Lab1

Tristen Tooming

2/13/2021

1

A

```
n <- 500 # Number of Observations (Rows)
p <- 20 # Number of "Variables"
ntr <- 5 # Number of datasets to be simulated
sig <- 0.2 # Scale parameter
X <- matrix(rnorm(n*p),ncol=p,nrow=n) # Generating values
set.seed(3) # Seed for the random generator
b_imp <- rnorm(ntr) # Random draw from normal distribution
b_zero <- rep(0,(p-ntr)) # zero vector
b_true <- c(b_imp,b_zero) #
y <- X%*%b_true + sig * rnorm(n) # Generating response values
```

B

```
# Data frame
data = data.frame(y=y, X=X)
# Splitting data
train_portition = 0.7
sample_size = floor(n * train_portition)
set.seed(2707)
train_index = sample(seq_len(n), size = sample_size)
train = data[train_index, ]
test = data[-train_index, ]
```

C

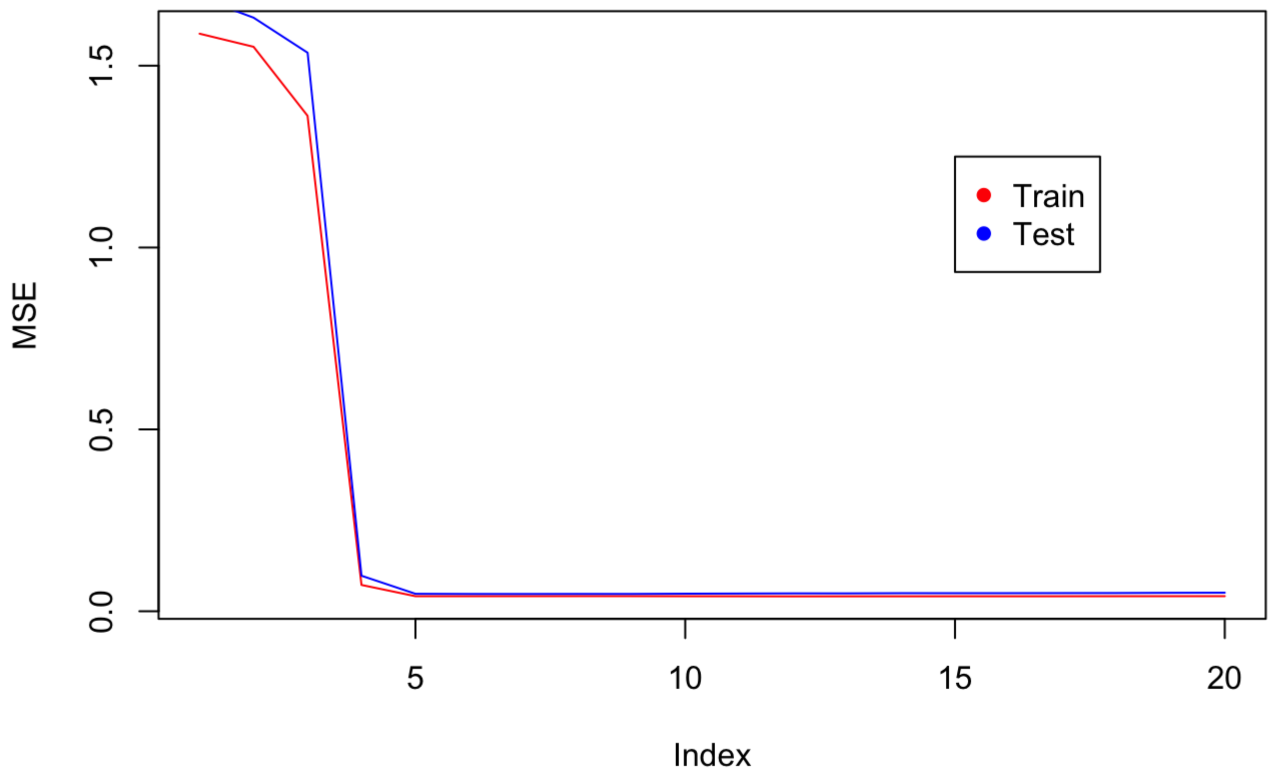
```
# Fitting the Model
linear_model = lm(y ~ ., data = train)
# Anova
anova(linear_model)
```

```
## Analysis of Variance Table
##
## Response: y
##            Df Sum Sq Mean Sq    F value  Pr(>F)
## X.1         1 214.78  214.78  5177.4062 < 2e-16 ***
## X.2         1  14.09   14.09   339.6449 < 2e-16 ***
## X.3         1  67.16   67.16  1618.9204 < 2e-16 ***
## X.4         1 446.41  446.41 10760.7778 < 2e-16 ***
## X.5         1  10.73   10.73   258.5844 < 2e-16 ***
## X.6         1   0.02    0.02     0.4666 0.49502
## X.7         1   0.02    0.02     0.3777 0.53926
## X.8         1   0.01    0.01     0.2682 0.60489
## X.9         1   0.05    0.05     1.1379 0.28688
## X.10        1   0.13    0.13     3.0392 0.08221 .
## X.11        1   0.08    0.08     1.9364 0.16500
## X.12        1   0.13    0.13     3.0201 0.08317 .
## X.13        1   0.00    0.00     0.0422 0.83745
## X.14        1   0.04    0.04     0.9600 0.32791
## X.15        1   0.01    0.01     0.1430 0.70555
## X.16        1   0.05    0.05     1.2942 0.25611
## X.17        1   0.02    0.02     0.4641 0.49621
## X.18        1   0.00    0.00     0.0575 0.81072
## X.19        1   0.02    0.02     0.5041 0.47821
## X.20        1   0.01    0.01     0.1498 0.69899
## Residuals 329  13.65    0.04
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

> Based on the F-test variables X.1, X.2, X.3, X.4 and X.5 are significant. Variable X.6 could be useful for improving model performance but needs more validation than simple F-test.

```r
mse_train = c()
mse_test = c()
variables = names(data)[-1]
for (i in 1:p) {
  lm_formula = as.formula(paste('y ~ ', paste(variables[1:i], collapse = '+')))
  fit_train = lm(lm_formula, data=train)
  mse_train = c(mse_train, anova(fit_train)['Residuals', 'Mean Sq'])
  mse_test = c(mse_test, mean((test$y - predict.lm(fit_train, test)) ^ 2))
}
```

```r
plot(mse_train, type = 'l', col = 'red', ylab = 'MSE')
lines(mse_test, type = 'l', col = 'blue')
legend(15, 1.25, col = c('red', 'blue'), legend = c('Train', 'Test'), pch = 16)
```




```
knitr::kable(data.frame(Iteration = 1:length(mse_test),MSE_Train = mse_train, MSE_Test = mse_te
st))
```

Iteration	MSE_Train	MSE_Test
1	1.5879552	1.6872759
2	1.5519261	1.6320610
3	1.3623063	1.5357965
4	0.0723210	0.0976254
5	0.0413473	0.0479758
6	0.0414114	0.0475739
7	0.0414867	0.0474908
8	0.0415757	0.0476685
9	0.0415592	0.0475990
10	0.0413098	0.0482969
11	0.0411944	0.0487205
12	0.0409448	0.0491813
13	0.0410615	0.0491542
14	0.0410652	0.0497245
15	0.0411704	0.0497033
16	0.0411328	0.0497686
17	0.0411987	0.0499699

Iteration	MSE_Train	MSE_Test
18	0.0413160	0.0501402
19	0.0413778	0.0507211
20	0.0414847	0.0510355

> Model performance increases significally when fourth term is included and reaches its maxium with fifth predictor. After that there is no significant increase in performance when comparing models with MSE. We choose to use model with formula y ~ X1 + X2 + X3 + X4 + X5. Minimum MSE test is at iteration #5

2

```r
# Reading data
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/00236/seeds_dataset.txt'
wheat_seeds = read.table(url)
# Factoring and Naming variables
column_names = c('Area_A',
'Åerimeter_P',
'Compactness_C',
'Length_of_kernel',
'Width_of_kernel',
'Asymmetry_coefficient',
'Length_of_kernel_groove',
'Variety')
colnames(wheat_seeds) = column_names
variety_names = c()
wheat_seeds$Variety = as.factor(wheat_seeds$Variety)
set.seed(3)

wheat_seeds_permuted = sample(wheat_seeds)
# Splitting data
train_portition = 0.7
n = nrow(wheat_seeds)
sample_size = floor(n * train_portition)
set.seed(2707)
train_index = sample(seq_len(n), size = sample_size)
wheat_seeds_train = wheat_seeds_permuted[train_index, ]
wheat_seeds_test = wheat_seeds_permuted[-train_index, ]

calc_acc = function(actual, predicted) {
accuracy = sum( actual == predicted)/length(actual) * 100
return(round(accuracy, 2))
}
```

```r
# Packages
library(nnet)
# Model fit
model_multinom = multinom(Variety ~ ., data = wheat_seeds_train)
```

```
## # weights:  27 (16 variable)
## initial  value 161.496006
## iter  10 value 17.735252
## iter  20 value 7.402402
## iter  30 value 6.999950
## iter  40 value 6.784102
## iter  50 value 6.629221
## iter  60 value 5.232829
## iter  70 value 5.014543
## iter  80 value 4.568702
## iter  90 value 4.411738
## iter 100 value 4.106498
## final  value 4.106498
## stopped after 100 iterations
```

```
predicted_multinom_wheat_seeds = predict(model_multinom, wheat_seeds_test)
acc_multinom = calc_acc(wheat_seeds_test$Variety, predicted_multinom_wheat_seeds)
cat(sprintf("\n\nModel Acc: %s", acc_multinom))
```

```
##
##
## Model Acc: 88.89
```

```
library(class)
accuracies = c() # Index = K-value
for (i in 1:25) {
fitted_knn = knn(train = wheat_seeds_train,
test = wheat_seeds_test,
k= i,
cl = wheat_seeds_train$Variety)
accuracies = c(accuracies, calc_acc(wheat_seeds_test$Variety, fitted_knn))
}
for (i in 1:length(accuracies)) {
cat(sprintf("K-value: %s, Accuracy: %s\n", i, accuracies[i]))
}
```

```
## K-value: 1, Accuracy: 100
## K-value: 2, Accuracy: 98.41
## K-value: 3, Accuracy: 98.41
## K-value: 4, Accuracy: 98.41
## K-value: 5, Accuracy: 98.41
## K-value: 6, Accuracy: 95.24
## K-value: 7, Accuracy: 96.83
## K-value: 8, Accuracy: 95.24
## K-value: 9, Accuracy: 96.83
## K-value: 10, Accuracy: 96.83
## K-value: 11, Accuracy: 96.83
## K-value: 12, Accuracy: 96.83
## K-value: 13, Accuracy: 96.83
## K-value: 14, Accuracy: 96.83
## K-value: 15, Accuracy: 95.24
## K-value: 16, Accuracy: 93.65
## K-value: 17, Accuracy: 93.65
## K-value: 18, Accuracy: 92.06
## K-value: 19, Accuracy: 92.06
## K-value: 20, Accuracy: 92.06
## K-value: 21, Accuracy: 92.06
## K-value: 22, Accuracy: 92.06
## K-value: 23, Accuracy: 93.65
## K-value: 24, Accuracy: 93.65
## K-value: 25, Accuracy: 93.65
```

```
cat(sprintf('\n\nBest Model: %s', which(accuracies==max(accuracies))
))
```

```
##
##
## Best Model: 1
```

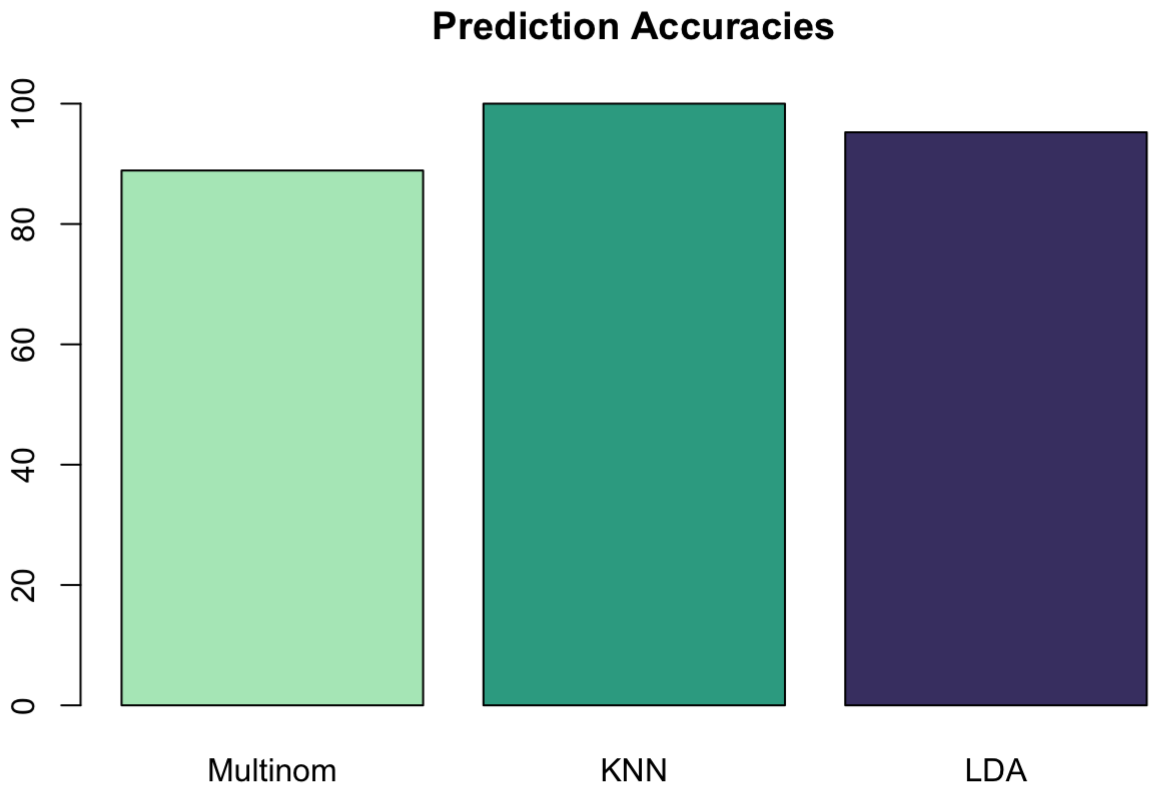
```
acc_knn = max(accuracies)
```

C

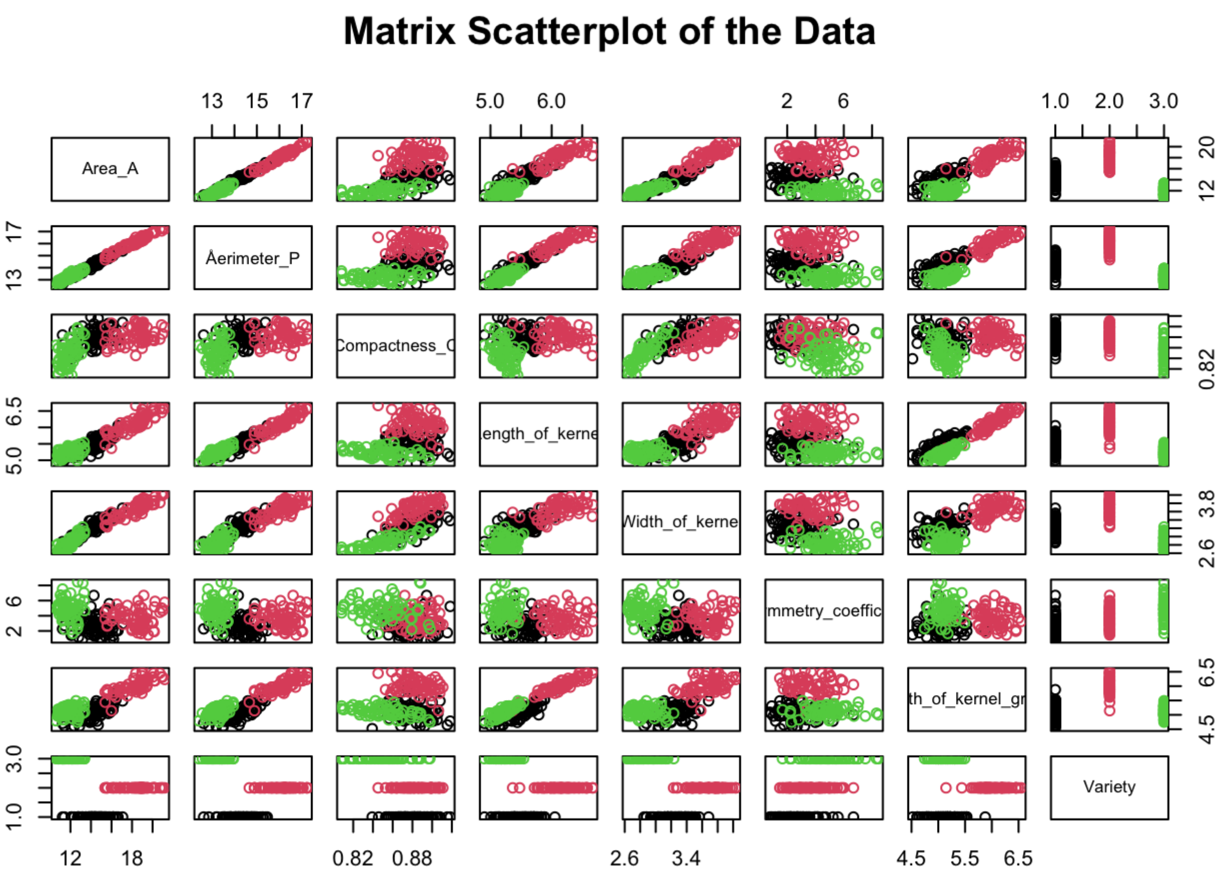
```
library(MASS)
model_lda = lda(Variety ~ ., data = wheat_seeds_train)
predicted_lda = predict(model_lda, wheat_seeds_test)$class
acc_lda = calc_acc(wheat_seeds_test$Variety, predicted_lda)
cat(sprintf('LDA accuracy %s', acc_lda))
```

```
## LDA accuracy 95.24
```

```
barplot(c(acc_multinom, acc_knn, acc_lda),
names.arg = c('Multinom', 'KNN', 'LDA'),
main = 'Prediction Accuracies',
col = c('#b2e8c2', '#33a892', '#483f72'))
```




```
pairs(wheat_seeds, col = wheat_seeds$Variety, main = "Matrix Scatterplot of the Data")
```



Conclusion With 100% (K-value = 1) accuracy KNN performs best then comes LDA with 95% accuracy and finally Multinomial with 89% accuracy. As seen in the scatter plot matrix varieties are forming distinct 6 clusters from each other and so on indicating good performance of the KNN. All in all I would choose KNN for this particular dataset for predicting wheat varieties for its good accuracy, cheap computing costs and specially for its simplicity