A4_DSC424_ateboul

Alex Teboul

November 5, 2019

##DSC 324/424 ##Assignment 4 (DUE SUNDAY, November 10th by Midnight)

Problem 2

2) (10 points, individual submission): Select one of the techniques covered in lectures 7 and 8 (i.e. LDA, CA, Cluster Analysis … any type) and apply it to some aspect of your data. Or research a new technique that we have not covered and apply it). Each team member should investigate a different aspect of the data.

```
library(MASS)
library(psych)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:psych':
##
##     outlier
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

```
library(AUC)
```

```
## AUC 0.3.0
```

```
## Type AUCNews() to see the change log and ?AUC to get an overview.
```

```
##
## Attaching package: 'AUC'
```

```
## The following objects are masked from 'package:caret':
##
##     sensitivity, specificity
```

```
library(caret)
library(readxl)
```

```
#Read in Datasets
housing_data_pre_lda<- read_xls("alex.xls")

#Remove columns
housing_data_lda <- subset(housing_data_pre_lda, select = -c(order, pid,miscval, saleprice, totalbsmtsf, grlivarea, garageco
nd))

#Get Numberic
nums_lda <- unlist(lapply(housing_data_lda, is.numeric))
housing_data_nums_lda<- housing_data_lda[ , nums_lda]

#Check Sample Size and Number of Variables
dim(housing_data_nums_lda)
```

```
## [1] 2930   54
```

```
#Missing values
sum(is.na(housing_data_nums_lda))
```

```
## [1] 0
```

```
#Calculate Missing Values by Attribute
colSums(is.na(housing_data_nums_lda))
```

```
##    mssubclass   lotfrontage       lotarea   overallqual   overallcond 
##             0             0             0             0             0 
##     yearbuilt  yearremodadd    masvnrarea    bsmtfinsf1    bsmtfinsf2 
##             0             0             0             0             0 
##     bsmtunfsf     x1stflrsf     x2ndflrsf  lowqualfinsf  bsmtfullbath 
##             0             0             0             0             0 
##  bsmthalfbath      fullbath      halfbath  bedroomabvgr  kitchenabvgr 
##             0             0             0             0             0 
##  totrmsabvgrd    fireplaces    garagecars    garagearea    wooddecksf 
##             0             0             0             0             0 
##   openporchsf enclosedporch    x3ssnporch   screenporch      poolarea 
##             0             0             0             0             0 
##        mosold        yrsold         alley      lotshape     exterqual 
##             0             0             0             0             0 
##     extercond      bsmtqual      bsmtcond  bsmtexposure  bsmtfintype1 
##             0             0             0             0             0 
##  bsmtfintype2     heatingqc   kitchenqual    functional   fireplacequ 
##             0             0             0             0             0 
##      garagequ        poolqc         fence    centralair        yr2006 
##             0             0             0             0             0 
##        yr2007        yr2008        yr2009        yr2010 
##             0             0             0             0
```

```
#get the pricelevel back in
housing_data_nums_lda$pricelevel <-housing_data_lda$pricelevel

#lda 1
lda_housing <- lda(pricelevel~., data=housing_data_nums_lda)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
lda_housing
```

```
## Call:
## lda(pricelevel ~ ., data = housing_data_nums_lda)
##
## Prior probabilities of groups:
##        high         low middle high  middle low
##   0.2494881   0.2494881   0.2498294   0.2511945
##
## Group means:
##             mssubclass lotfrontage   lotarea overallqual overallcond
## high          50.45828    79.23940 13338.137    7.674419    5.339261
## low           59.24077    60.26402  7839.302    4.790698    5.566347
## middle high   61.85109    69.14071 10000.007    6.420765    5.443989
## middle low    57.98913    67.44565  9419.427    5.497283    5.900815
##             yearbuilt yearremodadd masvnrarea bsmtfinsf1 bsmtfinsf2
## high         1994.363     1999.741  221.26539   700.6156   45.44460
## low          1944.878     1967.536   32.10534   241.1628   38.99179
## middle high  1984.178     1991.567   91.81557   404.5232   51.57104
## middle low   1962.053     1978.253   59.49457   423.7921   62.72283
##             bsmtunfsf x1stflrsf x2ndflrsf lowqualfinsf bsmtfullbath
## high         687.4391 1501.6895  501.1532     1.318741    0.6306430
## low          464.6156  903.2544  203.1874     8.621067    0.2476060
## middle high  609.3866 1158.1066  406.3128     3.424863    0.3934426
## middle low   475.3492 1075.7554  231.7826     5.339674    0.4524457
##             bsmthalfbath fullbath  halfbath bedroomabvgr kitchenabvgr
## high          0.03967168 1.989056 0.5499316     3.006840     1.008208
## low           0.05198358 1.155951 0.1545828     2.618331     1.083447
## middle high   0.07103825 1.821038 0.5204918     2.890710     1.019126
## middle low    0.08152174 1.301630 0.2934783     2.900815     1.066576
##             totrmsabvgrd fireplaces garagecars garagearea wooddecksf
## high            7.521204  1.0424077   2.452804   667.5746  149.52531
## low             5.601915  0.1997264   1.101231   301.9644   42.14090
## middle high     6.579235  0.7240437   1.964481   497.9358  106.73361
## middle low      6.072011  0.4320652   1.547554   423.4606   76.70652
##             openporchsf enclosedporch x3ssnporch screenporch poolarea
## high           80.16826      12.78796  4.0109439   21.473324 5.389877
## low            23.13406      36.19973  0.5280438    6.737346 0.000000
## middle high    54.75000      16.23497  2.7759563   19.760929 1.672131
## middle low     32.17663      26.80707  3.0516304   16.031250 1.914402
##               mosold   yrsold      alley lotshape exterqual extercond
## high        6.421341 2007.751 0.03009576 2.370725  3.997264  3.088919
## low         6.121751 2007.851 0.11901505 3.570451  2.990424  3.056088
## middle high 6.170765 2007.794 0.12704918 2.591530  3.486339  3.084699
## middle low  6.150815 2007.766 0.10054348 3.226902  3.122283  3.111413
##             bsmtqual bsmtcond bsmtexposure bsmtfintype1 bsmtfintype2
## high        4.222982 3.064295     2.212038     4.615595     1.348837
## low         3.173735 3.082079     1.496580     3.240766     1.797538
## middle high 3.740437 3.046448     1.704918     3.898907     1.463115
## middle low  3.326087 3.044837     1.546196     3.936141     1.639946
##             heatingqc kitchenqual functional fireplacequ garagequ
## high         4.783858    4.158687   6.948016   3.2079343 3.010944
## low          3.537620    3.039672   6.714090   0.5854993 2.381669
## middle high  4.370219    3.595628   6.894809   2.0942623 2.978142
## middle low   3.907609    3.252717   6.819293   1.1970109 2.835598
##                  poolqc     fence centralair    yr2006    yr2007    yr2008
## high        0.034199726 0.2845417  0.9958960 0.2188782 0.2407661 0.2147743
## low         0.000000000 1.0451436  0.7852257 0.2079343 0.2229822 0.2134063
## middle high 0.005464481 0.5464481  0.9904372 0.1980874 0.2540984 0.2144809
## middle low  0.009510870 1.2445652  0.9605978 0.2282609 0.2296196 0.2065217
##                yr2009    yr2010
## high        0.2216142 0.1039672
## low         0.2216142 0.1340629
## middle high 0.2226776 0.1106557
## middle low  0.2187500 0.1168478
##
## Coefficients of linear discriminants:
##                        LD1           LD2           LD3
## mssubclass     3.758993e-03  5.969745e-03  3.561336e-03
## lotfrontage   -2.590110e-03  1.041676e-02  4.164089e-03
## lotarea       -1.433195e-05 -9.781648e-06  8.588646e-06
## overallqual   -2.707421e-01 -9.853353e-02  8.335712e-02
## overallcond   -1.413598e-01  1.651743e-01  3.722822e-01
## yearbuilt     -1.763171e-02  2.029830e-02  7.849511e-03
## yearremodadd  -6.714298e-03  1.169089e-02  2.483927e-03
## masvnrarea     5.585540e-04 -1.381401e-03  1.855713e-05
## bsmtfinsf1     1.124201e-04 -4.196365e-04  2.819661e-04
## bsmtfinsf2    -1.889271e-04  2.249575e-04  6.051748e-04
## bsmtunfsf     -1.895130e-04  1.549299e-05  1.880099e-04
## x1stflrsf     -1.327940e-03 -5.696678e-04  7.748907e-04
## x2ndflrsf     -1.111766e-03 -1.396550e-03  3.177216e-04
## lowqualfinsf  -2.222472e-04  8.312234e-05 -1.431425e-04
## bsmtfullbath  -3.249781e-01  1.314037e-01  3.536010e-01
## bsmthalfbath  -1.440861e-01  4.564997e-01  1.252369e-01
## fullbath      -6.856102e-01  9.499326e-01 -1.505915e+00
## halfbath      -1.953491e-01  7.441659e-01 -3.663002e-01
## bedroomabvgr   6.564083e-02  1.041274e-01  6.469788e-01
## kitchenabvgr   7.756700e-01 -3.998930e-01  1.154659e+00
## totrmsabvgrd   1.893256e-02  5.229589e-02 -9.344506e-02
## fireplaces    -3.156379e-01  3.414928e-01 -4.049557e-01
## garagecars    -4.524641e-01  3.169834e-01 -2.508981e-01
## garagearea     1.960391e-04 -2.130253e-03  1.846670e-03
## wooddecksf    -3.719431e-04  3.340968e-04 -2.937824e-04
## openporchsf   -3.721370e-04 -3.024352e-04 -8.659353e-04
## enclosedporch -1.542482e-03  1.634615e-03  3.021148e-04
## x3ssnporch    -1.390880e-03  1.478960e-03  2.115085e-03
## screenporch   -1.985122e-03  2.637657e-03 -1.017069e-04
## poolarea      -2.010717e-03 -8.714909e-04 -9.600560e-04
## mosold        -2.523867e-03 -2.326052e-02 -3.374699e-03
## yrsold         6.276466e-03 -1.251888e-02 -3.100600e-02
## alley         -6.258478e-02  6.698640e-01 -1.600653e-03
## lotshape       8.497087e-02 -8.072169e-02  8.741618e-02
## exterqual     -1.652161e-01 -7.427846e-01 -2.774110e-01
## extercond      2.550590e-02 -4.020300e-02 -1.543744e-01
## bsmtqual       6.287087e-02 -4.185407e-01  1.405082e-01
## bsmtcond       1.043191e-01  6.114229e-02 -2.376376e-02
## bsmtexposure  -4.155776e-02 -5.311216e-02 -1.025642e-01
## bsmtfintype1  -7.325918e-02  1.536377e-02  6.349401e-02
## bsmtfintype2   2.257878e-02 -5.154931e-02 -7.676922e-02
## heatingqc     -1.579793e-01  3.065762e-02  4.970290e-02
## kitchenqual   -8.373626e-02 -4.055242e-01 -6.073768e-03
## functional    -1.771099e-01 -3.080019e-02 -1.426769e-01
## fireplacequ   -8.760915e-02 -7.356126e-02  9.099000e-02
## garagequ       7.779145e-02  5.630612e-01  1.768317e-01
## poolqc         6.703732e-01  4.649366e-02  6.830497e-02
## fence          3.700017e-02  1.332586e-02  1.299692e-01
## centralair     4.084033e-01  7.060096e-01  8.299580e-01
## yr2006        -3.776268e-02 -2.565318e-02  1.950961e-01
## yr2007         4.750022e-02  3.461803e-02 -1.066113e-01
## yr2008        -7.467601e-02  3.324595e-02 -6.868558e-02
## yr2009         7.951316e-02  5.228366e-02  2.071034e-02
## yr2010        -3.365356e-02 -1.607963e-01 -5.393892e-02
## 
## Proportion of trace:
##    LD1    LD2    LD3 
## 0.9152 0.0567 0.0281
```

```
#predict values
p_housing = predict(lda_housing, housing_data_nums_lda)$class
#p_housing
```

```
# Compare the results of the prediction
lda_housing_table <- table(p_housing, housing_data_nums_lda$pricelevel)
confusionMatrix(lda_housing_table)
```

```
## Confusion Matrix and Statistics
##
##
## p_housing     high low middle high middle low
##   high         590   0          38          4
##   low            0 545           1        108
##   middle high  133   9         570        105
##   middle low     8 177         123        519
##
## Overall Statistics
##
##                Accuracy : 0.759
##                  95% CI : (0.7431, 0.7744)
##     No Information Rate : 0.2512
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6787
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: high Class: low Class: middle high
## Sensitivity               0.8071     0.7456             0.7787
## Specificity               0.9809     0.9504             0.8876
## Pos Pred Value            0.9335     0.8333             0.6977
## Neg Pred Value            0.9386     0.9183             0.9233
## Prevalence                0.2495     0.2495             0.2498
## Detection Rate            0.2014     0.1860             0.1945
## Detection Prevalence      0.2157     0.2232             0.2788
## Balanced Accuracy         0.8940     0.8480             0.8332
##                      Class: middle low
## Sensitivity                     0.7052
## Specificity                     0.8596
## Pos Pred Value                  0.6276
## Neg Pred Value                  0.8968
## Prevalence                      0.2512
## Detection Rate                  0.1771
## Detection Prevalence            0.2823
## Balanced Accuracy               0.7824
```

- We added a columns to the dataset to categorize sale prices into 4 groups: low(<$129,499), low middle($129,500 - $160,000), high middle($160,001 - $213,500), and high(>$213,501). These cutoffs were generated by the saleprice quartiles such that each grouping had 25% of the data.
- From the results of the LDA, we see that classification to the different categories (on the training set itself) is 76%. Next we try a random train/test split and compare.
- Actually I will use Random Forest as another classification technique to and look at accuracy.

```
# Random sample indexes
train_index <- sample(1:nrow(housing_data_nums_lda), 0.7 * nrow(housing_data_nums_lda))
test_index <- setdiff(1:nrow(housing_data_nums_lda), train_index)

# Build X_train, y_train, X_test, y_test
X_train <- housing_data_nums_lda[train_index, -15]
y_train <- housing_data_nums_lda[train_index, "pricelevel"]
house_train <-X_train
house_train$pricelevel <-y_train$pricelevel

X_test <- housing_data_nums_lda[test_index, -15]
y_test <- housing_data_nums_lda[test_index, "pricelevel"]
house_test <- X_test
house_test$pricelevel <- y_test$pricelevel

dim(X_train)
```

```
## [1] 2051   54
```

```
dim(y_train)
```

```
## [1] 2051    1
```

```
dim(X_test)
```

```
## [1] 879  54
```

```
dim(y_test)
```

```
## [1] 879   1
```

```
dim(house_train) #same as x_train actually
```

```
## [1] 2051   54
```

```
dim(house_test) #same as x_test actually
```

```
## [1] 879  54
```

```
X_test_nopricelevel <- subset(X_test, select = -c(pricelevel))
dim(X_test_nopricelevel)
```

```
## [1] 879  53
```

```
str(house_train)
```

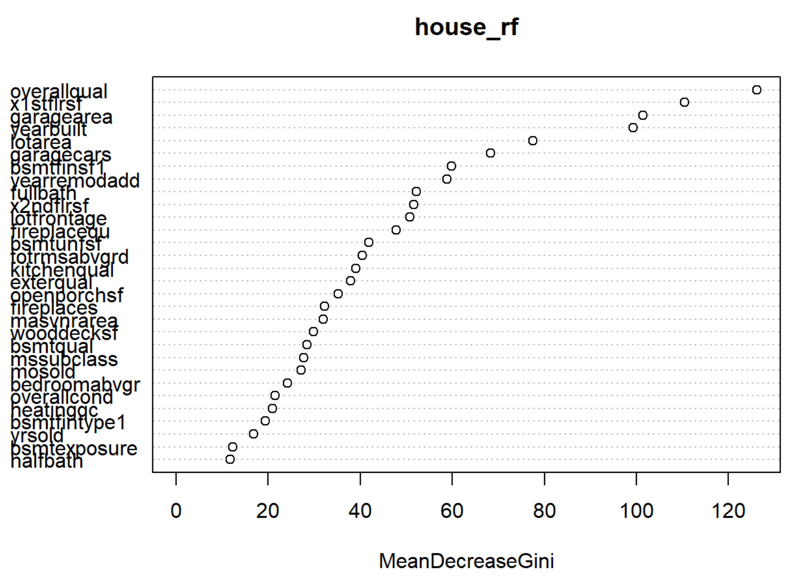
```
## Classes 'tbl_df', 'tbl' and 'data.frame':    2051 obs. of  54 variables:
##  $ mssubclass   : num  20 20 60 80 20 50 80 60 60 30 ...
##  $ lotfrontage  : num  75 95 68 68 60 75 68 65 74 50 ...
##  $ lotarea      : num  9464 13618 9240 12095 6600 ...
##  $ overallqual  : num  6 8 8 6 5 5 6 7 7 6 ...
##  $ overallcond  : num  7 5 5 6 6 5 5 5 5 8 ...
##  $ yearbuilt    : num  1958 2005 2001 1964 1982 ...
##  $ yearremodadd : num  1958 2006 2002 1964 2005 ...
##  $ masvnrarea   : num  135 198 396 115 0 0 0 0 0 0 ...
##  $ bsmtfinsf1   : num  570 1350 0 564 638 420 939 0 0 299 ...
##  $ bsmtfinsf2   : num  0 0 0 0 0 0 0 0 0 40 ...
##  $ bsmtunfsf    : num  510 378 1055 563 207 ...
##  $ x1stflrsf    : num  1080 1960 1055 1445 845 ...
##  $ x2ndflrsf    : num  0 0 1208 0 0 ...
##  $ lowqualfinsf : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ bsmthalfbath : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ fullbath     : num  1 2 2 1 1 1 1 2 2 1 ...
##  $ halfbath     : num  0 0 1 1 0 1 0 1 1 0 ...
##  $ bedroomabvgr : num  3 3 3 3 3 4 3 4 3 2 ...
##  $ kitchenabvgr : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ totrmsabvgrd : num  5 8 7 7 6 6 6 8 7 5 ...
##  $ fireplaces   : num  0 2 1 1 0 0 2 1 0 0 ...
##  $ garagecars   : num  1 3 2 2 1 1 2 2 2 1 ...
##  $ garagearea   : num  288 714 905 645 264 282 441 434 578 195 ...
##  $ wooddecksf   : num  0 172 0 180 0 289 0 100 0 0 ...
##  $ openporchsf  : num  0 38 45 0 0 0 69 48 105 0 ...
##  $ enclosedporch: num  0 0 0 0 0 0 0 0 0 116 ...
##  $ x3ssnporch   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ screenporch  : num  130 0 189 0 0 0 0 0 0 0 ...
##  $ poolarea     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ mosold       : num  6 11 9 8 7 6 11 7 8 7 ...
##  $ yrsold       : num  2008 2006 2008 2009 2008 ...
##  $ alley        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ lotshape     : num  4 4 4 1 4 4 1 4 1 4 ...
##  $ exterqual    : num  3 4 4 3 3 3 3 4 4 3 ...
##  $ extercond    : num  4 3 3 4 3 3 3 3 3 4 ...
##  $ bsmtqual     : num  3 5 4 3 3 3 3 4 4 3 ...
##  $ bsmtcond     : num  3 4 3 3 3 3 3 3 4 3 ...
##  $ bsmtexposure : num  1 3 1 4 1 1 1 1 2 1 ...
##  $ bsmtfintype1 : num  4 6 1 3 6 4 5 1 1 3 ...
##  $ bsmtfintype2 : num  1 1 1 1 1 1 1 1 1 6 ...
##  $ heatingqc    : num  4 5 5 3 4 3 3 5 5 3 ...
##  $ kitchenqual  : num  3 4 4 3 4 3 3 4 4 3 ...
##  $ functional   : num  7 7 7 7 7 7 7 7 7 7 ...
##  $ fireplacequ  : num  0 4 3 2 0 0 4 4 0 0 ...
##  $ garagequ     : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ poolqc       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ fence        : num  0 0 0 4 0 4 0 0 0 4 ...
##  $ centralair   : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ yr2006       : num  0 1 0 0 0 0 1 0 1 0 ...
##  $ yr2007       : num  0 0 0 0 0 1 0 1 0 0 ...
##  $ yr2008       : num  1 0 1 0 1 0 0 0 0 1 ...
##  $ yr2009       : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ yr2010       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ pricelevel   : chr  "middle low" "high" "high" "middle low" ...
```

```
#### Converting into factor
house_train$pricelevel <- as.factor(house_train$pricelevel)
str(house_train)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    2051 obs. of  54 variables:
##  $ mssubclass   : num  20 20 60 80 20 50 80 60 60 30 ...
##  $ lotfrontage  : num  75 95 68 68 60 75 68 65 74 50 ...
##  $ lotarea      : num  9464 13618 9240 12095 6600 ...
##  $ overallqual  : num  6 8 8 6 5 5 6 7 7 6 ...
##  $ overallcond  : num  7 5 5 6 6 5 5 5 5 8 ...
##  $ yearbuilt    : num  1958 2005 2001 1964 1982 ...
##  $ yearremodadd : num  1958 2006 2002 1964 2005 ...
##  $ masvnrarea   : num  135 198 396 115 0 0 0 0 0 0 ...
##  $ bsmtfinsf1   : num  570 1350 0 564 638 420 939 0 0 299 ...
##  $ bsmtfinsf2   : num  0 0 0 0 0 0 0 0 0 40 ...
##  $ bsmtunfsf    : num  510 378 1055 563 207 ...
##  $ x1stflrsf    : num  1080 1960 1055 1445 845 ...
##  $ x2ndflrsf    : num  0 0 1208 0 0 ...
##  $ lowqualfinsf : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ bsmthalfbath : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ fullbath     : num  1 2 2 1 1 1 1 2 2 1 ...
##  $ halfbath     : num  0 0 1 1 0 1 0 1 1 0 ...
##  $ bedroomabvgr : num  3 3 3 3 3 4 3 4 3 2 ...
##  $ kitchenabvgr : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ totrmsabvgrd : num  5 8 7 7 6 6 6 8 7 5 ...
##  $ fireplaces   : num  0 2 1 1 0 0 2 1 0 0 ...
##  $ garagecars   : num  1 3 2 2 1 1 2 2 2 1 ...
##  $ garagearea   : num  288 714 905 645 264 282 441 434 578 195 ...
##  $ wooddecksf   : num  0 172 0 180 0 289 0 100 0 0 ...
##  $ openporchsf  : num  0 38 45 0 0 0 69 48 105 0 ...
##  $ enclosedporch: num  0 0 0 0 0 0 0 0 0 116 ...
##  $ x3ssnporch   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ screenporch  : num  130 0 189 0 0 0 0 0 0 0 ...
##  $ poolarea     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ mosold       : num  6 11 9 8 7 6 11 7 8 7 ...
##  $ yrsold       : num  2008 2006 2008 2009 2008 ...
##  $ alley        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ lotshape     : num  4 4 4 1 4 4 1 4 1 4 ...
##  $ exterqual    : num  3 4 4 3 3 3 3 4 4 3 ...
##  $ extercond    : num  4 3 3 4 3 3 3 3 3 4 ...
##  $ bsmtqual     : num  3 5 4 3 3 3 3 4 4 3 ...
##  $ bsmtcond     : num  3 4 3 3 3 3 3 3 4 3 ...
##  $ bsmtexposure : num  1 3 1 4 1 1 1 1 2 1 ...
##  $ bsmtfintype1 : num  4 6 1 3 6 4 5 1 1 3 ...
##  $ bsmtfintype2 : num  1 1 1 1 1 1 1 1 1 6 ...
##  $ heatingqc    : num  4 5 5 3 4 3 3 5 5 3 ...
##  $ kitchenqual  : num  3 4 4 3 4 3 3 4 4 3 ...
##  $ functional   : num  7 7 7 7 7 7 7 7 7 7 ...
##  $ fireplacequ  : num  0 4 3 2 0 0 4 4 0 0 ...
##  $ garagequ     : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ poolqc       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ fence        : num  0 0 0 4 0 4 0 0 0 4 ...
##  $ centralair   : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ yr2006       : num  0 1 0 0 0 0 1 0 1 0 ...
##  $ yr2007       : num  0 0 0 0 0 1 0 1 0 0 ...
##  $ yr2008       : num  1 0 1 0 1 0 0 0 0 1 ...
##  $ yr2009       : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ yr2010       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ pricelevel   : Factor w/ 4 levels "high","low","middle high",..: 4 1 1 4 4 4 3 1 3 2 ...
```

```
library(MASS)
require(randomForest)

house_rf <- randomForest(pricelevel~., data = house_train)
importance <- importance(house_rf)
varImpPlot(house_rf)
```



- Important variables for the splitting make a lot of sense:overall quality, 1st floor square footage, yearbuilt, lotarea, yearremodeled are the top important variables and these make sense. Differences in these make sense to impact price overall in these price levels.

```
print(house_rf)
```

```
##
## Call:
##  randomForest(formula = pricelevel ~ ., data = house_train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 22.14%
## Confusion matrix:
##             high low middle high middle low class.error
## high         424   1          71          7   0.1570577
## low            0 429           1         83   0.1637427
## middle high   43   4         384         90   0.2629559
## middle low     4  90          60        360   0.2996109
```

- We've got an estimated error rate of 23% which isn't great, but it's not horrible either. All things considered. ALso looking at the confusion matrix we see that most of the errors are between neighboring classes. It's never mistaking a low priced home with a high priced one or vice versa.

```
housing_prediction <- predict(house_rf, X_test_nopricelevel)
housing_conf_table <- table(housing_prediction, X_test$pricelevel)
confusionMatrix(housing_conf_table)
```

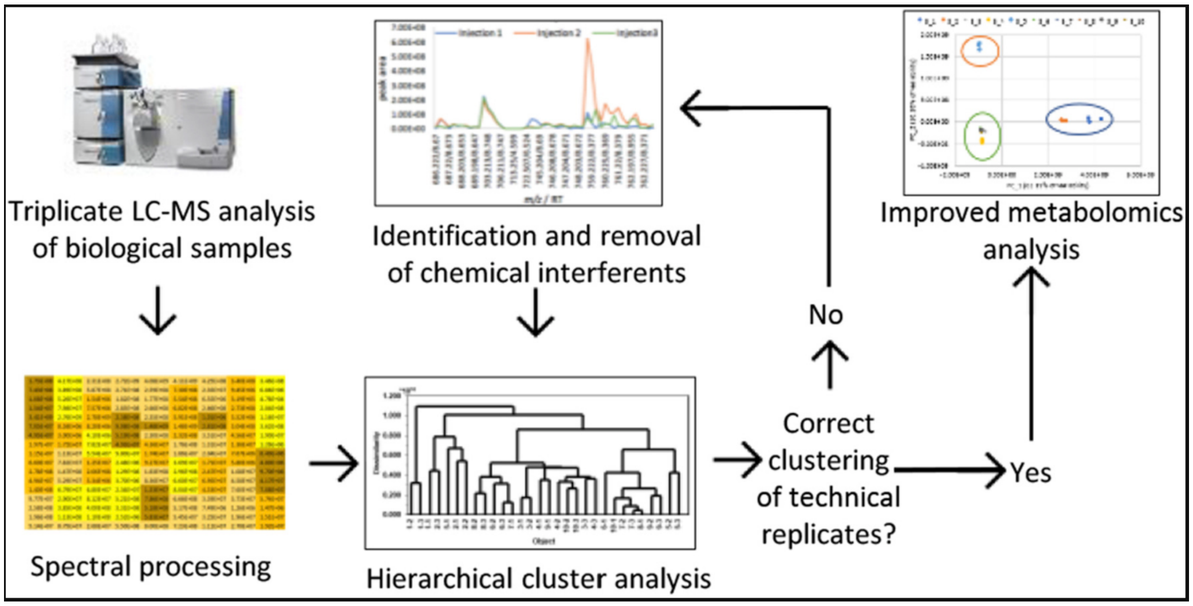
```
## Confusion Matrix and Statistics
##
##
## housing_prediction high low middle high middle low
##        high         194   0          16          0
##        low            0 172           0         52
##        middle high   34   3         156         26
##        middle low     0  43          39        144
##
## Overall Statistics
##
##                Accuracy : 0.7577
##                  95% CI : (0.7279, 0.7857)
##     No Information Rate : 0.2594
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6769
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: high Class: low Class: middle high
## Sensitivity               0.8509     0.7890             0.7393
## Specificity               0.9754     0.9213             0.9057
## Pos Pred Value            0.9238     0.7679             0.7123
## Neg Pred Value            0.9492     0.9298             0.9167
## Prevalence                0.2594     0.2480             0.2400
## Detection Rate            0.2207     0.1957             0.1775
## Detection Prevalence      0.2389     0.2548             0.2491
## Balanced Accuracy         0.9131     0.8552             0.8225
##                      Class: middle low
## Sensitivity                     0.6486
## Specificity                     0.8752
## Pos Pred Value                  0.6372
## Neg Pred Value                  0.8806
## Prevalence                      0.2526
## Detection Rate                  0.1638
## Detection Prevalence            0.2571
## Balanced Accuracy               0.7619
```

- The random forest model performed quite well on the test set! An accuracy of 92.7% and pretty good sensitivity/specificity on the difference price level classes as well. Best method so far for classification into groups. Does this tell you what price to sell for? No, but it gives a quick snapshot of the price level based on features: yes.

Problem 3

3) (10 points-Cluster Analysis): Using Google Scholar, locate a journal article, which uses cluster analysis in your field of interest. Write a summary of the journal article and how it utilizes the cluster analysis in two to three paragraphs. Cite the paper in APA format.

- Authors L.K Caesar, O.M. Kvalheim, and B.C. Cech use cluster analysis towards the identification of chemical interferents to help with metabolomics modeling. The stated goal of the study was to identify and remove chemical contaminants from mass spectral data sets using heirarchical cluster analysis. Basically, they use hierarchical cluster analysis in their workflow of chemical analysis when performing mass spectrometry-based studies. The cluster analysis is supposed to have clustered appropriately, but if clusters are not as expected, they know there are chemical interferents that must be filtered out before further analysis of metabolites is performed. This is not only a time saver but leads to improved models of how chemical samples are composed and may ultimately interact with one another.
- Metabolomics in general is a field that seeks to analyze and compare the quantities of metabolites, or small molecules, in biological samples. It has applications in drug interactions, diet, disease pathogenesis, and drug discovery. Because relatively few samples are collected in metabolomics experiments relative to the number of variables or different metabolites present, overfitting and other errors are common. Filtering contaminants is one way to improve models significantly, and it turns out that statistical analysis can help in this process, with cluster analysis being particularly useful when integrated into the process. They conclude that HCA is in fact a useful technique to help identify and remove certain contaminants/interferents in metabolomics applications and research.


Triplicate LC-MS analysis of biological samples
Identification and removal of chemical interferents
Improved metabolomics analysis
Spectral processing
Hierarchical cluster analysis
Correct clustering of technical replicates?
No
Yes
Injection 1
Injection 2
Injection3
peak area
m/z / RT
Dissimilarity
Object
problem3_image

Citation:

Caeser, L. K., Kvalheim, O. M., & Cech, B.C. (2019). Hierarchical cluster analysis of technical replicates to identify intereferents in untargeted mass spectrometry metabolomics.
Analytica Chimica Acta, *Volume 1021*, 69-77.
https://doi.org/10.1016/j.aca.2018.03.013 (https://doi.org/10.1016/j.aca.2018.03.013)

Problem 4

4) (Paper Review - 10 points-PCA and Linear Discriminant Analysis (LDA): An academic paper from a conference or Journal will be posted to the Homework 4 content section of D2L. Review the paper and evaluate their usage of PCA and LDA. In particular address the following: (See article on Face Recognition using Principle Component Analysis and Linear Discriminant Analysis)

• What is the application of this paper?

- PCA/LDA - Facial Recognition
- In this paper, the authors applied PCA and LDA to the task of facial recognition. Both techniques are linear algorithms that can be used for dimensionality reduction and the authors were building off of research that showed facial recognition accuracy and efficiency improvements when PCA or LDA were applied. They used data from the UMIST and ORL datasets, which contain headshots. In their results, they share accuracies for PCA vs LDA on each of the two datasets given different numbers of training images.

• What is the research question the authors wish to answer in this paper?

- The primary research question the authors wished to answer was - how do PCA and LDA compare in terms of accuracy at correctly classfying faces (eg performing facial recognition).

• How does this paper utilize PCA and LDA?

- The paper utilizes PCA and LDA by performing each on the UMIST and ORL face image datasets and comparing face classification accuracy between the two methods. They also run their experiment for different numbers of training images. As expected, and an important caveat to this technique is that the accuracy goes down with increasing numbers of samples, as it is harder for these methods to distinguish between the faces. The datasets used also differed in face presentation (how the faces were oriented), such that a comparison of PCA and LDA on different types of facial recognition would also be assessed.

• What are the results and conclusions from this paper?__

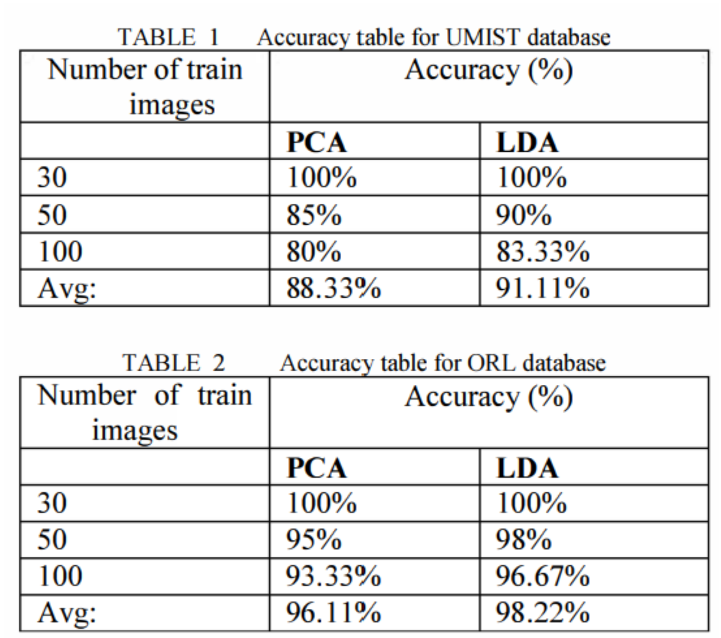


TABLE 1 Accuracy table for UMIST database

Number of train images	Accuracy (%)	
	PCA	**LDA**
30	100%	100%
50	85%	90%
100	80%	83.33%
Avg:	88.33%	91.11%

TABLE 2 Accuracy table for ORL database

Number of train images	Accuracy (%)	
	PCA	**LDA**
30	100%	100%
50	95%	98%
100	93.33%	96.67%
Avg:	96.11%	98.22%

problem4_image

- The authors conclude that for this task, there experiment shows LDA having a higher accuracy that PCA. They note that as the number of samples increases, both methods suffer in terms of accuracy, with PCA dropping in accuracy more so than LDA. They also conclude by noting that PCA has some advantages in terms of dimensionality reduction and ease of comparison between images (test against learned). LDA can also suffer when the dimensions are large relative to the number of samples. Ultimately they conclude that PCA and LDA are relevant techniques to use toward facial recognition.

Problem 5

5) (20 points): A common application of Discriminant Analysis is the classification of bonds into various bond rating classes. These ratings are intended to reflect the risk of the bond and influence the cost of borrowing for companies that issue bonds. Various financial ratios culled from annual reports are often used to help determine a company's bond rating. The Excel spreadsheet BondRating.xls (XLS) contains two sheets named Training data and Validation data. These are data from a sample of 95 companies selected from COMPUSTAT financial data tapes. The company bonds have been classified by Moody's Bond Ratings (1980) into seven classes of risk ranging from AAA, the safest, to C, the most risky. The data include ten financial variables for each company. These are:

-LOPMAR: Logarithm of the operating margin,

-LFIXMAR: Logarithm of the pretax fixed charge coverage,

-LTDCAP: Long-term debt to capitalization,

-LGERRAT: Logarithm of total long-term debt to total equity,

-LLEVER: Logarithm of the leverage,

-LCASHLTD: Logarithm of the cash flow to long-term debt,

-LACIDRAT: Logarithm of the acid test ratio,

-LCURRAT: Logarithm of the current assets to current liabilities,

-LRECTURN: Logarithm of the receivable turnover,

-LASSLTD: Logarithm of the net tangible assets to long-term debt.

The data are divided into 81 observations in the Training data sheet and 14 observations in the Validation data sheet. The bond ratings have been coded into numbers in the column with the title CODERTG, with AAA coded as 1, AA as 2, etc. Develop a Linear Discriminant Analysis model to classify the bonds in the Validation data sheet.

```r
library(MASS)
library(psych)
library(randomForest)
library(caret)
library(AUC)
library(caret)
```

```r
#Read in  Training Dataset
bond_training_all <- read_xls("BondRating.xls", skip=2, sheet="training")

#Remove columns
bond_training <- subset(bond_training_all, select = -c(OBS,CODERTG))

#Check Sample Size and Number of Variables
dim(bond_training)
```

```
## [1] 81 11
```

```
#Missing values
sum(is.na(bond_training))
```

```
## [1] 0
```

```
colSums(is.na(bond_training))
```

```
##   RATING   LOPMAR LFIXCHAR LGEARRAT   LTDCAP   LLEVER LCASHLTD LACIDRAT
##        0        0        0        0        0        0        0        0
##  LCURRAT LRECTURN  LASSLTD
##        0        0        0
```

```
head(bond_training)
```

```
## # A tibble: 6 x 11
##   RATING LOPMAR LFIXCHAR LGEARRAT LTDCAP LLEVER LCASHLTD LACIDRAT LCURRAT
##   <chr>   <dbl>    <dbl>    <dbl>  <dbl>  <dbl>    <dbl>    <dbl>   <dbl>
## 1 AAA     -1.66    0.749   -0.491  0.378  0.16    -1.23     0.433   1.12
## 2 AAA     -2.38    0.814    0.147  0.534  1.19    -1.55    -1.01    0.553
## 3 AAA     -1.40    2.56    -1.80   0.142 -0.531    0.496    0.314   1.01
## 4 AAA     -2.04    2.51    -1.53   0.178 -0.325    0.019    0.149   0.773
## 5 AAA     -1.36    2.43    -1.12   0.246 -0.085   -0.083    0.033   0.344
## 6 AAA     -1.69    2.89    -1.64   0.162  0.025    0.183   -0.051   0.328
## # ... with 2 more variables: LRECTURN <dbl>, LASSLTD <dbl>
```

```
#Read in Validation Dataset
bond_validation_all <- read_xls("BondRating.xls", skip=2, sheet="validation")

#Remove columns
bond_validation <- subset(bond_validation_all, select = -c(OBS,CODERTG))

#Check Sample Size and Number of Variables
dim(bond_validation)
```

```
## [1] 14 11
```

```
#Missing values
sum(is.na(bond_validation))
```

```
## [1] 0
```

```
colSums(is.na(bond_validation))
```

```
##   RATING   LOPMAR LFIXCHAR LGEARRAT   LTDCAP   LLEVER LCASHLTD LACIDRAT
##        0        0        0        0        0        0        0        0
##  LCURRAT LRECTURN  LASSLTD
##        0        0        0
```

```
head(bond_validation)
```

```
## # A tibble: 6 x 11
##   RATING LOPMAR LFIXCHAR LGEARRAT LTDCAP LLEVER LCASHLTD LACIDRAT LCURRAT
##   <chr>   <dbl>    <dbl>    <dbl>  <dbl>  <dbl>    <dbl>    <dbl>   <dbl>
## 1 AAA     -1.32    0.998   -0.936  0.281 -0.042   -0.187    0.001   0.863
## 2 AAA     -2.1     1.52    -1.65   0.159  0.251    0.342   -0.077   0.347
## 3 AA      -1.74    1.63    -1.21   0.23  -0.066   -0.266   -0.229   0.543
## 4 AA      -1.78    1.15    -0.45   0.389  0.171   -0.898   -0.073   0.44
## 5 A       -1.70    3.69    -3.16   0.04  -0.936    1.57     0.122   0.998
## 6 A       -1.77    0.887   -0.532  0.369  0.013   -0.929    0.07    0.781
## # ... with 2 more variables: LRECTURN <dbl>, LASSLTD <dbl>
```

- OBS and CODRTG were removed. OBS is the unique ID and RATING is explained by CODERTG numerically.

```
#describe the variables in training and validation
describe(bond_training)
```

```
## Warning in describe(bond_training): NAs introduced by coercion
## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning
## Inf
```

```
## Warning in FUN(newX[, i], ...): no non-missing arguments to max; returning
## -Inf
```

```
##          vars  n  mean   sd median trimmed  mad   min   max range  skew
## RATING*     1 81   NaN   NA     NA     NaN   NA   Inf  -Inf  -Inf    NA
## LOPMAR      2 81 -2.00 0.53  -2.10   -2.03 0.48 -3.40 -0.38  3.02  0.51
## LFIXCHAR    3 81  1.42 0.73   1.38    1.40 0.74  0.00  3.69  3.69  0.30
## LGEARRAT    4 81 -0.70 0.67  -0.70   -0.69 0.63 -3.15  0.95  4.10 -0.35
## LTDCAP      5 81  0.35 0.14   0.33    0.34 0.13  0.04  0.72  0.68  0.40
## LLEVER      6 81  0.16 0.41   0.13    0.13 0.37 -0.94  1.39  2.32  0.55
## LCASHLTD    7 81 -0.76 0.67  -0.76   -0.78 0.69 -2.33  1.57  3.90  0.39
## LACIDRAT    8 81  0.01 0.33   0.01    0.02 0.29 -1.01  0.74  1.75 -0.27
## LCURRAT     9 81  0.67 0.26   0.70    0.68 0.25  0.09  1.24  1.16 -0.13
## LRECTURN   10 81  1.98 0.53   1.95    1.99 0.46 -0.13  3.25  3.39 -0.68
## LASSLTD    11 81  1.52 0.49   1.47    1.49 0.44  0.58  3.49  2.91  0.90
##          kurtosis   se
## RATING*        NA   NA
## LOPMAR       1.17 0.06
## LFIXCHAR    -0.14 0.08
## LGEARRAT     1.06 0.07
## LTDCAP      -0.20 0.02
## LLEVER       0.71 0.05
## LCASHLTD     0.67 0.07
## LACIDRAT     0.64 0.04
## LCURRAT     -0.51 0.03
## LRECTURN     2.43 0.06
## LASSLTD      1.89 0.05
```

```
describe(bond_validation)
```

```
## Warning in describe(bond_validation): NAs introduced by coercion
```

```
## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning
## Inf
```

```
## Warning in FUN(newX[, i], ...): no non-missing arguments to max; returning
## -Inf
```

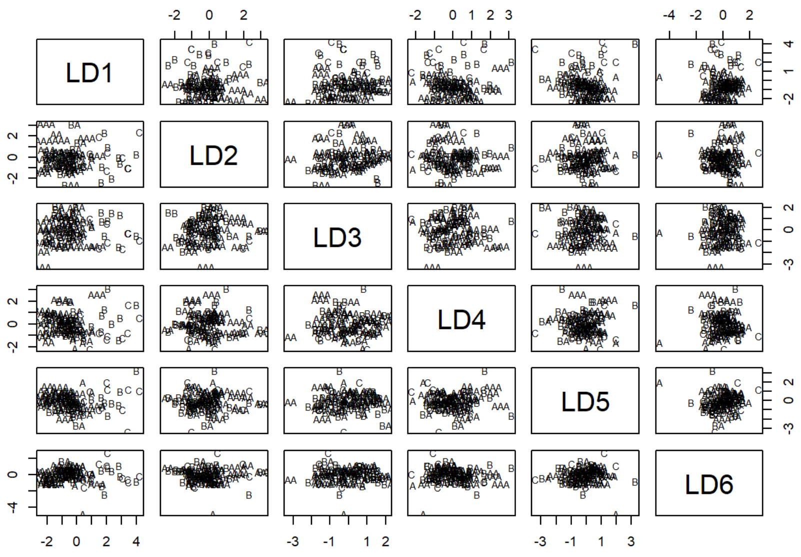
```
##          vars  n  mean   sd median trimmed  mad   min   max range  skew
## RATING*     1 14   NaN   NA     NA     NaN   NA   Inf  -Inf  -Inf    NA
## LOPMAR      2 14 -1.94 0.40  -1.94   -1.97 0.34 -2.48 -1.15  1.33  0.40
## LFIXCHAR    3 14  1.21 0.93   1.05    1.11 0.69  0.00  3.69  3.69  1.28
## LGEARRAT    4 14 -0.88 0.84  -0.92   -0.77 0.63 -3.15  0.03  3.19 -1.16
## LTDCAP      5 14  0.32 0.14   0.28    0.33 0.14  0.04  0.51  0.47 -0.17
## LLEVER      6 14  0.03 0.36  -0.01    0.06 0.26 -0.94  0.63  1.56 -0.99
## LCASHLTD    7 14 -0.60 0.86  -0.71   -0.68 0.76 -1.72  1.57  3.29  0.91
## LACIDRAT    8 14 -0.14 0.22  -0.08   -0.11 0.20 -0.71  0.12  0.83 -1.10
## LCURRAT     9 14  0.60 0.26   0.64    0.61 0.27  0.09  1.00  0.91 -0.36
## LRECTURN   10 14  2.00 0.41   1.99    1.97 0.34  1.35  3.01  1.66  0.62
## LASSLTD    11 14  1.63 0.70   1.64    1.54 0.59  0.89  3.49  2.61  1.20
##          kurtosis   se
## RATING*        NA   NA
## LOPMAR      -0.94 0.11
## LFIXCHAR     1.06 0.25
## LGEARRAT     1.15 0.23
## LTDCAP      -1.06 0.04
## LLEVER       1.39 0.10
## LCASHLTD     0.40 0.23
## LACIDRAT     0.65 0.06
## LCURRAT     -0.98 0.07
## LRECTURN     0.37 0.11
## LASSLTD      0.92 0.19
```

a) What is the performance of the classifier on the training data? Notice that there is order in the class variables (i.e., AAA is better than AA, which is better than A,…).

```
#lda 1
lda1 <- lda(RATING~., data=bond_training)
lda1
```

```
## Call:
## lda(RATING ~ ., data = bond_training)
##
## Prior probabilities of groups:
##         A        AA       AAA         B        BA       BAA         C
## 0.1481481 0.1604938 0.1111111 0.1358025 0.1604938 0.1604938 0.1234568
##
## Group means:
##        LOPMAR  LFIXCHAR    LGEARRAT    LTDCAP      LLEVER   LCASHLTD
## A   -2.017917 1.7306667 -0.94075000 0.3034167  0.04291667 -0.4003333
## AA  -2.094385 1.8042308 -1.05315385 0.2641538 -0.08338462 -0.3925385
## AAA -1.738889 1.6637778 -0.99555556 0.2881111  0.12388889 -0.3940000
## B   -2.078545 0.9529091 -0.07790909 0.4812727  0.44972727 -1.4103636
## BA  -1.981846 1.7073077 -0.75800000 0.3272308  0.07430769 -0.7765385
## BAA -2.213923 1.3204615 -1.01200000 0.2704615 -0.02153846 -0.5720769
## C   -1.783600 0.5873000  0.10860000 0.5248000  0.64370000 -1.4720000
##         LACIDRAT   LCURRAT LRECTURN  LASSLTD
## A    0.017500000 0.6387500 2.074250 1.693417
## AA  -0.003692308 0.6640769 2.266308 1.733462
## AAA  0.059888889 0.6932222 1.943889 1.804000
## B   -0.033181818 0.7031818 1.818182 1.103182
## BA   0.137076923 0.7471538 1.950000 1.510077
## BAA -0.063230769 0.7600769 2.032077 1.721769
## C   -0.031600000 0.4642000 1.650000 0.993700
##
## Coefficients of linear discriminants:
##                 LD1        LD2        LD3          LD4           LD5
## LOPMAR   -0.7720156   2.993776 -1.0902999   1.19056396   0.003079991
## LFIXCHAR  0.3309649   1.032219  2.0342609  -0.17225468  -0.566130362
## LGEARRAT  2.0228900  13.206606  4.3603205  30.56370258  19.296973115
## LTDCAP   27.6725970 -15.434851  1.0663233 -30.15183168   0.636947862
## LLEVER   -5.2113899  -4.540020 -5.2197916 -13.97013291 -12.485287860
## LCASHLTD -0.8040312  -3.684976 -0.6103313  -1.47884309   2.343115368
## LACIDRAT -0.2978150   3.360777 -0.7014467  -0.09884748   0.507853522
## LCURRAT  -2.0007312  -2.040593 -1.1419790   1.51718949  -2.677213623
## LRECTURN -1.1369903   2.245231 -0.6432160   0.81809242   0.686713979
## LASSLTD   5.2328461  14.461158  1.3481935  26.33072526  16.502239043
##                  LD6
## LOPMAR     1.0907388
## LFIXCHAR  -0.4446614
## LGEARRAT   8.6572293
## LTDCAP   -22.5703473
## LLEVER    -4.5123115
## LCASHLTD  -2.1285439
## LACIDRAT   0.9383520
## LCURRAT   -3.2930473
## LRECTURN   0.9182123
## LASSLTD    5.7011832
##
## Proportion of trace:
##    LD1    LD2    LD3    LD4    LD5    LD6
## 0.6309 0.1209 0.1005 0.0705 0.0587 0.0186
```

```
#lda1 plot
plot(lda1)
```



```
#predict values
p = predict(lda1, bond_training)$class
p
```

```
##  [1] AAA B   AA  AA  AA  AAA AAA BAA AAA A   BA  BAA AA  AAA AA  AA  AA 
## [18] AA  BA  BAA AA  AA  BA  AA  BAA AA  A   A   A   A   A   AA  A   BAA
## [35] BAA BAA AA  BAA BAA BAA BAA BA  BAA BAA BAA BAA BAA BA  A   BA  AAA
## [52] BA  BA  BA  BA  BA  BA  BAA AA  BAA C   B   B   B   B   B   B   AAA
## [69] BA  B   B   BAA C   C   A   C   B   A   C   C   C  
## Levels: A AA AAA B BA BAA C
```

```
# Compare the results of the prediction
lda1_table <- table(p, bond_training$RATING)
confusionMatrix(lda1_table)
```

```
## Confusion Matrix and Statistics
## 
##      
## p      A AA AAA  B BA BAA  C
##   A    6  1   0  0  1   0  2
##   AA   3  7   3  0  1   1  0
##   AAA  0  1   4  1  1   0  0
##   B    0  0   1  8  0   0  1
##   BA   1  2   0  1  8   1  0
##   BAA  2  2   1  0  2  11  1
##   C    0  0   0  1  0   0  6
## 
## Overall Statistics
##                                           
##                Accuracy : 0.6173          
##                  95% CI : (0.5026, 0.7231)
##     No Information Rate : 0.1605          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.5506          
##                                           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: AA Class: AAA Class: B Class: BA
## Sensitivity           0.50000   0.53846    0.44444  0.72727   0.61538
## Specificity           0.94203   0.88235    0.95833  0.97143   0.92647
## Pos Pred Value        0.60000   0.46667    0.57143  0.80000   0.61538
## Neg Pred Value        0.91549   0.90909    0.93243  0.95775   0.92647
## Prevalence            0.14815   0.16049    0.11111  0.13580   0.16049
## Detection Rate        0.07407   0.08642    0.04938  0.09877   0.09877
## Detection Prevalence  0.12346   0.18519    0.08642  0.12346   0.16049
## Balanced Accuracy     0.72101   0.71041    0.70139  0.84935   0.77093
##                      Class: BAA Class: C
## Sensitivity              0.8462  0.60000
## Specificity              0.8824  0.98592
## Pos Pred Value           0.5789  0.85714
## Neg Pred Value           0.9677  0.94595
## Prevalence               0.1605  0.12346
## Detection Rate           0.1358  0.07407
## Detection Prevalence     0.2346  0.08642
## Balanced Accuracy        0.8643  0.79296
```

- The question a) asked what the performance of the lda classifier was on the training data. Without using leave-one-out cross validation, the accuracy was 61.7%.
- Sensitivity and Specificity metrics are also given using the confusionMatrix function from the caret package. Should we need to tune the model to be more sensitive to a particular bond rating, we can focus on these values for example.

```
#lda1_withCV = lda(RATING ~ ., data=bond_training, CV=T)
#lda1_withCV
#p_cv = predict(lda1_withCV, bond_training)$class
#p_cv
#lda1_withCV_table <- table(p_cv, bond_training$RATING)
#confusionMatrix(lda1_withCV_table)
```

- The code above uses cross validation, but the accuracy is lower than without using it. I will not use leave-one-out cross validation in part b.

b) What is the performance of the classifier on the validation data?

```
p_validation = predict(lda1, bond_validation)$class
p_validation
```

```
##  [1] BAA AAA AA  AA  A   BAA BAA BAA AA  BAA B   B   C   C  
## Levels: A AA AAA B BA BAA C
```

```
lda1_val_table <- table(p_validation, bond_validation$RATING)
confusionMatrix(lda1_val_table)
```

```
## Confusion Matrix and Statistics
##
##
## p_validation A AA AAA B BA BAA C
##          A   1  0   0 0  0   0 0
##          AA  0  2   0 0  1   0 0
##          AAA 0  0   1 0  0   0 0
##          B   0  0   0 2  0   0 0
##          BA  0  0   0 0  0   0 0
##          BAA 1  0   1 0  1   2 0
##          C   0  0   0 0  0   0 2
##
## Overall Statistics
##
##                Accuracy : 0.7143
##                  95% CI : (0.419, 0.9161)
##     No Information Rate : 0.1429
##     P-Value [Acc > NIR] : 2.034e-06
##
##                   Kappa : 0.6667
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: AA Class: AAA Class: B Class: BA
## Sensitivity           0.50000    1.0000    0.50000   1.0000    0.0000
## Specificity           1.00000    0.9167    1.00000   1.0000    1.0000
## Pos Pred Value        1.00000    0.6667    1.00000   1.0000       NaN
## Neg Pred Value        0.92308    1.0000    0.92308   1.0000    0.8571
## Prevalence            0.14286    0.1429    0.14286   0.1429    0.1429
## Detection Rate        0.07143    0.1429    0.07143   0.1429    0.0000
## Detection Prevalence  0.07143    0.2143    0.07143   0.1429    0.0000
## Balanced Accuracy     0.75000    0.9583    0.75000   1.0000    0.5000
##                      Class: BAA Class: C
## Sensitivity              1.0000   1.0000
## Specificity              0.7500   1.0000
## Pos Pred Value           0.4000   1.0000
## Neg Pred Value           1.0000   1.0000
## Prevalence               0.1429   0.1429
## Detection Rate           0.1429   0.1429
## Detection Prevalence     0.3571   0.1429
## Balanced Accuracy        0.8750   1.0000
```

- The performance of the classifier on the validation data is given by the confusion matrix above, as well as class statistics and overall statistics. For the overall, accuracy was 71.4%. This is 10% points larger than the training set accuracy. This is a fairly large difference in accuracy and suggests that more data is probably needed in the training set - after all it only had 81 observations to train on.

c) Would certain misclassification errors be worse than others? If so, how would you suggest measuring this?

- Yes certain misclassification errors would be worse than others. If a bond rating is a measure of risk, then assigning a higher rating to a bond that should really be rated lower would subject investors to a higher level of risk than expected. For example, if a bond is classified by lda as AAA, but is really a B rating based on it’s riskiness, then investing in it is not as safe as the lda classifier is suggesting.

Extra Credit (10 points)

An academic paper from a conference or Journal will be posted to the Homework 4 content section of D2L. Review the paper and evaluate their usage of FA and Latent Dirichlet Allocation (the other LDA). In particular address the following: (See article on Comparison of Latent Dirichlet Modeling and Factor Analysis for Topic Extraction A Lesson of History)

• What is the application of this paper?

- Factor Analysis / Latent Dirichlet Allocation - Topic Extraction (unstructured text)
- In this paper, the authors apply Factor Analysis and Latent Dirichlet Allocation towards topic extraction from unstructured text. Their intention is to revisit Factor Analysis, a technique that has existed for decades, to topic extraction to compare to Latent Dirichlet Allocation, which has received more research attention in recent years in this area. For this paper, the authors used three datasets used for assessing topic models from the TREC AP corpus, Hawaii Internation Conference abstracts, and Vegas hotel reviews from Expedia.

• What is the research question the authors wish to answer in this paper?

- The research question they are trying to answer is whether Factor Analysis can perform as well, if not better, than Latent Dirichlet Allocation for the purpose of topic extraction on unstructured text. They are also creating a new evaluation method to perform this comparison.

• What is Natural Language Processing (NLP) and what can we learn from it?

- Natural Langauge Processing (NLP) is the application of statistical/data science/machine learning techniques towards the synthesis and analysis of human language data, usually in the form of either structured or unstructured text data. Everything from computing and synthesizing text data, to language translation, to performing sentiment analysis or topic modeling falls under the general umbrella of use cases for NLP techniques. We can learn about the form, structure, and information held in human language through these techniques. Given the ubiquity of unstructured text data, these techniques are becoming more and more relevant.

- **How does this paper utilize FA and LDA in Natural Language Processing?**
    - The paper utilizes FA and LDA for topic modeling on three different datasets, in combination with some participant evaluation. They generated topics for LDA using Mallet and WordStat software for FA.
- **What are the results and conclusions from this paper?**
    - The main conclusion from their results was that Factor Analysis can generate topics that are perceived as more cohert than those created using Latent Dirichlet Allocation.
    - They also argue that FA is more stable in its output compared to LDA. Specifically, multi-topic models generated using LDA can vary based on variation in the data or even the random seed selected.
    - Another proposed benefit of FA is that the topics / words extracted are more independent from eachother, whereas a topic in LDA may have the same words as another generated topic.
    - They note that tuning the parameters of the LDA model or the dataset input could improve their LDA topics but that there was not enough supporting literature to inform how to do this.
    - They do not claim that FA is better at information retrieval, classifcation, or document indexing in this paper, just focused on coherence and topic similarity between LDA and FA. They conclude that that FA is important to revisit in this field of NLP and text-mining.
- **What other areas or fields do you think would benefit from LDA?**
    - Other fields that could benefit from LDA besides Topic Modeling? If that is the question being asked I'm not really sure. I suppose that since probabalistic latent semantic analysis's extension with LDA offer the ability to decompose documents into probability distributions of topics, perhaps a similar process could be used on image data. Instead of taking in word vectors, you could bring in your pixel vectors and see if image topics could be formed that way. I doubt it would provide useful information but I can't think of another non-NLP field that could use LDA. Within the areas explored in this paper directly of abstract synthesis, review topic modeling, and the TREC AP corpus, another NLP topic modeling task could be to test it out on spam emails. Spam emails might have certain common topics or words that are frequent which could be helpful for creating spam filters.
- **What other thoughts do you have on topic modeling, NLP, and LDA?**
    - It is certainly a critical field with many applications. There is so much unstructured text data out there that coming up with means of interpretting and using it is sure to be very lucrative. I've read the arguments for how synthesizing large amounts of text can be useful in fields like research were one could not possibly read all the new papers that come out. Interestingly, the paper authors talk about LDA and how Factor Analysis solves a similar task albeit being an older technique. With better access to already researched methods, using NLP/topic modeling, perhaps some redundancies in the research world could be avoided. In general I'm not as interested in text as I am in imaging applications, but the two must work together for more complex applications. Human language isn't going away, so there will always be needs for processing it in computers.