The Analytics Edge

Test Your Knowledge of Clustering and Recommendation Systems

Note to all. I have compiled the answers in the following format – for each question, the qualitative or "written" solutions will be provided together with their sub-questions. The R scripts (as well as the console outputs) will be provided *after* each whole question, followed by all the relevant plots. If I have missed anything in the solutions, or if you have any questions, you may email me at benjamin_tanwj@mymail.sutd.edu.sg. Thank you!

1. In this question, we will use cluster-then-predict, a methodology in which you first cluster observations and then build cluster-specific prediction models. 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.

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

Solution. We have a total of 11580 observations.

(b) What proportion of the observations have positive returns in December?

Solution. 54.6% of the observations have positive returns in December.

(c) 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.

Solution. Dropping the values equal to 1, the maximum correlation is 0.1916727856 (between the months of October and November).

(d) 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?

Solution. April has the largest average return, September has the smallest average return.

(e) 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:

```
> set.seed(144)
> spl <− sample.split(stocks$PositiveDec, SplitRatio = 0.7)
> stocksTrain <− subset(stocks, spl == TRUE)
> stocksTest <− 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. What is the overall accuracy on the training set, using a threshold of 0.5?

Solution. The confusion matrix for the training set predictions (corresponding to $t = 0.5$) is given as:

	0	1
$p < 0.5$	990	787
$p \geq 0.5$	2689	3640

with an accuracy of 0.5711818.

(f) 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?

Solution. The confusion matrix for the test set predctions (corresponding to $t = 0.5$) is given as:

	0	1
$p < 0.5$	417	344
$p \geq 0.5$	1160	1553

with an accuracy of 0.5670697.

(g) What is the accuracy on the test set of a baseline model that always predicts the most common outcome in the training set?

Solution. In the training set, we have more positive return observations in December than non-positive return observations (most common training observation for *PositiveDec* is 1).

Hence, if we predict everything in the test set as 1's, the accuracy is basically just the proportion of actual 1's in the test set, which is given by $1897/(1897+1577) = 0.5460564$.

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

```
> limitedTrain <− stocksTrain
> limitedTrain$PositiveDec <− NULL
```

> limitedTest <− stocksTest
> limitedTest$PositiveDec <− NULL

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

i. Leaving in the dependent variable might lead to unbalanced clusters
ii. Removing the dependent variable decreases the computational effort needed to cluster
iii. Needing to know the dependent variable value to assign an observation to a cluster defeats the purpose of the methodology

Solution. We remove the dependent variable because needing to know the dependent variable value to assign an observation to a cluster defeats the purpose of the methodology.

(i) In some cases where we have a training and testing set, we might want to normalize by the mean and standard deviation of the variables in the training set. We can do this by using the **caret** package passing just the training set to the preProcess function, which normalizes variables by subtracting by the mean and dividing by the standard deviation.
> library(caret)
> preproc <− preProcess(limitedTrain)
> normTrain <− predict(preproc, limitedTrain)
> normTest <− predict(preproc, limitedTest)
What is the mean of the ReturnJan variable in normTrain?
What is the mean of the ReturnJan variable in normTest?

Solution. The mean of *ReturnJan* in *normTrain* is 1.330682e-17. The mean of *ReturnJan* in *normTest* is -0.0004185886.

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

i. Small rounding errors exist in the normalization procedure
ii. The distribution of the ReturnJan variable is different in the training and testing set
iii. The distribution of the dependent variable is different in the training and testing set

Solution. The distribution of the *ReturnJan* variable is different in the training and testing set.

(k) 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?

i. Cluster 1
ii. Cluster 2

iii. Cluster 3

Solution.

1	2	3
3157	4696	253

Cluster 2 has the largest number of observations.

(l) In this question, we use the flexclust package to obtain training set and testing set cluster assignments for our observations and to do the predictions. Use the following commands:

```
> library(flexclust)
> km.kcca <− as.kcca(km, normTrain)
> clusterTrain <− predict(km.kcca)
> clusterTest <− predict(km.kcca, newdata=normTest)
```

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

Solution.

1	2	3
1298	2080	96

We assigned 2080 observations to Cluster 2.

(m) 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. Which training set data frame has the highest average value of the dependent variable?

i. stocksTrain1
ii. stocksTrain2
iii. stocksTrain3

Solution. *stocksTrain1* has the observations with the highest average value of the dependent variable (0.6024707) as compared to *stocksTrain2* (0.5140545) and *stocksTrain3* (0.4387352).

(n) 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. 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?

Solution. Variables that have at least one positive sign and one negative sign in the three models are *ReturnJan, ReturnFeb, ReturnMar, ReturnJune, ReturnAug, ReturnOct*.

(o) 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.
What is the overall accuracy of StocksModel1 on the test set stocksTest1, using a threshold of 0.5?
What is the overall accuracy of StocksModel2 on the test set stocksTest2, using a threshold of 0.5?
What is the overall accuracy of StocksModel3 on the test set stocksTest3, using a threshold of 0.5?

Solution. The confusion matrix for the *StocksModel1* predictions on *stocksTest1* (corresponding to $t = 0.5$) is given as:

	0	1
$p < 0.5$	30	23
$p \geq 0.5$	471	774

with an accuracy of 0.6191.
The confusion matrix for the *StocksModel2* predictions on *stocksTest2* (corresponding to $t = 0.5$) is given as:

	0	1
$p < 0.5$	388	309
$p \geq 0.5$	626	757

with an accuracy of 0.55.
The confusion matrix for the *StocksModel3* predictions on *stocksTest3* (corresponding to $t = 0.5$) is given as:

	0	1
$p < 0.5$	49	21
$p \geq 0.5$	13	13

with an accuracy of 0.63.

(p) 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:
> AllPredictions <− c(PredictTest1, PredictTest2, PredictTest3)

> AllOutcomes <− c(stocksTest1$PositiveDec, stocksTest2$PositiveDec, stocksTest3$PositiveDec)
What is the overall test-set accuracy of the cluster-then-predict approach, again using a threshold of 0.5?

Solution. The confusion matrix for the accumulated predictions (corresponding to $t = 0.5$) is given as:

	0	1
$p < 0.5$	467	353
$p \geq 0.5$	1110	1544

with an accuracy of 0.578.

R Scripts.

```
> #1
> #a)
> stocks <- read.csv("StocksCluster.csv")
> str(stocks)
'data.frame': 11580 obs. of  12 variables:
 $ ReturnJan  : num  0.0807 -0.0107 0.0477 -0.074 -0.031 ...
 $ ReturnFeb  : num  0.0663 0.1021 0.036 -0.0482 -0.2127 ...
 $ ReturnMar  : num  0.0329 0.1455 0.0397 0.0182 0.0915 ...
 $ ReturnApr  : num  0.1831 -0.0844 -0.1624 -0.0247 0.1893 ...
 $ ReturnMay  : num  0.13033 -0.3273 -0.14743 -0.00604 -0.15385 ...
 $ ReturnJune : num  -0.0176 -0.3593 0.0486 -0.0253 -0.1061 ...
 $ ReturnJuly : num  -0.0205 -0.0253 -0.1354 -0.094 0.3553 ...
 $ ReturnAug  : num  0.0247 0.2113 0.0334 0.0953 0.0568 ...
 $ ReturnSep  : num  -0.0204 -0.58 0 0.0567 0.0336 ...
 $ ReturnOct  : num  -0.1733 -0.2671 0.0917 -0.0963 0.0363 ...
 $ ReturnNov  : num  -0.0254 -0.1512 -0.0596 -0.0405 -0.0853 ...
 $ PositiveDec: int  0 0 0 1 1 1 1 0 0 0 ...

> #b)
> table(stocks$PositiveDec)/nrow(stocks)

       0        1
0.453886 0.546114

> #c)
> cor(stocks[,1:11])
             ReturnJan   ReturnFeb    ReturnMar    ReturnApr    ReturnMay
ReturnJan   1.00000000  0.06677458 -0.090496798 -0.037678006 -0.044411417
ReturnFeb   0.06677458  1.00000000 -0.155983263 -0.191351924 -0.095520920
ReturnMar  -0.09049680 -0.15598326  1.000000000  0.009726288 -0.003892789
```

```
ReturnApr  -0.03767801 -0.19135192  0.009726288  1.000000000  0.063822504
ReturnMay  -0.04441142 -0.09552092 -0.003892789  0.063822504  1.000000000
ReturnJune  0.09223831  0.16999448 -0.085905486 -0.011027752 -0.021074539
ReturnJuly -0.08142976 -0.06177851  0.003374160  0.080631932  0.090850264
ReturnAug  -0.02279202  0.13155979 -0.022005400 -0.051756051 -0.033125658
ReturnSep  -0.02643715  0.04350177  0.076518327 -0.028920972  0.021962862
ReturnOct   0.14297723 -0.08732427 -0.011923758  0.048540025  0.017166728
ReturnNov   0.06763233 -0.15465828  0.037323535  0.031761837  0.048046590
             ReturnJune    ReturnJuly     ReturnAug     ReturnSep   ReturnOct
ReturnJan   0.09223831 -0.0814297650 -0.0227920187 -0.0264371526  0.14297723
ReturnFeb   0.16999448 -0.0617785094  0.1315597863  0.0435017706 -0.08732427
ReturnMar  -0.08590549  0.0033741597 -0.0220053995  0.0765183267 -0.01192376
ReturnApr  -0.01102775  0.0806319317 -0.0517560510 -0.0289209718  0.04854003
ReturnMay  -0.02107454  0.0908502642 -0.0331256580  0.0219628623  0.01716673
ReturnJune  1.00000000 -0.0291525996  0.0107105260  0.0447472692 -0.02263599
ReturnJuly -0.02915260  1.0000000000  0.0007137558  0.0689478037 -0.05470891
ReturnAug   0.01071053  0.0007137558  1.0000000000  0.0007407139 -0.07559456
ReturnSep   0.04474727  0.0689478037  0.0007407139  1.0000000000 -0.05807924
ReturnOct  -0.02263599 -0.0547089088 -0.0755945614 -0.0580792362  1.00000000
ReturnNov  -0.06527054 -0.0483738369 -0.1164890345 -0.0197197998  0.19167279
              ReturnNov
ReturnJan   0.06763233
ReturnFeb  -0.15465828
ReturnMar   0.03732353
ReturnApr   0.03176184
ReturnMay   0.04804659
ReturnJune -0.06527054
ReturnJuly -0.04837384
ReturnAug  -0.11648903
ReturnSep  -0.01971980
ReturnOct   0.19167279
ReturnNov   1.00000000
> sort(cor(stocks[,1:11]))
  [1] -0.1913519239 -0.1913519239 -0.1559832630 -0.1559832630 -0.1546582815
  [6] -0.1546582815 -0.1164890345 -0.1164890345 -0.0955209197 -0.0955209197
 [11] -0.0904967978 -0.0904967978 -0.0873242672 -0.0873242672 -0.0859054862
 [16] -0.0859054862 -0.0814297650 -0.0814297650 -0.0755945614 -0.0755945614
 [21] -0.0652705413 -0.0652705413 -0.0617785094 -0.0617785094 -0.0580792362
 [26] -0.0580792362 -0.0547089088 -0.0547089088 -0.0517560510 -0.0517560510
 [31] -0.0483738369 -0.0483738369 -0.0444114168 -0.0444114168 -0.0376780060
 [36] -0.0376780060 -0.0331256580 -0.0331256580 -0.0291525996 -0.0291525996
 [41] -0.0289209718 -0.0289209718 -0.0264371526 -0.0264371526 -0.0227920187
 [46] -0.0227920187 -0.0226359936 -0.0226359936 -0.0220053995 -0.0220053995
 [51] -0.0210745388 -0.0210745388 -0.0197197998 -0.0197197998 -0.0119237577
 [56] -0.0119237577 -0.0110277522 -0.0110277522 -0.0038927894 -0.0038927894
 [61]  0.0007137558  0.0007137558  0.0007407139  0.0007407139  0.0033741597
 [66]  0.0033741597  0.0097262876  0.0097262876  0.0107105260  0.0107105260
 [71]  0.0171667279  0.0171667279  0.0219628623  0.0219628623  0.0317618366
 [76]  0.0317618366  0.0373235349  0.0373235349  0.0435017706  0.0435017706
 [81]  0.0447472692  0.0447472692  0.0480465902  0.0480465902  0.0485400254
 [86]  0.0485400254  0.0638225039  0.0638225039  0.0667745831  0.0667745831
 [91]  0.0676323333  0.0676323333  0.0689478037  0.0689478037  0.0765183267
 [96]  0.0765183267  0.0806319317  0.0806319317  0.0908502642  0.0908502642
[101]  0.0922383068  0.0922383068  0.1315597863  0.1315597863  0.1429772286
[106]  0.1429772286  0.1699944833  0.1699944833  0.1916727856  0.1916727856
[111]  1.0000000000  1.0000000000  1.0000000000  1.0000000000  1.0000000000
[116]  1.0000000000  1.0000000000  1.0000000000  1.0000000000  1.0000000000
[121]  1.0000000000

> #d)
> sort(colMeans(stocks[,1:11]))
   ReturnSep    ReturnFeb   ReturnJuly    ReturnOct   ReturnJune    ReturnNov
-0.014720768 -0.007604784  0.003050863  0.005650844  0.005937902  0.011387440
   ReturnJan    ReturnAug    ReturnMar    ReturnMay    ReturnApr
 0.012631602  0.016198265  0.019402336  0.024736591  0.026308147

> #e)
> set.seed(144)
> library(caTools)
> spl <- sample.split(stocks$PositiveDec, SplitRatio = .7)
> stocksTrain <- subset(stocks, spl == TRUE)
> stocksTest <- subset(stocks, spl == FALSE)
> stocksModel <- glm(PositiveDec ~ ., data = stocksTrain, family = "binomial")
> stocksPredict <- predict(stocksModel, newdata = stocksTrain, type = "response")
> table(stocksPredict >= .5, stocksTrain$PositiveDec)

          0    1
  FALSE  990  787
  TRUE  2689 3640

> #f)
> stocksPredicttest <- predict(stocksModel, newdata = stocksTest, type = "response")
> table(stocksPredicttest >= .5, stocksTest$PositiveDec)

          0    1
  FALSE  417  344
  TRUE  1160 1553
```

```
> #g)
> table(stocksTrain$PositiveDec)

   0    1 
3679 4427 
> # 0    1
> # 3679 4427
> # The accuracy is 4427/(4427+3679) = 0.5461387
> table(stocksTest$PositiveDec)

   0    1 
1577 1897 

> #h)
> limitedTrain <- stocksTrain
> limitedTrain$PositiveDec <- NULL
> limitedTest <- stocksTest
> limitedTest$PositiveDec <- NULL

> #i)
> library(caret)
Loading required package: lattice
Loading required package: ggplot2
> preproc <- caret::preProcess(limitedTrain)
> normTrain <- predict(preproc, limitedTrain)
> normTest <- predict(preproc, limitedTest)
> colMeans(normTrain)
    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(normTest)
    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 

> #k)
> set.seed(144)
> km <- kmeans(normTrain, centers = 3)
> table(km$cluster)

   1    2    3 
3157 4696  253 

> #l)
> library(flexclust)
Loading required package: grid
Loading required package: modeltools
Loading required package: stats4
> km.kcca <- as.kcca(km, normTrain)
Found more than one class "kcca" in cache; using the first, from namespace 'kernlab'
Also defined by flexclust
Found more than one class "kcca" in cache; using the first, from namespace 'kernlab'
Also defined by flexclust
> clusterTrain <- predict(km.kcca)
Found more than one class "kcca" in cache; using the first, from namespace 'kernlab'
Also defined by flexclust
> clusterTest <- predict(km.kcca, newdata = normTest)
> table(clusterTest)
clusterTest
   1    2    3 
1298 2080   96 

> #m)
> stocksTrain1 <- subset(stocksTrain, clusterTrain == 1)
> stocksTrain2 <- subset(stocksTrain, clusterTrain == 2)
> stocksTrain3 <- subset(stocksTrain, clusterTrain == 3)
> stocksTest1 <- subset(stocksTest, clusterTest == 1)
> stocksTest2 <- subset(stocksTest, clusterTest == 2)
> stocksTest3 <- subset(stocksTest, clusterTest == 3)
> mean(stocksTrain1$PositiveDec) # 0.6024707
[1] 0.6024707
> mean(stocksTrain2$PositiveDec) # 0.5140545
[1] 0.5140545
> mean(stocksTrain3$PositiveDec) # 0.4387352
[1] 0.4387352

> #n)
> StocksModel1 <- glm(PositiveDec ~ ., data = stocksTrain1, family = "binomial")
> StocksModel2 <- glm(PositiveDec ~ ., data = stocksTrain2, family = "binomial")
> StocksModel3 <- glm(PositiveDec ~ ., data = stocksTrain3, family = "binomial")
> summary(StocksModel1)
```

```
Call:
glm(formula = PositiveDec ~ ., family = "binomial", data = stocksTrain1)

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

> summary(StocksModel2)

Call:
glm(formula = PositiveDec ~ ., family = "binomial", data = stocksTrain2)

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

> summary(StocksModel3)

Call:
glm(formula = PositiveDec ~ ., family = "binomial", data = stocksTrain3)

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

> #o)
> PredictTest1 <- predict(StocksModel1, newdata = stocksTest1, type = "response")
> PredictTest2 <- predict(StocksModel2, newdata = stocksTest2, type = "response")
> PredictTest3 <- predict(StocksModel3, newdata = stocksTest3, type = "response")
> table1 <- table(PredictTest1 >= .5, stocksTest1$PositiveDec )
> table1

          0   1
  FALSE  30  23
  TRUE  471 774
> # 0   1
> # FALSE  30  23
> # TRUE  471 774
> # Accuracy = 0.6191
> table2 <- table(PredictTest2 >= .5, stocksTest2$PositiveDec )
> table2

          0   1
  FALSE 388 309
  TRUE  626 757
> # 0   1
> # FALSE 388 309
> # TRUE  626 757
> # Accuracy = 0.55
> table3 <- table(PredictTest3 >= .5, stocksTest3$PositiveDec )
> table3

         0  1
  FALSE 49 21
  TRUE  13 13

> #p)
> AllPredictions <- c(PredictTest1, PredictTest2, PredictTest3)
> AllOutcomes <- c(stocksTest1$PositiveDec, stocksTest2$PositiveDec, stocksTest3$PositiveDec)
> table4 <- table(AllPredictions >= 0.5, AllOutcomes)
> table4
     AllOutcomes
         0    1
  FALSE  467  353
  TRUE  1110 1544
```

2. Clustering is commonly used to divide a broad target market of customers into smaller, similar groups and then to design marketing strategies specifically for each group. In this question, you will use clustering to study publicly available data from the New York Citi Bike sharing program. Citi Bike is the largest bike sharing program in the United States and serves various parts of the New York city. The data is provided in the file **citibike.csv** and contains trip information for the bike rides for the month of July 2013:

   - **tripduration**: Time duration of the trip (in seconds)
   - **startstation**: Name of the start station
   - **endstation**: Name of the end station
   - **gender**: Gender of user (1 = male, 2 = female)
   - **age**: Age of the user
   - **day**: Day on which the trip was started (Mon, Tue, Wed, Thu, Fri, Sat, Sun)
   - **starttime**: Start time of the trip in unit of hours (measured from 0 (12am) to 23 (11 pm))

   (a) Read the dataset into the dataframe **citi**. How many bike stations are there in this dataset?

   Solution. There are 329 bike stations.

   (b) On which day of the week, is the average duration of the trips taken by bikers the maximum?

   Solution.

Fri	Mon	Sat	Sun	Thu	Tue	Wed
832.3580	853.6248	894.2661	887.5528	865.7822	857.4895	843.5679

   We can see that the longest average trip is on Saturdays.

   (c) What is the start hour when the maximum number of bikes are rented? What is the start hour when the minimum number of bikes are rented?

   Solution. The start hour when the maximum number of bikes are rented is 6pm (18), and that for the minimum number of rented bikes is 4am (4).

   (d) In this dataset, what proportion of the bikes are rented by female users?

   Solution. 23.4%.

(e) One of the challenges to do clustering with this data is the presence of the categorical variable for the **day** variable. To tackle this, we will define seven new binary variables (**Mon** to **Sun**), each of which takes a value of 1 if the corresponding trip is taken on that day and 0 otherwise. Write down one such sample R command(s) that you use to do this for a given day of the week.

Solution.

```
citi$Mon <- as.integer(citi$day == "Mon")
citi$Tue <- as.integer(citi$day == "Tue")
citi$Wed <- as.integer(citi$day == "Wed")
citi$Thu <- as.integer(citi$day == "Thu")
citi$Fri <- as.integer(citi$day == "Fri")
citi$Sat <- as.integer(citi$day == "Sat")
citi$Sun <- as.integer(citi$day == "Sun")
```

(f) In clustering data, it is often important to normalize the variables so that they are all on the same scale. If you clustered this dataset without normalizing, which variable would you expect to dominate in the distance calculations?

- **tripduration**
- **gender**
- **age**
- **starttime**
- **Mon**

Solution. The trip duration.

(g) Normalize the variables **tripduration**, **gender**, **age**, **starttime**, **Mon**, ..., **Sun** by using the **scale()** function in R. We normalize such that each variable has mean 0 and standard deviation 1. What is the maximum value of **tripduration** in the normalized dataset?

Solution. The max value of *tripduration* in the normalised dataset is 402.9514.

(h) We will not use hierarchical clustering for this dataset. Why do you think hierarchical clustering might have a problem with this dataset?

- We have categorical variables in this dataset, so we cant use hierarchical clustering.
- We might have too many variables in the dataset for hierarchical clustering to handle.
- We might have too many observations in the dataset for hierarchical clustering to handle.

- We are sure of the number of clusters in this application, so using hierarchical clustering does not make sense.

Solution. There are too many observations.

(i) Run the k-means algorithm on the normalized dataset with 10 clusters. Use set.seed(100) in R just before running the code. You only want to use the variables **tripduration**, **gender**, **age**, **starttime**, **Mon**, ..., **Sun** in building your clusters. Use the default settings to build your model. How many trips are there in the largest and smallest clusters respectively?

Solution. Largest cluster has 107185 trips, and the smallest cluster has 18148 trips.

(j) Which cluster best fits the description "trips taken primarily by older users on Saturdays"? You can use the centers of the clusters to answer this question.

Solution. Cluster 5.

(k) Which cluster best fits the description "longer trips taken primarily by female users either on Tuesdays or Wednesdays"? You can use the centers of the clusters to answer this question.

Solution. Cluster 10.

(l) If we ran k-means clustering a second time without making any additional calls to set.seed, we would expect

- Different results from the first k-means clustering
- Identical results to the first k-means clustering

Solution. Different results – we can only replicate results if we use the same seed.

(m) If we ran k-means clustering a second time, again running the command set.seed(100) right before doing the clustering, we would expect

- Different results from the first k-means clustering
- Identical results to the first k-means clustering

Solution. Identical results.

(n) Suppose the marketing department at Citi Bike decided that instead of using the days of the week for clustering, they would like to use a single variable **weekday** which took a value of 1 if the trip started on Monday, Tuesday, Wednesday, Thursday, or Friday and 0 if it started on a Saturday or Sunday. Redo the clustering. As before, remember to normalize the **weekday** variable and run the k-means algorithm on the normalized dataset with 10 clusters. Use set.seed(100) in R before running the code. Which cluster best fits the description "longer trips taken by older female users on weekdays"? You can use the centers of the clusters to answer this question.

Solution. Cluster 1.

(o) Which cluster best fits the description "short trips taken by younger male users early on weekdays"? You can use the centers of the clusters to answer this question.

Solution. Cluster 9.

R Scripts.

```
> #2)
> #a)
> citi <- read.csv("citibike.csv")
> unique(citi$startstation)
  [1] E 14 St & Avenue B                  University Pl & E 14 St
  [3] Broadway & W 49 St                  1 Ave & E 15 St
  [5] Hudson St & Reade St                Mott St & Prince St
  [7] W 41 St & 8 Ave                     W 20 St & 8 Ave
  [9] W 13 St & 6 Ave                     W 59 St & 10 Ave
 [11] W 26 St & 8 Ave                     E 20 St & FDR Drive
 [13] Cleveland Pl & Spring St            W 56 St & 6 Ave
 [15] Broadway & E 14 St                  St Marks Pl & 2 Ave
 [17] E 11 St & 2 Ave                     Metropolitan Ave & Bedford Ave
 [19] W 14 St & The High Line             Cumberland St & Lafayette Ave
 [21] E 55 St & 2 Ave                     W 51 St & 6 Ave
 [23] Willoughby St & Fleet St            Pershing Square N
 [25] W 52 St & 9 Ave                     Greenwich St & N Moore St
 [27] Dean St & 4 Ave                     E 17 St & Broadway
 [29] Broadway & W 37 St                  Cliff St & Fulton St
 [31] 8 Ave & W 31 St                     E 11 St & Broadway
 [33] St James Pl & Oliver St             Washington Pl & Broadway
 [35] Broadway & W 39 St                  W 49 St & 8 Ave
 [37] 9 Ave & W 18 St                     W 20 St & 7 Ave
 [39] Murray St & West St                 W 38 St & 8 Ave
 [41] E 3 St & 1 Ave                    Spruce St & Nassau St
 [43] 9 Ave & W 14 St                   Laight St & Hudson St
 [45] Lefferts Pl & Franklin Ave        2 Ave & E 31 St
 [47] Stanton St & Chrystie St          Broadway & E 22 St
 [49] W 27 St & 7 Ave                   Washington Square E
 [51] Mercer St & Bleecker St           E 58 St & 3 Ave
 [53] W 4 St & 7 Ave S                  E 31 St & 3 Ave
 [55] E 32 St & Park Ave                Broadway & W 29 St
 [57] W 37 St & 10 Ave                  Suffolk St & Stanton St
 [59] 1 Ave & E 18 St                   E 39 St & 2 Ave
 [61] 2 Ave & E 58 St                   Park Pl & Church St
 [63] E 25 St & 2 Ave                   Sullivan St & Washington Sq
 [65] 6 Ave & Broome St                 W 22 St & 10 Ave
 [67] Clinton St & Joralemon St         Broadway & W 60 St
 [69] Jay St & Tech Pl                  Great Jones St
 [71] W 43 St & 6 Ave                   E 39 St & 3 Ave
 [73] Lawrence St & Willoughby St       W 37 St & 5 Ave
 [75] Clinton St & Tillary St           10 Ave & W 28 St
 [77] Greenwich Ave & 8 Ave             E 7 St & Avenue A
 [79] W 43 St & 10 Ave                  W 45 St & 6 Ave
 [81] E 2 St & 2 Ave                    9 Ave & W 22 St
 [83] W 54 St & 9 Ave                   W 24 St & 7 Ave
 [85] Broadway & W 55 St                E 56 St & 3 Ave
 [87] West St & Chambers St             LaGuardia Pl & W 3 St
 [89] Allen St & E Houston St           W 13 St & 7 Ave
 [91] Atlantic Ave & Fort Greene Pl     E 43 St & Vanderbilt Ave
 [93] Pearl St & Hanover Square         Broadway & W 51 St
 [95] Grand St & Havemeyer St           Greenwich St & Warren St
 [97] Franklin St & W Broadway          W 15 St & 7 Ave
 [99] Forsyth St & Broome St            Barrow St & Hudson St
[101] Hicks St & Montague St            W 20 St & 11 Ave
[103] Fulton St & William St            Henry St & Atlantic Ave
[105] Rivington St & Chrystie St        E 48 St & 3 Ave
[107] Church St & Leonard St            E 45 St & 3 Ave
[109] 8 Ave & W 33 St                   5 Ave & E 29 St
[111] 6 Ave & W 33 St                   W 29 St & 9 Ave
[113] E 13 St & Avenue A                E 19 St & 3 Ave
[115] Allen St & Rivington St           E 4 St & 2 Ave
[117] Bond St & Schermerhorn St         MacDougal St & Prince St
[119] Lafayette St & E 8 St             Vesey Pl & River Terrace
[121] Barclay St & Church St            Columbia St & Rivington St
[123] West Thames St                    Lafayette Ave & St James Pl
[125] 12 Ave & W 40 St                  Lexington Ave & E 24 St
[127] E 11 St & 1 Ave                   E 23 St & 1 Ave
```

[129] 1 Ave & E 30 St	Division St & Bowery
[131] Broadway & W 53 St	Broadway & Battery Pl
[133] Ashland Pl & Hanson Pl	E 30 St & Park Ave S
[135] Warren St & Church St	Front St & Gold St
[137] Bedford Ave & S 9th St	W 17 St & 8 Ave
[139] Christopher St & Greenwich St	South End Ave & Liberty St
[141] W 22 St & 8 Ave	St Marks Pl & 1 Ave
[143] Lexington Ave & E 26 St	Forsyth St & Canal St
[145] E 33 St & 1 Ave	Greenwich Ave & Charles St
[147] Broadway & W 41 St	E 47 St & 2 Ave
[149] E 25 St & 1 Ave	Centre St & Chambers St
[151] Norfolk St & Broome St	Centre St & Worth St
[153] Lispenard St & Broadway	W 33 St & 7 Ave
[155] Bayard St & Baxter St	W Broadway & Spring St
[157] S 5 Pl & S 4 St	W 11 St & 6 Ave
[159] Front St & Washington St	E 16 St & Irving Pl
[161] W 47 St & 10 Ave	Old Fulton St
[163] W 42 St & 8 Ave	Clermont Ave & Lafayette Ave
[165] W 18 St & 6 Ave	E 2 St & Avenue B
[167] Washington St & Gansevoort St	Broadway & Berry St
[169] 9 Ave & W 45 St	DeKalb Ave & Hudson Ave
[171] Macon St & Nostrand Ave	Madison St & Clinton St
[173] E 53 St & Lexington Ave	Watts St & Greenwich St
[175] Washington Pl & 6 Ave	Clark St & Henry St
[177] Fulton St & Waverly Ave	Bank St & Hudson St
[179] W 21 St & 6 Ave	Wythe Ave & Metropolitan Ave
[181] 6 Ave & Canal St	Broadway & W 58 St
[183] Broadway & W 36 St	W Houston St & Hudson St
[185] Lexington Ave & Classon Ave	W 45 St & 8 Ave
[187] E 51 St & Lexington Ave	E 9 St & Avenue C
[189] Carmine St & 6 Ave	Bank St & Washington St
[191] South St & Whitehall St	William St & Pine St
[193] E 52 St & 2 Ave	Willoughby Ave & Hall St
[195] W 39 St & 9 Ave	State St & Smith St
[197] Cherry St	Pershing Square S
[199] Canal St & Rutgers St	Mercer St & Spring St
[201] W 26 St & 10 Ave	E 10 St & Avenue A
[203] Montague St & Clinton St	Clinton Ave & Myrtle Ave
[205] W 52 St & 11 Ave	DeKalb Ave & Skillman St
[207] W 53 St & 10 Ave	Lafayette St & Jersey St
[209] Perry St & Bleecker St	Clinton St & Grand St
[211] Adelphi St & Myrtle Ave	Henry St & Poplar St
[213] W 31 St & 7 Ave	York St & Jay St
[215] W 25 St & 6 Ave	E 16 St & 5 Ave

```
[217] W 13 St & 5 Ave                   8 Ave & W 52 St
[219] E 47 St & 1 Ave                   11 Ave & W 27 St
[221] W 52 St & 5 Ave                   Broadway & W 32 St
[223] 11 Ave & W 41 St                  E 20 St & Park Ave
[225] E 6 St & Avenue B                 E 59 St & Sutton Pl
[227] Allen St & Hester St              Lafayette Ave & Classon Ave
[229] Maiden Ln & Pearl St              Avenue D & E 12 St
[231] Harrison St & Hudson St           Lafayette Ave & Fort Greene Pl
[233] Avenue D & E 8 St                 E 15 St & 3 Ave
[235] Elizabeth St & Hester St          E 37 St & Lexington Ave
[237] W 56 St & 10 Ave                  E 40 St & 5 Ave
[239] E 56 St & Madison Ave             Broadway & W 24 St
[241] Johnson St & Gold St              E 53 St & Madison Ave
[243] E 5 St & Avenue C                 MacDougal St & Washington Sq
[245] S Portland Ave & Hanson Pl        E 12 St & 3 Ave
[247] E 51 St & 1 Ave                   E 2 St & Avenue C
[249] Fulton St & Clermont Ave          Howard St & Centre St
[251] Washington Ave & Park Ave         Fulton St & Rockwell Pl
[253] St James Pl & Pearl St            Front St & Maiden Ln
[255] S 3 St & Bedford Ave              John St & William St
[257] Concord St & Bridge St            Avenue D & E 3 St
[259] Liberty St & Broadway             Duane St & Greenwich St
[261] Central Park S & 6 Ave            Madison St & Montgomery St
[263] Duffield St & Willoughby St       9 Ave & W 16 St
[265] Little West St & 1 Pl             W 46 St & 11 Ave
[267] W 16 St & The High Line           Emerson Pl & Myrtle Ave
[269] Carlton Ave & Park Ave            Joralemon St & Adams St
[271] State St                          1 Ave & E 44 St
[273] Rivington St & Ridge St           W 49 St & 5 Ave
[275] Bialystoker Pl & Delancey St      Reade St & Broadway
[277] Pitt St & Stanton St              E 10 St & 5 Ave
[279] E 33 St & 5 Ave                   Pike St & E Broadway
[281] Gallatin Pl & Livingston St       E 6 St & Avenue D
[283] Grand Army Plaza & Central Park S Pearl St & Anchorage Pl
[285] Monroe St & Bedford Ave           E 27 St & 1 Ave
[287] Shevchenko Pl & E 6 St            E 43 St & 2 Ave
[289] Broad St & Bridge St              Willoughby Ave & Walworth St
[291] S 4 St & Wythe Ave                DeKalb Ave & S Portland Ave
[293] Water - Whitehall Plaza           Kent Ave & S 11 St
[295] Stanton St & Mangin St            Old Slip & Front St
[297] Flushing Ave & Carlton Ave        E 47 St & Park Ave
[299] Atlantic Ave & Furman St          Washington Ave & Greene Ave
[301] Market St & Cherry St             3 Ave & Schermerhorn St
[303] E 20 St & 2 Ave                   FDR Drive & E 35 St
```

```
[305] Nassau St & Navy St              Pike St & Monroe St
[307] Fulton St & Grand Ave            South St & Gouverneur Ln
[309] Grand St & Greene St             DeKalb Ave & Vanderbilt Ave
[311] Clinton Ave & Flushing Ave       E 55 St & Lexington Ave
[313] Hancock St & Bedford Ave         Sands St & Gold St
[315] W 44 St & 5 Ave                  Washington Park
[317] Myrtle Ave & St Edwards St       Cadman Plaza E & Tillary St
[319] Franklin Ave & Myrtle Ave        Columbia Heights & Cranberry St
[321] Clermont Ave & Park Ave          W 34 St & 11 Ave
[323] Catherine St & Monroe St         Monroe St & Classon Ave
[325] Hanover Pl & Livingston St       Park Ave & St Edwards St
[327] Cadman Plaza E & Red Cross Pl    Railroad Ave & Kay Ave
[329] 7 Ave & Farragut St
329 Levels: 1 Ave & E 15 St 1 Ave & E 18 St 1 Ave & E 30 St ... York St & Jay St
> nlevels(unique(citi$startstation))
[1] 329
> unique(citi$endstation)
  [1] E 51 St & 1 Ave                  Suffolk St & Stanton St
  [3] 9 Ave & W 45 St                  E 20 St & FDR Drive
  [5] Front St & Maiden Ln             Canal St & Rutgers St
  [7] W 37 St & 5 Ave                  W 49 St & 8 Ave
  [9] 1 Ave & E 44 St                  E 47 St & 1 Ave
 [11] Barrow St & Hudson St            E 4 St & 2 Ave
 [13] Lafayette St & E 8 St            8 Ave & W 33 St
 [15] Forsyth St & Broome St           W 47 St & 10 Ave
 [17] E 10 St & Avenue A               Grand St & Havemeyer St
 [19] 1 Ave & E 18 St                  W Houston St & Hudson St
 [21] E 11 St & 1 Ave                  Carmine St & 6 Ave
 [23] DeKalb Ave & Vanderbilt Ave      E 15 St & 3 Ave
 [25] W Broadway & Spring St           W 45 St & 6 Ave
 [27] Adelphi St & Myrtle Ave          E 33 St & 1 Ave
 [29] Emerson Pl & Myrtle Ave          Broadway & W 32 St
 [31] W 15 St & 7 Ave                  E 5 St & Avenue C
 [33] Great Jones St                   Centre St & Worth St
 [35] 2 Ave & E 31 St                  E 6 St & Avenue B
 [37] 11 Ave & W 27 St                 E 12 St & 3 Ave
 [39] W 39 St & 9 Ave                  W 56 St & 10 Ave
 [41] W 54 St & 9 Ave                  Maiden Ln & Pearl St
 [43] E 39 St & 2 Ave                  Kent Ave & S 11 St
 [45] E 20 St & Park Ave               Allen St & Rivington St
 [47] E 56 St & 3 Ave                  University Pl & E 14 St
 [49] Mercer St & Bleecker St          Fulton St & William St
 [51] E 58 St & 3 Ave                  9 Ave & W 14 St
 [53] E 25 St & 2 Ave                  E 13 St & Avenue A
 [55] South End Ave & Liberty St       E 47 St & 2 Ave
 [57] W 31 St & 7 Ave                  W 52 St & 11 Ave
 [59] E 48 St & 3 Ave                  Washington Pl & Broadway
 [61] E 32 St & Park Ave               Broadway & Berry St
 [63] 11 Ave & W 41 St                 W 24 St & 7 Ave
 [65] Clark St & Henry St              Norfolk St & Broome St
 [67] 9 Ave & W 22 St                  Franklin St & W Broadway
 [69] W 59 St & 10 Ave                 Elizabeth St & Hester St
 [71] E 55 St & 2 Ave                  W 46 St & 11 Ave
 [73] E 16 St & 5 Ave                  E 3 St & 1 Ave
 [75] W 26 St & 8 Ave                  Greenwich Ave & 8 Ave
 [77] E 17 St & Broadway               Church St & Leonard St
 [79] W 20 St & 11 Ave                 E 2 St & 2 Ave
 [81] W 42 St & 8 Ave                  Spruce St & Nassau St
 [83] Columbia St & Rivington St       S 5 Pl & S 4 St
 [85] Lexington Ave & E 24 St          William St & Pine St
 [87] E 33 St & 5 Ave                  Cliff St & Fulton St
 [89] Cleveland Pl & Spring St         Mercer St & Spring St
 [91] W 20 St & 7 Ave                  Bayard St & Baxter St
 [93] Greenwich St & Warren St         Washington Pl & 6 Ave
 [95] Division St & Bowery             E 25 St & 1 Ave
 [97] Dean St & 4 Ave                  Washington St & Gansevoort St
 [99] W 18 St & 6 Ave                  W 37 St & 10 Ave
[101] E 31 St & 3 Ave                  Sullivan St & Washington Sq
[103] Flushing Ave & Carlton Ave       St Marks Pl & 2 Ave
[105] Allen St & E Houston St          Forsyth St & Canal St
[107] Hicks St & Montague St           1 Ave & E 15 St
[109] Lafayette Ave & St James Pl      Lawrence St & Willoughby St
[111] DeKalb Ave & Hudson Ave          S 3 St & Bedford Ave
[113] Hudson St & Reade St             E 19 St & 3 Ave
[115] Henry St & Atlantic Ave          Clinton Ave & Flushing Ave
[117] West Thames St                   LaGuardia Pl & W 3 St
[119] Washington Ave & Park Ave        Harrison St & Hudson St
[121] Broadway & W 41 St               E 43 St & Vanderbilt Ave
[123] Pershing Square N                Broadway & W 58 St
[125] W 11 St & 6 Ave                  1 Ave & E 30 St
[127] Fulton St & Clermont Ave         St Marks Pl & 1 Ave
[129] E 23 St & 1 Ave                  Clinton St & Grand St
[131] Rivington St & Ridge St          W 13 St & 7 Ave
[133] E 2 St & Avenue B                Vesey Pl & River Terrace
[135] E 53 St & Lexington Ave          8 Ave & W 31 St
[137] E 51 St & Lexington Ave          W 17 St & 8 Ave
[139] W 22 St & 10 Ave                 E 7 St & Avenue A
[141] Bialystoker Pl & Delancey St     Broadway & W 39 St
```

```
[143] W 13 St & 5 Ave                   W 41 St & 8 Ave
[145] E 11 St & 2 Ave                   Jay St & Tech Pl
[147] W 27 St & 7 Ave                   Pearl St & Anchorage Pl
[149] Avenue D & E 12 St                W 53 St & 10 Ave
[151] Macon St & Nostrand Ave           Clermont Ave & Lafayette Ave
[153] St James Pl & Pearl St            W 43 St & 6 Ave
[155] Warren St & Church St             Franklin Ave & Myrtle Ave
[157] 9 Ave & W 18 St                   W 52 St & 9 Ave
[159] Bond St & Schermerhorn St         W 43 St & 10 Ave
[161] Liberty St & Broadway             West St & Chambers St
[163] Murray St & West St               W 22 St & 8 Ave
[165] W 20 St & 8 Ave                   E 30 St & Park Ave S
[167] E 20 St & 2 Ave                   E 56 St & Madison Ave
[169] Central Park S & 6 Ave            Monroe St & Bedford Ave
[171] Metropolitan Ave & Bedford Ave    W 29 St & 9 Ave
[173] W 38 St & 8 Ave                   Bank St & Hudson St
[175] Broadway & W 36 St                W 13 St & 6 Ave
[177] State St                          E 10 St & 5 Ave
[179] DeKalb Ave & S Portland Ave       W 25 St & 6 Ave
[181] Broadway & W 60 St                Stanton St & Mangin St
[183] Fulton St & Grand Ave             Allen St & Hester St
[185] Avenue D & E 3 St                 Little West St & 1 Pl
[187] W 4 St & 7 Ave S                  MacDougal St & Prince St
[189] FDR Drive & E 35 St               MacDougal St & Washington Sq
[191] Fulton St & Waverly Ave           Clinton Ave & Myrtle Ave
[193] E 45 St & 3 Ave                   Greenwich St & N Moore St
[195] Avenue D & E 8 St                 Cherry St
[197] Stanton St & Chrystie St          E 52 St & 2 Ave
[199] Water - Whitehall Plaza           Broadway & W 29 St
[201] Willoughby Ave & Hall St          St James Pl & Oliver St
[203] 12 Ave & W 40 St                  2 Ave & E 58 St
[205] Broadway & W 53 St                S Portland Ave & Hanson Pl
[207] Washington Ave & Greene Ave       Shevchenko Pl & E 6 St
[209] Clinton St & Tillary St           South St & Whitehall St
[211] W 56 St & 6 Ave                   E 16 St & Irving Pl
[213] Pitt St & Stanton St              W 33 St & 7 Ave
[215] Watts St & Greenwich St           Lefferts Pl & Franklin Ave
[217] Perry St & Bleecker St            Front St & Washington St
[219] Johnson St & Gold St              Pike St & Monroe St
[221] Christopher St & Greenwich St     Broadway & W 51 St
[223] Broadway & W 24 St                8 Ave & W 52 St
[225] E 14 St & Avenue B                Lexington Ave & E 26 St
[227] Lafayette Ave & Classon Ave       Pearl St & Hanover Square
[229] Mott St & Prince St               Duane St & Greenwich St
```

```
[231] W 14 St & The High Line          Lafayette St & Jersey St
[233] Pershing Square S                E 39 St & 3 Ave
[235] E 11 St & Broadway               6 Ave & Broome St
[237] Lispenard St & Broadway          Madison St & Clinton St
[239] S 4 St & Wythe Ave               E 40 St & 5 Ave
[241] 10 Ave & W 28 St                 Willoughby Ave & Walworth St
[243] Madison St & Montgomery St       Concord St & Bridge St
[245] W 49 St & 5 Ave                  E 37 St & Lexington Ave
[247] Broadway & Battery Pl            Washington Square E
[249] Broadway & E 22 St               Greenwich Ave & Charles St
[251] Bank St & Washington St          E 9 St & Avenue C
[253] Broadway & E 14 St               Atlantic Ave & Fort Greene Pl
[255] Pike St & E Broadway             Lafayette Ave & Fort Greene Pl
[257] Rivington St & Chrystie St       6 Ave & W 33 St
[259] Laight St & Hudson St            Broadway & W 49 St
[261] Ashland Pl & Hanson Pl           E 47 St & Park Ave
[263] Barclay St & Church St           Broadway & W 37 St
[265] Lexington Ave & Classon Ave      Broadway & W 55 St
[267] E 2 St & Avenue C                Myrtle Ave & St Edwards St
[269] W 26 St & 10 Ave                 State St & Smith St
[271] Cadman Plaza E & Tillary St      9 Ave & W 16 St
[273] Duffield St & Willoughby St      Centre St & Chambers St
[275] Cumberland St & Lafayette Ave    John St & William St
[277] 6 Ave & Canal St                 Market St & Cherry St
[279] Wythe Ave & Metropolitan Ave     W 51 St & 6 Ave
[281] Howard St & Centre St            W 16 St & The High Line
[283] Reade St & Broadway              W 45 St & 8 Ave
[285] E 59 St & Sutton Pl              W 21 St & 6 Ave
[287] Park Pl & Church St              Clinton St & Joralemon St
[289] Broad St & Bridge St             Gallatin Pl & Livingston St
[291] Catherine St & Monroe St         Grand Army Plaza & Central Park S
[293] Fulton St & Rockwell Pl          E 43 St & 2 Ave
[295] Hancock St & Bedford Ave         E 6 St & Avenue D
[297] Washington Park                  W 34 St & 11 Ave
[299] E 27 St & 1 Ave                  Willoughby St & Fleet St
[301] 5 Ave & E 29 St                  DeKalb Ave & Skillman St
[303] Joralemon St & Adams St          Clermont Ave & Park Ave
[305] Montague St & Clinton St         Carlton Ave & Park Ave
[307] Atlantic Ave & Furman St         Bedford Ave & S 9th St
[309] W 44 St & 5 Ave                  Monroe St & Classon Ave
[311] Nassau St & Navy St              South St & Gouverneur Ln
[313] Old Slip & Front St              E 55 St & Lexington Ave
[315] Grand St & Greene St             York St & Jay St
[317] Henry St & Poplar St             W 52 St & 5 Ave
```

```
[319] E 53 St & Madison Ave            Front St & Gold St
[321] Old Fulton St                    Sands St & Gold St
[323] 3 Ave & Schermerhorn St          Columbia Heights & Cranberry St
[325] Hanover Pl & Livingston St       7 Ave & Farragut St
[327] Railroad Ave & Kay Ave           Cadman Plaza E & Red Cross Pl
[329] Park Ave & St Edwards St
329 Levels: 1 Ave & E 15 St 1 Ave & E 18 St 1 Ave & E 30 St ... York St & Jay St
> nlevels(unique(citi$endstation))
[1] 329

> #b)
> tapply(citi$tripduration,citi$day,mean)
     Fri      Mon      Sat      Sun      Thu      Tue      Wed
832.3580 853.6248 894.2661 887.5528 865.7822 857.4895 843.5679

> #c)
> max(table(citi$starttime))
[1] 65684
> which.max(table(citi$starttime))
18
19
> min(table(citi$starttime))
[1] 1151
> which.min(table(citi$starttime))
4
5

> #d)
> table(citi$gender)

     1      2
510826 156912

> #e)
> citi$Mon <- as.integer(citi$day == "Mon")
> citi$Tue <- as.integer(citi$day == "Tue")
> citi$Wed <- as.integer(citi$day == "Wed")
> citi$Thu <- as.integer(citi$day == "Thu")
> citi$Fri <- as.integer(citi$day == "Fri")
> citi$Sat <- as.integer(citi$day == "Sat")
> citi$Sun <- as.integer(citi$day == "Sun")

> #f)
> summary(citi)
```

```
  tripduration                     startstation   
 Min.   :    60.0   E 17 St & Broadway   :  6310  
 1st Qu.:   410.0   Pershing Square N    :  6295  
 Median :   653.0   W 20 St & 11 Ave     :  6054  
 Mean   :   859.1   Lafayette St & E 8 St:  5881  
 3rd Qu.:  1055.0   8 Ave & W 31 St      :  5741  
 Max.   :730955.0   Broadway & E 14 St   :  5727  
                    (Other)              :631730  
                 endstation         gender            age           day        
 E 17 St & Broadway   :  6847   Min.   :1.000   Min.   :16.00   Fri: 81695  
 W 20 St & 11 Ave     :  6123   1st Qu.:1.000   1st Qu.:29.00   Mon:107187  
 Lafayette St & E 8 St:  5820   Median :1.000   Median :35.00   Sat: 68866  
 Broadway & E 14 St   :  5712   Mean   :1.235   Mean   :37.47   Sun: 69153  
 8 Ave & W 31 St      :  5656   3rd Qu.:1.000   3rd Qu.:44.00   Thu: 84892  
 Pershing Square N    :  5102   Max.   :2.000   Max.   :75.00   Tue:127394  
 (Other)              :632478                                   Wed:128551  
   starttime          Mon              Tue              Wed        
 Min.   : 0.00   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:10.00   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000  
 Median :15.00   Median :0.0000   Median :0.0000   Median :0.0000  
 Mean   :14.25   Mean   :0.1605   Mean   :0.1908   Mean   :0.1925  
 3rd Qu.:18.00   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000  
 Max.   :23.00   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  

      Thu              Fri              Sat              Sun        
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000  
 Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000  
 Mean   :0.1271   Mean   :0.1223   Mean   :0.1031   Mean   :0.1036  
 3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  

> #g)
> citi1 <- citi
> citi1$tripduration <- scale(citi1$tripduration)
> citi1$gender <- scale(citi1$gender)
> citi1$age <- scale(citi1$age)
> citi1$starttime <- scale(citi1$starttime)
> citi1$Mon <- scale(citi1$Mon)
> citi1$Tue <- scale(citi1$Tue)
> citi1$Wed <- scale(citi1$Wed)
> citi1$Thu <- scale(citi1$Thu)
> citi1$Fri <- scale(citi1$Fri)
> citi1$Sat <- scale(citi1$Sat)
> citi1$Sun <- scale(citi1$Sun)
> max(citi1$tripduration)
[1] 402.9514

> #i)
> set.seed(100)
> cluster1 <- kmeans(citi1[,c("tripduration","gender","age","starttime","Mon","Tue","Wed",
                     "Thu","Fri","Sat","Sun")],centers=10)
> min(cluster1$size)
[1] 18148
> max(cluster1$size)
[1] 107185

> #j)
> cluster1$centers
   tripduration      gender         age    starttime        Mon        Tue
1   0.072858922  1.80299766 -0.10540533 -0.040093878 -0.4372836 -0.4855556
2   0.020245389  0.18761389 -0.64091446  0.052809861 -0.4372836 -0.4855556
3   0.002615291 -0.01315714  0.02627901 -0.010131298 -0.4372836 -0.4855556
4  -0.031234691 -0.55423157  0.07826060  0.004153716 -0.4372836 -0.4855556
5   0.017598755 -0.13967955  1.11985924 -0.050591617 -0.4372836 -0.4855556
6  -0.039763879 -0.55423157  0.03803946 -0.104875348 -0.4372836 -0.4855556
7  -0.029791852 -0.55423157  0.07311514  0.004761192 -0.4372836  2.0594932
8   0.015717426  0.08515270 -0.12675059  0.026080401 -0.4372836 -0.4855556
9  -0.004070735 -0.02634887  0.02757155  0.013132061  2.2868421 -0.4855556
10  0.087047159  1.80380832 -0.09463120  0.047256550 -0.4371895  0.7792304
          Wed        Thu        Fri        Sat       Sun
1  -0.4882786 -0.3816420  2.6783460 -0.3391057 -0.339893
2  -0.4882786 -0.3816420 -0.3733642  2.9489285 -0.339893
3  -0.4882786  2.6202529 -0.3733642 -0.3391057 -0.339893
4   2.0480080 -0.3816420 -0.3733642 -0.3391057 -0.339893
5  -0.4882786 -0.3816420 -0.3733642  2.9489285 -0.339893
6  -0.4882786 -0.3816420  2.6783460 -0.3391057 -0.339893
7  -0.4882786 -0.3816420 -0.3733642 -0.3391057 -0.339893
8  -0.4882786 -0.3816420 -0.3733642 -0.3391057  2.942098
9  -0.4882786 -0.3816420 -0.3733642 -0.3391057 -0.339893
10  0.7874450 -0.3815901 -0.3733642 -0.3391057 -0.339893

> #n)
> citi1$weekday <- 1-as.integer(citi$Sat==1|citi$Sun==1)
> citi1$weekday <- scale(citi1$weekday)
> set.seed(100)
> cluster2 <- kmeans(citi1[,c("tripduration","gender","age","starttime","weekday")],centers=10)
> cluster2$centers
```

```
   tripduration     gender        age  starttime    weekday
1   0.409925533  1.7800715  1.1147478 -0.1593482  0.5041004
2   0.003522478  0.2187535 -0.6427132  0.1959095 -1.9590838
3  -0.047614897 -0.5542316  1.0398396 -1.0983165  0.5104419
4  -0.078466769 -0.5208387 -0.5914811 -1.4382010  0.3764170
5   0.009448121 -0.1418490  1.1043608 -0.0903787 -1.9590838
6  -0.015523192 -0.5542316  0.4787926  0.7174943  0.5104419
7  -0.018969283 -0.5542316 -0.7725312  0.9527464  0.5104419
8  -0.010347262 -0.5542316  1.8753805  0.3917602  0.5104419
9  -0.096648552 -0.5542316 -0.5627644 -0.1497968  0.5104419
10  0.019099791  1.8042973 -0.6737577  0.1652332  0.5104419
```

3. In this question, you will extend the collaborative filtering model from the class. In the class we developed three models for collaborative filtering—using average item rating, using average user rating and by taking the average of the k nearest users using the Pearson correlation metric. In this question, you will extend the last model by using a weighted average where the weights are the similarity metric defined by the Pearson correlation. Develop an R code to do this and verify the quality of the fit by changing the number of neighbors in the set $\{10, 50, 100, 150, 200, 250\}$. Compute the root mean squared error in each of these cases.

   Solution. Things to note:

   - We need to keep track of the correlation scores, unlike in the class notes where they are discarded/overwritten as soon as the order is known.
   - Whenever a relevant neighbour of user u, let's say user v, has a missing entry (user v did not rate a certain movie), we should not consider the correlation between u and v when computing the sum of weights (in the denominator):

$$p_{u,i} = \frac{\sum_{v \in N_u} S_{u,v} r_{v,i}}{\sum_{v \in N_u} S_{u,v}}$$

   - Correlation scores are between -1 and 1. This means that the weighted average can be smaller than 0 or greater than 5. Let us simply project these values onto the feasible output set, i.e. negative scores will be changed to 0, and scores greater than 5 will be changed to 5.
   - Use *apply* to calculate the column means, with the user-defined function *w.avg* that calculates the proper weighted average, and projects the predicted scores onto the feasible output set:

```
w.avg <-  function(x,y){
  # If y'all want to use weighted.mean instead, take note to include
  # the separate argument "na.rm=TRUE" in the apply() function call
  # z <- weighted.mean(x,y,na.rm=TRUE)

  z <- sum(x*y, na.rm=TRUE)/sum(y[which(!is.na(x))])
  if (!is.na(z)){
  if(z>5){z <- 5}
  if(z<0){z <- 0}
  }
  return(z)
}
```

   Let us now look at the code:

R Scripts.

```
## Pre-process your data as in Week 10 Class 2 until the initialisation of the matrices

Cor <- matrix(nrow=length(spl1c),ncol=length(spl1))
Order <- matrix(nrow=length(spl1c), ncol=length(spl1))
for(i in 1:length(spl1c)){
  for(j in 1:length(spl1)){
    Cor[i,j] <- cor(Data[spl1c[i],spl2],Data[spl1[j],spl2],use = "pairwise.complete.obs")
  }
  V <- order(Cor[i,], decreasing=TRUE, na.last=NA)
  Order[i,] <- c(V, rep(NA, times=length(spl1)- length(V)))
}

w.avg <-  function(x,y){
  # If y'all want to use weighted.mean instead, take note to include
  # the separate argument "na.rm=TRUE" in the apply() function call
  # z <- weighted.mean(x,y,na.rm=TRUE)

  z <- sum(x*y, na.rm=TRUE)/sum(y[which(!is.na(x))])
  if (!is.na(z)){
  if(z>5){z <- 5}
  if(z<0){z <- 0}
  }
  return(z)
}

numk <- c(10,50,100,150,200,250)
RMSE <- rep(NA, times=length(numk))

for(ind in 1:length(numk)){
  k <- numk[ind]
  UserPred <- matrix(nrow=length(spl1c), ncol=length(spl2c))

  for(i in 1:length(spl1c)){
    w <- Cor[i,Order[i,1:k]]
    D <- Data[spl1[Order[i,1:k]],spl2c]
    UserPred[i,] <- apply(D,2,w.avg,y=w)
  }
  RMSE[ind] <- sqrt(mean((Data[spl1c,spl2c] - UserPred)^2,na.rm=TRUE))
}

RMSE
```

```
# 1.2085385 1.1566302 1.0403980 0.9834489 0.9231894 0.9003782
plot(numk,RMSE)
```