**Comprehensive Discussion of the Analytical Process on Stroke Data (Including Results)**

Discussion & Analysis by Yoyci I. Medina Villanueva

Professor Kourosh Dadgar

BUS 315 Data Mining for Business

October 6th, 2023

1. Introduction:

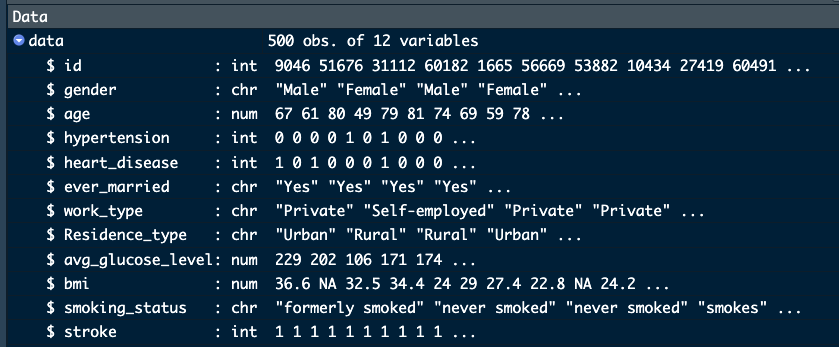
The purpose of this dataset under examination pertains to a demographic of patients and encompasses several attributes, potentially serving as indicators or risk factors for the occurrence of a stroke. These attributes include patient information, health history, and lifestyle factors.

2. Dataset Exploration:

Loading library packages & “stroke\_data.csv”:



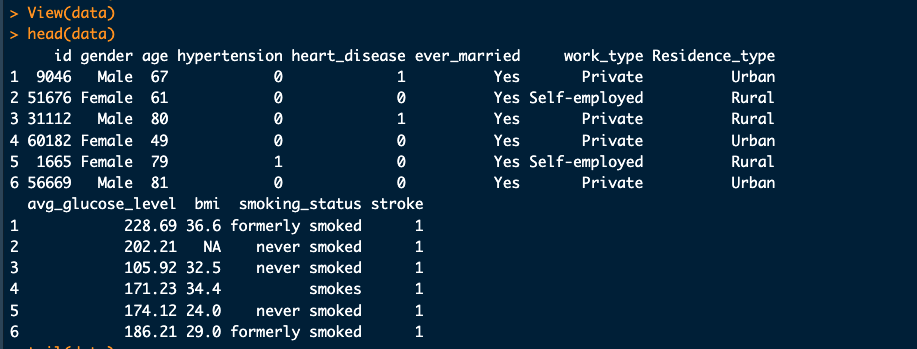
In this section of the analysis process, it is important to familiarize oneself with the dataset. This process initially involves identifying the data types, missing values, descriptive statistics, distribution of variables, correlation analysis, and investigative visualization. I began by loading the necessary packages, loading those library packages, and then lastly loading "stroke\_data.csv”.



* Dataset size: 500
* Attributes: 12 (ID, gender, age, hypertension, heart disease, marital status, work type, residence type, average glucose level, BMI, smoking status, stroke occurrence).

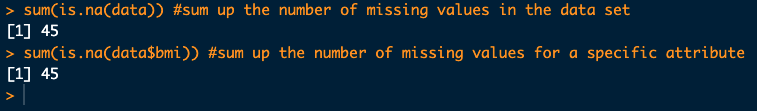
Investigation of data:

In the next step of the process, I viewed the dataset using the functions view(), head(), and tail(). These functions revealed a general overview of the data, ranging from the first couple of rows to the last couple of rows.



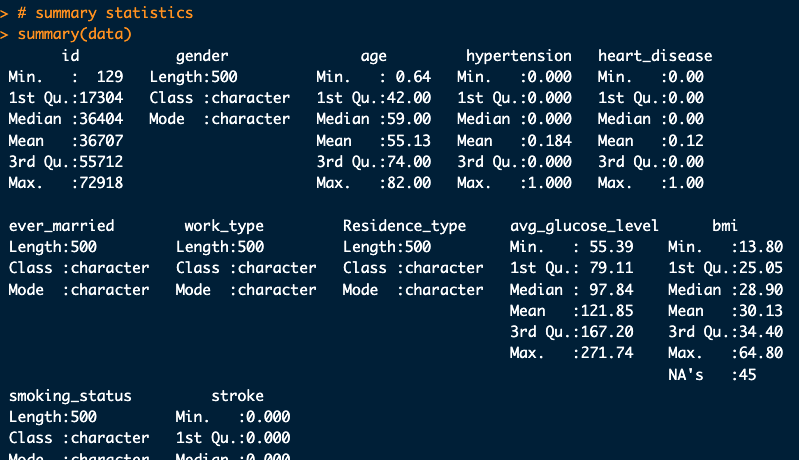


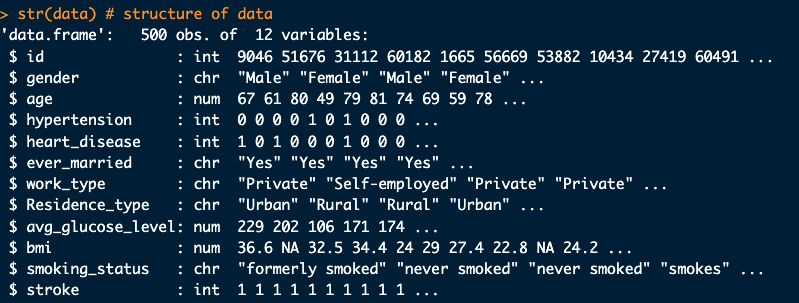
Following this step, I checked for any missing values, as in the first couple of rows it is evident that the bmi column had an N/A value. It is important to determine whether the dataset needs to be cleaned. I accomplished this task by using the “is.na(data)” function. The function showed that there were positive results for missing values in the bmi column. A further in depth view into the bmi column revealed that there were multiple missing values all within the bmi attribute.



* Missing Values: There were 45 missing values exclusively in the `bmi` attribute. This represents 9% of the data.

After identifying the missing values in the dataset, I moved on to view a complete summary of the data, the structure of the data, and its descriptive statistics. I believe further insight was necessary, as the functions used for this section would further my insights on the dataset.



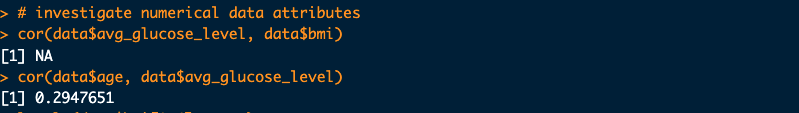


Results:

* Age: ranges from 0.64 to 82 years.
* Hypertension: Avg. 18.4% of the patients have hypertension.
* Heart Disease: Avg. 12% of the patients have heart disease.
* Average Glucose Level: Varies between 55.39 - 271.74, and Avg. 97.84.
* BMI: 45 missing values, ranging from 13.8 to 64.8, and Avg. 30.13.
* Stroke: Avg. 49.8% of the patients in the dataset have had a stroke.
* Data types (attributes): int(5), chr(5), and num(2).

Correlation analysis:

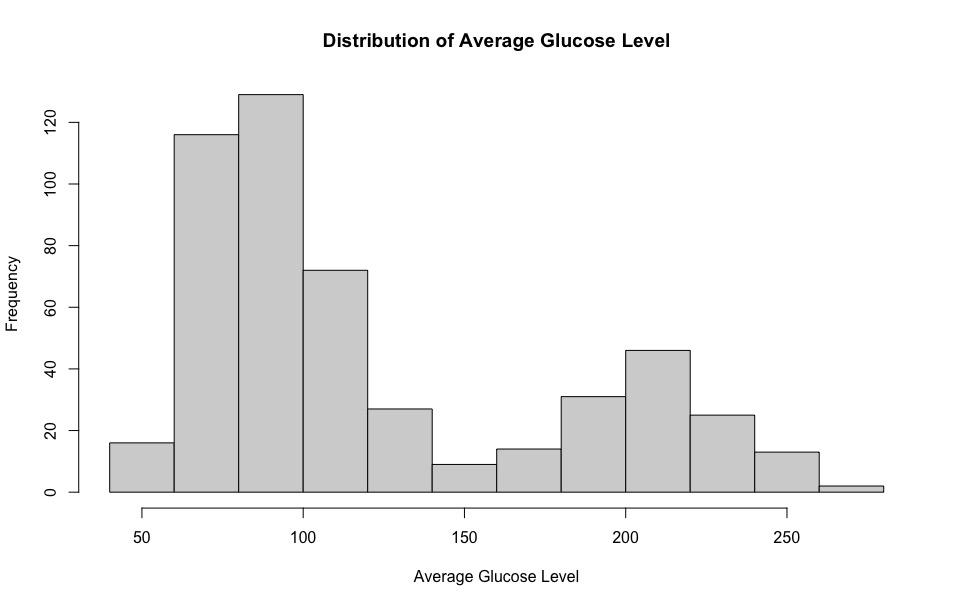
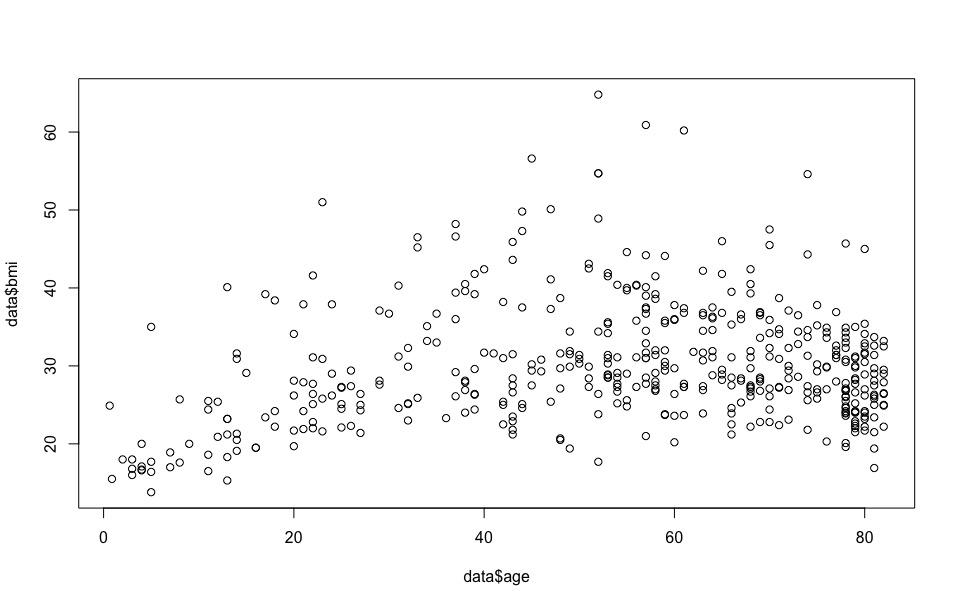
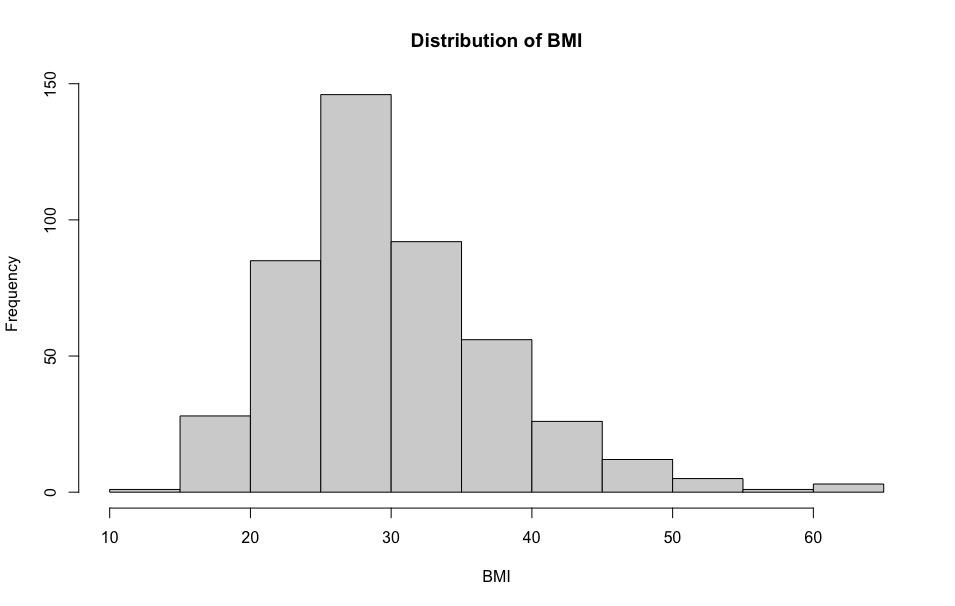
Continuing my investigation, I took a closer look at the numerical attributes for any insightful correlations between them. Correlation is important as it can reveal insights into numerical attributes that may be useful in real world applications in the medical field when conducting predictive analysis for patient symptoms. In addition, visualization can help understand those correlations.

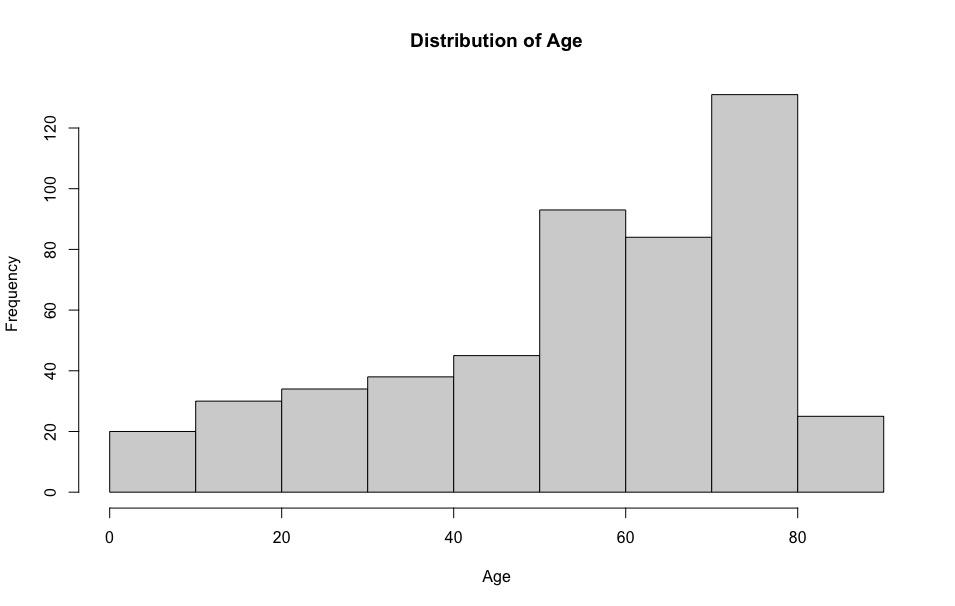


Results:

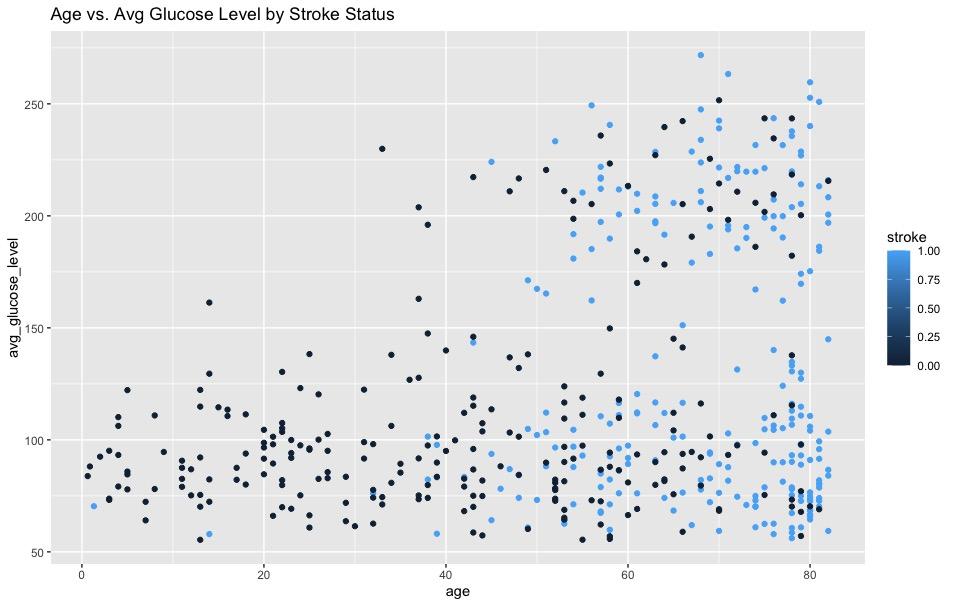
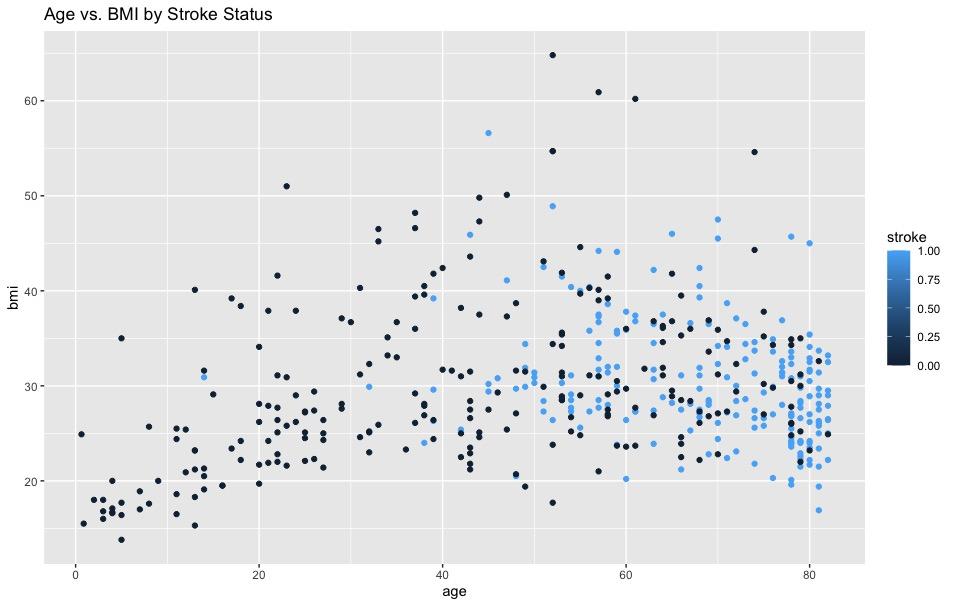
* There's no correlation between 'avg. glucose level' and 'BMI’ since there are missing values in the 'bmi' column.
* There's a positive correlation of approximately 0.2948 between 'age' and 'avg. glucose level'.

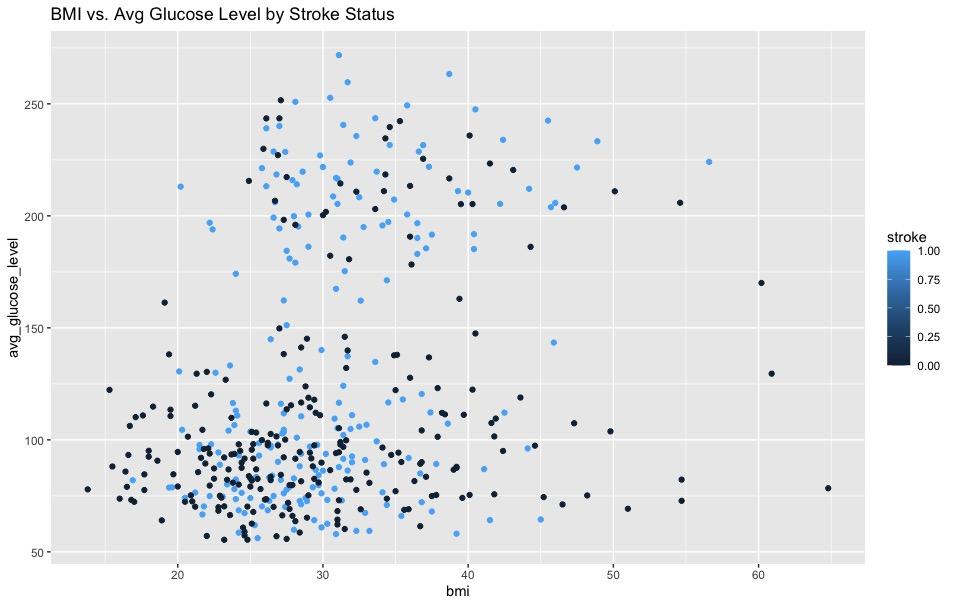
Visualization of numeric attributes:





* The distribution of ‘age' was visualized using a histogram. The histogram showed the age distribution of individuals in the dataset. Additionally, the data is skewed right, with more frequency between 50 - 80 years of age.
* A scatter plot was created between 'age' and 'avg. glucose level' to visualize any potential patterns or relationships. Further visual analysis using ggplot2
* Histograms were also plotted for 'avg. glucose level' and 'BMI' to understand their distributions.
  + Most patients within the dataset have a BMI between 25 - 30.
  + Most patients have an avg. glucose level between 50 - 120.

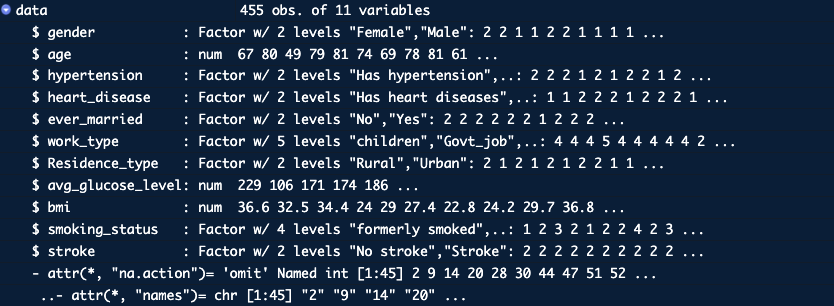




* Using ggplot2, I created scatter plots to show the relationship between (1: Stroke, 0: No stroke):
  + Age and avg. glucose level categorized by stroke status
    - Patients in the age range of 50 - 80 years with a BMI between 20 - 40 have a higher likelihood of suffering a stroke
  + Age and BMI are categorized by stroke status.
    - Patients in the age range of 75 -80 years with an avg. glucose level of 50 - 250 has a higher likelihood of suffering a stroke
  + BMI and avg. glucose level categorized by stroke status
    - Patients with a BMI of 20 - 40 with an avg. glucose level of 50 - 200 has a higher likelihood of suffering a stroke.

3. Preprocessing:

The preprocessing stage begins the initial step in the analysis that calls for cleaning up the data discrepancies. The preprocessing stage is critical, as it is the stage where the dataset's quality is established to extract accurate insights and results. I began by focusing on removing data that does not impact the analysis results; in this situation, the ‘id’ column meets the criteria as it holds data that acts as a unique identifier for each individual patient in the dataset. Secondly, after removing the ‘id’ attribute, I progressed to handling the missing values discovered in the investigation, 45 ‘bmi’ missing values using the ‘na.omit(data) function. I then moved on to the data transformation part of the cleaning process. To begin, the ‘bmi’ attribute data type was classified incorrectly and needed to be transformed into a numerical value. Secondly, to make the results more interpretable, I changed the descriptive class labels for the ‘hypertension’, 'heart\_disease’, and ‘stroke’ attributes from binary values of (1 and 0) to ("No hypertension", 'Has hypertension'), ("No heart diseases", 'Has heart diseases') (“Stroke”, ‘No Stroke’), as it makes presenting the analysis results more straightforward. To complete the cleaning process, I converted attributes to their appropriate data types if required.



Data Cleaning:

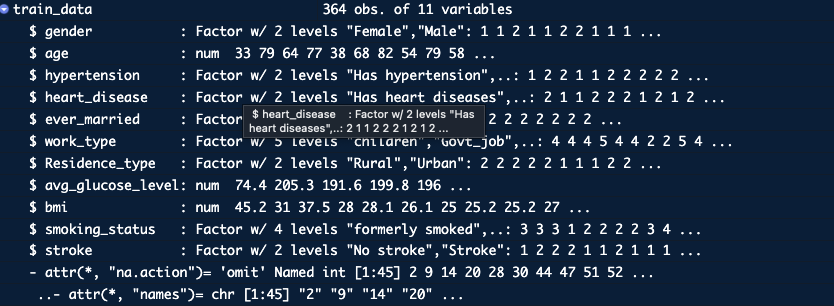
* ID Removal: The `id` column, which serves as a unique identifier for each individual, was dropped from the dataset as it's non-informative for the analysis.
* Handling missing values: These rows with missing values have been omitted using na.omit(data).
* Class labels for ‘hypertension’, 'heart\_disease’, and ‘stroke’ attribute‘stroke’ were converted.
* Data Transformation: Converted attributes to appropriate data types, including gender, hypertension, heart disease, marital status, work type, residence type, and smoking status, were converted to factor data types to facilitate analysis. BMI was converted to numerical.

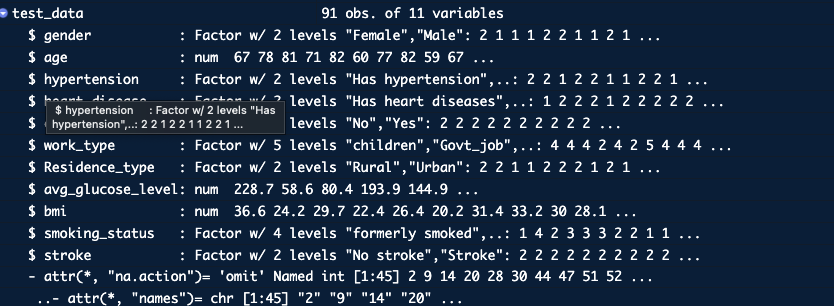
4. Creating Training & Testing Datasets

At this point of the process, the data is ready for classification model building but before the models are built I need to create a trained dataset and a testing dataset from the stroke\_data. Both the training and testing sets will play important roles in the modeling building process. The training set will be employed to train the model using the attributes in order to learn insights into their relationships with one another and the target variables. The testing set will then play its role in evaluating the performance of the trained model so that there is strong confidence that the model can make accurate predictions on the unseen data. The whole objective of this step of the process is to guarantee that the models are learning and not memorizing the data.

Splitting the Data:

To begin, I split the dataset into a training and testing set using random sampling using the ‘set.seed(123)’ function to ensure that the sequence of random numbers generated is the same every time my script is run. I then implemented the ‘train\_index <- sample(1:nrow(data), .8\*nrow(data)’ to perform the random sampling. The function then creates a sequence of numbers beginning with 1 to the number of rows in the data frame. The second portion of the function then calculates 80% of the total number of rows in the dataset. Resulting in the ‘train\_index’ that will store the randomly selected rows, the size of the sample of 80% of the total number of rows in the strokes dataset. The next step is to separate the target variable for testing, in this case ‘stroke\_test’ in order to evaluate the model's predictions so that the actual outcomes of stroke occurrences are extracted and stored in ‘stroke\_test’. This step is important as it gives insightful information by comparing the predicted values and the actual outcomes, ensuring an accurate measurement.





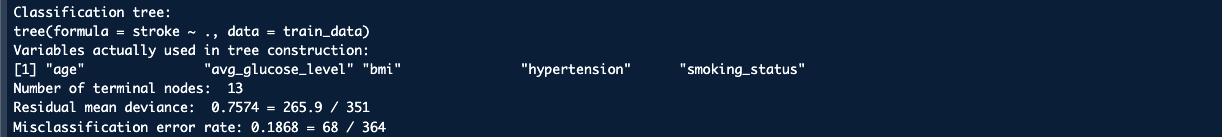
5. Build classification mode: train, test, cross validate, prune, use rainforest

Decision Tree Classification:

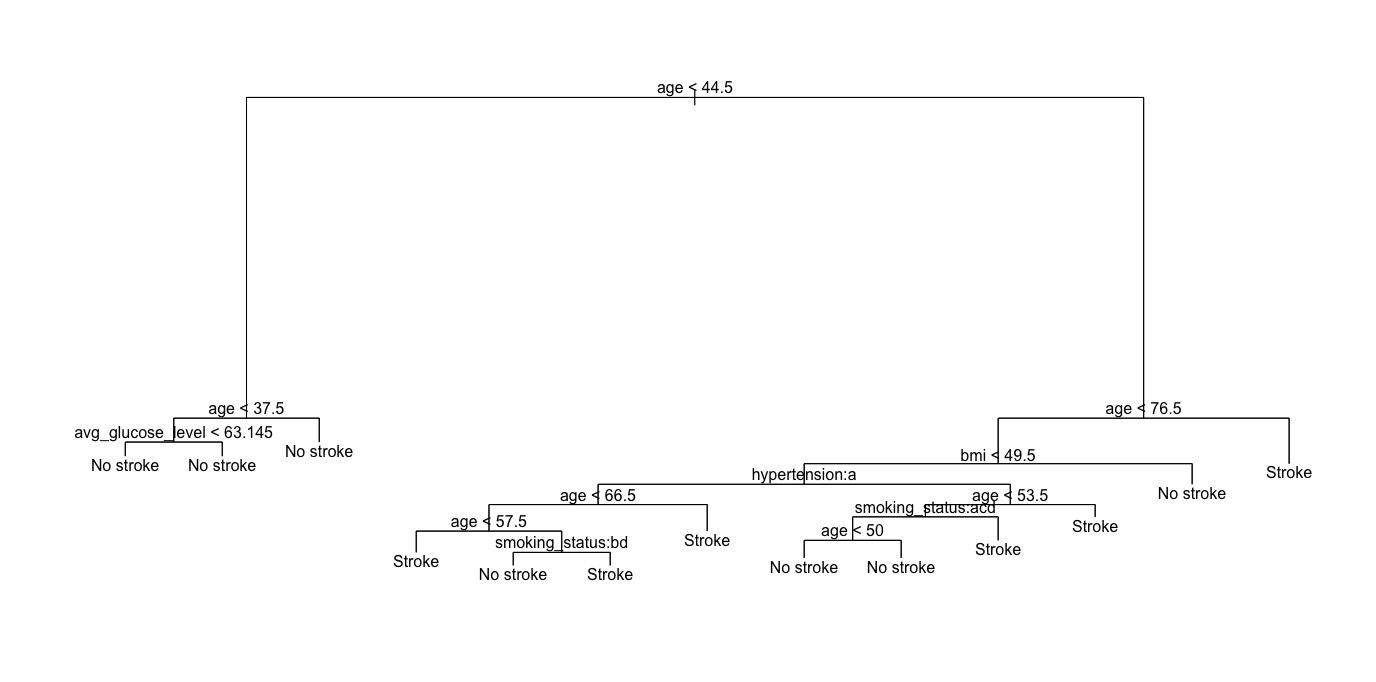
Training:

A decision tree was built using the tree function to predict if a person has experienced a stroke or not. The variables that were used as predictors to determine the stroke outcome were;

* ‘age’
* ‘avg. glucose\_level
* ‘bmi’
* ‘hypertension’
* ‘Smoking\_status
* The tree had a total of 13 terminal nodes.
* The residual mean deviance, which measures how well the model fits the data was determined to be 0.7574. (75.7%) This value was obtained by dividing 265.9 by 351. A lower residual mean deviance implies an alignment, with the data.
* The error rate, for misclassifying the tree was 18.68%. This implies that out of the training data, around 18.68% of the instances were classified wrongly by the model.



Visualization:



Testing:

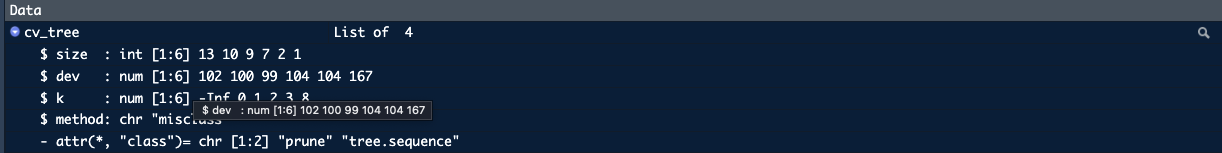
Once I trained the decision tree model using the training dataset it becomes crucial to evaluate its performance, on the data. This step ensures that the model not only performs well on the training data but also demonstrates generalization, to new data points. The accuracy achieved by the decision tree model on the test data was 72.53%. This indicates that the model correctly identified whether a patient had a stroke or not in, around 72.53% of cases. Decision trees are particularly beneficial when it comes to explaining results to for example; stakeholders without a technical background or expertise because they are easy to visualize and comprehend.



Pruning the Decision Tree:

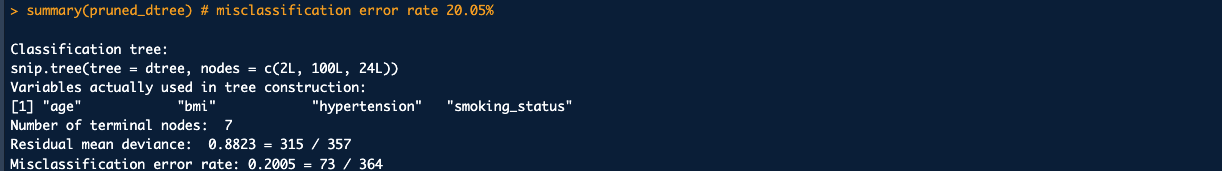
Once the initial decision tree model was trained and tested it became clear that there could be ways to enhance its performance. One approach is called "pruning," which involves reducing the depth of the tree by removing branches that may be causing the model to overfit the training data. Overfitting happens when a model becomes overly complex capturing details from the training data of focusing on the underlying patterns. This can lead to a decrease, in its ability to generalize well on data.

Cross Validation:

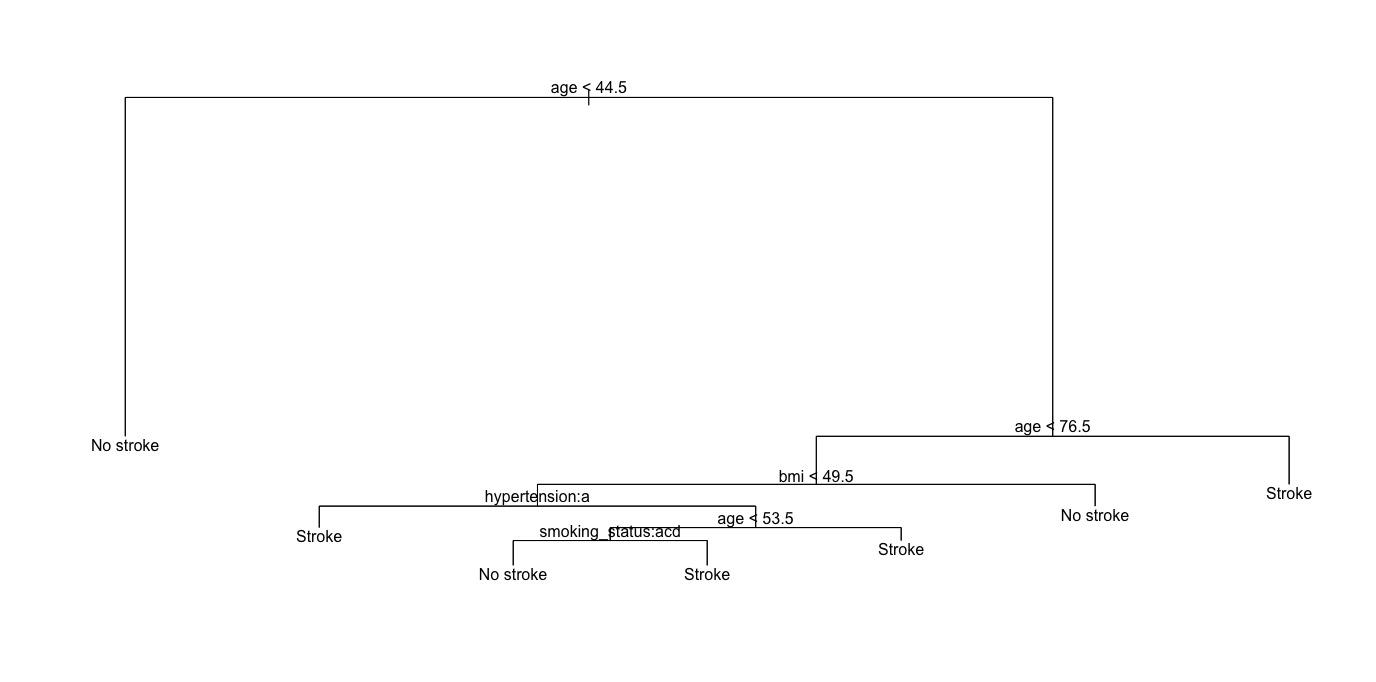


After pruning, the tree used the following variables:

* Age
* BMI (Body Mass Index)
* Hypertension status
* Smoking Status
* This pruned tree had 7 terminal nodes, fewer than the initial tree. The misclassification error rate for this pruned tree, when evaluated on the training data, was 20.05%. This means that the pruned model showed an indication that 20.05% of the training data was classified incorrectly.

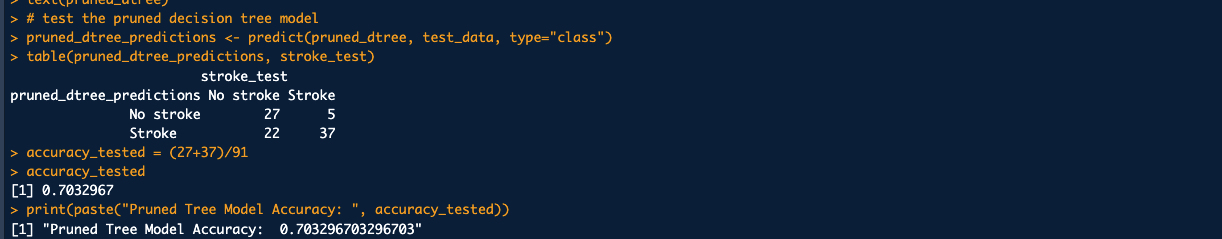


Visualization:



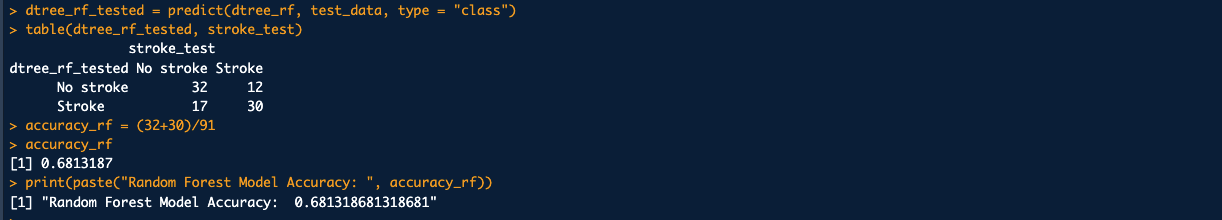
Testing:

Upon testing the pruned tree on the test data, the following results were observed in the confusion matrix. The model accurately predicted 27 cases where individuals did not have a stroke and 37 cases where individuals did have a stroke. However it mistakenly classified 5 stroke cases as non stroke. 22 non stroke cases, as stroke. Overall, the model managed to attain a 70.33% accuracy, on the test data. Although this is slightly lower compared to the accuracy of the tree (72.53%) the pruned tree is now simpler with nodes. This simplification can enhance interpretability. Reduce susceptibility to overfitting even though there was a compromise, in its accuracy.



Random Forest Model:

To potentially improve the prediction accuracy and leverage the power of ensemble learning, I ran a Random Forest classification model on the data. Random Forest classification model combines decision trees in order to make predictions. During training it considers the majority class of each tree, for classification tasks. The goal behind using trees is to capture the patterns and variations, in the data, which can ultimately enhance accuracy and mitigate overfitting issues. The model correctly predicted 32 cases where there was no stroke and 30 cases where there was a stroke. However it made mistakes by classifying 12 instances of strokes as non strokes and misclassifying 17 non stroke instances as strokes. The test data showed that the Random Forest model had an accuracy rate of 68.13%. Although its slightly lower, than both the pruned decision tree models the Random Forest has advantages when it comes to handling datasets and being more robust. The small decrease in accuracy can be attributed to the fact that this model's better, at generalizing to data by averaging out multiple decision trees, which helps reduce variability.



To sum up the results of the models built; the decision tree model resulted an accuracy of 72.53% when tested. When I pruned the tree to prevent overfitting the accuracy slightly dropped to 70.33%. The Random Forest model performed the least accurate, achieved an accuracy of 68.13%.