Classifying beans from images using machine learning in R

By Joshua Buchanan

The Dry Bean Dataset (https://archive.ics.uci.edu/ml/datasets/Dry+Bean+Dataset), obtained from the UCI Machine Learning Repository, has 16 features: 12 dimensions and 4 shapes forms. The features were obtained from high-resolution images of 13,611 grains of 7 different dry beans (Seker, Barbunya, Bombay, Cali, Dermosan, Horoz, and Sira). The dataset also has Class data which is the classification of the bean.

```
> beans = read.csv("Dry_Bean_Dataset.csv")
> dim(beans)
[1] 13611    17
> names(beans)
 [1] "Area"            "Perimeter"       "MajorAxisLength"
 [4] "MinorAxisLength" "AspectRation"    "Eccentricity"
 [7] "ConvexArea"      "EquivDiameter"   "Extent"
[10] "Solidity"        "roundness"       "Compactness"
[13] "ShapeFactor1"    "ShapeFactor2"    "ShapeFactor3"
[16] "ShapeFactor4"    "Class"
```

$n = 13611, C = 7, p = 16$

The classification problem is to be able to classify a test set (30% of the dry bean dataset) into the 7 classes based on the data from all 16 features of a training set (the other 70% of the dataset). Classification using LDA, QDA, KNN, SVMs, and MLR using cross-validation where appropriate will be used to determine the method which has the lowest misclassification rate.

```
> summary(beans)
      Area          Perimeter      MajorAxisLength
 Min.   : 20420   Min.   : 524.7   Min.   :183.6
 1st Qu.: 36328   1st Qu.: 703.5   1st Qu.:253.3
 Median : 44652   Median : 794.9   Median :296.9
 Mean   : 53048   Mean   : 855.3   Mean   :320.1
 3rd Qu.: 61332   3rd Qu.: 977.2   3rd Qu.:376.5
 Max.   :254616   Max.   :1985.4   Max.   :738.9
 MinorAxisLength  AspectRation    Eccentricity
 Min.   :122.5   Min.   :1.025   Min.   :0.2190
 1st Qu.:175.8   1st Qu.:1.432   1st Qu.:0.7159
 Median :192.4   Median :1.551   Median :0.7644
 Mean   :202.3   Mean   :1.583   Mean   :0.7509
```

```
 3rd Qu.:217.0   3rd Qu.:1.707   3rd Qu.:0.8105
 Max.   :460.2   Max.   :2.430   Max.   :0.9114
   ConvexArea     EquivDiameter       Extent
 Min.   : 20684   Min.   :161.2   Min.   :0.5553
 1st Qu.: 36714   1st Qu.:215.1   1st Qu.:0.7186
 Median : 45178   Median :238.4   Median :0.7599
 Mean   : 53768   Mean   :253.1   Mean   :0.7497
 3rd Qu.: 62294   3rd Qu.:279.4   3rd Qu.:0.7869
 Max.   :263261   Max.   :569.4   Max.   :0.8662
    Solidity        roundness       Compactness
 Min.   :0.9192   Min.   :0.4896   Min.   :0.6406
 1st Qu.:0.9857   1st Qu.:0.8321   1st Qu.:0.7625
 Median :0.9883   Median :0.8832   Median :0.8013
 Mean   :0.9871   Mean   :0.8733   Mean   :0.7999
 3rd Qu.:0.9900   3rd Qu.:0.9169   3rd Qu.:0.8343
 Max.   :0.9947   Max.   :0.9907   Max.   :0.9873
  ShapeFactor1       ShapeFactor2        ShapeFactor3
 Min.   :0.002778   Min.   :0.0005642   Min.   :0.4103
 1st Qu.:0.005900   1st Qu.:0.0011535   1st Qu.:0.5814
 Median :0.006645   Median :0.0016935   Median :0.6420
 Mean   :0.006564   Mean   :0.0017159   Mean   :0.6436
 3rd Qu.:0.007271   3rd Qu.:0.0021703   3rd Qu.:0.6960
 Max.   :0.010451   Max.   :0.0036650   Max.   :0.9748
  ShapeFactor4       Class
 Min.   :0.9477   Length:13611
 1st Qu.:0.9937   Class :character
 Median :0.9964   Mode  :character
 Mean   :0.9951
 3rd Qu.:0.9979
 Max.   :0.9997
```

The values in Area and ConvexArea seem very large compared to the other values and ShapeFactor1 and ShapeFactor2 seem very small, scaling the data would be ideal to prevent over-weighing and under-weighing these variables.

```
> beans[,1:16] = scale(beans[,1:16])
```

I created boxplots of each of the continuous variables against Class and took not of which ones were similar to each other.

```
boxplot(Area~Class,data=beans)
boxplot(Perimeter~Class,data=beans)
boxplot(MajorAxisLength~Class,data=beans)
boxplot(MinorAxisLength~Class,data=beans)
boxplot(AspectRation~Class,data=beans)
```

```
boxplot(Eccentricity~Class,data=beans)
boxplot(ConvexArea~Class,data=beans)
boxplot(EquivDiameter~Class,data=beans)
boxplot(Extent~Class,data=beans)
boxplot(Solidity~Class,data=beans)
boxplot(roundness~Class,data=beans)
boxplot(Compactness~Class,data=beans)
boxplot(ShapeFactor1~Class,data=beans)
boxplot(ShapeFactor2~Class,data=beans)
boxplot(ShapeFactor3~Class,data=beans)
boxplot(ShapeFactor4~Class,data=beans)
#The features that have similar boxplots are:
#perimeter, majoraxislength, convexarea, equivdiam
#Area and MinorAxisLength
#compactness, sf3
#From this, I would suggest removing MajorAL, ConvexArea, EquivDiameter,
 MinorAL, & ShapeFactor3. Creating a model with 11 features.
#Will try each method with 16 and 11 features.
```

```
> set.seed(23)
> train = sample(1:nrow(beans),0.7*nrow(beans))
> test = -train
```

Linear discriminant analysis (LDA) -

```
> lda.fit = lda(Class~., data=beans,subset=train)
> lda.pred=predict(lda.fit,beans[test,])
> table(lda.pred$class,beans[test,17])
```

	BARBUNYA	BOMBAY	CALI	DERMASON	HOROZ	SEKER	SIRA
BARBUNYA	341	0	2	2	1	1	0
BOMBAY	0	160	0	0	0	0	0
CALI	39	0	473	0	12	0	1
DERMASON	0	0	0	891	0	12	37
HOROZ	0	0	8	0	536	0	6
SEKER	3	0	0	10	0	559	2
SIRA	38	0	11	132	29	31	747

The misclassification rate is 377/4084 = 9.2% for LDA.

I tried LOOCV which resulted in 921/9527 = 9.7% misclassification rate. Doing model selection and removing each variable at a time did not result in any significant change in the LOOCV misclassification rate.

```
> set.seed(23)
> lda.fit = lda(Class~., data=beans,subset=train,CV=TRUE)
> table(lda.fit$class,beans[train,17])
```

```
          BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      742      1    8        0     3     9    5
  BOMBAY          0    361    0        0     0     0    0
  CALI           81      0 1085        0    25     0    2
  DERMASON        0      0    0     2144     8    18   99
  HOROZ           4      0    9        3  1262     0   14
  SEKER          10      0    2       33     0  1297    8
  SIRA           64      0   32      331    52   100 1715
>
> #LOOCV model selection
> #Without Area didn't change much
> #Perimeter didn't change much at all
> #MajorAL didn't change much
> #MinorAL didn't
> #Aspect didn't
> #Eccentrictiy didn't
> #Convex didn't
> #Equiv didn't
> #Extent didn't
> #Solidity dropped to 906/9527 = 9.5% misclassification
> #not enough difference. Gave 382/4084 on the test error rate
> #which is not better than with all variables.
> #roundness didn't
> #Compactness didn't
> #Sf1 didn't
> #Sf2 didn't
> #sf3 didn't
> #sf4 didn't
> lda.fit = lda(Class~.-Solidity, data=beans,subset=train,CV=TRUE)
> table(lda.fit$class,beans[train,17])

          BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      748      1    8        1     3     9    3
  BOMBAY          0    361    0        0     0     0    0
  CALI           81      0 1085        0    25     0    2
  DERMASON        0      0    0     2149     8    17   98
  HOROZ           4      0   10        3  1262     0   14
  SEKER          10      0    2       30     0  1298    8
  SIRA           58      0   31      328    52   100 1718

> lda.fit = lda(Class~.-MajorAxisLength-ConvexArea-EquivDiameter-MinorAxisLength-Shap
> table(lda.fit$class,beans[train,17])

          BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
```

```
  BARBUNYA      739      1    9        1     6    10    5
  BOMBAY          0    361    0        0     0     0    0
  CALI           78      0 1074        0    46     0    1
  DERMASON        0      0    0     2080    10    12   76
  HOROZ           1      0    6        2  1219     0    7
  SEKER           4      0    2       43     0  1293   12
  SIRA           79      0   45      385    69   109 1742

> #LDA with 11 features
> lda.fit = lda(Class~.-MajorAxisLength-ConvexArea-EquivDiameter-MinorAxisLength-Shap
> lda.pred=predict(lda.fit,beans[test,])
> table(lda.pred$class,beans[test,17])

           BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      336      0    2        1     1     1    0
  BOMBAY          0    160    0        0     0     0    0
  CALI           36      0  471        0    22     0    1
  DERMASON        0      0    0      862     1    12   24
  HOROZ           0      0    6        0   519     0    3
  SEKER           3      0    0       14     0   558    6
  SIRA           46      0   15      158    35    32  759
> #419/4084 = 10.3% misclassification rate.
```

I have decided a model with all 16 features is best and classifies better than the 11 feature model for LDA.

Quadratic discriminant analysis (QDA) -

```
 > set.seed(23)
> qda.fit = qda(Class~., data=beans,subset=train)
> qda.pred = predict(qda.fit,beans[test,])
> table(qda.pred$class,beans[test,17])

           BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      365      0   11        1     1     1    2
  BOMBAY          0    160    0        0     0     0    0
  CALI           41      0  471        0     8     0    1
  DERMASON        0      0    0      886     4     8   33
  HOROZ           1      0    8        4   554     0   18
  SEKER           3      0    0       17     0   577   10
  SIRA           11      0    4      127    11    17  729
> #342/4084 = 8.4% misclassification rate. Better than LDA.
>
```

```
> #QDA cross-validation
> set.seed(23)
> qda.fit = qda(Class~., data=beans,subset=train,CV=TRUE)
> table(qda.fit$class,beans[train,17])

           BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      793      1   29        0     2    14    8
  BOMBAY          1    361    0        0     0     0    0
  CALI           74      0 1077        0    20     0    8
  DERMASON        0      0    0     2173    11    16  107
  HOROZ           6      0   22        6  1293     0   36
  SEKER           6      0    2       44     0  1344   21
  SIRA           21      0    6      288    24    50 1663
> #823/9527 = 8.6 % misclassification rate.
>
> set.seed(23)
> qda.fit = qda(Class~.-Solidity, data=beans,subset=train,CV=TRUE)
> table(qda.fit$class,beans[train,17])

           BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      784      1   31        0     3    14    9
  BOMBAY          1    361    0        0     0     0    0
  CALI           79      0 1077        0    20     0    8
  DERMASON        0      0    0     2181    11    16  106
  HOROZ           6      0   19        6  1292     0   35
  SEKER           7      0    2       44     0  1343   21
  SIRA           24      0    7      280    24    51 1664
> #Removing Solidity to check improvements.
> #825/9527 = no improvement.
```

So far, QDA (8.4%) results in a lower misclassification rate than LDA (9.2%).

K-nearest neighbours classification (KNN) -

```
 > library(class)
> set.seed(3)
> knn.pred = knn(beans[train,-17],beans[test,-17],beans[train,17],k=7)
> table(knn.pred,beans[test,17])

knn.pred   BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      374      0   10        0     0     1    2
  BOMBAY          0    160    0        0     0     0    0
  CALI           28      0  472        0    12     0    1
```

```
  DERMASON          0        0     0       958      3      11    69
  HOROZ             0        0     7         1    547       1    12
  SEKER             4        0     0        13      0     575    11
  SIRA             15        0     5        63     16      15   698
```

KNN resulted in a misclassification rate of 300/4084 = 7.3% with k=7 (7 was the minimum LOOCV obtained using cross-validation). Better than QDA and LDA.

Support vector machines (SVMs) -

```
> y = as.factor(beans$Class)
> Xtrain = data.frame(y=y[train],area=beans$Area[train],
perimeter=beans$Perimeter[train], major = beans$MajorAxisLength[train],
minor=beans$MinorAxisLength[train], aspect=beans$AspectRation[train],
eccentricity=beans$Eccentricity[train], convex = beans$ConvexArea[train],
equiv = beans$EquivDiameter[train], extent = beans$Extent[train],
solidity = beans$Solidity[train], roundness =beans$roundness[train],
compactness=beans$Compactness[train], sf1=beans$ShapeFactor1[train],
sf2=beans$ShapeFactor2[train], sf3=beans$ShapeFactor3[train],
sf4=beans$ShapeFactor4[train])
>
> Xtest = data.frame(y=y[test],area=beans$Area[test],
perimeter=beans$Perimeter[test], major = beans$MajorAxisLength[test],
minor=beans$MinorAxisLength[test], aspect=beans$AspectRation[test],
eccentricity=beans$Eccentricity[test], convex = beans$ConvexArea[test],
equiv = beans$EquivDiameter[test], extent = beans$Extent[test],
solidity = beans$Solidity[test], roundness =beans$roundness[test],
compactness=beans$Compactness[test], sf1=beans$ShapeFactor1[test],
sf2=beans$ShapeFactor2[test], sf3=beans$ShapeFactor3[test],
sf4=beans$ShapeFactor4[test])
>
> svmfit = svm(y~area+perimeter+major+minor+aspect+eccentricity
+convex+equiv+extent+solidity+roundness+compactness+sf1
+sf2+sf3+sf4, data=Xtrain,kernel="linear")
>
> summary(svmfit)

Call:
svm(formula = y ~ area + perimeter + major + minor +
    aspect + eccentricity + convex + equiv + extent +
    solidity + roundness + compactness + sf1 + sf2 +
    sf3 + sf4, data = Xtrain, kernel = "linear")
```

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  1761

 ( 185 467 153 621 172 3 160 )


Number of Classes:  7

Levels:
 BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
```

Cross-validation -

```
> library(e1071)
> set.seed(32)
> tune.out = tune(svm, y~area+perimeter+major+
minor+aspect+eccentricity+convex+equiv+extent+
solidity+roundness+compactness+sf1+sf2+sf3+sf4,
data=Xtrain, kernel="linear",
ranges=list(cost=c(0.01, 0.03, 0.1, 0.3, 1, 3, 5, 7, 9, 12, 20, 30, 50)))
> tune.out

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters: cost: 30

- best performance: 0.07547065

> set.seed(69)
> pred = predict(tune.out$best.model, Xtest)
> table(pred, Xtest$y)

pred       BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      380      0   15        0     2     3    1
  BOMBAY          0    160    0        0     0     0    0
  CALI           23      0  464        0    10     0    1
  DERMASON        0      0    0      970     5    10   73
  HOROZ           1      0   11        0   547     1   10
```

```
  SEKER            5       0    1         8      0   576   11
  SIRA            12       0    3        57     14    13  697
> #290/4084 = 7.1% misclassification rate.
```

SVM using the best model from cross-validation gives a misclassification rate of 290/4084 = 7.1%.

Using kernlab with C =7 instead of the tune function.

```
> library(kernlab)
> set.seed(100)
> kernfit = ksvm(y~area+perimeter+major+minor+aspect+eccentricity+convex+
equiv+extent+solidity+roundness+compactness+sf1+sf2+sf3+sf4,
 data=Xtrain,type="C-svc", kernel="vanilladot", cross = 10, C = 7)
 Setting default kernel parameters
>
> kernfit
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 7

Linear (vanilla) kernel function.

Number of Support Vectors : 1683

Objective Function Value : -0.9519 -1519.665 -16.8632 -281.0301 -421.4618
-508.0839 -0.3789 -0.045 -0.0916 -0.0833 -0.0765 -10.1345 -767.9599
-155.9531 -425.052 -460.7409 -1301.24 -5721.973
-55.1227 -1283.438 -1471.921
Training error : 0.071901
Cross validation error : 0.074423

> set.seed(13)
> pred = predict(kernfit, Xtest)
> table(pred, Xtest$y)

pred       BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      381      0   15        1     2     3    1
  BOMBAY          0    160    0        0     0     0    0
  CALI           23      0  464        0    11     0    0
  DERMASON        0      0    0      967     5     9   72
  HOROZ           0      0   11        0   547     1    9
  SEKER           5      0    1        8     0   578   12
  SIRA           12      0    3       59    13    12  699
> #288/4084 = 7.1% misclassification rate.
```

This gives the same misclassification rate of 7.1% with a cost of 7 rather than 30. Therefore, using C=7 is optimal.

However, I tried using an SVM with Gaussian Radial Basis kernel function with varying costs and sigmas and I found one that has a lower misclassification rate.

```
> set.seed(2)
> kernfit2 = ksvm(y~., data=Xtrain, kernel="rbfdot",C=10000,
kpar=list(sigma=0.001))
> kernfit2
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 10000

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.001

Number of Support Vectors : 1602

Objective Function Value : -475.0477 -1909149 -8064.859 -326459.4
-406782 -624666.7 -190.386 -23.0269 -46.4842 -42.1935 -38.8656
-5069.58 -922754.8 -197908.9 -507927.1 -588345.9 -1785553
-7655377 -66092.38 -1616561 -1971842
Training error : 0.065603
>
> pred2=predict(kernfit2,Xtest)
> table(pred2,Xtest$y)

pred2      BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      385      0   16        0     3     3    2
  BOMBAY          0    160    0        0     0     0    0
  CALI           19      0  467        0     9     0    1
  DERMASON        0      0    0      970     5    10   73
  HOROZ           1      0    7        0   548     1    7
  SEKER           5      0    1       10     0   579   12
  SIRA           11      0    3       55    13    10  698
> #277/4084 = 6.8% misclassification rate.
```

This resulted in a 277/4048 = 6.8% misclassification rate.

Multinomial logistic regression (MLR) -

```
> library(nnet)
```

```
> mlr = multinom(Class~.,data=beans,subset=train)
# weights:  126 (102 variable)
initial  value 18538.685990
iter  10 value 4635.950933
iter  20 value 3905.234506
iter  30 value 2993.216879
iter  40 value 2370.068915
iter  50 value 2013.228202
iter  60 value 1922.167309
iter  70 value 1895.854803
iter  80 value 1890.009686
iter  90 value 1886.572357
iter 100 value 1882.693552
final  value 1882.693552
stopped after 100 iterations
> summary(mlr)
Call:
multinom(formula = Class ~ ., data = beans, subset = train)

Coefficients:
         (Intercept)       Area  Perimeter MajorAxisLength
BOMBAY    -1.6776208   3.622409  -3.320253        4.394378
CALI       1.4700346 -17.666601 -63.187330       40.213195
DERMASON  -2.6875233   3.432653  23.828866      -27.910458
HOROZ      3.4130483  10.764712  18.029888      -32.074625
SEKER     -1.2172481   5.407770  27.540835      -14.870259
SIRA       0.7808995   7.563053 -69.630390       24.785594
         MinorAxisLength AspectRation Eccentricity ConvexArea
BOMBAY          6.643984     3.622077     2.607257    2.61103
CALI           19.874347   -20.511927    -6.332277   -7.34406
DERMASON       -4.204785    12.305907    -1.606614  -11.85558
HOROZ          21.340765    23.747571    -1.816895   11.83518
SEKER         -25.949634    14.504570     0.398340    6.05386
SIRA           -1.044103     9.027093     9.488998  -10.61612
         EquivDiameter      Extent  Solidity roundness
BOMBAY        5.506230 -0.06452685 0.4853973  3.124242
CALI         41.810395  0.10849412 1.3663565 -7.067417
DERMASON     -8.123965 -0.67416727 1.1878749  5.370186
HOROZ       -35.749977 -0.30833396 2.2246506  3.404930
SEKER       -20.883234 -0.57435448 2.1165606  4.085993
SIRA         22.949194 -0.31736020 1.6065409 -7.784342
         Compactness ShapeFactor1 ShapeFactor2 ShapeFactor3
BOMBAY     2.7778670    17.121453    0.8413395     4.582186
CALI     -33.0011907     3.242868   24.5918391    -4.757390
DERMASON   9.1099750     1.589895   -0.5063269    -6.748992
```

```
HOROZ     -0.2452186   -4.272524  17.9091246  -14.168235
SEKER      0.3930022  -17.412043   6.1138614    9.454401
SIRA       5.7136256   -5.655352  -4.2437722   16.184107
         ShapeFactor4
BOMBAY     -0.9551396
CALI       -1.6966215
DERMASON   -1.8897967
HOROZ      -3.9380323
SEKER       1.0599345
SIRA       -3.1938309

Std. Errors:
         (Intercept)     Area Perimeter MajorAxisLength
BOMBAY      6.901843 41.59143 109.20229        68.70252
CALI        1.193052 69.43088  18.73232        62.56481
DERMASON    6.165431 66.81434  21.25720        72.92346
HOROZ       1.876360 80.05722  10.56148        57.90554
SEKER       3.001433 36.60998  14.62866        87.22786
SIRA        2.527878 70.90198  21.20580        63.86583
         MinorAxisLength AspectRation Eccentricity ConvexArea
BOMBAY          59.00447    104.54530    82.428567   48.99813
CALI            33.16054     21.41160     9.040369   64.03198
DERMASON        54.54923     24.09982     9.477552   72.47916
HOROZ           39.02107     22.83755    19.346735   73.79443
SEKER           39.28474     27.50116     7.415305   61.84366
SIRA            56.79916     22.65394    12.011938   74.46474
         EquivDiameter    Extent  Solidity roundness
BOMBAY        61.19456 1.9607032 3.5023904 22.366955
CALI          43.51640 0.1348117 0.7461399  3.152934
DERMASON      42.82426 0.1799967 0.5246391  2.933577
HOROZ         46.79967 0.1659326 0.7643740  1.846554
SEKER         35.92693 0.2271611 0.6286540  2.282252
SIRA          64.67710 0.1635484 0.7030290  2.989676
         Compactness ShapeFactor1 ShapeFactor2 ShapeFactor3
BOMBAY      46.52127     42.16167     57.88211     65.41005
CALI        87.86819     26.24803     24.26980     78.78173
DERMASON    41.56763     22.17767     33.03323     35.92616
HOROZ       58.38542     17.68657     28.00317     61.64676
SEKER       28.16225     31.92775     31.56151     16.30480
SIRA        79.54426     18.65850     30.54778     74.46877
         ShapeFactor4
BOMBAY      5.166725
CALI        1.032236
DERMASON    1.183567
HOROZ       1.206804
```

```
SEKER        1.147204
SIRA         1.408534

Residual Deviance: 3765.387
AIC: 3969.387
```

I will do model selection using AIC to determine if using less of the features results in no loss/improvement in classification.

```
> set.seed(5)
> pred4 = predict(mlr)
> table(pred4,beans$Class[train])
#Training error is 698/9527 = 7.3%

> pred5 = predict(mlr, newdata=beans[test,])
> table(pred5, beans$Class[test])
#298/4084 = 7.3% misclassification rate

#model selection with AIC
> mlrAIC = step(mlr)
> mlrAIC
Call:
multinom(formula = Class ~ Perimeter + MinorAxisLength + AspectRation +
    EquivDiameter + Extent + Solidity + roundness + ShapeFactor2 +
    ShapeFactor3 + ShapeFactor4, data = beans, subset = train)

Coefficients:
         (Intercept) Perimeter MinorAxisLength AspectRation
BOMBAY   -4.52861682 -20.32855        62.30731   19.3685333
CALI     -0.06089977 -46.55688       -33.04964   -0.6510886
DERMASON -2.09357073  19.54994        16.34721    5.7552174
HOROZ     3.20086878  26.04448        52.49295   16.4870624
SEKER     0.47778308  29.34116       -26.64829    7.1039648
SIRA      1.65326011 -68.39792       -48.89224   -1.2366713
         EquivDiameter      Extent Solidity roundness
BOMBAY      -23.776915  0.11588705 1.096693  3.428044
CALI         80.397719  0.08174032 1.431989 -4.284807
DERMASON    -58.555618 -0.73147436 1.261604  5.055283
HOROZ       -76.610768 -0.34658914 2.155829  4.776845
SEKER        -8.706622 -0.63447902 2.051840  4.409810
SIRA         93.530216 -0.34954047 1.679940 -7.426586
         ShapeFactor2 ShapeFactor3 ShapeFactor4
BOMBAY      39.062580   -31.261754    -1.669605
CALI         2.993181     5.986844    -3.615884
DERMASON     2.092961    -4.907563    -1.493863
HOROZ       25.061499   -24.825093    -3.199405
```

```
SEKER       -1.638613    18.830722     1.104004
SIRA       -11.229167    18.816874    -3.571549

Residual Deviance: 3771.65
AIC: 3903.65
```

The model with 10 features: Perimeter, MinorAL, AspectRation, EquivDiam, Extent, Solidity, roundness, Sf2, sf3, sf4 has the lowest AIC.

```
> mlr2 = multinom(Class~.-Area-MajorAxisLength
-Eccentricity-ConvexArea-Compactness-ShapeFactor1,data=beans,subset=train)
> set.seed(5)
> pred6 = predict(mlr2)
> table(pred6,beans$Class[train])

pred6      BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      817      0   33        0     2    10    5
  BOMBAY          0    362    0        0     0     0    0
  CALI           54      0 1072        0    21     0    3
  DERMASON        0      0    0     2298    12    17  161
  HOROZ           4      0   16        4  1286     0   24
  SEKER           7      0    3       42     0  1356   21
  SIRA           19      0   12      167    29    41 1629
> #Training error is 707/9527 = 7.4%
>
> pred7 = predict(mlr2, newdata=beans[test,])
> table(pred7, beans$Class[test])

pred7      BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      381      0   13        0     2     3    0
  BOMBAY          0    160    0        0     0     0    0
  CALI           22      0  466        0    11     0    2
  DERMASON        0      0    0      955     5    10   68
  HOROZ           0      0   11        2   546     1   10
  SEKER           5      0    1       10     0   577   12
  SIRA           13      0    3       68    14    12  701
> #298/4084 = 7.3% misclassification rate
```

Results in exactly the same misclassification rate (7.3%) with only using 10 of the 16 features. I will now try the SVM with Gaussian Radial Basis function using the model with 10 features.

```
> set.seed(2)
> kernfit3 = ksvm(y~.-area-major-eccentricity-convex-compactness-sf1,
 data=Xtrain, kernel="rbfdot",C=10000,kpar=list(sigma=0.001))
```

```
> kernfit3
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 10000

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.001

Number of Support Vectors : 1612

Objective Function Value : -704.7697 -1934480 -16324.87 -336047.2 -419790.6
-634297.2 -372.5205 -44.9948 -92.0169 -80.8133 -75.1647 -10569.63
-935796 -210520.5 -533505.1 -604021.3 -1812266 -7712119 -72815.92
-1707103 -2005748
Training error : 0.066023
>
> pred8=predict(kernfit3,Xtest)
> table(pred8,Xtest$y)

pred8      BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      384      0   15        0     2     3    2
  BOMBAY          0    160    0        0     0     0    0
  CALI           19      0  468        0     8     0    1
  DERMASON        0      0    0      970     4    11   65
  HOROZ           1      0    7        0   550     1    8
  SEKER           6      0    1       10     0   577   11
  SIRA           11      0    3       55    14    11  706
```

The misclassification rate was 269/4084 = 6.6% which is the best out of all the classifications I tried.

Because the cost on this function is so high, I want to try using linear SVMs with the 10 features found to classify the data.

```
> library(kernlab)
> set.seed(100)
> kernfit3 = ksvm(y~perimeter+minor+aspect+equiv+extent
+solidity+roundness+sf2+sf3+sf4,
data=Xtrain,type="C-svc", kernel="vanilladot", cross = 10, C = 7)
Setting default kernel parameters
>
> kernfit3
Support Vector Machine object of class "ksvm"
```

```
SV type: C-svc  (classification)
 parameter : cost C = 7

Linear (vanilla) kernel function.
Number of Support Vectors : 1715

Objective Function Value : -1.4099 -1521.67 -25.9658 -288.1448
 -421.9698 -516.2474 -0.743 -0.0889 -0.1814 -0.1605 -0.1489
  -18.2802 -777.9821 -159.4079 -443.0713 -467.7106 -1335.172
   -5750.088 -60.4047 -1380.205 -1498.658
Training error : 0.073475
Cross validation error : 0.075368
> pred2=predict(kernfit2,Xtest)
> table(pred2,Xtest$y)

pred2      BARBUNYA BOMBAY CALI DERMASON HOROZ SEKER SIRA
  BARBUNYA      385      0   16        0     3     3    2
  BOMBAY          0    160    0        0     0     0    0
  CALI           19      0  467        0     9     0    1
  DERMASON        0      0    0      970     5    10   73
  HOROZ           1      0    7        0   548     1    7
  SEKER           5      0    1       10     0   579   12
  SIRA           11      0    3       55    13    10  698
> #277/4084 = 6.8%
```

This resulted in a 6.8% misclassification rate.

The misclassification rate using LDA with all 16 features was 9.2%.
The misclassification rate using proposed 11 features was 10.3%.
The misclassification rate using QDA with all 16 features was 8.4%.
The misclassification rate using KNN with k=7 and all 16 features was 7.3%.
The misclassification rates using Linear SVMs with costs of 7 and 30 with all 16 features were both 7.1%.
The misclassification rate using SVMs with Gaussian Radial Basis kernel function, with cost 10000 and sigma 0.001, with all 16 features was 6.8%.
The misclassification rate using MLR with 16 features was 7.3%. The misclassification rate using MLR with 10 features as selected by AIC was also 7.3% but at a lower cost.
The misclassification rate using the SVMs with GRBkf with the 10 features as selected by AIC was 6.6%.
The misclassification rate using linear SVMs with a cost of 7 with the 10 features was 6.8%.

In conclusion, I would choose to use linear SVMs with the 10 features as it had the lowest misclassification rate except for the high-cost gaussian radial basis function SVMs. The linear SVMs cost is 7 and uses only 10 of the 16 features. The models all seem to have the most problem distinguishing between Dermason and Sira bean grains which must be the most similar on these features.

Dry Bean Dataset Reference
Source:
Murat KOKLU Faculty of Technology, Selcuk University, TURKEY. ORCID : 0000-0002-2737-2360 mkoklu '@' selcuk.edu.tr
Ilker Ali OZKAN Faculty of Technology, Selcuk University, TURKEY. ORCID : 0000-0002-5715-1040 ilkerozkan '@' selcuk.edu.tr
Found at URL: https://archive.ics.uci.edu/ml/datasets/Dry+Bean+Dataset