STATS 101C Midterm Project

Limit DNE: Yanhua Lin, Hana Yerin Lim, Yingzhen Zhao

November 2020

1 Abstract

- Kaggle Information

Kaggle Group Name: Limit DNE
Kaggle Score: 0.79471

- Final Model

class = $H4K20me1_{h}eight + Broad_{H}3K79me2_{p}ercentage + ncGERP + RVIS_{p}ercentile + dN_{t}o_{d}S_{r}atio + pLOF_{Z}score + Missense_{Z}score + intolerant_{p}Null + intolerant_{p}LI + Gene_{b}ody_{h}ypomethylation_{i}n_{c}ancer + Promoter_{h}ypermethylation_{i}n_{c}ancer + BioGRID_{l}og_{d}egree + BioGRID_{b}etweenness + Super_{E}nhancer_{p}ercentage + VEST_{s}core + Exon_{C}ons + NonSilent_{T}O_{S}ilent_{R}atio + Inframe_{i}ndel_{f}raction + Silent_{f}raction + Missense_{D}amaging_{T}O_{M}issense_{B}enign_{R}atio + Missense_{E}ntropy + N_{S}plice$

Total Number of Predictors Used: 22

2 Introduction

This is a study on distinguishing abnormal cells that cause cancer using classification methods. The predictors in the study are various features of the genes. The response under study involves three classes: a gene being OG, TSG or NG. The main goal is to identify both TSGs and OGs, which are genes that play a role in cancer. In the training set, which is used for creating models, there are 3177 IDs and 97 Variables. All the variables are the features of genes. There are 1363 IDs in test data needed to identify.

3 Methodology

This section is to discuss the establishment of the model and the predictors selection. In particular, the processing of using and choosing classification methods will be explained in detail.

3.1 Preprocessing of the data

- Data Set Cleaning

After exploring the raw data set, we observed that the data set was imbalanced and that the column predictors contained excessive 0s. Our first approach was to remove highly correlated predictors. We then cleaned the data by changing the values of extreme outliers to two times the interquartile(IQR) range so that the training dataset is more representative and changed the values of outliers accordingly.

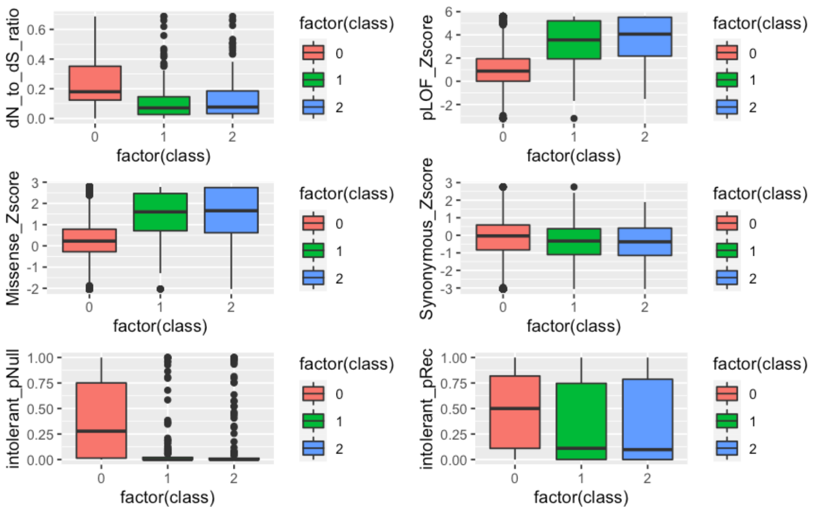
3.2 Description of the model

- Predictor Selection

The predictor selection of the model is based on two means, the correlation between variables and the boxplots between variables and class.

One of the important factors to consider is the presence of multicollinearity. When two or more highly correlated predictor variables are included in a regression model, they are effectively carrying very similar information about the response variable, and would lead to very few of the regression coefficients be statistically significant. This could hurt the accuracy of the prediction and therefore needs to be avoided. Thus, we dropped the variables that were highly correlated and narrowed our predictors down to 56 variables.

In order to obtain more effective variables, we selected the final predictors based on boxplots. Analyzing boxplots is one of the effective ways to detect significant impacts on one another. The boxplots with similar means and distributions of all three classes are dropped as it indicates insignificant affects on the class.

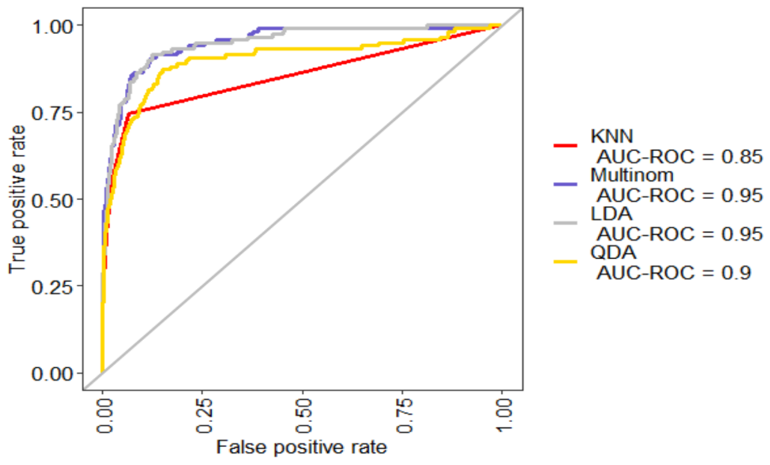


Example plots above are showing that features $dN_to_dS_ratio$, $Synonymous_Zscore$, $intolerant_pNull$, and $intolerant_pRec$ are not good predictors. Plots of $dN_to_dS_ratio$ and $intolerant_pNull$ group by class are showing that outliers of class 1 and 2 are in the main part of class 0, so that these outliers are very likely to be classfied as class 0. Plots of $Synonymous_Zscore$ and $intolerant_pRec$ group by class are showing that the distribution of each group are very similar so that it's hard to distinguish each class. Based on 56 plots(left 56 variables after dropping predictors are highly correlated) and the method we use to select predictors, we narrowed down to 22 variables as our final decision.

- Processing Classification Methods

After conducting comprehensive analysis using the WCA scoring method, LDA produced the most accurate prediction.

We applied classification metrics, such as KNN, LDA, QDA, Logistic Regression, and Naive Bayes classifier, to our selected predictors. To obtain the prediction accuracy of each algorithm, we used validation set to choose the best predictive method. We created partition on 3 to 7 ratio. The accuracy results based on the predictors used with KNN, LDA, QDA and multinomial naive bayes are shown as follows:
LDAfit: 0.9485
KNNfit: 0.938 when k=6
Multinomfit: 0.9548
QDAfit: 0.9002



From the AUC-ROC curve, we can observe that Multinomial and LDA attained the best performance with the highest rates of 0.95,
We analyzed that LDA predicts much better on the "TSG" cells and multinom predicts better on the "NG" cells. Since "OG" class is in between "NG" and "TSG" cells, the performance was approximately the same with both the metrics. Since the main goal is to predict well on TSG and OG cancer cells, we used LDA as our predictive model.

3.3 Best evaluation metric value with this model

The best WCA score with this model on Kaggle is 0.79471.

4 Conclusion and Limitation

Although there were 99 predictors given, we successfully narrowed our predictors by cleaning the data, removing highly correlated predictors, and analyzing the correlation relationship through boxplots. As a result, we were able to correctly identify 80 percent of the OGs, TSGs, and NGs using 22 predictors and Linear Discriminant Analysis model. To make a good prediction, both predictor selection and the choice of the method used to predict are important.
Limitation: Most of the features predict NGs and TSGs very well but only a few predict OGs well, thus most of the OGs are classfied as either NGs or TSGs.
Use the prediction and subsetting methods only covered from chapter 1 to 5

5 Statement of Contributions

All the members collaborated on every parts of the project.
Predictor Selection: Yanhua Lin, Hana Yerin Lim, Yingzhen Zhao
Processing Classification Methods: Hana Yerin Lim, Yanhua Lin
Report: Yingzhen Zhao, Hana Yerin Lim, Yanhua Lin

101C Midterm Kaggle Competition

Team Limit DNE: Yingzhen Zhao, Yanhua Lin, Hana Yerin Lim

11/7/2020

Data libraries & Loading data

Data Cleaning on training data set

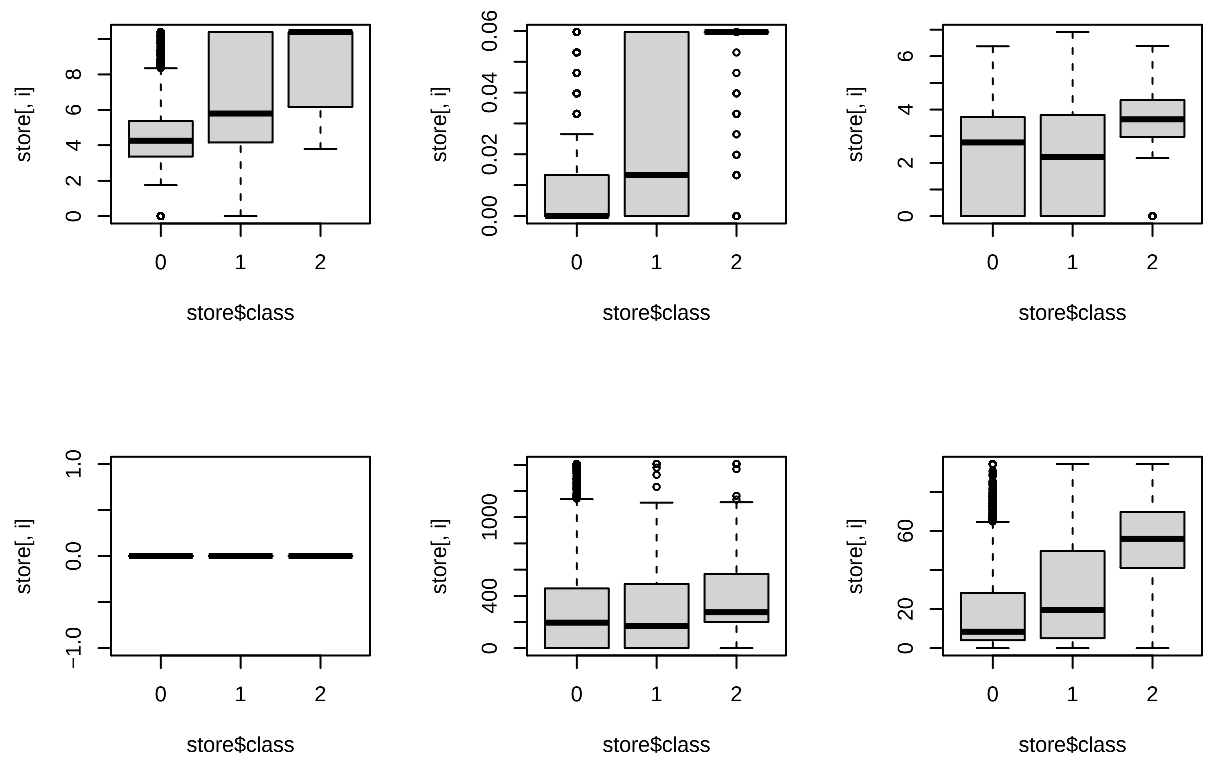
i) Removing highly correlated predictors

```r
train_scale <- as.data.frame(scale(train[ , -c(1, 99)]))
train_scale <- train_scale[ , 97 : 1]

# remove predictors highly correlated
save_features <- character()
while(ncol(train_scale) > 2) {

  correlation <- cor(train_scale[ , 1], train_scale[ , 2 : ncol(train_scale)])
  high_correlated <- names(which(correlation[1, ] > 0.7))
  left_features <- colnames(train_scale)[!(colnames(train_scale) %in% high_correlated)]
  left_features <- left_features[-1]
  save_features <- c(save_features, names(train_scale[1]))
  train_scale <- train_scale[ ,left_features]

}

train_left <- as.data.frame(scale(train[ ,save_features]))
```

ii) Changing values of outliers on the training data set

Predictors selection using boxplot

```r
# plot the predictors by group; Example plot
par(mfrow = c(2,3))
for (i in colnames(store)[1:6]) {
  boxplot(store[ ,i] ~ store$class)
}
```



```
# plot the predictors by group
par(mfrow = c(2,3))
for (i in colnames(store)[1:6]) {
  boxplot(store[ ,i] ~ store$class)
}
par(mfrow = c(2,3))
for (i in colnames(store)[7:12]) {
  boxplot(store[ ,i] ~ store$class)
}
par(mfrow = c(2,3))
for (i in colnames(store)[13:18]) {
  boxplot(store[ ,i] ~ store$class)
}
par(mfrow = c(2,3))
for (i in colnames(store)[19:24]) {
  boxplot(store[ ,i] ~ store$class)
}
par(mfrow = c(2,3))
for (i in colnames(store)[25:30]) {
  boxplot(store[ ,i] ~ store$class)
}
par(mfrow = c(2,3))
for (i in colnames(store)[31:36]) {
  boxplot(store[ ,i] ~ store$class)
}
par(mfrow = c(2,3))
```

```r
for (i in colnames(store)[37:42]) {
  boxplot(store[ ,i] ~ store$class)
}
par(mfrow = c(2,3))
for (i in colnames(store)[43:48]) {
  boxplot(store[ ,i] ~ store$class)
}
par(mfrow = c(2,3))
for (i in colnames(store)[49:54]) {
  boxplot(store[ ,i] ~ store$class)
}
par(mfrow = c(2,3))
for (i in colnames(store)[55:58]) {
  boxplot(store[ ,i] ~ store$class)
}
```

Dimension reduction

```r
# predictors picked from the plot
features <- colnames(train_left)[c(1,2,15,17,19,20,21,23,25,27,30,31,32,33,38,42,46,49,52,56,57,58)]
final_train <- store[features]
final_train$class <- train$class

final_train$class_3[which(final_train$class == 0)] <- "NG"
final_train$class_3[which(final_train$class == 1)] <- "OG"
final_train$class_3[which(final_train$class == 2)] <- "TSG"
final_train$class_3 <- as.factor(final_train$class_3)
final_train <- final_train[ ,-23]
```

Classification metrics

```r
# predicting class using different methods
set.seed(1000)
trainIndex <- createDataPartition(final_train$class_3, p = 0.7, list = FALSE)
train1 <- final_train[trainIndex, ]
validation <- final_train[-trainIndex, ]


train_control <- trainControl(method="cv", number = 10,
                              classProbs = TRUE,
                              savePredictions = TRUE)

LDAfit <- train(class_3 ~ .,
                data = train1,
                method = "lda",
                preProc = c("center", "scale"),
                trControl = train_control)
```

```
KNNfit <- train(class_3 ~ .,
                data = train1,
                method = 'knn',
                 preProc = c("center", "scale"),
                 trControl = train_control,
                 tuneGrid = expand.grid(k = seq(1, 50, by = 5)))

Multinomfit <- train(class_3 ~ .,
                 data = train1,
                 method = "multinom",
                 preProc = c("center", "scale"),
                 trControl = train_control,
                 trace = FALSE)

QDAfit <- train(class_3 ~ .,
                 data = train1,
                 method = "qda",
                 preProc = c("center", "scale"),
                 trControl = train_control)
```

Prediction of the metrics

```
# test the valitdation set
predLDA <- predict(LDAfit, newdata = validation)
confusionMatrix(validation$class_3, predLDA)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NG  OG TSG
##        NG  839   5   8
##        OG    8  32  10
##        TSG  11   7  32
##
## Overall Statistics
##
##                Accuracy : 0.9485
##                  95% CI : (0.9325, 0.9617)
##     No Information Rate : 0.9013
##     P-Value [Acc > NIR] : 7.438e-08
##
##                   Kappa : 0.7265
##
##  Mcnemar's Test P-Value : 0.638
##
## Statistics by Class:
##
##                      Class: NG Class: OG Class: TSG
## Sensitivity             0.9779   0.72727    0.64000
## Specificity             0.8617   0.98018    0.98004
## Pos Pred Value          0.9847   0.64000    0.64000
## Neg Pred Value          0.8100   0.98670    0.98004
## Prevalence              0.9013   0.04622    0.05252
## Detection Rate          0.8813   0.03361    0.03361
## Detection Prevalence    0.8950   0.05252    0.05252
## Balanced Accuracy       0.9198   0.85372    0.81002
```

```
predKNN <- predict(KNNfit, newdata = validation)
confusionMatrix(validation$class_3, predKNN)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NG  OG TSG
##        NG  850   0   2
##        OG   20  16  14
##        TSG  19   5  26
##
## Overall Statistics
##
##                Accuracy : 0.937
##                  95% CI : (0.9196, 0.9516)
##     No Information Rate : 0.9338
##     P-Value [Acc > NIR] : 0.3787
##
##                   Kappa : 0.608
##
##  Mcnemar's Test P-Value : 2.792e-08
##
## Statistics by Class:
##
##                      Class: NG Class: OG Class: TSG
## Sensitivity             0.9561   0.76190    0.61905
## Specificity             0.9683   0.96348    0.97363
## Pos Pred Value          0.9977   0.32000    0.52000
## Neg Pred Value          0.6100   0.99446    0.98226
## Prevalence              0.9338   0.02206    0.04412
## Detection Rate          0.8929   0.01681    0.02731
## Detection Prevalence    0.8950   0.05252    0.05252
## Balanced Accuracy       0.9622   0.86269    0.79634
```

```
predMultinom <- predict(Multinomfit, newdata = validation)
confusionMatrix(validation$class_3, predMultinom)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NG  OG TSG
##        NG  846   2   4
##        OG    9  32   9
##        TSG  14   5  31
##
## Overall Statistics
```

```
##
##                Accuracy : 0.9548
##                  95% CI : (0.9396, 0.9671)
##     No Information Rate : 0.9128
##     P-Value [Acc > NIR] : 3.947e-07
##
##                   Kappa : 0.7469
##
##  Mcnemar's Test P-Value : 0.01093
##
## Statistics by Class:
##
##                      Class: NG Class: OG Class: TSG
## Sensitivity             0.9735   0.82051    0.70455
## Specificity             0.9277   0.98028    0.97907
## Pos Pred Value          0.9930   0.64000    0.62000
## Neg Pred Value          0.7700   0.99224    0.98559
## Prevalence              0.9128   0.04097    0.04622
## Detection Rate          0.8887   0.03361    0.03256
## Detection Prevalence    0.8950   0.05252    0.05252
## Balanced Accuracy       0.9506   0.90040    0.84181
```

```
predQDA <- predict(QDAfit, newdata = validation)
confusionMatrix(validation$class_3, predQDA)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NG  OG TSG
##        NG  800  25  27
##        OG    7  32  11
##        TSG  11  14  25
##
## Overall Statistics
##
##                Accuracy : 0.9002
##                  95% CI : (0.8794, 0.9185)
##     No Information Rate : 0.8592
##     P-Value [Acc > NIR] : 9.142e-05
##
##                   Kappa : 0.5538
##
##  Mcnemar's Test P-Value : 0.0006362
##
## Statistics by Class:
##
##                      Class: NG Class: OG Class: TSG
## Sensitivity             0.9780   0.45070    0.39683
## Specificity             0.6119   0.97957    0.97188
## Pos Pred Value          0.9390   0.64000    0.50000
## Neg Pred Value          0.8200   0.95676    0.95787
## Prevalence              0.8592   0.07458    0.06618
## Detection Rate          0.8403   0.03361    0.02626
## Detection Prevalence    0.8950   0.05252    0.05252
## Balanced Accuracy       0.7950   0.71514    0.68435
```

Best metric with the highest predictive accuracy

```
# pick the model based on the result above
LDAfit <- train(class_3 ~ .,
                data = final_train,
                method = "lda",
                preProc = c("center", "scale"),
                trControl = train_control)
```

Data cleaning on test data set

```
# change outliers of test data
final_test <- test[features]

change_outlier <- function(x){
  lowerq <- quantile(x)[2]
  upperq <- quantile(x)[4]
  iqr = upperq - lowerq
  upper_outlier <- (iqr * 4) + upperq
  lower_outlier <- lowerq - (iqr * 4)
  x[x > upper_outlier] <- upper_outlier
  x[x < lower_outlier] <- lower_outlier
  x
}

store2 <- cbind()
for(i in colnames(final_test)){
  store2 <- cbind(store2, change_outlier(final_test[[i]]))
}

colnames(store2) <- c(colnames(final_test))
store2 <- as.data.frame(store2)
```

Final prediction

```
# prediction using test data
predLDA <- predict(LDAfit, newdata = store2)
table(predLDA)
```

```
## predLDA
##   NG   OG  TSG
## 1150   81  132
```

```r
result <- data.frame(test$id,predLDA)
colnames(result) <- c("id", "class")
result$class <- as.factor(result$class)
levels(result$class) <- c(0,1,2)

#write.csv(result, "result.csv", row.names = FALSE)
```