Stroke Dataset

Hunter Blum, Ben Earnest, & Andrew Kim

4/12/2022

Dataset:

https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset (https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset)

Libraries

```
library(randomForest)
```

```
## 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)
library(nnet)
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
library(DataExplorer)
library(Boruta)
library(rpart)
library(rpart.plot)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

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

```
library(e1071)
library(kableExtra)
library(tidyverse)
```

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

```
## v tibble  3.1.6     v dplyr   1.0.8
## 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

```
#Import the data set
Stroke <- read.csv("/Users/andre/OneDrive/Documents/ADS502/Stroke Dataset.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.facto
r)

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.
```

gen… <int>	ag <int>	hypertension <int>	heart_disease <int>	ever_married <int>	work_type <int>	Residence_type <int>	avg_g
0	0	0	0	0	0	0	

1 row | 1-9 of 11 columns

```
#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)
```

	gen… <fct>	ag <dbl>	hypertension <fct>	heart_disease <fct>	ever_married <fct>	work_type <fct>	Residence_type <fct>
1	Male	67	No	Yes	Yes	Private	Urban
3	Male	80	No	Yes	Yes	Private	Rural
4	Female	49	No	No	Yes	Private	Urban
5	Female	79	Yes	No	Yes	Self-employed	Rural
6	Male	81	No	No	Yes	Private	Urban
7	Male	74	Yes	Yes	Yes	Private	Rural

6 rows | 1-8 of 12 columns

Variable by Stroke

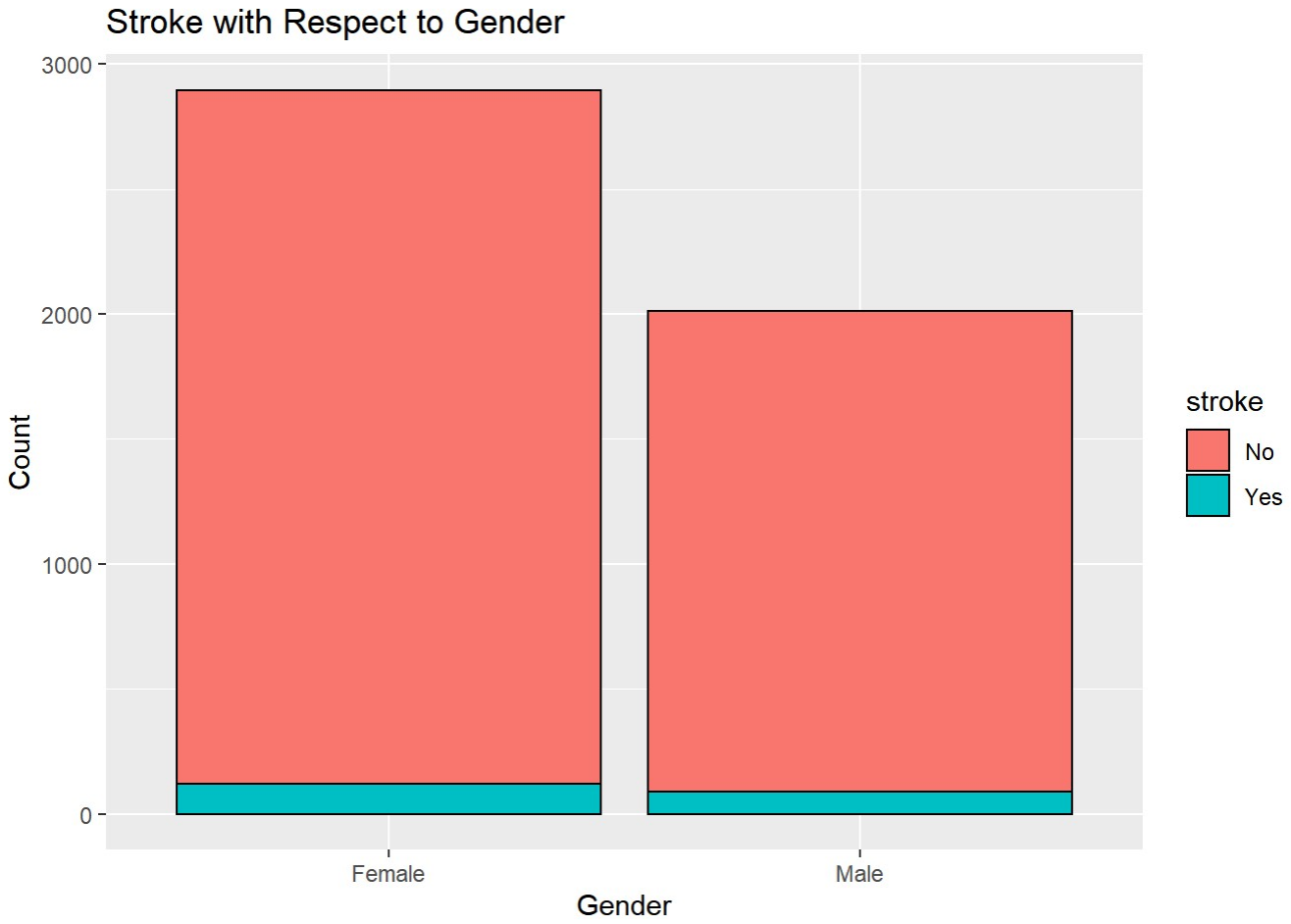
Make Functions

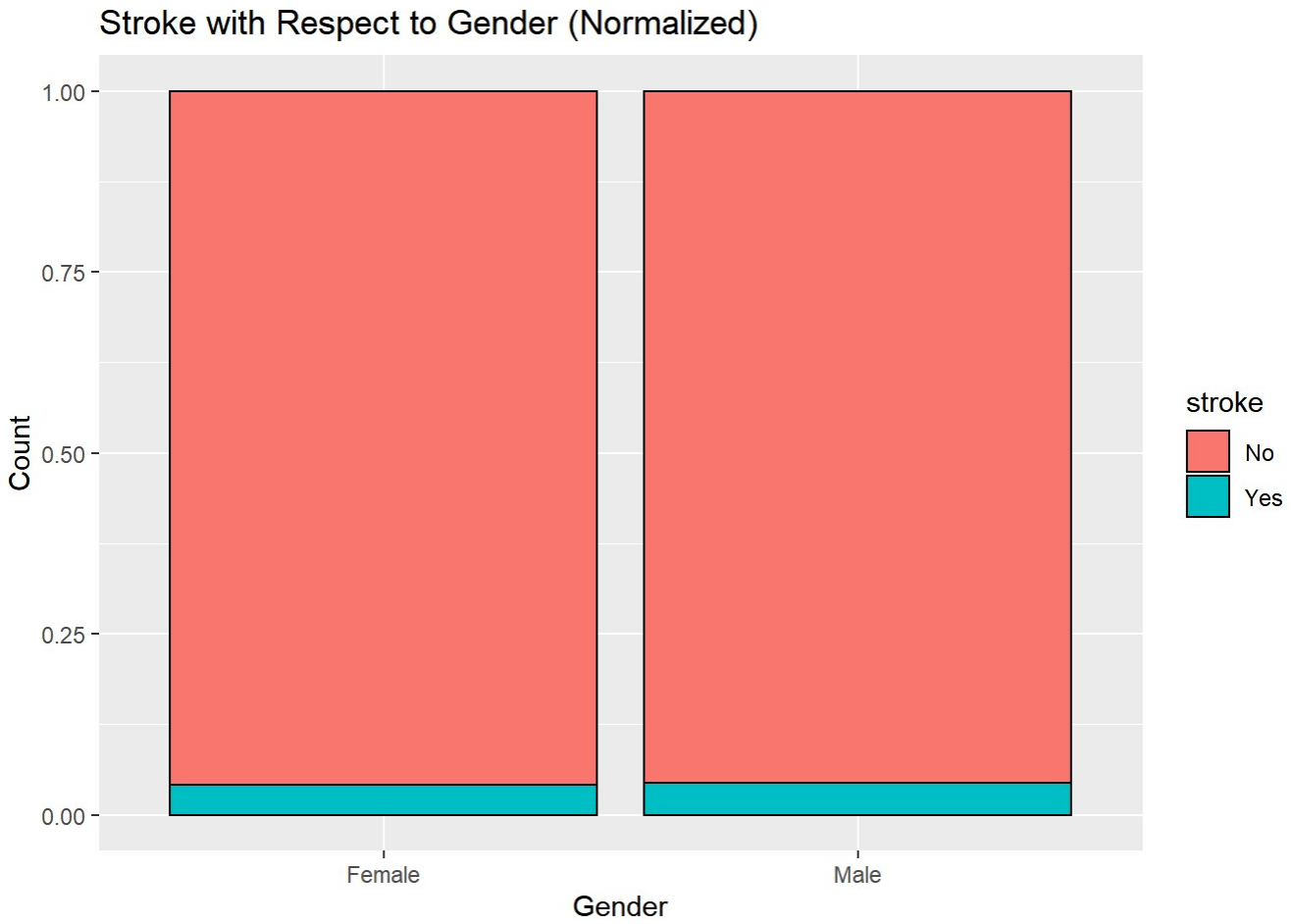
```
#Categorical
Cat_eda <- function(x, y) {
  p1 <- ggplot(Stroke_clean, aes(x={{x}})) +
    geom_bar(aes(fill=stroke) , color = "black") +
    ggtitle(paste0("Stroke with Respect to ", y)) +
    xlab(y) + ylab("Count")

  p2 <- ggplot(Stroke_clean, aes(x={{x}})) +
    geom_bar(aes(fill=stroke), position = "fill", color = "black") + ggtitle(paste0("Stroke w
ith 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)) +
   xlab(y) + ylab("Count")
 plot(p1)
 plot(p2)

}
```

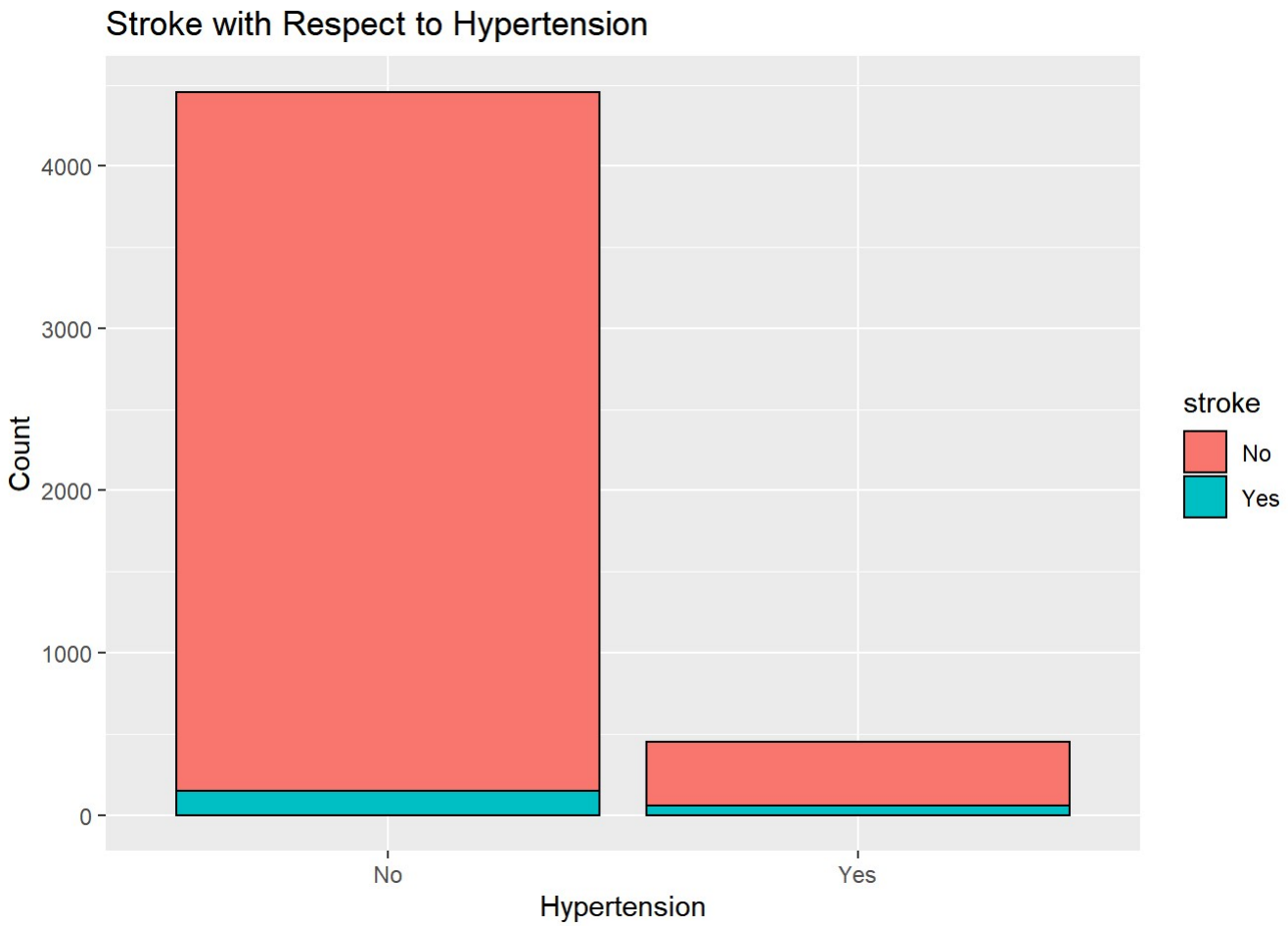
Categorical Variables

```
Cat_eda(gender, "Gender")
```

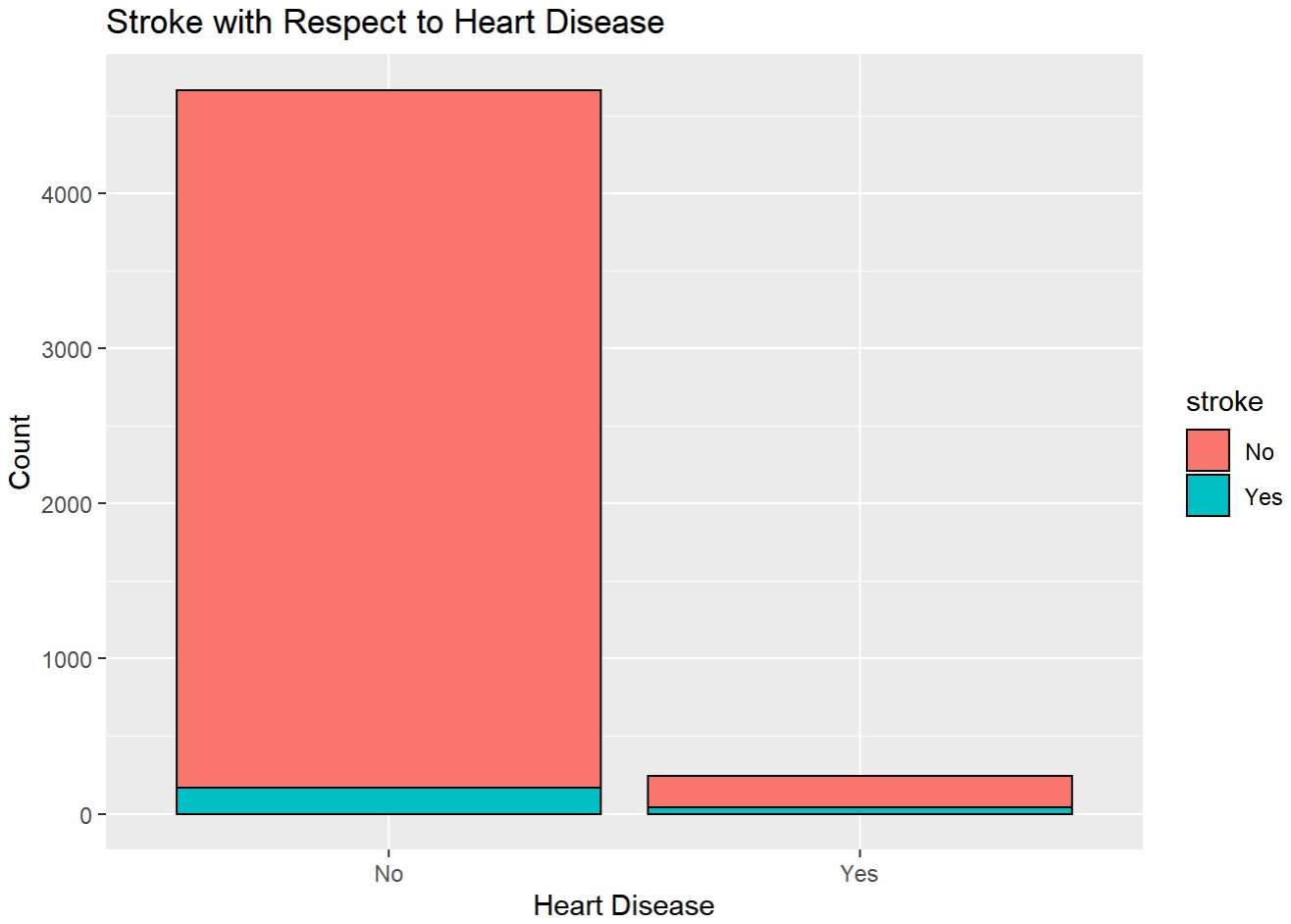

Stroke with Respect to Gender
3000
2000
1000
0
Count
Female
Male
Gender
stroke
No
Yes


Stroke with Respect to Gender (Normalized)
1.00
0.75
0.50
0.25
0.00
Count
Female
Male
Gender
stroke
No
Yes

```
Cat_eda(hypertension, "Hypertension")
```

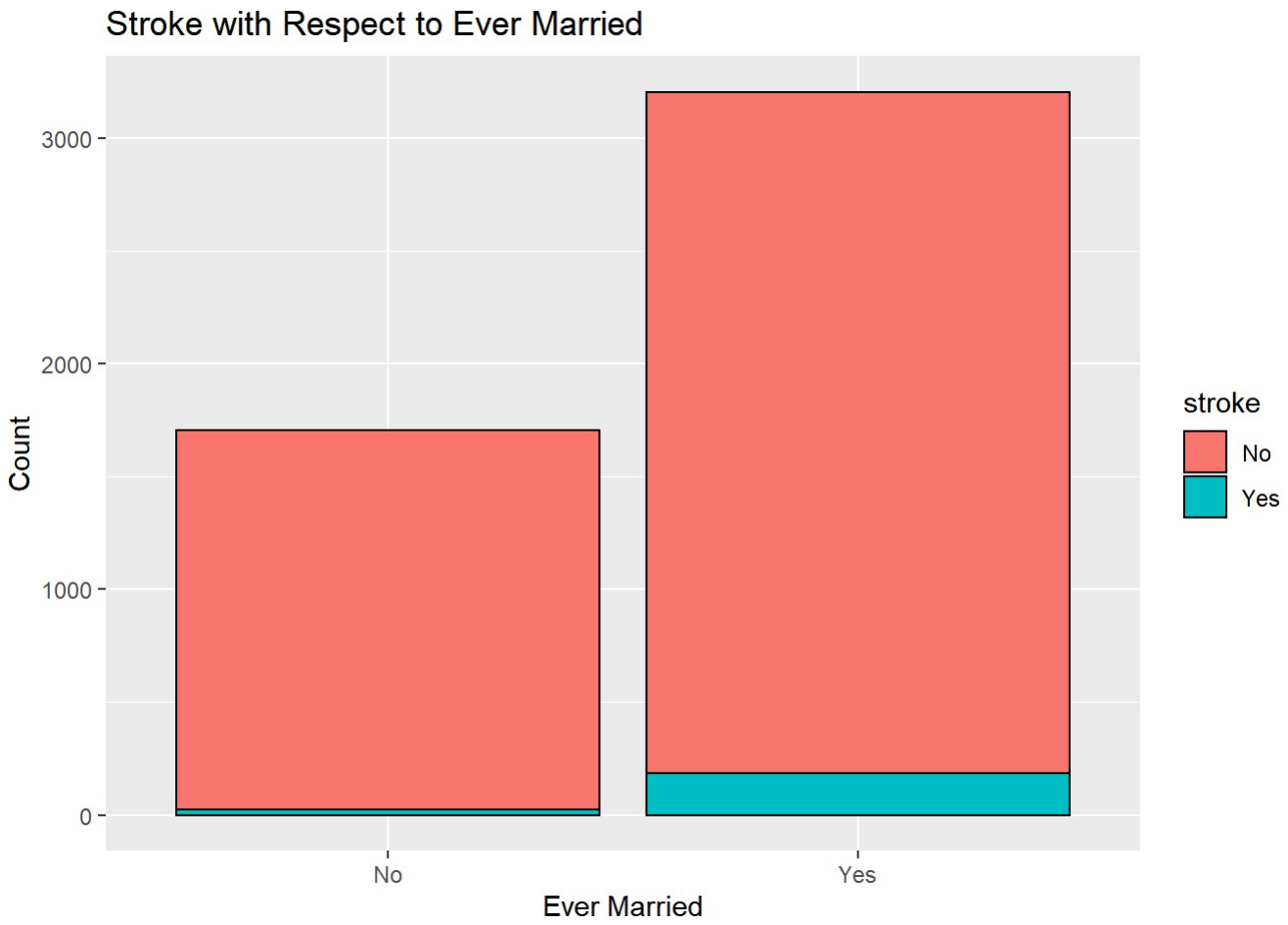


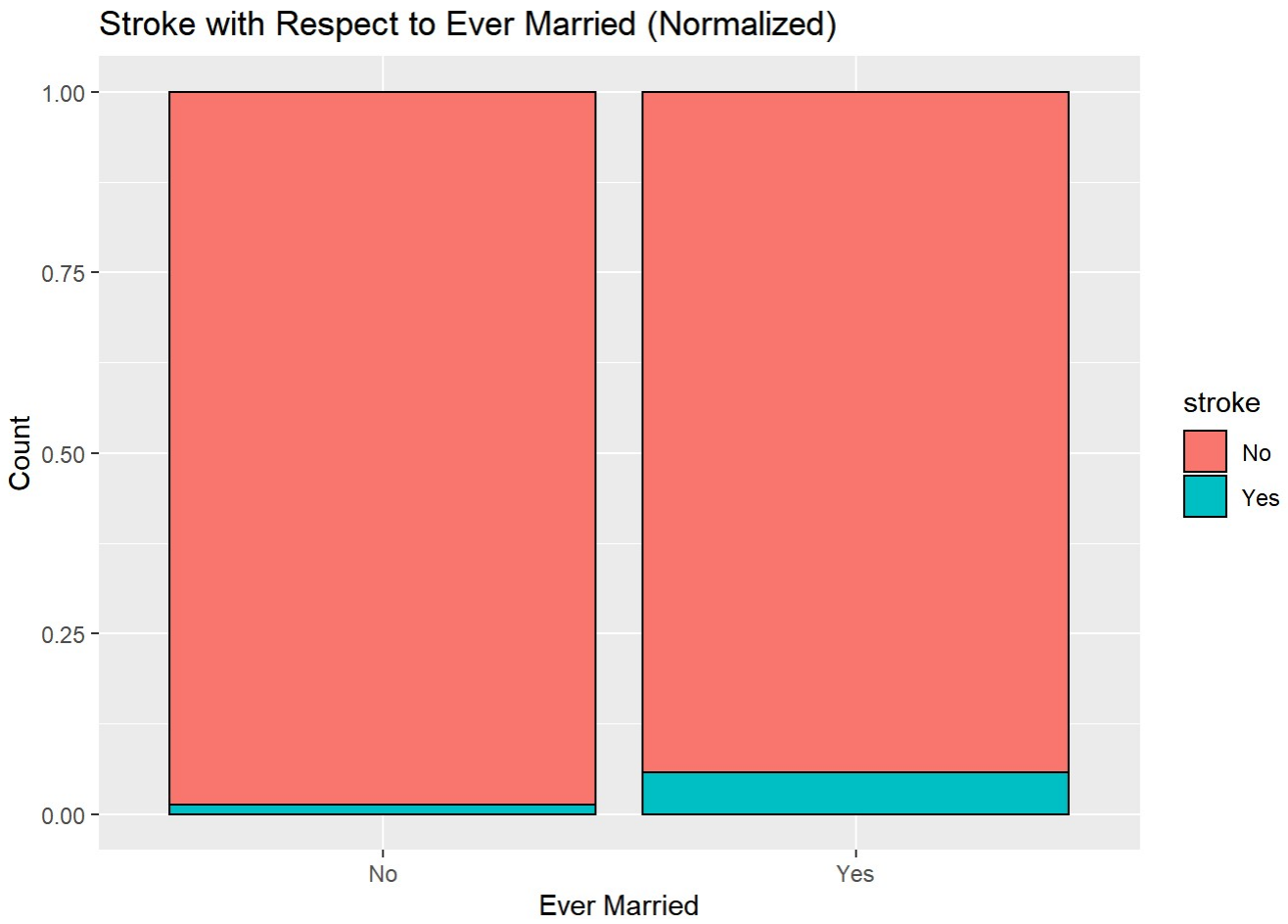

```
Cat_eda(heart_disease, "Heart Disease")
```


Stroke with Respect to Heart Disease
Count
4000
3000
2000
1000
0
No
Yes
Heart Disease
stroke
No
Yes

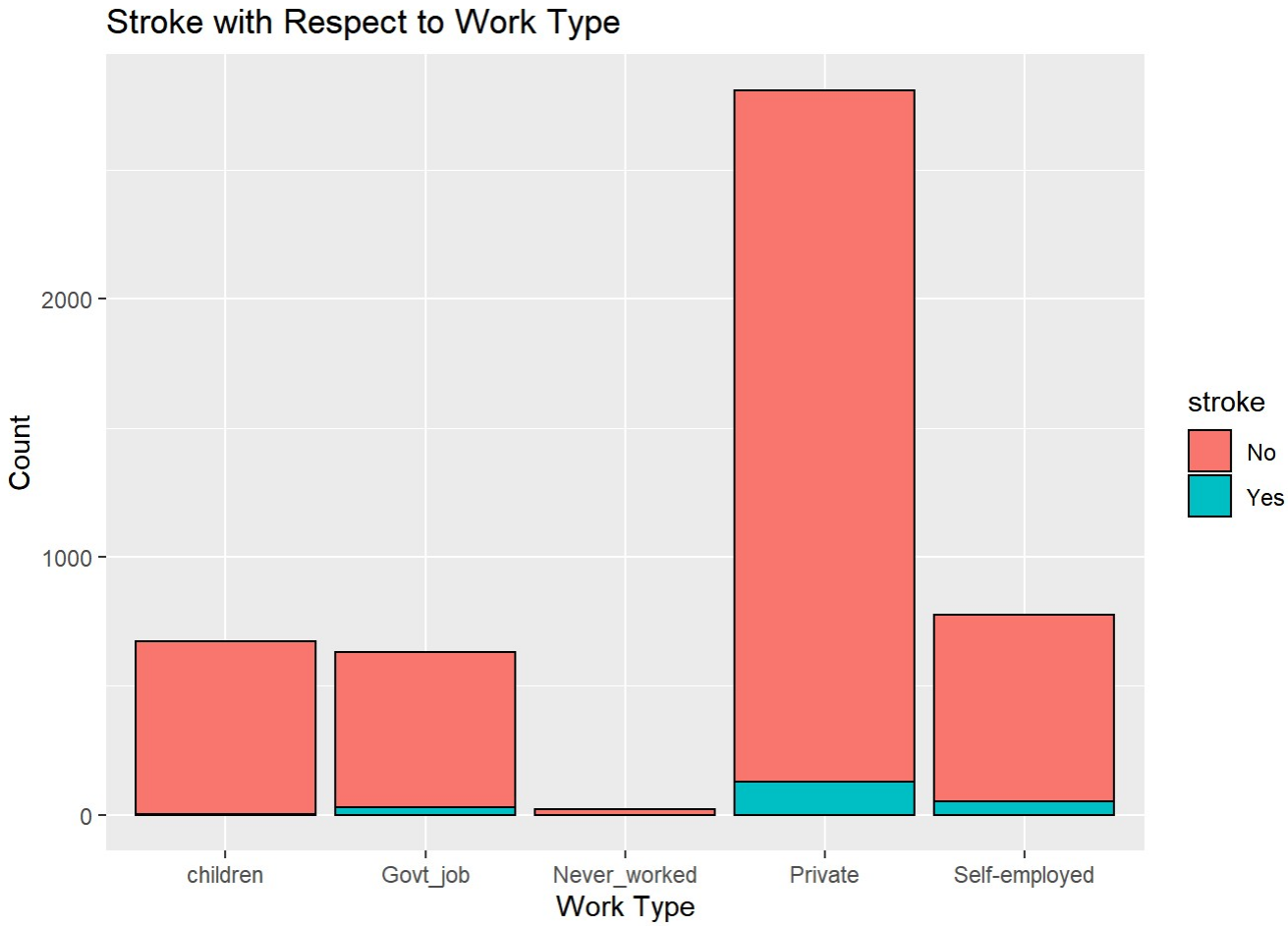

Stroke with Respect to Heart Disease (Normalized)
Count
1.00
0.75
0.50
0.25
0.00
No
Yes
Heart Disease
stroke
No
Yes

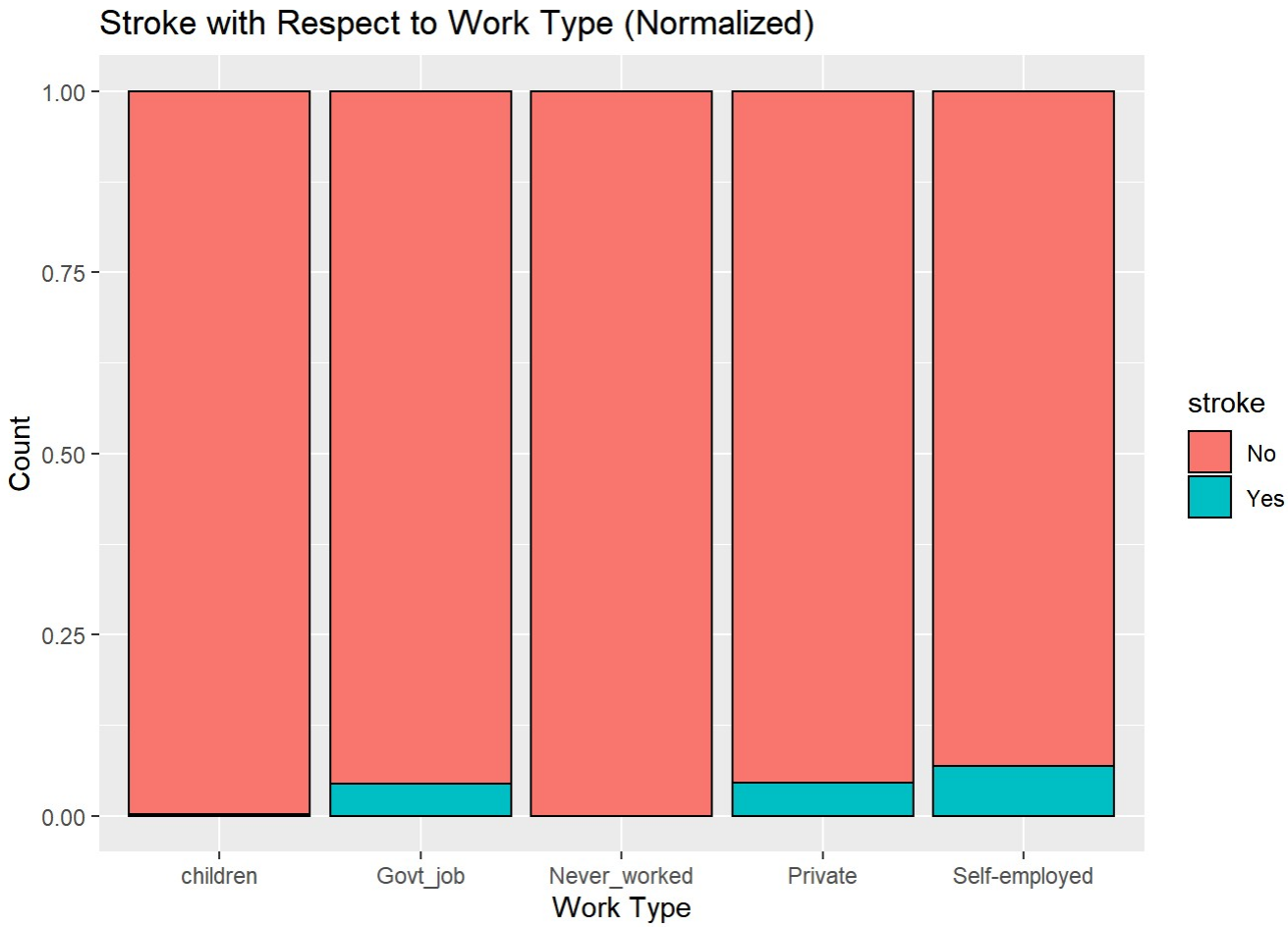
```
Cat_eda(ever_married, "Ever Married")
```



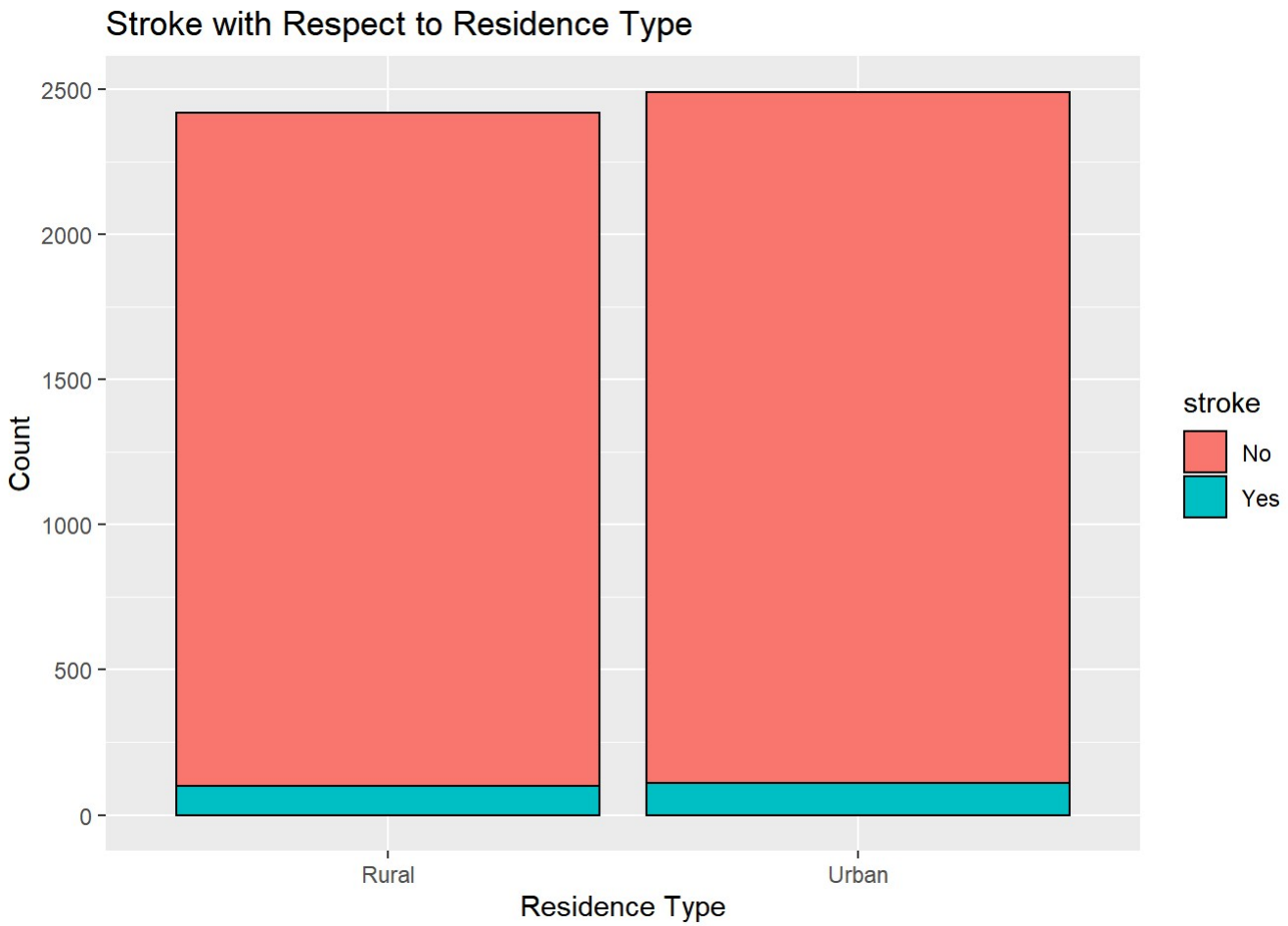



```
Cat_eda(work_type, "Work Type")
```

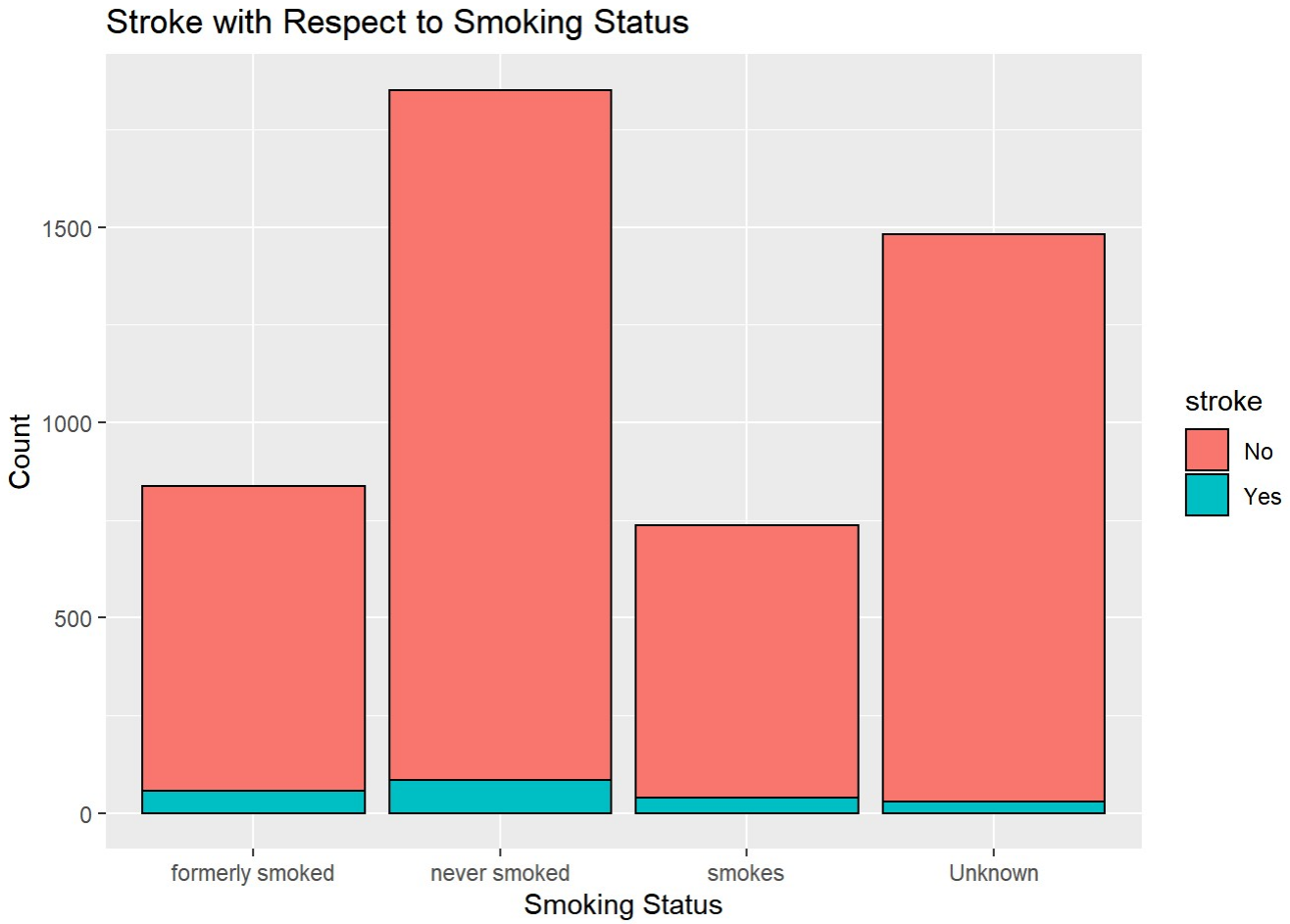

Stroke with Respect to Work Type
Count
2000
1000
0
children
Govt_job
Never_worked
Private
Self-employed
Work Type
stroke
No
Yes

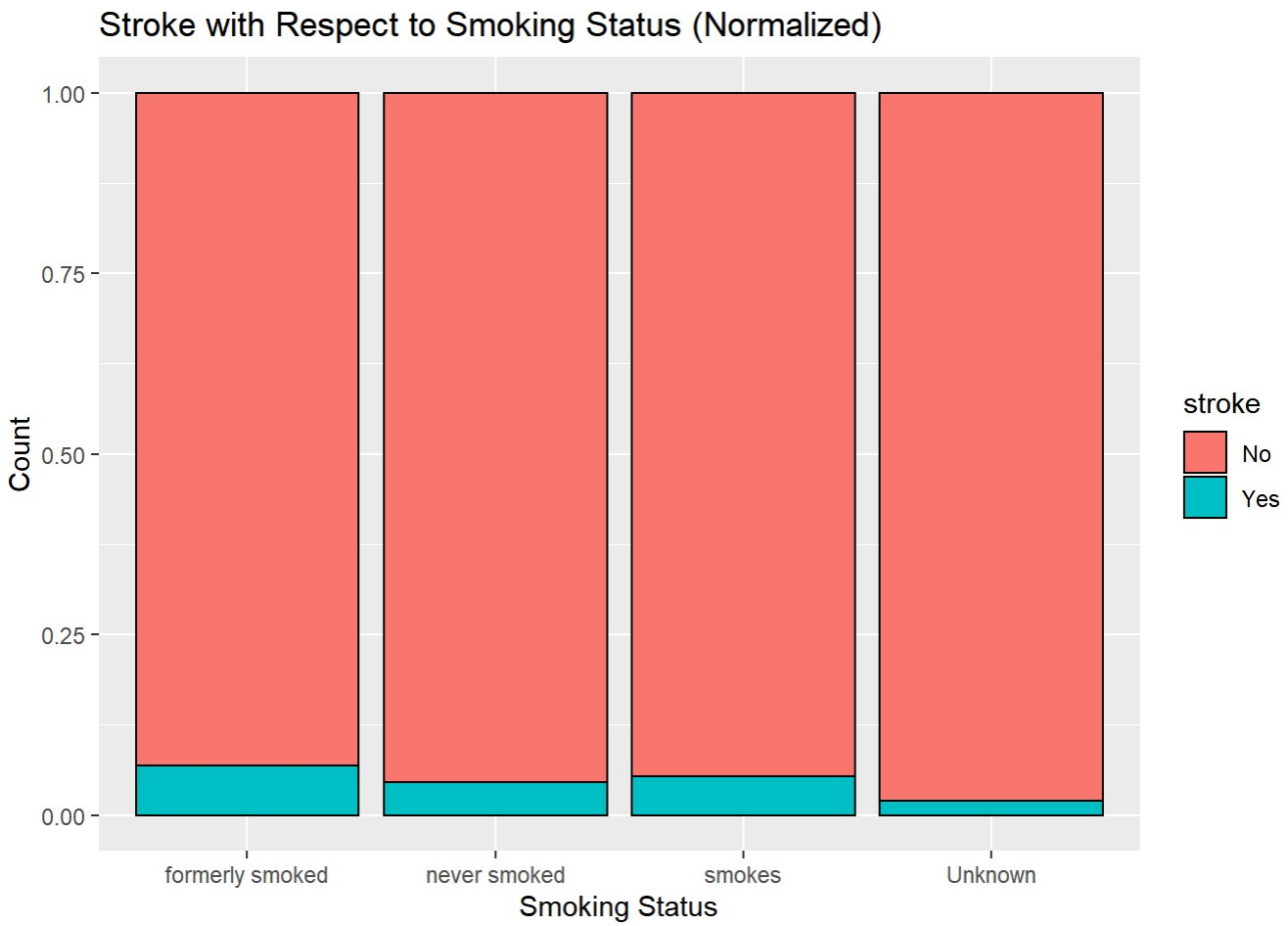

Stroke with Respect to Work Type (Normalized)
Count
1.00
0.75
0.50
0.25
0.00
children
Govt_job
Never_worked
Private
Self-employed
Work Type
stroke
No
Yes

```
Cat_eda(Residence_type, "Residence Type")
```




```
Cat_eda(smoking_status, "Smoking Status")
```

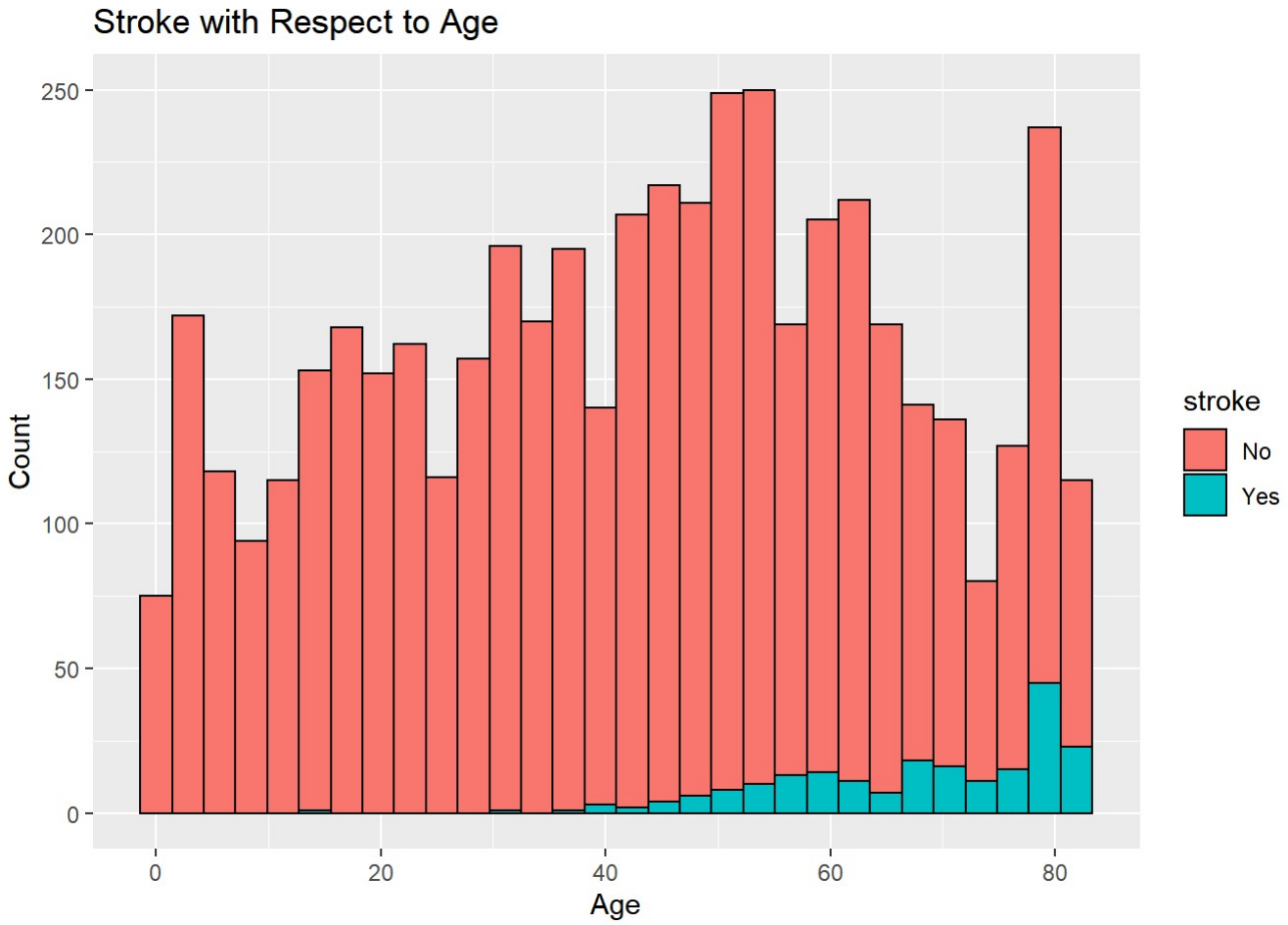

Stroke with Respect to Smoking Status
Count
1500
1000
500
0
formerly smoked
never smoked
smokes
Unknown
Smoking Status
stroke
No
Yes


Stroke with Respect to Smoking Status (Normalized)
Count
1.00
0.75
0.50
0.25
0.00
formerly smoked
never smoked
smokes
Unknown
Smoking Status
stroke
No
Yes

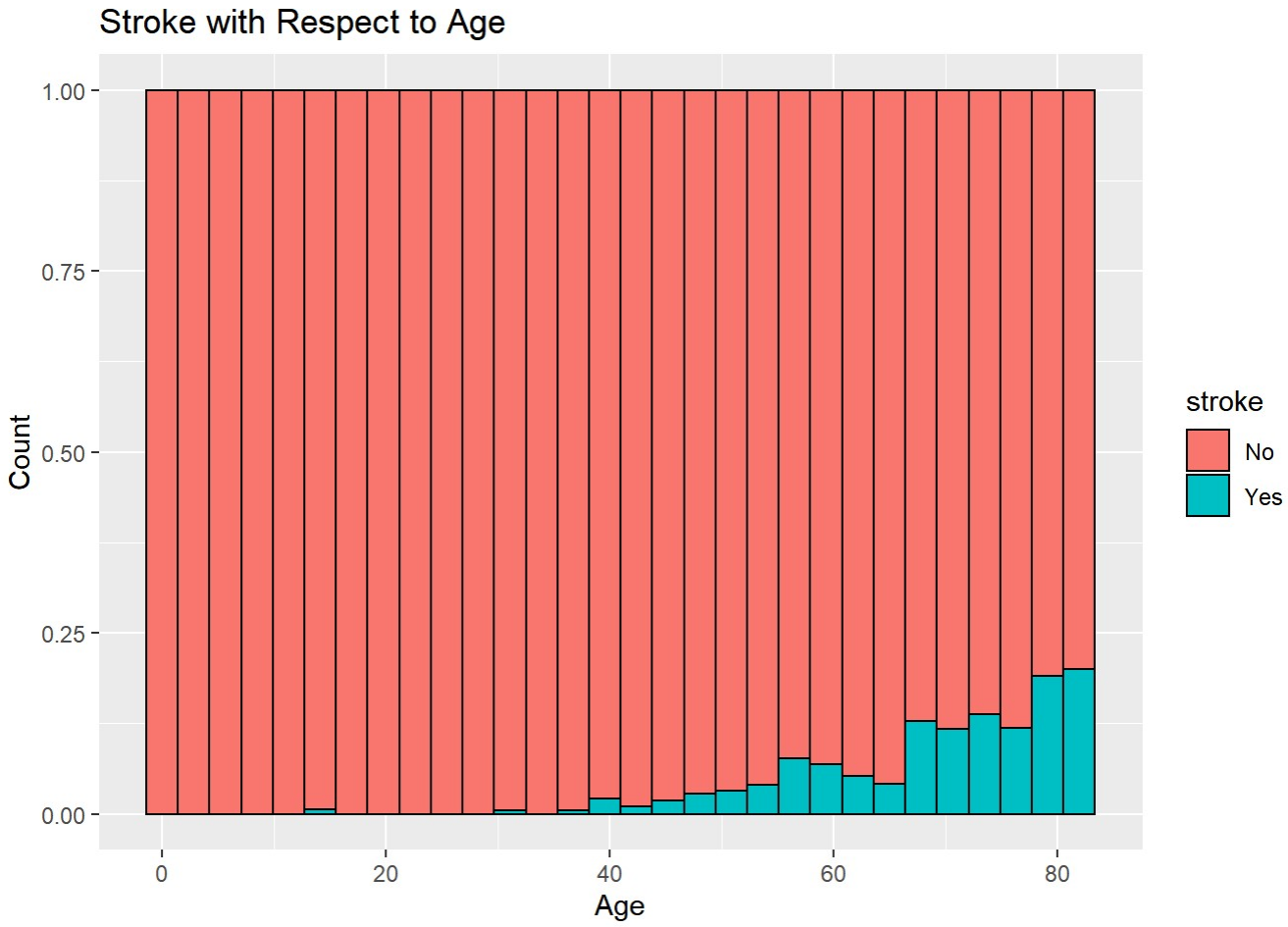
Numeric variables

```
Num_eda(age, "Age")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

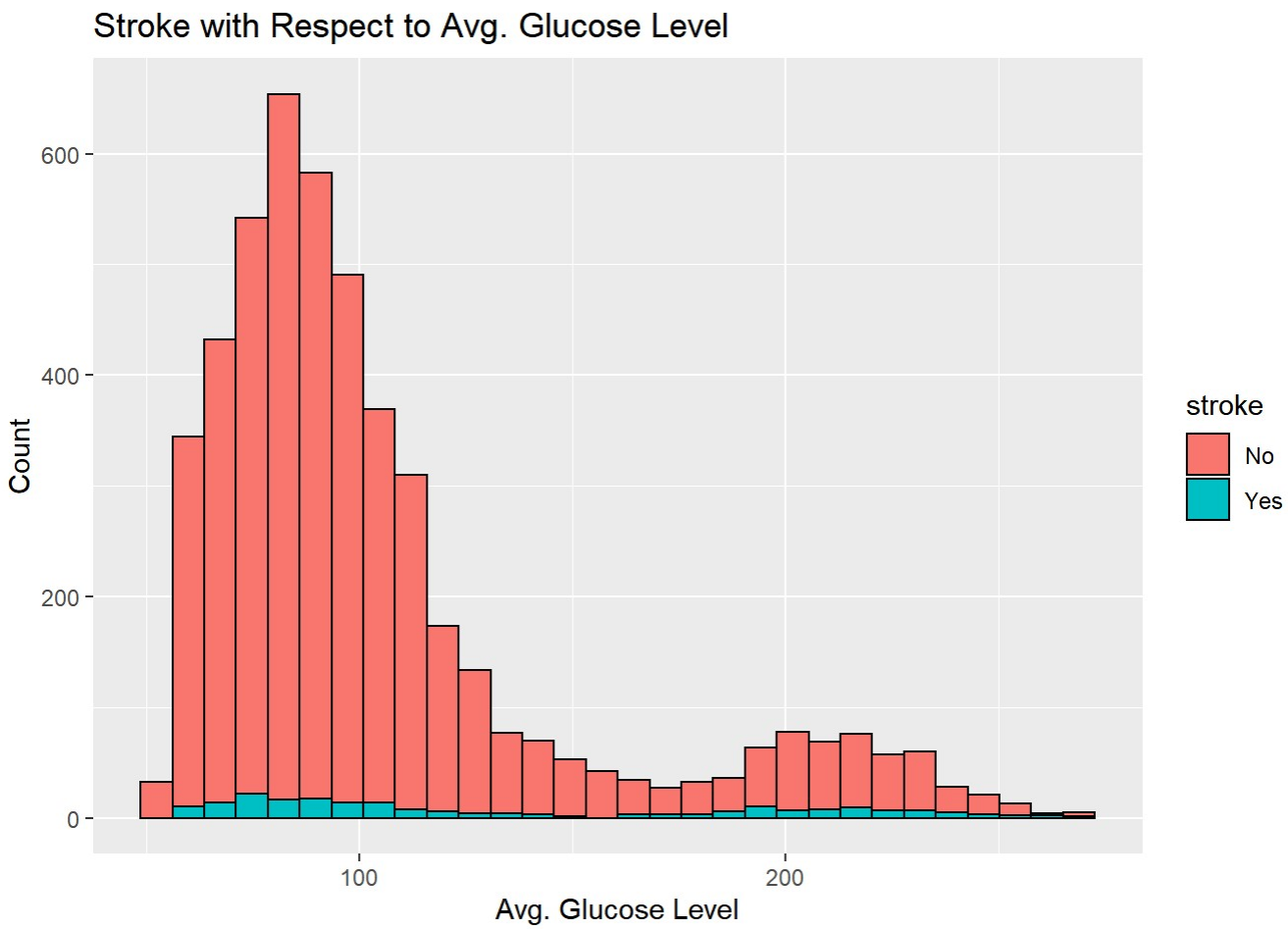



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

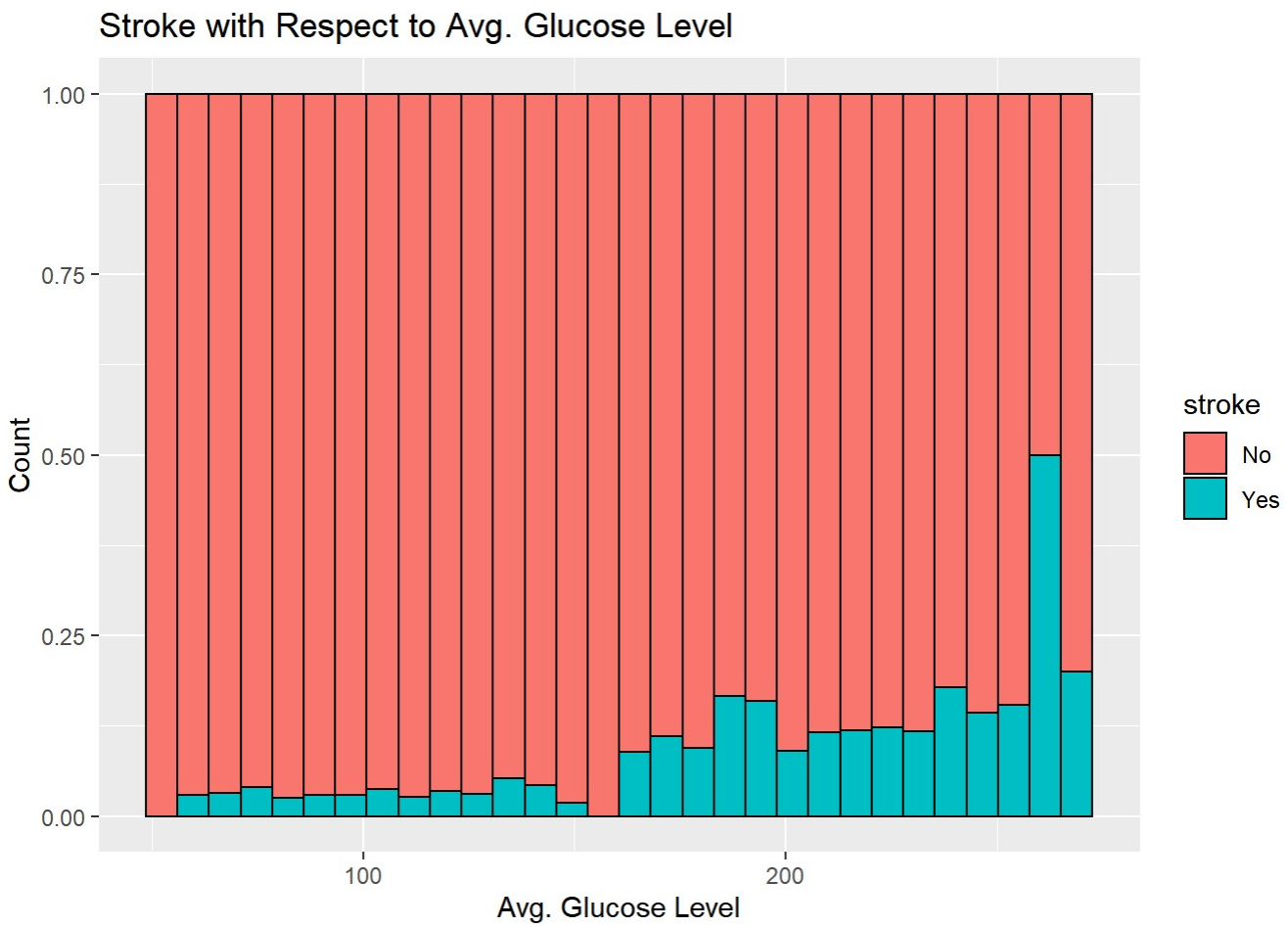



```
Num_eda(avg_glucose_level, "Avg. Glucose Level")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

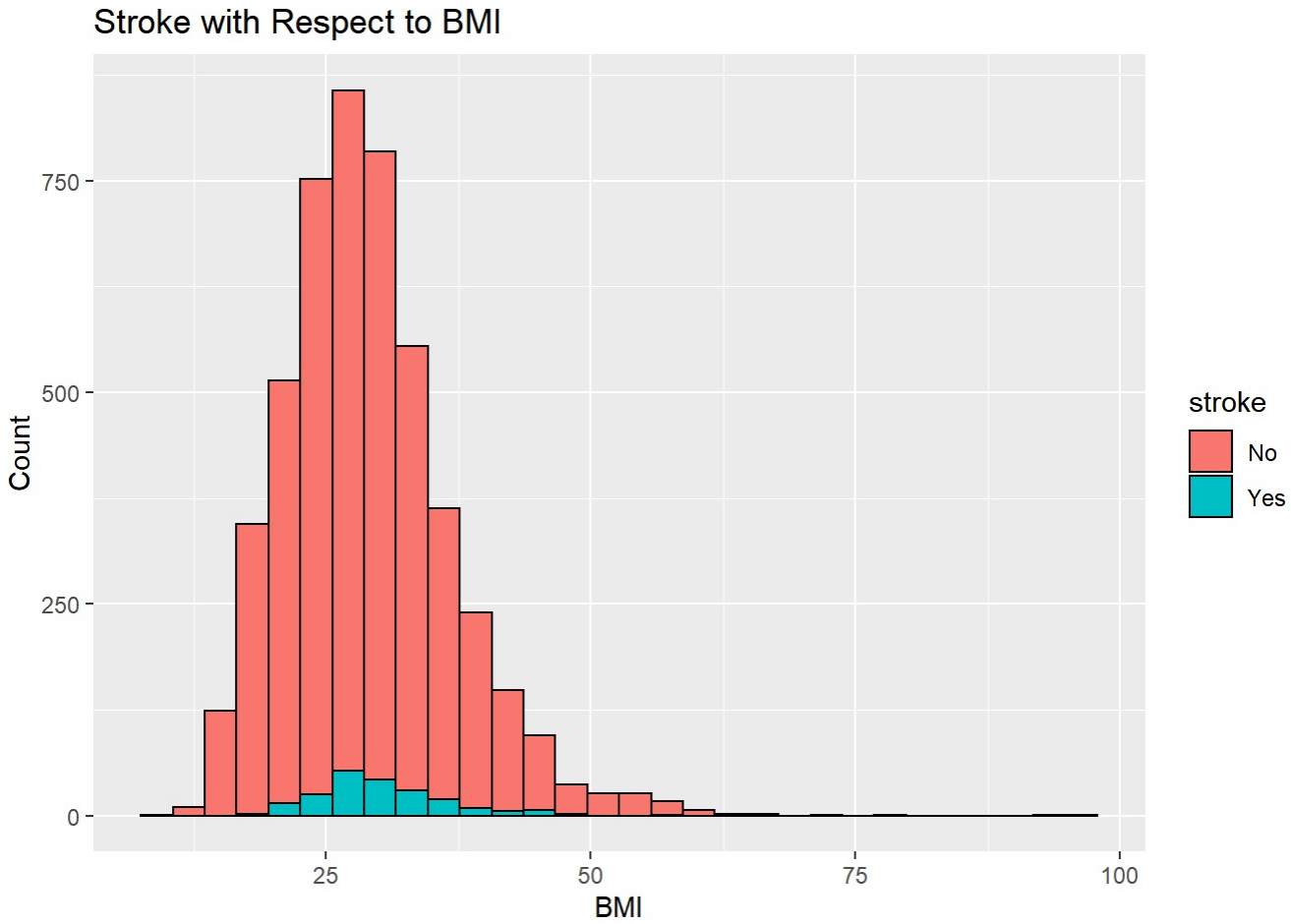



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



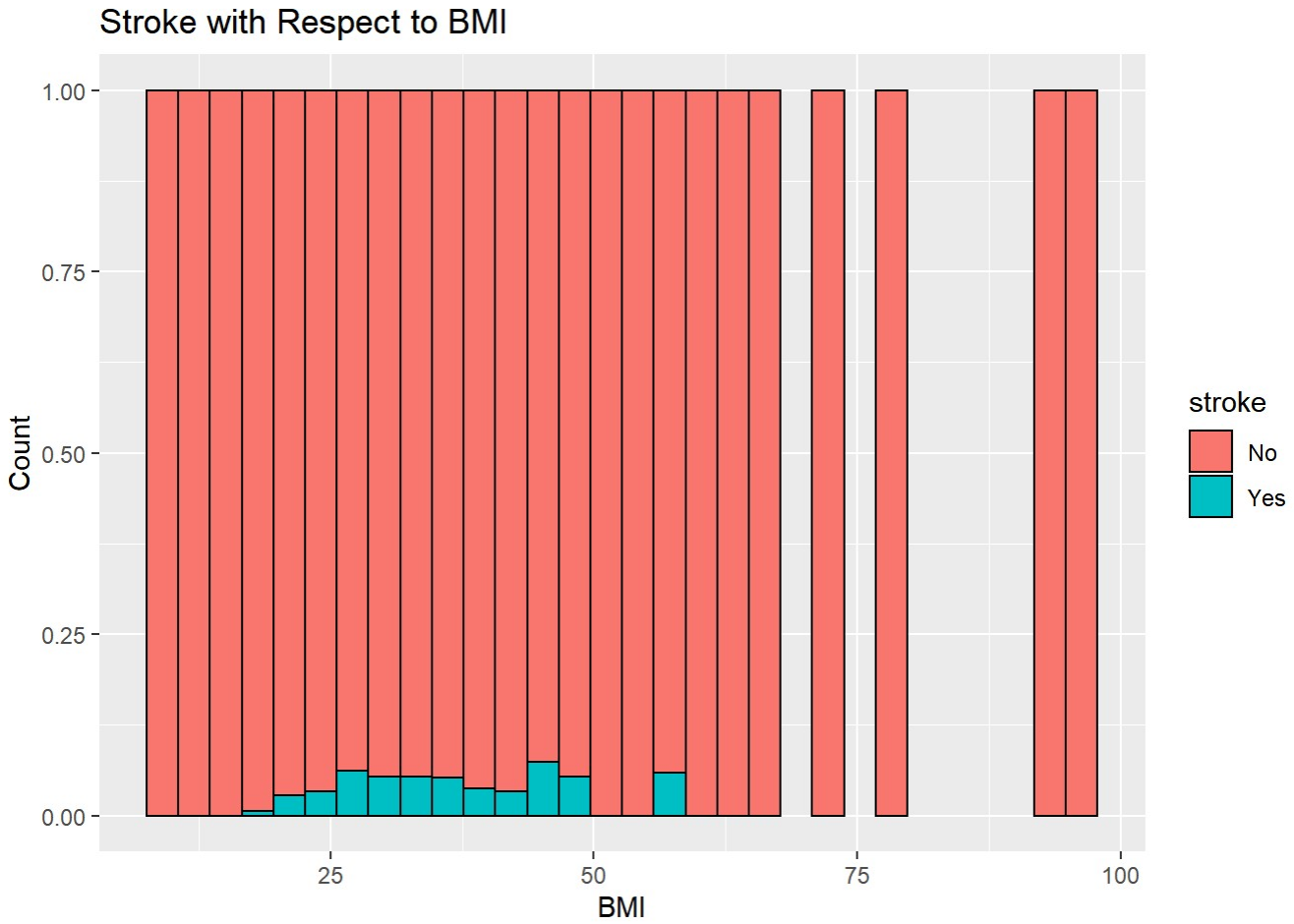

```
Num_eda(bmi, "BMI")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



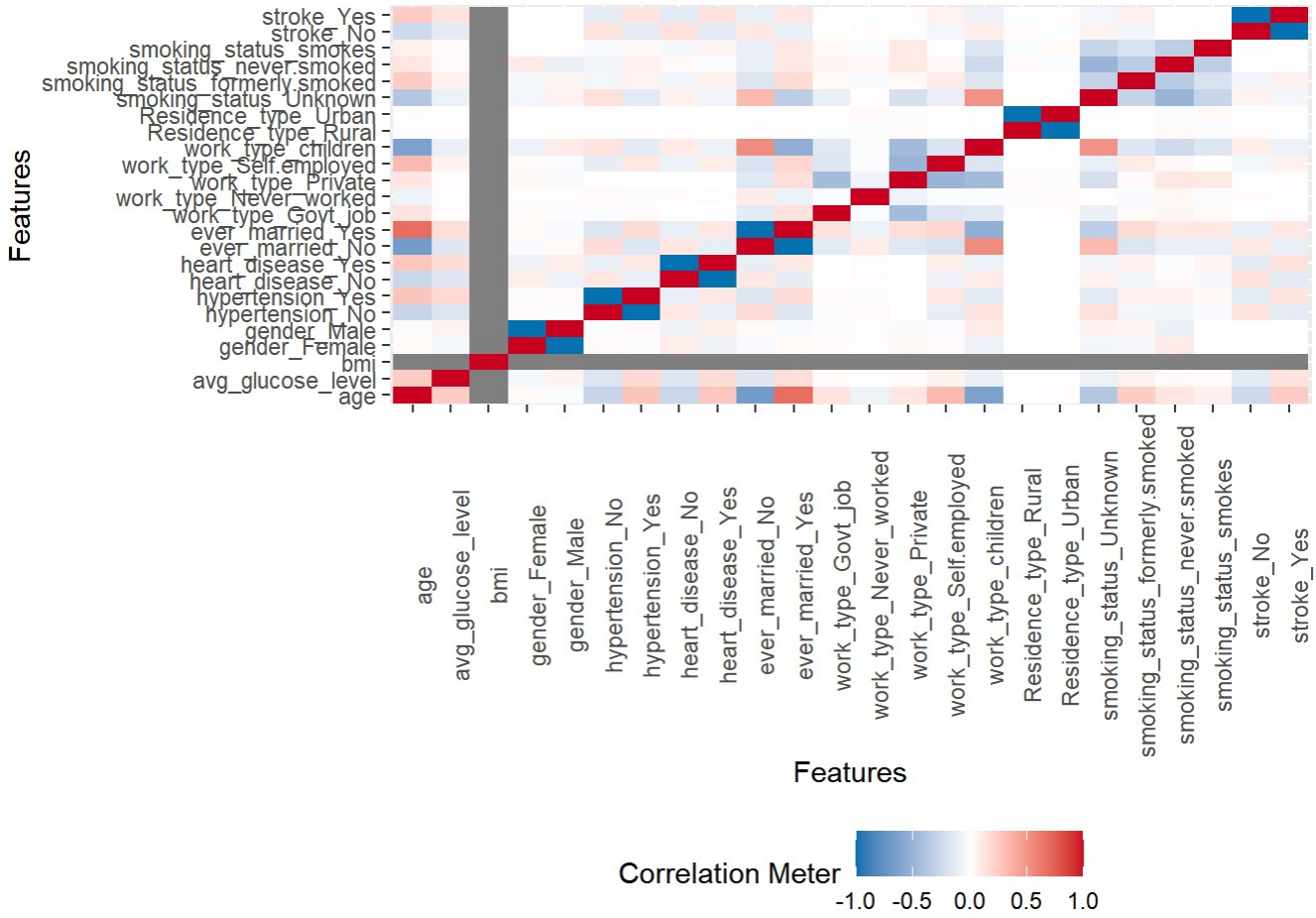

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 12 rows containing missing values (geom_bar).
```



Correlation Matrix

```
plot_correlation(Stroke)
```



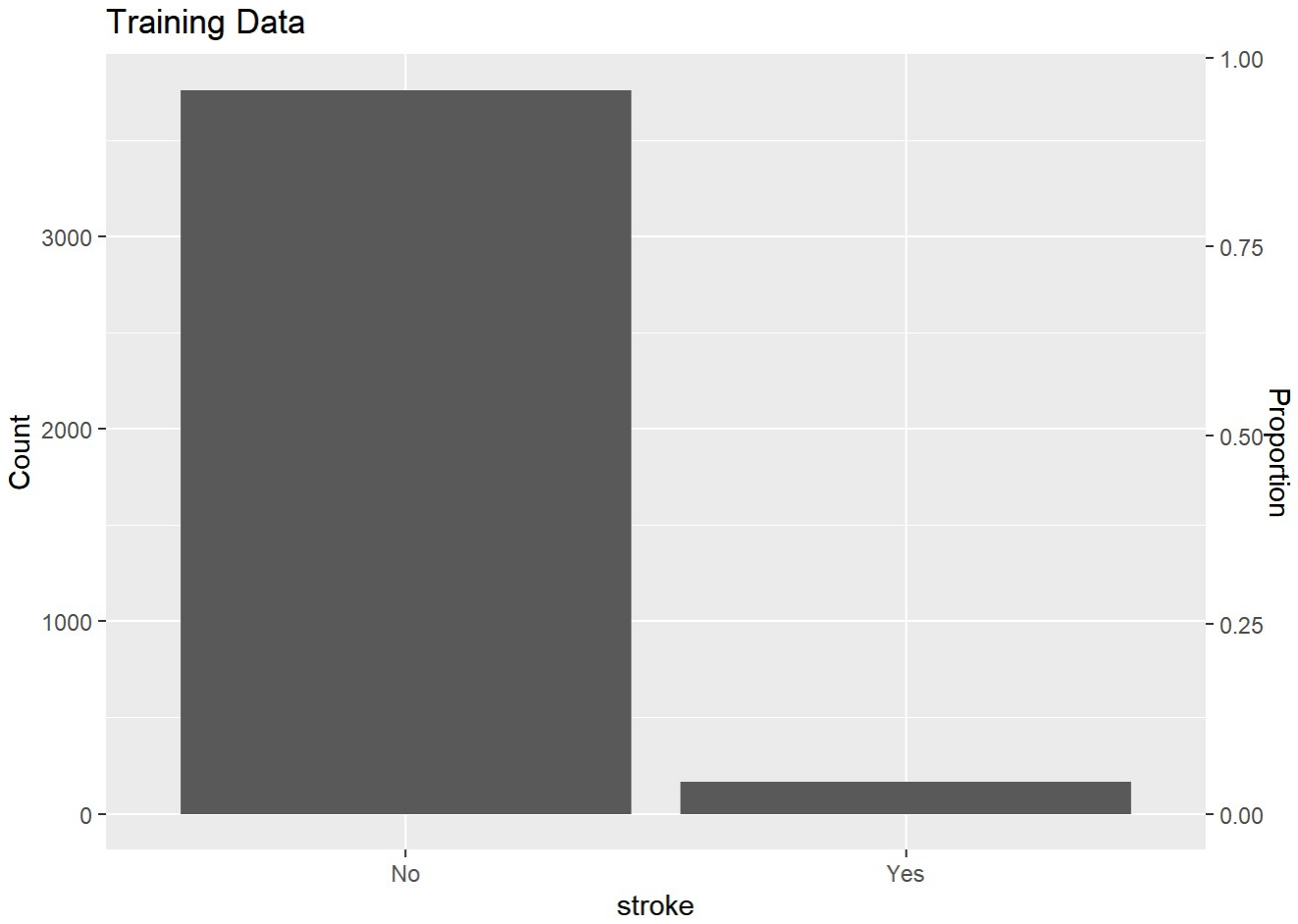
Data Preparation for modelling

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, ]
```

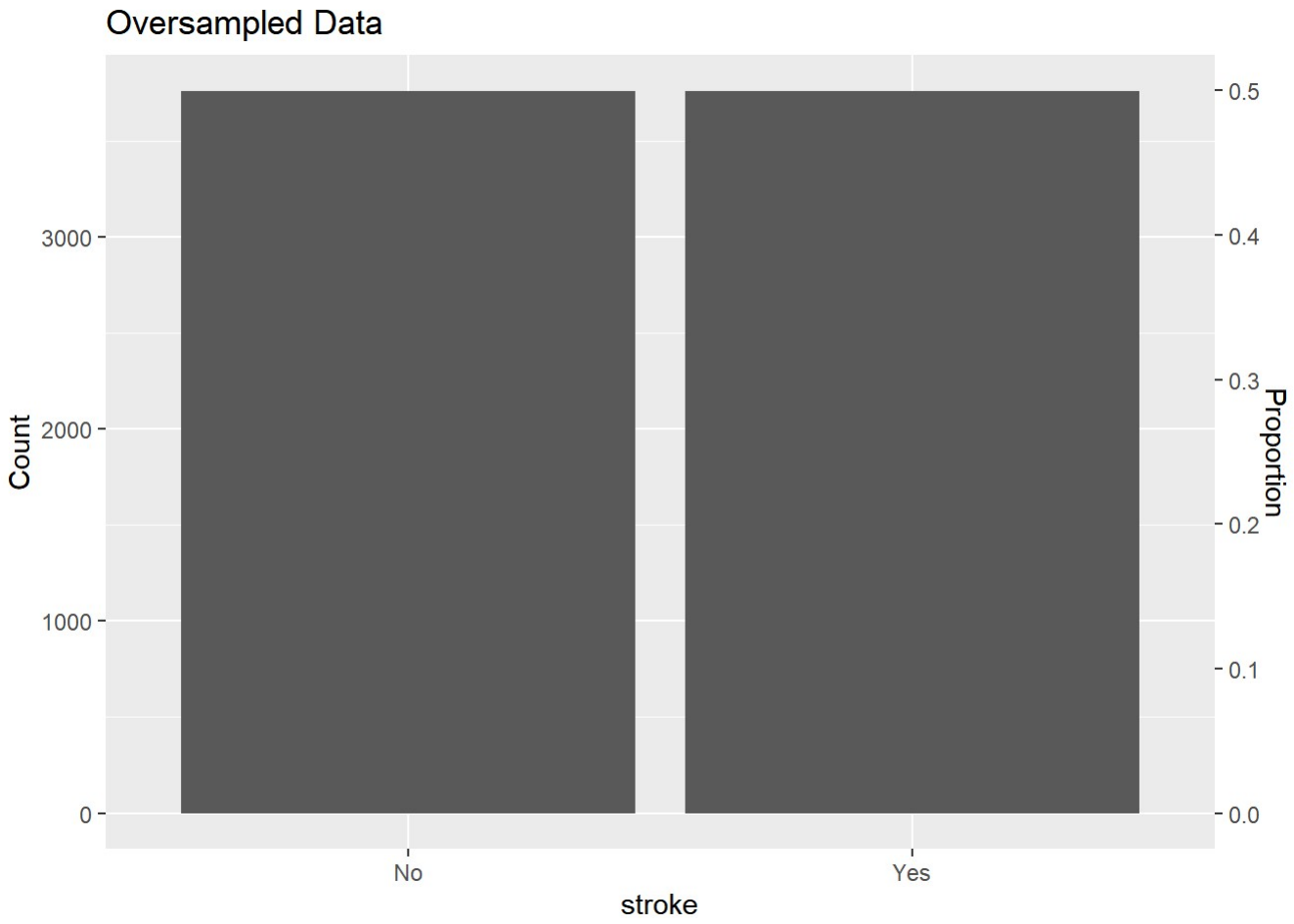
Visualize Stroke Balance in Training

```
#Dual Axis
ggplot(Stroke_tr, aes(x=stroke)) +
  geom_bar() +
  scale_y_continuous(
    name = "Count",
    sec.axis = sec_axis(~./nrow(Stroke_tr), name = "Proportion")
  ) + ggtitle("Training Data")
```



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 amount
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() +
  scale_y_continuous(
    name = "Count",
    sec.axis = sec_axis(~./nrow(Stroke_over), name = "Proportion")
  )  + ggtitle("Oversampled Data")
```



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, +13 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)
```

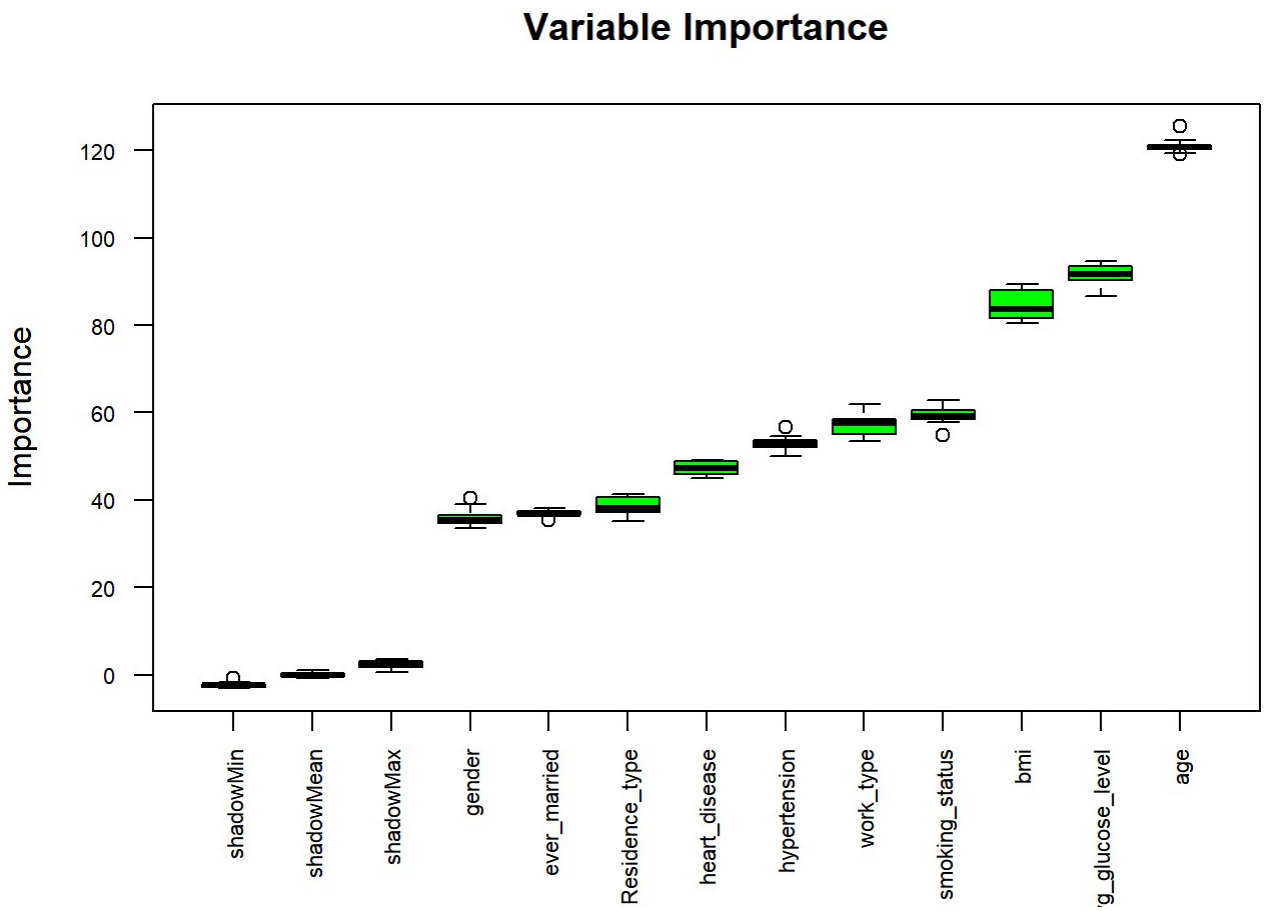
```
##  [1] "gender"            "hypertension"      "heart_disease"
##  [4] "ever_married"      "work_type"         "Residence_type"
##  [7] "smoking_status"    "age"               "avg_glucose_level"
## [10] "bmi"
```

```
imps <- attStats(boruta_out)
imps2 = imps[imps$decision != 'Rejected', c('meanImp', 'decision')]
imps2[order(-imps2$meanImp), ]
```

	meanImp <dbl>	decision <fct>
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

1-10 of 10 rows

```
plot(boruta_out, cex.axis=.7, las=2, xlab="", main="Variable Importance")
```



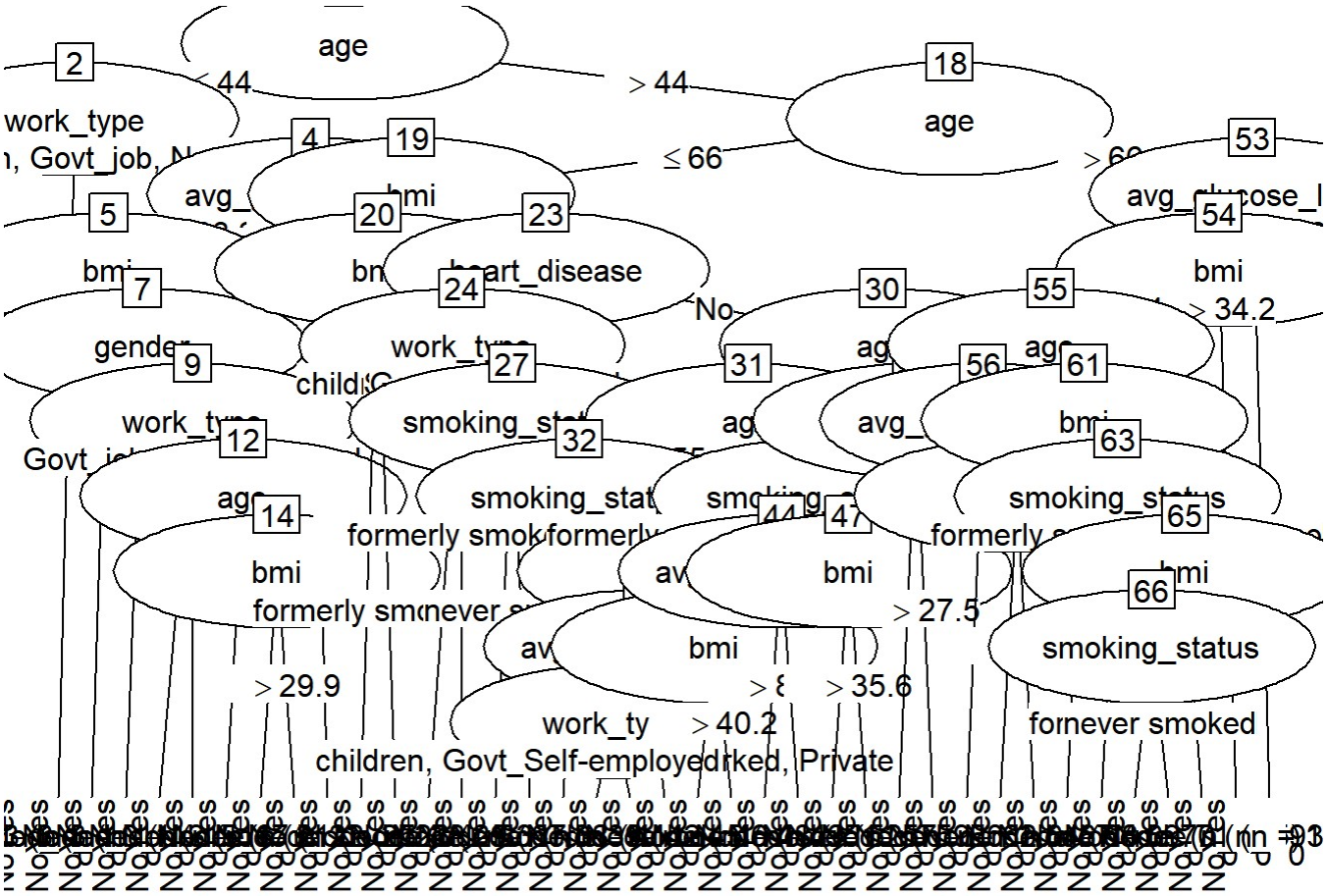

```
# All Variables were deemed important
```

Modeling

C5.0 - Andrew

```
library(C50)
C5 <- C5.0(formula = stroke ~ . , data = Stroke_over, control = C5.0Control(minCases = 75))
```

```
#Visualize the tree
plot(C5)
```




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

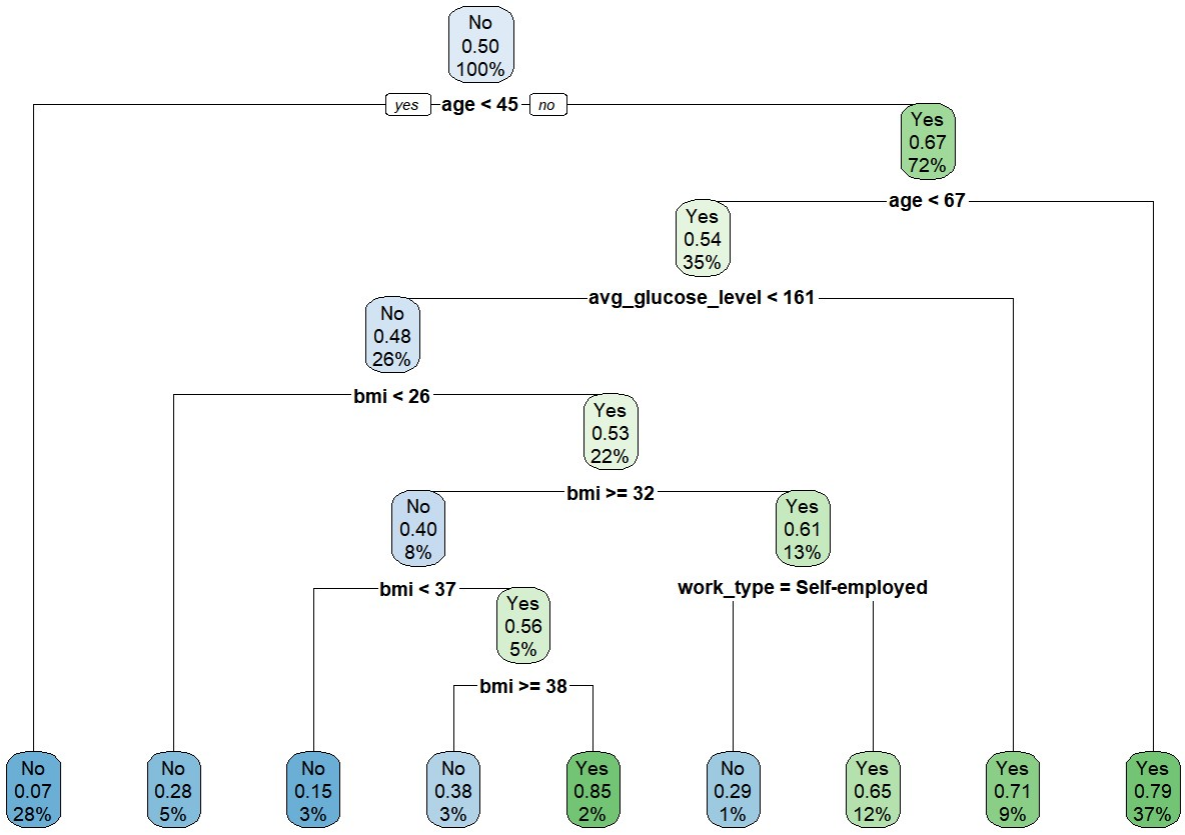
```
#Obtain model diagnostics
C5_bal <- confusionMatrix(data = predict(C5, Stroke_test), ref = Stroke_test$stroke, positive
= "Yes")
C5_bal
```

```
## 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)
```



Evaluate Model on Train and Test Data

```
train_cart <- confusionMatrix(data = predict(cart01, Stroke_over, type = "class"), ref = Stro
ke_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$stro
ke, 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_t
est$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)
```

	gen… <fct>	hypertension <fct>	heart_disease <fct>	ever_married <fct>	work_type <fct>	Residence_type <fct>	smo <fct>
1	Male	No	Yes	Yes	Private	Urban	form
3	Male	No	Yes	Yes	Private	Rural	neve
5	Female	Yes	No	Yes	Self-employed	Rural	neve

	gen… <fct>	hypertension <fct>	heart_disease <fct>	ever_married <fct>	work_type <fct>	Residence_type <fct>	smo <fct>
6	Male	No	No	Yes	Private	Urban	form
7	Male	Yes	Yes	Yes	Private	Rural	neve
11	Female	Yes	No	Yes	Private	Rural	neve

6 rows | 1-8 of 12 columns

```
Stroke_logreg_df <- subset(Stroke_tr_z_bal_lr, select = c("gender", "hypertension", "heart_di
sease", "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 training dataset target variables. Covert the probabilites to binary outputs in a new column called ‘pred’ on the Stroke_logreg_df dataframe:

```
# prediction
Stroke_logreg_df$pred_prob <- predict(object = logreg_stroke_subset, newdata = Stroke_logreg_
df, type='response')
Stroke_logreg_df$pred <- (Stroke_logreg_df$pred_prob > 0.5)*1
```

```
# Change pred variables to y/n
Stroke_logreg_df$pred[Stroke_logreg_df$pred=="1"]<-"Yes"
Stroke_logreg_df$pred[Stroke_logreg_df$pred=="0"]<-"No"
```

```
# create contingency table of predicted vs. actual
logreg_t1 <- table(Stroke_logreg_df$stroke, Stroke_logreg_df$pred)
row.names(logreg_t1) <- c("Actual: No", "Actual: Yes")
colnames(logreg_t1) <- c("Predicted: No", "Predicted: Yes")
logreg_t1 <- addmargins(A = logreg_t1, FUN = list(Total = sum), quiet = TRUE)
logreg_t1.df <- as.data.frame.matrix(logreg_t1)
logreg_t1.df
```

	Predicted: No <dbl>	Predicted: Yes <dbl>	Total <dbl>
Actual: No	2771	989	3760
Actual: Yes	799	2961	3760
Total	3570	3950	7520

3 rows

Evaluation metric for the logistic regression model

```
#calculate evaluation metrics
precision_lr <- logreg_t1.df[2,2] / logreg_t1.df[3,2]
Accuracy_lr <- (logreg_t1.df[1,1] + logreg_t1.df[2,2])/logreg_t1.df[3,3]
error_rate_lr <- 1 - Accuracy_lr
sensitivity_lr <- logreg_t1.df[2,2]/logreg_t1.df[1,3]
specificity_lr <- logreg_t1.df[1,1]/logreg_t1.df[1,3]
f1_lr <- 2*((precision_lr * specificity_lr)/(precision_lr+specificity_lr))
f2_lr <- 5*((precision_lr*specificity_lr)/((4*precision_lr)+specificity_lr))
f0.5_lr <- 1.25*((precision_lr * specificity_lr)/((0.25*precision_lr)*specificity_lr))
```

```
#create dataframe for evaluation metrics
eval.dflr <- data.frame(eval.measure = c("Accuracy", "error.rate", "sensitivity", "specificit
y", "precision", "f1", "f2", "f0.5"))
eval.dflr$model.lr <- round(c(Accuracy_lr, error_rate_lr, sensitivity_lr, specificity_lr, pre
cision_lr, f1_lr, f2_lr, f0.5_lr), 2)
eval.dflr
```

eval.measure <chr>	model.lr <dbl>
Accuracy	0.76
error.rate	0.24
sensitivity	0.79
specificity	0.74
precision	0.75
f1	0.74
f2	0.74
f0.5	5.00

8 rows

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$stroke, k=10)
rf_stroke_bal <- caret::train(stroke~., method="rf", data = Stroke_over, tuneLength = 5, trCo
ntrol = 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    0.9047872  0.8095745
##    5    0.9990691  0.9981383
##    8    0.9989362  0.9978723
##   11    0.9988032  0.9976064
##   15    0.9988032  0.9976064
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 5.
```

Confusion Matrices

```
rf_reg <- confusionMatrix(data = predict(rf_stroke, Stroke_test), ref = Stroke_test$stroke, p
ositive = "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$strok
e, positive = "Yes")
rf_bal
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  929  41
##        Yes  10   0
##
##                Accuracy : 0.948
##                  95% CI : (0.9321, 0.961)
##     No Information Rate : 0.9582
##     P-Value [Acc > NIR] : 0.9492
##
##                   Kappa : -0.0167
##
##  Mcnemar's Test P-Value : 2.659e-05
##
##             Sensitivity : 0.00000
##             Specificity : 0.98935
##          Pos Pred Value : 0.00000
##          Neg Pred Value : 0.95773
##              Prevalence : 0.04184
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.01020
##       Balanced Accuracy : 0.49468
##
##        '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
```

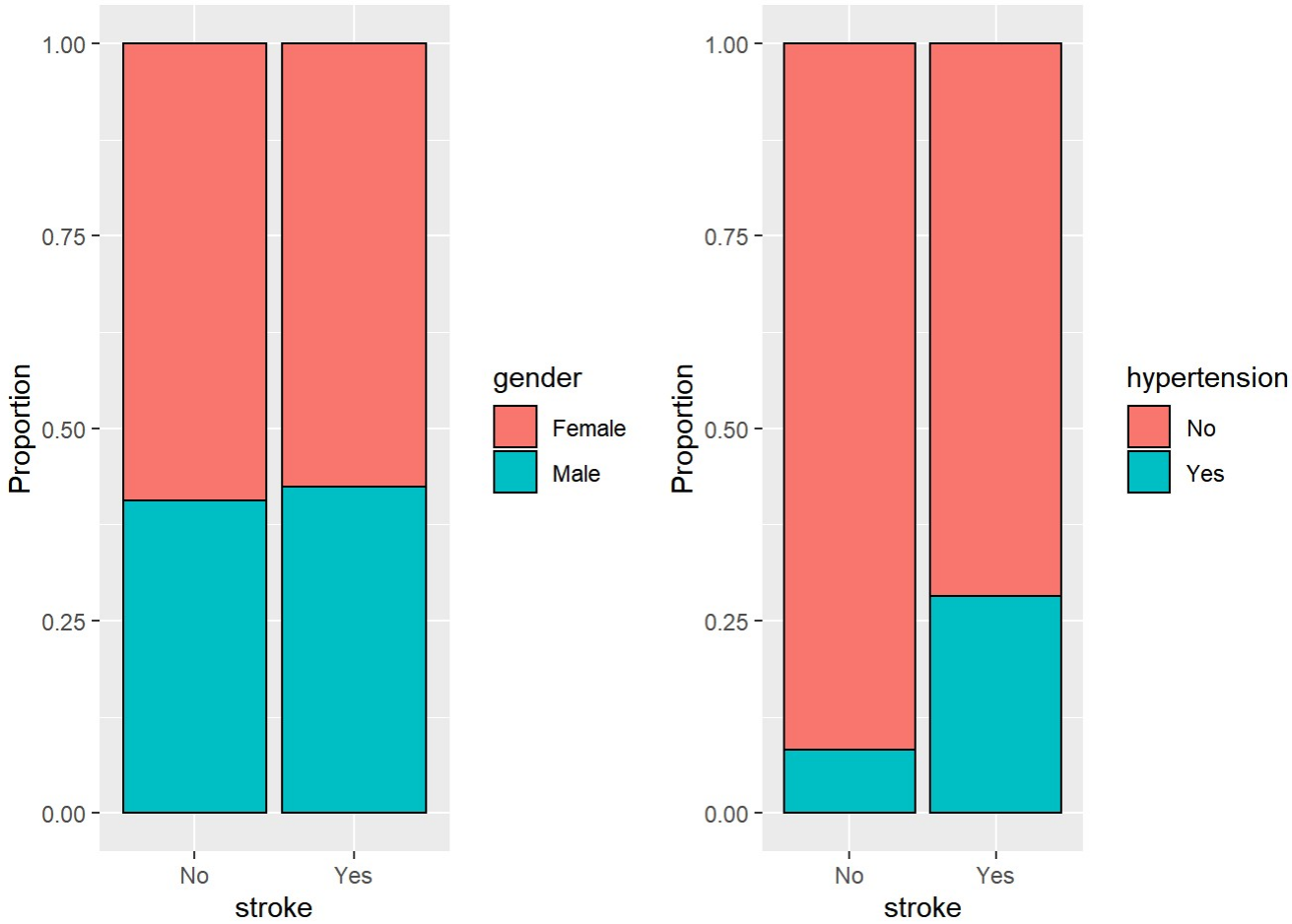
Gridlines for each variable in association with “stroke”.

Plot Function

```
nb_plot <- function(x, y){
  ggplot(x, aes(stroke)) + geom_bar(aes(fill=x[,y]), position = "fill", color = "black") + yl
ab("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)
```



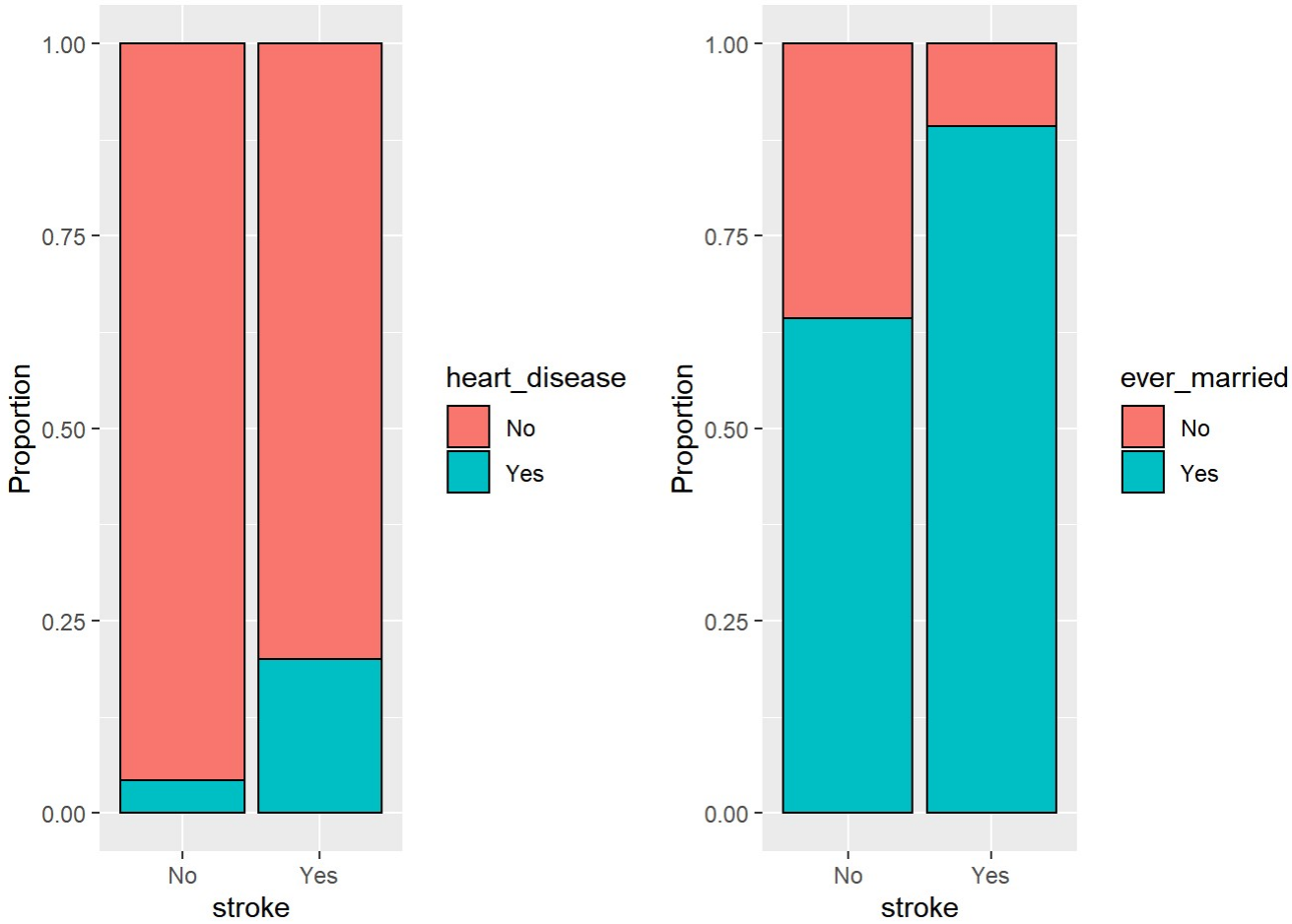
Run the Naive Bayes estimator for “stroke” in association with “gender” and “hypertension”.

```
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)
```



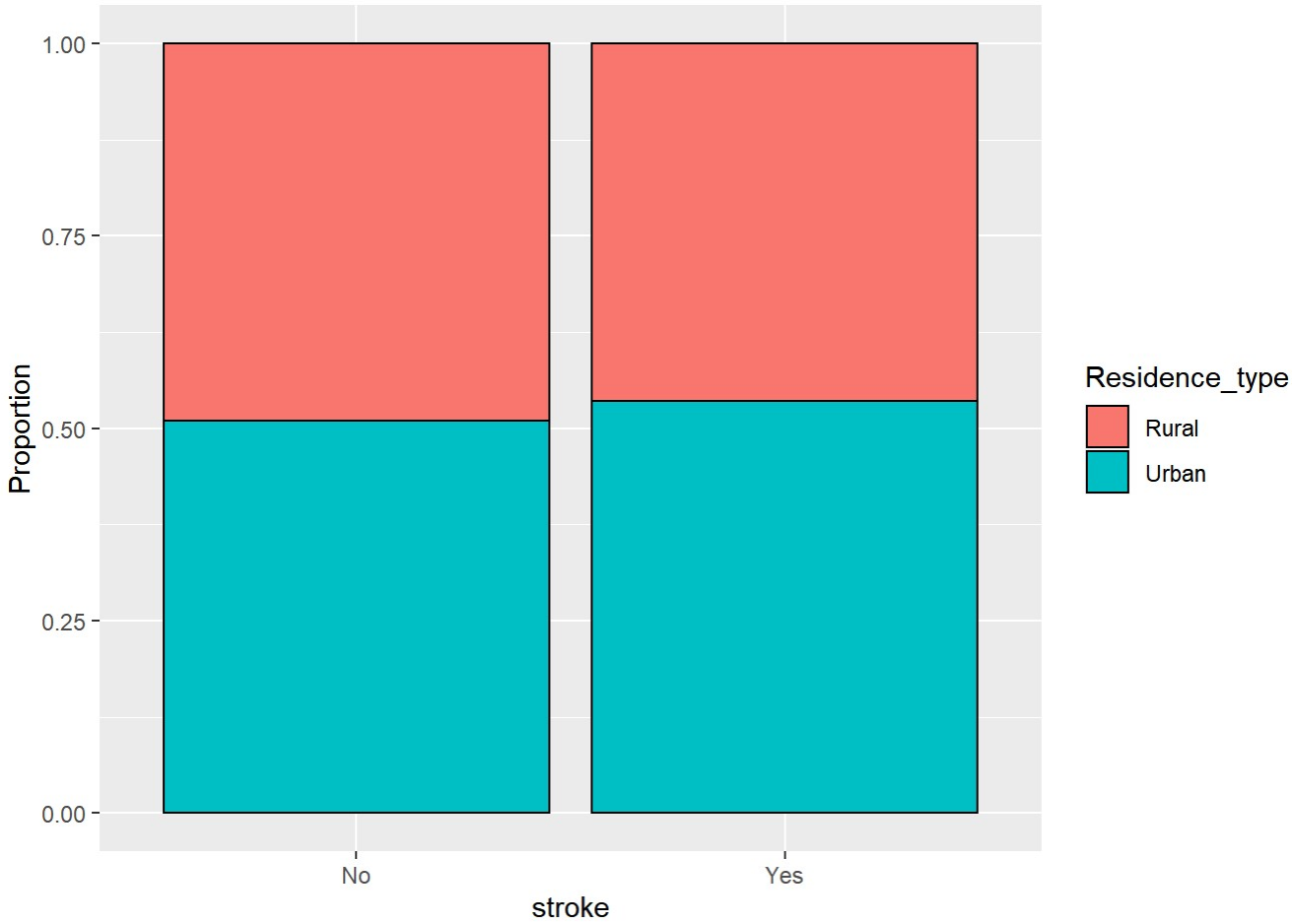
Run the Naive Bayes estimator for “stroke” in association with “heart disease” and “ever married”.

```
nb02 <- naiveBayes(formula = stroke ~ heart_disease + ever_married, data = Stroke_over)
nb_heart_marry <- confusionMatrix(data = predict(nb02, Stroke_test, type = "class"), ref = St
roke_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"))
```



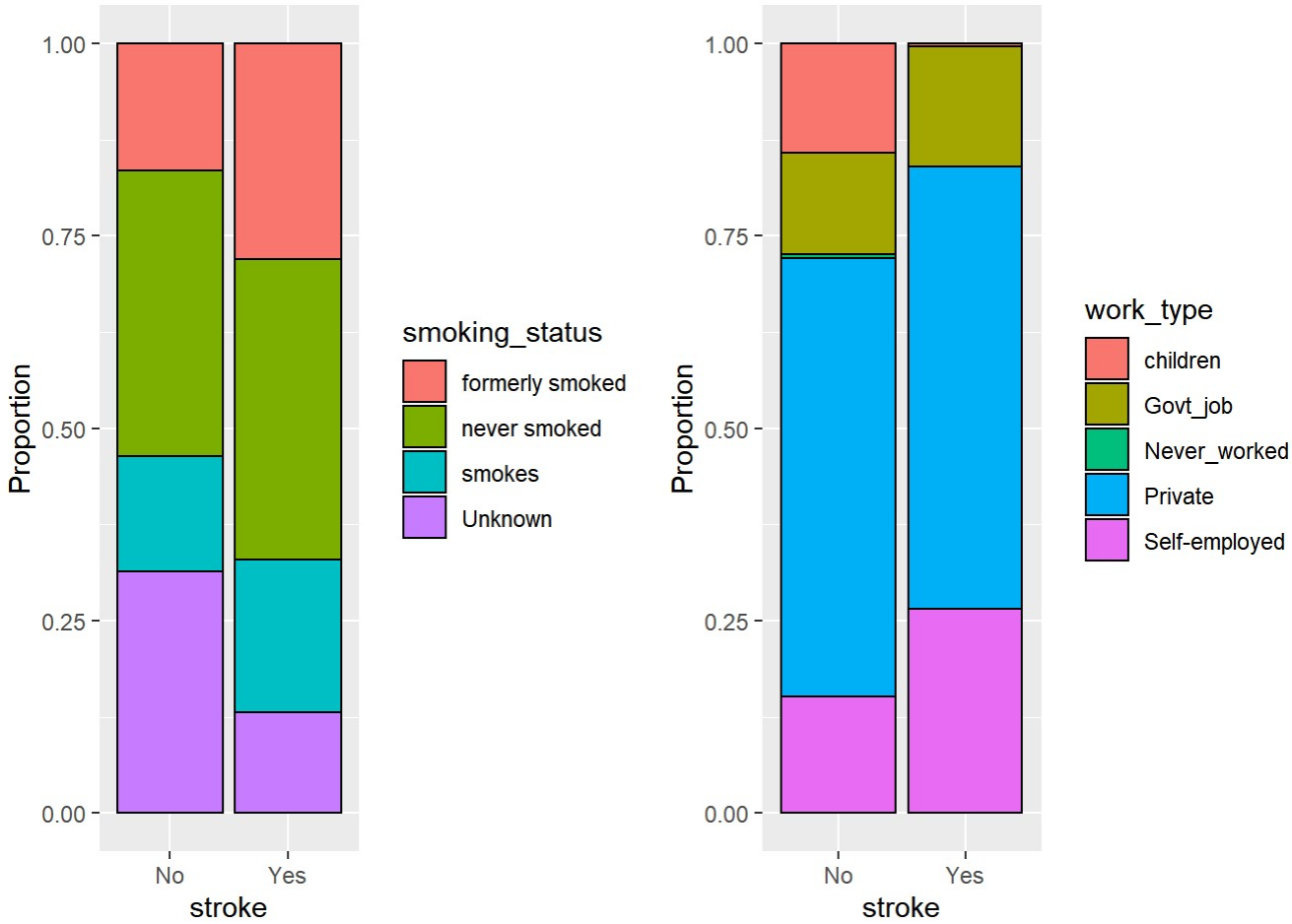
Run the Naive Bayes estimator for “stroke” in association with “Residence type”.

```
nb03 <- naiveBayes(formula = stroke ~ Residence_type, data = Stroke_over)
nb_Res <- confusionMatrix(data = predict(nb03, Stroke_test, type = "class"), ref = Stroke_tes
t$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)
```



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 = Str
oke_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

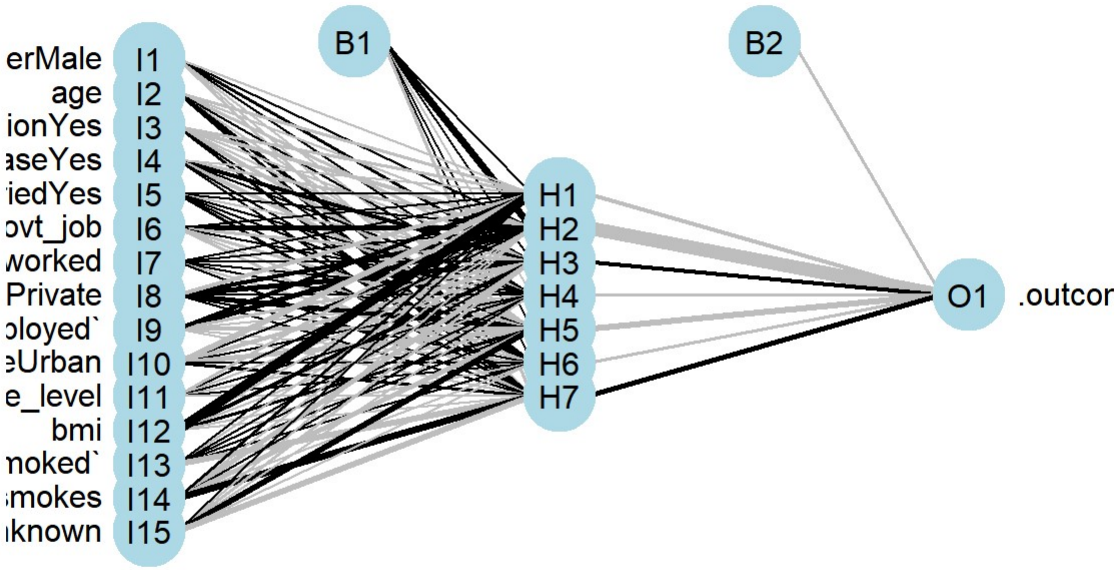
```
#Unbalanced
train <- createFolds(Stroke_tr$stroke, k=10)
nnet_stroke <- caret::train(stroke ~ ., method = "nnet", data = Stroke_tr,
    tuneLength = 5,
    trControl = trainControl(
        method = "cv", indexOut = train),
  trace = FALSE)

nnet_stroke
```

```
## Neural Network
##
## 3928 samples
##   10 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 3536, 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  0.9564678   0.0072675790
##   3     1e-04  0.9572311   0.0000000000
##   3     1e-03  0.9572311   0.0000000000
##   3     1e-02  0.9572311   0.0000000000
##   3     1e-01  0.9572311   0.0000000000
##   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   0.0199028777
##   5     1e-01  0.9574856   0.0208806818
##   7     0e+00  0.9572311   0.0000000000
##   7     1e-04  0.9572311   0.0000000000
##   7     1e-03  0.9569767  -0.0004829545
##   7     1e-02  0.9572311   0.0000000000
##   7     1e-01  0.9577400   0.0213636364
##   9     0e+00  0.9572311   0.0000000000
##   9     1e-04  0.9572311   0.0000000000
##   9     1e-03  0.9572311   0.0000000000
##   9     1e-02  0.9574856   0.0327937857
##   9     1e-01  0.9574856   0.0203858322
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 7 and decay = 0.1.
```

```
nnet_stroke$finalModel
```

```
## a 15-7-1 network with 120 weights
## inputs: genderMale age hypertensionYes heart_diseaseYes ever_marriedYes work_typeGovt_job
work_typeNever_worked work_typePrivate `work_typeSelf-employed` Residence_typeUrban avg_gluco
se_level bmi `smoking_statusnever smoked` smoking_statussmokes smoking_statusUnknown
## output(s): .outcome
## options were - entropy fitting  decay=0.1
```

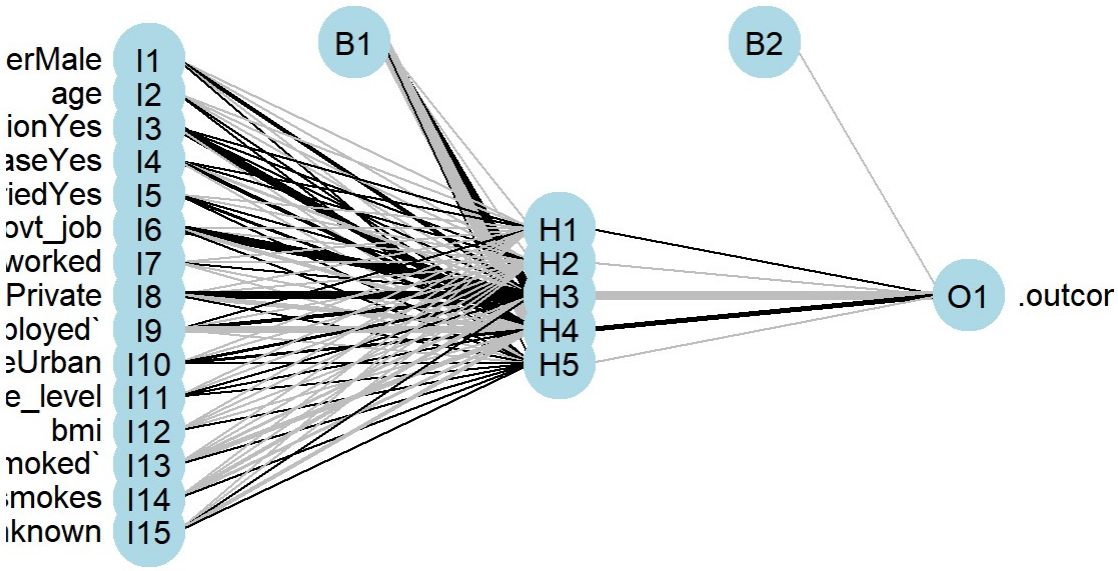
```
plotnet(nnet_stroke$finalModel)
```




```
#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  0.5864362  0.1728723
##   1     1e-04  0.6037234  0.2074468
##   1     1e-03  0.6046543  0.2093085
##   1     1e-02  0.6688830  0.3377660
##   1     1e-01  0.7614362  0.5228723
##   3     0e+00  0.6345745  0.2691489
##   3     1e-04  0.7041223  0.4082447
##   3     1e-03  0.7397606  0.4795213
##   3     1e-02  0.7648936  0.5297872
##   3     1e-01  0.7752660  0.5505319
##   5     0e+00  0.7021277  0.4042553
##   5     1e-04  0.6893617  0.3787234
##   5     1e-03  0.7771277  0.5542553
##   5     1e-02  0.7784574  0.5569149
##   5     1e-01  0.7898936  0.5797872
##   7     0e+00  0.7735372  0.5470745
##   7     1e-04  0.7765957  0.5531915
##   7     1e-03  0.7632979  0.5265957
##   7     1e-02  0.7807181  0.5614362
##   7     1e-01  0.7865691  0.5731383
##   9     0e+00  0.7880319  0.5760638
##   9     1e-04  0.7480053  0.4960106
##   9     1e-03  0.7703457  0.5406915
##   9     1e-02  0.7815160  0.5630319
##   9     1e-01  0.7836436  0.5672872
##
## 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.
```

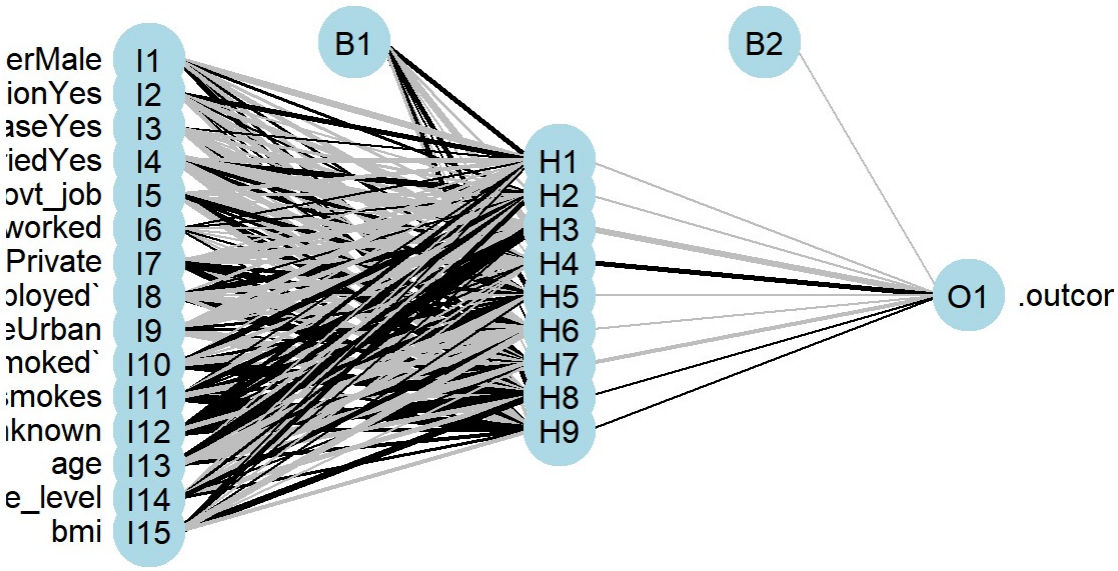
```
plotnet(nnet_stroke_balanced$finalModel)
```




```
#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, 3536, 3535, 3536, 3535, 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  0.9574856  0.02865220
##   3     1e-04  0.9574856  0.01068182
##   3     1e-03  0.9572311  0.00000000
##   3     1e-02  0.9572311  0.01019886
##   3     1e-01  0.9579945  0.06120925
##   5     0e+00  0.9577433  0.09128618
##   5     1e-04  0.9572318  0.06899412
##   5     1e-03  0.9579945  0.04174947
##   5     1e-02  0.9577400  0.07482302
##   5     1e-01  0.9567209  0.08353387
##   7     0e+00  0.9587605  0.06557616
##   7     1e-04  0.9592668  0.13753743
##   7     1e-03  0.9579945  0.05204695
##   7     1e-02  0.9597757  0.16249932
##   7     1e-01  0.9602859  0.18264625
##   9     0e+00  0.9605416  0.20227752
##   9     1e-04  0.9605384  0.22918691
##   9     1e-03  0.9595232  0.12795940
##   9     1e-02  0.9602839  0.18875636
##   9     1e-01  0.9602865  0.19182379
##
## 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.
```

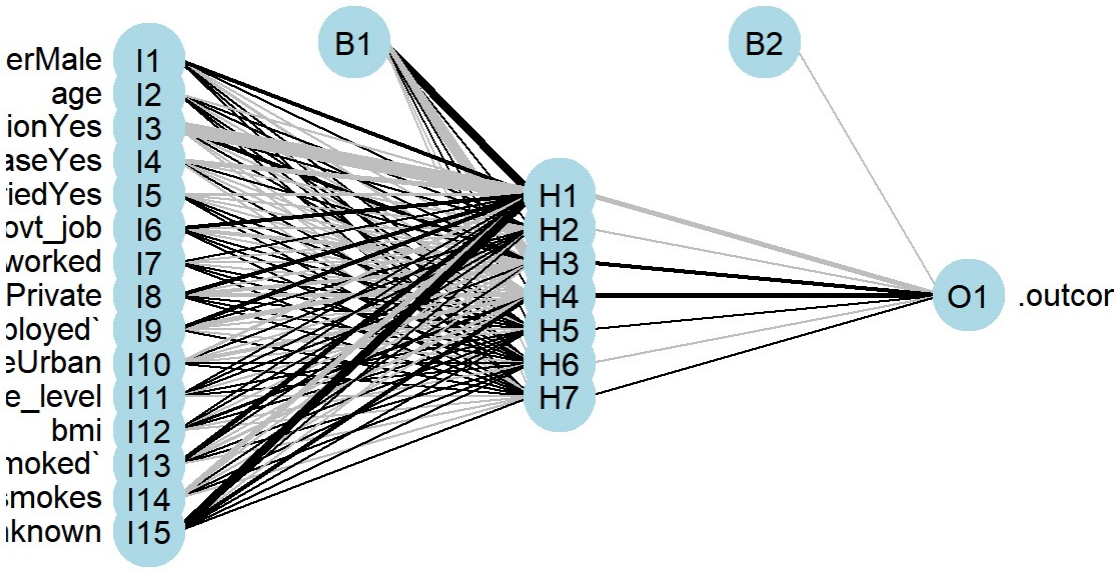
```
plotnet(nnet_stroke_z$finalModel)
```




```
#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  0.4454815  0.02623893
##   1     1e-04  0.5165609  0.02011860
##   1     1e-03  0.5445085  0.03335283
##   1     1e-02  0.6270590  0.10627450
##   1     1e-01  0.7105293  0.13800038
##   3     0e+00  0.6186452  0.11623991
##   3     1e-04  0.4751000  0.05919859
##   3     1e-03  0.5998773  0.07877419
##   3     1e-02  0.6829977  0.13257871
##   3     1e-01  0.6983084  0.13277474
##   5     0e+00  0.6881199  0.13766258
##   5     1e-04  0.5963461  0.11126958
##   5     1e-03  0.7286513  0.15696996
##   5     1e-02  0.6965623  0.14417382
##   5     1e-01  0.7454218  0.15408060
##   7     0e+00  0.6603501  0.12910040
##   7     1e-04  0.6714299  0.08810738
##   7     1e-03  0.6487148  0.12737101
##   7     1e-02  0.7163869  0.14580233
##   7     1e-01  0.7512593  0.16357615
##   9     0e+00  0.6993360  0.13641240
##   9     1e-04  0.6456568  0.14594495
##   9     1e-03  0.6960573  0.14016581
##   9     1e-02  0.7072610  0.13816701
##   9     1e-01  0.7329491  0.15837911
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 7 and decay = 0.1.
```

```
plotnet(nnet_stroke_z_bal$finalModel)
```



Evaluate NN

```
nnet_reg <- confusionMatrix(data = predict(nnet_stroke, Stroke_test), ref = Stroke_test$strok
e, 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_t
est$stroke, positive = "Yes")
nnet_bal
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  712   7
##        Yes 227  34
##
##                Accuracy : 0.7612
##                  95% CI : (0.7333, 0.7876)
##     No Information Rate : 0.9582
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1648
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.82927
##             Specificity : 0.75825
##          Pos Pred Value : 0.13027
##          Neg Pred Value : 0.99026
##              Prevalence : 0.04184
##          Detection Rate : 0.03469
##    Detection Prevalence : 0.26633
##       Balanced Accuracy : 0.79376
##
##        'Positive' Class : Yes
##
```

```
nnet_z <- confusionMatrix(data = predict(nnet_stroke_z, Stroke_test), ref = Stroke_test$strok
e, 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_te
st$stroke, positive = "Yes")
nnet_z_bal
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  697   5
##        Yes 242  36
##
##                Accuracy : 0.748
##                  95% CI : (0.7195, 0.7749)
##     No Information Rate : 0.9582
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1648
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.87805
##             Specificity : 0.74228
##          Pos Pred Value : 0.12950
##          Neg Pred Value : 0.99288
##              Prevalence : 0.04184
##          Detection Rate : 0.03673
##    Detection Prevalence : 0.28367
##       Balanced Accuracy : 0.81016
##
##        'Positive' Class : Yes
##
```

Model Evaluation

Add Baseline

```
Stroke_count <- Stroke_test %>% group_by(stroke) %>% tally()
Stroke_count
```

stroke <fct>	n <int>
No	939
Yes	41

2 rows

```
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)
```

```
#Get LogReg Model
LogReg <- pivot_wider(eval.dflr, values_from = model.lr, names_from = eval.measure)
colnames(LogReg) <- str_to_title(colnames(LogReg))
LogReg <- LogReg %>% dplyr::select(Accuracy, Sensitivity, Specificity, Precision, F1)

Model_comp <- rbind(`ANN Reg.`, `ANN Bal.`, `ANN Z`, `ANN Z Bal.`, `RF Reg.`, `RF Bal.`, `CAR
T Bal.`, `CART Cost`, `CART Cost Bal.`, `C5.0 Bal.`, `NB Gender`, `NB Heart + Marry`, `NB Res
ident`, `NB Smoke + Work`)
Model_comp <- data.frame(Model_comp)

Model_comp <- Model_comp %>% dplyr::select(Accuracy, Sensitivity, Specificity, Precision, F1)
Model_comp <- rbind(Baseline, LogReg, Model_comp)

rownames(Model_comp)[rownames(Model_comp) == 1] <- "Baseline"
```

```
## Warning: Setting row names on a tibble is deprecated.
```

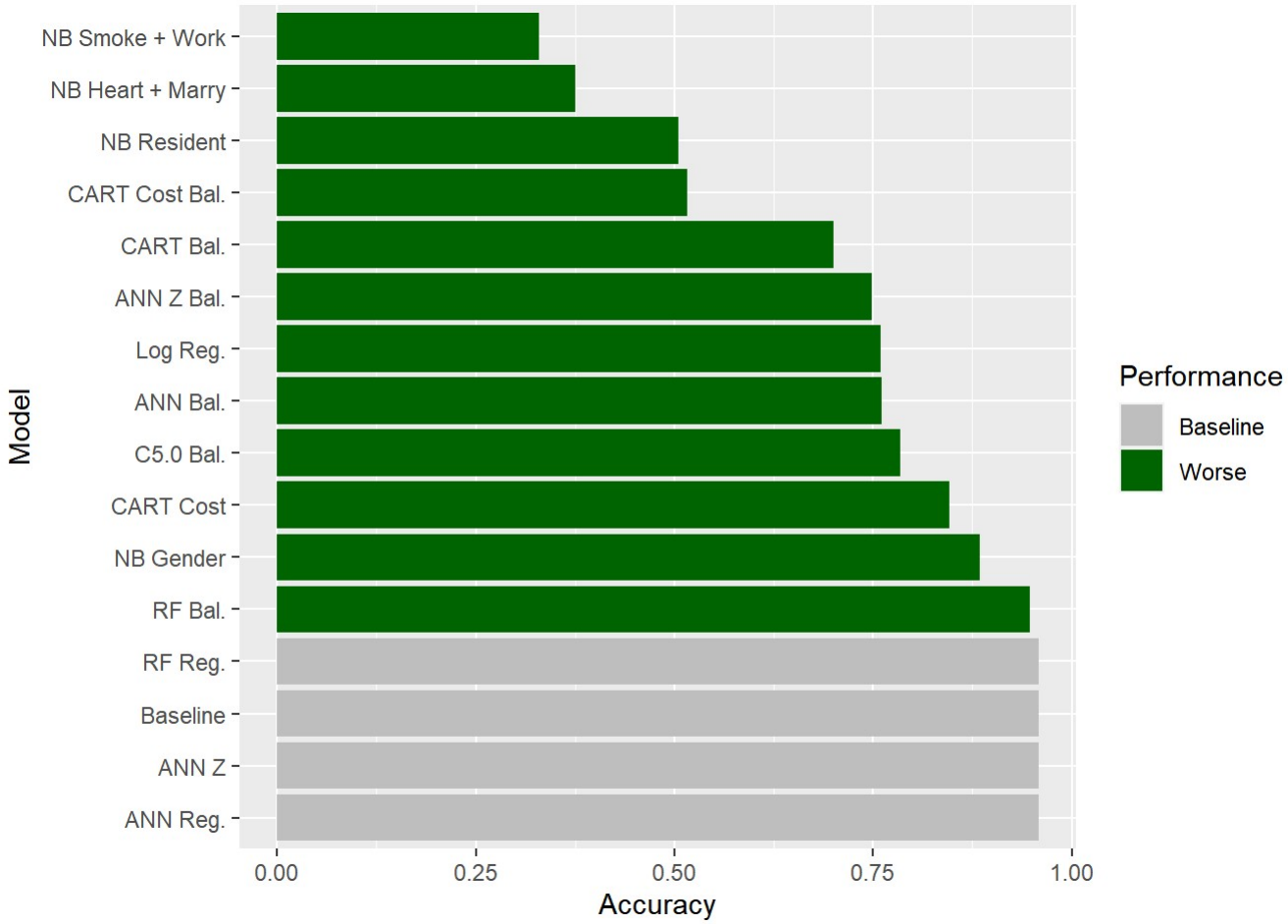
```
rownames(Model_comp)[rownames(Model_comp) == 2] <- "Log Reg."
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```
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
```

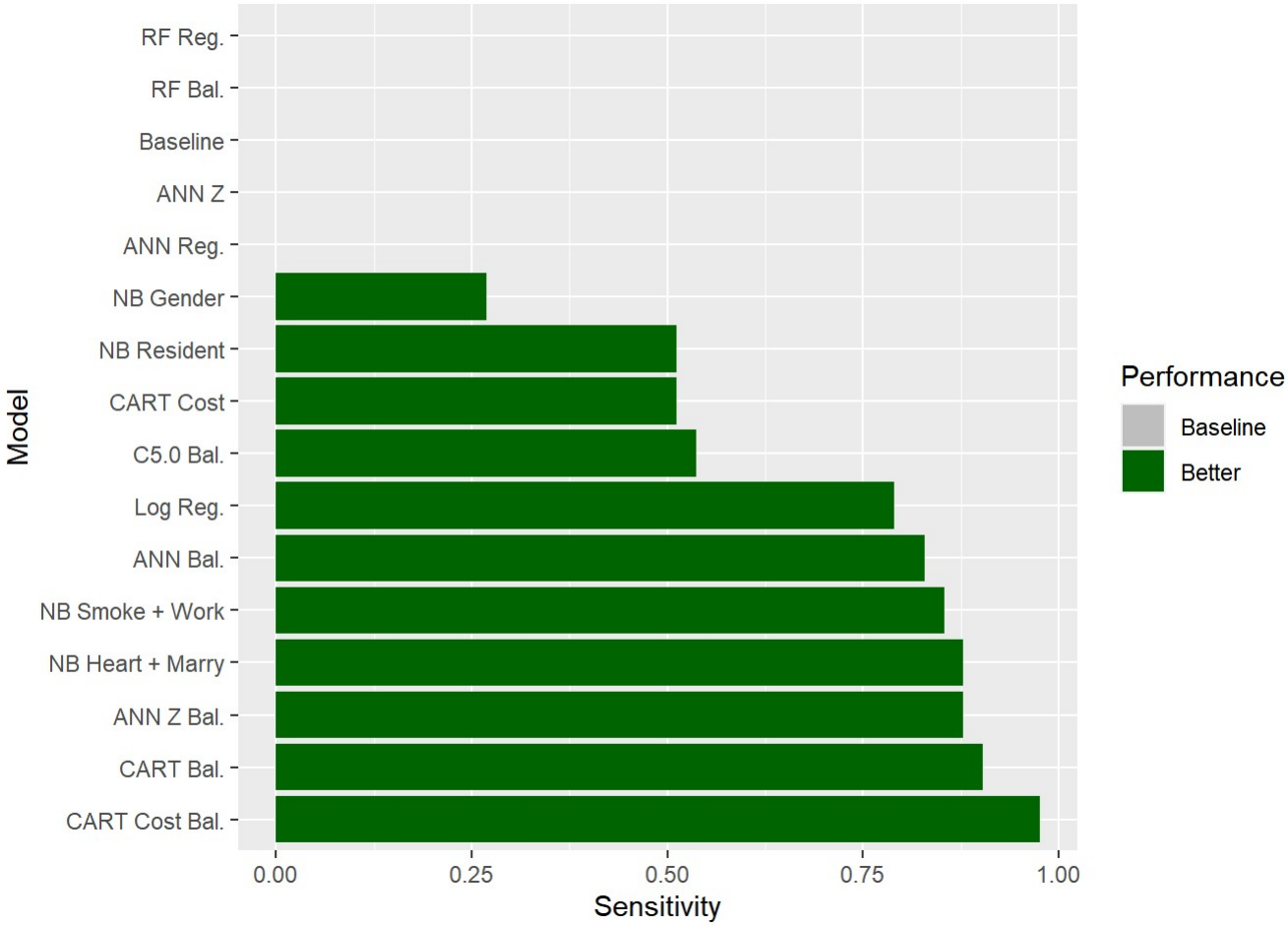
Accuracy Graph

```
Model_comp %>% mutate(Model = fct_reorder(Model, desc(Accuracy))) %>% mutate(Performance = if
else(Model_comp$Accuracy == Accuracy_base, "Baseline", ifelse(Model_comp$Accuracy < Accuracy_
base, "Worse", "Better"))) %>% ggplot(aes(x=Model, y=Accuracy, fill = Performance)) + geom_ba
r(stat = "identity") + coord_flip() + scale_fill_manual(values = c("grey", "darkgreen", "#c12
503"))
```



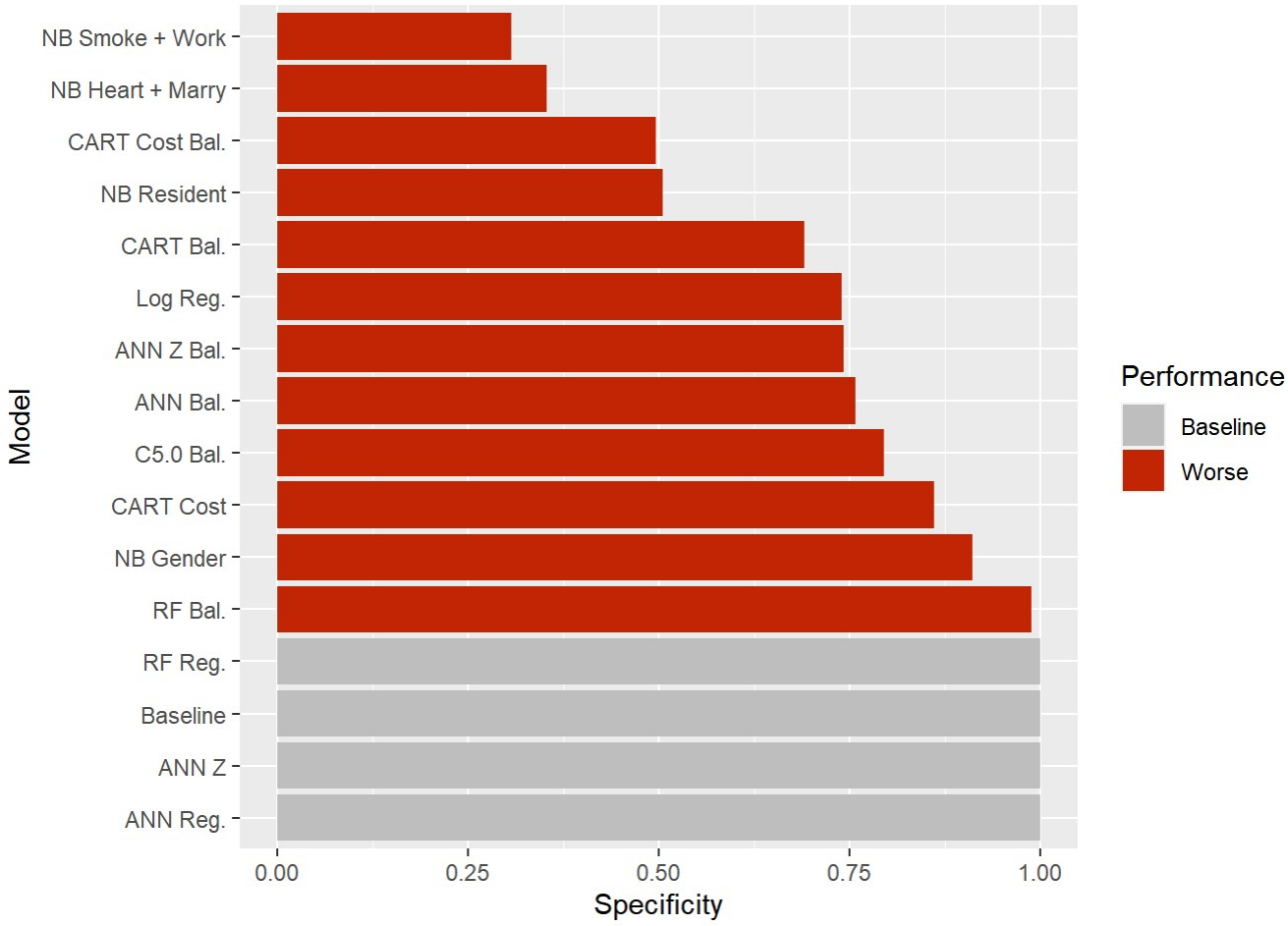
Sensitivity

```
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") + coo
rd_flip() + scale_fill_manual(values = c("Grey", "darkgreen"))
```



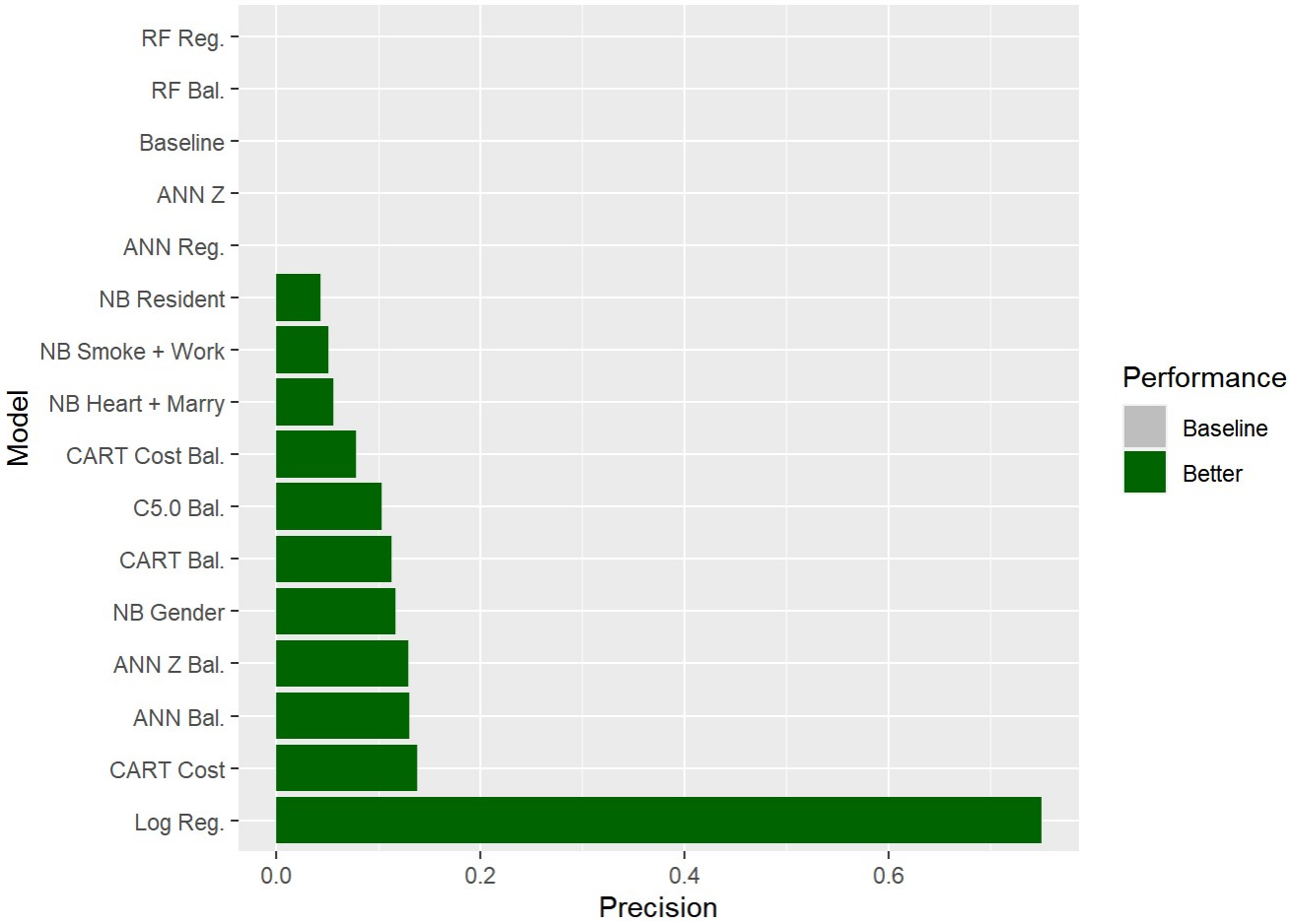
Specificity

```
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") + coo
rd_flip() + scale_fill_manual(values = c("Grey", "#c12503"))
```



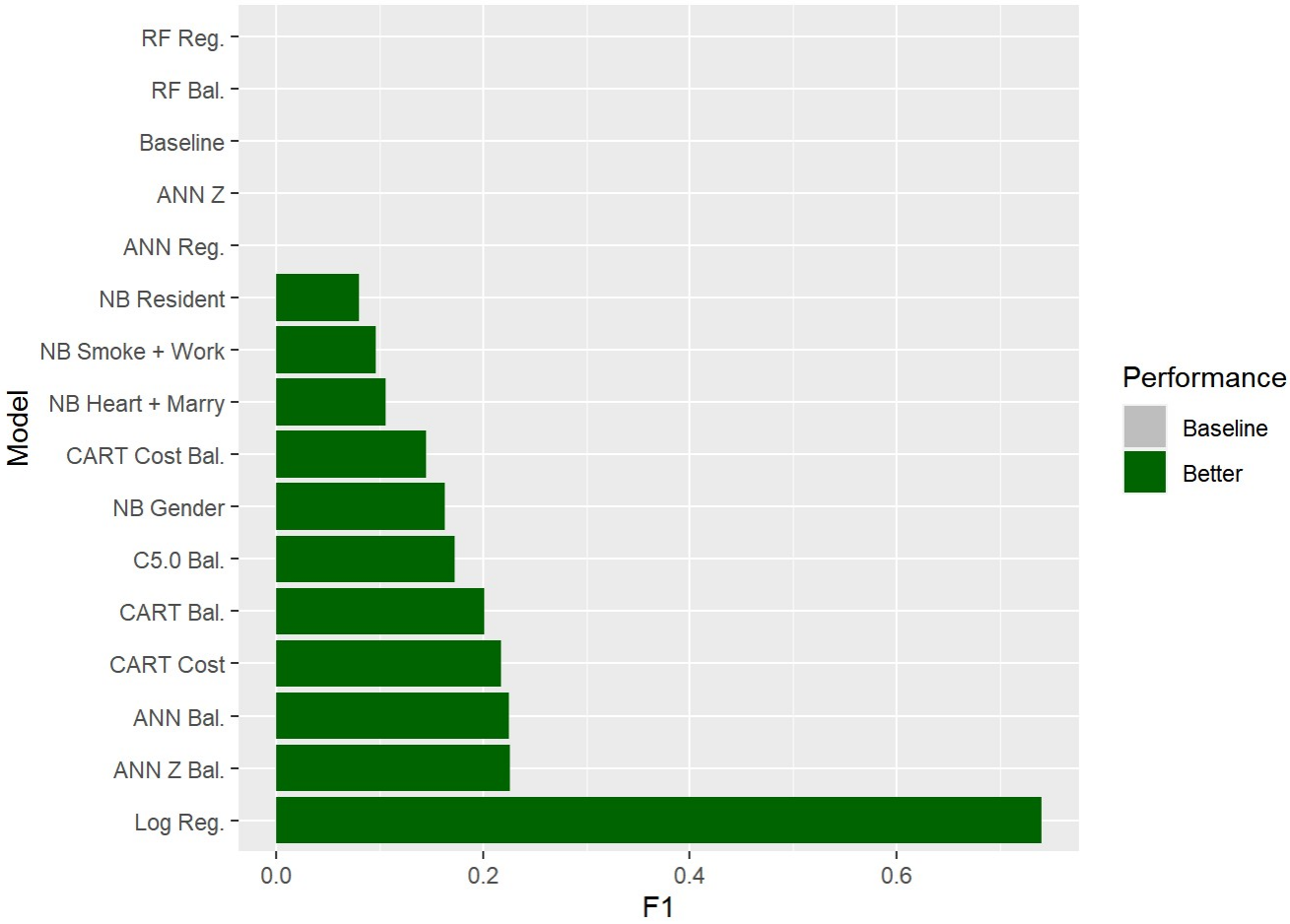
Precision

```
Model_comp %>% mutate(Model = fct_reorder(Model, desc(Precision))) %>% mutate(Performance = i
felse(Model_comp$Precision == 0, "Baseline", ifelse(Model_comp$Precision < 0, "Worse", "Bette
r"))) %>% ggplot(aes(x=Model, y=Precision, fill = Performance)) + geom_bar(stat = "identity")
+ coord_flip() + scale_fill_manual(values = c("Grey", "darkgreen"))
```



F1

```
Model_comp %>% mutate(Model = fct_reorder(Model, desc(F1))) %>% mutate(Performance = ifelse(M
odel_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"))
```



Comparison Table

```
kable(Model_comp, table.attr = "style = \"color: black;\"") %>% kable_material(c("striped", "hover")) %>% column_spec(1, bold = T, border_right = T)
```

Model	Accuracy	Sensitivity	Specificity	Precision	F1
Baseline	0.9581633	0.0000000	1.0000000	0.0000000	0.0000000
Log Reg.	0.7600000	0.7900000	0.7400000	0.7500000	0.7400000
ANN Reg.	0.9581633	0.0000000	1.0000000	0.0000000	0.0000000
ANN Bal.	0.7612245	0.8292683	0.7582535	0.1302682	0.2251656
ANN Z	0.9581633	0.0000000	1.0000000	0.0000000	0.0000000

Model	Accuracy	Sensitivity	Specificity	Precision	F1
ANN Z Bal.	0.7479592	0.8780488	0.7422790	0.1294964	0.2257053
RF Reg.	0.9581633	0.0000000	1.0000000	0.0000000	0.0000000
RF Bal.	0.9479592	0.0000000	0.9893504	0.0000000	0.0000000
CART Bal.	0.7000000	0.9024390	0.6911608	0.1131498	0.2010870
CART Cost	0.8459184	0.5121951	0.8604899	0.1381579	0.2176166
CART Cost Bal.	0.5163265	0.9756098	0.4962726	0.0779727	0.1444043
C5.0 Bal.	0.7846939	0.5365854	0.7955272	0.1028037	0.1725490
NB Gender	0.8846939	0.2682927	0.9116081	0.1170213	0.1629630
NB Heart + Marry	0.3755102	0.8780488	0.3535676	0.0559876	0.1052632
NB Resident	0.5051020	0.5121951	0.5047923	0.0432099	0.0796964
NB Smoke + Work	0.3295918	0.8536585	0.3067093	0.0510204	0.0962861