HW4

David Schultheiss

9/28/2020

##Problem 1

```
library(readr)
seeds <- read_csv("/Users/davidschultheiss/Downloads/seeds.csv")
```

```
## Parsed with column specification:
## cols(
##   Area = col_double(),
##   perimeter = col_double(),
##   compactness = col_double(),
##   length.of.kernel = col_double(),
##   width.of.kernel = col_double(),
##   asymmetry.coefficient = col_double(),
##   length.of.kernel.groove = col_double(),
##   Type = col_double()
## )
```

```
seeds$Type[which(seeds$Type==1)]="Kama"
seeds$Type[which(seeds$Type==2)]="Rosa"
seeds$Type[which(seeds$Type==3)]="Canadian"

seeds$Type = factor(seeds$Type)
```

a)

```
library(GGally)
```

```
## Warning: package 'GGally' was built under R version 3.6.2
```

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

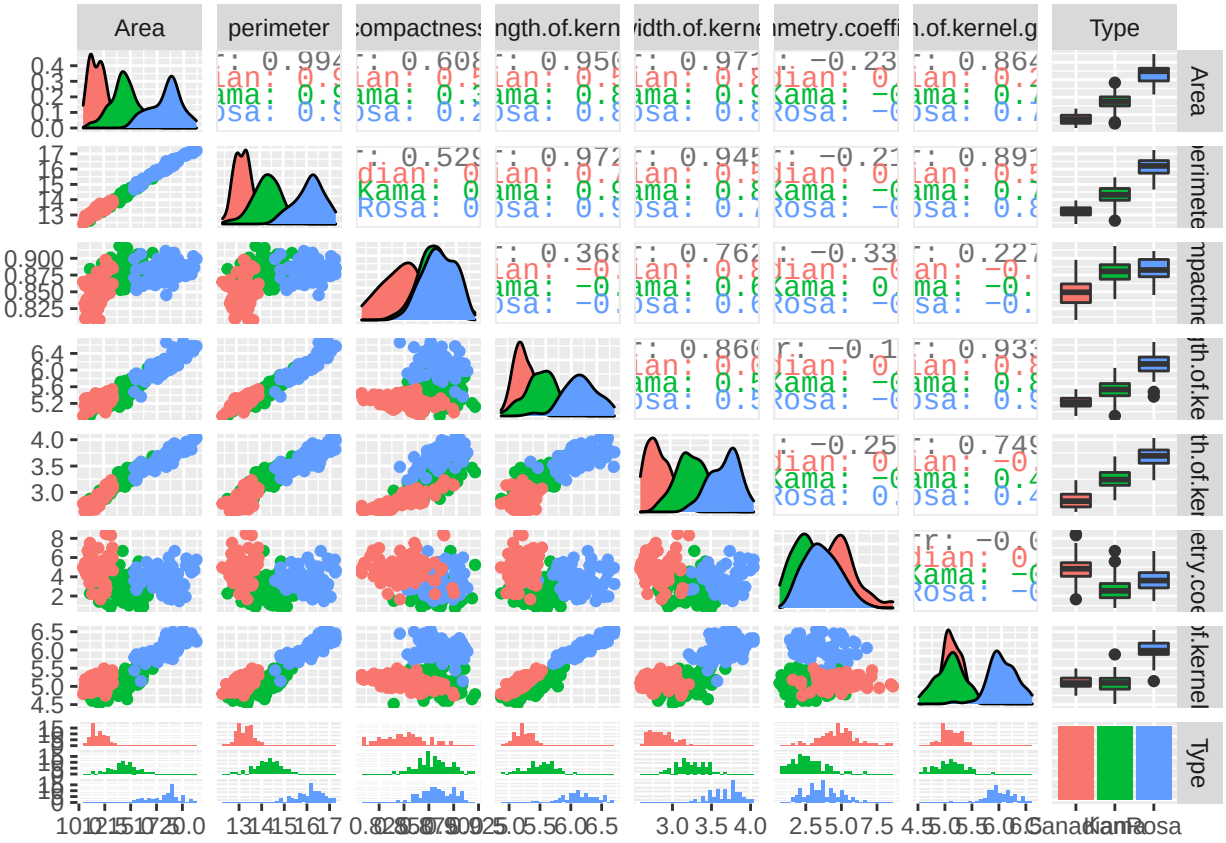
```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
ggpairs(data= seeds, mapping= aes(color= seeds$Type))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Length of Kernel Groove, Compactness, and Asymmetry Coefficient. I chose these because all have weak correlations (>.34), and clustering for different Types.

b)

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v purrr   0.3.3     v forcats 0.4.0
```

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

```
set.seed(1)
testsample = sample(1:nrow(seeds), .2*nrow(seeds))
test = seeds[testsample, ]
training = seeds[-testsample, ]

test = select(test, Type, length.of.kernel.groove, compactness,
              asymmetry.coefficient)
training = select(training, Type, length.of.kernel.groove, compactness,
              asymmetry.coefficient)
```

c) KNN -

```
library(class)

knn.fit = knn(training[,-1], test[,-1], training$Type, k=1)
miss = length(which(knn.fit != test$Type))
(rate = (miss / length(knn.fit)) * 100)
```

```
## [1] 30.95238
```

```
knn.fit = knn(training[,-1], test[,-1], training$Type, k=5)
miss = length(which(knn.fit != test$Type))
(rate = (miss / length(knn.fit)) * 100)
```

```
## [1] 28.57143
```

```
knn.fit = knn(training[,-1], test[,-1], training$Type, k=10)
miss = length(which(knn.fit != test$Type))
(rate = (miss / length(knn.fit)) * 100)
```

```
## [1] 23.80952
```

LDA -

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
ldafit = lda(data = training, Type ~ length.of.kernel.groove + compactness +
      asymmetry.coefficient)
lda.pred = predict(ldafit, test)

table(test$Type, lda.pred$class)
```

```
##
##            Canadian Kama Rosa
##   Canadian       11    3    0
##   Kama            4    9    1
##   Rosa            0    0   14
```

```
mean(test$Type != lda.pred$class)
```

```
## [1] 0.1904762
```

QDA -

```
qdafit = qda(data = training, Type ~ length.of.kernel.groove + compactness +
               asymmetry.coefficient)
qda.pred = predict(qdafit, test)

table(test$Type, qda.pred$class)
```

```
##
##            Canadian Kama Rosa
##   Canadian       12    2    0
##   Kama            5    8    1
##   Rosa            0    0   14
```

```
mean(test$Type != qda.pred$class)
```

```
## [1] 0.1904762
```

d) The discriminant analysis models both have lower misclassification rates than KNN, and perform better with our data.

##Problem 2

```
tumor <- read_csv("/Users/davidschultheiss/Downloads/tumor.csv")
```

```
## Parsed with column specification:
## cols(
##   Diagnosis = col_character(),
##   Radius = col_double(),
##   Texture = col_double(),
##   Perimeter = col_double(),
##   Area = col_double(),
##   Smoothness = col_double(),
```

```
##   Compactness = col_double(),
##   Concavity = col_double(),
##   `Concave Points` = col_double(),
##   Symmetry = col_double(),
##   `Fractal Dimension` = col_double()
## )
```

a)

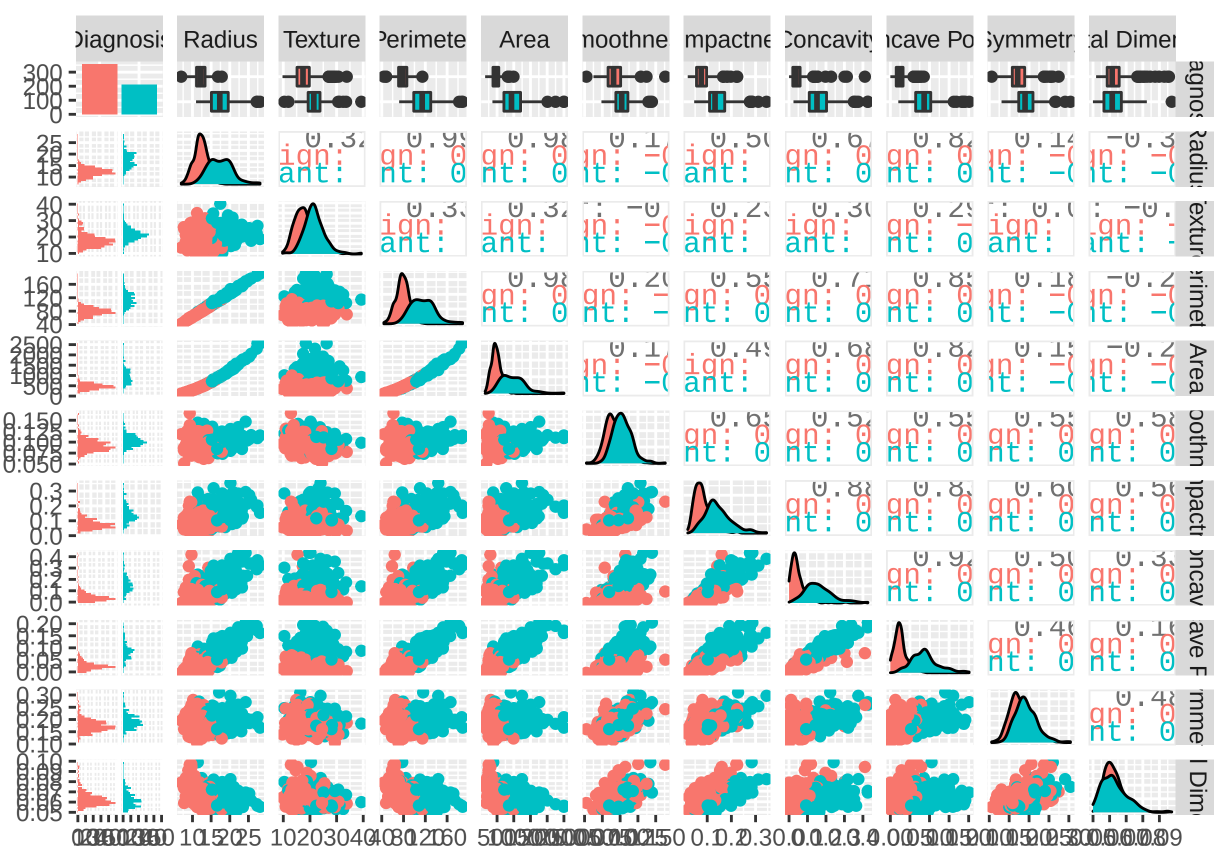
```r
is.factor(tumor$Diagnosis)
```

```
## [1] FALSE
```

```r
tumor$Diagnosis = factor(tumor$Diagnosis)

ggpairs(data= tumor, mapping= aes(color= Diagnosis))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Benign is red, and Malignant is teal.

Obviously variables like perimeter, area, and radius are highly correlated. Compactness and Concavity are also related.

No Variables look particularly reliable for predicting Diagnosis. Variables that could be useful are radius, texture, perimeter, area, compactness, concavity, and concave points.

I will be using three variables. Texture, area, and concavity. Area and concavity do have a .67 correlation, but most of our significant variables are highly related.

I think that KNN being non-parametric and having a decent amount of observations to train the data will make it the most reliable method.

b)

```
set.seed(1)
testsample = sample(1:nrow(tumor), .2*nrow(tumor))
test = tumor[testsample, ]
training = tumor[-testsample, ]
```

c)

```
glm.fit = glm(data= training, Diagnosis ~ Texture + Area + Concavity,
              family= binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Diagnosis ~ Texture + Area + Concavity, family = binomial,
##     data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1928  -0.2628  -0.0941   0.0384   3.0324
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -14.004711   1.652111  -8.477  < 2e-16 ***
## Texture       0.214226   0.051518   4.158 3.21e-05 ***
## Area          0.011207   0.001525   7.349 2.00e-13 ***
## Concavity    26.062158   3.799056   6.860 6.88e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 610.92  on 455  degrees of freedom
## Residual deviance: 172.77  on 452  degrees of freedom
## AIC: 180.77
##
## Number of Fisher Scoring iterations: 7
```

All of our variables are significant.

d)

```
glm.probs = predict(glm.fit, test, type= 'response')
glm.pred = rep(0, nrow(test))
glm.pred[glm.probs > 0.5] = 1

table(test$Diagnosis, glm.pred)
```

```
##             glm.pred
##               0  1
##   Benign     77  3
##   Malignant   4 29
```

Misclassification rate is 7/113 or 6.19%

e) False Negative Rate = 4/81 or 4.94%. False Positive Rate = 3/32 or 9.37%

f)

```
glm.pred = rep(0, nrow(test))
glm.pred[glm.probs > 0.3] = 1

table(test$Diagnosis, glm.pred)
```

```
##             glm.pred
##               0  1
##   Benign     74  6
##   Malignant   1 32
```

Misclassification rate holds at 6.19%. FNR falls to 1.33%. FPR rises to 15.79%.

g)

```
test = dplyr::select(test, Diagnosis, Texture, Area, Concavity)
training = dplyr::select(training, Diagnosis, Texture, Area, Concavity)

knn.fit = knn(training[,-1], test[,-1], training$Diagnosis, k=10)
miss = length(which(knn.fit != test$Diagnosis))
(rate = (miss / length(knn.fit)) * 100)
```

```
## [1] 8.849558
```

Misclassification for KNN is 8.85%

```
ldafit = lda(data = training, Diagnosis ~ Texture + Area + Concavity)
lda.pred = predict(ldafit, test)
table(test$Diagnosis, lda.pred$class)
```

```
##
##              Benign Malignant
##   Benign         80         0
##   Malignant       6        27
```

```r
mean(test$Diagnosis != lda.pred$class)
```

```
## [1] 0.05309735
```

Misclassification rate for LDA is 5.3%. LDA is the lowest overall.

h) LDA is the best for this researcher. It has better predictive capability with a lower misclassifcation rate than logistic regression. KNN is not able to examine individual variables.