Statistics 101C Midterm Report:

Classifying Cancer Genes using Mutation-Related, Genomic, Phenotype and Epigenetic Features

Team: Sunflower Cello Hotdog

Anna Piskun —

Derek Wang —

Tomi Rajninger –

1 Introduction

According to the National Cancer Institute, approximately 39.5% of American men and women will receive a cancer diagnosis at some point during their lifetimes.[1] Cancer itself is a group of diseases involving abnormal cell growth that can potentially spread to other parts of the body. Most cancers are due to mutations in the genome. While oncogenes (OGs) and tumor suppressor genes (TSGs) normally work together to prevent abnormal cell growth, their balance is disrupted by mutations that then cause cancer. Thus, early discovery and identification of these cancer-driving genes is crucial in the progression of cancer research and achievement of more effective cancer diagnoses in the future.[2] As such, this midterm project tasked us with building an effective predictive model to identify both OGs and TSGs from a sample of over 3000 genes.

2 Methodology

2.1 Data Preprocessing

Our first step in preprocessing the data was to run the str() function to get a general overview of the training data's structure and its predictors. We found that all columns were either numeric or integer types. This included the response variable, *class*, which we later converted into a factor because this variable is actually categorical with levels of 0, 1 and 2 corresponding to neutral genes, oncogenes and tumor suppressor genes, respectively. Initially, however, we kept the *class* variable as a numeric type in order to run an ANOVA test, which does not accept categorical response variables.

Additionally, we checked for any missing (NA) values in both the training and test files. We thus concluded that we did not have to remove any observations because all of the observations contained data (except for *class* in the test file, which is to be expected). Lastly, we transformed our dataset by scaling and summing up various groups of similar predictors to create 11 new "grouped" variables. (For example, the transformed *H3K4me3* variable was created from scaling and summing up the *Length_H3K4me3*, *Broad_H3K4me3_percentage*, and *H3K4me3_height* variables). These grouping transformations (which reduced the number of predictors) seemed to improve our accuracy by both potentially reducing multicollinearity and resulting in a less flexible model with a lower variance and, therefore, a more consistent performance.

2.2 Statistical Model

In the end, we chose to utilize a multinomial logistic regression model with a reduced number of predictors that drew data from 52 scaled predictors through 5-fold Cross-Validation. Specifically, we reduced the total number of predictors from 97 to 52, choosing to include 41 (stand-alone) statistically significant predictors[3] and 11 grouped/transformed predictors.[4] Ultimately, we used the train() function along with the preProcess argument to center, scale, and fit our final LR model.

---

[1] National Cancer Institute. (n.d.). Cancer Statistics. Retrieved November 09, 2020, from https://www.cancer.gov/about-cancer/understanding/statistics

[2] All information above on cancer genes and our specific dataset were extracted from the "Overview" section on Kaggle.

[3] Predictors with a p-value with a */**/*** generated from an initial `aov(class~.)` output.

[4] Refer to Appendix for full list of stand-alone and grouped/transformed predictors in `summary(anova)` output.

As mentioned above, we used 5-fold Cross-Validation to evaluate the performance of each model we constructed. Specifically, after deciding on our final set of predictors, we compared the accuracies from train() outputs for KNN, LDA, QDA and LR models with these same predictors. From these outputs (included in Figure 1 below), we concluded that the multinomial logistic regression model had the highest accuracy of all four models and, therefore, that it was the best model to move forward with.

```
k-Nearest Neighbors

3177 samples
  52 predictor
   3 classes: '0', '1', '2'

Pre-processing: centered (52), scaled (52)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2543, 2541, 2541, 2542, 2541
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.9200504  0.5356375
   6  0.9348422  0.5757408
  11  0.9291764  0.5054831
  16  0.9263412  0.4755577
  21  0.9266572  0.4721121
  26  0.9238245  0.4419982
  31  0.9225662  0.4261493
  36  0.9231961  0.4301646
  41  0.9206779  0.4055757
  46  0.9197355  0.3921074

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 6.
```

```
Linear Discriminant Analysis

3177 samples
  52 predictor
   3 classes: '0', '1', '2'

Pre-processing: centered (52), scaled (52)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2543, 2541, 2541, 2542, 2541
Resampling results:

  Accuracy   Kappa
  0.9480666  0.701043
```

```
Quadratic Discriminant Analysis

3177 samples
  51 predictor
   3 classes: '0', '1', '2'

Pre-processing: centered (51), scaled (51)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2543, 2541, 2541, 2542, 2541
Resampling results:

  Accuracy   Kappa
  0.9288619  0.5965765
```

```
Penalized Multinomial Regression

3177 samples
  52 predictor
   3 classes: '0', '1', '2'

Pre-processing: centered (52), scaled (52)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 2543, 2541, 2541, 2542, 2541
Resampling results across tuning parameters:

  decay  Accuracy   Kappa
  0e+00  0.9458639  0.7037863
  1e-04  0.9452344  0.7013020
  1e-01  0.9493270  0.7183759

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was decay = 0.1.
```

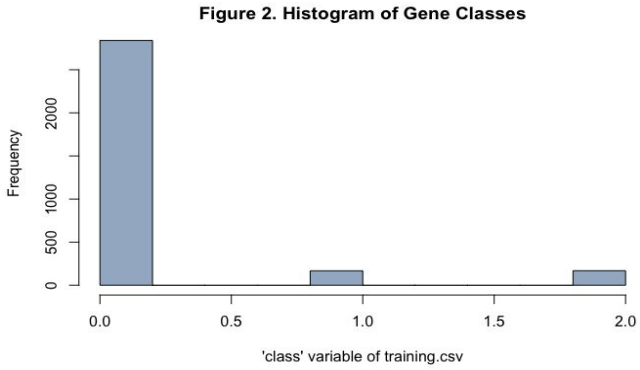
Figure 1. Cross-Validation Outputs. The outputs above indicate that the 5-fold Cross-Validation accuracies for the KNN (k = 6) , LDA, QDA and logistic regression (decay = 0.1) models using our final set of predictors were approximately 0.9348, 0.9481, 0.9289 and 0.9493, respectively. Again, the multinomial logistic regression model had the greatest accuracy and thus, was chosen as our final model for this project.

3 Results

Our final submission to Kaggle was a multinomial logistic regression model with the 41 stand-alone significant ANOVA predictors as well as an additional 11 transformed predictors. According to the public leaderboard on Kaggle, this model's value for the WCA metric was **0.76459**.

4 General Conclusions

Out of all methods we tested, the LDA and multinomial LR models performed the best. As such, we are able to assume that the provided data follows a more linear Bayes decision boundary line. This would also justify why QDA ended up performing the worst. Based on the clearly non-normal histogram to the right, we also believe that the multinomial LR model outperformed the LDA model because LDA requires a normality assumption while LR does not.



Analyzing our model's performance, we can see that this multinomial LR model works well for several reasons**.** First, this model significantly cut down the number of predictors in order to improve model accuracy. In our final model, we chose significant predictors based on the results of the fitted ANOVA model. Our motivation behind using the aov() function follows a simplified process of elimination. Since *class* is a factor/categorical variable with three levels (0, 1 and 2), we wanted to conduct a t-test between *class* and each of the other 97 predictors. These t-tests would more accurately consider class as a categorical variable compared to computing correlation coefficients between *class* and the other predictors, for example. However, because the *class* variable has *three* levels (not two), we were unable to use the t.test() function because it performs a Welch Two-Sample T-Test. Instead, we concluded that conducting an analysis of variance (ANOVA) between *class* and the other 97 predictors was the most appropriate method for comparing the three group means and ultimately choosing the best subset of predictors.

Secondly, we believe our model works well because the assumptions of logistic regression (LR) are satisfied. Unlike many other OLS-based linear models, LR does not make the assumptions of linearity, normality, nor homoscedasticity, for example. LR does *not* require a linear relationship between the response and predictor variables nor does it require the residuals to be normally distributed or for there to be constant variance (homoscedasticity) throughout the model.[5] Lastly, LR performs best with a large sample size, which we consider our training set to be. This all further justifies that our final submitted model is valid.

Taking stock of our final multinomial logistic regression model and its performance, it is also important to acknowledge the ways in which it could have performed better. First and foremost, a more exhaustive and statistically rigorous approach could have been utilized in order to select a more effective subset of predictors. Likewise, more advanced statistical modeling techniques would likely have increased our model's predictive ability. However, since this project limited us to content in the scope of this course, many of these other methods were unattainable.

[5] Assumptions of Logistic Regression. (2020, June 22). Retrieved November 09, 2020, from https://www.statisticssolutions.com/assumptions-of-logistic-regression/

6 Appendix — R Code

 # Step 1: Initial Set-Up and Exploration of Data

```
# load packages
library(caret)
library(dplyr)
library(nnet)

# download training and test data
training <- read.csv(file = "/Users/Tomi/Google Drive/Stats
101C/Midterm Project/ucla-stats101c-lec4/training.csv")

test <- read.csv(file = "/Users/Tomi/Google Drive/Stats
101C/Midterm Project/ucla-stats101c-lec4/test.csv")

# description of dataset
   # each row: gene
   # each column: feature of gene
   # 3177 rows (genes), 97 predictors (99 columns but id and class are not predictors)
   # response variable: class — 0 = NG; 1 = OG; 2 = TSG

# explore and clean dataset

   # 1. explore structure of dataset, types of predictors
   str(training)
     # all variables except class are initially stored as numeric or integer
     # class is stored as numeric, but should technically be factor (levels 0, 1, 2)
     # looked at codebook to identify if any other predictors should be changed to factors
     # conclusion: we don't need to change any other predictors to factors

   # 2. check for any missing values in training and test sets
   any(is.na(training))  # outputs FALSE, so no missing values in training.csv
   any(is.na(test))  # outputs TRUE, so some missing (NA) values in test.csv


   # investigate test.csv to identify which variables have NA values
   fun <- function(x){
        any(is.na(x))
      }
```

```
is_NA <- lapply(test, fun)
which(is_NA == TRUE) # outputs that class is only column with missing values
# this is expected because we know that test.csv doesn't include the true gene classes
# conclusion: we don't need to remove any missing observations
```

 # Note: Because class is a factor/categorical variable with 3 classes (0, 1 and 2), it doesn't make sense to calculate the correlation coefficient between *class* and other predictors. Specifically, the cor() function in R requires both variables to be numeric. So, instead of calculating correlation coefficients, we'll want to investigate whether each level of the *class* variable has a different mean value for each of the predictors. While this would normally be done using a Welch Two-Sample T-Test in R, ANOVA is the appropriate method for a comparison of more than two group means (*class* has 3 levels, not 2). Thus, below we conduct an analysis of variance (ANOVA) that predicts *class* using all other columns in the training.csv dataset.

```
anova <- aov(class ~ ., data = training)
summary(anova)
```

 # listed below are all *//*** significant ANOVA predictors of class ~ .**

 # Silent_KB_Ratio, N_Missense, N_LOF, N_Splice, Missense_KB_Ratio, LOF_KB_Ratio, Missense_Entropy, LOF_TO_Silent_Ratio, Missense_TO_Silent_Ratio, LOF_TO_Benign_Ratio, Missense_TO_Benign_Ratio, Missense_Damaging_TO_Benign_Ratio, Polyphen2, LOF_TO_Total_Ratio, Missense_TO_Total_Ratio, Splice_TO_Total_Ratio, LOF_TO_Missense_Ratio, Silent_fraction, Nonsense_fraction, Missense_fraction, Recurrent_missense_fraction, Frameshift_indel_fraction, Inactivating_mutations_fraction, log_gene_length, CDS_length, CNA_deletion, Exon_Cons, MGAentropy, VEST_score, Cell_proliferation_rate_CRISPR_KD, Super_Enhancer_percentage, BioGRID_betweenness, BioGRID_clossness, BioGRID_log_degree, Gene_body_hypermethylation_in_cancer, Canyon_genebody_hypermethylation, intolerant_pLI, Synonymous_Zscore, Missense_Zscore, pLOF_Zscore, ncGERP, Length_H3K4me3, Broad_H3K4me3_percentage, H3K4me3_height, H3K4me2_width + Broad_H3K4me2_percentage, H3K4me1_width, Broad_H3K4me1_percentage, H3K4me1_height, H3K36me3_width, Broad_H3K36me3_percentage, H3K36me3_height, Broad_H3K27ac_percentage, Broad_H3K27me3_percentage, H3K9me3_width, Broad_H3K9me2_percentage, H3K79me2_width, H3K79me2_height, H4K20me1_width, Broad_H4K20me1_percentage, H4K20me1_height

 # plot histogram of *class* variable to investigate its distribution

```
hist(training$class, main = "Figure 2. Histogram of Gene
Classes", xlab = "'class' variable of training.csv", col =
"slategray3")
```

 # Step 2: Transform specified *training.csv* predictors

 # all new transformed predictors: H3K4me3, H3K4me2, H3K4me1, H3K36me3, H3K27ac, H3K27me3, H3K9me3, H3K9ac, H3K9me2, H3K79me2, H4K20me1
 # create transformations by combining all predictors that were part of a "group"
 # specifically, we scaled and then combined all groups of variables with the same "prefix"

```
H3K4me3 <- rowSums(scale(training$Length_H3K4me3),
scale(training$Broad_H3K4me3_percentage))
H3K4me3 <- rowSums(scale(training$H3K4me3_height), H3K4me3)

H3K4me2 <- rowSums(scale(training$H3K4me2_width),
scale(training$Broad_H3K4me2_percentage))
H3K4me2 <- rowSums(scale(training$H3K4me2_height), H3K4me2)

H3K4me1 <- rowSums(scale(training$H3K4me1_width),
scale(training$Broad_H3K4me1_percentage))
H3K4me1 <- rowSums(scale(training$H3K4me1_height), H3K4me1)

H3K36me3 <- rowSums(scale(training$H3K36me3_width),
scale(training$Broad_H3K36me3_percentage))
H3K36me3 <- rowSums(scale(training$H3K36me3_height), H3K36me3)

H3K27ac <- rowSums(scale(training$H3K27ac_width),
scale(training$Broad_H3K27ac_percentage))
H3K27ac <- rowSums(scale(training$H3K27ac_height), H3K27ac)

H3K27me3 <-rowSums(scale(training$H3K27me3_width),
scale(training$Broad_H3K27me3_percentage))
H3K27me3 <- rowSums(scale(training$H3K27me3_height), H3K27me3)

H3K9me3 <- rowSums(scale(training$H3K9me3_width),
scale(training$Broad_H3K9me3_percentage))
H3K9me3 <- rowSums(scale(training$H3K9me3_height), H3K9me3)

H3K9ac <-  rowSums(scale(training$H3K9ac_width),
scale(training$Broad_H3K9ac_percentage))
```

```
H3K9ac <- rowSums(scale(training$H3K9ac_height),H3K9ac  )

H3K9me2 <- rowSums(scale(training$H3K9me2_width),
scale(training$Broad_H3K9me2_percentage))
H3K9me2 <- rowSums(scale(training$H3K9me2_height),H3K9me2 )

H3K79me2 <- rowSums(scale(training$H3K79me2_width),
scale(training$Broad_H3K79me2_percentage))
H3K79me2 <- rowSums(scale(training$H3K79me2_height), H3K79me2)

H4K20me1 <- rowSums(scale(training$H4K20me1_width),
scale(training$Broad_H4K20me1_percentage))
H4K20me1 <- rowSums(scale(training$H4K20me1_height), H4K20me1)
```

 # remove all individual "group" variables from *training.csv* (columns 66-98)
 # (*class* was 99th column, so store it in separate object, remove columns 66-99)
 # then, add *class* back to end of *training.csv* after adding the 11 transformed predictors

```
class <- training$class
training <- training[,-66:-99]
training <- data.frame(training, H3K4me3, H3K4me2, H3K4me1,
H3K36me3, H3K27ac, H3K27me3, H3K9me3, H3K9ac, H3K9me2, H3K79me2,
H4K20me1, class)
```

 # kept columns 1-65 and class, added 11 transformed variables = 77 columns in training.csv
 # *class* variable (outcome variable) is now 77th column in *training.csv*

```
dim(training)
```

Step 3: Fit 4 Models (KNN, LDA, QDA, LR) with Final Subset of Predictors and Compare Models using 5-Fold Cross-Validation

 # overview of our final set of predictors
 # 41 stand-alone *//*** predictors , not part of groups**
 # then, added 11 more transformed variables
 # 41 + 11 = 52 rpedictors total in this model

 # first, convert *class* into factor and establish settings for 5-fold Cross Validation

```
training$class <- factor(training$class)
train_control <- trainControl(method = "cv", number = 5)
```

 # fit KNN model

```
set.seed(123)
```

```
KNNfit <- train(class ~ Silent_KB_Ratio +
                        N_Missense +
                        N_LOF +
                        N_Splice +
                        Missense_KB_Ratio +
                        LOF_KB_Ratio +
                        Missense_Entropy +
                        LOF_TO_Silent_Ratio +
                        Missense_TO_Silent_Ratio +
                        LOF_TO_Benign_Ratio +
                        Missense_TO_Benign_Ratio +
                        Missense_Damaging_TO_Benign_Ratio +
                        Polyphen2 +
                        LOF_TO_Total_Ratio +
                        Missense_TO_Total_Ratio +
                        Splice_TO_Total_Ratio +
                        LOF_TO_Missense_Ratio +
                        Silent_fraction +
                        Nonsense_fraction +
                        Missense_fraction +
                        Recurrent_missense_fraction +
                        Frameshift_indel_fraction +
                        Inactivating_mutations_fraction +
                        log_gene_length +
                        CDS_length +
                        CNA_deletion +
                        Exon_Cons +
                        MGAentropy +
                        VEST_score +
                        Cell_proliferation_rate_CRISPR_KD +
                        Super_Enhancer_percentage +
                        BioGRID_betweenness +
                        BioGRID_clossness +
                        BioGRID_log_degree +
                        Gene_body_hypermethylation_in_cancer +
                        Canyon_genebody_hypermethylation +
                        intolerant_pLI +
                        Synonymous_Zscore +
                        Missense_Zscore +
                        pLOF_Zscore +
                        ncGERP +
```

```
                        #Length_H3K4me3 +
                        #Broad_H3K4me3_percentage +
                        #H3K4me3_height +
                        #H3K4me2_width +
                        #Broad_H3K4me2_percentage +
                        #H3K4me1_width +
                        #Broad_H3K4me1_percentage +
                        #H3K4me1_height +
                        #H3K36me3_width +
                        #Broad_H3K36me3_percentage +
                        #H3K36me3_height +
                        #Broad_H3K27ac_percentage +
                        #Broad_H3K27me3_percentage +
                        #H3K9me3_width +
                        #Broad_H3K9me2_percentage +
                        #H3K79me2_width +
                        #H3K79me2_height +
                        #H4K20me1_width +
                        #Broad_H4K20me1_percentage +
                        #H4K20me1_height +
                        H3K4me3 +
                        H3K4me2 +
                        H3K4me1 +
                        H3K36me3 +
                        H3K27ac +
                        H3K27me3 +
                        H3K9me3 +
                        H3K9ac +
                        H3K9me2 +
                        H3K79me2 +
                        H4K20me1,
                        data = training, method = "knn",
                        preProc = c("center", "scale"),
                        trControl = train_control,
                        tuneGrid = expand.grid(k=seq(1,50,by=5)))

# fit LDA model
set.seed(123)
LDAfit <- train(class ~
                        Silent_KB_Ratio +
                        N_Missense +
```

```
N_LOF +
N_Splice +
Missense_KB_Ratio +
LOF_KB_Ratio +
Missense_Entropy +
LOF_TO_Silent_Ratio +
Missense_TO_Silent_Ratio +
LOF_TO_Benign_Ratio +
Missense_TO_Benign_Ratio +
Missense_Damaging_TO_Benign_Ratio +
Polyphen2 +
LOF_TO_Total_Ratio +
Missense_TO_Total_Ratio +
Splice_TO_Total_Ratio +
LOF_TO_Missense_Ratio +
Silent_fraction +
Nonsense_fraction +
Missense_fraction +
Recurrent_missense_fraction +
Frameshift_indel_fraction +
Inactivating_mutations_fraction +
log_gene_length +
CDS_length +
CNA_deletion +
Exon_Cons +
MGAentropy +
VEST_score +
Cell_proliferation_rate_CRISPR_KD +
Super_Enhancer_percentage +
BioGRID_betweenness +
BioGRID_clossness +
BioGRID_log_degree +
Gene_body_hypermethylation_in_cancer +
Canyon_genebody_hypermethylation +
intolerant_pLI +
Synonymous_Zscore +
Missense_Zscore +
pLOF_Zscore +
ncGERP +
#Length_H3K4me3 +
#Broad_H3K4me3_percentage +
```

```
                        #H3K4me3_height +
                        #H3K4me2_width +
                        #Broad_H3K4me2_percentage +
                        #H3K4me1_width +
                        #Broad_H3K4me1_percentage +
                        #H3K4me1_height +
                        #H3K36me3_width +
                        #Broad_H3K36me3_percentage +
                        #H3K36me3_height +
                        #Broad_H3K27ac_percentage +
                        #Broad_H3K27me3_percentage +
                        #H3K9me3_width +
                        #Broad_H3K9me2_percentage +
                        #H3K79me2_width +
                        #H3K79me2_height +
                        #H4K20me1_width +
                        #Broad_H4K20me1_percentage +
                        #H4K20me1_height +
                        H3K4me3 +
                        H3K4me2 +
                        H3K4me1 +
                        H3K36me3 +
                        H3K27ac +
                        H3K27me3 +
                        H3K9me3 +
                        H3K9ac +
                        H3K9me2 +
                        H3K79me2 +
                        H4K20me1,
                        data = training,
                        method = "lda",
                        preProcess = c("center", "scale"),
                        trControl = train_control)

# fit QDA model
set.seed(123)
QDAfit <- train(class ~
                        Silent_KB_Ratio +
                        N_Missense +
                        N_LOF +
                        N_Splice +
```

```
Missense_KB_Ratio +
LOF_KB_Ratio +
Missense_Entropy +
LOF_TO_Silent_Ratio +
Missense_TO_Silent_Ratio +
LOF_TO_Benign_Ratio +
Missense_TO_Benign_Ratio +
Missense_Damaging_TO_Benign_Ratio +
Polyphen2 +
LOF_TO_Total_Ratio +
Missense_TO_Total_Ratio +
Splice_TO_Total_Ratio +
LOF_TO_Missense_Ratio +
Silent_fraction +
Nonsense_fraction +
Missense_fraction +
#Recurrent_missense_fraction
# removed above variable b/c not running
Frameshift_indel_fraction +
Inactivating_mutations_fraction +
log_gene_length +
CDS_length +
CNA_deletion +
Exon_Cons +
MGAentropy +
VEST_score +
Cell_proliferation_rate_CRISPR_KD +
Super_Enhancer_percentage +
BioGRID_betweenness +
BioGRID_clossness +
BioGRID_log_degree +
Gene_body_hypermethylation_in_cancer +
Canyon_genebody_hypermethylation +
intolerant_pLI +
Synonymous_Zscore +
Missense_Zscore +
pLOF_Zscore +
ncGERP +
#Length_H3K4me3 +
#Broad_H3K4me3_percentage +
#H3K4me3_height +
```

```
                        #H3K4me2_width +
                        #Broad_H3K4me2_percentage +
                        #H3K4me1_width +
                        #Broad_H3K4me1_percentage +
                        #H3K4me1_height +
                        #H3K36me3_width +
                        #Broad_H3K36me3_percentage +
                        #H3K36me3_height +
                        #Broad_H3K27ac_percentage +
                        #Broad_H3K27me3_percentage +
                        #H3K9me3_width +
                        #Broad_H3K9me2_percentage +
                        #H3K79me2_width +
                        #H3K79me2_height +
                        #H4K20me1_width +
                        #Broad_H4K20me1_percentage +
                        #H4K20me1_height +
                        H3K4me3 +
                        H3K4me2 +
                        H3K4me1 +
                        H3K36me3 +
                        H3K27ac +
                        H3K27me3 +
                        H3K9me3 +
                        H3K9ac +
                        H3K9me2 +
                        H3K79me2 +
                        H4K20me1,
                        data = training,
                        method = "qda",
                        preProcess = c("center", "scale"),
                        trControl = train_control)

# fit LR model
set.seed(123)
LRfit <- train(class ~
                        Silent_KB_Ratio +
                        N_Missense +
                        N_LOF +
                        N_Splice +
                        Missense_KB_Ratio +
```

```
LOF_KB_Ratio +
Missense_Entropy +
LOF_TO_Silent_Ratio +
Missense_TO_Silent_Ratio +
LOF_TO_Benign_Ratio +
Missense_TO_Benign_Ratio +
Missense_Damaging_TO_Benign_Ratio +
Polyphen2 +
LOF_TO_Total_Ratio +
Missense_TO_Total_Ratio +
Splice_TO_Total_Ratio +
LOF_TO_Missense_Ratio +
Silent_fraction +
Nonsense_fraction +
Missense_fraction +
Recurrent_missense_fraction +
Frameshift_indel_fraction +
Inactivating_mutations_fraction +
log_gene_length +
CDS_length +
CNA_deletion +
Exon_Cons +
MGAentropy +
VEST_score +
Cell_proliferation_rate_CRISPR_KD +
Super_Enhancer_percentage +
BioGRID_betweenness +
BioGRID_clossness +
BioGRID_log_degree +
Gene_body_hypermethylation_in_cancer +
Canyon_genebody_hypermethylation +
intolerant_pLI +
Synonymous_Zscore +
Missense_Zscore +
pLOF_Zscore +
ncGERP +
#Length_H3K4me3 +
#Broad_H3K4me3_percentage +
#H3K4me3_height +
#H3K4me2_width +
#Broad_H3K4me2_percentage +
```

```
                        #H3K4me1_width +
                        #Broad_H3K4me1_percentage +
                        #H3K4me1_height +
                        #H3K36me3_width +
                        #Broad_H3K36me3_percentage +
                        #H3K36me3_height +
                        #Broad_H3K27ac_percentage +
                        #Broad_H3K27me3_percentage +
                        #H3K9me3_width +
                        #Broad_H3K9me2_percentage +
                        #H3K79me2_width +
                        #H3K79me2_height +
                        #H4K20me1_width +
                        #Broad_H4K20me1_percentage +
                        #H4K20me1_height +
                        H3K4me3 +
                        H3K4me2 +
                        H3K4me1 +
                        H3K36me3 +
                        H3K27ac +
                        H3K27me3 +
                        H3K9me3 +
                        H3K9ac +
                        H3K9me2 +
                        H3K79me2 +
                        H4K20me1,
                        data = training,
                        method = "multinom",
                        preProcess = c("center", "scale"),
                        trControl =  train_control)
```

 # compare accuracies of the 4 models using 5-fold CV

```
KNNfit
LDAfit
QDAfit
LRfit
```

 # conclusion: LRfit had highest accuracy, so we should use LRfit to predict test.csv data

 # Step 4: Transform specified *test.csv* predictors

 # transform the test.csv (will lose previous data from training.csv transformations)
 # all new: H3K4me3, H3K4me2, H3K4me1, H3K36me3, H3K27ac, H3K27me3, H3K9me3, H3K9ac, H3K9me2, H3K79me2, H4K20me1

```
H3K4me3 <- rowSums(scale(test$Length_H3K4me3),
scale(test$Broad_H3K4me3_percentage))
H3K4me3 <- rowSums(scale(test$H3K4me3_height), H3K4me3)

H3K4me2 <- rowSums(scale(test$H3K4me2_width),
scale(test$Broad_H3K4me2_percentage))
H3K4me2 <- rowSums(scale(test$H3K4me2_height), H3K4me2)

H3K4me1 <- rowSums(scale(test$H3K4me1_width),
scale(test$Broad_H3K4me1_percentage))
H3K4me1 <- rowSums(scale(test$H3K4me1_height), H3K4me1)

H3K36me3 <- rowSums(scale(test$H3K36me3_width),
scale(test$Broad_H3K36me3_percentage))
H3K36me3 <- rowSums(scale(test$H3K36me3_height), H3K36me3)

H3K27ac <- rowSums(scale(test$H3K27ac_width),
scale(test$Broad_H3K27ac_percentage))
H3K27ac <- rowSums(scale(test$H3K27ac_height), H3K27ac)

H3K27me3_ <-rowSums(scale(test$H3K27me3_width),
scale(test$Broad_H3K27me3_percentage))
H3K27me3 <- rowSums(scale(test$H3K27me3_height), H3K27me3)

H3K9me3 <-  rowSums(scale(test$H3K9me3_width),
scale(test$Broad_H3K9me3_percentage))
H3K9me3 <-  rowSums(scale(test$H3K9me3_height), H3K9me3)

H3K9ac <-  rowSums(scale(test$H3K9ac_width),
scale(test$Broad_H3K9ac_percentage))
H3K9ac <- rowSums(scale(test$H3K9ac_height),H3K9ac )

H3K9me2 <-  rowSums(scale(test$H3K9me2_width),
scale(test$Broad_H3K9me2_percentage))
H3K9me2 <- rowSums(scale(test$H3K9me2_height),H3K9me2)
```

```
H3K79me2 <-  rowSums(scale(test$H3K79me2_width),
scale(test$Broad_H3K79me2_percentage))
H3K79me2 <- rowSums(scale(test$H3K79me2_height), H3K79me2)

H4K20me1 <-  rowSums(scale(test$H4K20me1_width),
scale(test$Broad_H4K20me1_percentage))
H4K20me1 <- rowSums(scale(test$H4K20me1_height), H4K20me1)

# remove all individual "group" variables from test.csv (columns 66-98)
# (class was 99th column, so store it in separate object, remove columns 66-99)
# then, add class back to end of test.csv after adding the 11 transformed predictors
class <- test$class
test <- test[,-66:-99]
test <- data.frame(test, H3K4me3, H3K4me2, H3K4me1, H3K36me3,
H3K27ac, H3K27me3, H3K9me3, H3K9ac, H3K9me2, H3K79me2, H4K20me1,
class)

# kept columns 1-65 and class, added 11 transformed variables = 77 columns in training.csv
# class variable (outcome variable) is now 77th column in test.csv
dim(test)

# convert class in test.csv into factor (LR requires categorical outcome variable)
test$class <- as.factor(test$class)

# make predictions on test.csv data, write.csv for last submission
set.seed(123)
LRpred_test <- predict(LRfit, test[, -77])
table(LRpred_test)

# predictions have 1259 for class 0 (NG), 54 for class 1 (OG), 50 for class 2 (TSG)

t_last_sub <- data.frame("id" = test$id, "class" = LRpred_test)
write.csv(t_last_sub, file = "t_last_submission.csv", row.names
= FALSE)
```