King Saud University

College of Computer and Information Sciences

Information Technology Department

IT 326 DATA MINING

First semester, 2023

STROKE PREDICITION

section: 1234567890-098765432345678909876 5	
Leader Email: 443200668@student.ksu.edu.sa	
<Bashair Abdullah Alsadhan >	443200668
<Maryam Faisal Altuwaijri>	443200235
<Rama Fahad Alshebel>	443200929

Supervised By: DR. Mashael aldayel

Contents

Problem 3
Data mining task 3
Dataset information 4
Data preprocessing 10
Imbalanced dataset problem: 13
Data mining techniques 14
Classification: 15
Evaluation 16
Gini index 16
IG ratio 22
Information gain 28
Clustering: 34
Finding: 45
Classification: 45
Clustering: 48
Code 50
References 50

Problem

Stroke is the second-leading cause of death and the most common global cause of disability. WHO estimates that 1 in 4 persons may experience a stroke during their lifetime; because strokes can occur at any time and to anyone, regardless of age, we have chosen to concentrate on this dataset. Given the sudden nature of strokes, we intend to investigate and analyze the data to provide predictions on what are some risk factors and shed light on the types of people who are likely to experience one, allowing for future changes in lives. This dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age,bmi, various diseases such as hypertension and heart disease, smoking status, marital status and residence type. Each row in the data provides relavant information about the patient.

Data mining task

Data mining plays a crucial role in predicting the probability of having a stroke through classification and clustering techniques. By applying data mining algorithms to a large dataset containing various health-related features, valuable patterns and relationships can be discovered. In the classification aspect, data mining aids in building models that can accurately classify individuals into different categories, such as 1 for "stroke" or 0 for "non-stroke," based on their attributes and risk factors. This helps in identifying individuals who are more likely to experience a stroke, enabling proactive interventions and preventive measures. On the other hand, clustering techniques assist in identifying groups or clusters of individuals with similar characteristics, allowing for a deeper understanding of stroke risk factors and potential subgroups within the population. By leveraging data mining in stroke prediction, healthcare professionals and researchers can gain valuable insights and develop effective strategies for stroke prevention, early detection, and personalized treatments.

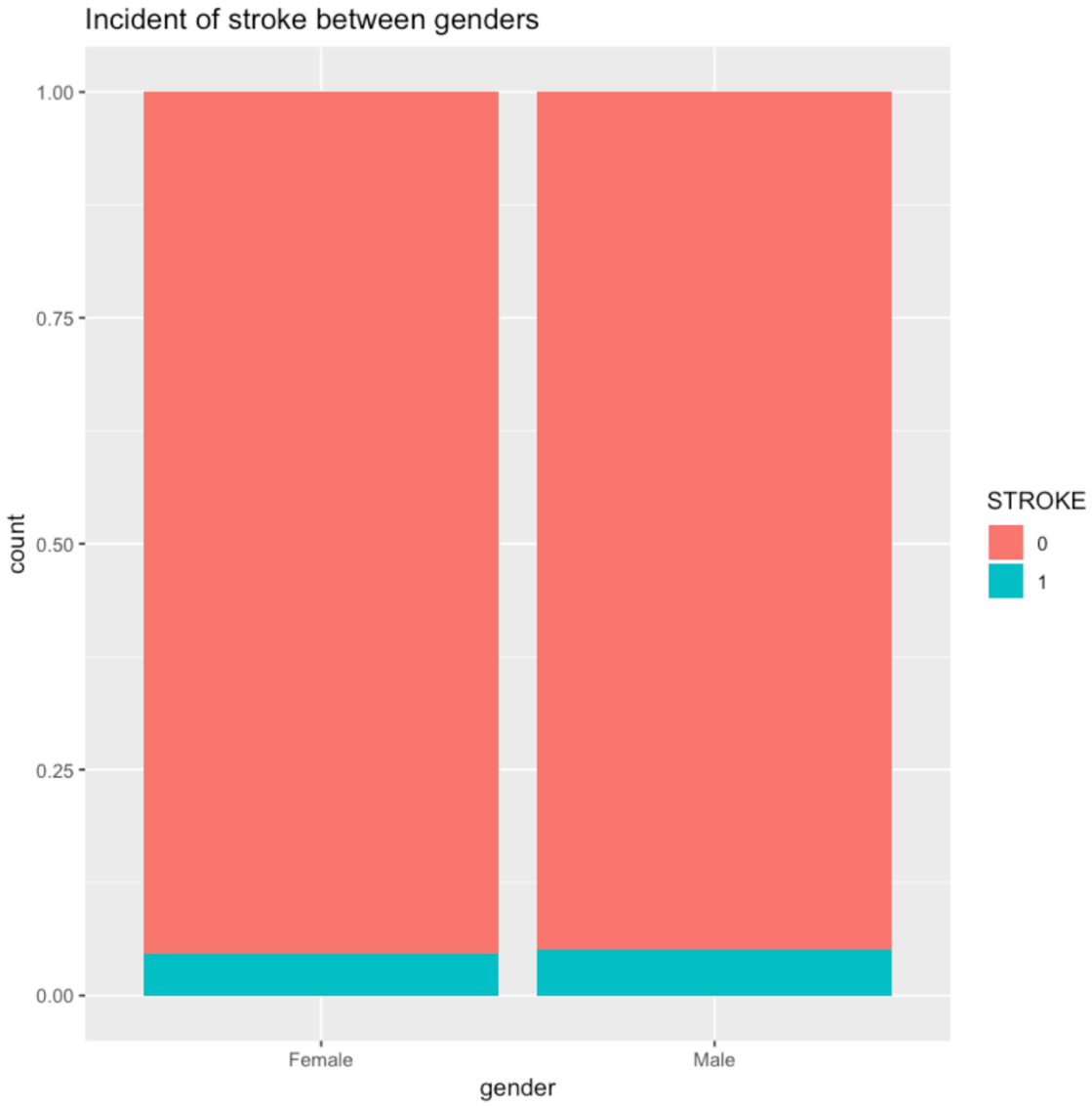
Dataset information

Our dataset source is : https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

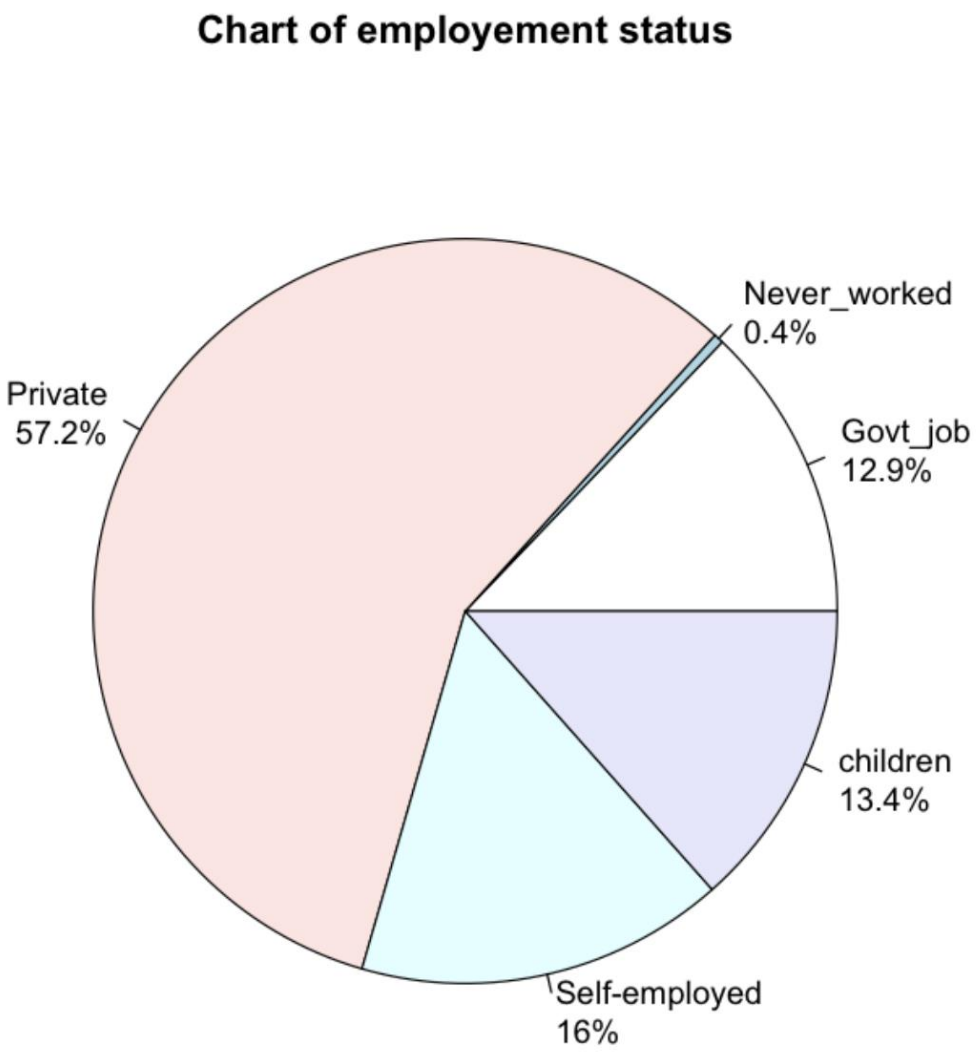
Among the 5110 objects in our dataset sample, 12 attributes are used to describe them. Our characteristics' values are utilized to identify their types, such as the nominal for id, binary for gender, and numeric for age. Additionally, we had two attributes for hypertension and heart disease that took two values 1 and 0 to indicate whether they are suffered from it or not, respectively. The last attribute, **"stroke"**, was described by two values 0 and 1 for the possibility of having a stroke or not as a result of analysis of the previous data, , which is what we aim to train our model to predict.The following table represent the data dictionary:

Attribute Name	Description	Data Type	Possible values
id	Unique id of the patient	Nominal	Range between 67-72940
gender	Gender of the patient	Binary	Female Male
age	Age of the patient	Numeric	Range between 0.08-82
hypertension	Hypertension binary feature, 1 means the patient has hypertension, 0 means they do not.	Binary	0,1
heart_disease	Heart disease binary feature, 1 means the patient has heart disease, 0 means they do not.	Binary	0,1
ever_married	Has the patient ever been married?	Binary	Yes No
work_type	Work type of the patient	Nominal	"Private" "Self-employed" "children" "Govt_job" "Never_worked"
residence_type	Residence type of the patient	Binary	"Urban" "Rural"
avg_glucose_level	Average glucose level in blood	Numeric	Range between 55.1-272
bmi	Body Mass Index	Numeric	Range between 10.3-97.6
smoking_status	Smoking status of the patient	Nominal	"never smoked" "Unknown" "formerly smoked" "smokes"
stroke	Stroke event, 1 means the patient had a stroke, 0 means not	Binary	0,1

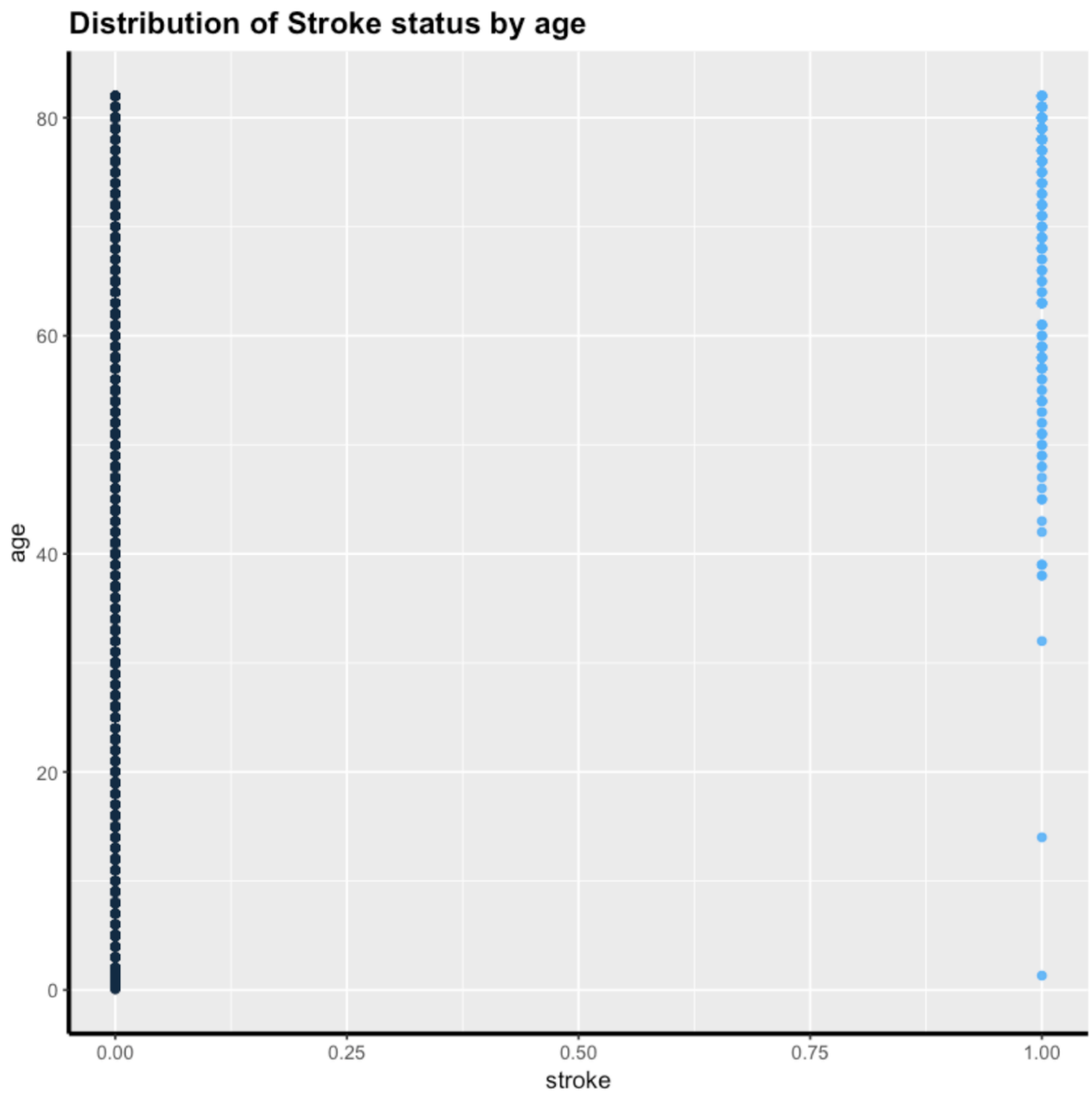
Understanding the data through graph representations:



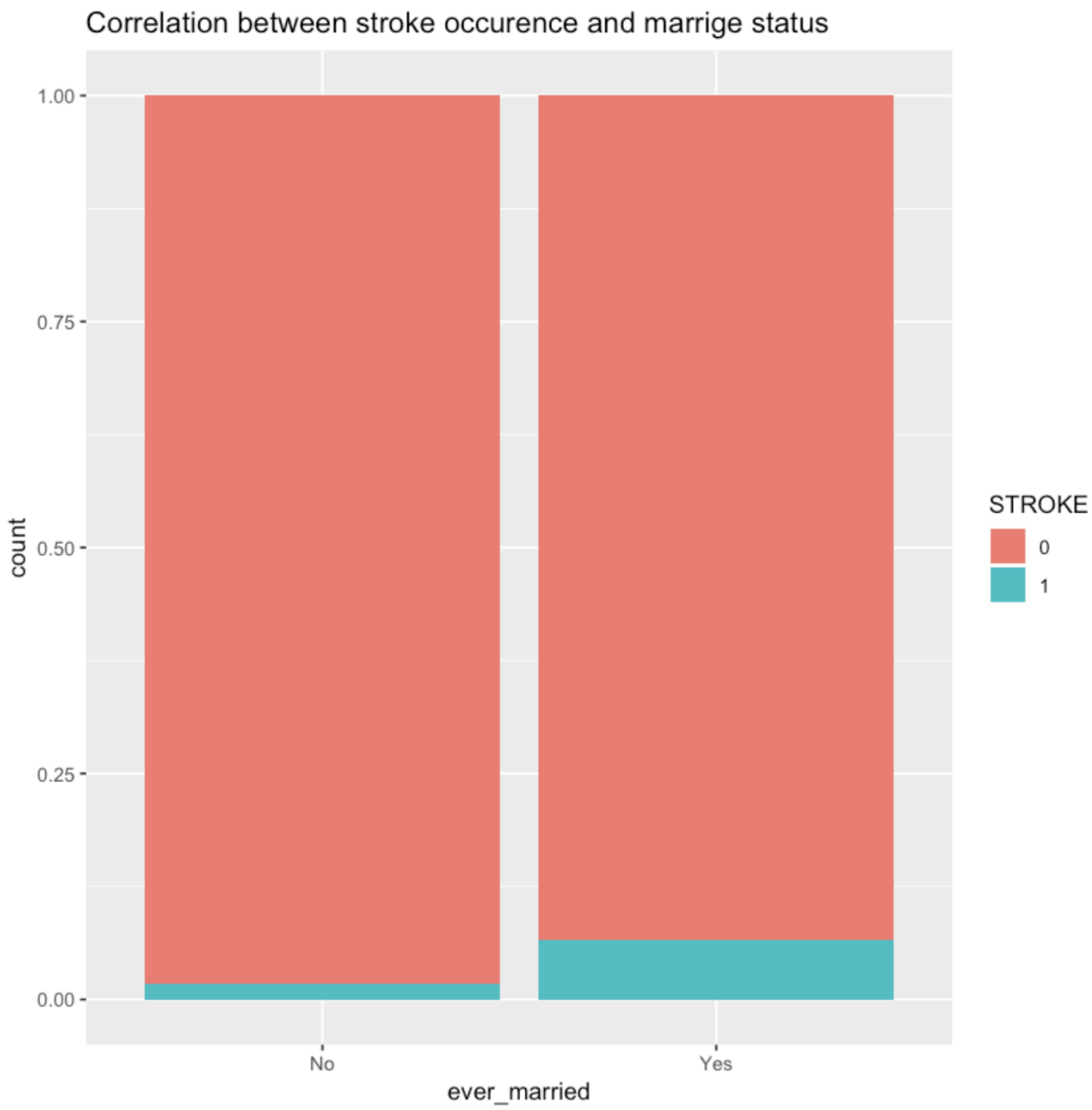
The total number of strokes by gender in our data set are shown in the bar chart. It demonstrates if there is a correlation between gender and the frequency of stroke cases in each gender. We can see that men are slightly more likely than women to experience a stroke.



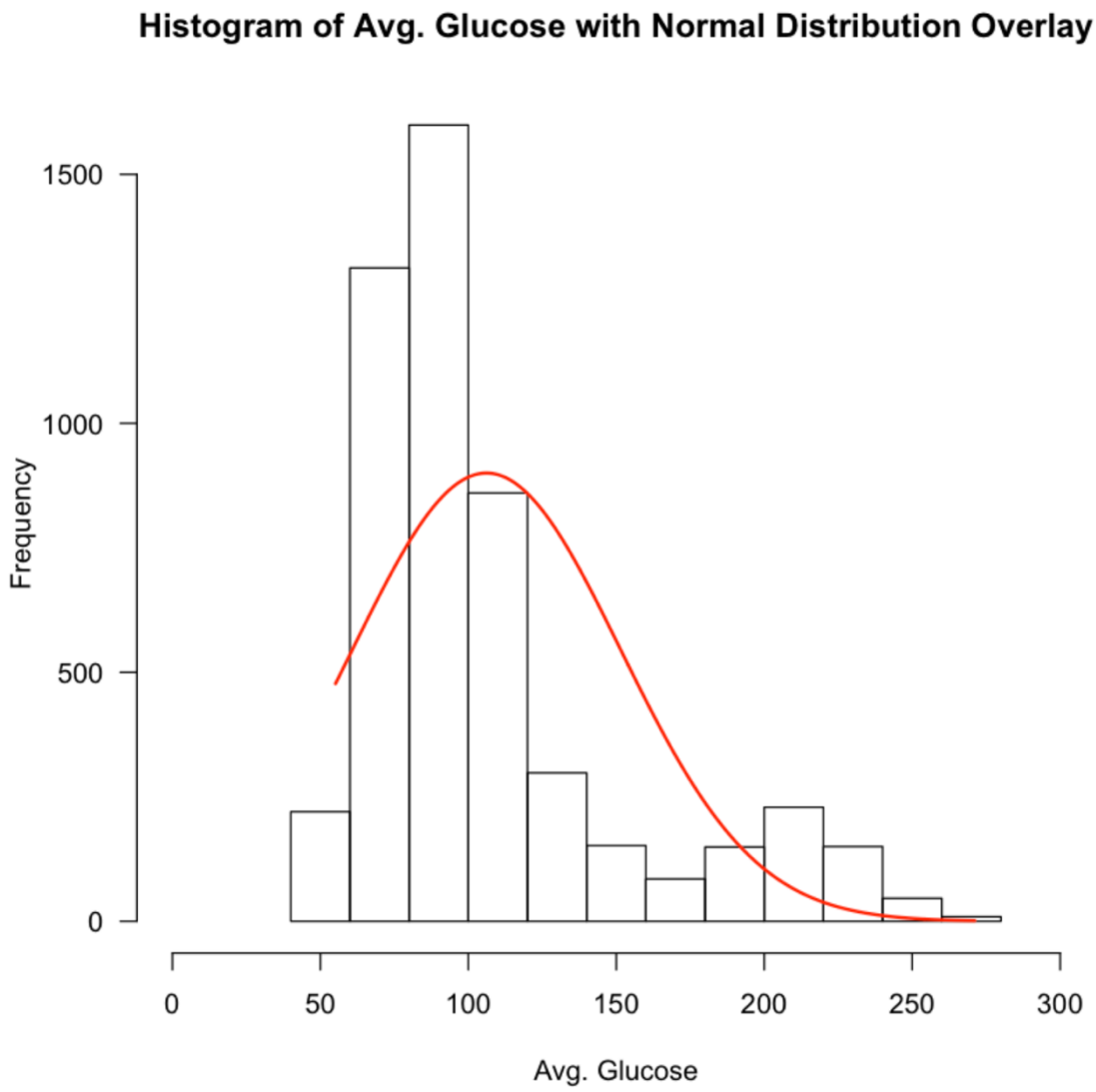
This pie chart illustrates the various worker types according to the employment sector type in our data collection. The summarization of our data collection's is nominal data, which shows people who work for the private sector are present at a higher percentages (57.2%) than those who work for the self-employed sector (16%) and so on.



The above chart shows the age distribution of stroke victims. Our results showed a correlation between age and stroke, showing a greater likelihood of stroke with older age.



The correlation between having a stroke and being married is illustrated in a bar graph. We discovered that people who are married have a higher risk of having a stroke than people who are not married.



The average glucose levels of the patients in the study are right skewed, with mean of 106.15 from the summary () function earlier. We notice that there is an increase in frequency when the glucose level reaches 200, which makes us wonder if this elevation is a factor for a stroke.

Data preprocessing

In our data, we performed several data preprocessing techniques such as data cleaning, data normalization, removing outliers, etc. for more information please refer to our juypter notebook `as it was discussed in detail.

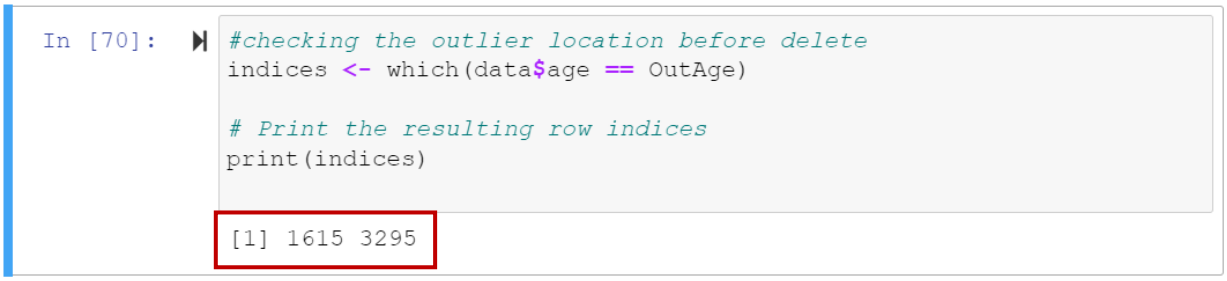


Figure 1

After detecting the outliers from the Age attribute, we took the necessary steps to verify the location and impact of these outliers before proceeding with their deletion. This approach allowed us to observe the dataset both before and after removing the outliers.

Figure 2

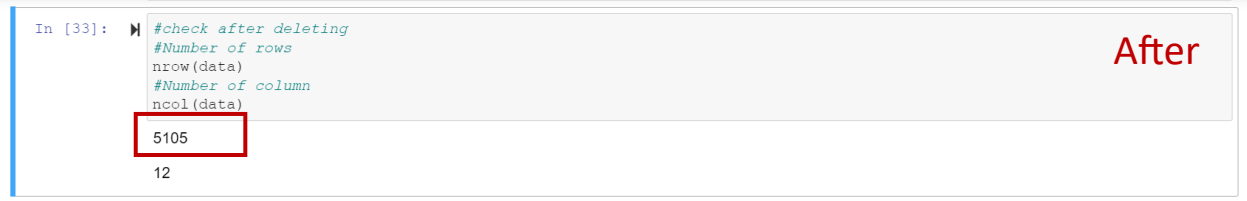


Figure 3

As a double check, we counted the rows before and after deletion and compared them to each other and confirmed that the deletion process is completed, as five rows were deleted according to the outlier of different attributes (Age, Glucose level, BMI).

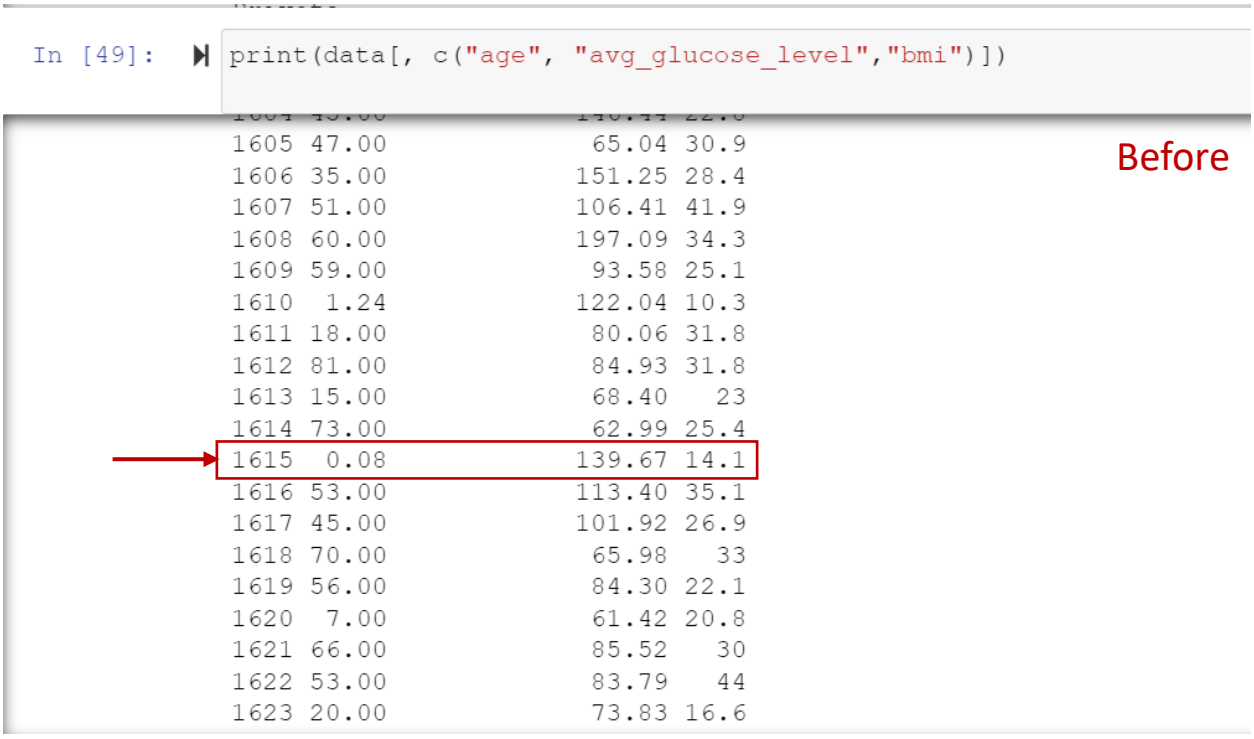


Figure 4

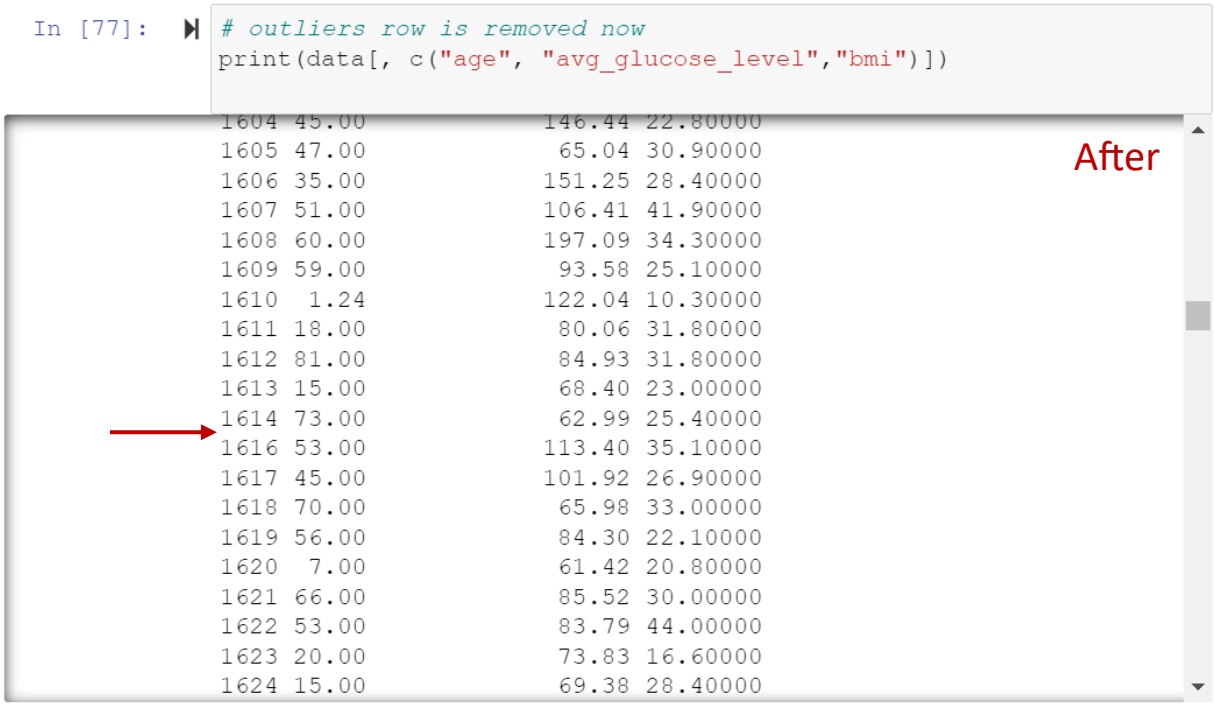


Figure 5

We also printed the dataset before and after and compared the values. We noticed that the patient row whose age was 0.8 had disappeared, which indicates the success of the deletion.

In [78]:
```
indices <- which(data$age == OutAge)

# Print the resulting row indices
print(indices)
```

```
integer(0)
```

In [79]:
```
indices3 <- which(data$avg_glucose_level == OutAvg)

# Print the resulting row indices
print(indices3)
```

```
integer(0)
```

In [80]:
```
indices2 <- which(data$bmi == 97.6)

# Print the resulting row indices
print(indices2)
```

```
integer(0)
```

Figure 6

To make sure that the deletion was successful, we searched for the rows that contain the Outlier values, and the results were all zero, which confirms to us that the deletion was successful.

5- Normalize Data using Min-Max Scaling:

In [61]:
```
normalize <- function(x)
{
 return ((x - min(x))/ (max(x)- min(x)))
}

data$avg_glucose_level= normalize(data$avg_glucose_level)
data$age= normalize(data$age)
data$bmi= normalize(data$age)
head(data)
```

id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
9046	1	0.8167155	0	1	1	4	1	0.8162622	0.8167155	3	1
51676	2	0.7434018	0	0	1	3	2	0.6917325	0.7434018	2	1
31112	1	0.9755621	0	1	1	4	2	0.2389014	0.9755621	2	1
60182	2	0.5967742	0	0	1	4	1	0.5460403	0.5967742	4	1
1665	2	0.9633431	1	0	1	3	2	0.5596313	0.9633431	2	1
56669	1	0.9877810	0	0	1	4	1	0.6164880	0.9877810	3	1

Figure 7

a normalization step was performed to ensure consistent scaling of the data. The normalization technique applied was the max-min normalization. This technique rescales the values of specific attributes within a defined range between 0 and 1.

The following attributes were selected for normalization: age, average glucose level, and BMI (Body Mass Index). We can use the normalized dataset to provide a more uniform and comparable representation of the attributes, enabling accurate analysis and modeling for stroke prediction with result as shown.

Imbalanced dataset problem:

In the given dataset, it is observed that only around 5% of all the individuals have experienced a stroke at some point. Consequently, our baseline dummy model achieves an accuracy of 95% by consistently predicting that individuals do not have a stroke. When evaluating our model, an essential metric to consider is sensitivity, also known as recall or the probability of detection. Low sensitivity indicates that our model struggles to identify true positive cases, even if the overall accuracy is high. In this case, the dummy model has a sensitivity of 0 since it fails to identify any true positives.

Addressing the class imbalance issue, there are various approaches available. Considering the limited size of the dataset, oversampling the minority class is deemed the most suitable strategy. By oversampling, we increase the representation of the minority class instances, enabling the model to learn and generalize better for this class.

Data mining techniques

We conducted both supervised and unsupervised learning on our dataset, utilizing classification and clustering techniques.

For classification, we employed a decision tree algorithm, which recursively constructs a tree with leaf nodes representing the final decisions. Our goal was to predict the class label "stroke," which has two categories: "yes" and "no." The prediction was based on selected attributes derived from the feature selection results, namely "ever_married," "hypertension," "avg_glucose," and "ages."

The classification technique involved splitting the dataset into two subsets:

Training dataset: Used for constructing the decision tree model.

Testing dataset: Employed to assess the performance of the constructed model.

To evaluate the effectiveness of our model, we utilized a confusion matrix and measured both accuracy and cost-sensitive metrics on the dataset.

Classification:

As known classification is supervised learning, so we need training data to train the model, we tried three different size of partitions and three attribute selection measures (IG, IG ratio, Gini index).

For Gini index we used Rpart which is a powerful machine learning library in R that is used for building classification and regression trees. This library implements recursive partitioning and is very easy to use. Rpart has many useful methods which assist us in building our model, for example rpart() for constructing the model and rpart.plot() for representing the tr

For IG we used ctree() in party package which depend in their performance on the concept of entropy and calculating the information gain. in addition, caret library was used for constructing the confusion matrix which helped as in evaluate the obtained model.

For gain ratio we used a C5.0 algorithm and C5.0() method which aims to find the feature that provides the most informative and balanced splits and construct the tree , considering both the reduction in uncertainty and the characteristics of the feature. This helps in avoiding biases and making more robust decisions during the tree construction process.

our training procedure was implemented first by using the method sample () for splitting the dataset into two subsets which are training data and testing data. We tried 3 different sizes of training subsets which are 70%, 80%, and 85% to get the best accuracy.

since our dataset size is limited, we set the value of replace attribute "TRUE" so the tuples of the training data will be selected with replacement and the rest will be included in the test data portion. We always gave the training subset the biggest portion of our dataset because of our model's ability to predict. class label correctly for the new tuples is depending on the operation of constructing and training the model. Our model represented by a decision tree algorithm needs to be fed with a large enough portion of data to be able to build the rules accurately so it can easily predict the labels when testing the model using the test data portion also in future uses. Choosing the attributes involved in classification was based on the results of feature selection.we specifically chose the first four attributes based on the results obtained.

Evaluation

Gini index

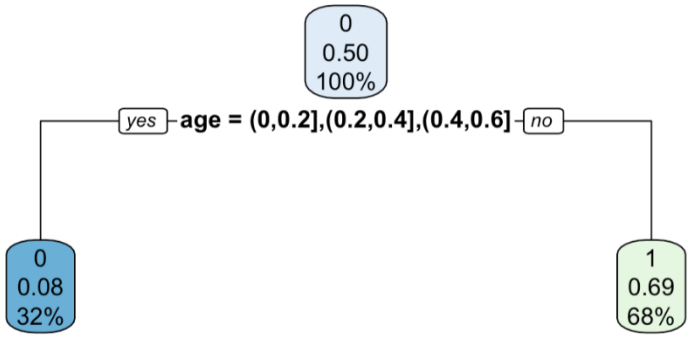
Gini index is a measure of impurity or the degree of disorder in a dataset. It is commonly used in decision tree algorithms to evaluate the quality of a split when constructing the tree.

When building a decision tree, each potential split is assessed using the Gini index. The Gini index calculates the probability of misclassifying a randomly chosen element from a dataset if it were randomly labeled according to the distribution of classes in that subset. A lower Gini index indicates a more pure or homogeneous subset, meaning that the classes within that subset are similar.

To use the Gini index in a decision tree, the algorithm considers various potential splits based on different features in the dataset. It calculates the Gini index for each split and selects the one with the lowest value. The chosen split results in the highest possible purity or homogeneity of the resulting subsets.

For Gini index we used Rpart which is a powerful machine learning library in R that is used for building classification and regression trees. This library implements recursive partitioning and is very easy to use. Rpart has many useful methods which assist us in building our model, for example rpart() for constructing the model and rpart.plot() for representing the tree

- partitioning the data into (70% training, 30% testing)



Decision tree

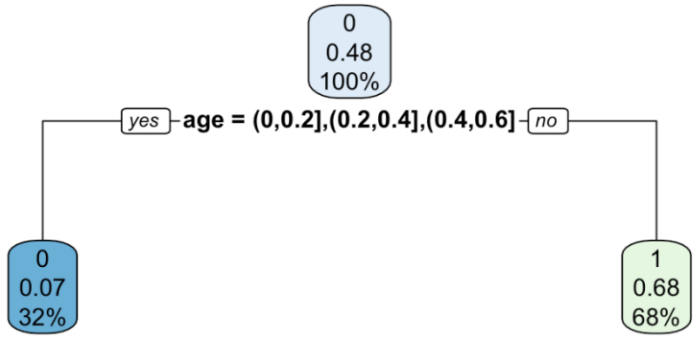
We can observe that we have 2rules as we have 2leaf nodes and 1 level, these are rules from the above tree :

- IF age =(0-0.2,0.2-0.4,0.4,0.6)THEN stroke= no.
- IF age > 0.6 THEN stroke=yes.

```
Accuracy
76.15894
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  90  14
         1  58 140

               Accuracy : 0.7616
                 95% CI : (0.7094, 0.8085)
    No Information Rate : 0.5099
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5202

 Mcnemar's Test P-Value : 4.029e-07

            Sensitivity : 0.9091
            Specificity : 0.6081
         Pos Pred Value : 0.7071
         Neg Pred Value : 0.8654
             Prevalence : 0.5099
         Detection Rate : 0.4636
   Detection Prevalence : 0.6556
      Balanced Accuracy : 0.7586

       'Positive' Class : 1

          Reference
Prediction   0   1
         0  90  14
         1  58 140
   0   1
0 90  14
1 58 140
                      [,1]
Accuracy       7.615894e-01
Kappa          5.202118e-01
AccuracyLower  7.094382e-01
AccuracyUpper  8.085273e-01
AccuracyNull   5.099338e-01
AccuracyPValue 2.304687e-19
McnemarPValue  4.028658e-07
                         [,1]
Sensitivity          0.9090909
Specificity          0.6081081
Pos Pred Value       0.7070707
Neg Pred Value       0.8653846
Precision            0.7070707
Recall               0.9090909
F1                   0.7954545
Prevalence           0.5099338
Detection Rate       0.4635762
Detection Prevalence 0.6556291
Balanced Accuracy    0.7585995
```

Confusion matrix and statistics

- partitioning the data into (80% training, 20% testing)



Decision tree

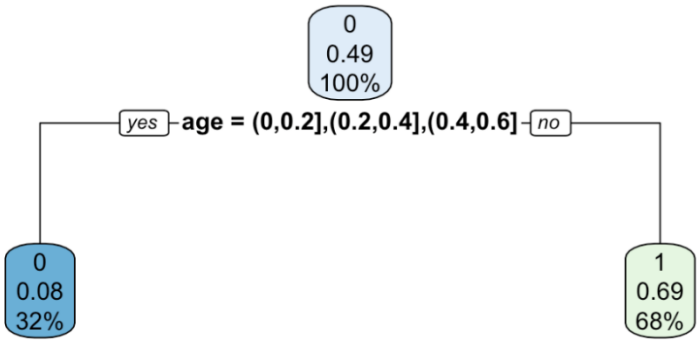
We can observe that we have 2rules as we have 2leaf nodes and 1 level, these are rules from the above tree :

- IF age =(0-0.2,0.2-0.4,0.4,0.6)THEN stroke= no.
- IF age > 0.6 THEN stroke=yes.

```
Accuracy
76.44231
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  55  12
         1  37 104

               Accuracy : 0.7644
                 95% CI : (0.7008, 0.8203)
    No Information Rate : 0.5577
    P-Value [Acc > NIR] : 4.505e-10

                  Kappa : 0.5087

 Mcnemar's Test P-Value : 0.0006068

            Sensitivity : 0.8966
            Specificity : 0.5978
         Pos Pred Value : 0.7376
         Neg Pred Value : 0.8209
             Prevalence : 0.5577
         Detection Rate : 0.5000
   Detection Prevalence : 0.6779
      Balanced Accuracy : 0.7472

       'Positive' Class : 1

          Reference
Prediction   0   1
         0  55  12
         1  37 104
   0   1
0 55  12
1 37 104
                      [,1]
Accuracy       7.644231e-01
Kappa          5.086772e-01
AccuracyLower  7.007972e-01
AccuracyUpper  8.203478e-01
AccuracyNull   5.576923e-01
AccuracyPValue 4.504835e-10
McnemarPValue  6.067668e-04
                         [,1]
Sensitivity          0.8965517
Specificity          0.5978261
Pos Pred Value       0.7375887
Neg Pred Value       0.8208955
Precision            0.7375887
Recall               0.8965517
F1                   0.8093385
Prevalence           0.5576923
Detection Rate       0.5000000
Detection Prevalence 0.6778846
Balanced Accuracy    0.7471889
```

Confusion matrix and statistics

- partitioning the data into (85% training, 15% testing)



Decision tree

We can observe that we have 2 rules as we have 2 leaf nodes and 1 level, these are rules from the above tree :

- IF age =(0-0.2,0.2-0.4,0.4,0.6)THEN stroke= no.
- IF age > 0.6 THEN stroke=yes.

```
Accuracy
76.10063
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 43  9
         1 29 78

               Accuracy : 0.761
                 95% CI : (0.687, 0.825)
    No Information Rate : 0.5472
    P-Value [Acc > NIR] : 1.91e-08

                  Kappa : 0.5059

 Mcnemar's Test P-Value : 0.002055

            Sensitivity : 0.8966
            Specificity : 0.5972
         Pos Pred Value : 0.7290
         Neg Pred Value : 0.8269
             Prevalence : 0.5472
         Detection Rate : 0.4906
   Detection Prevalence : 0.6730
      Balanced Accuracy : 0.7469

       'Positive' Class : 1

          Reference
Prediction  0  1
         0 43  9
         1 29 78
   0  1
0 43  9
1 29 78
                        [,1]
Accuracy       7.610063e-01
Kappa          5.058881e-01
AccuracyLower  6.870373e-01
AccuracyUpper  8.249851e-01
AccuracyNull   5.471698e-01
AccuracyPValue 1.909866e-08
McnemarPValue  2.054719e-03
                          [,1]
Sensitivity          0.8965517
Specificity          0.5972222
Pos Pred Value       0.7289720
Neg Pred Value       0.8269231
Precision            0.7289720
Recall               0.8965517
F1                   0.8041237
Prevalence           0.5471698
Detection Rate       0.4905660
Detection Prevalence 0.6729560
Balanced Accuracy    0.7468870
```

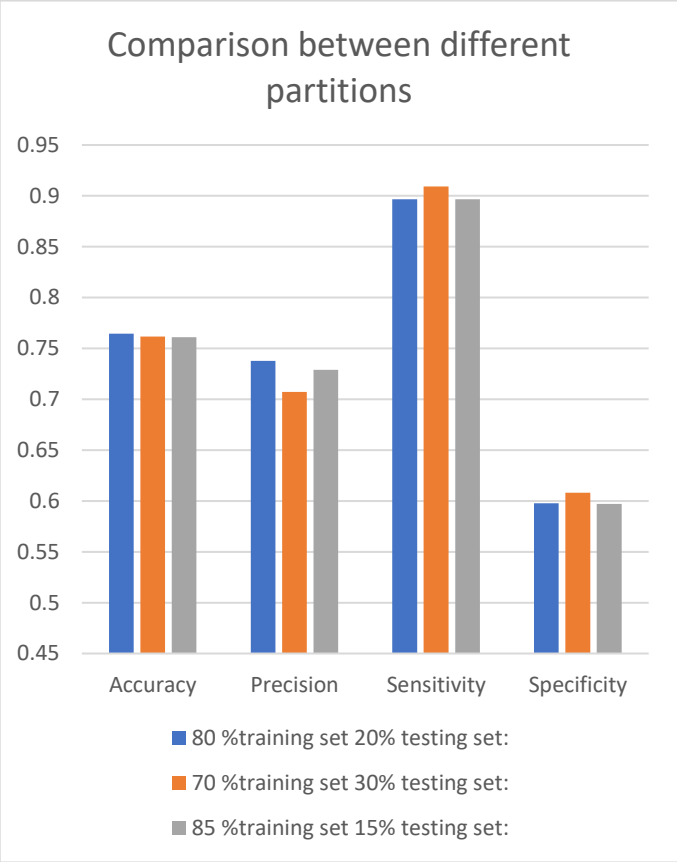
Confusion matrix and statistics

now let's compare between the different partitions results in GINI INDEX:

	80 %training set 20% testing set:	70 %training set 30% testing set:	85 %training set 15% testing set:
Accuracy	0.7644	0.7616	0.7610
precision	0.7376	0.7071	0.7290
sensitivity	0.8966	0.9091	0. 8966
specificity	0.5978	0.6081	0.5972

Looking at the accuracy values, the model trained on the 80% training set and tested on the 20% testing set achieved the highest accuracy (0.7644), followed closely by the model trained on the 70% training set and 30% testing set (0.7616), and the model trained on the 85% training set and 15% testing set (0.7610).

In terms of precision, which measures the proportion of correctly predicted positive instances, the model trained on the 80% training set and 20% testing set also obtained the highest precision (0.7376), followed by the model trained on the 85% training set and 15% testing set (0.7290), and the model trained on the 70% training set and 30% testing set (0.7071).



For sensitivity, which represents the ability to correctly identify positive instances, the model trained on the 70% training set and 30% testing set had the highest sensitivity (0.9091), followed by the models trained on the 80% training set and 20% testing set, as well as the 85% training set and 15% testing set, which both had a sensitivity of 0.8966.

In terms of specificity, which measures the ability to correctly identify negative instances, the model trained on the 70% training set and 30% testing set achieved the highest specificity (0.6081), followed by the model trained on the 80% training set and 20% testing set (0.5978), and the model trained on the 85% training set and 15% testing set (0.5972).

It is important to highlight that the decision tree constructed using the Gini index solely relied on the "age" attribute, which is not an ideal approach. Depending on a single attribute for building the decision tree can lead to a limited ability to predict accurately, and it is advisable to include additional relevant attributes. This raises the question of whether the "age" attribute truly has a stronger impact compared to other attributes in the feature selection results.

IG ratio

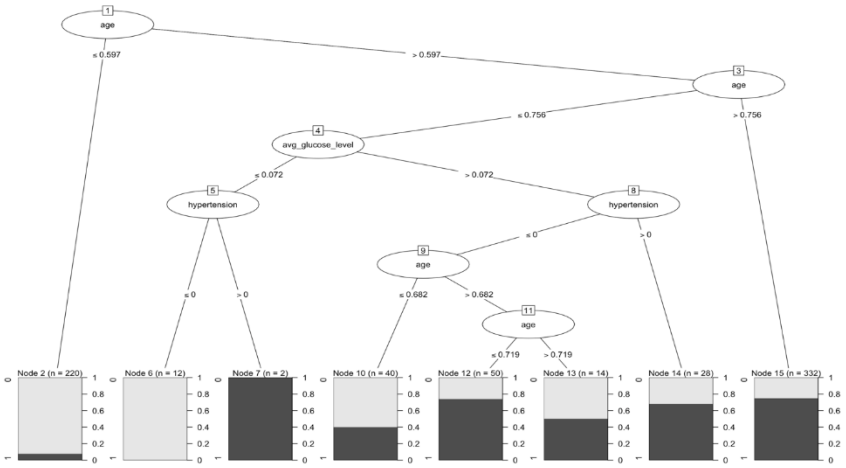
Gain ratio is a metric used in decision tree algorithms to evaluate the quality of a split based on the information gain and the intrinsic information of a feature. It takes into account the entropy or impurity of a dataset and the potential information gained by splitting the data based on a specific feature.

The gain ratio is calculated by dividing the information gain by the split information. Information gain measures the reduction in entropy achieved by splitting the dataset based on a particular feature. Split information quantifies the potential information generated by the feature itself.

Using the gain ratio in a decision tree, the algorithm compares the gain ratios of different features and selects the feature with the highest ratio as the best split. This approach helps prevent bias towards features with a large number of values or categories.

For gain ratio we used a C5.0 algorithm and C5.0() method which aims to find the feature that provides the most informative and balanced splits and construct the tree, considering both the reduction in uncertainty and the characteristics of the feature. This helps in avoiding biases and making more robust decisions during the tree construction process.

- partitioning the data into (70% training, 30% testing)



Decision tree

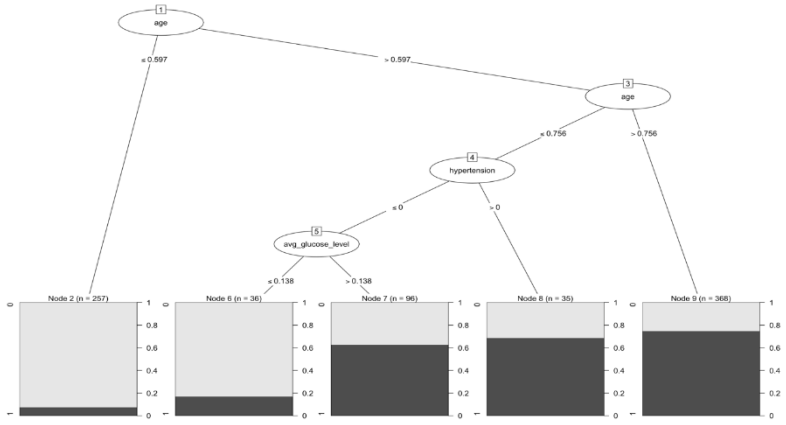
We can observe that we have 8rules as we have 8leaf nodes and 6 level, these are some rules from the above tree :

- IF age <=0.597 THEN stroke= no.
- IF age > 0.597 AND age >0.756 THEN stroke=yes.

Confusion matrix and statistics

```
Accuracy
78.47682
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 109  26
         1  39 128

               Accuracy : 0.7848
                 95% CI : (0.7341, 0.8298)
    No Information Rate : 0.5099
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5686

 Mcnemar's Test P-Value : 0.1366

            Sensitivity : 0.8312
            Specificity : 0.7365
         Pos Pred Value : 0.7665
         Neg Pred Value : 0.8074
             Prevalence : 0.5099
         Detection Rate : 0.4238
   Detection Prevalence : 0.5530
      Balanced Accuracy : 0.7838

       'Positive' Class : 1

          Reference
Prediction   0   1
         0 109  26
         1  39 128
    0   1
0 109  26
1  39 128
                      [,1]
Accuracy       7.847682e-01
Kappa          5.686283e-01
AccuracyLower  7.340723e-01
AccuracyUpper  8.297802e-01
AccuracyNull   5.099338e-01
AccuracyPValue 5.560128e-23
McnemarPValue  1.366410e-01
                         [,1]
Sensitivity         0.8311688
Specificity         0.7364865
Pos Pred Value      0.7664671
Neg Pred Value      0.8074074
Precision           0.7664671
Recall              0.8311688
F1                  0.7975078
Prevalence          0.5099338
Detection Rate      0.4238411
Detection Prevalence 0.5529801
Balanced Accuracy   0.7838277
```

- partitioning the data into (80% training, 20% testing)



Decision tree

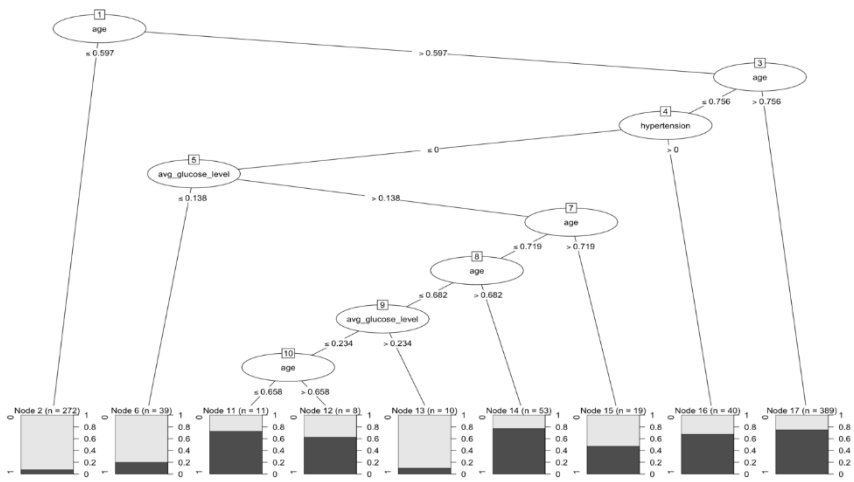
We can observe that we have 5 rules as we have 5 leaf nodes and 3 level, these are some rules from the above tree :

- IF age <=0.597 THEN stroke= no.
- IF age > 0.597 AND age >0.756 THEN stroke=yes.

```
Accuracy
      75
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 59 19
         1 33 97

               Accuracy : 0.75
                 95% CI : (0.6854, 0.8073)
    No Information Rate : 0.5577
    P-Value [Acc > NIR] : 7.029e-09

                  Kappa : 0.4851

 Mcnemar's Test P-Value : 0.07142

            Sensitivity : 0.8362
            Specificity : 0.6413
         Pos Pred Value : 0.7462
         Neg Pred Value : 0.7564
             Prevalence : 0.5577
         Detection Rate : 0.4663
   Detection Prevalence : 0.6250
      Balanced Accuracy : 0.7388

       'Positive' Class : 1

          Reference
Prediction  0  1
         0 59 19
         1 33 97
   0  1
0 59 19
1 33 97
                      [,1]
Accuracy       7.500000e-01
Kappa          4.851485e-01
AccuracyLower  6.854271e-01
AccuracyUpper  8.072990e-01
AccuracyNull   5.576923e-01
AccuracyPValue 7.029496e-09
McnemarPValue  7.142346e-02
                         [,1]
Sensitivity          0.8362069
Specificity          0.6413043
Pos Pred Value       0.7461538
Neg Pred Value       0.7564103
Precision            0.7461538
Recall               0.8362069
F1                   0.7886179
Prevalence           0.5576923
Detection Rate       0.4663462
Detection Prevalence 0.6250000
Balanced Accuracy    0.7387556
```

Confusion matrix and statistics

- partitioning the data into (85% training, 15% testing)



Decision tree

We can observe that we have 9 rules as we have 9 leaf nodes and 7 level, these are some rules from the above tree :

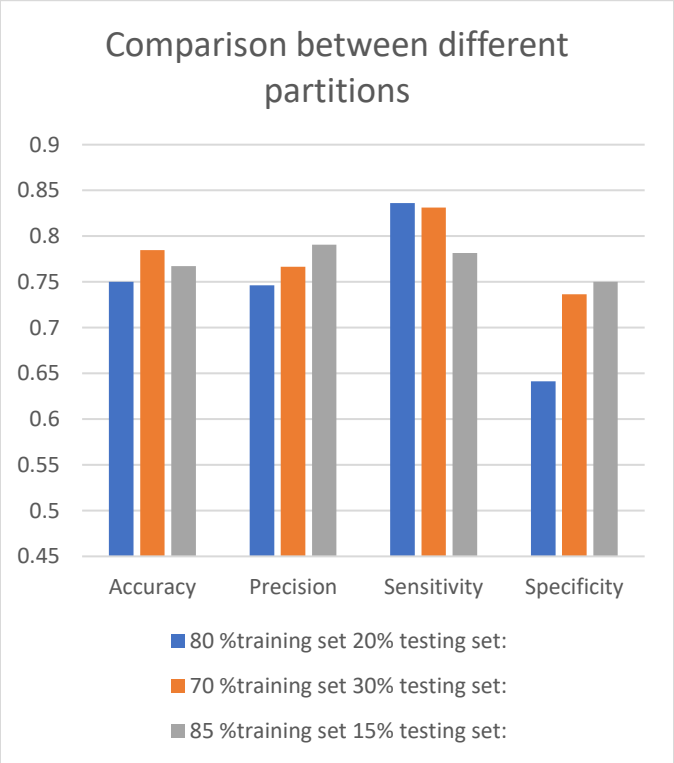
- IF age <=0.597 THEN stroke= no.
- IF age > 0.597 AND age >0.756 THEN stroke=yes.

```
Accuracy
76.72956
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 54 19
         1 18 68

               Accuracy : 0.7673
                 95% CI : (0.6938, 0.8306)
    No Information Rate : 0.5472
    P-Value [Acc > NIR] : 7.049e-09

                  Kappa : 0.531

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.7816
            Specificity : 0.7500
         Pos Pred Value : 0.7907
         Neg Pred Value : 0.7397
             Prevalence : 0.5472
         Detection Rate : 0.4277
   Detection Prevalence : 0.5409
      Balanced Accuracy : 0.7658

       'Positive' Class : 1

          Reference
Prediction  0  1
         0 54 19
         1 18 68
   0  1
0 54 19
1 18 68
                      [,1]
Accuracy       7.672956e-01
Kappa          5.309735e-01
AccuracyLower  6.938015e-01
AccuracyUpper  8.305536e-01
AccuracyNull   5.471698e-01
AccuracyPValue 7.049462e-09
McnemarPValue  1.000000e+00
                          [,1]
Sensitivity          0.7816092
Specificity          0.7500000
Pos Pred Value       0.7906977
Neg Pred Value       0.7397260
Precision            0.7906977
Recall               0.7816092
F1                   0.7861272
Prevalence           0.5471698
Detection Rate       0.4276730
Detection Prevalence 0.5408805
Balanced Accuracy    0.7658046
```

Confusion matrix and statistics

now lets compare between the different partitions results in GAIN RATIO

	80 %training set 20% testing set:	70 %training set 30% testing set:	85 %training set 15% testing set:
Accuracy	0.7500	0.7848	0.7673
Precision	0.7462	0.7665	0.7907
Sensitivity	0.8362	0.8312	0.7816
Specificity	0.6413	0.7365	0.7500



Among these partitioning ratios, the model trained on the 70% training set and 30% testing set achieved the highest accuracy (0.7848), followed by the model trained on the 85% training set and 15% testing set (0.7673), and the model trained on the 80% training set and 20% testing set (0.7500).

In terms of precision, the model trained on the 85% training set and 15% testing set achieved the highest precision (0.7907), followed by the model trained on the 70% training set and 30% testing set (0.7665), and the model trained on the 80% training set and 20% testing set (0.7462).

For sensitivity, the model trained on the 80% training set and 20% testing set had the highest sensitivity (0.8362), followed by the model trained on the 70% training set and 30% testing set (0.8312), and the model trained on the 85% training set and 15% testing set (0.7816).

In terms of specificity, the model trained on the 70% training set and 30% testing set achieved the highest specificity (0.7365), followed by the model trained on the 85% training set and 15% testing set (0.7500), and the model trained on the 80% training set and 20% testing set (0.6413).

Based on these results, the model trained on the 70% training set and 30% testing set appears to be the best partitioning ratio, as it showed the highest accuracy and a balanced performance in terms of precision, sensitivity, and specificity.

Overall, the results obtained from the decision tree using the gain ratio as the splitting criterion are realistic and promising. The accuracy values are reasonably high, indicating that the model can make correct predictions on the testing data. The precision, sensitivity, and specificity measures also demonstrate a good balance between correctly identifying the positive instances, correctly identifying the negative instances, and avoiding false positives and false negatives.

Information gain

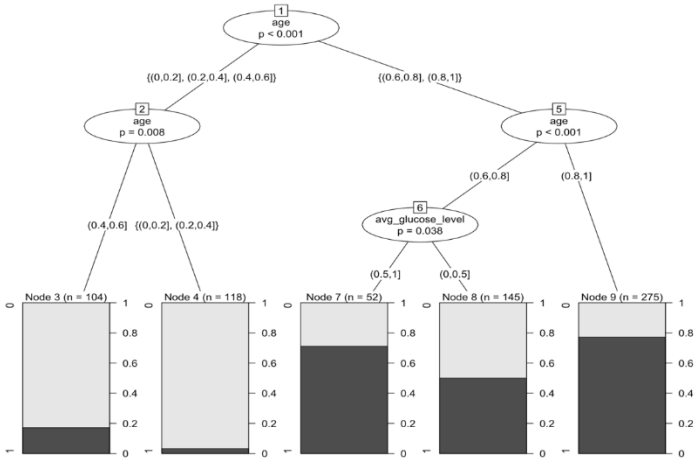
Information gain is a measure used in decision tree algorithms to evaluate the usefulness of a feature in splitting the data. It quantifies the reduction in entropy or impurity achieved by splitting the dataset based on that feature.

Entropy is a measure of the disorder or randomness in a dataset, specifically the uncertainty of class labels. Information gain calculates the difference between the entropy of the parent node (before the split) and the weighted average of the entropies of the child nodes (after the split).

To use information gain in a decision tree, the algorithm examines different features and calculates the information gain for each one. The feature with the highest information gain is selected as the best choice for splitting the data.

By selecting features with high information gain, the decision tree algorithm aims to create subsets that are more homogeneous or pure in terms of the class labels. This allows for better classification or prediction within each subset.

- partitioning the data into (70% training, 30% testing)



Decision tree

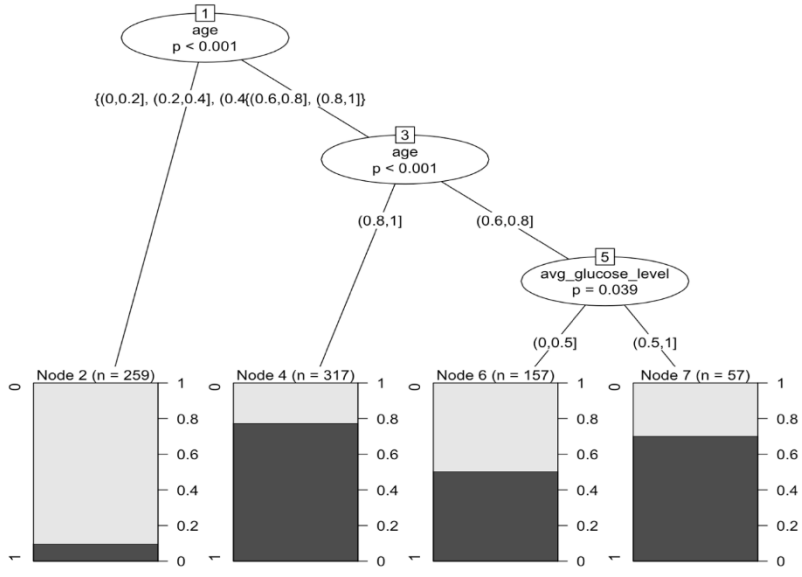
```
        Conditional inference tree with 5 terminal nodes

Response:  stroke
Inputs:  age, hypertension, ever_married, avg_glucose_level
Number of observations:  694

1) age == {(0,0.2], (0.2,0.4], (0.4,0.6]}; criterion = 1, statistic = 232.462
  2) age == {(0.4,0.6]}; criterion = 0.992, statistic = 12.527
    3)*  weights = 104
  2) age == {(0,0.2], (0.2,0.4]}
    4)*  weights = 118
1) age == {(0.6,0.8], (0.8,1]}
  5) age == {(0.6,0.8]}; criterion = 1, statistic = 24.775
    6) avg_glucose_level == {(0.5,1]}; criterion = 0.962, statistic = 6.687
      7)*  weights = 52
    6) avg_glucose_level == {(0,0.5]}
      8)*  weights = 145
  5) age == {(0.8,1]}
    9)*  weights = 275
```

Rules

```
Accuracy
78.10458
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  93   9
         1  58 146

               Accuracy : 0.781
                 95% CI : (0.7305, 0.8261)
    No Information Rate : 0.5065
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5602

 Mcnemar's Test P-Value : 4.515e-09

            Sensitivity : 0.9419
            Specificity : 0.6159
         Pos Pred Value : 0.7157
         Neg Pred Value : 0.9118
             Prevalence : 0.5065
         Detection Rate : 0.4771
   Detection Prevalence : 0.6667
      Balanced Accuracy : 0.7789

       'Positive' Class : 1

          Reference
Prediction   0   1
         0  93   9
         1  58 146
   0   1
0 93   9
1 58 146
                      [,1]
Accuracy       7.810458e-01
Kappa          5.601751e-01
AccuracyLower  7.304639e-01
AccuracyUpper  8.260940e-01
AccuracyNull   5.065359e-01
AccuracyPValue 3.827606e-23
McnemarPValue  4.514846e-09
                        [,1]
Sensitivity        0.9419355
Specificity        0.6158940
Pos Pred Value     0.7156863
Neg Pred Value     0.9117647
Precision          0.7156863
Recall             0.9419355
F1                 0.8133705
Prevalence         0.5065359
Detection Rate     0.4771242
Detection Prevalence 0.6666667
Balanced Accuracy  0.7789148
```

Confusion matrix and statistics

- partitioning the data into (80% training, 20% testing)



Decision tree

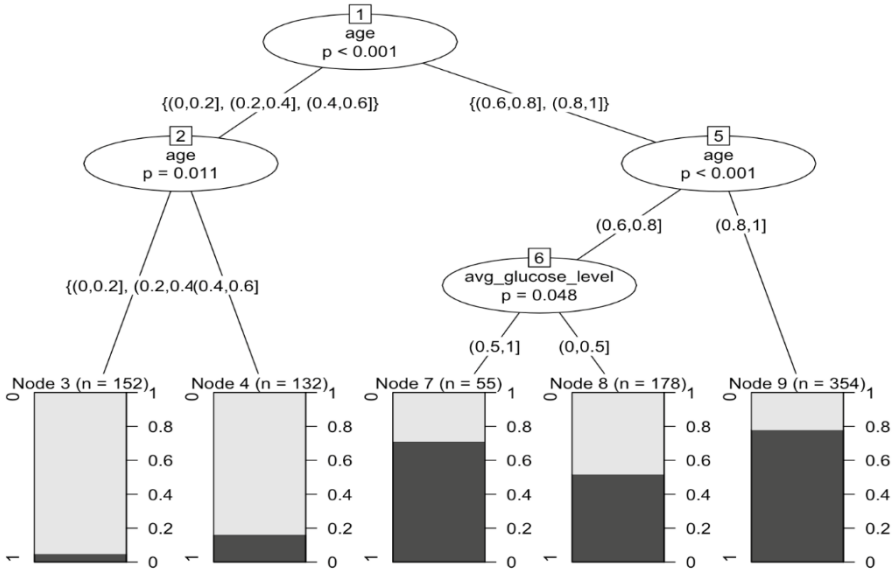
```
        Conditional inference tree with 4 terminal nodes

Response:  stroke
Inputs:  age, hypertension, ever_married, avg_glucose_level
Number of observations:  790

1) age == {(0,0.2], (0.2,0.4], (0.4,0.6]}; criterion = 1, statistic = 268.154
  2)*  weights = 259
1) age == {(0.6,0.8], (0.8,1]}
  3) age == {(0.8,1]}; criterion = 1, statistic = 27.799
    4)*  weights = 317
  3) age == {(0.6,0.8]}
    5) avg_glucose_level == {(0,0.5]}; criterion = 0.961, statistic = 6.648
      6)*  weights = 157
    5) avg_glucose_level == {(0.5,1]}
      7)*  weights = 57
```

Rules

```
Accuracy
78.09524
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  59   6
         1  40 105

               Accuracy : 0.781
                 95% CI : (0.7188, 0.8349)
    No Information Rate : 0.5286
    P-Value [Acc > NIR] : 3.117e-14

                  Kappa : 0.5522

 Mcnemar's Test P-Value : 1.141e-06

            Sensitivity : 0.9459
            Specificity : 0.5960
         Pos Pred Value : 0.7241
         Neg Pred Value : 0.9077
             Prevalence : 0.5286
         Detection Rate : 0.5000
   Detection Prevalence : 0.6905
      Balanced Accuracy : 0.7710

       'Positive' Class : 1

          Reference
Prediction   0   1
         0  59   6
         1  40 105
   0   1
0 59   6
1 40 105
                      [,1]
Accuracy       7.809524e-01
Kappa          5.521558e-01
AccuracyLower  7.188499e-01
AccuracyUpper  8.349412e-01
AccuracyNull   5.285714e-01
AccuracyPValue 3.116877e-14
McnemarPValue  1.141190e-06
                     [,1]
Sensitivity     0.9459459
Specificity     0.5959596
Pos Pred Value  0.7241379
Neg Pred Value  0.9076923
Precision       0.7241379
Recall          0.9459459
F1              0.8203125
Prevalence      0.5285714
Detection Rate  0.5000000
Detection Prevalence 0.6904762
Balanced Accuracy    0.7709528
```

Confusion matrix and statistics

- partitioning the data into (85% training, 15% testing)



Decision tree

```
        Conditional inference tree with 5 terminal nodes

Response:  stroke
Inputs:  age, hypertension, ever_married, avg_glucose_level
Number of observations:  871

1) age == {(0,0.2], (0.2,0.4], (0.4,0.6]}; criterion = 1, statistic = 301.227
  2) age == {(0,0.2], (0.2,0.4]}; criterion = 0.989, statistic = 11.709
    3)*  weights = 152
  2) age == {(0.4,0.6]}
    4)*  weights = 132
1) age == {(0.6,0.8], (0.8,1]}
  5) age == {(0.6,0.8]}; criterion = 1, statistic = 31.191
    6) avg_glucose_level == {(0.5,1]}; criterion = 0.952, statistic = 6.282
      7)*  weights = 55
    6) avg_glucose_level == {(0,0.5]}
      8)*  weights = 178
  5) age == {(0.8,1]}
    9)*  weights = 354
```

Rules

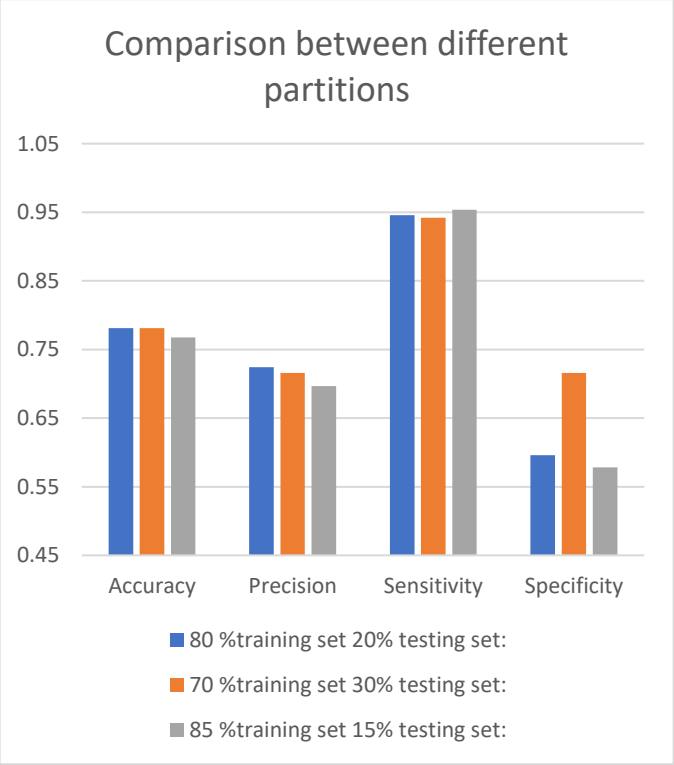
```
Accuracy
76.74419
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 37  3
         1 27 62

               Accuracy : 0.7674
                 95% CI : (0.6849, 0.8373)
    No Information Rate : 0.5039
    P-Value [Acc > NIR] : 7.166e-10

                  Kappa : 0.5335

 Mcnemar's Test P-Value : 2.679e-05

            Sensitivity : 0.9538
            Specificity : 0.5781
         Pos Pred Value : 0.6966
         Neg Pred Value : 0.9250
             Prevalence : 0.5039
         Detection Rate : 0.4806
   Detection Prevalence : 0.6899
      Balanced Accuracy : 0.7660

       'Positive' Class : 1

          Reference
Prediction  0  1
         0 37  3
         1 27 62
   0  1
0 37  3
1 27 62
                      [,1]
Accuracy       7.674419e-01
Kappa          5.335101e-01
AccuracyLower  6.849483e-01
AccuracyUpper  8.372617e-01
AccuracyNull   5.038760e-01
AccuracyPValue 7.166227e-10
McnemarPValue  2.678522e-05
                      [,1]
Sensitivity          0.9538462
Specificity          0.5781250
Pos Pred Value       0.6966292
Neg Pred Value       0.9250000
Precision            0.6966292
Recall               0.9538462
F1                   0.8051948
Prevalence           0.5038760
Detection Rate       0.4806202
Detection Prevalence 0.6899225
Balanced Accuracy    0.7659856
```

Confusion matrix and statistics

now lets compare between the different partitions results in INFORMATION GAIN

	80 %training set 20% testing set:	70 %training set 30% testing set:	85 %training set 15% testing set:
Accuracy	0.781	0.781	0.7674
precision	0.7241	0.7157	0.6966
sensitivity	0.9459	0.9419	0.9537
specificity	0.5960	0.7157	0.5781

Among these partitioning ratios, both the models trained on the 80% training set and 20% testing set, as well as the 70% training set and 30% testing set, achieved the same accuracy score (0.781), which is the highest among the three ratios. The model trained on the 85% training set and 15% testing set had a slightly lower accuracy (0.7674).



In terms of precision, the model trained on the 80% training set and 20% testing set obtained the highest precision (0.7241), followed by the model trained on the 70% training set and 30% testing set (0.7157), and the model trained on the 85% training set and 15% testing set (0.6966).

For sensitivity, the model trained on the 85% training set and 15% testing set achieved the highest sensitivity (0.9537), followed by the models trained on the 80% training set and 20% testing set (0.9459), and the 70% training set and 30% testing set (0.9419).

In terms of specificity, the model trained on the 70% training set and 30% testing set obtained the highest specificity (0.7157), followed by the model trained on the 80% training set and 20% testing set (0.5960), and the model trained on the 85% training set and 15% testing set (0.5781).

Based on these results, both the models trained on the 80% training set and 20% testing set, and the 70% training set and 30% testing set show similar and better performance compared to the model trained on the 85% training set and 15% testing set. The accuracy, precision, sensitivity, and specificity measures are relatively high, indicating that the models can make reasonably accurate predictions.

Overall, the results obtained from the decision tree using information gain as the splitting criterion are realistic and indicate good performance.

Clustering:

The clustering process involves grouping similar data points together based on their inherent characteristics or similarities. One popular clustering algorithm is k-means, which partitions the dataset into k clusters, where each data point belongs to the cluster with the nearest mean value based on Euclidean distance. We can implement clustering by following a few steps.

```
> ###################################Cluster
> library(factoextra)
> sr<-(data$stroke)
> data <- data[,!names(data) %in% c("stroke","id","Residence_type","gender","heart_disease","bmi","work_type","smoking_status")]
> head(data)
         age hypertension ever_married avg_glucose_level
1 0.03470186            0            1        0.18811136
2 0.70674487            1            2        0.15443943
3 0.09579668            0            1        0.26227427
4 0.85337243            0            2        0.06546275
5 0.16911046            0            1        0.49924755
6 0.57233627            0            2        0.73283484
> str(data)
'data.frame':   9714 obs. of  4 variables:
 $ age              : num  0.0347 0.7067 0.0958 0.8534 0.1691 ...
 $ hypertension     : num  0 1 0 0 0 0 0 0 0 1 ...
 $ ever_married     : num  1 2 1 2 1 2 2 2 2 2 ...
 $ avg_glucose_level: num  0.1881 0.1544 0.2623 0.0655 0.4992 ...
```

First, we remove the class label "stroke" from the dataset since clustering is an unsupervised learning task and does not require labeled data. In addition, we remove certain attributes (heart_disease, work_type, smoking_status, and bmi) and selected the top four attributes (age, hypertension, ever_married, avg_glucose_level) after applying feature selection.

```
> data$age<- as.numeric(data$age )
> data$hypertension <- as.numeric(data$hypertension )
> data$avg_glucose_level <- as.numeric(data$avg_glucose_level)
> data$ever_married <- as.numeric(data$ever_married)
> #for ease creating a data set without color column
> data_no_color <- data[1:4]
> #Let us see the structure again
> str(data_no_color)
'data.frame':   9714 obs. of  4 variables:
 $ age              : num  0.0347 0.7067 0.0958 0.8534 0.1691 ...
 $ hypertension     : num  0 1 0 0 0 0 0 0 0 1 ...
 $ ever_married     : num  1 2 1 2 1 2 2 2 2 2 ...
 $ avg_glucose_level: num  0.1881 0.1544 0.2623 0.0655 0.4992 ...
>
```

we convert all the remaining attributes in the dataset to numeric format. This step ensures compatibility with the k-means algorithm, which operates on numerical data.

Once the preprocessing is complete, we can proceed with implementing the k-means clustering algorithm. We start by selecting a suitable value for k, which represents the number of clusters we want to identify in the data. To determine the optimal choice of k, we can try different key sizes such as 2, 3, or 5, and evaluate the clustering results for each value.

After running the k-means algorithm with different values of k, we can assess the quality of the clustering using various evaluation metrics, such as within-cluster sum of squares, silhouette analysis, and BCubed precision and recall. These metrics help us determine the optimal number of clusters that provide the most meaningful and well-separated groups.

By applying the k-means algorithm to the Stroke Prediction Dataset, while removing the stroke class label and converting attributes to numeric format, we can effectively perform clustering analysis to identify patterns and groupings within the data. This process aids in uncovering valuable insights and potentially discovering hidden relationships among the variables, which

can further enhance our understanding of stroke risk factors and contribute to better healthcare strategies.

K= 2

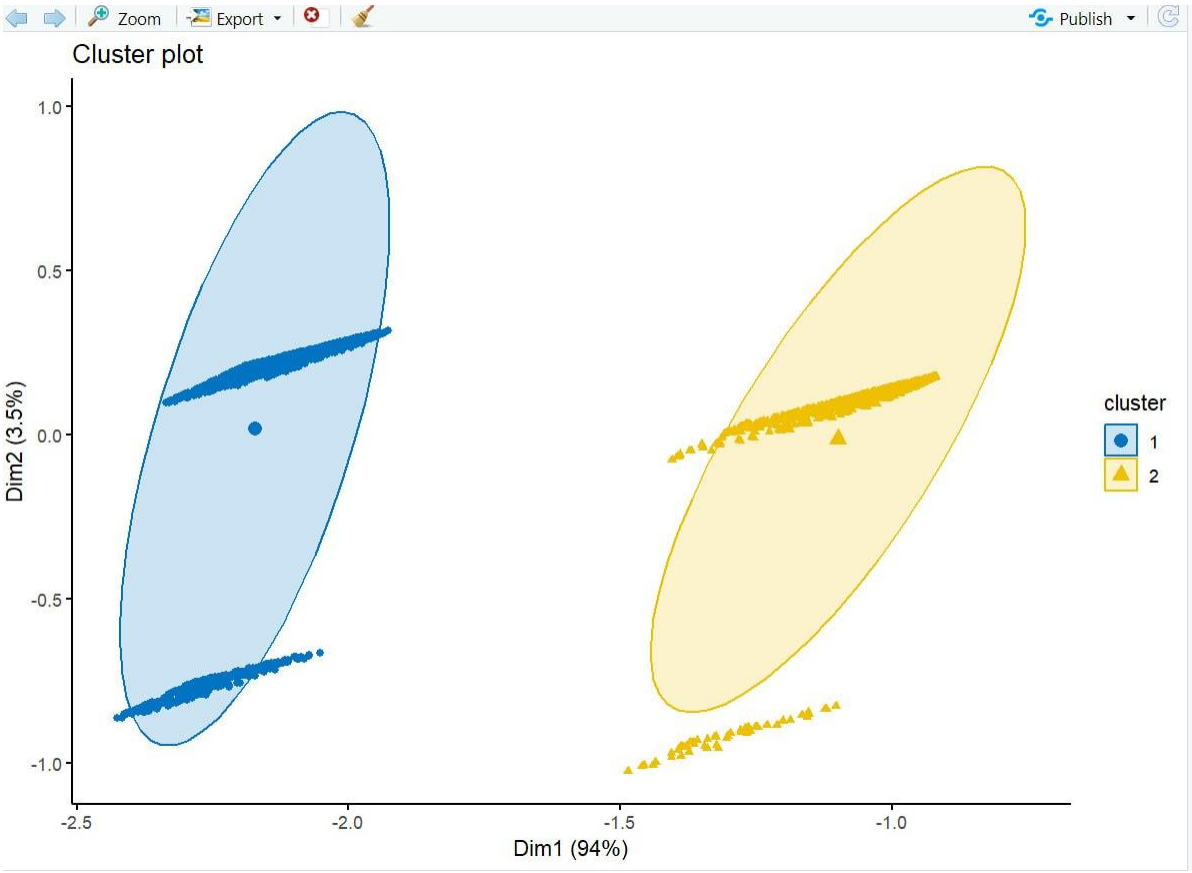


Figure 1

The analysis of the clustering results reveals several significant findings. Firstly, the observation of two non-overlapping clusters in the image indicates the successful separation of data points into distinct groups by the employed clustering algorithm. This outcome is indicative of favorable grouping results, as it demonstrates the algorithm's ability to identify and delineate cohesive clusters within the dataset.

Furthermore, the considerable distance observed between the two clusters signifies a clear separation in the feature space. This characteristic is highly desirable in clustering tasks, as it suggests a substantial dissimilarity between the data points belonging to different clusters. The notable distance between clusters enhances the interpretability of the results and indicates the presence of distinctive patterns or characteristics within each group.

Additionally, the nearly identical sizes of the two clusters are a notable observation. The presence of similar cluster sizes is advantageous in clustering analysis as it implies a balanced distribution of data points across the clusters. This balance facilitates a fair representation of the underlying data and simplifies the interpretation of the unique characteristics exhibited by each cluster.

```
> fviz_silhouette(sil)
  cluster size ave.sil.width
1       1 2274          0.58
2       2 7440          0.51
```

Figure 2

These distinct values suggest the presence of characteristic attributes that differentiate the clusters from each other. Such differentiation is indicative of reasonable clustering outcomes, as it implies the identification of distinct subgroups within the dataset based on the selected attributes. Moreover, the average silhouette widths, measuring 0.58 and 0.51, offer further evaluation of the clustering quality.

Average silhouette:

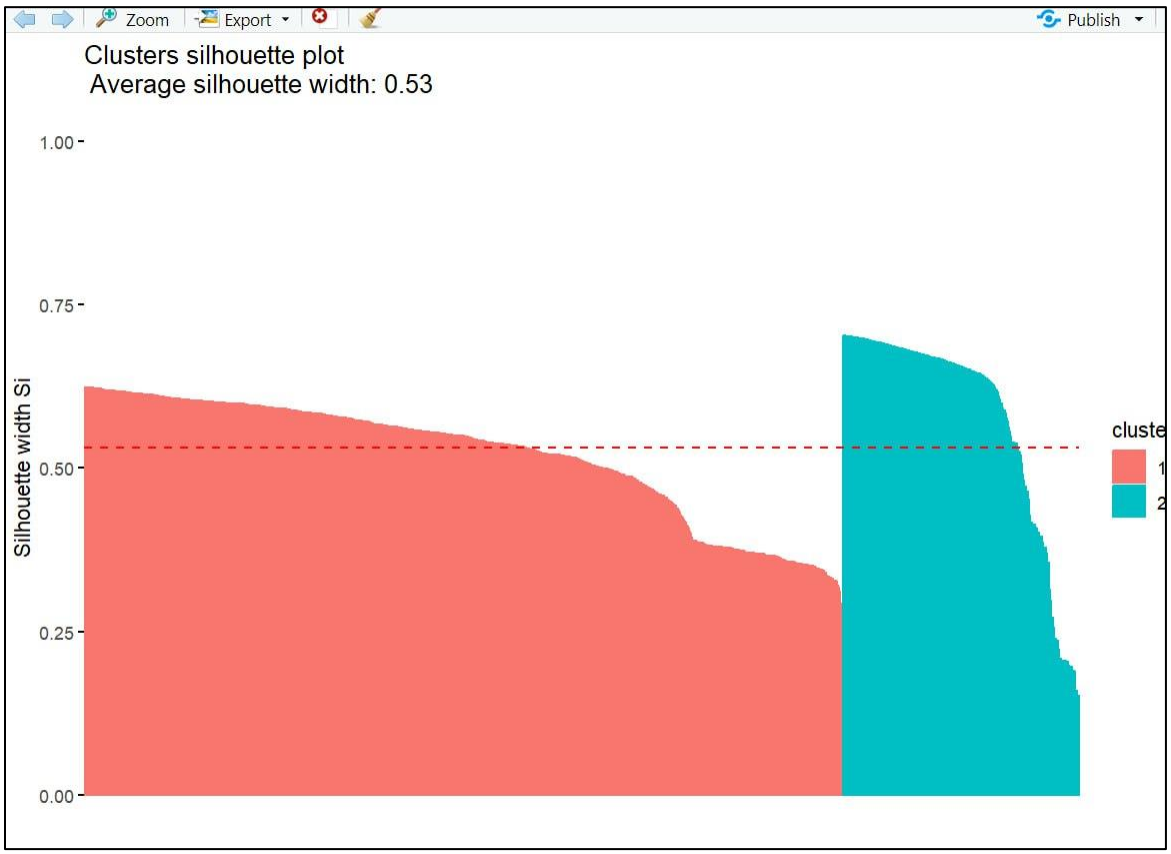


Figure 3

A silhouette width above 0.5 is generally considered indicative of a reasonable degree of separation between data points within their assigned clusters. In this analysis, the observed silhouette widths surpass this threshold, suggesting a satisfactory level of discrimination and cohesion within each cluster. Consequently, the clustering algorithm has likely achieved a suitable level of within-cluster similarity and between-cluster dissimilarity, reinforcing the reliability of the clustering results.

Total within-cluster sum of square:

```
> # Total sum of squares
> km$tot.withinss
[1] 2521.881
```

Figure 4

Total Sum of Squares (WSS), which serves as a metric for assessing the compactness of the clusters. However, the absolute value of WSS alone may not provide a definitive assessment of the clustering quality and should be interpreted relative to other evaluations.

Average BCubed precision and recall

```
> BCubed_metric(sr, km$cluster, 0.5)
[1] 0.5986005
>
```

Figure 5

The function returns Bcubed F score of the clustering method. The higher the value is, the better performance the clustering method can get, our value (0.5986005) is generally desirable, as it indicates better agreement between the clustering results and the reference class label “stroke.

K= 3

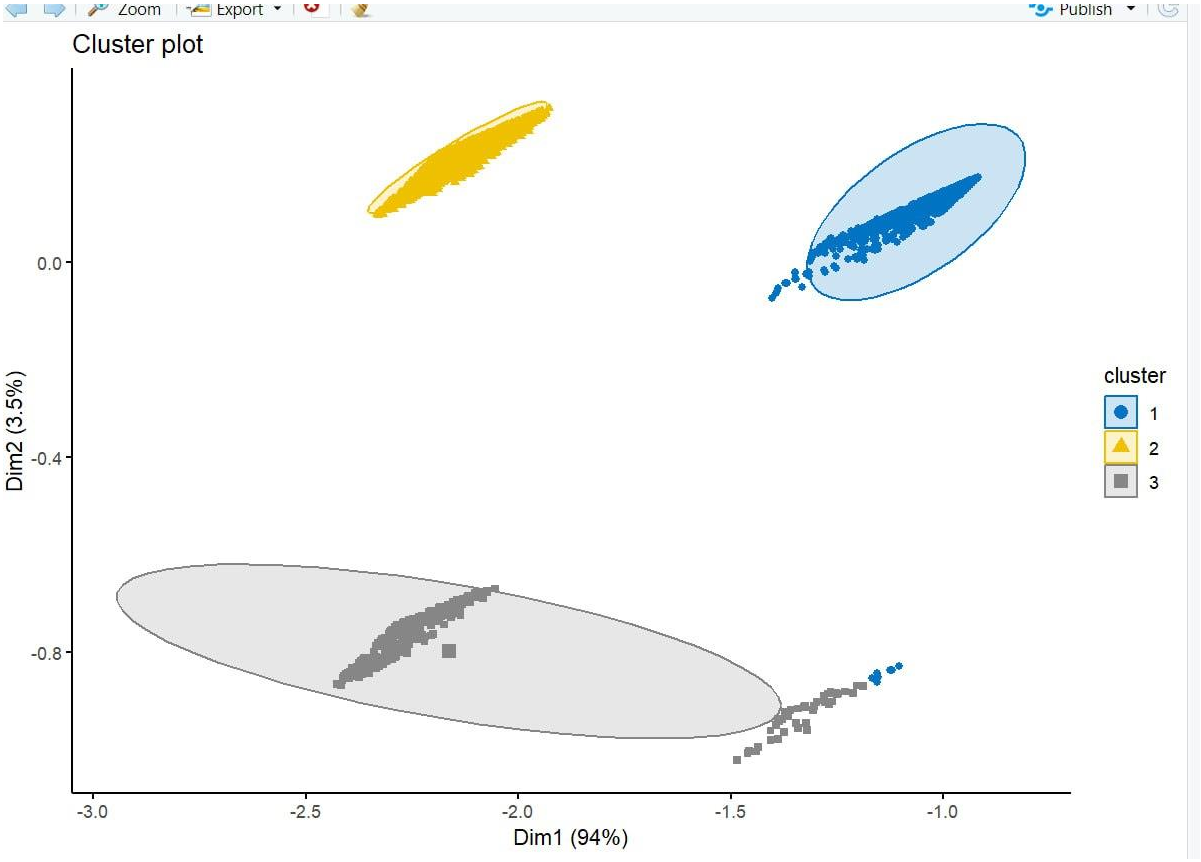


Figure 6

We performed clustering analysis with a value of k set to 3 using the 3-mean clustering method. The resulting cluster figure revealed three clusters of unequal size, with one cluster appearing significantly larger compared to the other two. However, it is worth noting since that each cluster contained an approximately equal number of objects.

The clusters displayed clear separation from each other, indicating distinct groupings within the dataset. This separation is a positive outcome as it suggests that the clustering algorithm successfully identified different patterns or characteristics among the data points.

Average silhouette:

Figure 7

The average silhouette width, calculated as 0.62, indicates a reasonably high level of separation and cohesion within the clusters. As we mention before that a silhouette width above 0.5 is generally considered indicative of a reasonable degree of separation between data points within their assigned clusters. Therefore, the observed average silhouette width of 0.62 suggests that the clustering algorithm achieved a satisfactory level of discrimination. Comparing this value with the average silhouette width obtained from the 2-mean clustering solution would provide a better understanding of the relative improvement.

Total within-cluster sum of square:

```
> km$tot.withinss
[1] 1239.568
```

Figure 8

A lower WSS value generally indicates better clustering results, suggesting that the data points within each cluster are more tightly grouped around their centroids. And our value here is less than the 2-mean value which mean this clustering is better.

Average BCubed precision and recall

```
> BCubed_metric(sr, km$cluster, 0.5)
[1] 0.5209976
>
```

Figure 9

The Average BCubed precision and recall value of 0.5209976 provides an evaluation of the clustering method's agreement with the reference class label “stroke”. While a higher BCubed goodness value is generally desirable.

K= 5

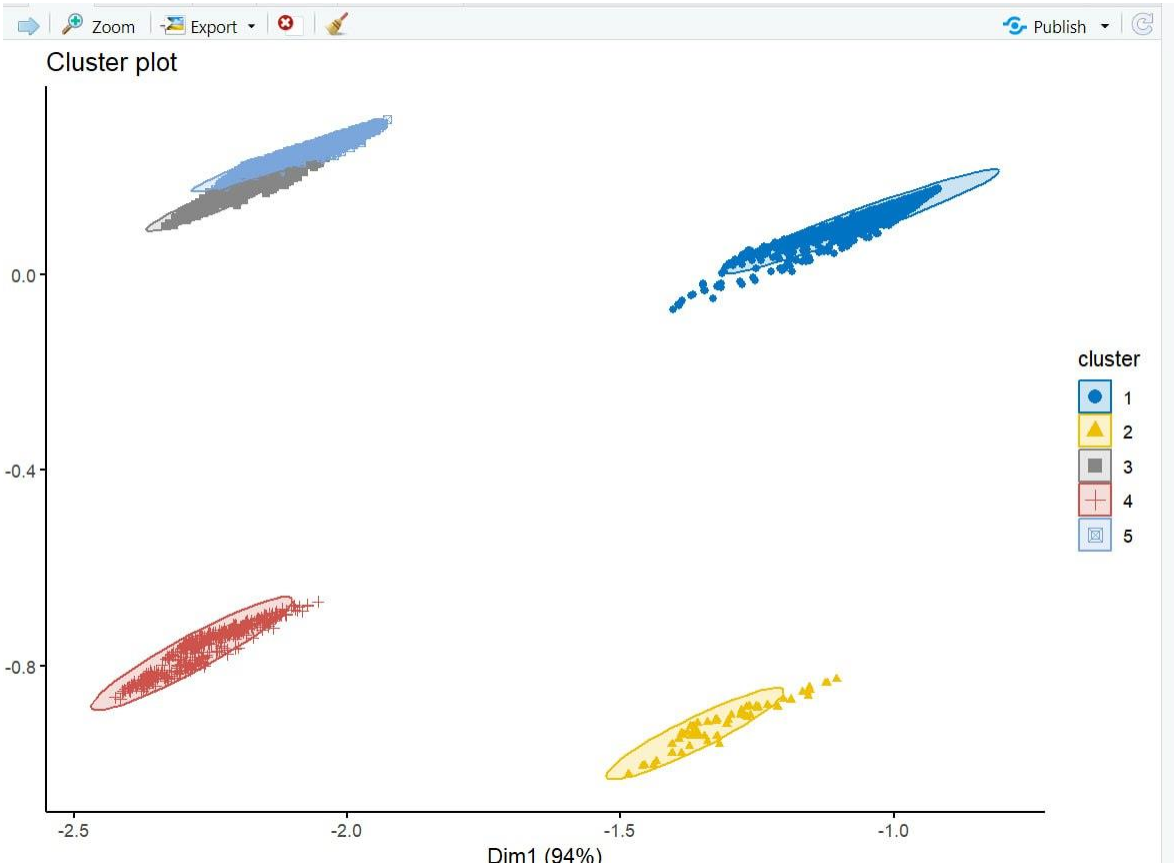


Figure 10

We utilized a value of k equal to 5, employing the 5-mean clustering method. The resulting cluster figure displayed five clusters of equal size. However, two of these clusters appeared to be in close proximity to each other, while the remaining clusters exhibited clear separation.

The closeness between two clusters can occur due to several reasons. It is possible that the data points in these clusters share similar characteristics or have overlapping features, leading to a reduced degree of separation. Alternatively, it could be an indication of limitations in the clustering algorithm's ability to distinguish between these clusters effectively.

```
  cluster size ave.sil.width
1       1 2038          0.66
2       2  237          0.70
3       3 1534          0.61
4       4 1489          0.59
5       5 4416          0.54
```

Figure 11

Average silhouette:

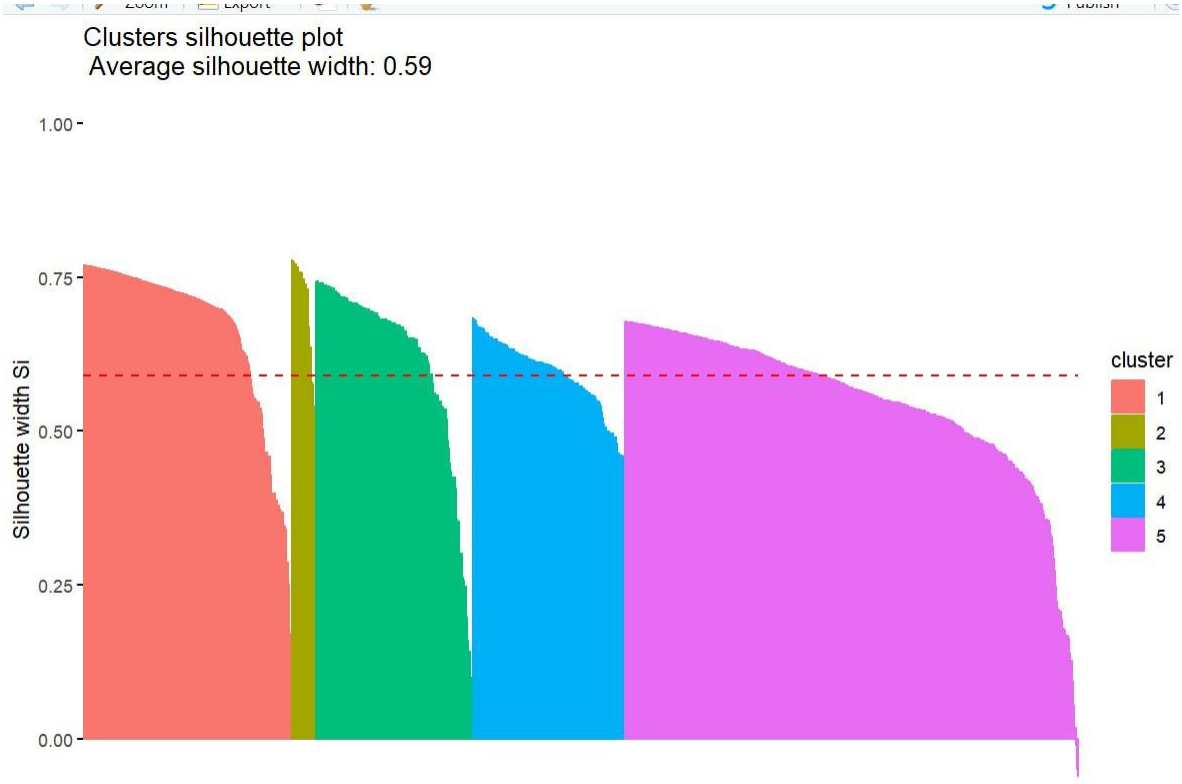


Figure 12

the observed value of 0.59 indicates a slightly lower level of discrimination and cohesion compared to the 3-mean clustering solution. The difference in average silhouette width between these two solutions suggests that the 3-mean clustering approach achieved a relatively higher level of separation and cohesion within each cluster.

Total within-cluster sum of square:

```
> # Total sum of squares
> km$tot.withinss
[1] 671.4406
```

Figure 13

The lower WSS value compared to the 3-mean clustering solution (1239.568) suggests that the clusters in the 5-mean solution are more compact, indicating that the data points within each cluster are more tightly grouped around their centroids. This can be seen as a positive outcome, indicating improved clustering results in terms of compactness.

Average BCubed precision and recall

```
> BCubed_metric(sr, km$cluster, 0.5)
[1] 0.4236021
>
```

Figure 14

the obtained BCubed value of 0.4236021 suggests that there is scope for improvement in terms of the clustering method's agreement with the reference labels. This indicates that the clustering algorithm may not accurately capture the underlying number of clusters k.

Optimal number of clusters

Elbow Method:

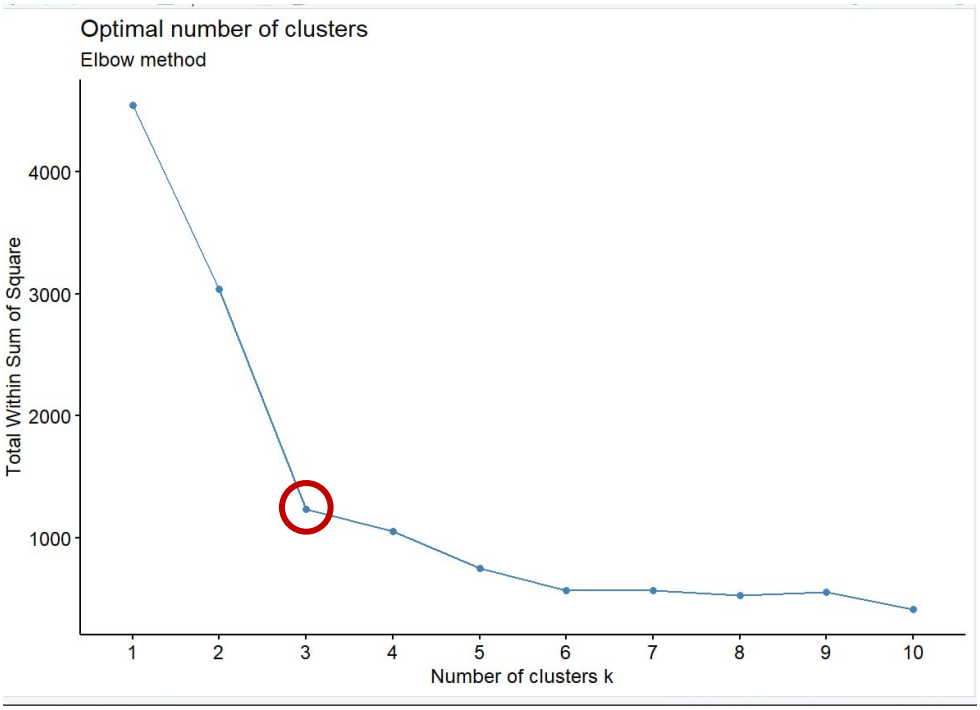


Figure 15

Average silhouette method:

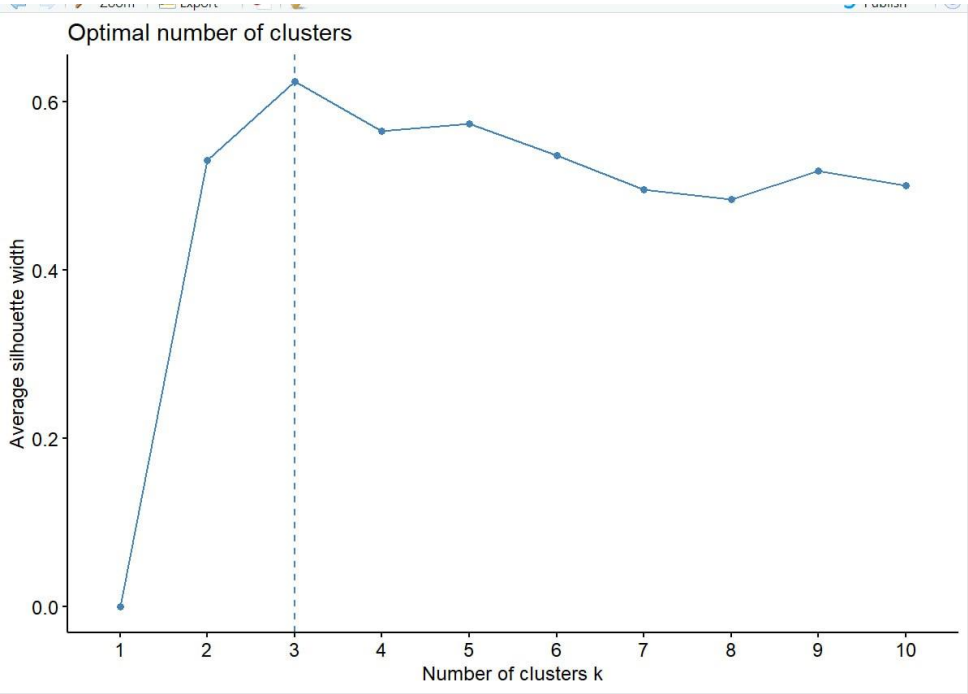


Figure 16

The average silhouette approach determines how well each object lies within its cluster. A high average silhouette width (k=3) indicates a good clustering.

Packages:

- ggplot2: is a plotting package that provides helpful commands to create complex plots from data in a data frame.
- factoextra: flexible and easy-to-use methods to extract quickly, in a human readable standard data format.
- DPBBM: Beta-binomial Mixture Model is used to infer the pattern from count data. It can be used for clustering of RNA methylation sequencing data.

Libraires:

- ggplot2: the grammar of graphics.
- factoextra: to visualize the cluster.
- cluster: to use silhouette method.

Methods:

- as.numeric(): transformed into numeric types before clustering.
- as.integer(): helps return their values as integer objects.

- rownames(): to set rownames in the data frame
- Kmeans(): run kmeans clustering to find N clusters.
- fviz_cluster(): visualization of the clusters.
- Silhouette(): calculate the average for each cluster.
- fviz_silhouette(): visualization of clusters Silhouette and average Silhouette.
- fviz_nbclust(): finding and visualization the best number of clusters.
- BCubed_metric(): F metric parameter which used to **average** precision and recall.

Although BCubed_metric() gives average values of recall and precision, we decided to choose to use this method because it works with our dataset and gives results that we can infer about the validity of the clustering.

-

Mining task	Comparison Criteria
Classification	
Clustering	

K-means			
Number of clusters	**K=2**	**K=3**	**K=5**
Average silhouette width for each cluster	0.53	0.62	0.59
total within-cluster sum of square	2521.881	1239.568	671.4406
Average BCubed precision and recall	0.5986005	0.5209976	0.4236021
Visualization	***Figure 1***	***Figure 6***	***Figure 10***

Finding:

Classification:

At the outset, our team carefully selected a dataset that represents valuable information about patients. Our goal was to utilize this data to predict the probability of individuals experiencing a stroke. By doing so, we aimed to provide people with the necessary knowledge and preventive measures to improve their overall well-being.

To ensure accurate and reliable results, we applied various preprocessing techniques to refine the dataset. These techniques helped us enhance the efficiency of the data and prepare it for analysis. Additionally, we employed several plotting methods to visually explore the dataset, allowing us to gain a deeper understanding of its characteristics and determine the most appropriate preprocessing steps.

Based on our observations from the plots and utilizing other relevant commands, we took steps to address any issues such as missing or outlier values. We removed these problematic instances from the dataset to prevent them from negatively impacting the accuracy of our predictions. Furthermore, we performed data transformation, which involved normalizing and discretizing certain attributes. This process aimed to ensure that all attributes carried equal weight and simplified data handling during subsequent data mining tasks.

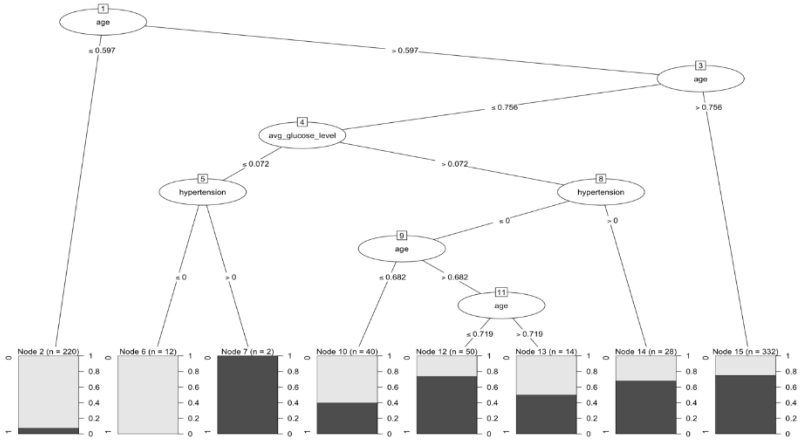
Through these efforts, we strived to create an efficient and reliable predictive model that could effectively assist individuals in taking proactive measures to lead healthier lives.

Following the preprocessing stage, we proceeded to apply various methods, including the Gini index, gain ratio, and information gain, using different partitioning techniques. We carefully evaluated the outcomes of each method to determine the most suitable approach for our specific dataset.

After thorough analysis, we determined that the gain ratio method with (70% training set and 30% testing set) yielded the most accurate and reliable results for our dataset. Therefore, we confidently selected it as the optimal method for constructing a decision tree.

As discussed, Gain ratio is considered superior to information gain in certain scenarios as it addresses the potential bias introduced by attributes with high cardinality or numerous distinct values. Information gain measures the reduction in uncertainty achieved by splitting data based on an attribute, but it tends to favor attributes with more distinct values or partitions, which can introduce more information.

In contrast, the gain ratio normalizes the information gain by considering the intrinsic information of the attribute. It takes into account the number of distinct values an attribute can have, penalizing attributes with high cardinality. By doing so, gain ratio provides a fairer comparison among attributes, considering both their information gain and the partitions they create.



Decision tree by gain ratio with 70% training and 30% testing.

The decision tree model yielded the following metrics:

- Accuracy: The accuracy of the model is 78.48%.
- Precision: The precision of the model is 76.65%.
- Sensitivity: The sensitivity (or recall) of the model is 83.12%.
- Specificity: The specificity of the model is 73.65%.

From the analysis of the decision tree, several findings can be observed:

- Number of Leaf Nodes: The decision tree consists of 8 leaf nodes, indicating that 8 rules or distinct conditions have been extracted from the tree.

- Important Attributes: The decision nodes in the tree utilize the attributes of age, average glucose, and hypertension to make decisions and classify instances.

- Age and Stroke: The analysis reveals that older individuals are more likely to have a stroke. This suggests that age is an important factor in determining the risk of stroke.

- Hypertension and Stroke: The tree also indicates that individuals with higher levels of hypertension are more prone to suffering from a stroke. This implies that hypertension is a significant contributing factor to stroke risk.

- Combined Impact: Furthermore, the findings suggest that individuals who have both hypertension and higher average glucose levels may be particularly susceptible to experiencing a stroke.

These insights provide valuable information about the relationships between specific attributes and the occurrence of strokes. They can be utilized to raise awareness, inform preventive measures, and guide healthcare interventions for individuals at higher risk of stroke.

Clustering:

For Clustering, we used K-means algorithm with 3 different K to find the optimal number of clusters, we calculated the average silhouette width for each K, and we concluded the following results:

- Number of cluster(K)= 2, the average silhouette width=0.53
- Number of cluster(K)= 3, the average silhouette width=0.62
- Number of cluster(K)= 5, the average silhouette width=0.59

Based on the analysis of the clustering results, it is evident that the choice of the number of clusters significantly impacts the quality of the clustering solution. The 2-mean clustering approach aligns better with the nature of our dataset, which includes class labels indicating the presence or absence of a stroke (0 for no stroke and 1 for stroke). Therefore, the optimal number of clusters for this dataset is 2, despite the 3-mean clustering solution appearing visually better.

The class labels in our dataset indicate that a binary classification problem is at hand, separating instances into two distinct categories: stroke and no stroke. Since the class labels suggest the existence of two clusters, it is more appropriate to choose a 2-mean clustering solution to accurately capture these distinct patterns.

However, when evaluating the clustering results using metrics such as silhouette analysis, within-cluster sum of squares (WSS), and BCubed precision and recall, the 3-mean clustering solution may appear to have better performance. This discrepancy can be attributed to factors such as the inherent complexity of the data or the influence of other variables not directly captured by the class labels.

While the 3-mean clustering solution visually appears superior, it is crucial to prioritize the alignment with the class labels and the nature of the problem at hand. Therefore, based on the stroke classification problem and the binary class labels, the 2-mean clustering solution is deemed better in this scenario.

It is worth noting that the clustering algorithm's ability to identify three clusters might be influenced by other factors present in the dataset that are not explicitly captured by the class labels. These factors could introduce additional complexity and patterns that the clustering

algorithm attempts to capture by suggesting three clusters. However, for the specific problem of stroke prediction, our numbers choose 30-mean clustering but the binary nature of the class labels suggests that a 2-mean clustering solution is more appropriate and aligned with the desired outcome.

Code

To review the code used in the data mining , please refer to the following files:

- Jupyter Notebook: This notebook contains code for data representation and preprocessing. It includes steps for data visualization, manipulation, cleaning, and any necessary preprocessing tasks.
- R Notebook: The R notebook focuses on classification and clustering tasks. It contains code for building and evaluating classification models, as well as performing clustering analysis on the data.

By examining these files, you will be able to gain a comprehensive understanding of the code implementation and the specific steps taken in the data analysis process.

References

- https://www.geeksforgeeks.org/conditional-inference-trees-in-r-programming/
- https://www.gormanalysis.com/blog/decision-trees-in-r-using-rpart/
- http://mercury.webster.edu/aleshunas/R_learning_infrastructure/Classification%20of%20data%20using%20decision%20tree%20and%20regression%20tree%20methods.html
- https://www.datacamp.com/tutorial/decision-trees-R
- https://medium.com/mlearning-ai/cluster-analysis-6757d6c6acc9
- https://www.geeksforgeeks.org/data-mining-cluster-analysis/