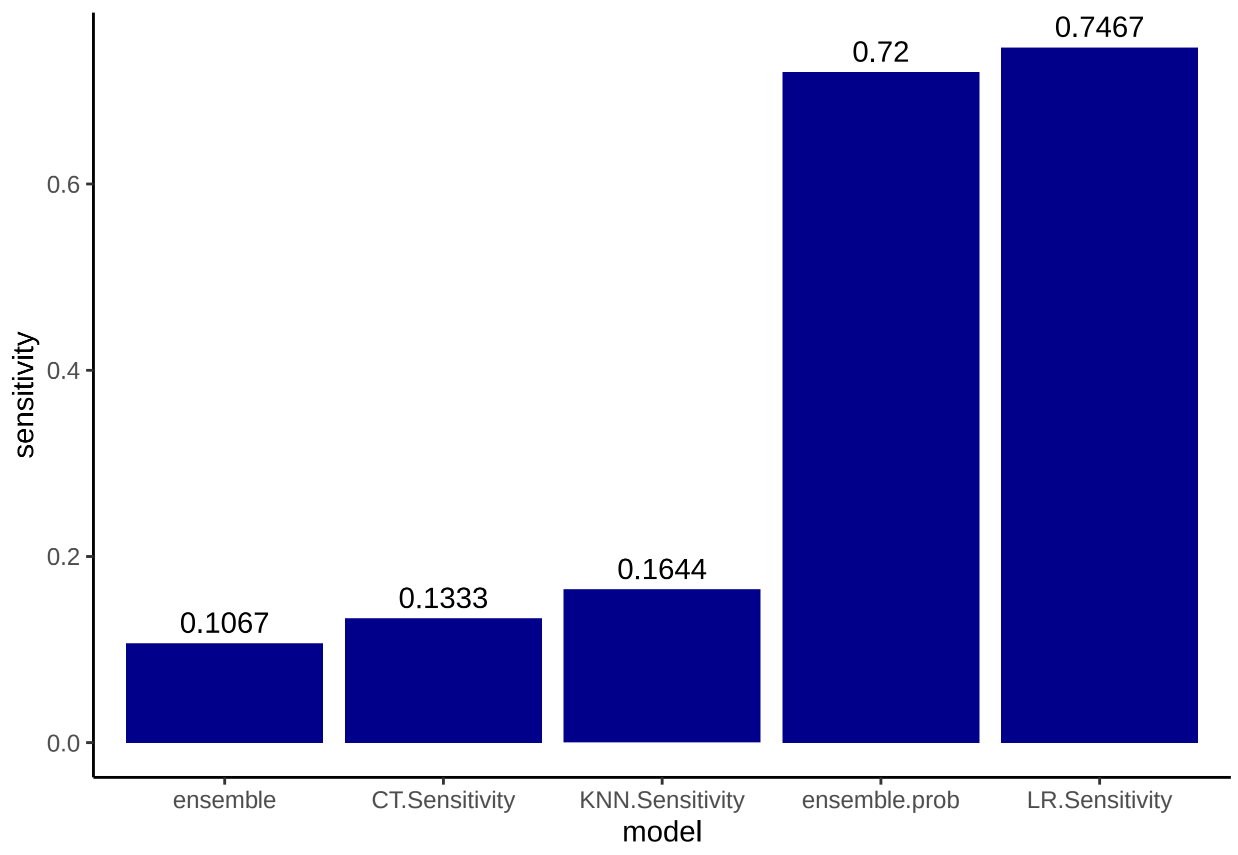
```
## [1] 0.03
```

```
accuracy.df = cbind(accuracy.df, ensemble.prob = confusionMatrix(as.factor(ifelse(results.df$average.pro
sensitivity.df = cbind(sensitivity.df, ensemble.prob = confusionMatrix(as.factor(ifelse(results.df$avera
```

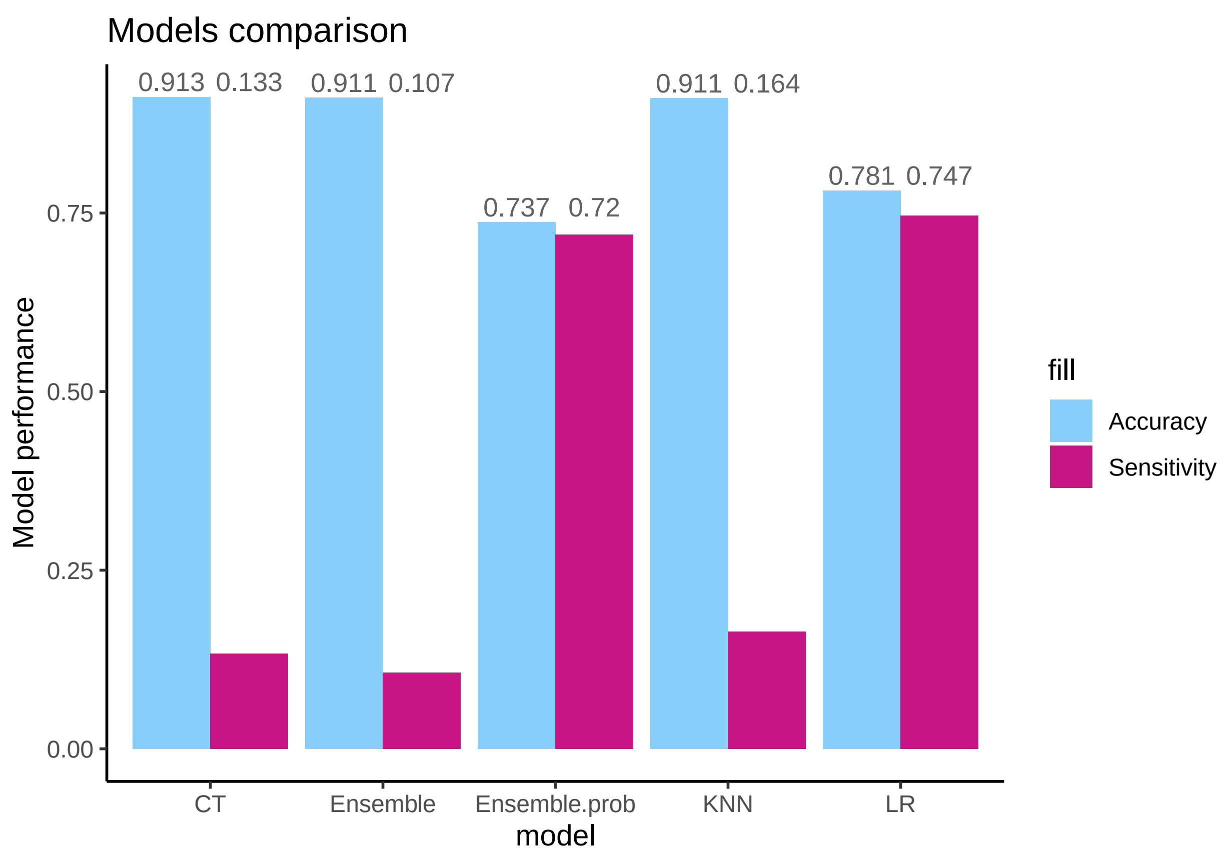
```
accuracy.df.melt = melt(accuracy.df)
accuracy.plot = ggplot(aes(x=reorder(variable, value), y=value), data = accuracy.df.melt) + geom_col(fil
geom_text(aes(label = round(value, 4)), vjust = -0.5) + theme_classic()
accuracy.plot
```



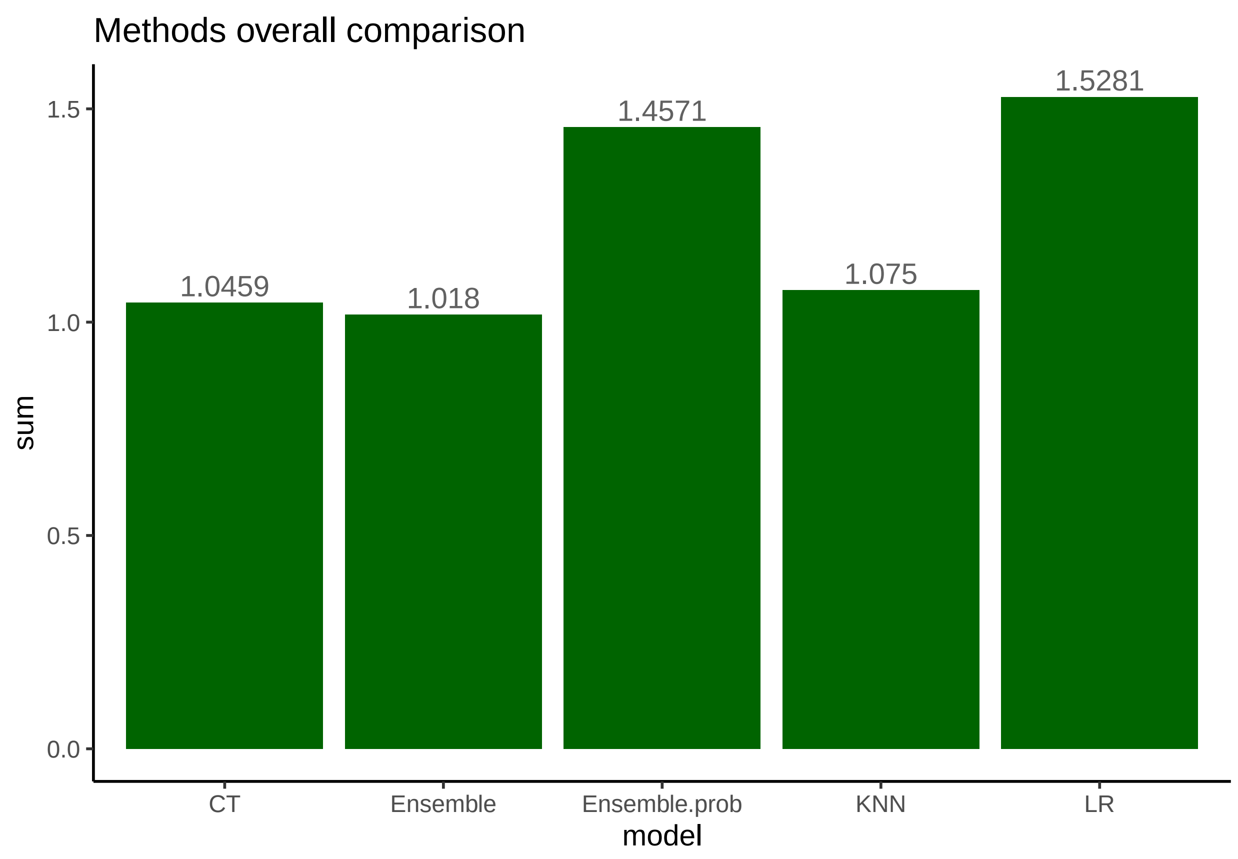
```
sensitivity.df.melt = melt(sensitivity.df)
sensitivity.plot = ggplot(aes(x=reorder(variable, value), y=value), data = sensitivity.df.melt) + geom_c
geom_text(aes(label = round(value, 4)), vjust = -0.5) + theme_classic()
sensitivity.plot
```



```
both.melt = accuracy.df.melt
both.melt[,1] = c("LR", "CT", "KNN", "Ensemble", "Ensemble.prob")
both.melt[,3] <- sensitivity.df.melt$value
colnames(both.melt) <- c("method", "accuracy", "sensitivity")

both.melt.plot = ggplot(both.melt, aes(x = method, y= accuracy)) + geom_col(aes(fill = "Accuracy"), wid
  geom_col(aes(y = sensitivity, fill = "Sensitivity"), width = 0.45, position = position_nudge(x = 0.225
  ylab("Model performance") + xlab("model") + ggtitle("Models comparison") + scale_fill_manual(values =
both.melt.plot
```



```
acc = c(accuracy.df[1,1], accuracy.df[1,2], accuracy.df[1,3], accuracy.df[1,4], accuracy.df[1,5])
sens = c(sensitivity.df[1,1], sensitivity.df[1,2], sensitivity.df[1,3], sensitivity.df[1,4], sensitivit
method = c("LR", "CT", "KNN", "Ensemble", "Ensemble.prob")
both.df = data.frame(method, "accuracy"= acc, "sensitivity"=sens)

both.df$sum = both.df$accuracy + both.df$sensitivity

#barplot(sum ~ method, both.df, main = "Methods overall comparison", ylab = "accuracy + sensitivity", x

both.plot = ggplot(aes(x=method, y=sum), data = both.df) + geom_col(fill = "darkgreen") + ylab("sum") +
both.plot
```



COMPARISON OF ALL METHODS: As the results of the full models of the CT and KNN are so low in sensitivity, there are inadequate for this highly unbalanced dataset. Therefore, our assumption was that doing a ensemble method by majority & average would be counterproductive, as the results yield would be lower than the logistic regression. The ensemble prob method, that allows for a cutoff, is actually much better than we thought. Its balanced predictive power is good, though still lower than LR. => Blindly, ensemble methods allow for a good result.

5) Best model: Prediction LR is the best fitted model => use this method as the basis for prediction.

```
set.seed(1)
# Reduced GLM
logit.reg2 <- glm(stroke ~ age + hypertension + heart_disease + avg_glucose_level , data = train.lr, fa
options(scipen = 999)
summary(logit.reg2)
```

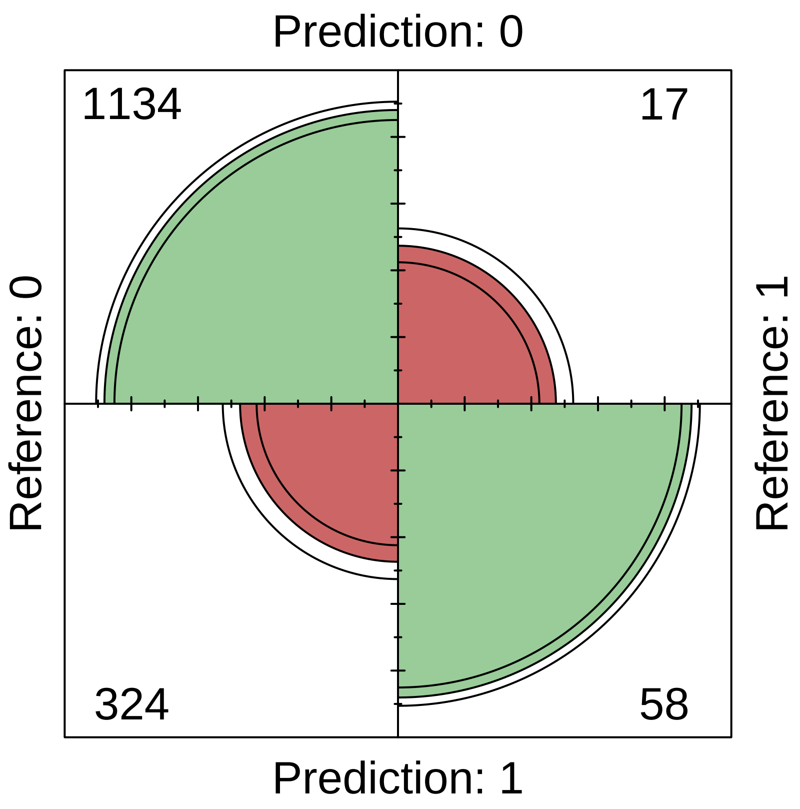
```
##
## Call:
## glm(formula = stroke ~ age + hypertension + heart_disease + avg_glucose_level,
##     family = "binomial", data = train.lr)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2155  -0.3263  -0.1693  -0.0782   3.7872
##
## Coefficients:
```

```
##                     Estimate Std. Error z value             Pr(>|z|)
## (Intercept)       -7.475759   0.512806 -14.578 < 0.0000000000000002 ***
## age                0.069560   0.007352   9.462 < 0.0000000000000002 ***
## hypertension       0.656781   0.218815   3.002              0.00269 **
## heart_disease      0.604553   0.245133   2.466              0.01365 *
## avg_glucose_level  0.003033   0.001654   1.834              0.06663 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1033.36  on 2553  degrees of freedom
## Residual deviance:  809.96  on 2549  degrees of freedom
## AIC: 819.96
##
## Number of Fisher Scoring iterations: 7
```

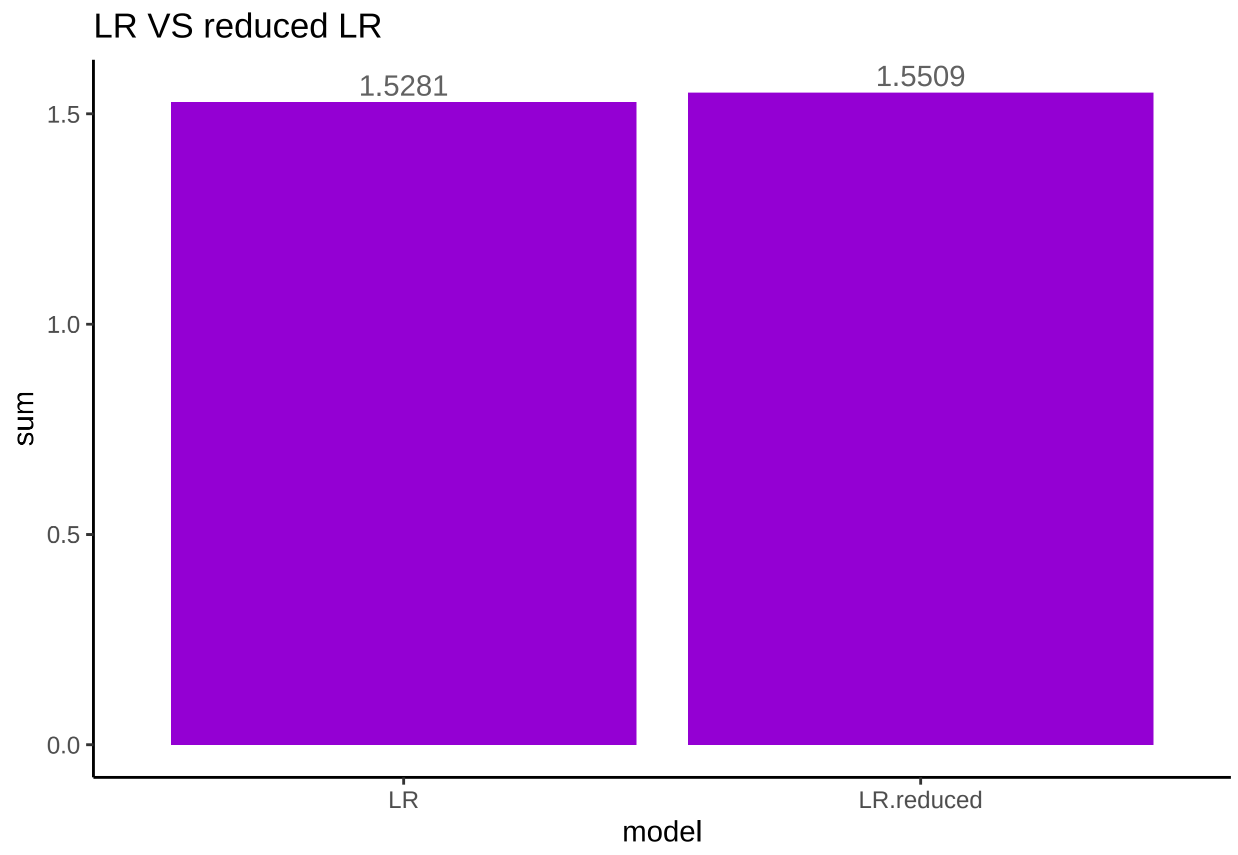
```
pred.lr2 = predict(logit.reg2, valid.lr[,-10], type = "response")
# Confusion matrix with best cut-off (0.4):
conf.lr2 = confusionMatrix(as.factor(ifelse(pred.lr2 > cut_off.best, 1, 0)), valid.lr$stroke, positive =
conf.lr2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1134   17
##          1  324   58
##
##                Accuracy : 0.7776
##                  95% CI : (0.7559, 0.7982)
##     No Information Rate : 0.9511
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1874
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.77333
##             Specificity : 0.77778
##          Pos Pred Value : 0.15183
##          Neg Pred Value : 0.98523
##              Prevalence : 0.04892
##          Detection Rate : 0.03783
##    Detection Prevalence : 0.24918
##       Balanced Accuracy : 0.77556
##
##        'Positive' Class : 1
##
```

```
fourfoldplot(confusionMatrix(as.factor(ifelse(pred.lr2 > cut_off.best, 1, 0)), valid.lr$stroke, positive
```



```
lr.plots = both.df[1,c(1,4)]
lr.plots[2,1] = c("LR.reduced")
lr.plots[2,2] = conf.lr2$overall[1]+conf.lr2$byClass[1]

lr.plots.plot = ggplot(aes(x=method, y=sum), data = lr.plots) + geom_col(fill = "darkviolet") + ylab("su
lr.plots.plot
```



The reduced model has more stroke prediction, for a number of less accurate non-stroke predictions. Keeping avg_glucose_level increases the overall accuracy-sensitivity mix, even though the variable is at the border of significance.

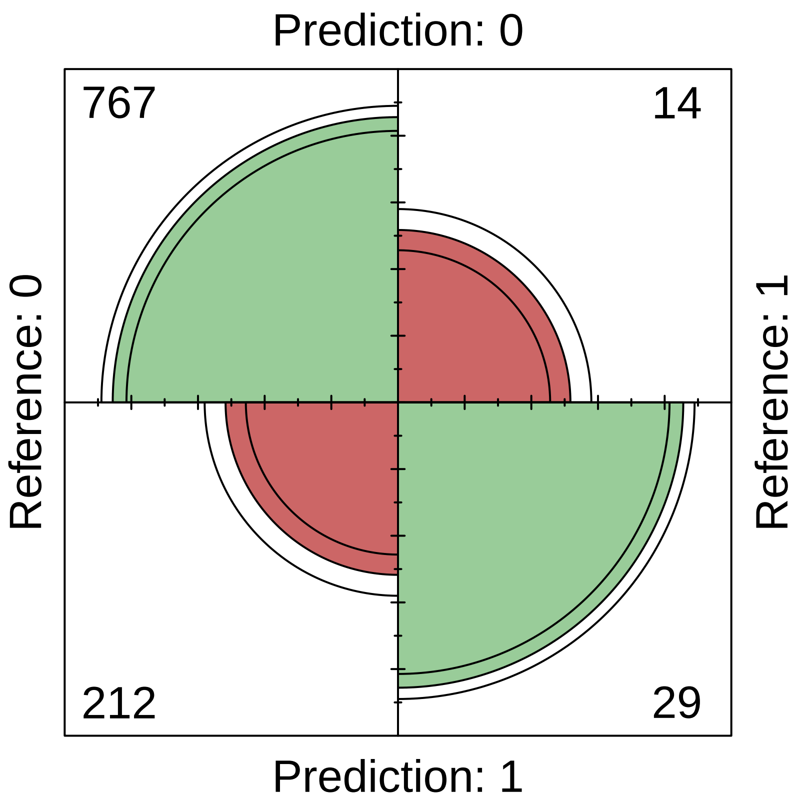
Test the model on test set

```
# LR on test set:
pred.test.lr = predict(logit.reg, test.lr[,-10], type = "response")
conf.test.lr = confusionMatrix(as.factor(ifelse(pred.test.lr > cut_off.best, 1, 0)), test.lr$stroke, po
conf.test.lr
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 767  14
##          1 212  29
##
##                Accuracy : 0.7789
##                  95% CI : (0.7521, 0.804)
##     No Information Rate : 0.9579
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.143
```

```
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.67442
##             Specificity : 0.78345
##          Pos Pred Value : 0.12033
##          Neg Pred Value : 0.98207
##              Prevalence : 0.04207
##          Detection Rate : 0.02838
##    Detection Prevalence : 0.23581
##       Balanced Accuracy : 0.72894
##
##        'Positive' Class : 1
##
```

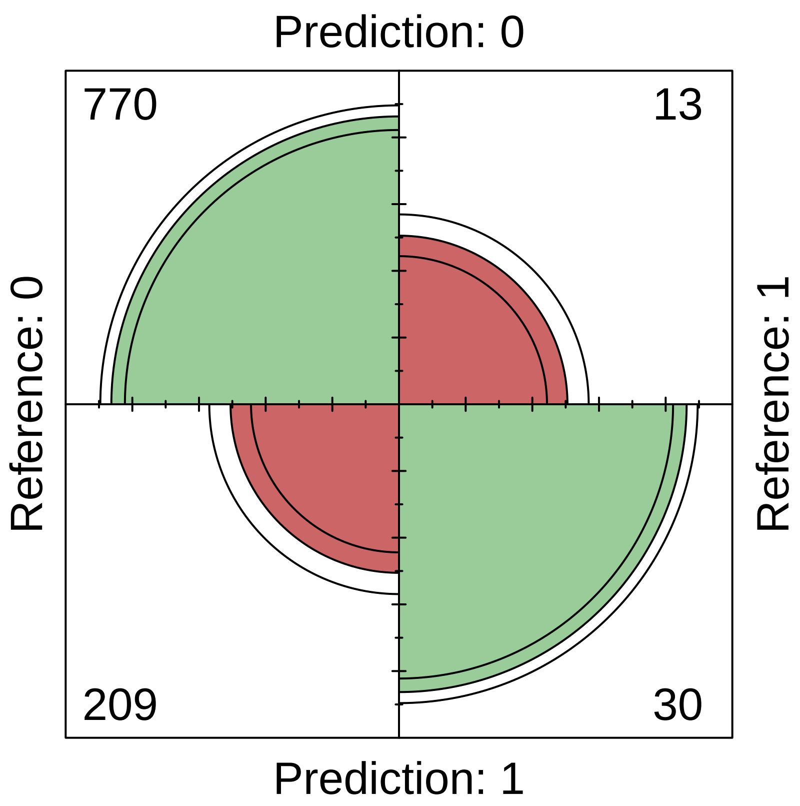
```
fourfoldplot(confusionMatrix(as.factor(ifelse(pred.test.lr > cut_off.best, 1, 0)), test.lr$stroke, posi
```



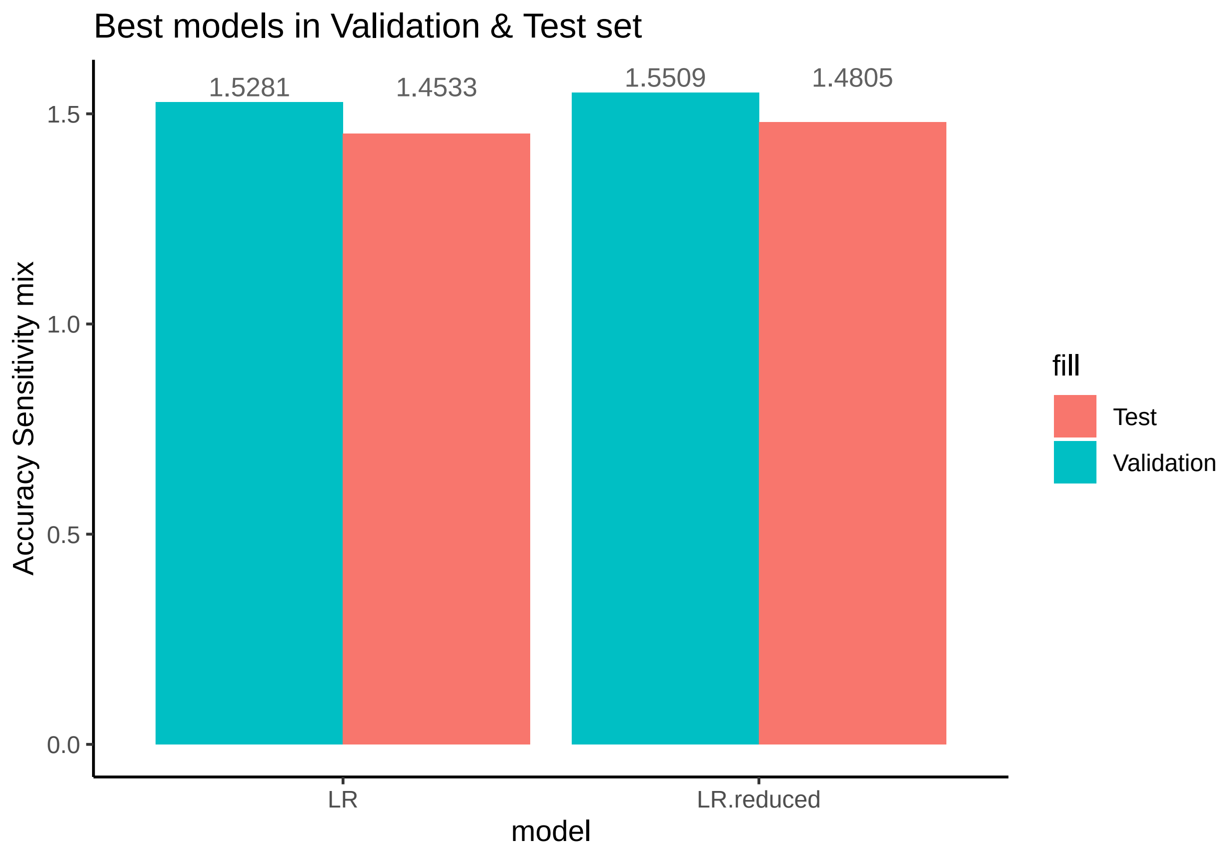
```
# Reduced LR on test set:
pred.test.lr2 = predict(logit.reg2, test.lr[,-10], type = "response")
conf.test.lr2 = confusionMatrix(as.factor(ifelse(pred.test.lr2 > cut_off.best, 1, 0)), test.lr$stroke, 
conf.test.lr2
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0   1
##          0 770  13
##          1 209  30
##
##                Accuracy : 0.7828
##                  95% CI : (0.7562, 0.8077)
##     No Information Rate : 0.9579
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1523
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.69767
##             Specificity : 0.78652
##          Pos Pred Value : 0.12552
##          Neg Pred Value : 0.98340
##              Prevalence : 0.04207
##          Detection Rate : 0.02935
##    Detection Prevalence : 0.23386
##       Balanced Accuracy : 0.74210
##
##        'Positive' Class : 1
##
```

```
fourfoldplot(confusionMatrix(as.factor(ifelse(pred.test.lr2 > cut_off.best, 1, 0)), test.lr$stroke, pos
```



```
lr.plots[1,3] = conf.test.lr$overall[1]+conf.test.lr$byClass[1]
lr.plots[2,3] = conf.test.lr2$overall[1]+conf.test.lr2$byClass[1]
colnames(lr.plots) <- c("method", "valid", "test")


lr.test.plots = ggplot(lr.plots, aes(x = method, y= valid)) + geom_col(aes(fill = "Validation"), width =
  geom_col(aes(y = test, fill = "Test"), width = 0.45, position = position_nudge(x = 0.225)) + geom_tex
  ylab("Accuracy Sensitivity mix") + xlab("model") + ggtitle("Best models in Validation & Test set") + 


lr.test.plots
```

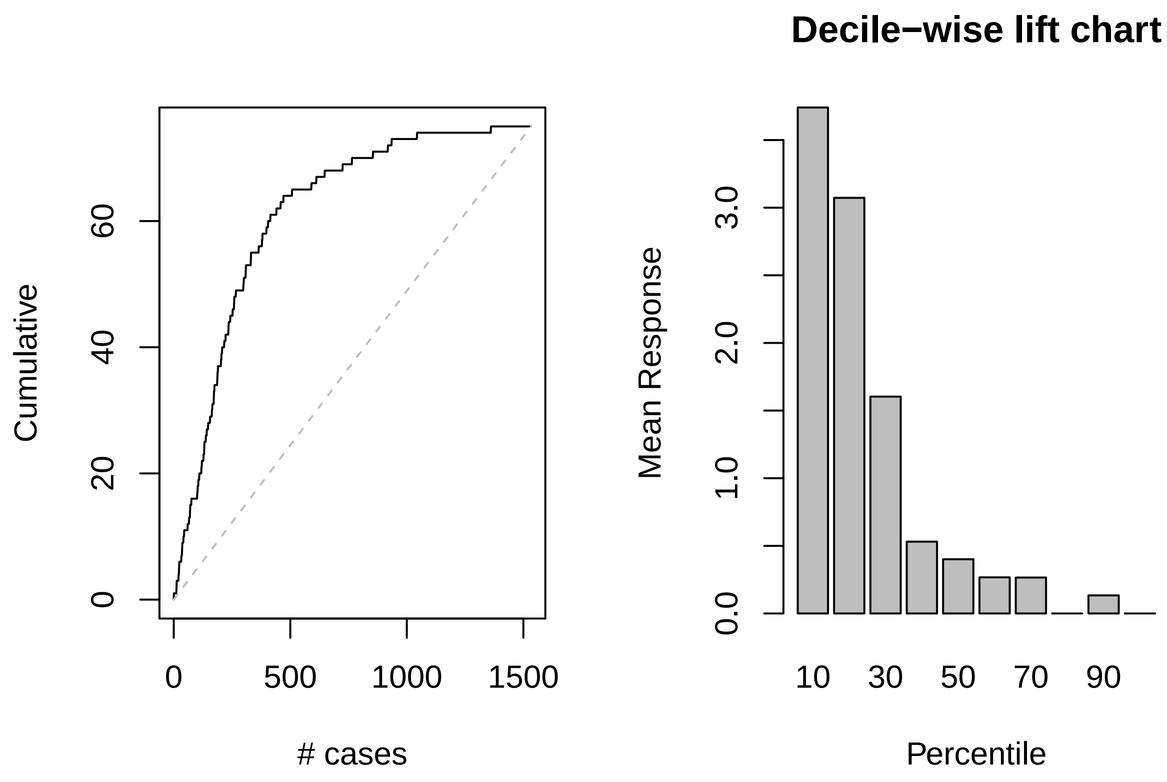


Gain & decile for best model

We want to know the performance of the best reduced LR model

```
outcome = data.frame(actual = valid.lr$stroke, prob = pred.lr2)

lift <- lift(actual ~ prob, data = outcome)

gain <- gains(predicted = outcome$prob, actual = as.numeric(as.character(outcome$actual)), groups=dim(ou

gain.decile <- gains(predicted = outcome$prob, actual = as.numeric(as.character(outcome$actual)))
```

```
par(mfrow=c(1,2))

plot(c(0, gain$cume.pct.of.total*sum(as.numeric(as.character(outcome$actual)))) ~ c(0, gain$cume.obs),
xlab = "# cases", ylab = "Cumulative", type="l")
lines(c(0,sum(as.numeric(as.character(outcome$actual))))~c(0,dim(outcome)[1]), col="gray", lty=2)

barplot(gain.decile$mean.resp / mean(as.numeric(as.character(outcome$actual))), names.arg = gain.decile
```

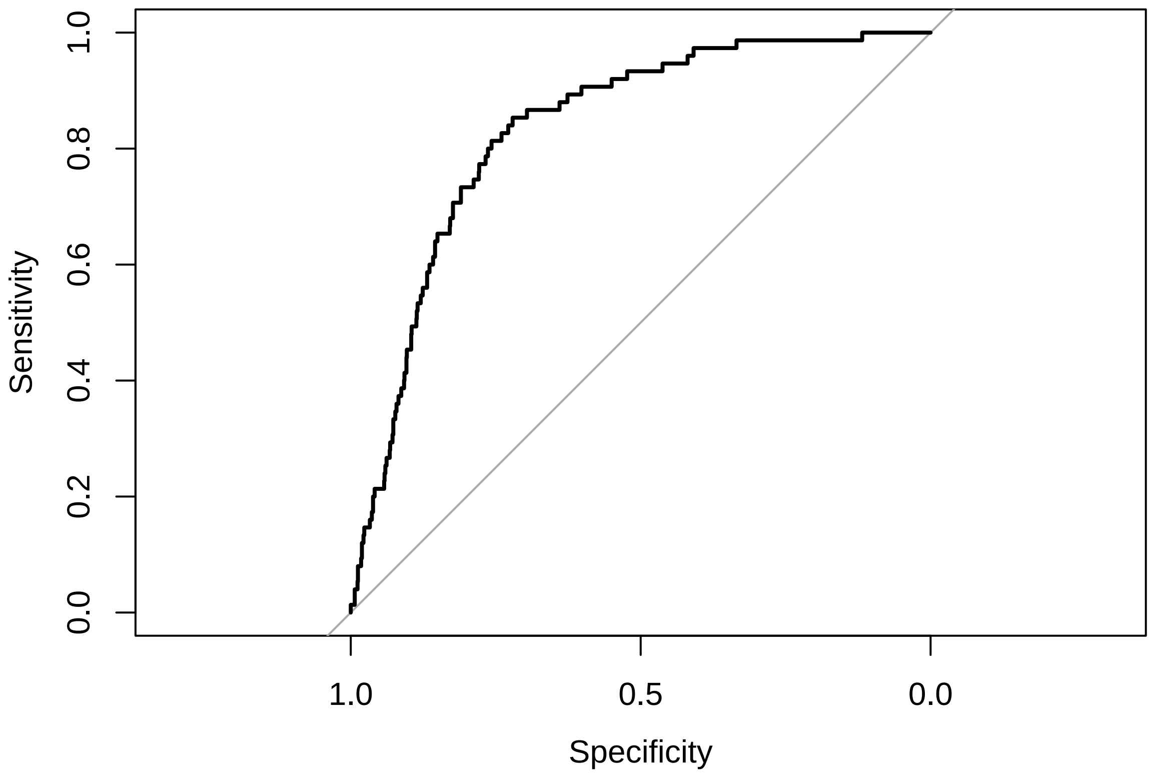


```
# ROC curve
ROC <- roc(outcome$actual, outcome$prob)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot.roc(ROC)
```



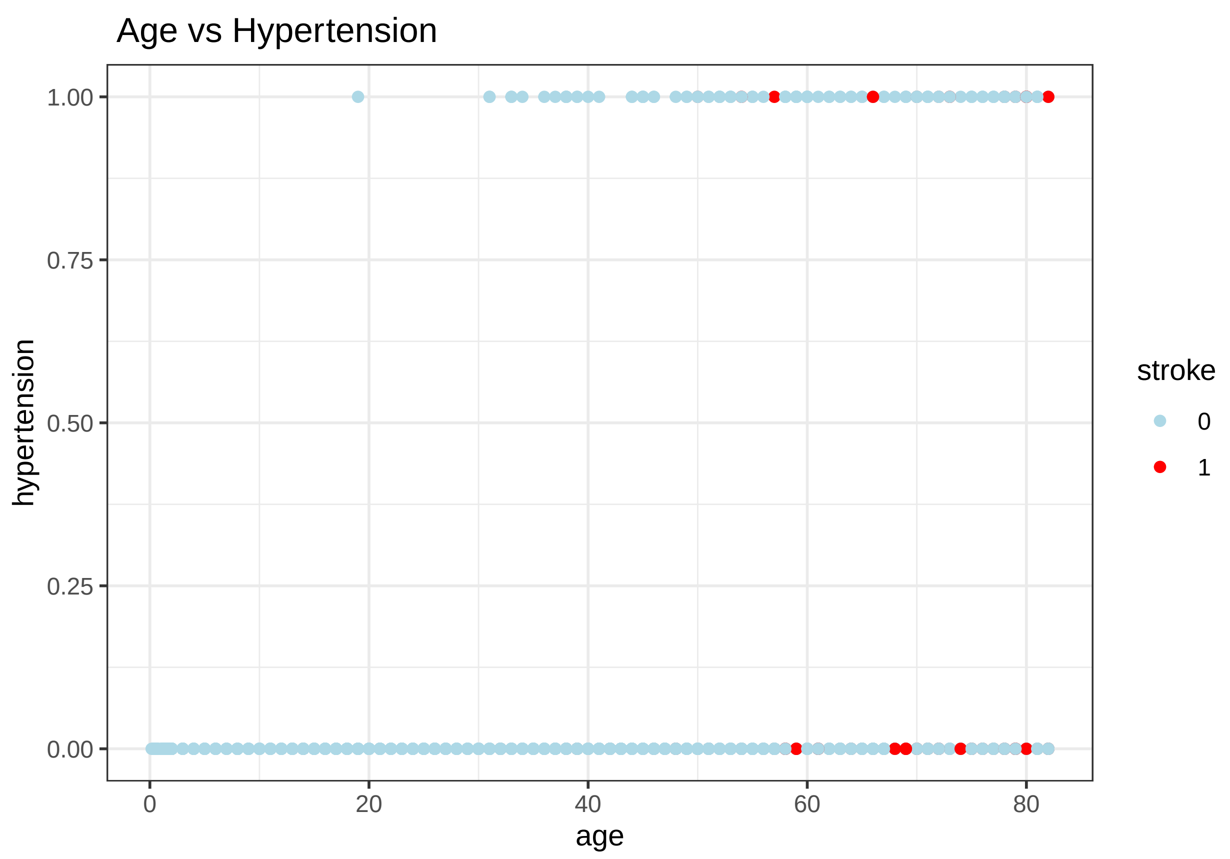
```
# Compute AUC
auc(ROC)
```

```
## Area under the curve: 0.8336
```

The closer the AUC to 1, the better the performance of the model. In our case where sensitivity is more important than specificity, our best reduced LR model does a good job at predicting the majority of stroke without penalizing the overall accuracy.

6) LDA - Linear Discriminant Analysis The significant predictors are in order (from most influencer to least) age, hypertension, heart_disease, and avg_glucose_level.

```
# Draw a colored scatterplot of age and avg_glucose_level:
plot.LDA <- ggplot(data=valid.lr, aes(x=age, y=hypertension)) +
    geom_point(aes(color=stroke)) + labs(title = " Age vs Hypertension") + scale_color_manual(values = 
plot.LDA
```



The higher the age, the higher the occurrences of strokes (more red dots on the right). In case of high glucose level, the likelihood of strokes is also higher (a bit more red dots top right).

```r
set.seed(1)
# 1) Compute the LDA model using lda() function
lda <- lda(formula = stroke ~ age + hypertension, data = valid.lr)
lda
```

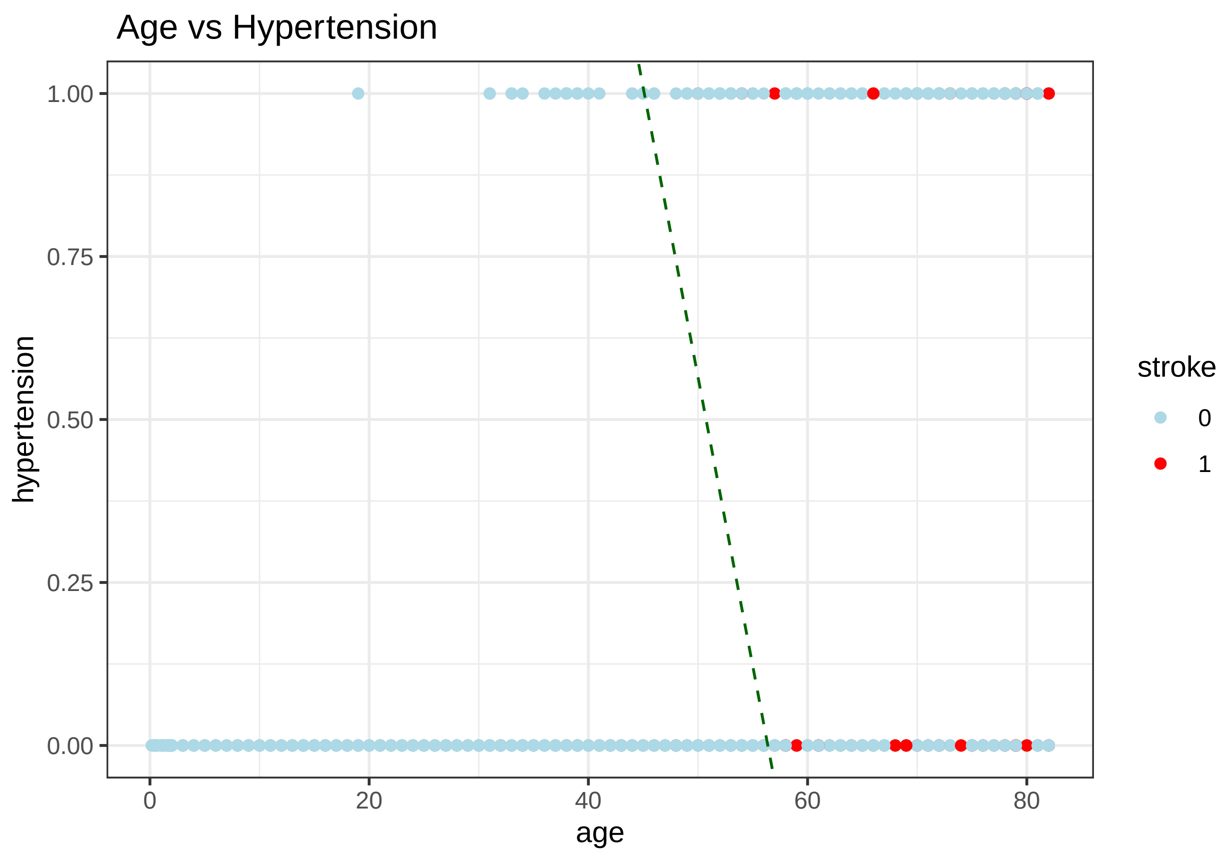
```
## Call:
## lda(stroke ~ age + hypertension, data = valid.lr)
##
## Prior probabilities of groups:
##          0          1
## 0.95107632 0.04892368
##
## Group means:
##        age hypertension
## 0 41.88702   0.08367627
## 1 67.50667   0.21333333
##
## Coefficients of linear discriminants:
##                     LD1
## age          0.04368914
## hypertension 0.49233089
```

```
set.seed(1)
# 2) Compute LDA with linDA() function:
linDA <- linDA(valid.lr[,c(2,3)], valid.lr[,10], prior= c(ratio_0, ratio_1))
linDA
```

```
##
## Linear Discriminant Analysis
## -------------------------------------------
## $functions        discrimination functions
## $confusion        confusion matrix
## $scores           discriminant scores
## $classification   assigned class
## $error_rate       error rate
## -------------------------------------------
##
## $functions
##                0        1
## constant      -1.9071  -7.7924
## age            0.0901   0.1418
## hypertension  -0.7375  -0.1550
##
##
## $confusion
##         predicted
## original     0     1
##        0  1458     0
##        1    75     0
##
##
## $error_rate
## [1] 0.04892368
##
##
## $scores
##                0           1
## 4119  1.7889121  -1.9771291
## 1380  1.6086201  -2.2608013
## 1050  0.8874523  -3.3954902
## 1708  0.1662844  -4.5301790
## 1087  2.6903719  -0.5587681
## 2566  2.9608099  -0.1332598
## ...
##
## $classification
## [1] 0 0 0 0 0 0
## Levels: 0 1
## ...
```

```
# Estimate DA line (book page 300)
# 1) we must calculate the difference between each classification function coefficients
a_age = linDA$functions[2,2] - linDA$functions[2,1]
a_avg_glucose = linDA$functions[3,2] - linDA$functions[3,1]
# 2) compute the value of intercept and slope
```

```
intercept.DA = a_age/a_avg_glucose * (lda$means[1,1] + lda$means[2,1])/2 + (lda$means[1,2] + lda$means[
  #(mean(system_admin$Experience) + mean(system_admin$Training))
slope.DA = - a_age/a_avg_glucose

plot.LDA <- plot.LDA + geom_abline(intercept = intercept.DA, slope = slope.DA, color = "dark green", li
plot.LDA
```



Since the cost of misclassifying a stroke is different from the cost of misclassifying a non-stroke, we want to minimize the expected cost of misclassification rather than the simple error rate (which does not account for unequal misclassification costs).

- q(0) the cost of misclassifying a class 0 member (into class 1);
- q(1) the cost of misclassifying a class 1 member (into class 0);

The cost must be integrated into the constants of the classification functions by adding log(q1) to the constant for class 1 (book, page 303). Thus, the score for class 1 membership would increase: observations become more easily classified as stroke (= belonging to class 1) with the modified (increased) constant coefficients, in order to avoid the cost of a mistake (= misclassifying stroke as non-stroke).

To determine the relationship cost requires domain knowledge: is classifying a non-stroke as stroke 20, 30, 40 times higher, than not detecting it? In medical field, it is the false negatives that are dangerous.

xxx The end xxx