Predicting Stock Returns with Cluster-Then-Predict

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.4.2

## -- Attaching packages ---------------------------------------------------------------------------------

## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.0     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0

## Warning: package 'ggplot2' was built under R version 3.4.4

## Warning: package 'tibble' was built under R version 3.4.4

## Warning: package 'tidyr' was built under R version 3.4.3

## Warning: package 'purrr' was built under R version 3.4.4

## Warning: package 'dplyr' was built under R version 3.4.4

## Warning: package 'forcats' was built under R version 3.4.3

## -- Conflicts ------------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.4.3
```

```r
library(caTools)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4

## Loading required package: lattice

## Warning in as.POSIXlt.POSIXct(Sys.time()): unknown timezone 'zone/tz/2018i.
## 1.0/zoneinfo/America/Chicago'

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(flexclust)
```

```
## Warning: package 'flexclust' was built under R version 3.4.4

## Loading required package: grid

## Loading required package: modeltools

## Loading required package: stats4
```

In the second lecture sequence this week, we heard about cluster-then-predict, a methodology in which you first cluster observations and then build cluster-specific prediction models. In the lecture sequence, we saw how this methodology helped improve the prediction of heart attack risk. In this assignment, we'll use cluster-then-predict to predict future stock prices using historical stock data.

When selecting which stocks to invest in, investors seek to obtain good future returns. In this problem, we will first use clustering to identify clusters of stocks that have similar returns over time. Then, we'll use logistic regression to predict whether or not the stocks will have positive future returns.

For this problem, we'll use StocksCluster.csv, which contains monthly stock returns from the NASDAQ stock exchange. The NASDAQ is the second-largest stock exchange in the world, and it lists many technology companies. The stock price data used in this problem was obtained from infochimps, a website providing access to many datasets.

Each observation in the dataset is the monthly returns of a particular company in a particular year. The years included are 2000-2009. The companies are limited to tickers that were listed on the exchange for the entire period 2000-2009, and whose stock price never fell below $1. So, for example, one observation is for Yahoo in 2000, and another observation is for Yahoo in 2001. Our goal will be to predict whether or not the stock return in December will be positive, using the stock returns for the first 11 months of the year.

This dataset contains the following variables:

ReturnJan = the return for the company's stock during January (in the year of the observation).

ReturnFeb = the return for the company's stock during February (in the year of the observation).

ReturnMar = the return for the company's stock during March (in the year of the observation).

ReturnApr = the return for the company's stock during April (in the year of the observation).

ReturnMay = the return for the company's stock during May (in the year of the observation).

ReturnJune = the return for the company's stock during June (in the year of the observation).

ReturnJuly = the return for the company's stock during July (in the year of the observation).

ReturnAug = the return for the company's stock during August (in the year of the observation).

ReturnSep = the return for the company's stock during September (in the year of the observation).

ReturnOct = the return for the company's stock during October (in the year of the observation).

ReturnNov = the return for the company's stock during November (in the year of the observation).

PositiveDec = whether or not the company's stock had a positive return in December (in the year of the observation). This variable takes value 1 if the return was positive, and value 0 if the return was not positive.

For the first 11 variables, the value stored is a proportional change in stock value during that month. For instance, a value of 0.05 means the stock increased in value 5% during the month, while a value of -0.02 means the stock decreased in value 2% during the month.

1.1) Exploring the Dataset

Load StocksCluster.csv into a data frame called "stocks". How many observations are in the dataset?

```
stocks = read.csv('StocksCluster.csv')

glimpse(stocks)
```

```
## Observations: 11,580
## Variables: 12
## $ ReturnJan   <dbl> 0.080677966, -0.010679889, 0.047741935, -0.0740402...
## $ ReturnFeb   <dbl> 0.06625000, 0.10211539, 0.03598972, -0.04816956, -...
## $ ReturnMar   <dbl> 0.03294118, 0.14549595, 0.03970223, 0.01821862, 0....
```

```
## $ ReturnApr   <dbl> 0.18309859, -0.08442804, -0.16235294, -0.02467917,...
## $ ReturnMay   <dbl> 0.130333952, -0.327300392, -0.147426982, -0.006036...
## $ ReturnJune  <dbl> -0.017642342, -0.359266055, 0.048589342, -0.025303...
## $ ReturnJuly  <dbl> -0.020517029, -0.025321312, -0.135384615, -0.09400...
## $ ReturnAug   <dbl> 0.02467587, 0.21129000, 0.03339192, 0.09529025, 0....
## $ ReturnSep   <dbl> -0.02040816, -0.58000326, 0.00000000, 0.05668016, ...
## $ ReturnOct   <dbl> -0.17331768, -0.26714125, 0.09169550, -0.09633911,...
## $ ReturnNov   <dbl> -0.02538531, -0.15123457, -0.05956113, -0.04051173...
## $ PositiveDec <int> 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,...
```

1.2) Exploring the Dataset

What proportion of the observations have positive returns in December?

```r
table(stocks$PositiveDec)
```

```
##
##    0    1
## 5256 6324
```

```r
stocks %>%
  select(PositiveDec) %>%
  summarize(prop = (sum(PositiveDec == 1)) / nrow(stocks))
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

```
##       prop
## 1 0.546114
```

1.3) Exploring the Dataset

What is the maximum correlation between any two return variables in the dataset? You should look at the pairwise correlations between ReturnJan, ReturnFeb, ReturnMar, ReturnApr, ReturnMay, ReturnJune, ReturnJuly, ReturnAug, ReturnSep, ReturnOct, and ReturnNov.

```r
sort(cor(stocks), decreasing = TRUE)
```

```
##   [1]  1.0000000000  1.0000000000  1.0000000000  1.0000000000  1.0000000000
##   [6]  1.0000000000  1.0000000000  1.0000000000  1.0000000000  1.0000000000
##  [11]  1.0000000000  1.0000000000  0.1916727856  0.1916727856  0.1699944833
##  [16]  0.1699944833  0.1429772286  0.1429772286  0.1315597863  0.1315597863
##  [21]  0.0943535281  0.0943535281  0.0922383068  0.0922383068  0.0908502642
##  [26]  0.0908502642  0.0806319317  0.0806319317  0.0765183267  0.0765183267
##  [31]  0.0743642097  0.0743642097  0.0689478037  0.0689478037  0.0676323333
##  [36]  0.0676323333  0.0667745831  0.0667745831  0.0638225039  0.0638225039
##  [41]  0.0582019336  0.0582019336  0.0485400254  0.0485400254  0.0480465902
##  [46]  0.0480465902  0.0447472692  0.0447472692  0.0435017706  0.0435017706
##  [51]  0.0416302863  0.0416302863  0.0373235349  0.0373235349  0.0317618366
##  [56]  0.0317618366  0.0234097451  0.0234097451  0.0224086612  0.0224086612
##  [61]  0.0219628623  0.0219628623  0.0171667279  0.0171667279  0.0107105260
##  [66]  0.0107105260  0.0097262876  0.0097262876  0.0047285182  0.0047285182
##  [71]  0.0041669657  0.0041669657  0.0033741597  0.0033741597  0.0007407139
##  [76]  0.0007407139  0.0007137558  0.0007137558 -0.0038927894 -0.0038927894
##  [81] -0.0110277522 -0.0110277522 -0.0119237577 -0.0119237577 -0.0197197998
##  [86] -0.0197197998 -0.0210745388 -0.0210745388 -0.0220053995 -0.0220053995
##  [91] -0.0226359936 -0.0226359936 -0.0227920187 -0.0227920187 -0.0264371526
```

```
##  [96] -0.0264371526 -0.0289209718 -0.0289209718 -0.0291525996 -0.0291525996
## [101] -0.0331256580 -0.0331256580 -0.0376780060 -0.0376780060 -0.0381731839
## [106] -0.0381731839 -0.0444114168 -0.0444114168 -0.0483738369 -0.0483738369
## [111] -0.0517560510 -0.0517560510 -0.0525749563 -0.0525749563 -0.0547089088
## [116] -0.0547089088 -0.0580792362 -0.0580792362 -0.0617785094 -0.0617785094
## [121] -0.0623465559 -0.0623465559 -0.0652705413 -0.0652705413 -0.0755945614
## [126] -0.0755945614 -0.0814297650 -0.0814297650 -0.0859054862 -0.0859054862
## [131] -0.0873242672 -0.0873242672 -0.0904967978 -0.0904967978 -0.0955209197
## [136] -0.0955209197 -0.1164890345 -0.1164890345 -0.1546582815 -0.1546582815
## [141] -0.1559832630 -0.1559832630 -0.1913519239 -0.1913519239
```

1.4) Exploring the Dataset

Which month (from January through November) has the largest mean return across all observations in the dataset?

Which month (from January through November) has the smallest mean return across all observations in the dataset?

```r
sort(colMeans(stocks), decreasing = TRUE)
```

```
##  PositiveDec   ReturnApr   ReturnMay   ReturnMar   ReturnAug
##  0.546113990 0.026308147 0.024736591 0.019402336 0.016198265
##    ReturnJan   ReturnNov  ReturnJune   ReturnOct  ReturnJuly
##  0.012631602 0.011387440 0.005937902 0.005650844 0.003050863
##    ReturnFeb   ReturnSep
## -0.007604784 -0.014720768
```

Explanation

These can be determined using the summary function:

```r
summary(stocks)
```

```
##    ReturnJan           ReturnFeb           ReturnMar
##  Min.   :-0.7616205   Min.   :-0.690000   Min.   :-0.712994
##  1st Qu.:-0.0691663   1st Qu.:-0.077748   1st Qu.:-0.046389
##  Median : 0.0009965   Median :-0.010626   Median : 0.009878
##  Mean   : 0.0126316   Mean   :-0.007605   Mean   : 0.019402
##  3rd Qu.: 0.0732606   3rd Qu.: 0.043600   3rd Qu.: 0.077066
##  Max.   : 3.0683060   Max.   : 6.943694   Max.   : 4.008621
##    ReturnApr          ReturnMay          ReturnJune
##  Min.   :-0.826503   Min.   :-0.92207   Min.   :-0.717920
##  1st Qu.:-0.054468   1st Qu.:-0.04640   1st Qu.:-0.063966
##  Median : 0.009059   Median : 0.01293   Median :-0.000880
##  Mean   : 0.026308   Mean   : 0.02474   Mean   : 0.005938
##  3rd Qu.: 0.085338   3rd Qu.: 0.08396   3rd Qu.: 0.061566
##  Max.   : 2.528827   Max.   : 6.93013   Max.   : 4.339713
##    ReturnJuly          ReturnAug           ReturnSep
##  Min.   :-0.7613096   Min.   :-0.726800   Min.   :-0.839730
##  1st Qu.:-0.0731917   1st Qu.:-0.046272   1st Qu.:-0.074648
##  Median :-0.0008047   Median : 0.007205   Median :-0.007616
##  Mean   : 0.0030509   Mean   : 0.016198   Mean   :-0.014721
##  3rd Qu.: 0.0718205   3rd Qu.: 0.070783   3rd Qu.: 0.049476
##  Max.   : 2.5500000   Max.   : 3.626609   Max.   : 5.863980
##    ReturnOct          ReturnNov          PositiveDec
##  Min.   :-0.685504   Min.   :-0.747171   Min.   :0.0000
```

```
##  1st Qu.:-0.070915   1st Qu.:-0.054890   1st Qu.:0.0000
##  Median : 0.002115   Median : 0.008522   Median :1.0000
##  Mean   : 0.005651   Mean   : 0.011387   Mean   :0.5461
##  3rd Qu.: 0.074542   3rd Qu.: 0.076576   3rd Qu.:1.0000
##  Max.   : 5.665138   Max.   : 3.271676   Max.   :1.0000
```

If you look at the mean value for each variable, you can see that April has the largest mean value (0.026308), and September has the smallest mean value (-0.014721).

2.1) Initial Logistic Regression Model

Run the following commands to split the data into a training set and testing set, putting 70% of the data in the training set and 30% of the data in the testing set:

```r
set.seed(144)

spl = sample.split(stocks$PositiveDec, SplitRatio = 0.7)

stocks_train = subset(stocks, spl == TRUE)

stocks_test = subset(stocks, spl == FALSE)
```

Then, use the stocksTrain data frame to train a logistic regression model (name it StocksModel) to predict PositiveDec using all the other variables as independent variables. Don't forget to add the argument family=binomial to your glm command.

```r
stocks_model = glm(PositiveDec ~ .,
                   data = stocks_train,
                   family = binomial)
```

What is the overall accuracy on the training set, using a threshold of 0.5?

```r
stocks_pred_train = predict(stocks_model, type = 'response', newdata = stocks_train)

table(stocks_train$PositiveDec, stocks_pred_train > 0.5)
```

```
##
##     FALSE TRUE
##   0   990 2689
##   1   787 3640
```

```r
(3640 + 990) / nrow(stocks_train)
```

```
## [1] 0.5711818
```

Explanation

We can train the model with:

StocksModel = glm(PositiveDec ~ ., data=stocksTrain, family=binomial)

Then, we can compute our predictions on the training set with:

PredictTrain = predict(StocksModel, type="response")

And construct a classification matrix with the table function:

table(stocksTrain$PositiveDec, PredictTrain > 0.5)

The overall accuracy of the model is (990 + 3640)/(990 + 2689 + 787 + 3640) = 0.571.

2.2) Initial Logistic Regression Model

Now obtain test set predictions from StocksModel. What is the overall accuracy of the model on the test, again using a threshold of 0.5?

```
stocks_pred_test = predict(stocks_model,
                           type = 'response',
                           newdata = stocks_test)

table(stocks_test$PositiveDec, stocks_pred_test >= 0.5)
```

```
##
##     FALSE TRUE
##   0   417 1160
##   1   344 1553
```

```
(417 + 1553) / nrow(stocks_test)
```

```
## [1] 0.5670697
```

Explanation

You can compute predictions on the test set using the predict function:

PredictTest = predict(StocksModel, newdata=stocksTest, type="response")

Then, you can compute the classification matrix on the test set with the table function:

table(stocksTest$PositiveDec, PredictTest > 0.5)

The overall accuracy of the model on the test set is $(417 + 1553)/(417 + 1160 + 344 + 1553) = 0.567$

2.3) Initial Logistic Regression Model

What is the accuracy on the test set of a baseline model that always predicts the most common outcome (PositiveDec = 1)?

```
table(stocks_test$PositiveDec)
```

```
##
##    0    1
## 1577 1897
```

```
1897 / (1897 + 1577)
```

```
## [1] 0.5460564
```

Explanation

This can be computed by making a table of the outcome variable in the test set:

table(stocksTest$PositiveDec)

The baseline model would get all of the PositiveDec = 1 cases correct, and all of the PositiveDec = 0 cases wrong, for an accuracy of $1897/(1577 + 1897) = 0.5460564$.

3.1) Clustering Stocks

Now, let's cluster the stocks. The first step in this process is to remove the dependent variable using the following commands:

```r
limited_train = stocks_train

limited_train$PositiveDec = NULL

limited_test = stocks_test

limited_test$PositiveDec = NULL
```

Why do we need to remove the dependent variable in the clustering phase of the cluster-then-predict methodology?

```r
include_graphics('3.1.png')
```

- ○ Leaving in the dependent variable might

- ○ Removing the dependent variable decrea

- ◉ Needing to know the dependent variable
the methodology ✔

Explanation

In cluster-then-predict, our final goal is to pre
prediction. Therefore, if we need to know the
longer useful for prediction of an unknown ou
This is an important point that is sometimes m
might conclude your method strongly outper
the outcome to determine the clusters, which

3.2) Clustering Stocks

In the market segmentation assignment in this week's homework, you were introduced to the preProcess command from the caret package, which normalizes variables by subtracting by the mean and dividing by the standard deviation.

In cases where we have a training and testing set, we'll want to normalize by the mean and standard deviation of the variables in the training set. We can do this by passing just the training set to the preProcess function:

```
preproc = preProcess(limited_train)

norm_train = predict(preproc, limited_train)

norm_test = predict(preproc, limited_test)
```

What is the mean of the ReturnJan variable in normTrain?

What is the mean of the ReturnJan variable in normTest?

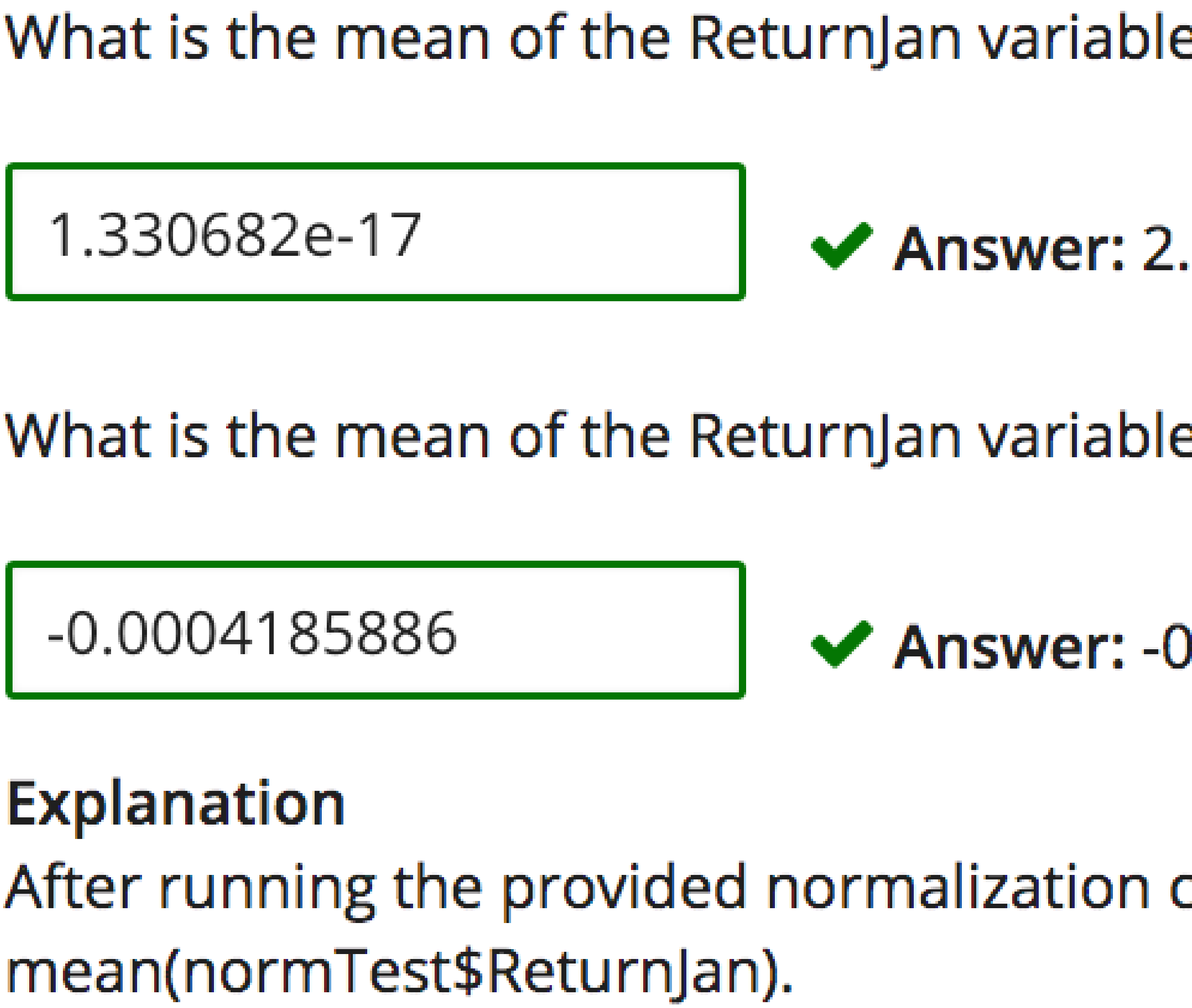
```
colMeans(norm_train)
```

```
##     ReturnJan     ReturnFeb     ReturnMar     ReturnApr     ReturnMay
##  1.330682e-17 -1.008584e-17 -8.424944e-18 -1.460048e-19 -9.386254e-19
##    ReturnJune    ReturnJuly     ReturnAug     ReturnSep     ReturnOct
## -7.332770e-18  3.542209e-18  2.075997e-17 -6.795511e-18 -5.161583e-18
##     ReturnNov
## -6.470330e-18
```

```
colMeans(norm_test)
```

```
##     ReturnJan     ReturnFeb     ReturnMar     ReturnApr     ReturnMay
## -0.0004185886 -0.0038621679  0.0058299150 -0.0363806373  0.0265120925
##    ReturnJune    ReturnJuly     ReturnAug     ReturnSep     ReturnOct
##  0.0431544402  0.0060164183 -0.0497332436  0.0293887872  0.0296723768
##     ReturnNov
##  0.0171281833
```

```
include_graphics('3.2.png')
```



What is the mean of the ReturnJan variable

1.330682e-17

Answer: 2.

What is the mean of the ReturnJan variable

-0.0004185886

Answer: -0

Explanation

After running the provided normalization c
mean(normTest$ReturnJan).

3.3) Clustering Stocks

Why is the mean ReturnJan variable much closer to 0 in normTrain than in normTest?

```
include_graphics('3.3.png')
```

- ○ Small rounding errors exist in the norma
- ◉ The distribution of the ReturnJan variabl
- ○ The distribution of the dependent variab

Explanation

From mean(stocksTrain$ReturnJan) and mea slightly higher in the training set than in the t ReturnJan value from the training set, this ex normTest.

3.4) Clustering Stocks

Set the random seed to 144 (it is important to do this again, even though we did it earlier). Run k-means clustering with 3 clusters on normTrain, storing the result in an object called km.

Which cluster has the largest number of observations?

```
set.seed(144)

km = kmeans(norm_train, centers = 3)
```

```r
table(km$cluster)
```

```
##
##    1    2    3
## 3157 4696  253
```

Cluster 2

3.5) Clustering Stocks

Recall from the recitation that we can use the flexclust package to obtain training set and testing set cluster assignments for our observations (note that the call to as.kcca may take a while to complete):

```r
km.kcca = as.kcca(km, norm_train)

cluster_train = predict(km.kcca)

cluster_test = predict(km.kcca, newdata = norm_test)
```

How many test-set observations were assigned to Cluster 2?

```r
table(cluster_test)
```

```
## cluster_test
##    1    2    3
## 1298 2080   96
```

4.1) Cluster-Specific Predictions

Using the subset function, build data frames stocksTrain1, stocksTrain2, and stocksTrain3, containing the elements in the stocksTrain data frame assigned to clusters 1, 2, and 3, respectively (be careful to take subsets of stocksTrain, not of normTrain). Similarly build stocksTest1, stocksTest2, and stocksTest3 from the stocksTest data frame.

```r
stocks_train1 = subset(stocks_train, cluster_train == 1)
stocks_train2 = subset(stocks_train, cluster_train == 2)
stocks_train3 = subset(stocks_train, cluster_train == 3)
```

Which training set data frame has the highest average value of the dependent variable?

```r
mean(stocks_train1$PositiveDec)
```

```
## [1] 0.6024707
```

```r
mean(stocks_train2$PositiveDec)
```

```
## [1] 0.5140545
```

```r
mean(stocks_train3$PositiveDec)
```

```
## [1] 0.4387352
```

```r
stocks_test1 = subset(stocks_test, cluster_test == 1)
stocks_test2 = subset(stocks_test, cluster_test == 2)
stocks_test3 = subset(stocks_test, cluster_test == 3)
```

```r
mean(stocks_test1$PositiveDec)
```

```
## [1] 0.6140216
```

```r
mean(stocks_test2$PositiveDec)
```

```
## [1] 0.5125
```

```r
mean(stocks_test3$PositiveDec)
```

```
## [1] 0.3541667
```

4.2) Cluster-Specific Predictions

Build logistic regression models StocksModel1, StocksModel2, and StocksModel3, which predict PositiveDec using all the other variables as independent variables. StocksModel1 should be trained on stocksTrain1, StocksModel2 should be trained on stocksTrain2, and StocksModel3 should be trained on stocksTrain3.

```r
stocks_model1 = glm(PositiveDec ~ .,
                    data = stocks_train1,
                    family = binomial)

stocks_model2 = glm(PositiveDec ~ .,
                    data = stocks_train2,
                    family = binomial)

stocks_model3 = glm(PositiveDec ~ .,
                    data = stocks_train3,
                    family = binomial)
```

Which variables have a positive sign for the coefficient in at least one of StocksModel1, StocksModel2, and StocksModel3 and a negative sign for the coefficient in at least one of StocksModel1, StocksModel2, and StocksModel3? Select all that apply.

```r
summary(stocks_model1)
```

```
##
## Call:
## glm(formula = PositiveDec ~ ., family = binomial, data = stocks_train1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7307  -1.2910   0.8878   1.0280   1.5023
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.17224    0.06302   2.733  0.00628 **
## ReturnJan    0.02498    0.29306   0.085  0.93206
## ReturnFeb   -0.37207    0.29123  -1.278  0.20139
## ReturnMar    0.59555    0.23325   2.553  0.01067 *
## ReturnApr    1.19048    0.22439   5.305 1.12e-07 ***
## ReturnMay    0.30421    0.22845   1.332  0.18298
## ReturnJune  -0.01165    0.29993  -0.039  0.96901
## ReturnJuly   0.19769    0.27790   0.711  0.47685
## ReturnAug    0.51273    0.30858   1.662  0.09660 .
## ReturnSep    0.58833    0.28133   2.091  0.03651 *
## ReturnOct   -1.02254    0.26007  -3.932 8.43e-05 ***
## ReturnNov   -0.74847    0.28280  -2.647  0.00813 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4243.0  on 3156  degrees of freedom
## Residual deviance: 4172.9  on 3145  degrees of freedom
## AIC: 4196.9
##
## Number of Fisher Scoring iterations: 4
```

```r
summary(stocks_model2)
```

```
##
## Call:
## glm(formula = PositiveDec ~ ., family = binomial, data = stocks_train2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2012  -1.1941   0.8583   1.1334   1.9424
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.10293    0.03785   2.719 0.006540 **
## ReturnJan    0.88451    0.20276   4.362 1.29e-05 ***
## ReturnFeb    0.31762    0.26624   1.193 0.232878
## ReturnMar   -0.37978    0.24045  -1.579 0.114231
## ReturnApr    0.49291    0.22460   2.195 0.028189 *
## ReturnMay    0.89655    0.25492   3.517 0.000436 ***
## ReturnJune   1.50088    0.26014   5.770 7.95e-09 ***
## ReturnJuly   0.78315    0.26864   2.915 0.003554 **
## ReturnAug   -0.24486    0.27080  -0.904 0.365876
## ReturnSep    0.73685    0.24820   2.969 0.002989 **
## ReturnOct   -0.27756    0.18400  -1.509 0.131419
## ReturnNov   -0.78747    0.22458  -3.506 0.000454 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6506.3  on 4695  degrees of freedom
## Residual deviance: 6362.2  on 4684  degrees of freedom
## AIC: 6386.2
##
## Number of Fisher Scoring iterations: 4
```

```r
summary(stocks_model3)
```

```
##
## Call:
## glm(formula = PositiveDec ~ ., family = binomial, data = stocks_train3)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9146  -1.0393  -0.7689   1.1921   1.6939
##
## Coefficients:
```

```
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.181896   0.325182  -0.559   0.5759
## ReturnJan   -0.009789   0.448943  -0.022   0.9826
## ReturnFeb   -0.046883   0.213432  -0.220   0.8261
## ReturnMar    0.674179   0.564790   1.194   0.2326
## ReturnApr    1.281466   0.602672   2.126   0.0335 *
## ReturnMay    0.762512   0.647783   1.177   0.2392
## ReturnJune   0.329434   0.408038   0.807   0.4195
## ReturnJuly   0.774164   0.729360   1.061   0.2885
## ReturnAug    0.982605   0.533158   1.843   0.0653 .
## ReturnSep    0.363807   0.627774   0.580   0.5622
## ReturnOct    0.782242   0.733123   1.067   0.2860
## ReturnNov   -0.873752   0.738480  -1.183   0.2367
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 346.92  on 252  degrees of freedom
## Residual deviance: 328.29  on 241  degrees of freedom
## AIC: 352.29
##
## Number of Fisher Scoring iterations: 4
```

Explanation

We can build the models with:

StocksModel1 = glm(PositiveDec ~ ., data=stocksTrain1, family=binomial)

StocksModel2 = glm(PositiveDec ~ ., data=stocksTrain2, family=binomial)

StocksModel3 = glm(PositiveDec ~ ., data=stocksTrain3, family=binomial)

From summary(StocksModel1), summary(StocksModel2), and summary(StocksModel3), ReturnJan, ReturnFeb, ReturnMar, ReturnJune, ReturnAug, and ReturnOct differ in sign between the models.

4.3) Cluster-Specific Predictions

Using StocksModel1, make test-set predictions called PredictTest1 on the data frame stocksTest1. Using StocksModel2, make test-set predictions called PredictTest2 on the data frame stocksTest2. Using StocksModel3, make test-set predictions called PredictTest3 on the data frame stocksTest3.

```r
predict_test1 = predict(stocks_model1,
                        type = 'response',
                        newdata = stocks_test1)

predict_test2 = predict(stocks_model2,
                        type = 'response',
                        newdata = stocks_test2)

predict_test3 = predict(stocks_model3,
                        type = 'response',
                        newdata = stocks_test3)
```

What is the overall accuracy of StocksModel1 on the test set stocksTest1, using a threshold of 0.5?

```r
table(stocks_test1$PositiveDec, predict_test1 >= 0.5)
```

```
##
##     FALSE TRUE
##   0    30  471
##   1    23  774
```

```r
(30 + 774)/ (30 + 774 + 471 + 23)
```

```
## [1] 0.6194145
```

What is the overall accuracy of StocksModel2 on the test set stocksTest2, using a threshold of 0.5?

```r
table(stocks_test2$PositiveDec, predict_test2 >= 0.5)
```

```
##
##     FALSE TRUE
##   0   388  626
##   1   309  757
```

```r
(388 + 757) / (388 + 757 + 626 + 309)
```

```
## [1] 0.5504808
```

What is the overall accuracy of StocksModel3 on the test set stocksTest3, using a threshold of 0.5?

```r
table(stocks_test3$PositiveDec, predict_test3 >= 0.5)
```

```
##
##     FALSE TRUE
##   0    49   13
##   1    21   13
```

```r
(49 + 13) / (49 + 13 + 13 + 21)
```

```
## [1] 0.6458333
```

4.4) Cluster-Specific Predictions

To compute the overall test-set accuracy of the cluster-then-predict approach, we can combine all the test-set predictions into a single vector and all the true outcomes into a single vector:

```r
all_predictions = c(predict_test1, predict_test2, predict_test3)

all_outcomes = c(stocks_test1$PositiveDec, stocks_test2$PositiveDec, stocks_test3$PositiveDec)
```

What is the overall test-set accuracy of the cluster-then-predict approach, again using a threshold of 0.5?

```r
table(all_outcomes, all_predictions >= 0.5)
```

```
##
## all_outcomes FALSE TRUE
##            0   467 1110
##            1   353 1544
```

```r
(467 + 1544) / (467 + 1544 + 1110 + 353)
```

```
## [1] 0.5788716
```

We see a modest improvement over the original logistic regression model. Since predicting stock returns is a notoriously hard problem, this is a good increase in accuracy. By investing in stocks for which we are more confident that they will have positive returns (by selecting the ones with higher predicted probabilities), this cluster-then-predict model can give us an edge over the original logistic regression model.