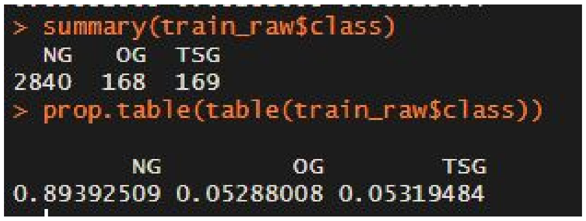
101C Midterm Project Report

1. The Data

In this project, our goal is to use statistical methods that we have learned in this course to identify cancer driver genes (TSGs and OGs) from a gene feature dataset. The response variable 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. The training dataset includes 3,177 genes [rows] and 97 predictors [columns]. This training data includes the responses in the column labeled class. The values of the response are the true class of each gene in the training set: NG (labeled as 0), OG (1) and TSG (2). The test data are in file test.csv. The dataset includes 1,363 genes [rows] and 97 predictors [columns]. The test data does not include the gene classes.

2. Exploratory Data Analysis and Data Pre-Processing

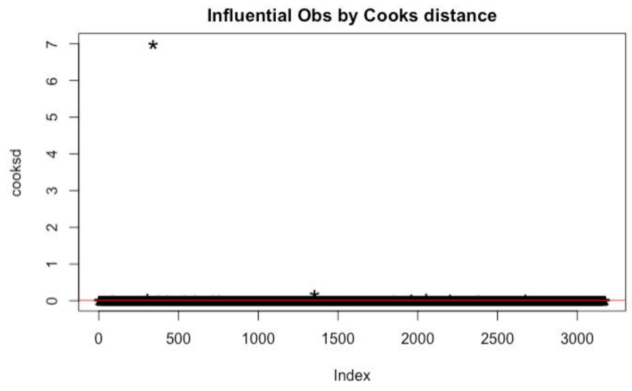
1. There are no missing/NA values and our dataset is tidy, where each row is an observation and each column is a feature.
2. Using the glimpse() function to check the data types of our variables, we see that our target variable 'class' is numeric. We convert the response variable 'class' from numeric to a factor with three levels (0,1,2) and rename the factor levels to their respective gene category ('NG', 'OG', 'TSG'). There are no other categorical variables in the dataset.
3. Remove the first column 'id' as it is not helpful in predicting the class of a gene
4. Since we have a multi-class classification problem, it is helpful to examine the proportions of each class in our training data.



```
> summary(train_raw$class)
  NG   OG  TSG
2840  168  169
> prop.table(table(train_raw$class))

        NG         OG        TSG
0.89392509 0.05288008 0.05319484
```

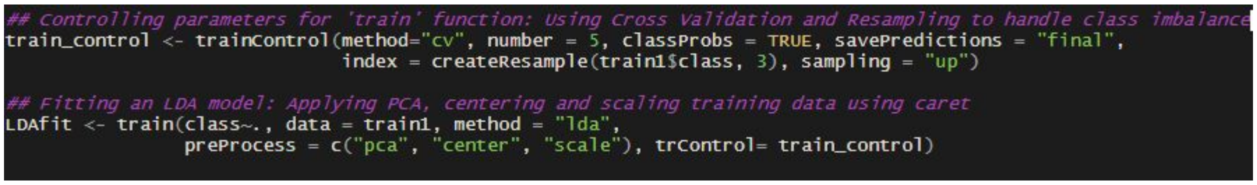
From the above table, we can see that we have a high class imbalance with 90% of the observations belonging to the class 'NG' and around 10% of observations split between 'OG' and 'TSG'. It is important to address this while fitting our model because a model which predicts all observations as class 'NG' would still achieve around 90% accuracy. Therefore, we should consider our metrics and model selection given that we have a large number of predictors and an imbalanced class.



5. Check for outliers and inconsistent data points and remove them. We use Cook's distance as our metric to estimate the influence of a data point. From the plot above, we see that we have some points that are inconsistent from the rest of the dataset and hence, we remove these observations.
6. Before fitting our model, we will check for multicollinearity between our predictor variables and remove the highly correlated variables to decrease bias and have a simpler model. For this project, we removed all variables with an absolute correlation of more than 0.7. After preprocessing the data, we finally get a dataset with 3072 rows and 55 variables.

3. The Model

Since we have a multiclass classification problem, *discriminant analysis* techniques are best suited for our purpose. We achieved the highest optimal informedness score of 0.63850113680935 with a Linear Discriminant Analysis (LDA) model implemented using the 'caret' package. Since LDA operates on the assumptions that predictors are normally distributed (Gaussian distribution), it is important to standardize the variables (scaling and centering data).



```
## Controlling parameters for 'train' function: Using Cross Validation and Resampling to handle class imbalance
train_control <- trainControl(method="cv", number = 5, classProbs = TRUE, savePredictions = "final",
                              index = createResample(train1$class, 3), sampling = "up")

## Fitting an LDA model: Applying PCA, centering and scaling training data using caret
LDAfit <- train(class~., data = train1, method = "lda",
                preProcess = c("pca", "center", "scale"), trControl= train_control)
```

1. In the above model, we use the trainControl() function and the 'sampling' argument to increase the number of minority class samples by bootstrapping new samples to deal with the class imbalance. Using this technique resulted in a much higher score as the model is now trained on more samples of the minority classes (OG and TSG) and generates more accurate predictions.
2. In terms of pre-processing, we use the caret package to apply Principal Component Analysis (PCA) to reduce the dimensionality of our large dataset. In addition to scaling and centering the data, using PCA resulted in an improved score in terms of the performance of the model.

3a. Model Evaluation Metric

For the given classification problem, accuracy is not a good metric to consider given the class imbalance issue. Instead, we compare the **AUC: Area under the ROC curve** as our metric for evaluating our models. From Figure 1 below, we can see that we get a AUC of 0.88 for the above model, which means we have a very good measure of separability between our classes. In comparison, Figure 2 shows us the AUC for our final LDA model in comparison to the less successful QDA and KNN models.

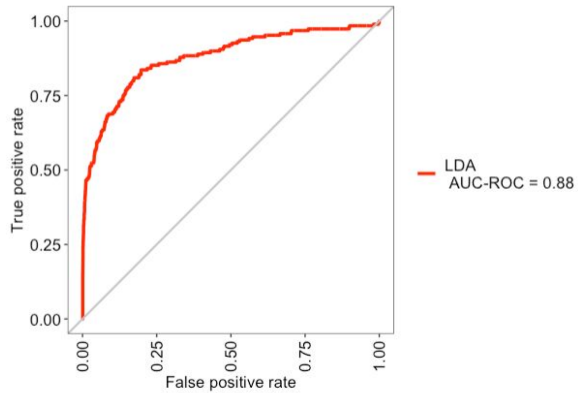


Figure 1: AUC for the best LDA model.

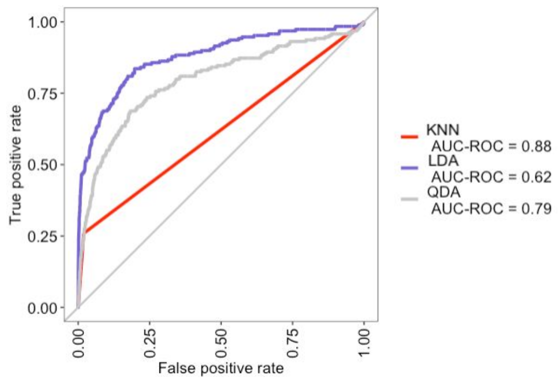


Figure 2: AUC for LDA, KNN, and QDA

3b. Why It Works

Our model works well because we thoroughly pre-processed the data by removing multicollinear variables, removing outliers, and handling class imbalance while fitting our LDA model. However, we still use a large number of predictors in our model and the performance of our model can be improved by reducing the number of predictors. An approach to this would be to plot the distribution of each variable and ensure that they are normally distributed so that we satisfy the LDA assumption. For variables that have skewed or exponential distributions, applying suitable transformations such as Box-Cox or log-transform would significantly improve the model's accuracy.

4. Statement of Contribution

Anshuman Mahalley

- Performed exploratory data analysis and data preprocessing (removing outliers, multicollinear variables, checking assumptions) to clean and transform the training dataset before fitting model.
- Achieved best model performance using LDA model, bootstrapping new samples from minority classes to solve the problem of class imbalance and applying preprocessing methods (PCA, scaling and centering).

Catherine Jennifer

- Applied preprocessing methods such as PCA, scaling, and centering; and created the KNN and QDA models. Determined that they were outperformed by the LDA model that Anshu created.
- Plotted the AUC-ROC graphs for all 3 tried classification models: KNN, QDA, and LDA.
- Acted as project manager throughout the competition, including scheduling and hosting Zoom meetings as well as conducting daily check-ins with team members.
- Wrote and edited the report together with Anshu.

Serena Tan

- Carried out initial data analysis through the creation of correlation matrices
- Rendered exploratory QDA models using the 5 variables most correlated with class; however, these models were outperformed by later models submitted by fellow members
- Created and merged teams on Kaggle

5. Appendix

```
# loading libraries
library(tidyverse)
library(haven)
library(caret)
library(MLeval)
library(mlbench)

# Importing training and test data
```

```
train_raw <- read_csv("training.csv")
test <- read_csv("test.csv")

# Converting the variable 'class' to a factor from numeric (Classification Problem)
train_raw$class <- as.factor(train_raw$class)
levels(train_raw$class) <- c(“NG”, “OG”, “TSG”)  # Renaming factor levels to the respective
gene classes

train_raw <- na.omit(train_raw)

# Checking dimensions of training and test data
dim(train_raw)
dim(test)

# Removing the first column 'id'
train_raw <- train_raw[,-1]

# EXPLORATORY DATA ANALYSIS

glimpse(train_raw)  # Checking data type of all the variables in the dataset

# Creating a table to check class balance and the proportion of each class
summary(train_raw$class)
prop.table(table(train_raw$class))

# Checking for outliers using Cook's distance
cooksd <- cooks.distance(glm(class ~ .,family = "binomial", data = train_raw))

# Plotting Cook's distance for each observation
plot(cooksd,
    pch = "*",
    cex = 2,
    main = "Influential Obs by Cooks distance")
abline(h = 4*mean(cooksd, na.rm = T), col = "red")

outliers <- rownames(train_raw[cooksd > 4*mean(cooksd, na.rm = T), ]) # Finding index of
outlier points
print(outliers)

# Removing outliers
train_raw <- train_raw[-c(1:35),]
```

```r
# Feature selection - Method 1: Removing highly correlated variables with correlation > 0.7
correlationMatrix <- cor(train_raw[,-98])
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff = 0.70)
train1 <- train_raw[,-highlyCorrelated]

# Setting seed to reproduce model
set.seed(123)

# Controlling parameters for our model - Bootstrapping samples from minority classes to
handle class imbalance and using Cross-Validation
train_control <- trainControl(method = "cv", number = 5, classProbs = TRUE, savePredictions
= "final", index = createResample(train1$class, 3), sampling = "up")

# LDA Model: Using PCA, Centering and Scaling to preprocess training dataset
LDAfit <- train(class~., data = train1, method = "lda", preProcess = c("pca", "center", "scale"),
trControl = train_control)

# KNN
KNNfit <- train(class~., data = train1, method = "knn", preProcess = c("center", "scale"),
            trControl = train_control, tuneGrid = expand.grid(k = seq(1, 50, by = 5)))

# QDA
QDAfit <- train(class ~ ., data = train1, method = "qda", preProcess = c("pca", "center",
"scale"), trControl = train_control)


# Evaluating models performance and plotting ROC-AUC curves for our models
res <- evalm(list(LDAfit, KNNfit, QDAfit), gnames = c('LDA', 'KNN', 'QDA'))
res$roc

# Code snippet to prepare data for Kaggle Submission

copy_test <- test # Creating a copy of test dataset
copy_test$class <- predict(LDAfit, newdata = copy_test)  # Assigning predicted outputs to
'class' column of copy_test dataframe

submission <- copy_test %>% dplyr:: select(id, class) # Subsetting copy_test dataframe to
get relevant columns for submission
levels(submission$class) <- c(0, 1, 2) # Renaming factor levels to original values
colnames(submission) <- c("id", "class")
head(submission)
```

```r
write.csv(submission, "m.csv", row.names = FALSE) # Saving file in .CSV format
```