Final

Zhang_Lugg

```r
library(r02pro)     #INSTALL IF NECESSARY
library(tidyverse)  #INSTALL IF NECESSARY
library(MASS)
library(naniar)
library(caret)
library(glmnet)
library(plotmo)
library(pROC)
library(e1071)
library(randomForest)
library(table1)
rm (list = ls()) # Clearing the memory
setwd("Y:/NYU/MLPH")
```

```r
stroke <- read.csv("healthcare-dataset-stroke-data.csv") |>
  replace_with_na_all(~. %in% c("","N/A")) |>
  na.omit()
stroke$bmi <- as.numeric(stroke$bmi)

# Set seed
set.seed(0923)
# Set the size if training and test data, here is 1:1
split_ratio <- 0.5
# Generate random number and index
indices <- sample(1:nrow(stroke), size = round(split_ratio * nrow(stroke)), replace = FALSE)
stroke.training <- stroke[indices, ]
stroke.test <- stroke[-indices, ]
str(stroke.training)
```

```
## tibble [2,454 x 12] (S3: tbl_df/tbl/data.frame)
##  $ id               : int [1:2454] 24245 49277 16938 72911 50522 40242 1192 58567 17277 64817 ...
##  $ gender           : chr [1:2454] "Male" "Female" "Female" "Female" ...
##  $ age              : num [1:2454] 55 34 40 57 72 5 31 42 4 39 ...
##  $ hypertension     : int [1:2454] 0 0 0 1 0 0 0 0 0 0 ...
##  $ heart_disease    : int [1:2454] 0 0 0 0 0 0 0 0 0 0 ...
##  $ ever_married     : chr [1:2454] "Yes" "No" "Yes" "Yes" ...
##  $ work_type        : chr [1:2454] "Private" "Private" "Self-employed" "Private" ...
##  $ Residence_type   : chr [1:2454] "Urban" "Urban" "Rural" "Rural" ...
##  $ avg_glucose_level: num [1:2454] 91 70.9 213 129.5 131.4 ...
##  $ bmi              : num [1:2454] 32.1 55.7 49.8 60.9 28.4 16.3 27.2 22.8 22 34.3 ...
##  $ smoking_status   : chr [1:2454] "Unknown" "formerly smoked" "formerly smoked" "smokes" ...
##  $ stroke           : int [1:2454] 0 0 0 0 1 0 0 0 0 0 ...
##  - attr(*, "na.action")= 'omit' Named int [1:201] 2 9 14 20 28 30 44 47 51 52 ...
##   ..- attr(*, "names")= chr [1:201] "2" "9" "14" "20" ...
```

```r
table(stroke.training$stroke)
```

```
##
##    0    1
## 2359   95
```

```r
str(stroke.test)
```

```
## tibble [2,455 x 12] (S3: tbl_df/tbl/data.frame)
##  $ id               : int [1:2455] 1665 53882 10434 60491 12109 12095 12175 5317 27458 38047 ...
##  $ gender           : chr [1:2455] "Female" "Male" "Female" "Female" ...
##  $ age              : num [1:2455] 79 74 69 78 81 61 54 79 60 65 ...
##  $ hypertension     : int [1:2455] 1 1 0 0 1 0 0 0 0 0 ...
##  $ heart_disease    : int [1:2455] 0 1 0 0 0 1 0 1 0 0 ...
##  $ ever_married     : chr [1:2455] "Yes" "Yes" "No" "Yes" ...
##  $ work_type        : chr [1:2455] "Self-employed" "Private" "Private" "Private" ...
##  $ Residence_type   : chr [1:2455] "Rural" "Rural" "Urban" "Urban" ...
##  $ avg_glucose_level: num [1:2455] 174.1 70.1 94.4 58.6 80.4 ...
##  $ bmi              : num [1:2455] 24 27.4 22.8 24.2 29.7 36.8 27.3 28.2 37.8 28.2 ...
##  $ smoking_status   : chr [1:2455] "never smoked" "never smoked" "never smoked" "Unknown" ...
##  $ stroke           : int [1:2455] 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "na.action")= 'omit' Named int [1:201] 2 9 14 20 28 30 44 47 51 52 ...
##   ..- attr(*, "names")= chr [1:201] "2" "9" "14" "20" ...
```

```r
table(stroke.test$stroke)
```

```
##
##    0    1
## 2341  114
```

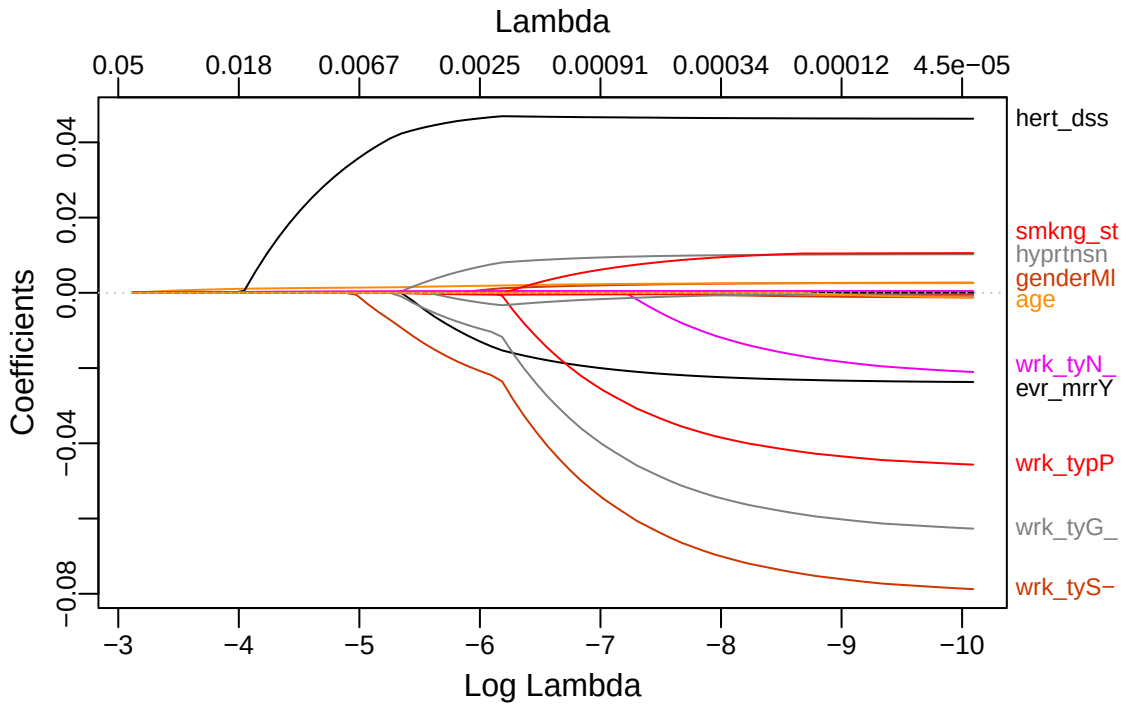
```r
t <- stroke
t$gender <- factor(t$gender)
t$hypertension <-
  factor(t$hypertension,
         levels = c(0,1),
         labels = c("No","Yes"))
t$heart_disease <-
  factor(t$heart_disease,
         levels = c(0,1),
         labels = c("No","Yes"))
t$work_type <-
  factor(t$work_type,
         levels = c("children","Govt_job","Never_worked",
                    "Private","Self-employed"),
         labels = c("Children","Govenment Job","Never Worked",
                    "Private","Self-employed"))
t$stroke <-
  factor(t$stroke,
         levels = c(1,0),
         labels = c("Stroke Patient","Not Stroke"))


column_labels <- c("ID","Gender","Age","Hypertension",
                   "Heart disease","Ever married","Work Type",
                   "Residence Type","Average Glucose Level",
                   "BMI(Body Mass Index)","Smoking Status","Stroke")

for (i in seq_along(t)) {
  attr(t[[i]], "label") <- column_labels[i]
}


pvalue <- function(x, ...) {
    # Construct vectors of data y, and groups (strata) g
    y <- unlist(x)
    g <- factor(rep(1:length(x), times=sapply(x, length)))
    if (is.numeric(y)) {
        # For numeric variables, perform a standard 2-sample t-test
        p <- t.test(y ~ g)$p.value
    } else {
        # For categorical variables, perform a chi-squared test of independence
        p <- wilcox.test(table(y, g))$p.value
    }
    # Format the p-value, using an HTML entity for the less-than sign.
    # The initial empty string places the output on the line below the variable label.
    c("", sub("<", "&lt;", format.pval(p, digits=3, eps=0.001)))
}
table1(~.|stroke,t,
```

```
       extra.col=list(`P-value`=pvalue),
       overall = F)
```

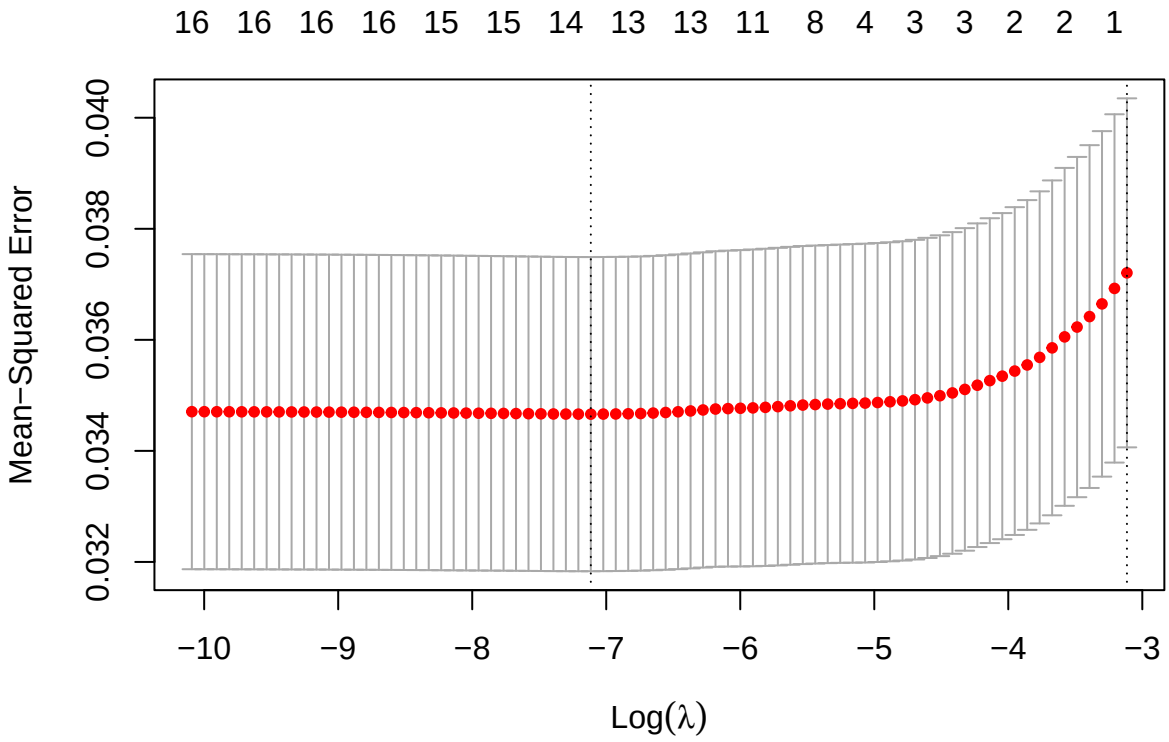
```
## Warning in wilcox.test.default(table(y, g)): cannot compute exact p-value with
## zeroes

## Warning in wilcox.test.default(table(y, g)): cannot compute exact p-value with
## zeroes

## Get nicer `table1` LaTeX output by simply installing the `kableExtra` package
```

	Stroke Patient	Not Stroke	P-value
	(N=209)	(N=4700)	
ID			
Mean (SD)	37500 (22100)	37000 (20900)	0.747
Median [Min, Max]	36900 [210, 72900]	37600 [77.0, 72900]	
Gender			
Female	120 (57.4%)	2777 (59.1%)	0.0591
Male	89 (42.6%)	1922 (40.9%)	
Other	0 (0%)	1 (0.0%)	
Age			
Mean (SD)	67.7 (12.4)	41.8 (22.3)	<0.001
Median [Min, Max]	70.0 [14.0, 82.0]	43.0 [0.0800, 82.0]	
Hypertension			
No	149 (71.3%)	4309 (91.7%)	0.125
Yes	60 (28.7%)	391 (8.3%)	
Heart disease			
No	169 (80.9%)	4497 (95.7%)	0.125
Yes	40 (19.1%)	203 (4.3%)	
Ever married			
No	23 (11.0%)	1682 (35.8%)	0.125
Yes	186 (89.0%)	3018 (64.2%)	
Work Type			
Children	1 (0.5%)	670 (14.3%)	0.00915
Govenment Job	28 (13.4%)	602 (12.8%)	
Never Worked	0 (0%)	22 (0.5%)	
Private	127 (60.8%)	2684 (57.1%)	
Self-employed	53 (25.4%)	722 (15.4%)	
Residence Type			
Rural	100 (47.8%)	2319 (49.3%)	0.125
Urban	109 (52.2%)	2381 (50.7%)	
Average Glucose Level			
Mean (SD)	135 (62.5)	104 (43.0)	<0.001
Median [Min, Max]	107 [56.1, 272]	91.2 [55.1, 268]	
BMI(Body Mass Index)			
Mean (SD)	30.5 (6.33)	28.8 (7.91)	<0.001
Median [Min, Max]	29.7 [16.9, 56.6]	28.0 [10.3, 97.6]	
Smoking Status			
formerly smoked	57 (27.3%)	780 (16.6%)	0.00781
never smoked	84 (40.2%)	1768 (37.6%)	
smokes	39 (18.7%)	698 (14.9%)	
Unknown	29 (13.9%)	1454 (30.9%)	

```r
x <- model.matrix(stroke ~ ., data = stroke.training)[, -1]  # Exclude intercept
y <- stroke.training$stroke

lasso_model <- glmnet(x,y,alpha = 1)
plot_glmnet(lasso_model)
```




```r
lasso_model <- cv.glmnet(x, y, alpha = 1)
plot(lasso_model)
```




```
opt_lambda <- lasso_model$lambda.min  # or cv$lambda.1se for the 1 standard error rule
```

```r
coef(lasso_model, s = opt_lambda)
```

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                                        s1
## (Intercept)                -6.572814e-02
## id                          1.023841e-07
## genderMale                  2.059242e-03
## age                         2.334119e-03
## hypertension                9.516878e-03
## heart_disease               4.664113e-02
## ever_marriedYes            -2.042050e-02
## work_typeGovt_job          -4.227642e-02
## work_typeNever_worked       .
## work_typePrivate           -2.757998e-02
## work_typeSelf-employed     -5.671392e-02
## Residence_typeUrban         .
## avg_glucose_level           4.781311e-04
## bmi                        -4.054594e-04
## smoking_statusnever smoked -1.535199e-03
## smoking_statussmokes        6.704251e-03
## smoking_statusUnknown       .
```

```r
lm <- lm(stroke~.-id ,stroke.training)
summary(lm)
```

```
##
## Call:
## lm(formula = stroke ~ . - id, data = stroke.training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.25036 -0.06720 -0.02009  0.00881  0.99483
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -5.589e-02  2.144e-02  -2.607  0.00918 **
## genderMale                  2.597e-03  7.751e-03   0.335  0.73756
## age                         2.604e-03  2.814e-04   9.253  < 2e-16 ***
## hypertension                1.030e-02  1.379e-02   0.747  0.45512
## heart_disease               4.639e-02  1.755e-02   2.643  0.00826 **
## ever_marriedYes            -2.378e-02  1.119e-02  -2.125  0.03366 *
## work_typeGovt_job          -6.398e-02  1.991e-02  -3.214  0.00133 **
## work_typeNever_worked      -2.080e-02  5.998e-02  -0.347  0.72874
## work_typePrivate           -4.686e-02  1.674e-02  -2.799  0.00516 **
## work_typeSelf-employed     -8.052e-02  2.019e-02  -3.987 6.89e-05 ***
## Residence_typeUrban        -1.214e-03  7.505e-03  -0.162  0.87155
## avg_glucose_level           4.803e-04  8.683e-05   5.532 3.50e-08 ***
## bmi                        -3.345e-04  5.508e-04  -0.607  0.54379
## smoking_statusnever smoked -8.369e-04  1.099e-02  -0.076  0.93930
## smoking_statussmokes        1.068e-02  1.312e-02   0.814  0.41594
## smoking_statusUnknown      -1.556e-03  1.238e-02  -0.126  0.90001
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.1852 on 2438 degrees of freedom
## Multiple R-squared:  0.08415,    Adjusted R-squared:  0.07851
## F-statistic: 14.93 on 15 and 2438 DF,  p-value: < 2.2e-16
```

```
# Define the formula
formula <- stroke ~ age + heart_disease + avg_glucose_level + hypertension + work_type

#### Logistic
logi.fit <- glm(formula, data = stroke.training, family='binomial')
# summary(logi.fit)
pred_train_prob <- predict(logi.fit,newdata = stroke.test, type = 'response')
#define object to plot
rocobj_logi <- roc(stroke.test$stroke, pred_train_prob)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
#create ROC plot
auc_logi <- auc(rocobj_logi)

#### LDA
lda.fit <- lda(formula, data = stroke.training)
lda.pred <- predict(lda.fit,newdata = stroke.test)$posterior[, 2]
rocobj_lda <- roc(stroke.test$stroke, lda.pred)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
auc_lda <- auc(rocobj_lda)

#### KNN
knn.fit <- knn3(formula, stroke.training, k = 7, prob = TRUE)
knn.pred <- predict(knn.fit, newdata = stroke.test, type = "prob")
rocobj_knn <- roc(stroke.test$stroke, knn.pred[ ,2])
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
auc_knn <- auc(rocobj_knn)

#### SVM
svm.fit <- svm(formula, data = stroke.training, kernel = "radial",probability = TRUE)
svm.pred <- predict(svm.fit, newdata = stroke.test, probability = TRUE)
rocobj_svm <- roc(stroke.test$stroke, svm.pred)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

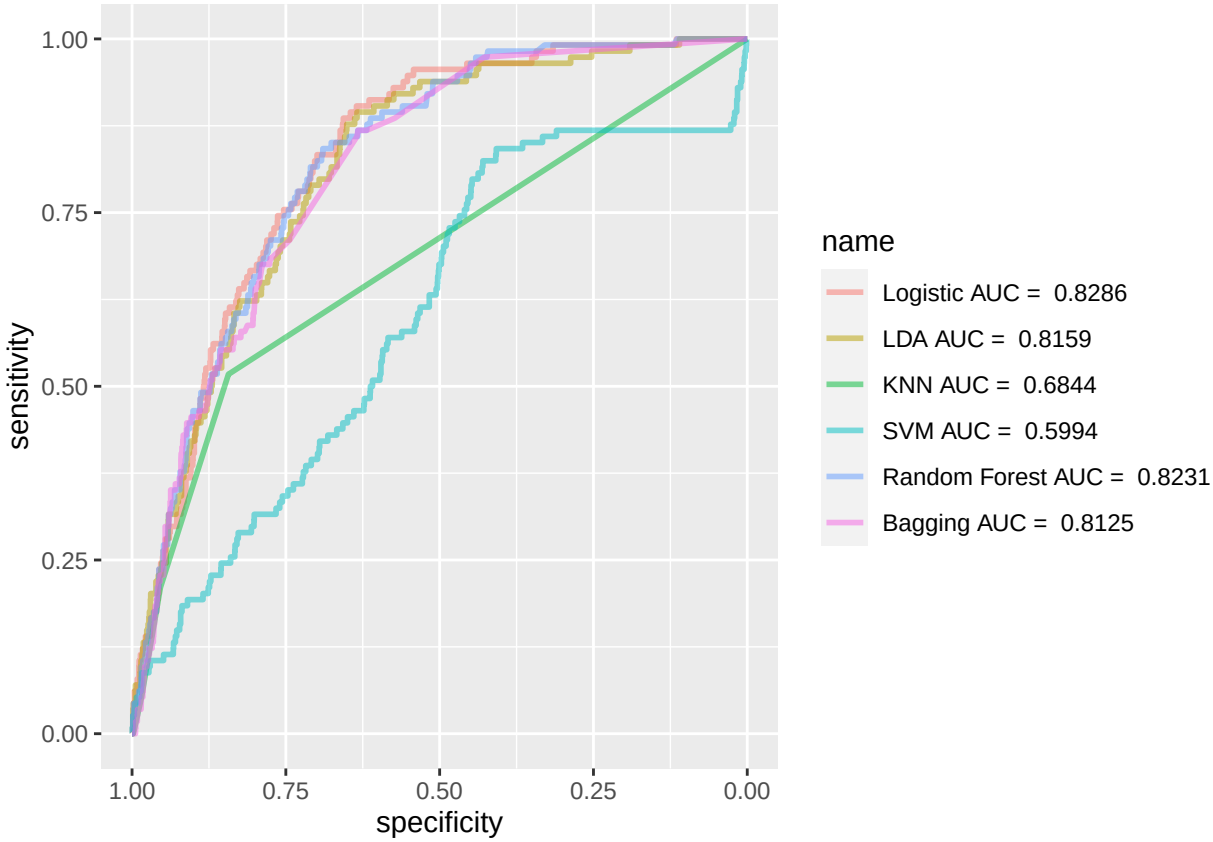
```
auc_svm <- auc(rocobj_svm)

#### Random Forest
rf.fit <- randomForest(formula, data = stroke.training)
rf.pred <- predict(rf.fit, newdata = stroke.test)
rocobj_rf <- roc(stroke.test$stroke, rf.pred)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
auc_rf <- auc(rocobj_rf)

#### Bagging
bg.fit <- train(formula, data = stroke.training, method = "treebag")
bg.pred <- predict(bg.fit, newdata = stroke.test)
rocobj_bg <- roc(stroke.test$stroke, bg.pred)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
auc_bg <- auc(rocobj_bg)
#### plot
rocobjs <- list(Logistic = rocobj_logi, LDA = rocobj_lda,KNN = rocobj_knn, SVM = rocobj_svm, RandomFores
methods_auc <- paste(c("Logistic", "LDA","KNN","SVM","Random Forest","Bagging"),
                     "AUC = ",
                     round(c(auc_logi, auc_lda, auc_knn,auc_svm,auc_rf,auc_bg),4))
ggroc(rocobjs, size = 1, alpha = 0.5) +
scale_color_discrete(labels = methods_auc)
```




```r
# Predicted probabilities for each model
logi_pred <- predict(logi.fit, newdata = stroke.training, type = "response")
lda_pred <- predict(lda.fit, newdata = stroke.training)$posterior[, 2]
knn_pred <- predict(knn.fit, newdata = stroke.training, type = "prob")
svm_pred <- predict(svm.fit, newdata = stroke.training, probability = TRUE)
rf_pred <- predict(rf.fit, newdata = stroke.training)
bg_pred <- predict(bg.fit, newdata = stroke.training)
```

```r
# Convert predicted probabilities to class labels
logi_class <- ifelse(logi_pred > 0.5, 1, 0)  # Logistic Regression
lda_class <- ifelse(lda_pred > 0.5, 1, 0)    # LDA
knn_class <- knn_pred[,2]                    # KNN
svm_class <- ifelse(svm_pred > 0.5, 1, 0)  # SVM
rf_class <- ifelse(rf_pred > 0.5, 1, 0)                          # Random Forest
bg_class <- ifelse(bg_pred > 0.5, 1, 0)                         # Bagging

# True class labels
true_class <- stroke.training$stroke

# Calculate accuracy for each model
train_error_logi <- mean(logi_class!=true_class)  # Logistic Regression
train_error_lda <- mean(lda_class!=true_class)    # LDA
train_error_knn <- mean(knn_class!=true_class)    # KNN
train_error_svm <- mean(svm_class!=true_class)    # SVM
train_error_rf <- mean(rf_class!=true_class)      # Random Forest
train_error_bg <- mean(bg_class!=true_class)      # Bagging
```

```r
# Predicted probabilities for each model
logi_pred <- predict(logi.fit, newdata = stroke.test, type = "response")
lda_pred <- predict(lda.fit, newdata = stroke.test)$posterior[, 2]
knn_pred <- predict(knn.fit, newdata = stroke.test)
svm_pred <- predict(svm.fit, newdata = stroke.test, probability = TRUE)
rf_pred <- predict(rf.fit, newdata = stroke.test)
bg_pred <- predict(bg.fit, newdata = stroke.test)

# Convert predicted probabilities to class labels
logi_class <- ifelse(logi_pred > 0.5, 1, 0)  # Logistic Regression
lda_class <- ifelse(lda_pred > 0.5, 1, 0)    # LDA
knn_class <- knn_pred[,2]                    # KNN
svm_class <- ifelse(svm_pred > 0.5, 1, 0)  # SVM
rf_class <- ifelse(rf_pred > 0.5, 1, 0)                          # Random Forest
bg_class <- ifelse(bg_pred > 0.5, 1, 0)                         # Bagging

# True class labels
true_class <- stroke.test$stroke

# Calculate accuracy for each model
accuracy_logi <- mean(logi_class == true_class)  # Logistic Regression
test_error_logi <- 1 - accuracy_logi
accuracy_lda <- mean(lda_class == true_class)    # LDA
test_error_lda <- 1 - accuracy_lda
accuracy_knn <- mean(knn_class == true_class)    # KNN
test_error_knn <- 1 - accuracy_knn
accuracy_svm <- mean(svm_class == true_class)    # SVM
test_error_svm <- 1 - accuracy_svm
accuracy_rf <- mean(rf_class == true_class)      # Random Forest
test_error_rf <- 1 - accuracy_rf
accuracy_bg <- mean(bg_class == true_class)      # Bagging
```

```r
test_error_bg <- 1 - accuracy_bg
```

```r
# Display Results
cat("Logistic Regression:\n",
    "Training Error:", train_error_logi, "\n",
    "Test Error:", test_error_logi, "\n\n")
```

```
## Logistic Regression:
##  Training Error: 0.03830481
##  Test Error: 0.04643585
```

```r
cat("LDA:\n",
    "Training Error:", train_error_lda, "\n",
    "Test Error:", test_error_lda, "\n\n")
```

```
## LDA:
##  Training Error: 0.04278729
##  Test Error: 0.05173116
```

```r
cat("KNN:\n",
    "Training Error:", train_error_knn, "\n",
    "Test Error:", test_error_knn, "\n\n")
```

```
## KNN:
##  Training Error: 0.1919315
##  Test Error: 0.196334
```

```r
cat("SVM:\n",
    "Training Error:", train_error_svm, "\n",
    "Test Error:", test_error_svm, "\n\n")
```

```
## SVM:
##  Training Error: 0.03871231
##  Test Error: 0.04643585
```

```r
cat("Random Forest:\n",
    "Training Error:", train_error_rf, "\n",
    "Test Error:", test_error_rf, "\n\n")
```

```
## Random Forest:
##  Training Error: 0.03871231
##  Test Error: 0.04643585
```

```r
cat("Bagging:\n",
    "Training Error:", train_error_bg, "\n",
    "Test Error:", test_error_bg, "\n")
```

```
## Bagging:
##  Training Error: 0.03789731
##  Test Error: 0.04725051
```

```r
# Sensitivity (True Positive Rate)
sensitivity_logi <- sum(logi_class == 1 & true_class == 1) / sum(true_class == 1)
sensitivity_lda <- sum(lda_class == 1 & true_class == 1) / sum(true_class == 1)
sensitivity_knn <- sum(knn_class == 1 & true_class == 1) / sum(true_class == 1)
sensitivity_svm <- sum(svm_class == 1 & true_class == 1) / sum(true_class == 1)
sensitivity_rf <- sum(rf_class == 1 & true_class == 1) / sum(true_class == 1)
sensitivity_bg <- sum(bg_class == 1 & true_class == 1) / sum(true_class == 1)
```

```r
# Specificity (True Negative Rate)
specificity_logi <- sum(logi_class == 0 & true_class == 0) / sum(true_class == 0)
specificity_lda <- sum(lda_class == 0 & true_class == 0) / sum(true_class == 0)
specificity_knn <- sum(knn_class == 0 & true_class == 0) / sum(true_class == 0)
specificity_svm <- sum(svm_class == 0 & true_class == 0) / sum(true_class == 0)
specificity_rf <- sum(rf_class == 0 & true_class == 0) / sum(true_class == 0)
specificity_bg <- sum(bg_class == 0 & true_class == 0) / sum(true_class == 0)

cat("Logistic Regression:\n",
    "Sensitivity:", sensitivity_logi, "\n",
    "Specificity:", specificity_logi, "\n\n")
```

```
## Logistic Regression:
##  Sensitivity: 0
##  Specificity: 1
```

```r
cat("LDA:\n",
    "Sensitivity:", sensitivity_lda, "\n",
    "Specificity:", specificity_lda, "\n\n")
```

```
## LDA:
##  Sensitivity: 0.07017544
##  Specificity: 0.9910295
```

```r
cat("KNN:\n",
    "Sensitivity:", sensitivity_knn, "\n",
    "Specificity:", specificity_knn, "\n\n")
```

```
## KNN:
##  Sensitivity: 0
##  Specificity: 0.8428022
```

```r
cat("SVM:\n",
    "Sensitivity:", sensitivity_svm, "\n",
    "Specificity:", specificity_svm, "\n\n")
```

```
## SVM:
##  Sensitivity: 0
##  Specificity: 1
```

```r
cat("Random Forest:\n",
    "Sensitivity:", sensitivity_rf, "\n",
    "Specificity:", specificity_rf, "\n\n")
```

```
## Random Forest:
##  Sensitivity: 0
##  Specificity: 1
```

```r
cat("Bagging:\n",
    "Sensitivity:", sensitivity_bg, "\n",
    "Specificity:", specificity_bg, "\n")
```

```
## Bagging:
##  Sensitivity: 0
##  Specificity: 0.9991457
```

```
rbind(
  c("Logistic Regression",train_error_logi,test_error_logi,accuracy_logi,sensitivity_logi,specificity_l
  c("LDA",train_error_lda,test_error_lda,accuracy_lda,sensitivity_lda,specificity_lda),
  c("KNN",train_error_knn,test_error_knn,accuracy_knn,sensitivity_knn,specificity_knn),
  c("SVM",train_error_svm,test_error_svm,accuracy_svm,sensitivity_svm,specificity_svm),
  c("Random Forest",train_error_rf,test_error_rf,accuracy_rf,sensitivity_rf,specificity_rf),
  c("Bagging",train_error_bg,test_error_bg,accuracy_bg,sensitivity_bg,specificity_bg)
) |>
  data.frame() -> tab
colnames(tab) <- c("Model","Train Error","Test Error","Accuracy","Sensitivity","Specificity")
tab |>
  mutate_at(vars(-Model), as.numeric) %>%
  mutate_at(vars(-Model), ~ round(., 4))
```

```
p <- ncol(stroke.training)-2
bag <- randomForest(formula, stroke.training, mtry = p, importance=TRUE)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values.  Are you sure you want to do regression?
```

```
## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range
```

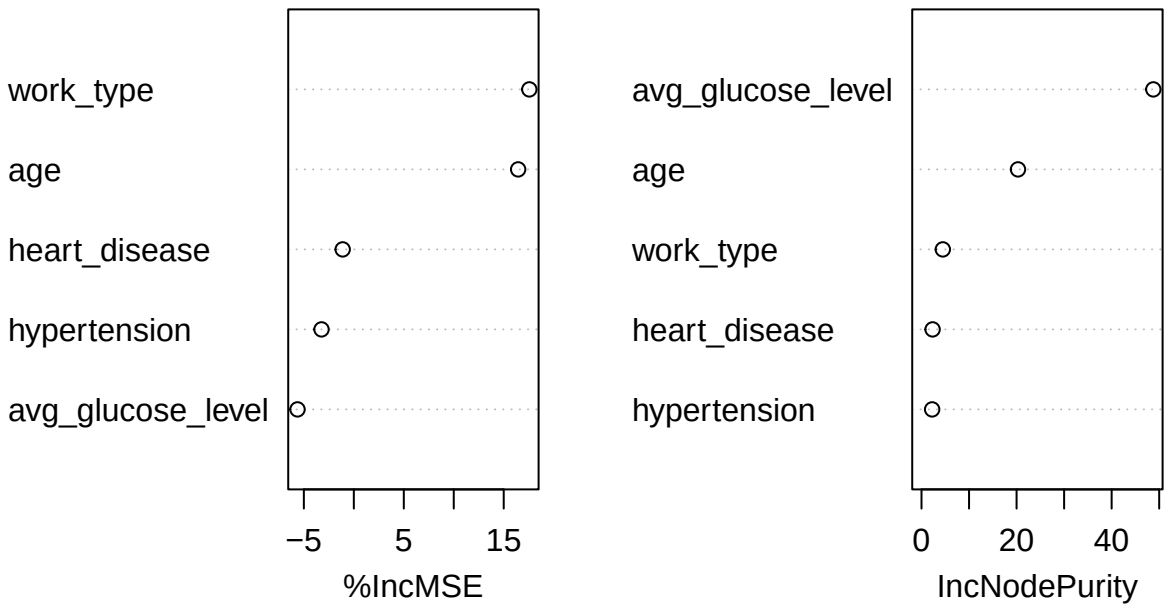
```
bag
```

```
##
## Call:
##  randomForest(formula = formula, data = stroke.training, mtry = p,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 0.03844101
##                     % Var explained: -3.3
```

```
importance(bag)
```

```
##                     %IncMSE IncNodePurity
## age               16.443210     20.296815
## heart_disease     -1.114219      2.331923
## avg_glucose_level -5.636165     48.767344
## hypertension      -3.233440      2.236379
## work_type         17.563035      4.479784
```

```
varImpPlot(bag)
```


bag
work_type
age
heart_disease
hypertension
avg_glucose_level
−5
5
15
%IncMSE
avg_glucose_level
age
work_type
heart_disease
hypertension
0
20
40
IncNodePurity