Stats 101C Midterm Project

Team MPGA
Crystal Li, Wenru Shi, Chongxuan Bi

Contents

1 Introduction

Healthcare holds infinite possibilities for mankind, and especially recent health issues have become a hot topic in the current society. The Big-Data tools provide a comprehensive discovery of potential cancer-driven genes, tumor suppressor genes (TSGs) and oncogenes (OGs). The OGs and TSGs work together to keep the balance between cell growth and apoptosis. However, when key genetic alterations accumulate, this balance is disrupted and cancer will happen.

Although general public recognize the importance of mutations, genetics, epigenetics, and phenotypes that make key alterations of our genes causing cancer, there is still unclear to figure out the specific individual contributors identifying and classifying potential features of cancer-related biological features. Here, we preprocessed data by excluding out the highly-correlated variables and conducting significance selection based on linear model. Then we tried K-Nearest Neighbors, Linear Discriminant Analysis and Logistic Regression models to fit, while we integrated diverse threshold setting methods. Meanwhile, we employed interaction between variables and removal of outliers in the selected version of variables. With builtin cross-validation, our score in the confusion matrix achieves 0.8236 and stay at that score even though we tried different approaches. It is default not to use lasso model and more detailed feature modification and selection.

2 Preprocessing

We preprocessed the data in 'training.csv'.

1) We removed the 'id' column because it was irrelevant and it was not a gene's inherent feature.

2) We converted the class column to a factor so that we can make classifications.

3) We split the data in 'training.csv' into 70% 'train' and 30%'valid' in order to perform cross validation later.

3 Feature Selection

3.1 Ideas from Paper

At the beginning, we read a lot of bioinformatics paper in order to figure out the possible relationship among mutation, phenotypes, epigenetics and genetics to identify tumor suppressor genes (TGSs) and oncogenes (OGs) from tumor DNA sequences.

As Devoli states in his paper, Cumulative Haploinsufficiency and Triplosensitivity Drive Aneuploidy Patterns to Shape the Cancer Genome, his group members state that LOF/Benign, Splice/Benign, Deletion frequency as well as Amplification frequency from mutations work efficiently in distinguishing the tsg genes and og genes.

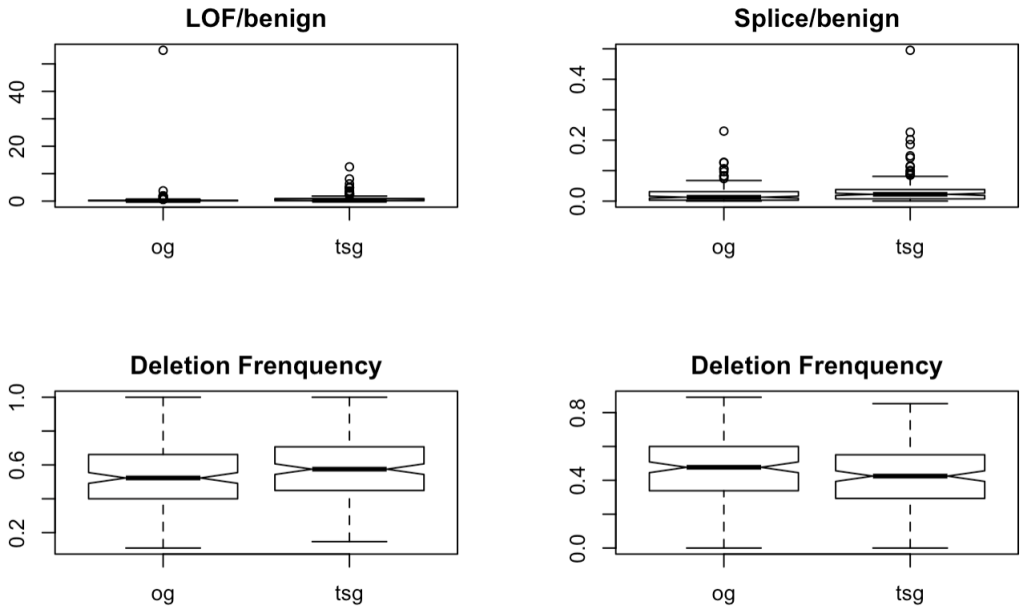

Figure 1: The Variables Suggested by Devoli's paper

However, when we applied into our own training data sets, there is no obvious difference between these variables in relation to the gene class. Thus, we can not employ those variables in our model.

Meanwhile, we also explore Jingyi Jessica Li's paper,DORGE: Discovery of Oncogenes and Tumor Suppressor Genes Using Genetic and Epigenetic Features, she highlights the importance of peak lengths of histone marks such as H3K4me3 peak length, H3K79me2, H3K36me3, H4K20me1, and H3K9ac, which functions as the initializer of gene transcription. In addition, VEST, Gene-body methylation canyons, and PolyPhen-2 score appears a huge differences of TSGs and OGs.

Based on all the paper resources, we tries our variables in three steps: group formation, removal of highly-correlated variables, and significance selection.

3.2 Group Formation and Removal of highly-correlated Variables

From our training dataset, we could observe that all of those variables belong to four different main groups: mutation, phenotypes, epigenetics and genetics. Thus, we firstly classified those variables into those four groups and ranked their significance levels in the linear model for future reference.

Moreover, we realized the high-correlated variables existing in those 96 variables. As a result, we firstly built a correlation matrix and then deleted correlation over 0.7 variables saving as the final variable. Since we have tried 0.6, 0.65 and 0.9 correlation selection criteria correspondingly, none of those performed better than our 0.7 correlation version in our confusion matrix) 0.7 criteria becomes our final criteria.

At first, we employed cor() R builtin function to get all the correlation among every variables. Then we converted the upper triangular matrix into 0 in order to avoid the potential repetition. Then, we select the rest of data with the criteria above 0.7

3.3 Significance Selection

Selecting non-highly-correlated variables, we tried the significance selection based on linear model, which further shrinks our variables group size to 23, which includes the importance ranks from *** to . in linear model.

	Silent_KB_Ratio	N_Missense	N_LOF	N_Splice	Missense_KB_Ratio	LOF_KB_Ratio
Silent_KB_Ratio	0.000000000	0.000000000	0.000000000	0.000000000	0.000000e+00	0.000000000
N_Missense	0.467483853	0.000000000	0.000000000	0.000000000	0.000000e+00	0.000000000
N_LOF	0.302485860	0.788043918	0.000000000	0.000000000	0.000000e+00	0.000000000
N_Splice	0.078194135	0.486902326	0.512669974	0.000000000	0.000000e+00	0.000000000
Missense_KB_Ratio	0.180859765	0.166012559	0.052843479	0.004265471	0.000000e+00	0.000000000
LOF_KB_Ratio	0.072165016	0.120525681	0.302207205	0.122728412	2.354388e-02	0.000000000
Missense_Entropy	0.313952997	0.377558050	0.197196104	0.103576382	4.030154e-01	0.056589912
LOF_TO_Silent_Ratio	-0.027778870	0.073335955	0.270986857	0.115024788	1.694744e-03	0.966250694
Splice_TO_Silent_Ratio	-0.187519209	-0.016111523	0.059939350	0.524172446	-1.539512e-02	0.088091990
Missense_TO_Silent_Ratio	0.112678061	0.216988644	0.082872561	0.030069887	9.328424e-01	0.034276429

	Missense_Entropy	LOF_TO_Silent_Ratio	Splice_TO_Silent_Ratio	Missense_TO_Silent_Ratio
Silent_KB_Ratio	0.000000e+00	0.0000000000	0.0000000000	0.0000000000
N_Missense	0.000000e+00	0.0000000000	0.0000000000	0.0000000000
N_LOF	0.000000e+00	0.0000000000	0.0000000000	0.0000000000
N_Splice	0.000000e+00	0.0000000000	0.0000000000	0.0000000000
Missense_KB_Ratio	0.000000e+00	0.0000000000	0.0000000000	0.0000000000
LOF_KB_Ratio	0.000000e+00	0.0000000000	0.0000000000	0.0000000000
Missense_Entropy	0.000000e+00	0.0000000000	0.0000000000	0.0000000000
LOF_TO_Silent_Ratio	2.047048e-02	0.0000000000	0.0000000000	0.0000000000
Splice_TO_Silent_Ratio	-1.741320e-02	0.1561329978	0.0000000000	0.0000000000
Missense_TO_Silent_Ratio	5.776099e-01	0.0242205673	0.0243206458	0.0000000000

	Missense_Damaging_TO_Missense_Benign_Ratio	LOF_TO_Benign_Ratio	Splice_TO_Benign_Ratio
Silent_KB_Ratio	0.000000000	0.000000000	0.000000000
N_Missense	0.000000000	0.000000000	0.000000000
N_LOF	0.000000000	0.000000000	0.000000000
N_Splice	0.000000000	0.000000000	0.000000000
Missense_KB_Ratio	0.000000000	0.000000000	0.000000000
LOF_KB_Ratio	0.000000000	0.000000000	0.000000000

Figure 2: The Variables Suggested by Devoli's paper

4 Model Fitting

Summary: After trials, we found that Logistic Regression Methods without using interactions term has the best fit result.

We have attempted three different classification methods to fit the data and predict result by using predictors (variables) we select from the previous section.

We used K Nearest Neighbors Model (KNN Method), Linear Discriminant Analysis (LDA Method), and Logistic Regression (LR Method) to predict variables.

From these three models, we found that Logistic Regression has the best predicted result compared to these of KNN Method and LDA Method. After trying out a bunch of interactions, LR predicted by 23 variables without interaction terms shows the best prediction result.

4.1 Trial 1: KNN Model

At the first attempt, we utilized KNN method to predict our validation test's results. With the caret library, we are able to use 10 folds cross validation to try KNN method from K = 1 to K = 100.

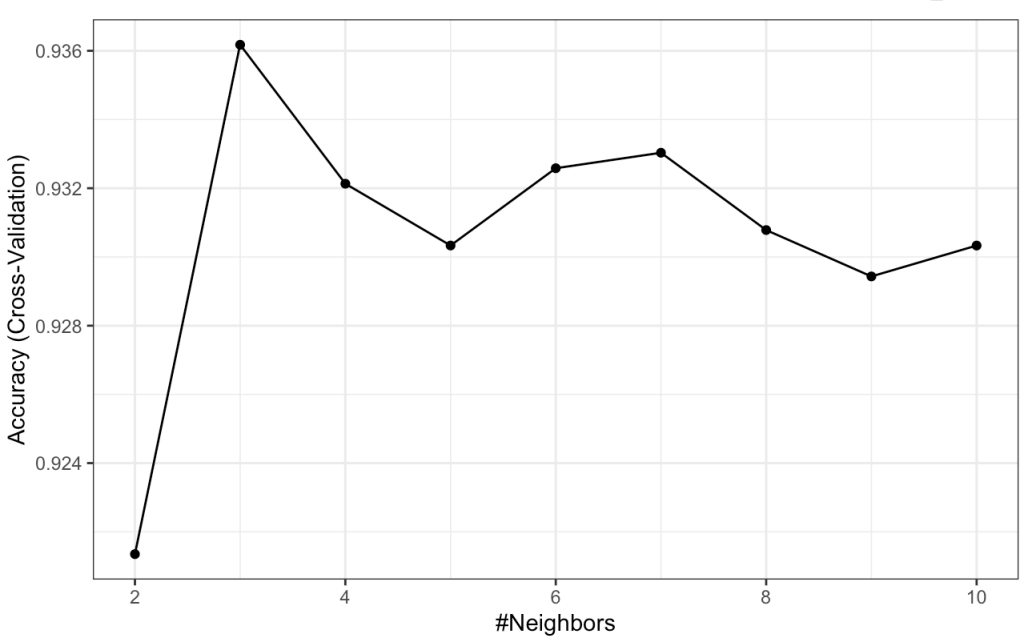

Figure 3: Choice of different K in KNN Model

Among all KNN methods from K = 1 to K = 100, the KNN method with K = 3, conducted with 23 variables selected from the previous section, presents the best predict result. However, this model only has an accuracy rate of 0.515

```
          Reference
Prediction  ng  og tsg
       ng  849  32  27
       og    1  13   5
       tsg   2   5  18
[1] 0.5150771
```

Figure 4: Trial 1 KNN Model with K = 3

At the same time it has a lot of wrong predictions of NGs. Thus, we progressed to our next model trial: Linear Discriminant Analysis.

4.2 Trial 2: LDA Model

In the second attempt, we used LDA method with 23 variables selected above. We still used package caret to conduct LDA with 10 folds cross validation. After fitting the model, we observed that the accuracy rate increases significantly compared with KNN model, which is around 0.646.

```
          Reference
Prediction  ng  og tsg
       ng  842  20  17
       og    1  23   6
       tsg   9   7  27
[1] 0.6458626
```

Figure 5: Trial 2 LDA Model

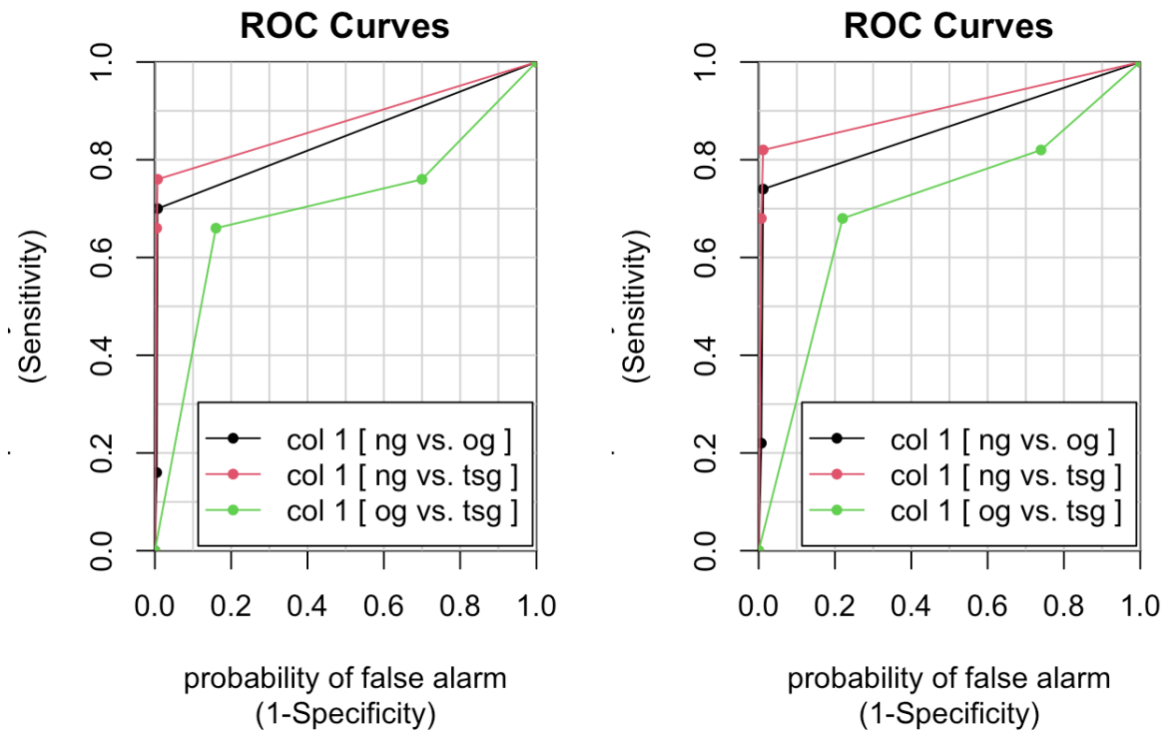

Figure 6: ROC Curve - KNN(Left) vs. LDA(Right)

From the ROC curve we can also observe that LDA has a better performance in detecting all three relationships: NG vs. OG, NG vs. TSG, OG vs. TSG.

However, even though the detection of OG and TSG increases compared to KNN model, we still have a large number of OG and TSG being detected as NG. Thus, we progressed to apply Logistic Regression Model.

4.3 Trial 3: Logistic Regression without Interaction

In the third attempt, we applied Logistic Regression Model (LR). By using function "multinom", we were able to predict results with three classifications. We got the following result with an accuracy rate of 0.683.

```
          Reference
Prediction  ng  og tsg
       ng  848  16  16
       og    0  28   7
       tsg   4   6  27
[1] 0.6830295
```

Figure 7: Trial 3 Logistic Regression

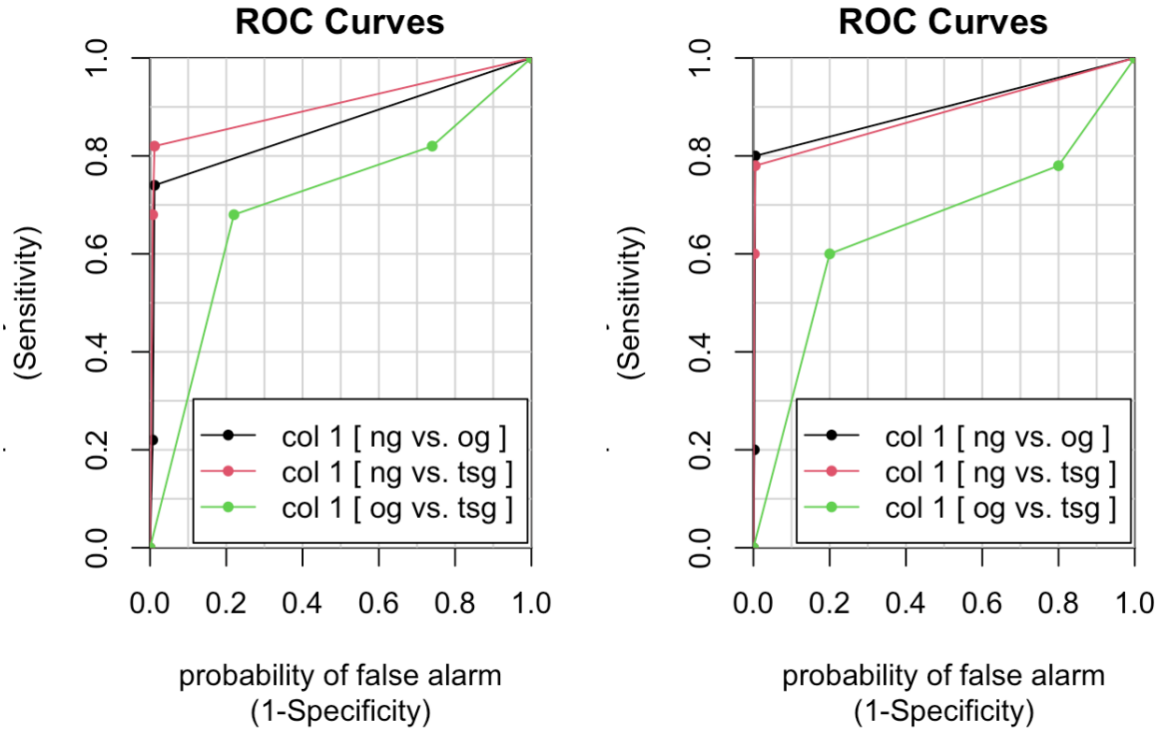

Figure 8: ROC Curve - LDA(Left) vs. LR(Right)

From the ROC Curve, we can observe that the performance in detecting NG vs. OG/TSG improves, while the performance of detecting OG vs. TSG decreases. It is vital to keep in mind the relationships between detecting OG/TSG from NG is important.

Thus, compared to KNN model and LDA model, LR indicates a better prediction in validation data. However, the same issue of high numbers of incorrect detection of OG and NG still occurred. Since we have tried out three possible classification methods, we progressed to consider the choice of interactions terms.

4.4 Trial 4: Logistic Regression with Interaction

To confirm that whether we should choose interaction terms, we adjusted the variables selected above. By observing boxplots and patterns, and reading related research papers, we believe that there exists certain interactions between some of those selected variable. For example, in employed Broad_H3K27me3_percentage* H3K27me3_width, percentage of broad H3K27me3 peaks in ENCODE samples are corresponding to H3K27me3 peak length, since the peak and percentage represent the importance of those cell line samples. Thus, we include following interactions terms in our LR model:

Missense_Entropy*N_Splice*Splice_TO_Total_Ratio
GDI*Missense_Damaging_TO_Benign_Ratio
S50_score_replication_timing*One_Minus_S50_score_replication_timing
Broad_H3K36me3_percentage* Broad_H3K4me1_percentage*H3K4me1_height
Broad_H3K27me3_percentage*H3K27me3_width

After fitting the LR model by including these interaction terms, we got the accuracy rate of prediction result of 0.736.

```
lrfit_pred  ng  og tsg
       ng  839  17  16
       og    3  30   1
       tsg  10   3  33
> 0.73597475
```

Figure 9: Trial 4 Logistic Regression with interaction terms

However, even though the accuracy rate improves from Trial 3 of LR model without interaction terms, its incorrect detection of OG and TSG as NG increases. In addition, consider about the overfitting issue of the model with interactions, LR model without interaction terms presents a better picture for our prediction.

4.5 Final Model: Logistic Regression

After applying KNN Model, LDA Model, and LR model, we found that LR predicted by 23 variables selected (Trial 3) without interactions shows the best prediction result. Thus, the model we selected is the Logistic Regression Model by using R's function "multinom".

```
          Reference
Prediction  ng  og tsg
       ng  848  16  16
       og    0  28   7
       tsg   4   6  27
[1] 0.6830295
```

Figure 10: Trial 3 Logistic Regression

5 Adjustments

5.1 Cross Validation

To adjust data analysis method and improve efficiency, we utilized 10-folds cross validation to test all three classification models: KNN, LDA, and LR.

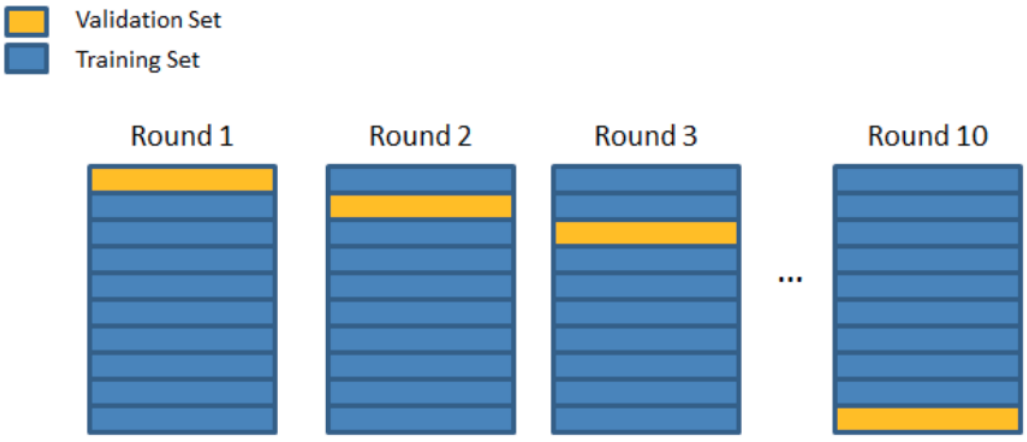

Figure 11: 10-fold Cross Validation. Source: Computer Vision and Machine Learning Projects and Tutorials

By using the caret library and its 'train' function, we were able to conduct 10-folds cross validation. We could observe from the ROC curve plot after Cross validation that the ROC curve indicates that LR still has a better performance even after cross validation as LR model has a high True Positive Rate and a low False Positive Rate.

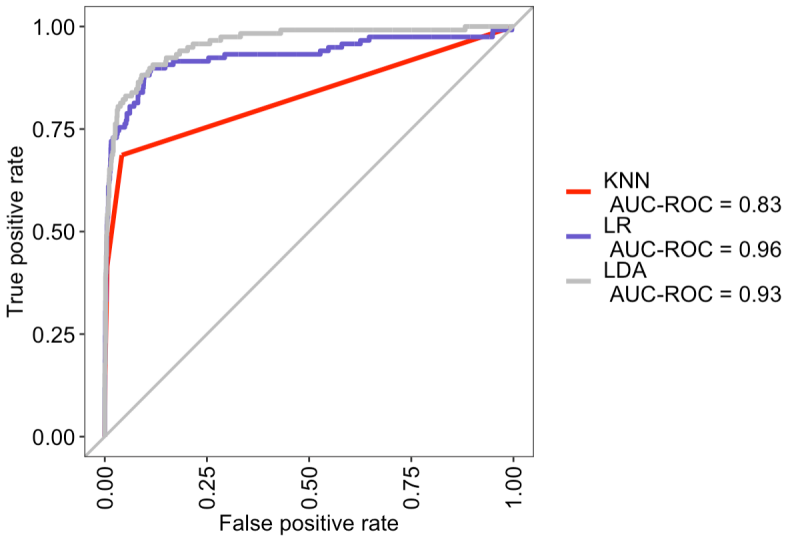

Figure 12: ROC Curve after Cross Validation

5.2 Threshold Setting

For each algorithm, we fit the model to the training data. Without threshold adjustments, the default algorithm chooses whichever class that appears most likely compared to the other two. Although this makes sense intuitively, this disproportionately results in predicting too many neutral genes(ng) and not enough tumor suppressor genes (tsg) and oncogenes (og).

As our priority is to correctly identify tumor suppressor genes and oncogenes, adjustment in threshold is necessary. By varying threshold parameters, we selected the best parameters based on Weighted Categorization Accuracy(WCA) of the validation data. We tried different thresholds for differentiating neutral genes from the other two but did not specifically set thresholds for classification between oncogenes and tumor suppressor genes. If the probability of a given observation being a neutral gene is greater than the threshold, then we classify it as a neutral gene. Otherwise, we classify it as either a oncogene or a tumor suppressor gene, whichever is more probable. On the one hand, we believed that by setting a high threshold, predicted neutral genes would almost always be real neutral genes and more observations would be classified as either oncogenes or tumor suppressor genes. Therefore, more genes of our interest would be identified. On the other hand, since the other two types of genes are equally important to us, we simply chose whichever that seemed more likely.

5.2.1 K-Nearest Neighbor Model

Below is the confusion matrix of the predictions that results from a K-Nearest Neighbor model with unmodified threshold. The first line shows that among predicted neutral genes, a large proportion of them are actually oncogenes or tumor suppressor genes. The last line shows the WCA score, which is only 0.515.

```
          Reference
Prediction  ng  og tsg
       ng  849  32  27
       og    1  13   5
       tsg   2   5  18
[1] 0.5150771
```

Figure 13: KNN No Threshold Adjustments

By varying the threshold,

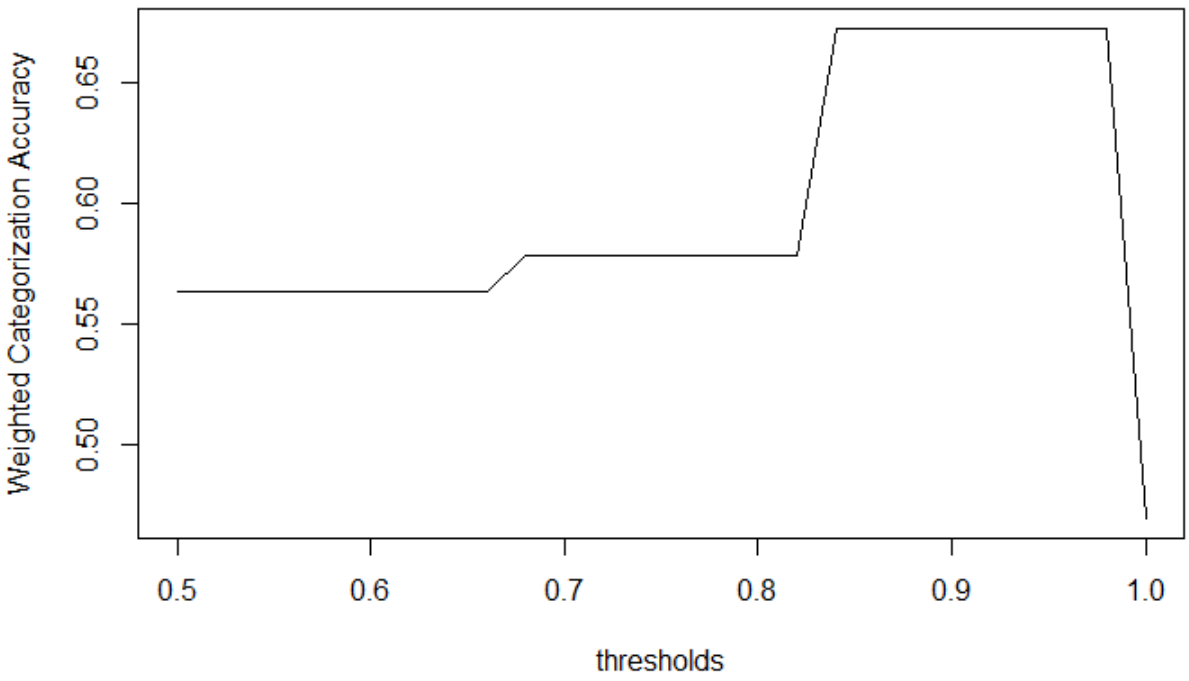

Figure 14: KNN Threshold Adjustments

Therefore, the K-Nearest Neighbor model with threshold of roughly 0.9 achieves the best classification outcome with WCA being over 0.65.

5.2.2 Logistic Regression Model

Below is the confusion matrix of the predictions that results from a Logistic Regression model with unmodified threshold. The first line shows that among predicted neutral genes, a large proportion of them are actually oncogenes or tumor suppressor genes. The last line shows the WCA score, which is only 0.683.

```
          Reference
Prediction  ng  og tsg
       ng  848  16  16
       og    0  28   7
       tsg   4   6  27
[1] 0.6830295
```

Figure 15: LR No Threshold Adjustments

By varying the threshold,

Therefore, the Logistic Regression model with threshold of roughly 0.96 achieves the best classification outcome with WCA being over 0.75.

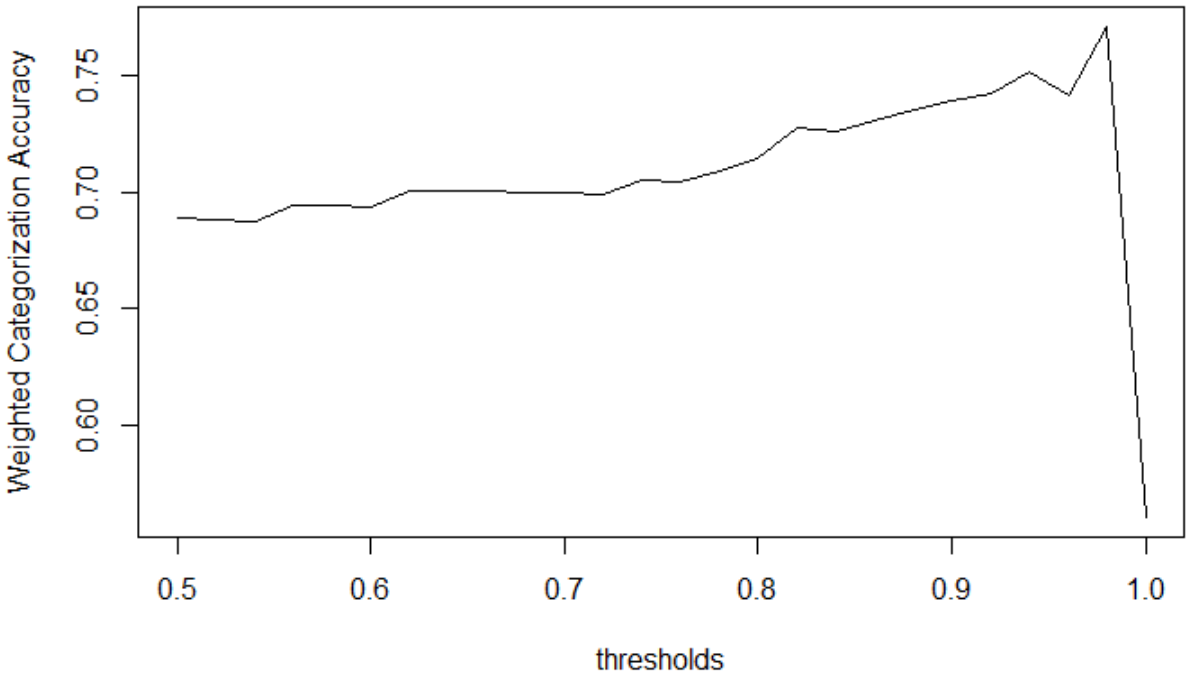

Figure 16: LR Threshold Adjustments

5.2.3 Linear Discriminant Analysis Model

Below is the confusion matrix of the predictions that results from a Logistic Regression model with unmodified threshold. The first line shows that among predicted neutral genes, a large proportion of them are actually oncogenes or tumor suppressor genes. The last line shows the WCA score, which is only 0.645.

```
          Reference
Prediction  ng  og tsg
       ng  842  20  17
       og    1  23   6
       tsg   9   7  27
[1] 0.6458626
```

Figure 17: LDA No Threshold Adjustments

By varying the threshold,

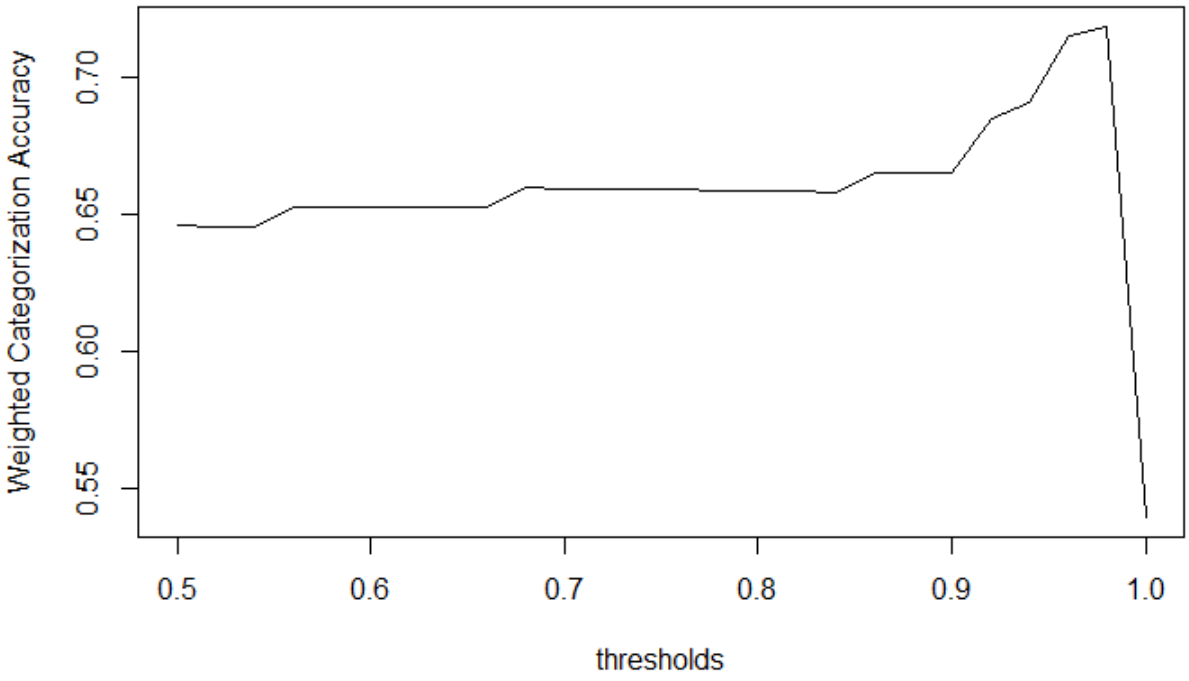

Figure 18: LDA Threshold Adjustments

Therefore, the Linear Discriminant Analysis model with threshold of roughly 0.96 achieves the best classification outcome with WCA being over 0.71.

5.2.4 Logistic Regression Model with Interaction Terms

After we choose Logistic Regression as our best model, we tried adding interaction terms. Below is the confusion matrix of the predictions that results from a Logistic Regression model(with interaction terms) with unmodified threshold. The first line shows that among predicted neutral genes, a large proportion of them are actually oncogenes or tumor suppressor genes. The last line shows the WCA score, which is only 0.736.

```
          Reference
Prediction  ng  og tsg
       ng  839  17  16
       og    3  30   1
       tsg  10   3  33
[1] 0.7359748
```

Figure 19: LR No Threshold Adjustments

By varying the threshold,

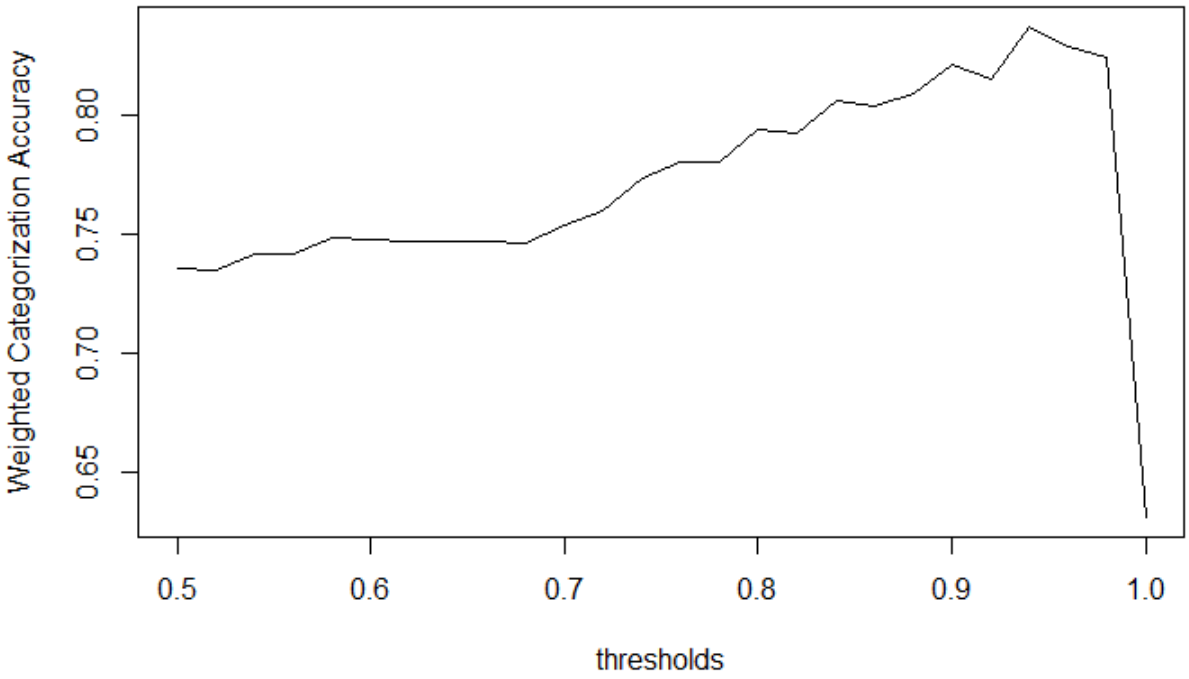

Figure 20: LR Threshold Adjustments

Therefore, the Logistic Regression model with threshold of roughly 0.94 achieves the best classification outcome with WCA being over 0.8.

5.2.5 Logistic Regression Model with Scaled & Logged Probabilities

We also tried scaling and logging the probabilities. Below is a plot showing thresholds of scaled and logged probabilities and corresponding WCAs.

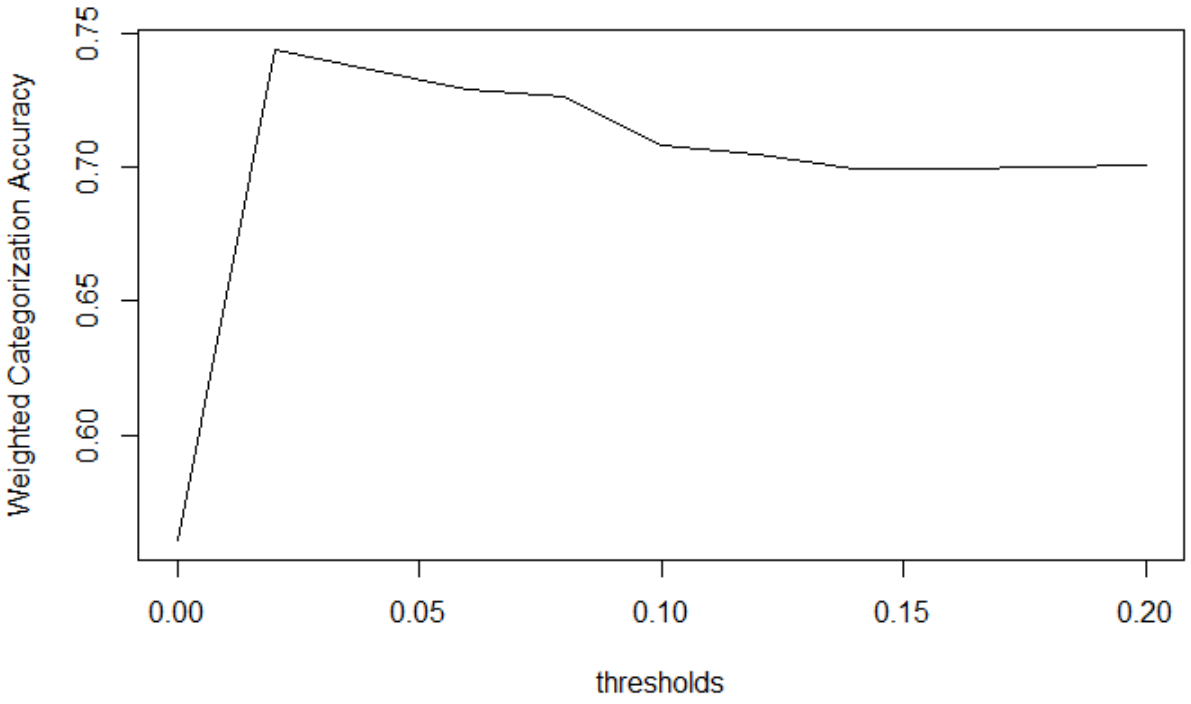

Figure 21: LR Scaled Probabilities

Therefore, the Logistic Model with threshold of 0.02 achieves the best classification outcome with WCA being about 0.75. However, after playing with the test data and adjusting the thresholds, we found that 0.08 also performed well with the test data. Therefore, we chose 0.08 as our final threshold.

5.3 Outliers

We were also concerned about the outliers effects on our overall predictions. Thus, we approached this from two ways.

After selecting the variables classified as the outliers by (0.025,0.975), we observed certain observations appear as multiple-outlier in diverse variables. Handling such a matter, we deleted those observations. On the other hand, most of those outlier observations are single-outlier. As a result, we decided to use the median of that variable to replace that value of the outlier feature of the single data. Even though we thought that KNN could be a good application in substituting potential outliers, there were time shortage and difficulties getting in our way. Thus, we ended up with median replacing method, which did not yield a significantly better result before the outlier removal.

6 Limitations

Although we achieved prediction score of around 82% based on Weighted Categorization Accuracy on the public leaderboard, we still have weaknesses in our approach.

1) In our feature selection process, due to the large size of predictors, we filtered predictors only based on their p-values of individual effects without considering important p-values of interactions. This could lead to neglecting important interaction effects and thus deteriorating the model's predictive ability.

2) Also in the feature selection process, we selected predictors based on linear model and removed highly correlated ones to alleviate multicollinearity. We deemed that our selected predictors are the best for all models. However, each model could have its own best selection of variables, and excluding those predictors,

which are not significant with linear model but might be significant when other models are used, could damage the prediction outcomes.

3) We tried to resolve close probabilities after fitting logistic regression model by applying KNN model again to double check (e.g. one class with probability 0.46 and another with 0.47), but the approach was not effective.

4) We tried first classifying genes into neutral genes and harmful ones, and then classifying harmful ones into oncogenes and tumor suppressor genes, but the approach was not effective.

5) We did not successfully apply Quadratic Discriminant Analysis due to multicollinearity error that it reported, even after we had selected a limited number of predictors.

7 Advantages of the Model

Our final model of logistic regression with scaled/logged probabilities and controlled thresholds performs well predicting the test data and achieves a score of 81.315% on the public leaderboard. Besides its predictive capability, it is also a simplified model based on only 23 predictors, which dramatically enhances the model's interpretability. With significantly less variables than the original training data with nearly 100, the model also achieves a relative bias-variance tradeoff balance by sacrificing a small amount of accuracy for a large amount of variance reduction. We also scaled and logged the probabilities to deal with issues that stem from the distribution of probabilities. Moreover, we applied threshold adjustments in order to give different classes different weighted importance. This manipulation allows us to more precisely identify and recognize harmful genes. Not only does this model predict well on the training data, it also outperforms other models including KNN, LDA, LR with interactions, and LR combined with KNN. Comparing different models validates our final model's performance and reliance.

In summary, with our multiple intricately tuned parameters and outcome performance measurements, the model accomplishes accurate predictions without sacrificing its interpretability, and reaches a score of 81.315%.

8 Conclusion

Employing all the listed methods above, our group achieve a 81.315% score on the public leader board. Since we combine several paper suggestions with self-made high-correlated variables and thus select final importance-related variables, we dive further into model selection. Scanning through KNN, LDA, Logistic Regression, we employ cross-validation and threshold settings to compute WCA score. It is important to realize the intricate threshold settings yielding diverse results, which includes scale/log method and OG-TST first method. While we do consider the impact from outliers and interaction hidden in certain variables, the removal of outliers and adding interaction terms do not improve our score any better. As a result, there are limitations in our approach as well, such as employing linear model on selected data and neglecting QDA analysis. Facing skyrocketing-high health demand nowadays, scientists compel to avoid the threats posed by cancer. Applying statistics, and Big data to bioinformatics field, they find out more related features from phenotypes, genetics, and epigenetics, the bedrock of solutions to future cancer issues.

9 Reference

[1] Davoli, T., Xu, A. W., Mengwasser, K. E., Sack, L. M., Yoon, J. C., Park, P. J., Elledge, S. J. (2013). Cumulative haploinsufficiency and triplosensitivity drive aneuploidy patterns and shape the cancer genome. Cell, 155(4), 948–962. https://doi.org/10.1016/j.cell.2013.10.011

[2] Esteller M. (2006). Epigenetics provides a new generation of oncogenes and tumour-suppressor genes. British journal of cancer, 94(2), 179–183. https://doi.org/10.1038/sj.bjc.6602918

10 Statement of Contributions

Crystal Li worked on feature selection and outliers.

Wenru Shi worked on model fitting and cross validation.

Chongxuan Bi worked on adjusting thresholds and comparing resulting models.

11 Code Appendix

```
# Import data
library(readr)
data = read_csv("training.csv")
test = read_csv("test.csv")
data = data[,-1]
test = test[,-1]


# Import libraries
library(MASS)
library(caret)
library(MLeval)
library(nnet)
library(glmnet)
library(readr)

# Convert class to factors:0 is ng, 1 is og, 2 is tsg
data$class = as.factor(data$class)
levels(data$class) = c('ng', 'og', 'tsg')

# Train/validation data split
set.seed(1)
train_index = createDataPartition(data$class, p = 0.7, list = F)
train = data[train_index,]
valid = data[-train_index,]

# Feature Selection
# Remove Collinear Variables for the training data
correlation_matrix = cor(train[,-98])
hc = findCorrelation(correlation_matrix,cutoff = 0.7)
hc = sort(hc)
reduced_train = train[,-c(hc)]

# Trial 1: Fit KNN model (caret) with threshold 0.5
train_control <- trainControl(method="cv", number = 5, classProbs = TRUE,
savePredictions = TRUE)
knnfit = train(class ~
                    Silent_KB_Ratio+
                    N_Splice+
                    Missense_Entropy+
                    Splice_TO_Total_Ratio+
                    Silent_fraction+
                    Recurrent_missense_fraction+
                    Inactivating_mutations_fraction+
                    MGAentropy+VEST_score+
                    Cell_proliferation_rate_CRISPR_KD+
                    Super_Enhancer_percentage+
                    BioGRID_log_degree+
                    Gene_body_hypermethylation_in_cancer+
                    Canyon_genebody_hypermethylation+
                    intolerant_pLI+
                    Synonymous_Zscore+
                    GDI+ncGERP+
                    Broad_H3K9me3_percentage+
                    Broad_H3K9ac_percentage+
                    Broad_H3K9me2_percentage+
```

```
                    H4K20me1_height+
                    S50_score_replication_timing, data = train, method = 'knn', preProc = c('cent
trControl = train_control, tuneGrid = expand.grid(k = seq(1, 100, 5)))

knn_pred = predict(knnfit, valid, type = 'prob')
knn_pred_no_prob = predict(knnfit, valid)
confusionMatrix(knn_pred_no_prob, valid$class)

# Manually adjust threshold
knn_pred_manual = ifelse(knn_pred_prob$ng == 1, 'ng',
ifelse(knn_pred_prob$og > 0.6 | knn_pred_prob$tsg == 0, 'og', 'tsg'))
knn_pred_manual = as.factor(knn_pred_manual)
levels(knn_pred_manual) = c('ng', 'og', 'tsg')
confusionMatrix(knn_pred_manual, valid$class)

# Trial 2: Fit LDA Model (caret) with manually adjusted threshold

ldafit = lda(class ~
                    Silent_KB_Ratio+
                    N_Splice+
                    Missense_Entropy+
                    Splice_TO_Total_Ratio+
                    Silent_fraction+
                    Recurrent_missense_fraction+
                    Inactivating_mutations_fraction+
                    MGAentropy+VEST_score+
                    Cell_proliferation_rate_CRISPR_KD+
                    Super_Enhancer_percentage+
                    BioGRID_log_degree+
                    Gene_body_hypermethylation_in_cancer+
                    Canyon_genebody_hypermethylation+
                    intolerant_pLI+
                    Synonymous_Zscore+
                    GDI+ncGERP+
                    Broad_H3K9me3_percentage+
                    Broad_H3K9ac_percentage+
                    Broad_H3K9me2_percentage+
                    H4K20me1_height+
                    S50_score_replication_timing,data = train)
ldafit_pred = predict(ldafit, valid)
ldafit_pred_prob = predict(ldafit, valid, type = 'prob')
ldafit_pred_manual = rep(NA, 953)
ldafit_pred_manual = ifelse(ldafit_pred_prob$posterior[,1] >= 0.9, 0, as.numeric(apply(ldafit_pr
table(ldafit_pred_manual, valid$class)

# Trial 3: Fit Basic LR models with manually adjusted threshold

lrfit = multinom(class ~
                    Silent_KB_Ratio+
                    N_Splice+
                    Missense_Entropy+
                    Splice_TO_Total_Ratio+
                    Silent_fraction+
                    Recurrent_missense_fraction+
                    Inactivating_mutations_fraction+
                    MGAentropy+VEST_score+
                    Cell_proliferation_rate_CRISPR_KD+
                    Super_Enhancer_percentage+
                    BioGRID_log_degree+
```

```
                    Gene_body_hypermethylation_in_cancer+
                    Canyon_genebody_hypermethylation+
                    intolerant_pLI+
                    Synonymous_Zscore+
                    GDI+ncGERP+
                    Broad_H3K9me3_percentage+
                    Broad_H3K9ac_percentage+
                    Broad_H3K9me2_percentage+
                    H4K20me1_height+
                    S50_score_replication_timing,data = train)
lrfit_pred_prob = predict(lrfit, valid, type = 'prob')
lrfit_pred_manual = rep(NA, 953)
lrfit_pred_manual = ifelse(s_lrfit_pred_prob[,1] >= 0.94, 0,
as.numeric(apply(s_lrfit_pred_prob [,2:3], 1, which.max)))
print(table(lrfit_pred_manual, valid$class))



# Trial 4: Fit LR models with interaction terms & manually adjusted threshold

lrfit = multinom(class ~
                    BioGRID_betweenness+Silent_KB_Ratio+
                    Silent_fraction+
                    Missense_Entropy*N_Splice*Splice_TO_Total_Ratio+MGAentropy+
                    GDI*Missense_Damaging_TO_Benign_Ratio+
                    Recurrent_missense_fraction+
                    Inactivating_mutations_fraction+
                    Cell_proliferation_rate_CRISPR_KD+
                    VEST_score+
                    Gene_body_hypermethylation_in_cancer+
                    Canyon_genebody_hypermethylation+
                    Super_Enhancer_percentage+intolerant_pLI+
                    Synonymous_Zscore+
                    ncGERP+
                    S50_score_replication_timing*One_Minus_S50_score_replication_timing+
                    BioGRID_log_degree+
                    H4K20me1_height+
                    Broad_H3K79me2_percentage+
                    Broad_H3K36me3_percentage*
                    Broad_H3K4me1_percentage*H3K4me1_height+
                    Broad_H3K27ac_percentage+
                    Broad_H3K27me3_percentage*H3K27me3_width+
                    H3K4me3_height+
                    LOF_TO_Benign_Ratio, data = train)
lrfit_pred = predict(lrfit, valid)
lrfit_pred_prob = predict(lrfit, valid, type = 'prob')
lrfit_pred_manual = rep(NA, 953)
lrfit_pred_manual = ifelse(lrfit_pred_prob[,1] >= 0.94,
0, as.numeric(apply(lrfit_pred_prob[,2:3], 1, which.max)))
table(lrfit_pred_manual, valid$class)

# Additional Trial: Fit LR models with manually adjusted threshold accompanied by KNN method

levels(knn_pred) = c(0,1,2)
knn_pred = as.numeric(knn_pred) - 1
lrfit_pred_manual = rep(NA, 952)
  for(i in 1:nrow(valid)){
    if(lrfit_pred_prob[i,1] >= t){
      lrfit_pred_manual[i] = 0
```

```
    } else if (lrfit_pred_prob[i,1]>= 0.90 && lrfit_pred_prob[i,1] < t){
      if(abs(lrfit_pred_prob[i,2]-lrfit_pred_prob[i,3]) <= 0.05){
        lrfit_pred_manual[i] = 0
      } else {
        lrfit_pred_manual[i] = as.numeric(which.max(lrfit_pred_prob [i,2:3]))
        if(as.numeric(which.max(lrfit_pred_prob [i,2:3]))!= as.numeric(which.max(ldafit_pred[i,2
          diff_lr = abs(lrfit_pred_prob [i,2]-lrfit_pred_prob [i,3])
          diff_lda = abs(ldafit_pred[i,2]-ldafit_pred[i,3])
          lrfit_pred_manual[i] = ifelse(diff_lr>diff_lda,as.numeric(which.max(lrfit_pred_prob [i
        }
      }
    } else if (lrfit_pred_prob[i,1]>= 0.70 && lrfit_pred_prob[i,1] < 0.90 ){
      if(abs(lrfit_pred_prob[i,2]-lrfit_pred_prob[i,3]) > 0.05){
        lrfit_pred_manual[i] = as.numeric(which.max(lrfit_pred_prob [i,2:3]))
      } else {
        lrfit_pred_manual[i] = 0
      }
    } else if (lrfit_pred_prob[i,1] < 0.7) {
      if(abs(lrfit_pred_prob[i,2]-lrfit_pred_prob[i,3]) > 0.05){
        lrfit_pred_manual[i] = as.numeric(which.max(lrfit_pred_prob [i,2:3]))
      } else {
        lrfit_pred_manual[i] = as.numeric(knn_pred[i])
      }
    }
  }
  print(table(lrfit_pred_manual, valid$class))

# Final Model: Fit LR models with scaling prediction probabilities & manually adjust threshold

lrfit = multinom(class ~
                   Silent_KB_Ratio+
                   N_Splice+
                   Missense_Entropy+
                   Splice_TO_Total_Ratio+
                   Silent_fraction+
                   Recurrent_missense_fraction+
                   Inactivating_mutations_fraction+
                   MGAentropy+VEST_score+
                   Cell_proliferation_rate_CRISPR_KD+
                   Super_Enhancer_percentage+
                   BioGRID_log_degree+
                   Gene_body_hypermethylation_in_cancer+
                   Canyon_genebody_hypermethylation+
                   intolerant_pLI+
                   Synonymous_Zscore+
                   GDI+ncGERP+
                   Broad_H3K9me3_percentage+
                   Broad_H3K9ac_percentage+
                   Broad_H3K9me2_percentage+
                   H4K20me1_height+
                   S50_score_replication_timing,data = train)

lrfit_pred_prob = predict(lrfit, valid, type = 'prob')
s_lrfit_pred_prob = scale(abs(log(lrfit_pred_prob)),center = F)
lrfit_pred_manual = rep(NA, 953)
threshold = 0.08
lrfit_pred_manual = ifelse(s_lrfit_pred_prob[,1] <= threshold,
0, as.numeric(apply(s_lrfit_pred_prob [,2:3], 1, which.min)))
print(table(lrfit_pred_manual, valid$class))
```

```
# Cross Validation

LRfit_all <- train(class ~
                    Silent_KB_Ratio+
                    N_Splice+
                    Missense_Entropy+
                    Splice_TO_Total_Ratio+
                    Silent_fraction+
                    Recurrent_missense_fraction+
                    Inactivating_mutations_fraction+
                    MGAentropy+VEST_score+
                    Cell_proliferation_rate_CRISPR_KD+
                    Super_Enhancer_percentage+
                    BioGRID_log_degree+
                    Gene_body_hypermethylation_in_cancer+
                    Canyon_genebody_hypermethylation+
                    intolerant_pLI+
                    Synonymous_Zscore+
                    GDI+ncGERP+
                    Broad_H3K9me3_percentage+
                    Broad_H3K9ac_percentage+
                    Broad_H3K9me2_percentage+
                    H4K20me1_height+
                    S50_score_replication_timing,
data = train, method = "multinom",
preProc = c("center", "scale"), trControl = train_control)

lrfit_cv_pred = predict(LRfit_cv, valid, type = 'prob')
s_lrfit_cv_pred = scale(abs(log(lrfit_cv_pred)))
lrfit_cv_pred_manual = rep(NA, 953)
threshold = 0.08
lrfit_pred_manual =
ifelse(s_lrfit_cv_pred[,1] <= x, 0, as.numeric(apply(s_lrfit_cv_pred [,2:3], 1, which.min)))
print(table(lrfit_cv_pred_manual, valid$class))

#Ourlier Removal
p<-"class␣~Silent_KB_Ratio+N_Splice+Missense_Entropy+Splice_TO_Total_Ratio+Silent_fraction
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣+Recurrent_missense_fraction+Inactivating_mutations_fraction+MGAentropy+VEST_sc
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣+Cell_proliferation_rate_CRISPR_KD+Super_Enhancer_percentage+BioGRID_log_degree
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣+Gene_body_hypermethylation_in_cancer+Canyon_genebody_hypermethylation+intolera
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣+Synonymous_Zscore+GDI+ncGERP+Broad_H3K9me3_percentage+Broad_H3K9ac_percentage
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣+Broad_H3K9me2_percentage+H4K20me1_height
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣+S50_score_replication_timing"
library(stringr)
m<-str_extract_all(p, "\\w+")
f<-m[[1]]
outllier_finder<-data.frame(data[,f])

for( i in 2: dim(outllier_finder)[2]){
current<-outllier_finder[,i]
lower_bound <- quantile(current, 0.025)
upper_bound <- quantile(current, 0.975)
outlier_ind <- which(current < lower_bound | current > upper_bound)

me<-median(current[-outlier_ind])
print(me)
for(j in 1:length(outlier_ind)){
   outllier_finder[outlier_ind[j],i]<-me
```

```
  }
}

set.seed(1)
train_index = createDataPartition(data$class, p = 0.7, list = F)
train_outlier = outllier_finder[train_index,]
valid_outlier = outllier_finder[-train_index,]

# Fit to whole data and predict on test

lrfit_test = multinom(class ~
                    Silent_KB_Ratio+
                    N_Splice+
                    Missense_Entropy+
                    Splice_TO_Total_Ratio+
                    Silent_fraction+
                    Recurrent_missense_fraction+
                    Inactivating_mutations_fraction+
                    MGAentropy+VEST_score+
                    Cell_proliferation_rate_CRISPR_KD+
                    Super_Enhancer_percentage+
                    BioGRID_log_degree+
                    Gene_body_hypermethylation_in_cancer+
                    Canyon_genebody_hypermethylation+
                    intolerant_pLI+
                    Synonymous_Zscore+
                    GDI+ncGERP+
                    Broad_H3K9me3_percentage+
                    Broad_H3K9ac_percentage+
                    Broad_H3K9me2_percentage+
                    H4K20me1_height+
                    S50_score_replication_timing,data = data)

lrfit_pred_prob_test = predict(lrfit_test, test, type = 'prob')
s_lrfit_pred_prob_test = scale(abs(log(lrfit_pred_prob_test)),center = F)
lrfit_pred_manual_test = rep(NA, 953)
threshold = 0.08
lrfit_pred_manual_test = ifelse(s_lrfit_pred_prob_test[,1] <= threshold,
0, as.numeric(apply(s_lrfit_pred_prob_test [,2:3], 1, which.min)))
print(table(lrfit_pred_manual_test, valid$class))


# Generate CSV file
result= data.frame("id" = c(3178:4540), "class"= as.numeric(lrfit_pred_manual_test),
row.names = NULL)

result_csv = write.csv(result, file = "result_test.csv",row.names = FALSE)
```