DATA ANALYTICS

Project Report

on

Heart Stroke Prediction

Professor: Gokhan Egilmez

Team Saffola Gold

Project by:

Adarsh Patel

Siva Chandan Chakka

Sun Gajiwala

Contents:

1. Abstract

- Heart Strokes have increased rapidly over the past few years for adults and for the young ones. Stroke prediction is complex and there is a need to automate this process.
- The project aims at proposing a heart stroke prediction model based on a dataset acquired from Kaggle.
- The risk of getting a heart stroke will be classified using various models like Random Forest, Logistic Regression and K-Nearest Neighbor (KNN).
- A comparative analysis will be presented to identify the most effective model.

2. Problem statement

- Due to increasing heart strokes during the past few years, there is a need to create a model that can predict whether a person is at a risk of having a heart stroke.
- Our project aims for the same by using some heart related and some general features to train models and using the same features as input to predict heart stroke.

Research questions

- How can we effectively predict when a person is going to suffer from a heart stroke?
- Which feature has the most effect on a person getting heart stroke?

3. Exploratory Data Analysis

3.1 Understanding the Dataset

The dataset contains 12 features of 5110 patients. The features contain the patients (id, gender, age, heart diseases, work type, tension, ever_married, residence_type, avg. glucose level, BMI, smoking status and stroke).

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5110 non-null   int64
 1   gender             5110 non-null   object
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64
 4   heart_disease      5110 non-null   int64
 5   ever_married       5110 non-null   object
 6   work_type          5110 non-null   object
 7   Residence_type     5110 non-null   object
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object
 11  stroke             5110 non-null   int64
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
An overview of the DataFrame None
```

- Here from the screenshot of the R - console output we can clearly see that each variable (column) types have data entry like:
  - Gender, ever_maried, work_type, Residence_type, Smoking_status is having variable type as text or object.
  - Age, avg_glucose_level, and bmi(body mass index) is having variable type as float.
  - Stroke variable has a data entry as a binary or index type.

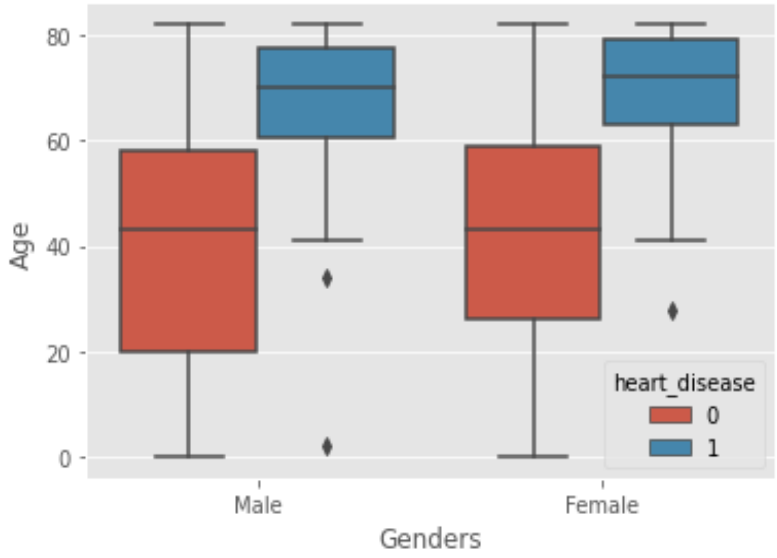
3.2. A statistical overview of the dataset

- Statical overview of the data set is basically calculating the mean, standard deviation and interquartile of all the features. This can be obtained from the R – console and the following are the screenshot of it.

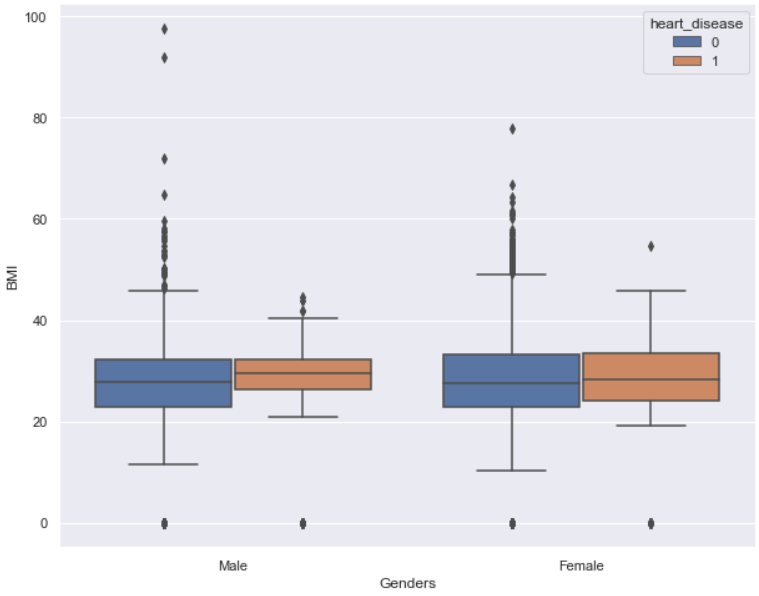
	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.000000
mean	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	22.612647	0.296607	0.226063	45.283560	7.854067	0.215320
min	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	25.000000	0.000000	0.000000	77.245000	23.500000	0.000000
50%	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	61.000000	0.000000	0.000000	114.090000	33.100000	0.000000
max	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

3.3 Understanding the Distribution of the features

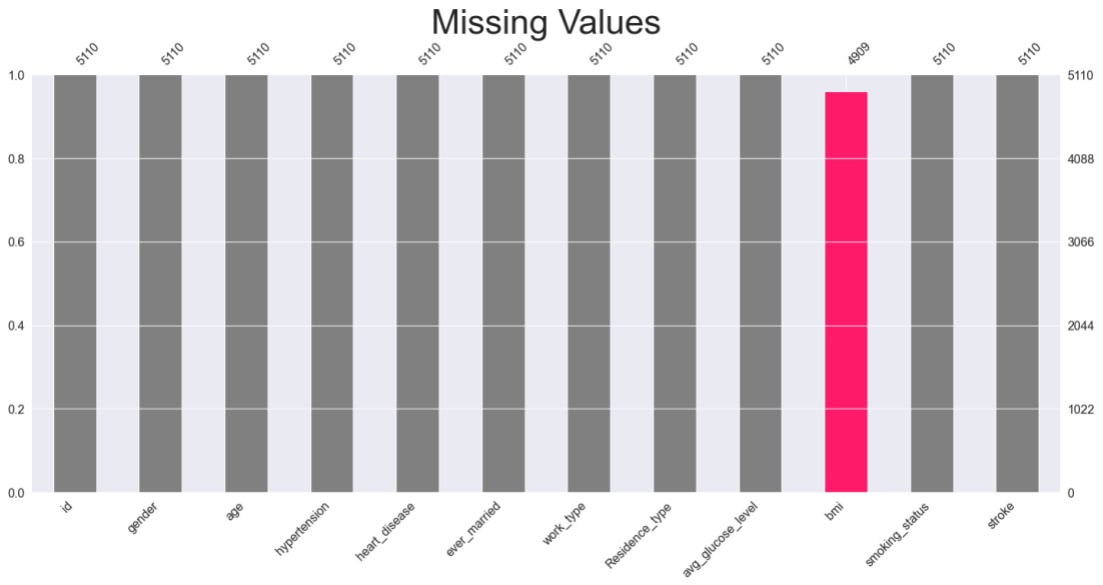
- Male and Female suffering from heart diseases according to age.
- We can see that people above age 60 suffered most from heart diseases.



- Male and Female suffering from heart diseases according to Body Mass Index.
- We can see that people below the BMI of 40 suffered most from heart diseases.



3.4 Identifying the Missing Values



- As we can see the missing values are mostly present in BMI column.

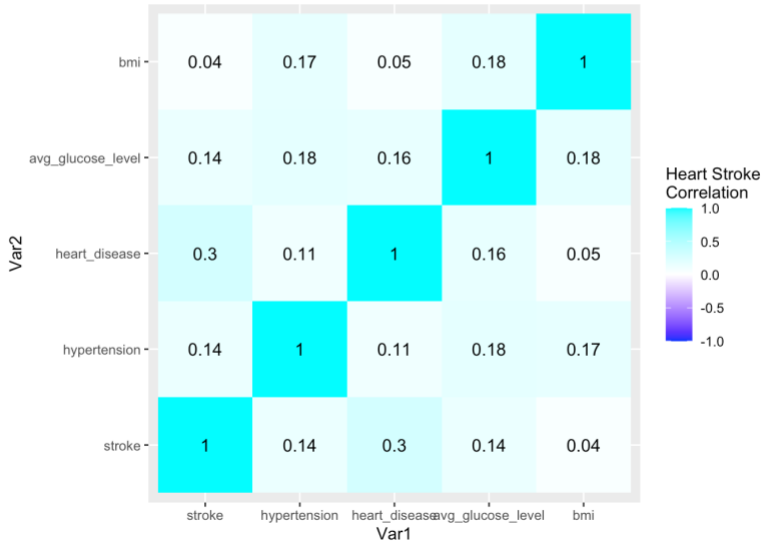
3.4.1 Dealing with the Missing Values

```
library(naniar)

# Removing the Id column as it is not relevant for our analysis.
d<-d[,-1]

# We need to check for the null values in our Dataset.
# Since we have N/A values in our BMI column, lets remove the rows with N/A values in them.

d<-d[!d$bmi == "N/A", ]
d$bmi<-as.integer(d$bmi)
```

3.5 Correlation Analysis

- Having a look at Heat map of Correlation matrix of the features will help us in selecting the Model

```
# We use the function geom_title to visualize the correlation matrix
library(ggplot2)
ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()+
  geom_text(aes(Var2, Var1, label = value), color = "black", size = 4)+
  scale_fill_gradient2(low = "blue", high = "cyan", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Heart Stroke\nCorrelation")
```



- From the above heat map, we can say that the features are reasonably independent with each other.

4.Methods (Models, Experiments, Analysis) and Results

4.1 Naive Bayes:

Why Naive Bayes?

As we have seen in the correlation matrix the features seem independent with each other and the problem in our hand is binary classification. Generally Naive Bayes works better than Logistic Regression.

What is Naïve Bayes?

Naive Bayes is a simple, yet effective and commonly used, machine learning classifier. It is a probabilistic classifier that makes classifications using the Maximum A Posteriori decision rule in a Bayesian setting. It can also be represented using a very simple Bayesian network. Naive Bayes classifiers have been especially popular for text classification and are a traditional solution for problems such as spam detection.

The Model

The goal of any probabilistic classifier is, with features x_0 through x_n and classes c_0 through c_k, to determine the probability of the features occurring in each class, and to return the most likely class. Therefore, for each class, we want to be able to calculate P(c_i | x_0, ..., x_n).

To do this, we use **Bayes rule**. Recall that Bayes rule is the following:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

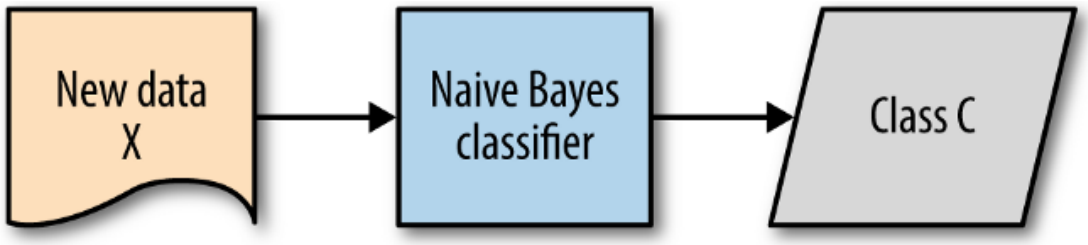
In the context of classification, you can replace A with a class, c_i, and B with our set of features, x_0 through x_n. Since P(B) serves as normalization, and we are

usually unable to calculate P(x_0, ..., x_n), we can simply ignore that term, and instead just state that P(c_i | x_0, ..., x_n) ∝ P(x_0, ..., x_n | c_i) * P(c_i), where ∝ means "is proportional to". P(c_i) is simple to calculate; it is just the proportion of the data-set that falls in class i. P(x_0, ..., x_n | c_i) is more difficult to compute. In order to simplify its computation, we make the assumption that x_0 through x_n are **conditionally independent** given c_i, which allows us to say that P(x_0, ..., x_n | c_i) = P(x_0 | c_i) * P(x_1 | c_i) * ... * P(x_n | c_i). This assumption is most likely not true — hence the name *naive* Bayes classifier, but the classifier nonetheless performs well in most situations. Therefore, our final representation of class probability is the following:

$$\begin{aligned} P(c_i|x_0,\ldots,x_n) &\propto P(x_0,\ldots,x_n|c_i)P(c_i) \\ &\propto P(c_i)\prod_{j=1}^{n} P(x_j|c_i) \end{aligned}$$

Calculating the individual P(x_j | c_i) terms will depend on what distribution your features follow. In the context of text classification, where features may be word counts, features may follow a **multinomial distribution**. In other cases, where features are continuous, they may follow a **Gaussian distribution**.

Note that there is very little explicit training in Naive Bayes compared to other common classification methods. The only work that must be done before prediction is finding the parameters for the features' individual probability distributions, which can typically be done quickly and deterministically. This means that Naive Bayes classifiers can perform well even with high-dimensional data points and/or a large number of data points.



Classification

Now that we have a way to estimate the probability of a given data point falling in a certain class, we need to be able to use this to produce classifications. Naive Bayes handles this in a very simple manner; simply pick the c_i that has the largest probability given the data point's features.

$$y = \underset{c_i}{argmax}\ P(c_i) \prod_{j=1}^{n} P(x_j|c_i)$$

This is referred to as the **Maximum A Posteriori** decision rule. This is because, referring back to our formulation of Bayes rule, we only use the P(B|A) and P(A) terms, which are the likelihood and prior terms, respectively. If we only used P(B|A), the likelihood, we would be using a **Maximum Likelihood** decision rule.

4.2. Data preprocessing

4.2.1 Converting the features into Categorical Values

- Naïve Bayes works only if the features are categorical and hence, we will be converting the features which are not categorical into categorical features.

```
# Since we plan on using the Naive Bayes classifier, we need to convert our numerical variables into ca
tegorical variables.
# The outcome variable is binary [0,1] which means yes or no.
data <- d
d$stroke <- ifelse(d$stroke == 1,"Y","N")
class(d$stroke)
```

```
## [1] "character"
```

```
stroke.count = sum(d$stroke == "Y")
stroke.count
```

```
## [1] 209
```

```
percent.stroke = stroke.count/dim(d)[1]
percent.stroke * 100
```

```
## [1] 4.257486
```

```
# Converting Numerical to Categorical for naive Bayes Classification

d$avg_glucose_level <- round(d$avg_glucose_level/10)
d$bmi               <- round(d$bmi/10)
str(d)
```

```
## 'data.frame':    4909 obs. of  11 variables:
##  $ gender           : chr  "Male" "Male" "Female" "Female" ...
##  $ age              : num  67 80 49 79 81 74 69 78 81 61 ...
##  $ hypertension     : int  0 0 0 1 0 1 0 0 1 0 ...
##  $ heart_disease    : int  1 1 0 0 0 1 0 0 0 1 ...
##  $ ever_married     : chr  "Yes" "Yes" "Yes" "Yes" ...
##  $ work_type        : chr  "Private" "Private" "Private" "Self-employed" ...
##  $ Residence_type   : chr  "Urban" "Rural" "Urban" "Rural" ...
##  $ avg_glucose_level: num  23 11 17 17 19 7 9 6 8 12 ...
##  $ bmi              : num  4 3 3 2 3 3 2 2 3 4 ...
##  $ smoking_status   : chr  "formerly smoked" "never smoked" "smokes" "never smoked" ...
##  $ stroke           : chr  "Y" "Y" "Y" "Y" ...
```

```
d$avg_glucose_level <-factor(d$avg_glucose_level)
d$age<-factor(d$age)
d$hypertension      <-factor(d$hypertension)
d$heart_disease     <-factor(d$heart_disease)
d$bmi               <-factor(d$bmi)
```

4.2.2 Splitting the data into Training and Validation Data

- Model performance can be truly measured when testing it on unseen data.
- To achieve this, splitting data into training and validation is a good practice.

Splitting Data for Training and Validation

```
# Setting seed to 1 for reproduciblity of randomness
set.seed(1)
train.rows       <- sample(row.names(d), 0.6*dim(d)[1])
valid.rows       <- setdiff(row.names(d), train.rows)
train.df         <- d[train.rows, ]
valid.df         <- d[valid.rows, ]
View(train.df)
```

4.3 Training the Model

Using the training data, we will Train the Naïve Bayes Model.

Naive Bayes classifier

```
# Since Stroke is our Target column, we divide the dataset according to that

library(e1071)
stroke.nb    <- naiveBayes(stroke ~ ., data = train.df)
stroke.nb
```

4.4 Results

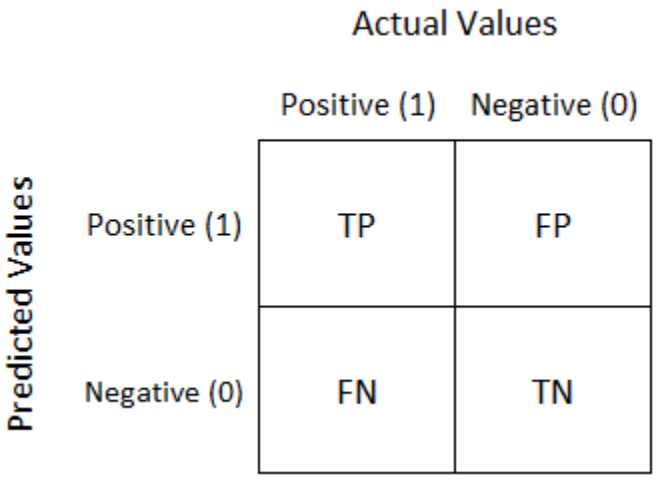
4.4.1 Metrics to be used to evaluate the Model

Accuracy:

Accuracy is most widely used metric when we want to measure the Classification Model.

Confusion Matrix:

Well, it is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.



4.4.2 Performance of the Model on Training Data:

The accuracy score for our training model Is around 93%.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    Y
##          N 2678   70
##          Y  141   56
##
##                Accuracy : 0.9284
##                  95% CI : (0.9184, 0.9374)
##     No Information Rate : 0.9572
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3108
##
##  Mcnemar's Test P-Value : 1.443e-06
##
##             Sensitivity : 0.44444
##             Specificity : 0.94998
##          Pos Pred Value : 0.28426
##          Neg Pred Value : 0.97453
##              Prevalence : 0.04278
##          Detection Rate : 0.01902
##    Detection Prevalence : 0.06689
##       Balanced Accuracy : 0.69721
##
##        'Positive' Class : Y
##
```

4.4.3: Performance of the Model on Validation Data

The accuracy score of the model on our testing data is around 92%.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    Y
##          N 1783   50
##          Y   98   33
##
##                Accuracy : 0.9246
##                  95% CI : (0.9121, 0.9359)
##     No Information Rate : 0.9577
##     P-Value [Acc > NIR] : 1.0000000
##
##                   Kappa : 0.2707
##
##  Mcnemar's Test P-Value : 0.0001118
##
##             Sensitivity : 0.39759
##             Specificity : 0.94790
##          Pos Pred Value : 0.25191
##          Neg Pred Value : 0.97272
##              Prevalence : 0.04226
##          Detection Rate : 0.01680
##    Detection Prevalence : 0.06670
##       Balanced Accuracy : 0.67275
##
##        'Positive' Class : Y
##
```

5. Conclusion

1. The model worked with an accuracy of over 90% and sensitivity of over 30% for both training and validation datasets.

2. 3 out of 8 were predicted to be at a risk of heart stroke from the test dataset.
3. Cleaning the dataset was challenging as there were some anomalies in BMI.
4. Bin of with range 10 as group was created to make the naïve bayes approach more optimal.
5. The number of people having stroke risk is very less in the dataset as compared to the number of people with no risk thereby hampering the machine learning process.

Although the accuracy is over 90% more data on the positive response of strokes could have made the algorithm a lot more accurate than it is.