Classification models for stroke prediction

Felipe Maggi

23/7/2021

Abstract

According to the World Health Organization, strokes are responsible for approximately 11% of total deaths worldwide, and were the second leading cause of mortality in 2019, only behind ischaemic heart disease[1].

In this report we describe the process for training various classification models with the intention of predicting whether a patient is prone to stroke, and taking appropriate action before it occurs.

The best model, according to the selected comparison criteria, turned out to be Flexible Discriminant Analysis, together with a data balancing method based on the random selection of variables that is supposed to guarantee the best estimates.

---

1. Introduction

The objective of this project is to find a model capable of predicting whether a patient has a tendency to suffer a stroke, based on the following variables:

1. Gender (binary)
2. Age (numerical)
3. Hypertension (binary)
4. Heart disease (binary)
5. Ever married (binary)
6. Type of residence (binary)
7. Average glucose level (numerical)
8. Body mass index (numerical)
9. Smoking status (categorical)

The selected model must have a high Sensitivity, without sacrificing too much Specificity.

1.1 data set description

The data set used (Stroke Prediction Data Data Set) contains 5,110 observations. It is available in Kaggle [2], and was uploaded to the platform by Federico Soriano (fedesorian) [3].

Originally, the variables and their respective classes are as follows:

```
str(stroke_data)
```

[1] *The top 10 causes of death.* https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death
[2] https://www.kaggle.com/fedesoriano/stroke-prediction-data set
[3] https://www.kaggle.com/fedesoriano

```
## '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 ...
```

In the following sections we will explain the changes that were made to the original data set, in order to apply the selected models.

To train the models, the final data set was divided into a training set and a test set, in a ratio of 80-20. We decided to reserve at least 20% of the observations for the test set because the data set is not excessively large and, as we will see, it is highly unbalanced. With a lower percentage of observations, we ran the risk of not having enough stroke cases in the test set, or that certain variables would see their proportions significantly affected.

Unfortunately, we think we don't have enough observations to reserve a part of the data set and use it as a validation set. Because of this, we have only worked with the training and test sets, and we have relied on cross-validation.

1.2 Goals and key steps

As we have already mentioned, the objective of this project is to train a model capable of predicting whether a patient has a high probability of suffering a stroke, taking into account the mentioned predictors.

The selected model must have a high Sensitivity, without sacrificing too much Specificity. In other words, we are looking for a model with a high Balanced Accuracy, compared to the rest.

As we will see, this will not be an easy task. The prevalence of negative class is staggering, and the predictor variables do not separate classes clearly.

Due to the nature of this project, we have trained several models with the original data and with balanced data. The underlying idea was to experiment with various algorithms, and compare their performance in this particular case.

In an exercise that could be described as brute force, six models have been trained with the unbalanced data, and nine models with each of the balancing methods.

The key steps in the development of this project were:

1. Treat the original data set, cleaning data and eliminating superfluous variables like "id".
2. Carry out an exploratory analysis of the data.
3. Determine the importance of the selected variables.
4. Center and scale the numeric variables.
5. Test provisional results, training and visualizing various models with the original unbalanced data set.
6. Balance the training set using various techniques.
7. Train several classification models with the different training sets obtained with the techniques used in the previous point.
8. Create ensembles of the best performing models for each balancing technique, and select a stroke prediction criterion based on the balanced accuracy metric.

2. Methods and Analysis

2.1 Data wrangling

During the development of this project, the data cleansing was carried out in several phases. In the first place, the variable "id" was eliminated, for not providing relevant information. At the same time, observations containing N/A (a problem affecting the BMI variable) were filtered, and categorical variables were factored (0 and 1 were converted to "No" or "Yes", respectively).

After eliminating the N/As the BMI variable was transformed into numeric (in the original data set it had been considered as a character by coercion).

Finally, the stroke variable was factored ("no_stroke", "stroke"), and the levels were reordered so that "stroke" became the positive class.

The resulting data set now contains 4,909 observations. The 201 deleted observations correspond to those in which BMI was reported as N/A:

```
# _________________________########
# DATA WRANGLING ########
# _________________________########

# Categorical and binary data as.factors #####

stroke_data <- stroke_data %>%
  filter(!bmi == "N/A")  %>% # filtering bmi = N/A
  mutate(gender = as.factor(gender),
         hypertension = as.factor(ifelse(hypertension == 0, "No", "Yes")),
         heart_disease = as.factor(ifelse(heart_disease == 0, "No", "Yes")),
         ever_married = as.factor(ever_married),
         work_type = as.factor(work_type),
         Residence_type = as.factor(Residence_type),
         bmi = as.numeric(bmi),
         smoking_status = as.factor(smoking_status),
         stroke = as.factor(ifelse(stroke == 1, "stroke", "no_stroke"))) %>%
  select(!id) # Removing "id" variable

# Relevel "stroke" "no_stroke" factors: positive class: "stroke" ####
# It makes "stroke" the positive class, in order to facilitate interpretations

stroke_data$stroke <- relevel(stroke_data$stroke, ref = "stroke")

str(stroke_data)
```

```
## 'data.frame':    4909 obs. of  11 variables:
##  $ gender           : Factor w/ 3 levels "Female","Male",..: 2 2 1 1 2 2 1 1 1 1 ...
##  $ age              : num  67 80 49 79 81 74 69 78 81 61 ...
##  $ hypertension     : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 1 1 2 1 ...
##  $ heart_disease    : Factor w/ 2 levels "No","Yes": 2 2 1 1 1 2 1 1 1 2 ...
##  $ ever_married     : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 1 2 2 2 ...
##  $ work_type        : Factor w/ 5 levels "children","Govt_job",..: 4 4 4 5 4 4 4 4 4 2 ...
##  $ Residence_type   : Factor w/ 2 levels "Rural","Urban": 2 1 2 1 2 1 2 2 1 1 ...
##  $ avg_glucose_level: num  229 106 171 174 186 ...
##  $ bmi              : num  36.6 32.5 34.4 24 29 27.4 22.8 24.2 29.7 36.8 ...
##  $ smoking_status   : Factor w/ 4 levels "formerly smoked",..: 1 2 3 2 1 2 2 4 2 3 ...
##  $ stroke           : Factor w/ 2 levels "stroke","no_stroke": 1 1 1 1 1 1 1 1 1 1 ...
```

During the exploration, we discovered an observation whose variable "gender" had the value of "Other". This observation was also removed. The hypothesis here is that this observation would add noise if gender turns out to be an important variable. We do not want this to be misunderstood. Filtering this class is for training purposes only. A single observation can generate significant changes in the output of the models (something we check). The author unconditionally respects any sexual preference.

Before starting to train models, the numerical variables were scaled and centralized.

The last modifications made to the data set was to balance the "stroke" and "no_stroke" classes, with different methods. We will see the details later.

2.2 Data exploration

For certain parts of this exploratory analysis we have followed the recommendations of Joaquín Amat Rodrigo, exposed on the site *Machine Learning with R and caret*[4]. Specifically, with regard to:

1. The statistical data of the numerical variables
2. The Density and box plots
3. The distribution of the qualitative variables
4. The correlation between numerical variables
5. The contrast of proportions for categorical variables
6. The random forest method
7. The Near Zero Variance Analysis

2.2.1 Viariable distributions

The first thing that stands out during the exploration of the data, is the clear imbalance between the classes "stroke" and "no_stroke". To maintain the clarity of the exposition, and the correspondence with the code, we show the data before filtering the observation with gender "Other".

Var1	Freq
stroke	209
no_stroke	4700

Var1	Freq
stroke	0.0425749
no_stroke	0.9574251

Only 4% of the observations are labeled "stroke", so the prevalence of the "no_stroke" class is staggering (close to 96%).

This represents the first hurdle to overcome. With such a prevalence of the negative class, training any model keeping the proportions of the original data set will not give good results.

Obviously, the "accuracy" metric loses all sense. A model that predicts "no_stroke" in all cases will have an accuracy close to 96%. The important thing here is the Sensitivity of the model (remember that "stroke" is the positive class). But, at the same time, we want to obtain a model that has a correct Specificity. Predicting "stroke" in all cases would raise the Sensitivity to 100%, but would leave us with another useless model unable to correctly classify any negative cases.

Anyway, we will analyze the distribution of the variables in this data set, and then we will apply various methods to balance the training set. This step is necessary to check if the changes made to the training set

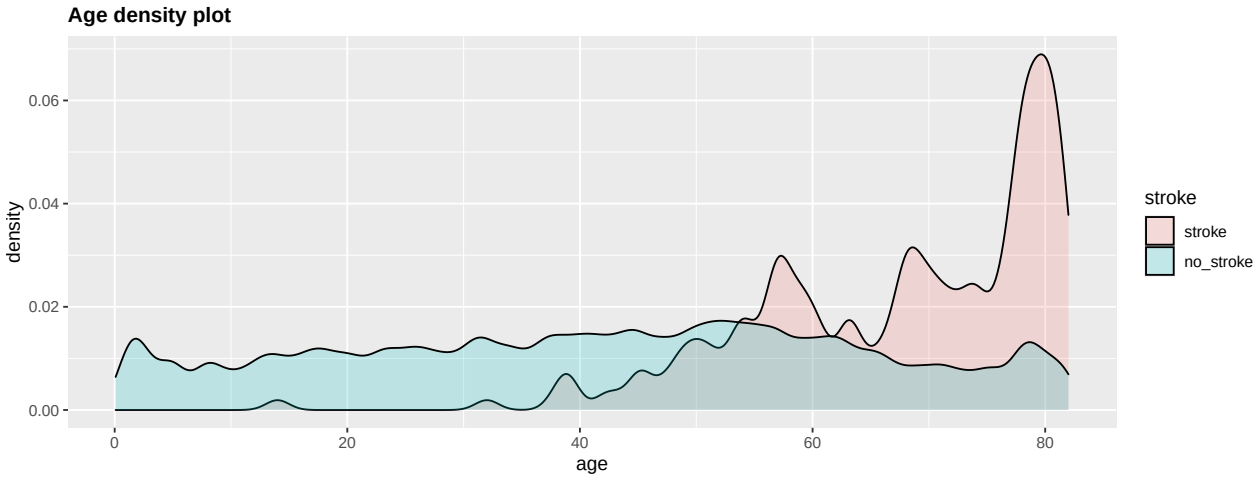
[4]Amat, Rodrigo: *Machine Learning con R y caret.* cienciadedatos.net, 2018: https://www.cienciadedatos.net/documentos/41_machine_learning_con_r_y_caret

do not alter the statistical data of the predictor variables.

2.2.1.1 Age The age statistics, depending on the response variables, are the following:

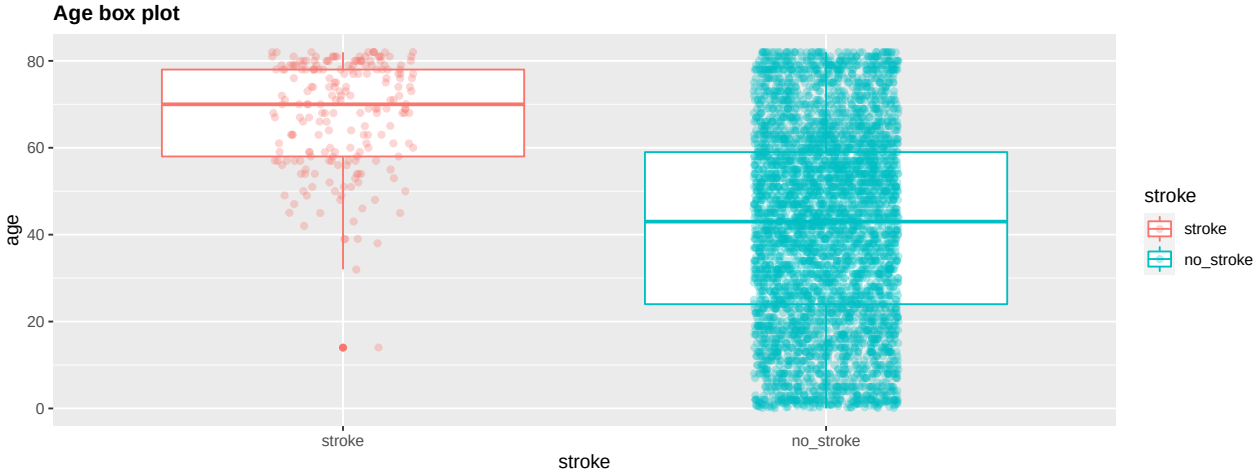
stroke	avg_age	median_age	min_age	max_age
stroke	67.7	70	14.00	82
no_stroke	41.8	43	0.08	82

It is not surprising that age, as we shall see, is the most important variable when classifying subjects according to "stroke" and "no_stroke". However, the overlap in the age range already indicates another obstacle that affect the models obtained from the selected data set. This overlap is clearly observed in the following density graph:



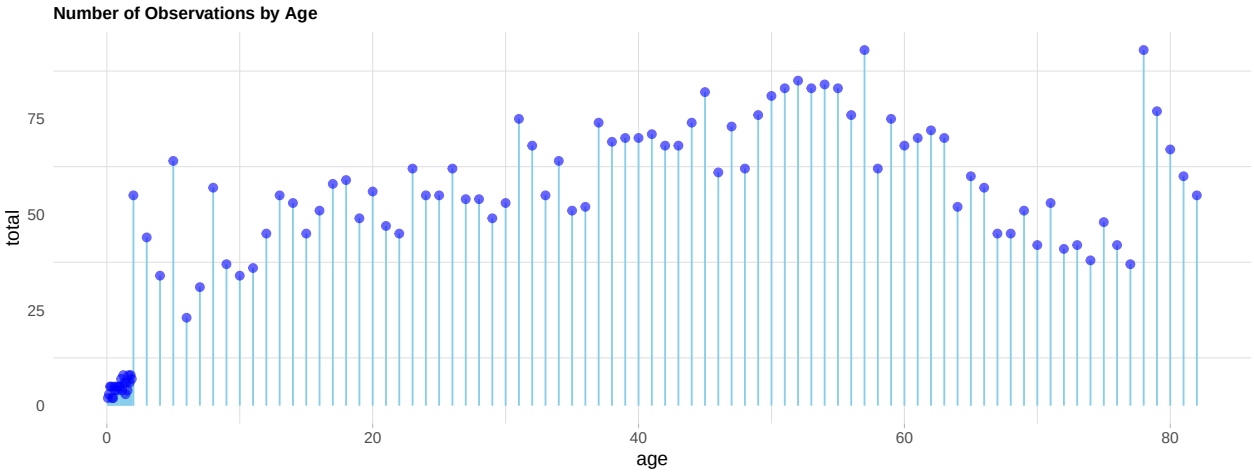
It is clear that age is an important factor, but as a variable it is far from dividing the observations clearly between one class and another.

The same is observed by means of box plots. As we are working with the original data set, the prevalence of the "no_stroke" class represented by the point density is also clearly visible.

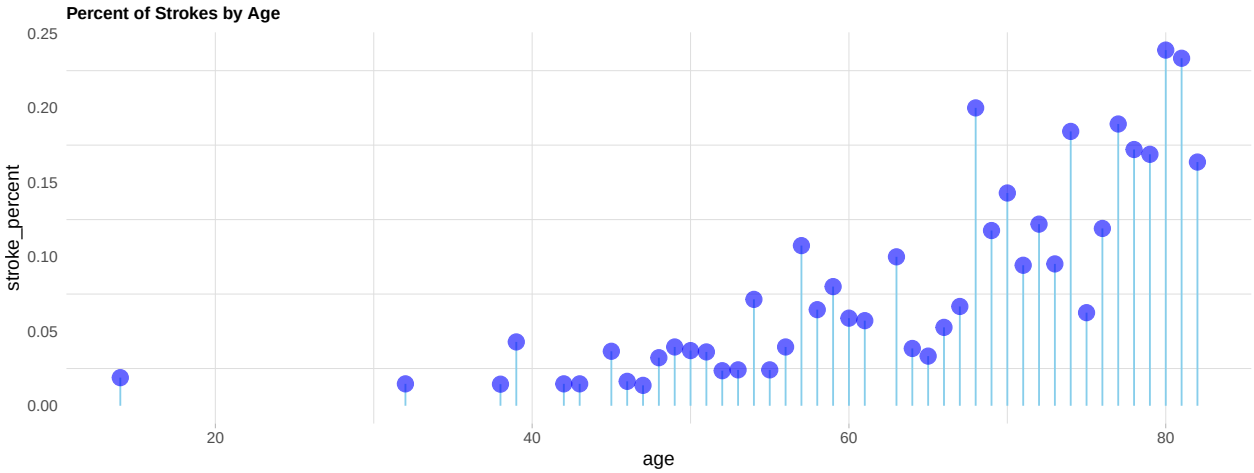


Regarding the number of observations according to age, we see that for very short ages (up to approximately 2 years old) there are few data. From two years old onwards we have, with few exceptions, over 35 observations.

The models were trained without grouping ages, and this tactic could be one of the changes to be tested to improve the results obtained.



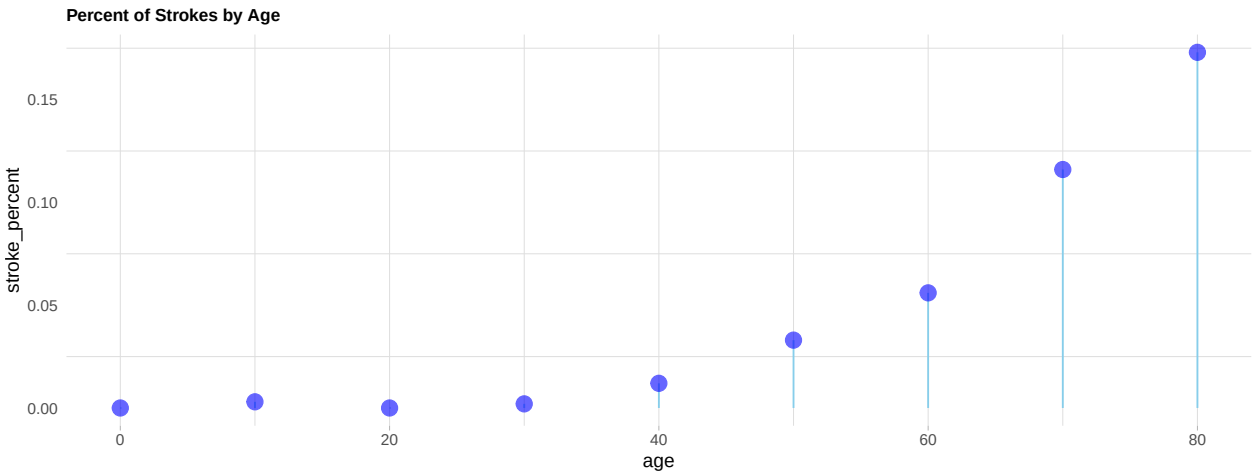
The percentage of strokes by age shows some correlation between this variable and the tendency to suffer from this disease:



As we have said, the models were trained with the ungrouped age data. If we group the ages to the nearest ten, the information acquires much more meaning and representativeness.

round_age	total	percent	strokes	stroke_percent
0	311	0.063	0	0.000
10	371	0.076	1	0.003
20	582	0.119	0	0.000
30	534	0.109	1	0.002
40	749	0.153	9	0.012
50	688	0.140	23	0.033
60	781	0.159	44	0.056
70	414	0.084	48	0.116
80	479	0.098	83	0.173

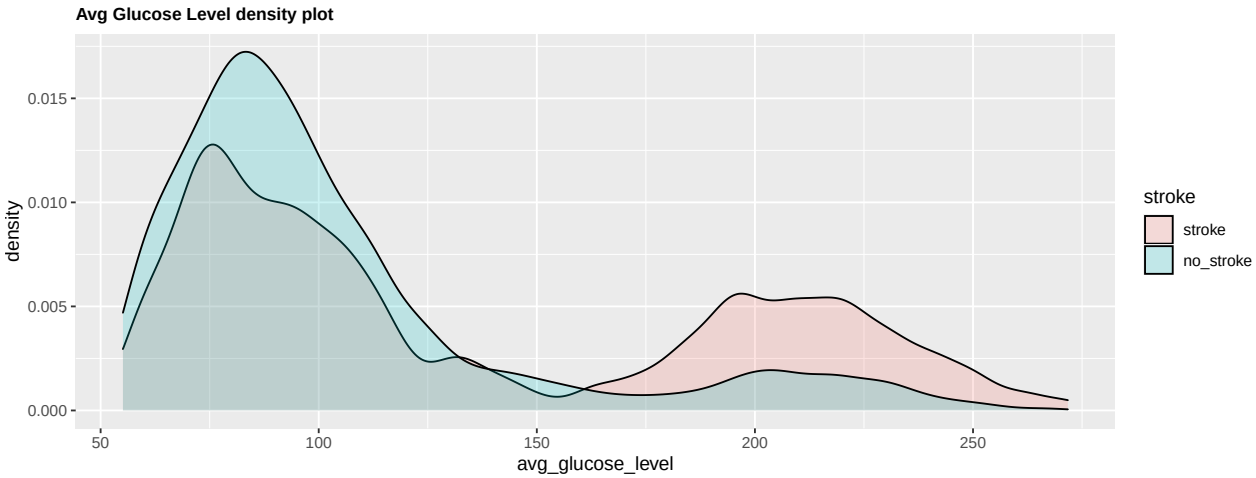
The correlation between age and incidence of strokes now looks much better:



Why, if knowing this, were the models trained the ungrouped age? First, to determine to what extent the models are able to deal with the original data and, second, to have improvement options.

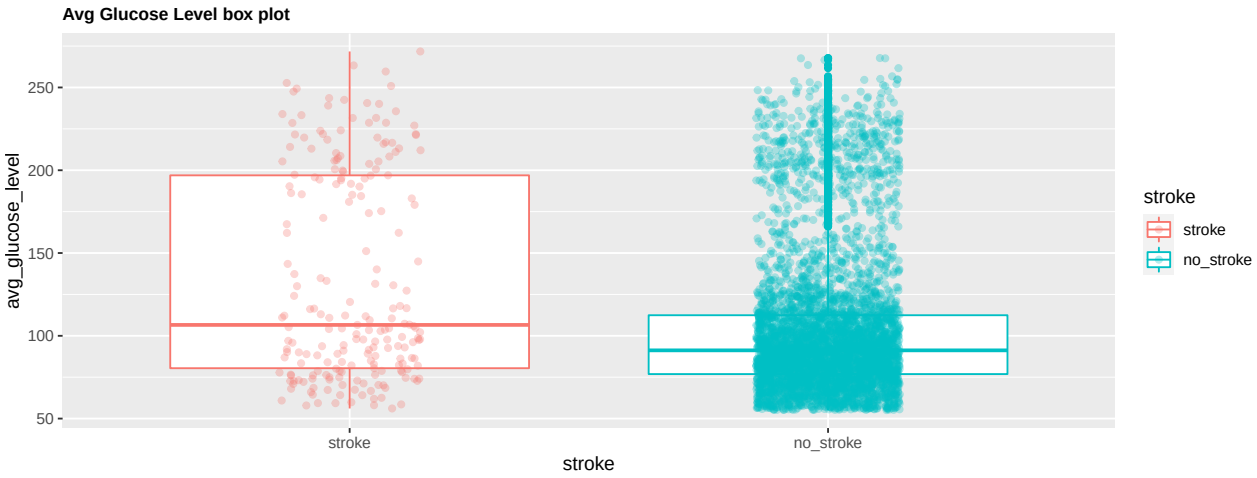
2.2.1.2 Average Glucose Level Some of the problems detected in the case of age are even more serious in the case of the average glucose level. There are minor differences in mean and median, but the ranges between minimum and maximum clearly overlap:

stroke	avg_glucose	median_glucose	min_glucose	max_glucose
stroke	134.6	107	56.11	271.74
no_stroke	104.0	91	55.12	267.76



In addition, we are facing a clear bimodal case, which affects both classes.

The box plot shows that the medians do not differ that much, and that the interquartile ranges overlap, despite having different sizes:

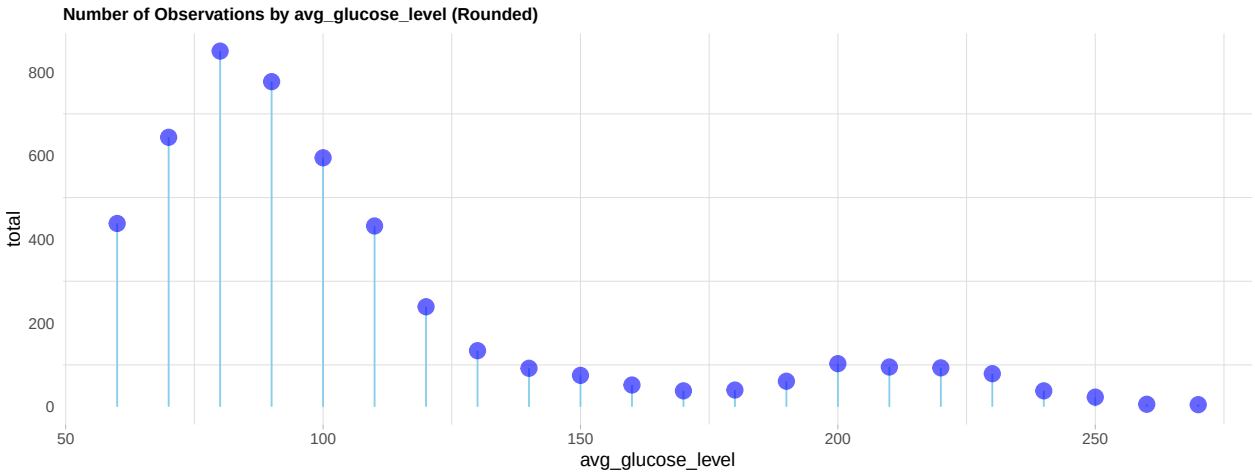


It is also important to highlight that with so few observations belonging to the "stroke" class, their variability is much greater, and it is perfectly possible that the data are not representative of the population. The latter is something that affects all variables, and balancing techniques do not solve. It is true that they increase the number of observations of the "stroke" class, but they do so by randomly (or according to best estimate criteria) repeating the available observations. Representativeness, or its lack of it, we understand that it is maintained.

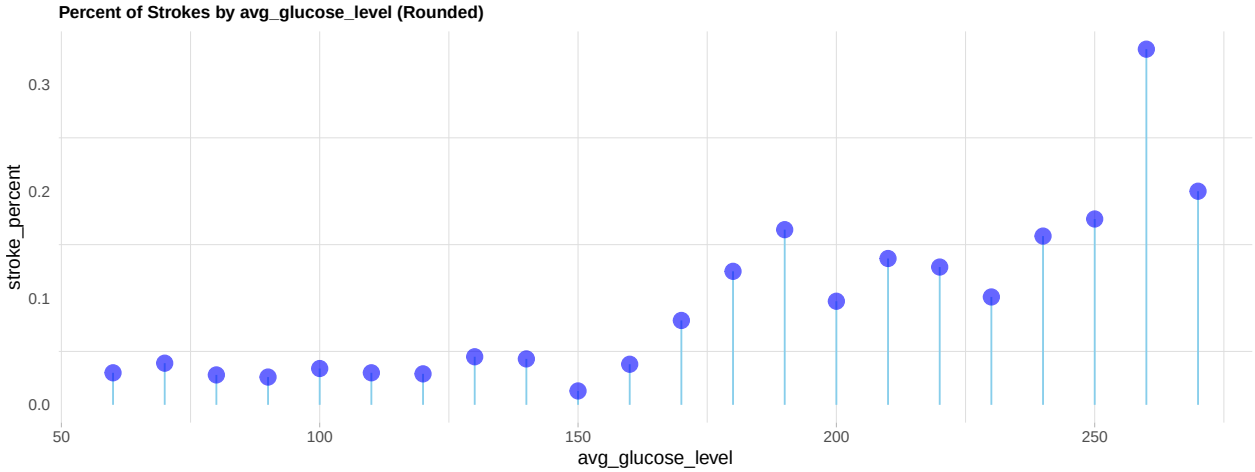
In this case we have rounded to the nearest ten directly, although the models have been trained with the ungrouped data:

round_avg_glucose_level	total	percent	strokes	stroke_percent
60	438	0.089	13	0.030
70	644	0.131	25	0.039
80	850	0.173	24	0.028
90	777	0.158	20	0.026
100	595	0.121	20	0.034
110	432	0.088	13	0.030
120	239	0.049	7	0.029
130	134	0.027	6	0.045
140	92	0.019	4	0.043
150	75	0.015	1	0.013
160	52	0.011	2	0.038
170	38	0.008	3	0.079
180	40	0.008	5	0.125
190	61	0.012	10	0.164
200	103	0.021	10	0.097
210	95	0.019	13	0.137
220	93	0.019	12	0.129
230	79	0.016	8	0.101
240	38	0.008	6	0.158
250	23	0.005	4	0.174
260	6	0.001	2	0.333
270	5	0.001	1	0.200

The number of observations according to the rounded glucose average are concentrated between 60 and 110, from there they are much scarcer, although they show a slight increase between 190 and 230:



Regarding the incidence of strokes according to rounded average glucose level, we observe a positive correlation (not as clear as in the case of age):

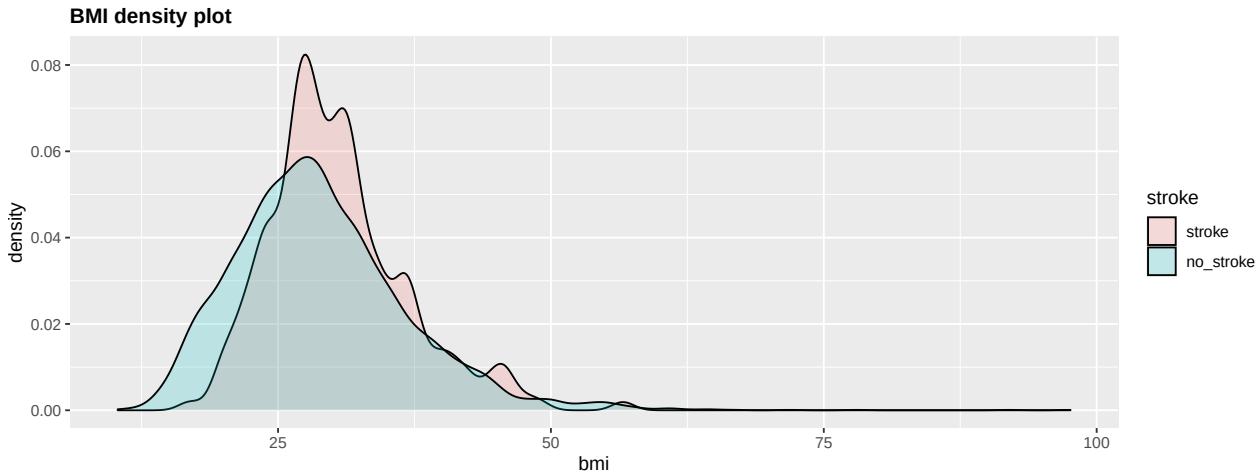


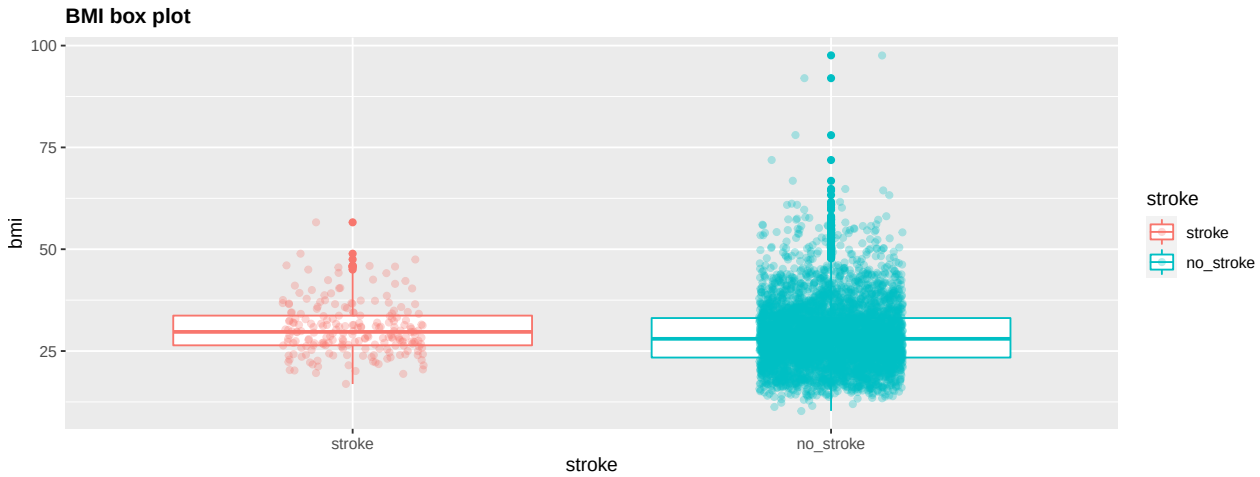
2.2.1.3 Body Mass Index The last of the numeric variables is the most complex, in the sense that it doesn't seem to separate the classes at all.

The mean and median are very similar, and the minimum-maximum range for the "no_stroke" class completely covers the range for the "stroke" class:

stroke	avg_bmi	median_bmi	min_bmi	max_bmi
stroke	30.5	30	16.9	56.6
no_stroke	28.8	28	10.3	97.6

The density and box plots make this much clearer:





This variable, despite everything, seems to be important according to the random forest variable selection method (Amat, 2018), which we will see later.

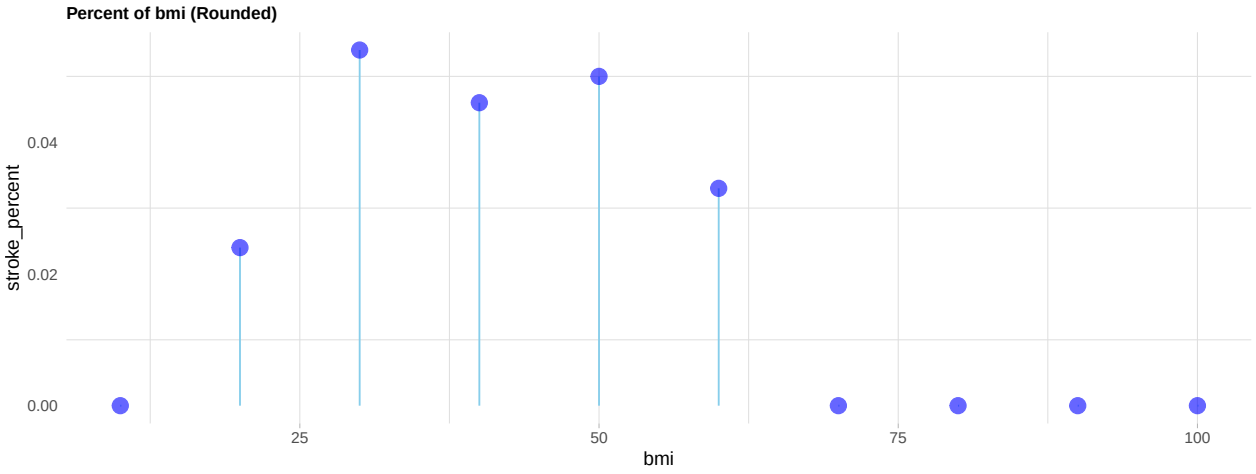
In any case, the random forest model did not yield good results with any of the balancing techniques. A possible course of action when it comes to improving the models is to test the training without this variable.

The BMI table rounded to the nearest ten shows very few observations starting at 70, and that the incidence is concentrated between 20 and 40.

round_bmi	total	percent	strokes	stroke_percent
10	42	0.009	0	0.000
20	1565	0.319	38	0.024
30	2382	0.485	129	0.054
40	765	0.156	35	0.046
50	120	0.024	6	0.050
60	30	0.006	1	0.033
70	2	0.000	0	0.000
80	1	0.000	0	0.000
90	1	0.000	0	0.000
100	1	0.000	0	0.000

In any case, we are talking about percentages ranging from 2% to 5%, which reinforces the hypothesis that

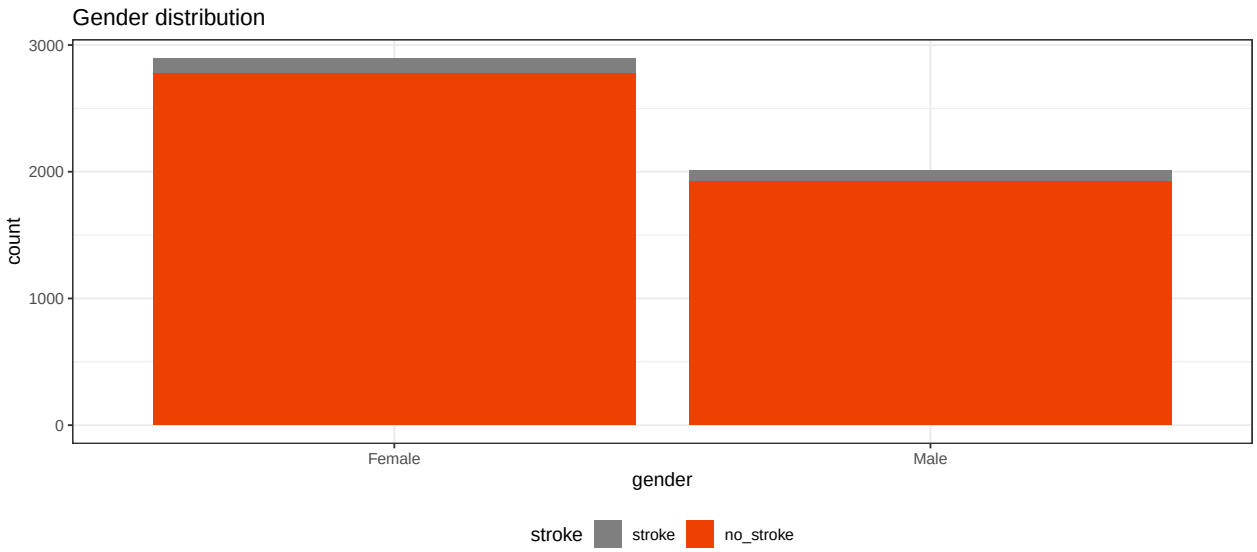
this variable, in this specific data set, should not have a great weight (but the fact is that it does, as we will see later).



2.2.1.4 Gender Based on the available data, the incidence of strokes among men is slightly higher than among women. The difference is not great, and as we will see later, the variable actually has little weight.

gender	total	percent	strokes	stroke_percent
Female	2897	0.59	120	0.041
Male	2011	0.41	89	0.044
Other	1	0.00	0	0.000

The bar graphs clearly show the distribution of the variables, but due to the high prevalence of the "no_stroke" class, it is difficult to see that the incidence is actually somewhat higher among men, as shown in the table:

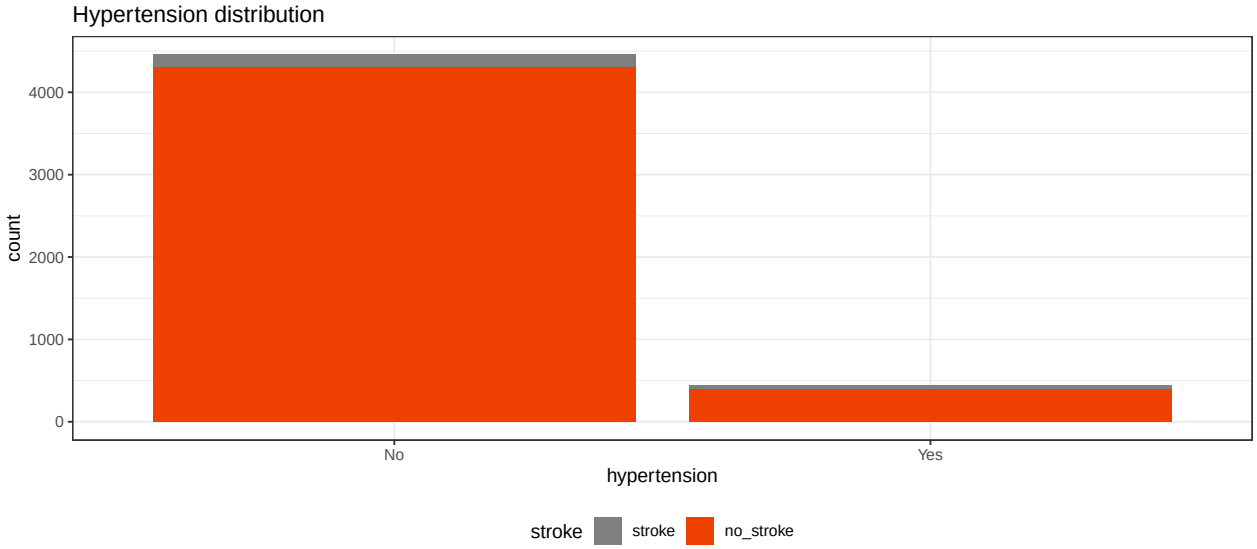


2.2.1.5 Hypertension Due to the clear difference in the incidence of strokes depending on whether or not one suffers from hypertension, it is logical to think that this variable will have an important weight:

hypertension	total	percent	strokes	stroke_percent
No	4457	0.908	149	0.033
Yes	451	0.092	60	0.133

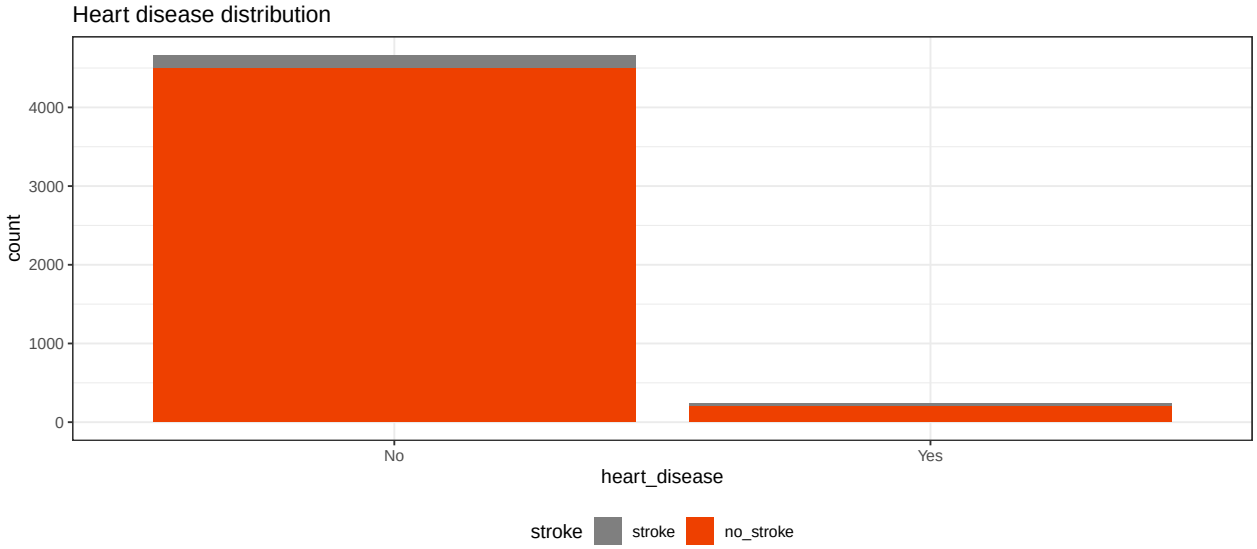
Among those who do not suffer from hypertension, the incidence is 3%. Among those who do suffer it, it reaches 13%.

Again, it is difficult to see this reality in bar charts:



2.2.1.6 Heart Disease In the group suffering from Heart Disease, the incidence of stroke reaches 17%. Those who do not have heart problems have the same chances of having a stroke as the base population (4%).

heart_disease	total	percent	strokes	stroke_percent
No	4665	0.95	169	0.036
Yes	243	0.05	40	0.165



Due to the clear difference in the probability of suffering a stroke depending on whether or not one has heart

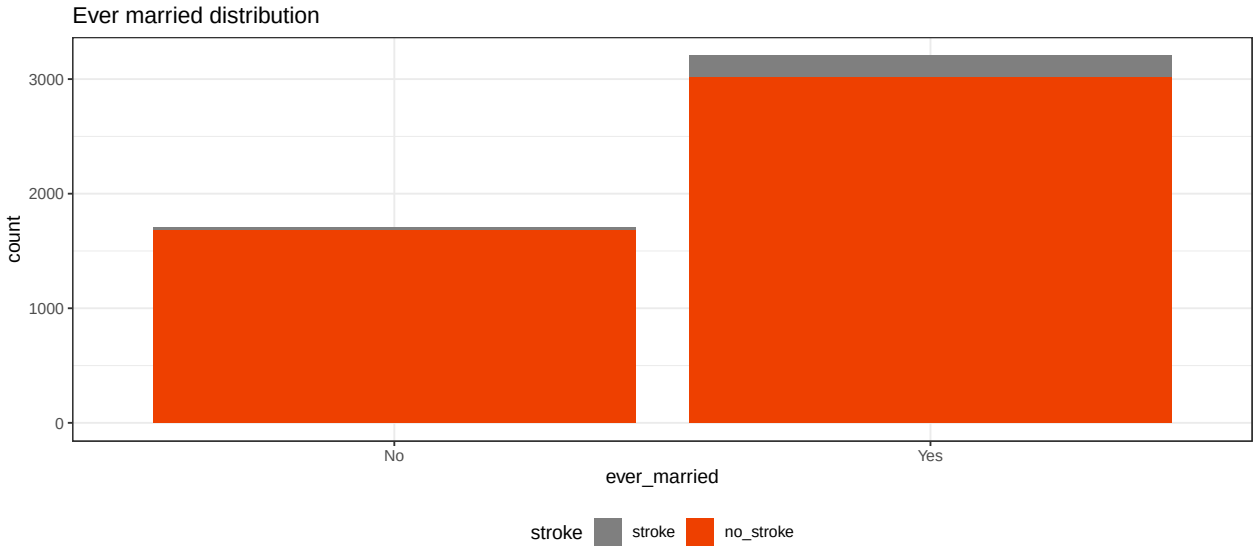
disease, it is reasonable to think that the variable will have a certain weight for the classification models.

When we get to the point of assessing the importance of variables, we will find certain surprises. Some variables that with this analysis seemed firm candidates to have a clear weight, in reality they do not, at least for those models in which the selection of variables is evident.

2.2.1.7 Ever Married There is also some difference in the probability of suffering a stroke whether you have been married or not. This difference is only partly surprising, if one takes into account that being married or not is directly related to age. Young people are obviously less likely to have been married than older people, and we have already seen that there is a clear relationship between age and strokes.

ever_married	total	percent	strokes	stroke_percent
No	1704	0.347	23	0.013
Yes	3204	0.653	186	0.058

For those who have been married, the percentage of strokes is around 6%. Among those who have always been single, the percentage barely exceeds 1%.

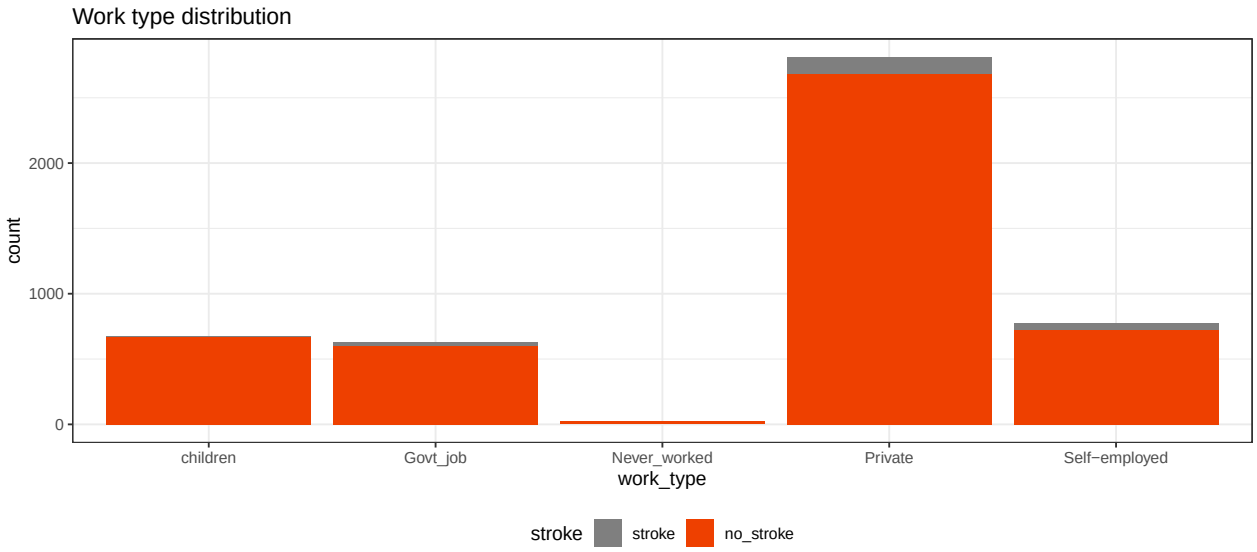


2.2.1.8 Work Type Those who are self-employed appear to be at slightly higher risk of strokes than other types of work.

work_type	total	percent	strokes	stroke_percent
children	671	0.137	1	0.001
Govt_job	630	0.128	28	0.044
Never_worked	22	0.004	0	0.000
Private	2810	0.572	127	0.045
Self-employed	775	0.158	53	0.068

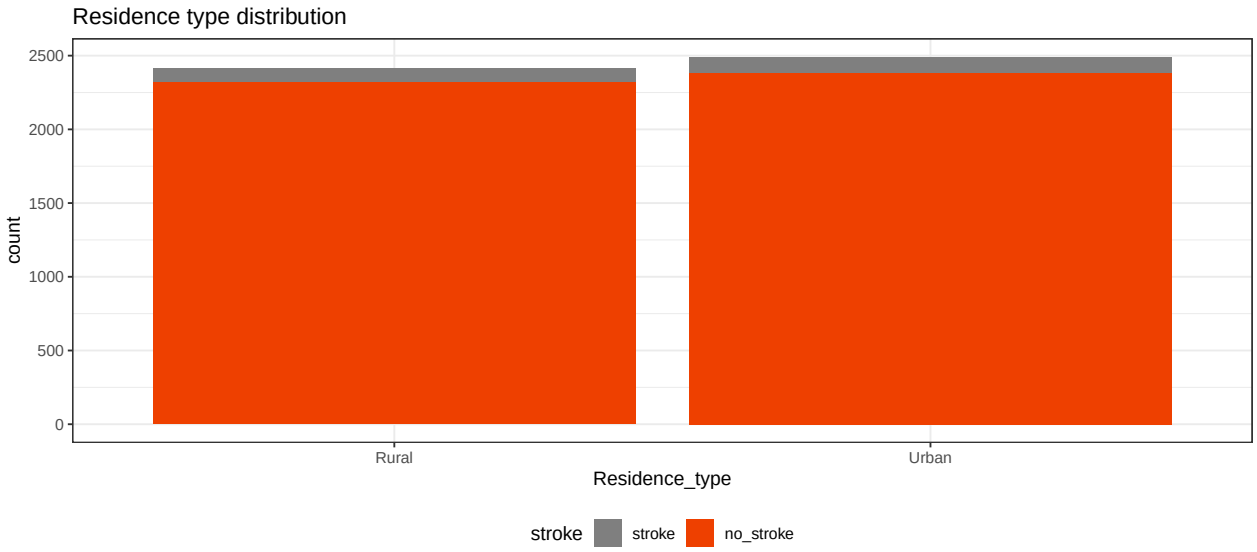
The cases of never worked are very few, and there are no incidences of strokes in this group. Children also do not present cases. This would explain why certain models, when they take this variable into account, use the group formed by Govt_job, Private and Self-employed as a single decision element.

Another future course of action could be to delete the observations tagged as Never Worked. They are very few and can add noise.



2.2.1.9 Residense Type The type of residence does not seem to influence the risk of suffering a stroke. Both groups are very balanced, and the percentage of strokes in the two cases is very similar to the baseline incidence.

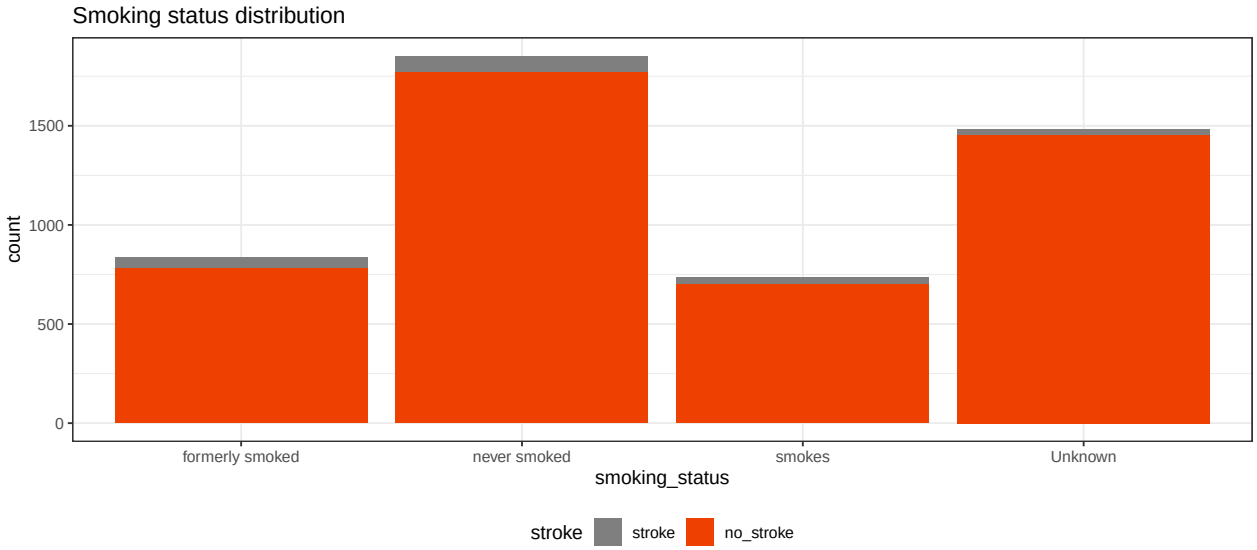
Residence_type	total	percent	strokes	stroke_percent
Rural	2418	0.493	100	0.041
Urban	2490	0.507	109	0.044



2.2.1.10 Smoking Status Finally, and as expected, the incidence of strokes is higher in the group of smokers and in those who have ever smoked. However, the incidence is not clearly higher. Where there is an important difference with the base incidence percentage in the "unknown" group. As we will see, in the models that consider smoking_status, "unknown" is used as a decision element.

smoking_status	total	percent	strokes	stroke_percent
formerly smoked	836	0.170	57	0.068
never smoked	1852	0.377	84	0.045

smoking_status	total	percent	strokes	stroke_percent
smokes	737	0.150	39	0.053
Unknown	1483	0.302	29	0.020



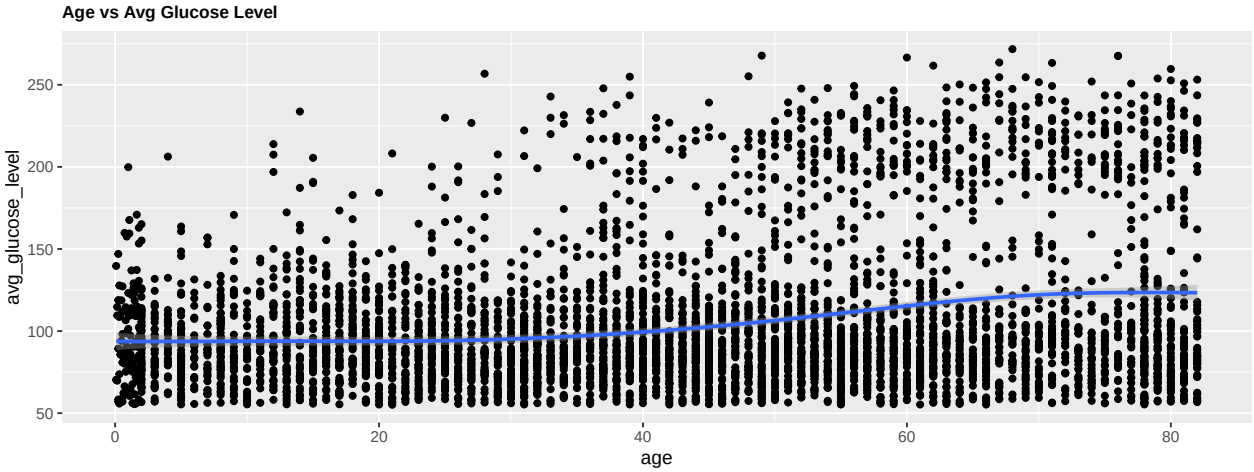
2.2.2 Variable importance

The exploratory analysis above allows to get an idea of which variables will have more weight when classifying and predicting the probabilities of suffering a stroke. By the way, it allows us to find those groups with few cases, which might not be present in both data sets (training and test), and which might be worth filtering.

However, and as Rodrigo Amat points out in *Machine Learning with R and caret*, this approach says nothing about the relationship between variables, and their joint effects.

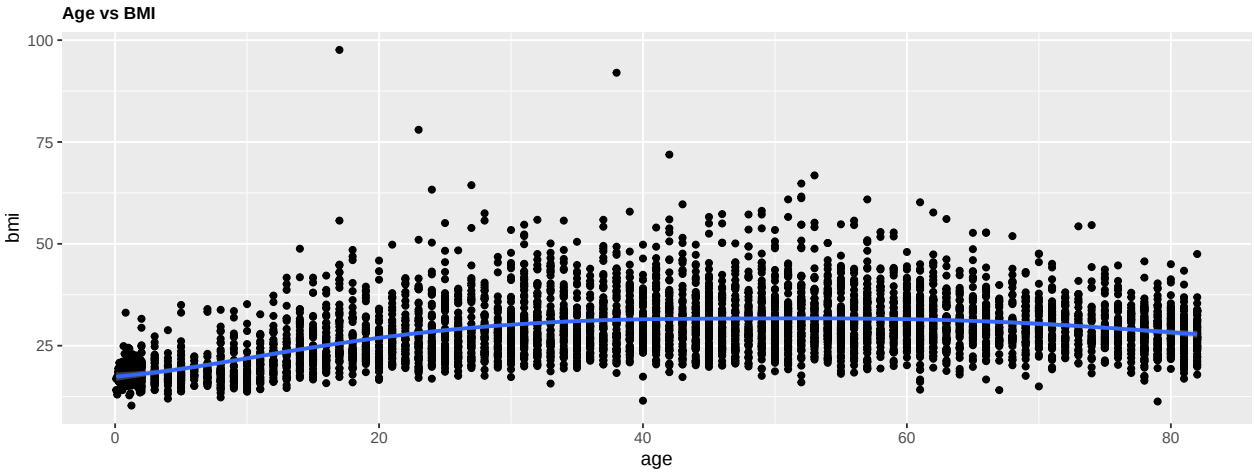
Next we will analyze the correlation between numerical variables, the importance of the variables with the random forest method, and we will make sure that none of the variables have a variance close to zero. Again, it is necessary to refer to Rodrigo Amat as the source of this approach.

2.2.2.1 Correlation between numerical variables As is well known, a strong correlation between numerical predictors can harm the results of a model. At this point, we find a new stumbling block, in addition to those found previously. Although visually the correlations do not seem strong, numerically they have some importance. In all cases they are less than 0.5, but we cannot say that there is no correlation at all. The values obtained are at that point where we believe that all numerical variables should be kept for modeling purposes, although it might be worth testing by eliminating those that are more correlated in future iterations.



The numerical correlation between Age and Average Glucose Level is close to 0.24:

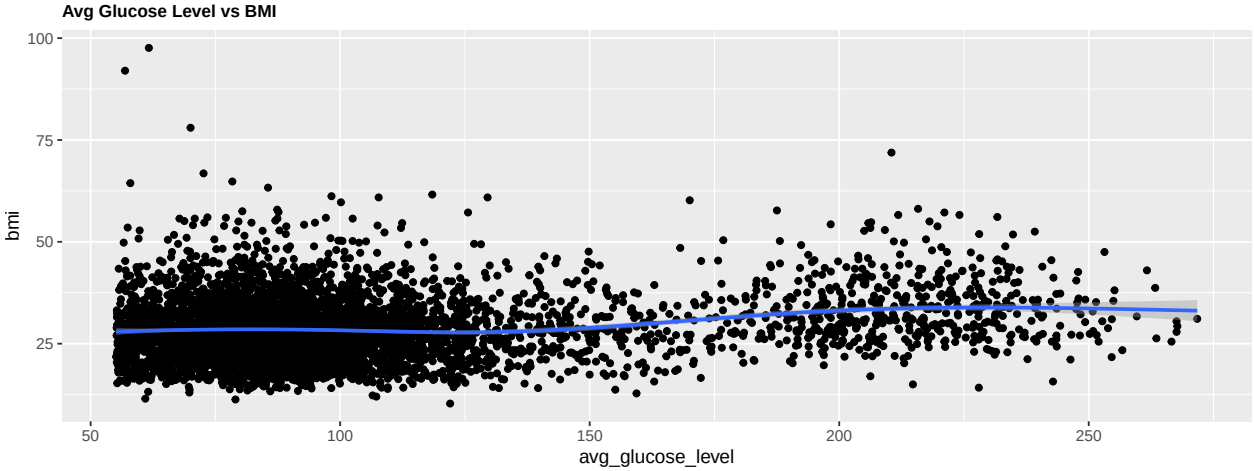
```
##
##  Pearson's product-moment correlation
##
## data:  stroke_data$age and stroke_data$avg_glucose_level
## t = 17.011, df = 4906, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2094044 0.2622460
## sample estimates:
##       cor
## 0.2359996
```



The correlation between age and body mass index is even slightly stronger, exceeding 0.33.

```
##
##  Pearson's product-moment correlation
##
## data:  stroke_data$age and stroke_data$bmi
## t = 24.762, df = 4906, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3082105 0.3579540
```

```
## sample estimates:
##       cor
## 0.3333142
```



Finally, the correlation between Average Glucose Level and Mass Index is around 0.18.

```
##
##  Pearson's product-moment correlation
##
## data:  stroke_data$avg_glucose_level and stroke_data$bmi
## t = 12.499, df = 4906, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1484233 0.2026536
## sample estimates:
##       cor
## 0.1756717
```

2.2.2.1 Contrast of proportions With respect to categorical variables, Amat's approach is based on performing a proportion test. It is about seeing which variables present a proportion of cases that are really different from the baseline level, taking into account the number of cases. The results are ordered from lowest to highest p-value.

Next we see the table with the first 10 results. Although for some variables it is noted that the Chi-squared approximation may be incorrect (not shown in this table), this exercise confirms part of what we had seen previously: suffering from hypertension or heart disease considerably increases the risk of suffering a stroke.

On the contrary, not having been married, or being a child who does not work, reduces the risk considerably.

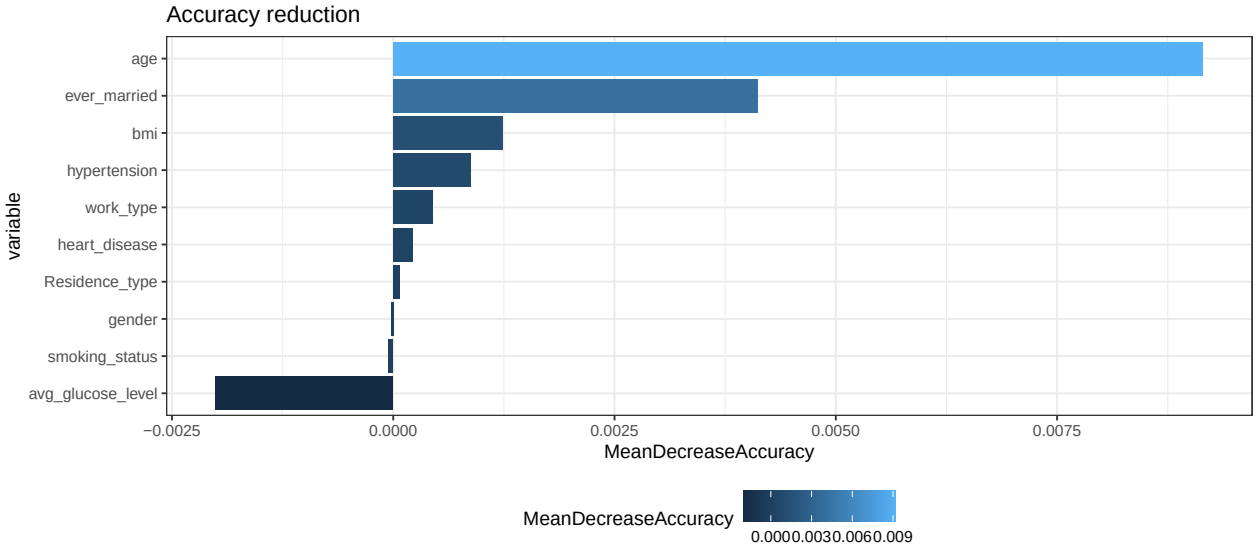
Again, we must note that this approach does not take into account the joint effect of the variables, nor the relationship between them (Amat, 2018).

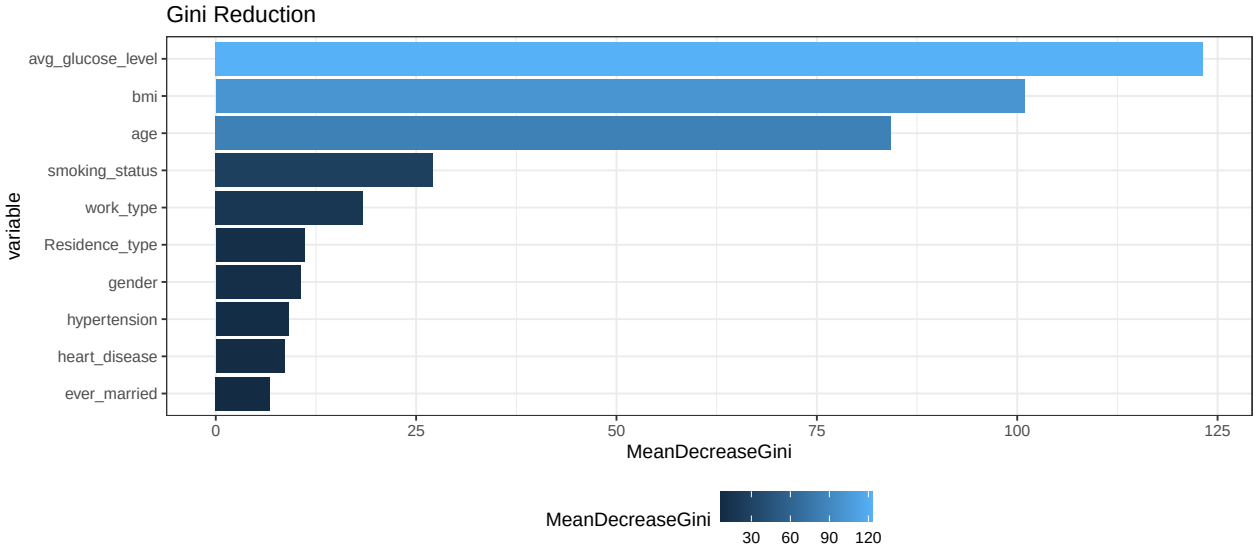
```
## # A tibble: 10 x 3
## # Groups:   group_variable [10]
##    group_variable                 p_value prop_strokes
##    <chr>                            <dbl>        <dbl>
##  1 hypertension_Yes              5.52e-21      0.133
##  2 heart_disease_Yes             1.98e-20      0.165
##  3 ever_married_No               3.98e- 9      0.0135
##  4 work_type_children            2.27e- 7      0.00149
##  5 smoking_status_Unknown        1.51e- 5      0.0196
```

```
##  6 ever_married_Yes                1.74e- 5      0.0581
##  7 smoking_status_formerly smoked 3.42e- 4      0.0682
##  8 work_type_Self-employed        5.20e- 4      0.0684
##  9 hypertension_No                2.82e- 3      0.0334
## 10 heart_disease_No               3.48e- 2      0.0362
```

2.2.2.3 The random forest method Next, we will review the importance of the variables using the random forest method. It is important to note that this analysis has been carried out on the original, strongly unbalanced data set. Results will vary when techniques are applied to balance the training set.

This method can be applied in two ways: by Accuracy Reduction, or Gini Reduction. Here we will apply both to see where they coincide and where they differ.





In the case of Accuracy Reduction, the most important variable is Age, followed by Ever Married. BMI and Hypertension are next on the list, but they are not given great weight.

Despite being an unbalanced data set, this approach yields surprising results to say the least. It is difficult to understand why Ever Married is preferred over other variables such as BMI or Hypertension.

Gini Reduction shows results more related to what we have seen so far. Age is an important variable in both cases. At the top of the list is Avg Glucose Level and BMI. It is still surprising that Hypertension and Heart

Disease carry so little weight in comparison.

If we look more deeply at the case of Heart Disease, we see that its incidence is clearly related to age:

round_age	total	percent	heart_disease	heart_disease_percent
0	311	0.063	1	0.003
10	371	0.076	0	0.000
20	582	0.119	0	0.000
30	533	0.109	1	0.002
40	749	0.153	4	0.005
50	688	0.140	28	0.041
60	781	0.159	53	0.068
70	414	0.084	65	0.157
80	479	0.098	91	0.190

This would explain why, once age is used as a decision factor, Heart Disease loses importance.

The same goes for Hypertension. The incidence, as is obvious, is closely related to age:

round_age	total	percent	hypertension	hypertension_percent
0	311	0.063	0	0.000
10	371	0.076	0	0.000
20	582	0.119	6	0.010
30	533	0.109	7	0.013
40	749	0.153	43	0.057
50	688	0.140	78	0.113
60	781	0.159	117	0.150
70	414	0.084	89	0.215
80	479	0.098	111	0.232

It is necessary, in any case, that the random forest model did not obtain good results. In this case, the selection of variables effected by this method does not seem the most appropriate, at least for Accuracy Reduction.

2.2.2.3 Near Zero Variance Analysis The last of the analyzes recommended by Amat, before training models, is the one destined to eliminate the variables that present, before scaling and centering the numerical data, variables close to zero.

Using the original unbalanced data set, the only variable that presents a variance close to zero is Heart Disease (which would explain its bad positions in the importance analysis carried out previously):

```
##                   freqRatio percentUnique zeroVar   nzv
## gender             1.440577    0.04074980   FALSE FALSE
## age                1.000000    2.11898941   FALSE FALSE
## hypertension       9.882483    0.04074980   FALSE FALSE
## heart_disease     19.197531    0.04074980   FALSE  TRUE
## ever_married       1.880282    0.04074980   FALSE FALSE
## work_type          3.625806    0.10187449   FALSE FALSE
## Residence_type     1.029777    0.04074980   FALSE FALSE
## avg_glucose_level  1.200000   78.46373268   FALSE FALSE
## bmi                1.078947    8.51670742   FALSE FALSE
## smoking_status     1.248820    0.08149959   FALSE FALSE
```

This changes when we apply techniques to balance the training set, and none of the variables presents variance close to zero.

2.2.2.4 Classification Tree Visualization Another way to easily visualize important variables is through decision trees. Next we experiment with the unbalanced data set, aware that it is only a test, and that the final models cannot be trained without first balancing the classes.

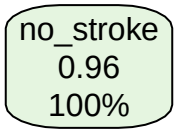
The first experiment is to visualize the tree generated by the default "rpart" model. It uses Gini Reduction and applies the following parameters:

1. Complexity parameter = 0.01
2. Minsplit = 20
3. Minbucket = 20/3
4. MaxDepth = 30

As explained in *Learn by marketing*[5], the complexity parameter (cp) in rpart is the minimum improvement in the model needed at each node. Minsplit is the minimum number of samples at each node, and MaxDepth is the maximum depth of the tree.

As expected, the result is not a tree but a single node that classifies all instances as "no_stroke", given the prevalence of that class:

```
# __Gini Tree, CP = 0.01, minsplit = 20, minbucket round 20/3, maxdepht = 30 ####
rpart.plot(rpart(stroke ~ .,
                 data = stroke_data)) # Default rpart tree
```

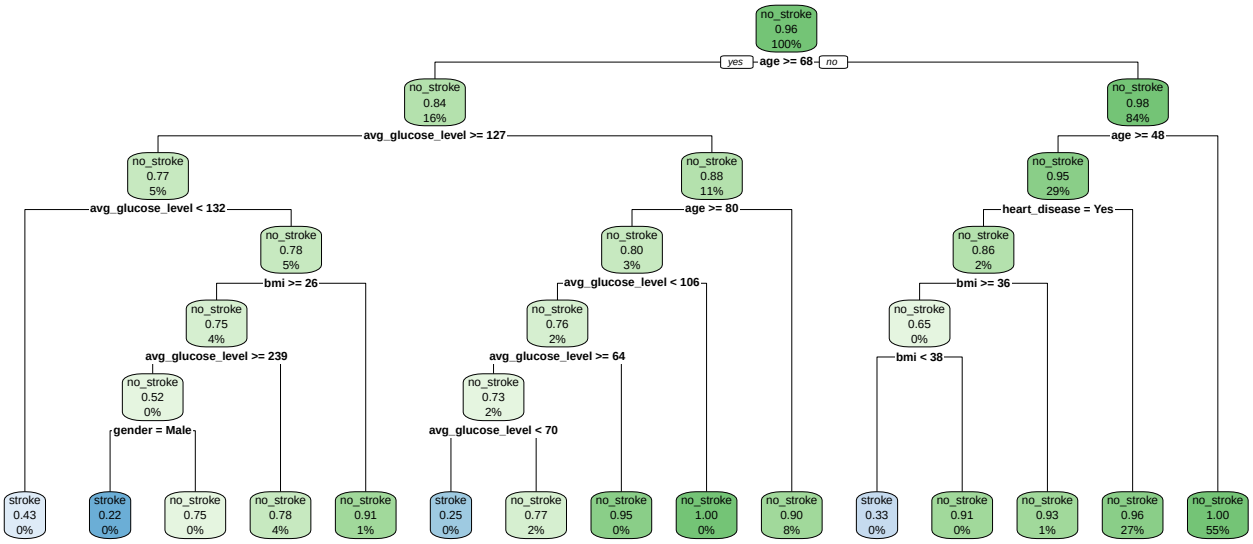


To see something, we must resort to overtraining. Again, it is important to note that for now we are only experimenting to see if we can see some variables. We are not training any models yet.

To create the following tree we have set the Complexity Parameter to 0.001, and the MaxDeth to 6:

```
# __Gini Tree, CP = 0.001, minslit = 20, minbucket round 20/3, maxdepht = 6 #####
rpart.plot(rpart(stroke ~.,
                 data = stroke_data,
                 parms=list(split=c("gini")),
                 cp = 0.001,
                 maxdepth = 6))
```

[5] *Learn by Marketing*: https://www.learnbymarketing.com/tutorials/rpart-decision-trees-in-r/



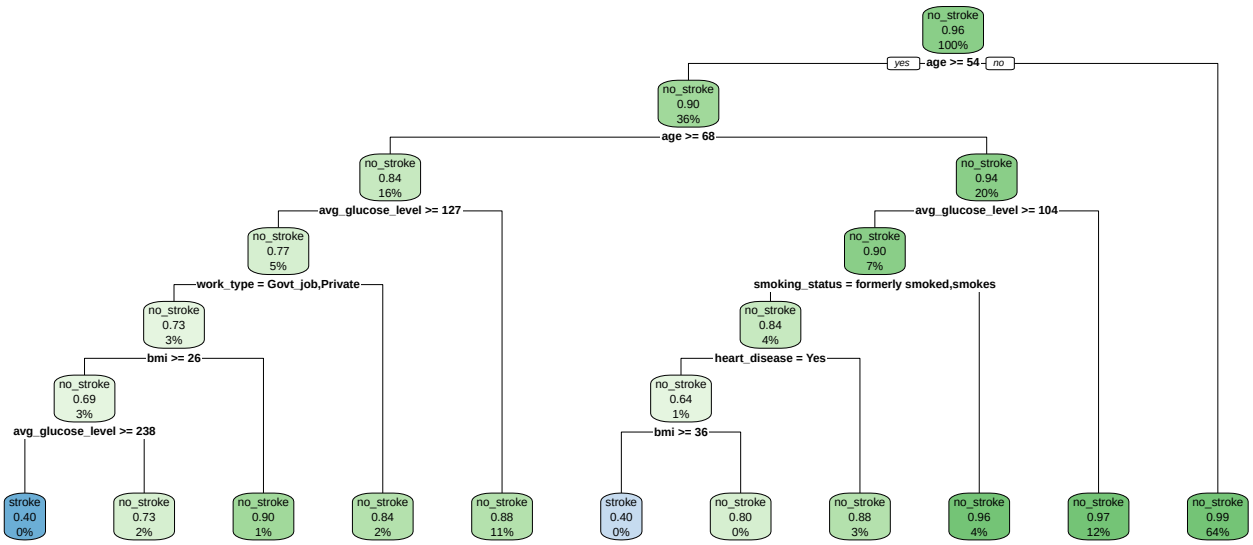
Although this tree requires pruning, we can already see that the variables that are repeated the most are Age, Average Glucose Level and BMI. As for the categorical ones, Heart Disease and Gender appear.

If instead of Gini Reduction we opt for Information Gain, we see that the main numerical variables are Age and Average Glucose Level. With respect to categorical variables, Work Type, Smoking Status and Heart Disease appear.

Both approaches, although we are clearly overtraining, yield results that, intuitively, seem more accurate than those obtained with the random forest method.

The following tree uses the Information Gain method, the CP is set to 0.0023, and the MaxDepth is equal to 6:

```
# __Information Tree, CP = 0.0023. minslit = 20, minbucket round 20/3, maxdepht = 6 #####
rpart.plot(rpart(stroke ~.,
                 data = stroke_data,
                 parms=list(split=c("information")),
                 cp = 0.0023,
                 maxdepth = 6))
```



It is interesting to note that if we try to create a tree only using categorical variables, no results are obtained with any method. This may be due to the fact that these types of variables are not capable of classifying between "stroke" and "no_stroke" by themselves, and that they need to interact with numeric variables.

The latter, on the other hand, are capable of generating trees when they interact with each other, without the need for categorical variables.

2.2.2.5 Balanced Data But what happens if we balance the data? For this exercise, we have used three methods to match the number of instances classified as "stroke" and "no_stroke", using the "ROSE" library.

1. Oversamplig: observations from the minority class are randomly added.
2. Both: the distributions are equalized by randomly adding observations from the minority class, and randomly removing observations from the majority class.
3. Better estimates: the library has a function (ROSE) that generates synthetic data to balance the classes. *The data generated using ROSE is considered to provide better estimate of original data*[6].

Next we review, for the case of Oversamplig, that although the classes have been balanced, the statistics and the distribution of the variables are maintained. This is true for the other two methods used.

After applying the "ovun.sample" function, with the "over" method, the new frequencies are as follows:

```
# ________________________########
# Over Sampling: all data ######
# ________________________########

n_over = sum(stroke_data == "no_stroke")

set.seed(1969, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1969)`

stroke_data_over <- ovun.sample(stroke ~ ., data = stroke_data,
                                method = "over", N = n_over*2)$data

# Relevel "stroke" "no_stroke" factors: positive class: "stroke"
stroke_data_over$stroke <- relevel(stroke_data_over$stroke, ref = "stroke")

table(stroke_data_over$stroke) %>%
  kable()
```

Var1	Freq
stroke	4699
no_stroke	4699

```
prop.table(table(stroke_data_over$stroke))  %>%
  kable()
```

Var1	Freq
stroke	0.5
no_stroke	0.5

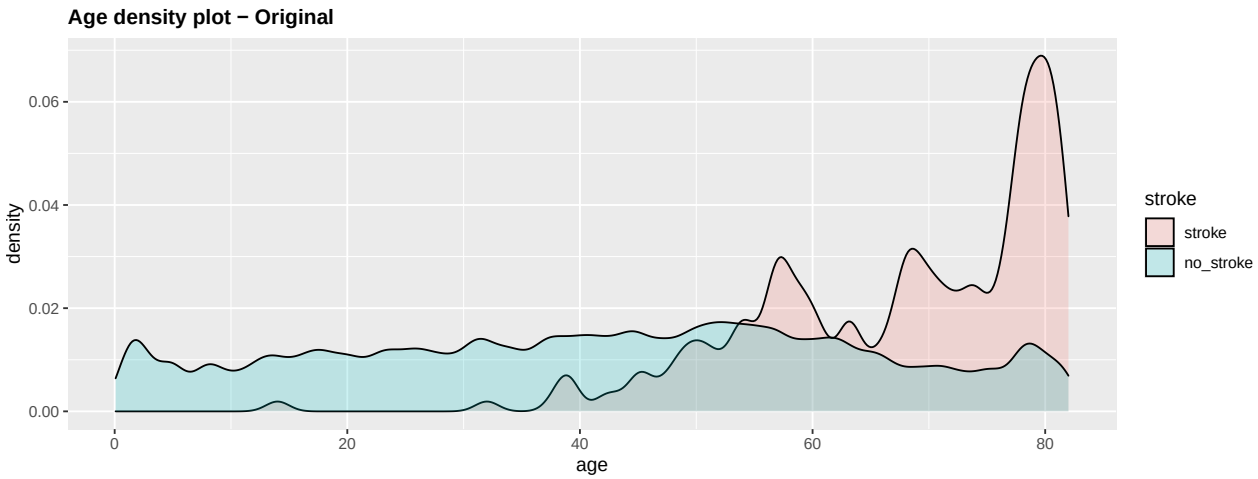
This, as has been said, does not change the statistics or the distributions of the numerical variables. Below for the Age variable, we show the original tables and graphs, compared with balanced data using the oversampling

[6] *Practical Guide to deal with Imbalanced Classification Problems in R.* Analytics Vidhya, 2016: https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/

method.

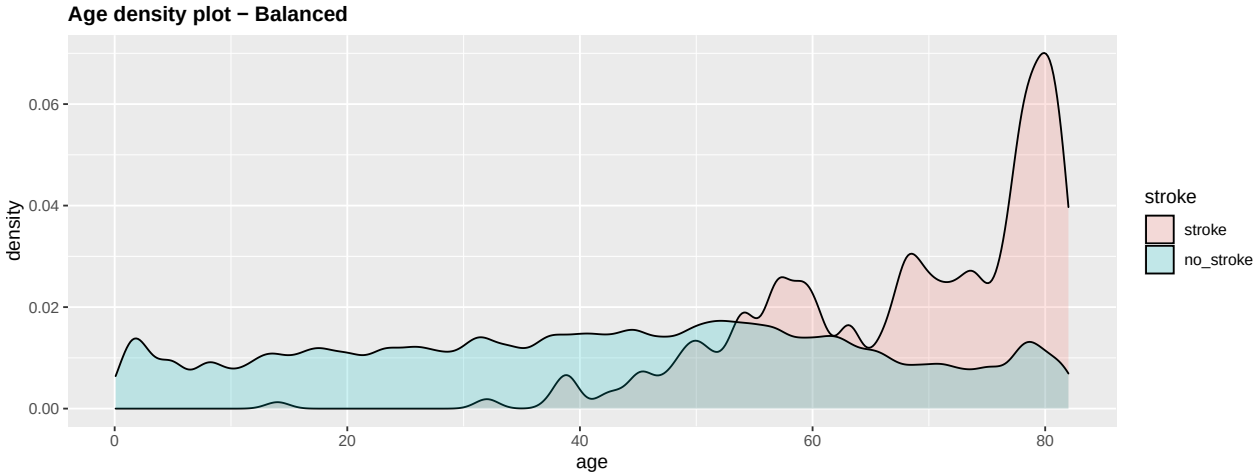
Original data

stroke	avg_age	median_age	min_age	max_age
stroke	67.7	70	14.00	82
no_stroke	41.8	43	0.08	82



Balanced data (oversampling method)

stroke	avg_age	median_age	min_age	max_age
stroke	68.1	71	14.00	82
no_stroke	41.8	43	0.08	82



In the case of categorical variables, similar proportions of the groups are maintained, but the percentage of "strokes" changes. What used to be 0.6% becomes 60%.

To exemplify the case of categorical variables, we have selected Smoking Status:

Original data

smoking_status	total	percent	strokes	stroke_percent
formerly smoked	836	0.170	57	0.068
never smoked	1852	0.377	84	0.045
smokes	737	0.150	39	0.053
Unknown	1483	0.302	29	0.020

Balanced data (oversampling method)

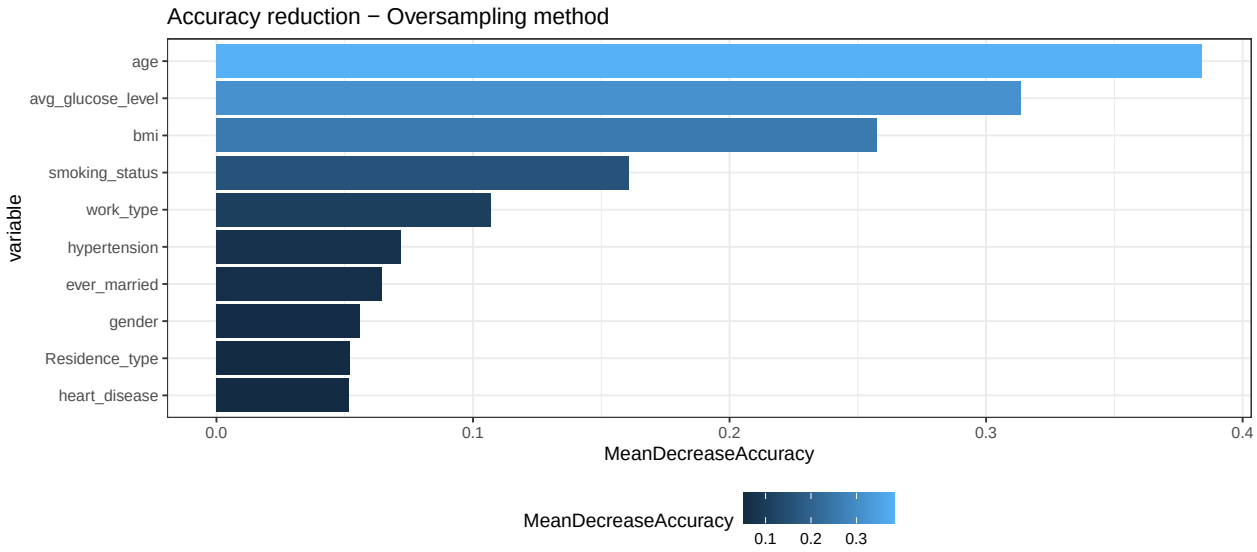
smoking_status	total	percent	strokes	stroke_percent
formerly smoked	2067	0.220	1288	0.623
never smoked	3657	0.389	1889	0.517
smokes	1581	0.168	883	0.559
Unknown	2093	0.223	639	0.305

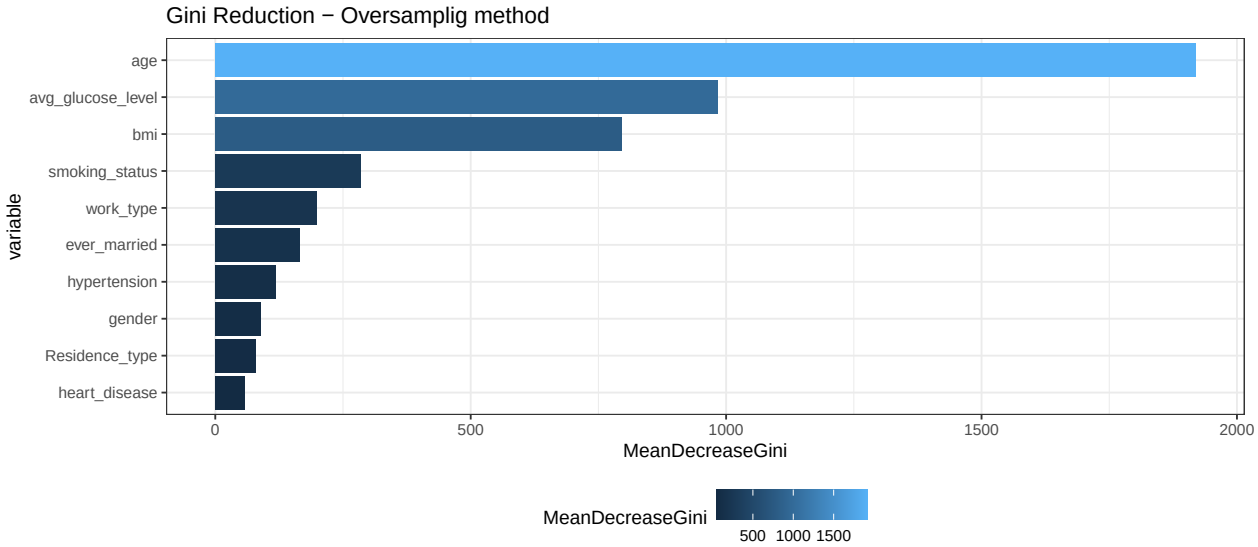
This same check has been carried out for all methods and all variables (numerical and categorical ones).

We have also verified that the correlation between numerical variables is similar to that found with the original data.

So what changes? With the data balanced using the oversampling method, we see changes in the list of variables obtained with the contrast of proportions (here we show the first ten), and with the random forest method. This is precisely what we are looking for: determining the importance of variables without the prevalence problems of the original data set.

```
## # A tibble: 10 x 3
## # Groups:   group_variable [10]
##    group_variable                   p_value prop_strokes
##    <chr>                              <dbl>        <dbl>
##  1 ever_married_No                1.49e-147       0.221
##  2 work_type_children             8.24e-138       0.0219
##  3 hypertension_Yes               6.95e-114       0.773
##  4 heart_disease_Yes              2.52e- 97       0.816
##  5 smoking_status_Unknown         8.06e- 71       0.305
##  6 ever_married_Yes               2.57e- 45       0.583
##  7 smoking_status_formerly smoked 5.49e- 29       0.623
##  8 hypertension_No                5.68e- 27       0.439
##  9 work_type_Self-employed        4.60e- 26       0.621
## 10 heart_disease_No               2.34e- 14       0.458
```



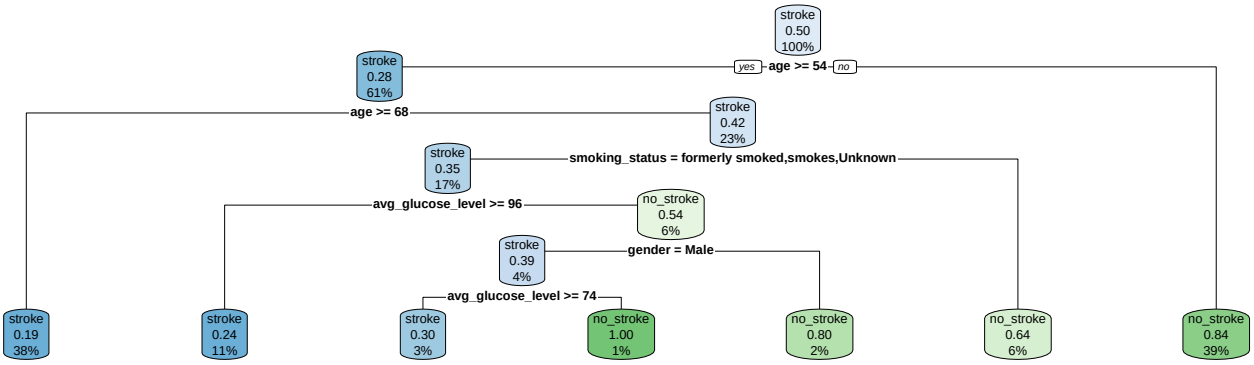


Finally, we have verified that the balanced data do not present variables with variation close to zero. Again, we show the results for the balanced data with the oversampling method:

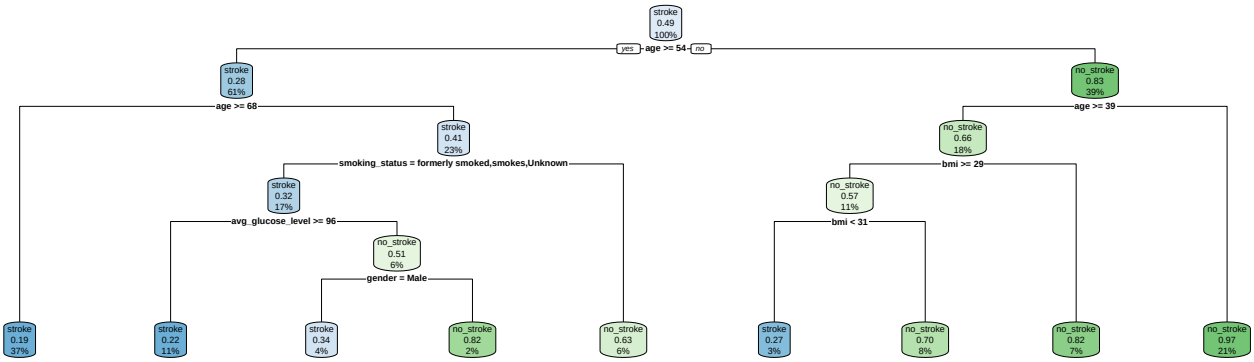
```
##                  freqRatio percentUnique zeroVar   nzv
## gender            1.370838    0.02128112   FALSE FALSE
## age               1.011494    1.10661843   FALSE FALSE
## hypertension      4.448116    0.02128112   FALSE FALSE
## heart_disease     7.528131    0.02128112   FALSE FALSE
## ever_married      3.352941    0.02128112   FALSE FALSE
## work_type         2.905561    0.05320281   FALSE FALSE
## Residence_type    1.080584    0.02128112   FALSE FALSE
## avg_glucose_level 1.055556   40.97680358   FALSE FALSE
## bmi               1.145985    4.44775484   FALSE FALSE
## smoking_status    1.747253    0.04256225   FALSE FALSE
```

These outputs are similar for all the methods used to balance the data, although their small differences cause different results in the applied models, as we will in the next chapters. To illustrate the differences between the results obtained with the different methods to balance the data, we show the trees that are now generated with the default "rpart" algorithm.

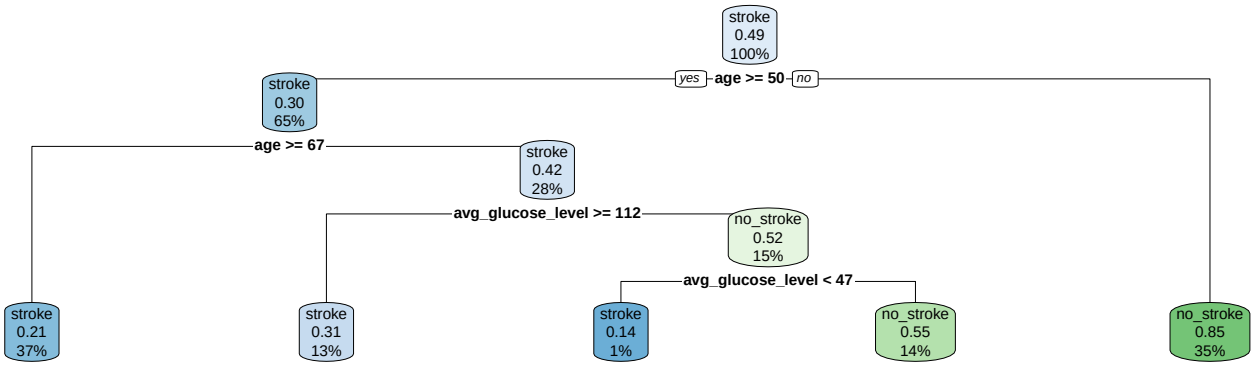
Oversampling



Both



Better Estimates



As we can see, “Age” is the most important variable in all cases. Average Glucose Level" is also common in trees generated by all three methods.

The “oversampling” and “both” methods generate trees in which smoking status is relevant. “BMI” only appears when using the “Both” method. Better Estimates only takes into account the variables “Age” and “Average Glucose Level”. Finally, the tree generated with the oversampling method incorporates the variable “Gender” as a decision element.

It is important to note that the processes for balancing the data are random. Every time we apply the function by selecting the "Oversampling" and "Both" methods, we obtain different data sets, and this can cause certain variables to acquire more or less prominence.

2.2.3 Summary of Exploratory Analysis

At this point, we know that:

1. The original data set is not large. We have just over 5,000 observations.
2. The original data is strongly unbalanced. Only 4% of the observations (about 200) are classified as "stroke".
3. There are no variables that completely separate the two classes. The numerical variables overlap, and the categorical variables are present in both classes (some of them with balanced distributions). This means that predictors with very similar values are present in observations classified as "stroke" and "no_stroke".
4. The numerical variables (Age, Average Glucose Level and BMI), which are important according to various analysis techniques, show a certain correlation.
5. Variables that present a clearly different incidence from the baseline, such as Heart Disease and Hypertension, do not seem to be important when analyzed together with the rest of the variables. Their relationship with Age is so strong that once Age is selected as the main variable, the others lose weight.

The situation, therefore, does not look very promising when it comes to finding a model that yields good results. Without having trained any model yet, we can already anticipate that algorithms based on neighbors, for example, should not be suitable in this case.

2.3 Modeling Approach

2.3.1 Train and test sets

As anticipated in the introduction, for the creation of the training set we reserve 80% of the data. Because the positive class "stroke" is very minor, before training the models we made sure that both classes were present in the test set, and that they had a similar distribution.

```
# Creating train and test set #####

# We will use 80% of data to train, and 20% of data to test.
set.seed(1970, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1970)`
test_index <- createDataPartition(y = stroke_data$stroke, times = 1, p = 0.2,
                                  list = FALSE)

# __train_stroke ####
train_stroke <- stroke_data[-test_index,]

# __test_stroke ####
test_stroke <- stroke_data[test_index,]
```

```
# Review #####
# Comparing test and train set to be sure that proportions are similar

# __Percent of strokes in train set ####
mean(train_stroke$stroke == "stroke")
```

```
## [1] 0.04253693
```

```
# __Percent of strokes in test set ####
mean(test_stroke$stroke == "stroke")
```

```
## [1] 0.04276986
```

A similar analysis was done for the rest of the variables. Only some important distribution differences were found in the case of Average_Glucose_Level.

2.3.2 Cross Validation and Criteria to Select the Best Tuning Parameters

Before continuing, we show here the selected cross-validation parameters to train all the models of the caret package:

```
# ________________________########
# TRAIN CONTOL  ########
# For Caret Trains ########
# ________________________########

ctrl <- trainControl(method="repeatedcv",
                     number = 10,
                     repeats = 5, summaryFunction=twoClassSummary, classProbs=T,
                     savePredictions = T)
# Parameters ClassProbs and SavePred: needed to plot ROC Curves.
```

We should also mention that the criterion to obtain the best parameters was in all cases the **Area Under the ROC Curve**.

It is clear that in this case, Accuracy is not useful for determining the quality of the models, due to the prevalence of the negative class. If the model predicts "no_stroke" in all cases, it will be 96% accurate.

Furthermore, this is the typical situation where sensitivity is more important than specificity. A false positive is preferable to a false negative. However, a model that predicts "stroke" in all cases is also not useful.

A high ROC, together with other indicators, is usually obtained when there is a certain balance between sensitivity and specificity (except in some exceptions), and this is what we wanted to achieve in the development of this project. Always giving priority to sensitivity, we wanted to maintain specificity at values above 70%.

2.3.3 Training Models

Despite the comments in point 5 of the *Summary of Exploratory Analysis section*, all the variables contribute to positive reductions (in Accuracy and Gini). This, together with the fact that depending on the balancing method some variables acquire more importance than others, has made us believe that it is worth training and comparing models taking into account all the variables. We will let the models capable of doing so, select the predictor variables through cross validation.

Given the nature of this project, we think the main goal is to experiment with various models. This allows us to analyze their differences, not only in terms of results, but also with respect to other aspects, such as computation times, tune parameters, and explicability.

In this report we will focus on the models that allow us to exemplify the process, whether it be by contrast between results, or by their quality.

As part of this experimentation, the first step was to train a series of models with the original data (after scaling and centralizing the numerical variables). At this stage, we were fully aware that due to the prevalence of the negative class, the results were not going to be good.

```
# PreProcessing: Centering and Scaling numerical variables ######
#(t stands for transformed)#####

preProcValues <- preProcess(train_stroke, method = c("center", "scale"))

# __train_stroke_t ####
train_stroke_t <- predict(preProcValues, train_stroke)
```

```
# __test_stroke_t ####
test_stroke_t <- predict(preProcValues, test_stroke)
```

With the original umbalanced data, the following models were trained:

- Rpart native
  - Gini, CP = 0.01, minslit = 20, minbucket round 20/3, maxdepht = 30 (default)
  - Gini Tree, CP = 0.001, minslit = 20, minbucket round 20/3, maxdepht = 30
  - Gini Tree, CP = 0.0024, minslit = 20, minbucket round 20/3, maxdepht = 30
  - Gini Tree, CP = 0.001, minslit = 20, minbucket round 20/3, maxdepht = 5
  - Information Tree, CP = 0.01, minslit = 20, minbucket round 20/3, maxdepht = 30
  - Information Tree, CP = 0.001. minslit = 20, minbucket round 20/3, maxdepht = 30
  - Information Tree, CP = 0.0023. minslit = 20, minbucket round 20/3, maxdepht = 30
- Caret Rpart:
- Caret K-Nearest-Neighbor: (KNN)
- Random Forest (RF)
- Flexible Discriminant Analysis (FDA)

As objectives, this previous exercise had:

1. For classification trees, test complexity and pruning parameters by hand to see what each one does.
2. Compare the performance of the "rpart" method of the caret package with the tree generated by the default rpart function.
3. Check the results of three classification models, two of them seen in the course documentation (KNN, RF), and another not included in it (FDA).

Once we had verified what we already knew (with the unbalanced data we were not going to obtain correct results), we balanced the training set with the methods described in the previous section, and we trained the following models for each of the cases (oversampling, both and better estimates):

- RPART native
  - Gini Tree, CP = 0.01, minslit = 20, minbucket round 20/3, maxdepht = 30
  - Gini Tree, CP = 0.001, minslit = 20, minbucket round 20/3, maxdepht = 30
  - Default Gini Tree & Cost Matrix 3 to 1
- RPART caret
- K-Nearest-Neighbor (KNN)
- Random Forest (RF)
- Neural Network (NNET)
- Flexible Discriminant Analysis (FDA)
- Naives Bayes (NB): We know that in this case Naives Bayes is not a good choice. Being a generative model, more than two numerical predictors are too many, and its distribution, as we have seen, is not normal in all cases. We have included it for experimentation.

For balanced data, we have added two other models: Neural Network and Naive Bayes.

Once the aforementioned models were trained, for each balancing method, we were able to determine which model, by itself, yields the best results.

However, to see if we could further improve those results, we created two Ensambles with the best-performing models for each balancing method.

The criteria for the first Ensamble was the "Area Under de Roc Curve" (AUC). For the second Ensamble we opted for "Balanced Accuracy" to select the models. Once the Ensembles were created, we selected "Balaced Acccuracy" as the comparison metric in both cases, and we looked for the cutoff (x) that optimized that metric (if x or more models predict "stroke", then predict "stroke").

3. Results

3.1 Unbalanced data

We are not going to stop long here, because as expected the results are not good. With native rpart models, overtraining is required to obtain a sensitivity of just 2% (1 correct classification of 42). With a Complexity Parameter (CP) set to 0.01, the model is not able to classify any instance as "stroke" (with an Accuracy of close to 96%, which is the prevalence of the negative class):

3.1.1 Gini, CP = 0.01, minslit = 20, minbucket round 20/3, maxdepht = 30

```
# Gini, CP = 0.01, minslit = 20, minbucket round 20/3, maxdepht = 30 ####
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`

gini_tree_cp0.01 <- rpart(stroke ~ .,
                          data = train_stroke_t) # Default rpart tree

y_hat_gini_tree_cp0.01 <- predict(gini_tree_cp0.01, test_stroke_t, type = "class")

# Model Evaluation :::::::

# Confusion Matrix
cm_gini_tree_cp0.01 <- confusionMatrix(y_hat_gini_tree_cp0.01, test_stroke$stroke)
cm_gini_tree_cp0.01
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke         0         0
##   no_stroke     42       940
##
##                Accuracy : 0.9572
##                  95% CI : (0.9426, 0.969)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 0.5409
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 2.509e-10
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.95723
##              Prevalence : 0.04277
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : stroke
##
```

This is true for all models based on the native rpart function. If we set the CP to below 0.0023, then 1 instance is classified as "stroke". We are still far from optimal results. Below we show the results of the

Gini-based model, with a CP of 0.001:

3.1.2 Gini Tree, CP = 0.001, minslit = 20, minbucket round 20/3, maxdepht = 30

```
# Gini Tree, CP = 0.001, minslit = 20, minbucket round 20/3, maxdepht = 30 #####
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
gini_tree_cp0.001 <- rpart(stroke ~.,
                           data = train_stroke_t,
                           parms=list(split=c("gini")),
                           cp = 0.001)

y_hat_gini_tree_cp0.001 <- predict(gini_tree_cp0.001, test_stroke_t, type = "class")

# Model Evaluation :::::::

# Confusion Matrix
cm_y_hat_gini_tree_cp0.001 <- confusionMatrix(y_hat_gini_tree_cp0.001,
                                              test_stroke_t$stroke)
cm_y_hat_gini_tree_cp0.001
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke         1         8
##   no_stroke     41       932
##
##                Accuracy : 0.9501
##                  95% CI : (0.9346, 0.9629)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 0.88
##
##                   Kappa : 0.0245
##
##  Mcnemar's Test P-Value : 4.844e-06
##
##             Sensitivity : 0.023810
##             Specificity : 0.991489
##          Pos Pred Value : 0.111111
##          Neg Pred Value : 0.957862
##              Prevalence : 0.042770
##          Detection Rate : 0.001018
##    Detection Prevalence : 0.009165
##       Balanced Accuracy : 0.507649
##
##        'Positive' Class : stroke
##
```

These results are similar for most of the cases in which the CP is less than 0.0023

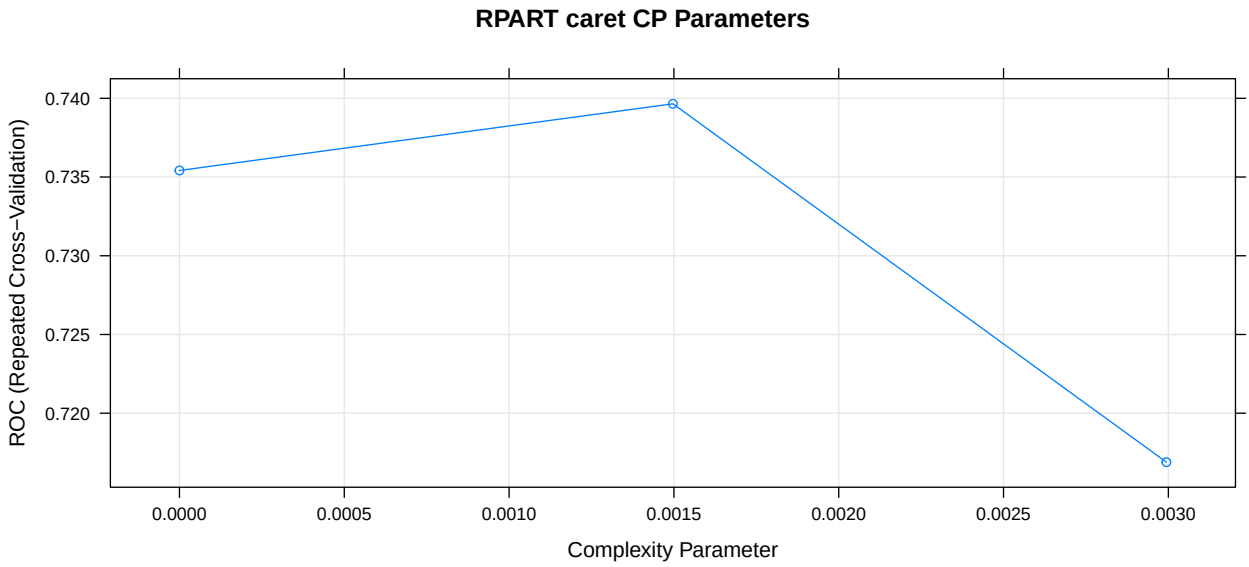
3.1.3 RPART caret

The results of the "train" function (caret package), with the "rpart" method are practically the same as the previous ones:

```r
# RPART caret ######
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
train_caret_tree <- train(stroke ~ ., method = "rpart",
                          data = train_stroke_t,
                          trControl = ctrl,
                          metric = "ROC")

y_hat_caret_tree <- predict(train_caret_tree, test_stroke_t)

# Model Evaluation ::::::::

cm_caret_tree <- confusionMatrix(y_hat_caret_tree, test_stroke_t$stroke)
cm_caret_tree
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke         1         6
##   no_stroke     41       934
##
##                Accuracy : 0.9521
##                  95% CI : (0.9369, 0.9646)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 0.8089
##
##                   Kappa : 0.029
##
##  Mcnemar's Test P-Value : 7.071e-07
##
##             Sensitivity : 0.023810
##             Specificity : 0.993617
##          Pos Pred Value : 0.142857
##          Neg Pred Value : 0.957949
##              Prevalence : 0.042770
##          Detection Rate : 0.001018
##    Detection Prevalence : 0.007128
##       Balanced Accuracy : 0.508713
##
##        'Positive' Class : stroke
##
```

In this case, the CP selected by cross validation, and which optimizes the AUC, is 0.0015:



3.1.4 K-Nearest-Neighbor

KNN does not classify any instance as "stroke":

```
# KNN ########
# K-Nearest-Neighbor #######
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
train_knn <- train(stroke ~ ., method = "knn",
                   data = train_stroke_t,
                   trControl = ctrl,
                   metric = "ROC")

y_hat_knn <- predict(train_knn, test_stroke_t)

# Model Evaluation ::::::::

cm_knn <- confusionMatrix(y_hat_knn, test_stroke_t$stroke)
cm_knn
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke         0         0
##   no_stroke     42       940
##
##                Accuracy : 0.9572
##                  95% CI : (0.9426, 0.969)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 0.5409
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 2.509e-10
##
##             Sensitivity : 0.00000
```

```
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.95723
##              Prevalence : 0.04277
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : stroke
##
```

3.1.5 Random Forest

RF gives results similar to those seen for the overtrained decision trees (a Sensitivity of around 2%).

```
# RANDOM FOREST ########
# it takes time! ######
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
train_rf <- train(stroke ~ ., method = "rf",
                  data = train_stroke_t,
                  trControl = ctrl,
                  metric = "ROC")

y_hat_rf <- predict(train_rf, test_stroke_t, type = "raw")

# Model Evaluation ::::::::

cm_rf <- confusionMatrix(y_hat_rf, test_stroke_t$stroke)
cm_rf
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke         1         0
##   no_stroke     41       940
##
##                Accuracy : 0.9582
##                  95% CI : (0.9438, 0.9699)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 0.4781
##
##                   Kappa : 0.0446
##
##  Mcnemar's Test P-Value : 4.185e-10
##
##             Sensitivity : 0.023810
##             Specificity : 1.000000
##          Pos Pred Value : 1.000000
##          Neg Pred Value : 0.958206
##              Prevalence : 0.042770
##          Detection Rate : 0.001018
##    Detection Prevalence : 0.001018
##       Balanced Accuracy : 0.511905
##
```

```
##        'Positive' Class : stroke
##
```

3.1.6 Flexible Discriminant Analysis (FDA)

The only model that stands out in this phase, comparatively speaking, is FDA. This model is able to correctly classify as "stroke" 14% of the instances (6 of 42):

```
# FDA #####
# Flexible Discriminant Analysis #####
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
train_fda <- train(stroke ~ ., method = "fda",
                   data = train_stroke_t,
                   trControl = ctrl,
                   metric = "ROC")

y_hat_fda <- predict(train_fda, test_stroke_t)

# Model Evaluation ::::::::

cm_fda <- confusionMatrix(y_hat_fda, test_stroke_t$stroke)
cm_fda
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke         6        29
##   no_stroke     36       911
##
##                Accuracy : 0.9338
##                  95% CI : (0.9164, 0.9485)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 0.9997
##
##                   Kappa : 0.1217
##
##  Mcnemar's Test P-Value : 0.4568
##
##             Sensitivity : 0.14286
##             Specificity : 0.96915
##          Pos Pred Value : 0.17143
##          Neg Pred Value : 0.96199
##              Prevalence : 0.04277
##          Detection Rate : 0.00611
##    Detection Prevalence : 0.03564
##       Balanced Accuracy : 0.55600
##
##        'Positive' Class : stroke
##
```

The results are still deplorable, but this model already makes a difference with the others.

3.2 Balanced data

It is time to get serious and train models after solving one of the main obstacles: the imbalance of the data.

This, as we will see, improves the metrics, but we cannot expect outstanding results (a high Accuracy, along with high Sensitivity and Specificity). This, as we saw in the exploratory section, does not seem possible, since there are no predictor variables capable of clearly separating the classes.

In our approach, we have preferred to sacrifice Accuracy in favor of Sensitivity. At the same time, as already mentioned, we wanted to keep the Specificity above 70%.

It is important to say that for modeling purposes, the balancing methods have been applied only on the scaled and centralized training set. The test set has remained unbalanced.

3.2.1 Oversampling Method

For this and the other methods to generate a balanced training set, the steps described in *Practical Guide to deal with Imbalanced Classification Problems in R* have been followed.

The code bellow illustrates how to generate the oversampled training set, and print the table with the resulting new frequencies:

```
# _________________________########
# Oversampling #####
# _________________________########

n_over = sum(train_stroke_t$stroke == "no_stroke")

set.seed(1969, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1969)`
# Every time we run the code, we get a different ovun.sample
# Accuracy and balanced accuracy strongly depends on this random process.

train_stroke_over <- ovun.sample(stroke ~ ., data = train_stroke_t,
                                 method = "over", N = n_over*2)$data

# Relevel "stroke" "no_stroke" factors: positive class: "stroke" ####
train_stroke_over$stroke <- relevel(train_stroke_over$stroke, ref = "stroke")

# Strokes distribution #####
table(train_stroke_over$stroke) %>%
  kable()
```

Var1	Freq
stroke	3759
no_stroke	3759

We will now review the results of the different models trained with the set generated by oversampling.

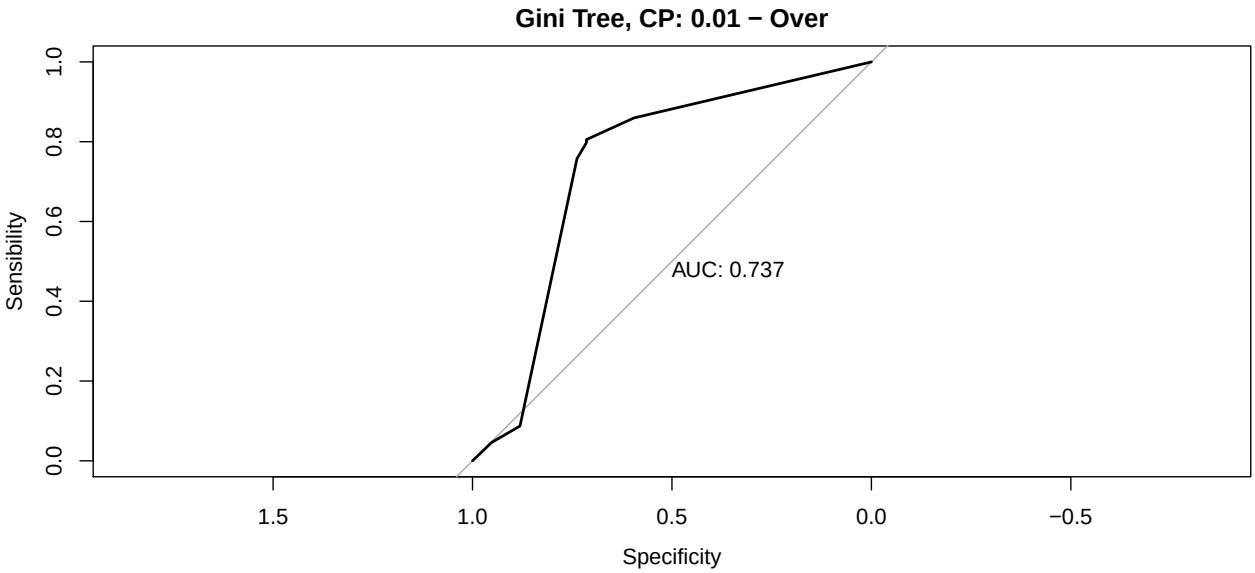
3.2.1.1 Gini Tree, CP = 0.01, minslit = 20, minbucket round 20/3, maxdepht = 30 This model has an Accuracy close to 76%. Its Sensitivity is around 74%, and its Specificity is 76%. The Balanced Accuracy is close to 75%. They are much better results than the previous ones. However, for the type of problem in question, the sensitivity is low. It has wrongly classified as negative 11 of 42 cases.

```
# RPART native #######
# Recursive Partitioning and Regression Trees ######

# Gini Tree, CP = 0.01, minslit = 20, minbucket round 20/3, maxdepht = 30 ####
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
gini_tree_cp0.01_over <- rpart(stroke ~ .,
```

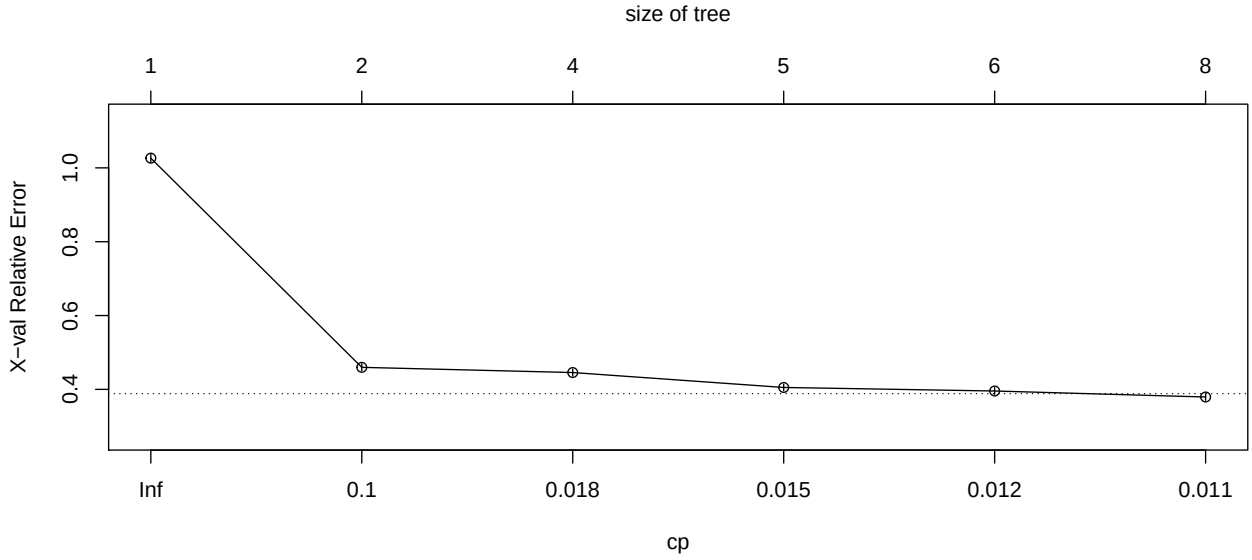
```
                                data = train_stroke_over)

y_hat_gini_tree_cp0.01_over <- predict(gini_tree_cp0.01_over, test_stroke_t,
                                       type = "class")
# Model Evaluation ::::::::

cm_gini_tree_cp0.01_over <- confusionMatrix(y_hat_gini_tree_cp0.01_over,
                                            test_stroke_t$stroke)
cm_gini_tree_cp0.01_over
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        31       228
##   no_stroke     11       712
##
##                Accuracy : 0.7566
##                  95% CI : (0.7285, 0.7832)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1429
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.73810
##             Specificity : 0.75745
##          Pos Pred Value : 0.11969
##          Neg Pred Value : 0.98479
##              Prevalence : 0.04277
##          Detection Rate : 0.03157
##    Detection Prevalence : 0.26375
##       Balanced Accuracy : 0.74777
##
##        'Positive' Class : stroke
##
```

The area under the ROC curve is around 0.74.



To obtain these results, the model has selected as best tunes: CP = 0.011, size of tree = 8:



From a CP of 0.015 the improvement is minimal.

The variables considered have been:

Variable importance

age	ever_married	work_type	bmi	smoking_status	avg_glucose_level	gender
52	14	9	9	8	7	1

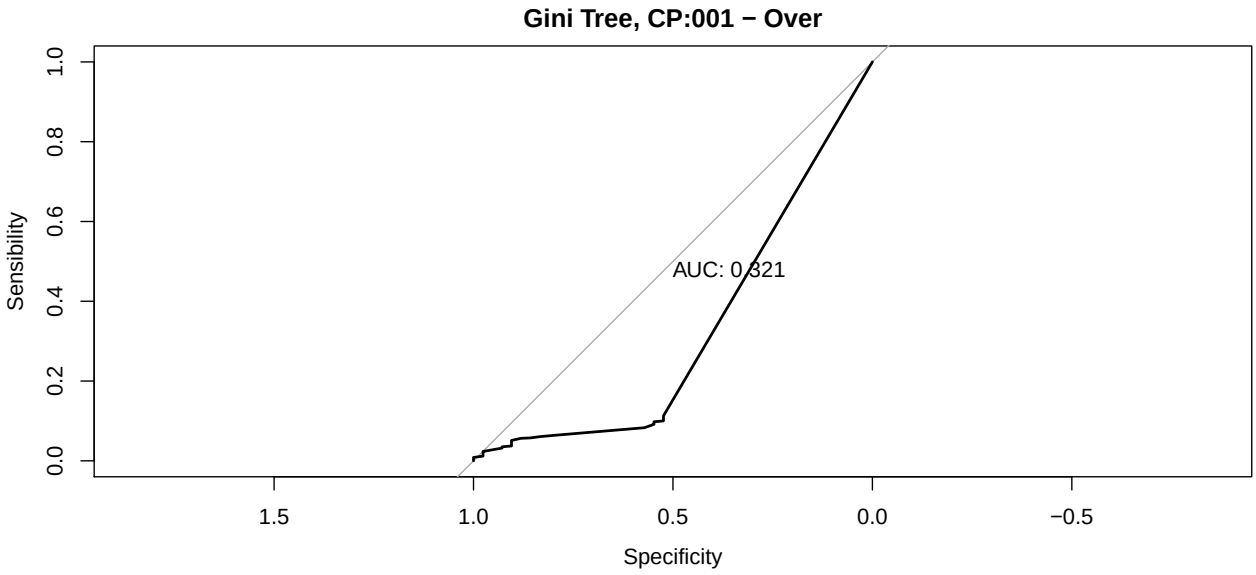
3.2.1.2 Gini Tree, CP = 0.001, minslit = 20, minbucket round 20/3, maxdepht = 3 With balanced data, overtraining does not give good results in terms of Sensitivity. Although we get better Accuracy, the sensitivity drops to 48%, and the Balanced Accuracy does not reach 70%:

```
# Gini Tree, CP = 0.001, minslit = 20, minbucket round 20/3, maxdepht = 30 #####
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

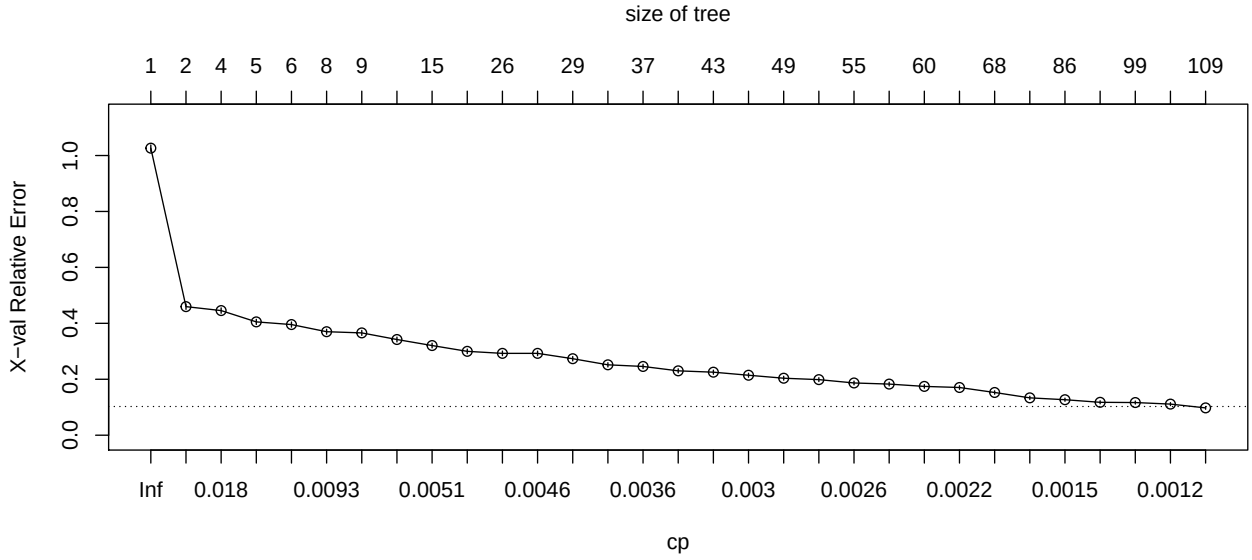
```r
gini_tree_cp0.001_over <- rpart(stroke ~.,
                                data = train_stroke_over,
                                parms=list(split=c("gini")),
                                cp = 0.001)

y_hat_gini_tree_cp0.001_over <- predict(gini_tree_cp0.001_over, test_stroke_t,
                                        type = "class")

# Model Evaluation ::::::::

cm_gini_tree_cp0.001_over <- confusionMatrix(y_hat_gini_tree_cp0.001_over,
                                             test_stroke_t$stroke)
cm_gini_tree_cp0.001_over
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        20       106
##   no_stroke     22       834
##
##                Accuracy : 0.8697
##                  95% CI : (0.847, 0.8901)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1859
##
##  Mcnemar's Test P-Value : 2.197e-13
##
##             Sensitivity : 0.47619
##             Specificity : 0.88723
##          Pos Pred Value : 0.15873
##          Neg Pred Value : 0.97430
##              Prevalence : 0.04277
##          Detection Rate : 0.02037
##    Detection Prevalence : 0.12831
##       Balanced Accuracy : 0.68171
##
##        'Positive' Class : stroke
##
```

The area under the curve (bellow 0.5) tells us that the model is clearly deficient:



In this case, overtraining is evident if we look at the size of the tree (109).



3.2.1.3 Default Gini Tree & Cost Matrix 3 to 1 In this model, we have experimented with a cost matrix, giving preference to Sensitivity (a false negative is 3 times worse than a false positive). If we increase the cost, we gain Sensitivity, but we lose Balanced Accuracy):

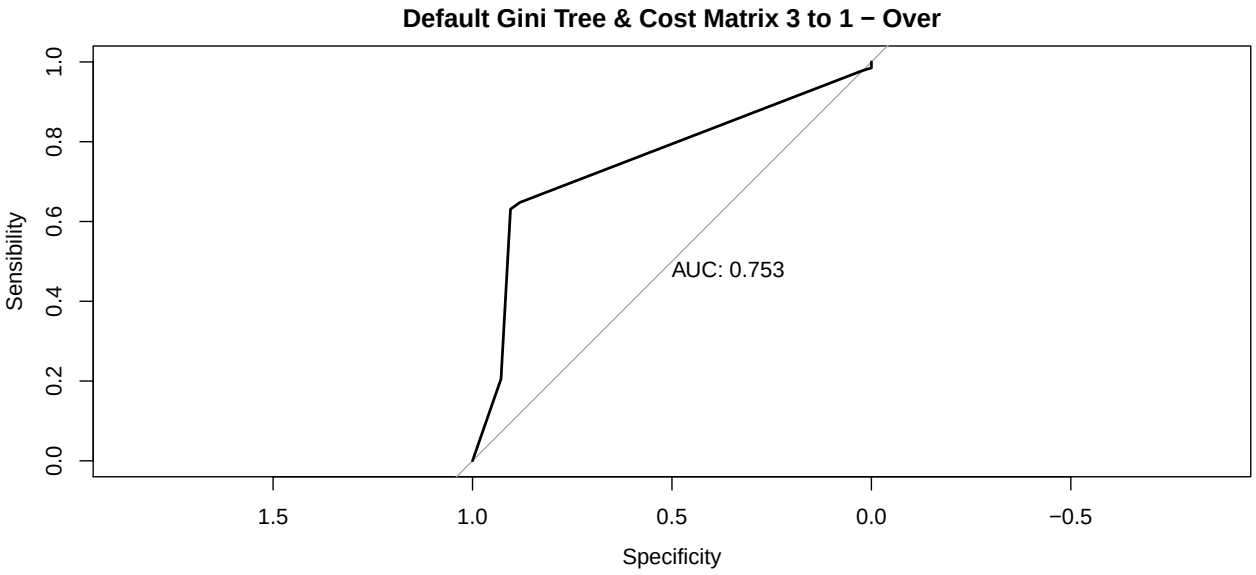
```
# Default Gini Tree & Cost Matrix 3 to 1 #####
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
cost_matrix_tree_over <- rpart(stroke ~ .,
                               data = train_stroke_over,
                               parms=list(
                                 loss=matrix(c(0,3,1,0),
                                             # A false negative is 3 times worse than a false positive
                                             byrow=TRUE,
                                             nrow=2)))

y_hat_cost_matrix_tree_over <- predict(cost_matrix_tree_over, test_stroke_t,
```

```
                                        type = "class")

# Model Evaluation ::::::::

cm_cost_matrix_tree_over <- confusionMatrix(y_hat_cost_matrix_tree_over,
                                            test_stroke_t$stroke)
cm_cost_matrix_tree_over
```

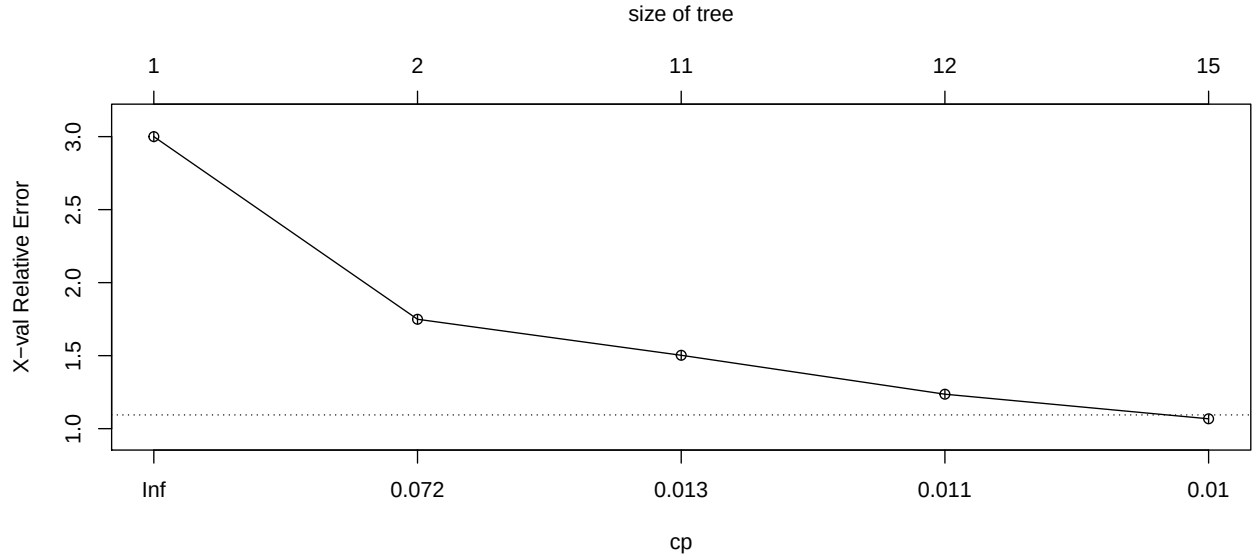
```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        38       347
##   no_stroke      4       593
##
##                Accuracy : 0.6426
##                  95% CI : (0.6117, 0.6726)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1093
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.90476
##             Specificity : 0.63085
##          Pos Pred Value : 0.09870
##          Neg Pred Value : 0.99330
##              Prevalence : 0.04277
##          Detection Rate : 0.03870
##    Detection Prevalence : 0.39206
##       Balanced Accuracy : 0.76781
##
##        'Positive' Class : stroke
##
```

With this model, the Sensitivity rises to 90%, but the Specificity barely reaches 63%.

The area under the ROC curve exceeds 0.75.



The CP is 0.01, as specified, and the tree size is reasonable:



The variables considered have been:

Variable importance

age	bmi	work_type	ever_married	work_type	avg_glucose_level	gender
46	18	14	9	13	4	2

Smoking Status and Hypertension also appear, but with less importance.

3.2.1.4 RPART caret As you might expect, the results of this model are similar to those of native default rpart (Gini Tree, CP = 0.01, minslit = 20, minbucket round 20/3, maxdepht = 30).

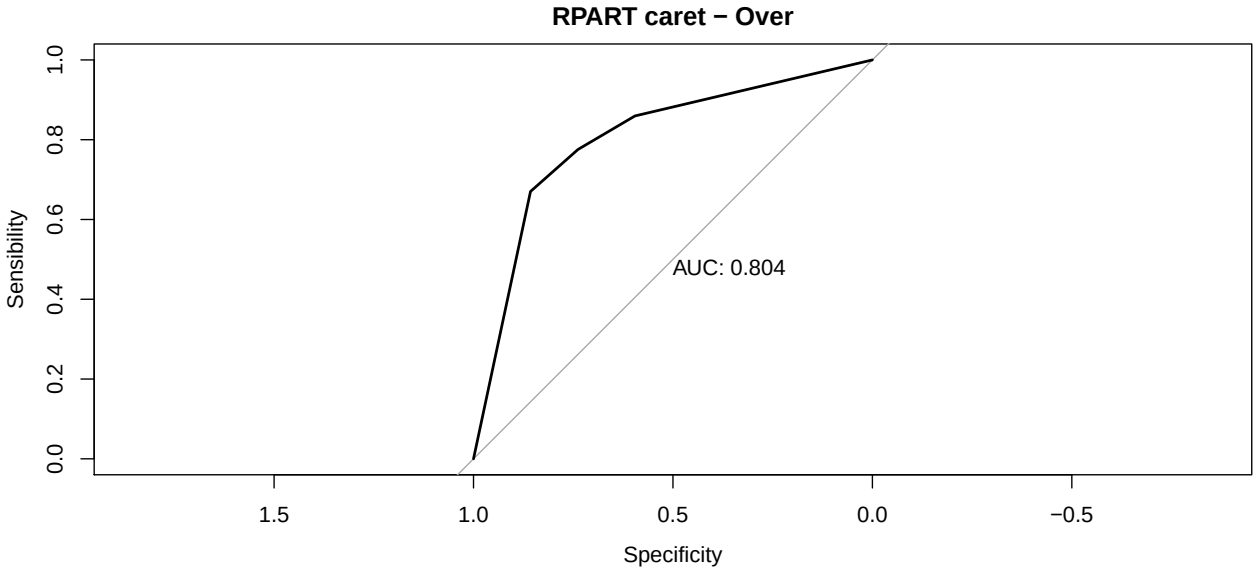
```
# RPART caret ####
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
train_caret_tree_over <- train(stroke ~ ., method = "rpart",
```

```
                                data = train_stroke_over,
                                trControl = ctrl,
                                metric="ROC")

y_hat_caret_tree_over <- predict(train_caret_tree_over, test_stroke_t)

# Model Evaluation ::::::::

cm_caret_tree_over <- confusionMatrix(y_hat_caret_tree_over, test_stroke_t$stroke)
cm_caret_tree_over
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        31       211
##   no_stroke     11       729
##
##                Accuracy : 0.7739
##                  95% CI : (0.7465, 0.7997)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1569
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.73810
##             Specificity : 0.77553
##          Pos Pred Value : 0.12810
##          Neg Pred Value : 0.98514
##              Prevalence : 0.04277
##          Detection Rate : 0.03157
##    Detection Prevalence : 0.24644
##       Balanced Accuracy : 0.75681
##
##        'Positive' Class : stroke
##
```

With a Sensitivity of around 74%, and a Specificity of 78%, this is the best model achieved so far. The Balanced Accuracy is around 76%.

The area under the ROC curve reaches 0.8:



The final CP was 0.01729183, and the variables considered were:

Variable importance

age	ever_marriedYes	bmi	smoking_statusUnknown	avg_glucose_level
61	18	8	6	6

avg_glucose_level	smoking_statussmokes
6	1

3.2.1.5 K-Nearest-Neighbor As we anticipated in the exploratory section, a model based on the nearest neighbors is not suitable in this case.

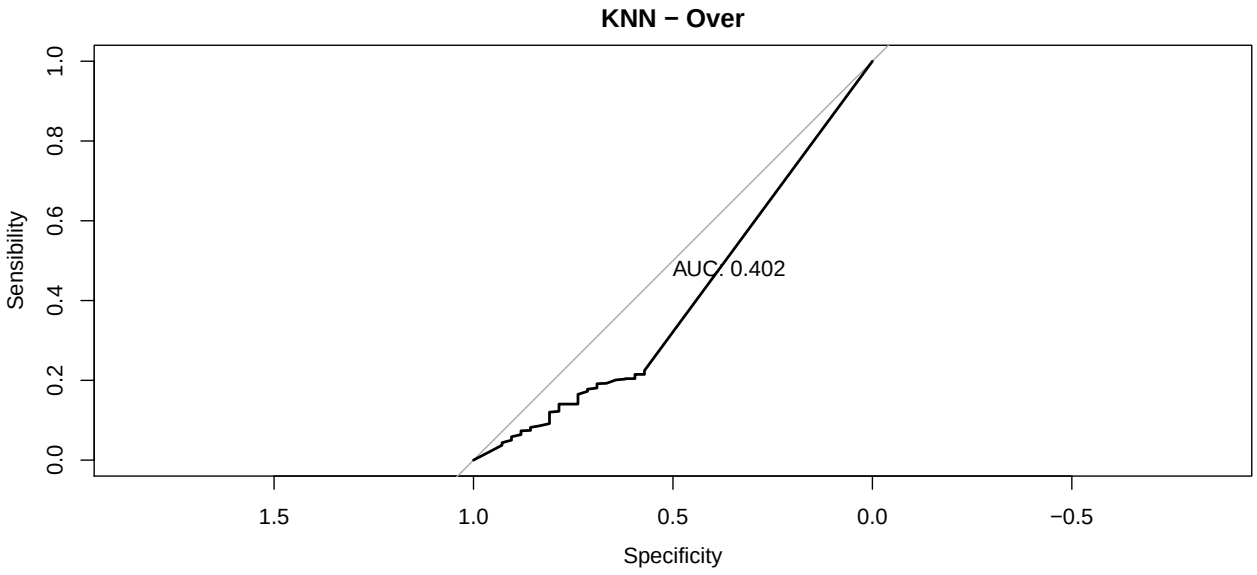
```
# K-Nearest-Neighbor #######
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
train_knn_over <- train(stroke ~ ., method = "knn",
                        data = train_stroke_over,
                        trControl = ctrl,
                        metric="ROC")

y_hat_knn_over <- predict(train_knn_over, test_stroke_t)

# Model Evaluation ::::::::

cm_knn_over <- confusionMatrix(y_hat_knn_over, test_stroke_t$stroke)
cm_knn_over
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        18       211
##   no_stroke     24       729
##
```

```
##                Accuracy : 0.7607
##                  95% CI : (0.7327, 0.7871)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0653
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.42857
##             Specificity : 0.77553
##          Pos Pred Value : 0.07860
##          Neg Pred Value : 0.96813
##              Prevalence : 0.04277
##          Detection Rate : 0.01833
##    Detection Prevalence : 0.23320
##       Balanced Accuracy : 0.60205
##
##        'Positive' Class : stroke
##
```

Although the Accuracy exceeds 76% the Sensitivity only touches 43%. With a Specificity of 78%, the Balanced Accuracy just exceeds 60%

The area under the ROC curve is less than 0.5:



In this case, the best tune for k = 9.

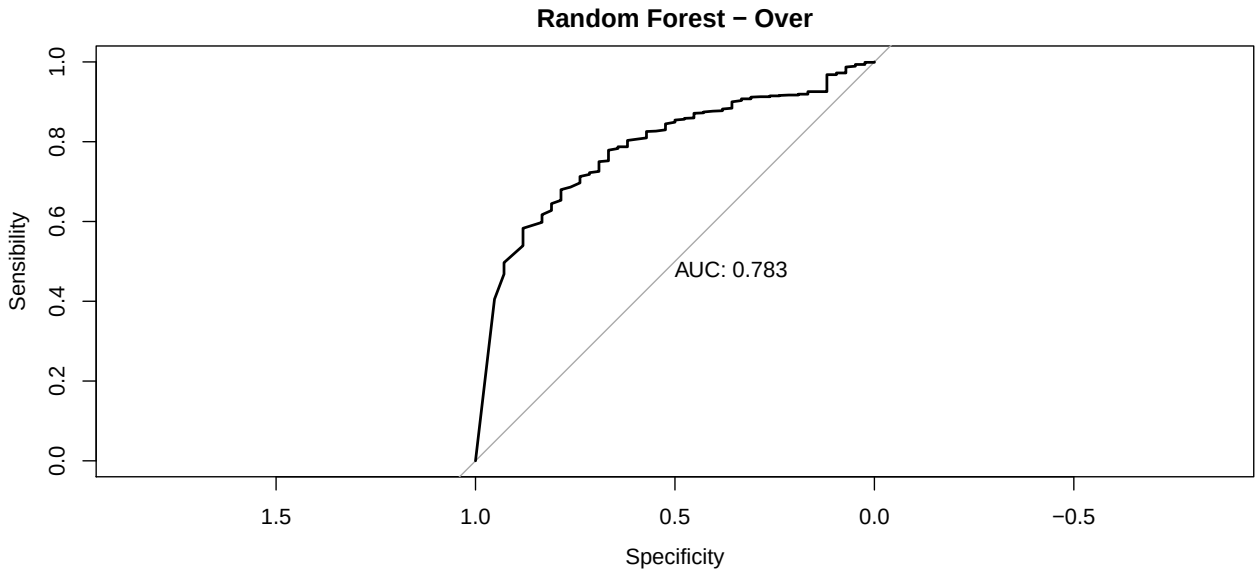
3.2.1.6 Random Forest This model requires the longest calculation time. Despite that, it is one of the worst performers in terms of Sensitivity and Balanced Accuracy.

```
# RANDOM FOREST ########
# it takes time! ######
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
train_rf_over <- train(stroke ~ ., method = "rf",
                       data = train_stroke_over,
                       trControl = ctrl,
```

```
                        metric="ROC")

y_hat_rf_over <- predict(train_rf_over, test_stroke_t, type = "raw")

# Model Evaluation ::::::::

cm_rf_over <- confusionMatrix(y_hat_rf_over, test_stroke_t$stroke)
cm_rf_over
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke         3        13
##   no_stroke     39       927
##
##                Accuracy : 0.947
##                  95% CI : (0.9311, 0.9602)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 0.9473063
##
##                   Kappa : 0.0818
##
##  Mcnemar's Test P-Value : 0.0005265
##
##             Sensitivity : 0.071429
##             Specificity : 0.986170
##          Pos Pred Value : 0.187500
##          Neg Pred Value : 0.959627
##              Prevalence : 0.042770
##          Detection Rate : 0.003055
##    Detection Prevalence : 0.016293
##       Balanced Accuracy : 0.528799
##
##        'Positive' Class : stroke
##
```

The sensitivity of this model barely reaches 7%. Surprisingly, it has an area under the ROC curve of 0.78 (something we must analyze in future steps to fully understand).




```
## Area under the curve: 0.7827
```

In this case, the best tune for "mtry" parameter (randomly selected predictors) was 9.

3.2.1.7 Neural Network This is another model with a comparatively good performance.

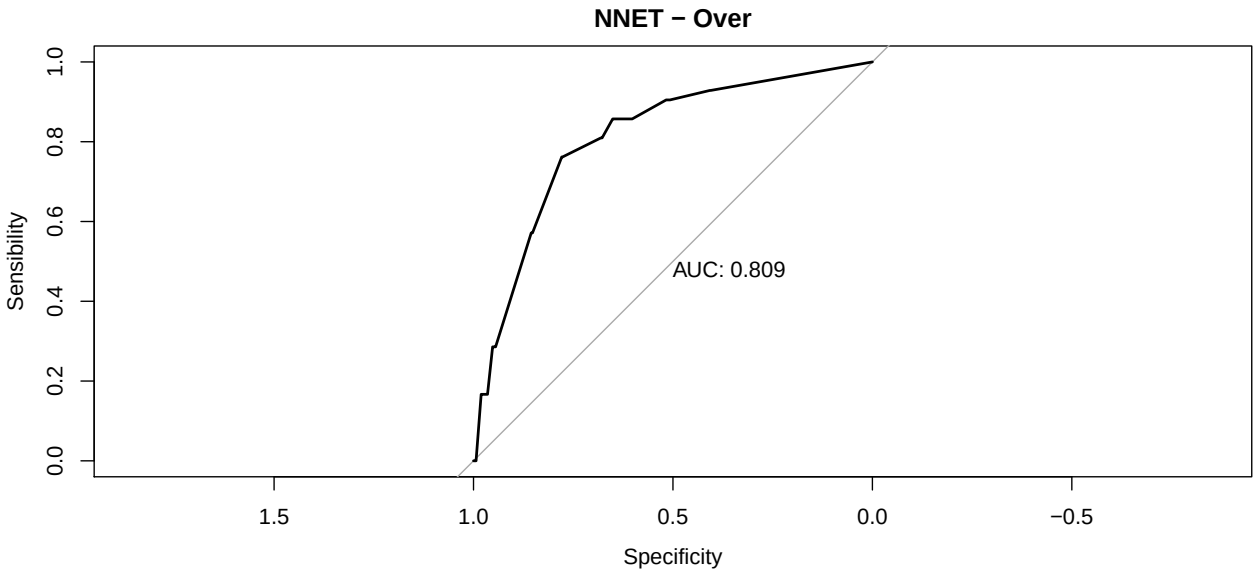
```
# NNET: great variability !!!#####
# Neural Network ####
# It takes some time!

set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
train_nnet_over <- train(stroke ~ ., method = "nnet",
                         data = train_stroke_over,
                         trControl = ctrl,
                         trace = FALSE,
                         metric="ROC")

y_hat_nnet_over <- predict(train_nnet_over, test_stroke_t, type = "raw")

# Model Evaluation ::::::::

cm_nnet_over <- confusionMatrix(as.factor(y_hat_nnet_over), test_stroke_t$stroke)
cm_nnet_over
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        32       209
##   no_stroke     10       731
##
##                Accuracy : 0.777
##                  95% CI : (0.7496, 0.8027)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1653
```

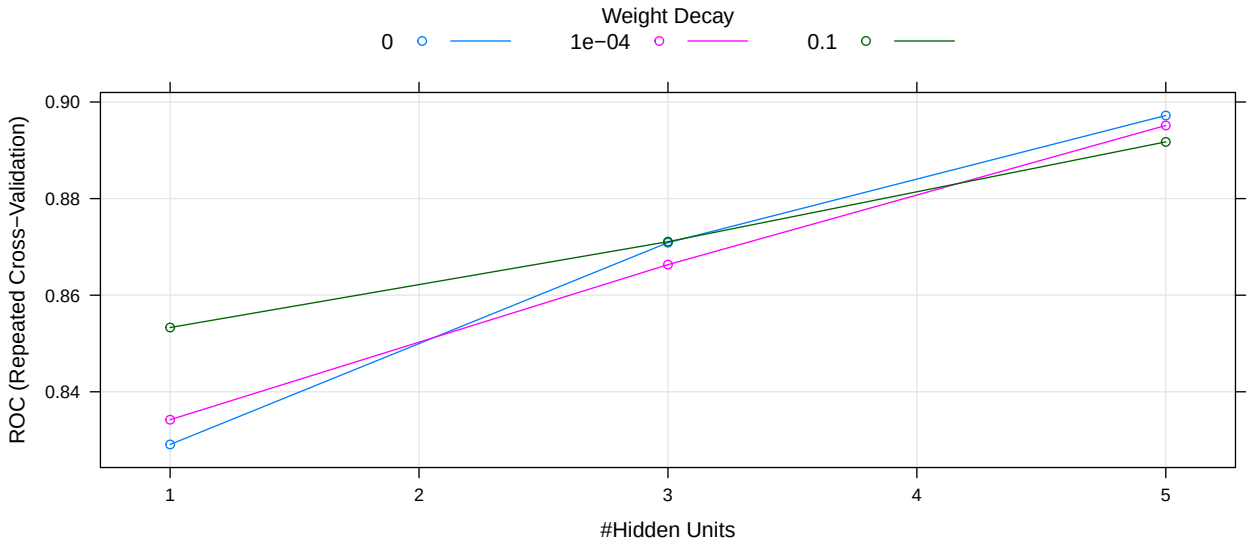
```
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.76190
##             Specificity : 0.77766
##          Pos Pred Value : 0.13278
##          Neg Pred Value : 0.98650
##              Prevalence : 0.04277
##          Detection Rate : 0.03259
##    Detection Prevalence : 0.24542
##       Balanced Accuracy : 0.76978
##
##        'Positive' Class : stroke
##
```

Its Accuracy is almost 77%. The Sensitivity exceeds 76%, and the Specificity is close to 78%. Balanced Accuracy is also around 77%.

The area under de ROC curve is 0.81:



The selected parameters were Weight Decay = 0 and Hidden Units = 5:



3.2.1.8 Flexible Discriminant Analysis This model is another one that stands out, comparatively speaking. As we will see later, it is the one that gives the best results with the "better estimates" method to generate balanced training data.

With the oversampling method, the results are as follows:

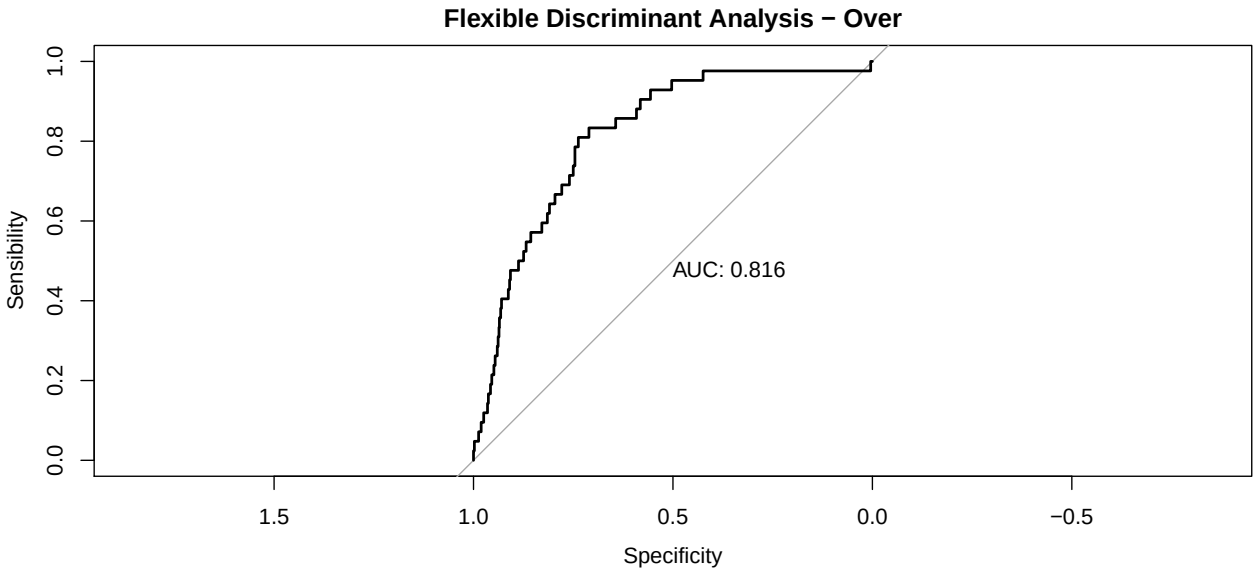
```
# FDA #####
# Flexible Discriminant Analysis #####
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
train_fda_over <- train(stroke ~ ., method = "fda",
                        data = train_stroke_over,
                        trControl = ctrl,
                        metric="ROC")

y_hat_fda_over <- predict(train_fda_over, test_stroke_t)

# Model Evaluation ::::::::

cm_fda_over <- confusionMatrix(y_hat_fda_over, test_stroke_t$stroke)
cm_fda_over
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        31       239
##   no_stroke     11       701
##
##                Accuracy : 0.7454
##                  95% CI : (0.717, 0.7724)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1347
##
##  Mcnemar's Test P-Value : <2e-16
##
```

```
##             Sensitivity : 0.73810
##             Specificity : 0.74574
##          Pos Pred Value : 0.11481
##          Neg Pred Value : 0.98455
##              Prevalence : 0.04277
##          Detection Rate : 0.03157
##    Detection Prevalence : 0.27495
##       Balanced Accuracy : 0.74192
##
##        'Positive' Class : stroke
##
```

Accuracy is close to 75%. Sensitivity is 74%, Specificity reaches 75%, and Balanced Accuracy is 74%

Area under the ROC curve reaches 0.82:



The final values used for the model were degree = 1 and nprune = 16.

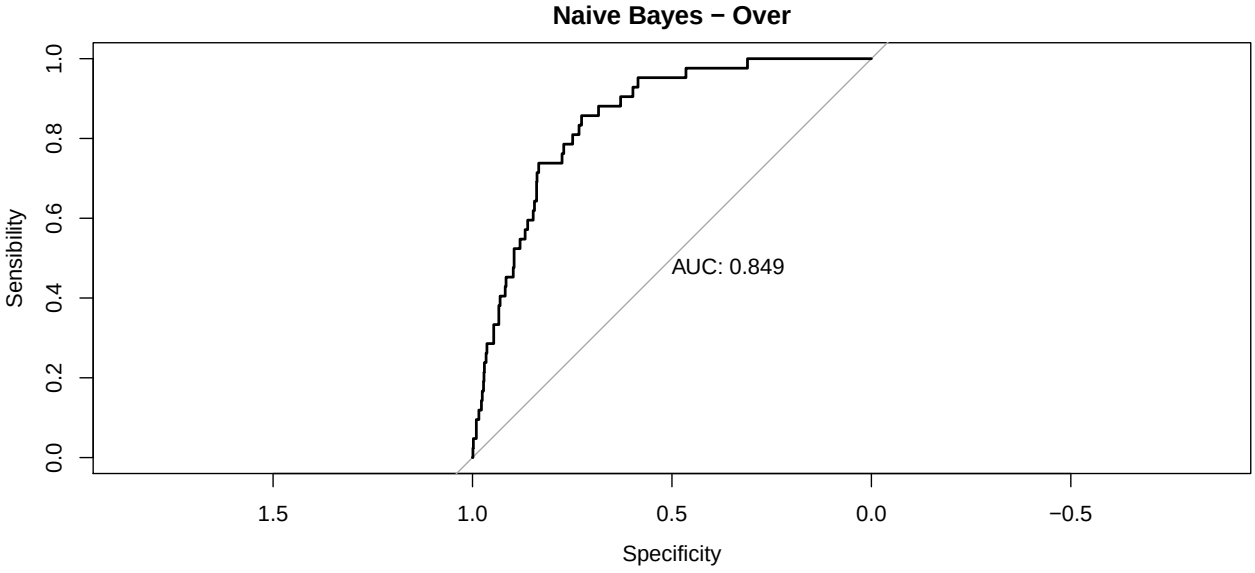
3.2.1.9 Naive Bayes Finally, the Naives Bayes model is another that has some surprises in store. Its Accuracy is very high (greater than 93%), but the Sensitivity is very low (26%). Balanced Accuracy stays at 61%.

```
# NAIVE BAYES ######
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
train_naiveBayes_over <- train(stroke ~ ., method = "naive_bayes",
                               data = train_stroke_over,
                               trControl = ctrl,
                               metric="ROC")

y_hat_naiveBayes_over <- predict(train_naiveBayes_over, test_stroke_t)

# Model Evaluation ::::::::

cm_naiveBayes_over <- confusionMatrix(y_hat_naiveBayes_over, test_stroke_t$stroke)
cm_naiveBayes_over
```

```
## Confusion Matrix and Statistics
```

```
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        11        33
##   no_stroke     31       907
##
##                Accuracy : 0.9348
##                  95% CI : (0.9175, 0.9495)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 0.9996
##
##                   Kappa : 0.2218
##
##  Mcnemar's Test P-Value : 0.9005
##
##             Sensitivity : 0.26190
##             Specificity : 0.96489
##          Pos Pred Value : 0.25000
##          Neg Pred Value : 0.96695
##              Prevalence : 0.04277
##          Detection Rate : 0.01120
##    Detection Prevalence : 0.04481
##       Balanced Accuracy : 0.61340
##
##        'Positive' Class : stroke
##
```

Despite these results, the area under the ROC curve is close to 0.85. One of the next steps after the delivery of this report to analyze this result to fully understand it.



The final values used for the model were laplace = 0, usekernel = TRUE and adjust = 1.

As we have already warned, the results, although better, are not excessively good. To find a sensitivity greater than 85%, together with a Specificity greater than 75%, we will have to wait for the FDA model with the better estimates method.

3.2.2 Both Method

The following code illustrates the how to generate the balanced training set with the "both" method, and print the table with the resulting new frequencies:

```
# ________________________########
# Oversamplig and Undersampling: both #####
# ________________________########

n_both = sum(train_stroke$stroke == "stroke") + sum(train_stroke$stroke == "no_stroke")

set.seed(1969, sample.kind="Rounding")
# Every time we run the code, we get a different ovun.sample
# Accuracy and balanced accuracy strongly depends on this ramdom process.

train_stroke_both <- ovun.sample(stroke ~ ., data = train_stroke_t,
                                 method = "both", p = 0.5, N = n_both)$data

# Relevel "stroke" "no_stroke" factors: positive class: "stroke" ####
train_stroke_both$stroke <- relevel(train_stroke_both$stroke, ref = "stroke")

# Strokes distribution #####
table(train_stroke_both$stroke) %>%
  kable()
```

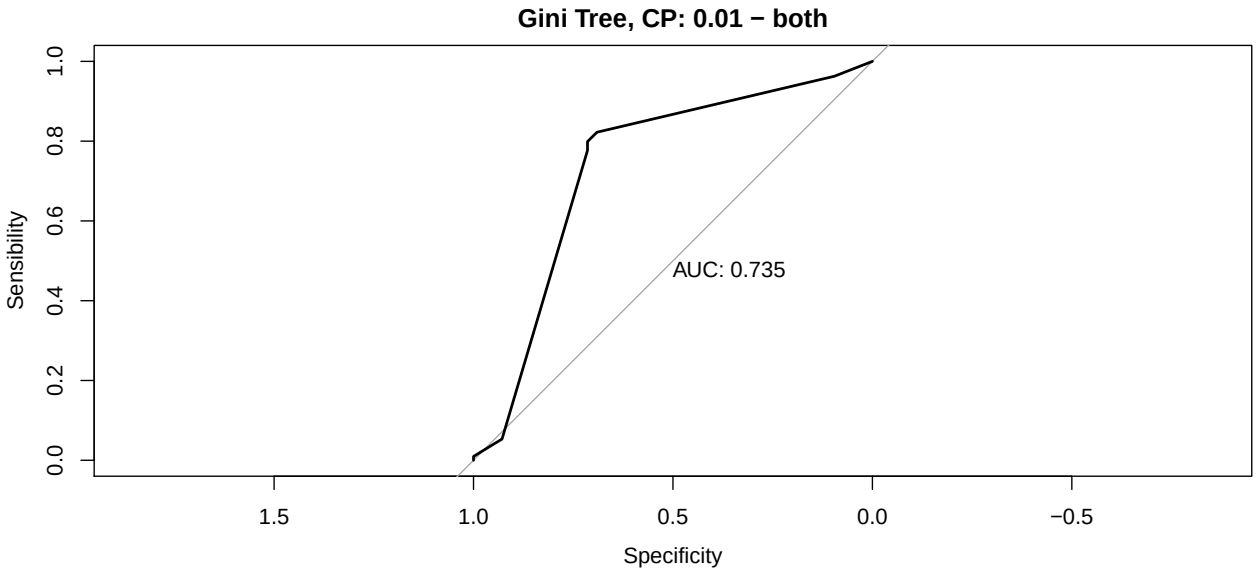
Var1	Freq
stroke	1987
no_stroke	1939

Here, we will only show the results.

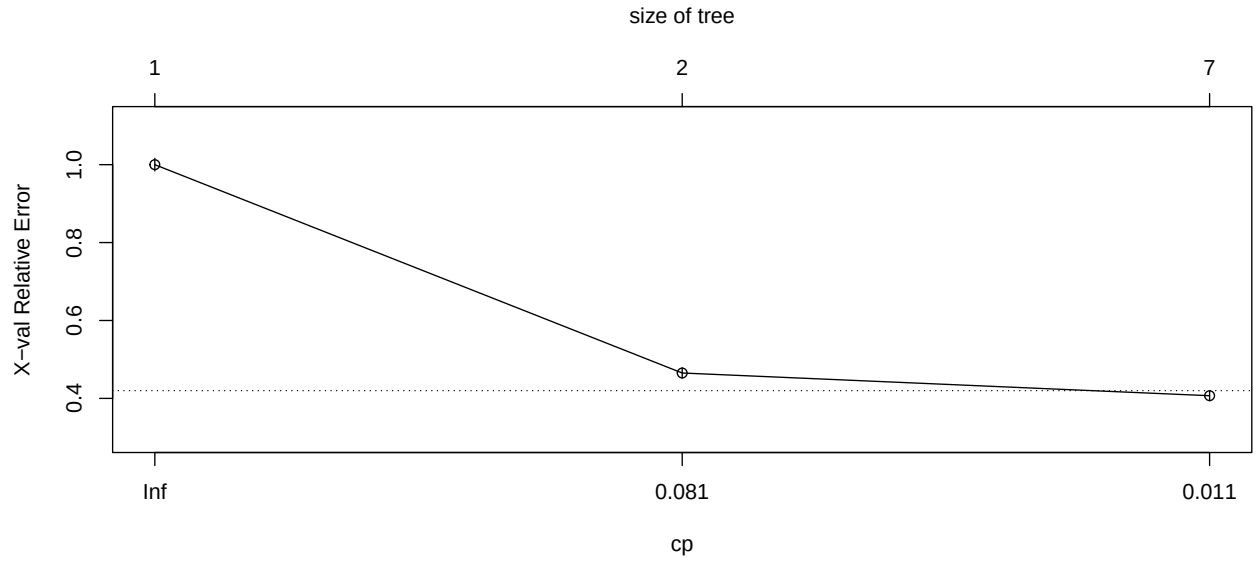
3.2.2.1 Gini Tree, CP = 0.01, minslit = 20, minbucket round 20/3, maxdepht = 30

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        30       189
##   no_stroke     12       751
##
##                Accuracy : 0.7953
##                  95% CI : (0.7687, 0.8201)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1703
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.71429
##             Specificity : 0.79894
##          Pos Pred Value : 0.13699
##          Neg Pred Value : 0.98427
##              Prevalence : 0.04277
```

```
##          Detection Rate : 0.03055
##    Detection Prevalence : 0.22301
##       Balanced Accuracy : 0.75661
##
##        'Positive' Class : stroke
##
```



To obtain these results, the model has selected as best tunes: CP = 0.011, size of tree = 7:



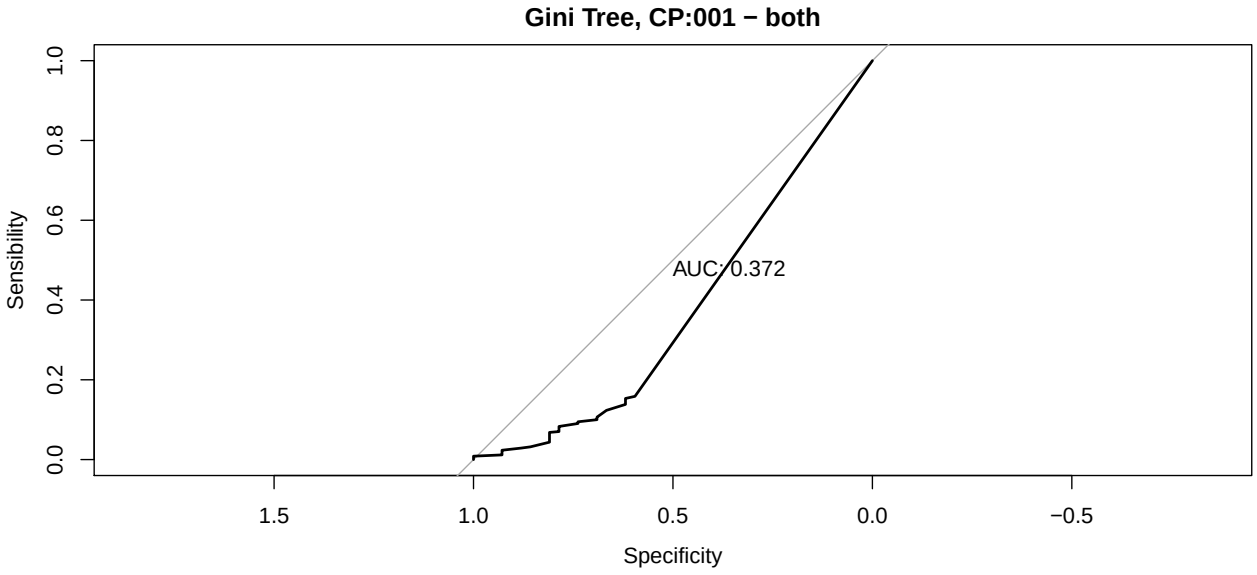
The variables considered have been:

Variable importance

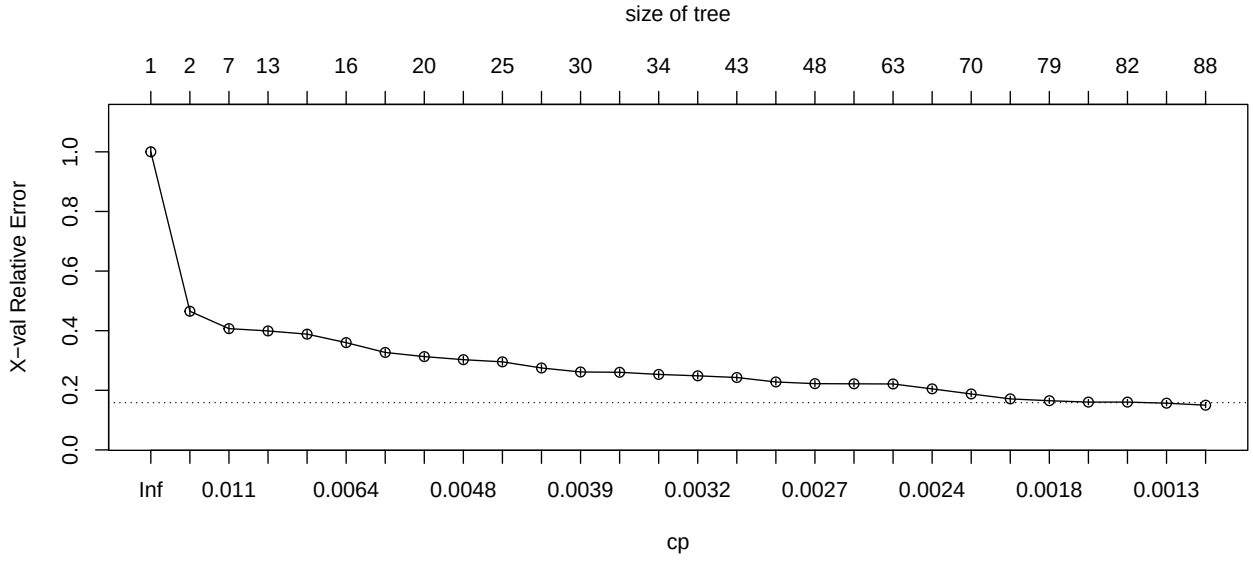
age	ever_married	smoking_status	avg_glucose_level	bmi	work_type
54	13	9	8	8	8

3.2.2.2 Gini Tree, CP = 0.001, minslit = 20, minbucket round 20/3, maxdepht = 3

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        16       144
##   no_stroke     26       796
##
##                Accuracy : 0.8269
##                  95% CI : (0.8017, 0.85)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0973
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.38095
##             Specificity : 0.84681
##          Pos Pred Value : 0.10000
##          Neg Pred Value : 0.96837
##              Prevalence : 0.04277
##          Detection Rate : 0.01629
##    Detection Prevalence : 0.16293
##       Balanced Accuracy : 0.61388
##
##        'Positive' Class : stroke
##
```

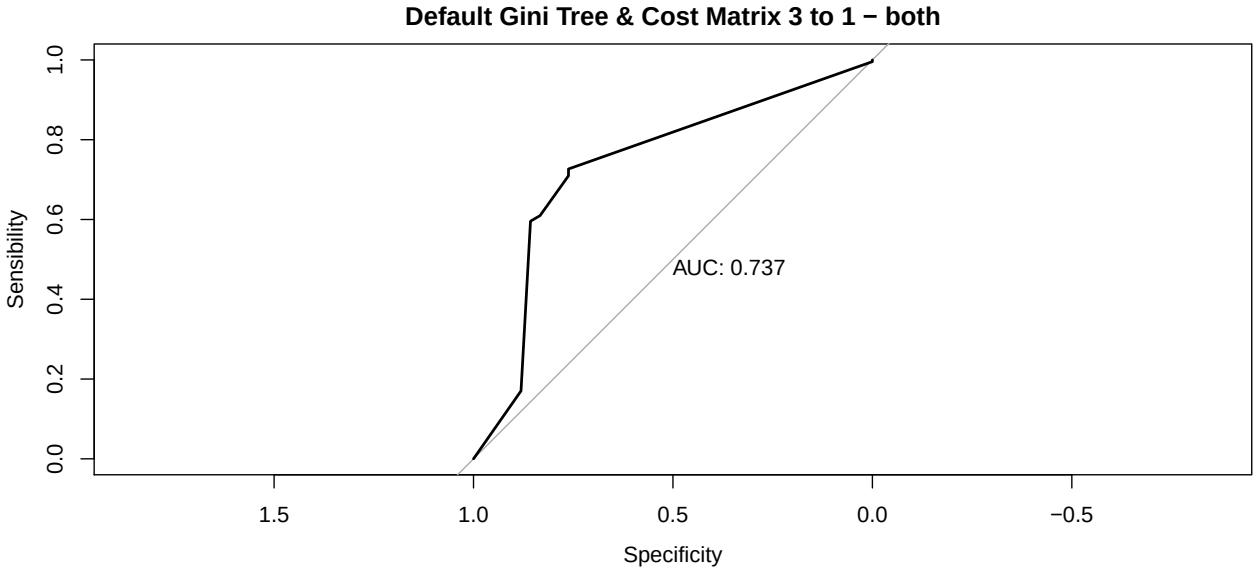


To obtain these results, the model has selected as best tunes: CP = 0.0013, size of tree 88:

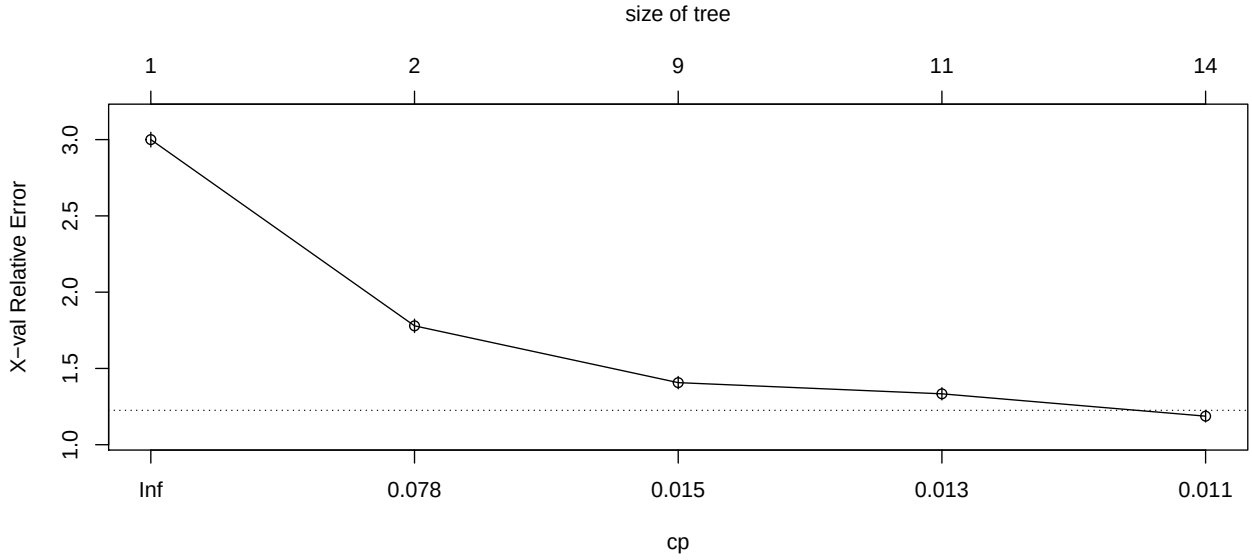


3.2.2.3 Default Gini Tree & Cost Matrix 3 to 1

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        36       380
##   no_stroke      6       560
##
##                Accuracy : 0.6069
##                  95% CI : (0.5756, 0.6376)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0862
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.85714
##             Specificity : 0.59574
##          Pos Pred Value : 0.08654
##          Neg Pred Value : 0.98940
##              Prevalence : 0.04277
##          Detection Rate : 0.03666
##    Detection Prevalence : 0.42363
##       Balanced Accuracy : 0.72644
##
##        'Positive' Class : stroke
##
```



To obtain these results, the model has selected as best tunes: CP = 0.011, size of tree = 14



The variables considered have been:

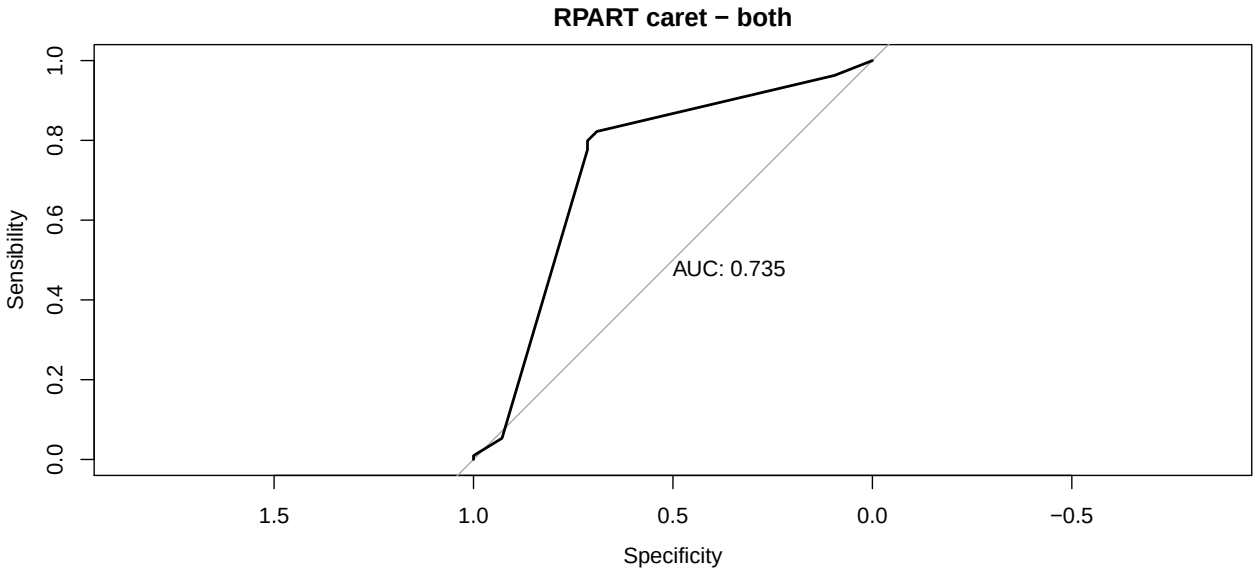
Variable importance

age	work_type	ever_married	bmi	avg_glucose_level	smoking_status
50	15	13	9	8	4

3.2.2.4 RPART caret

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        30       189
##   no_stroke     12       751
```

```
##
##                Accuracy : 0.7953
##                  95% CI : (0.7687, 0.8201)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1703
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.71429
##             Specificity : 0.79894
##          Pos Pred Value : 0.13699
##          Neg Pred Value : 0.98427
##              Prevalence : 0.04277
##          Detection Rate : 0.03055
##    Detection Prevalence : 0.22301
##       Balanced Accuracy : 0.75661
##
##        'Positive' Class : stroke
##
```



The final value used for the model was cp = 0.00988482.

The variables considered have been:

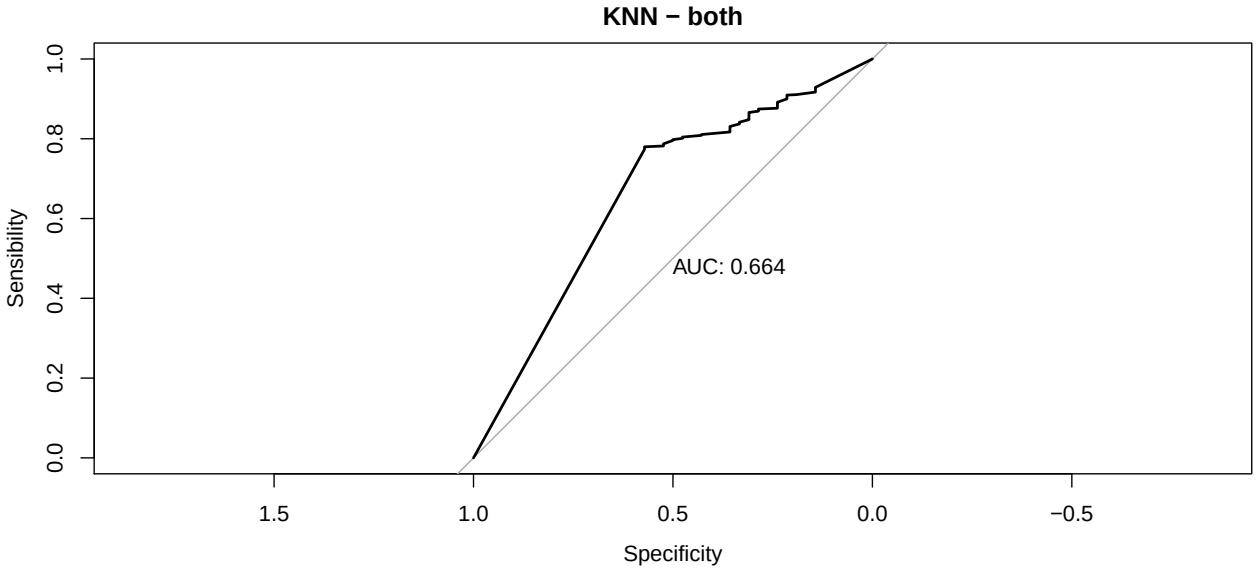
Variable importance

age	ever_marriedYes	avg_glucose_level	bmi	smoking_statusUnknown	smoking_statusnever smoked
58	14	9	9	6	3

3.2.2.5 K-Nearest-Neighbor

```
## Confusion Matrix and Statistics
```

```
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        24       213
##   no_stroke     18       727
##
##                Accuracy : 0.7648
##                  95% CI : (0.737, 0.791)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1072
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.57143
##             Specificity : 0.77340
##          Pos Pred Value : 0.10127
##          Neg Pred Value : 0.97584
##              Prevalence : 0.04277
##          Detection Rate : 0.02444
##    Detection Prevalence : 0.24134
##       Balanced Accuracy : 0.67242
##
##        'Positive' Class : stroke
##
```

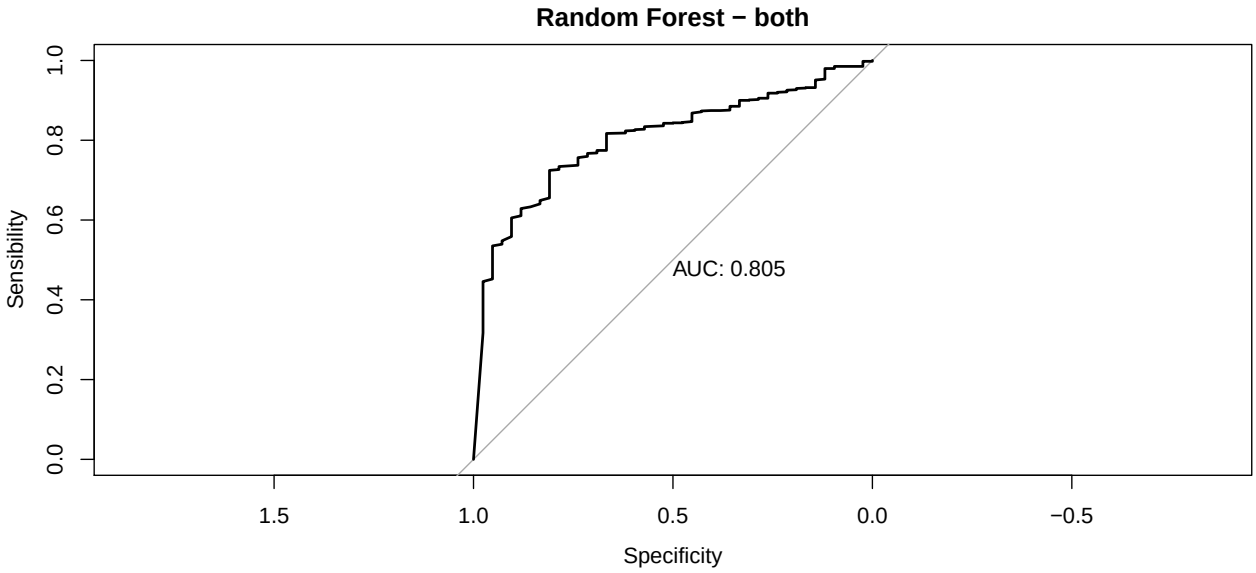


The final value used for the model was k = 5.

3.2.2.6 Ramdom Forest

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke         5        34
```

```
##   no_stroke     37       906
##
##                Accuracy : 0.9277
##                  95% CI : (0.9097, 0.9431)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : 0.0858
##
##  Mcnemar's Test P-Value : 0.8124
##
##             Sensitivity : 0.119048
##             Specificity : 0.963830
##          Pos Pred Value : 0.128205
##          Neg Pred Value : 0.960764
##              Prevalence : 0.042770
##          Detection Rate : 0.005092
##    Detection Prevalence : 0.039715
##       Balanced Accuracy : 0.541439
##
##        'Positive' Class : stroke
##
```

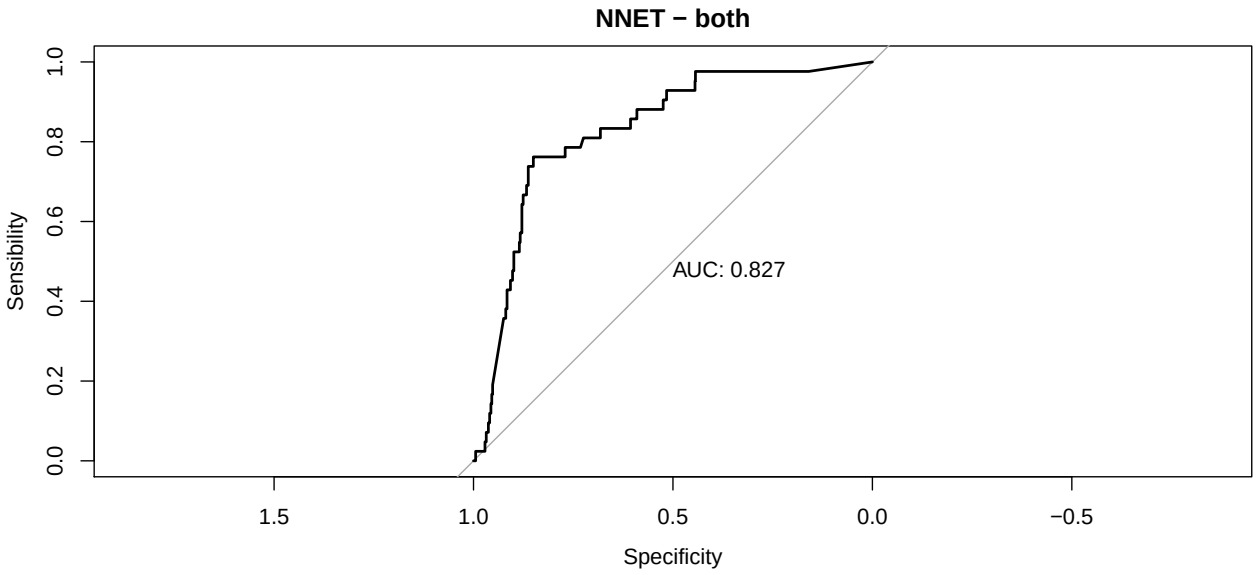


The final value used for the model was mtry = 9.

3.2.2.7 Neural Network

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        32       206
##   no_stroke     10       734
##
##                Accuracy : 0.78
##                  95% CI : (0.7528, 0.8056)
```

```
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1681
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.76190
##             Specificity : 0.78085
##          Pos Pred Value : 0.13445
##          Neg Pred Value : 0.98656
##              Prevalence : 0.04277
##          Detection Rate : 0.03259
##    Detection Prevalence : 0.24236
##       Balanced Accuracy : 0.77138
##
##        'Positive' Class : stroke
##
```

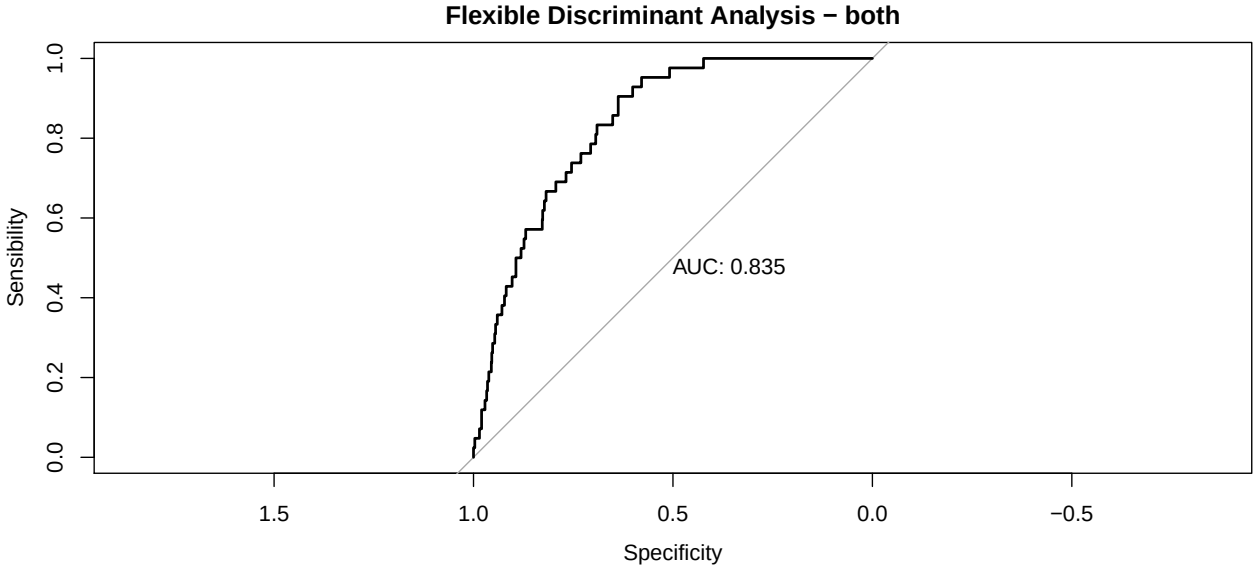


The final values used for the model were size = 5 and decay = 0.

3.2.2.8 Flexible Discriminant Analysis

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        31       243
##   no_stroke     11       697
##
##                Accuracy : 0.7413
##                  95% CI : (0.7127, 0.7685)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1318
```

```
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.73810
##             Specificity : 0.74149
##          Pos Pred Value : 0.11314
##          Neg Pred Value : 0.98446
##              Prevalence : 0.04277
##          Detection Rate : 0.03157
##    Detection Prevalence : 0.27902
##       Balanced Accuracy : 0.73979
##
##        'Positive' Class : stroke
##
```

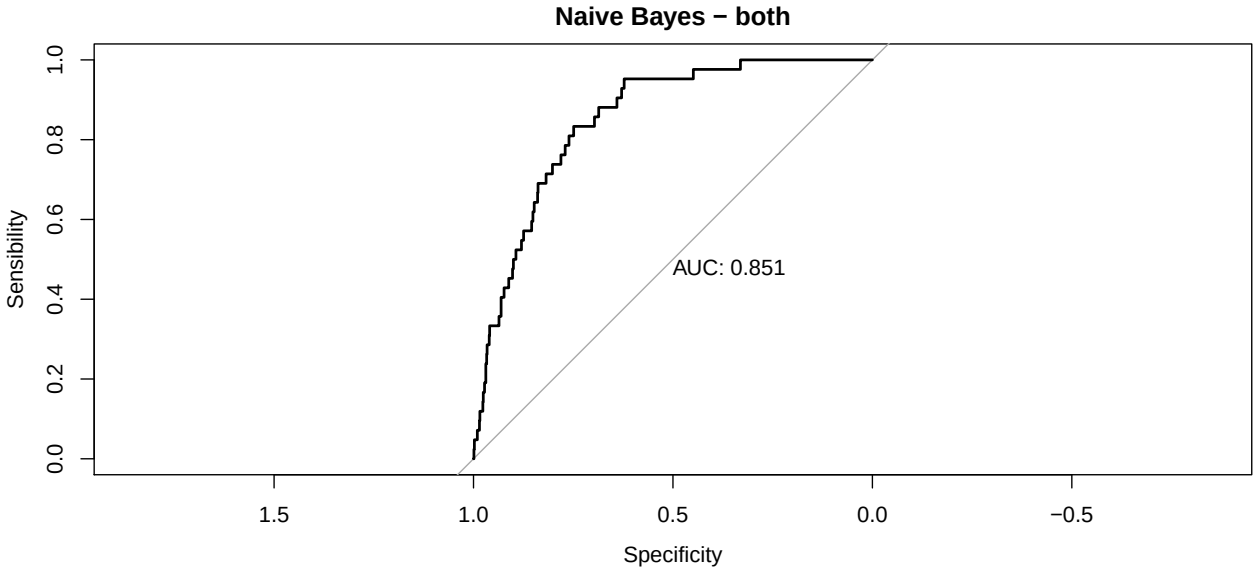


The final values used for the model were degree = 1 and nprune = 17.

3.2.2.9 Naive Bayes

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        11        32
##   no_stroke     31       908
##
##                Accuracy : 0.9358
##                  95% CI : (0.9187, 0.9504)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 0.9993
##
##                   Kappa : 0.2253
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.26190
```

```
##             Specificity : 0.96596
##          Pos Pred Value : 0.25581
##          Neg Pred Value : 0.96699
##              Prevalence : 0.04277
##          Detection Rate : 0.01120
##    Detection Prevalence : 0.04379
##       Balanced Accuracy : 0.61393
##
##        'Positive' Class : stroke
##
```



The final values used for the model were laplace = 0, usekernel = TRUE and adjust = 1.

3.2.3 Better Estimates Method

The following code illustrates the how to generate the balanced training set with better estimates, and print the table with the resulting frequencies:

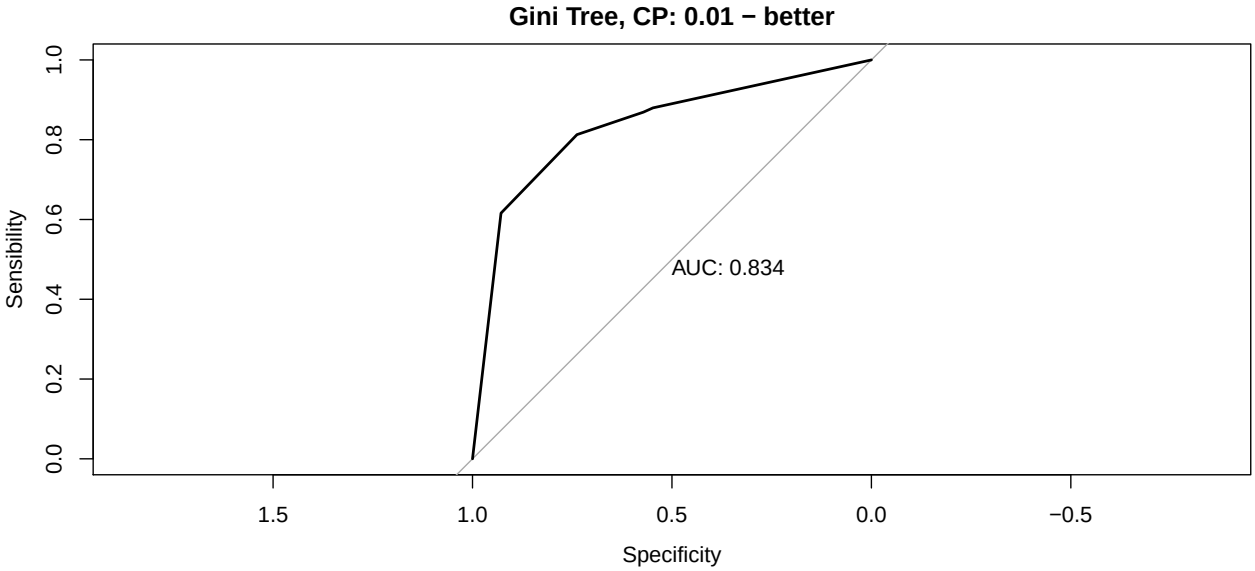
```
# ________________________########
# Better estimates ######
# ________________________########

set.seed(1969, sample.kind="Rounding")
# Every time we run the code, we get a different ovun.sample
# Accuracy and balanced accuracy strongly depends on this ramdom process.

train_stroke_better <- ROSE(stroke ~ ., data = train_stroke_t)$data

# Relevel "stroke" "no_stroke" factors: positive class: "stroke"
train_stroke_better$stroke <- relevel(train_stroke_better$stroke, ref = "stroke")

# Strokes distribution #####
table(train_stroke_better$stroke) %>%
  kable()
```

Var1	Freq
stroke	1987
no_stroke	1939

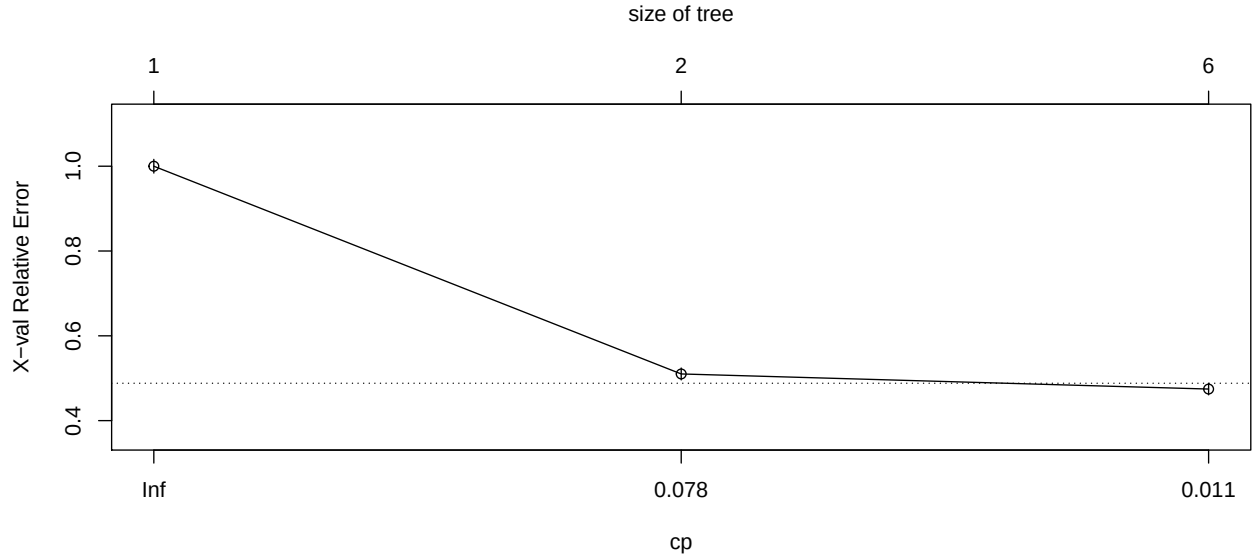
Again, we show only the results.

3.2.3.1 Gini Tree, CP = 0.01, minslit = 20, minbucket round 20/3, maxdepht = 30

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        31       176
##   no_stroke     11       764
##
##                Accuracy : 0.8096
##                  95% CI : (0.7836, 0.8337)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1915
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.73810
##             Specificity : 0.81277
##          Pos Pred Value : 0.14976
##          Neg Pred Value : 0.98581
##              Prevalence : 0.04277
##          Detection Rate : 0.03157
##    Detection Prevalence : 0.21079
##       Balanced Accuracy : 0.77543
##
##        'Positive' Class : stroke
##
```



To obtain these results, the model has selected as best tunes: CP = 0.011, size of tree = 6:



The variables considered have been:

Variable importance

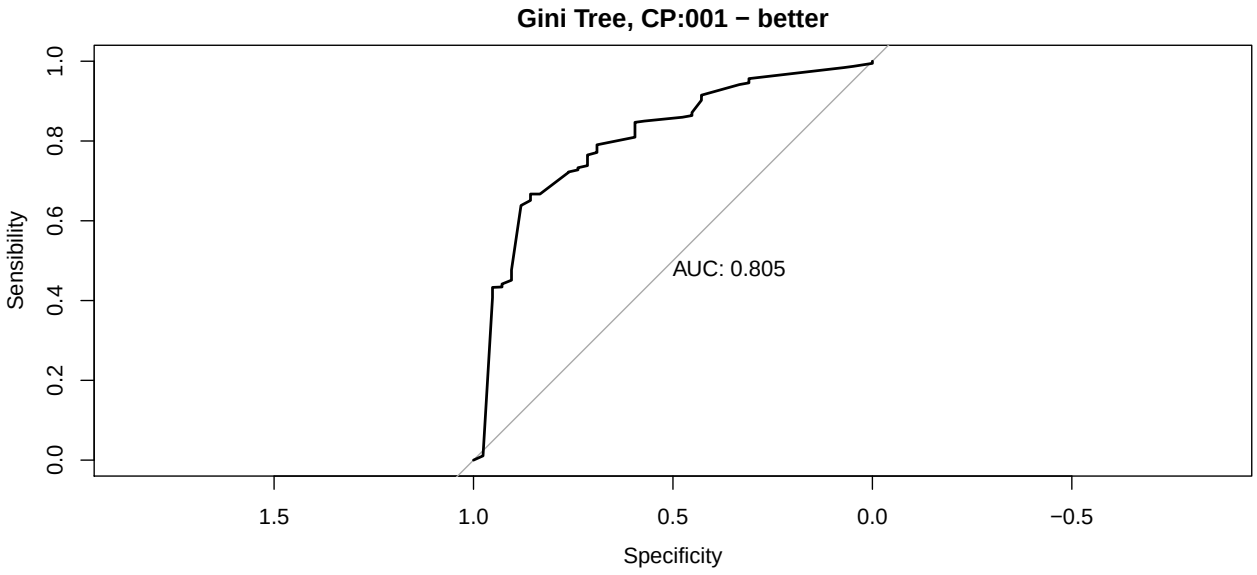
age	ever_married	work_type	smoking_status	avg_glucose_level	bmi	heart_disease	hypertension
55	15	11	6	6	5	2	1

3.2.3.2 Gini Tree, CP = 0.001, minslit = 20, minbucket round 20/3, maxdepht = 30

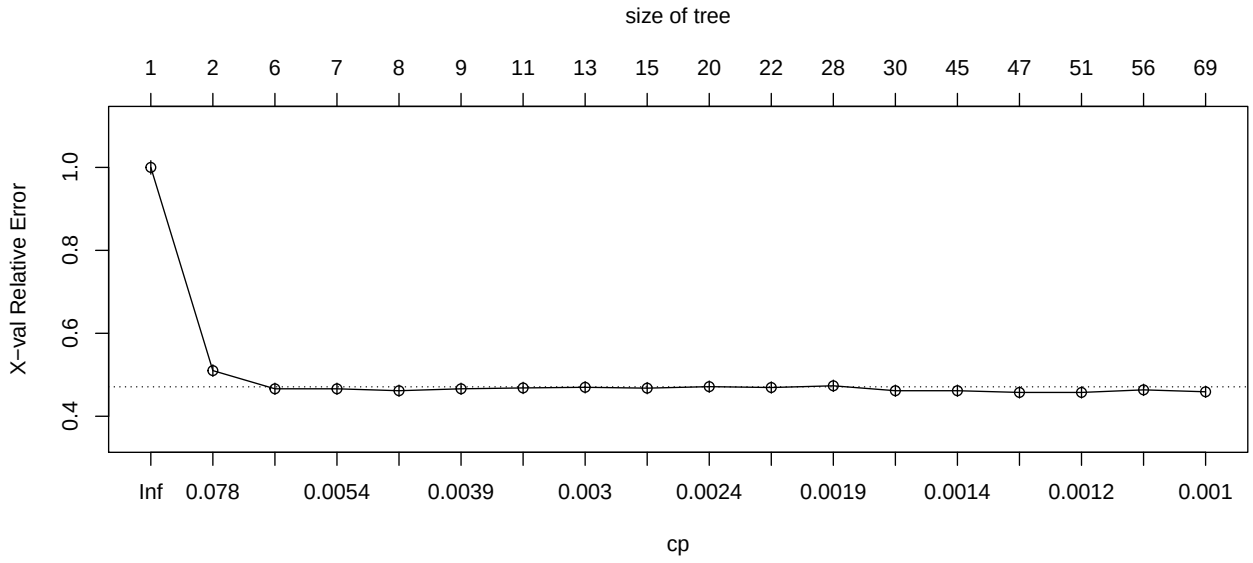
```
##  Accuracy
## 0.8268839
```

```
## Confusion Matrix and Statistics
##
##            Reference
```

```
## Prediction  stroke no_stroke
##   stroke        16       144
##   no_stroke     26       796
##
##                Accuracy : 0.8269
##                  95% CI : (0.8017, 0.85)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0973
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.38095
##             Specificity : 0.84681
##          Pos Pred Value : 0.10000
##          Neg Pred Value : 0.96837
##              Prevalence : 0.04277
##          Detection Rate : 0.01629
##    Detection Prevalence : 0.16293
##       Balanced Accuracy : 0.61388
##
##        'Positive' Class : stroke
##
```

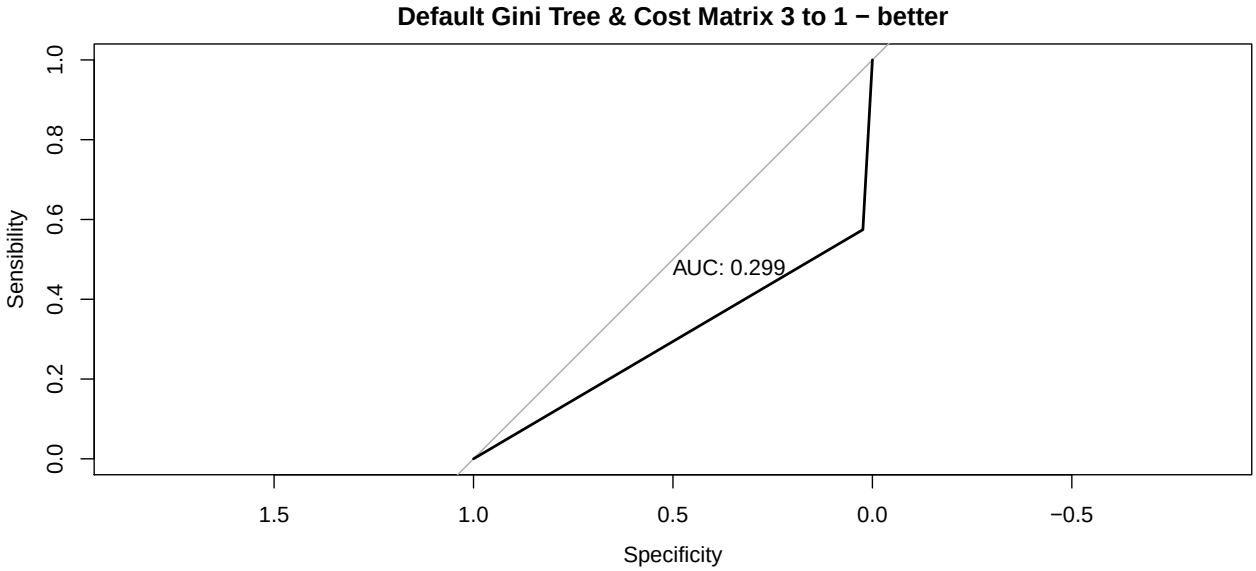


To obtain these results, the model has selected as best tunes: CP = 0.001, size of tree 69:

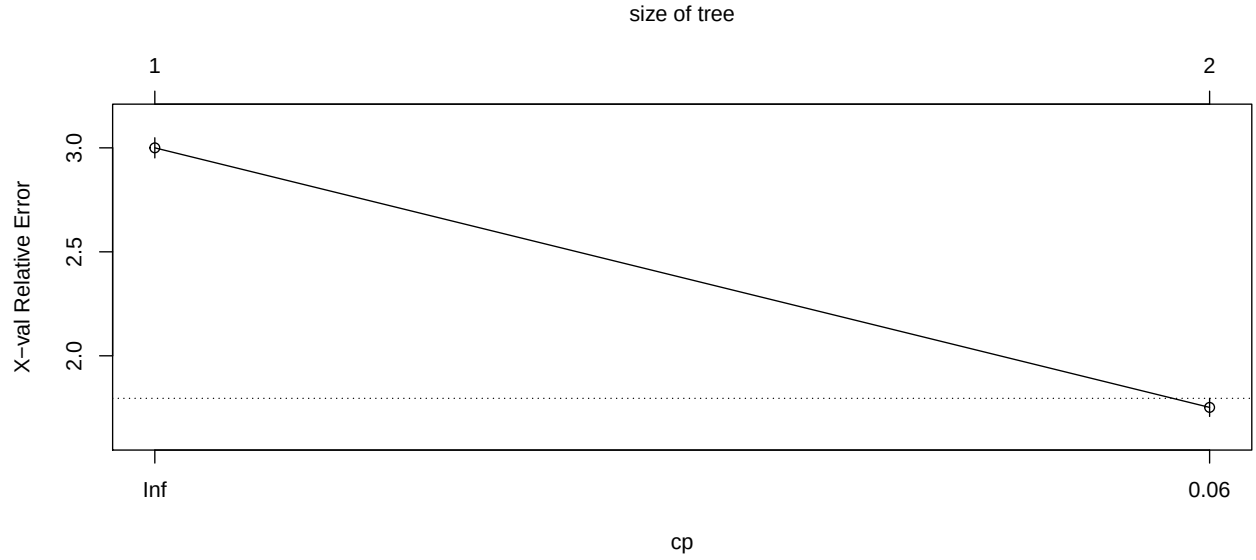


3.2.3.3 Default Gini Tree & Cost Matrix 3 to 1

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        36       380
##   no_stroke      6       560
##
##                Accuracy : 0.6069
##                  95% CI : (0.5756, 0.6376)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0862
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.85714
##             Specificity : 0.59574
##          Pos Pred Value : 0.08654
##          Neg Pred Value : 0.98940
##              Prevalence : 0.04277
##          Detection Rate : 0.03666
##    Detection Prevalence : 0.42363
##       Balanced Accuracy : 0.72644
##
##        'Positive' Class : stroke
##
```



To obtain these results, the model has selected as best tunes: CP = 0.06, size of tree = 2



The variables considered have been:

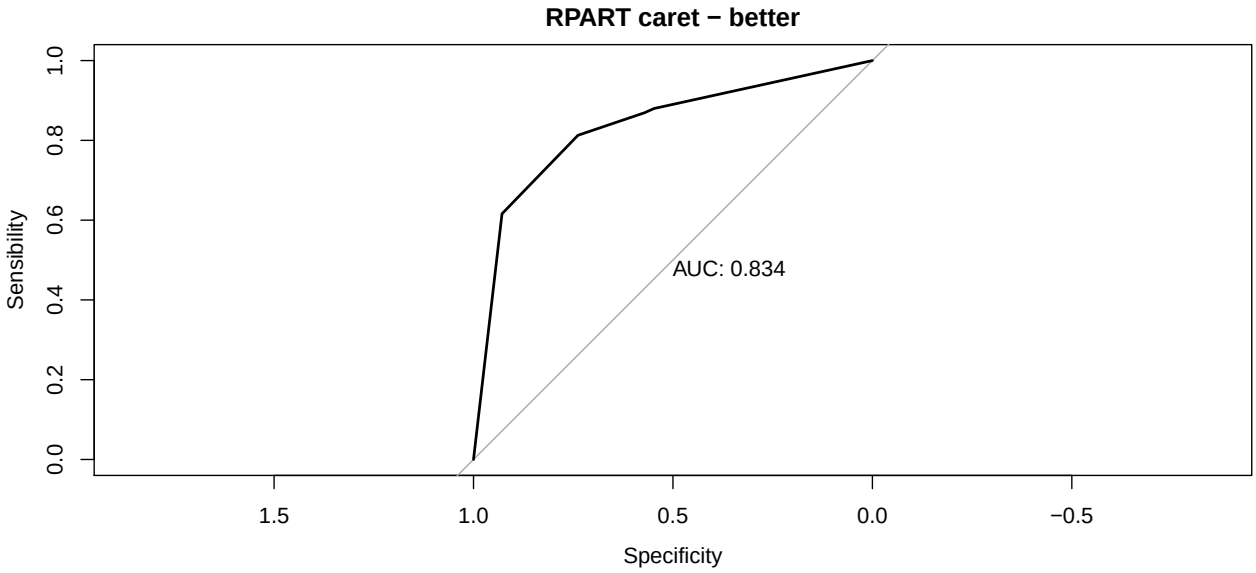
Variable importance

age	work_type	ever_married	bmi
57	19	19	5

3.2.3.4 RPART caret

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        31       176
##   no_stroke     11       764
```

```
##
##                Accuracy : 0.8096
##                  95% CI : (0.7836, 0.8337)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1915
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.73810
##             Specificity : 0.81277
##          Pos Pred Value : 0.14976
##          Neg Pred Value : 0.98581
##              Prevalence : 0.04277
##          Detection Rate : 0.03157
##    Detection Prevalence : 0.21079
##       Balanced Accuracy : 0.77543
##
##        'Positive' Class : stroke
##
```



The final value used for the model was cp = 0.005673027.

The variables considered have been:

Variable importance

age	ever_marriedYes	avg_glucose_level	bmi	smoking_statusUnknown
61	16	6	6	6

In addition to:

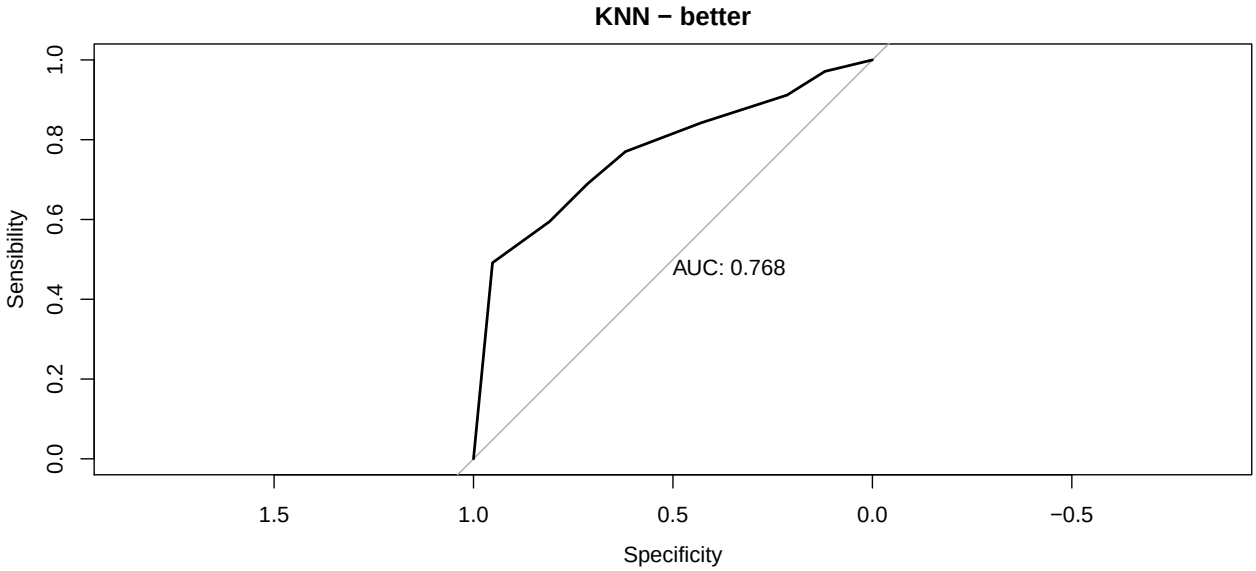
- heart_diseaseYes: 2
- smoking_statusnever smoked: 1

- smoking_statussmokes: 1
- work_typeSelf-employed: 1

3.2.3.5 K-Nearest-Neighbor

```
##            Reference
## Prediction  stroke no_stroke
##   stroke        26       216
##   no_stroke     16       724
```

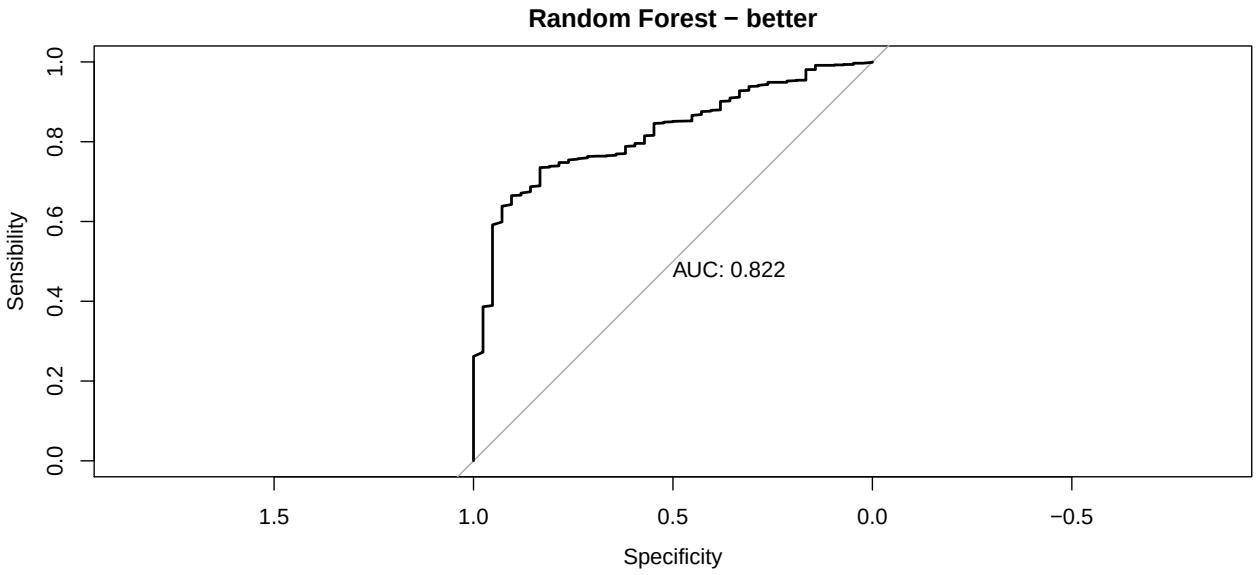
```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        26       216
##   no_stroke     16       724
##
##                Accuracy : 0.7637
##                  95% CI : (0.7359, 0.79)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1189
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.61905
##             Specificity : 0.77021
##          Pos Pred Value : 0.10744
##          Neg Pred Value : 0.97838
##              Prevalence : 0.04277
##          Detection Rate : 0.02648
##    Detection Prevalence : 0.24644
##       Balanced Accuracy : 0.69463
##
##        'Positive' Class : stroke
##
```



The final value used for the model was k = 7.

3.2.3.6 Ramdom Forest

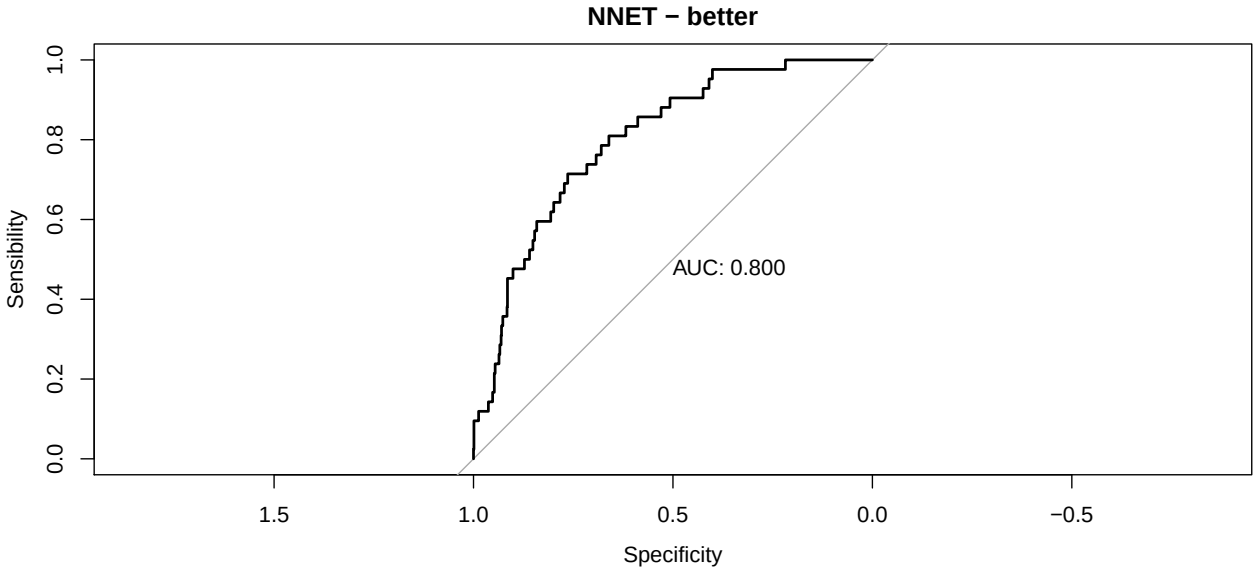
```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        25       192
##   no_stroke     17       748
##
##                Accuracy : 0.7872
##                  95% CI : (0.7602, 0.8124)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1308
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.59524
##             Specificity : 0.79574
##          Pos Pred Value : 0.11521
##          Neg Pred Value : 0.97778
##              Prevalence : 0.04277
##          Detection Rate : 0.02546
##    Detection Prevalence : 0.22098
##       Balanced Accuracy : 0.69549
##
##        'Positive' Class : stroke
##
```



The final value used for the model was mtry = 9.

3.2.3.7 Neural Network

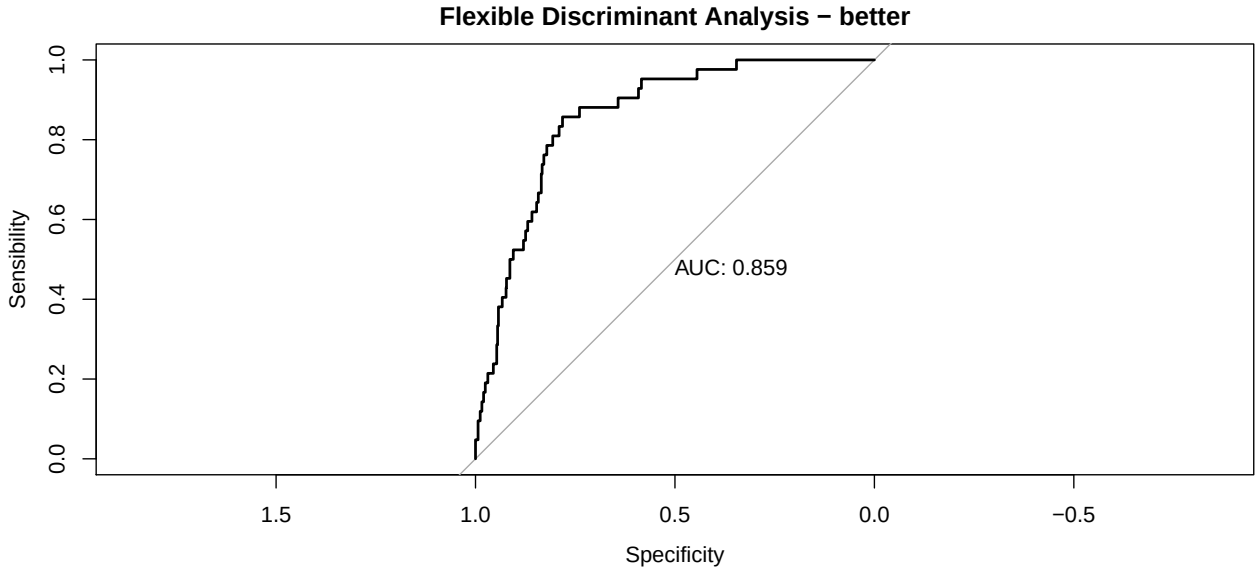
```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        30       225
##   no_stroke     12       715
##
##                Accuracy : 0.7587
##                  95% CI : (0.7306, 0.7851)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1388
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.71429
##             Specificity : 0.76064
##          Pos Pred Value : 0.11765
##          Neg Pred Value : 0.98349
##              Prevalence : 0.04277
##          Detection Rate : 0.03055
##    Detection Prevalence : 0.25967
##       Balanced Accuracy : 0.73746
##
##        'Positive' Class : stroke
##
```



The final values used for the model were size = 5 and decay = 0.1.

3.2.3.8 Flexible Discriminant Analysis

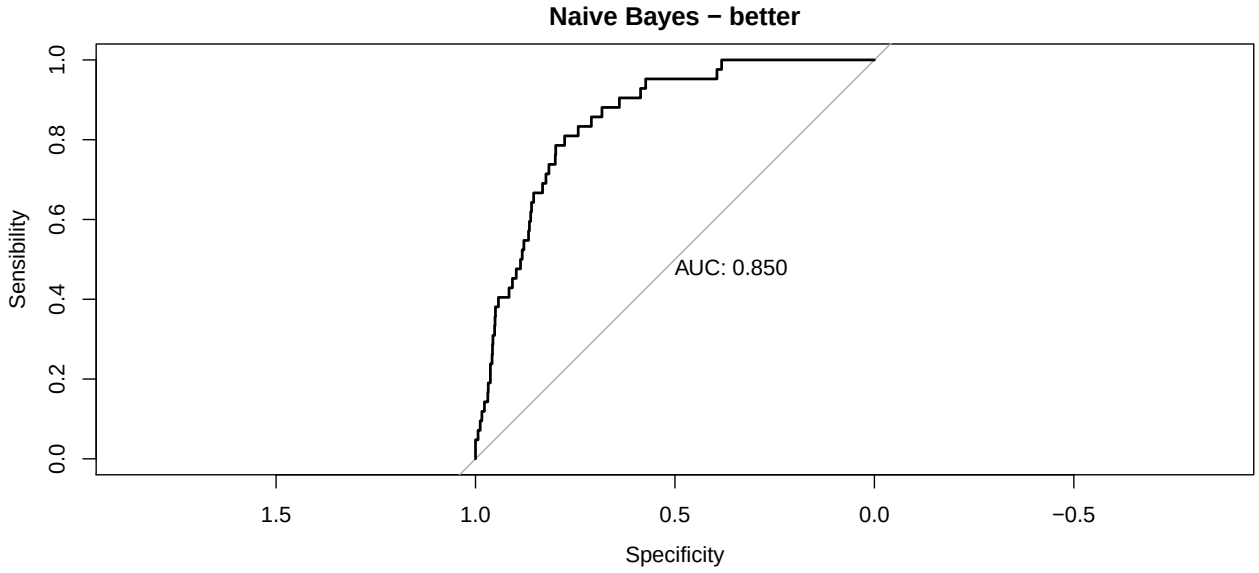
```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke        36       231
##   no_stroke      6       709
##
##                Accuracy : 0.7587
##                  95% CI : (0.7306, 0.7851)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1718
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.85714
##             Specificity : 0.75426
##          Pos Pred Value : 0.13483
##          Neg Pred Value : 0.99161
##              Prevalence : 0.04277
##          Detection Rate : 0.03666
##    Detection Prevalence : 0.27189
##       Balanced Accuracy : 0.80570
##
##        'Positive' Class : stroke
##
```



The final values used for the model were degree = 1 and nprune = 13.

3.2.3.9 Naive Bayes

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  stroke no_stroke
##   stroke         6        25
##   no_stroke     36       915
##
##                Accuracy : 0.9379
##                  95% CI : (0.9209, 0.9522)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 0.9982
##
##                   Kappa : 0.1329
##
##  Mcnemar's Test P-Value : 0.2004
##
##             Sensitivity : 0.14286
##             Specificity : 0.97340
##          Pos Pred Value : 0.19355
##          Neg Pred Value : 0.96215
##              Prevalence : 0.04277
##          Detection Rate : 0.00611
##    Detection Prevalence : 0.03157
##       Balanced Accuracy : 0.55813
##
##        'Positive' Class : stroke
##
```



3.2.4 Results Summary

Below we show the table with the results of all the models. The list is displayed in descending order based on the area under the ROC curve.

models	sensitivity	specificity	balanced_accuracy	AUC
fda_better	0.85714	0.75426	0.80570	0.8588
nb_both	0.26190	0.96596	0.61393	0.8505
nb_better	0.14286	0.97340	0.55813	0.8503
nb_over	0.26190	0.96489	0.61340	0.8489
fda_both	0.73810	0.74149	0.73979	0.8346
gini_tree_cp0.01_better	0.73810	0.81277	0.77543	0.8338
caret_tree_better	0.73810	0.81277	0.77543	0.8338
rf_better	0.59524	0.79468	0.69496	0.8196
fda_over	0.73810	0.74574	0.74192	0.8163
nnet_both	0.76190	0.78085	0.77138	0.8138
nnet_over	0.76190	0.77766	0.76978	0.8095
gini_tree_cp0.001_better	0.38095	0.84681	0.61388	0.8052
rf_both	0.11905	0.96383	0.54144	0.8050
caret_tree_over	0.73810	0.77553	0.75681	0.8042
nnet_better	0.71429	0.76064	0.73746	0.8000
rf_over	0.07143	0.98617	0.52880	0.7827
knn_better	0.61905	0.77021	0.69463	0.7685
cost_matrix_tree_over	0.90476	0.63085	0.76781	0.7525
gini_tree_cp0.01_over	0.73810	0.75745	0.74777	0.7372
cost_matrix_tree_both	0.85714	0.59574	0.72644	0.7368
gini_tree_cp0.01_both	0.71429	0.79894	0.75661	0.7352
caret_tree_both	0.71429	0.79894	0.75661	0.7352
knn_both	0.57143	0.77340	0.67242	0.6639
knn_over	0.42857	0.77553	0.60205	0.4015
gini_tree_cp0.001_both	0.38095	0.84681	0.61388	0.3724
gini_tree_cp0.001_over	0.47619	0.88723	0.68171	0.3212
cost_matrix_tree_better	0.85714	0.59574	0.72644	0.2991

So far the best model, in terms of Sensitivity and Balanced Accuracy, has turned out to be Flexible Discrimimant Analysis, along with the better estimates method for balancing the data.

As explained in the summary of the paper by Tevor Hastie, Robert Tibishirani and Andreas Buja, *Fisher's linear discriminant analysis is a valuable tool for multigroup classification. With a large number of predictors, one can find a reduced number of discriminant coordinate functions that are "optimal" for separating groups. With two such functions, one can produce a classification map that partitions the reduced space into regions that are identified with group membership, and the decision boundaries are linear* (Tevor Hastie, et al. 2021)[7]:

This model. As we have seen, it returns the following results when training with balanced data according to better estimates:

Accuracy	Sensitivity	Specificity	Balanced Accuracy	AUC
0.7587	0.85714	0.75426	0.80570	0.8588

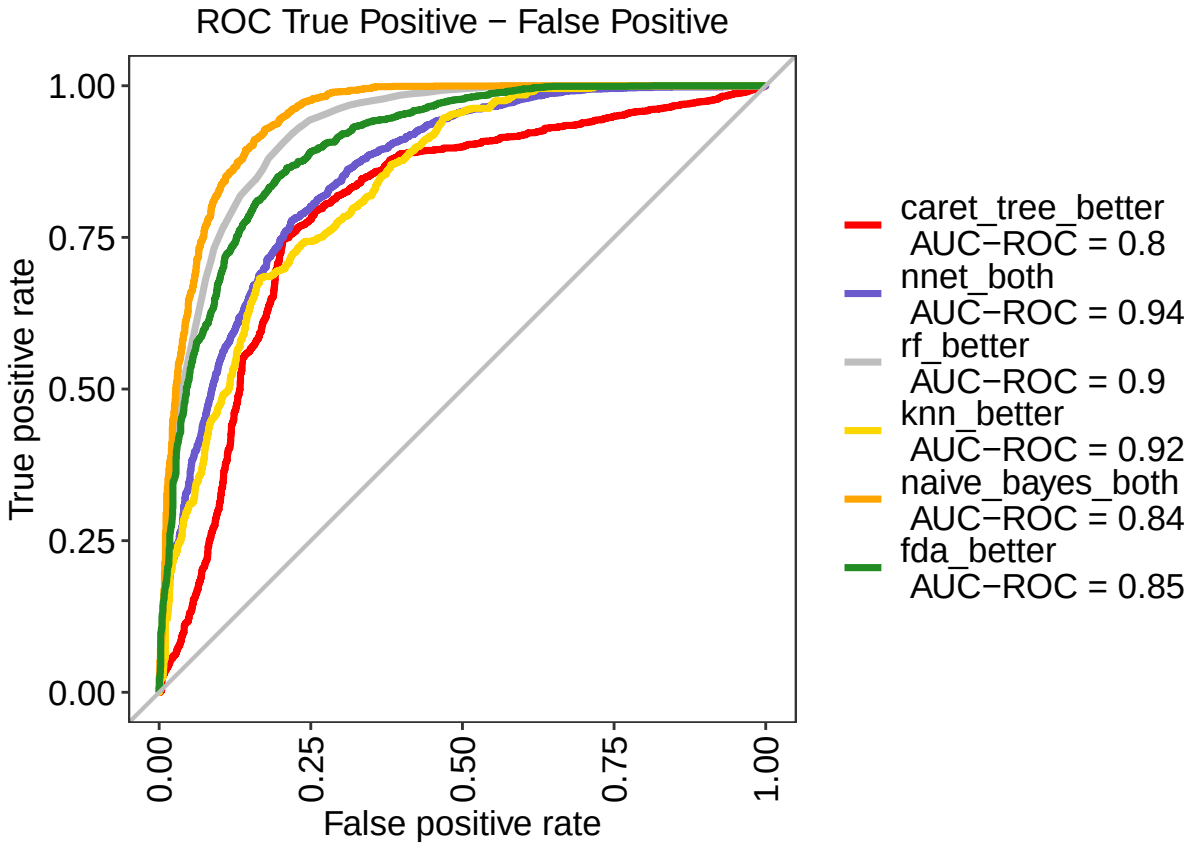
Comparing the results with the original test set, he is able to correctly classify 36 out of 42 observations as "stroke", keeping the Specificity at a reasonable 75%.

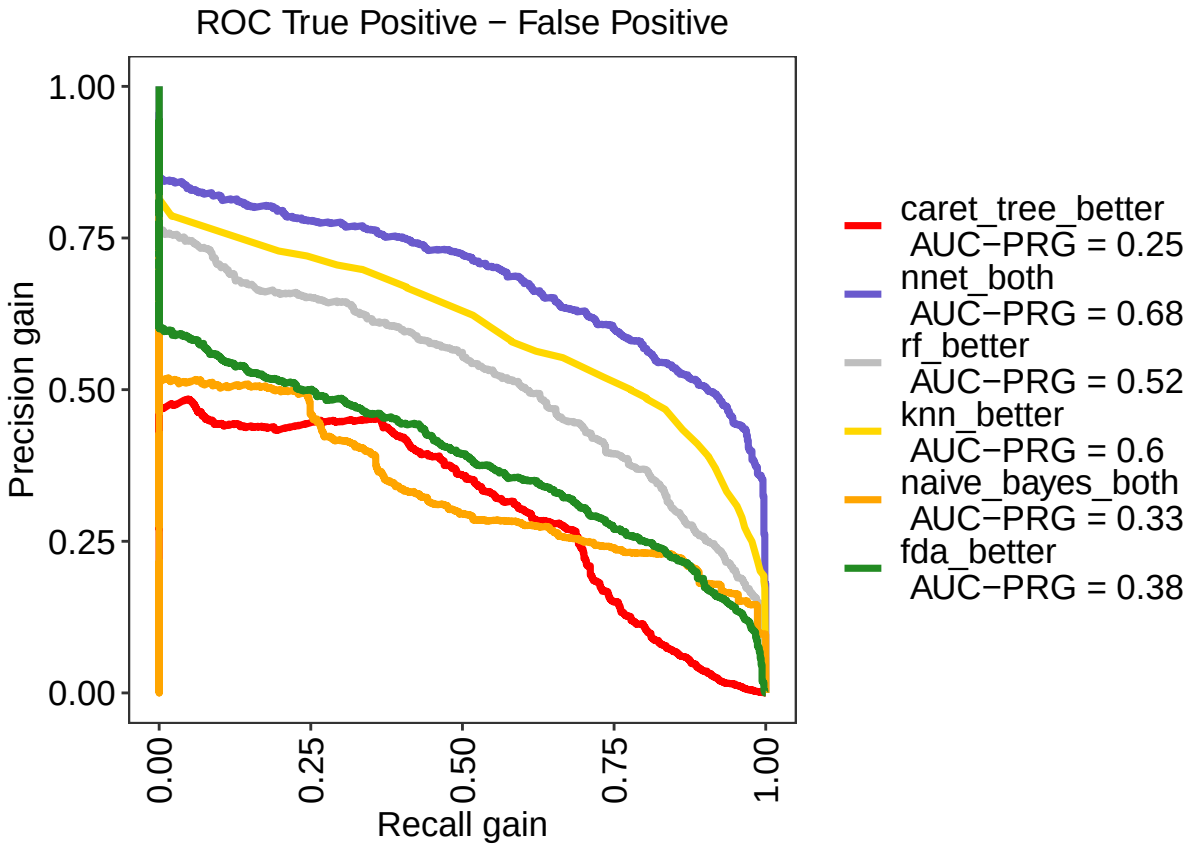
If we order the list according to balanced accuracy, the same model is the winner:

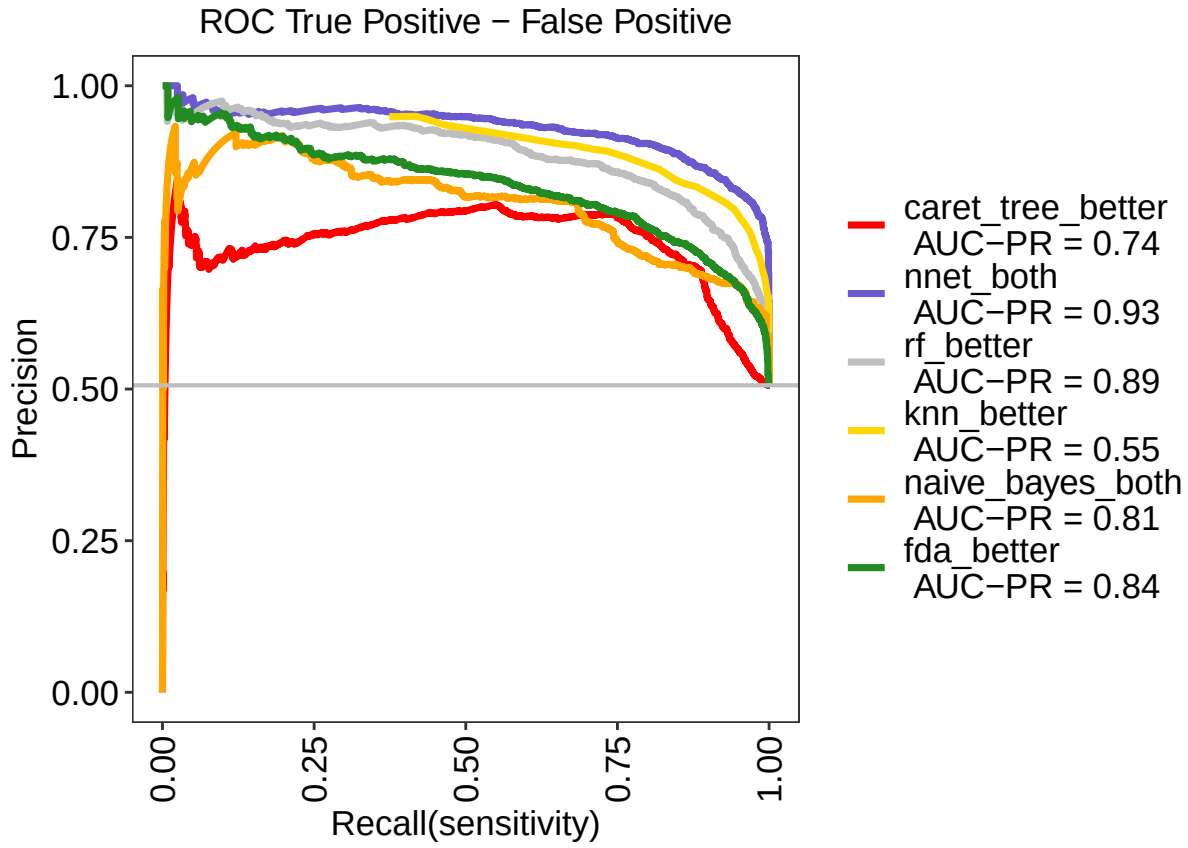
models	sensitivity	specificity	balanced_accuracy	AUC
fda_better	0.85714	0.75426	0.80570	0.8588
nb_both	0.26190	0.96596	0.61393	0.8505
nb_better	0.14286	0.97340	0.55813	0.8503
nb_over	0.26190	0.96489	0.61340	0.8489
fda_both	0.73810	0.74149	0.73979	0.8346
gini_tree_cp0.01_better	0.73810	0.81277	0.77543	0.8338
caret_tree_better	0.73810	0.81277	0.77543	0.8338
rf_better	0.59524	0.79468	0.69496	0.8196
fda_over	0.73810	0.74574	0.74192	0.8163
nnet_both	0.76190	0.78085	0.77138	0.8138
nnet_over	0.76190	0.77766	0.76978	0.8095
gini_tree_cp0.001_better	0.38095	0.84681	0.61388	0.8052
rf_both	0.11905	0.96383	0.54144	0.8050
caret_tree_over	0.73810	0.77553	0.75681	0.8042
nnet_better	0.71429	0.76064	0.73746	0.8000
rf_over	0.07143	0.98617	0.52880	0.7827
knn_better	0.61905	0.77021	0.69463	0.7685
cost_matrix_tree_over	0.90476	0.63085	0.76781	0.7525
gini_tree_cp0.01_over	0.73810	0.75745	0.74777	0.7372
cost_matrix_tree_both	0.85714	0.59574	0.72644	0.7368
gini_tree_cp0.01_both	0.71429	0.79894	0.75661	0.7352
caret_tree_both	0.71429	0.79894	0.75661	0.7352
knn_both	0.57143	0.77340	0.67242	0.6639
knn_over	0.42857	0.77553	0.60205	0.4015
gini_tree_cp0.001_both	0.38095	0.84681	0.61388	0.3724
gini_tree_cp0.001_over	0.47619	0.88723	0.68171	0.3212
cost_matrix_tree_better	0.85714	0.59574	0.72644	0.2991

However, under other criteria there are models that perform better. Below we compare the selected model with the best versions of the caret RPART (caret_tree), NNET, RF and NB models.

[7] *Flexible Discriminant Analysis by Optimal Scoring.* Tevor Hastie, Robert Tibishirani & Andreas Buja. http://ww.web.stanford.edu/~hastie/Papers/fda.pdf


ROC True Positive − False Positive
True positive rate
False positive rate
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
caret_tree_better
AUC−ROC = 0.8
nnet_both
AUC−ROC = 0.94
rf_better
AUC−ROC = 0.9
knn_better
AUC−ROC = 0.92
naive_bayes_both
AUC−ROC = 0.84
fda_better
AUC−ROC = 0.85


ROC True Positive − False Positive
Precision gain
Recall gain
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
caret_tree_better
AUC−PRG = 0.25
nnet_both
AUC−PRG = 0.68
rf_better
AUC−PRG = 0.52
knn_better
AUC−PRG = 0.6
naive_bayes_both
AUC−PRG = 0.33
fda_better
AUC−PRG = 0.38



Therefore, the selection of the FDA model is due to the fact that it is the one that best meets the established criteria, although other models present better results in aspects such as areas under the curves. We will try to solve this problem in the next section. Before that, let us take a closer look at the results table.

It is clear that the results of the caret RPART model are practically the same as those of the default rpart function. So we will remove the gini_tree_cp_0.01 models from the list.

The results of NNET Over, and NNET Both are practically the same, so we will eliminate the second of them. The same is true for FDA Over and FDA Both. We will only keep the first one.

Cost Matrix Tree Both and Better only differ on the AUC (something we need to investigate further in the future). Let's just stick with Cost Matrix Tree Both.

The same goes for gini_tree_cp0.001 Both and Better. In this case we will stick with the Better version, which has a higher AUC.

After eliminating models with the same results, the final list is as follows (ordered according to Balanced Accuracy):

models	sensitivity	specificity	balanced_accuracy	AUC
fda_better	0.85714	0.75426	0.80570	0.8588
caret_tree_better	0.73810	0.81277	0.77543	0.8338
nnet_both	0.76190	0.78085	0.77138	0.8138
cost_matrix_tree_over	0.90476	0.63085	0.76781	0.7525
caret_tree_over	0.73810	0.77553	0.75681	0.8042
caret_tree_both	0.71429	0.79894	0.75661	0.7352
fda_over	0.73810	0.74574	0.74192	0.8163
nnet_better	0.71429	0.76064	0.73746	0.8000
cost_matrix_tree_both	0.85714	0.59574	0.72644	0.7368

models	sensitivity	specificity	balanced_accuracy	AUC
rf_better	0.59524	0.79468	0.69496	0.8196
knn_better	0.61905	0.77021	0.69463	0.7685
gini_tree_cp0.001_over	0.47619	0.88723	0.68171	0.3212
knn_both	0.57143	0.77340	0.67242	0.6639
nb_both	0.26190	0.96596	0.61393	0.8505
gini_tree_cp0.001_better	0.38095	0.84681	0.61388	0.8052
nb_over	0.26190	0.96489	0.61340	0.8489
knn_over	0.42857	0.77553	0.60205	0.4015
nb_better	0.14286	0.97340	0.55813	0.8503
rf_both	0.11905	0.96383	0.54144	0.8050
rf_over	0.07143	0.98617	0.52880	0.7827

The best models, after FDA, are caret RPART (caret_tree), and NNET.

As can be seen, in this case a classification tree responds in a similar way to our objectives than a complex neural network (and in a significantly shorter calculation time). It is true that the network has a little more Sensitivity, but in terms of Balanced Accuracy and AUC, the results are very similar.

In any case, it is fair to acknowledge that our data set has relatively little data, and a neural network usually needs a lot of observations.

The first finding we get from this is that sometimes a simpler model can offer similar results to a more complex one, especially if we don't have a lot of data. Between one model and another, we can choose to keep the simplest one, not only for calculation times but also for explicability. A decision tree is really easy to explain, whereas the criteria used by the neural network to classify observations are a black box.

The second we see is that, as we suspected after the exploratory analysis, the KNN model is not suitable for this data set, where the points belonging to both classes share common neighbors.

Third, we can say that Random Forest and Naive Bayes definitely do not perform well with this data set, and perhaps in similar cases.

Finally, if we apply the "better estimates" method, it seems that we are somewhat more likely to optimize the results. 6 of the 9 groups of trained models present better results with this method. In 2 groups the "both" method is better, and in 1 group the "over" method is better (the criterion in this case is the AUC):

models	sensitivity	specificity	balanced_accuracy	AUC
fda_better	0.85714	0.75426	0.80570	0.8588
fda_both	0.73810	0.74149	0.73979	0.8346
fda_over	0.73810	0.74574	0.74192	0.8163

models	sensitivity	specificity	balanced_accuracy	AUC
caret_tree_better	0.73810	0.81277	0.77543	0.8338
caret_tree_over	0.73810	0.77553	0.75681	0.8042
caret_tree_both	0.71429	0.79894	0.75661	0.7352

models	sensitivity	specificity	balanced_accuracy	AUC
gini_tree_cp0.01_better	0.73810	0.81277	0.77543	0.8338
gini_tree_cp0.01_both	0.71429	0.79894	0.75661	0.7352
gini_tree_cp0.01_over	0.73810	0.75745	0.74777	0.7372

models	sensitivity	specificity	balanced_accuracy	AUC
nnet_both	0.76190	0.78085	0.77138	0.8138
nnet_over	0.76190	0.77766	0.76978	0.8095
nnet_better	0.71429	0.76064	0.73746	0.8000

models	sensitivity	specificity	balanced_accuracy	AUC
cost_matrix_tree_over	0.90476	0.63085	0.76781	0.7525
cost_matrix_tree_both	0.85714	0.59574	0.72644	0.7368
cost_matrix_tree_better	0.85714	0.59574	0.72644	0.2991

models	sensitivity	specificity	balanced_accuracy	AUC
gini_tree_cp0.001_better	0.38095	0.84681	0.61388	0.8052
gini_tree_cp0.001_both	0.38095	0.84681	0.61388	0.3724
gini_tree_cp0.001_over	0.47619	0.88723	0.68171	0.3212

models	sensitivity	specificity	balanced_accuracy	AUC
rf_better	0.59524	0.79468	0.69496	0.8196
rf_both	0.11905	0.96383	0.54144	0.8050
rf_over	0.07143	0.98617	0.52880	0.7827

models	sensitivity	specificity	balanced_accuracy	AUC
knn_better	0.61905	0.77021	0.69463	0.7685
knn_both	0.57143	0.77340	0.67242	0.6639
knn_over	0.42857	0.77553	0.60205	0.4015

models	sensitivity	specificity	balanced_accuracy	AUC
nb_both	0.26190	0.96596	0.61393	0.8505
nb_better	0.14286	0.97340	0.55813	0.8503
nb_over	0.26190	0.96489	0.61340	0.8489

In any case, the decision to use one method or another also depends on the amount of data available. If applying a method we are left with very little data, it is better to choose "over".

In fact, there is a fourth method to balance the data: "under". With this system, observations of the majority class are randomly eliminated to match their frequency with the minority class. In this particular case, the resulting data set was reduced too much to train any model with a certain level of confidence.

3.2.5 Ensembles

The winning model meets part of our objectives, but still has a low Sensitivity considering the nature of the project (predicting the tendency to suffer a stroke).

In addition, the Accuracy barely touches 76%, and the comparison with other models shows that some, with lower Sensitivity and Balanced Accuracy, present better areas under the curves (ROC, and Precision-Recall).

We can try to improve the results by building ensembles of models that meet a certain condition. We have already said that we must give priority to Sensitivity, but we cannot sacrifice Specificity in such a way that the model is not able to predict negative cases.

To build our ensembles, we have selected from the definitive list all the models whose AUC is higher than 0.8 (Ensemble A), and all those with a Balanced Accuracy higher than 0.7 (Ensemble B):

Ensemble A

models	sensitivity	specificity	balanced_accuracy	AUC
fda_better	0.85714	0.75426	0.80570	0.8588
caret_tree_better	0.73810	0.81277	0.77543	0.8338
nnet_both	0.76190	0.78085	0.77138	0.8138
caret_tree_over	0.73810	0.77553	0.75681	0.8042
fda_over	0.73810	0.74574	0.74192	0.8163
nnet_better	0.71429	0.76064	0.73746	0.8000
rf_better	0.59524	0.79468	0.69496	0.8196
nb_both	0.26190	0.96596	0.61393	0.8505
gini_tree_cp0.001_better	0.38095	0.84681	0.61388	0.8052
nb_over	0.26190	0.96489	0.61340	0.8489
nb_better	0.14286	0.97340	0.55813	0.8503
rf_both	0.11905	0.96383	0.54144	0.8050

Ensemble B

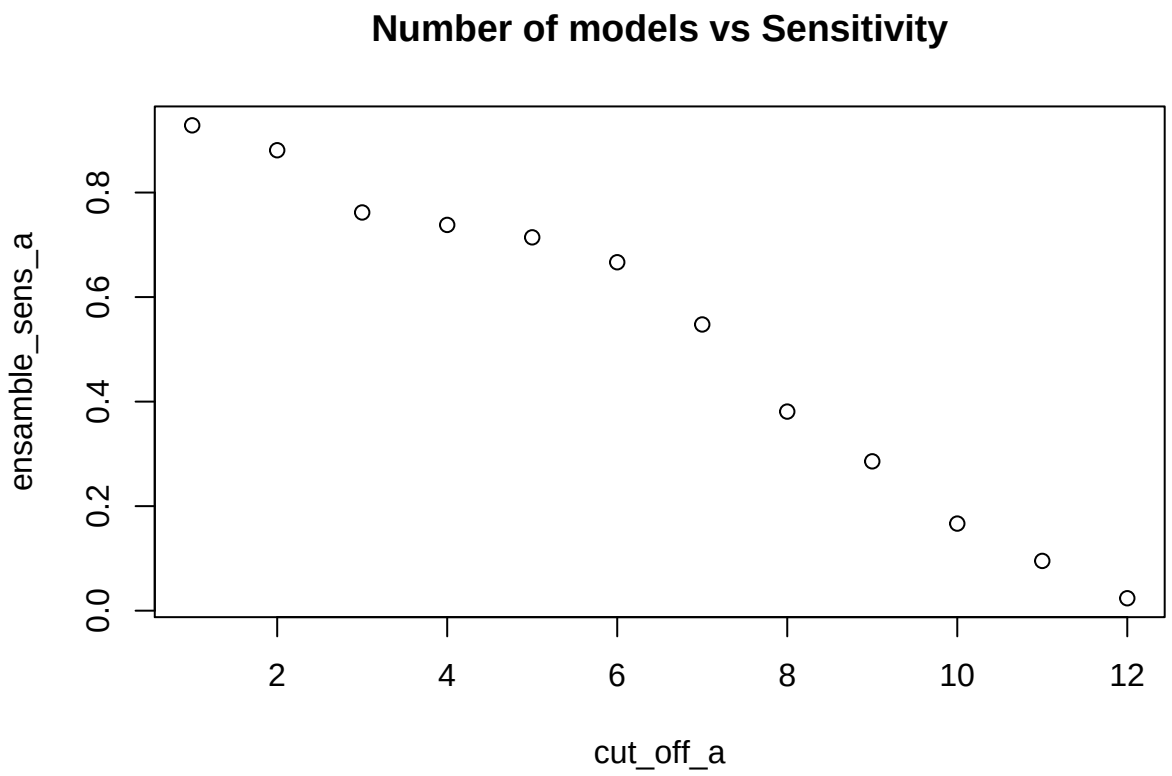
models	sensitivity	specificity	balanced_accuracy	AUC
fda_better	0.85714	0.75426	0.80570	0.8588
caret_tree_better	0.73810	0.81277	0.77543	0.8338
nnet_both	0.76190	0.78085	0.77138	0.8138
cost_matrix_tree_over	0.90476	0.63085	0.76781	0.7525
caret_tree_over	0.73810	0.77553	0.75681	0.8042
caret_tree_both	0.71429	0.79894	0.75661	0.7352
fda_over	0.73810	0.74574	0.74192	0.8163
nnet_better	0.71429	0.76064	0.73746	0.8000
cost_matrix_tree_both	0.85714	0.59574	0.72644	0.7368

In both cases, the results depend on the selection of the number of models that have predicted “stroke”, to predict “stroke”.

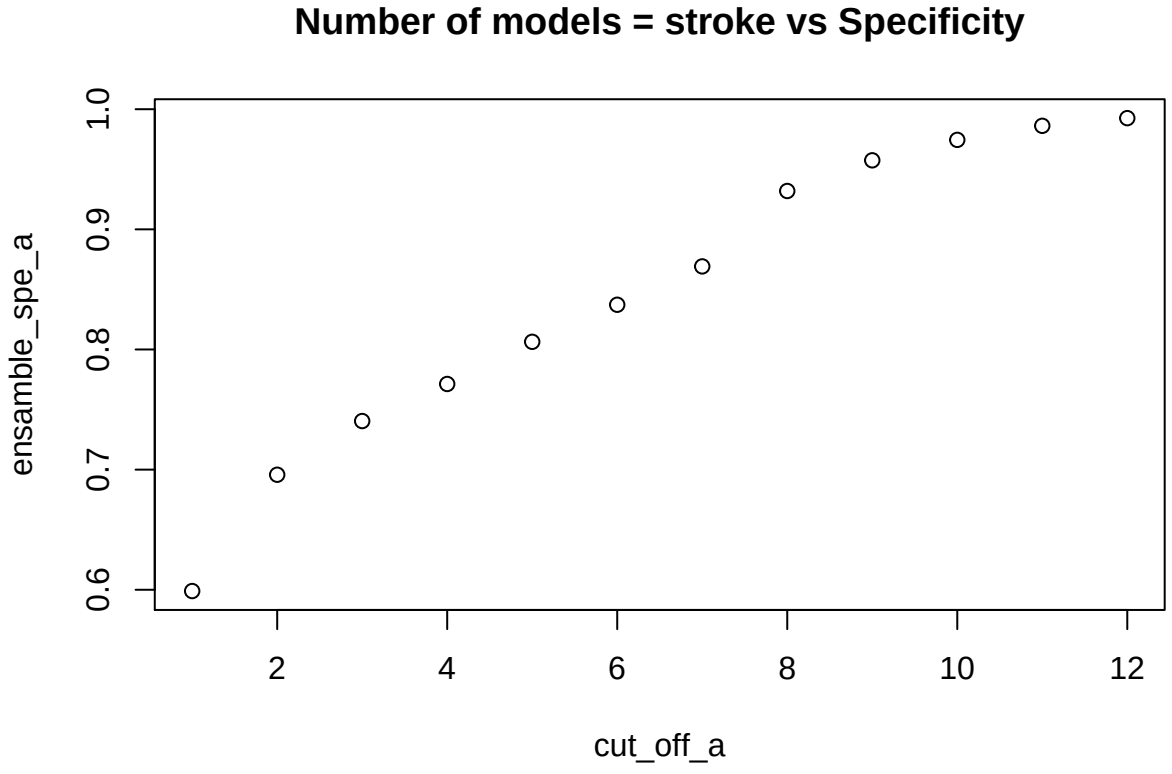
The smaller the number, the higher the Sensitivity at the expense of Specificity. As that number grows, you gain in Specificity at the cost of Sensitivity.

Below we show the resulting graphs for Ensembe A, (made up of 12 models) according to the number of models that must predict “stroke”, to classify an observation as “stroke”.

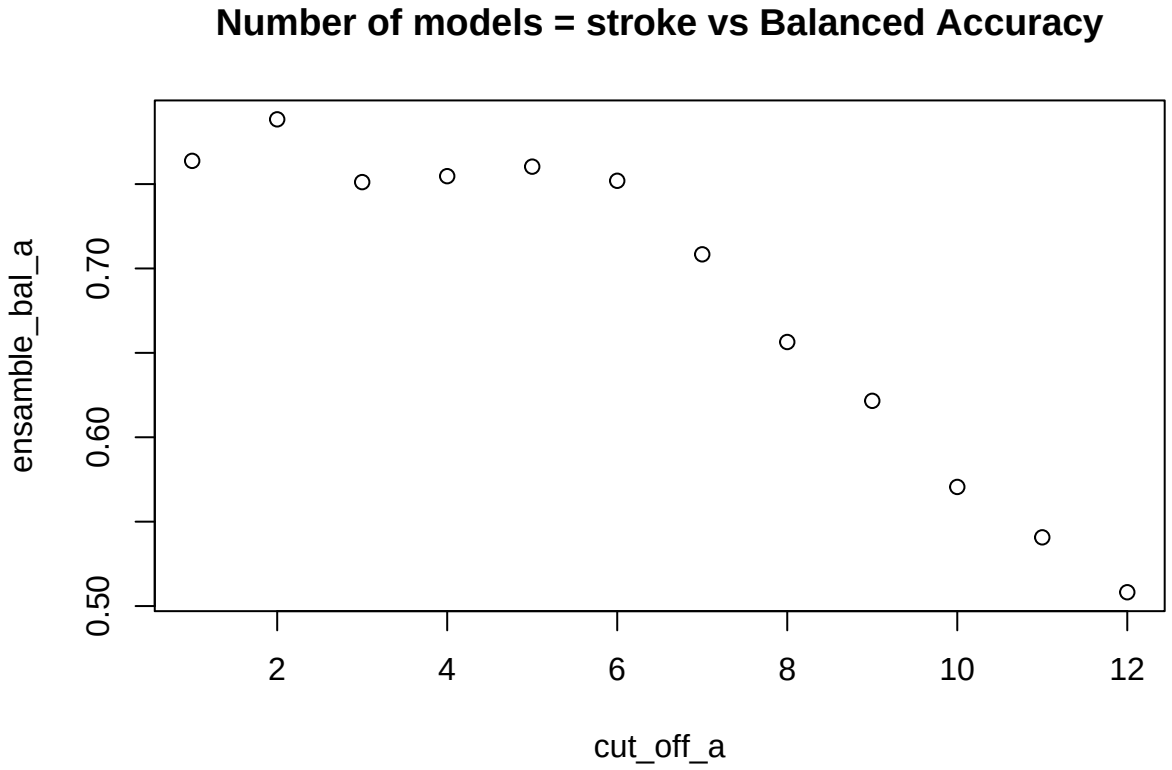
If we predict “stroke” when only one of the models has classified the observation as such, the Sensitivity exceeds 90%. If at the other extreme, we require that all 12 models have predicted “stroke” to classify an observation as “stroke”, the Sensitivity drops to 2% (only 1 of 42 cases is classified correctly).



The graph is inverted for the case of Specificity, although in this case the lower Specificity is close to 60%, given the prevalence of the "no_stroke" class:



The highest Balanced Accuracy is obtained by specifying that only 2 out of 12 models have classified an observation as "stroke" to predict "stroke":



The same analysis has been carried out for Ensemble B, which we do not reproduce here.

If we choose the Balanced Accuracy criteria to select the best cutoff the results, compared to the FDA better model, are as follows:

3.2.5.1 Ensemble A, cutoff = 2

Models	Accuracy	Sensitivity	Specificity	Balanced Accuracy
Ensemble A	0.7037	0.88095	0.69574	0.78835
fda_better	0.7587	0.85714	0.75426	0.80570

We gain some Sensitivity, at the cost of a lot of Accuracy and Specificity. Balanced Accuracy also suffers.

3.2.5.2 Ensemble A, cutoff = 1

If we choose to further increase the Sensitivity, the results are as follows:

Models	Accuracy	Sensitivity	Specificity	Balanced Accuracy
Ensemble A	0.6130	0.92857	0.59894	0.76375
fda_better	0.7587	0.85714	0.75426	0.80570

In terms of Sensitivity, the Ensemble A is obviously better, but at the cost of obtaining a Specificity of less than 60%. With a cutoff of 1, the Ensemble results are similar to those obtained with the Cost Matrix Over model:

Models	Accuracy	Sensitivity	Specificity	Balanced Accuracy
Ensemble A	0.6130	0.92857	0.59894	0.76375
cost_matrix_tree_over	0.6426	0.90476	0.63085	0.76781

Obviously, if we had to bring one of the two models into production, we would opt for the Cost Matrix. It is much easier to train a single model. The Ensemble is only justified if it improves the results substantially, and in this case it does not appear to do so.

3.2.5.3 Ensemble B, cutoff = 2

Below we show the results of Assembly B,compared to FDA Better model, according to the cutoff that maximizes Balanced Accuracy and Sensitivity at the same time (cutoff = 2):

Models	Accuracy	Sensitivity	Specificity	Balanced Accuracy
Ensemble B	0.6375	0.95238	0.62340	0.78789
fda_better	0.7587	0.85714	0.75426	0.80570

In this case, we obtain a Sensitivity greater than 95%. The FDA model is still the most balanced, but if we want to correctly classify the largest possible number of positive cases, this Ensemble could be the answer.

Ensemble B presents similar results in terms of Accuracy and Balacend Accuracy, compared to Cost Matrix, But the Sensitivity is higher:

Models	Accuracy	Sensitivity	Specificity	Balanced Accuracy
Ensemble B	0.6375	0.95238	0.62340	0.78789
cost_matrix_tree_over	0.6426	0.90476	0.63085	0.76781

Forced to decide on a very Sensitive model, Ensemble B could be the right choice.

4. Conclusions

4.1 Summary

Throughout this project we have experimented with various classification models, training them with unbalanced data (just to check the problems that this generates in terms of performance), and with balanced data using various methods.

In the case of balanced data, we have applied balancing on the training data set (not on the test data set). This has raised doubts, but we have followed the example of the sources consulted.

The results on the unbalanced data are, unsurprisingly, very poor. With the balanced training set we obtain much better results, although in no case does the Balanced Accuracy exceed 80% (our main indicator, along with AUC).

In terms of balanced results (high Accuracy, high Sensitivity, high Specificity, high Balanced Accuracy, and high AUC) the best model has turned out to be FDA, trained with data obtained with the better estimates method.

4.2 Limitations

Given the prevalence of the negative class ("no_stroke"), and because we have decided to keep the test data set unbalanced, the correct classification of 1 case up or down has a great impact on Sensitivity.

For this same reason, the minimum Specificity can never go below 60% (the same as Accuracy), and models with Specificity and Accuracy around that percentage are not performing really well in that regard.

Furthermore, since the confusion matrices are so unbalanced, the F-Meas metric is always low. Even so in general the "good" models have a comparatively higher F-Meas than the others. In any case, we decided not to consider this metric when evaluating the models.

In terms of Sensitivity, the Cost Matrix models (those that penalize false negatives) are the best, but at the cost of presenting a Specificity that is around the lower limit of 60% which, as we have already said, cannot be lower given the prevalence of the negative class. In this sense, our Ensembles perform similarly to Cost Matrix models.

4.3 Potential Impact

The applicability of either model for this project is debatable. The best of them, from our point of view, has a Sensitivity of 85% (calculated on 42 positive cases, which is what the test data set contains).

Taking into account the incidence of strokes in the population (suppose it is 4%, the same prevalence as in the original data set), the real possibility of suffering a stroke if the FDA better model predict "stroke" is 13% if we apply Bayes. This means that with a positive test, we have 3.25 more options to suffer a stroke than the population base.

According to data from the world health organization, in 2019 about 6 million people died of stroke. That represents approximately 0.08% of the world's population for that year. Therefore, if our model predicts "stroke", the possibility of dying (not just suffering) from that disease is, according to Bayes, close to 0.27%. Again, we are talking about 3.5 more options than base population.

If the decision to keep the training data set unbalanced was the right one, a positive result is enough to warn a person to watch their glucose level and body mass index. If the person smokes, it would be time to quit. Age and marital status "ever married = yes" are factors that we cannot alter.

4.4 Future Work

To improve the results, there are least are three lines of action that would be worth testing:

1. Create dummy variables from the categorical variables, and transform them to 0 and 1, instead of factors.
2. Create age groups, "babies", "children", "young people" and "old people" to train the models using these groups, or at least group the ages to the nearest ten.
3. Train the models eliminating one by one variables with apparently little weight (Hypertension, Heart Disease, Residence Type and Gender, for example).

However, to continue with this work with certain guarantees of improvement, we believe that it would be necessary to add other variables that may not be contemplated in the original data set.

References

The top 10 causes of death. World Health Organization, 2020. https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death

Tevor Hastie, et al. (2021). *Flexible Discriminant Analysis by Optimal Scoring.* http://ww.web.stanford.edu/~hastie/Papers/fda.pdf

Amat, Rodrigo (2020): *Machine Learning con R y caret.* https://www.cienciadedatos.net/documentos/41_machine_learning_con_r_y_caret#Introducci%C3%B3n

Bibliography and resources

Irizarry, Rafael. *Introduction to Data Science. Data Analysis and Prediction Algorithms with R*, 2021. https://rafalab.github.io/dsbook/

Kuhn, Max. *The caret Package*, 2019. http://topepo.github.io/caret/index.html

Parra, Francisco. *Estadística y Machine Learning con R*, 2019. https://bookdown.org/content/2274/portada.html

Delgado, Ronald. *Introducción a los Modelos de Clasificación en R*, 2018. https://rpubs.com/rdelgado/397838

Dinov, Ivo. *Data Science and Predictive Analytics (UMich HS650). Improving Model Performance*, 2020. https://www.socr.umich.edu/people/dinov/courses/DSPA_notes/14_ImprovingModelPerformance.html

Santillan, Carlos. *Parameter tuning caret*, 2017. https://csantill.github.io/RTuningModelParameters/

Practical Guide to deal with Imbalanced Classification Problems in R, 2016. Analytics Vidhya: https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/

Brownlee, Jason. *Cost-Sensitive Learning for Imbalanced Classification*, Machine Learning Mastery, 2020. https://machinelearningmastery.com/cost-sensitive-learning-for-imbalanced-classification/

Brownlee, Jason. *Non-Linear Classification in R*, Machine Learning Mastery, 2020. https://machinelearningmastery.com/non-linear-classification-in-r/

How to easily make a ROC curve in R, 2019. INTOBIOINFORMATICS. https://intobioinformatics.wordpress.com/2019/11/26/how-to-easily-make-a-roc-curve-in-r/

Rpart decision trees in r. Learn by Marketing. https://www.learnbymarketing.com/methods/classification-and-regression-decision-trees-explained/

Custom lollipop chart. The R Graph Gallery. https://www.r-graph-gallery.com/301-custom-lollipop-chart.html