Andrew Rothman
anr248@stanford.edu

STATS 202 – Homework #3

October 19th, 2018
SUID: 06325118

Problem #1:

Exercise 3 from section 4.7 of ITSL textbook

This problem relates to the QDA model, in which the observations within each class are drawn from a normal distribution with a class-specific mean vector and a class specific covariance matrix. We consider the simple case where *p*=1; i.e. there is only one feature.

Suppose that we have K classes, and that if an observation belongs to the kth class then X comes from a one-dimensional normal distribution, $X \sim X(\mu_k, {\sigma_k}^2)$. Recall that the density function for the one-dimensional normal distribution is given in (4.11). Prove that in this case, the Bayes' classifier is *not* linear. Argue that it is in fact quadratic.

Answer:

The derivation below provides insight as to why for the QDA model with *p*=1, the Bayes' classifier is not linear but is in fact quadratic. This provides the QDA with quadratic, and not linear, decision boundaries:

- Assume we know $P(Y = k) = \pi_k$ exactly
- $P(X = x|Y = k) = f_k(x) = \frac{1}{\sqrt{2\pi{\sigma_k}^2}} e^{-\frac{1}{2{\sigma_k}^2}(x-\mu_k)^2}$

$$P(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} = \frac{\pi_k \frac{1}{\sqrt{2\pi{\sigma_k}^2}} e^{-\frac{1}{2{\sigma_k}^2}(x-\mu_k)^2}}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi{\sigma_l}^2}} e^{-\frac{1}{2{\sigma_l}^2}(x-\mu_l)^2}}$$

, note that the denominator does not depend on the response "k", so it can be absorbed into constant "C":

$$C = \sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi{\sigma_l}^2}} e^{-\frac{1}{2{\sigma_l}^2}(x-\mu_l)^2}$$

$P(Y = k|X = x) = C\pi_k \frac{1}{\sqrt{2\pi{\sigma_k}^2}} e^{-\frac{1}{2{\sigma_k}^2}(x-\mu_k)^2}$, take the ln of both sides:

$$ln\big(P(Y = k|X = x)\big) = ln\left(C\pi_k \frac{1}{\sqrt{2\pi{\sigma_k}^2}} e^{-\frac{1}{2{\sigma_k}^2}(x-\mu_k)^2}\right)$$

$$= ln(C) + ln(\pi_k) + ln\left(\frac{1}{\sqrt{2\pi{\sigma_k}^2}}\right) - \frac{1}{2{\sigma_k}^2}(x-\mu_k)^2$$

$$= ln(C) + ln(\pi_k) + ln(1) - ln\left((2\pi{\sigma_k}^2)^{\frac{1}{2}}\right) - \frac{1}{2{\sigma_k}^2}(x-\mu_k)^2$$

$$= ln(C) + ln(\pi_k) + 0 - \frac{1}{2}ln(2\pi) - ln(\sigma_k) - \frac{1}{2{\sigma_k}^2}(x-\mu_k)^2$$

, absorb terms that don't depend on the response "k" into constant "C'":

$$C' = ln(C) - \frac{1}{2}ln(2\pi)$$

$$P(Y = k|X = x) = C' + ln(\pi_k) - ln(\sigma_k) - \frac{1}{2{\sigma_k}^2}(x - \mu_k)^2$$

$$P(Y = k|X = x) = C' + ln(\pi_k) - ln(\sigma_k) - \frac{1}{2{\sigma_k}^2}(x^2 - 2x\mu_k + {\mu_k}^2)$$

$$P(Y = k|X = x) = C' + ln(\pi_k) - ln(\sigma_k) - \frac{{\mu_k}^2}{2{\sigma_k}^2} - \frac{x^2}{2{\sigma_k}^2} + \frac{\mu_k x}{2{\sigma_k}^2}$$

We define the objective as:

$\delta_k(x) = ln(\pi_k) - ln(\sigma_k) - \frac{{\mu_k}^2}{2{\sigma_k}^2} - \frac{x^2}{2{\sigma_k}^2} + \frac{\mu_k x}{2{\sigma_k}^2}$, where at an input x we predict the response with the highest $\delta_k(x)$

The decision boundary in this case is the set of points in which 2 classes do just as well:

$$\delta_k(x) = \delta_l(x)$$

$$ln(\pi_k) - ln(\sigma_k) - \frac{{\mu_k}^2}{2{\sigma_k}^2} - \frac{x^2}{2{\sigma_k}^2} + \frac{\mu_k x}{2{\sigma_k}^2} = ln(\pi_l) - ln(\sigma_l) - \frac{{\mu_l}^2}{2{\sigma_l}^2} - \frac{x^2}{2{\sigma_l}^2} + \frac{\mu_l x}{2{\sigma_l}^2}$$

From the equation above, we can see the boundary is quadratic in x (and is not linear in x). We have therefore shown that the Bayes' classifier in this case is not linear, but rather is quadratic.

Problem #2:

Exercise 5 from section 4.7 of ITSL textbook

We now examine the differences between LDA and QDA.

(a) If the Bayes decision boundary is linear, do we expect LDA or QDA to perform better on the training set? On the test set?

Answer:
We would likely expect QDA to perform on the training set. However, LDA will likely perform better on the test set. On the training set, assuming there is sufficiently large noise in the data (i.e. noise with respect to the true population level signal that is trying to be captured), QDA will be more flexible in mistakenly fitting this noise than LDA. This will lead to QDA likely fitting the training set more closely. However if the boundary is truly linear, when testing against an outside test-set, LDA will likely perform better.

(b) If the Bayes decision boundary is non-linear, do we expect LDA or QDA to perform better on the training set? On the test set?

Answer:
In this case, we would likely expect QDA to perform better than LDA on both the training set and test set. Because the decision boundary is truly non-linear, we know LDA is parametrically missspecified. And even if the decision boundary is not quadratic, QDA will still likely be able to more closely fit the true signal as compared to LDA (given the greater flexibility of QDA to fit quadratic boundaries). Therefore QDA will likely perform better on both the training and test sets.

(c) In general, as the sample size *n* increases, do we expect the test prediction accuracy of QDA relative to LDA to improve, decline, or be unchanged? Why?

Answer:
I think this is a bit of an odd question, because it depends on what the true decision boundary is? Here's what I will say: In the case that the true decision boundary is linear, then both LDA and QDA are correctly specified parametrically and as the sample size "n" approaches infinity, the prediction accuracy of QDA and LDA become identical. If the true decision boundary is quadratic, then as n approaches infinity the prediction accuracy of QDA continues to increase, while the prediction accuracy of LDA does not (because in this case, LDA would be parametrically misspecified).

(d) True or False: Even if the Bayes decision boundary for a given problem is linear, we will probably achieve a superior test error rate using QDA rather than LDA because QDA is flexible enough to model a linear decision boundary. Justify your answer.

Answer:
I would argue that in the case the decision boundary is truly linear, that QDA will still likely have a superior training error rate than LDA. Though I would argue this is **not the case for the test error rate**. Yes, it is true that QDA can model linear boundaries. In fact, even in the case that the true decision boundary is linear, QDA is still correctly specified from a parametric perspective. The difference is that, in the case of a true linear decision boundary, with QDA we are wasting degrees of freedom estimating additional parameters we don't actually need to estimate (i.e. we are estimating separate covariance structures in each class of "Y' when it is in fact sufficient to estimate a common covariance structure across all classes). These wasted degrees of freedom result in a likely poorer test error rate of QDA relative to LDA in the case of a linear decision boundary.

Problem #3:

Exercise 6 from section 4.7 of ITSL textbook

Suppose we collect data for a group of students in a statistics class with variables:

- $X_1 = hours\ studied$
- $X_2 = undergrad\ GPA$
- $Y = receive\ an\ "A"$

We fit a logistic regression and produce estimated coefficient:

- $\hat{\beta}_0 = -6$
- $\hat{\beta}_1 = 0.05$
- $\hat{\beta}_2 = 1$

(a) Estimate the probability that a student who studies for 40 h and has an undergrad GPA of 3.5 gets an A in the class.

Answer:
The logistic model that was fit is:

$$logit(Pr[Y = 1]) = ln\left(\frac{Pr[Y = 1]}{1 - Pr[Y = 1]}\right) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2$$
$$= -6 + 0.05(hours\ studied) + 1(undergrad\ GPA)$$
$$= -6 + 0.05(40) + 1(3.5)$$
$$= -6 + 2 + 3.5 = -0.5$$

$$ln\left(\frac{P[Y = 1]}{1 - P[Y = 1]}\right) = -0.5$$

$$P[Y = 1] = \frac{e^{-0.5}}{1 + e^{-0.5}} = 0.3775$$

A student who studies for 40 hours and has an undergraduate GPA of 3.5 has an estimated probability of 37.75% of receiving an A in the class.

(b) How many hours would the student in part (a) need to study to have a 50% chance of getting an A in the class?

Answer:

$$ln\left(\frac{Pr[Y=1]}{1-Pr[Y=1]}\right)=\hat{\beta}_0+\hat{\beta}_1X_1+\hat{\beta}_2X_2$$

$$ln\left(\frac{0.5}{1-0.5}\right)=-6+0.05X_1+3.5$$

$$ln(1)=-6+0.05X_1+3.5$$

$$0=-6+0.05X_1+3.5$$

$$0.05X_1=2.5$$

$$X_1=50$$

A student with an undergraduate GPA of 3.5 needs to study 50 hours to have an estimated probability of 50% of receiving an A in the class.

<u>**Problem #4**</u>:

Exercise 9 from section 4.7 of ITSL textbook

This problem has to do with *odds*.

(a) On average, what fraction of people with an odds of 0.37 of defaulting on their credit card payment will in fact default?

Answer:

$$Odds = \frac{Probability}{1 - Probability}$$

$$(Odds) * (1 - Probability) = Probability$$

$$(Odds * 1) - (Odds * Probability) = Probability$$

$$Probability + (Odds * Probability) = Odds$$

$$Probability = \frac{Odds}{1 + Odds}$$

$$Probability = \frac{0.37}{1 + 0.37} \approx 0.2701$$

Approximately 27.01% will default

(b) Suppose that an individual has a 16% chance of defaulting on her credit card payment. What are the odds that she will default?

Answer:

$$Odds = \frac{Probability}{1 - Probability}$$

$$Odds = \frac{0.16}{1 - 0.16} \approx 0.1905$$

The odds that she will default is approximately 0.1905

Problem #5:

Exercise 10 from section 4.7 of ITSL textbook

This question should be answered using the "Weekly" data set, which is part of the "ISLR" package. This data is similar in nature to the "Smarket" data from this chapter's lab, except that it contains 1,089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

(a) Produce some numerical and graphical summaries of the "Weekly" data. Do there appear to be any patterns?

Answer:

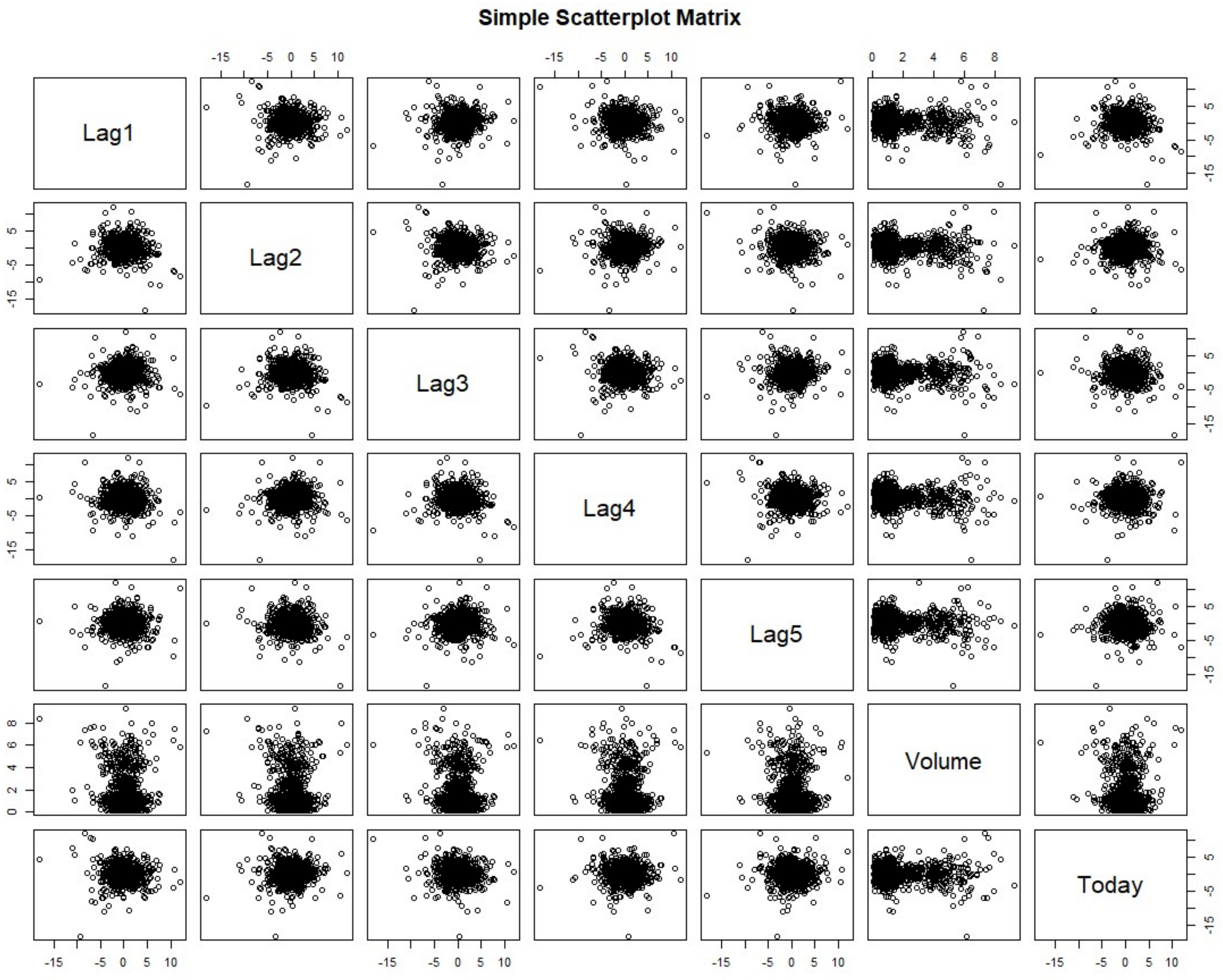
Descriptive Statistics of "Weekly" dataset (n=1089)

Variable	Mean	Standard Deviation	Median	Min	Max
Lag1	0.150585	2.357013	0.241	-18.195	12.026
Lag2	0.151079	2.357254	0.241	-18.195	12.026
Lag3	0.147205	2.360502	0.241	-18.195	12.026
Lag4	0.145818	2.360279	0.238	-18.195	12.026
Lag5	0.139893	2.361285	0.234	-18.195	12.026
Volume	1.574618	1.686636	1.00268	0.087465	9.328214
Today	0.149899	2.356927	0.241	-18.195	12.026

Regarding descriptive statistics for discrete variables "Year" and "Direction":

- "Year" ranges all years from 1990 to 2010, with 1990 containing 47 observations and all other years containing 52 observations
- For "Direction", 484 observations are classified as "Down" with the remaining 605 observations classified as "Up"

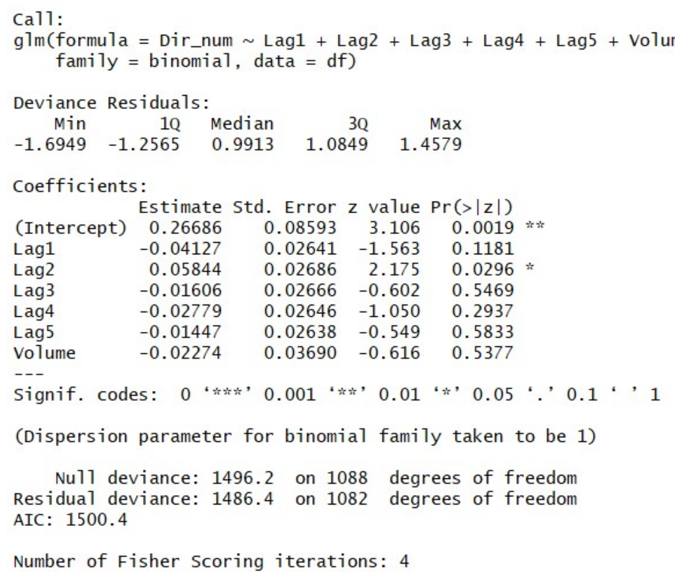
Shown below are scatterplots of the continuous variables shown in the table above.



In terms of potential patterns in the data, looking at the scatterplots above it seems there is some (perhaps) general direct relationship between all of the five “lag” variables. There also appears to (perhaps) be a direct relationship between the variables “Today” and the five lag variables.

(b) Use the full data set to perform a logistic regression with "Direction" as the response and the five lag variables plus "Volume" as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

Answer:
Shown below is the summary output from the logistic regression:



```
Call:
glm(formula = Dir_num ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
    family = binomial, data = df)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6949  -1.2565   0.9913   1.0849   1.4579

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.26686    0.08593   3.106   0.0019 **
Lag1        -0.04127    0.02641  -1.563   0.1181
Lag2         0.05844    0.02686   2.175   0.0296 *
Lag3        -0.01606    0.02666  -0.602   0.5469
Lag4        -0.02779    0.02646  -1.050   0.2937
Lag5        -0.01447    0.02638  -0.549   0.5833
Volume      -0.02274    0.03690  -0.616   0.5377
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1496.2  on 1088  degrees of freedom
Residual deviance: 1486.4  on 1082  degrees of freedom
AIC: 1500.4

Number of Fisher Scoring iterations: 4
```

As can be seen in the output above, the predictor "Lag2" is statistically significant at an alpha threshold of alpha=0.05 for the null hypothesis $H_0: \beta_{Lag2} = 0$, with a calculated p-value of 0.0296.

(c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

Answer:
Of the predicted probabilities of a given observation being classified as "Up" from the output of the logistic regression model, estimated probabilities >0.5 were classified as "Up" with those ≤0.5 classified as "Down". The resulting confusion matrix is shown below:

		Actual	
		Down	Up
Predicted	Down	54	48
	Up	430	557

From the confusion matrix above, the fraction of correct predictions is:

$$(54 + 557)/_{1089} \approx 0.5611$$

The fraction of correction predictions made by the logistic regression is approximately 56.11%

Regarding the "mistakes" the logistic regression (and 0.5 cutoff) is making classifying each observation into the "Down" or "Up" class:

- The probability of the model predicting a false positive (i.e. predicting "Up" when in fact the true class is "Down") is: $^{430}/_{(54+430)} \approx 0.8884$
- The probability of the model predicting a false negative (i.e. predicting "Down" when in fact the true class is "Up") is: $^{48}/_{(48+557)} \approx 0.0793$

(d) Now fit the logistic regression model using a training data period from 1990 to 2008, with "Lag2" as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 to 2010).

Answer:
The resulting confusion matrix of the logistic regression is shown below:

		Actual	
		Down	Up
Predicted	Down	9	5
	Up	34	56

From the confusion matrix above, the fraction of correct predictions is:

$$^{(9+56)}/_{104} = 0.625$$

(e) Repeat (d) using LDA.

Answer:
The resulting confusion matrix of the LDA is shown below:

		Actual	
		Down	Up
Predicted	Down	9	5
	Up	34	56

From the confusion matrix above, the fraction of correct predictions is:

$$^{(9+56)}/_{104} = 0.625$$

(f) Repeat (d) using QDA.

Answer:
The resulting confusion matrix of the QDA is shown below:

		Actual	
		Down	Up
Predicted	Down	0	0
	Up	43	61

From the confusion matrix above, the fraction of correct predictions is:

$$^{61}/_{104} \approx 0.5865$$

(g) Repeat (d) using KNN with K=1.

Answer:
The resulting confusion matrix of the KNN with K=1 is shown below:

		Actual	
		Down	Up
Predicted	Down	21	29
	Up	22	32

From the confusion matrix above, the fraction of correct predictions is:

$$^{(21+32)}/_{104} \approx 0.5096$$

(h) Which of these methods appears to provide the best results on this data?

Answer:
I find this to be a poorly specified question, given that the resulting confusion matrices for logistic, LDA, and QDA are enumerated based on an assumed cutoff of 0.5 to specify estimated probabilities from these models as belonging to class “Up” or “Down”. Specification of this cutoff point to a value other than 0.5 may (and will most likely) result in different confusion matrices.

Assuming we are using a 0.5 cutoff for logistic, LDA, and QDA (which I will remined the reader has NOT been explicitly specified in the problem statement…), the method(s) that provide the best results appear to be both logistic and LDA. Both approaches resulted in identical confusion matrices, and the same fraction of correct predictions in the test set (62.5%), which was higher than in QDA or KNN.

(i) Experiment with different combinations of predictors, including possible transformations and interactions, for each of the methods. Report the variables, method, and associated confusion matrix that appears to provide the best results on the held out data. Note that you should also experiment with values for K in the KNN classifier.

Answer:
For this problem, I took a bit of a "brute-force" approach. I first looked at any combinations of the following 6 variables:

- Lag1
- Lag2
- Lag3
- Lag4
- Lag5
- Volume

Ranging from one variable at a time, to all 6 variables, and any combination in-between. Then using that given set of combination of predictors, Logistic Regression, LDA, QDA, and KNN (with k=1,5,10,20,50) were all fit on the training data and tested on the testing data.

Inspection showed that the best performing model was KNN (with k=20) using predictors "Lag1, Lag2, and Lag3". The fraction of observations in the test set classified with the correct outcome using this method was 62.5%.

To see if we could increase the accuracy, additional models were explored using interaction terms and data transformations using only variables "Lag1, Lag2, and Lag3".

For this next part of the analysis the following 9 variables were specified:

- Lag1
- Lag2
- Lag3
- $(\text{Lag1})^3$
- $(\text{Lag2})^3$
- $(\text{Lag3})^3$
- Lag1*Lag2
- Lag1*Lag3
- Lag2*Lag3

Ranging from one variable at a time, to all 9 variables, and any combination in-between, Logistic Regression, LDA, QDA, and KNN (with k=1,5,10,20,50) were all fit on the training data and tested on the testing data.

The best performing model was KNN (with k=10) using predictors "Lag1, Lag2^3, and Lag2*Lag3". The fraction of observations in the test set classified with the correct outcome using this method was approximately 68.269%.

The confusion matrix for this method (KNN with k=10 among predictors "Lag1, Lag2^3, and Lag2*Lag3") is shown below:

		Actual	
		Down	Up
Predicted	Down	25	15
	Up	18	46

R Code:
```
################
## Problem #5 ##
################
library(ISLR)
data(Weekly)
df <- Weekly
??Weekly

## Part A
library(psych)

dim(df)
head(df)
colnames(df)

summary(df)
describe(df)
table(df$Year, exclude=NULL)
table(df$Direction, exclude=NULL)

write.csv(describe(df), "D:\\classes\\Stanford\\STATS 202 (Fall
2018)\\Homework\\HW#3\\descrip_tableA.csv")

pairs(~Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume + Today, data=df, main="Simple Scatterplot
Matrix")


## Part B
df$Dir_num <- 0
df$Dir_num[df$Direction=='Up'] <- 1
table(df$Direction, exclude=NULL)
table(df$Dir_num, exclude=NULL)
table(df$Direction, df$Dir_num, exclude=NULL)

summary(glm(Dir_num ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, family=binomial, data=df))

## Part C
df$Dir_probability <- predict(glm(Dir_num ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
family=binomial, data=df), type="response")
df$Dir_pred_class <- 0
df$Dir_pred_class[df$Dir_probability > 0.5] <- 1

table(df$Dir_pred_class, df$Direction)


## Part D
library(dplyr)

head(df)
df_train <- dplyr::filter(dplyr::select(df, -Dir_probability, -Dir_pred_class), Year<=2008)
df_test <- dplyr::filter(dplyr::select(df, -Dir_probability, -Dir_pred_class), Year>2008)
```

```
df_test$logistic_p <- predict(glm(Dir_num ~ Lag2, family=binomial, data=df_train), df_test,
type="response")
df_test$logistic_class <- 0
df_test$logistic_class[df_test$logistic_p > 0.5] <- 1

table(df_test$logistic_class, df_test$Direction)

## Part E
library(MASS)
df_test$lda_class <- predict(lda(Dir_num ~ Lag2, data=df_train), df_test)$class
table(df_test$lda_class, df_test$Direction)

## Part F
df_test$qda_class <- predict(qda(Dir_num ~ Lag2, data=df_train), df_test)$class
table(df_test$qda_class, df_test$Direction)


## Part G
library(class)
set.seed(1234)
Y_train <- as.matrix(df_train$Dir_num)
X_train <- as.matrix(df_train$Lag2)
X_test <- as.matrix(df_test$Lag2)

df_test$KNN_class <- knn(X_train, X_test, Y_train, k=1)
table(df_test$KNN_class, df_test$Direction)




## Part I
CV_results <- function(df, comb_matrix){

  num_variables <- rep(dim(comb_matrix)[1],dim(comb_matrix)[2])

  results_df <- data.frame(num_variables)
  rm(num_variables)
  results_df$var_string <- NA
  results_df$CV_logit_frac <- NA
  results_df$CV_LDA_frac <- NA
  results_df$CV_QDA_frac <- NA
  results_df$CV_KNN_1_frac <- NA
  results_df$CV_KNN_5_frac <- NA
  results_df$CV_KNN_10_frac <- NA
  results_df$CV_KNN_20_frac <- NA
  results_df$CV_KNN_50_frac <- NA

  for(i in 1:(dim(comb_matrix)[2])){
    print(i)

    ## take the "ith" column of covariates from the "covariate matrix"
    cov_sub <- comb_matrix[,i]
    results_df$var_string[i] <- paste(cov_sub, collapse=" + ")

    ## create training and testing sets:
    df_train <- dplyr::select(dplyr::filter(df, Year<=2008), Dir_num, cov_sub)
    df_test <- dplyr::select(dplyr::filter(df, Year>2008), Dir_num, cov_sub)

    ## Fit the logistic regression:
    df_test$pred_logit <- predict(glm(Dir_num ~ ., family=binomial, data=df_train), df_test,
type="response")
    df_test$pred_logit_class <- 0
    df_test$pred_logit_class[df_test$pred_logit > 0.5] <- 1
    cm_logit <- table(df_test$pred_logit_class, df_test$Dir_num)
    results_df$CV_logit_frac[i] <- dim(dplyr::filter(df_test,Dir_num==pred_logit_class))[1] /
dim(df_test)[1]
```

```
    rm(cm_logit)

    ## Fit the LDA:
    df_test$pred_LDA <- predict(lda(Dir_num ~ ., data=df_train), df_test)$class
    cm_LDA <- table(df_test$pred_LDA, df_test$Dir_num)
    #results_df$CV_LDA_frac[i] <- (cm_LDA[1,1]+cm_LDA[2,2])/sum(cm_LDA)
    results_df$CV_LDA_frac[i] <- dim(dplyr::filter(df_test,Dir_num==pred_LDA))[1] /
dim(df_test)[1]
    rm(cm_LDA)

    ## Fit the QDA:
    df_test$pred_QDA <- predict(qda(Dir_num ~ ., data=df_train), df_test)$class
    cm_QDA <- table(df_test$pred_QDA, df_test$Dir_num)
    #results_df$CV_QDA_frac[i] <- (cm_QDA[1,1]+cm_QDA[2,2])/sum(cm_QDA)
    results_df$CV_QDA_frac[i] <- dim(dplyr::filter(df_test,Dir_num==pred_QDA))[1] /
dim(df_test)[1]
    rm(cm_QDA)

    ## fit the KNN and output prediction
    Y_train <- as.matrix(df_train$Dir_num)
    X_train <- as.matrix(dplyr::select(df_train, cov_sub))
    X_test <- as.matrix(dplyr::select(df_test, cov_sub))
    ## for k=1
    df_test$pred_KNN_1 <- knn(X_train, X_test, Y_train, k=1)
    cm_KNN_1 <- table(df_test$pred_KNN_1, df_test$Dir_num)
    results_df$CV_KNN_1_frac[i] <- dim(dplyr::filter(df_test,Dir_num==pred_KNN_1))[1] /
dim(df_test)[1]
    rm(cm_KNN_1)
    ## for k=5
    df_test$pred_KNN_5 <- knn(X_train, X_test, Y_train, k=5)
    cm_KNN_5 <- table(df_test$pred_KNN_5, df_test$Dir_num)
    results_df$CV_KNN_5_frac[i] <- dim(dplyr::filter(df_test,Dir_num==pred_KNN_5))[1] /
dim(df_test)[1]
    rm(cm_KNN_5)
    ## for k=10
    df_test$pred_KNN_10 <- knn(X_train, X_test, Y_train, k=10)
    cm_KNN_10 <- table(df_test$pred_KNN_10, df_test$Dir_num)
    results_df$CV_KNN_10_frac[i] <- dim(dplyr::filter(df_test,Dir_num==pred_KNN_10))[1] /
dim(df_test)[1]
    rm(cm_KNN_10)
    ## for k=20
    df_test$pred_KNN_20 <- knn(X_train, X_test, Y_train, k=20)
    cm_KNN_20 <- table(df_test$pred_KNN_20, df_test$Dir_num)
    results_df$CV_KNN_20_frac[i] <- dim(dplyr::filter(df_test,Dir_num==pred_KNN_20))[1] /
dim(df_test)[1]
    rm(cm_KNN_20)
    ## for k=50
    df_test$pred_KNN_50 <- knn(X_train, X_test, Y_train, k=50)
    cm_KNN_50 <- table(df_test$pred_KNN_50, df_test$Dir_num)
    results_df$CV_KNN_50_frac[i] <- dim(dplyr::filter(df_test,Dir_num==pred_KNN_50))[1] /
dim(df_test)[1]
    rm(cm_KNN_50)

  }

  return(results_df)
}


df$Lag1_cubed <- df$Lag1^3
df$Lag2_cubed <- df$Lag2^3
df$Lag3_cubed <- df$Lag3^3
df$Lag4_cubed <- df$Lag4^3
df$Lag5_cubed <- df$Lag5^3
df$Volume_cubed <- df$Volume^3
```

```
df$Lag1_2 <- df$Lag1 * df$Lag2
df$Lag1_3 <- df$Lag1 * df$Lag3
df$Lag1_4 <- df$Lag1 * df$Lag4
df$Lag1_5 <- df$Lag1 * df$Lag5
df$Lag1_Volume <- df$Lag1 * df$Volume

df$Lag2_3 <- df$Lag2 * df$Lag3
df$Lag2_4 <- df$Lag2 * df$Lag4
df$Lag2_5 <- df$Lag2 * df$Lag5
df$Lag2_Volume <- df$Lag2 * df$Volume

df$Lag3_4 <- df$Lag3 * df$Lag4
df$Lag3_5 <- df$Lag3 * df$Lag5
df$Lag3_Volume <- df$Lag3 * df$Volume

df$Lag4_5 <- df$Lag4 * df$Lag5
df$Lag4_Volume <- df$Lag4 * df$Volume

df$Lag5_Volume <- df$Lag5 * df$Volume


covariate_labels <- c("Lag1", "Lag2", "Lag3", "Lag4", "Lag5", "Volume")



results_1 <- CV_results(df, combn(covariate_labels, 1))
results_2 <- CV_results(df, combn(covariate_labels, 2))
results_3 <- CV_results(df, combn(covariate_labels, 3))
results_4 <- CV_results(df, combn(covariate_labels, 4))
results_5 <- CV_results(df, combn(covariate_labels, 5))
results_6 <- CV_results(df, combn(covariate_labels, 6))

results_df <- rbind(results_1, results_2, results_3, results_4, results_5, results_6)



covariate_labels_sub <- c("Lag1", "Lag2", "Lag3",
                      "Lag1_cubed", "Lag2_cubed", "Lag3_cubed",
                      "Lag1_2", "Lag1_3", "Lag2_3")

results_1_sub <- CV_results(df, combn(covariate_labels_sub, 1))
results_2_sub <- CV_results(df, combn(covariate_labels_sub, 2))
results_3_sub <- CV_results(df, combn(covariate_labels_sub, 3))
results_4_sub <- CV_results(df, combn(covariate_labels_sub, 4))
results_5_sub <- CV_results(df, combn(covariate_labels_sub, 5))
results_6_sub <- CV_results(df, combn(covariate_labels_sub, 6))
results_7_sub <- CV_results(df, combn(covariate_labels_sub, 7))
results_8_sub <- CV_results(df, combn(covariate_labels_sub, 8))
results_9_sub <- CV_results(df, combn(covariate_labels_sub, 9))


results_df_sub <- rbind(results_1_sub, results_2_sub, results_3_sub, results_4_sub,
results_5_sub, results_6_sub, results_7_sub, results_8_sub, results_9_sub)
```

Problem #6:

Exercise 13 from section 4.7 of ITSL textbook

Using the “Boston” data set, fit classification models in order to predict whether a given suburb has a crime rate above or below the median. Explore logistic regression, LDA, and KNN models using various subsets of the predictors. Describe your findings.

Answer:

For this problem, I took a bit of a “brute-force” approach. I first examined the descriptive statistics of the Boston Dataset, including the “median” crime rate. Descriptive statistics are shown in the table below:

Descriptive Statistics of Boston Housing Dataset *(n=506)*

Variable Name	Description	Mean (SD)	Count (%)
crim	per capita crime rate by town	3.61 (8.60)	-
zn	proportion of residential land zoned for lots over 25,000 sq.ft	11.36 (23.32)	-
indus	proportion of non-retail business acres per town	11.14 (6.86)	-
chas	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)	0.07 (0.25)	-
nox	nitrogen oxides concentration (parts per 10 million)	0.55 (0.12)	-
rm	average number of rooms per dwelling	6.28 (0.70)	-
age	proportion of owner-occupied units built prior to 1940	68.57 (28.15)	-
dis	weighted mean of distances to five Boston employment centres	3.80 (2.11)	-
rad	index of accessibility to radial highways	9.55 (8.71)	-
tax	full-value property-tax rate per \$10,000	408.24 (168.54)	-
ptratio	pupil-teacher ratio by town	18.46 (2.16)	-
black	1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town	356.67 (91.29)	-
lstat	lower status of the population (percent)	12.65 (7.14)	-
medv	median value of owner-occupied homes in \$1000s	22.53 (9.20)	-
outcome	crime rate above the median *(median=0.25651)*	-	253 (50%)

From the table above, there are a total of 13 possible predictor variables with the "outcome" of interest being "crime rate above the median crime rate *(0.25651)*". To determine the best fitting prediction model for said outcome using the above predictors, the following brute-force approach was taken:

1. Every linear combination of the 13 predictors was determined and listed (from 1 predictor to all 13 predictors, and every combination in-between). In total there are 8,191 combinations of the 13 predictors (i.e. $\sum_{i=1}^{13} \binom{13}{i} = 8191$)
2. For all 8191 combinations of the 13 predictors, the following 7 models were run and assessed on training/testing sets via 5-fold cross validation:
   - Logistic Regression
   - Linear Discriminant Analysis
   - K Nearest Neighbors *(with k=1)*
   - K Nearest Neighbors *(with k=5)*
   - K Nearest Neighbors *(with k=10)*
   - K Nearest Neighbors *(with k=20)*
   - K Nearest Neighbors *(with k=50)*
3. With a probability cutoff of 0.5, the confusion-matrix and fraction of correctly predicted outcomes for each cross-validated model were enumerated. The average of the enumerated fractions of correctly predicted outcomes (i.e. the "Cross Validated scores") were enumerated and stored.
4. The maximum prediction score for each model type for each number of variables used (1-13) is shown below

Note, that for this analysis, only the possible additive combinations of the 13 predictor variables were considered. Any transformations or interactions terms between these 13 variables were not considered for this analysis.

Maximum Cross-Validated Score

Number of predictors in the model	Logistic Regression	LDA	KNN *(k=1)*	KNN *(k=5)*	KNN *(k=10)*	KNN *(k=20)*	KNN *(k=50)*
1	0.824502	0.826344	0.950685	0.936753	0.901014	0.887112	0.818788
2	0.865154	0.851826	0.958538	0.94858	0.942163	0.934381	0.856113
3	0.871371	0.863922	0.958538	0.956598	0.952425	0.942704	0.854071
4	0.883807	0.870644	0.961184	0.962276	0.970365	0.915728	0.858419
5	0.890878	0.872749	0.963053	0.962276	0.96826	0.909747	0.862193
6	0.893356	0.870644	0.963053	0.958502	0.965746	0.904688	0.861957
7	0.897602	0.872749	0.963053	0.958502	0.964181	0.904617	0.857207
8	0.909363	0.872749	0.963053	0.954693	0.953987	0.898938	0.859381
9	0.911301	0.870644	0.961184	0.952486	0.942739	0.877494	0.859381
10	0.913105	0.864262	0.960676	0.950617	0.940565	0.873077	0.859381
11	0.910428	0.861957	0.95099	0.946102	0.936827	0.873852	0.859381
12	0.909028	0.857878	0.937121	0.935359	0.920721	0.871173	0.855338
13	0.907228	0.851425	0.922823	0.921157	0.885207	0.865565	0.828898

From the results above, it appears KNN with the (k-parameter specified somewhere between 1 to 10) in general performed better than Logistic Regression or LDA. For each of the seven model types, the actual model with the best Cross-validated Score is show below:

Model Type	Number of Predictors	List of Predictors	Cross-Validated Score
Logistic Regression	10	zn, indus, nox, age, dis, rad, tax, ptratio, black, medv	0.9131046
LDA	5	zn, nox, rad, tax, medv	0.8727494
KNN *(k=1)*	5	zn, indus, rm, dis, rad	0.9630533
KNN *(k=5)*	4	indus, rm, dis, rad	0.9622764
KNN *(k=10)*	4	indus, dis, rad, ptratio	0.9703651
KNN *(k=20)*	3	chas, nox, rad	0.9427036
KNN *(k=50)*	5	Chas, rm, dis, rad, ptratio	0.8621926

As show in the table above, the model that produced the highest Cross-Validated Score (i.e. average of 5-fold cross-validation of percentage of binary outcomes classified correctly) is the K-Nearest Neighbors model with 4 variables included as predictors *(indus, dis, rad, and ptratio)* and the k-parameter specified to k=10. The estimated Cross-Validated Score from the analysis above for this model was 97.03651%

I would like to note, I find this a bit of an odd HW question. If I were approaching this problem in practice, and not for the purposes of a HW problem, I would approach it quite differently. A couple of thoughts:

- Because we have not yet covered any stepwise selection or regularization methods (LASSO, Ridge, Elastic Net) in the course, I purposely did not use those methods here. In practice, I would almost certainly try regularization approaches paired with some of the methods used above. We have also not yet covered tree-based methods or GAMs (which I know we are planning to cover later in the course), and purposely did not use those methods here either.
- Additionally, all of the methods used in the analysis above force me (as the analyst) to specify a-priori what variables are in the model and (at least in the case of logistic regression) specify certain aspects of the functional form of the model. If I was approaching this problem from purely a prediction standpoint, did not care about the descriptive interpretability of the model parameters, and had little a-priori knowledge about the functional form of the model, I would probably try some (at least shallow) Neural Network approaches.

R Code:

```
CV_results <- function(df, comb_matrix){

  num_variables <- rep(dim(comb_matrix)[1],dim(comb_matrix)[2])

  results_df <- data.frame(num_variables)
  rm(num_variables)
  results_df$var_string <- NA
  results_df$CV_logit_frac <- NA
  results_df$CV_LDA_frac <- NA
  results_df$CV_KNN_1_frac <- NA
  results_df$CV_KNN_5_frac <- NA
  results_df$CV_KNN_10_frac <- NA
  results_df$CV_KNN_20_frac <- NA
  results_df$CV_KNN_50_frac <- NA

  for(i in 1:(dim(comb_matrix)[2])){
    print(i)

    ## take the "ith" column of covariates from the "covariate matrix"
    cov_sub <- comb_matrix[,i]
    results_df$var_string[i] <- paste(cov_sub, collapse=" + ")

    ## create "sub" dataframe with the appropriate covariates and the outcome
    df_sub <- dplyr::select(df, outcome, cov_sub)

    ## create the for loop for 5-fold CV:

    logit_frac <- rep(NA, 5)
    LDA_frac <- rep(NA, 5)
    KNN_1_frac <- rep(NA, 5)
    KNN_5_frac <- rep(NA, 5)
    KNN_10_frac <- rep(NA, 5)
    KNN_20_frac <- rep(NA, 5)
    KNN_50_frac <- rep(NA, 5)

    for(j in 1:5){
      df_train <- dplyr::select(dplyr::filter(df, CV_id!=j), outcome, cov_sub)
      df_test <- dplyr::select(dplyr::filter(df, CV_id==j), outcome, cov_sub)
      ## fit the logistic model and output prediction
      df_test$pred_logit <- predict(glm(outcome ~ ., family=binomial, data=df_train), df_test,
type="response")
      df_test$pred_logit_class <- 0
      df_test$pred_logit_class[df_test$pred_logit > 0.5] <- 1
      cm_logit <- table(df_test$pred_logit_class, df_test$outcome)
      logit_frac[j] <- (cm_logit[1,1]+cm_logit[2,2])/sum(cm_logit)
      rm(cm_logit)

      ## fit the LDA and output prediction
      df_test$pred_LDA <- predict(lda(outcome ~ ., data=df_train), df_test)$class
      cm_LDA <- table(df_test$pred_LDA, df_test$outcome)
      LDA_frac[j] <- (cm_LDA[1,1]+cm_LDA[2,2])/sum(cm_LDA)
      rm(cm_LDA)

      ## fit the KNN and output prediction
      Y_train <- as.matrix(df_train$outcome)
      X_train <- as.matrix(dplyr::select(df_train, cov_sub))
      X_test <- as.matrix(dplyr::select(df_test, cov_sub))
      ## for k=1
      df_test$pred_KNN_1 <- knn(X_train, X_test, Y_train, k=1)
      cm_KNN_1 <- table(df_test$pred_KNN_1, df_test$outcome)
      KNN_1_frac[j] <- (cm_KNN_1[1,1]+cm_KNN_1[2,2])/sum(cm_KNN_1)
      rm(cm_KNN_1)
      ## for k=5
      df_test$pred_KNN_5 <- knn(X_train, X_test, Y_train, k=5)
      cm_KNN_5 <- table(df_test$pred_KNN_5, df_test$outcome)
      KNN_5_frac[j] <- (cm_KNN_5[1,1]+cm_KNN_5[2,2])/sum(cm_KNN_5)
      rm(cm_KNN_5)
      ## for k=10
      df_test$pred_KNN_10 <- knn(X_train, X_test, Y_train, k=10)
      cm_KNN_10 <- table(df_test$pred_KNN_10, df_test$outcome)
      KNN_10_frac[j] <- (cm_KNN_10[1,1]+cm_KNN_10[2,2])/sum(cm_KNN_10)
      rm(cm_KNN_10)
      ## for k=20
      df_test$pred_KNN_20 <- knn(X_train, X_test, Y_train, k=20)
      cm_KNN_20 <- table(df_test$pred_KNN_20, df_test$outcome)
      KNN_20_frac[j] <- (cm_KNN_20[1,1]+cm_KNN_20[2,2])/sum(cm_KNN_20)
      rm(cm_KNN_20)
      ## for k=50
      df_test$pred_KNN_50 <- knn(X_train, X_test, Y_train, k=50)
      cm_KNN_50 <- table(df_test$pred_KNN_50, df_test$outcome)
      KNN_50_frac[j] <- (cm_KNN_50[1,1]+cm_KNN_50[2,2])/sum(cm_KNN_50)
      rm(cm_KNN_50)
      rm(Y_train, X_train, X_test)

      rm(df_train, df_test)
    }

    results_df$CV_logit_frac[i] <- mean(logit_frac)
    results_df$CV_LDA_frac[i] <- mean(LDA_frac)
    results_df$CV_KNN_1_frac[i] <- mean(KNN_1_frac)
    results_df$CV_KNN_5_frac[i] <- mean(KNN_5_frac)
    results_df$CV_KNN_10_frac[i] <- mean(KNN_10_frac)
    results_df$CV_KNN_20_frac[i] <- mean(KNN_20_frac)
    results_df$CV_KNN_50_frac[i] <- mean(KNN_50_frac)

    rm(logit_frac, LDA_frac, KNN_1_frac, KNN_5_frac, KNN_10_frac, KNN_20_frac, KNN_50_frac)
  }

  return(results_df)
}
```

```
################
## Problem #6 ##
################
library(MASS)
library(dplyr)
library(psych)
set.seed(1234)
df <- Boston

## explore the Boston dataset
dim(df)
head(df)
colnames(df)
?Boston
summary(df)

## specify the median of the crime variable
median_cr <- median(df$crim)

## add in CV ineger varibal for later Cross Validation
df$CV_id <- sample(1:5, dim(df)[1], replace=TRUE)
table(df$CV_id, exclude=NULL)

## specify the binary outcome
df$outcome <- 0
df$outcome[df$crim>median_cr] <- 1
table(df$outcome, exclude=NULL)

covariate_labels <- c("zn", "indus", "chas", "nox", "rm", "age", "dis", "rad", "tax", "ptratio",
"black", "lstat", "medv")

################################################################
################################################################
## for the 13 possible covariates                             ##
## take all possible linear combinations of those covariates  ##
## and perorm:                                                ##
## Logistic Regression                                        ##
## LDA                                                        ##
## KNN with k=1,5,10,20,50                                    ##
## using 5-fold cross-validation                              ##
################################################################
################################################################

results_1 <- CV_results(df, combn(covariate_labels, 1))  ## 13 possible combiations
results_2 <- CV_results(df, combn(covariate_labels, 2))  ## 78 possible combiations
results_3 <- CV_results(df, combn(covariate_labels, 3))  ## 286 possible combiations
results_4 <- CV_results(df, combn(covariate_labels, 4))  ## 715 possible combiations
results_5 <- CV_results(df, combn(covariate_labels, 5))  ## 1287 possible combiations
results_6 <- CV_results(df, combn(covariate_labels, 6))  ## 1716 possible combiations
results_7 <- CV_results(df, combn(covariate_labels, 7))  ## 1716 possible combiations
results_8 <- CV_results(df, combn(covariate_labels, 8))  ## 1287 possible combiations
results_9 <- CV_results(df, combn(covariate_labels, 9))  ## 715 possible combiations
results_10 <- CV_results(df, combn(covariate_labels, 10))  ## 286 possible combiations
results_11 <- CV_results(df, combn(covariate_labels, 11))  ## 78 possible combiations
results_12 <- CV_results(df, combn(covariate_labels, 12))  ## 13 possible combiations
results_13 <- CV_results(df, combn(covariate_labels, 13))  ## 1 possible combiations

results_df <- rbind(results_1, results_2, results_3, results_4, results_5, results_6, results_7,
results_8, results_9, results_10, results_11, results_12, results_13)

head(dplyr::arrange(results_df, desc(CV_logit_frac)))
head(dplyr::arrange(results_df, desc(CV_LDA_frac)))
head(dplyr::arrange(results_df, desc(CV_KNN_1_frac)))
head(dplyr::arrange(results_df, desc(CV_KNN_5_frac)))
head(dplyr::arrange(results_df, desc(CV_KNN_10_frac)))
head(dplyr::arrange(results_df, desc(CV_KNN_20_frac)))
head(dplyr::arrange(results_df, desc(CV_KNN_50_frac)))

results_aggregated <- results_df %>% group_by(num_variables) %>% summarise(max_logit =
max(CV_logit_frac), max_LDA = max(CV_LDA_frac), max_KNN1 = max(CV_KNN_1_frac), max_KNN5 =
max(CV_KNN_5_frac), max_KNN10 = max(CV_KNN_10_frac), max_KNN20 = max(CV_KNN_20_frac), max_KNN50 =
max(CV_KNN_50_frac))
write.csv(results_aggregated, "D:\\classes\\Stanford\\STATS 202 (Fall
2018)\\Homework\\HW#3\\results_aggregated.csv")
```