A Stroking Analysis of Strokes

Team 4: Andrew Kim, Benjamin Earnest, and Hunter Blum

Shiley-Marcos School of Engineering, University of San Diego

**Abstract**

Strokes are a common cause of death and disability across the world, costing billions in the United States every year (Murphy and Werring, 2020, p.1; Centers for Disease Control and Prevention 2022). Our project focused on constructing a sensitive and accurate model to identify what patients are most likely to have a stroke to capitalize on the success of early prevention techniques. The data was collected from Kaggle.com and consisted of 12 attributes describing traditional stroke risk factors (Palacios, 2021). After reading and cleaning the data in R, the cases of stroke were balanced using random duplicate oversampling (R Core Team, 2021). Each feature in the data was found to be important through the Boruta algorithm. 15 models were constructed using a variety of methods and mainly evaluated using accuracy and sensitivity. Based on sensitivity, a cost-sensitive CART with balanced data performed the best with a 97.6% sensitivity and 51.6%. However, an artificial neural network created with balanced data performed better overall, with a 74.4% accuracy and 90.2% sensitivity. Healthcare providers should note that the recommended artificial neural network has a low precision at only 13.0%. These models do not predict when a patient may have a stroke, they identify patients that are at a much higher risk of having a stroke. Therefore, this model should be utilized as a screener to find patients for further stroke testing, rather than a recommendation for prescribing medication or immediate, invasive stroke treatment measures.

# Predicting Stroke Risk Factors

# Caused by vascular injuries in the brain, strokes are the second leading cause of death and disability in the world (Murphy and Werring, 2020, p. 1). In the United States, somebody dies of a stroke every 3.5 minutes resulting in $53 billion in costs annually (Centers for Disease Control and Prevention, 2022). However, prevention strategies can have a massive impact, with current prevention techniques mitigating up to 80% of strokes (Pandian et al., 2018). A major part of beginning prevention is identifying patients who are at risk of stroke. Current model accuracies for predicting strokes in patients can range from 88% to 97% (Singh and Choudhary, 2017; Emon et al., 2020). This project considered existing data based on patient risk factors, identified the key predictor variables, and evaluated model efficacy. The goal was to identify which modeling algorithm provided the most accurate and sensitive predictions of strokes in patients when a tested with novel data.

# Methodology

# Our goal was to develop a model to predict strokes in patients, utilizing a secondary data analysis. We utilized R to read in a data set from Kaggle.com, consisting of 5,110 records and 12 different attributes (R Core Team, 2021; Palacios, 2021). The arbitrary “id” attribute was removed from the data and attributes were converted to the correct type. 201 observations were missing BMI values and were subsequently removed from the data.

# To explore the data, we created a regular and normalized bar plot of each categorical attribute overlaid with the target stroke attribute. When normalized the presence of heart disease or hypertension exhibited the largest relationship with having a stroke (Figure 2.1). For numeric attributes a regular and normalized histogram was coded. After normalizing, age had a clear, direct relationship with a stroke occurring in patients (Figure 2.2).

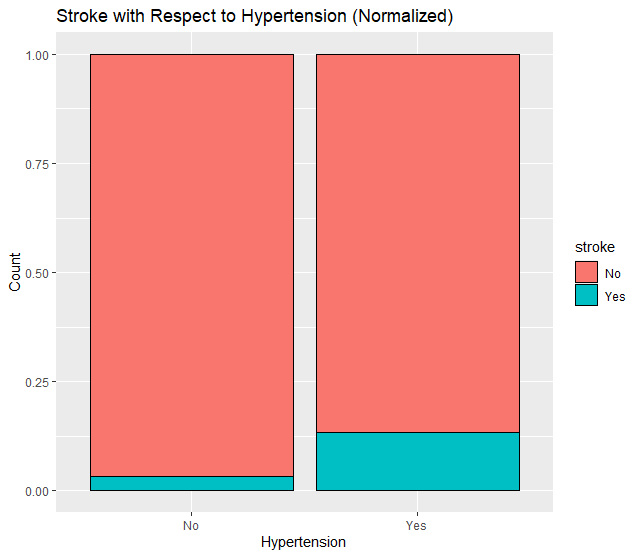
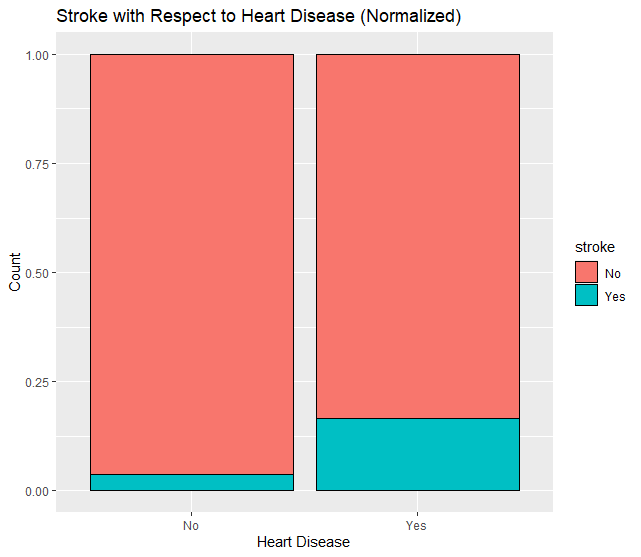


Figure 2.1. Normalized bar plots of hypertension (left) and heart disease (right) overlaid by stroke (blue indicates a stroke occurred).

Chart, bar chart, histogram

Description automatically generated

Figure 2.2. Normalized histogram of age overlaid by stroke (blue indicates a stroke occurred).

# The data was split into a training and testing dataset, making up 80% and 20% of the data respectively. The training data had a large imbalance for the target stroke attribute, with over 95% of observations not having a stroke. The stroke class was balanced in the training data using random duplicate oversampling, making each class of stroke equal. We selected important explanatory attributes by using the Boruta algorithm on the balanced data. Each variable had a much higher importance than the shadow data, so every attribute was considered in modelling (Figure 2.3).

Chart, waterfall chart

Description automatically generated

Figure 2.3. Output of Boruta on balanced data. Explanatory attributes are in green and the shadow data are in blue.

The attributes considered for model development were gender, age, hypertension, heart disease, marital status, occupation, residence type, average glucose level, BMI, and smoking status. A variety of models will be evaluated on a many metrics, particularly accuracy and sensitivity.

# C5.0

# The C5.0 model utilized all 11 variables, which generated multiple decision nodes. Figure 3.1 below features the C5.0 decision tree output from the balanced data set. Based on the model, the “age” variable serves as the root node, which splits and determines if a patient is above or not above the age of a certain value. In this case, the first age split occurs at 44. The decision tree indicates that if a patient is less than or equal to the age of 44, it proceeds to determine the patients’ occupation. If a patient is greater than the age of 44, the decision tree proceeds to another age value of 66, which determines the patients’ BMI and average glucose level. Despite the decision tree and the variables being packed tightly, the tree includes 66 nodes and a data frame was created that included the predictor variables of the records to classify (Figure 3.1). When diagnosing the model, the C5.0 model was shown to generate a 78% accuracy with a sensitivity rate of 54% and a specificity rate of 80%.

A picture containing chart

Description automatically generated

Figure 3.1. C5.0 decision tree predicting strokes in patients.

# CART

The CART model also utilized all 11 variables. However, the CART algorithm found the root node to be age, followed by avg\_glucose\_level, bmi, and work\_type as the decision nodes. This is similar to the C5.0 model. The CART model generated 9 leaf nodes in total. The root node considered age greater than 45 as an initial decision point, with only 28% being below age 45, and low risk of having a stroke. The remaining 72% were then split for those above or below age 67. The model considered anyone over the age of 67 to be at risk of a stroke, suggesting age is a leading factor in determine stroke risk. There were 35% of this group between the ages of 45 and 67, that would need other factors considered to determine stroke risk by the CART model. For this age group, the model considered avg\_glucose\_level, bmi, and if a person was Self-employed. Interestingly, the model found that a person who was between ages 45 and 67, had a bmi greater than 32, and is Self-employed represented a stroke risk (Figure 4.1). When evaluating the model on test data, we found it to have an accuracy of 80%, sensitivity of 89%, and specificity of 70%. This suggests the model generalized well and has the potential for future predictive model use.



Figure 4.1. Output of CART model.

To achieve maximum sensitivity, a cost-sensitive CART model was trained utilizing the balanced data and “rpart” method in caret. The cost-matrix considered a false-negative to be ten times as costly as a false-positive. When evaluated with the test data, the highest accuracy model produced an accuracy of 51.6% and a sensitivity of 97.6%.

# Logistic Regression

Given the binary nature of our target variable and many of the predictor variables, a logistic regression model was considered and evaluated. To do this, the non-binary continuous variables were standardized. This included age, bmi, and avg\_glucose\_level. Once standardized, an initial run of the logistic regression model was run to look for statistically significant predictor variables. To avoid issues from an imbalanced data set, the balanced, standardized training data set was used. The model found that gender, hypertension, heart\_disease, smoking\_status, age, and avg\_glucose\_level were statistically significant, or had p-values < 0.05 (Equation 5.1). The algorithm was then rerun to validate statistical significance of the variables and generate the logistic regression equation for our model.

Equation 5.1. Equation for final linear regression.

The test data was run through the model to evaluate performance. The results suggest the model does not predict well based on the variables used, as each test observation was predicted to have a stroke, or greater than a 50% chance of having a stroke.

# Random Forest

Two random forest models were trained utilizing the “rf” method in the caret package. The first model utilized the unbalanced data, while taking into account the class probabilities of the stroke. The final model had an accuracy of 95.8% on the test data. However, the unbalanced model had a sensitivity of zero. In hopes of improving the sensitivity, another random forest model was produced using the balanced data set. However, this model resulted in the same output, resulting in an accuracy of 95.8% and a sensitivity of zero.

# Naïve Bayes

# Several Naïve Bayes models were created and evaluated utilizing different predictor variables. The first model that evaluated “stroke” in association with “gender” and “hypertension” generated the highest accuracy rating of 88% and the highest specificity rating of 91%. This indicates that evaluating the gender type and hypertension level of each patient are leading indicators of a patient to have a stroke. The second model that evaluated “stroke” in association with “heart disease” and “ever married” generated the highest sensitivity rating at 88%.

# Neural Network

Four neural network models were trained using the “nnet” method in the caret package and ten-fold cross-validation. Each model began with 15 inputs, representing every attribute and class level from the data. Two models were made based on the unbalanced data, one with the raw data and the other with z-score standardized numeric attributes. Both models resulted in all-negative models, producing 95.8% accuracy and zero sensitivity when used with the test set. The third model utilized balanced data. This model considered seven nodes in its hidden layer (Figure 8.1). When run on the test data set, the balanced neural network had an accuracy of 78.7% and an 80.5% sensitivity. Finally, a fourth model was made, this time using the balanced data with z-score standardized numeric attributes. The fourth model considered only four nodes in the hidden layer (Figure 8.1). This model had a lower accuracy at just 61.3%, but a higher sensitivity at 87.8%.

Diagram

Description automatically generated

Figure 8.1. Results of each neural network. Unbalanced and unstandardized on the top left, unbalanced and standardized on the top right, balanced and unstandardized on the bottom left, balanced and standardized on the bottom right.

# Results

# The models were evaluated with accuracy, sensitivity, specificity, precision, and F1 score against a baseline all-negative model. The unbalanced artificial neural network (ANN Reg.), unbalanced and standardized artificial neural network (ANN Z), and unbalanced random forest (RF Reg.) all produced all-negative predictions, equaling the Baseline model in every measure (Table 9.1). The balanced random forest (RF Bal.) model generated the second-highest accuracy rate of approximately 95%, however, the model also did not predict any patients that actually had a stroke correctly. The strongest model according to sensitivity was the cost-sensitive CART (CART Cost Bal.), which generated the highest sensitivity at 97.6%. The highest precision rating was found through Logistic Regression at 75%, which indicated the highest proportion of predicted strokes to actual strokes. The balanced artificial neural had the highest F1 score at 23%.

**Table 9.1**

*Model Evaluation Table for Stroke Prediction*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Sensitivity** | **Specificity** | **Precision** | **F1** |
| **Baseline** | 0.958 | 0.000 | 1.000 | 0.000 | 0.000 |
| **Log Reg.** | 0.042 | 1.000 | 0.000 | 0.418 | 0.080 |
| **ANN Reg.** | 0.958 | 0.000 | 1.000 | 0.000 | 0.000 |
| **ANN Bal.** | 0.744 | 0.902 | 0.737 | 0.130 | 0.228 |
| **ANN Z** | 0.958 | 0.000 | 1.000 | 0.000 | 0.000 |
| **ANN Z Bal.** | 0.613 | 0.878 | 0.602 | 0.088 | 0.160 |
| **RF Reg.** | 0.958 | 0.000 | 1.000 | 0.000 | 0.000 |
| **RF Bal.** | 0.948 | 0.000 | 0.989 | 0.000 | 0.000 |
| **CART Bal.** | 0.700 | 0.902 | 0.691 | 0.113 | 0.201 |
| **CART Cost** | 0.845 | 0.512 | 0.860 | 0.138 | 0.218 |
| **CART Cost Bal.** | 0.516 | 0.976 | 0.496 | 0.078 | 0.144 |
| **C5.0 Bal.** | 0.785 | 0.537 | 0.796 | 0.103 | 0.173 |
| **NB Gender** | 0.885 | 0.268 | 0.912 | 0.117 | 0.163 |
| **NB Heart + Marry** | 0.376 | 0.878 | 0.354 | 0.056 | 0.105 |
| **NB Resident** | 0.505 | 0.512 | 0.505 | 0.0432 | 0.080 |
| **NB Smoke + Work** | 0.330 | 0.854 | 0.308 | 0.051 | 0.096 |

Chart, bar chart

Description automatically generated

Figure 9.2. Accuracy (left) and sensitivity (right) of each model. Colors are based on the performance against the baseline, with grey being equal, red being worse, and green being better.

# Conclusion

Our goal was to find a model with high accuracy and sensitivity to predict if a patient would have a stroke. There are a few models that we would recommend based on the circumstances. If we were solely focused on capturing the most people who would actually have a stroke, we would recommend the balanced, cost-sensitive CART (CART Cost Bal.) This model had the highest sensitivity, however, it also struggled to correctly predict patients who would not have a stroke, resulting in a poor accuracy (Figure 9.2). If we were focused on a more balanced model, the balanced artificial neural network (ANN Bal.) would be our recommendation. This model still had a high sensitivity at 90.2% and had a much higher accuracy than the CART Cost Bal. at 74.4%. Choosing one, we would recommend the ANN Bal. because of its better accuracy. However, the model has a low precision at just 13%, meaning that 87% of patients predicted as having a stroke by this model did not have a stroke. Healthcare providers should take this into account if using this model to suggest any medications with potentially harmful side effects. The model does serve as a good screener to capture all patients that will have a stroke, so healthcare providers can use this to narrow down patients for more advanced stroke risk evaluation.

# References

Centers for Disease Control and Prevention. (2022, April 5). Stroke facts. Centers for Disease Control and Prevention. Retrieved April 13, 2022, from <https://www.cdc.gov/stroke/facts.htm>

Emon, M. U., Keya, M. S., Meghla, T. I., Rahman, M. M., Al Mamun, M. S., & Kaiser, M. S. (2020, November). Performance analysis of machine learning approaches in stroke prediction. In 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA) (pp. 1464-1469). IEEE.

Markus, H. (2008). Stroke: causes and clinical features. Medicine. 36(11). Elsevier Ltd.

Murphy, S. J., & Werring, D. J. (2020). Stroke: causes and clinical features. Medicine, 48(9), 561-566.

Palacios, F. S. (2021, January 26). Stroke prediction dataset. Kaggle. Retrieved April 14, 2022, from <https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset>

Pandian, J. D., Gall, S. L., Kate, M. P., Silva, G. S., Akinyemi, R. O., Ovbiagele, B. I., … & Thrift, A. G. (2018). Prevention of stroke: a global perspective. The Lancet, 392(10154), 1269-1278.

R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.

Singh, M. S., & Choudhary, P. (2017, August). Stroke prediction using artificial intelligence. In 2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON) (pp. 158-161). IEEE.

**Appendix**

R Code Utilized for Project

A Stroking Analysis of Strokes

Hunter Blum, Ben Earnest, Andrew Pak Kim

4/12/2022

Dataset:

<https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset>

# Libraries

library(randomForest)

## Warning: package 'randomForest' was built under R version 4.1.3

## randomForest 4.7-1

## Type rfNews() to see new features/changes/bug fixes.

library(caret)

## Loading required package: ggplot2

##   
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':  
##   
## margin

## Loading required package: lattice

library(NeuralNetTools)

## Warning: package 'NeuralNetTools' was built under R version 4.1.3

library(nnet)

## Warning: package 'nnet' was built under R version 4.1.3

library(doParallel)

## Warning: package 'doParallel' was built under R version 4.1.3

## Loading required package: foreach

## Loading required package: iterators

## Loading required package: parallel

library(DataExplorer)

## Warning: package 'DataExplorer' was built under R version 4.1.3

library(Boruta)

## Warning: package 'Boruta' was built under R version 4.1.3

library(rpart)  
library(rpart.plot)  
library(gridExtra)

##   
## Attaching package: 'gridExtra'

## The following object is masked from 'package:randomForest':  
##   
## combine

library(e1071)

## Warning: package 'e1071' was built under R version 4.1.3

library(kableExtra)

## Warning: package 'kableExtra' was built under R version 4.1.3

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble 3.1.6 v dplyr 1.0.7  
## v tidyr 1.2.0 v stringr 1.4.0  
## v readr 2.1.2 v forcats 0.5.1  
## v purrr 0.3.4

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x purrr::accumulate() masks foreach::accumulate()  
## x dplyr::combine() masks gridExtra::combine(), randomForest::combine()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::group\_rows() masks kableExtra::group\_rows()  
## x dplyr::lag() masks stats::lag()  
## x purrr::lift() masks caret::lift()  
## x ggplot2::margin() masks randomForest::margin()  
## x purrr::when() masks foreach::when()

set.seed(123)

# Data

Stroke <- read.csv("stroke.csv")

# Multicore Support - Need to have Java Installed that is the same bit as your CPU (Probably 64)

registerDoParallel()  
getDoParWorkers()

## [1] 3

# Cleaning the Data

The data is pretty clean after running this code. All variables are the correct type after running it. The only variable missing data is BMI, which only has 200/5100 observations missing.

## Structure

str(Stroke)

## 'data.frame': 5110 obs. of 12 variables:  
## $ id : int 9046 51676 31112 60182 1665 56669 53882 10434 27419 60491 ...  
## $ gender : chr "Male" "Female" "Male" "Female" ...  
## $ age : num 67 61 80 49 79 81 74 69 59 78 ...  
## $ hypertension : int 0 0 0 0 1 0 1 0 0 0 ...  
## $ heart\_disease : int 1 0 1 0 0 0 1 0 0 0 ...  
## $ ever\_married : chr "Yes" "Yes" "Yes" "Yes" ...  
## $ work\_type : chr "Private" "Self-employed" "Private" "Private" ...  
## $ Residence\_type : chr "Urban" "Rural" "Rural" "Urban" ...  
## $ avg\_glucose\_level: num 229 202 106 171 174 ...  
## $ bmi : chr "36.6" "N/A" "32.5" "34.4" ...  
## $ smoking\_status : chr "formerly smoked" "never smoked" "never smoked" "smokes" ...  
## $ stroke : int 1 1 1 1 1 1 1 1 1 1 ...

#Get rid of one other observation in gender  
Stroke <- Stroke %>% filter(gender!="Other")  
  
#Fix specific variables  
Stroke$hypertension <- as.factor(Stroke$hypertension)  
Stroke$heart\_disease <- as.factor(Stroke$heart\_disease)  
Stroke$bmi <- as.numeric(Stroke$bmi)

## Warning: NAs introduced by coercion

Stroke$stroke <- as.factor(Stroke$stroke)  
  
#Make all character variables into factors  
Stroke[sapply(Stroke, is.character)] <- lapply(Stroke[sapply(Stroke, is.character)], as.factor)  
  
  
str(Stroke)

## 'data.frame': 5109 obs. of 12 variables:  
## $ id : int 9046 51676 31112 60182 1665 56669 53882 10434 27419 60491 ...  
## $ gender : Factor w/ 2 levels "Female","Male": 2 1 2 1 1 2 2 1 1 1 ...  
## $ age : num 67 61 80 49 79 81 74 69 59 78 ...  
## $ hypertension : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 2 1 1 1 ...  
## $ heart\_disease : Factor w/ 2 levels "0","1": 2 1 2 1 1 1 2 1 1 1 ...  
## $ ever\_married : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 1 2 2 ...  
## $ work\_type : Factor w/ 5 levels "children","Govt\_job",..: 4 5 4 4 5 4 4 4 4 4 ...  
## $ Residence\_type : Factor w/ 2 levels "Rural","Urban": 2 1 1 2 1 2 1 2 1 2 ...  
## $ avg\_glucose\_level: num 229 202 106 171 174 ...  
## $ bmi : num 36.6 NA 32.5 34.4 24 29 27.4 22.8 NA 24.2 ...  
## $ smoking\_status : Factor w/ 4 levels "formerly smoked",..: 1 2 2 3 2 1 2 2 4 4 ...  
## $ stroke : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...

#Rename Factors for Easier Understanding  
levels(Stroke$hypertension) <- c("No", "Yes")  
levels(Stroke$heart\_disease) <- c("No", "Yes")  
levels(Stroke$stroke) <- c("No", "Yes")  
  
#Get Rid of id  
Stroke$id <- NULL

## NAs

Stroke %>%   
 select(everything()) %>%   
 summarise\_all(funs(sum(is.na(.))))

## Warning: `funs()` was deprecated in dplyr 0.8.0.  
## Please use a list of either functions or lambdas:   
##   
## # Simple named list:   
## list(mean = mean, median = median)  
##   
## # Auto named with `tibble::lst()`:   
## tibble::lst(mean, median)  
##   
## # Using lambdas  
## list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was generated.

## gender age hypertension heart\_disease ever\_married work\_type Residence\_type  
## 1 0 0 0 0 0 0 0  
## avg\_glucose\_level bmi smoking\_status stroke  
## 1 0 201 0 0

#Just 201 missing observations in bmi   
  
#We'll just delete the NAs for now  
Stroke\_clean <- na.omit(Stroke)

# Exploratory Data Analysis

Our target feature is stroke, where 1 indicates that a stroke occurred. For any binary attributes 1 is always the variable occurred (eg. 1 for heart disease means the patient had heart disease).

## Dataset Overview

summary(Stroke\_clean)

## gender age hypertension heart\_disease ever\_married  
## Female:2897 Min. : 0.08 No :4457 No :4665 No :1704   
## Male :2011 1st Qu.:25.00 Yes: 451 Yes: 243 Yes:3204   
## Median :44.00   
## Mean :42.87   
## 3rd Qu.:60.00   
## Max. :82.00   
## work\_type Residence\_type avg\_glucose\_level bmi   
## children : 671 Rural:2418 Min. : 55.12 Min. :10.30   
## Govt\_job : 630 Urban:2490 1st Qu.: 77.07 1st Qu.:23.50   
## Never\_worked : 22 Median : 91.68 Median :28.10   
## Private :2810 Mean :105.30 Mean :28.89   
## Self-employed: 775 3rd Qu.:113.50 3rd Qu.:33.10   
## Max. :271.74 Max. :97.60   
## smoking\_status stroke   
## formerly smoked: 836 No :4699   
## never smoked :1852 Yes: 209   
## smokes : 737   
## Unknown :1483   
##   
##

head(Stroke\_clean)

## gender age hypertension heart\_disease ever\_married work\_type  
## 1 Male 67 No Yes Yes Private  
## 3 Male 80 No Yes Yes Private  
## 4 Female 49 No No Yes Private  
## 5 Female 79 Yes No Yes Self-employed  
## 6 Male 81 No No Yes Private  
## 7 Male 74 Yes Yes Yes Private  
## Residence\_type avg\_glucose\_level bmi smoking\_status stroke  
## 1 Urban 228.69 36.6 formerly smoked Yes  
## 3 Rural 105.92 32.5 never smoked Yes  
## 4 Urban 171.23 34.4 smokes Yes  
## 5 Rural 174.12 24.0 never smoked Yes  
## 6 Urban 186.21 29.0 formerly smoked Yes  
## 7 Rural 70.09 27.4 never smoked Yes

## Variable by Stroke

### Make Functions

#Categorical  
Cat\_eda <- function(x, y) {  
 p1 <- ggplot(Stroke\_clean, aes(x={{x}})) +  
 geom\_bar(aes(fill=stroke) , color = "black") +  
 ggtitle(paste0("Stroke with Respect to ", y)) +  
 xlab(y) + ylab("Count")  
  
 p2 <- ggplot(Stroke\_clean, aes(x={{x}})) +  
 geom\_bar(aes(fill=stroke), position = "fill", color = "black") + ggtitle(paste0("Stroke with Respect to ",y, " (Normalized)")) + xlab(y) + ylab("Count")  
   
 plot(p1)  
 plot(p2)  
}  
  
#Numeric  
Num\_eda <- function(x, y) {  
 p1 <- ggplot(Stroke\_clean, aes(x={{x}})) +  
 geom\_histogram(aes(fill=stroke), color = "black") +  
 ggtitle(paste0("Stroke with Respect to ", y)) +  
 xlab(y) + ylab("Count")  
  
 p2 <- ggplot(Stroke\_clean, aes(x={{x}})) +   
 geom\_histogram(aes(fill=stroke), color = "black", position = "fill") +  
 ggtitle(paste0("Stroke with Respect to ", y, " (Normalized)")) +  
 xlab(y) + ylab("Count")  
   
 plot(p1)  
 plot(p2)  
   
}

### Categorical Variables

Cat\_eda(gender, "Gender")

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

Cat\_eda(hypertension, "Hypertension")

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

Cat\_eda(heart\_disease, "Heart Disease")

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

Cat\_eda(ever\_married, "Ever Married")

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

Cat\_eda(work\_type, "Work Type")

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

Cat\_eda(Residence\_type, "Residence Type")

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

Cat\_eda(smoking\_status, "Smoking Status")

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

### Numeric variables

Num\_eda(age, "Age")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

Chart, histogram

Description automatically generated

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

Chart, bar chart, histogram

Description automatically generated

Num\_eda(avg\_glucose\_level, "Avg. Glucose Level")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

Chart, histogram

Description automatically generated

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

Chart, bar chart, histogram

Description automatically generated

Num\_eda(bmi, "BMI")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

Chart, histogram

Description automatically generated

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 12 rows containing missing values (geom\_bar).

Chart, bar chart

Description automatically generated

### Correlation Matrix

plot\_correlation(Stroke)

Text

Description automatically generated

# Data Preparation for modelling

## Partition Data

#Partition Data  
trainIndex <- createDataPartition(y=Stroke\_clean$stroke, p=0.8, list = F, times = 1)  
  
Stroke\_tr <- Stroke\_clean[trainIndex, ]  
Stroke\_test <- Stroke\_clean[-trainIndex, ]

### Visualize Stroke Balance in Training

#Dual Axis  
ggplot(Stroke\_tr, aes(x=stroke)) +   
 geom\_bar(fill = "#eb746c") +  
 scale\_y\_continuous(  
 name = "Count",  
 sec.axis = sec\_axis(~./nrow(Stroke\_tr), name = "Proportion")  
 ) + ggtitle("Training Data Pre-balancing")

Chart, bar chart

Description automatically generated

### Oversampling

#Get count of yes in training  
minority <- Stroke\_tr %>% group\_by(stroke) %>% tally() %>% filter(stroke =="Yes")  
  
#Change this to change balance to desired yes proportion  
increase\_to <- 0.5  
  
#Calculate resample amouunt  
oversample\_n <- (increase\_to\*nrow(Stroke\_tr)-minority$n)/(1-increase\_to)  
  
#Resample  
to\_oversample <- which(Stroke\_tr$stroke == "Yes")  
our\_oversample <- sample(x = to\_oversample, size = oversample\_n, replace = T)  
our\_oversample <- Stroke\_tr[our\_oversample, ]  
Stroke\_over <- rbind(Stroke\_tr, our\_oversample)  
  
#Evaluate  
ggplot(Stroke\_over, aes(x=stroke)) +   
 geom\_bar(fill = "#72cf69") +  
 scale\_y\_continuous(  
 name = "Count",  
 sec.axis = sec\_axis(~./nrow(Stroke\_over), name = "Proportion")  
 ) + ggtitle("Oversampled Data")

Chart, bar chart

Description automatically generated

## Standardization

### Min-Max Standardization Function - Use standard.df() to create your own data set for model if you feel standardization is necessary.

#Function to Standardize One Variable  
standard.mm <- function(x){  
 (x - min(x)) / (max(x) - min(x))  
}  
  
#Function to Standardize all Numeric Variables in Data Frame  
standard.mm.df <- function(x){  
 #Split Data  
 tr\_num <- x %>% select(where(is.numeric))  
 tr\_non <- x %>% select(!where(is.numeric))  
   
 #Run Standardization Function Across Numeric  
 tr\_num\_mm <- apply(X = tr\_num, FUN = standard.mm, MARGIN = 2)  
   
 #Recombine  
 tr\_mm <- cbind(tr\_non, tr\_num\_mm)  
}

### Z-Score Standardization

#Z-Score Function  
standard.z <- function(x){  
 (x-mean(x))/sd(x)  
}  
  
#Function to Standardize all Numeric Variables in Data Frame  
standard.z.df <- function(x){  
 #Split Data  
 tr\_num <- x %>% select(where(is.numeric))  
 tr\_non <- x %>% select(!where(is.numeric))  
   
 #Run Standardization Function Across Numeric  
 tr\_num\_mm <- apply(X = tr\_num, FUN = standard.z, MARGIN = 2)  
   
 #Recombine  
 tr\_mm <- cbind(tr\_non, tr\_num\_mm)  
}

### Feature Selection

# Run the boruta  
Stroke\_over\_z <- standard.z.df(Stroke\_over)  
  
boruta\_out <- Boruta(stroke ~ ., data = Stroke\_over\_z, doTrace = 2)

## 1. run of importance source...

## 2. run of importance source...

## 3. run of importance source...

## 4. run of importance source...

## 5. run of importance source...

## 6. run of importance source...

## 7. run of importance source...

## 8. run of importance source...

## 9. run of importance source...

## 10. run of importance source...

## After 10 iterations, +16 secs:

## confirmed 10 attributes: age, avg\_glucose\_level, bmi, ever\_married, gender and 5 more;

## no more attributes left.

boruta\_sig <- getSelectedAttributes(boruta\_out, withTentative = T)  
  
print(boruta\_sig)

## [1] "gender" "hypertension" "heart\_disease"   
## [4] "ever\_married" "work\_type" "Residence\_type"   
## [7] "smoking\_status" "age" "avg\_glucose\_level"  
## [10] "bmi"

imps <- attStats(boruta\_out)  
imps2 = imps[imps$decision != 'Rejected', c('meanImp', 'decision')]  
imps2[order(-imps2$meanImp), ]

## meanImp decision  
## age 121.07528 Confirmed  
## avg\_glucose\_level 91.64216 Confirmed  
## bmi 84.51178 Confirmed  
## smoking\_status 59.33260 Confirmed  
## work\_type 57.09723 Confirmed  
## hypertension 53.04735 Confirmed  
## heart\_disease 47.30415 Confirmed  
## Residence\_type 38.55393 Confirmed  
## ever\_married 36.90999 Confirmed  
## gender 36.01437 Confirmed

plot(boruta\_out, cex.axis=.55, las=2, xlab="", main="Variable Importance")

Chart, waterfall chart

Description automatically generated

# All Variables were deemed important

# Modeling

## C5.0 - Andrew

library(C50)  
C5 <- C5.0(formula = stroke ~ . , data = Stroke\_over, control = C5.0Control(minCases = 75))

#Visualize the tree  
plot(C5)

Diagram, schematic

Description automatically generated

#Create a data frame that includes the predictor variables of the records to classify.  
X = Stroke\_over %>% select(!stroke)

#Obtain model diagnostics  
C5\_bal <- confusionMatrix(data = predict(C5, Stroke\_test), ref = Stroke\_test$stroke, positive = "Yes")  
C5\_bal

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 747 19  
## Yes 192 22  
##   
## Accuracy : 0.7847   
## 95% CI : (0.7576, 0.8101)  
## No Information Rate : 0.9582   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.1101   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.53659   
## Specificity : 0.79553   
## Pos Pred Value : 0.10280   
## Neg Pred Value : 0.97520   
## Prevalence : 0.04184   
## Detection Rate : 0.02245   
## Detection Prevalence : 0.21837   
## Balanced Accuracy : 0.66606   
##   
## 'Positive' Class : Yes   
##

## CART - Ben

### Run CART decision tree model:

cart01 <- rpart(formula = stroke ~ ., data = Stroke\_over, method = "class")  
rpart.plot(cart01)

Diagram

Description automatically generated

### Evaluate Model on Train and Test Data

train\_cart <- confusionMatrix(data = predict(cart01, Stroke\_over, type = "class"), ref = Stroke\_over$stroke, positive = "Yes")  
train\_cart

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 2660 408  
## Yes 1100 3352  
##   
## Accuracy : 0.7995   
## 95% CI : (0.7902, 0.8085)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5989   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.8915   
## Specificity : 0.7074   
## Pos Pred Value : 0.7529   
## Neg Pred Value : 0.8670   
## Prevalence : 0.5000   
## Detection Rate : 0.4457   
## Detection Prevalence : 0.5920   
## Balanced Accuracy : 0.7995   
##   
## 'Positive' Class : Yes   
##

cart\_bal <- confusionMatrix(data = predict(cart01, Stroke\_test, type = "class"), ref = Stroke\_test$stroke, positive = "Yes")  
cart\_bal

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 649 4  
## Yes 290 37  
##   
## Accuracy : 0.7   
## 95% CI : (0.6702, 0.7286)  
## No Information Rate : 0.9582   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.1369   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.90244   
## Specificity : 0.69116   
## Pos Pred Value : 0.11315   
## Neg Pred Value : 0.99387   
## Prevalence : 0.04184   
## Detection Rate : 0.03776   
## Detection Prevalence : 0.33367   
## Balanced Accuracy : 0.79680   
##   
## 'Positive' Class : Yes   
##

### Cost Sensitive CARET -

Probably only useful if we want to hyperfocus on sensitivity, but loses a lot of specificity.

cost <- matrix(c(  
 0, 1,  
 10, 0  
), byrow = TRUE, nrow = 2)  
cost

## [,1] [,2]  
## [1,] 0 1  
## [2,] 10 0

### Create cost sensitive model

train <- createFolds(Stroke\_tr$stroke, k=10)  
  
cart\_stroke <- caret::train(stroke~., method="rpart", data = Stroke\_tr, tuneLength = 5, parms = list(loss = cost),trControl = trainControl(  
 method = "cv", indexOut = train  
))  
  
cart\_stroke

## CART   
##   
## 3928 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 3536, 3535, 3535, 3535, 3535, 3535, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.0000000000 0.8971452 0.3854536  
## 0.0008267196 0.8951109 0.3804238  
## 0.0016534392 0.8956191 0.3802099  
## 0.0024801587 0.8943410 0.3792570  
## 0.0033068783 0.8984149 0.3785458  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.003306878.

cart\_stroke\_bal <- caret::train(stroke~., method="rpart", data = Stroke\_over, tuneLength = 5, parms = list(loss = cost),trControl = trainControl(  
 method = "cv", indexOut = train  
))  
  
cart\_stroke\_bal

## CART   
##   
## 7520 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 6768, 6768, 6768, 6768, 6768, 6768, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.007978723 0.56365737 0.09491299  
## 0.011436170 0.52649231 0.08103147  
## 0.013696809 0.51783183 0.07830490  
## 0.018882979 0.51528730 0.07762580  
## 0.484574468 0.04276886 0.00000000  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.007978723.

### Evaluate

cart\_cost <- confusionMatrix(data = predict(cart\_stroke, Stroke\_test), ref = Stroke\_test$stroke, positive = "Yes")  
cart\_cost

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 808 20  
## Yes 131 21  
##   
## Accuracy : 0.8459   
## 95% CI : (0.8218, 0.868)  
## No Information Rate : 0.9582   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.1624   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.51220   
## Specificity : 0.86049   
## Pos Pred Value : 0.13816   
## Neg Pred Value : 0.97585   
## Prevalence : 0.04184   
## Detection Rate : 0.02143   
## Detection Prevalence : 0.15510   
## Balanced Accuracy : 0.68634   
##   
## 'Positive' Class : Yes   
##

cart\_cost\_bal <- confusionMatrix(data = predict(cart\_stroke\_bal, Stroke\_test), ref = Stroke\_test$stroke, positive = "Yes")  
cart\_cost\_bal

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 466 1  
## Yes 473 40  
##   
## Accuracy : 0.5163   
## 95% CI : (0.4845, 0.548)  
## No Information Rate : 0.9582   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0725   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.97561   
## Specificity : 0.49627   
## Pos Pred Value : 0.07797   
## Neg Pred Value : 0.99786   
## Prevalence : 0.04184   
## Detection Rate : 0.04082   
## Detection Prevalence : 0.52347   
## Balanced Accuracy : 0.73594   
##   
## 'Positive' Class : Yes   
##

## Logistic Regression - Ben

#unbalanced  
Stroke\_tr\_z\_lr <- standard.z.df(Stroke\_tr)  
logreg\_stroke <- glm(formula = stroke ~ ., data = Stroke\_tr\_z\_lr, family = binomial)  
summary(logreg\_stroke)

##   
## Call:  
## glm(formula = stroke ~ ., family = binomial, data = Stroke\_tr\_z\_lr)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.2010 -0.2952 -0.1626 -0.0815 3.5082   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -3.700489 1.074733 -3.443 0.000575 \*\*\*  
## genderMale -0.028491 0.172729 -0.165 0.868985   
## hypertensionYes 0.629895 0.194744 3.234 0.001219 \*\*   
## heart\_diseaseYes 0.454585 0.230574 1.972 0.048662 \*   
## ever\_marriedYes 0.030847 0.287125 0.107 0.914445   
## work\_typeGovt\_job -0.455028 1.132155 -0.402 0.687748   
## work\_typeNever\_worked -10.927599 521.627746 -0.021 0.983286   
## work\_typePrivate -0.461907 1.118231 -0.413 0.679556   
## work\_typeSelf-employed -0.757860 1.140530 -0.664 0.506383   
## Residence\_typeUrban 0.003971 0.167234 0.024 0.981055   
## smoking\_statusnever smoked -0.052834 0.212144 -0.249 0.803322   
## smoking\_statussmokes 0.375812 0.254124 1.479 0.139180   
## smoking\_statusUnknown -0.310082 0.277384 -1.118 0.263618   
## age 1.457749 0.155625 9.367 < 2e-16 \*\*\*  
## avg\_glucose\_level 0.248710 0.063996 3.886 0.000102 \*\*\*  
## bmi -0.036858 0.103632 -0.356 0.722097   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1387.8 on 3927 degrees of freedom  
## Residual deviance: 1106.7 on 3912 degrees of freedom  
## AIC: 1138.7  
##   
## Number of Fisher Scoring iterations: 15

#balanced  
Stroke\_tr\_z\_bal\_lr <- standard.z.df(Stroke\_over)  
logreg01\_stroke <- glm(formula = stroke ~ ., data = Stroke\_tr\_z\_bal\_lr, family = binomial)  
summary(logreg01\_stroke)

##   
## Call:  
## glm(formula = stroke ~ ., family = binomial, data = Stroke\_tr\_z\_bal\_lr)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.5291 -0.7329 0.1129 0.7519 2.6018   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.17828 0.31070 0.574 0.56611   
## genderMale -0.16544 0.06114 -2.706 0.00682 \*\*   
## hypertensionYes 0.73662 0.07963 9.250 < 2e-16 \*\*\*  
## heart\_diseaseYes 0.43637 0.10281 4.244 2.19e-05 \*\*\*  
## ever\_marriedYes 0.09102 0.09499 0.958 0.33797   
## work\_typeGovt\_job -0.39750 0.31635 -1.257 0.20893   
## work\_typeNever\_worked -11.65478 191.15688 -0.061 0.95138   
## work\_typePrivate -0.41659 0.31145 -1.338 0.18103   
## work\_typeSelf-employed -0.61030 0.32231 -1.894 0.05829 .   
## Residence\_typeUrban 0.10418 0.05865 1.776 0.07567 .   
## smoking\_statusnever smoked -0.20604 0.07664 -2.688 0.00718 \*\*   
## smoking\_statussmokes 0.25144 0.09091 2.766 0.00568 \*\*   
## smoking\_statusUnknown -0.44780 0.09344 -4.792 1.65e-06 \*\*\*  
## age 1.53390 0.05057 30.331 < 2e-16 \*\*\*  
## avg\_glucose\_level 0.25093 0.03290 7.626 2.42e-14 \*\*\*  
## bmi 0.01968 0.03315 0.594 0.55275   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 10424.9 on 7519 degrees of freedom  
## Residual deviance: 7189.3 on 7504 degrees of freedom  
## AIC: 7221.3  
##   
## Number of Fisher Scoring iterations: 13

### Remove variables with a p-value < .05:

head(Stroke\_tr\_z\_bal\_lr)

## gender hypertension heart\_disease ever\_married work\_type Residence\_type  
## 1 Male No Yes Yes Private Urban  
## 3 Male No Yes Yes Private Rural  
## 5 Female Yes No Yes Self-employed Rural  
## 6 Male No No Yes Private Urban  
## 7 Male Yes Yes Yes Private Rural  
## 11 Female Yes No Yes Private Rural  
## smoking\_status stroke age avg\_glucose\_level bmi  
## 1 formerly smoked Yes 0.5770671 1.9093525 0.97141531  
## 3 never smoked Yes 1.1674606 -0.2580567 0.40689722  
## 5 never smoked Yes 1.1220457 0.9459614 -0.76344518  
## 6 formerly smoked Yes 1.2128755 1.1594010 -0.07500847  
## 7 never smoked Yes 0.8949713 -0.8906076 -0.29530822  
## 11 never smoked Yes 1.2128755 -0.7080629 0.02137266

### Retrain model

Stroke\_logreg\_df <- subset(Stroke\_tr\_z\_bal\_lr, select = c("gender", "hypertension", "heart\_disease", "smoking\_status", "age", "avg\_glucose\_level", "stroke"))  
  
logreg\_stroke\_subset <- glm(formula = stroke ~ ., data = Stroke\_logreg\_df, family = binomial)  
summary(logreg\_stroke\_subset)

##   
## Call:  
## glm(formula = stroke ~ ., family = binomial, data = Stroke\_logreg\_df)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.52959 -0.74160 0.05061 0.74884 2.68731   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.12931 0.06864 -1.884 0.05958 .   
## genderMale -0.15016 0.06076 -2.471 0.01346 \*   
## hypertensionYes 0.71806 0.07879 9.114 < 2e-16 \*\*\*  
## heart\_diseaseYes 0.45246 0.10226 4.425 9.66e-06 \*\*\*  
## smoking\_statusnever smoked -0.22301 0.07606 -2.932 0.00337 \*\*   
## smoking\_statussmokes 0.23934 0.09033 2.650 0.00806 \*\*   
## smoking\_statusUnknown -0.42955 0.09208 -4.665 3.09e-06 \*\*\*  
## age 1.49247 0.04353 34.283 < 2e-16 \*\*\*  
## avg\_glucose\_level 0.26549 0.03123 8.501 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 10424.9 on 7519 degrees of freedom  
## Residual deviance: 7203.5 on 7511 degrees of freedom  
## AIC: 7221.5  
##   
## Number of Fisher Scoring iterations: 5

### Compare the logreg predictions to the test dataset target variables.

#Create Test Subset  
Stroke\_logreg\_test <- Stroke\_test %>% select(gender, hypertension, heart\_disease, smoking\_status, age, avg\_glucose\_level, stroke)  
# prediction  
Stroke\_logreg\_test$pred\_prob <- predict(object = logreg\_stroke\_subset, newdata = Stroke\_logreg\_test, type='response')  
Stroke\_logreg\_test$pred <- (Stroke\_logreg\_test$pred\_prob > 0.5)\*1  
  
# Change pred variables to y/n  
Stroke\_logreg\_test$pred[Stroke\_logreg\_test$pred=="1"]<-"Yes"  
Stroke\_logreg\_test$pred[Stroke\_logreg\_test$pred=="0"]<-"No"  
Stroke\_logreg\_test$pred <- as.factor(Stroke\_logreg\_test$pred)

### Confusion Matrix and Metrics

LogReg <- confusionMatrix(data = Stroke\_logreg\_test$pred, ref = Stroke\_logreg\_test$stroke, positive = "Yes")

## Warning in confusionMatrix.default(data = Stroke\_logreg\_test$pred, ref =  
## Stroke\_logreg\_test$stroke, : Levels are not in the same order for reference and  
## data. Refactoring data to match.

LogReg

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 0 0  
## Yes 939 41  
##   
## Accuracy : 0.0418   
## 95% CI : (0.0302, 0.0563)  
## No Information Rate : 0.9582   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 1.00000   
## Specificity : 0.00000   
## Pos Pred Value : 0.04184   
## Neg Pred Value : NaN   
## Prevalence : 0.04184   
## Detection Rate : 0.04184   
## Detection Prevalence : 1.00000   
## Balanced Accuracy : 0.50000   
##   
## 'Positive' Class : Yes   
##

## Random Forest - Hunter

### Create Models

train <- createFolds(Stroke\_tr$stroke, k=10)  
  
rf\_stroke <- caret::train(stroke~., method="rf", data = Stroke\_tr, tuneLength = 5, trControl = trainControl(  
 method = "cv", indexOut = train, classProbs = TRUE  
))  
  
rf\_stroke

## Random Forest   
##   
## 3928 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 3535, 3535, 3535, 3535, 3536, 3535, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.9577400 0.02032707  
## 5 0.9961800 0.94974337  
## 8 0.9961800 0.94974337  
## 11 0.9959255 0.94678592  
## 15 0.9959255 0.94678592  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 5.

train <- createFolds(Stroke\_over\_z$stroke, k=10)  
  
rf\_stroke\_bal <- caret::train(stroke~., method="rf", data = Stroke\_over\_z, tuneLength = 5, trControl = trainControl(  
 method = "cv", indexOut = train  
))  
  
  
rf\_stroke\_bal

## Random Forest   
##   
## 7520 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 6768, 6768, 6768, 6768, 6768, 6768, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.9027926 0.8055851  
## 5 0.9990691 0.9981383  
## 8 0.9990691 0.9981383  
## 11 0.9985372 0.9970745  
## 15 0.9986702 0.9973404  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 5.

rf\_stroke\_bal$finalModel

##   
## Call:  
## randomForest(x = x, y = y, mtry = min(param$mtry, ncol(x)))   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 5  
##   
## OOB estimate of error rate: 0.51%  
## Confusion matrix:  
## No Yes class.error  
## No 3722 38 0.01010638  
## Yes 0 3760 0.00000000

### Confusion Matrices

rf\_reg <- confusionMatrix(data = predict(rf\_stroke, Stroke\_test), ref = Stroke\_test$stroke, positive = "Yes")  
rf\_reg

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 939 41  
## Yes 0 0  
##   
## Accuracy : 0.9582   
## 95% CI : (0.9437, 0.9698)  
## No Information Rate : 0.9582   
## P-Value [Acc > NIR] : 0.5414   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 4.185e-10   
##   
## Sensitivity : 0.00000   
## Specificity : 1.00000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.95816   
## Prevalence : 0.04184   
## Detection Rate : 0.00000   
## Detection Prevalence : 0.00000   
## Balanced Accuracy : 0.50000   
##   
## 'Positive' Class : Yes   
##

rf\_bal <- confusionMatrix(data = predict(rf\_stroke\_bal, Stroke\_test), ref = Stroke\_test$stroke, positive = "Yes")  
rf\_bal

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 939 41  
## Yes 0 0  
##   
## Accuracy : 0.9582   
## 95% CI : (0.9437, 0.9698)  
## No Information Rate : 0.9582   
## P-Value [Acc > NIR] : 0.5414   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 4.185e-10   
##   
## Sensitivity : 0.00000   
## Specificity : 1.00000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.95816   
## Prevalence : 0.04184   
## Detection Rate : 0.00000   
## Detection Prevalence : 0.00000   
## Balanced Accuracy : 0.50000   
##   
## 'Positive' Class : Yes   
##

## Naive Bayes - Andrew

### Create the tables that will allow calculation of necessary probabilities

#### Table Function

#For any individual dataset  
#x = data set, y = non stroke variable  
nb\_table <- function(x, y) {  
 gen <- table(x[,"stroke"], x[,y])  
 colnames(gen) <- levels(x[,y])  
 rownames(gen) <- c("stroke = Yes", "stroke = No")  
 names(dimnames(gen)) <- list(" ", y)  
 addmargins(A = gen, FUN = list(Total = sum), quiet = TRUE)  
}

#### First table is the contingency table of “stroke” and “gender”. The value 1 indicates “yes” while 0 indicates “no.”

nb\_table(Stroke\_over, "gender")

## gender  
## Female Male Total  
## stroke = Yes 2233 1527 3760  
## stroke = No 2167 1593 3760  
## Total 4400 3120 7520

#### Second table is the contingency table of “stroke” and “hypertension”. The value 1 indicates “yes” while 0 indicates “no.”

nb\_table(Stroke\_over, "hypertension")

## hypertension  
## No Yes Total  
## stroke = Yes 3452 308 3760  
## stroke = No 2701 1059 3760  
## Total 6153 1367 7520

#### Third table is the contingency table of “stroke” and “heart disease”. The value 1 indicates “yes” while 0 indicates “no.”

nb\_table(Stroke\_over, "heart\_disease")

## heart\_disease  
## No Yes Total  
## stroke = Yes 3600 160 3760  
## stroke = No 3005 755 3760  
## Total 6605 915 7520

#### Fourth table is the contingency table of “stroke” and “ever married”.

nb\_table(Stroke\_over, "ever\_married")

## ever\_married  
## No Yes Total  
## stroke = Yes 1344 2416 3760  
## stroke = No 405 3355 3760  
## Total 1749 5771 7520

#### Fifth table is the contingency table of “stroke” and “residence type”.

nb\_table(Stroke\_over, "Residence\_type")

## Residence\_type  
## Rural Urban Total  
## stroke = Yes 1844 1916 3760  
## stroke = No 1748 2012 3760  
## Total 3592 3928 7520

#### Sixth table is the contingency table of “stroke” and “smoking status”.

nb\_table(Stroke\_over, "smoking\_status")

## smoking\_status  
## formerly smoked never smoked smokes Unknown Total  
## stroke = Yes 618 1399 561 1182 3760  
## stroke = No 1053 1467 749 491 3760  
## Total 1671 2866 1310 1673 7520

#### Seventh table is the contingency table of “stroke” and “work type”.

nb\_table(Stroke\_over, "work\_type")

## work\_type  
## children Govt\_job Never\_worked Private Self-employed Total  
## stroke = Yes 532 496 21 2140 571 3760  
## stroke = No 14 587 0 2159 1000 3760  
## Total 546 1083 21 4299 1571 7520

### Gridlines for each variable in association with “stroke”.

#### Plot Function

nb\_plot <- function(x, y){  
 ggplot(x, aes(stroke)) + geom\_bar(aes(fill=x[,y]), position = "fill", color = "black") + ylab("Proportion") + labs(fill = y)  
}

#### Graphs of “stroke” in association with “gender” and “hyptertension”.

grid.arrange(nb\_plot(Stroke\_over, "gender"), nb\_plot(Stroke\_over,"hypertension"), nrow = 1)

Chart, bar chart

Description automatically generated

#### Run the Naive Bayes estimator for “stroke” in association with “gender” and “hypertension”.

nb01 <- naiveBayes(formula = stroke ~ gender + hypertension, data = Stroke\_over)  
  
nb\_gender <- confusionMatrix(data = predict(nb01, Stroke\_test, type = "class"), ref = Stroke\_test$stroke, positive = "Yes")  
nb\_gender

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 856 30  
## Yes 83 11  
##   
## Accuracy : 0.8847   
## 95% CI : (0.863, 0.904)  
## No Information Rate : 0.9582   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.1112   
##   
## Mcnemar's Test P-Value : 9.994e-07   
##   
## Sensitivity : 0.26829   
## Specificity : 0.91161   
## Pos Pred Value : 0.11702   
## Neg Pred Value : 0.96614   
## Prevalence : 0.04184   
## Detection Rate : 0.01122   
## Detection Prevalence : 0.09592   
## Balanced Accuracy : 0.58995   
##   
## 'Positive' Class : Yes   
##

#### Graphs of “stroke” in association with “heart disease” and “ever married”.

grid.arrange(nb\_plot(Stroke\_over, "heart\_disease"), nb\_plot(Stroke\_over,"ever\_married"), nrow = 1)

Chart, bar chart, waterfall chart

Description automatically generated

#### Run the Naive Bayes estimator for “stroke” in association with “heart disease” and “ever married”.

nb02 <- naiveBayes(formula = stroke ~ heart\_disease + ever\_married, data = Stroke\_over)  
  
nb\_heart\_marry <- confusionMatrix(data = predict(nb02, Stroke\_test, type = "class"), ref = Stroke\_test$stroke, positive = "Yes")  
nb\_heart\_marry

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 332 5  
## Yes 607 36  
##   
## Accuracy : 0.3755   
## 95% CI : (0.3451, 0.4067)  
## No Information Rate : 0.9582   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0289   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.87805   
## Specificity : 0.35357   
## Pos Pred Value : 0.05599   
## Neg Pred Value : 0.98516   
## Prevalence : 0.04184   
## Detection Rate : 0.03673   
## Detection Prevalence : 0.65612   
## Balanced Accuracy : 0.61581   
##   
## 'Positive' Class : Yes   
##

#### Graph of “stroke” in association with “Residence type”.

grid.arrange(nb\_plot(Stroke\_over, "Residence\_type"))

Chart, bar chart

Description automatically generated

#### Run the Naive Bayes estimator for “stroke” in association with “Residence type”.

nb03 <- naiveBayes(formula = stroke ~ Residence\_type, data = Stroke\_over)  
  
nb\_Res <- confusionMatrix(data = predict(nb03, Stroke\_test, type = "class"), ref = Stroke\_test$stroke, positive = "Yes")  
nb\_Res

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 474 20  
## Yes 465 21  
##   
## Accuracy : 0.5051   
## 95% CI : (0.4733, 0.5369)  
## No Information Rate : 0.9582   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0027   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.51220   
## Specificity : 0.50479   
## Pos Pred Value : 0.04321   
## Neg Pred Value : 0.95951   
## Prevalence : 0.04184   
## Detection Rate : 0.02143   
## Detection Prevalence : 0.49592   
## Balanced Accuracy : 0.50849   
##   
## 'Positive' Class : Yes   
##

#### Graph of “stroke” in association with “smoking status” and “work type”.

grid.arrange(nb\_plot(Stroke\_over, "smoking\_status"), nb\_plot(Stroke\_over, "work\_type"), nrow = 1)

Chart, bar chart

Description automatically generated

#### Run the Naive Bayes estimator for “stroke” in association with “smoking status” and “work type”.

nb04 <- naiveBayes(formula = stroke ~ smoking\_status + work\_type, data = Stroke\_over)  
  
nb\_smoke\_work <- confusionMatrix(data = predict(nb04, Stroke\_test, type = "class"), ref = Stroke\_test$stroke, positive = "Yes")  
nb\_smoke\_work

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 288 6  
## Yes 651 35  
##   
## Accuracy : 0.3296   
## 95% CI : (0.3002, 0.36)  
## No Information Rate : 0.9582   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0188   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.85366   
## Specificity : 0.30671   
## Pos Pred Value : 0.05102   
## Neg Pred Value : 0.97959   
## Prevalence : 0.04184   
## Detection Rate : 0.03571   
## Detection Prevalence : 0.70000   
## Balanced Accuracy : 0.58018   
##   
## 'Positive' Class : Yes   
##

## Neural Network - Hunter

### Fitting Models

#Unbalanced  
train <- createFolds(Stroke\_tr$stroke, k=10)  
  
nnet\_stroke <- caret::train(stroke ~ ., method = "nnet", data = Stroke\_tr,  
 tuneLength = 5,  
 trControl = trainControl(  
 method = "cv", indexOut = train),  
 trace = FALSE)   
   
nnet\_stroke

## Neural Network   
##   
## 3928 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 3535, 3535, 3535, 3535, 3535, 3535, ...   
## Resampling results across tuning parameters:  
##   
## size decay Accuracy Kappa   
## 1 0e+00 0.9572311 0.0000000000  
## 1 1e-04 0.9572311 0.0000000000  
## 1 1e-03 0.9572311 0.0000000000  
## 1 1e-02 0.9572311 0.0000000000  
## 1 1e-01 0.9572311 0.0000000000  
## 3 0e+00 0.9572311 0.0000000000  
## 3 1e-04 0.9572311 0.0000000000  
## 3 1e-03 0.9572311 0.0000000000  
## 3 1e-02 0.9572311 0.0000000000  
## 3 1e-01 0.9577400 0.0203270712  
## 5 0e+00 0.9572311 0.0000000000  
## 5 1e-04 0.9572311 0.0000000000  
## 5 1e-03 0.9572311 0.0000000000  
## 5 1e-02 0.9572311 0.0000000000  
## 5 1e-01 0.9572311 0.0101988636  
## 7 0e+00 0.9569754 0.0180328507  
## 7 1e-04 0.9572311 0.0000000000  
## 7 1e-03 0.9572311 0.0000000000  
## 7 1e-02 0.9572311 0.0000000000  
## 7 1e-01 0.9574862 0.0215378262  
## 9 0e+00 0.9572311 0.0000000000  
## 9 1e-04 0.9572311 0.0000000000  
## 9 1e-03 0.9572311 0.0000000000  
## 9 1e-02 0.9569767 -0.0004829545  
## 9 1e-01 0.9574862 0.0215378262  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 3 and decay = 0.1.

nnet\_stroke$finalModel

## a 15-3-1 network with 52 weights  
## inputs: genderMale age hypertensionYes heart\_diseaseYes ever\_marriedYes work\_typeGovt\_job work\_typeNever\_worked work\_typePrivate `work\_typeSelf-employed` Residence\_typeUrban avg\_glucose\_level bmi `smoking\_statusnever smoked` smoking\_statussmokes smoking\_statusUnknown   
## output(s): .outcome   
## options were - entropy fitting decay=0.1

plotnet(nnet\_stroke$finalModel, pad\_x = 0.25)

Diagram, text

Description automatically generated

#Balanced  
  
train <- createFolds(Stroke\_over$stroke, k=10)  
  
nnet\_stroke\_balanced <- caret::train(stroke ~ ., method = "nnet", data = Stroke\_over,  
 tuneLength = 5,  
 trControl = trainControl(  
 method = "cv", indexOut = train),  
 trace = FALSE)   
   
nnet\_stroke\_balanced

## Neural Network   
##   
## 7520 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 6768, 6768, 6768, 6768, 6768, 6768, ...   
## Resampling results across tuning parameters:  
##   
## size decay Accuracy Kappa   
## 1 0e+00 0.6424202 0.2848404  
## 1 1e-04 0.5768617 0.1537234  
## 1 1e-03 0.5791223 0.1582447  
## 1 1e-02 0.7412234 0.4824468  
## 1 1e-01 0.7135638 0.4271277  
## 3 0e+00 0.6622340 0.3244681  
## 3 1e-04 0.6922872 0.3845745  
## 3 1e-03 0.6682181 0.3364362  
## 3 1e-02 0.7807181 0.5614362  
## 3 1e-01 0.7847074 0.5694149  
## 5 0e+00 0.6676862 0.3353723  
## 5 1e-04 0.7381649 0.4763298  
## 5 1e-03 0.7763298 0.5526596  
## 5 1e-02 0.7783245 0.5566489  
## 5 1e-01 0.7823138 0.5646277  
## 7 0e+00 0.7630319 0.5260638  
## 7 1e-04 0.7555851 0.5111702  
## 7 1e-03 0.7299202 0.4598404  
## 7 1e-02 0.7873670 0.5747340  
## 7 1e-01 0.7836436 0.5672872  
## 9 0e+00 0.7804521 0.5609043  
## 9 1e-04 0.7150266 0.4300532  
## 9 1e-03 0.7835106 0.5670213  
## 9 1e-02 0.7840426 0.5680851  
## 9 1e-01 0.7845745 0.5691489  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 7 and decay = 0.01.

plotnet(nnet\_stroke\_balanced$finalModel, pad\_x = 0.25, circle\_col = c("#F784FF", "#F784FF"))

Diagram, text

Description automatically generated

#Z-Score Standardized  
  
Stroke\_tr\_z\_nnet <- standard.z.df(Stroke\_tr)  
  
train <- createFolds(Stroke\_tr\_z\_nnet$stroke, k=10)  
  
  
nnet\_stroke\_z <- caret::train(stroke ~ ., method = "nnet", data = Stroke\_tr\_z\_nnet,  
 tuneLength = 5,  
 trControl = trainControl(  
 method = "cv", indexOut = train),  
 trace = FALSE)   
   
nnet\_stroke\_z

## Neural Network   
##   
## 3928 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 3535, 3535, 3535, 3535, 3536, 3535, ...   
## Resampling results across tuning parameters:  
##   
## size decay Accuracy Kappa   
## 1 0e+00 0.9572311 0.00000000  
## 1 1e-04 0.9572311 0.00000000  
## 1 1e-03 0.9572311 0.00000000  
## 1 1e-02 0.9572311 0.00000000  
## 1 1e-01 0.9572311 0.00000000  
## 3 0e+00 0.9572318 0.02015567  
## 3 1e-04 0.9582489 0.04169071  
## 3 1e-03 0.9572311 0.00000000  
## 3 1e-02 0.9592681 0.09085595  
## 3 1e-01 0.9574849 0.03058514  
## 5 0e+00 0.9590123 0.08382559  
## 5 1e-04 0.9595212 0.13774851  
## 5 1e-03 0.9579945 0.06793554  
## 5 1e-02 0.9574856 0.01068182  
## 5 1e-01 0.9587598 0.10046361  
## 7 0e+00 0.9579945 0.05737656  
## 7 1e-04 0.9597783 0.14420028  
## 7 1e-03 0.9597757 0.09652999  
## 7 1e-02 0.9602865 0.16699041  
## 7 1e-01 0.9592687 0.13054370  
## 9 0e+00 0.9587579 0.06201778  
## 9 1e-04 0.9610486 0.16087842  
## 9 1e-03 0.9600334 0.16256469  
## 9 1e-02 0.9628310 0.23777756  
## 9 1e-01 0.9605390 0.16287208  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 9 and decay = 0.01.

plotnet(nnet\_stroke\_z$finalModel, pad\_x = 0.25, circle\_col = c("#FF9984", "#FF9984"))

Text

Description automatically generated

#Z-score standardized and balanced  
  
Stroke\_tr\_z\_bal <- standard.z.df(Stroke\_over)  
  
train <- createFolds(Stroke\_tr\_z\_nnet$stroke, k=10)  
  
  
nnet\_stroke\_z\_bal <- caret::train(stroke ~ ., method = "nnet", data = Stroke\_over,  
 tuneLength = 5,  
 trControl = trainControl(  
 method = "cv", indexOut = train),  
 trace = FALSE)   
   
nnet\_stroke\_z\_bal

## Neural Network   
##   
## 7520 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 6768, 6768, 6768, 6768, 6768, 6768, ...   
## Resampling results across tuning parameters:  
##   
## size decay Accuracy Kappa   
## 1 0e+00 0.2694864 0.04028738  
## 1 1e-04 0.5660546 0.02643556  
## 1 1e-03 0.5782456 0.02741529  
## 1 1e-02 0.6357597 0.10744617  
## 1 1e-01 0.7179221 0.14517540  
## 3 0e+00 0.6488069 0.09219538  
## 3 1e-04 0.6659254 0.07940954  
## 3 1e-03 0.7091902 0.12047621  
## 3 1e-02 0.7164369 0.16033342  
## 3 1e-01 0.7293848 0.15185464  
## 5 0e+00 0.6309511 0.12989159  
## 5 1e-04 0.7238634 0.12407983  
## 5 1e-03 0.5336618 0.09631729  
## 5 1e-02 0.7104851 0.14957540  
## 5 1e-01 0.7482104 0.15358422  
## 7 0e+00 0.6980961 0.11185244  
## 7 1e-04 0.6022479 0.11323999  
## 7 1e-03 0.7147881 0.15895386  
## 7 1e-02 0.7031579 0.13301399  
## 7 1e-01 0.7245352 0.15497244  
## 9 0e+00 0.6108902 0.11884824  
## 9 1e-04 0.6846296 0.14122883  
## 9 1e-03 0.7253207 0.16466484  
## 9 1e-02 0.7006004 0.15032576  
## 9 1e-01 0.7425988 0.15671316  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 5 and decay = 0.1.

plotnet(nnet\_stroke\_z\_bal$finalModel, pad\_x = 0.25, circle\_col= c("#90F88D", "#90F88D"))

Diagram, text

Description automatically generated

### Evaluate NN

nnet\_reg <- confusionMatrix(data = predict(nnet\_stroke, Stroke\_test), ref = Stroke\_test$stroke, positive = "Yes")  
nnet\_reg

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 939 41  
## Yes 0 0  
##   
## Accuracy : 0.9582   
## 95% CI : (0.9437, 0.9698)  
## No Information Rate : 0.9582   
## P-Value [Acc > NIR] : 0.5414   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 4.185e-10   
##   
## Sensitivity : 0.00000   
## Specificity : 1.00000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.95816   
## Prevalence : 0.04184   
## Detection Rate : 0.00000   
## Detection Prevalence : 0.00000   
## Balanced Accuracy : 0.50000   
##   
## 'Positive' Class : Yes   
##

nnet\_bal <- confusionMatrix(data = predict(nnet\_stroke\_balanced, Stroke\_test), ref = Stroke\_test$stroke, positive = "Yes")  
nnet\_bal

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 692 4  
## Yes 247 37  
##   
## Accuracy : 0.7439   
## 95% CI : (0.7153, 0.7709)  
## No Information Rate : 0.9582   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.1668   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.90244   
## Specificity : 0.73695   
## Pos Pred Value : 0.13028   
## Neg Pred Value : 0.99425   
## Prevalence : 0.04184   
## Detection Rate : 0.03776   
## Detection Prevalence : 0.28980   
## Balanced Accuracy : 0.81970   
##   
## 'Positive' Class : Yes   
##

nnet\_z <- confusionMatrix(data = predict(nnet\_stroke\_z, Stroke\_test), ref = Stroke\_test$stroke, positive = "Yes")  
nnet\_z

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 939 41  
## Yes 0 0  
##   
## Accuracy : 0.9582   
## 95% CI : (0.9437, 0.9698)  
## No Information Rate : 0.9582   
## P-Value [Acc > NIR] : 0.5414   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 4.185e-10   
##   
## Sensitivity : 0.00000   
## Specificity : 1.00000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.95816   
## Prevalence : 0.04184   
## Detection Rate : 0.00000   
## Detection Prevalence : 0.00000   
## Balanced Accuracy : 0.50000   
##   
## 'Positive' Class : Yes   
##

nnet\_z\_bal <- confusionMatrix(data = predict(nnet\_stroke\_z\_bal, Stroke\_test), ref = Stroke\_test$stroke, positive = "Yes")  
nnet\_z\_bal

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 695 5  
## Yes 244 36  
##   
## Accuracy : 0.7459   
## 95% CI : (0.7174, 0.7729)  
## No Information Rate : 0.9582   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.1632   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.87805   
## Specificity : 0.74015   
## Pos Pred Value : 0.12857   
## Neg Pred Value : 0.99286   
## Prevalence : 0.04184   
## Detection Rate : 0.03673   
## Detection Prevalence : 0.28571   
## Balanced Accuracy : 0.80910   
##   
## 'Positive' Class : Yes   
##

# Model Evaluation

## Add Baseline

Stroke\_count <- Stroke\_test %>% group\_by(stroke) %>% tally()  
Stroke\_count

## # A tibble: 2 x 2  
## stroke n  
## <fct> <int>  
## 1 No 939  
## 2 Yes 41

TN <- as.numeric(Stroke\_count[1,2])  
FN <- as.numeric(Stroke\_count[2,2])  
TP <- 0  
FP <- 0  
  
Accuracy\_base <- TN/(TN+FN)  
Sensitivity\_base <- TP/(TP+FN)  
Specificity\_base<- TN/(TN+FP)  
Precision\_base <- TP/(TP+FP)  
F1\_base <- 0  
  
Baseline <- c(Accuracy\_base, Sensitivity\_base, Specificity\_base, Precision\_base, F1\_base)

## Model Comparison Data Frame

#Models  
"ANN Reg." <- c(nnet\_reg$overall, nnet\_reg$byClass)  
"ANN Bal." <- c(nnet\_bal$overall, nnet\_bal$byClass)  
"ANN Z" <- c(nnet\_z$overall, nnet\_z$byClass)  
"ANN Z Bal." <- c(nnet\_z\_bal$overall, nnet\_z\_bal$byClass)  
"RF Reg." <- c(rf\_reg$overall, rf\_reg$byClass)  
"RF Bal." <- c(rf\_bal$overall, rf\_bal$byClass)  
"CART Bal." <- c(cart\_bal$overall, cart\_bal$byClass)  
"C5.0 Bal." <- c(C5\_bal$overall, C5\_bal$byClass)  
"NB Gender" <- c(nb\_gender$overall, nb\_gender$byClass)  
"NB Heart + Marry" <- c(nb\_heart\_marry$overall, nb\_heart\_marry$byClass)  
"NB Resident" <- c(nb\_Res$overall, nb\_Res$byClass)  
"NB Smoke + Work" <- c(nb\_smoke\_work$overall, nb\_smoke\_work$byClass)  
"CART Cost" <- c(cart\_cost$overall, cart\_cost$byClass)  
"CART Cost Bal." <- c(cart\_cost\_bal$overall, cart\_cost\_bal$byClass)  
"Log Reg." <- c(LogReg$overall, LogReg$byClass)  
  
  
Model\_comp <- rbind(`ANN Reg.`, `ANN Bal.`, `ANN Z`, `ANN Z Bal.`, `RF Reg.`, `RF Bal.`, `CART Bal.`, `CART Cost`, `CART Cost Bal.`, `C5.0 Bal.`, `NB Gender`, `NB Heart + Marry`, `NB Resident`, `NB Smoke + Work`, `Log Reg.`)  
  
Model\_comp <- data.frame(Model\_comp)  
  
Model\_comp <- Model\_comp %>% dplyr::select(Accuracy, Sensitivity, Specificity, Precision, F1)  
  
Model\_comp <- rbind(Baseline , Model\_comp)  
  
rownames(Model\_comp)[rownames(Model\_comp) == 1] <- "Baseline"  
  
Model\_comp <- cbind(Model = rownames(Model\_comp), Model\_comp)  
rownames(Model\_comp) <- 1:nrow(Model\_comp)  
  
Model\_comp$Model <- as.factor(Model\_comp$Model)  
Model\_comp[is.na(Model\_comp)] <- 0

## Accuracy Graph

Plot\_acc <- Model\_comp %>% mutate(Model = fct\_reorder(Model, desc(Accuracy))) %>% mutate(Performance = ifelse(Model\_comp$Accuracy == Accuracy\_base, "Baseline", ifelse(Model\_comp$Accuracy < Accuracy\_base, "Worse", "Better"))) %>% ggplot(aes(x=Model, y=Accuracy, fill = Performance)) + geom\_bar(stat = "identity") + coord\_flip() + scale\_fill\_manual(values = c("grey", "#c12503", "darkgreen"))  
plot(Plot\_acc)

Chart, bar chart

Description automatically generated

## Sensitivity

Plot\_sens <- Model\_comp %>% mutate(Model = fct\_reorder(Model, desc(Sensitivity))) %>% mutate(Performance = ifelse(Model\_comp$Sensitivity == Sensitivity\_base, "Baseline", ifelse(Model\_comp$Sensitivity < Sensitivity\_base, "Worse", "Better"))) %>%   
 ggplot(aes(x=Model, y=Sensitivity, fill = Performance)) + geom\_bar(stat = "identity") + coord\_flip() + scale\_fill\_manual(values = c("Grey", "darkgreen"))  
plot(Plot\_sens)

Chart

Description automatically generated ## Side by side for paper

grid.arrange(Plot\_acc, Plot\_sens, ncol = 2)

Chart, waterfall chart

Description automatically generated ## Specificity

Model\_comp %>% mutate(Model = fct\_reorder(Model, desc(Specificity))) %>% mutate(Performance = ifelse(Model\_comp$Specificity == Specificity\_base, "Baseline", ifelse(Model\_comp$Specificity < Specificity\_base, "Worse", "Better"))) %>%   
 ggplot(aes(x=Model, y=Specificity, fill = Performance)) + geom\_bar(stat = "identity") + coord\_flip() + scale\_fill\_manual(values = c("Grey", "#c12503"))

Chart

Description automatically generated

## Precision

Model\_comp %>% mutate(Model = fct\_reorder(Model, desc(Precision))) %>% mutate(Performance = ifelse(Model\_comp$Precision == 0, "Baseline", ifelse(Model\_comp$Precision < 0, "Worse", "Better"))) %>% ggplot(aes(x=Model, y=Precision, fill = Performance)) + geom\_bar(stat = "identity") + coord\_flip() + scale\_fill\_manual(values = c("Grey", "darkgreen"))

Chart

Description automatically generated

## F1

Model\_comp %>% mutate(Model = fct\_reorder(Model, desc(F1))) %>% mutate(Performance = ifelse(Model\_comp$F1 == F1\_base, "Baseline", ifelse(Model\_comp$F1 < F1\_base, "Worse", "Better"))) %>%   
 ggplot(aes(x=Model, y=F1, fill = Performance)) + geom\_bar(stat = "identity") + coord\_flip() + scale\_fill\_manual(values = c("Grey", "darkgreen"))

Chart

Description automatically generated

## Comparison Table

Model\_comp

## Model Accuracy Sensitivity Specificity Precision F1  
## 1 Baseline 0.95816327 0.0000000 1.0000000 0.00000000 0.00000000  
## 2 ANN Reg. 0.95816327 0.0000000 1.0000000 0.00000000 0.00000000  
## 3 ANN Bal. 0.74387755 0.9024390 0.7369542 0.13028169 0.22769231  
## 4 ANN Z 0.95816327 0.0000000 1.0000000 0.00000000 0.00000000  
## 5 ANN Z Bal. 0.74591837 0.8780488 0.7401491 0.12857143 0.22429907  
## 6 RF Reg. 0.95816327 0.0000000 1.0000000 0.00000000 0.00000000  
## 7 RF Bal. 0.95816327 0.0000000 1.0000000 0.00000000 0.00000000  
## 8 CART Bal. 0.70000000 0.9024390 0.6911608 0.11314985 0.20108696  
## 9 CART Cost 0.84591837 0.5121951 0.8604899 0.13815789 0.21761658  
## 10 CART Cost Bal. 0.51632653 0.9756098 0.4962726 0.07797271 0.14440433  
## 11 C5.0 Bal. 0.78469388 0.5365854 0.7955272 0.10280374 0.17254902  
## 12 NB Gender 0.88469388 0.2682927 0.9116081 0.11702128 0.16296296  
## 13 NB Heart + Marry 0.37551020 0.8780488 0.3535676 0.05598756 0.10526316  
## 14 NB Resident 0.50510204 0.5121951 0.5047923 0.04320988 0.07969639  
## 15 NB Smoke + Work 0.32959184 0.8536585 0.3067093 0.05102041 0.09628611  
## 16 Log Reg. 0.04183673 1.0000000 0.0000000 0.04183673 0.08031342