STSCI 4740 Final Project

Names/NetIDs:

Jieon (Geena) Lee (jl3257)

Bin Jang (bj225)

Anthony Wu (aw526)

Hana Gabrielle Bidon (hrb56)

Introduction:

For our final project, we chose to use the cancer data set. The goal is to predict the cancer levels (low/medium/high) of any given patient, using the predictors given. We aimed to use the algorithms we learned in class and compare their test errors to choose the best model and apply it to our data set.

Initial Thoughts:

First, we had to decide which algorithms to apply to the data set. This is a classification problem rather than a regression problem, therefore we eliminated regression algorithms like Linear Regression. We then were left with algorithms like KNN, LDA, QDA, and logistic regression. We will discuss them more specifically throughout the report, but ultimately these were the core four algorithms we wanted to try out.

In order to avoid overfitting, we wanted to ensure some method of validation. At first, we tried the train/validation split method by dividing the data set in a 70:30 split. We later also did cross-validation, which we will discuss more in-depth within the report. We did not do leave-one-out cross-validation because we felt it would take too long and an attempt to reproduce that code would be inefficient. Now we can look into our methods and results.

PART I: Using All Features

As the instructions stated, we tried using all the features for KNN, Logistic Regression, LDA, and QDA. We did, however, omit the patient ID feature, because the value of the patient's ID should have zero impact on the label (in other words, we could change everybody's patient ID number, but their cancer risk level should not change at all).

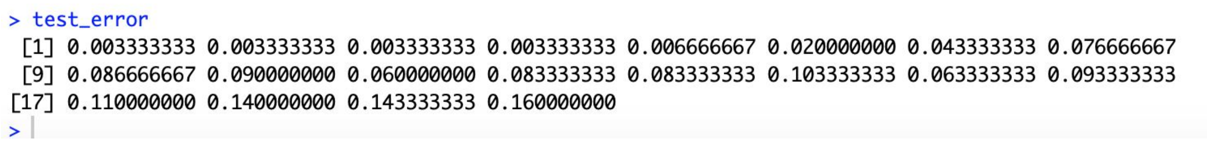
However, Logistic Regression resulted in an error - likely because this is not a binary classification problem and Logistic Regression does not work well for classification beyond 2 classes. QDA also resulted in an error - likely because we had too many features. Therefore, we were left to compare KNN and LDA. What we saw however, was that the error rates on the test set (validation set) were extremely small. For LDA, our error rate was only 0.013 while for KNN the error rate was only 0.0033, which meant only 1 of our 300 test data was off.

This seemed fairly peculiar as we also tried a 50:50 split and the training error was still extremely low, although we were supposed to have a less accurate model compared to the 70:30 split.

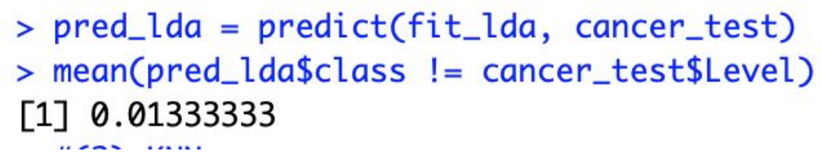
We then found out that there were many duplicate rows in the given data set. With 23 features in each row (excluding ID and Level), it was not logical to have multiple duplicates of the same data within the same age group. A simple calculation of 2^{23} would tell us that we have extremely low possibility for duplicates, even when we assume that there are 2 options for every feature (which is still an underestimate).

Using the unique() function in R, and disregarding ID and Level, we found that there were only 152 unique rows, meaning 848 rows were exact duplicates of other rows. After a simple ordering of the rows in Excel, sorting by the Age column from youngest to oldest, we found, for instance, that all patients aged 14 had the same values for all 23 features (i.e. all these observations were the same.) Since more than 80% of the data were composed of duplicates of a different row, we decided to extract only the rows with a unique pair of data, and train on that "unique dataset" for a 50:50 split. Although this could potentially alter the age (and other features) distribution of the original dataset, we thought removing duplicates would do more benefit than harm, for two reasons. First, removing duplicates could improve the quality of the learning, because training with unique data instead of duplicates will increase the possibility that more unique cases will be learned.

Also, removing duplicates could improve the accuracy of the test error, because if there were a row in a test set that is a duplicate of the same row that was included in the training set, it would always guess correctly on the test for the duplicates (because it's a duplicate of what it has trained). This could also account for the extremely low test error that we got when we tested our model on the original dataset (with duplicate rows not removed) -- it was because the duplicates in the testing "diluted" the number of errors.

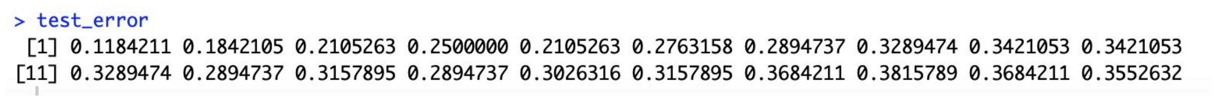


```
> test_error
 [1] 0.003333333 0.003333333 0.003333333 0.003333333 0.006666667 0.020000000 0.043333333 0.076666667
 [9] 0.086666667 0.090000000 0.060000000 0.083333333 0.083333333 0.103333333 0.063333333 0.093333333
[17] 0.110000000 0.140000000 0.143333333 0.160000000
> 
```

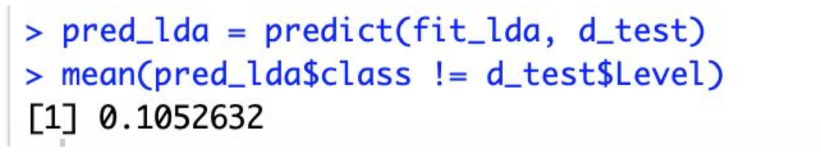


```
> pred_lda = predict(fit_lda, cancer_test)
> mean(pred_lda$class != cancer_test$Level)
[1] 0.01333333
```

(All features, original dataset)



```
> test_error
 [1] 0.1184211 0.1842105 0.2105263 0.2500000 0.2105263 0.2763158 0.2894737 0.3289474 0.3421053 0.3421053
[11] 0.3289474 0.2894737 0.3157895 0.2894737 0.3026316 0.3157895 0.3684211 0.3815789 0.3684211 0.3552632
```



```
> pred_lda = predict(fit_lda, d_test)
> mean(pred_lda$class != d_test$Level)
[1] 0.1052632
```

(All features, unique dataset)

PART II: Feature Selection

We wanted to reduce the number of features because we believed that some variables may be unnecessary or unhelpful when determining classification. Though often more data leads to a better classification, we thought that reducing the number of features we had and utilizing the unique samples could potentially reduce our error rates. It would also allow us to use more algorithms like QDA which could not be used on the original data set.

We found feature selection by running three things: heterogeneous correlation, correlation, and ANOVA tests. We wanted to run three different types because often the correlation coefficient is not completely indicative of how variables affect one another. Additionally, correlation coefficients are more often used for numeric variable comparison, whereas we are using a categorical variable here for our label.

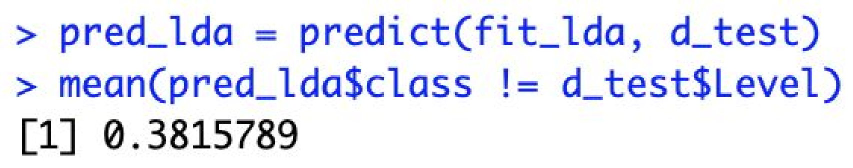
We found through documentation that hetcor() or the heterogeneous correlation matrix, calculates correlations differently depending on the type of variables being used. Because of our categorical variable for our label, we thought the correlation coefficients here could be more meaningful than running cor() alone. For the heterogeneous correlations, the highest values with respect to Level were Passive Smoker, Smoking, Coughing of Blood, Balanced Diet, Air Pollution, Obesity, Chest Pain, Fatigue, Alcohol Use, and Chronic Lung Disease. For the correlations we ran cor() and the highest values with respect to Level were Passive Smoker, Coughing of Blood, Balanced Diet, Smoking, Obesity, Air Pollution, Chest Pain, Alcohol Use, Chronic Lung Disease, and Genetic Risk. Finally, we looped through each variable and ran an ANOVA test.

ANOVA tests are made to find out if the group means among different groups deviate from one another in a statistically significant manner. In other words, ANOVA tests can measure how likely it is that the numerical variable you measure affects the means of the categorical variable. The ANOVA test in R can be run using aov(), which we did with all features and Level. We extracted the p-values from each test. This p-value is a measurement of how likely it is for your test statistic to occur by random chance. Therefore, we found the features that resulted in the lowest p-values. We found that Obesity, Coughing of Blood, Passive Smoker, Balanced Diet, Dust Allergy, Alcohol Use, Genetic Risk, Air Pollution, Occupational Hazards, and Chest Pain had the lowest p-values from their ANOVA tests with Level.

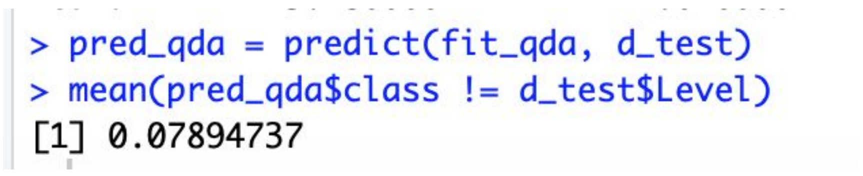
We wanted to limit our features to about 5, so we took the "top 10" features from hetcor, cor, and ANOVA and cross-referenced them to find the five features. Passive Smoker had the highest hetcor and cor values, and had the third smallest p-value from ANOVA. Coughing of Blood had the third highest hetcor, second highest cor, and second smallest p-value. We used similar logic to conclude that Passive Smoker, Coughing of Blood, Balanced Diet, Air Pollution, and Obesity should be the five features to use. Through cross-referencing different methods, we felt that these five were most likely the features that influenced Level the most and therefore should be utilized in classification. We discuss the results of running our algorithms using these features and our unique data set below.

PART III: Model with the Selected Features

Through the feature selection methods mentioned above in Part II, we decided on five key features: passive smoker, coughing of blood, balanced diet, air pollution, and obesity, and reran the algorithms on the cancer data set with only these five predictors on Level. This time, since the number of features has been reduced from 24 to 5, we were able to utilize both the LDA and QDA algorithms. QDA worked this time because now we have reduced the number of features, compared to when we attempted to use QDA with all features. After randomly splitting the new dataset that only contained unique observations into an equally-sized training and test set, we found that LDA yielded a moderately high test error rate of 0.382, while QDA yielded a test error rate of only 0.079, a significant reduction. Thus, it seemed that QDA was the better option of the two.



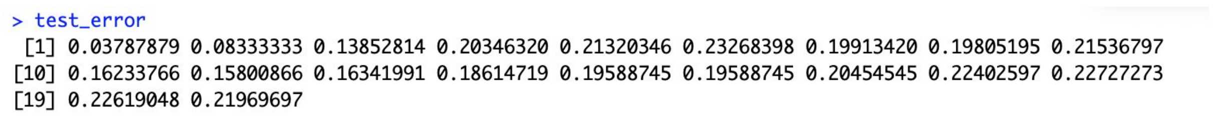
```
> pred_lda = predict(fit_lda, d_test)
> mean(pred_lda$class != d_test$Level)
[1] 0.3815789
```



```
> pred_qda = predict(fit_qda, d_test)
> mean(pred_qda$class != d_test$Level)
[1] 0.07894737
```

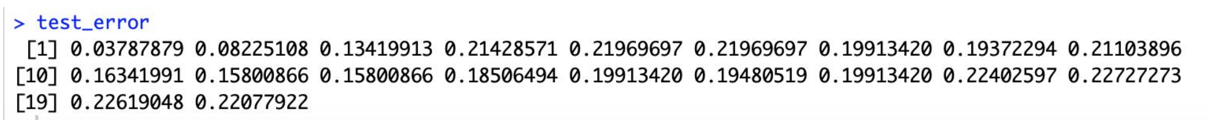
We then ran KNN twice. We used the same training set for both, but different test sets: one that was only on the unique observations (i.e. the same sets used for LDA and QDA above), and one on the whole cancer data set (minus the training set observations). Both yielded similar test error results, due to the fact that the majority of the dataset had duplicated observations (rows), and it would be likely that if KNN was able to accurately predict Level on our test set, given the five features, it could accurately predict those duplicated observations as well. Here, we found in both instances that the lowest test error of 0.0379 corresponded to K = 1. However, the test error rates for when K = 2 or K = 3 are still considered relatively low.

KNN, training set - 5 features, test set



```
> test_error
 [1] 0.03787879 0.08333333 0.13852814 0.20346320 0.21320346 0.23268398 0.19913420 0.19805195 0.21536797
[10] 0.16233766 0.15800866 0.16341991 0.18614719 0.19588745 0.19588745 0.20454545 0.22402597 0.22727273
[19] 0.22619048 0.21969697
```

KNN, training set - 5 features, whole data set



```
> test_error
 [1] 0.03787879 0.08225108 0.13419913 0.21428571 0.21969697 0.21969697 0.19913420 0.19372294 0.21103896
[10] 0.16341991 0.15800866 0.15800866 0.18506494 0.19913420 0.19480519 0.19913420 0.22402597 0.22727273
[19] 0.22619048 0.22077922
```

Conclusion:

We concluded that both QDA and KNN models have low test error rates, and therefore the most valid among the models that we had tested. To validate the accuracy of our selected-feature-model, we ran our 5-feature KNN model onto our original dataset as the testing set. We could see that the test error rate for all ranges of K was extremely close to the error rates on our previous test set with only unique rows. Although KNN is likely to overfit the training data on lower values of K, because we are getting stable test error rates after K = 4, it is safe to assume that our 5-feature KNN model has about 80% accuracy. This means that our 5-feature KNN model was even more accurate compared to the all-feature KNN model that we tested in part 1, which had about 70% accuracy for K>5 (when tested on unique values).

Code:

```
library(readxl)

cancer_data <- read_excel("cancer_data.xlsx")

cancer_data <- read.csv("cancer_data.csv")



# create 70% training data and 30% test data

cancer_data = cancer_data[,-1]

num_total = nrow(cancer_data)

set.seed(18)

train_idxs = sample(num_total,ceiling(0.7*num_total))

cancer_train = cancer_data[train_idxs,]

cancer_test = cancer_data[-train_idxs,]


#(1) LDA

library(MASS)

fit_lda = lda(Level ~ ., data = cancer_train)

fit_lda


pred_lda = predict(fit_lda, cancer_test)

mean(pred_lda$class != cancer_test$Level)


#(2) KNN

library(class)
```

```
features = c(1:(ncol(cancer_data)-1))

kset<-c(1:9,seq(10,60,5))

test_error<-c()

for(num_k in kset)

  {

  predknn_k1 = knn(cancer_train[features], cancer_test[features], cancer_train$Level,
k=num_k)

  test_error<-c(test_error,mean(predknn_k1 != cancer_test$Level))

}

test_error


kset[which.min(test_error)]


####### with unique rows

d <- read.csv("cancer_data.csv")

d = unique(cancer_data)

d$Level = as.factor(d$Level)

num_total = nrow(d)

set.seed(5)

train_idxs = sample(num_total,ceiling(0.5*num_total))

d_train = d[train_idxs,]

d_test = d[-train_idxs,]


library(MASS)

fit_lda = lda(Level ~ ., data = d_train)
```

```
fit_lda

pred_lda = predict(fit_lda, d_test)

mean(pred_lda$class != d_test$Level)


library(class)

features = c(1:(ncol(cancer_data)-1))

kset<-c(1:20)

test_error<-c()

for(num_k in kset)

{

  predknn_k1 = knn(d_train[features], d_test[features], d_train$Level, k=num_k)

  test_error<-c(test_error,mean(predknn_k1 != d_test$Level))

}

test_error


kset[which.min(test_error)]


#passive smoker, coughing up blood, balanced diet, air pollution, obesity

d <- read.csv("cancer_data.csv")

d = unique(cancer_data)

d$Level = as.factor(d$Level)

d = d[,c(12,14,9,3,10,24)]

num_total = nrow(d)

set.seed(5)

train_idxs = sample(num_total,ceiling(0.5*num_total))
```

```
d_train = d[train_idxs,]
d_test = d[-train_idxs,]

library(MASS)
fit_lda = lda(Level ~ ., data = d_train)
fit_lda
pred_lda = predict(fit_lda, d_test)
mean(pred_lda$class != d_test$Level)

fit_qda = qda(Level ~ ., data = d_train)
fit_qda
pred_qda = predict(fit_qda, d_test)
mean(pred_qda$class != d_test$Level)

library(class)
features = c(1:(ncol(d)-1))
kset<-c(1:20)
test_error<-c()
for(num_k in kset){
  predknn_k1 = knn(d_train[features], d_test[features], d_train$Level, k=num_k)
  test_error<-c(test_error,mean(predknn_k1 != d_test$Level))
}
test_error

library(class)
```

```r
features = c(1:(ncol(d)-1))

e = cancer_data[-train_idxs,c(12,14,9,3,10,24)]

kset<-c(1:20)

test_error<-c()

for(num_k in kset){

  predknn_k1 = knn(d_train[features], e[features], d_train$Level, k=num_k)

  test_error<-c(test_error,mean(predknn_k1 != e$Level))

}

test_error


#hetcor coefficients

library(polycor)

hetcor(cancer_data)


#correlation coefficients

cancer_data$Level <- as.numeric(cancer_data$Level)

cor(cancer_data)


# ANOVA test results

pVals <- c()

cols <- colnames(cancer_data)

for (i in 1:length(cancer_data)) {

  pVals[i] <-summary(aov(cancer_data[[i]]~ cancer_data$Level))[[1]][["Pr(>F)"]]

}
```