# Building a Stroke Prediction System

# 1. Introduction

According to a WHO survey, worldwide, cerebrovascular accidents (stroke) are the second leading cause of death and the third leading cause of disability. Stroke, the sudden death of some brain cells due to lack of oxygen when the blood flow to the brain is lost by blockage or rupture of an artery to the brain, is also a leading cause of dementia and depression.
Strokes mainly affect individuals at the peak of their productive life. Despite its enormous impact on countries' socio-economic development, this growing crisis has received very little attention to date. (Ref: *Bulletin of the World Health Organization 2016; 94:634-634A*)

The purpose of this project is to use a publicly available dataset to build a stroke prediction system as a part of the Harvardx Professional Data Scientist program. I believe that informative websites and stroke prediction models available for everyone online, can increase the awareness of the population for this terrible disease.

The project is based on the Stroke Prediction Dataset publicly available on (https://www.kaggle.com/fedesoriano/stroke-prediction-dataset). Unfortunately, the descrioption of the dataset is not very detailed. We do not know what survey this dataset is based on, I suppose the survey was conducted in one of the countries of the developed world. The dataset is eligible 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 relevant information for more than 5000 patients.

I divided the dataset to two subsets. The larger subset (called *stroke*) contains the 80% of the data and can be used as a training set during the development of the machine algorithms. The smaller subset (called *validation*) contains the remaining data. It cannot be used for training purposes, it is only for validation of the algorithms.

In this project I tried to build a prediction system, which predict whether a patient is likely to get stroke on the validation data. The goal during the training of the algorithms is not to achieve as much overall accuracy as possible, but to build an algorithm with balanced performance measures.

# 2. Exploratory Data Analysis

At first, we need to be familiarized with the two datasets (*stroke* and *validation*) we are working with.

## 2.1 Dataset Dimensions

I used the *stroke* as the training set and saved the *validation* for evaluating the performance metrics of the final algorithm.

Dataset Dimensions

| Dataset | No. of Rows | No of Columns |
|---|---:|---:|
| stroke | 4087 | 12 |
| validation | 1022 | 12 |

## 2.2 Missing Data

It is important to know if the dataset contains any missing values, because they can cause us difficulties during the development of the algorithm, so we have to address them.

Number of N/A-s

| | **Stroke** | **Validation** |
|---|---:|---:|
| id | 0 | 0 |
| gender | 0 | 0 |
| age | 0 | 0 |
| hypertension | 0 | 0 |
| heart_disease | 0 | 0 |
| ever_married | 0 | 0 |
| work_type | 0 | 0 |
| Residence_type | 0 | 0 |
| avg_glucose_level | 0 | 0 |
| bmi | 159 | 42 |
| smoking_status | 0 | 0 |
| stroke | 0 | 0 |

The above table tells us we there are some missing values, in the *bmi* columns. To handle this problem, I substituted the *NA* values with the median bmi of the whole dataset.

## 2.3 Dataset Structure

The dataset is in tidy format. It contains 12 columns (features) and give us the following information:

- id <numeric>: Unique identifier of the patient
- gender <character>: The gender of the patient: “Male”, “Female” or “Other”
- age <numeric>: The age of the patient
- hypertension <numeric>: Binary variable: “0” if the patient doesn’t have hypertension, “1” if the patient has hypertension
- heart_disease <numeric>: Binary variable: “0” if the patient doesn’t have any heart diseases, “1” if the patient has a heart disease
- ever_married <character>: “No” or “Yes”
- work_type <character>: The employment status of the patient: “children”, “Govt_jov”, “Never_worked”, “Private” or “Self-employed”
- Residence_type <character>: The residence type of the patient: “Rural” or “Urban”

- avg_glucose_level <numeric>: The average glucose level in the patient's blood
- bmi <character>: The body mass index of the patient
- smoking_status <character>: The smoking status of the patient: "formerly smoked", "never smoked", "smokes" or "Unknown"*
- stroke <numeric>: Binary variable: "1" if the patient had a stroke or "0" if not

Notes:
1. "Unknown" in smoking_status means that the information is unavailable for this patient
2. The description of the dataset does not define what gender "Other" means. There is only 1 patient in the dataset with this gender, so I removed it.

Preview of the dataset (Columns: 1-6)

| | id | gender | age | hypertension | heart_disease | ever_married |
|---|---|---|---|---|---|---|
| 1 | 9046 | Male | 67 | 0 | 1 | Yes |
| 2 | 51676 | Female | 61 | 0 | 0 | Yes |
| 3 | 31112 | Male | 80 | 0 | 1 | Yes |
| 4 | 60182 | Female | 49 | 0 | 0 | Yes |
| 5 | 1665 | Female | 79 | 1 | 0 | Yes |
| 7 | 53882 | Male | 74 | 1 | 1 | Yes |

Preview of the dataset (Columns: 7-12)

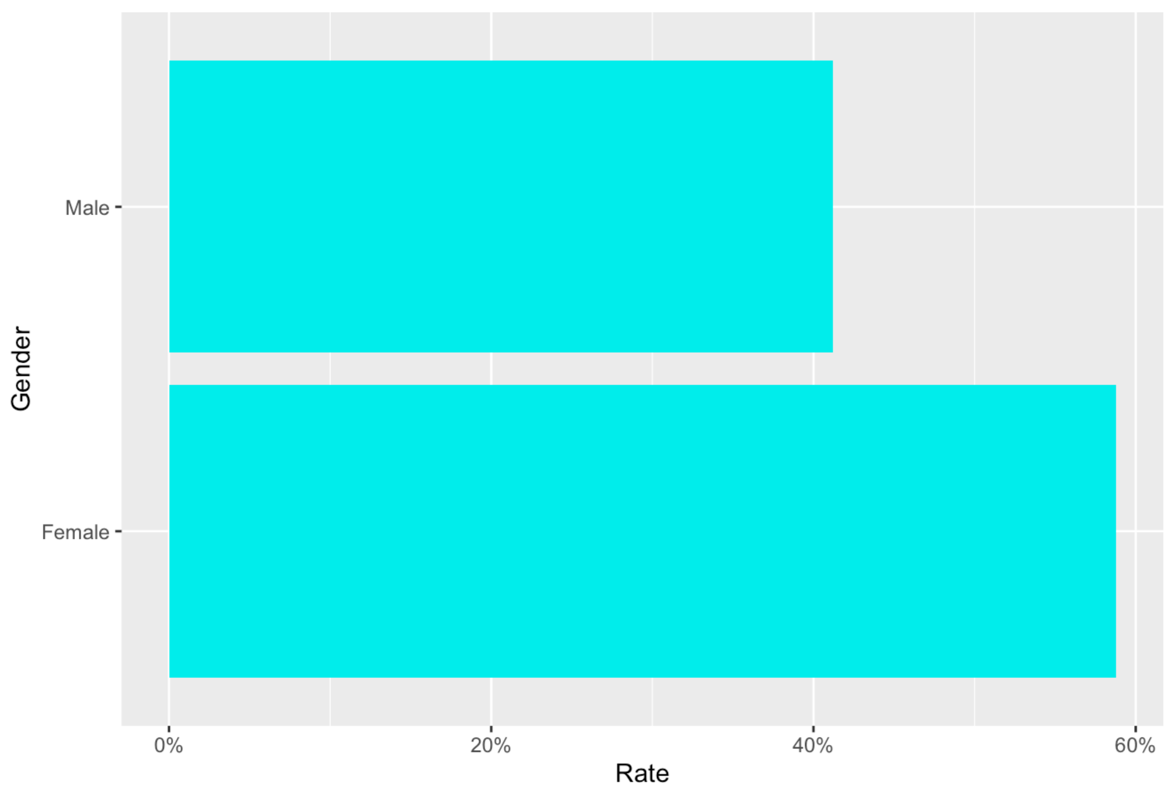
| | work_type | Residence_type | avg_glucose_level | bmi | smoking_status | stroke |
|---|---|---|---|---|---|---|
| 1 | Private | Urban | 228.69 | 36.6 | formerly smoked | 1 |
| 2 | Self-employed | Rural | 202.21 | 28.1 | never smoked | 1 |
| 3 | Private | Rural | 105.92 | 32.5 | never smoked | 1 |
| 4 | Private | Urban | 171.23 | 34.4 | smokes | 1 |
| 5 | Self-employed | Rural | 174.12 | 24.0 | never smoked | 1 |
| 7 | Private | Rural | 70.09 | 27.4 | never smoked | 1 |

Now let us delve into the details and see the distribution of some features. We need to get a clearer picture of the dataset, and the group of the patients whose data the dataset was created from.

## 2.4 Distribution of Genders

The shares of women and men in the human population is approximately 50-50%, but it varies a bit across regions. The plot below shows us nearly the same gender proportions in the dataset.

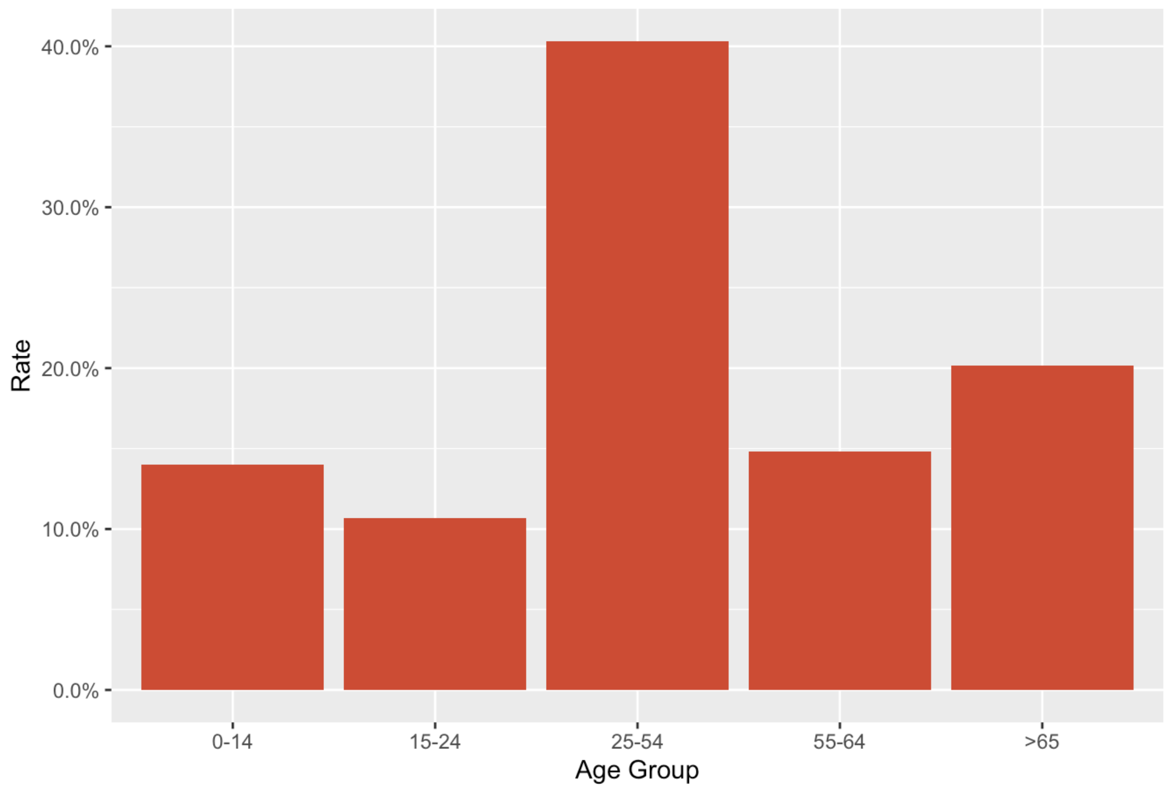
Distribution of Genders



## 2.5 Distribution of Age

The age structure of the dataset is the following:

Distribution of Age



The above plot shows us that all age groups are represented in the dataset, however the stroke incidents in the youngest age group are extremely rare.

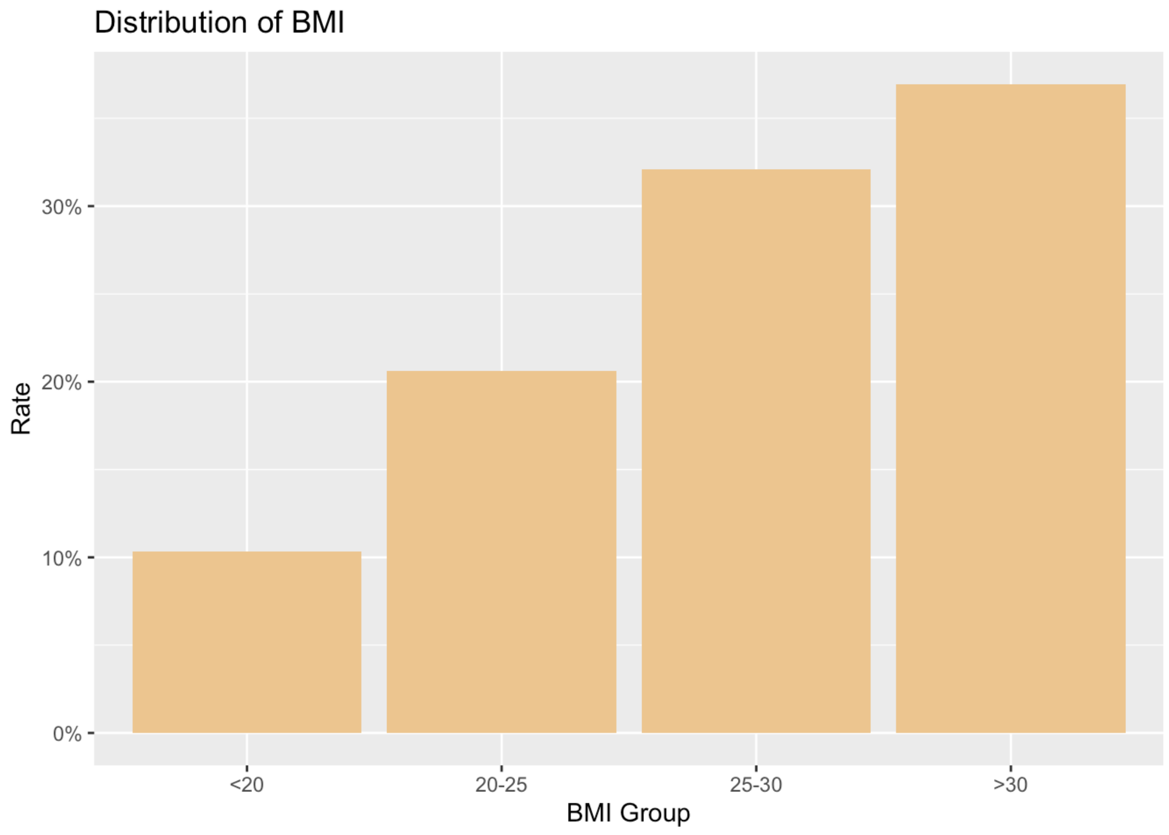
## 2.6 Distribution of the Body Mass Index

Body Mass Index (BMI) is a measure for indicating nutritional status in adults. It is defined as a person's weight in kilograms divided by the square of the person's height in metres (kg/m2).

BMI Categories

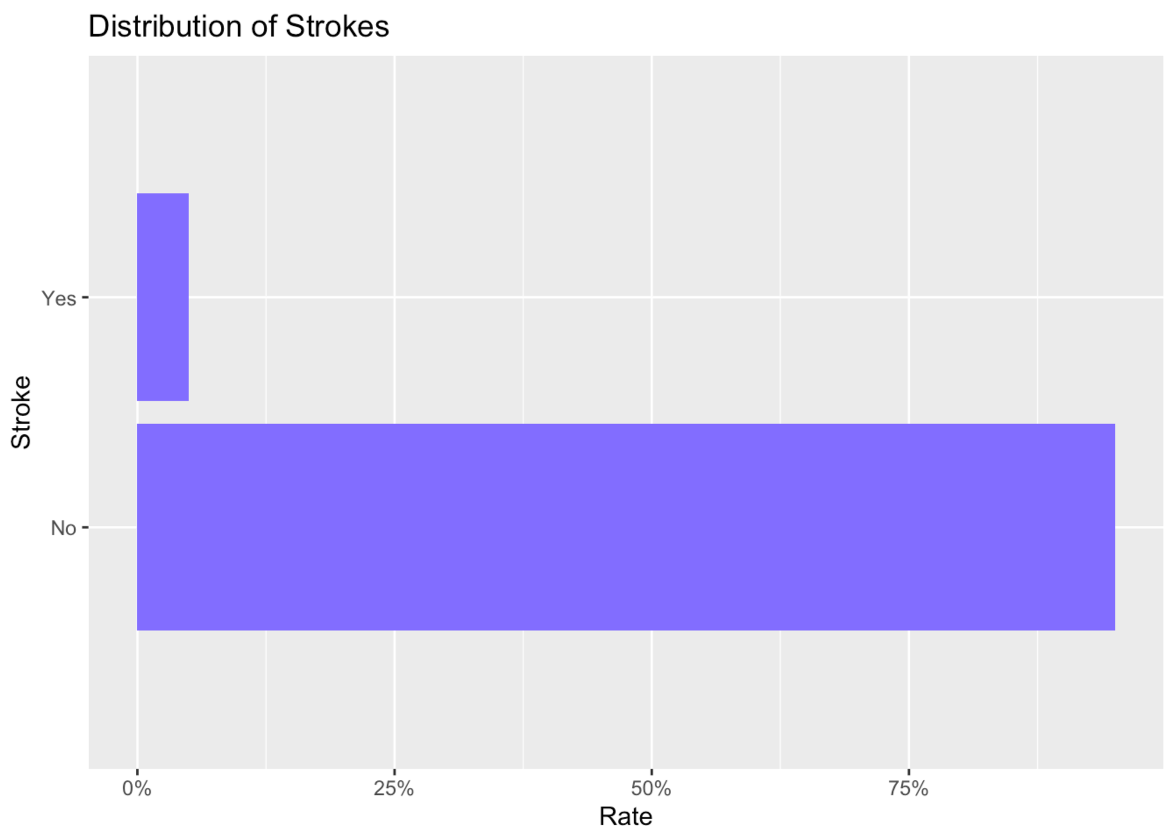
| BMI Category | BMI Value |
|---|---|
| Underweight | <20 |
| Normal Weight | 20-25 |
| Overweight | 25-30 |
| Obese | >30 |

The BMI distribution of the dataset is the following:



We can see that the majority of the patients are overweight (25<BMI<30) or obese (BMI>30). This confirms my suspicion that the dataset is from a high-income country.

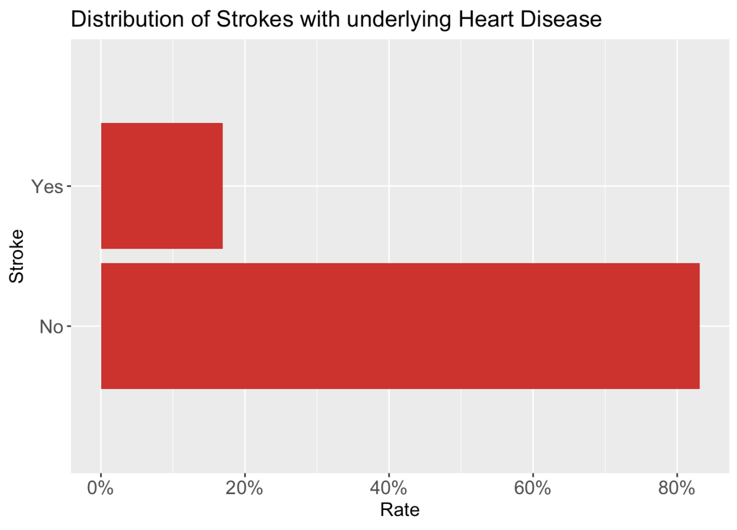
## 2.7 Distribution of Strokes

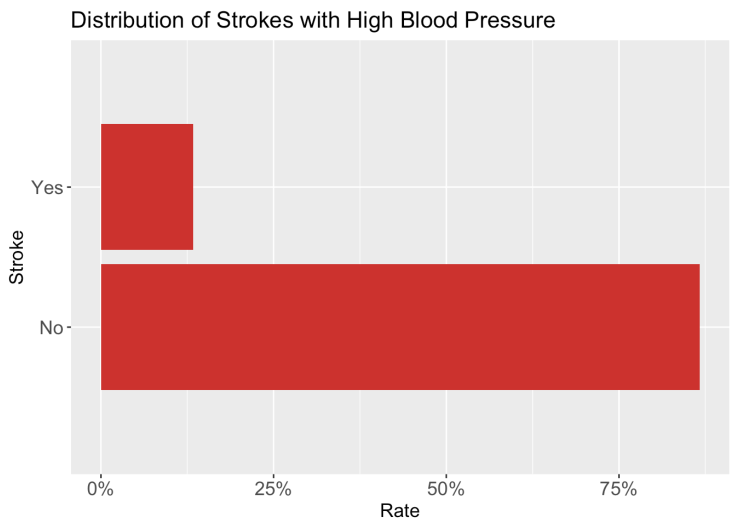


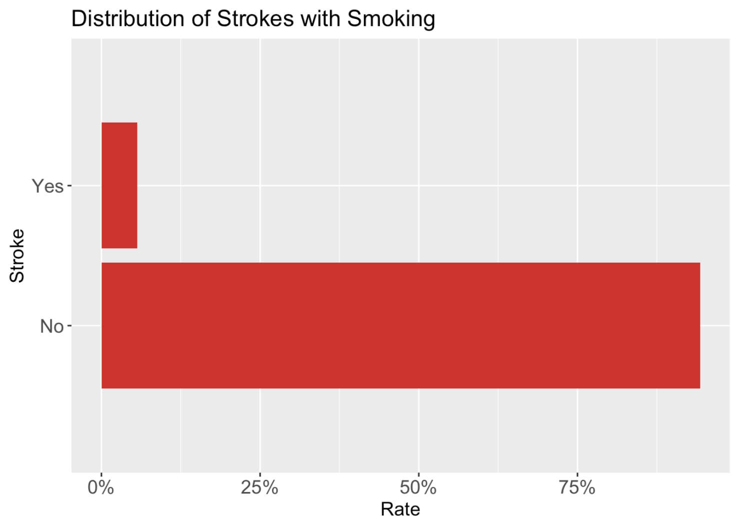
From the plot above we can see that the percentage of the patients who had stroke is 5%. But this is only a summary value. We know from several reports that the heart disease, high blood pressure and smoking are increase the probability of stroke. To get a clearer picture, let us see how these factors affect the distribution of stroke.

## 2.8 Effects of Heart disease, High Blood Pressure and Smoking on Stroke

The underlying heart diseases, high blood pressure and smoking increase the probability of having stroke. These factors are mentioned in all reports on stroke as high-risk factors. The charts below show the percentage of patients with one of these factors had stroke.



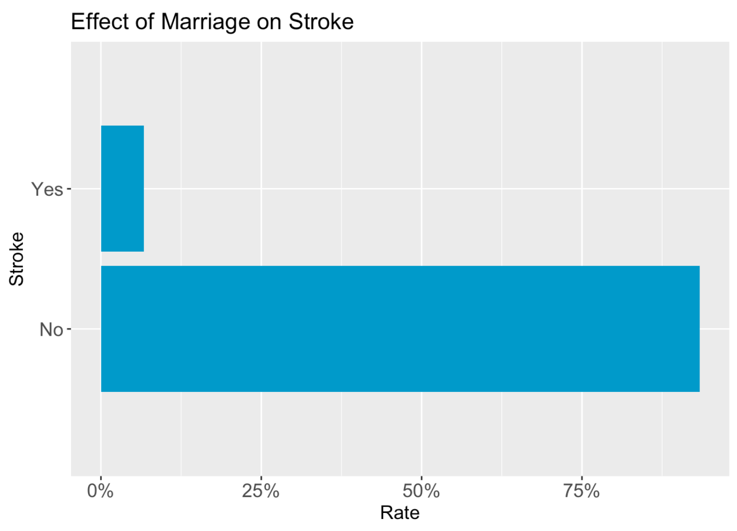


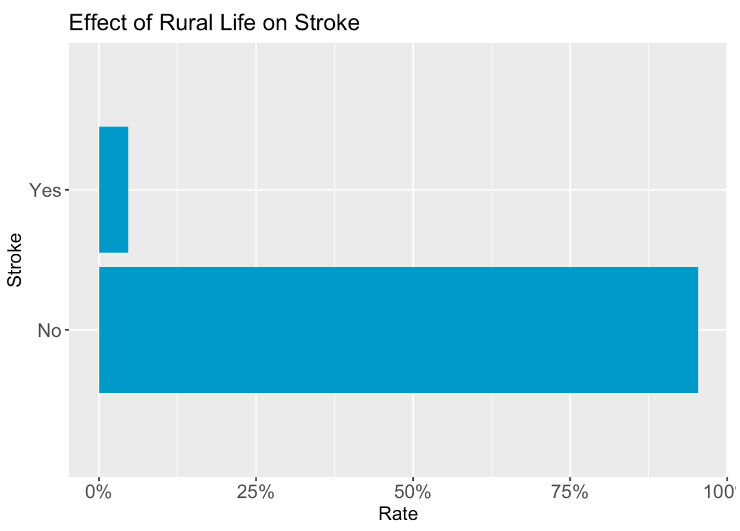


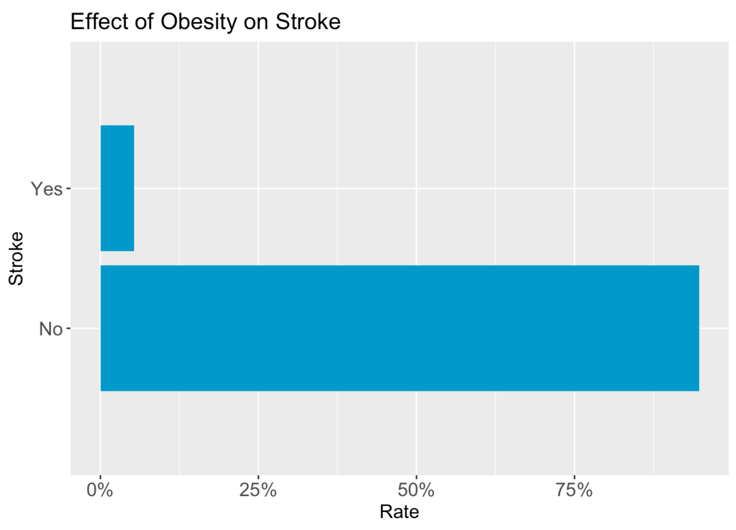
We can see from the charts above how badly these three factors affect our chance to suffer stroke.
Heart disease has the greatest impact, it tripled the chance, increased it to 16.9%.
High blood pressure also has a significant effect, patients with high blood pressure have double chance to have stroke as the average. It increased the rate to 13.3%.
Smoking does not have a significant effect in our dataset however it causes cancer, heart disease, lung diseases, diabetes, chronic obstructive pulmonary disease (COPD) and also increases the chance of stroke. It increased the rate to 5.7%.

## 2.9 Effects of Lifestyle on Stroke

Let us see how the patients' lifestyle affect the probability of stroke. The effect of smoking was mentioned in the previous paragraph, here I analyzed only a few other aspects of their lifestyle. The below charts show effect of the marriage, rural life and the obesity (high BMI values).

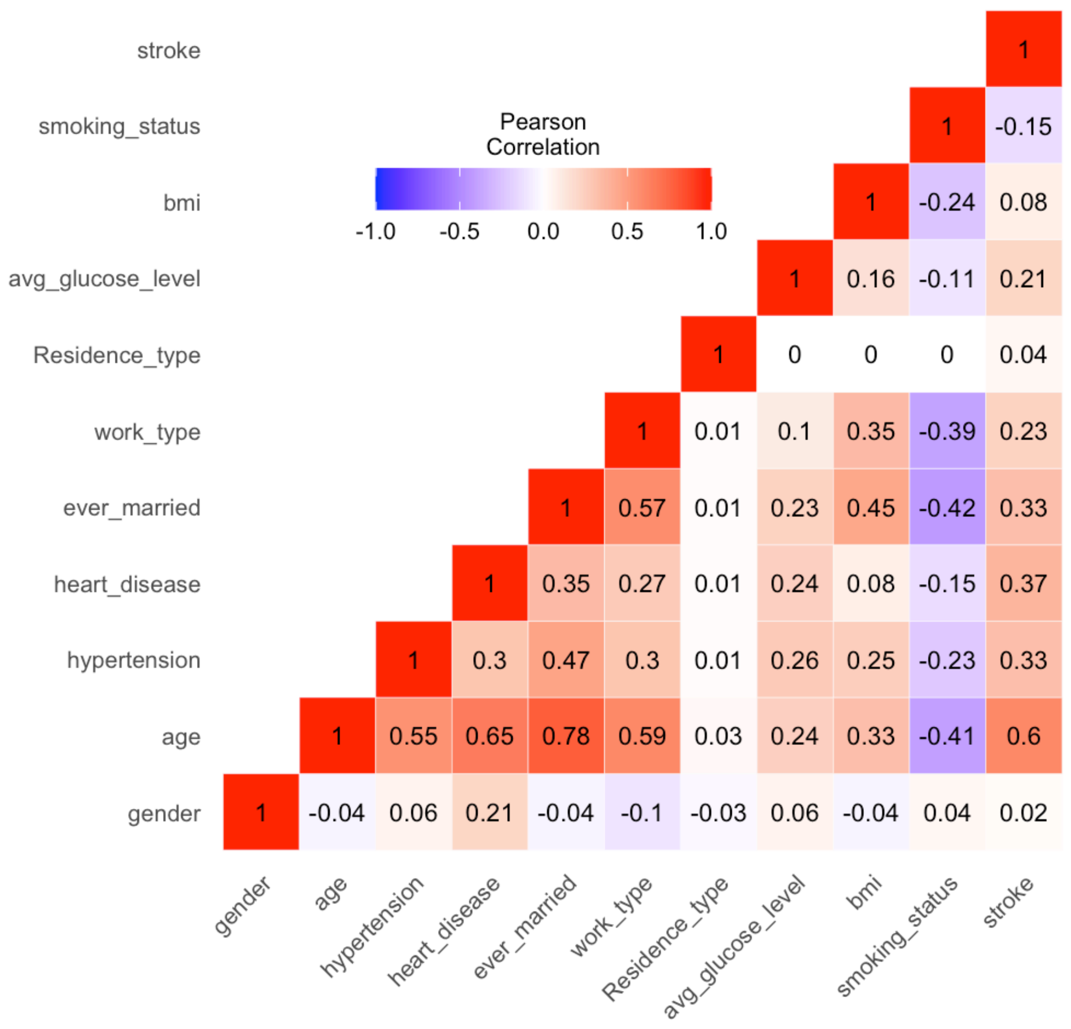






We can see that the probability of stroke among married patients is higher than the average 6.7%. Maybe married people live a more stressful life.
The obesity is increased the stroke chances as we expected. Knowing that obese people live a sedentary lifestyle which often leads to cardiovascular problems and high blood pressure, the result is not surprising. The rate is 5.3%.
Rural life, living in a calm environment has a slight positive effect on our health as the chart shows above. The probability of having stroke among rural people is a bit lower than the average, 4.6%.

## 2.10 Correlation Matrix



|  | gender | age | hypertension | heart_disease | ever_married | work_type | Residence_type | avg_glucose_level | bmi | smoking_status | stroke |
|---|---|---|---|---|---|---|---|---|---|---|---|
| stroke |  |  |  |  |  |  |  |  |  |  | 1 |
| smoking_status |  |  |  |  |  |  |  |  |  | 1 | -0.15 |
| bmi |  |  |  |  |  |  |  |  | 1 | -0.24 | 0.08 |
| avg_glucose_level |  |  |  |  |  |  |  | 1 | 0.16 | -0.11 | 0.21 |
| Residence_type |  |  |  |  |  |  | 1 | 0 | 0 | 0 | 0.04 |
| work_type |  |  |  |  |  | 1 | 0.01 | 0.1 | 0.35 | -0.39 | 0.23 |
| ever_married |  |  |  |  | 1 | 0.57 | 0.01 | 0.23 | 0.45 | -0.42 | 0.33 |
| heart_disease |  |  |  | 1 | 0.35 | 0.27 | 0.01 | 0.24 | 0.08 | -0.15 | 0.37 |
| hypertension |  |  | 1 | 0.3 | 0.47 | 0.3 | 0.01 | 0.26 | 0.25 | -0.23 | 0.33 |
| age |  | 1 | 0.55 | 0.65 | 0.78 | 0.59 | 0.03 | 0.24 | 0.33 | -0.41 | 0.6 |
| gender | 1 | -0.04 | 0.06 | 0.21 | -0.04 | -0.1 | -0.03 | 0.06 | -0.04 | 0.04 | 0.02 |

The above plot shows us that the majority of the correlations between the outcome (*stroke*) and the features are moderate or weak. It has the strongest correlations with *age* and *heart_disease*. On the other hand, the correlation is weak, almost 0 with the *gender* and *residence_type*. From the aspect of machine learning we wouldn't lose much information if we removed these two features. The outcome has negative correlation with *smoking_status*.

# 3. Preprocessing

In machine learning, we must examine the predictors before running the machine algorithm. It is often needed to transform the predictors for some reason. We also remove predictors that are clearly not useful.

## 3.1 Converting and Removing Features

If we take a closer look on the dataset, we can see that some features are not useful in their default class, some others are not useful at all. We must examine the features one by one and handle them in different ways.

- The *id* column contains a unique identifier for each patient, so it is not useful in our models. I removed it from the dataset.

- As we saw in paragraph 2.3, most of the features of the dataset are character type. This type of features is not useful when we use machine learning algorithms, because not all of these algorithms accept this class of data as an input. In our case these are the *hypertension*, *heart_disease*, *bmi*, *stroke*, *gender*, *ever_married*, *Residence_type*, *work_type* and *smoking_status* columns. They were converted to factors.

## 3.2 Examination of the variability of the predictors

The near zero variance predictors are clearly useless in machine learning, because they are nearly or entirely constant. To identify these predictors I used the *nearZeroVar()* function. This function accepts data frames with all numeric data, so I removed the factor type columns before running the function. The function does not recommend any feature to be removed, so I left all features in the dataset.

# 4. Addressing the Prevalence

As we saw in paragraph 2.4, only 5% of the patients have stroke in our dataset, which means the prevalence of patients who do not have stroke is very high. In other words, we have a high imbalance between the observed classes. This imbalance can have a significant negative impact on model fitting. If we train an algorithm on a highly imbalanced train set, the overall accuracy of the model will be very high, but the sensitivity or the specificity could be very low, or even 0. In our case, if a a patient has no stroke, the algorithm will likely to predict correctly. But if the patient has stroke, the algorithm will very likely to fail. During the optimization of the algorithms I focused on accuracy, sensitivity, specificity and F1 score to build models with balanced performance. One technique for resolving such a class imbalance is to subsample the train set in a manner that mitigates the issues. I chose the up-sampling technique, because the the dataset is small. The *caret* package contains a function (*upSample*) to do this. The function randomly samples (with replacement) the minority class (*stroke* =1) to be the same size as the majority class (*stroke*=0).
I divided the dataset to a train set (called *stroke*), and a validation set (called *validation*). I up-sampled the *stroke* set to mitigate the imbalance.

Structure of the Balanced Train Set

| Stroke | Number |
|---|---:|
| 0 | 3883 |
| 1 | 3883 |

The following machine algorithms were trained on the new up-sampled train set (called *balanced_stroke*), and they were tested on the validation set.

# 5. Models

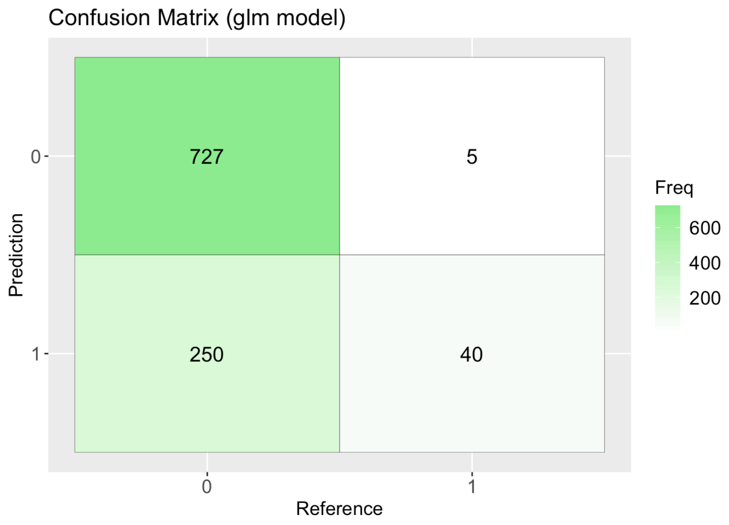
In this project I had to predict a binary (classified) outcome, so I tried the 5 of the most popular classification algorithms. These are the *glm*, *LDA*, *kNN*, *rpart* and the *randomForest* algorithms. Each algorithm based on different ideas and concepts, I compared them to find out which is the most eligible in our case.

## 5.1 Logistic Regression model (glm)

Logistic regression is a model that makes predictions using a logistic function to find the dependency between the output and input variables. It is a calculation used to predict a binary outcome, so it is a very good choice for this project. The *glm* function has no tuning parameters.

The results are the following:

The overall accuracy I achieved on the validation set with this simple model is 0.7505. This is not bad, considering the high prevalence of the dataset. All performance measures balanced. It would be much worse if the train set wasn't balanced.



Performance Metrics

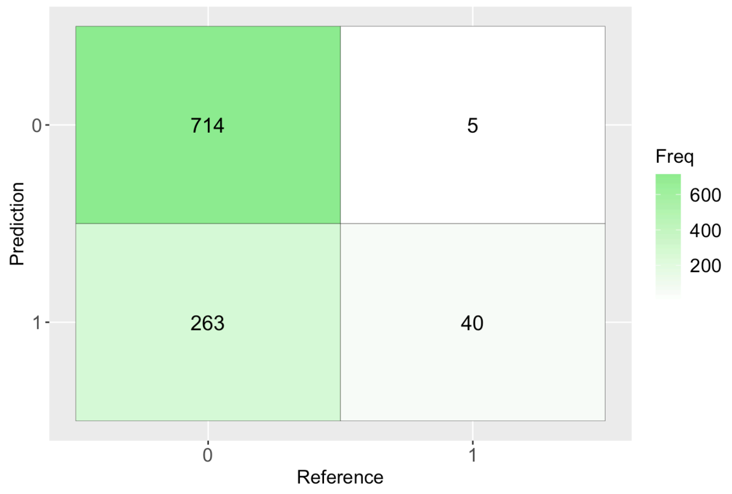
| | glm |
|---|---|
| Accuracy | 0.7505 |
| Sensitivity | 0.7441 |
| Specificity | 0.8889 |
| F1 | 0.8508 |

## 5.2 LDA model

Linear Discriminant Analysis (LDA) is a well-established machine learning technique and classification method for predicting categories. Its main advantages, compared to other classification algorithms such as neural networks and random forests, are that the model is interpretable and that prediction is easy. LDA is used to predict the probability of belonging to a given class (or category) based on one or multiple predictor variables. It works with continuous and/or categorical predictor variables. It uses linear combinations of predictors to predict the class of a given observation.

The overall accuracy, sensitivity and F1 score I achieved is a bit lower than those of the glm model. The specificity is 0.8889 exactly the same as that of the first model. The performance of the model is well balanced, however a bit worse than the first algorithm.

Confusion Matrix (LDA model)

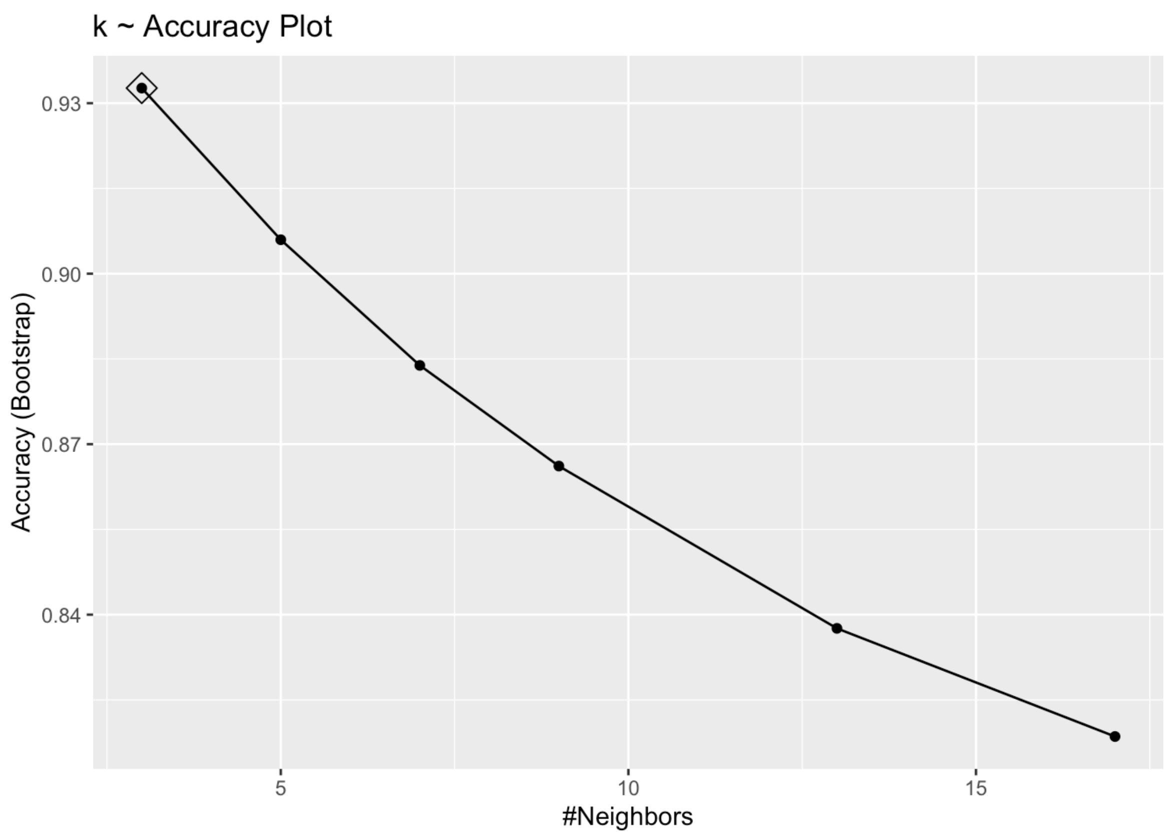


Performance Metrics

| | glm | LDA |
|---|---|---|
| Accuracy | 0.7505 | 0.7378 |
| Sensitivity | 0.7441 | 0.7308 |
| Specificity | 0.8889 | 0.8889 |
| F1 | 0.8508 | 0.8420 |

## 5.3 kNN model

K-nearest neighbors (kNN) is a pattern recognition algorithm that uses training datasets to find the k closest relatives in future examples. When kNN is used in classification, we calculate to place data within the category of its nearest neighbor. I used the *train* function from the *caret* package to train the algorithm. By default, the *train* function performs cross validation by taking 25 bootstrap samples comprised of 25% of the observations. For the kNN algorithm the tuning parameter is *k*. The default of the *train* function is to try *k*= 5, 7, 9. I tried it on a bit wider scale: *k*= 3, 5, 7, 9, 13, 17.

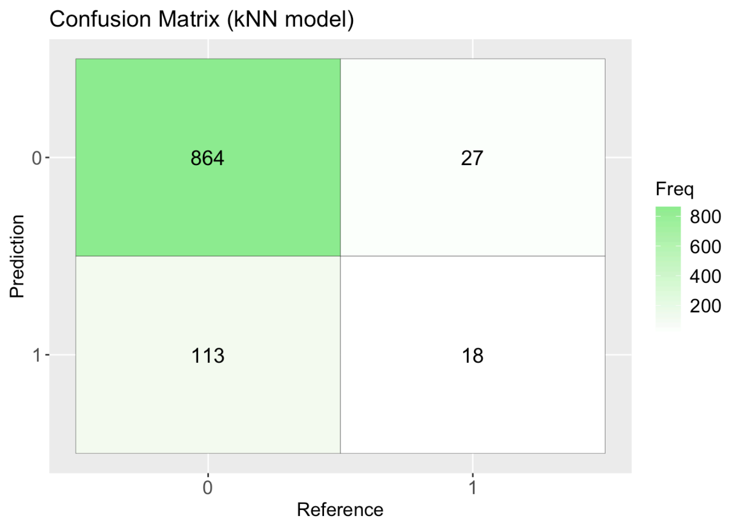


We can see that the *k* value that gives the best accuracy is

| | **k** |
|---|---|
| | <dbl> |
| 1 | 3 |

1 row

. This model has a very good accuracy, the F1 score is also excellent, but the specificity is much worse than that of the first two models. It is only 0.4. Unfortunately the performance is not balanced, it fails to predict stroke too often.

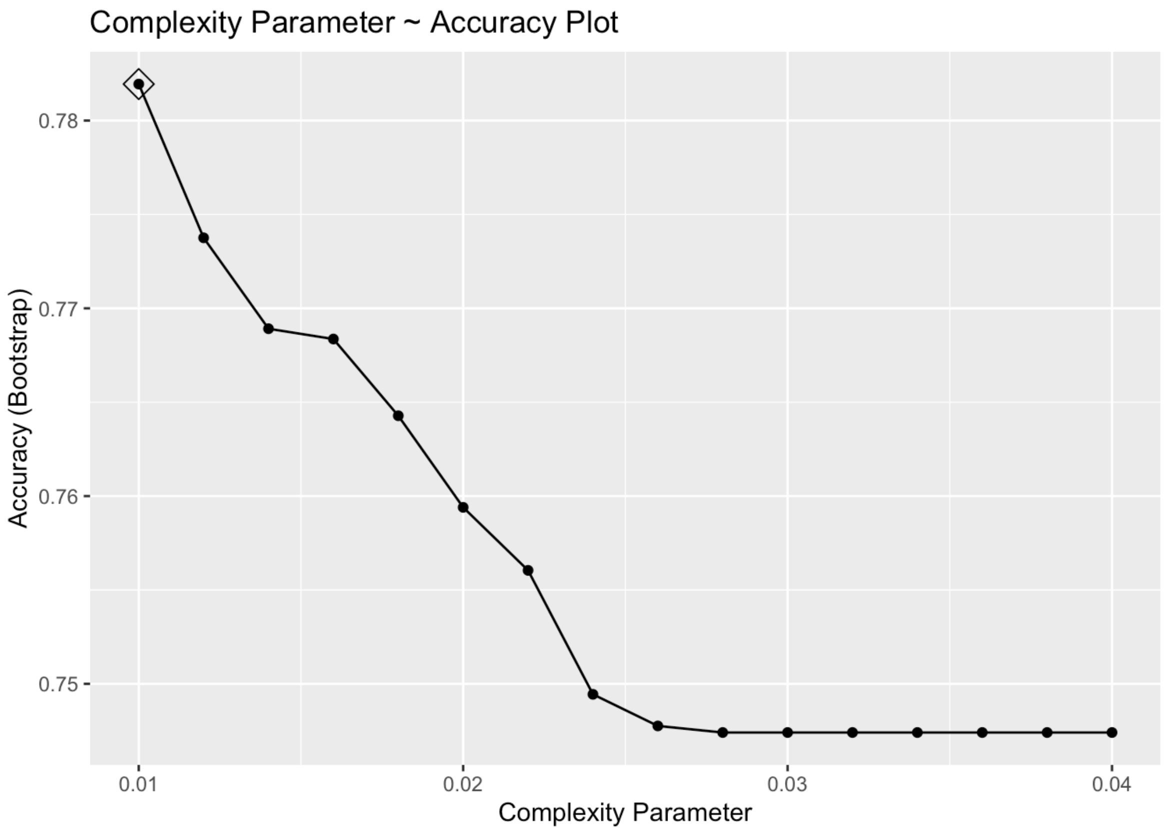


Performance Metrics

| | glm | LDA | kNN |
|---|---|---|---|
| Accuracy | 0.7505 | 0.7378 | 0.8630 |
| Sensitivity | 0.7441 | 0.7308 | 0.8843 |
| Specificity | 0.8889 | 0.8889 | 0.4000 |
| F1 | 0.8508 | 0.8420 | 0.9251 |

## 5.4 CART model (rpart)

Classification and regression tree (CART) is a type of supervised learning algorithm that can be used in both regression and classification problems. It works for both categorical and continuous input and output variables. They are very powerful algorithms, capable of fitting complex datasets.
The major advantage of using decision trees is that they are intuitively very easy to explain. They closely mirror human decision-making compared to other regression and classification approaches. They can be displayed graphically.
Besides, decision trees are fundamental components of random forests, which are among the most potent Machine Learning algorithms available today. I used the complexity parameter (*cp*) to tune the algorithm, I tried it between 0.01 and 0.04.



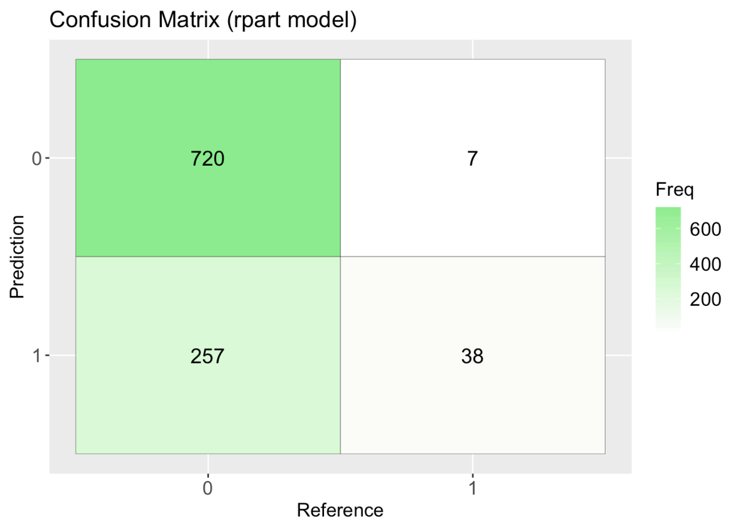
By further reducing the *cp* value we could increase the accuracy of the algorithm, but on the other hand we could loose it’s balance. With *cp*=

| cp<br>&lt;dbl&gt; |
|---|

| | |
|---|---|
| 1 | 0.01 |

1 row

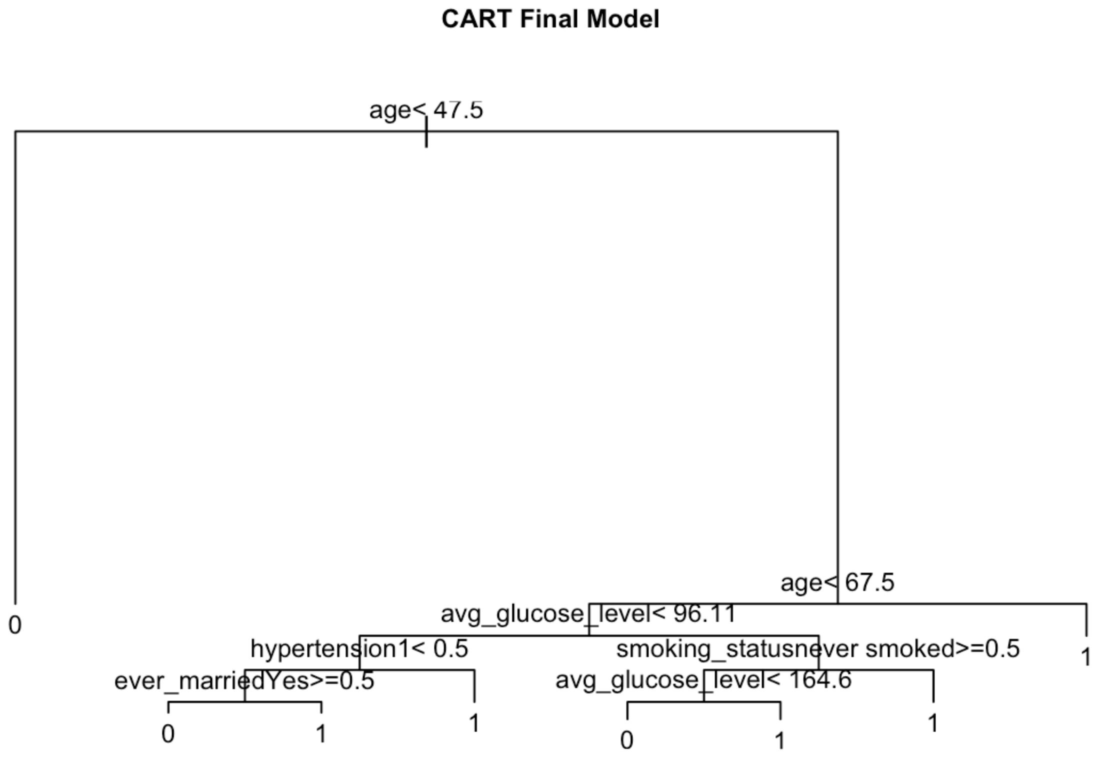
, we have a well balanced model, with a performance somewhere between the first two models'.



Performance Metrics

| | glm | LDA | kNN | rpart |
|---|---|---|---|---|
| Accuracy | 0.7505 | 0.7378 | 0.8630 | 0.7417 |
| Sensitivity | 0.7441 | 0.7308 | 0.8843 | 0.7369 |
| Specificity | 0.8889 | 0.8889 | 0.4000 | 0.8444 |
| F1 | 0.8508 | 0.8420 | 0.9251 | 0.8451 |

The final decision tree is the following:



## 5.5 Random Forest model

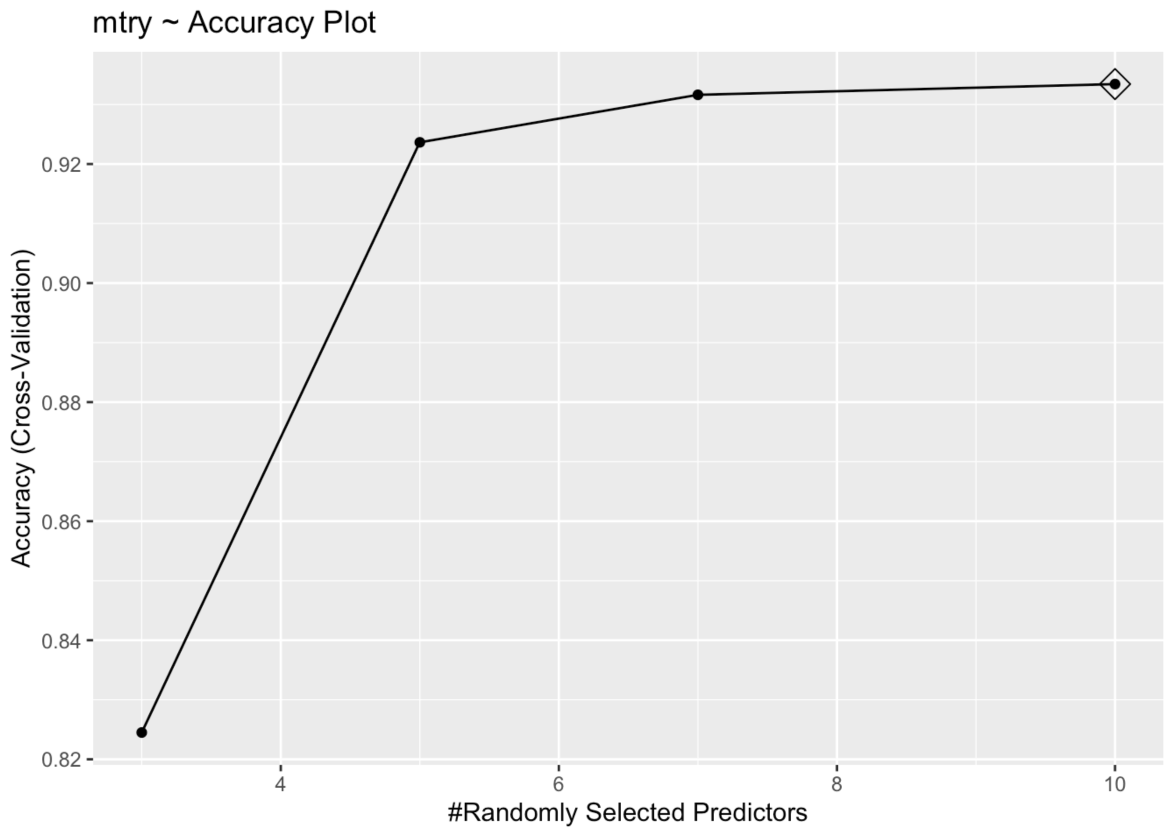
The random forest algorithm is an expansion of decision tree. A decision tree is a learning algorithm that is perfect for classification problems, as it is able to order classes on a precise level. It works like a flow chart, separating data points into two similar categories at a time from the “tree trunk” to “branches,” to “leaves,” where the categories become more finitely similar. This creates categories within categories, allowing for organic classification with limited human supervision.
The randomForest improves prediction performance of the decision tree by averaging multiple decision trees (a forest of trees constructed with randomness).
To increase accuracy, I used 5-fold cross validation, and tried the tuning parameter *mtry* values 1, 5, 10, 25, 50. To mitigate the imbalance of the dataset I changed the cutoff for classification from the default 50-50% to 80-20%.

As we can see in the following plot the optimal *mtry* is

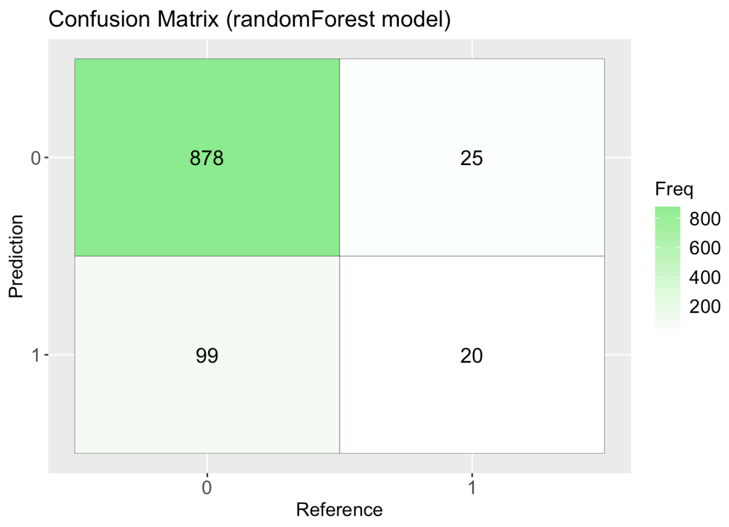
| | **mtry**<br><dbl> |
|---|---|
| 4 | 10 |

1 row

. It means we get the highest accuracy if all predictors selected in each tree.



This model's performance is close to the kNN's. All performance metrics are very good except the specificity which less than the half of the others.



Performance Metrics

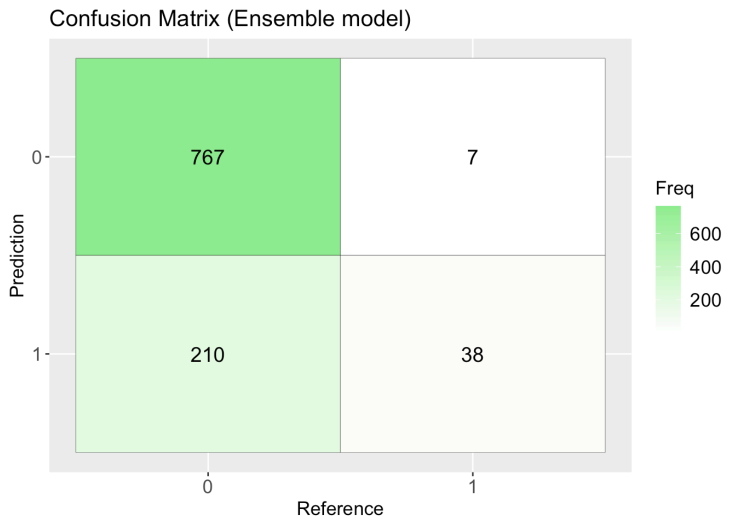
| | glm | LDA | kNN | rpart | randomForest |
|---|---|---|---|---|---|
| Accuracy | 0.7505 | 0.7378 | 0.8630 | 0.7417 | 0.8787 |
| Sensitivity | 0.7441 | 0.7308 | 0.8843 | 0.7369 | 0.8987 |
| Specificity | 0.8889 | 0.8889 | 0.4000 | 0.8444 | 0.4444 |

| | | | | | |
|---|---|---|---|---|---|
| F1 | 0.8508 | 0.8420 | 0.9251 | 0.8451 | 0.9340 |

## 5.6 Ensemble model

In general, ensembling is a technique of combining two or more algorithms of similar or dissimilar types called base learners. This is done to make a more robust system which incorporates the predictions from all the base learners. By using this technique, we can take all of these predictions into account while making the final decision. This will make our final decision more robust, accurate and less likely to be biased. There are different types of ensembling. In this project I took the prediction with maximum vote from all model's predictions as the final prediction.

As the figures show below this is the most balanced and most robust method. It has a bit higher accuracy, sensitivity and F1 score than the *glm*, *LDA* and *rpart* models. Besides, from the aspect of specificity it much better than the *kNN* and *randomForest* models.



Performance Metrics

| | glm | LDA | kNN | rpart | randomForest | Ensemble |
|---|---|---|---|---|---|---|
| Accuracy | 0.7505 | 0.7378 | 0.8630 | 0.7417 | 0.8787 | 0.7877 |
| Sensitivity | 0.7441 | 0.7308 | 0.8843 | 0.7369 | 0.8987 | 0.7851 |
| Specificity | 0.8889 | 0.8889 | 0.4000 | 0.8444 | 0.4444 | 0.8444 |
| F1 | 0.8508 | 0.8420 | 0.9251 | 0.8451 | 0.9340 | 0.8761 |

# 6. Summary

The goal of this project was to build a prediction system based on the Stroke Prediction dataset, which predict whether a patient is likely to get stroke or not. Because of the dataset is highly imbalanced, the goal during the training of the algorithms was to build an algorithm with balanced performance metrics. I used the accuracy, sensitivity, specificity and the F1 score to evaluate the models. Predicting the minority (*stroke*=1) and the majority class (*stroke*=0) accurately were considered equally important.

There are several classifiers available in R, I chose the *glm*, *LDA*, *rpart*, *kNN* and *randomForest* models to use in this project.
The simple methods were more eligible to achieve the goal. The *glm*, *LDA* and *rpart* models' performance were more balanced, besides their overall accuracy were good too. I was a bit disappointed at the performance of the *kNN* and *randomForest* models. I expected that these complex and computationally expensive models will outperform the simple ones. That is not what happened. However I achieved better accuracy with them, but on the other hand their specificity were very low. This means they are very good at predicting majority class (*stroke*=0), but they often fail to predict the minority class (*stroke*=1) accurately.
Finally, I built an ensemble model which includes all the models using majority voting. I achieved the most balanced performance metrics with this final model, all of them around 80%. This is clearly the best model built in this project.

# 7. Future Considerations

However I achieved pretty good results with the methods I used in this project, there are many other ways that worth to try if we would like to further increase performance in the future. For example we could try the following techniques:

- **Preprocessing**: I used the up-sampling method to mitigate the class imbalance of the training set. But there are other ways to threat imbalanced datasets such as Synthetic Data Generation. The *ROSE* function instead of replicating and adding the observations from the minority class, it overcomes imbalances by generating artificial data. It uses bootstrapping to generate artificial data. This method could achieve higher accuracy as compared to sampling methods.
- **Models**: I used 5 classification models in this project, but there are several other models that we could try. For example the *Naive Bayes*, *SVM*, and *QDA* models are very good and very popular classifiers. All of them based on different ideas.
- **Training**: For training of the algorithms I used the *train* function of the *caret* package. By default, the *train* function maximizes the accuracy for classification. Changing the optimality criterion to Cohen's Kappa can improve the quality of the final model, when we are dealing with a highly imbalanced datatset.
  If it does not improve our models, the *train* function allows us to create our own optimalty criterion.
- **Ensembling**: There are different types of ensembling. In this project I used majority voting, but we could try other ensemling techniques:
  - By using weighted voting we could give higher weightage to the votes of one or more best performing models. We could also penalize the weak ones.
  - Boosting is one of the ensemble learning techniques in machine learning and it is widely used in regression and classification problems. The main concept of this method is to improve (boost) the weak learners sequentially and increase the model accuracy with a combined model. There are several boosting algorithms such as *GBM* (Gradient boosting), *AdaBoost* (Adaptive Boost), *XGBoost* and others.