My_own_Project

Yassin Zeraoulia

20 5 2021

#Introduction

According to the World Health Organization (WHO) stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. This dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relavant information about the patient.

VAriable inside the data set: 1) id: unique identifier 2) gender: "Male", "Female" or "Other" 3) age: age of the patient 4) hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension 5) heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease 6) ever_married: "No" or "Yes" 7) work_type: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed" 8) Residence_type: "Rural" or "Urban" 9) avg_glucose_level: average glucose level in blood 10) bmi: body mass index 11) smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"* 12) stroke: 1 if the patient had a stroke or 0 if not

In order to predict effectively if a person is more likely to have a stroke we'll have to take into consideration relevant variables of the data set, some of them are not relevant because they don't have impact on the physiology of the human being. Next we'll have to gain some insight from summary statistics of the data set and from the plots that will help visualize some correlation and trends of the variable of the data set. Finally we'll implement some predictive model and test them to asses which model is able to predict whit satisfactory accuracy strokes in the population. The first and most simple model that will be implemented is logistic regression while the second one will be based on K-nearest neighbors algorithm.

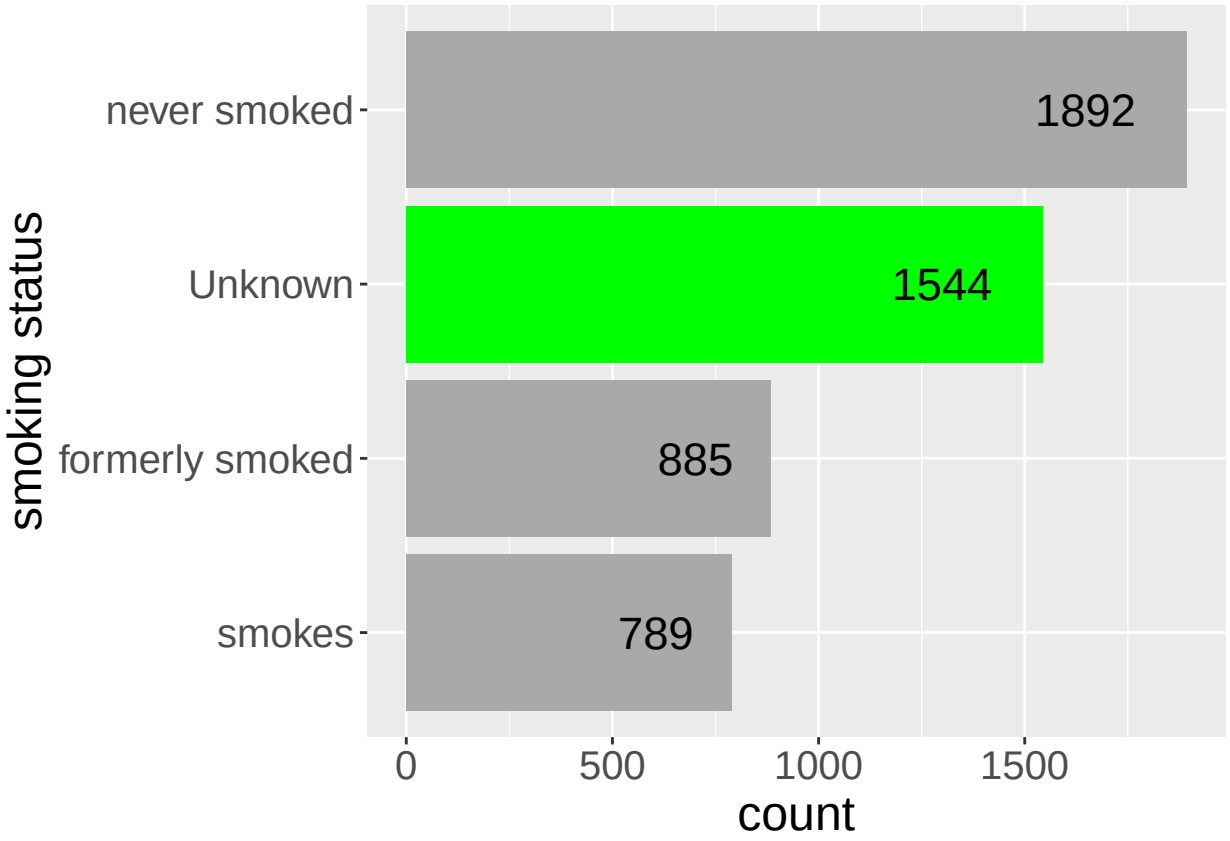
We start by removing unnecessary variable inside the dataset and by checking wheter NA are present in the data set. We see that we have 1544 unknown values for smoking status and therefore are missing a lot of information in a potentially informative predictor.

```
# how many "N/A" values are in my dataset per column?
miss_scan_count(data = stroke_data, search = list("N/A", "Unknown"))
```

```
## # A tibble: 12 x 2
##    Variable             n
##    <chr>            <int>
##  1 id                   0
##  2 gender               0
##  3 age                  0
##  4 hypertension         0
##  5 heart_disease        0
##  6 ever_married         0
##  7 work_type            0
##  8 Residence_type       0
##  9 avg_glucose_level    0
## 10 bmi                201
```

```
## 11 smoking_status     1544
## 12 stroke                0
```

```
#There are 201 "N/A" values in the bmi column that likely caused this column to be parsed as character,


###there are a lot of "Unknown" values in smoking_status

fig(15, 8)

stroke_data %>%
  group_by(smoking_status) %>%
  summarise(count = length(smoking_status)) %>%
  mutate(smoking_status = factor(smoking_status)) %>%
  ggplot(aes(x = fct_reorder(smoking_status, count), y = count, fill = factor(ifelse(smoking_status=="U
  geom_col() +
  geom_text(aes(label = count, x = smoking_status, y = count), size = 6, hjust = 1.5) +
  coord_flip() +
  scale_fill_manual(values = c("Unknown" = "green", "Known" = "darkgrey")) +
  labs(x = "smoking status") +
  theme(legend.position = "none") +
  theme_bigfont
```




```
# replace the "N/A" in bmi
stroke_data_clean <- replace_with_na(data = stroke_data, replace = list(bmi = c("N/A"), smoking_status =
```

```
  # change bmi to numeric
  mutate(bmi = as.numeric(bmi))

# check
summary(stroke_data_clean)
```

```
##        id            gender               age         hypertension
##  Min.   :   67   Length:5110        Min.   : 0.08   Min.   :0.00000
##  1st Qu.:17741   Class :character   1st Qu.:25.00   1st Qu.:0.00000
##  Median :36932   Mode  :character   Median :45.00   Median :0.00000
##  Mean   :36518                      Mean   :43.23   Mean   :0.09746
##  3rd Qu.:54682                      3rd Qu.:61.00   3rd Qu.:0.00000
##  Max.   :72940                      Max.   :82.00   Max.   :1.00000
##
##  heart_disease     ever_married        work_type         Residence_type
##  Min.   :0.00000   Length:5110        Length:5110        Length:5110
##  1st Qu.:0.00000   Class :character   Class :character   Class :character
##  Median :0.00000   Mode  :character   Mode  :character   Mode  :character
##  Mean   :0.05401
##  3rd Qu.:0.00000
##  Max.   :1.00000
##
##  avg_glucose_level      bmi        smoking_status         stroke
##  Min.   : 55.12    Min.   :10.30   Length:5110        Min.   :0.00000
##  1st Qu.: 77.25    1st Qu.:23.50   Class :character   1st Qu.:0.00000
##  Median : 91.89    Median :28.10   Mode  :character   Median :0.00000
##  Mean   :106.15    Mean   :28.89                      Mean   :0.04873
##  3rd Qu.:114.09    3rd Qu.:33.10                      3rd Qu.:0.00000
##  Max.   :271.74    Max.   :97.60                      Max.   :1.00000
##                    NA's   :201
```

```
unique(stroke_data_clean$smoking_status)
```

```
## [1] "formerly smoked" "never smoked"    "smokes"          NA
```

```
knitr::opts_chunk$set(echo = TRUE)
```

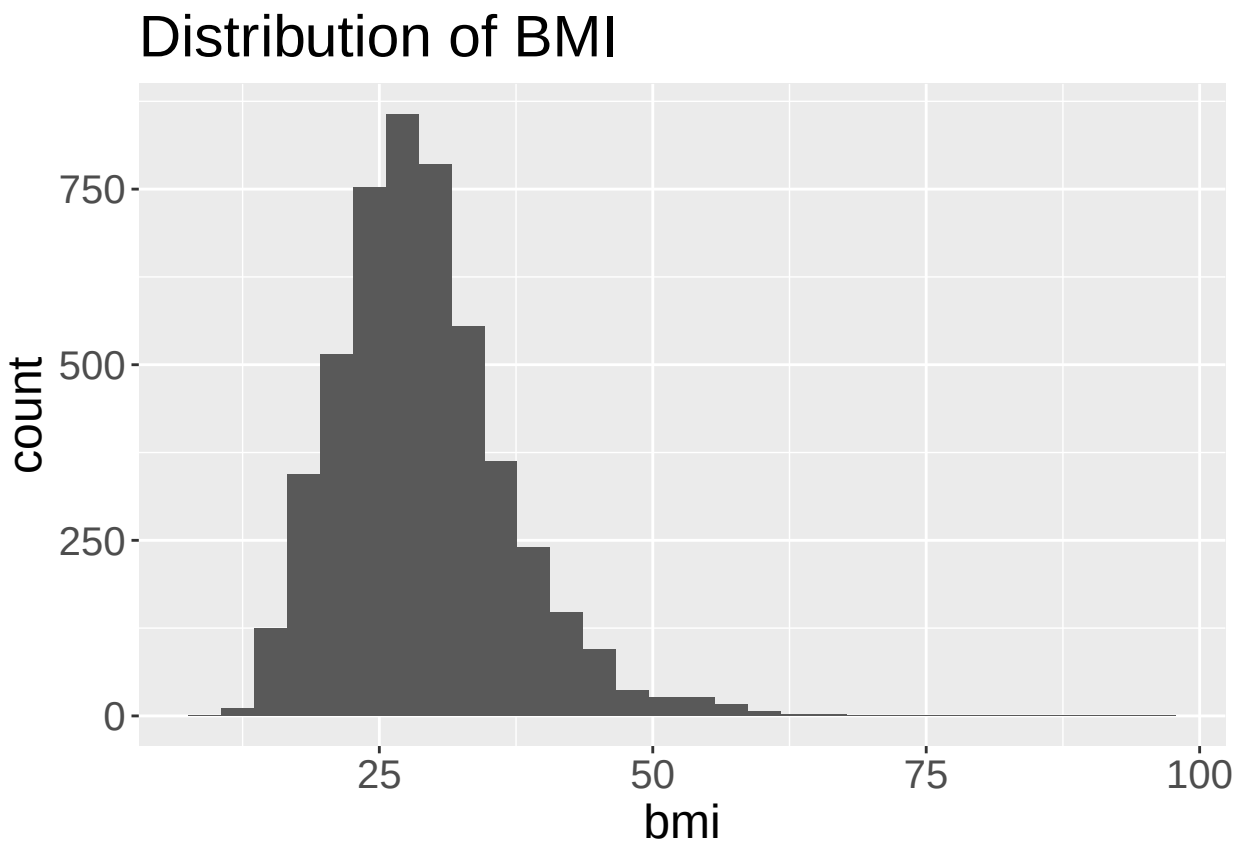
Now we can gain some initial insight on the variables inside the data set by generating some plots and analizyng distributions of the data set:

The distribution of bmi is right skewed (long tail to the right). Because this is the only variable with missing data (at least of the numerical variables) we can impute the median on the missing data without losing too much information.

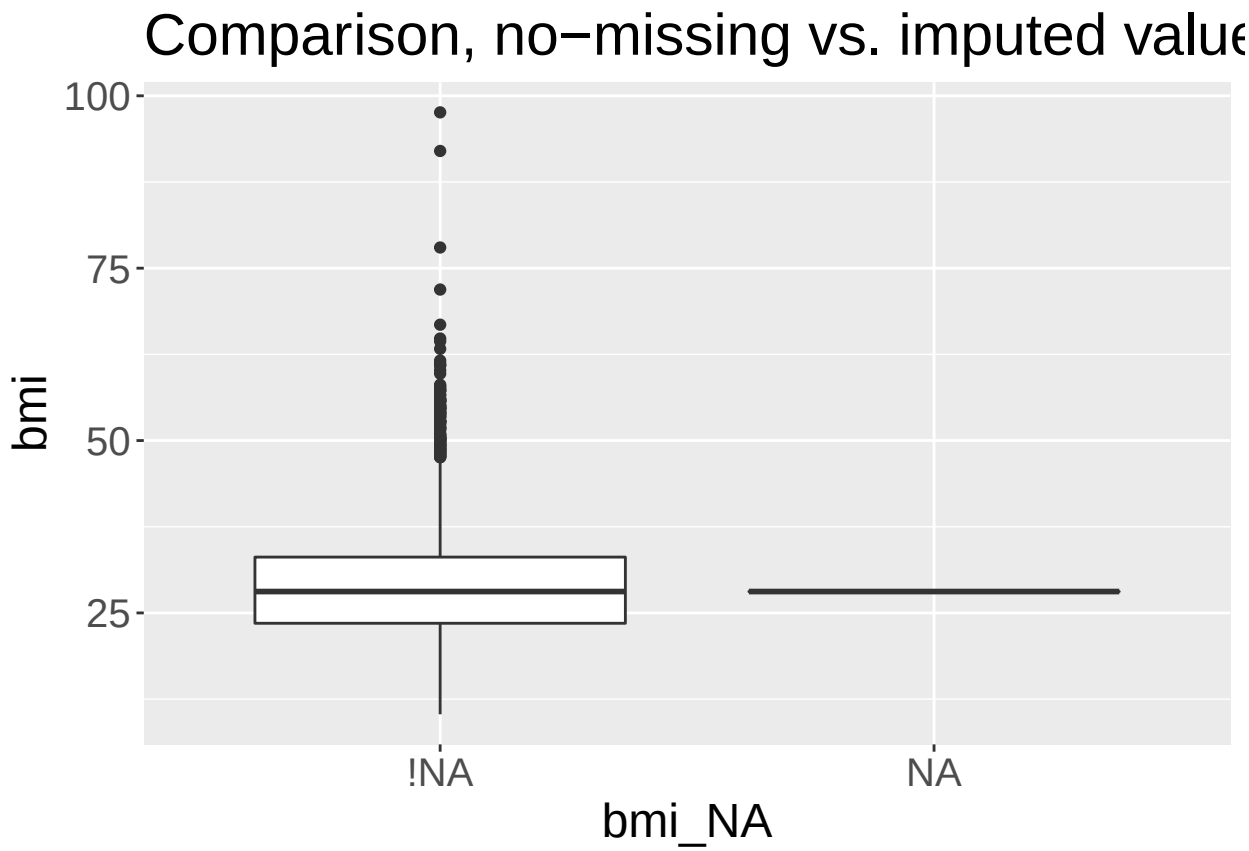
```
# check distribution of bmi
ggplot(stroke_data_clean, aes(x = bmi)) +
  geom_histogram() +
  labs(title = "Distribution of BMI") +
  theme_bigfont
```

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

```
## Warning: Removed 201 rows containing non-finite values (stat_bin).
```

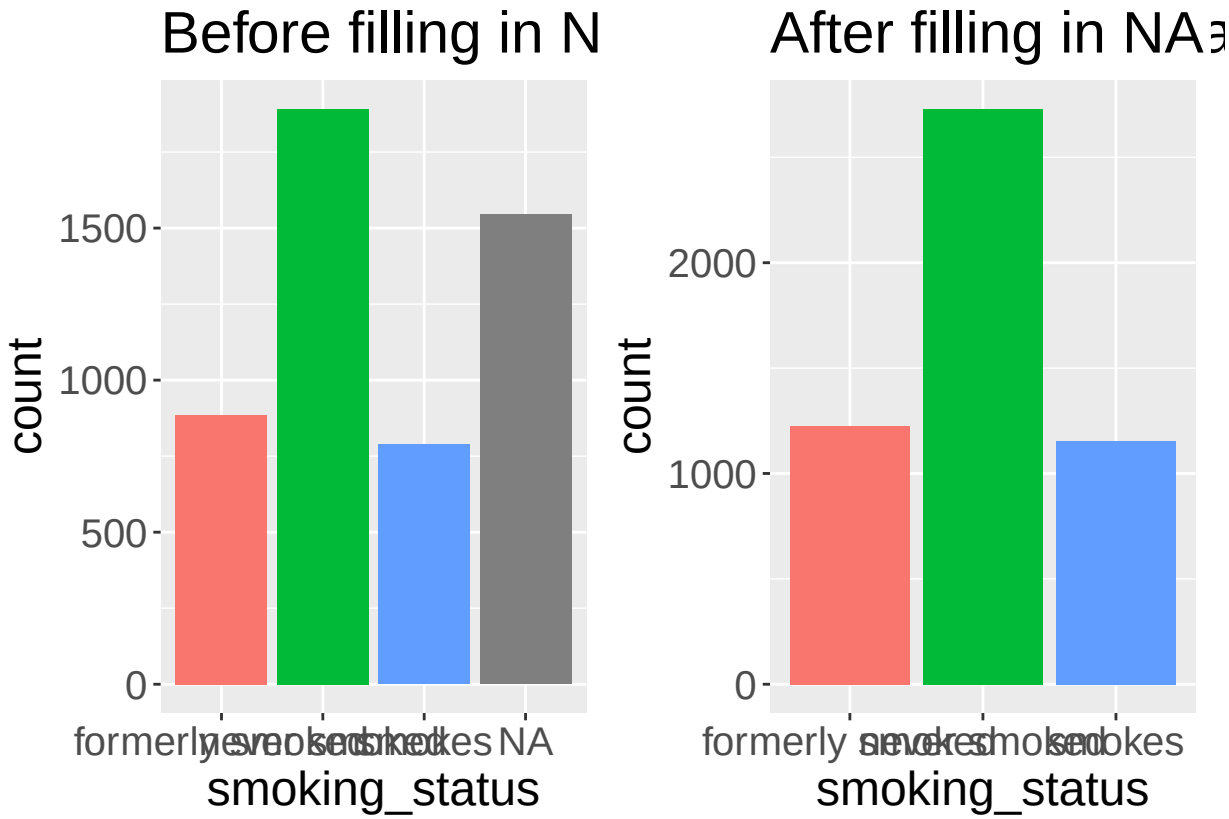



```
# impute median and bind shadow to evaluate imputation
stroke_data_imp <- bind_shadow(stroke_data_clean) %>%
  impute_median_at(.vars = c("bmi")) %>%
  add_label_shadow()

# Explore the median values in bmi in the imputed dataset
ggplot(stroke_data_imp,
       aes(x = bmi_NA, y = bmi)) +
  geom_boxplot() +
  labs(title = "Comparison, no-missing vs. imputed values for BMI") +
  theme_bigfont
```




```
stroke_data_imp <- impute_median_at(stroke_data_clean, .vars = c("bmi"))


fig(16,8)

p1 <- ggplot(stroke_data_imp,
             aes(x = smoking_status, fill = smoking_status)) +
  geom_bar() +
  labs(title = "Before filling in NA values in smoking_status") +
  theme(legend.position = "none") +
  theme_bigfont

# fill imputation based on previous unique value in "smoking_status" column
after <- stroke_data_imp %>%
  fill(smoking_status)
# mode imputation which leads to worse performance of models:
#mutate(across(c(smoking_status)), replace(., is.na(.), "never smoked"))

# Explore the median values in bmi in the imputed dataset
p2 <- ggplot(after,
             aes(x = smoking_status, fill = smoking_status)) +
  geom_bar() +
  labs(title = "After filling in NA values in smoking_status") +
  theme(legend.position = "none") +
  theme_bigfont
```

```
p1 + p2
```



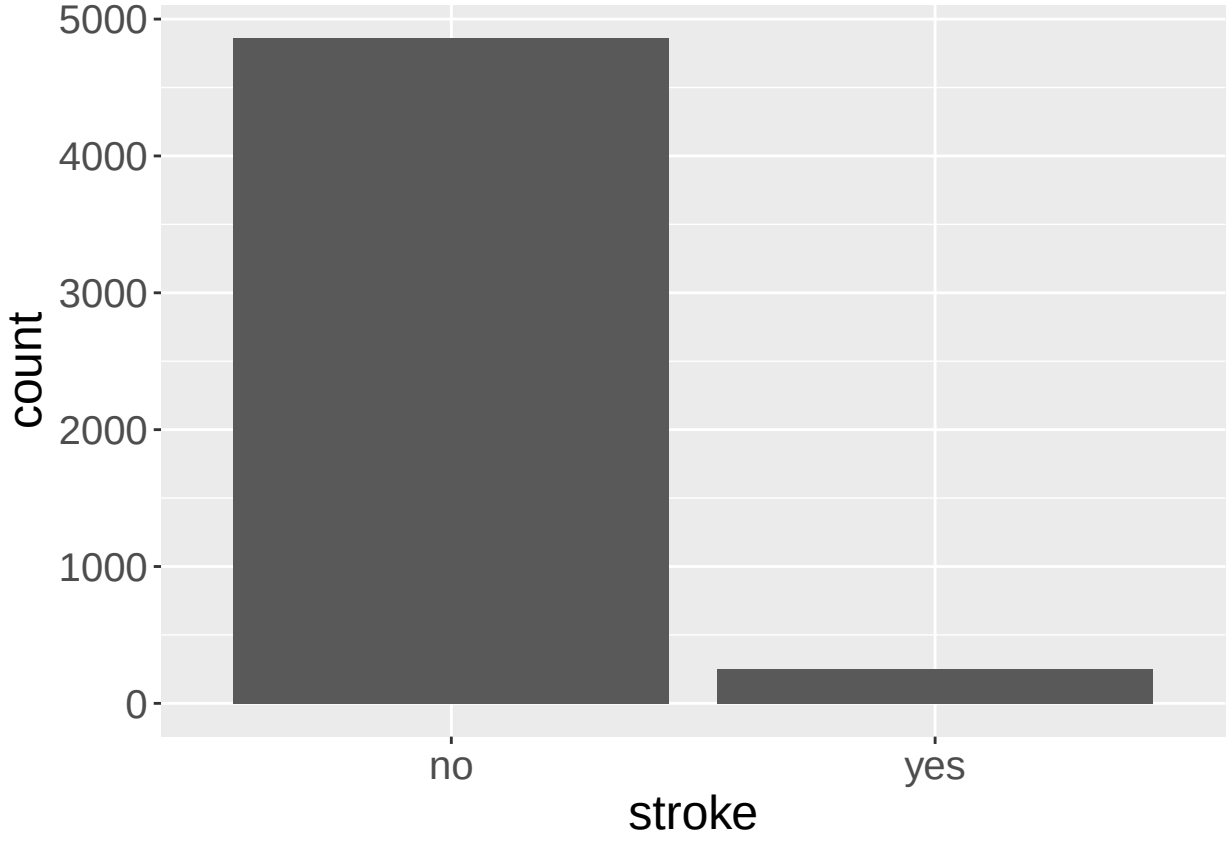

```
knitr::opts_chunk$set(echo = TRUE)
```

Here we link each values of some variable to the respective categorical values.In particular we convert bmi from a continuous variable to a factor according to the bmi categories of the CDC.This will be useful running the random forest mode, it will increase slightly the performance.

```
stroke_data_imp2 <- stroke_data_imp %>%
  fill(smoking_status) %>%
  mutate(across(c(smoking_status)), replace(., is.na(.), "never smoked")) %>%
  mutate(across(c(hypertension, heart_disease), factor),
         across(where(is.character), as.factor),
         across(where(is.factor), as.numeric),
         stroke = factor(ifelse(stroke == 0, "no", "yes")))

stroke_data_imp2 <- stroke_data_imp2 %>%
  mutate(bmi = case_when(bmi < 18.5 ~ "underweight",
                         bmi >= 18.5 & bmi < 25 ~ "normal weight",
                         bmi >= 25 & bmi < 30 ~ "overweight",
                         bmi >= 30 ~ "obese"),
         bmi = factor(bmi, levels = c("underweight", "normal weight", "overweight", "obese"), order = TR

knitr::opts_chunk$set(echo = TRUE)
```

Only 5% of the people inside the data set had a stroke, This means that our baseline dummy model has an accuracy of 95%.That is if we would predict a person to not have a stroke all the time.

```
#Only 5% of the people inside the data set had a stroke

fig(10, 8)

# plot prop of people who had a stroke
stroke_data_imp2 %>%
  select(stroke) %>%
  ggplot(aes(x = stroke)) +
  geom_bar() +
  theme_bigfont
```




```
# count how many people had a stroke and the prop
stroke_data_imp2 %>%
  group_by(stroke) %>%
  summarize(n = n()) %>%
  mutate(prop = round(n / sum(n), 2))
```

```
## # A tibble: 2 x 3
##   stroke     n  prop
##   <fct>  <int> <dbl>
## 1 no      4861  0.95
## 2 yes      249  0.05
```

```
knitr::opts_chunk$set(echo = TRUE)
```

We go further balancing the imbalance

```
# check imbalance ratio
imbalanceRatio(as.data.frame(stroke_data_imp2), classAttr = "stroke")
```

```
## [1] 0.05122403
```

```
stroke_test <- stroke_data_imp2 %>%
  mutate(
    stroke = as.character(stroke),
    across(where(is.factor), as.numeric),
    stroke = factor(stroke)
  )

stroke_oversampled <- oversample(as.data.frame(stroke_test), classAttr = "stroke", ratio = 1, method =

head(stroke_oversampled)
```

```
##      id gender age hypertension heart_disease ever_married work_type
## 1  9046      2  67            1             2            2         4
## 2 51676      1  61            1             1            2         5
## 3 31112      2  80            1             2            2         4
## 4 60182      1  49            1             1            2         4
## 5  1665      1  79            2             1            2         5
## 6 56669      2  81            1             1            2         4
##   Residence_type avg_glucose_level bmi smoking_status stroke
## 1              2            228.69   4              1    yes
## 2              1            202.21   3              2    yes
## 3              1            105.92   4              2    yes
## 4              2            171.23   4              3    yes
## 5              1            174.12   2              2    yes
## 6              2            186.21   3              1    yes
```

```
stroke_oversampled %>%
  group_by(stroke) %>%
  summarize(n = n()) %>%
  mutate(prop = round(n / sum(n), 2))
```

```
## # A tibble: 2 x 3
##   stroke     n  prop
##   <fct>  <int> <dbl>
## 1 no      4861   0.5
## 2 yes     4861   0.5
```

```
stroke_data_final <- stroke_oversampled %>% select(-id)
knitr::opts_chunk$set(echo = TRUE)
```

In order to build our model we'll split the data set in two chunk, train (70%) and test (30%)

```r
# total number of observations
n_obs <- nrow(stroke_data_final)

# shuffle the dataset randomly
permuted_rows <- sample(n_obs)

# Randomly order data
stroke_shuffled <- stroke_data_final[permuted_rows,]

# Identify row to split on
split <- round(n_obs * 0.7)

# Create train
train <- stroke_shuffled[1:split,]

# Create test
test <- stroke_shuffled[(split + 1):nrow(stroke_shuffled),]

# check if train is really 70% of the original
nrow(train) / nrow(stroke_data_final)
```

```
## [1] 0.6999589
```

```r
knitr::opts_chunk$set(echo = TRUE)
```

#Random forset model

```r
######random forest
mm_test <- test %>% select(-stroke)

rfGrid <- data.frame(
  .mtry = c(2,3,5,6),
  .splitrule = "gini",
  .min.node.size = 5
)

rfControl <- trainControl(
  method = "oob",
  number = 5,
  verboseIter = TRUE
)

rf_model <- train(
  stroke ~ .,
  train,
  method = "ranger",
  tuneLength = 3,
  tuneGrid = rfGrid,
  trControl = rfControl
)
```

```
## + : mtry=2, splitrule=gini, min.node.size=5
```

```
## - : mtry=2, splitrule=gini, min.node.size=5
## + : mtry=3, splitrule=gini, min.node.size=5
## - : mtry=3, splitrule=gini, min.node.size=5
## + : mtry=5, splitrule=gini, min.node.size=5
## - : mtry=5, splitrule=gini, min.node.size=5
## + : mtry=6, splitrule=gini, min.node.size=5
## - : mtry=6, splitrule=gini, min.node.size=5
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 3, splitrule = gini, min.node.size = 5 on full training set
```

```
rf_model
```

```
## Random Forest
##
## 6805 samples
##   10 predictor
##    2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.9620867  0.9241321
##   3     0.9714916  0.9429589
##   5     0.9714916  0.9429614
##   6     0.9713446  0.9426694
##
## Tuning parameter 'splitrule' was held constant at a value of gini
##
## Tuning parameter 'min.node.size' was held constant at a value of 5
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 3, splitrule = gini
##  and min.node.size = 5.
```

```
rf_pred <- predict(rf_model, newdata = mm_test)

confusionMatrix(rf_pred, factor(test[["stroke"]]), positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  1405   71
##        yes   17 1424
##
##                Accuracy : 0.9698
##                  95% CI : (0.963, 0.9757)
##     No Information Rate : 0.5125
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9397
##
```

```
##  Mcnemar's Test P-Value : 1.606e-08
##
##             Sensitivity : 0.9525
##             Specificity : 0.9880
##          Pos Pred Value : 0.9882
##          Neg Pred Value : 0.9519
##              Prevalence : 0.5125
##          Detection Rate : 0.4882
##    Detection Prevalence : 0.4940
##       Balanced Accuracy : 0.9703
##
##        'Positive' Class : yes
##
```

```
knitr::opts_chunk$set(echo = TRUE)
```

Error is low for the random forest model, it's accuracy is higher than baseline for all of the mtry parameters. The random forest model has an accuracy of (0.97) after evaluating it on the unseen test data. It's recall is also high (0.98) which means it will classify most true negatives correctly. The same goes for classifying true positives (Specificity: 0.96)

#Conclusions

The random forest model is great at classifying true negative cases, but performs poorly on classifying true positive cases which is what we are interested in (we want to detect people with stroke, so we can be confident in telling a patient they are at risk of stroke when we supply his/her information to the model).