Homework 08

STAT 430, Fall 2017

Due: Friday, November 10, 11:59 PM

Exercise 1 (Classifying Leukemia)

[10 points] For this question we will use the data in `leukemia.csv` which originates from Golub et al. 1999.

The response variable `class` is a categorical variable. There are two possible responses: `ALL` (acute myeloid leukemia) and `AML` (acute lymphoblastic leukemia), both types of leukemia. We will use the many feature variables, which are expression levels of genes, to predict these classes.

Note that, this dataset is rather large and you may have difficultly loading it using the "Import Dataset" feature in RStudio. Instead place the file in the same folder as your `.Rmd` file and run the following command. (Which you should be doing anyway.) Again, since this dataset is large, use 5-fold cross-validation when needed.

```
library(readr)
leukemia = read_csv("leukemia.csv", progress = FALSE)
```

For use with the `glmnet` package, it will be useful to create a factor response variable `y` and a feature matrix `X` as seen below. We won't test-train split the data since there are so few observations.

```
y = as.factor(leukemia$class)
X = as.matrix(leukemia[, -1])
```

Do the following:

- Set a seed equal to your UIN.
- Fit the full path of a logistic regression with both a lasso penalty and a ridge penalty. (Don't use cross-validation. Also let `glmnet` choose the λ values.) Create side-by-side plots that shows the features entering (or leaving) the models.
- Use cross-validation to tune an logistic regression with a lasso penalty. Again, let `glmnet` choose the λ values. Store both the λ that minimizes the deviance, as well as the λ that has a deviance within one standard error. Create a plot of the deviances for each value of λ considered. Use these two λ values to create a grid for use with `train()` in `caret`. Use `train()` to get cross-validated classification accuracy for these two values of λ. Store these values.
- Use cross-validation to tune an logistic regression with a ridge penalty. Again, let `glmnet` choose the λ values. Store both the λ that minimizes the deviance, as well as the λ that has a deviance within one standard error. Create a plot of the deviances for each value of λ considered. Use these two λ values to create a grid for use with `train()` in `caret`. Use `train()` to get cross-validated classification accuracy for these two values of λ. Store these values.
- Use cross-validation to tune k-nearest neighbors using `train()` in `caret`. Do not specify a grid of k values to try, let `caret` do so automatically. (It will use 5, 7, 9.) Store the cross-validated accuracy for each. Scale the predictors.
- Summarize these **seven** models in a table. (Two lasso, two ridge, three knn.) For each report the cross-validated accuracy and the standard deviation of the accuracy.

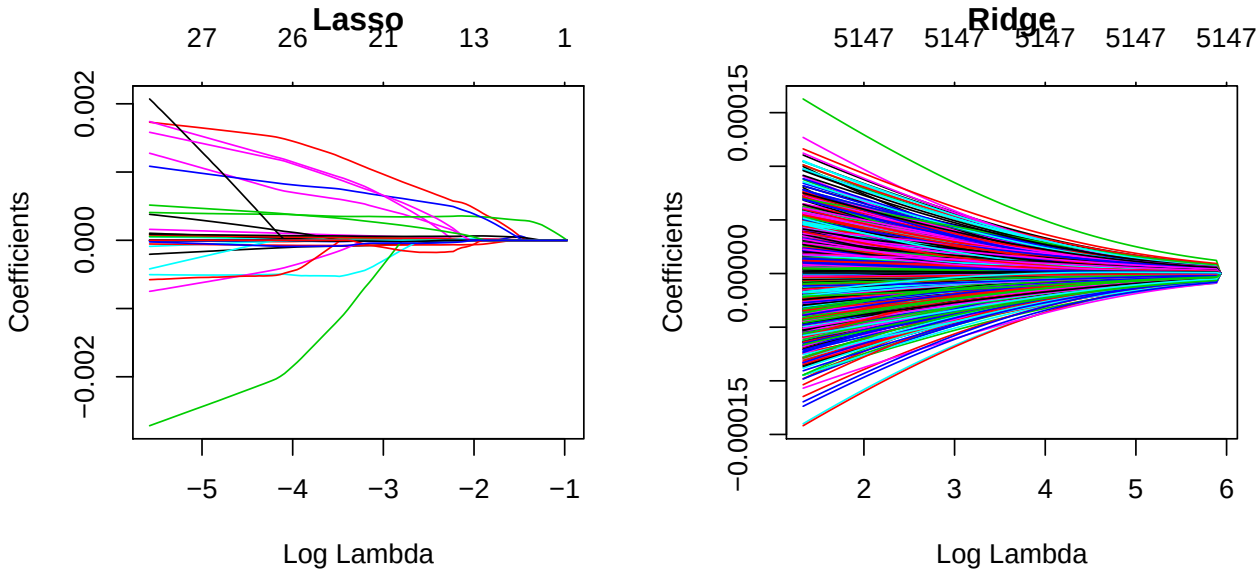
Solution:

```
library(glmnet)
library(caret)
```

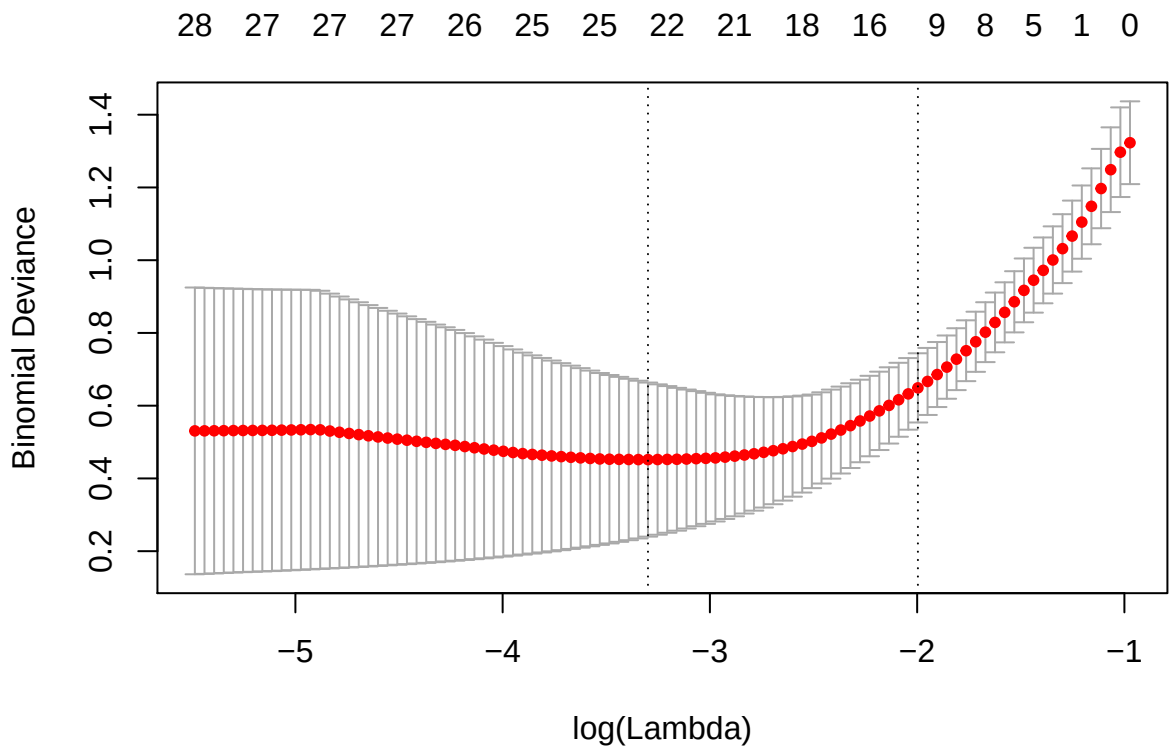
```
uin = 123456789
set.seed(uin)
cv_5 = trainControl(method = "cv", number = 5)
```

```
fit_lasso = glmnet(X, y, family = "binomial", alpha = 1)
fit_ridge = glmnet(X, y, family = "binomial", alpha = 0)
```

```
par(mfrow = c(1, 2))
plot(fit_lasso, xvar = "lambda", main = "Lasso")
plot(fit_ridge, xvar = "lambda", main = "Ridge")
```



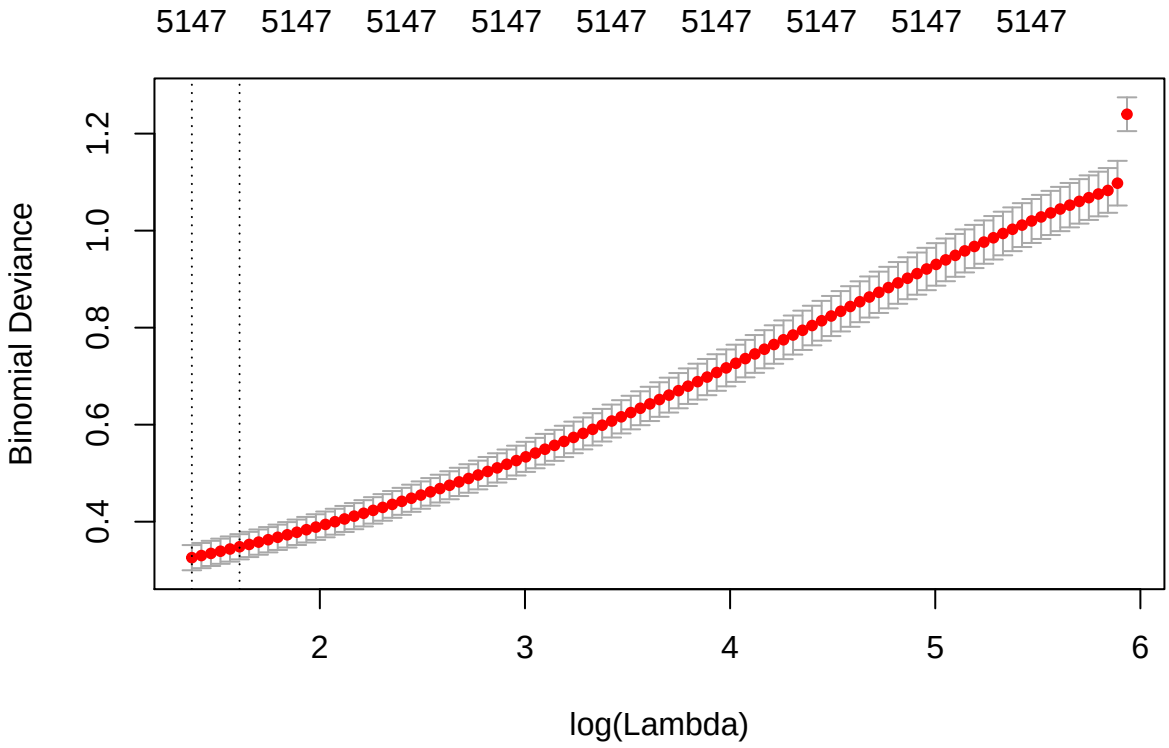

```
fit_lasso_cv = cv.glmnet(X, y, family = "binomial", alpha = 1, nfolds = 5)
plot(fit_lasso_cv)
```




```
lambda_lasso = expand.grid(alpha = 1,
                           lambda = c(fit_lasso_cv$lambda.min,
                                      fit_lasso_cv$lambda.1se))
```

```
train_lasso = train(X, y,
                    method = "glmnet",
                    trControl = cv_5,
                    tuneGrid = lambda_lasso)
```

```
fit_ridge_cv = cv.glmnet(X, y, family = "binomial", alpha = 0, nfolds = 5)
plot(fit_ridge_cv)
```




```
lambda_ridge = expand.grid(alpha = 0,
                           lambda = c(fit_ridge_cv$lambda.min,
                                      fit_ridge_cv$lambda.1se))
```

```
train_ridge = train(X, y,
                    method = "glmnet",
                    trControl = cv_5,
                    tuneGrid = lambda_ridge)
```

```
train_knn = train(X, y,
                  method = "knn",
                  preProc = c("center", "scale"),
                  trControl = cv_5)
```

Method	Parameter	Parameter Value	CV-5 Accuracy	Standard Deviation
Lasso	lambda	0.037	0.930	0.003
Lasso	lambda	0.136	0.917	0.058
Ridge	lambda	3.960	0.986	0.032
Ridge	lambda	4.996	0.986	0.032
KNN	k	5.000	0.876	0.090
KNN	k	7.000	0.876	0.090
KNN	k	9.000	0.833	0.063

Exercise 2 (The Cost of College)

[10 points] For this exercise, we will use the `College` data from the `ISLR` package. Familiarize yourself with this dataset before performing analyses. We will attempt to predict the `Outstate` variable.

Test-train split the data using this code.

```
set.seed(42)
library(caret)
library(ISLR)
index = createDataPartition(College$Outstate, p = 0.75, list = FALSE)
college_trn = College[index, ]
college_tst = College[-index, ]
```

Train a total of **six** models using five-fold cross validation.

- An additive linear model.
- An elastic net model using additive predictors. Use a `tuneLength` of `10`.
- An elastic net model that also considers all two-way interactions. Use a `tuneLength` of `10`.
- A well-tuned KNN model.
- A well-tuned KNN model that also considers all two-way interactions. (Should this work?)
- A default-tuned random forest.

Before beginning, set a seed equal to your UIN.

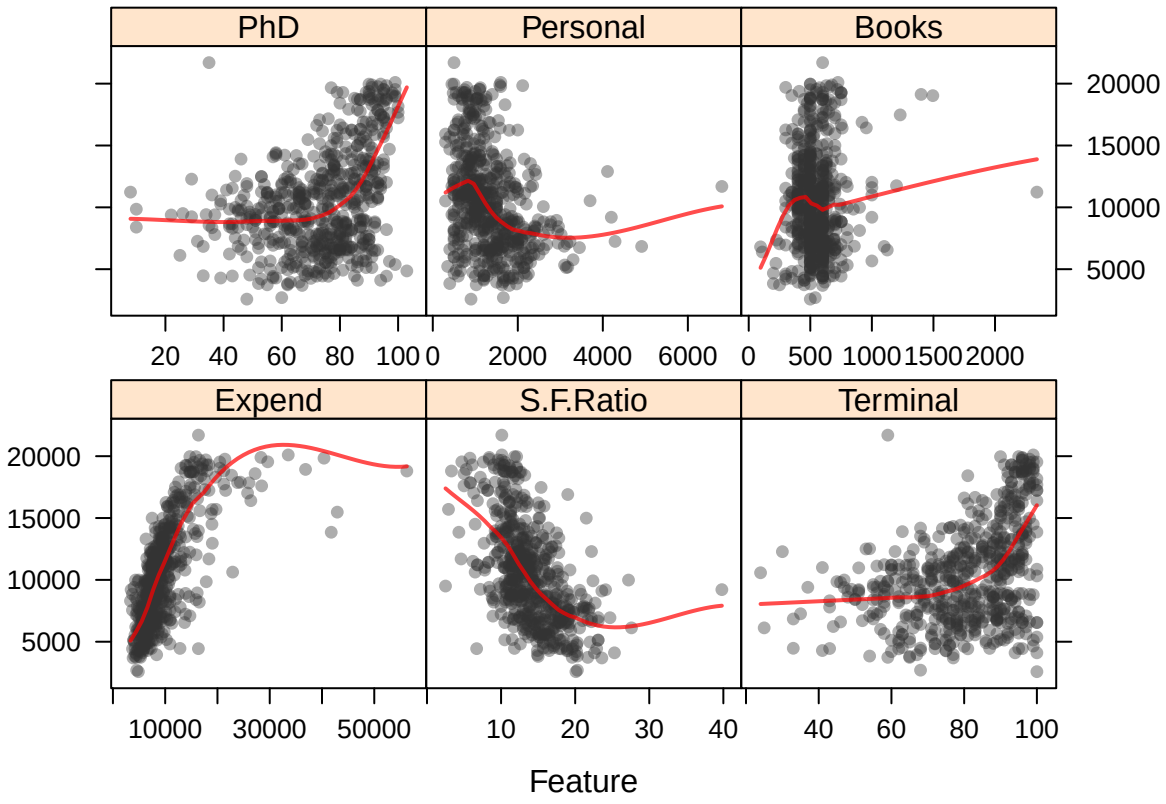
```
uin = 123456789
set.seed(uin)
```

- Create a table which reports CV and Test RMSE for each.

Solution:

Note that some code, for plotting and summarizing, is hidden. See the .Rmd file for code.

```
library(glmnet)
library(randomForest)
```




```
calc_rmse = function(actual, predicted) {
  sqrt(mean((actual - predicted) ^ 2))
}
```

```
cv_5 = trainControl(method = "cv", number = 5)
```

```
set.seed(uin)
fit_lm         = train(Outstate ~ ., data = college_trn, method = "lm",
                       trControl = cv_5)
fit_glmnet     = train(Outstate ~ ., data = college_trn, method = "glmnet",
                       trControl = cv_5, tuneLength = 10)
fit_glmnet_int = train(Outstate ~ . ^ 2, data = college_trn, method = "glmnet",
                       trControl = cv_5, tuneLength = 10)
fit_knn        = train(Outstate ~ ., data = college_trn, method = "knn",
                       trControl = cv_5, tuneLength = 25,
                       preProcess = c("center", "scale"))
fit_knn_int    = train(Outstate ~ . ^ 2, data = college_trn, method = "knn",
                       trControl = cv_5, tuneLength = 25,
                       preProcess = c("center", "scale"))
fit_rf         = train(Outstate ~ ., data = college_trn, method = "rf",
                       trControl = cv_5)
```

```
get_best_result = function(caret_fit) {
  best = which(rownames(caret_fit$results) == rownames(caret_fit$bestTune))
  best_result = caret_fit$results[best, ]
  rownames(best_result) = NULL
  best_result
}
```

Method	CV RMSE	Test RMSE
Linear Model	1968.034	2069.087
Elastic Net	1982.386	2080.690
Elastic Net with Interactions	1872.169	1964.739
KNN	1916.154	1971.330
KNN with Interactions	1916.701	2060.803
Random Forest	1785.113	1733.767

Exercise 3 (Concept Checks)

[**1 point each**] Answer the following questions based on your results from the three exercises.

Leukemia

(a) How many observations are in the dataset? How many predictors are in the dataset?

- Predictors: 5147
- Observations: 72

(b) Based on the deviance plots, do you feel that `glmnet` considered enough λ values for lasso?

Yes. We see a nice U-shaped CV error curve.

(c) Based on the deviance plots, do you feel that `glmnet` considered enough λ values for ridge?

No. This plot suggests that if we were to try smaller lambda, we could achieve a lower devience (error).

(d) How does k-nearest neighbor compare to the penalized methods? Can you explain any difference?

KNN performs worse. This is expected in a high-dimensional setting that we have here due to the curse of dimensionality. (Although, it turns out that if we were to scale the predictors, KNN would actually work reasonably well.)

(e) Based on your results, which model would you choose? Explain.

The **ridge** model with the **larger** lambda. (Using the 1-SE rule.) This model performs much better than the lasso and KNN models. It also achieves the same CV accuracy as the ridge with a smaller lambda, but we prefer the model with a greater penalty, thus less chance of overfitting.

College

(f) Based on the table, which model do you prefer? Justify your answer.

Random forest, as it achieves the lowest error. Although, predicting college tuition to within 1700 versus 2000 dollars is not all that different, so an argument could be made for a more interpretable model.

(g) For both of the elastic net models, report the best tuning parameters from `caret`. For each, is this ridge, lasso, or somewhere in between? If in between, closer to which?

```
fit_glmnet$bestTune
```

```
##   alpha   lambda
## 5   0.1 35.91052
```

```
fit_glmnet_int$bestTune
```

```
##   alpha   lambda
## 7   0.1 220.8521
```

Both have an α value of 0.1, so are in-between, but are closer to ridge.

(h) Did you scale the predictors when you used KNN? Should you have scaled the predictors when you used KNN?

Yes. Yes. A lower error is found using scaled predictors.

(i) Of the two KNN models which works better? Can you explain why?

Without interactions seems to work better. Adding all the interactions creates a high dimensional dataset, so we're suffering from the curse of dimensionality.

(j) What year is this dataset from? What was out-of-state tuition at UIUC at that time?

According to the documentation, 1995.

```
library(dplyr)
College %>% filter(rownames(College) == "University of Illinois - Urbana") %>% select(Outstate)
```

```
##   Outstate
## 1     7560
```

Wow. Remember, this is for out-of-state.