# STATS 101C - Midterm Report

Red Team (Aaron Radparvar, Darren Tsang, Jingwei Liang, Yonatan Khalil), Lecture 4

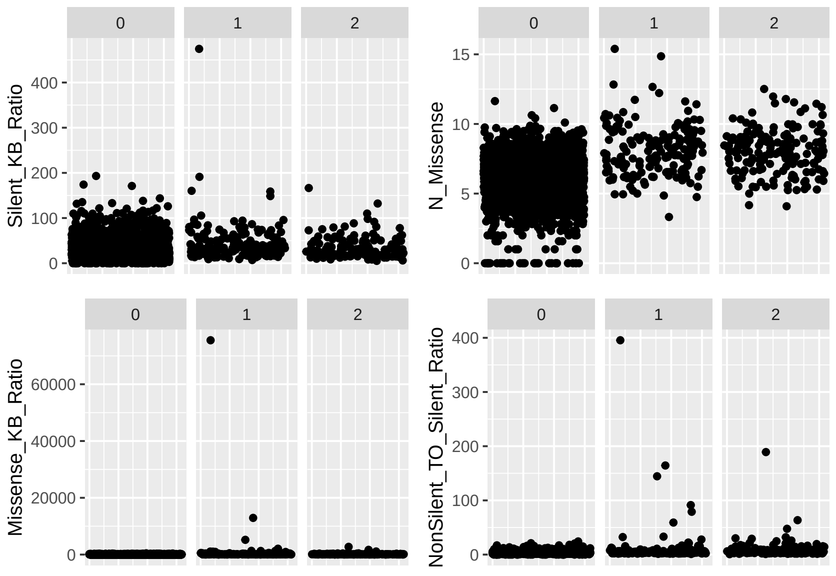
Produced on Monday, Nov. 09 2020 @ 11:28:02 PM

## 1 Introduction

Cancer is a category of diseases present in all biological life caused by when cells divide uncontrollably and spread into surrounding tissues. Cancer is caused by changes to DNA. Most cancer-causing DNA changes occur in sections of DNA called genes. These genes will sometimes be prone to mutations, which will then often cause permanent changes to its comprising DNA structures. Mutations can either be inherited or be caused due to exposure to certain materials, such as long-term exposure to unstable and radioactive elements and materials or ingestion of carcinogenic materials. The discovery of cancer driver genes, including oncogenes (OGs) and tumor suppressor genes (TSGs), is vital for detection, diagnosis, and treatment of cancer.

In this Kaggle project, we are given a training dataset of 3177 NGs, OGs, and TSGs in total. We are also given 97 properties associated with each of the 3177 genes. We are limited to using the following four methods: logistic regression (LR), k-nearest neighbors (kNN), linear discriminant analysis (LDA), and quadratic discriminant analysis (QDA). Our goal is to use the training dataset to train a model (or a combination of models) to identify which genes are NG, OG, or TSG. With the trained model, we put it to the test on the testing dataset, which has the same properties as the training dataset, but without the class. The evaulation metric used to evaluate the goodness of a model was a weighted categorization average (WCA); predicting an NG, OG, TSG correctly is worth 1, 20, 20 points respectively.

To begin the process, it is always a good idea to perform some exploratory data analysis (EDA) to get a better sense of how the data looks like. Below are some plots that show a couple of predictors, which are split by their class.



## 2 Methodology

Our team initially tackled this classification problem with a rather naive approach; we chose our predictors by taking the 20, 32, 64, etc. most correlated predictors with class and used LDA with these different set of predictors. This approach yielded us with surprisingly high WCA scores of ~.73 on the Kaggle public leaderboard. In fact, as it stands currently (November 9, 5 PM PST), a WCA score of .73 would be 16th place on the the public leaderboard. Our team used this as a starting point for future methods and adjustments, as we were highly aware that much improvements could be made.

First, our team created the find_pred() function. In short, find_pred() takes in a potential set of predictors and removes predictors such that the correlation between all predictors were less than a threshold set by us. Furthermore, creating a function like this not only allowed us to quickly remove highly correlated predictors, but also easily experiment with different correlation thresholds. This was pivotal in removing the problem with multicollinearity we saw in our initial naive approach. Please note that the implementation and a more detailed explanation of our find_pred() function can be found in the Appendix. The complete list of predictors used can also be found in the Appendix.

Second, we dealt with the unbalanced nature of our training data, which contained 2840 NGs, 168 OGs, and 169 TSGs. It is very clear that the amount of NGs greatly outnumber the amount OGs and TSGs, which would lead to any model to overfit. To deal with this, we created duplicates of the OGs and TSGs so that their count would be closer to NGs'. This essentially has the effect of putting more weights on the OGs and TSGs. Please note that this process can be found in the Appendix under the section titled *Training LDA to identify if NG or not.*

Third, from our initial EDA, we noticed that there were some variables where the OGs and TSGs were clearly distinguishable from the NGs. We can see such example in the variable *N_Missenese*, where there are no OGs or TSGs that have a value less than 2.5. To use this finding to our advantage, we decided to go with a two model approach; the first model would identify the NGs and the second model would identify everything that was not identified as NG earlier as either OG or TSG.

Fourth, we decided to preform cross-validation when training our models. This is to prevent our model from simply overfitting and remembering our training dataset.

We used the 4 techniques described above and tried a combination of LR, kNN, LDA, and QDA. Our best results are discussed in the following section, which is conveniently titled *Results.*

## 3 Results

Using LDA first and QDA second yielded our best WCA score of .81069 on the Kaggle leaderboard. Furthermore, we experimented with different values of correlation thresholds and found that .8 and .7 worked best for the first and second correlation threshold respectively.

Below is the confusion matrix using our best method on the training data and comparing it with the actual class. We can see that the most frequent misclassification occurs in the 0 column, but we can live with this because predicting an NG incorrectly is not as bad as predicting OG or TSG incorrectly.

| | | Actual | | |
|---|---|---|---|---|
| | | 0 (NG) | 1 (OG) | 2 (TSG) |
| Predicted | 0 (NG) | 2606 | 17 | 17 |
| | 1 (OG) | 77 | 121 | 9 |
| | 2 (TSG) | 157 | 30 | 143 |

Below is a table of our predictions on the testing data using our best method (which is what we have submitted to Kaggle):

| 0 (NG) | 1 (OG) | 2 (TSG) |
|---|---|---|
| 1126 | 74 | 163 |

We would like to note that various combinations of LR, kNN, LDA, and QDA had a pretty wide range of WCA scores from the Kaggle public leaderboard. We got WCA scores ranging from ~.43 to ~.81 (which was our best).

## 4 General Conclusions

Our highest WCA score on the Kaggle public leaderboard was .81069. This was obtained by using the following method: using LDA to identify the NGs from the OGs and TSGs, then using QDA to differentiate between OGs and TSGs.

We believe that this two model approach worked well because through our EDA, we discovered that it was easier to distinguish NG from the other two. Then, since the NGs were out of the picture after the first model, our remaining task was to take the "leftovers" and identify them as OG or TSG. Another reason why we believe our method worked well was the fact that we dealt with multicollinearity. The fact that multicollinearity did not exist among our selected predictors allowed our model to better learn from our predictors without having interference from other predictors.

On the other hand, there are a couple of potential areas where we could have improved our method. For example, creating duplicates of OGs and TSGs may have led to OGs and TSGs having too much weight. This would have caused more NGs to be identified to be either OG or TSG. Though, this may not have caused too much damage to our model since NGs have a much lower point value than OGs and TSGs. (NGs were 1 point if correct identified, while OGs and TSGs were 20 points if correctly identified.) Another way to improve our method is if we removed outliers among NGs, OGs, and TSGs during the training process. This may help our model in correctly identifying classes by not having to worry about extreme values of the predictors used.

Overall, we are happy with our results, but at the same time, we acknowledge that there are flaws in our approach that could have been improved given more time and a wider range of methods.

## Appendix

The code below was used to generate our best WCA score on the Kaggle public leaderboard.

### Loading caret and setting up cross validation for later

```r
library(caret)

train_control <- trainControl(method="cv", number = 5,
                              classProbs = TRUE,
                              savePredictions = TRUE)
```

### find_pred() function

find_pred() is a function that takes in *df*, which is a correlation matrix, *check*, which is a list of potential predictors sorted from most to least correlated with class, and *correlation*, which is the maximum of correlation allowed between each predictor. find_pred() returns a subset of *check*, where each predictor's correlation value with the other predictors is less than *correlation*.

```r
find_pred <- function(df, check, correlation){
  # include all initially
  include <- rep(T, length(check))
  for (i in seq(1, length(check))){
    cur_col = check[i]
    if(include[i]){
      # check which columns have higher correlation than cur_col
      a <- (abs(unname(df[, cur_col])) > correlation)
      # check which columns in a also are in check
      in_both <- intersect(check, names(df[a, cur_col]))
      if(!is.null(in_both)){
        # get check's position that contains each element of in_both
        pos <- match(in_both, check)
        for (j in pos){
          if(i < j){
            # set to false (not including)
            include[j] = F
          }
        }
      }
    }
  }

  return(check[include])
}
```

```r
# read in file
initial_data <- read.csv("ucla-stats101c-lec4/training.csv")

# change 0, 1, 2 to NG, OG, TSG, respectively
initial_data$class <- ifelse(initial_data$class == 0, "NG",
                             ifelse(initial_data$class == 1, "OG", "TSG"))
```

```
# find correlation of predictors, do not include id and class
cor_matrix <- cor(initial_data[, -c(1, 99)])
```

### Training LDA to identify if NG or not

```
# make a copy of initial_data
first_dataset <- initial_data

# convert NG to 0 and else to 1
first_dataset$class <- ifelse(first_dataset$class == "NG", 0, 1)

# get 64 of the most correlated predictors with class in first_dataset
x <- names(sort(abs(cor(first_dataset)[, "class"]), decreasing = T)[-1][1:64])

# set threshold to .8 and remove predictors such that
# no predictor has a correlation with another predictor
# higher than .8
first_threshold <- .8
first_predictors <- find_pred(cor_matrix, x, first_threshold)
print(first_predictors)
```

```
##  [1] "BioGRID_log_degree"
##  [2] "Broad_H4K20me1_percentage"
##  [3] "Broad_H3K9ac_percentage"
##  [4] "pLOF_Zscore"
##  [5] "H3K79me2_height"
##  [6] "H4K20me1_height"
##  [7] "intolerant_pLI"
##  [8] "N_LOF"
##  [9] "H3K4me1_width"
## [10] "H3K36me3_width"
## [11] "VEST_score"
## [12] "N_Missense"
## [13] "Missense_Entropy"
## [14] "Length_H3K4me3"
## [15] "H3K27ac_height"
## [16] "ncGERP"
## [17] "H3K9ac_height"
## [18] "N_Splice"
## [19] "LOF_TO_Total_Ratio"
## [20] "BioGRID_clossness"
## [21] "H3K4me3_height"
## [22] "Missense_Zscore"
## [23] "RVIS_percentile"
## [24] "H3K4me1_height"
## [25] "Silent_fraction"
## [26] "S50_score_replication_timing"
## [27] "intolerant_pNull"
## [28] "BioGRID_betweenness"
## [29] "Super_Enhancer_percentage"
## [30] "Missense_Damaging_TO_Missense_Benign_Ratio"
## [31] "LOF_KB_Ratio"
## [32] "NonSilent_TO_Silent_Ratio"
## [33] "CDS_length"
## [34] "Cell_proliferation_rate_CRISPR_KD"
## [35] "Exon_Cons"
## [36] "Polyphen2"
## [37] "Nonsense_fraction"
## [38] "Gene_age"
## [39] "Inframe_indel_fraction"
## [40] "Gene_body_hypermethylation_in_cancer"
```

```
# convert 0 to NG, else to OG_TSG
first_dataset$class <- ifelse(first_dataset$class == 0, "NG", "OG_TSG")
table(first_dataset$class)
```

```
##
##     NG OG_TSG
##   2840    337
```

```
# get rows of OG_TSG and add them to the bottom of first_dataset
base <- first_dataset[first_dataset$class != "NG",]
for (i in seq(1:8)){
 first_dataset <- rbind(first_dataset, base)
}
table(first_dataset$class)
```

```
##
##     NG OG_TSG
##   2840   3033
```

```
# train to identify whether NG or not
lda_first <- train(class~ .,
                   data = first_dataset[, c(first_predictors, "class")],
                   method='lda',
                   preProc = c("center", "scale"),
                   trControl = train_control)
```

### Training QDA to classify between OG and TSG

```
# create second_dataset set from everything that is not NG in initial_data
second_dataset <- initial_data[initial_data$class != "NG", ]
# convert OG to 1 and TSG to 2
second_dataset$class <- ifelse(second_dataset$class == "OG", 1, 2)

# get 64 of the most correlated predictors with class in second_dataset
y <- names(sort(abs(cor(second_dataset)[, "class"]), decreasing = T)[-1][1:64])

# set threshold to .7 and remove predictors such that
# no predictor has a correlation with another predictor
# higher than .7
```

```r
second_threshold <- .7
second_predictors <- find_pred(cor_matrix, y, second_threshold)
print(second_predictors)
```

```
##  [1] "Broad_H3K36me3_percentage"
##  [2] "H3K4me2_height"
##  [3] "VEST_score"
##  [4] "H3K79me2_width"
##  [5] "H3K27ac_height"
##  [6] "H3K9me2_height"
##  [7] "Length_H3K4me3"
##  [8] "Broad_H3K27ac_percentage"
##  [9] "Inactivating_mutations_fraction"
## [10] "H4K20me1_height"
## [11] "Gene_body_hypermethylation_in_cancer"
## [12] "H3K9me3_height"
## [13] "Missense_fraction"
## [14] "Missense_Entropy"
## [15] "N_LOF"
## [16] "N_Splice"
## [17] "S50_score_replication_timing"
## [18] "Splice_TO_Silent_Ratio"
## [19] "H3K9me3_width"
## [20] "Silent_KB_Ratio"
## [21] "CDS_length"
## [22] "Missense_Damaging_TO_Missense_Benign_Ratio"
## [23] "CNA_deletion"
## [24] "H3K27me3_height"
## [25] "Gene_age"
## [26] "H3K9me2_width"
## [27] "Canyon_genebody_hypermethylation"
## [28] "Super_Enhancer_percentage"
## [29] "Broad_H3K9me2_percentage"
## [30] "Gene_expression_Z_score"
## [31] "pLOF_Zscore"
## [32] "BioGRID_log_degree"
```

```r
# convert 1 to OG and else to TSG
second_dataset$class <- ifelse(second_dataset$class == 1, "OG", "TSG")

# train to identify whether OG or TSG
qda_second <- train(class~ .,
                   data = second_dataset[, c(second_predictors, "class")],
                   method='qda',
                   preProc = c("center", "scale"),
                   trControl = train_control, sep = "")
```

### Making predictons on test data

```r
# read in test data
test_data_first <- read.csv("ucla-stats101c-lec4/test.csv")
```

```r
# test_data_first <- read.csv("ucla-stats101c-lec4/training.csv")

table(test_data_first$class)
```

```
## < table of extent 0 >
```

```r
# make predictions whether NG or not
p1 <- predict(lda_first, newdata = test_data_first)

# take all the ones predicted not NG
test_data_second <- test_data_first[which(p1 == "OG_TSG"), ]

# classify the ones not NG as either OG or TSG
p2 <- predict(qda_second, newdata = test_data_second)

# change into desired format, initially in factor form, but changing to 0, 1, 2
results <- as.numeric(p1) - 1
p2 <- as.numeric(p2)
counter <- 1
for (i in seq(1, length(results))){
  if(results[i] != 0){
    results[i] <- p2[counter]
    counter <- counter + 1
  }
}

table(results)
```

```
## results
##    0    1    2
## 1126   74  163
```

```r
# create dataframe and name columns
submit <- data.frame(test_data_first$id, results)
colnames(submit) <- c("id", "class")

# write to file which will be submitted to Kaggle for evaluation
write.csv(submit, "submit_this.csv", row.names=FALSE)
```

### Code to create plots

```r
library(lattice)
library(ggplot2)
library(gridExtra)

training <- read.csv("ucla-stats101c-lec4/training.csv")
plot1 <- ggplot(training, aes(x = id, y = Silent_KB_Ratio)) +
  geom_point() +
  facet_grid(~class) +
  ylab("Silent_KB_Ratio") +
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())

plot2 <- ggplot(training, aes(x = id, y = N_Missense)) +
  geom_point() +
  facet_grid(~class) +
  ylab("N_Missense") +
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())


plot3 <- ggplot(training, aes(x = id, y = Missense_KB_Ratio)) +
  geom_point() +
  facet_grid(~class) +
  ylab("Missense_KB_Ratio") +
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())


plot4 <- ggplot(training, aes(x = id, y = NonSilent_TO_Silent_Ratio)) +
  geom_point() +
  facet_grid(~class) +
  ylab("NonSilent_TO_Silent_Ratio") +
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())


grid.arrange(plot1, plot2, plot3,plot4, ncol=2)
```

## Statement of Contribution

Aaron: I focused on logistic regression. To do this, I had to do some research on how multinomial logistic regression for classification worked.

Darren: I created the general code framework (such as the find_pred() function) which allowed for testing of other methods and thresholds with the change of only a few lines. I focused mainly on the LDA method.

Jingwei: I mainly focused on the kNN method, but unfortunately the results were not as good as the other methods.

Yonatan: I planned out our schedule and divided workload along with testing the QDA method.

When creating our report, we split the work pretty evenly and worked on it collaboratively.