Classification of Cancer Causing Genes

Stats 101C Midterm Project

Atif Farook, Julia Wood, Conner Ching

Contents

I. Background

Successful discovery of cancer-driving genes, such as tumor suppressor genes (TSGs) and oncogenes (OGs), is crucial to preventing, diagnosing, and treating cancer. In this project, we will use statistical methods that we have learned in this course to identify both TSGs and OGs. We will use a comprehensive gene feature dataset as our training set. The predictors are various features of the genes. The response under study involves three classes: a gene being OG, TSG or NG. If we can have an accurate prediction of the class of genes, we can use this model to discover new TSGs and OGs, which will be very useful in cancer diagnosis.

II. Processing

1. Cleaning and Transforming Our Data

Since the data did not contain any NA values, we did not need to significantly clean the data. In addition, since we did not use a KNN model, we did not need to standardize or transform the predictors. Instead, our main preprocessing was choosing which predictors to use in our model. While it would have been possible to create a model with all 96 available predictors, such a model would be both computationally expensive and fraught with multicollinearity. Thus, to select predictors, we first looped through all pairs of predictor variables and excluded predictor pairs with high correlation (based on a threshold of 0.8).

2. Feature Selection and Cross Validation

To select our final predictors from this list of potential variables, we first created a logistic regression model with all predictors and then identified the predictors that have coefficients with extremely significant p-values (< 0.00001). Then, we applied cross validation to create a multinomial regression model with all predictors, and similarly identified the predictors with coefficients with significant p-values. We combined these lists of predictors to create a concise dataset with only these 13 predictors and the "class" variable.

3. Visualizing the Variables

After reducing our dataset, we conducted exploratory data analysis by constructing scatter and jitter plots for each predictor variable (Appendix a.) against the different classes to understand their distribution. We also created side-by-side boxplots ((Appendix b.) to inspect the distribution of predictors segmented by class, which allowed us to identify predictors that varied notably by class and were thus likely associated with class.

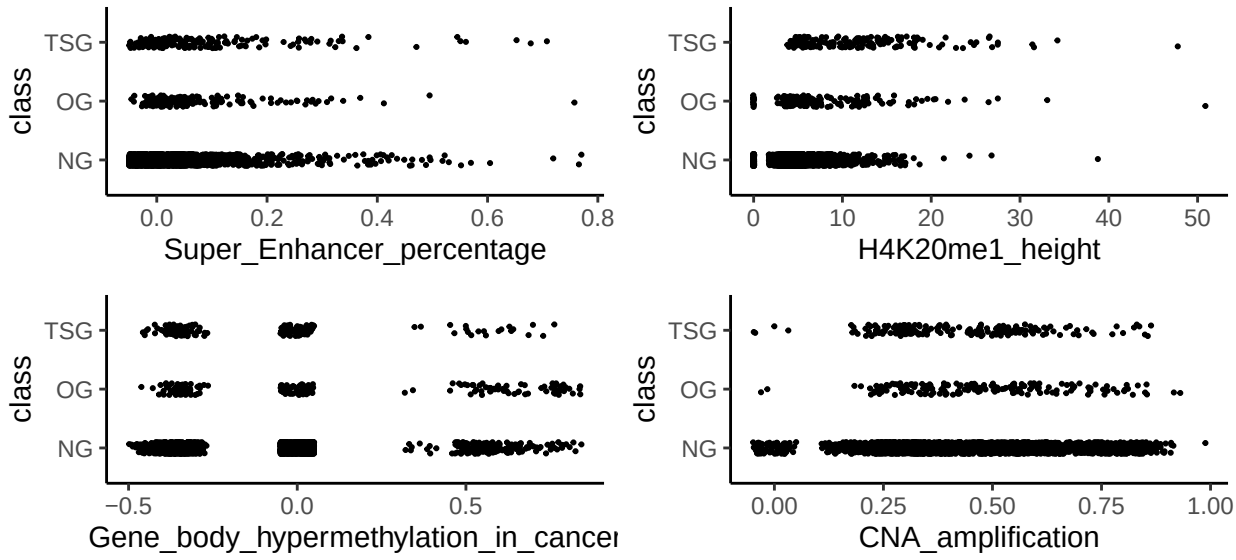


Figure 1: Examples of a few Scatter and Jitter Plots

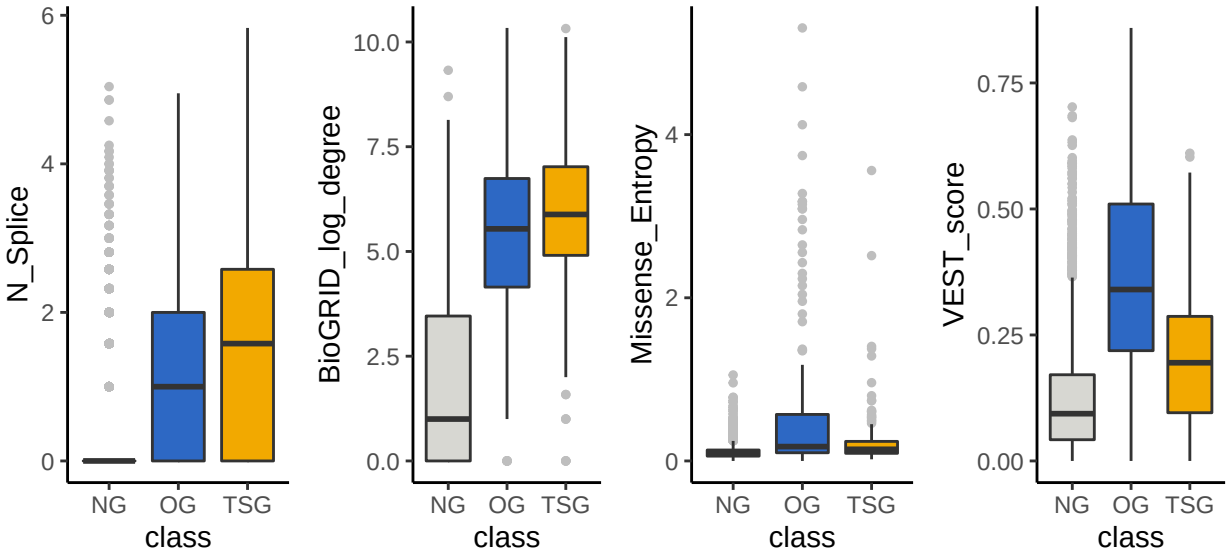


Figure 2: Examples of a few Box Plots

4. Further reduction using Cross Validation, Loops and Combinations

After completeing the processing above, we were left with 13 variables (Appendix c.). To further reduce the number of variables we created a loop that models and tests every possible combination of the 13 selected variables. That is, the code loops through every $\binom{13}{n}$ combination of predictors where n = 1,2, 3, . . . , 13, and outputs a list of coefficients and probabilities based on simple threshold values and then finally returns an test error rate. Based on the results from this loop we were able to further reduce our model to 8 variables.

III. Description of the Model

We used a multinomial regression model, built using the multinom function from the nnet package on 70% of the training data. We ultimately used 8 predictors:

Predictor	Coefficients (OG)	Coefficients (TSG)
(Intercept)	-9.273682	-8.443234
N_Splice	0.4994907	0.7702348
Super_Enhancer_percentage	2.780101	3.385923
H4K20me1_height	0.1668887	0.2351836
BioGRID_log_degree	0.5396548	0.6951637
Missense_Entropy	6.783425	6.169923
VEST_score	7.420182	1.444010
Gene_body_hypermethylation_in_cancer	2.467435	1.092625
CNA_amplification	0.7979840	-0.9524778

Table 1: Summary of Multinomial regression model

This model outputs a list of coefficients that represent the expected increase in the log odds of being an OG vs. an NG or a TSG vs. an NG based on a unit increase in a given predictor. For example, based on our model, for a unit increase in N_Splice (the number of splicing mutations), we would expect the log odds of a gene being an OG vs. an NG to increase by 0.499.

We then applied this model to our validation data (the remaining 30% of the training data). The model utilizes these coefficients to compute the probabilities of being an OG, TS, or NG (neutral gene) for a given gene. Rather than using the default threshold values or setting a blanket threshold to classify a gene based

on its probabilities, we created a multidimensional threshold as follows:

If P(NG) > 0.85 classify the gene as an NG, else,
If P(OG) > P(TSG) classify the gene as an OG, else,
If P(OG) < P(TSG) classify the gene as an TSG, else

We set these thresholds manually by repeatedly inspecting the model's incorrect predictions and adjusting the thresholds accordingly.

IV. Evaluation of the Model

1. Test error rate

The test error rate was a good baseline metric to understand if our model was producing accurate predictions. However, because the test error rate does not distinguish between the correct prediction rates of NGs, OGs, and TSGs, we found this evaluation metric to be limited in accurately measuring the success of our model.

2. Weighted Categorization Accuracy (WCA)

Instead of using the test error rate, we used WCA as an evaluation metric because it weights a correct OG or TSG classification higher than a correct NG classification. We believed it was important to prioritize correct OG and TSG classification over NG classification because, in reality, it is more dangerous to incorrectly classify a cancer gene than it is to incorrectly classify a non-cancer gene. In other words, we were willing to accept a high false positive rate (where "positive" = classified as a cancer gene) to minimize the false negative rate. This also maximizes our chances of achieving a high score in the Kaggle competition, since successful prediction of each OG or TSG is worth 20 points, whereas successful prediction of each NG is worth only 1 point. Thus, in order to choose our final model, we calculated the WCA for the top five models and submitted the two models with the highest WCA.

V. Success of the Model

We believe the success of our model is rooted in 2 things:

1. Multidimensional Threshold:

By closely inspecting our model's incorrect predictions, we were able to fine-tune our threshold to maximize our success.

2. Cross Validation and Testing Various Combinations:

With our exploratory analysis and careful selection of predictors, we were able to create a highly effective model with relatively few predictors. By using cross validation and testing out every possible combination we effectively reduced the number of predictors further and focused on selecting only the best performing ones. This allowed us to achieve **both low bias and low variance**: the model is flexible enough to successfully capture the variability in the data, but it is not so flexible that it is overfitted to the training data.

VI. Statement of Contributions

Atif Farook
Helped create, train and tune the logistic regression model. Wrote a function that could evaluate probabilites at different threshold values and output test error rates and a cross validation & combination loop that reduced the number of predictors to 8. Also helped compile the final document.

Julia Wood
I conducted a lot of preprocessing to investigate which predictors we should use in our model. I also explored LDA, QDA, and KNN models in addition to our multinomial regression, but we found that the multinomial logistic regression was the best choice.

Conner Ching
I wrote the majority of the report by taking the model that Atif and Julia built, writing an explanation of how it works and the work they did to create it, and organizing the flow to match with the R code.

VII. R Source Code

```
## LOADING REQUIRED PACKAGES
require(foreign)
require(nnet)
require(reshape2)
library(class)
library(MASS)
library(ggplot2)
library(tidyverse)
library(ggcorrplot)
library(grid)
library(gridExtra)
library(caret)
library(mclust)
library(boot)
library(MLeval)

## LOADING DATA SET
data <- read.csv("training.csv")

## Finding variables with high correlation (> 0.85)
## Note: Used feature selection using P-value in the end


var_names = names(data)
num_vars = length(var_names)

for (i1 in 1:(num_vars-1))
{
  v1 = var_names[i1]
  for (i2 in (i1+1):num_vars)
  {
    v2 = var_names[i2]

    c = cor(data[,v1], data[,v2])
    if (abs(c) > 0.85)
    {
```

```
      print(paste(v1, v2, sep=' - '))
      print(c)
    }
  }
}

set.seed(1)
## SELECTING VARIABLES USING LOGISTIC REGRESSION
## USING GML TO SELECT VARIABLES (Selection method 1)

logistic_reg = glm(class ~ ., data=data)
pvals = coef(summary(logistic_reg))[,4]
pvals = pvals[2:length(pvals)]

## STORING VARIABLES WITH HIGH P-VALUES (FINAL-1)
vars_to_keep <- names(data)[pvals < .00001]


## MULTINOMIAL CROSS VALIDATIONS WITH ENTIRE DATA SET (Selection method 2)

## METRICS FOR EVALUATION
train_control <- trainControl(method="cv", number = 5,
                              classProbs = TRUE,
                              savePredictions = TRUE)



## CONVERTING CLASS TO A FACTOR
data$class <- factor(data$class)
levels(data$class) <- c("NG", "OG", "TSG")

LRfit <- train(class~.,
               data = data, method = "multinom",
               preProc = c("center", "scale"),
               trControl = train_control)

## ESTIMATING P-VALUES
z <- summary(LRfit)$coefficients/summary(LRfit)$standard.errors
p <- (1 - pnorm(abs(z), 0, 1)) * 2

##STORING VARIABLES WITH HIGH P-VALUES  (FINAL 2)
vars_to_keep2 <- c("BioGRID_log_degree", "Missense_Entropy", "VEST_score",
                   "Missense_Damaging_TO_Missense_Benign_Ratio",
                   "Gene_body_hypermethylation_in_cancer", "CNA_amplification")

## COMBINING OUR BEST VARIABLES
model_1 <- c(vars_to_keep, vars_to_keep2)

variables <- model_1

data = data[,c(variables, "class")]
data$class <- factor(data$class)
levels(data$class) <- c("NG", "OG", "TSG")
```

```
## CREATING TRAINING DATA SET

#70% of data for train and 30% of data for test
train_size = floor(0.7 * nrow(data))

#get training indices
train_ind = sample(seq_len(nrow(data)), size = train_size)

data_train = data[train_ind, ]
data_test = data[-train_ind, ]

## VISUALIZING OUR VARIABLES - JITTER PLOTS
plots <- list(0)
box_plots <- list(0)
for (feat in variables)
{
    plots[[feat]] = ggplot(data, aes_string(feat, "class")) +
      geom_jitter(width=0.05, height=0.1, size=0.4) +
      theme_classic()
}

## VISUALIZING OUR VARIABLES - BOX PLOTS

for (feat in variables)
{
    box_plots[[feat]] = ggplot(data,
                               aes_string(x = "class",
                                          feat,
                                          fill = "class")) +
      geom_boxplot(outlier.colour="grey",
                   outlier.shape=19,
                   outlier.size=1) +
      scale_fill_manual(values=c("#D7D7D2",
                                 "#2D68C4",
                                 "#F2A900")) +
      theme_classic()  +  theme(legend.position = "none")
}


loop_predictors <- variables
entire_table <- list(0)
best_pred <- list(0)
best_pred_CV <- list(0)

## CROSS VALIDATION FOR ALL COMBINATIONS OF VARIABLES
## OUTPUT:
## 1. BEST PREDICTORS FOR EACH n
## 2. ACCURACY LEVELS

set.seed(1)
for (k in 1:13) {
pred <- k
```

```
loop_pred <- combn(loop_predictors, pred)
n <- (length(loop_pred)/pred)
results_pred <- matrix(nrow = n,
                       ncol = 3)
colnames(results_pred) <- c("Index", "Naive", "Diff.Thres")

for (i in 1:n) {
  results_pred[i,1] <- i
  data_loop <- data[c(loop_pred[,i], "class")]
  data_train_loop = data_loop[train_ind, ]
  data_test_loop = data_loop[-train_ind, ]

  logistic_reg_loop = nnet::multinom(class ~ ., data=data_train_loop)
  pred_logic_test <- predict(logistic_reg_loop, data_test_loop[, -(pred+1), drop = FALSE])

  results_pred[i,2] <- mean(pred_logic_test == data_test_loop$class)


  pp_raw <- predict(logistic_reg_loop, data_test_loop[, -(pred+1), drop = FALSE], "probs")
  pp <- pp_raw*100
  testing.set <- rep(0, length(pp[,1]))
  for(j in 1:length(testing.set)){
    if(pp[j,1] > 85 ){
      testing.set[j] <- 0
      }else{
        if(pp[j,2] >  pp[j,3] | pp[j,2] > 40){
          testing.set[j] <- 1
          }else{
            if(pp[j,2] <  pp[j,3]){
              testing.set[j] <- 2
            }
          }
      }
  }
  testing.set <- as.factor(testing.set)
  levels(testing.set) <- c("NG", "OG", "TSG")

  results_pred[i,3] <- mean(testing.set == data_test_loop$class)
}

final <- as.data.frame(results_pred) %>%
  arrange(desc(Naive))

entire_table[[pred]] <- final

best_pred[[pred]] <- c(loop_pred[,final[1,1]], final[1,2])
}

## BASED ON CV
model_2 <- c("N_Splice", "Super_Enhancer_percentage", "H4K20me1_height",
             "BioGRID_log_degree", "Missense_Entropy", "VEST_score",
             "Gene_body_hypermethylation_in_cancer", "CNA_amplification")
```

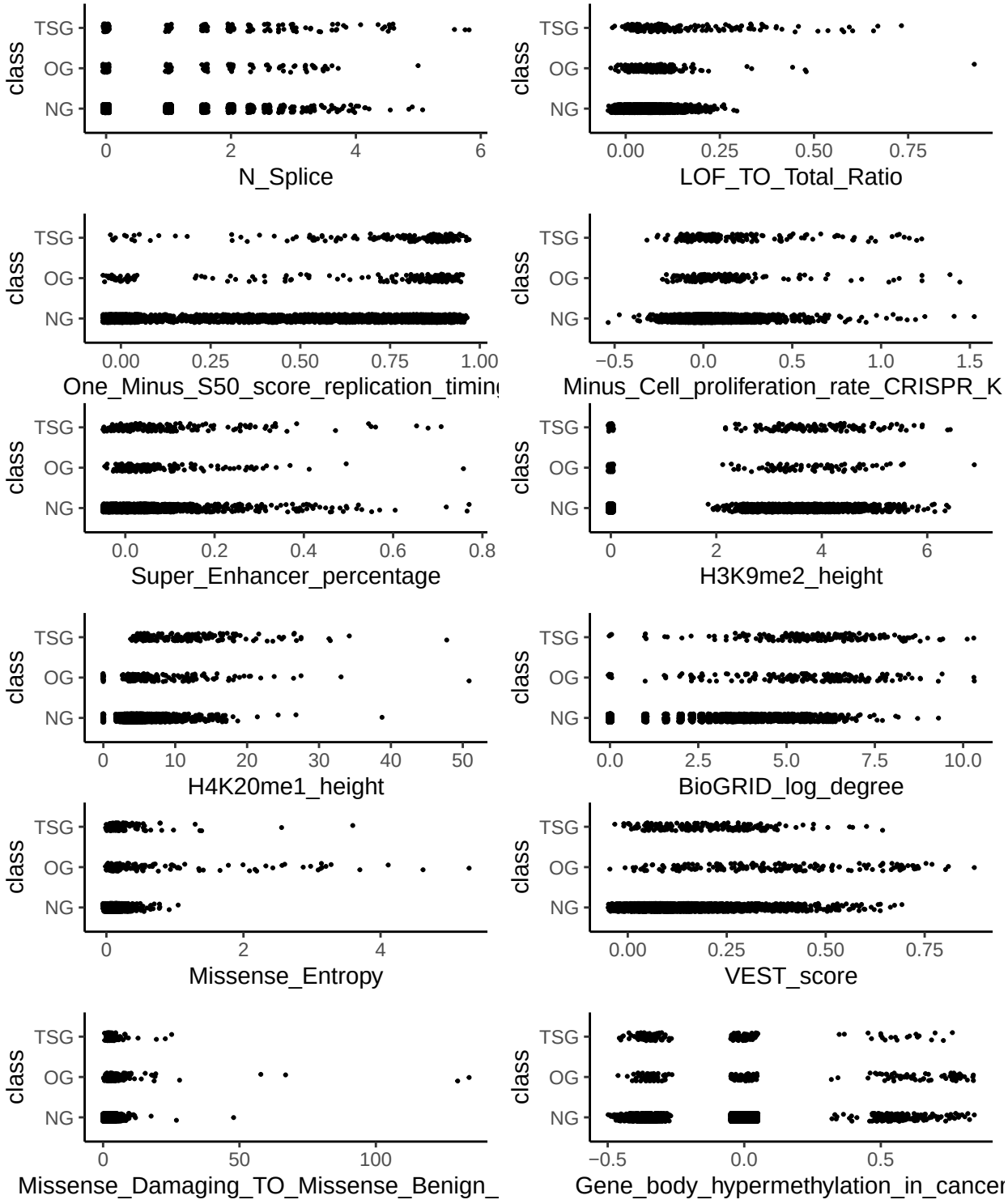
```r
variables <- model_2
data = data[,c(variables, "class")]
data$class <- factor(data$class)
levels(data$class) <- c("NG", "OG", "TSG")

set.seed(1234)
train_size = floor(0.7 * nrow(data))
train_ind = sample(seq_len(nrow(data)), size = train_size)


data_train = data[train_ind, ]
data_test = data[-train_ind, ]

logistic_reg = nnet::multinom(class ~ ., data=data_train)
summary(logistic_reg)


pred_logic_test <- predict(logistic_reg, data_test[, -(length(variables) + 1)])

pp_raw <- predict(logistic_reg, data_test[,-(length(variables) + 1)], "probs")

pp <- pp_raw*100
testing.set <- rep(0, length(pp[,1]))


## SIMPLE VERSION
for(i in 1:length(testing.set)){
  if(pp[i,1] > 85){
    testing.set[i] <- 0
  }else{
    if(pp[i,2] >  pp[i,3]){
      testing.set[i] <- 1
    }else{
      if(pp[i,2] <  pp[i,3]){
      testing.set[i] <- 2
      }
    }
  }
}

incorrect_index <- which(pred_logic_test != data_test$class)
incorrect_probs <- pp[c(incorrect_index),]

testing.set.final <- as.factor(testing.set)
levels(testing.set.final) <- c("NG", "OG", "TSG")

mean(pred_logic_test == data_test$class)
mean(testing.set.final == data_test$class)

sum(pred_logic_test != testing.set.final)
sum(pred_logic_test != data_test$class)
sum(testing.set.final != data_test$class)
```

```r
testing_incorrect_index <- which(testing.set.final != data_test$class)
testing_incorrect_probs <- pp[c(testing_incorrect_index),]

table(data_test$class[c(incorrect_index)])
table(data_test$class[c(testing_incorrect_index)])


testing.incorrect <- cbind(incorrect_probs,
                           data.frame(
                             "pred" = pred_logic_test[c(incorrect_index)]),
                           "actual" = data_test$class[c(incorrect_index)])


retesting.incorrect <- cbind(testing_incorrect_probs,
                             data.frame(
                               "pred" = testing.set.final[c(testing_incorrect_index)]),
                             "actual" = data_test$class[c(testing_incorrect_index)])
## CHECK YOUR DATA
retesting.incorrect ##threshold
testing.incorrect #without threshold

## FUNCTION TO EVALUATE OUR MODELS
get_wca <- function(pred, true)
{
 #get max score
 max_score = sum(true == "NG")*1 + sum(true == "OG")*20 + sum(true=="TSG")*20

 #get achieved score
 score = sum((true == "NG")&(pred == "NG"))*1 +
   sum((true == "OG")&(pred == "OG"))*20 +
   sum((true == "TSG")&(pred == "TSG"))*20
 #get wca

 return (score/max_score)
}

get_wca(testing.set.final, data_test$class)

## NEW FINAL OUTPUT
test <- read.csv("test.csv")
sample <- read.csv("sample.csv")

## CHOSEN PREDICTORS
sample.prob <- predict(logistic_reg, test[,variables], "probs")

sample_pp <- sample.prob*100
testing.sample <- rep(0, length(sample_pp[,1]))

## CHOSEN THRESHOLD VALUES
for(i in 1:length(testing.sample)){
  if(sample_pp[i,1] > 85){
    testing.sample[i] <- 0
  }else{
```

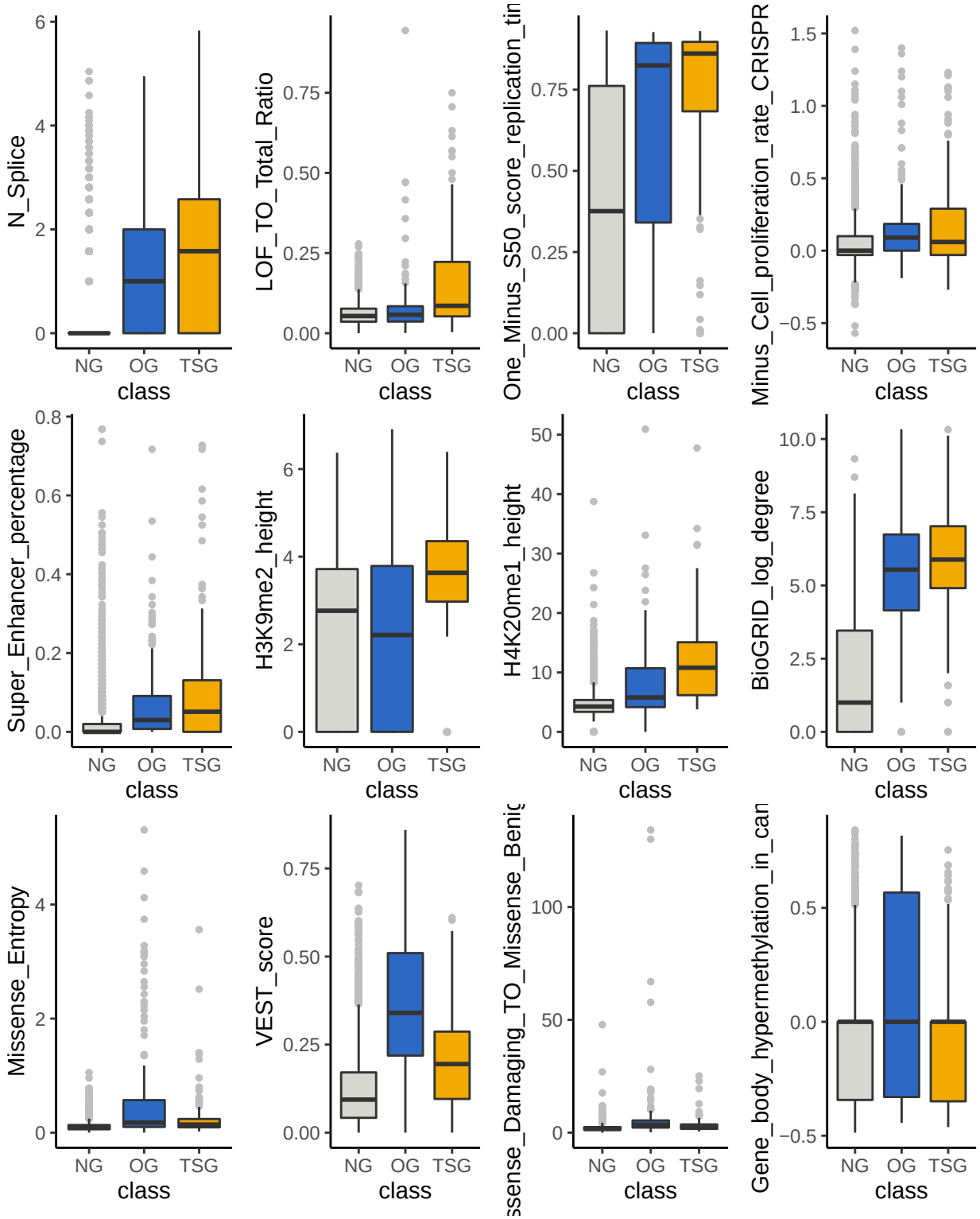
```
    if(sample_pp[i,2] >  sample_pp[i,3]){
      testing.sample[i] <- 1
    }else{
      if(sample_pp[i,2] <  sample_pp[i,3]){
      testing.sample[i] <- 2
      }
    }
  }
}

## CROSS CHECKING AND CONVERTING

sample$class <- as.integer(testing.sample)
table(sample$class)

## FINAL DOCUMENT
write.csv(sample,"cvlogistic.csv", row.names = FALSE)
```

VIII. Appendix

a. Jitter Plots



b. Box Plots


N_Splice
LOF_TO_Total_Ratio
One_Minus_S50_score_replication_tin
Minus_Cell_proliferation_rate_CRISPR
Super_Enhancer_percentage
H3K9me2_height
H4K20me1_height
BioGRID_log_degree
Missense_Entropy
VEST_score
ssense_Damaging_TO_Missense_Benig
Gene_body_hypermethylation_in_can
NG
OG
TSG
class

c. Model 1

Predictors		
N_Splice	Super_Enhancer_percentage	H4K20me1_height
Missense_Damaging_TO_Missense_Benign_Ratio	Missense_Entropy	VEST_score
Gene_body_hypermethylation_in_cancer	CNA_amplification	LOF_TO_Total_Ratio
One_Minus_S50_score_replication_timing	H3K9me2_height	BioGRID_log_degree
Minus_Cell_proliferation_rate_CRISPR_KD		

d. Model 2

Predictor	Coefficients (OG)	Coefficients (TSG)
(Intercept)	-9.273682	-8.443234
N_Splice	0.4994907	0.7702348
Super_Enhancer_percentage	2.780101	3.385923
H4K20me1_height	0.1668887	0.2351836
BioGRID_log_degree	0.5396548	0.6951637
Missense_Entropy	6.783425	6.169923
VEST_score	7.420182	1.444010
Gene_body_hypermethylation_in_cancer	2.467435	1.092625
CNA_amplification	0.7979840	-0.9524778