

Project

Victoria (Tori) Widjaja & Jeremiah Fa'atiligā

2024-06-19

R Markdown

```
library(dplyr)
library(tidyr)
library(tibble)
library(readr)
library(tidyverse)
library(ggplot2)
library(GGally)
library(corrplot)
library(plotmo)
library(caret)
library(kernlab)
library(earth)
library(skimr)
library(psych)
library(reshape2)
library(gt)
library(ROSE)

rseed <- 123

dir_prefix <- getwd()
print(dir_prefix)

## [1] "C:/Users/ToriT/ADS503-Group6"

### Connection info for GitHub File
url <- paste(dir_prefix, 'healthcare-dataset-stroke-data.csv', sep = '/')
df_orig <- read_csv(url)
print(url)

## [1] "C:/Users/ToriT/ADS503-Group6/healthcare-dataset-stroke-data.csv"

describe(df_orig)
```

	vars	n	mean	sd	median	trimmed	mad	min
## id	1	5110	36517.83	21161.72	36932.00	36542.26	27413.27	67.00
## gender*	2	5110	1.41	0.49	1.00	1.39	0.00	1.00
## age	3	5110	43.23	22.61	45.00	43.61	26.69	0.08
## hypertension	4	5110	0.10	0.30	0.00	0.00	0.00	0.00
## heart_disease	5	5110	0.05	0.23	0.00	0.00	0.00	0.00
## ever_married*	6	5110	1.66	0.48	2.00	1.70	0.00	1.00

```
## work_type*          7 5110      3.50      1.28      4.00      3.62      0.00      1.00
## Residence_type*     8 5110      1.51      0.50      2.00      1.51      0.00      1.00
## avg_glucose_level   9 5110    106.15    45.28    91.88    97.85    26.06    55.12
## bmi*               10 5110    172.19    88.96   158.00   163.08    74.13    1.00
## smoking_status*    11 5110      2.59      1.09      2.00      2.61      1.48    1.00
## stroke             12 5110      0.05      0.22      0.00      0.00      0.00    0.00
##
##           max      range  skew kurtosis      se
## id          72940.00 72873.00 -0.02   -1.21 296.03
## gender*         3.00      2.00  0.35   -1.86  0.01
## age           82.00   81.92 -0.14   -0.99  0.32
## hypertension     1.00      1.00  2.71    5.37  0.00
## heart_disease     1.00      1.00  3.94   13.57  0.00
## ever_married*     2.00      1.00 -0.66   -1.57  0.01
## work_type*        5.00      4.00 -0.91   -0.49  0.02
## Residence_type*    2.00      1.00 -0.03   -2.00  0.01
## avg_glucose_level 271.74   216.62  1.57    1.68  0.63
## bmi*            419.00   418.00  0.97    0.87  1.24
## smoking_status*    4.00      3.00  0.08   -1.35  0.02
## stroke           1.00      1.00  4.19   15.57  0.00
```

Exploratory Data Analysis (EDA)

###graphical and non-graphical representations of relationships between the response variable and predi

```
df_eda <- df_orig
```

```
rownames(df_eda) <- df_eda$id
```

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

```
df_eda <- dplyr::select(df_eda, -id)
```

```
print(df_eda)
```

```
## # A tibble: 5,110 x 11
```

```
##   gender  age hypertension heart_disease ever_married work_type Residence_type
##   <chr>  <dbl>         <dbl>         <dbl> <chr>         <chr>      <chr>
## 1 Male    67           0           1 Yes      Private      Urban
## 2 Female  61           0           0 Yes      Self-emp~    Rural
## 3 Male    80           0           1 Yes      Private      Rural
## 4 Female  49           0           0 Yes      Private      Urban
## 5 Female  79           1           0 Yes      Self-emp~    Rural
## 6 Male    81           0           0 Yes      Private      Urban
## 7 Male    74           1           1 Yes      Private      Rural
## 8 Female  69           0           0 No        Private      Urban
## 9 Female  59           0           0 Yes      Private      Rural
## 10 Female 78           0           0 Yes      Private      Urban
```

```
## # i 5,100 more rows
```

```
## # i 4 more variables: avg_glucose_level <dbl>, bmi <chr>, smoking_status <chr>,
```

```
## #   stroke <dbl>
```

Histograms

```

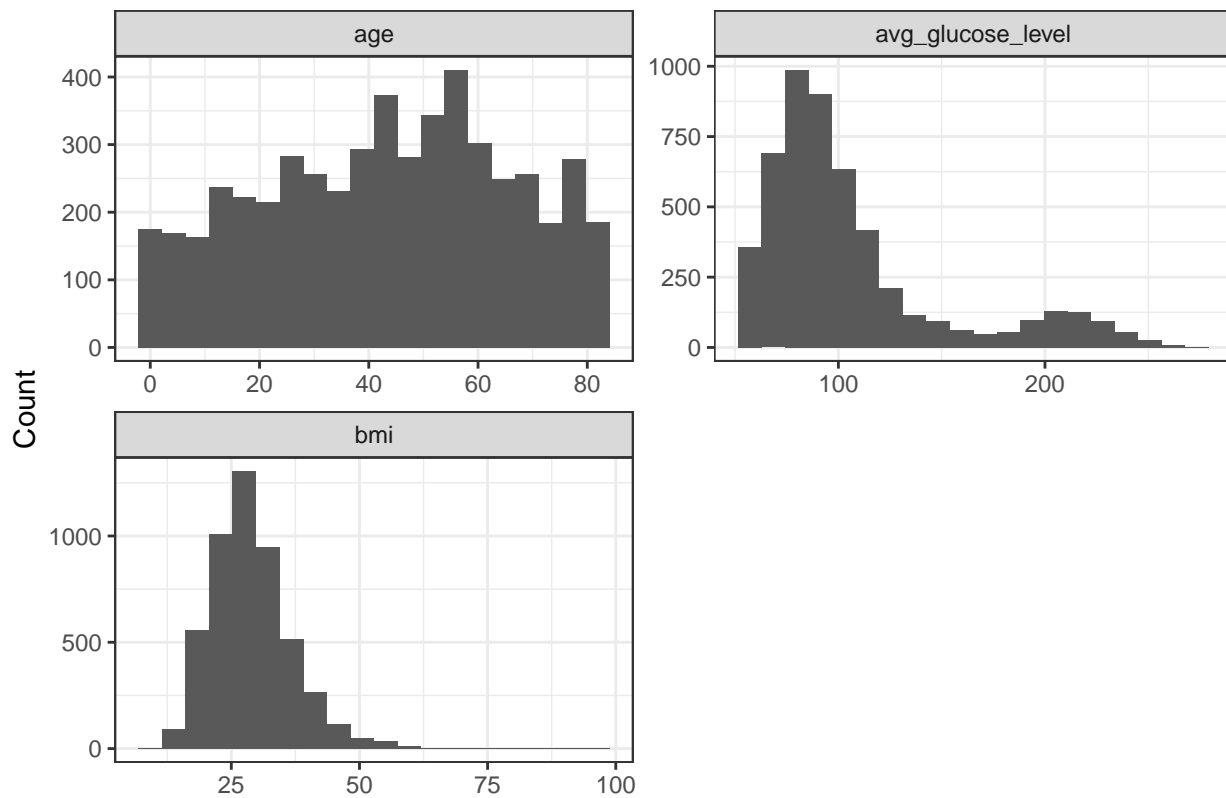
# Filter out N/A values for bmi and convert to numeric
df_eda <- df_eda %>%
  filter(!is.na(bmi)) %>%
  mutate(bmi = as.numeric(bmi))

# Pivot longer and convert value column to numeric if possible
df_long <- df_eda %>%
  pivot_longer(-c(stroke, ever_married, gender, hypertension, heart_disease, Residence_type, work_type),
  mutate(value = as.numeric(value))

# Plot histograms
ggplot(df_long, aes(x = value)) +
  geom_histogram(bins = 20) +
  facet_wrap(~Variable, scales = "free", ncol = 2) +
  theme_bw() +
  labs(title = 'Histograms of Variables Distributions', x = NULL, y = "Count")

```

Histograms of Variables Distributions



Corrplot

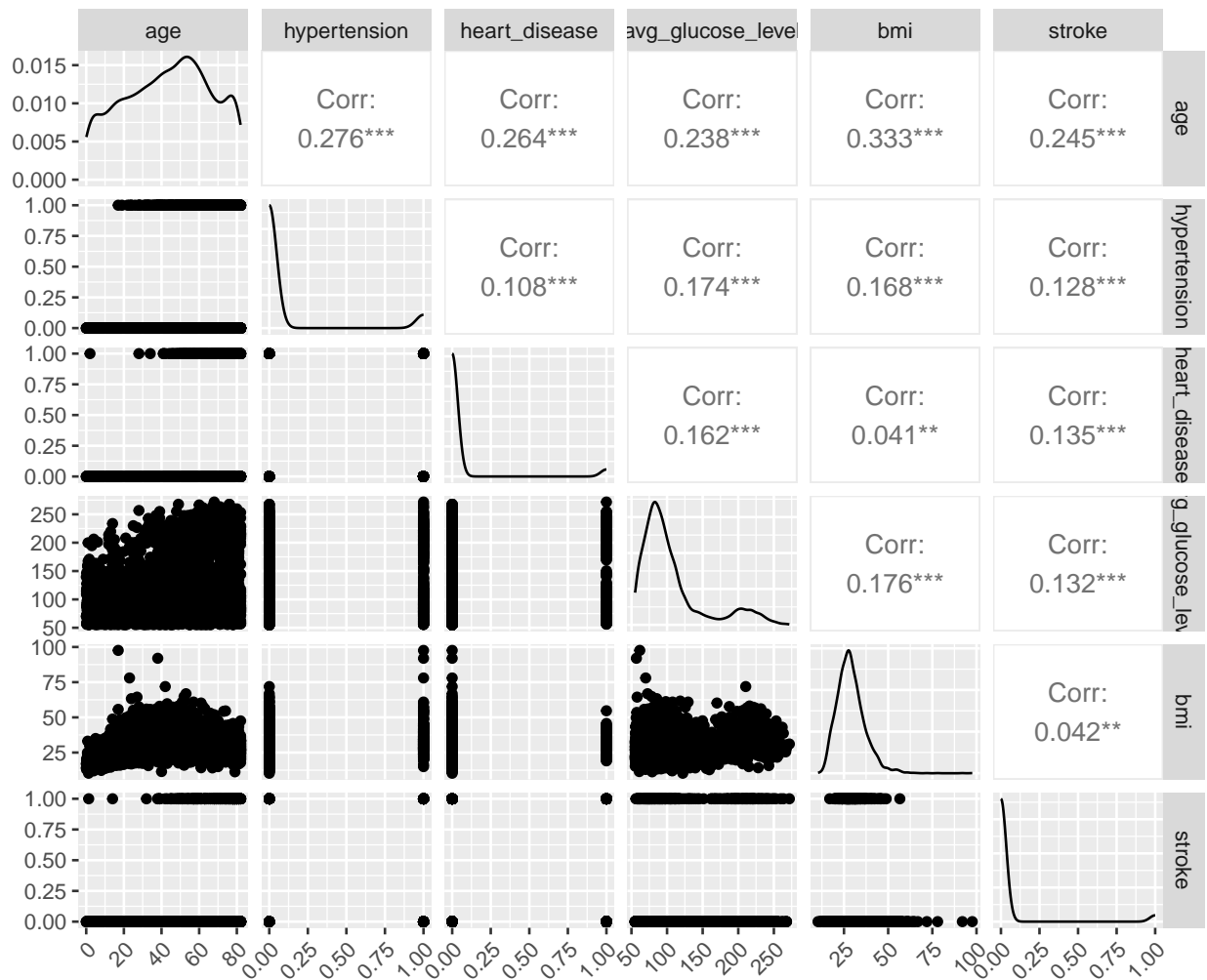
```

numeric_df <- df_eda[sapply(df_eda, is.numeric)]

numeric_df |>
  ggpairs(title = "Relationship Between Predictors", progress = FALSE)+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

```

Relationship Between Predictors



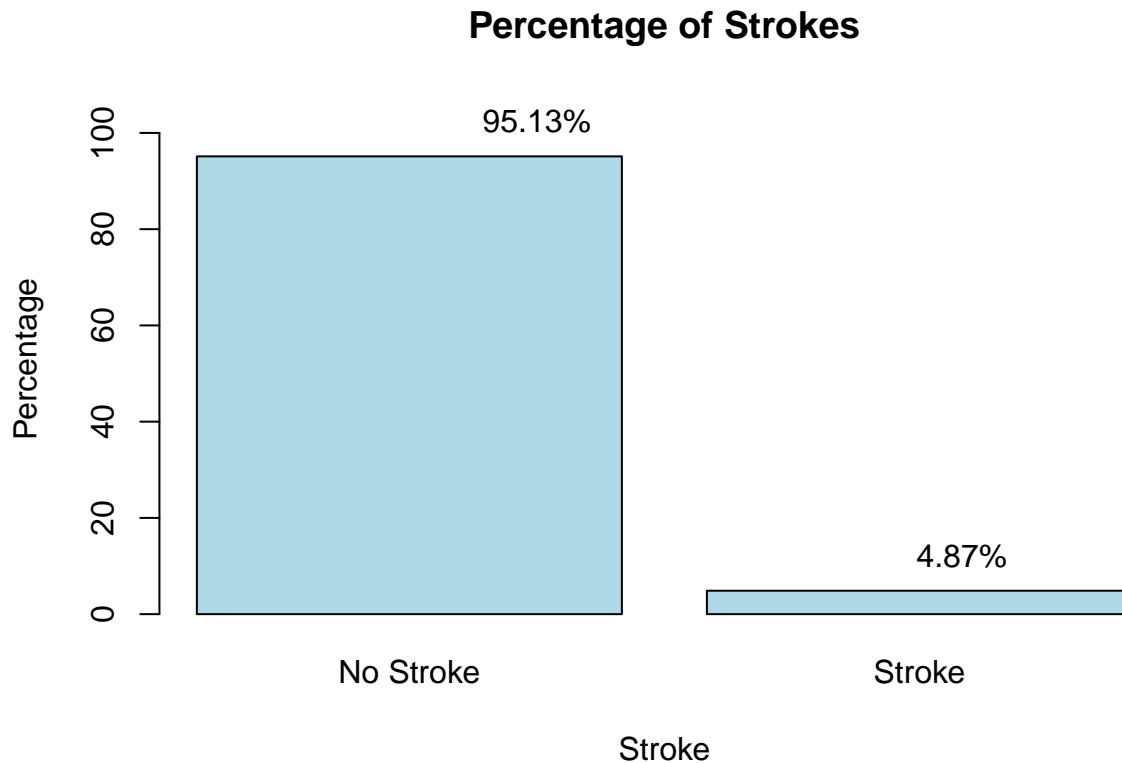
Class Balance

```
stroke_percent <- prop.table(table(df_eda$stroke)) * 100

barplot(stroke_percent,
        main = "Percentage of Strokes",
        xlab = "Stroke",
        ylab = "Percentage",
        col = "lightblue",
        ylim = c(0, max(stroke_percent) + 11),
        names.arg = c("No Stroke", "Stroke"))

label_pos <- stroke_percent + 1

### Add labels
text(x = 1:length(stroke_percent),
     y = label_pos,
     labels = paste0(round(stroke_percent, 2), "%"),
     pos = 3)
```



Distributions of Values (Counts)

```
wrap_text <- function(x, width) {
  sapply(strwrap(x, width = width, simplify = FALSE), paste, collapse = "\n")
}

# Select non-numeric columns
non_numeric_columns <- names(df_eda)[sapply(df_eda, is.factor) | sapply(df_eda, is.character)]

# Function to create plots
create_plots <- function(col) {
  counts <- table(df_eda[[col]], df_eda$stroke)
  counts_df <- as.data.frame(counts)
  names(counts_df) <- c(col, "stroke", "count")

  ggplot(counts_df, aes(x = !!sym(col), y = count, fill = factor(stroke))) +
    geom_bar(stat = "identity", position = "dodge") +
    geom_text(aes(label = paste0(count),
      vjust = -0.5, size = 3, position = position_dodge(width = 0.9))) + # Add count / percent
    labs(x = col, y = "Count of Records",
      title = paste("Bar Graph of Stroke by", col, " by Count")) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
      panel.spacing = unit(3, "lines"),
      legend.text = element_text(size = 7), # Adjust legend text size
      legend.title = element_text(size = 9)) # Adjust legend title size
}
```

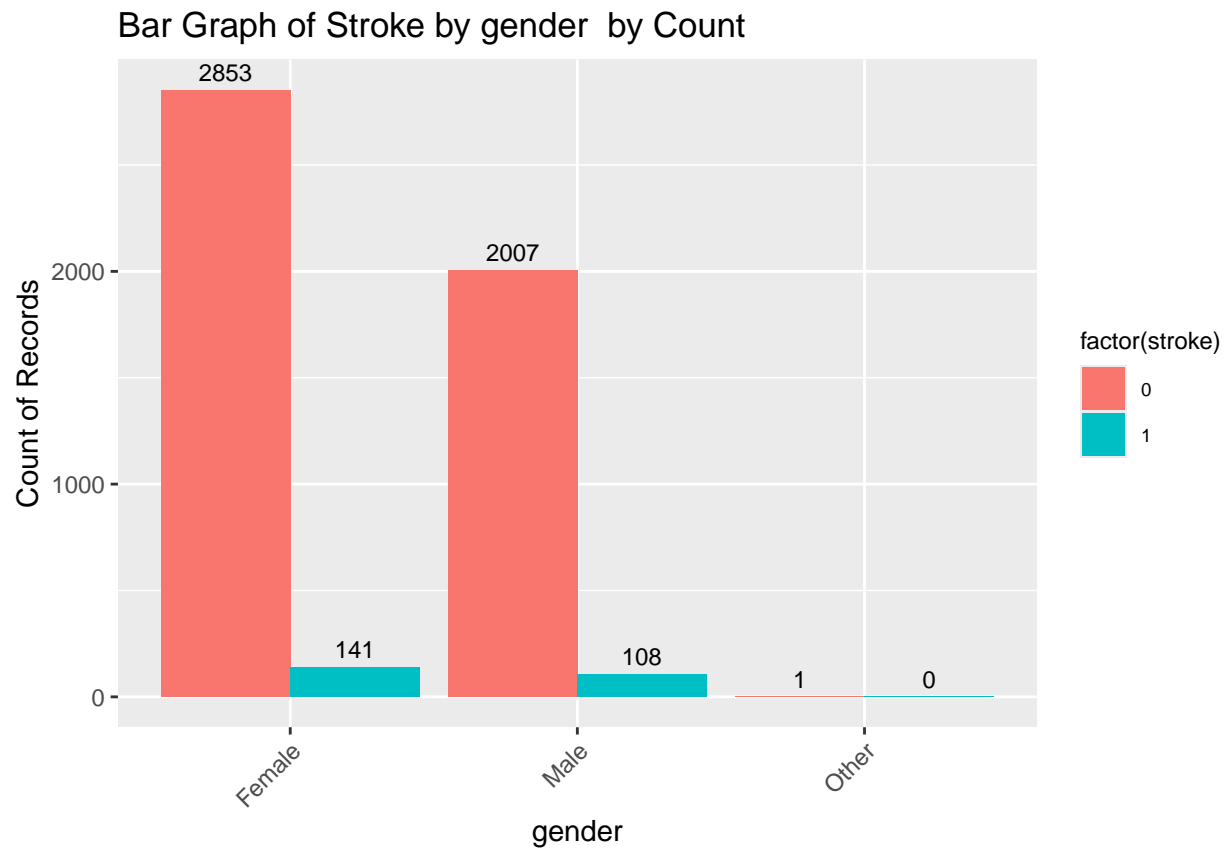
```

}

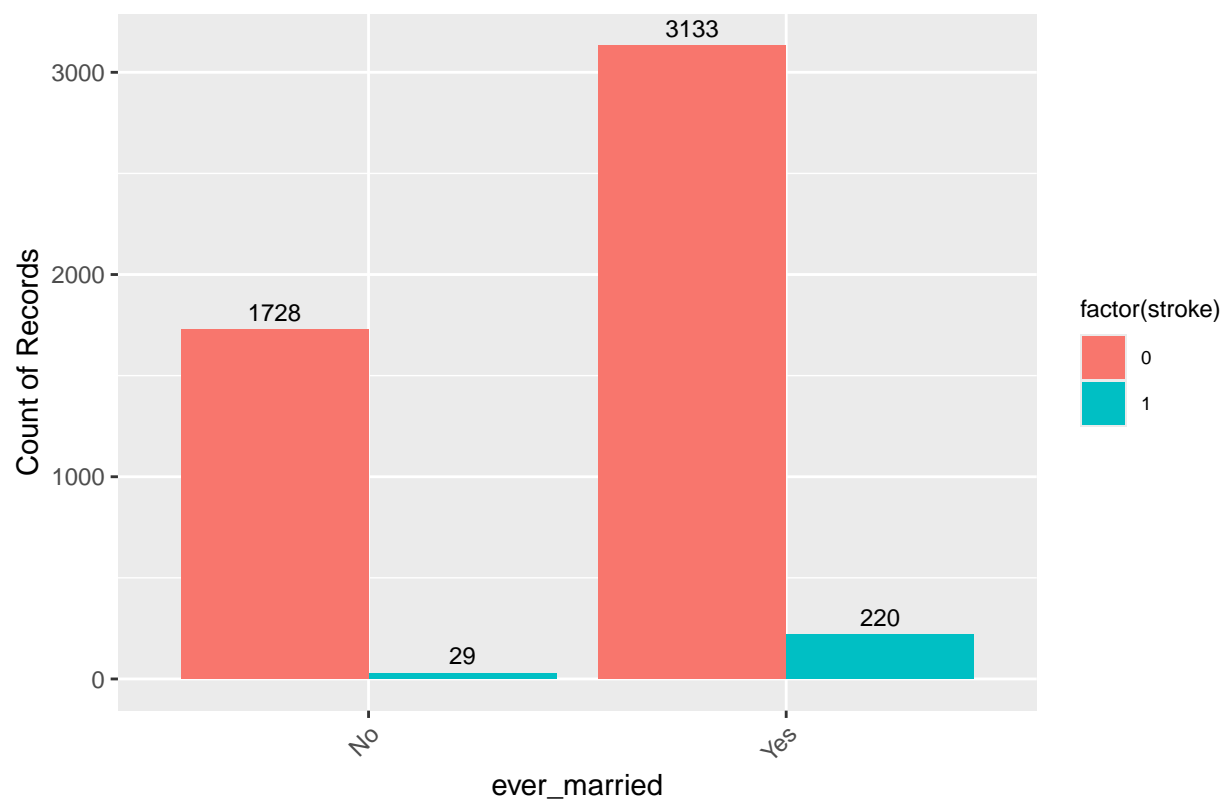
# Create plots for each non-numeric column
plots_list <- map(non_numeric_columns, ~ create_plots(.x))

# Print plots
walk(plots_list, print)

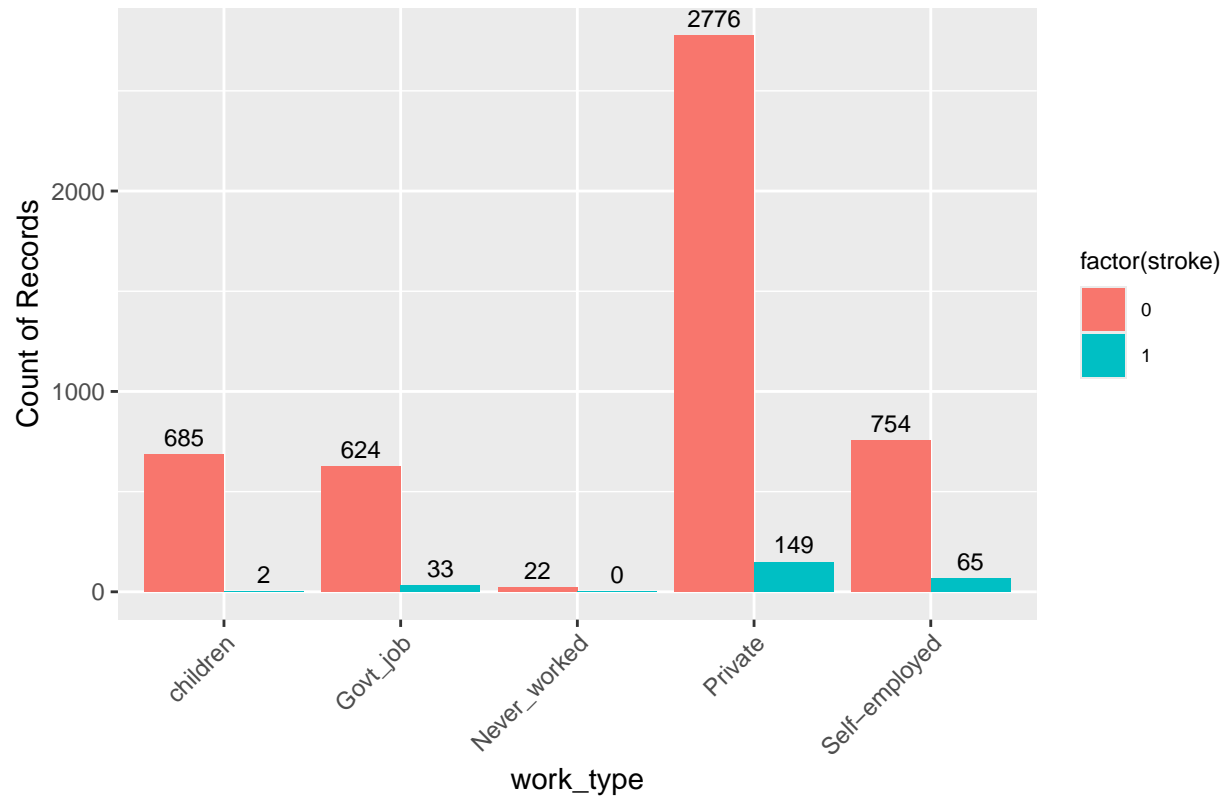
```

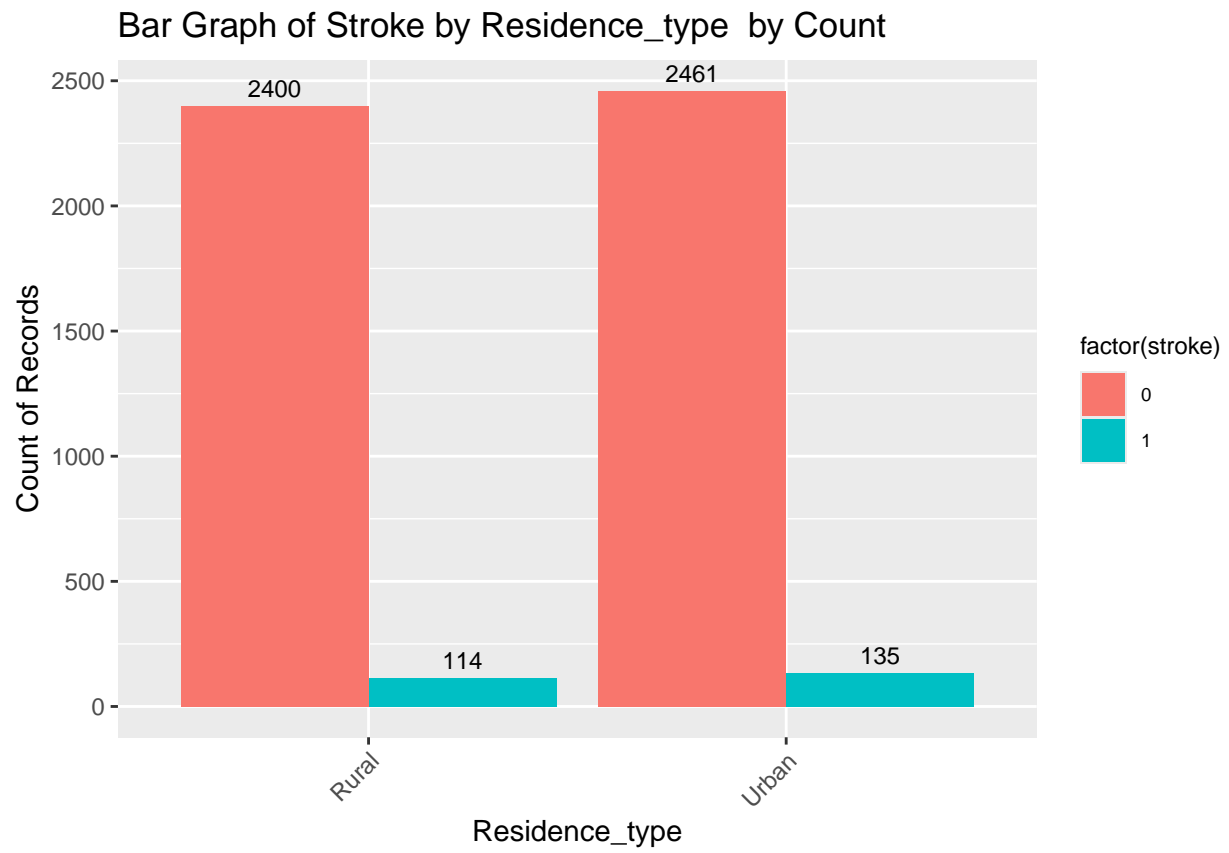


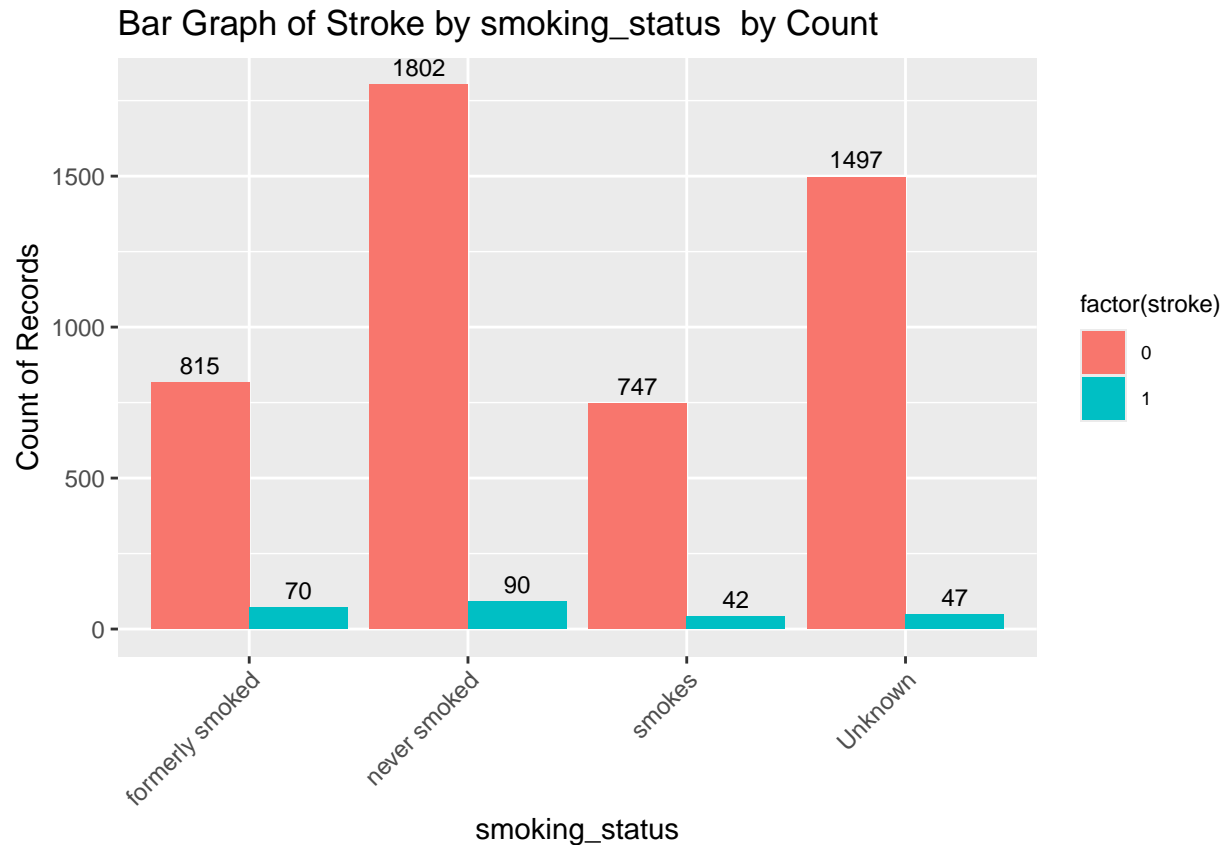
Bar Graph of Stroke by ever_married by Count



Bar Graph of Stroke by work_type by Count







Distributions of Values (Percentages)

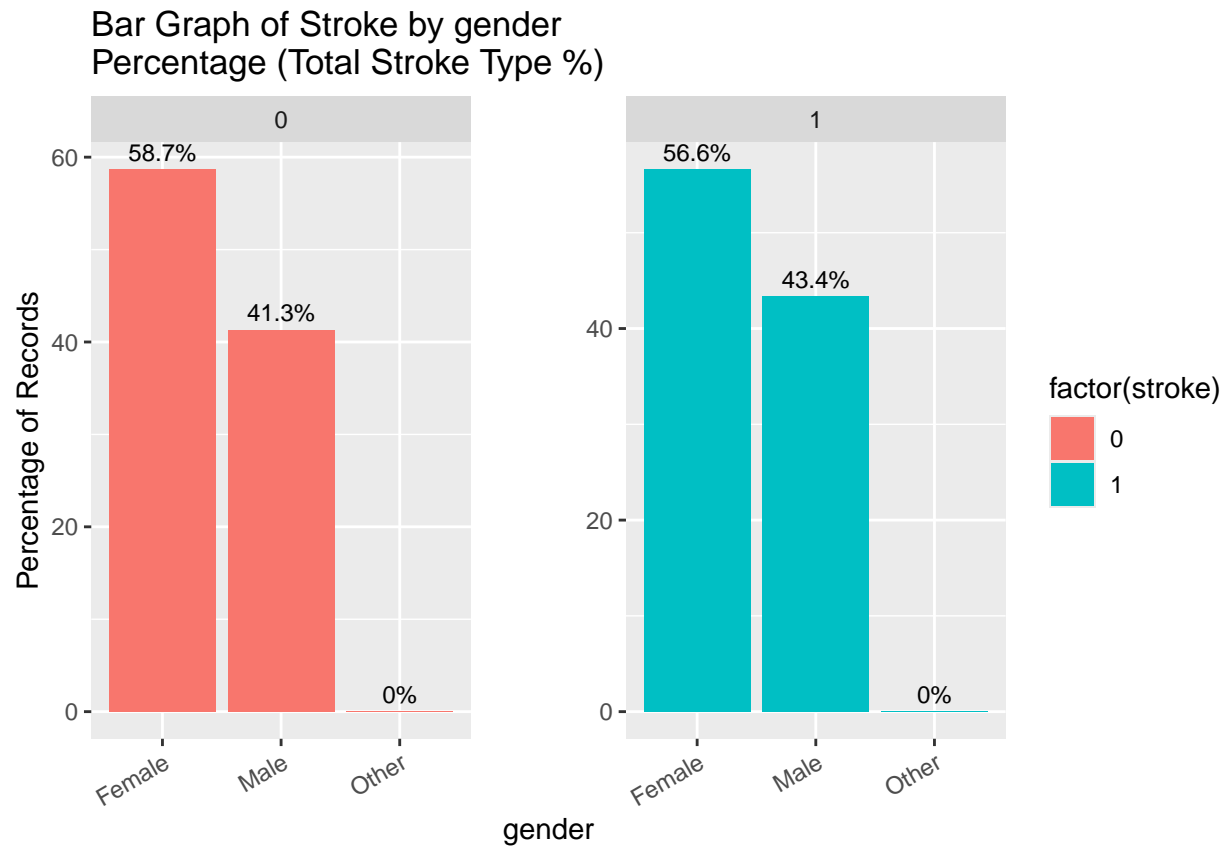
```
# Function to create plots
create_plots <- function(col) {
  counts <- table(df_eda[[col]], df_eda$stroke)
  counts_df <- as.data.frame(counts)
  names(counts_df) <- c(col, "stroke", "count")

  # Calculate percentages within each category
  counts_df <- counts_df %>%
    group_by(stroke) %>%
    mutate(percent = count / sum(count) * 100) %>%
    ungroup()

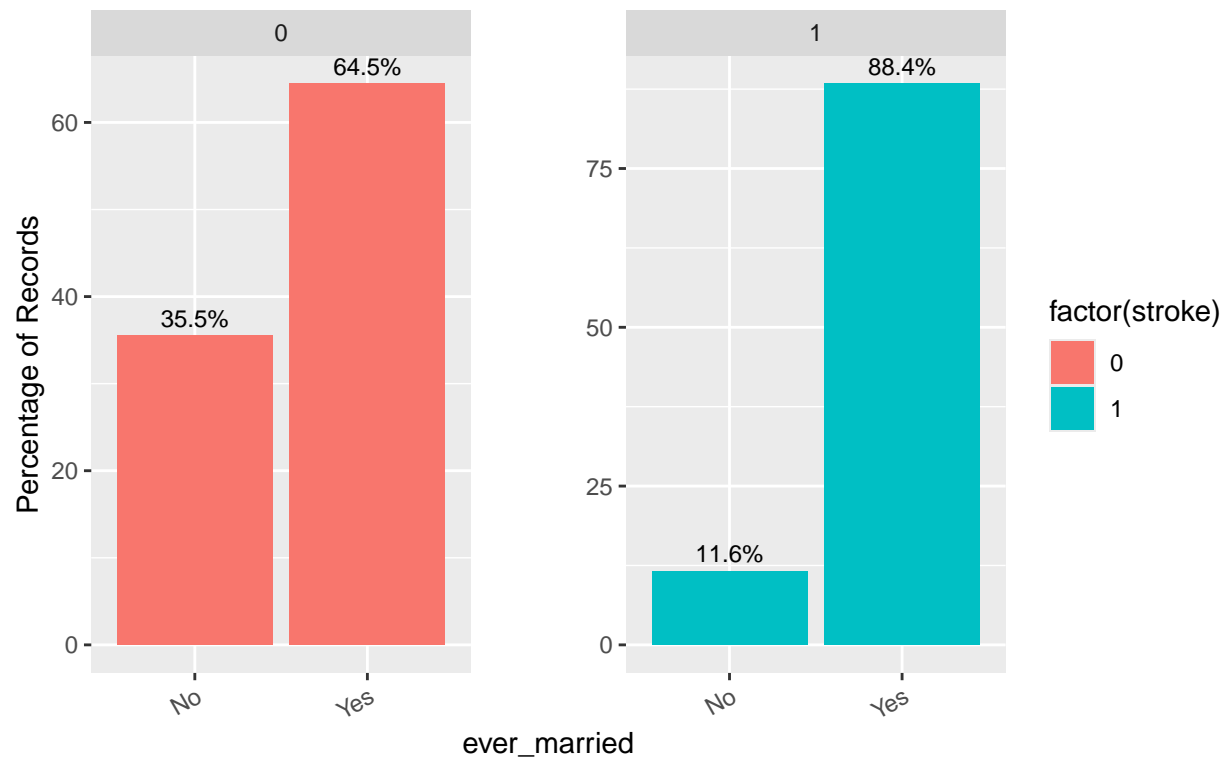
  ggplot(counts_df, aes(x = !!sym(col), y = percent, fill = factor(stroke))) +
    geom_bar(stat = "identity") +
    geom_text(aes(label = paste0(round(percent, 1), "%"),
      vjust = -0.5, size = 3) + # Add percent labels
    labs(x = col, y = "Percentage of Records",
      title = paste("Bar Graph of Stroke by", col, "\nPercentage (Total Stroke Type %)") +
    facet_wrap(~stroke, scales = "free") + # Facet by stroke
    theme(axis.text.x = element_text(angle = 30, hjust = 1),
      panel.spacing = unit(3, "lines"))
}
```

```
# Create plots for each non-numeric column
plots_list <- map(non_numeric_columns, ~ create_plots(.x))

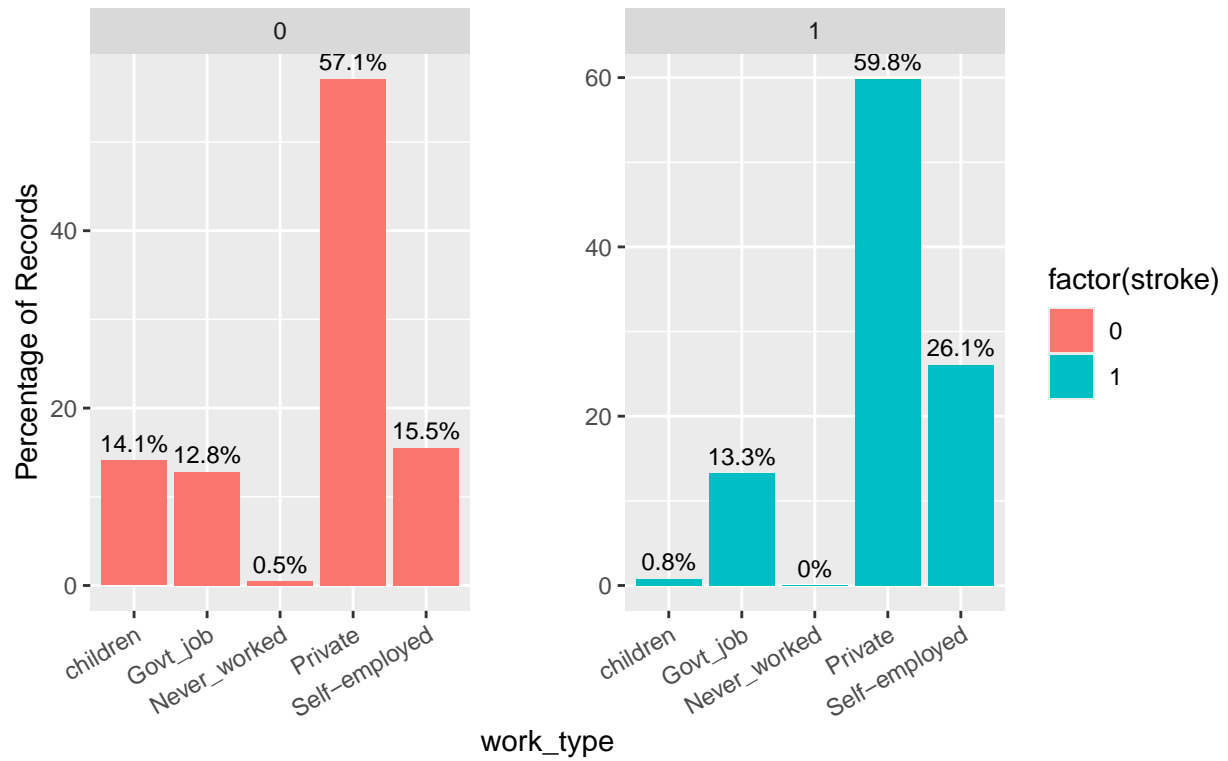
# Print plots
walk(plots_list, print)
```



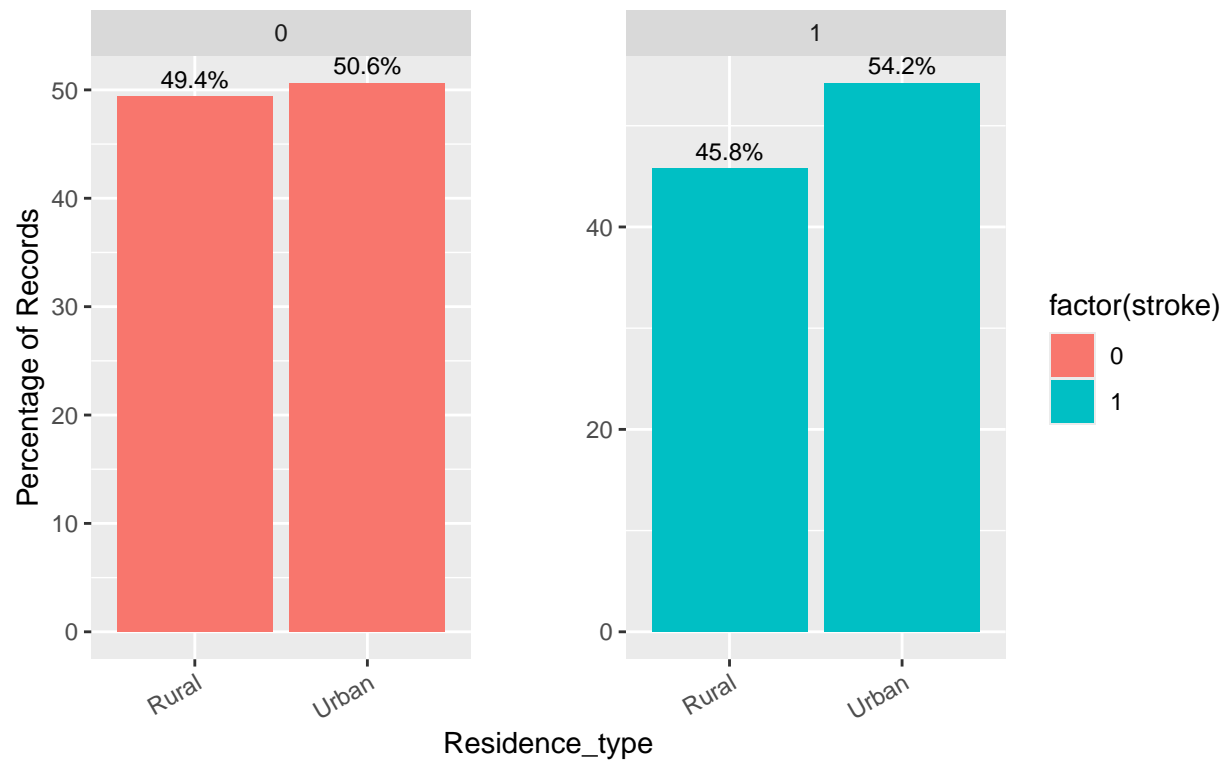
Bar Graph of Stroke by ever_married
Percentage (Total Stroke Type %)



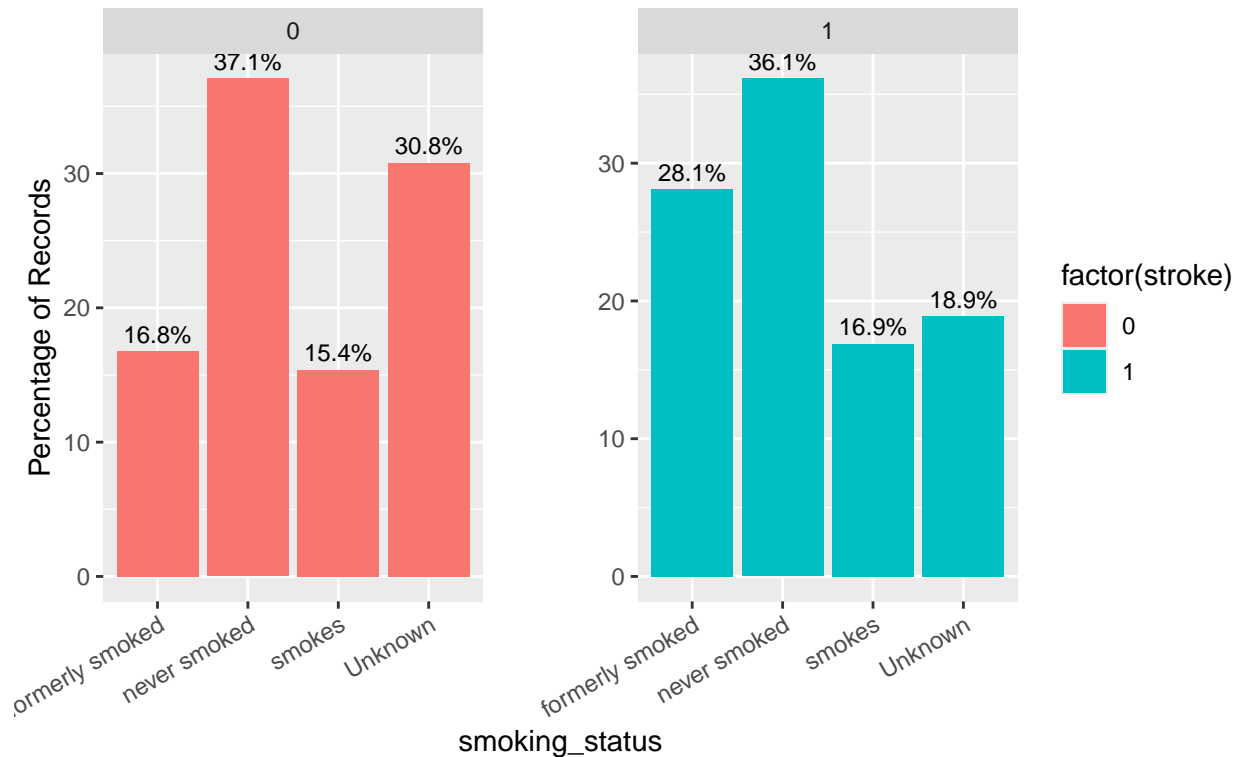
Bar Graph of Stroke by work_type
Percentage (Total Stroke Type %)



Bar Graph of Stroke by Residence_type
Percentage (Total Stroke Type %)



Bar Graph of Stroke by smoking_status
Percentage (Total Stroke Type %)

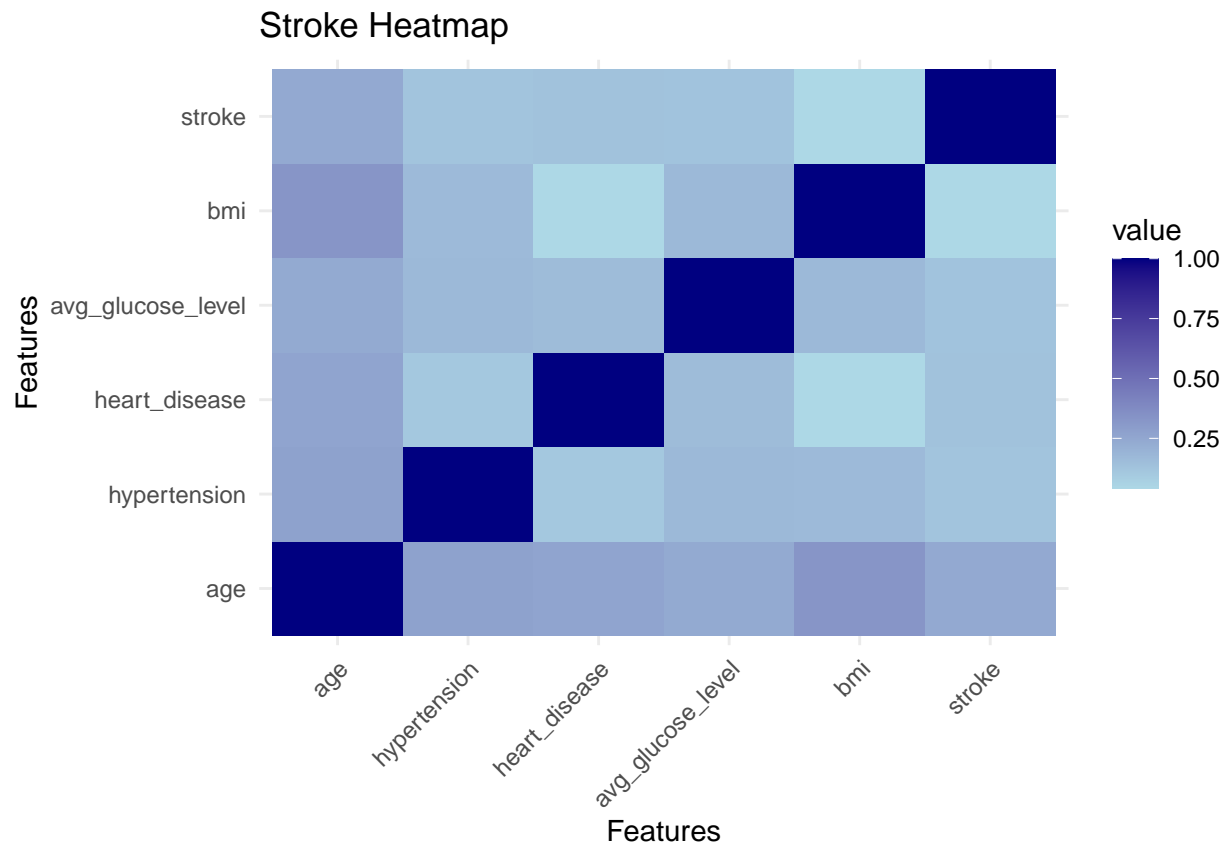


Heatmap

```
##### HEAT MAP ##### For Feature Selection #####
# Numeric Columns for EDA
heatmap_data <- df_eda %>%
  select(age, hypertension, heart_disease, avg_glucose_level, bmi, stroke)

# Correlation Matrix
correlation_matrix <- cor(heatmap_data, use = "pairwise.complete.obs")

# Heatmap Plot
ggplot(data = melt(correlation_matrix), aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient(low = "Light Blue", high = "Navy Blue") +
  labs(x = "Features", y = "Features", title = "Stroke Heatmap") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Data Splitting - Training & Test

```
###training, validation, and test sets
###Split
trainIndex <- createDataPartition(df_eda$stroke, p = 0.8, ### saving 20% for test
                                  list = FALSE,
                                  times = 1)

# Subset data into training and testing sets
trainData <- df_eda[trainIndex, ]
testData <- df_eda[-trainIndex, ]
```

Data Wrangling and Pre-Processing

Missing Values

```
table(trainData$gender)

##
## Female    Male
##   2395    1693

trainData$gender[trainData$gender == "Other"] <- {
  ux <- unique(trainData$gender)
  mode_gender <- ux[which.max(tabulate(match(trainData$gender, ux)))]
  mode_gender
```



```

}

testData$gender[testData$gender == "Other"] <- {
  ux <- unique(testData$gender)
  mode_gender <- ux[which.max(tabulate(match(testData$gender, ux)))]
  mode_gender
}
table(trainData$gender)

##
## Female    Male
##   2395    1693

# Check for missing data
colSums(is.na(trainData))

##           gender           age      hypertension      heart_disease
##             0             0             0             0
## ever_married      work_type  Residence_type avg_glucose_level
##             0             0             0             0
##           bmi      smoking_status           stroke
##           162             0             0

### bagImpute used because missing data is random
bag_missing <- preProcess(trainData, method = "bagImpute") ##### CHANGE? knn ??
trainData <- predict(bag_missing, newdata = trainData)

# bagImpute for Test Data
testData <- predict(bag_missing, newdata = testData)

colSums(is.na(trainData))

##           gender           age      hypertension      heart_disease
##             0             0             0             0
## ever_married      work_type  Residence_type avg_glucose_level
##             0             0             0             0
##           bmi      smoking_status           stroke
##             0             0             0

```

Setting up Factors

```

### Factor Train
non_numeric_cols <- sapply(trainData, function(x) !is.numeric(x))
# Convert non-numeric columns to factors
trainData <- trainData %>%
  mutate_if(non_numeric_cols, as.factor)

### Factor Test
non_numeric_cols <- sapply(testData, function(x) !is.numeric(x))
# Convert non-numeric columns to factors Test
testData <- testData %>%
  mutate_if(non_numeric_cols, as.factor)

#str(trainData)
#str(testData)

```

Dummy Variables

```
### dummy variables for Trfactors
dummy_model1 <- dummyVars(stroke ~ gender + ever_married + work_type + Residence_type + smoking_status,

trainData_dummy <- as.data.frame(predict(dummy_model1, newdata = trainData))
trainData_dummy <- as.data.frame(lapply(trainData_dummy, as.factor))

### dummy variables for test factors
dummy_model2 <- dummyVars(stroke ~ gender + ever_married + work_type + Residence_type + smoking_status,

testData_dummy <- as.data.frame(predict(dummy_model2, newdata = testData))
testData_dummy <- as.data.frame(lapply(testData_dummy, as.factor))

#head(trainData_dummy)
#head(testData_dummy)

### Dropping columns that can be inferred from the others to avoid multicollinearity
trainData_dummy <-select(trainData_dummy, -gender.Female, -ever_married.No, -work_type.children, -Residence_type.Children)

#head(trainData_dummy)
trainData_selected <- select(trainData, age, hypertension, heart_disease, avg_glucose_level, bmi, stroke)
#str(trainData_selected)

### Test Dropping columns that can be inferred from the others to avoid multicollinearity
testData_dummy <-select(testData_dummy, -gender.Female, -ever_married.No, -work_type.children, -Residence_type.Children)
testData_selected <- select(testData, age, hypertension, heart_disease, avg_glucose_level, bmi, stroke)
#str(testData_dummy)

# Train
trainData_ready <- cbind(trainData_dummy, trainData_selected)
trainData_ready <- trainData_ready %>%
  mutate(across(all_of(c("hypertension", "heart_disease", "stroke")), as.factor))
names(trainData_ready) <- gsub("\\\\.", "_", names(trainData_ready))
names(trainData_ready) <- gsub("_Yes", "", names(trainData_ready))
trainData_ready$stroke <- as.factor(make.names(as.character(trainData_ready$stroke)))
head(trainData_ready)

##   gender_Male ever_married work_type_Govt_job work_type_Never_worked
## 1           1           1                   0                      0
## 2           0           1                   0                      0
## 3           1           1                   0                      0
## 4           0           1                   0                      0
## 5           0           1                   0                      0
## 6           1           1                   0                      0
##   work_type_Private work_type_Self-employed Residence_type_Urban
## 1                 1                 0                 1
## 2                 0                 1                 0
## 3                 1                 0                 0
## 4                 1                 0                 1
## 5                 0                 1                 0
## 6                 1                 0                 1
##   smoking_status_formerly_smoked smoking_status_never_smoked
## 1                             1                             0
## 2                             0                             1
```

```
## 3          0          1
## 4          0          0
## 5          0          1
## 6          1          0
##   smoking_status_smokes age hypertension heart_disease avg_glucose_level
## 1          0 67          0          1          228.69
## 2          0 61          0          0          202.21
## 3          0 80          0          1          105.92
## 4          1 49          0          0          171.23
## 5          0 79          1          0          174.12
## 6          0 81          0          0          186.21
##           bmi stroke
## 1 36.60000    X1
## 2 34.58408    X1
## 3 32.50000    X1
## 4 34.40000    X1
## 5 24.00000    X1
## 6 29.00000    X1
```

```
# Test
testData_ready <- cbind(testData_dummy, testData_selected)
testData_ready <- testData_ready %>%
  mutate(across(all_of(c("hypertension", "heart_disease", "stroke")), as.factor))
names(testData_ready) <- gsub("\\.", "_", names(testData_ready))
names(testData_ready) <- gsub("_Yes", "", names(testData_ready))
testData_ready$stroke <- as.factor(make.names(as.character(testData_ready$stroke)))
str(testData_ready)
```

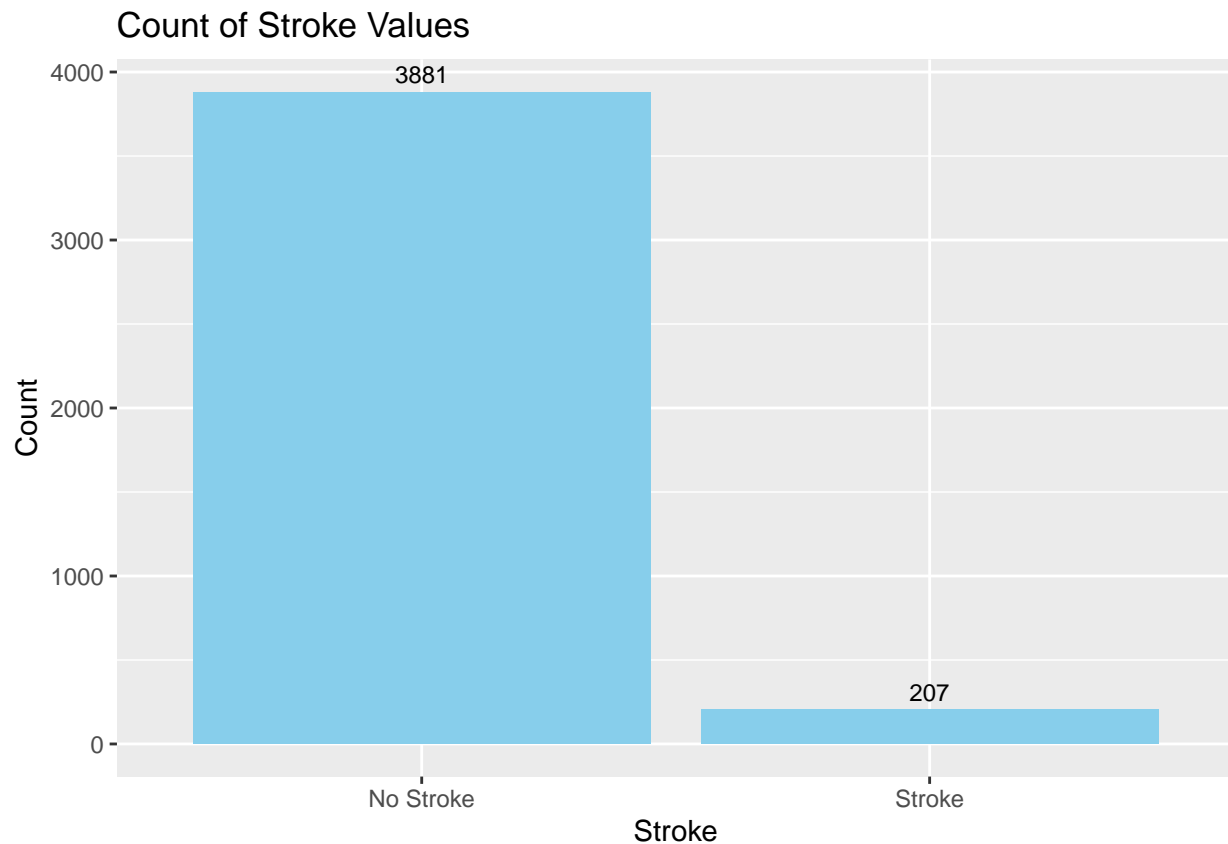
```
## 'data.frame': 1022 obs. of 16 variables:
## $ gender_Male : Factor w/ 2 levels "0","1": 2 2 1 2 2 2 2 2 1 1 ...
## $ ever_married : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 2 2 2 ...
## $ work_type_Govt_job : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 1 ...
## $ work_type_Never_worked : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
## $ work_type_Private : Factor w/ 2 levels "0","1": 1 2 2 1 2 2 2 2 2 1 ...
## $ work_type_Self-employed : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 1 2 ...
## $ Residence_type_Urban : Factor w/ 2 levels "0","1": 2 1 1 2 1 2 1 2 1 2 ...
## $ smoking_status_formerly_smoked: Factor w/ 2 levels "0","1": 1 1 2 1 2 1 2 1 1 1 ...
## $ smoking_status_never_smoked : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
## $ smoking_status_smokes : Factor w/ 2 levels "0","1": 1 1 1 2 1 2 1 2 2 1 ...
## $ age : num 57 82 65 69 59 57 74 82 39 77 ...
## $ hypertension : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 2 2 ...
## $ heart_disease : Factor w/ 2 levels "0","1": 2 2 1 2 1 1 1 2 1 1 ...
## $ avg_glucose_level : num 217 208 101 195 212 ...
## $ bmi : num 35 32.5 28.2 28.3 35 ...
## $ stroke : Factor w/ 2 levels "X0","X1": 2 2 2 2 2 2 2 2 2 2 ...
```

Class Imbalance (ROSE Method)

```
# Calculate counts and convert to data frame
stroke_counts <- as.data.frame(table(trainData_ready$stroke))
colnames(stroke_counts) <- c("Stroke", "Count")
stroke_counts$Stroke <- factor(stroke_counts$Stroke, levels = c('X0', 'X1'), labels = c("No Stroke", "S

### Bar Chart
ggplot(stroke_counts, aes(x = Stroke, y = Count)) +
```

```
geom_bar(stat = "identity", fill = "skyblue") +
labs(title = "Count of Stroke Values",
      x = "Stroke",
      y = "Count") +
geom_text(aes(label = Count),
          vjust = -0.5, # Position above the bars
          size = 3, # Size of the text
          color = "black") # Color of the text
```



```
set.seed(rseed)

# Apply ROSE to balance the dataset
rose_train <- ROSE(stroke ~ ., data = trainData_ready)$data

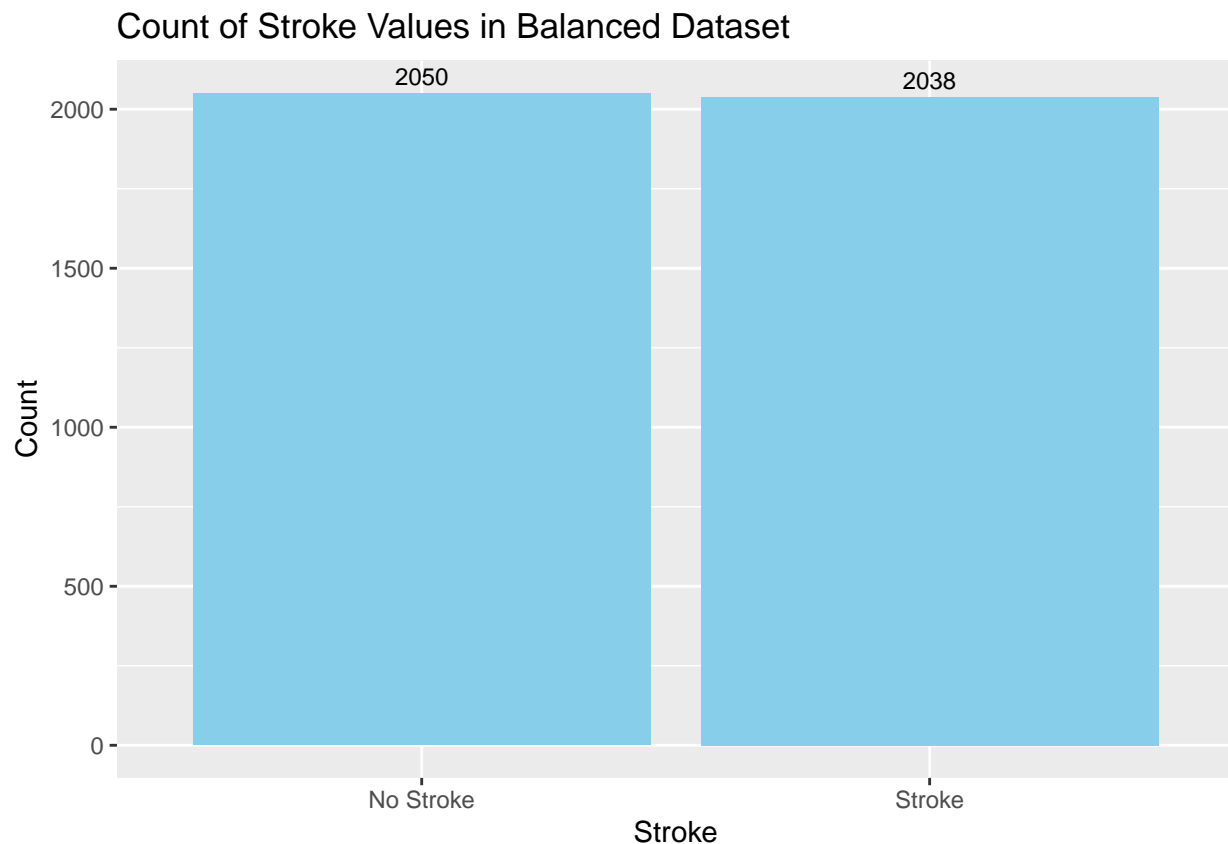
# Separate predictors (features) and target variable (stroke)
train_X <- rose_train[, !(names(rose_train) %in% "stroke")]
train_y <- rose_train$stroke

train_y_df <- data.frame(stroke = train_y)

stroke_counts <- as.data.frame(table(train_y_df$stroke))
colnames(stroke_counts) <- c("Stroke", "Count")
stroke_counts$Stroke <- factor(stroke_counts$Stroke,
                              levels = c('X0', 'X1'), labels = c("No Stroke", "Stroke"))

### Bar chart
ggplot(stroke_counts, aes(x = Stroke, y = Count)) +
```

```
geom_bar(stat = "identity", fill = "skyblue") +
labs(title = "Count of Stroke Values in Balanced Dataset",
      x = "Stroke",
      y = "Count") +
geom_text(aes(label = Count),
          vjust = -0.5,
          size = 3,
          color = "black")
```



```
trainData_ready <- data.frame(rose_train)
```

Baseline Model to Beat

```
set.seed(123)
### Parameters for Tuning
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 3,
                     verboseIter = FALSE,
                     classProbs = TRUE,
                     summaryFunction = twoClassSummary)

### Base Model Logistic Regression Model

# Fit logistic regression model
lrm_base_model <- train(stroke ~ .,
```

```

data = trainData_ready,
method = "glm",
family = "binomial",
trControl = ctrl)

## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.

# View model summary
summary(lrm_base_model)

##
## Call:
## NULL
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -3.988e+00  2.959e-01 -13.477  < 2e-16 ***
## gender_Male1    -3.107e-01  8.271e-02  -3.756  0.000172 ***
## ever_married1    2.337e-01  1.206e-01   1.937  0.052764 .
## work_type_Govt_job1 -8.208e-01  3.333e-01  -2.462  0.013799 *
## work_type_Never_worked1 -1.189e+01  2.780e+02  -0.043  0.965879
## work_type_Private1 -6.458e-01  3.209e-01  -2.012  0.044208 *
## work_type_Self-employed1 -8.378e-01  3.395e-01  -2.468  0.013591 *
## Residence_type_Urban1  3.330e-01  7.935e-02   4.196  2.72e-05 ***
## smoking_status_formerly_smoked1  4.729e-02  1.181e-01   0.400  0.688843
## smoking_status_never_smoked1  -2.223e-01  1.104e-01  -2.013  0.044127 *
## smoking_status_smokes1  1.164e-02  1.304e-01   0.089  0.928875
## age             6.739e-02  2.891e-03  23.306  < 2e-16 ***
## hypertension1    6.785e-01  1.102e-01   6.159  7.33e-10 ***
## heart_disease1    4.971e-01  1.344e-01   3.698  0.000218 ***
## avg_glucose_level  3.746e-03  7.264e-04   5.157  2.51e-07 ***
## bmi             3.214e-03  5.895e-03   0.545  0.585639
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##    Null deviance: 5667.1  on 4087  degrees of freedom
## Residual deviance: 3948.9  on 4072  degrees of freedom
## AIC: 3980.9
##
## Number of Fisher Scoring iterations: 13

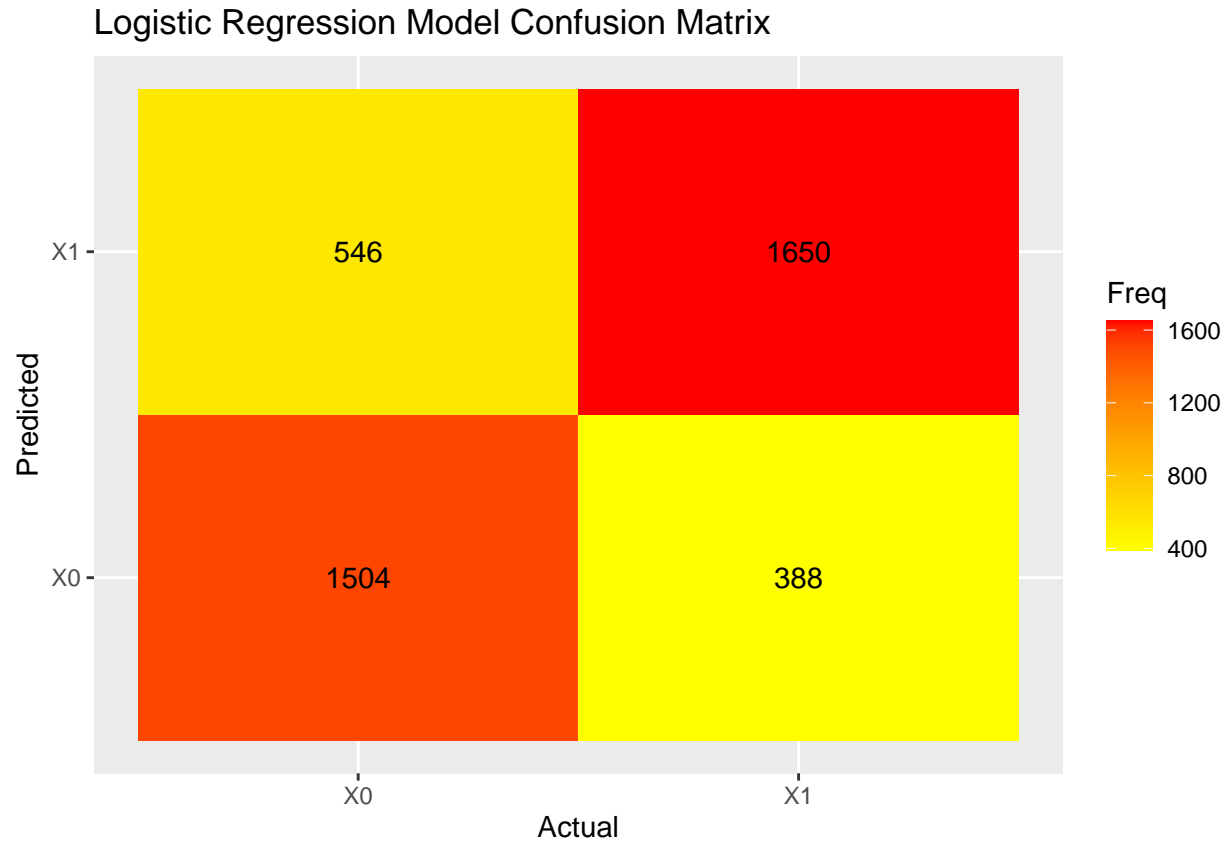
# LogReg Predictions
predictions <- predict(lrm_base_model, newdata = trainData_ready)

# LogReg Confusion Matrix
logregCM <- confusionMatrix(predictions, trainData_ready$stroke)

# Confusion Matrix Plot
conf_matrix <- as.data.frame(logregCM$table)
ggplot(data = conf_matrix, aes(x = Reference, y = Prediction)) +
  geom_tile(aes(fill = Freq)) +
  geom_text(aes(label = Freq)) +

```

```
scale_fill_gradient(low = 'yellow', high = 'red') +
labs(title = 'Logistic Regression Model Confusion Matrix', x = 'Actual', y = 'Predicted')
```



Data Prep

```
# Exclude stroke and Identify and remove variables with zero variance
Zero_Var_vars <- nearZeroVar(trainData_ready[, -which(names(trainData_ready) == "stroke"), drop = FALSE]
trainData_ready <- trainData_ready[, -Zero_Var_vars, drop = FALSE]
print(names(Zero_Var_vars))
```

```
## NULL
```

Models

Model #1 - Linear Discriminant Analysis (LDR)

```
rseed <- 123
set.seed(rseed)
# Linear Discriminant Analysis
ldaFit_stroke <- train(stroke ~ .,
  data = trainData_ready,
  method = 'lda',
  preProc = c("center", "scale"),
  metric = 'ROC',
  trControl = ctrl)

ldaFit_stroke
```

```

## Linear Discriminant Analysis
##
## 4088 samples
## 14 predictor
## 2 classes: 'X0', 'X1'
##
## Pre-processing: centered (14), scaled (14)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 3680, 3679, 3679, 3679, 3679, 3679, ...
## Resampling results:
##
## ROC          Sens          Spec
## 0.8409107 0.7265041 0.8151768

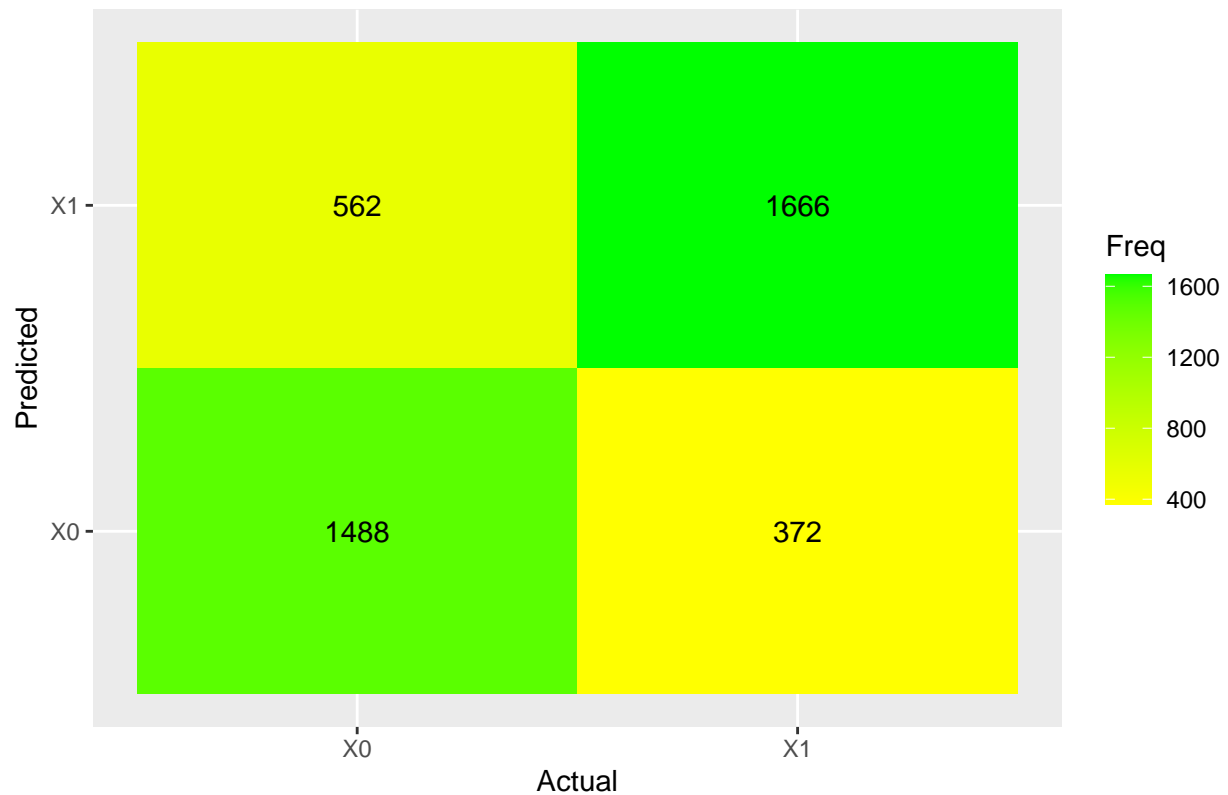
# LDA Predictions
predictions <- predict(ldaFit_stroke, newdata = trainData_ready)

# LDA Confusion Matrix
ldaCM_stroke <- confusionMatrix(predictions, trainData_ready$stroke)

# Confusion Matrix Plot
conf_matrix <- as.data.frame(ldaCM_stroke$table)
ggplot(data = conf_matrix, aes(x = Reference, y = Prediction)) +
  geom_tile(aes(fill = Freq)) +
  geom_text(aes(label = Freq)) +
  scale_fill_gradient(low = 'yellow', high = 'green') +
  labs(title = 'Linear Discriminant Analysis Model Confusion Matrix', x = 'Actual', y = 'Predicted')

```


Linear Discriminant Analysis Model Confusion Matrix



Model #2 - Penalized Logistic Regression (PLR)

```
set.seed(rseed)

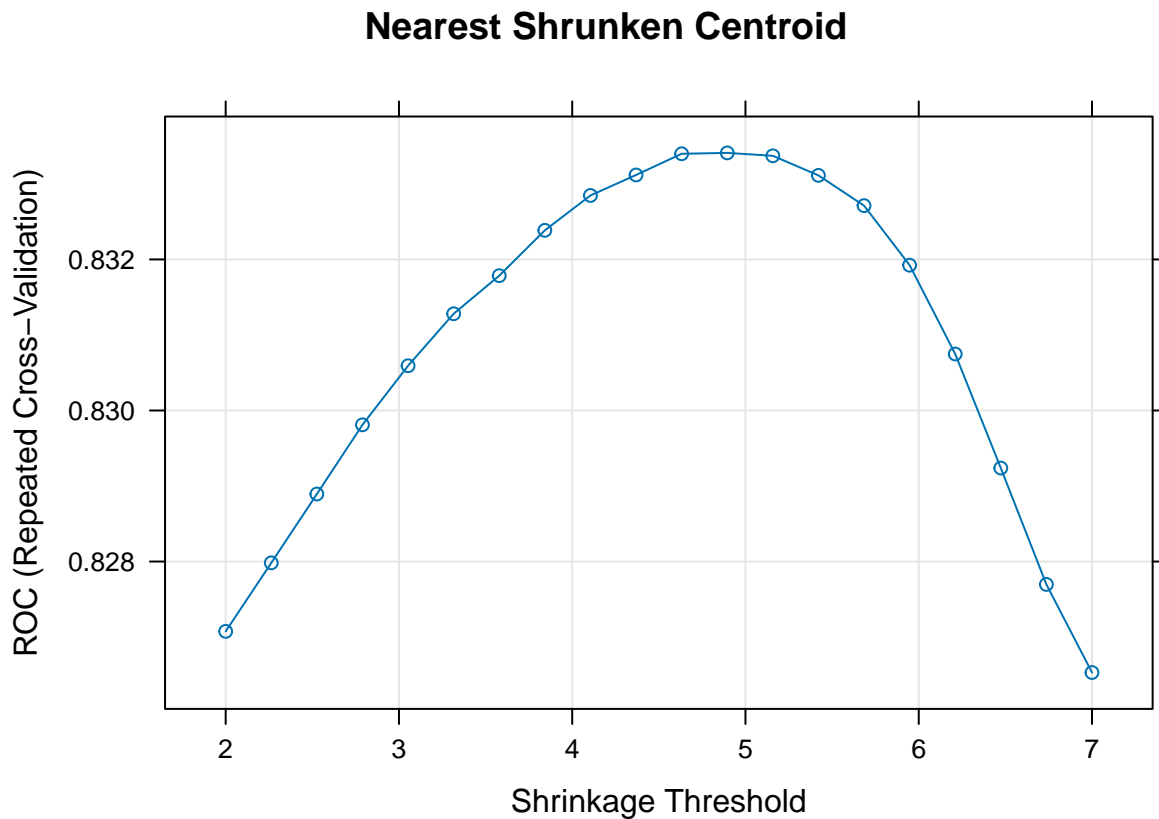
glmGrid <- expand.grid(alpha = c( 0, .5, 1),
  lambda = seq(.001, 2, length = 20))

# Penalized Logistic Regression
plrFit_stroke <- train(stroke ~ .,
  data = trainData_ready,
  method = 'glmnet',
  tuneGrid = glmGrid,
  preProc = c('center', 'scale'),
  metric = 'ROC',
  trControl = ctrl)

plot(plrFit_stroke, main = 'Penalized Logistic Regression')
```



```
plot(nscFit_stroke, main = 'Nearest Shrunk Centroid')
```



```
# Best tuning parameters
optimal_nsc_tune <- nscFit_stroke$bestTune
print(paste('Best threshold tuning parameter for Nearest Shrunk Centroids:', paste(optimal_nsc_tune, 0)))

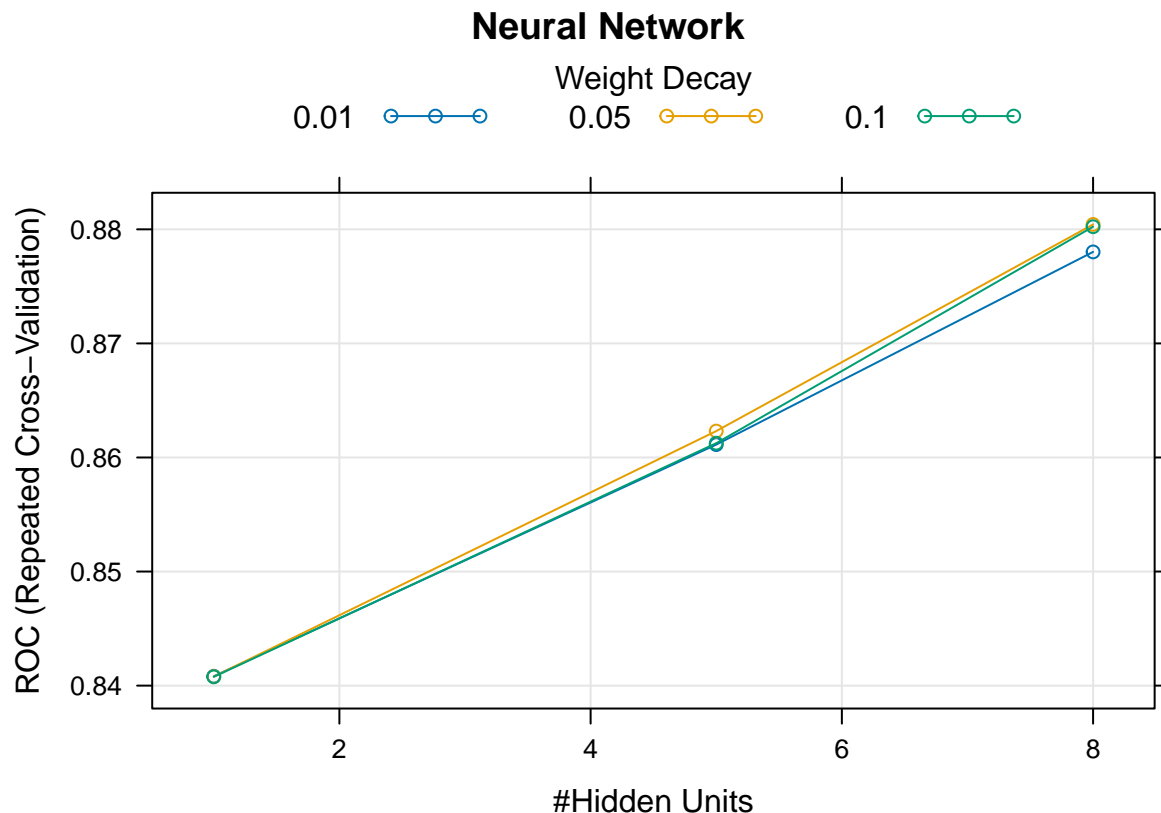
## [1] "Best threshold tuning parameter for Nearest Shrunk Centroids: 4.89473684210526"
```

Model #4 - Neural Network

```
set.seed(rseed)

grid <- expand.grid(size = c(1, 5, 8),      # Number of hidden units
                   decay = c( 0.01, 0.05, .1)) # Weight decay

# Neural Networks
nnFit_stroke <- train(stroke ~ .,
                     data = trainData_ready,
                     method = 'nnet',
                     maxit = 500,
                     preProcess = c('center', 'scale'),
                     metric = 'ROC',
                     trControl = ctrl,
                     tuneGrid = grid,
                     trace = FALSE)
plot(nnFit_stroke, main = 'Neural Network')
```



```
# Best tuning parameters
optimal_nn_tune <- nnFit_stroke$bestTune
print(paste('Best Size and Decay tuning parameters for Neural Network:', paste(optimal_nn_tune, collapse=', ')))

## [1] "Best Size and Decay tuning parameters for Neural Network: 8,0.05"
```

Model #5 - Random Forest

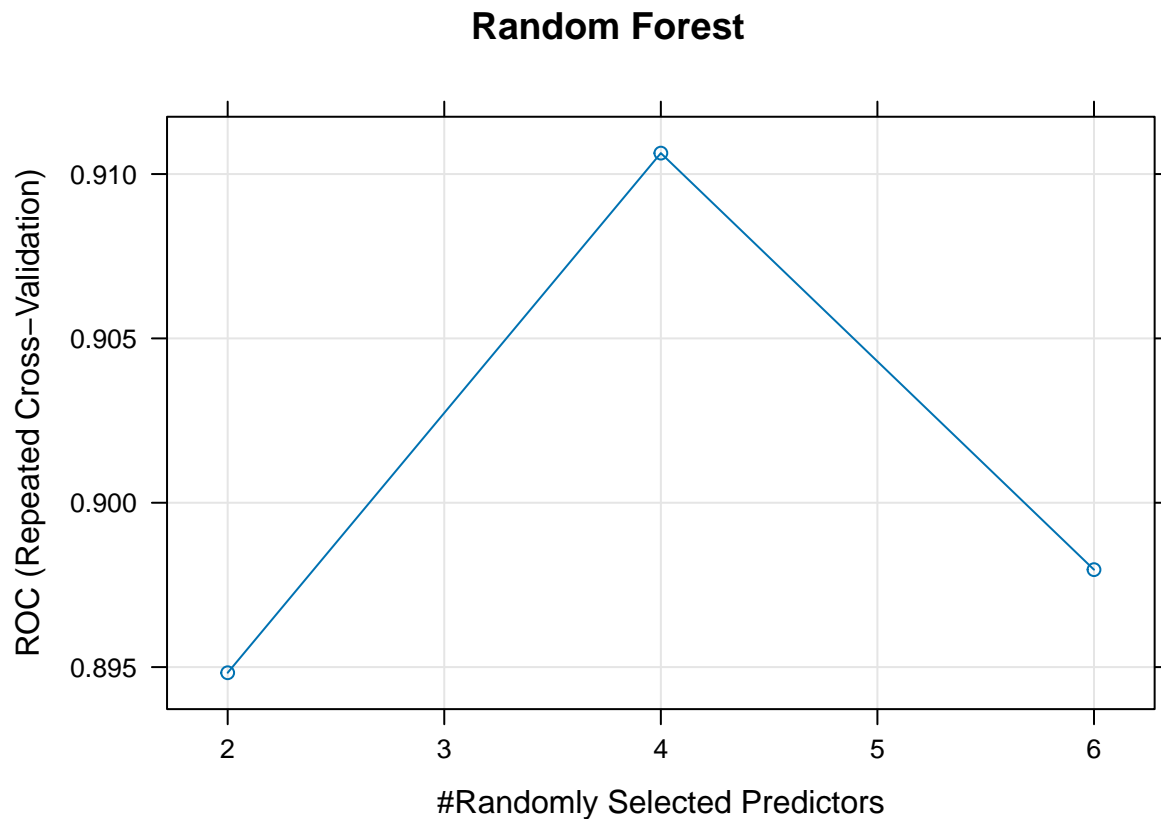
```
# Set seed for reproducibility
set.seed(rseed)

# Random Forest
rf_grid <- expand.grid(
  mtry = c(2, 4, 6)
)

rfFit_stroke <- train(
  stroke ~ .,
  data = trainData_ready,
  method = "rf",
  trControl = ctrl,
  tuneGrid = rf_grid,
  ntree = 100,
  trace = FALSE
)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
```

```
## in the result set. ROC will be used instead.
# Print the best parameters and performance metrics
plot(rfFit_stroke, main = 'Random Forest')
```



```
# Best tuning parameters
optimal_rf_tune <- rfFit_stroke$bestTune
print(paste('Best Size and Decay tuning parameters for Random Forest:', paste(optimal_rf_tune, collapse = ' ')))

## [1] "Best Size and Decay tuning parameters for Random Forest: 4"
```

Results - Summary Table

```
# 1.Make predictions for Linear Discriminant Analysis
lda_pred <- predict(ldaFit_stroke, newdata = testData_ready)
lda_cm <- confusionMatrix(lda_pred, testData_ready$stroke)

# 2.Make predictions for Penalized Logistic Regression
plr_pred <- predict(plrFit_stroke, newdata = testData_ready)
plr_cm <- confusionMatrix(plr_pred, testData_ready$stroke)

# 3.Make predictions for Nearest Shrunken Centroids
nsc_pred <- predict(nscFit_stroke, newdata = testData_ready)
nsc_cm <- confusionMatrix(nsc_pred, testData_ready$stroke)

# 4.Make predictions for Neural Network
nn_pred <- predict(nnFit_stroke, newdata = testData_ready)
```

```

nn_cm <- confusionMatrix(nn_pred, testData_ready$stroke)

# 5. Make predictions for Random Forest
rf_pred <- predict(rfFit_stroke, newdata = testData_ready)
rf_cm <- confusionMatrix(rf_pred, testData_ready$stroke)

extract_accuracy <- function(model, cm) {
  tibble(
    model = model,
    Accuracy = cm$overall['Accuracy'],
    `CI Lower` = cm$overall['AccuracyLower'],
    `CI Upper` = cm$overall['AccuracyUpper']
  )
}

# Combine accuracies into a table
accuracies <- bind_rows(
  extract_accuracy('Linear Discriminant Analysis', lda_cm),
  extract_accuracy('Penalized Logistic Regression', plr_cm),
  extract_accuracy('Nearest Shrunken Centroids', nsc_cm),
  extract_accuracy('Neural Network', nn_cm),
  extract_accuracy('Random Forest', rf_cm)
)

# Display the table using gt package
accuracies %>%
  arrange(-Accuracy) %>%
  gt() %>%
  fmt_number(columns = c(Accuracy, `CI Lower`, `CI Upper`), decimals = 3)

```

model	Accuracy	CI Lower	CI Upper
Random Forest	0.748	0.720	0.774
Neural Network	0.735	0.707	0.762
Penalized Logistic Regression	0.717	0.689	0.745
Linear Discriminant Analysis	0.711	0.682	0.739
Nearest Shrunken Centroids	0.663	0.634	0.692

Tuned Models

```

set.seed(rseed)
# Penalized Logistic Regression with optimal parameters
plrFit_final <- train(stroke ~ .,
  data = trainData_ready,
  method = 'glmnet',
  preProc = c('center', 'scale'),
  metric = 'ROC',
  trControl = ctrl,
  tuneGrid = expand.grid(
    alpha = optimal_plr_tune$alpha,
    lambda = optimal_plr_tune$lambda
  ))
set.seed(rseed)

```



```

nsc_cm2 <- confusionMatrix(nsc_pred2, testData_ready$stroke)

# Make predictions for Neural Network
nn_pred2 <- predict(nnFit_final, newdata = testData_ready)
nn_cm2 <- confusionMatrix(nn_pred2, testData_ready$stroke)

# Make predictions for Random Forest
rf_pred2 <- predict(rfFit_final, newdata = testData_ready)
rf_cm2 <- confusionMatrix(rf_pred2, testData_ready$stroke)

extract_accuracy <- function(model, cm) {
  tibble(
    model = model,
    Accuracy = cm$overall['Accuracy'],
    `CI Lower` = cm$overall['AccuracyLower'],
    `CI Upper` = cm$overall['AccuracyUpper']
  )
}

# Combine accuracies into a table
accuracies <- bind_rows(
  extract_accuracy('Linear Discriminant Analysis', lda_cm2),
  extract_accuracy('Penalized Logistic Regression', plr_cm2),
  extract_accuracy('Nearest Shrunken Centroids', nsc_cm2),
  extract_accuracy('Neural Network', nn_cm2),
  extract_accuracy('Random Forest', rf_cm2),
)

# Display the table using gt package
accuracies %>%
  arrange(-Accuracy) %>%
  gt() %>%
  fmt_number(columns = c(Accuracy, `CI Lower`, `CI Upper`), decimals = 3)

```

model	Accuracy	CI Lower	CI Upper
Random Forest	0.746	0.718	0.772
Neural Network	0.720	0.692	0.748
Penalized Logistic Regression	0.717	0.689	0.745
Linear Discriminant Analysis	0.711	0.682	0.739
Nearest Shrunken Centroids	0.663	0.634	0.692

Final Model

```

rfFit_final$finalModel

##
## Call:
## randomForest(x = x, y = y, ntree = 100, mtry = param$mtry)
##           Type of random forest: classification
##           Number of trees: 100
## No. of variables tried at each split: 4

```



```
##
##          OOB estimate of  error rate: 17.51%
## Confusion matrix:
##          X0   X1 class.error
## X0 1569  481   0.2346341
## X1   235 1803   0.1153091
```