STAT 639 HW 3

Jack Cunningham (jgavc@tamu.edu)

4/2/24

1)

a) K-fold cross-validation is implemented by splitting the observed data into k non-overlapping sets with close to equal size called folds. For each step in the procedure a fold is kept out as a validation set. A chosen model is fit using the k-1 remaining folds and evaluated using the validation set. The evaluation criteria depends on the setting and objective. For instance a regression problem may utilize mean-squared error, while a classification problem may use the error rate. This procedure is done k times and each of the k folds is used as a validation set once. The evaluation metric is then averaged and returned.

b)

i) The k-fold cross-validation approach has notable advantages over the validation set approach.

Firstly, the validation set approach struggles with the variability inherent to the selection of the held out set. For example consider 20% of the data is held out. There are instances where the selected 20% are on the higher end, where they are on the lower end, where outliers are selected and so on. If the same procedure was ran again results could be wildly different depending on the randomly selected validation set. The k-fold cross-validation approach on the other hand uses each of the k splits as a validation set and averages the evaluation criteria. This reduces the variance of the approach in its estimate of test error.

Secondly, the validation set approach often overestimates the test error we would see if the entire data set was used to train. Continuing the previous example, only 80% of our observed data was utilized in training our model and 20% was held out. There is a balance between having enough data to adequately train the model and enough to adequately evaluate it. The k-fold cross-validation approach allows us to utilize all of the observed data to train the model and thus does not tend to overestimate the test error.

The main disadvantage the k-fold cross-validation approach has is its increased computational requirement. With the validation set approach a model needs to be fit only one time, the K-fold cross validation approach requires it be fit k times. Depending on the computational complexity of a model this can be restrictive. But with a reasonable number k chosen this

disadvantage can be adjusted for. Additionally for many models the computational tax being paid is not too high.

c)

i) The LOOCV approach is a special version of the K-fold cross validation approach where k = n.

The first strength of the LOOCV approach is its sensibility and thus its lack of bias in estimating test error. The LOOCV approach uses everything but one observation each time a model is fit and evaluates its performance on that one observation. This is repeated n times and the evaluation criteria is averaged out. It is similar to the actual process of seeing a new observation and attempting to estimate it, we are using all the other information possible to make our prediction. Therefore the estimate of test error is approximately unbiased.

The benefit of being unbiased can become a curse in many scenarios. The trade-off between bias and variance is a paramount decision for creating a prediction. The variance of the LOOCV approach is high due to the n fitted models all being very similar, therefore correlated, to each other. The average of highly correlated quantities has high variance compared to moderately correlated quantities. The k-fold-cross-validation approach strikes more of a middle ground between bias and variance. In many cases this is an advantage in its ability to estimate test error.

A main drawback of the LOOCV approach is its computational intensity. As previously discussed the LOOCV approach fits a model n times. This makes it unrealistic for situations where a model is computationally taxing or n is large. K-fold cross-validation has a clear advantage in this area, the difference between fitting a model n times and 5 or 10 times (commonly selected values of k) is a world of difference computationally.

2)

a)

```
lm.fit <- glm(default ~ income + balance, data = Default, family = binomial)
summary(lm.fit)$coefficients
```

```
                 Estimate   Std. Error    z value      Pr(>|z|)
(Intercept) -1.154047e+01 4.347564e-01 -26.544680 2.958355e-155
income       2.080898e-05 4.985167e-06   4.174178  2.990638e-05
balance      5.647103e-03 2.273731e-04  24.836280 3.638120e-136
```

After fitting a logistic model we see that the standard deviation of the income and balance estimate is 4.9852e-06 and 2.2737e-04 respectively.

b)

```
boot.fn <- function(data, index){
  coef(glm(default ~ income + balance, data = Default, family = binomial,
           subset = index))
}
```

c)

```
set.seed(5)
boot(Default, boot.fn, 1000)
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = Default, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
         original        bias     std. error
t1* -1.154047e+01 -3.787648e-02 4.380326e-01
t2*  2.080898e-05  2.733337e-07 4.909003e-06
t3*  5.647103e-03  1.512143e-05 2.332406e-04
```

d)

The standard errors obtained from the glm function and bootstrap function are very close to each other.

3)

a) iii. is correct. Least squares has zero bias if linear assumptions are met. The lasso introduces bias and is also a linear model, so it is less flexible. If the squared bias is less than the decrease in variance we would see improved predictions.

b) iii. is correct. Least squares has zero bias if linear assumptions are met. Ridge regression introduces bias and is also a linear model, so it is less flexible. If the squared bias is less than the decrease in variance we would see improved predictions.

c) ii. is correct. A non-linear method makes less assumptions about the form of the true function than a linear method, therefore it is more flexible. We would expect prediction accuracy to improve if the bias we reduced is more than the increase in variance.

4)

a) iv. Steadily decrease is correct. When s = 0 we have a null model with coefficients equal to zero. That is when bias is at its highest and variance is 0. As s increases the flexibility of the model is increased. Training RSS will decrease as flexibility increases, we are following the training data closer and heading towards the least squares estimate (which solely focuses on minimizes RSS).

b) ii. Decrease initially and then start increasing in a U shape is correct. . When s = 0 we have a null model with coefficients equal to zero. That is when bias is at its highest and variance is 0. As s increases the flexibility of the model is increased. Test RSS, similar to other test errors, will be minimized somewhere in the middle ground on the bias and variance trade-off. After that point the increased variance will outpace the reduction in bias and test RSS will increase.

c) iii. Steadily increase is correct. When s = 0 we have a null model with coefficients equal to zero. That is when bias is at its highest and variance is 0. As s increases the flexibility of the model is increased, naturally variance is also increased.

d) iv. Steadily decrease is correct. When s = 0 we have a null model with coefficients equal to zero. That is when bias is at its highest and variance is 0. As s increases the flexibility of the model is increased, naturally bias is decreasing.

e) v. Remain constant is correct. Irreducible error cannot, as the name suggests, be reduced through increasing the flexibility of a model.

5)

a)

Splitting data and creating necessary objects for further steps.

```
#|ooutput: false
library(ISLR2)
library(glmnet)
```

```
Warning: package 'glmnet' was built under R version 4.3.3
```

```
Loading required package: Matrix
```

```
Loaded glmnet 4.1-8
```

```
x <- model.matrix(Apps ~ ., College)[, -1]
y <- College$Apps

set.seed(11)
```

```
train <- sample(1:nrow(x), nrow(x) / 2)
test <- (-train)
y.test <- y[test]

College_train <- College[train, ]
College_test <- College[test, ]
```

b)

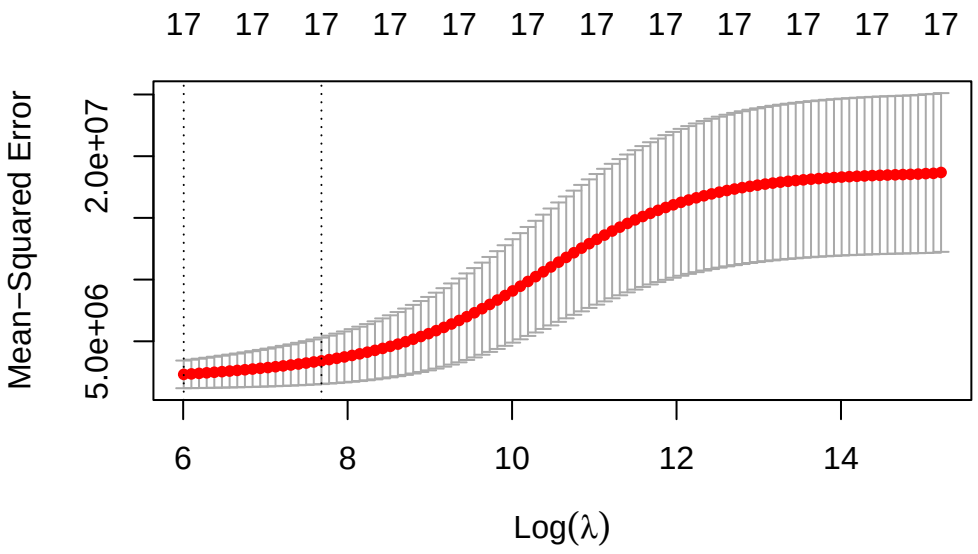
```
apps_lse <- lm(Apps ~ ., College[train, ])
apps_lse_pred <- predict(apps_lse, College[test, ])
mean((apps_lse_pred - y.test)^2)
```

```
[1] 1026096
```

The test MSE we end up with is 1026096.

c)

```
cv.out_ridge <- cv.glmnet(x[train, ], y[train], alpha  = 0)
plot(cv.out_ridge)
```



Finding lambda corresponding to minimum cross validation error.

```
bestlam_ridge <- cv.out_ridge$lambda.min
bestlam_ridge
```

```
[1] 406.2152
```

Calculating test error.

```
apps_ridge <- glmnet(x[train, ], y[train], alpha = 0,
                     lambda = bestlam_ridge)
apps_ridge_pred <- predict(apps_ridge, newx = x[test, ])
mean((apps_ridge_pred - y.test)^2)
```

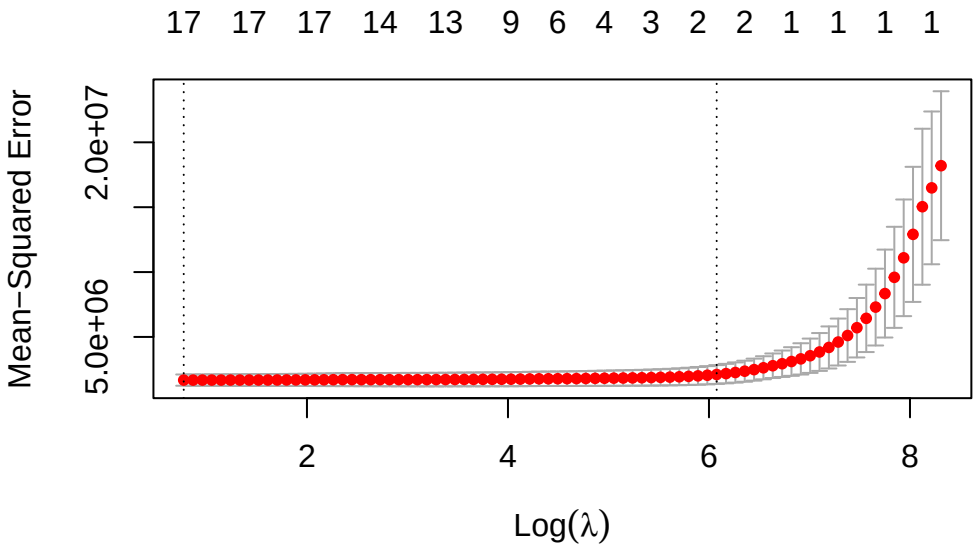
```
[1] 970029.2
```

The test error we end up with is 970029.2, an improvement over least squares.

d)

Finding lambda with lowest cross validation error.

```
cv.out_lasso <- cv.glmnet(x[train, ], y[train], alpha = 1)
plot(cv.out_lasso)
```



Grabbing lambda corresponding to lowest cross validation error.

```
bestlam_lasso <- cv.out_lasso$lambda.min
bestlam_lasso
```

```
[1] 2.167848
```

Fitting lasso and predicting on test set.

```
apps_lasso <- glmnet(x[train, ], y[train], alpha = 1,
                     lambda = bestlam_lasso)
apps_lasso_pred <- predict(apps_lasso, newx = x[test, ])
mean((apps_lasso_pred - y.test)^2)
```

```
[1] 1013435
```

The test error for lasso is 1013435, in-between least squares ridge regression.

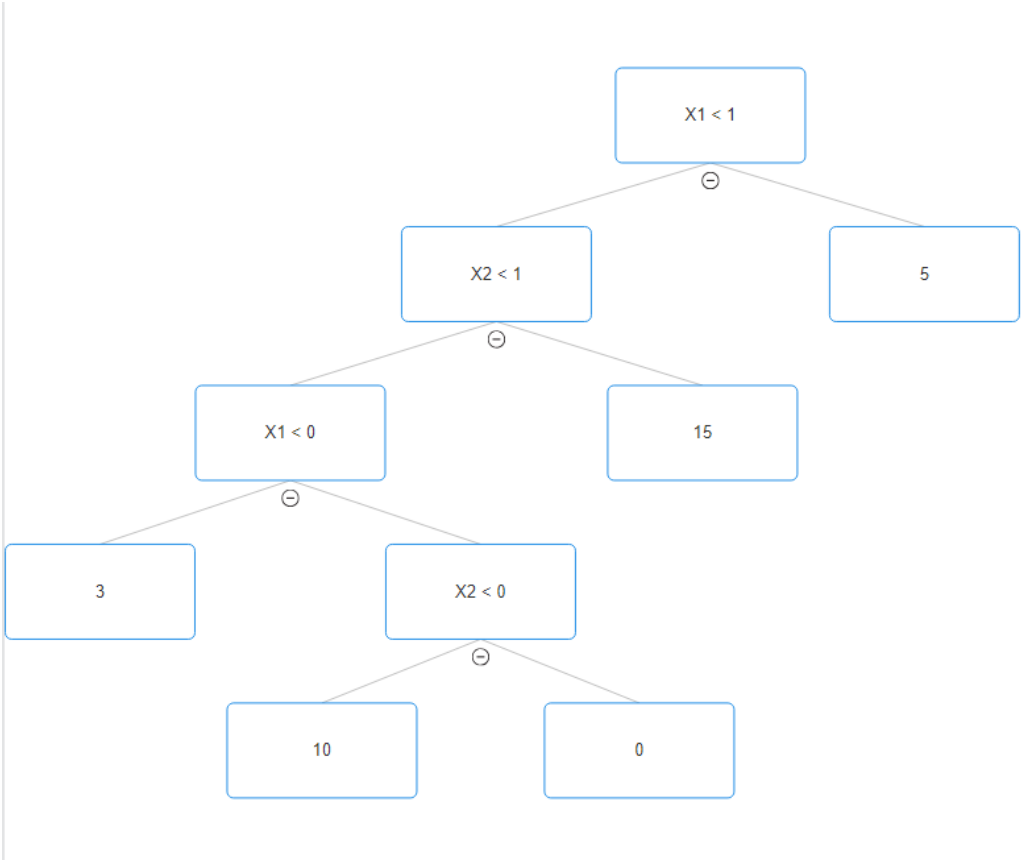
```
apps_lasso_coef <- predict(apps_lasso, type = "coefficients")[1:18,]
apps_lasso_coef
```

(Intercept)	PrivateYes	Accept	Enroll	Top10perc
-5.85082731	-551.94564528	1.73590532	-1.24765293	63.63848772
Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board
-20.95805152	0.08704623	0.01933786	-0.08426781	0.15180916
Books	Personal	PhD	Terminal	S.F.Ratio
-0.09987712	0.15879104	-13.94997700	-1.05751968	3.84655292
perc.alumni	Expend	Grad.Rate		
-0.37110003	0.05516770	8.60115434		

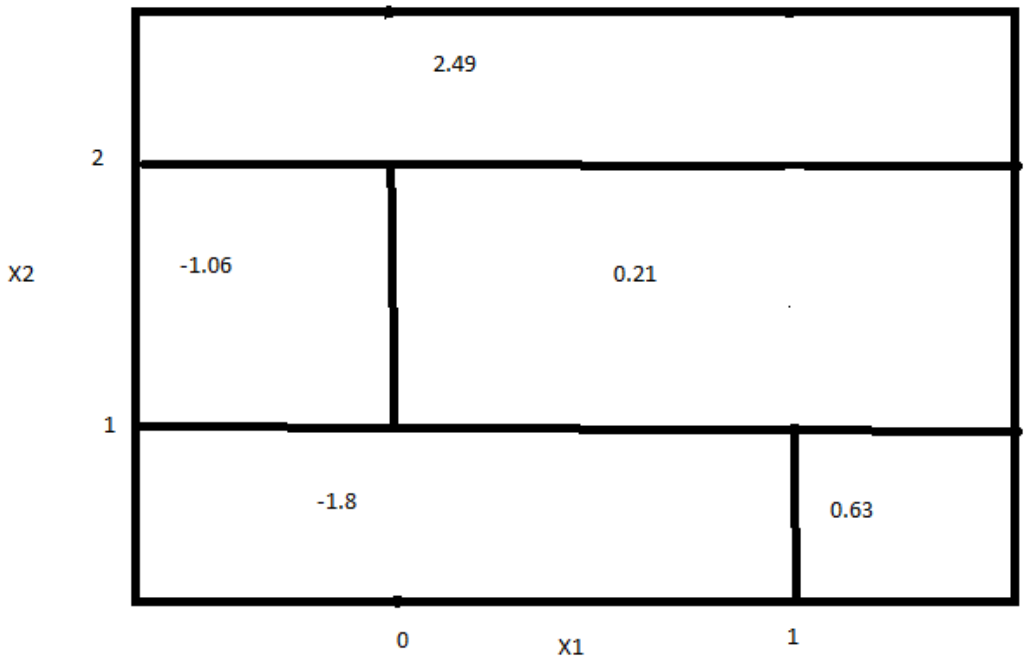
All coefficients are non-zero, this is due to the lambda chosen through cross validation being small.

6)

a)



b)



7)

In the majority vote approach we have 6 votes for Red and 4 votes for Green. We would recommend Red.

In the average probability approach we have the below:

```
probs <- c(0.1,0.15,0.2,0.2,0.55,0.6,0.6,0.65,0.7,0.75)
average_prob <- sum(probs)/length(probs)
average_prob
```

```
[1] 0.45
```

The average probability, $P(\text{Class is Red}|X)$, is 0.45. So we would recommend Green.