BST260-Final_Project

Chunlei Yang

2022-12-11

Regression Analysis and Machine Learning for Stroke Prediction

Introduction

Dataset and Question

According to WHO, stroke has been the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. It is meaningful to predict stroke incidence based on some health indicators.

The dataset I used is from Kaggle, which includes categorical variables, such as gender, the incidence of hypertension and heart disease, marriage status, work type, residence type, and smoking status, and continuous variables, like age, the average glucose level in the blood, and BMI. The outcome variable is whether the patient had a stroke or not. This dataset contains 5110 observations in total.

We want to study what characteristic of a patient is more likely to get a stroke and how to predict the incidence of stroke based on these factors. We will use all variables as predictors and stroke incidence as the outcome to build machine learning models and determine the model with the best accuracy.

Data Cleaning

Overall, this dataset is ready to use, but some places still need to be cleaned. BMI is stored as the character, so we need to change it to numeric data. Also, there are some NAs in the BMI column. Because only 3.93% of BMI records are missing, we can simply use the gender-specific BMI average to fill in those missing data. Also, there are 30.2% of smoking statuses missing. From missing data analysis, we can find out that the missing data for smoking may not be due to MCAR, so we cannot simply use mean or median to generate so much missing data. We need to use the mice (Multiple Imputation by Chained Equations) package, providing missing value imputation methods to generate missing values.

EDA & Data Pre-Processing

The Frequency of Categorical Levels plot displays the proportion of each variable and its categories. We can notice that in this sample, female is more than male (59% vs. 41%), and ever-married people are more than never-married people (66% vs. 34%). Also, 90% of people have no incidence of hypertension, and 95% of participants never had heart disease. More than half people never smoked, and 20% of the participants are smokers now. Note that 13% of participants are children.

The Histograms of Numeric Columns and data summary show that half of the people in this sample are aged between 25 to 61, with a mean of 43.2. Half of the people's average glucose levels are between 77 and 114, with a mean of 106. Most people have a BMI of 20 to 40.

What's more, we calculate the correlation between stroke and all other factors and make the plot. From this initial study, we can find there are four main driving factors that lead to the incidence of stroke, which are age (older compared to younger), heart disease, higher glucose level, and hypertension, which is expected based on our intuition. But we need to further study the association in our regression model.

There is a very important issue with imbalanced data. We can notice that only 249 (4.9%) subjects have the incidence of stroke, which means even guessing the person does not have a stroke can achieve an accuracy of 95%. If we do not deal with this imbalance, even if the accuracy is high, the sensitivity may be rather low, which means the model cannot predict true positive cases very well, so we need to deal with this imbalance. I use the MWMOTE (majority Weighted Minority Oversampling Technique for Imbalanced Data Set Learning) method in the imbalance package to generate stroke data. The simulation results show that our method is better than or comparable with some other existing methods in terms of various assessment metrics.

Methodology

I will split my data to train data and test data into 80% and 20% and use 10-fold validation to perform cross-validation when we train the data. For the model choices, I use logistic regression, which is suitable for categorical data. Also, for the machine learning part, I use the random forest, boosted trees (XGBoost), KNN, SVM (support vector machine), LDA, and Neural Network methods in machine learning to train and test data. I will mainly use the caret package, providing me train function. And for some models, I will also use a tuning grid to find the best tune for this model to increase the accuracy. In the end, I will use the test data to predict the outcome and calculate the confusion matrix and get the accuracy of all these methods and find the best model.

Results

Logistic Regression

First, I fit a logistic regression including all variables. Based on the summary table of logistic regression, we can notice that age, hypertension, heart disease, average glucose level, BMI, and smoking are significantly associated with the incidence of stroke, controlling for all other variables. Furthermore, I chose a stepwise method based on AIC to remove non-significant variables. In the stepwise logistic regression model, gender and residence type are removed. We find that age (older), average glucose level (higher), BMI (lower), heart disease, hypertension, and smoking are a statistically significant impact on whether a person has a stroke or not. Most of them are expected, while BMI is negatively associated with the incidence of stroke. It may be due to some selection bias in this data sample. One of the most detrimental factors is smoking. The odds ratio of the incidence of stroke among people who are smokers now is 2.03 times higher than those who have never smoked. Even the odds ratio among former smokers is 2.07 times higher. Also, the odds ratio of the incidence of stroke increases by 11% for every year increase in age, and the p-value is almost 0, so age is highly associated with the incidence of stroke. What's more, the odds ratio among people who has hypertension and among people with heart disease is 45.3% and 34.2% higher. Those estimates agree well with our EDA.

Logistic regression is very useful for us to determine what characteristic of a patient is more likely to get a stroke, but it is not a good method to predict because the accuracy for logistic regression and stepwise logistic regression is 0.827 and 0.809, not good enough. We need to use some other machine learning methods to get the better prediction.

Random Forest

Random forest is a great machine learning approach to address the shortcomings of decision trees, improving prediction performance and reducing instability by averaging multiple decision trees. Also, instead of the "rf" method, I use the "ranger" method to increase the speed of calculation for random forest algorithms. From the plot of random forest fit, we can see the error does not decrease when the number of trees is larger than 300. The random forest model gets a very high accuracy of 0.9393. The sensitivity is 0.9208, and the specificity is 0.9578

I also evaluate the importance of the variables in terms of making predictions. From the plot of importance, we can find the dominant predictor is age, which importance is far beyond other variables. Also, the other two main predictors are BMI and average glucose level. It seems that only continous variables have more importance.

Gradient Boosted Trees

Like random forests, gradient boosting trees also combine decision trees but start the combining process at the beginning instead of at the end. Random forests build each tree independently, while gradient boosting builds one tree at a time. Usually, gradient boosting trees will get better accuracy than random forests. For the stroke dataset, the gradient boosting trees model has an accuracy of 0.9594, with a sensitivity of 0.9578 and specificity of 0.9609, which are high enough.

KNN

In KNN, the outcome is classified by a plurality vote of its neighbors, which is a very popular model. However, KNN seems to perform not so well dealing with categorical data. This is true when many of the scales in my model are categorical. It is hard to tell the distance between numerical and categorical variables like age and marital status, or within categorical variables, like private work and self-employed. We want to find out the optimized k for our KNN model, and we find 2 is the optimized k. In conclusion, although the accuracy is 0.8704, I think it is not appropriate to use KNN for this dataset.

SVM

SVM (Support Vector Machine) is a model that draws a boundary that clearly separates two or more classes for classification. We need to choose different C values to tune our model. The accuracy for SVM is relatively low, 0.8266.

LDA

LDA (Linear Discriminant Analysis) is a generalization of Fisher's linear discriminant to find a linear combination of features that characterizes or separates two or more classes of objects or events. We also fit the LDA model using caret, and the accuracy is 0.8272, which is not good enough.

Neural Network

A neural network is a biologically inspired method for computers to learn through analyzing data. We can use the nnet package to fit a single-hidden-layer feed-forward neural network using multinomial log-linear models. Also, we use the caret package to train the model. We need to set the tuning grid. The tuning parameters are weight (decay), and the number of hidden units (size). Also, set the maximum iteration to 1000. The accuracy is 0.9208, which is relatively high.

Conclusion

From our logistic regression and stepwise regression model, age, higher average glucose level, heart disease, hypertension, and smoking are the main factors to increase the risk of stroke. People with these characteristics should pay much more attention to the potential incidence of stroke. Even as a former smoker, the odds ratio of the incidence of stroke is 2.07 times higher than those who never smoke. Therefore, it is important to choose not to smoke at the beginning to decrease the risk. Also, there is an 11% increase in adds ratio for every year increase in age, so aged people should have regular body exams for potential stroke. Also, other diseases, like hypertension, heart disease, and high glucose level, are also highly associated with the increasing risk of stroke, and this may cause more severe outcomes if the disease is composite. Although we cannot change our age, we could form a good life from healthy diet or regular training to diminish other risk factors such as heart disease, hypertension, or average glucose level, and be a non-smoker, for your health and decrease the risk of stroke.

Based on our machine learning model, including random forest, gradient boosting trees (XGBoost), KNN, SVM (support vector machine), LDA, and Neural Network, we train these models using caret package and find that random forest, gradient boosting trees, and Neural Network performs relatively better. The model with the best accuracy is gradient boosting trees, with an accuracy of 0.9594, sensitivity of 0.9578, and specificity of 0.9609. The accuracy is higher than 95%, which could be a satisfactory result. The models with the worst accuracy is SVM and LDA, with accuracy of 0.8266 and 0.8272. In conclusion, I would choose gradient boosting trees model for stroke prediction.

The potential issue is the imbalanced data problem. I use MWMOTE to generate data, but they are not categorical, so I just round them to the nearest integer, which may cause some inaccuracy. Also, from the importance plot for the random forest, categorical data does not contribute much to the prediction compared to the continuous variable, and it may limit the accuracy of the random forest. I want to investigate further how to deal with imbalanced data and how to better train categorical data in machine learning nad learn some advanced models in the future.

Appendix (codes and plots)

load data

```
library(tidyverse)
library(dplyr)
library(caret)
library(randomForest)
library(imbalance)
library(cvms)
library(inspectdf)
library(ggimage)
library(rsvg)
```

```
stroke <- read_csv("healthcare-dataset-stroke-data.csv")
```

data cleaning

Data preview

```
str(stroke)
```

```
## spc_tbl_ [5,110 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ id                : num [1:5110] 9046 51676 31112 60182 1665 ...
##  $ gender            : chr [1:5110] "Male" "Female" "Male" "Female" ...
##  $ age               : num [1:5110] 67 61 80 49 79 81 74 69 59 78 ...
##  $ hypertension      : num [1:5110] 0 0 0 0 1 0 1 0 0 0 ...
##  $ heart_disease    : num [1:5110] 1 0 1 0 0 0 1 0 0 0 ...
##  $ ever_married     : chr [1:5110] "Yes" "Yes" "Yes" "Yes" ...
##  $ work_type        : chr [1:5110] "Private" "Self-employed" "Private" "Private" ...
##  $ Residence_type   : chr [1:5110] "Urban" "Rural" "Rural" "Urban" ...
##  $ avg_glucose_level: num [1:5110] 229 202 106 171 174 ...
##  $ bmi              : chr [1:5110] "36.6" "N/A" "32.5" "34.4" ...
##  $ smoking_status   : chr [1:5110] "formerly smoked" "never smoked" "never smoked" "smokes" ...
##  $ stroke           : num [1:5110] 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   id = col_double(),
##   ..   gender = col_character(),
##   ..   age = col_double(),
##   ..   hypertension = col_double(),
##   ..   heart_disease = col_double(),
##   ..   ever_married = col_character(),
##   ..   work_type = col_character(),
##   ..   Residence_type = col_character(),
##   ..   avg_glucose_level = col_double(),
##   ..   bmi = col_character(),
##   ..   smoking_status = col_character(),
##   ..   stroke = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Drop id variable

```
stroke <- stroke |> dplyr::select(-id)
```

Change bmi to numeric data

```
stroke <- stroke |> mutate_at("bmi", as.numeric)
```

Remove Other gender because it only has one row

```
stroke |> filter(gender == "Other") |> nrow() # only one row
```

```
## [1] 1
```

```
stroke <- filter(stroke, gender != "Other")
```

Change some columns to categorical data

```
stroke <- stroke |> mutate(gender = factor(gender, levels = c("Female", "Male")),
                           hypertension = factor(hypertension, labels = c("No", "Yes")),
                           heart_disease = factor(heart_disease, labels = c("No", "Yes")),
                           ever_married = factor(ever_married, levels = c("No", "Yes")),
                           work_type = factor(work_type,
                                              levels = c("children", "Never_worked", "Self-employed", "Priva
                           Residence_type = factor(Residence_type, levels = c("Rural", "Urban")),
                           stroke = factor(stroke, labels = c("No", "Yes")))
```

Find NA

```r
colSums(is.na(stroke))
```

```
##            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 
##               201                 0                 0
```

For BMI, Replace NA to mean BMI

```r
stroke <- stroke |> group_by(gender) |> mutate(bmi = ifelse(is.na(bmi), mean(bmi, na.rm=TRUE), bmi))
```

We can further notice that there is “Unknown” category in Smoking status. Change Unknown to NA.

```r
stroke <- stroke |> mutate(smoking_status = ifelse(smoking_status == "Unknown", NA, smoking_status)) |>
```

NA account for 30%

```r
prop.table(table(stroke$smoking_status, exclude=NULL))
```

```
## 
##    never smoked formerly smoked          smokes            <NA> 
##       0.3703269       0.1730280       0.1544334       0.3022118
```

Missing data analysis for smoking_status

```r
library(finalfit)
var <- names(stroke)
explanatory <- var[var != "smoking_status" & var != "bmi"]
dependent <- var[var == "smoking_status"]
stroke |> missing_compare(dependent, explanatory)
```

```
##  Missing data analysis: smoking_status                 Not missing     Missing
##                                gender        Female 2158 (72.1)  836 (27.9)
##                                                Male 1407 (66.5)  708 (33.5)
##                                   age     Mean (SD) 48.9 (18.9) 30.2 (25.1)
##                          hypertension            No 3119 (67.6) 1492 (32.4)
##                                                 Yes  446 (89.6)   52 (10.4)
##                         heart_disease            No 3337 (69.0) 1496 (31.0)
##                                                 Yes  228 (82.6)   48 (17.4)
##                          ever_married            No  855 (48.7)  901 (51.3)
##                                                 Yes 2710 (80.8)  643 (19.2)
##                             work_type      children   69 (10.0)  618 (90.0)
##                                        Never_worked   14 (63.6)    8 (36.4)
##                                       Self-employed  663 (81.0)  156 (19.0)
##                                             Private 2284 (78.1)  640 (21.9)
##                                            Govt_job  535 (81.4)  122 (18.6)
##                        Residence_type         Rural 1751 (69.7)  762 (30.3)
##                                              Urban  1814 (69.9)  782 (30.1)
##                        avg_glucose_level     Mean (SD) 109.0 (48.3) 99.6 (36.6)
##                                   stroke            No  3363 (69.2) 1497 (30.8)
##                                                    Yes   202 (81.1)   47 (18.9)
##       p
##  <0.001
##
##  <0.001
##  <0.001
##
##  <0.001
##
##  <0.001
##
##  <0.001
##
##
##
##
##   0.901
##
##  <0.001
##  <0.001
##
```

We will use mice package to fill the NA values for smoking status

```r
library(mice)
stroke <- mice(stroke, seed = 260)
```

```
##
##  iter imp variable
##   1   1  smoking_status
##   1   2  smoking_status
##   1   3  smoking_status
##   1   4  smoking_status
##   1   5  smoking_status
##   2   1  smoking_status
##   2   2  smoking_status
##   2   3  smoking_status
##   2   4  smoking_status
##   2   5  smoking_status
##   3   1  smoking_status
##   3   2  smoking_status
##   3   3  smoking_status
##   3   4  smoking_status
##   3   5  smoking_status
##   4   1  smoking_status
##   4   2  smoking_status
##   4   3  smoking_status
##   4   4  smoking_status
##   4   5  smoking_status
##   5   1  smoking_status
##   5   2  smoking_status
##   5   3  smoking_status
##   5   4  smoking_status
##   5   5  smoking_status
```

```
stroke <- complete((stroke))
```

EDA

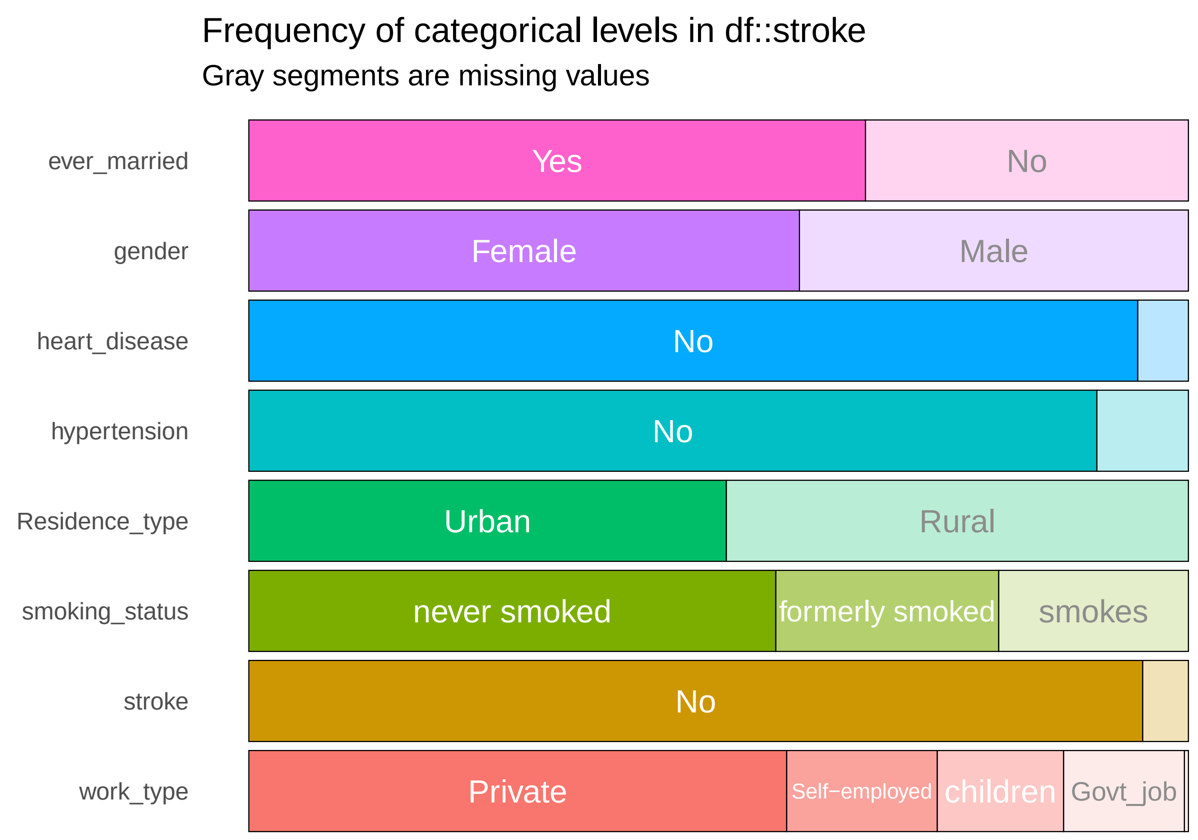
Summary of the data now

```
summary(stroke)
```

```
##     gender          age         hypertension heart_disease ever_married
##  Female:2994   Min.   : 0.08   No :4611     No :4833      No :1756    
##  Male  :2115   1st Qu.:25.00   Yes: 498     Yes: 276      Yes:3353    
##                Median :45.00                                          
##                Mean   :43.23                                          
##                3rd Qu.:61.00                                          
##                Max.   :82.00                                          
##          work_type    Residence_type avg_glucose_level      bmi       
##  children     : 687   Rural:2513     Min.   : 55.12    Min.   :10.30  
##  Never_worked :  22   Urban:2596     1st Qu.: 77.24    1st Qu.:23.80  
##  Self-employed: 819                  Median : 91.88    Median :28.40  
##  Private      :2924                  Mean   :106.14    Mean   :28.89  
##  Govt_job     : 657                  3rd Qu.:114.09    3rd Qu.:32.80  
##                                      Max.   :271.74    Max.   :97.60  
##          smoking_status stroke    
##  never smoked   :2865   No :4860  
##  formerly smoked:1212   Yes: 249  
##  smokes         :1032             
##                                   
##                                   
##
```

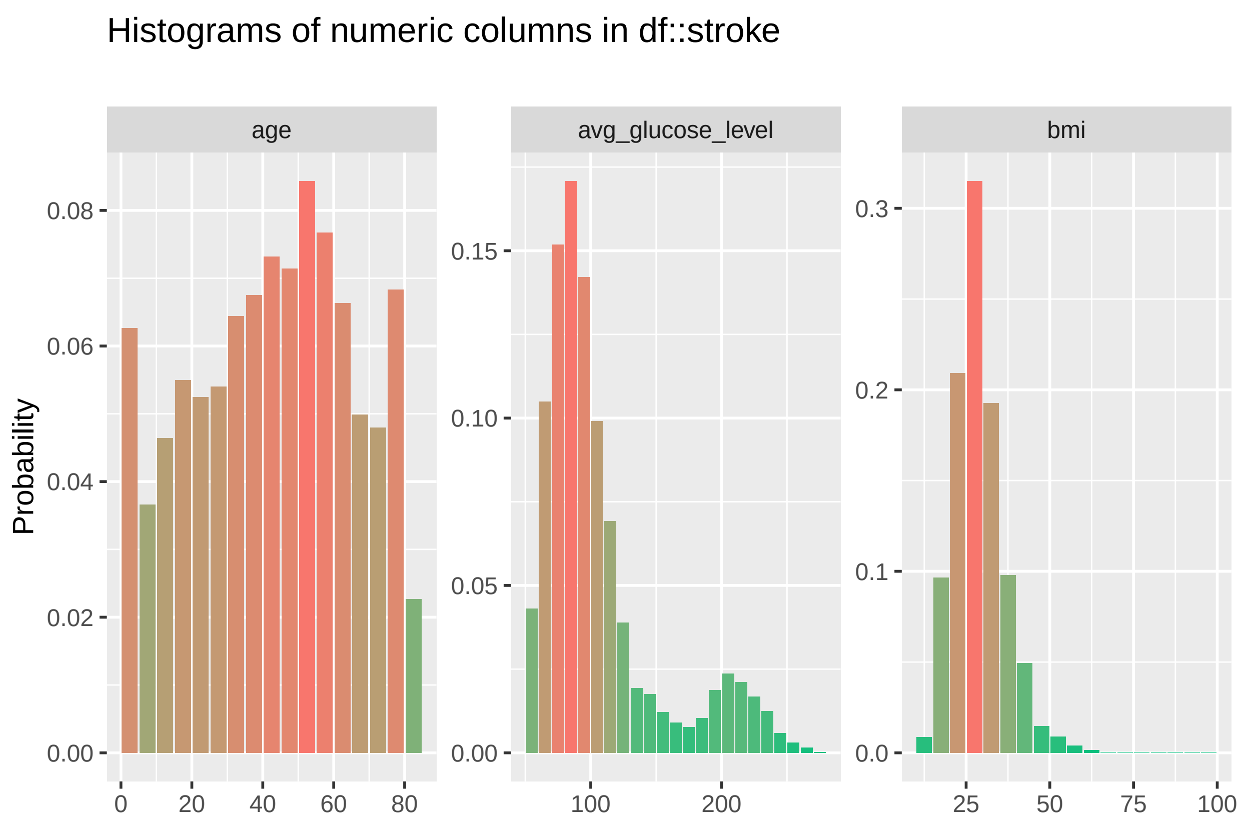
Frequency of Categorical Levels plot

```
stroke |> inspect_cat() |> show_plot()
```



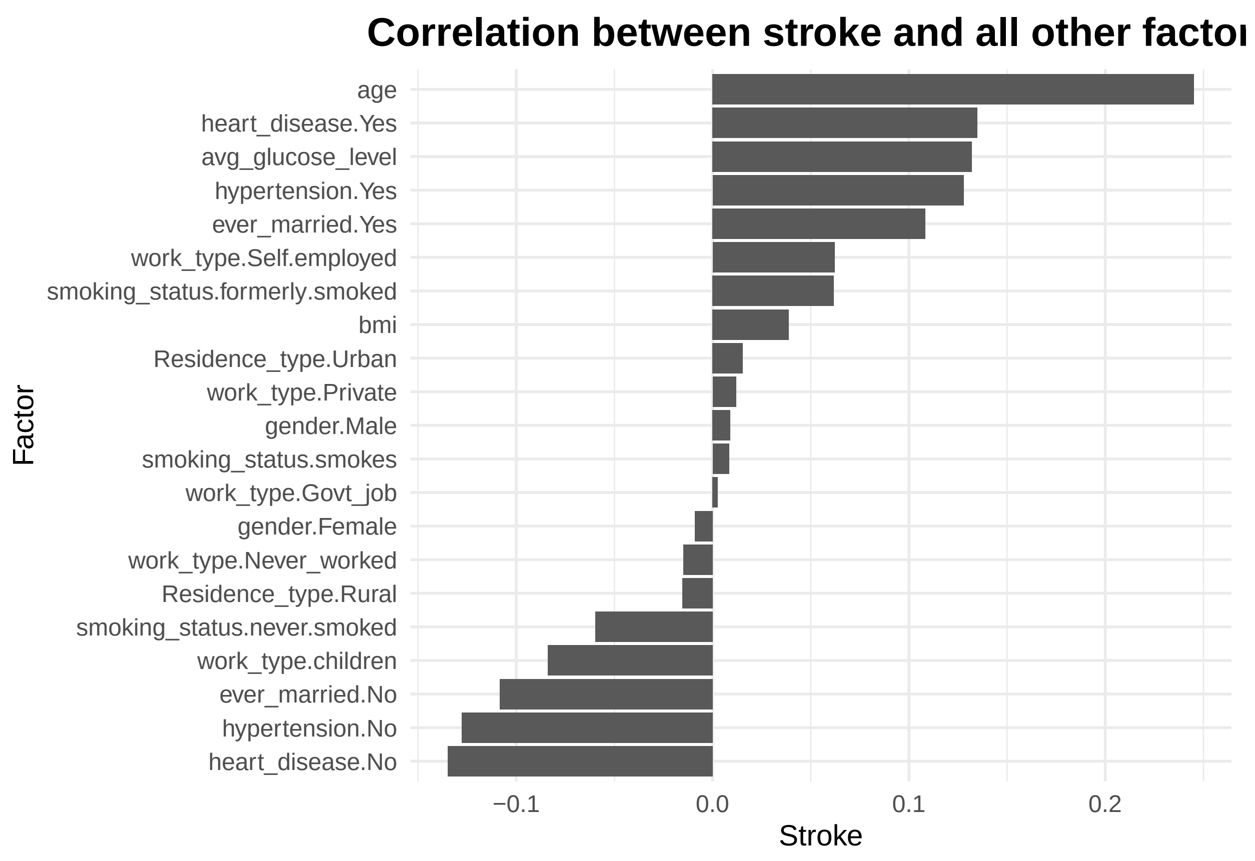
Histograms of Numeric Columns

```
stroke |> inspect_num() |> show_plot()
```



Correlation between stroke and all other factors

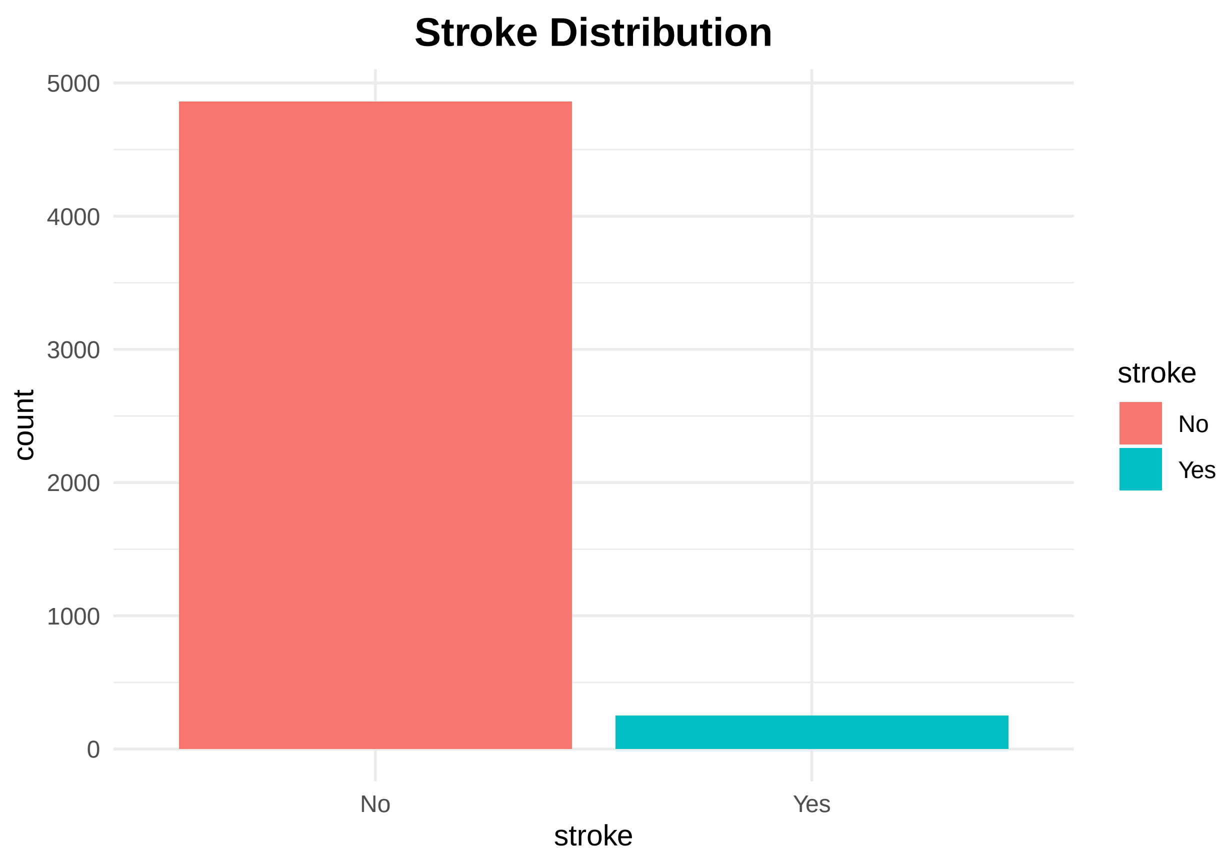
```
library(corrr)
library(ggcorrplot)
stroke1 <- stroke |> mutate(stroke = as.numeric(stroke))
dmy <- dummyVars("~ .", data = stroke1)
onehot <- data.frame(predict(dmy, stroke1))
cor <-correlate(onehot, use = "pairwise.complete.obs")
cor |> focus(stroke) |>
  ggplot(aes(x = reorder(term, stroke), stroke)) +
  geom_col() + coord_flip() +
  theme_minimal() +
  theme(plot.title = element_text(size = 15, face = "bold", hjust = 0.5)) +
  labs(y = "Stroke", x = "Factor", title = "Correlation between stroke and all other factors")
```



```r
rm(stroke1)
```

Imbalanced data for stroke outcome

```r
stroke |>
      ggplot() +
      geom_bar(aes(x = stroke, fill = stroke), position = "dodge") +
      theme_minimal() +
      theme(plot.title = element_text(size = 15, face = "bold", hjust = 0.5)) +
      labs(y = "count", title = "Stroke Distribution")
```



Only 5% of people had a stroke, which means even guessing the person does not have stroke can achieve the accuracy of 95%. If we do not deal with this imbalance, even the accuracy is high, the sensitivity may be rather low, which means the model cannot predict true positive cases very well, so we need to deal with this imbalance.

Generate new sample using MWMOTE

```
library(imbalance)
set.seed(260)
stroke <- stroke |> mutate(across(where(is.factor), as.numeric))
stroke$stroke <- as.numeric(stroke$stroke)
newSample <- mwmote(stroke, classAttr = "stroke", numInstances = 4611)
newSample <- round(newSample)
```

Add generated sample to stroke database

```
stroke <- rbind(stroke, newSample)
rm(newSample)
```

Change stroke variables back to factor

```
stroke <- stroke |> mutate(gender = factor(gender, levels = c(1, 2),
                                           labels = c("Female", "Male")),
                           hypertension = factor(hypertension, levels = c(1, 2),
                                                 labels = c("No", "Yes")),
                           heart_disease = factor(heart_disease, levels = c(1, 2),
                                              labels = c("No", "Yes")),
                           ever_married = factor(ever_married, levels = c(1, 2),
                                                     labels = c("No", "Yes")),
                           work_type = factor(work_type, levels = c(1, 2, 3, 4, 5),
                              labels = c("children", "Never_worked", "Self-employed", "Private", "Govt_
                           Residence_type = factor(Residence_type, levels = c(1, 2),
                                     labels = c("Rural", "Urban")),
                           smoking_status = factor(smoking_status, levels = c(1, 2, 3), labels = c("neve
                           stroke = factor(stroke, levels = c(1, 2),
                                           labels = c("No", "Yes")))
```

```
table(stroke$stroke)
```

```
##
##   No  Yes
## 4860 4860
```

Model

- Data spliting

```
set.seed(260)
train_index <- createDataPartition(y = stroke$stroke, p = 0.8, list = FALSE)
train_set <- stroke[train_index, ]
test_set <- stroke[-train_index, ]
```

- 10-fold validation

```
control <- trainControl(method = "cv", number = 10, p = .9)
```

1. Logistic regression

```
train_glm <- train(stroke ~ ., data = train_set,
                 method = "glm",
                 trControl = control,
                 family = "binomial")
summary(train_glm)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6452  -0.4784   0.0889   0.6406   3.5827
##
## Coefficients:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    -7.520e+00  1.011e+00  -7.439 1.02e-13 ***
## genderMale                     -6.242e-02  6.667e-02  -0.936  0.34914
## age                              1.065e-01  2.939e-03  36.227  < 2e-16 ***
## hypertensionYes                  3.798e-01  8.427e-02   4.506 6.59e-06 ***
## heart_diseaseYes                 3.070e-01  1.015e-01   3.026  0.00248 **
## ever_marriedYes                 -1.617e-01  1.075e-01  -1.504  0.13255
## work_typeNever_worked           -8.869e+00  2.237e+02  -0.040  0.96837
## `work_typeSelf-employed`         3.228e-01  1.024e+00   0.315  0.75267
## work_typePrivate                 7.449e-01  1.020e+00   0.730  0.46533
## work_typeGovt_job                3.964e-01  1.022e+00   0.388  0.69820
## Residence_typeUrban              6.998e-02  6.451e-02   1.085  0.27796
## avg_glucose_level                6.947e-03  6.578e-04  10.562  < 2e-16 ***
## bmi                             -2.554e-02  6.217e-03  -4.108 4.00e-05 ***
## `smoking_statusformerly smoked`  7.341e-01  7.414e-02   9.901  < 2e-16 ***
## smoking_statussmokes             7.129e-01  9.036e-02   7.889 3.05e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10779.8  on 7775  degrees of freedom
## Residual deviance:  6116.2  on 7761  degrees of freedom
## AIC: 6146.2
##
## Number of Fisher Scoring iterations: 13
```

- Stepwise logistic regression

```
library(MASS)
fit_glm <- glm(stroke~., data = train_set,
               family = "binomial")
step_glm <- stepAIC(fit_glm)
```

```
## Start:  AIC=6146.17
## stroke ~ gender + age + hypertension + heart_disease + ever_married +
##     work_type + Residence_type + avg_glucose_level + bmi + smoking_status
##
##                     Df Deviance    AIC
## - gender             1   6117.0 6145.0
## - Residence_type     1   6117.3 6145.3
## <none>                   6116.2 6146.2
## - ever_married       1   6118.4 6146.4
## - heart_disease      1   6125.6 6153.6
## - bmi                1   6133.3 6161.3
## - hypertension       1   6136.9 6164.9
## - work_type          4   6152.3 6174.3
## - avg_glucose_level  1   6232.2 6260.2
## - smoking_status     2   6235.5 6261.5
## - age                1   8283.5 8311.5
##
## Step:  AIC=6145.04
## stroke ~ age + hypertension + heart_disease + ever_married +
##     work_type + Residence_type + avg_glucose_level + bmi + smoking_status
##
##                     Df Deviance    AIC
## - Residence_type     1   6118.3 6144.3
## <none>                   6117.0 6145.0
## - ever_married       1   6119.4 6145.4
## - heart_disease      1   6125.8 6151.8
## - bmi                1   6134.3 6160.3
## - hypertension       1   6137.5 6163.5
## - work_type          4   6152.7 6172.7
## - avg_glucose_level  1   6232.4 6258.4
## - smoking_status     2   6235.9 6259.9
## - age                1   8283.8 8309.8
##
## Step:  AIC=6144.3
## stroke ~ age + hypertension + heart_disease + ever_married +
##     work_type + avg_glucose_level + bmi + smoking_status
##
##                     Df Deviance    AIC
## <none>                   6118.3 6144.3
## - ever_married       1   6120.9 6144.9
## - heart_disease      1   6127.1 6151.1
## - bmi                1   6135.5 6159.5
## - hypertension       1   6138.4 6162.4
## - work_type          4   6154.3 6172.3
## - avg_glucose_level  1   6234.6 6258.6
## - smoking_status     2   6238.1 6260.1
## - age                1   8301.1 8325.1
```

```
summary(step_glm)
```

```
##
## Call:
## glm(formula = stroke ~ age + hypertension + heart_disease + ever_married +
##     work_type + avg_glucose_level + bmi + smoking_status, family = "binomial",
##     data = train_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6445  -0.4788   0.0896   0.6407   3.5809
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -7.515e+00  1.010e+00  -7.439 1.01e-13 ***
## age                           1.065e-01  2.935e-03  36.306  < 2e-16 ***
## hypertensionYes               3.739e-01  8.418e-02   4.441 8.94e-06 ***
## heart_diseaseYes              2.940e-01  1.005e-01   2.927  0.00343 **
## ever_marriedYes              -1.729e-01  1.068e-01  -1.619  0.10546
## work_typeNever_worked        -8.848e+00  2.237e+02  -0.040  0.96845
## work_typeSelf-employed        3.365e-01  1.024e+00   0.329  0.74246
## work_typePrivate              7.598e-01  1.020e+00   0.745  0.45637
## work_typeGovt_job             4.191e-01  1.022e+00   0.410  0.68174
## avg_glucose_level             6.949e-03  6.571e-04  10.575  < 2e-16 ***
## bmi                          -2.555e-02  6.214e-03  -4.111 3.94e-05 ***
## smoking_statusformerly smoked  7.268e-01  7.351e-02   9.886  < 2e-16 ***
## smoking_statussmokes          7.080e-01  8.954e-02   7.907 2.63e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10779.8  on 7775  degrees of freedom
## Residual deviance:  6118.3  on 7763  degrees of freedom
## AIC: 6144.3
##
## Number of Fisher Scoring iterations: 13
```

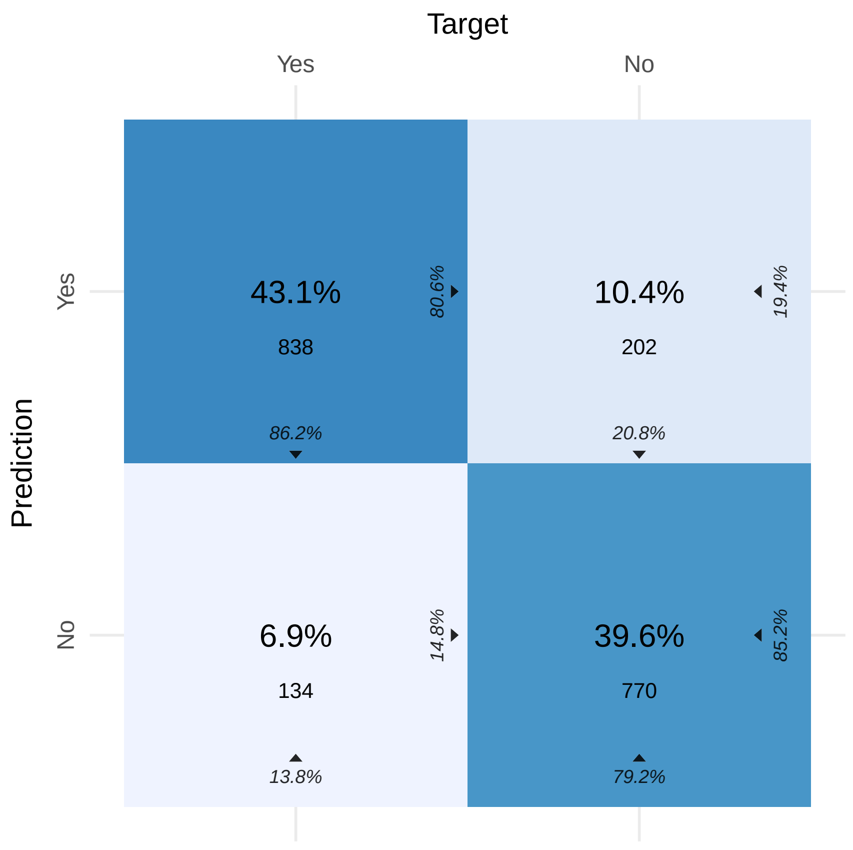
Prediction for logistic regression

```
test_set$stroke_hat_glm <- predict(train_glm, test_set)
cm <- confusionMatrix(test_set$stroke_hat_glm, test_set$stroke)
cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.8271605
```

Confusion matrix plot

```
test_set$stroke_hat_glm <- as.character(test_set$stroke_hat_glm)
conf_mat <- evaluate(test_set, target_col = "stroke", prediction_cols = "stroke_hat_glm",
                        type = "binomial", positive = "Yes")
plot_confusion_matrix(conf_mat)
```



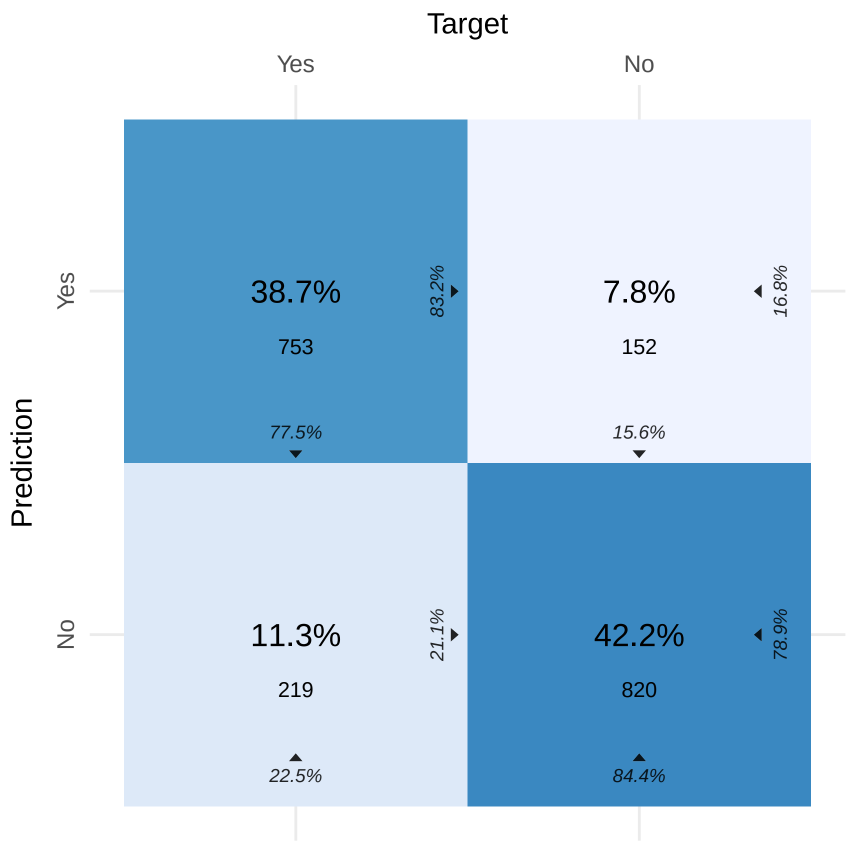
Prediction for stepwise logistic regression

```
p_hat_step <- predict(step_glm, test_set)
test_set$stroke_hat_step <- ifelse(p_hat_step > 0.5, "Yes", "No") |> factor()
cm <- confusionMatrix(test_set$stroke_hat_step, test_set$stroke)
cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.8091564
```

Confusion matrix plot

```
test_set$stroke_hat_step <- as.character(test_set$stroke_hat_step)
conf_mat <- evaluate(test_set, target_col = "stroke", prediction_cols = "stroke_hat_step",
                        type = "binomial", positive = "Yes")
plot_confusion_matrix(conf_mat)
```



2. Random Forest

```
library(ranger)
```

```
##
## Attaching package: 'ranger'
## The following object is masked from 'package:randomForest':
##
##     importance
```

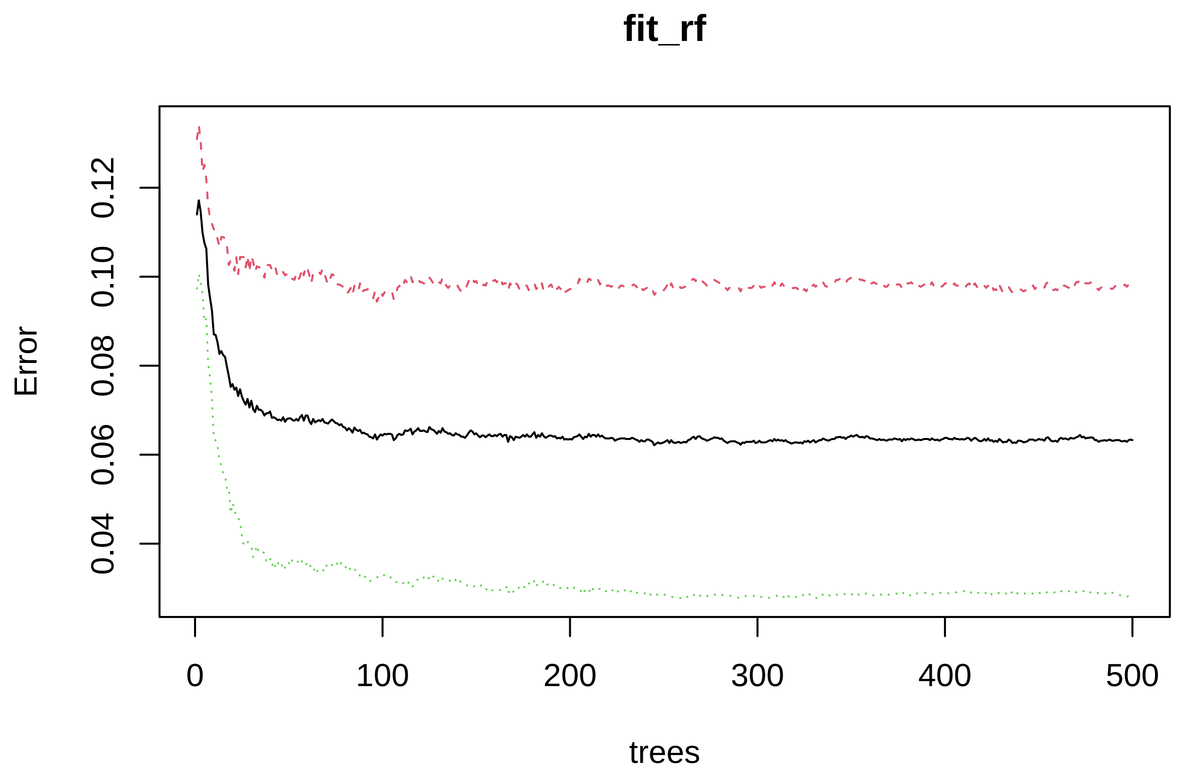
```
set.seed(260)
grid <- data.frame(
  .mtry = seq(1, 11, 1),
  .splitrule = "gini",
  .min.node.size = 0
)

train_rf <- train(stroke ~ ., data = train_set,
                  method = "ranger",
                  trControl = control,
                  tuneGrid = grid)
```

```
set.seed(260)
fit_rf <- randomForest(stroke ~ ., data = train_set,
                       mtry = train_rf$bestTune$mtry)
```

Error vs # trees

```
plot(fit_rf)
```



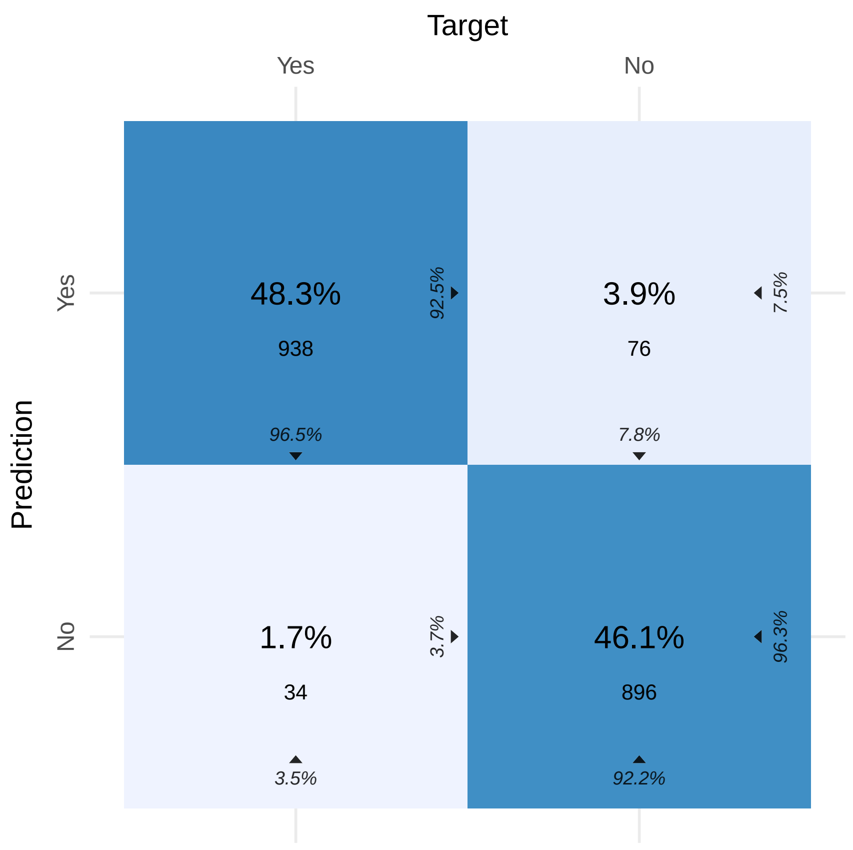
predict

```r
test_set$stroke_hat_rf <- predict(fit_rf, test_set)
cm <- confusionMatrix(test_set$stroke_hat_rf, test_set$stroke)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  896  34
##        Yes  76 938
##
##                Accuracy : 0.9434
##                  95% CI : (0.9322, 0.9533)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8868
##
##  Mcnemar's Test P-Value : 9.26e-05
##
##             Sensitivity : 0.9218
##             Specificity : 0.9650
##          Pos Pred Value : 0.9634
##          Neg Pred Value : 0.9250
##              Prevalence : 0.5000
##          Detection Rate : 0.4609
##    Detection Prevalence : 0.4784
##       Balanced Accuracy : 0.9434
##
##        'Positive' Class : No
##
```

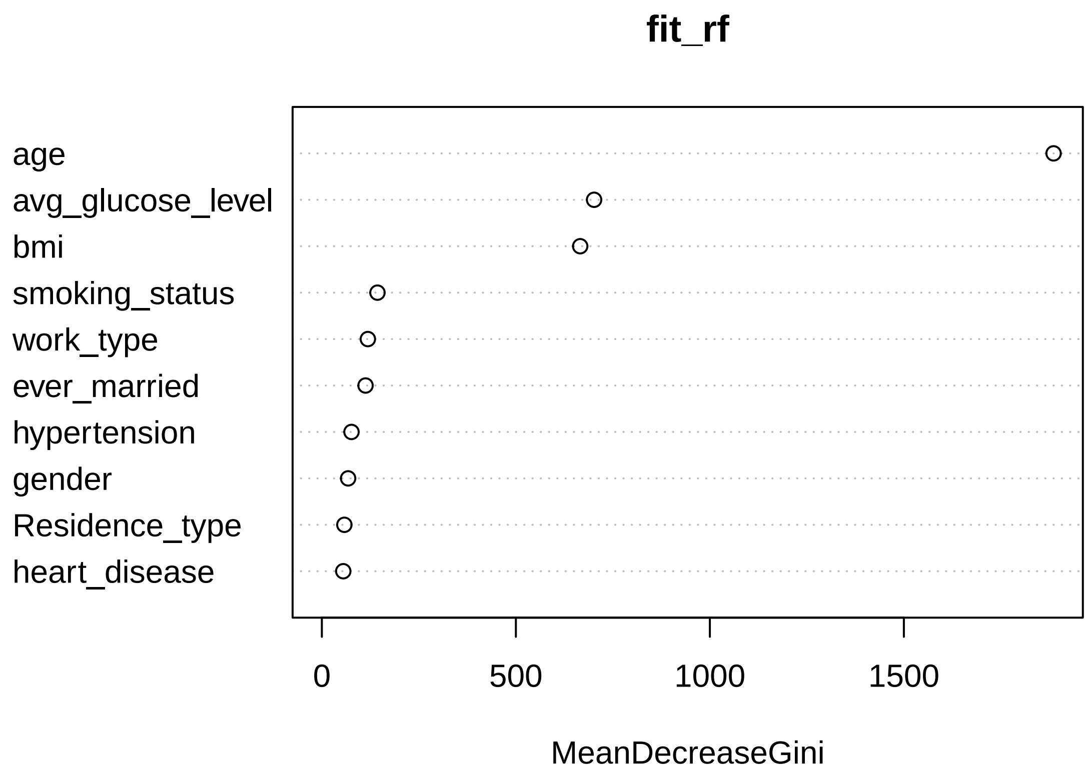
Confusion matrix plot

```r
test_set$stroke_hat_rf <- as.character(test_set$stroke_hat_rf)
conf_mat <- evaluate(test_set, target_col = "stroke", prediction_cols = "stroke_hat_rf",
                     type = "binomial", positive = "Yes")
plot_confusion_matrix(conf_mat)
```



Importance of the variables in terms of making predictions

```
varImpPlot(fit_rf)
```



4. Gradient Boosted Trees

```
library(xgboost)
set.seed(260)

xgbGrid <- expand.grid(
    nrounds = 3500,
    max_depth = 7,
    eta = 0.01,
    gamma = 0.01,
    colsample_bytree = 0.75,
    min_child_weight = 0,
    subsample = 0.5
)

xgbControl <- trainControl(
    method = "cv",
    number = 5
)

train_xgb <- train(
    stroke ~ ., data = train_set,
    method = "xgbTree",
    tuneLength = 10,
    tuneGrid = xgbGrid,
    trControl = xgbControl
)
```

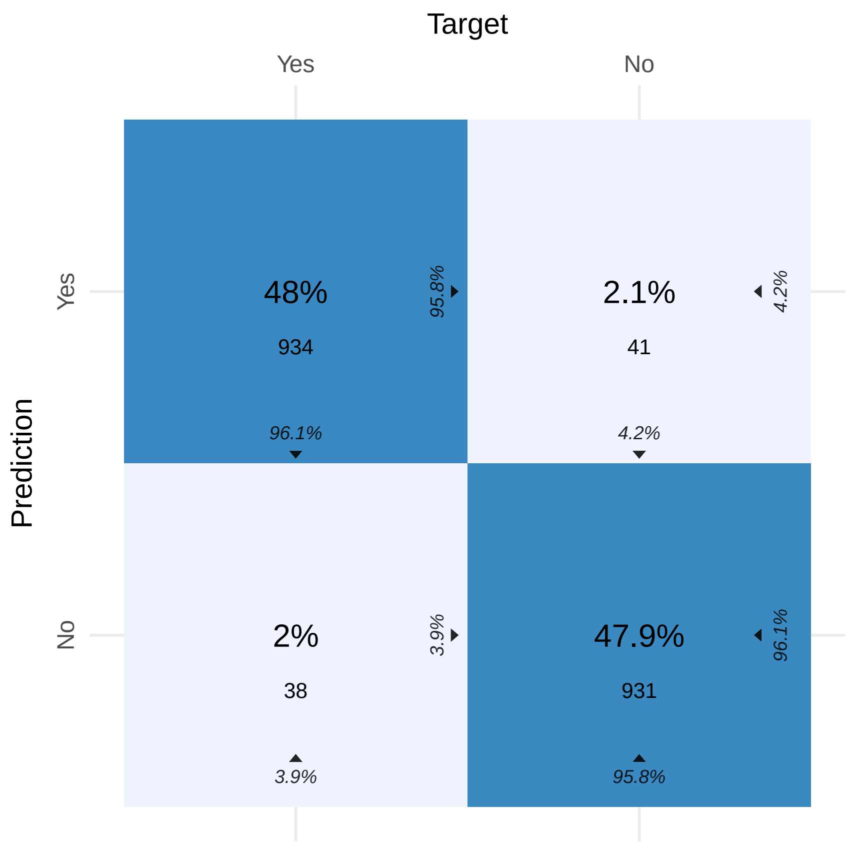
predict

```
test_set$stroke_hat_xgb <- predict(train_xgb, test_set)
cm <- confusionMatrix(test_set$stroke_hat_xgb, test_set$stroke)
cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.9593621
```

Confusion matrix plot

```
test_set$stroke_hat_xgb <- as.character(test_set$stroke_hat_xgb)
conf_mat <- evaluate(test_set, target_col = "stroke", prediction_cols = "stroke_hat_xgb",
                     type = "binomial", positive = "Yes")
plot_confusion_matrix(conf_mat)
```

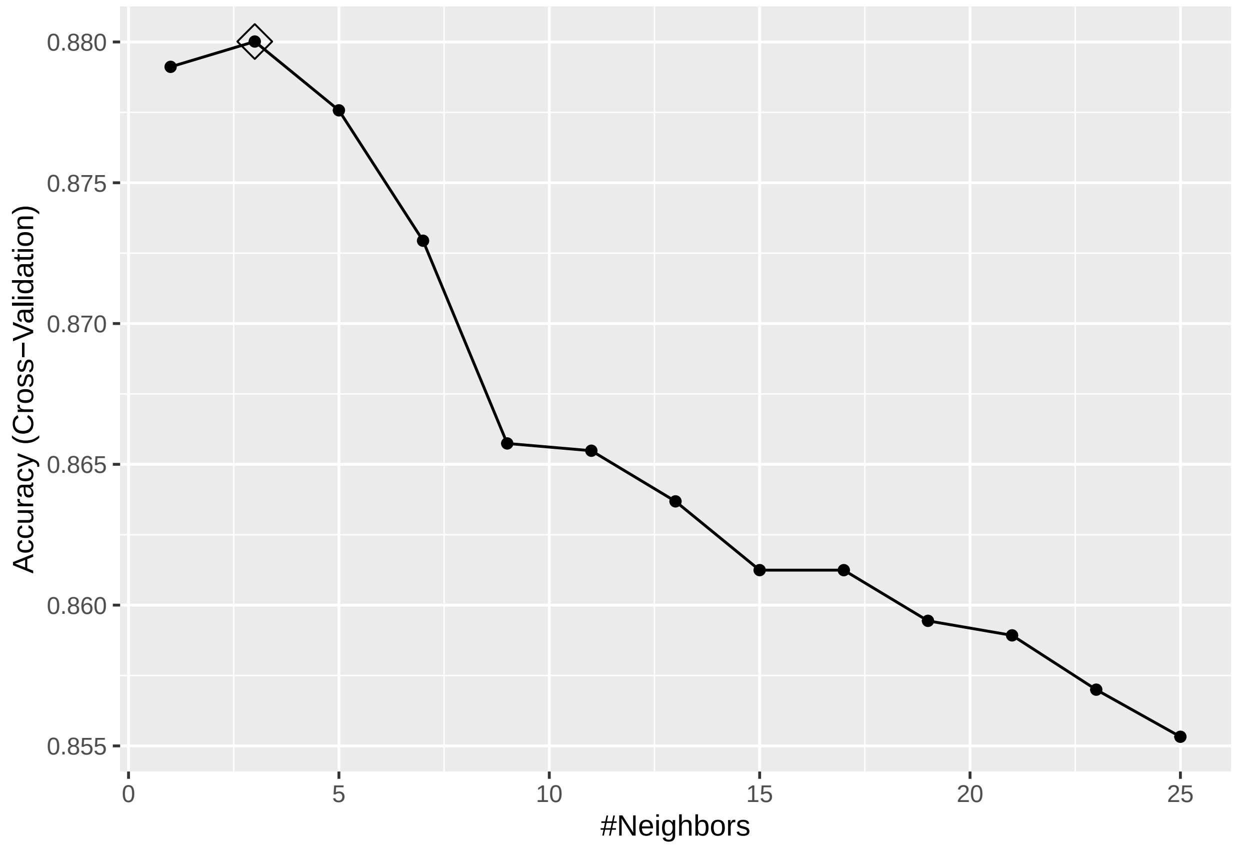


5. KNN

```r
set.seed(260)

train_knn <- train(stroke ~ ., data = train_set,
                   method = "knn",
                   trControl = control,
                   tuneGrid = data.frame(k = seq(1, 25, 2)))

ggplot(train_knn, highlight = TRUE)
```



fit knn model

```r
fit_knn <- knn3(stroke ~ ., data = train_set, k = 2)
```

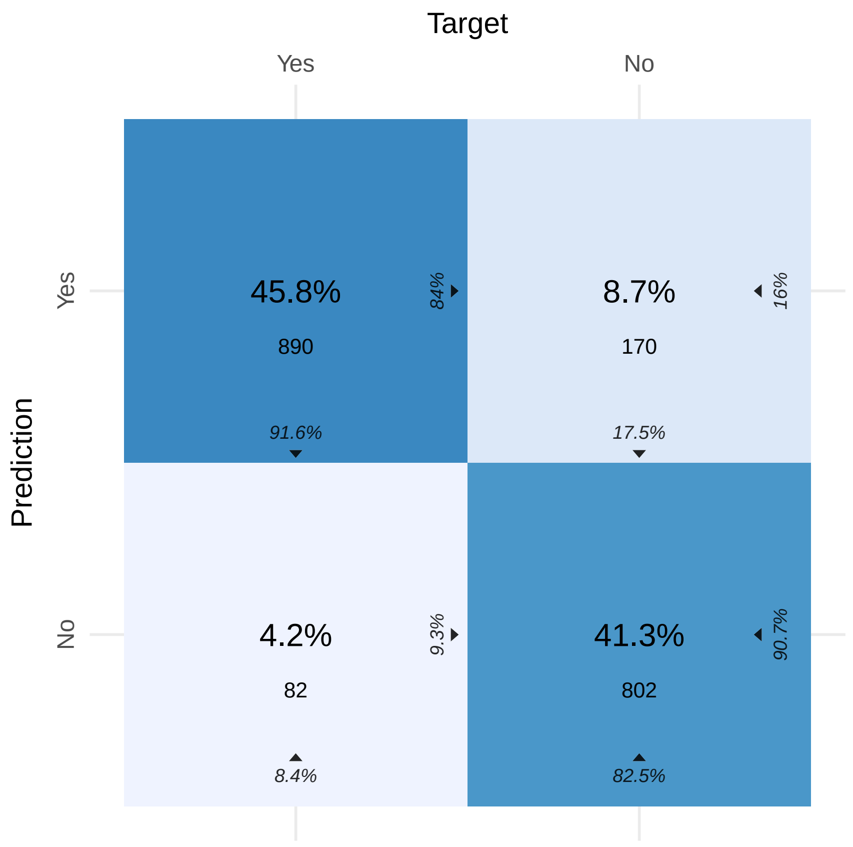
predict

```r
test_set$stroke_hat_knn <- predict(fit_knn, test_set, type="class")
cm <- confusionMatrix(test_set$stroke_hat_knn, test_set$stroke)
cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.8703704
```

Confusion matrix plot

```
test_set$stroke_hat_knn <- as.character(test_set$stroke_hat_knn)
conf_mat <- evaluate(test_set, target_col = "stroke", prediction_cols = "stroke_hat_knn",
                        type = "binomial", positive = "Yes")
plot_confusion_matrix(conf_mat)
```



6. SVM

```
set.seed(260)
library(kernlab)
grid <- data.frame(C = seq(0.1, 1, 0.1))
train_svm <- train(stroke ~ ., data = train_set,
                   method = "svmLinear",
                   trControl = control,
                   tuneGrid = grid)
train_svm
```

```
## Support Vector Machines with Linear Kernel
##
## 7776 samples
##   10 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 6998, 6998, 6999, 6998, 6999, 6999, ...
## Resampling results across tuning parameters:
##
##   C    Accuracy   Kappa
##   0.1  0.8251036  0.6502085
##   0.2  0.8254894  0.6509802
##   0.3  0.8252325  0.6504666
##   0.4  0.8251038  0.6502091
##   0.5  0.8251038  0.6502091
##   0.6  0.8252325  0.6504661
##   0.7  0.8252325  0.6504661
##   0.8  0.8253610  0.6507232
##   0.9  0.8256181  0.6512373
##   1.0  0.8254896  0.6509802
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.9.
```

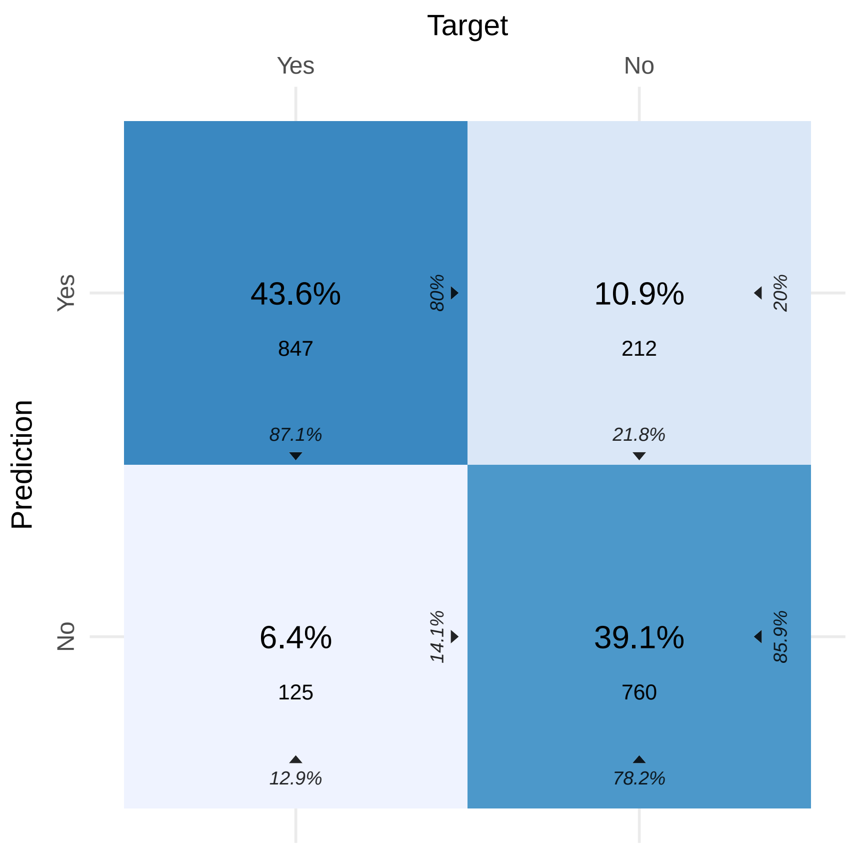
predict

```
test_set$stroke_hat_svm <- predict(train_svm, test_set)
cm <- confusionMatrix(test_set$stroke_hat_svm, test_set$stroke)
cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.8266461
```

Confusion matrix plot

```
test_set$stroke_hat_svm <- as.character(test_set$stroke_hat_svm)
conf_mat <- evaluate(test_set, target_col = "stroke", prediction_cols = "stroke_hat_svm",
                     type = "binomial", positive = "Yes")
plot_confusion_matrix(conf_mat)
```



7. Linear discriminant analysis

```
set.seed(260)
train_lda <- train(stroke ~ ., data = train_set,
                   method = "lda",
                   family = "binomial",
                   trControl = control)
train_lda
```

```
## Linear Discriminant Analysis
##
## 7776 samples
##   10 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 6998, 6998, 6999, 6998, 6999, 6999, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8220168  0.6440344
```

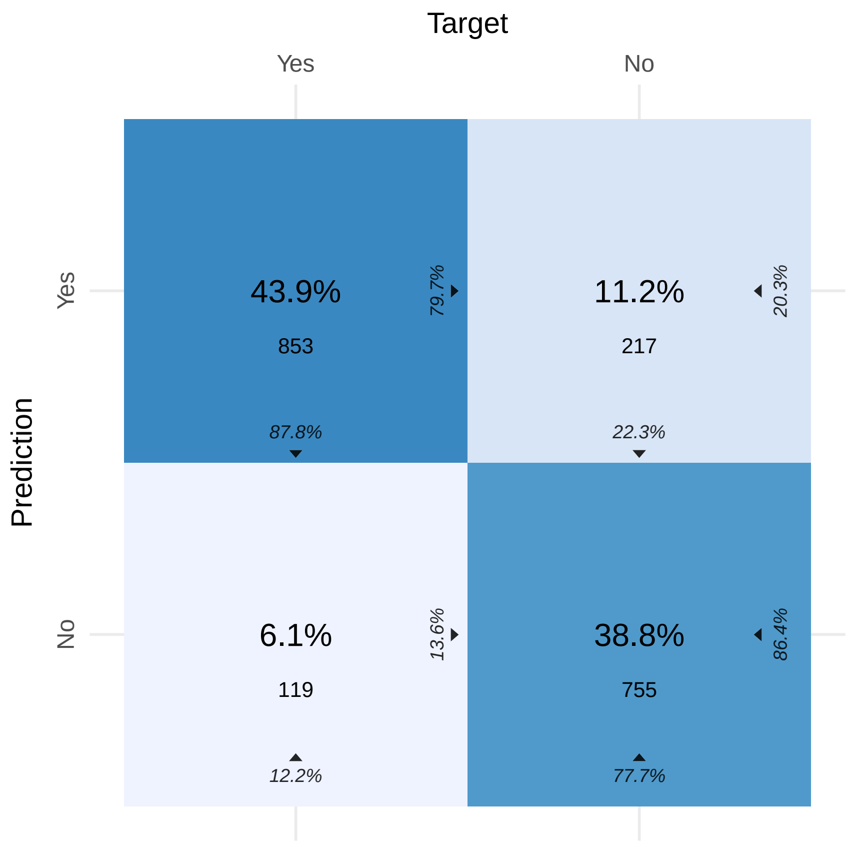
predict

```
test_set$stroke_hat_lda <- predict(train_lda, test_set)
cm <- confusionMatrix(test_set$stroke_hat_lda, test_set$stroke)
cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.8271605
```

Confusion matrix plot

```
test_set$stroke_hat_lda <- as.character(test_set$stroke_hat_lda)
conf_mat <- evaluate(test_set, target_col = "stroke", prediction_cols = "stroke_hat_lda",
                        type = "binomial", positive = "Yes")
plot_confusion_matrix(conf_mat)
```



8. Neural Network

```
set.seed(260)
grid <- expand.grid(decay = c(0.05, 0.1),
                        size = seq(30, 60, 10))
control <- trainControl(method = "cv", number = 5)

train_nnet <- train(stroke ~ ., data = train_set,
                    method = "nnet",
                    trControl = control,
                    maxit = 1000, # maximum iteration
                    preProcess = c('center', 'scale'), # standardize data
                    trace = FALSE, # hide the training trace
                    tuneGrid = grid)
train_nnet
```

```
## Neural Network
##
## 7776 samples
##   10 predictor
##    2 classes: 'No', 'Yes'
##
## Pre-processing: centered (14), scaled (14)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 6220, 6222, 6221, 6221, 6220
## Resampling results across tuning parameters:
##
##   decay  size  Accuracy   Kappa
##   0.05   30    0.9014935  0.8029875
##   0.05   40    0.9124222  0.8248445
##   0.05   50    0.9104952  0.8209906
##   0.05   60    0.9110081  0.8220164
##   0.10   30    0.9036791  0.8073574
##   0.10   40    0.9110069  0.8220139
##   0.10   50    0.9140947  0.8281895
##   0.10   60    0.9139672  0.8279344
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 50 and decay = 0.1.
```

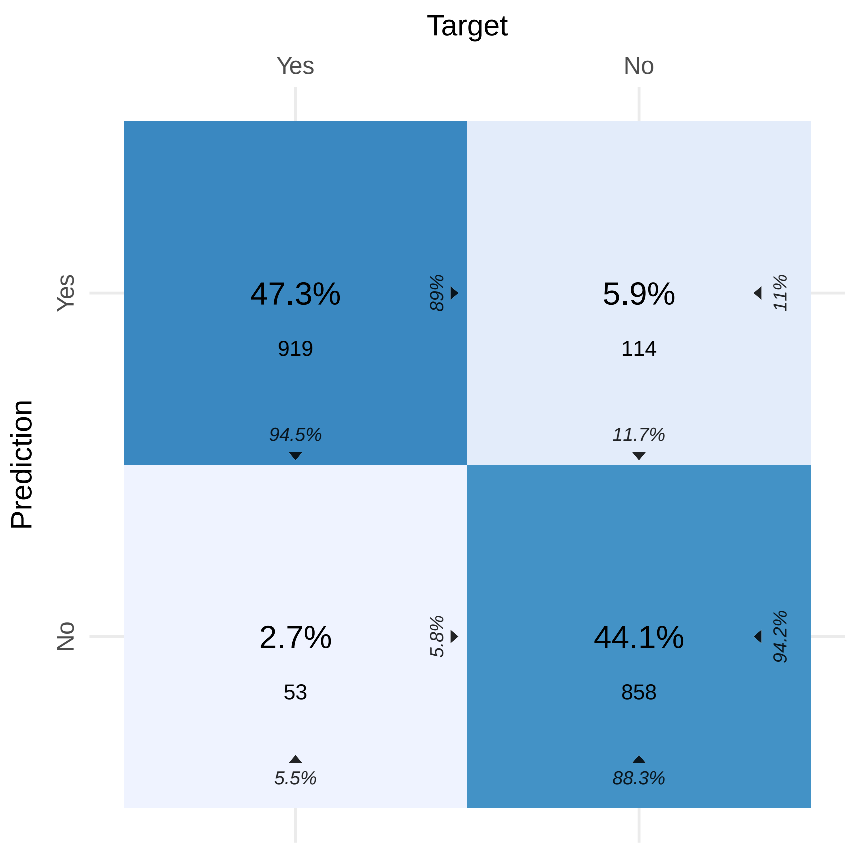
predict

```
test_set$stroke_hat_nnet <- predict(train_nnet, test_set)
cm <- confusionMatrix(test_set$stroke_hat_nnet, test_set$stroke)
cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.9140947
```

Confusion matrix plot

```
test_set$stroke_hat_nnet <- as.character(test_set$stroke_hat_nnet)
conf_mat <- evaluate(test_set, target_col = "stroke", prediction_cols = "stroke_hat_nnet",
                        type = "binomial", positive = "Yes")
plot_confusion_matrix(conf_mat)
```


Target
Yes
No
Prediction
Yes
No
47.3%
919
89%
94.5%
5.9%
114
11%
11.7%
2.7%
53
5.8%
5.5%
44.1%
858
94.2%
88.3%