Final Project

Ethan Lee - 505997405

2024-12-02

```
Sys.setlocale("LC_ALL", "C")
Sys.setenv(LANG="en")
library(ggplot2)
library(class)
library(boot)
library(crossval)
library(MASS)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:boot':
##
##     melanoma
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:crossval':
##
##     confusionMatrix
```

#Q3) Download the training and the testing data sets.

```
ObesityTr <- read.csv("/Users/ethanlee/Desktop/STATS 101C/ObesityTrain2.csv", header = TRUE)
ObesityTs <- read.csv("/Users/ethanlee/Desktop/STATS 101C/ObesityTestNoY2.csv", header = TRUE)
```

##a) Report the dimensions of both the training and the testing data sets.

```
cat("The dimensions of training data set: ", dim(ObesityTr), "\n")
```

```
## The dimensions of training data set:  32014 30
```

```
cat("The dimensions of testing data set: ", dim(ObesityTs))
```

```
## The dimensions of testing data set:  10672 29
```

##b) How many numerical predictors does your data have? List them.

```
numerical_columns <- sapply(ObesityTr, is.numeric)
numerical_predictors <- names(ObesityTr)[numerical_columns]
num_numerical_predictors <- length(numerical_predictors)
cat("The  data has ", num_numerical_predictors, "numerical predictors. They are: ", paste(strsplit(nume
```

```
## The  data has  11 numerical predictors. They are:  Age , Height , FCVC , NCP , CH2O , FAF , TUE , Re
```

##c) How many categorical predictors does your data have? List them.

```
categorical_predictors <- names(ObesityTr[, !(names(ObesityTr) %in% numerical_predictors)])
categorical_predictors <- categorical_predictors[-length(categorical_predictors)]
num_categorical_predictors <- length(categorical_predictors)
cat("The  data has ", num_categorical_predictors, "categorical predictors. They are: ", paste(strsplit(
```

```
## The  data has  18 categorical predictors. They are:  Gender , family_history_with_overweight , FAVC
```

##d) Report the size of missing values (frequency or percentage or both) in both data sets (Training and Testing).

```
#function
calculate_missing_values <- function(data){
  total_missing = sum(is.na(data))
  missing_percentage <- sum(is.na(data)) / prod(dim(data)) *100
  column_missing_freq <- colSums(is.na(data))
  column_missing_percentage <- (column_missing_freq / nrow(data)) * 100
  result <- list(total_missing = total_missing,
    missing_percentage = missing_percentage,
    col_missing = cbind(column_missing_freq, column_missing_percentage)
  )
  return(result)
}
print(calculate_missing_values(ObesityTr))
```

```
## $total_missing
## [1] 74272
##
## $missing_percentage
## [1] 7.733283
##
## $col_missing
##                                 column_missing_freq column_missing_percentage
## Age                                            2520                 7.871556
## Gender                                         2572                 8.033985
## Height                                         2538                 7.927782
## family_history_with_overweight                 2547                 7.955894
## FAVC                                           2579                 8.055851
## FCVC                                           2574                 8.040232
## NCP                                            2583                 8.068345
## CAEC                                           2537                 7.924658
## SMOKE                                          2609                 8.149560
## CH2O                                           2549                 7.962142
## SCC                                            2551                 7.968389
## FAF                                            2560                 7.996502
## TUE                                            2592                 8.096458
## CALC                                           2614                 8.165178
## MTRANS                                         2490                 7.777847
## Race                                           2588                 8.083963
## RestingBP                                      2509                 7.837196
## Cholesterol                                    2556                 7.984007
## FastingBS                                      2556                 7.984007
## RestingECG                                     2617                 8.174549
## MaxHR                                          2586                 8.077716
## ExerciseAngina                                 2527                 7.893422
```

```
## HeartDisease                                2596                 8.108952
## hypertension                                2615                 8.168301
## ever_married                                2538                 7.927782
## work_type                                   2667                 8.330730
## Residence_type                              2550                 7.965265
## avg_glucose_level                           2500                 7.809084
## stroke                                      2452                 7.659149
## ObStatus                                       0                 0.000000
```

```
print(calculate_missing_values(ObesityTs))
```

```
## $total_missing
## [1] 24759
##
## $missing_percentage
## [1] 7.999987
##
## $col_missing
##                                 column_missing_freq column_missing_percentage
## Age                                             860                 8.058471
## Gender                                          885                 8.292729
## Height                                          859                 8.049100
## family_history_with_overweight                  849                 7.955397
## FAVC                                            854                 8.002249
## FCVC                                            850                 7.964768
## NCP                                             828                 7.758621
## CAEC                                            872                 8.170915
## SMOKE                                           874                 8.189655
## CH2O                                            855                 8.011619
## SCC                                             834                 7.814843
## FAF                                             800                 7.496252
## TUE                                             840                 7.871064
## CALC                                            846                 7.927286
## MTRANS                                          888                 8.320840
## Race                                            818                 7.664918
## RestingBP                                       856                 8.020990
## Cholesterol                                     826                 7.739880
## FastingBS                                       863                 8.086582
## RestingECG                                      893                 8.367691
## MaxHR                                           858                 8.039730
## ExerciseAngina                                  845                 7.917916
## HeartDisease                                    846                 7.927286
## hypertension                                    827                 7.749250
## ever_married                                    843                 7.899175
## work_type                                       918                 8.601949
## Residence_type                                  843                 7.899175
## avg_glucose_level                               861                 8.067841
## stroke                                          868                 8.133433
```

##e) Report the frequency and proportions of the categories in your response variable (in the Training and Testing data).

```
freq_table <- table(ObesityTr$ObStatus)
prop_table <- prop.table(freq_table)
print(freq_table)
```

```
##
## Not Obese     Obese
##     19531     12483
```

```
print(prop_table)
```

```
##
## Not Obese     Obese
## 0.6100768 0.3899232
```

```
#Testing data do not have response variable
```

##f) What is your maximum error rate allowed based on your Training data?

```
max_error_rate <- sum(ObesityTr$ObStatus == "Not Obese") / nrow(ObesityTr)
cat("Maximum error rate allowed based on your Training data is: ", max_error_rate)
```

```
## Maximum error rate allowed based on your Training data is:  0.6100768
```

##g) Plot densities of your best three numerical predictors based on the response variable.

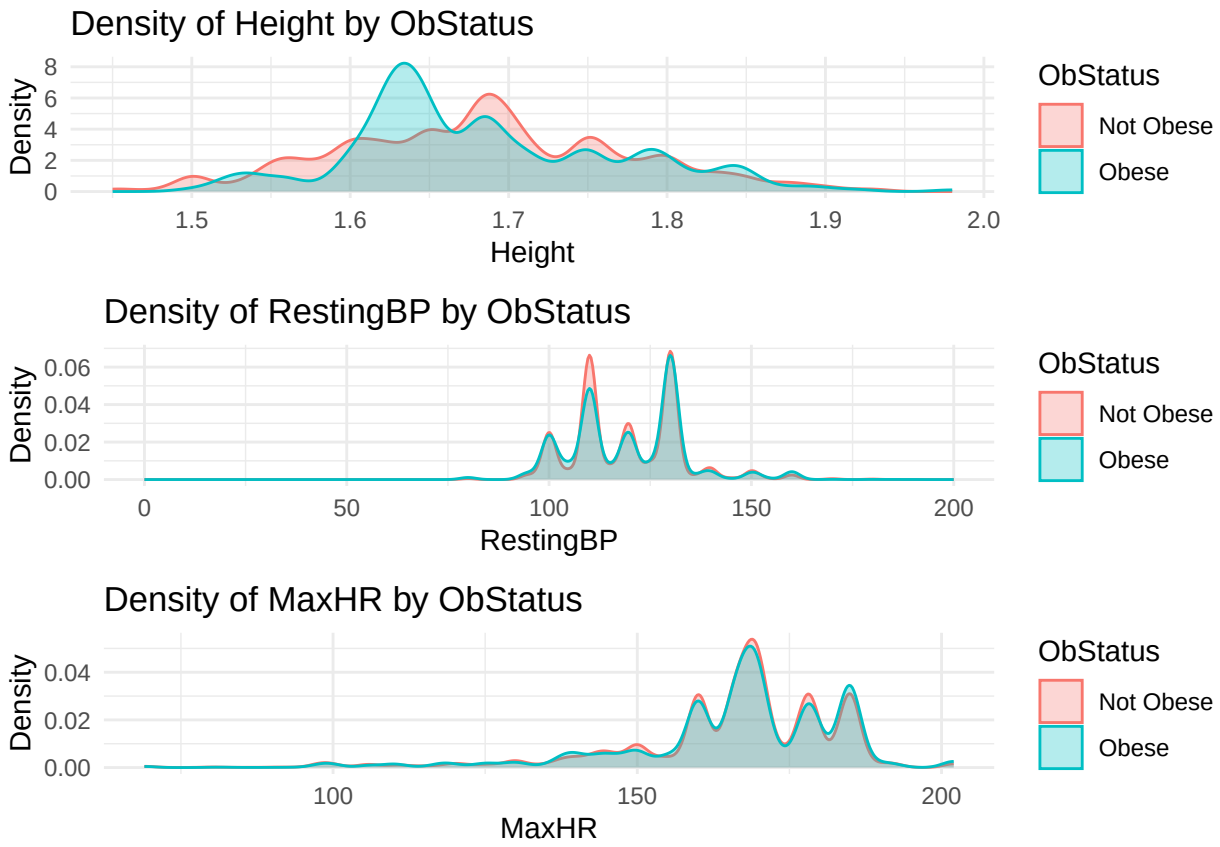
```
#deal with the missing values
for (col in names(ObesityTr)) {
  if (is.numeric(ObesityTr[[col]])) {
    # missing value = the average
    mean_value <- mean(ObesityTr[[col]], na.rm = TRUE)
    ObesityTr[[col]][is.na(ObesityTr[[col]])] <- mean_value
  }
}
#choose the best 3 numerical predictor
###correlations <- cor(ObesityTr[, numerical_columns],  as.numeric(as.factor(ObesityTr$ObStatus)), use =

p_values <- sapply(ObesityTr[, numerical_columns], function(x) {
  anova_result <- aov(x ~ ObesityTr$ObStatus)
  summary(anova_result)[[1]]["Pr(>F)"][1]
})
p_values <- as.data.frame(p_values)
top_num_predictors <- names(p_values[, order(as.matrix(p_values[1,]), decreasing = TRUE)])[1:3]
```

The best 3 numerical predictor are: Height, RestingBP, MaxHR

```
#plot densities
top_num_predictors <- c("Height", "RestingBP", "MaxHR")
plots <- lapply(top_num_predictors, function(pred) {
  ggplot(ObesityTr, aes_string(x = pred, color = "ObStatus", fill = "ObStatus")) +
    geom_density(alpha = 0.3) +
    labs(title = paste("Density of", pred, "by ObStatus"),
         x = pred, y = "Density") +
    theme_minimal()
})
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
library(gridExtra)
grid.arrange(grobs = plots, ncol = 1)
```

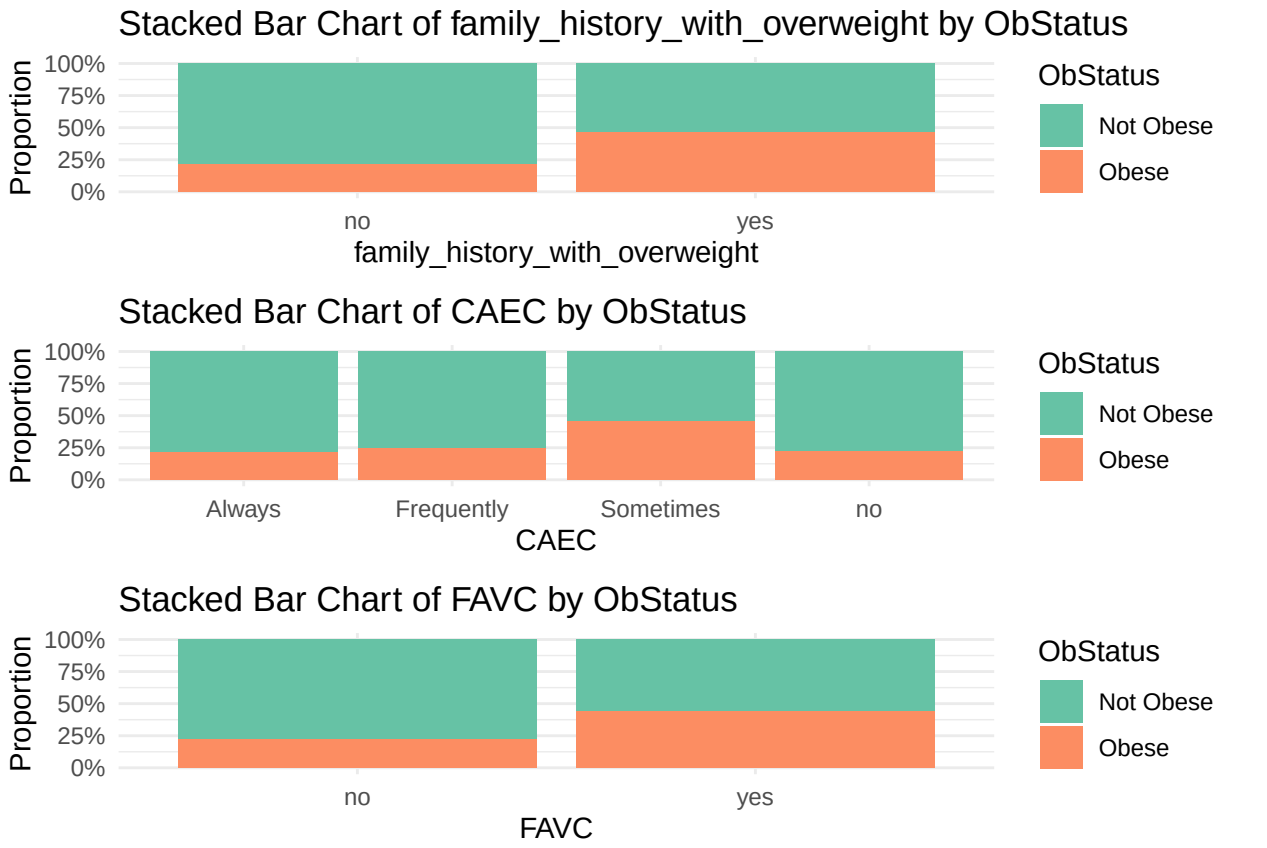


##h) Create stacked par charts for your best three categorical predictors based on your response variable.

```
#deal with the missing values
set.seed(12345)
for (col in names(ObesityTr)) {
  if (is.factor(ObesityTr[[col]]) || is.character(ObesityTr[[col]])) {
    proportions <- prop.table(table(ObesityTr[[col]], useNA = "no"))
    categories <- names(proportions)
    probs <- as.numeric(proportions)
    # randomly assign
    missing_indices <- which(is.na(ObesityTr[[col]]))
    ObesityTr[[col]][missing_indices] <- sample(categories, length(missing_indices), replace = TRUE, pr
  }
}
#choose the best categorical predictors
chisq_p_values <- sapply(categorical_predictors, function(var) {
  chisq_test <- chisq.test(table(ObesityTr[[var]], ObesityTr$ObStatus))
  return(chisq_test$p.value)
})
sorted_p_values <- sort(chisq_p_values)
best_three_cate_predictors <- names(sorted_p_values)[1:3]
cat("Best three categorical predictors based on chi-square test:", best_three_cate_predictors)
```

```
## Best three categorical predictors based on chi-square test: family_history_with_overweight CAEC FAVC
```

```
#Create stacked par charts
stacked_bar_charts <- lapply(best_three_cate_predictors, function(pred) {
  ggplot(ObesityTr, aes_string(x = pred, fill = "ObStatus")) +
    geom_bar(position = "fill") +
    labs(title = paste("Stacked Bar Chart of", pred, "by ObStatus"),
         x = pred, y = "Proportion") +
    theme_minimal() +
    scale_y_continuous(labels = scales::percent) +
    scale_fill_brewer(palette = "Set2")
})
library(gridExtra)
grid.arrange(grobs = stacked_bar_charts, ncol = 1)
```



##i) List predictors that are not in your data, but you wished they were (based on context) Sleep Quality; Income level; Education level.

#Q4) Build a classifier of your choice and predict the class of the unknown Y variable "ObStatus" in the testing data. Create a submission file (similar to the submission file example and submit your prediction on Kaggle. If you already have a group, each member must submit his/her own file.

##a) User Name of your Kaggle Account Lany Lan ##b) Report your training model (summary)

```
ObesityTr <- read.csv("/Users/ethanlee/Desktop/STATS 101C/ObesityTrain2.csv", header = TRUE)
ObesityTs <- read.csv("/Users/ethanlee/Desktop/STATS 101C/ObesityTestNoY2.csv", header = TRUE)
ObesitySampleSol <- read.csv("/Users/ethanlee/Desktop/STATS 101C/ObesitySampleSolKaggle.csv", header =
```

```
#output function
csv_output <- function(predictions,file_name){
  ID <- c(1:10672)
```

```
  test_preds <- data.frame("ID" = ID, "ObStatus" = predictions)
  write.csv(test_preds, file = as.character(file_name), row.names = FALSE)
}
```

```
#deal with the missing values
missing_fill <- function(data){
  for (col in names(data)) {
    if (is.numeric(data[[col]])) {
      # missing value = the average
      median_value <- median(data[[col]], na.rm = TRUE)
      data[[col]][is.na(data[[col]])] <- median_value
    }
    if (is.factor(data[[col]]) || is.character(data[[col]])) {
      proportions <- prop.table(table(data[[col]], useNA = "no"))
      categories <- names(proportions)
      probs <- as.numeric(proportions)
      # randomly assign
      missing_indices <- which(is.na(data[[col]]))
      data[[col]][missing_indices] <- sample(categories, length(missing_indices), replace = TRUE, prob =
    }
  }
  return(data)
}
ObesityTr <- missing_fill(ObesityTr)
ObesityTs <- missing_fill(ObesityTs)
```

```
#sclae the numerical predictors
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

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

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
train_predictors <- select(ObesityTr, where(is.numeric), -ObStatus)
test_predictors <- select(ObesityTs, where(is.numeric))

train_mean <- sapply(train_predictors, mean, na.rm = TRUE)
train_sd <- sapply(train_predictors, sd, na.rm = TRUE)

train_predictors_scaled <- as.data.frame(scale(train_predictors, center = train_mean, scale = train_sd))
test_predictors_scaled <- as.data.frame(scale(test_predictors, center = train_mean, scale = train_sd))
```

```r
ObesityTr <- ObesityTr %>%
  select(-where(is.numeric), -ObStatus) %>%
  bind_cols(train_predictors_scaled) %>%
  mutate(ObStatus = ObesityTr$ObStatus)

ObesityTs <- ObesityTs %>%
  select(-where(is.numeric)) %>%
  bind_cols(test_predictors_scaled)
```

```r
#outlier
pro_outlier <- function(data){
  for (col in names(data)) {
    Q1 <- quantile(data$col, 0.25)
    Q3 <- quantile(data$col, 0.75)
    IQR_value <- Q3 - Q1
    outliers <- which(data$col < (Q1 - 1.5 * IQR_value) | data$col >
                        (Q3 + 1.5 * IQR_value))
    data$col[outliers] <- median(data$col, na.rm = TRUE)
  }
  return(data)
}
ObesityTr <- pro_outlier(ObesityTr)
ObesityTs <- pro_outlier(ObesityTs)
```

```r
#Subset Selection
library(caret)
ObesityTr$ObStatus <- as.factor(ObesityTr$ObStatus)
#recursive feature elimination
control <- rfeControl(functions = rfFuncs, method = "cv", number = 5)
predictors <- ObesityTr[, names(ObesityTr) != "ObStatus"]
set.seed(123)
subset_selection <- rfe(
  predictors,
  ObesityTr$ObStatus,
  sizes = c(5, 10, 15, 20, 25),
  rfeControl = control
)
selected_vars <- predictors(subset_selection)
print(selected_vars)
```

```
##  [1] "Height"                          "Race"
##  [3] "Age"                             "CALC"
##  [5] "FAF"                             "CH2O"
##  [7] "MTRANS"                          "NCP"
##  [9] "FCVC"                            "family_history_with_overweight"
## [11] "CAEC"                            "TUE"
## [13] "SMOKE"                           "FAVC"
## [15] "Gender"
```

selected variables are: “Height” “Race” “Age” “CALC” “FAF” “CH2O” “MTRANS” “NCP” “FCVC” “SMOKE”

```r
#PCA
X_train <- model.matrix(ObStatus ~ ., ObesityTr)[ ,-1]
pca <- prcomp(X_train, center = TRUE, scale. = TRUE)
```

```r
num_components <- 10
X_train_pca <- pca$x[, 1:num_components]
```

```r
#Lasso Regression
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```r
X <- model.matrix(ObStatus ~ ., ObesityTr)
y <- ObesityTr$ObStatus
lasso_model <- cv.glmnet(X, y, alpha = 1, family = "binomial")
best_lambda <- lasso_model$lambda.min
print(best_lambda)
```

```
## [1] 0.0003568467
```

```r
lasso_final_model <- glmnet(X, y, alpha = 1, lambda = best_lambda, family = "binomial")
summary(lasso_final_model)
```

```
##            Length Class     Mode
## a0          1     -none-    numeric
## beta       43     dgCMatrix S4
## df          1     -none-    numeric
## dim         2     -none-    numeric
## lambda      1     -none-    numeric
## dev.ratio   1     -none-    numeric
## nulldev     1     -none-    numeric
## npasses     1     -none-    numeric
## jerr        1     -none-    numeric
## offset      1     -none-    logical
## classnames  2     -none-    character
## call        6     -none-    call
## nobs        1     -none-    numeric
```

```r
#test
lasso_train_preds <- predict(lasso_final_model, newx = X, type = "class")
lasso_train_accuracy <- mean(lasso_train_preds == y)
cat("error rate", 1 - lasso_train_accuracy, "\n")
```

```
## error rate 0.2575436
```

```r
lasso_test_preds <-predict(lasso_final_model, newx = model.matrix( ~ ., ObesityTs), , type = "class")
csv_output(lasso_test_preds, "lasso_preds.csv")
```

```r
#linear regression
logistic_model <- glm(as.factor(ObStatus) ~ ., data = ObesityTr, family = binomial)
#summary(logistic_model)

logistic_probs <- predict(logistic_model, type = "response")
logistic_preds <- ifelse(logistic_probs > 0.5, "Obese", "Not Obese")

log_table <- table(Predicted = logistic_preds, Actual = ObesityTr$ObStatus)
log_error_rate <- (log_table[1,2] + log_table[2,1]) / sum(log_table)
cat("The error rate is: ", log_error_rate)
```

```
## The error rate is:  0.258262
```

```r
#cv
set.seed(12345)
cvlogistic_model <- cv.glm(ObesityTr, logistic_model, K = 10)
cv_log_error_rate <- cv.glm(ObesityTr, logistic_model, K = 10)$delta
cat("The error rate is: ",cv_log_error_rate)
```

```
## The error rate is:  0.1814107 0.1813827
```

```r
#test
logistic_model <- glm(as.factor(ObStatus) ~ ., data = ObesityTr, family = binomial)
summary(logistic_model)
```

```
##
## Call:
## glm(formula = as.factor(ObStatus) ~ ., family = binomial, data = ObesityTr)
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      -3.7302967  1.0798288  -3.455 0.000551 ***
## GenderMale                       -0.5177403  0.0349094 -14.831  < 2e-16 ***
## family_history_with_overweightyes  0.8086727  0.0312053  25.915  < 2e-16 ***
## FAVCyes                           0.7349432  0.0358788  20.484  < 2e-16 ***
## CAECFrequently                    0.3105502  0.0637029   4.875 1.09e-06 ***
## CAECSometimes                     0.9894235  0.0577402  17.136  < 2e-16 ***
## CAECno                            0.5172771  0.1015389   5.094 3.50e-07 ***
## SMOKEyes                          0.0727949  0.0612005   1.189 0.234263
## SCCyes                           -0.5825370  0.0557244 -10.454  < 2e-16 ***
## CALCFrequently                    1.4606124  1.0754163   1.358 0.174406
## CALCSometimes                     1.7570877  1.0741530   1.636 0.101884
## CALCno                            1.8517177  1.0742155   1.724 0.084746 .
## MTRANSBike                       -0.1883487  0.1577794  -1.194 0.232577
## MTRANSMotorbike                   0.3661726  0.1078200   3.396 0.000683 ***
## MTRANSPublic_Transportation       0.3163603  0.0386471   8.186 2.70e-16 ***
## MTRANSWalking                    -0.7414331  0.0703352 -10.541  < 2e-16 ***
## RaceBlack                        -0.1128976  0.0586606  -1.925 0.054281 .
## RaceHispanic                     -0.3726480  0.0565762  -6.587 4.50e-11 ***
## RaceWhite                        -0.2639791  0.0542114  -4.869 1.12e-06 ***
## FastingBSYes                     -0.2191178  0.0575402  -3.808 0.000140 ***
## RestingECGNormal                 -0.0323790  0.0340905  -0.950 0.342217
## RestingECGST                      0.0872292  0.0484879   1.799 0.072020 .
## ExerciseAnginaY                  -0.0880781  0.0314325  -2.802 0.005077 **
## HeartDiseaseYes                  -0.0268895  0.0347766  -0.773 0.439400
## hypertensionYes                  -0.0455273  0.1049631  -0.434 0.664473
## ever_marriedYes                  -0.0240920  0.0350306  -0.688 0.491615
## work_typeNever_worked             0.1442244  0.1074109   1.343 0.179358
## work_typePrivate                 -0.1144093  0.0470388  -2.432 0.015006 *
## work_typeSelf-employed           -0.0968556  0.0776797  -1.247 0.212450
## work_typechildren                 0.0468754  0.2327604   0.201 0.840394
## Residence_typeUrban              -0.0035774  0.0263981  -0.136 0.892204
## strokeYes                        -0.4213880  0.3300913  -1.277 0.201750
## Age                               0.0045479  0.0177260   0.257 0.797512
## Height                            0.3109782  0.0178143  17.457  < 2e-16 ***
## FCVC                              0.2500583  0.0142838  17.506  < 2e-16 ***
## NCP                               0.1805836  0.0136574  13.222  < 2e-16 ***
## CH2O                              0.3566599  0.0139880  25.498  < 2e-16 ***
```

```
## FAF                              -0.4526365  0.0154109 -29.371  < 2e-16 ***
## TUE                              -0.0202815  0.0142521  -1.423 0.154723
## RestingBP                        -0.0009173  0.0134386  -0.068 0.945581
## Cholesterol                       0.0566590  0.0138475   4.092 4.28e-05 ***
## MaxHR                             0.0086577  0.0142736   0.607 0.544148
## avg_glucose_level                 0.0575352  0.0144306   3.987 6.69e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 42816  on 32013  degrees of freedom
## Residual deviance: 35157  on 31971  degrees of freedom
## AIC: 35243
##
## Number of Fisher Scoring iterations: 5
```

```
logistic_probs <- predict(logistic_model, newdata = ObesityTs)
logistic_preds <- ifelse(logistic_probs > 0.5, "Obese", "Not Obese")

csv_output(logistic_preds, "logistic_preds.csv")
```

```
#LDA
#lda_model <- lda(ObStatus ~ ., data = ObesityTr)
#lda_preds <- predict(lda_model)$class
#table(Predicted = lda_preds, Actual = ObesityTr$ObStatus)
set.seed(12345)
train_control <- trainControl(method = "cv", number = 20)
cv_lda_model <- train(ObStatus ~ .,
                      data = ObesityTr,
                      method = "lda",
                      trControl = train_control)
cv_lda_error_rate <- 1 - cv_lda_model$results$Accuracy
cat("The error rate is: ",cv_lda_error_rate)
```

```
## The error rate is:  0.2612901
```

```
summary(cv_lda_model)
```

```
##             Length Class      Mode
## prior        2     -none-     numeric
## counts       2     -none-     numeric
## means       84     -none-     numeric
## scaling     42     -none-     numeric
## lev          2     -none-     character
## svd          1     -none-     numeric
## N            1     -none-     numeric
## call         3     -none-     call
## xNames      42     -none-     character
## problemType  1     -none-     character
## tuneValue    1     data.frame list
## obsLevels    2     -none-     character
## param        0     -none-     list
```

```
#test
lda_preds <- predict(cv_lda_model, newdata = ObesityTs)
```

```r
csv_output(lda_preds, "lda_preds.csv")
```

```r
#KNN
set.seed(12345)
train_control <- trainControl(method = "cv", number = 5)
cv_knn_model <- train(ObStatus ~ .,
                      data = ObesityTr,
                      method = "knn",
                       tuneGrid = expand.grid(k = 25),
                      trControl = train_control)
cv_knn_error_rate <- 1 - cv_knn_model$results$Accuracy
cat("The error rate is: ",cv_knn_error_rate)
```

```
## The error rate is:  0.0738739
```

```r
summary(cv_knn_model)
```

```
##             Length Class      Mode
## learn        2     -none-     list
## k            1     -none-     numeric
## theDots      0     -none-     list
## xNames      42     -none-     character
## problemType  1     -none-     character
## tuneValue    1     data.frame list
## obsLevels    2     -none-     character
## param        0     -none-     list
```

```r
#test
knn_preds <- predict(cv_knn_model, newdata = ObesityTs)

csv_output(knn_preds, "knn_preds.csv")
```

```r
#Random Forest classifier
library(randomForest)
```

```
## randomForest 4.7-1.2
```

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

```
##
## Attaching package: 'randomForest'
```

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

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

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(caret)
ObesityTr$ObStatus <- as.factor(ObesityTr$ObStatus)

rf_model <- randomForest(ObStatus ~ ., data = ObesityTr, ntree = 100, mtry = 3)
```

```
rf_pred_tr <-  predict(rf_model, newdata = ObesityTr)
rf_df_mtx <- table(rf_pred_tr, ObesityTr$ObStatus)
rf_accuracy <- (rf_df_mtx[1,1] + rf_df_mtx[2, 2]) / sum(rf_df_mtx)
cat("Accuracy: ", rf_accuracy)
```

```
## Accuracy:  0.9998126
```

```
f_predictions <- predict(rf_model, newdata = ObesityTs)
summary((rf_model))
```

```
##                 Length Class  Mode
## call                 5  -none- call
## type                 1  -none- character
## predicted        32014  factor numeric
## err.rate           300  -none- numeric
## confusion            6  -none- numeric
## votes            64028  matrix numeric
## oob.times        32014  -none- numeric
## classes              2  -none- character
## importance          29  -none- numeric
## importanceSD         0  -none- NULL
## localImportance      0  -none- NULL
## proximity            0  -none- NULL
## ntree                1  -none- numeric
## mtry                 1  -none- numeric
## forest              14  -none- list
## y                32014  factor numeric
## test                 0  -none- NULL
## inbag                0  -none- NULL
## terms                3  terms  call
```

```
csv_output(f_predictions, "f_preds.csv")
```

##c) Report your accuracy based on your training data.

```
cat("Accuracy of logisstic: ", 1 - log_error_rate, "\n")
```

```
## Accuracy of logisstic:  0.741738
```

```
cat("Accuracy of lasso regression", lasso_train_accuracy, "\n")
```

```
## Accuracy of lasso regression 0.7424564
```

```
cat("Accuracy of lda: ", 1 - cv_lda_error_rate, "\n")
```

```
## Accuracy of lda:  0.7387099
```

```
cat("Accuracy of knn: ", 1 - cv_knn_error_rate, "\n")
```

```
## Accuracy of knn:  0.9261261
```

```
cat("Accuracy of random forest: ", rf_accuracy)
```

```
## Accuracy of random forest:  0.9998126
```

##d) Report your accuracy based on your testing (public score) on Kaggle logistic_preds.csv: 0.72891

lda_preds.csv: 0.74962

knn_preds.csv: 0.97366

f_preds.csv: 1.00000

##e) Report your rank on Kaggle at the time the predictions were submitted based on your public score.
rank 2