Stats 101C Midterm Project

Chelsea Miao Eleanor Jiang Mengyu Zhang Tyler Wu

November 10, 2020

Contents

1 Data Pre-Process

At the beginning, Let's take a look at the distribution of our response variable, class.

NG	OG	TSG
0.89471199	0.05429651	0.05099150

According to the table, we could notice that the class NG appears the most often. In fact, it dominates about 90% of the training data. We now have a preliminary understanding of the prior probability on how the three classes are distributed. In future classification, we likely want to determine a unique threshold to deal with this situation.

1.1 Variable Selection

Our first criteria for selecting predictors was to build a multinomial model on all 97 predictors to view those variables whose p-values are smaller than 0.05, this does not address collinearity issues, so next, we created a correlation matrix to determine which variables to keep.

After finding collinear sets of variables, we selected one variable from each set. Our criteria for which predictor was selected was based on which had a lower p-value. If these were approximately equal, we then compared each variable's average correlation with all other predictors as the tiebreaker.

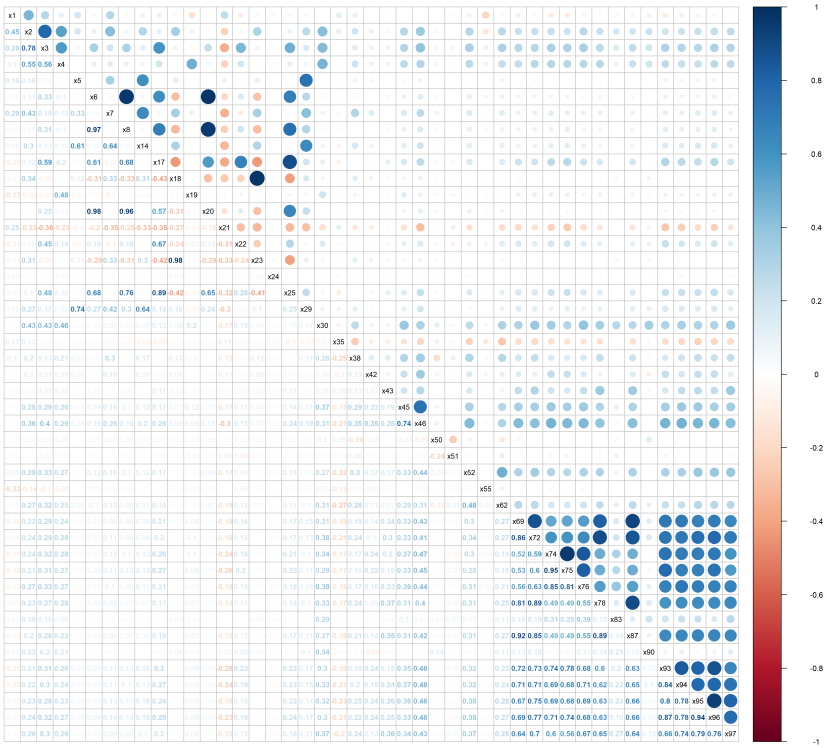

Figure 1: Correlation plot between variables selected based on p-values

1.2 Data Transformation

In the correlation plot (page 3), the last column/row shows the correlation of class with other variables. We can see that there are some variables that have correlation with our response variable, but none of these exceed 0.6, meaning some feature engineering may be necessary to find significant predictors. We can also see that some variables are highly correlated with each other, which may lead to collinearity issues.

When checking the distribution of the predictors we used, we found that some of them were skewed heavily to the right. Therefore, we applied a log transformation to these variables to reduce the magnitude of the larger values and make the distribution more normal. See figure 2 (page 4), for examples of the skewed distributions.

Figure 2: Example of Distributions of Predictors

2 Model Selection and Validation

2.1 Model Description

In the end, We chose 19 predictors in our final model, coming from the following categories: mutations, function, and epigenetics. Our final model was

$$class \sim x1 + log(x3) + log(x4) + log(x7) + log(x19) + x20 + x21 + log(x22) + log(x25) + log(x38) + x42 + log(x43) + x45 + x51 + x52 + log(x75) + log(x78) + log(x83) + log(x97)$$ The number corresponds to the column number of each x

To make the final classification of the data, we wrote a function that used a custom threshold based on the distributions (prior probabilities) of the classes in our training set discussed in the data pre-processing section. We found these values to be around 0.9 NG, 0.05 OG, 0.05 TSG, and set the thresholds to match these probabilities.

```
LR_fit <- caret::train(class~.,_, data=train)
```

2.2 Validity of the Model

By applying the cross validation method in caret, we randomly divided the training data into ten stratified folds in order to simulate predicting new data and check for problems like over-fitting. After the model was built, we train the the model with nine folds and test on the remaining one fold for each of the ten folds.

In addition to the cross-validation built into caret, we also split our training data into a training and validation set to validate our model on. After reviewing the validation accuracies of all the models we have learned (i.e. QDA, LDA, KNN, and LogReg), we decided to use logistic regression for our final prediction model. Moreover, by comparing the weighted categorization accuracy (WCA) for each model on our validation set, the logistic model is also the best model among the four. When we tested our model on the validation set, we received a WCA of 7036 out of 9580 possible, so a prediction accuracy of 73.45%. The reason we were not able to improve on this accuracy was because we were using the default accuracy metric for the majority of our tests, and we realized too late that we should be validating on WCA instead. Therefore, the model may not have the best prediction accuracy, but we were able to validate it on the correct metric.

Finally, by applying a bootstrap test, we used the limited training data to repeat sampling multiple times to recreate a new sample that represents the distribution of the maternal sample. Since we cannot using R to give out a directly model, this method should works in the estimation of variables. As a result we got an average 0.81231106 points and a standard error as 0.01707938, which shows that our model always output an acceptable prediction.

3 Further Discussion

As mentioned before, we are using *RStudio* to help analyze all the data we have. The analyzing tool we used can tell us when what should consider when taking action, but it cannot tell what action to take. Although we have already choosing the predictors cautiously after rigorous discussion, it is still to make the promise that all the rest variables that we did not use in the model are statistically meaningless to appear in the prediction model. Due to lacking of biogenetical related knowledge, the model lacks a bit in interpretability as the predictors were all chosen based on importance in prediction. Moreover, the predictors we took are acceptable but may not be the best combination over the total number of 97 variables in our training data.

Nevertheless, our model makes a valid model with good approaching of the overall prediction. Overall, since the prompt was focused on prediction accuracy, our model does decent for this purpose with a WCA of 7036 out of 9580. By applying the data, we successfully predict most of the gene class with an acceptable tolerance. Combining the all the predictors together with the several specific interaction terms to get the best fitted model with highest efficiency.

A R Codes

```
library(caret)
library(corrplot)
library(plyr)

## load data
train <- read.csv("training.csv")
test <- read.csv("test.csv")

## change column names
colnames(train) <- c("id", paste("x",1:97,sep=""),"class")
colnames(test) <- c("id", paste("x",1:97,sep=""),"class")


## functions
mm_scale <- function(x){
    # normalizes values of a vector x such that all values range from 0 to 1
    # params: x: the vector to be normalized
    (x-min(x))/(max(x)-min(x))
}

add_log_cols <- function(df,cols){
    # function that will create a log column from a column that has been normalized
    # params: df: the dataframe to add the columns to
    #         cols:
    # return: returns a vector containing the prediction (0, 1, or 2) for each gene
    log_df <- log(df[cols]+1) # scale by 1 for 0s then log
    colnames(log_df) <- paste("log", colnames(log_df), sep = "_") # add log_ prefix
    df <- cbind(df,log_df)
    return(df)
}

WCA <- function(data,lev=NULL,model=NULL){
    # custom WCA metric to use in cross validation
    # in place of the default Accuracy metric
    # params: data: a dataframe containing the observed
    # and predicted values for class
    #       (other params are boilerplate)
    # return: the WCA score for the model

    # check that data is correct format
    if(length(data$obs) != length(data$pred)){
    stop("length(data$obs) != length(data$pred)")
    }
    # Scores for correct predictions
    score_vec <- c(1, 20, 20) # 1 NG, 20 OG, 20 TSG
    gene_vec <- c("NG","OG", "TSG")
```

```
    # Running total of WCA score
    WCA <- 0
    for(i in 0:2){
    # check that at least one appearance of class i
    obs_i <- data$pred==gene_vec[i+1]
    if(sum(obs_i) == 0){
      stop("sum(obs_i) == 0 (data mislabeled?)")
    }
    # Calculate points scored for each class
    score_i <- score_vec[i + 1]*sum(data$obs[obs_i] == data$pred[obs_i])
    # Add to running total
    WCA <- WCA + score_i
    }
    names(WCA) = "WCA"
    return(WCA)
}

pred_LR <- function(ptable){
    # function that will classify genes based on custom thresholds
    # params: ptable: a dataframe object containing probabilities of each gene
    #         type (NG, OG, TSG) as columns
    # return: returns a vector containing the prediction (0, 1, or 2) for each gene
    i = 0
    preds = list() # vector to hold predictions
    for(row in 1:nrow(ptable)){
    if(ptable[row,1] >= 0.9){ # if prob NG >= 0.90, classify as NG
      preds[row] = 0
    }
    else{
      if(ptable[row,2] >= ptable[row,3]){ # classify OG if OG prob higher
        preds[row] = 1
      }
      else{ # otherwise classify TSG
        preds[row] = 2
      }
    }
    }
    preds <- unlist(preds)
    return(preds)
}

## variable selection
# first narrow down to important variables, by choosing those with p-values < 0.10
glm.sig_p <- glm(class~.-id, data = train[1:99])
summary(glm.sig_p)

# of the remaining, filter out those with > 0.7 correlation
vars <- c("x1","x2","x3","x4","x5","x6","x7","x8","x14","x17","x18","x19","x20",
            "x21","x22","x23","x24","x25","x29","x30","x35","x38","x42","x43",
```

```
            "x45","x46","x50","x51","x52","x55","x62","x69","x72","x74","x75",
            "x76","x78","x83","x87","x90","x93","x94","x95","x96","x97")
corr <- data.frame(cor(train[vars]))
corr <- corr[]
# only keep variables that do not have a correlation > 0.7 with any other vars
corr[!rowSums(corr > 0.70 & corr != 1),]


## transformations
# normalizing predictor variables
train[2:98] <- apply(train[2:98],2,mm_scale)
test[2:98] <- apply(test[2:98],2,mm_scale)

# adding log-transformed cols to train and test
train <- add_log_cols(train,paste("x",c(3,4,5,7,17,19,22,24,25,29,38,43,75,78,83,90,97),
                        sep=""))
test <- add_log_cols(test,paste("x",c(3,4,5,7,17,19,22,24,25,29,38,43,75,78,83,90,97),
                        sep=""))

# adding factor version of class
train$class_f <- as.factor(train$class)
train$class_f <- revalue(train$class_f, c("0"="NG", "1"="OG","2"="TSG"))
test$class_f <- test$class


## model and predicted probabilites
train_control <- trainControl(method="cv",
                        number = 10,
                        summaryFunction = WCA,
                        selectionFunction="best",
                        classProbs = TRUE,
                        savePredictions = TRUE)

log_fit <- train(
    form = class_f ~x1 +log_x3+log_x4+log_x7+log_x19+x20+x21+log_x22+log_x25+log_x38
            +x42+log_x43+x45+x51+x52+log_x75+log_x78+log_x83+log_x97,
    data = train,
    trControl = train_control,
    method = "multinom",
    metric = "WCA",
    maximize = TRUE
)
p.table <- predict(log_fit,newdata = test,"prob")

# using classifier function to classify probabilities for test set
test_preds <- pred_LR(p.table)

# pair predictions with test set ids and write to csv
df <- data.frame("id" = test$id, "class" = test_preds)
```

```
write.csv(df,"submission.csv", row.names = FALSE)
```

B Statement of Contribution

Tyler Wu: wrote a few models, function for classifying predictions with custom threshold, and function for including WCA metric into cross-validation

Eleanor Jiang (zoe1trick): did the variable selection, tried few feature engineering by transforming continuous numeric data to factors

Mengyu zhang (onezmy on Kaggle) : Trying use principle component analysis to help with variable selection, using bootstrap to figure out a better method. Doing report writing work.

Chelsea Miao: tried using LASSO to help with variable selection, wrote a function to evaluate the final models to help with the validation.

C Sources

References

[1] Jason Brownlee.
https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your\-machine-learning-dataset/

[2] Max Kuhn
https://topepo.github.io/caret/index.html

[3] caret demo.R from CCLE