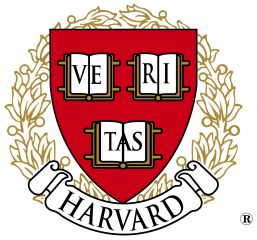


Stroke Report

Ing. José Ramón Riesgo Escovar

May 2021

INTRODUCTION TO STROKE

Stroke is one of the leading causes of death globally. Sometimes it is also called a brain attack. A stroke happens when the blood supply to part of the brain is blocked. Also, a stroke happens when a blood vessel in the brain bursts. In either case, this stops the flow of blood and due to this, part of the brain becomes damaged or dies. Unfortunately, when this happens it can cause lasting brain damage, long-term disability, and in some cases, death. Our brain controls our movements, stores our memories, is the source of our thoughts, emotions, and our language. Besides these functions the brain also controls functions like breathing.

Our brain uses 20% of the oxygen we breathe, and the arteries deliver our oxygen-rich blood to all the sections of our brain.

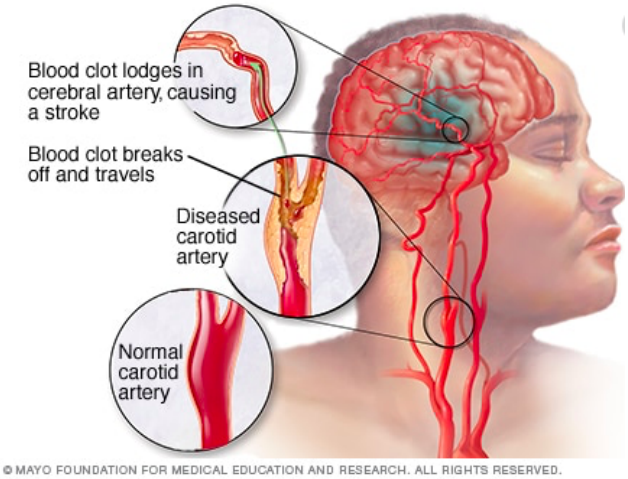


Image taken from Mayo Foundation for Medical Education and Research

A stroke happens when a blood clot blocks blood flow to the
brain or when there is a brain blood vessel burst that prevents the flow of blood

Stroke is the third major cause of disability. Long term disability affects people severely, in terms of their productive life. The aim of this report is to identify risk factors, and then, using them, to be able to predict if someone has a high risk of having a stroke.

The patient dataset was obtained from **Kaggle**. The processes and methods to ascertain whether a variable is a risk factor will be evaluated and described. We will visualize and discover insights of the dataset, ending with a conclusion, some ideas and suggestions for future work on research for early symptoms that could help prevent an actual stroke

From Kaggle we get the description of the columns in our dataset:

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

*Note: "Unknown" in smoking_status means that the information is unavailable for this patient

In the next section we will first do data preprocessing, then we will execute the exploratory data analysis, and then work on the models.

DATA PREPROCESSING ANALYSIS:

The Dataset consists of:

```
stroke_data <- read.csv("stroke.csv", header = TRUE)
dim(stroke_data)
```

```
## [1] 5110   12
```

We have 5110 records and 12 columns, 11 potential predictors, and the column that indicates if the patient actually suffered a stroke or not.

Now we will be exploring one by one the potential predictors:

1) ID : Unique Identifier

This is a number to identify the patient but it is irrelevant because it does not provide any meaningful information for our future models so we will delete this column from our dataset by applying this code:

```
stroke_data <- subset( stroke_data, select = -id )
```

2) Gender: The specific gender of the patients

With the following we will generate the actual distribution of the gender within the dataset:

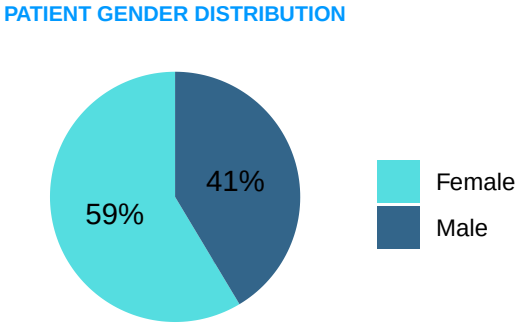
```
stroke_data %>%
  group_by(gender) %>%
  summarise(total = n())
```

```
## # A tibble: 3 x 2
##   gender total
##   <chr>  <int>
## 1 Female  2994
## 2 Male    2115
## 3 Other      1
```

We see that there is just one record of an "Other" gender. Having just one record is insignificant to the dataset and for the future prediction models, so we will eliminate it from the dataset using this code:

```
stroke_data <- subset (stroke_data, stroke_data$gender !="Other")
```

Now the dataset has the following distribution of Gender of Patients:



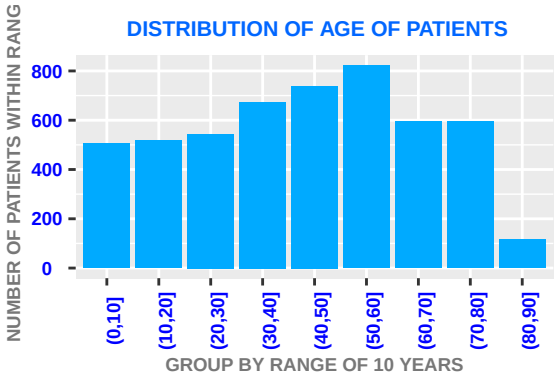
3) Age: Distribution of the age of the patients

Generating the summary of how the age is distributed within our patients

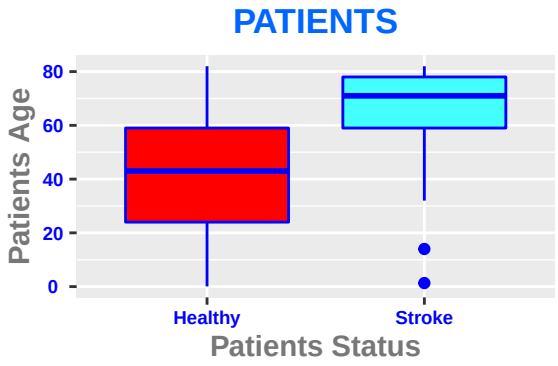
```
summary(stroke_data$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.08   25.00   45.00   43.23   61.00   82.00
```

The next chart shows how the patients are distribute by grouping them in ranges of 10 years of age



With the following boxplot we can see very clearly that there is a tendency for older patient to have a stroke:



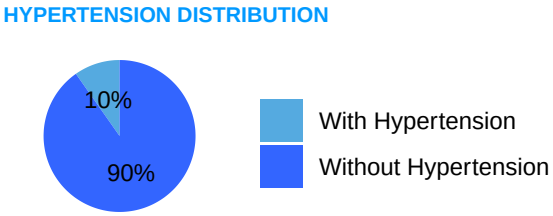
4) Hypertension: Information if the patient had hypertension or not

With the following code we visualized the total patients that have hypertension and the ones without hypertension.

```
stroke_data %>%
  mutate(text = ifelse(hypertension==0,"Without Hypertension","With Hypertension")) %>%
  group_by(text) %>%
  summarise(total = n())
```

```
## # A tibble: 2 x 2
##   text                total
##   <chr>               <int>
## 1 With Hypertension     498
## 2 Without Hypertension  4611
```

Pie chart with the hypertension distribution within our set:



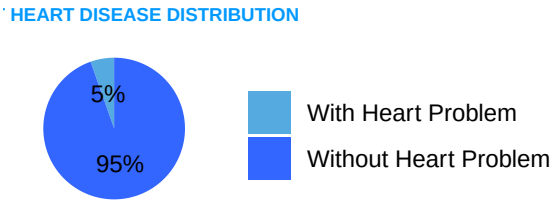
5) Heart Disease: Information if the patient had a heart problem or not

With the following code we visualized the total patients that have had heart problems and the ones without.

```
stroke_data %>%
  mutate(text = ifelse(heart_disease==0,"Without Heart Problem","With Heart Problem")) %>%
  group_by(text) %>%
  summarise(total = n())
```

```
## # A tibble: 2 x 2
##   text                  total
##   <chr>                 <int>
## 1 With Heart Problem      276
## 2 Without Heart Problem  4833
```

The following pie chart shows the distribution of patients with heart problems:



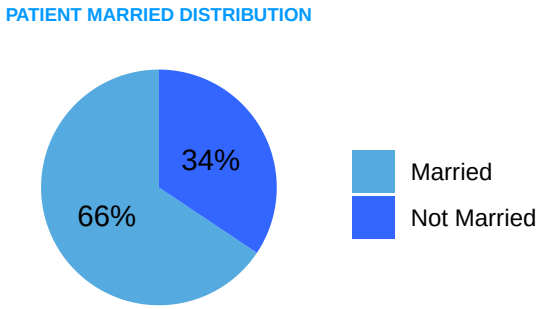
6) ever_married: Information if the patient had been married or not

With the following code we visualize how many patients were married and how many never married:

```
stroke_data %>%
  mutate(text = ifelse(ever_married=="Yes","Married","Not Married")) %>%
  group_by(text) %>%
  summarise(total = n())
```

```
## # A tibble: 2 x 2
##   text        total
##   <chr>       <int>
## 1 Married      3353
## 2 Not Married  1756
```

The following pie chart shows the distribution of the patients that were married versus the ones that never married:



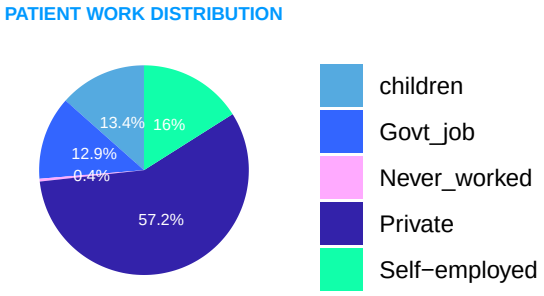
7) work_type: We will show the information if the patient have work and what type of work

The following code shows the amount of patients separated by their work situation, unless they are children:

```
stroke_data %>%
  group_by(work_type) %>%
  summarise(total = n())
```

```
## # A tibble: 5 x 2
##   work_type     total
##   <chr>         <int>
## 1 children        687
## 2 Govt_job        657
## 3 Never_worked     22
## 4 Private        2924
## 5 Self-employed   819
```

The following code shows a pie chart with the work type distribution of the patients:



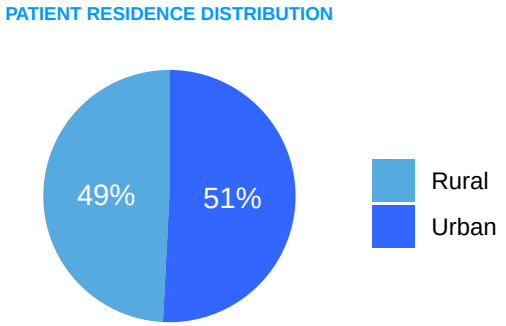
8) Residence_type: defines where the patient's live in urban or rural areas

The following code summarizes where the patients' residences are:

```
stroke_data %>%
  group_by(Residence_type) %>%
  summarise(total = n())
```

```
## # A tibble: 2 x 2
##   Residence_type total
##   <chr>          <int>
## 1 Rural           2513
## 2 Urban           2596
```

A pie chart showing the distribution of the patients' residences:



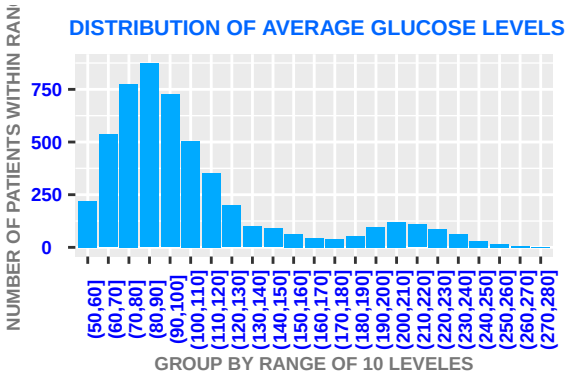
9) avg_glucose_level: defines the glucose levels of the patients in the set

With the following code we generate the summary of of the statistics of the average glucose level with the dataset:

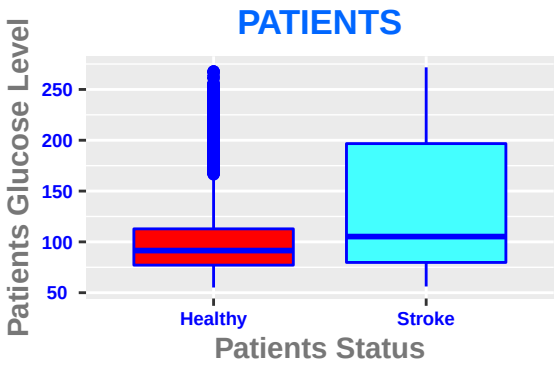
```
summary(stroke_data$avg_glucose_level)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   55.12   77.24   91.88  106.14  114.09  271.74
```

This chart shows the distribution, in ranges, from the lower limit to the higher limit in 10 units increments:



We show a box plot of the patients that had a stroke as well as the ones healthy:



Patients with high glucose levels are more prone to have a stroke.

10) bmi: defines the Body Mass Index of the patients in the set

Reviewing Body Mass Index (bmi) levels, there are patients without this information with an N/A in these rows. We also identify that the bmi values are strings within the dataset, instead of numbers. With the following code we are showing the summary of bmi

```
# To avoid warning due to N/A will disable them during this code execution
options(warn = -1) # To avoid Warning due to N/A
summary(as.numeric(stroke_data$bmi), na.rm=TRUE)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   10.30   23.50   28.10   28.89   33.10   97.60     201
```

The following code allows review of what patients do not have bmi information and had not had any stroke:

```
# How many records we have without bmi information and that have not had any stroke
sum(stroke_data$bmi=="N/A" & stroke_data$stroke==0)
```

```
## [1] 161
```

The following code allows review of what patients that do not have bmi information and had a stroke:

```
# How many records we have without bmi information and that had a stroke
sum(stroke_data$bmi=="N/A" & stroke_data$stroke==1)
```

```
## [1] 40
```

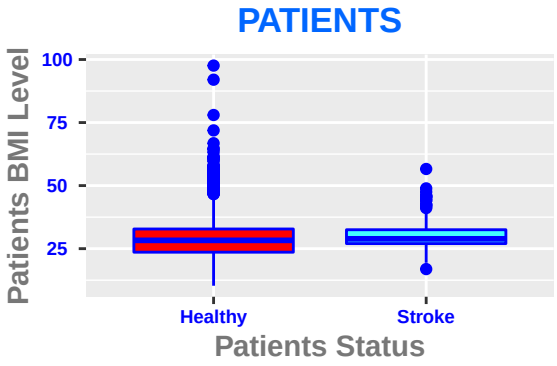
Since these 201 registers (161 that did not suffer a stroke and 40 that suffered a stroke) are relevant for our study and predictions of stroke, we will be calculate the average bmi of the dataset using this code:

```
# Calculate the mean of bmi
bmimean <- mean(as.numeric(stroke_data$bmi), na.rm = TRUE)
```

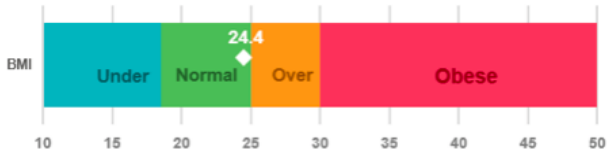
We add a new column in our dataset with the patients bmi number, instead of the string, For the patients without bmi information the mean of bmi will be assigned to them. The following code achieves this:

```
stroke_data <- stroke_data %>%
  mutate( bmi_num = ifelse(bmi=="N/A",bmimean,as.numeric(bmi)))
# Return to normal warnings
options(warn = 0L)
# Clearing the temporary variable bmimean to keep the environment clean
rm(bmimean)
```

The following boxplot shows the distribution of the patients that had a stroke and the healthy ones:



Analyzing BMI information, we see the following pictures:

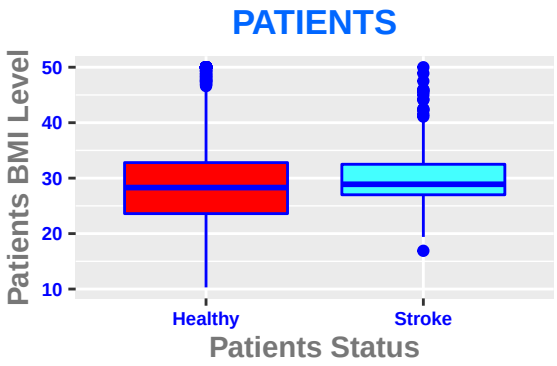


As seen here, anyone above 50 is extremely obese, so values above 56 of BMI seem out of range. All patient outliers above 50 will be adjusted 56 to avoid distortion in the models.

With the following code we will adjust this:

```
stroke_data <- stroke_data %>%
  mutate(bmi_num = ifelse(bmi_num >=50,50, bmi_num))
```

Showing the boxplot after the adjustment:



We do not see any specific trend in BMI distribution between healthy patients and patients that had had a stroke.

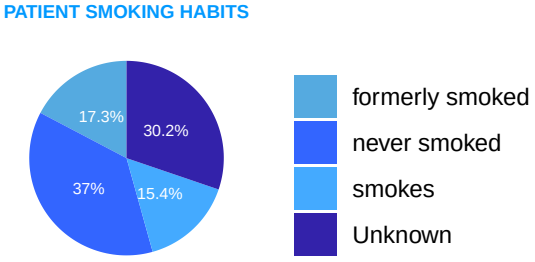
11) smoking_status: defines the patient relation to smoking

The following code shows the distribution of the patients based on their smoking status:

```
stroke_data %>%
  group_by(smoking_status) %>%
  summarise(total = n())
```

```
## # A tibble: 4 x 2
##   smoking_status  total
##   <chr>           <int>
## 1 formerly smoked   884
## 2 never smoked     1892
## 3 smokes            789
## 4 Unknown          1544
```

In the following pie chart, we observe the distribution of the smoking habits of the patients:



Note: "Unknown" in smoking_status means that the information is unavailable for this patient

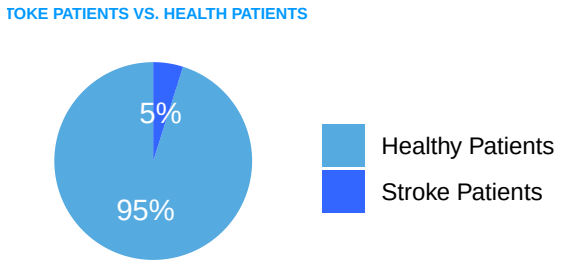
Stoke Patient in the Dataset:

The following code visualizes the amount of smoking patients versus the healthy patients in the dataset:

```
stroke_data %>%
  mutate(text = ifelse(stroke==0,"Healthy Patients","Stroke Patients")) %>%
  group_by(text) %>%
  summarise(total = n())
```

```
## # A tibble: 2 x 2
##   text             total
##   <chr>            <int>
## 1 Healthy Patients  4860
## 2 Stroke Patients    249
```

The following Pie chart show the distribution between Stroke Patients and Healthy Patients



General and preparation for models

A correlation check only accepts numerical variables, which are also used for fitting models, so we are preprocessing all categorical variables to numbers, encoding them. Also, we will scale age, avg_glucose_level and bmi because if we keep predictors that are measured at different scales they will not contribute equally to our fitting models and could create a bias. To deal with this possible problem we will standardized the age, avg_glucose_level and bmi to have a $(\mu = 0, \sigma = 1)$ before we start the fitting models.

The following code achieves all the transformation:

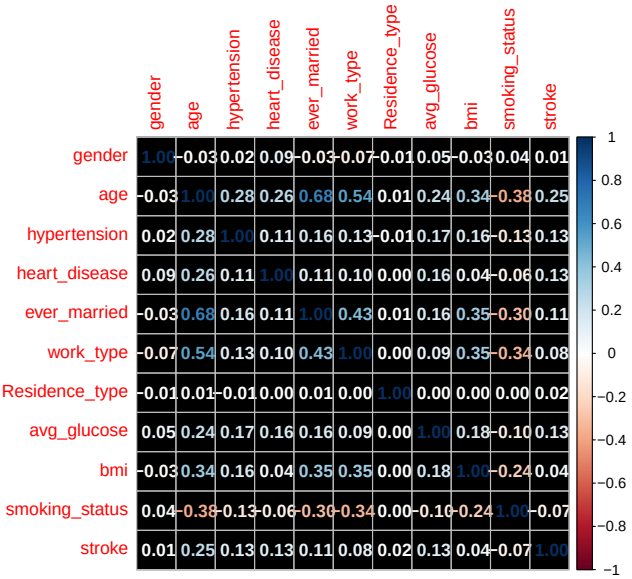
```
# mean of age
age_mean <- mean(stroke_data$age)
# sd of age
age_sd <- sd(stroke_data$age)
# mean of glucose
glucose_mean <- mean(stroke_data$avg_glucose_level)
# sd of glucose
glucose_sd <- sd(stroke_data$avg_glucose_level)
# mean of bmi
bmi_mean <- mean(stroke_data$bmi_num)
# sd of bmi
bmi_sd <- sd(stroke_data$bmi_num)

# We need to change our categorical variables too
stroke_data_num <- stroke_data %>%
  # Gender: Female 0, Male 1
  mutate(gender_num=ifelse(gender=="Female",0,1)) %>%
```

```
  # Married: Not_Married 0, Married 1
  mutate(married_num=ifelse(ever_married=="Yes",1,0)) %>%
  # In the following section we will be passing from text to numbers of work type
  # children 0, Govt_job 1, Never_worked2, Private 3, Self-employed 4
  mutate(work_type_num=sapply(work_type, function(x)
    switch(x,"children"= 0,"Govt_job"= 1,"Never_worked"= 2,"Private"= 3,
           "Self-employed"= 4))) %>%
  # Residence_type: Rural 0, Urban 1
  mutate(Residence_type_num=ifelse(Residence_type=="Urban",1,0)) %>%
  # In the following section we will be passing from text to numbers of smoking status
  # formerly smoked 0, never smoked 1, smokes 2, Unknown 3
  mutate(smoking_status_num=sapply(smoking_status, function(x)
    switch(x,"formerly smoked"= 0,"never smoked"= 1,"smokes"= 2,"Unknown"= 3))) %>%
  # Adjust/Fit the values of age
  mutate(age_fit=((age - age_mean)/age_sd)) %>%
  # Adjust/Fit the values of glucose
  mutate(glucose_fit=((avg_glucose_level - glucose_mean)/glucose_sd)) %>%
  # Adjust/Fit the values of bmi
  mutate(bmi_fit=((bmi_num - bmi_mean)/bmi_sd)) %>%
  dplyr::select(gender=gender_num, age=age_fit, hypertension,
         heart_disease, ever_married=married_num,
         work_type=work_type_num, Residence_type=Residence_type_num,
         avg_glucose= glucose_fit, bmi = bmi_fit,
         smoking_status=smoking_status_num, stroke )

# We remove the temporary values to keep as clean as possible the environment
rm(age_mean,age_sd,bmi_mean,bmi_sd,glucose_mean,glucose_sd)
```

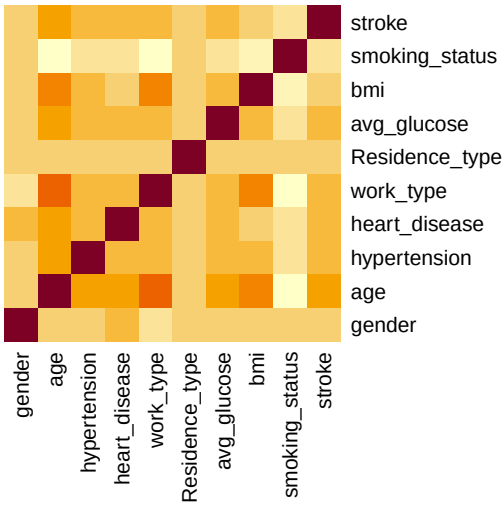
This is the correlation matrix:



There seems to be multicollinearity between age and ever_married because we have a high correlation of 0.68. In principle, age contains more information if a patient is susceptible to a stroke and we will discard the ever married category from our predictors. The following code will execute this:

```
stroke_data_num <- subset( stroke_data_num, select = -ever_married )
```

The following is the Headmap of the predictors:



The full dataset will be split: we use 80% as a training set, and 20% as a test set by using the following code:

```
# Set seed to 5 to have always the same results and it generate
# and already review that there is a  balance betweeen the Stroke patients of
# the train and test set
set.seed(5, sample.kind = "Rounding")
#We need to first create a partition of the dataset for training 80% and 20% testing
test_index <- createDataPartition(stroke_data_num$stroke, times = 1, p = 0.2, list = FALSE)
# Generate the sets for the models:
# For convenience and more clarity we will split in "x" the predictors and
# "y" the actual value of stroke or not stoke
test_x <- stroke_data_num[test_index,1:9]
test_y <- stroke_data_num[test_index,10]
train_x <- stroke_data_num[-test_index,1:9]
train_y <- stroke_data_num[-test_index,10]
```

We assign “S” to the patients with Stroke and and “H” to the Healthy patients. This will help in having a categorical response instead of 0 and 1, so that the models generate “S” or “H” as the output of the models. This code will generate these changes to the dataset of Testing and Training.

```
train_y <- ifelse(train_y==1,"S","H")
test_y  <- ifelse(test_y==1,"S","H")
```

Verify the percentage of stroke patients in the training set and also the amount of stroke patients by executing this code:

```
sum(train_y=="S")/length(train_y)
```

```
## [1] 0.04869097
```

```
# How many stroke patients in Training set
sum(train_y=="S")
```

```
## [1] 199
```

Verify the percentage of stroke patients in the test set and also the amount of stroke patients by executing this code:

```
sum(test_y=="S")/length(test_y)
```

```
## [1] 0.04892368
```

```
# How many stroke patients in Test set
sum(test_y=="S")
```

```
## [1] 50
```

We have a very reasonable balance between both sets, almost the same percentage of stroke patients.

MODELS WITH THE FULL SET

We will run the following models:

1) Generalized Linear Model:

The following code executes the model and shows the accuracy and the actual predictions for stroke:

```
# We apply the method "glm" to the training set
generate_glm <- train(train_x, train_y, method = "glm")
# Then with the generated model we create the predictions for the test set
glm_predictions <- predict(generate_glm, test_x)
# We calculate the accuracy of the prediction
mean(glm_predictions == test_y)
```

```
## [1] 0.9510763
```

```
# How many patients calculate with stroke
sum(glm_predictions=="S")
```

```
## [1] 0
```

2) Generalized Additive Model for LOESS:

The following code executes the model and shows the accuracy and the actual predictions for stroke:

```
# We apply the method "gamLoess" to the training set
generate_gamloess <- train(train_x, train_y, method = "gamLoess")
# Then with the generated model we create the predictions for the test set
gamloess_predictions <- predict(generate_gamloess, test_x)
# We calculate the accuracy of the prediction
mean(gamloess_predictions == test_y)
```

```
## [1] 0.9510763
```

```
# How many patients calculate with Stroke
sum(gamloess_predictions=="S")
```

```
## [1] 0
```

3) K-Nearest Neighbor (knn):

The following code executes the model and shows the accuracy and the actual predictions for stroke:

```
# We apply the method knn to the training set
generate_knn <- train(train_x, train_y, method = "knn",
                      tuneGrid = data.frame(k = seq(1,40,2)))
# Then with the generated model we create the prediction for the test set
knn_predictions <- predict(generate_knn, test_x)
# We calculate the accuracy of the prediction
mean(knn_predictions == test_y)
```

```
## [1] 0.9510763
```

```
# How many patients calculate with Stroke
sum(knn_predictions=="S")
```

```
## [1] 0
```

```
#Cleaning all the used variables to keep the environment tidy
rm(generate_glm,generate_knn,generate_gamloess)
rm(gamloess_predictions,glm_predictions,knn_predictions)
```

The following code executes the model and shows the accuracy and the actual predictions for stroke: Because we are not really predicting, just generating all predictions as Healthy, the dataset has a bias, because there is a big distribution of 95% of the dataset as healthy, so we are not generating any predictions for stroke patients. To solve this, we are going to adjust the dataset to have close to 50% patients with stroke and 50% of aleatory healthy patients, so that we can test our models to predict the potential stroke and healthy patients based on the 9 predictors.

Adjusting the dataset to produce a more balance dataset between Stroke and Healthy Patients:

Passing all the stroke patients to a temporary dataset with the following code:

```
# Pass all Stroke patients to a dataset subset
positive_stroke_patients <- stroke_data_num %>%
  filter(stroke==1)
```

Calculating the number of patients with strokes with the following code:

```
nrow(positive_stroke_patients)
```

```
## [1] 249
```

The set has 249 patients with Stroke

Passing all the Healthy patients to a temporary dataset with the following code:

```
# Generate a subset of all helthy patients
health_stroke_patients <- stroke_data_num %>%
  filter(stroke==0)
```

Based on the full size of this healthy dataset we are going to partition 6% of the set and with that produce a similar size dataset to the stroke dataset with the following code:

```
# Set the seed to 3
set.seed(3, sample.kind = "Rounding")
# Generate a partition of 6% of the healthy patients around 250 registries
health_index <-createDataPartition(health_stroke_patients$age, times = 1, p = 0.06, list = FALSE)
# Generate this new healthy patients subset
health_stroke_patients <- health_stroke_patients[health_index,1:10]
```

Calculating the number of healthy patients with the following code:

```
nrow(health_stroke_patients)
```

```
## [1] 294
```

The set has 294 healthy patients

Now with the following code we combine both sets to have a more balance dataset, without any bias for healthy patient. We also calculate the count of patients:

```
# Combine the stroke patients with the generated subset
stroke_data_num_adj <- positive_stroke_patients %>%
  union(health_stroke_patients)
# Visualize the new size of the dataset
nrow(stroke_data_num_adj)
```

```
## [1] 543
```

```
# Cleaning the environment of the temporary objects
rm(health_stroke_patients,positive_stroke_patients, health_index)
```

With the following code we split again the patients, but this time with the adjusted dataset 80% of records in the training set and 20% in the test set:

```r
# Set seed to 5 to have always the same results and it generate
# and already review that there is a  balance betweeen the Stroke patients of
# the train and test set
set.seed(5, sample.kind = "Rounding")
#We need to first create a partition of the dataset for training 80% and 20% testing
test_index <- createDataPartition(stroke_data_num_adj$stroke, times = 1, p = 0.2, list = FALSE)
# Generate the sets for the models:
# For convenience and more clarity we will split in "x" the predictors and
# "y" the actual value of stroke or not stoke
test_x <- stroke_data_num_adj[test_index,1:9]
test_y <- stroke_data_num_adj[test_index,10]
train_x <- stroke_data_num_adj[-test_index,1:9]
train_y <- stroke_data_num_adj[-test_index,10]

# Cleaning the environment
rm(test_index, stroke_data_num_adj)
```

Applying the same logic we used before to adjust the output of the models to categorical values with the following code, and verifying that we have a reasonable balance between both sets of the stroke patients as well as the amount of stroke patients in each set:

```r
train_y <- ifelse(train_y==1,"S","H")
test_y  <- ifelse(test_y==1,"S","H")
# Calculate the % of stoke patients in the train set
sum(train_y=="S")/length(train_y)
```

```
## [1] 0.4585253
```

```r
# How many stroke patients in Training set
sum(train_y=="S")
```

```
## [1] 199
```

```r
# Calculate the % of stoke patients in the test set
sum(test_y=="S")/length(test_y)
```

```
## [1] 0.4587156
```

```r
# How many stroke patients in Test set
sum(test_y=="S")
```

```
## [1] 50
```

Looks a good balance between both sets, almost the same percentage of stroke patients, so we will now perform the models again:

MODELS EXECUTED WITH ADJUSTED SET

Now we will re-execute the 3 models plus others to try to get the more accurate model with the adjusted dataset.

1) Generalized Linear Model (glm):

The following code executes the Model and shows the accuracy and the actual predictions for stroke:

```
# We apply the method "glm" to the training set
generate_glm <- train(train_x, train_y, method = "glm")
# Then with the generated model we create the predictions for the test set
glm_predictions <- predict(generate_glm, test_x)
# We calculate the accuracy of the prediction
model_accuracy <- mean(glm_predictions == test_y)
# How many patients calculate with stroke
stroke_patients_predicted <- sum(glm_predictions=="S")
# I generate a Table with all the Methods and the Accuracy of each
accuracy_results <- data.frame(METHOD = "glm", ACCURACY = model_accuracy,
                               STROKE_PATIENTS = stroke_patients_predicted)
# Table of output
accuracy_results %>% knitr::kable()
```

METHOD	ACCURACY	STROKE_PATIENTS
glm	0.7431193	58

2) Quadratic Discriminant Analysis (qda):

The following code executes the Model and shows the accuracy and the actual predictions for stroke:

```
# We apply the method "qda" to the training set
generate_qda <- train(train_x, train_y, method = "qda")
# Then with the generated model we create the predictions for the test set
qda_predictions <- predict(generate_qda, test_x)
# We calculate the accuracy of the prediction
model_accuracy <- mean(qda_predictions == test_y)
# How many patients calculate with Stroke
stroke_patients_predicted <- sum(qda_predictions=="S")
# Adding the results of the model qda
accuracy_results <- bind_rows(accuracy_results,
                              data.frame(METHOD = "qda", ACCURACY = model_accuracy,
                                         STROKE_PATIENTS = stroke_patients_predicted))
# Table of output
accuracy_results %>% knitr::kable()
```

METHOD	ACCURACY	STROKE_PATIENTS
glm	0.7431193	58
qda	0.6880734	54

3) Generalized Additive Model for LOESS:

The following code executes the Model and shows the accuracy and the actual predictions for stroke:

```
# We apply the method "gamLoess" to the training set
generate_loess <- train(train_x, train_y, method = "gamLoess")
```

```
# Then with the generated model we create the predictions for the test set
gamloess_predictions <- predict(generate_loess, test_x)
# We calculate the accuracy of the prediction
model_accuracy <- mean(gamloess_predictions == test_y)
# How many patients calculate with Stroke
stroke_patients_predicted <- sum(gamloess_predictions=="S")
# Adding the results of the model gamLoess
accuracy_results <- bind_rows(accuracy_results,
                              data.frame(METHOD = "gamLoess", ACCURACY = model_accuracy,
                                         STROKE_PATIENTS = stroke_patients_predicted))
# Table of output
accuracy_results %>% knitr::kable()
```

METHOD	ACCURACY	STROKE_PATIENTS
glm	0.7431193	58
qda	0.6880734	54
gamLoess	0.7614679	60

4) K-Nearest Neighbor (knn):

The following code executes the Model and shows the accuracy and the actual predictions for stroke:

```
# We apply the method knn to the training set
generate_knn <- train(train_x, train_y, method = "knn",
                      tuneGrid = data.frame(k = seq(1,40,2)))
# Then with the generated model we create the prediction for the test set
knn_predictions <- predict(generate_knn, test_x)
# We calculate the accuracy of the prediction
model_accuracy <- mean(knn_predictions == test_y)
# How many patients calculate with Stroke
stroke_patients_predicted <- sum(knn_predictions=="S")
# Adding the results of the model knn
accuracy_results <- bind_rows(accuracy_results,
                              data.frame(METHOD = "knn", ACCURACY = model_accuracy,
                                         STROKE_PATIENTS = stroke_patients_predicted))
# Table of output
accuracy_results %>% knitr::kable()
```

METHOD	ACCURACY	STROKE_PATIENTS
glm	0.7431193	58
qda	0.6880734	54
gamLoess	0.7614679	60
knn	0.7247706	58

5) Random Forest (rf):

The following code executes the Model and shows the accuracy and the actual predictions for stroke:

```
# We apply the method rf to the training set
# We apply the tuning from 9 to 11
tuning <- data.frame(mtry = c(  9, 11))
# We apply the method rf
```

```
generate_rf <- train(train_x, train_y, method = "rf",
                     tuneGrid = tuning, importance = TRUE)
# Then with the model we create the prediction for the test set
rf_predictions <- predict(generate_rf, test_x)
# How many patients calculate with Stroke
model_accuracy <- mean(rf_predictions == test_y)
# How many patients calculate with Stroke
stroke_patients_predicted <- sum(rf_predictions=="S")
# Adding the results of the model rf
accuracy_results <- bind_rows(accuracy_results,
                              data.frame(METHOD = "rf", ACCURACY = model_accuracy,
                                         STROKE_PATIENTS = stroke_patients_predicted))
# Table of output
accuracy_results %>% knitr::kable()
```

METHOD	ACCURACY	STROKE_PATIENTS
glm	0.7431193	58
qda	0.6880734	54
gamLoess	0.7614679	60
knn	0.7247706	58
rf	0.7981651	54

6) K-Means clustering algorithm (k-means):

The following code executes the Model and shows the accuracy and the actual predictions for stroke:

```
# The predict_kmeans() function defined here takes two arguments - a matrix of
# observations x and a k-means object k - and assigns each row of x to a cluster
# from k.
predict_kmeans <- function(x, k) {
  centers <- k$centers    # extract cluster centers
  # calculate distance to cluster centers
  distances <- sapply(1:nrow(x), function(i){
    apply(centers, 1, function(y) dist(rbind(x[i,], y)))
  })
  max.col(-t(distances))  # select cluster with min distance to center
}
# Perform k-means clustering on the training set with 2
# centers and assign the output to k.
k <- kmeans(train_x, centers = 2)
# Calculate the predictions with the different "k"
kmean_predictions <- ifelse(predict_kmeans(test_x,k)==1,"S", "H")
# Calculate the accuracy of the model kmean
model_accuracy <- mean(kmean_predictions == test_y)
# How many patients calculate with Stroke
stroke_patients_predicted <- sum(kmean_predictions=="S")
# Adding the results of the model k-means
accuracy_results <- bind_rows(accuracy_results,
                              data.frame(METHOD = "k-means", ACCURACY = model_accuracy,
                                         STROKE_PATIENTS = stroke_patients_predicted))
# Table of output
accuracy_results %>% knitr::kable()
```

METHOD	ACCURACY	STROKE_PATIENTS
glm	0.7431193	58
qda	0.6880734	54
gamLoess	0.7614679	60
knn	0.7247706	58
rf	0.7981651	54
k-means	0.6513761	72

7) More optimal Generalized Additive Model (bam):

The following code executes the Model and shows the accuracy and the actual predictions for stroke:

```
# We apply the method "bam" to the training set
generate_bam <- train(train_x, train_y, method = "bam")
# Then with the generated model we create the predictions for the test set
bam_predictions <- predict(generate_bam, test_x)
# We calculate the accuracy of the prediction
model_accuracy <- mean(bam_predictions == test_y)
# How many patients calculate with Stroke
stroke_patients_predicted <- sum(bam_predictions=="S")
# Adding the results of the model bam
accuracy_results <- bind_rows(accuracy_results,
                              data.frame(METHOD = "bam", ACCURACY = model_accuracy,
                                         STROKE_PATIENTS = stroke_patients_predicted))
# Table of output
accuracy_results %>% knitr::kable()
```

METHOD	ACCURACY	STROKE_PATIENTS
glm	0.7431193	58
qda	0.6880734	54
gamLoess	0.7614679	60
knn	0.7247706	58
rf	0.7981651	54
k-means	0.6513761	72
bam	0.7522936	59

8) Conditional Random Forest (cForest):

The following code executes the Model and shows the accuracy and the actual predictions for stroke:

```
# We apply the method "cforest" to the training set
# We apply the tuning from 3 to 7 of odd numbers
tuning <- data.frame(mtry = c(3, 5, 7))
# apply the model
generate_cforest <- train(train_x, train_y, method = "cforest",
                          tuneGrid = tuning)
# Then with the generated model we create the predictions for the test set
cforest_predictions <- predict(generate_cforest, test_x)
# We calculate the accuracy of the prediction
model_accuracy <- mean(cforest_predictions == test_y)
# How many patients calculate with Stroke
stroke_patients_predicted <- sum(cforest_predictions=="S")
# Adding the results of the model cforest
```

```r
accuracy_results <- bind_rows(accuracy_results,
                              data.frame(METHOD = "cforest", ACCURACY = model_accuracy,
                                         STROKE_PATIENTS = stroke_patients_predicted))
# Table of output
accuracy_results %>% knitr::kable()
```

METHOD	ACCURACY	STROKE_PATIENTS
glm	0.7431193	58
qda	0.6880734	54
gamLoess	0.7614679	60
knn	0.7247706	58
rf	0.7981651	54
k-means	0.6513761	72
bam	0.7522936	59
cforest	0.7339450	61

9) Bayesian generalized linear models (bayesglm):

The following code executes the Model and shows the accuracy and the actual predictions for stroke:

```r
# We apply the method "bayesglm" to the training set
generate_bayesglm <- train(train_x, train_y, method = "bayesglm")
# Then with the generated model we create the predictions for the test set
bayesglm_predictions <- predict(generate_bayesglm, test_x)
# We calculate the accuracy of the prediction
model_accuracy <- mean(bayesglm_predictions == test_y)
# How many patients calculate with stroke
stroke_patients_predicted <- sum(bayesglm_predictions=="S")
# Adding the results of the model bayesglm
accuracy_results <- bind_rows(accuracy_results,
                              data.frame(METHOD = "bayesglm", ACCURACY = model_accuracy,
                                         STROKE_PATIENTS = stroke_patients_predicted))
# Table of output
accuracy_results %>% knitr::kable()
```

METHOD	ACCURACY	STROKE_PATIENTS
glm	0.7431193	58
qda	0.6880734	54
gamLoess	0.7614679	60
knn	0.7247706	58
rf	0.7981651	54
k-means	0.6513761	72
bam	0.7522936	59
cforest	0.7339450	61
bayesglm	0.7431193	58

10) Penalized Discriminant Analysis (pda2):

The following code executes the Model and shows the accuracy and the actual predictions for stroke:

```r
# We apply the method "pda2" to the training set
generate_pda2 <- train(train_x, train_y, method = "pda2")
```

```r
# Then with the generated model we create the predictions for the test set
pda2_predictions <- predict(generate_pda2, test_x)
# We calculate the accuracy of the prediction
model_accuracy <- mean(pda2_predictions == test_y)
# How many patients calculate with stroke
stroke_patients_predicted <- sum(pda2_predictions=="S")
# Adding the results of the model pda2
accuracy_results <- bind_rows(accuracy_results,
                              data.frame(METHOD = "pda2", ACCURACY = model_accuracy,
                                         STROKE_PATIENTS = stroke_patients_predicted))
# Show the table with all the values
accuracy_results %>% knitr::kable()
```

METHOD	ACCURACY	STROKE_PATIENTS
glm	0.7431193	58
qda	0.6880734	54
gamLoess	0.7614679	60
knn	0.7247706	58
rf	0.7981651	54
k-means	0.6513761	72
bam	0.7522936	59
cforest	0.7339450	61
bayesglm	0.7431193	58
pda2	0.7339450	59

11) General Ensemble with 10 models:

This tries to analyze if the combination of the 10 models provides a better accuracy for the prediction.

The following code executes the Model and shows the accuracy and the actual predictions for stroke:

```r
# We will generate an ensemble with the 10 models used to try to enhance our
# predictions

# This function returns a value of 1 if the patient under review was predicted with a
# stroke, if not 0 if he was healthy
change_value <- function(item) {
  values <- ifelse(item == "S",1,0)
}
# This value evaluates if 6 or more give a "Stroke" value we return "Stroke" if
# not the patient is Healthy based on the combination of predictions from all the models
return_value <- function(item) {
  valores <- ifelse(item >= 6 , "S","H")
}
# This value evaluates if 3 or more give a "Stroke" value we return "Stroke" if
# not the patient is Healthy based on the combination of predictions from the 5 more
# accurate models models
return_value_opt <- function(item) {
  valores <- ifelse(item >= 3 , "S","H")
}
# Combination of all predictions to try to get a better prediction in the ensemble
ensemble_res <- return_value(  change_value(glm_predictions) +
                               change_value(qda_predictions) +
```

```
                                change_value(gamloess_predictions) +
                                change_value(knn_predictions) +
                                change_value(rf_predictions) +
                                change_value(kmean_predictions) +
                                change_value(bam_predictions) +
                                change_value(cforest_predictions) +
                                change_value(bayesglm_predictions) +
                                change_value(pda2_predictions))

# General ensemble accuracy
model_accuracy <- mean(ensemble_res == test_y)
stroke_patients_predicted <- sum(ensemble_res=="S")
# Adding the results of the general ensemble model
accuracy_results <- bind_rows(accuracy_results,
                              data.frame(METHOD = "General Ensemble", ACCURACY = model_accuracy,
                                         STROKE_PATIENTS = stroke_patients_predicted))
# Show the table with all the values
accuracy_results %>% knitr::kable()
```

METHOD	ACCURACY	STROKE_PATIENTS
glm	0.7431193	58
qda	0.6880734	54
gamLoess	0.7614679	60
knn	0.7247706	58
rf	0.7981651	54
k-means	0.6513761	72
bam	0.7522936	59
cforest	0.7339450	61
bayesglm	0.7431193	58
pda2	0.7339450	59
General Ensemble	0.7339450	59

12) Optimized Ensemble with best 5 models models:

This tries to analyze if the combination of the 5 best accuracy models provides a better accuracy for the prediction.

The following code executes the model and shows the accuracy and the actual predictions for stroke:

```
# Combination of the better 5 models in the "optimal" ensemble
ensemble_res_optimal <- return_value_opt( change_value(glm_predictions) +
                                          change_value(gamloess_predictions) +
                                          change_value(bam_predictions) +
                                          change_value(rf_predictions) +
                                          change_value(bayesglm_predictions))
# Optimal ensemble accuracy
model_accuracy <- mean(ensemble_res_optimal == test_y)
stroke_patients_predicted <- sum(ensemble_res_optimal=="S")
accuracy_results <- bind_rows(accuracy_results,
                              data.frame(METHOD = "Optimal Ensemble", ACCURACY = model_accuracy,
                                         STROKE_PATIENTS = stroke_patients_predicted))
# Show the table with all the values
accuracy_results %>% knitr::kable()
```

METHOD	ACCURACY	STROKE_PATIENTS
glm	0.7431193	58
qda	0.6880734	54
gamLoess	0.7614679	60
knn	0.7247706	58
rf	0.7981651	54
k-means	0.6513761	72
bam	0.7522936	59
cforest	0.7339450	61
bayesglm	0.7431193	58
pda2	0.7339450	59
General Ensemble	0.7339450	59
Optimal Ensemble	0.7522936	59

```
# Cleaning of Variables
rm(k,generate_bam,generate_bayesglm,generate_cforest,generate_glm,generate_knn)
rm(generate_loess, generate_pda2, generate_qda, generate_rf,tuning)
rm(bam_predictions,bayesglm_predictions,cforest_predictions,gamloess_predictions)
rm(glm_predictions,knn_predictions,qda_predictions,rf_predictions,pda2_predictions)
```

CONCLUTION

The best model and prediction were the Random Forest model. I believe that we might be missing additional information or predictors to define more accurately when a patient will potentially get a stroke. With the maximum accuracy of 79% based on the model, we are still far from an optimal target of 90% to 95%, that is ideal. There might be better algorithms. However, we can confirm that as age progresses it tends to increment the probability of a stroke as well as high levels of glucose combined with patients that have worked. Nevertheless, it seems that because of the relatively small number of patients with perhaps limited predictors, we were not able to be more accurate than 79%. Maybe other predictors, like cholesterol and vascular risk factor, and other metabolic factors, as well as depression and anxiety, could complement the dataset and could help in being more accurate in future studies.

Stroke is really a very big problem and while we are not able to make more accurate predictions, at least we should be aware of how to react very soon if there are any signals of a potential stroke... FAST is key:

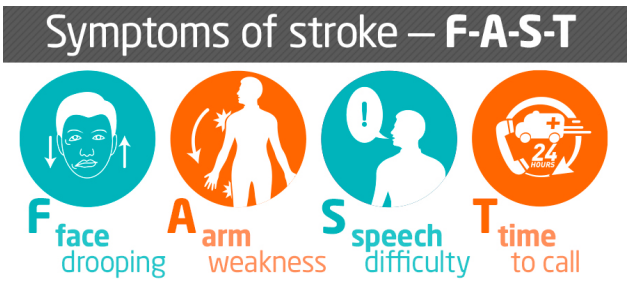


Image taken from CarePoint Health

Dedicated to all the patients that are still struggling to return to a normal live after a stroke.