Lab05

Amanda Mentesana

2025-03-21

Firstly, set up libraries and read dataset.

```
knitr::opts_chunk$set(echo = FALSE)

#install libraries
library(readr)
```

```
## Warning: package 'readr' was built under R version 4.4.2
```

```
library(EnvStats)
```

```
## Warning: package 'EnvStats' was built under R version 4.4.2
```

```
##
## Attaching package: 'EnvStats'
```

```
## The following objects are masked from 'package:stats':
##
##     predict, predict.lm
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.2
```

```
library(ggfortify)
```

```
## Warning: package 'ggfortify' was built under R version 4.4.3
```

```
library(class)
```

```
## Warning: package 'class' was built under R version 4.4.2
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.2
```

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

```
## Warning: package 'lattice' was built under R version 4.4.2
```

```
## Registered S3 method overwritten by 'lava':
##   method         from
##   print.estimate EnvStats
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.4.2
```

```
##
## Attaching package: 'e1071'
```

```
## The following objects are masked from 'package:EnvStats':
##
##     kurtosis, skewness
```

```
library(readr)


#read the wine data set
wine <- read_csv("C:/Users/amanda/Downloads/wine/wine.data")
```

```
## Rows: 177 Columns: 14
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (14): 1, 14.23, 1.71, 2.43, 15.6, 127, 2.8, 3.06, .28, 2.29, 5.64, 1.04,...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
colnames(wine) <- c("class","Alcohol","Malic acid","Ash","Alcalinity of ash","Magnesium","Total phenols

summary(wine)
```

```
##      class          Alcohol        Malic acid        Ash
##  Min.   :1.000   Min.   :11.03   Min.   :0.74   Min.   :1.360
##  1st Qu.:1.000   1st Qu.:12.36   1st Qu.:1.60   1st Qu.:2.210
##  Median :2.000   Median :13.05   Median :1.87   Median :2.360
##  Mean   :1.944   Mean   :12.99   Mean   :2.34   Mean   :2.366
##  3rd Qu.:3.000   3rd Qu.:13.67   3rd Qu.:3.10   3rd Qu.:2.560
##  Max.   :3.000   Max.   :14.83   Max.   :5.80   Max.   :3.230
##  Alcalinity of ash   Magnesium      Total phenols     Flavanoids
##  Min.   :10.60     Min.   : 70.00   Min.   :0.980   Min.   :0.340
##  1st Qu.:17.20     1st Qu.: 88.00   1st Qu.:1.740   1st Qu.:1.200
##  Median :19.50     Median : 98.00   Median :2.350   Median :2.130
##  Mean   :19.52     Mean   : 99.59   Mean   :2.292   Mean   :2.023
##  3rd Qu.:21.50     3rd Qu.:107.00   3rd Qu.:2.800   3rd Qu.:2.860
##  Max.   :30.00     Max.   :162.00   Max.   :3.880   Max.   :5.080
##  Nonflavanoid phenols Proanthocyanins Color intensity       Hue
```

```
##  Min.   :0.1300       Min.   :0.410   Min.   : 1.280   Min.   :0.480
##  1st Qu.:0.2700       1st Qu.:1.250   1st Qu.: 3.210   1st Qu.:0.780
##  Median :0.3400       Median :1.550   Median : 4.680   Median :0.960
##  Mean   :0.3623       Mean   :1.587   Mean   : 5.055   Mean   :0.957
##  3rd Qu.:0.4400       3rd Qu.:1.950   3rd Qu.: 6.200   3rd Qu.:1.120
##  Max.   :0.6600       Max.   :3.580   Max.   :13.000   Max.   :1.710
##  OD280/OD315 of diluted wines    Proline
##  Min.   :1.270                Min.   : 278.0
##  1st Qu.:1.930                1st Qu.: 500.0
##  Median :2.780                Median : 672.0
##  Mean   :2.604                Mean   : 745.1
##  3rd Qu.:3.170                3rd Qu.: 985.0
##  Max.   :4.000                Max.   :1680.0
```

```
str(wine)
```

```
## spc_tbl_ [177 x 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ class                       : num [1:177] 1 1 1 1 1 1 1 1 1 1 ...
##  $ Alcohol                     : num [1:177] 13.2 13.2 14.4 13.2 14.2 ...
##  $ Malic acid                  : num [1:177] 1.78 2.36 1.95 2.59 1.76 1.87 2.15 1.64 1.35 2.16 ...
##  $ Ash                         : num [1:177] 2.14 2.67 2.5 2.87 2.45 2.45 2.61 2.17 2.27 2.3 ...
##  $ Alcalinity of ash           : num [1:177] 11.2 18.6 16.8 21 15.2 14.6 17.6 14 16 18 ...
##  $ Magnesium                   : num [1:177] 100 101 113 118 112 96 121 97 98 105 ...
##  $ Total phenols               : num [1:177] 2.65 2.8 3.85 2.8 3.27 2.5 2.6 2.8 2.98 2.95 ...
##  $ Flavanoids                  : num [1:177] 2.76 3.24 3.49 2.69 3.39 2.52 2.51 2.98 3.15 3.32 ...
##  $ Nonflavanoid phenols        : num [1:177] 0.26 0.3 0.24 0.39 0.34 0.3 0.31 0.29 0.22 0.22 ...
##  $ Proanthocyanins             : num [1:177] 1.28 2.81 2.18 1.82 1.97 1.98 1.25 1.98 1.85 2.38 ...
##  $ Color intensity             : num [1:177] 4.38 5.68 7.8 4.32 6.75 5.25 5.05 5.2 7.22 5.75 ...
##  $ Hue                         : num [1:177] 1.05 1.03 0.86 1.04 1.05 1.02 1.06 1.08 1.01 1.25 ...
##  $ OD280/OD315 of diluted wines: num [1:177] 3.4 3.17 3.45 2.93 2.85 3.58 3.58 2.85 3.55 3.17 ...
##  $ Proline                     : num [1:177] 1050 1185 1480 735 1450 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   `1` = col_double(),
##   ..   `14.23` = col_double(),
##   ..   `1.71` = col_double(),
##   ..   `2.43` = col_double(),
##   ..   `15.6` = col_double(),
##   ..   `127` = col_double(),
##   ..   `2.8` = col_double(),
##   ..   `3.06` = col_double(),
##   ..   `.28` = col_double(),
##   ..   `2.29` = col_double(),
##   ..   `5.64` = col_double(),
##   ..   `1.04` = col_double(),
##   ..   `3.92` = col_double(),
##   ..   `1065` = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

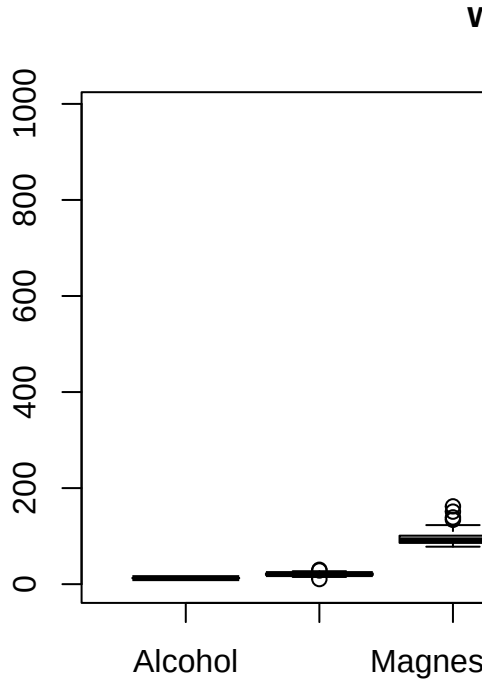
```
wine <- wine[wine$class != 1, ]
wine$class <- as.factor(wine$class)
wine <- na.omit(wine)
```

```
wine <- wine[ , -c(3, 4, 9, 10, 12, 13)] #cleaning taken from Lab04 PCA
```

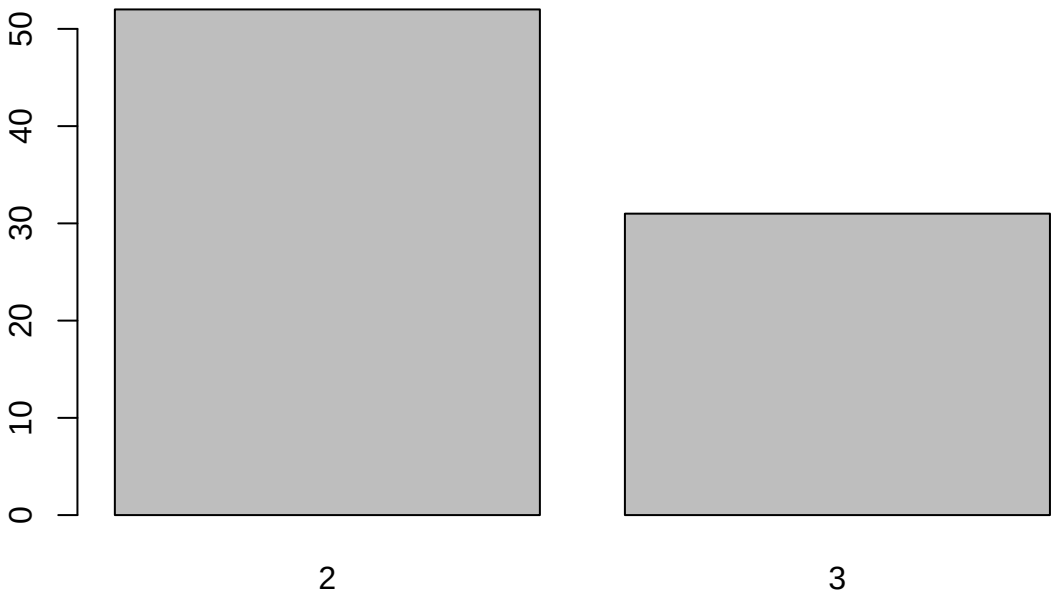
Train 2 SVM classifiers to predict the type of wine using a subset of the other 13 variables. You may choose the subset based on previous analysis. One using a linear kernel and another of your choice.

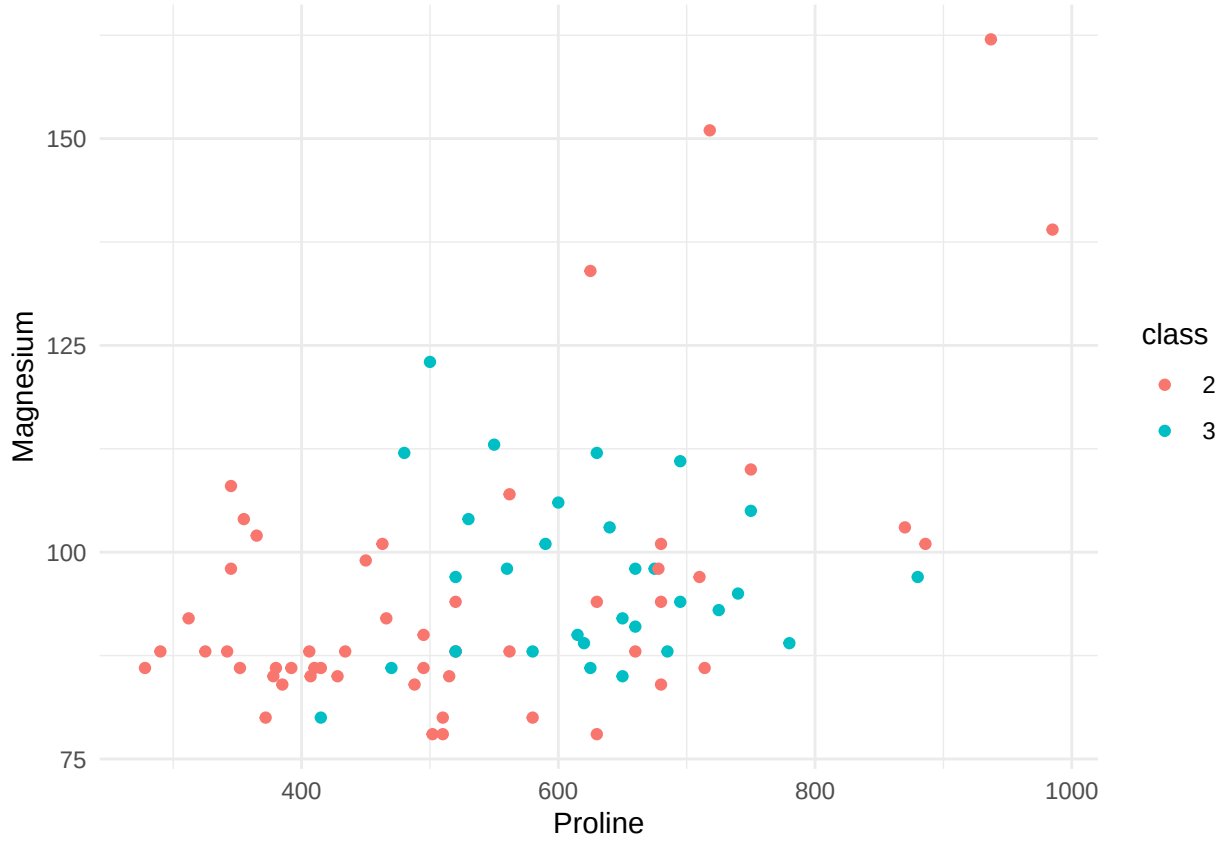
Train 2 SVM classifiers to predict the type of wine using a subset of the other 13 variables.

Based on a comparison to svm classifiers with recall, precision and f1 metrics, our first model with a polyno-



mial kernel outperforms the model with a polynomial kernel in each category.


0
10
20
30
40
50
2
3



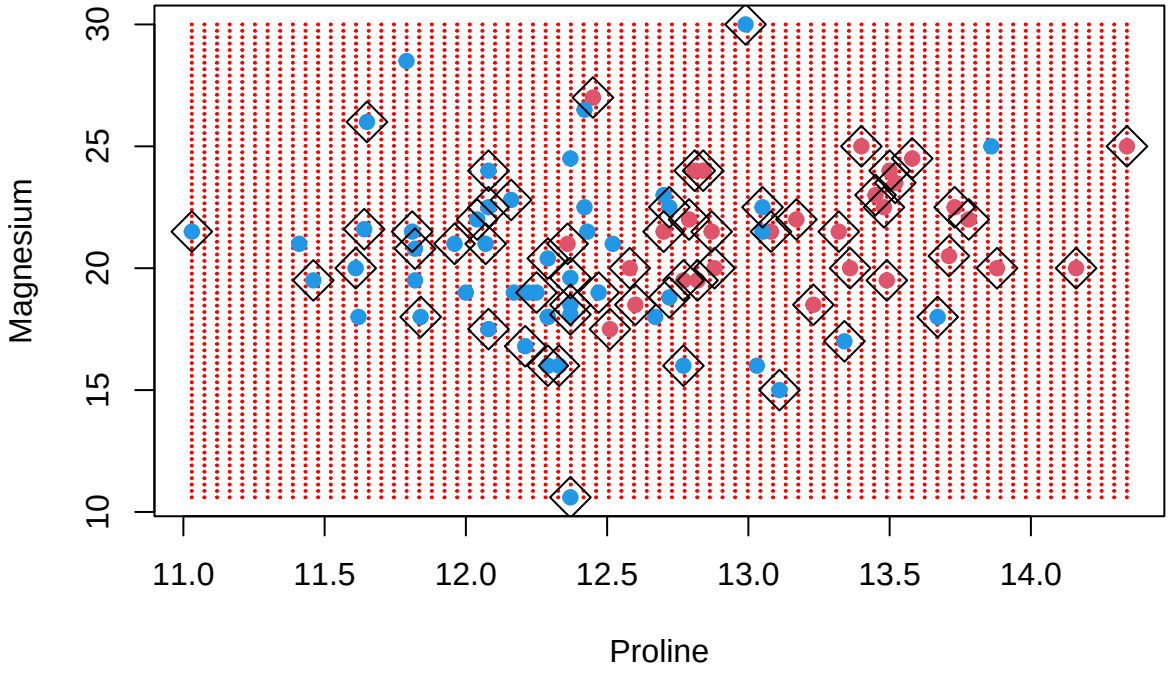

```
##
## Call:
## svm(formula = class ~ Proline + Magnesium, data = train, kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  64
```

```
##       Predicted
## Actual  2  3
##      2 52  0
##      3 31  0
```

```
##   precision_1 recall_1      f1_1
## 2    0.626506        1 0.7703704
## 3         NaN        0       NaN
```

```
##    Proline Magnesium
## 1 11.03000      10.6
## 2 11.07473      10.6
## 3 11.11946      10.6
```

```
## 4 11.16419      10.6
## 5 11.20892      10.6
## 6 11.25365      10.6
## 7 11.29838      10.6
## 8 11.34311      10.6
```



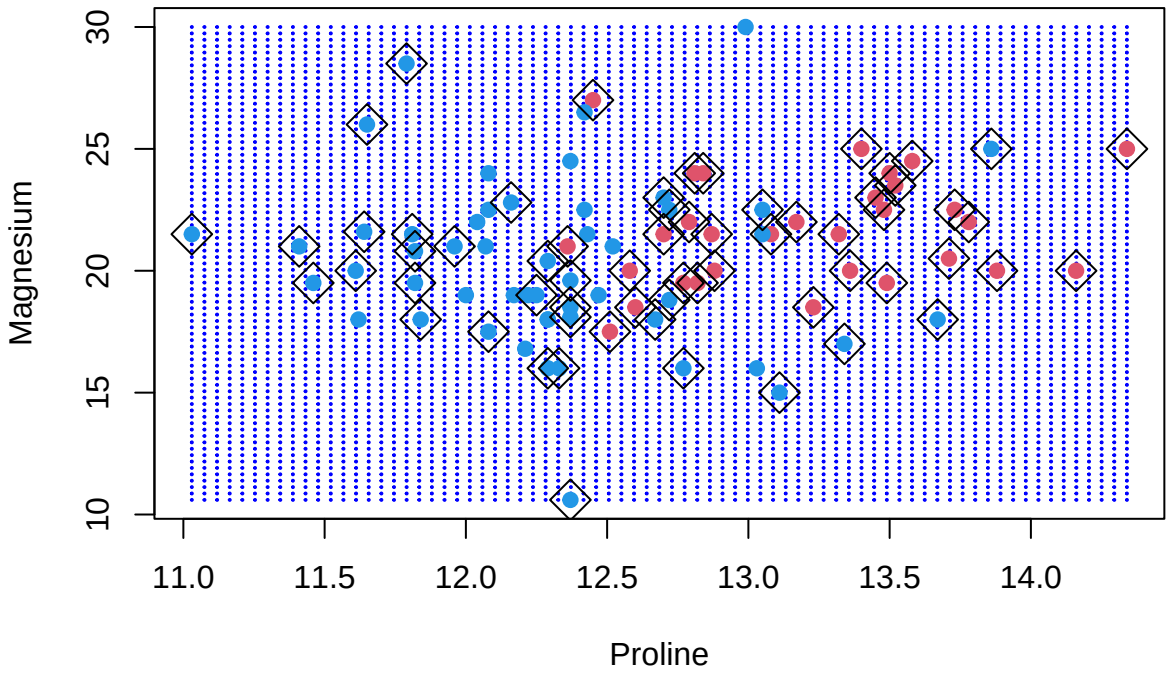

```
##
## Call:
## svm(formula = class ~ Proline + Magnesium, data = train, kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  63

##       Predicted
## Actual  2  3
##      2 50  2
##      3 30  1

##   precision_2   recall_2       f1_2
```

```
## 2   0.6250000 0.96153846 0.75757576
## 3   0.3333333 0.03225806 0.05882353
```

```
##    Proline Magnesium
## 1 11.03000      10.6
## 2 11.07473      10.6
## 3 11.11946      10.6
## 4 11.16419      10.6
## 5 11.20892      10.6
## 6 11.25365      10.6
## 7 11.29838      10.6
## 8 11.34311      10.6
```




```
##   precision_1 precision_2 recall_1   recall_2      f1_1       f1_2
## 2    0.626506   0.6250000        1 0.96153846 0.7703704 0.75757576
## 3         NaN   0.3333333        0 0.03225806       NaN 0.05882353
```

Use tune.svm to find optimum C and Gamma values.

When using the tune function, the model with "optimal parameters" for Gamma and cost do not perform as well as the svm.mod2.

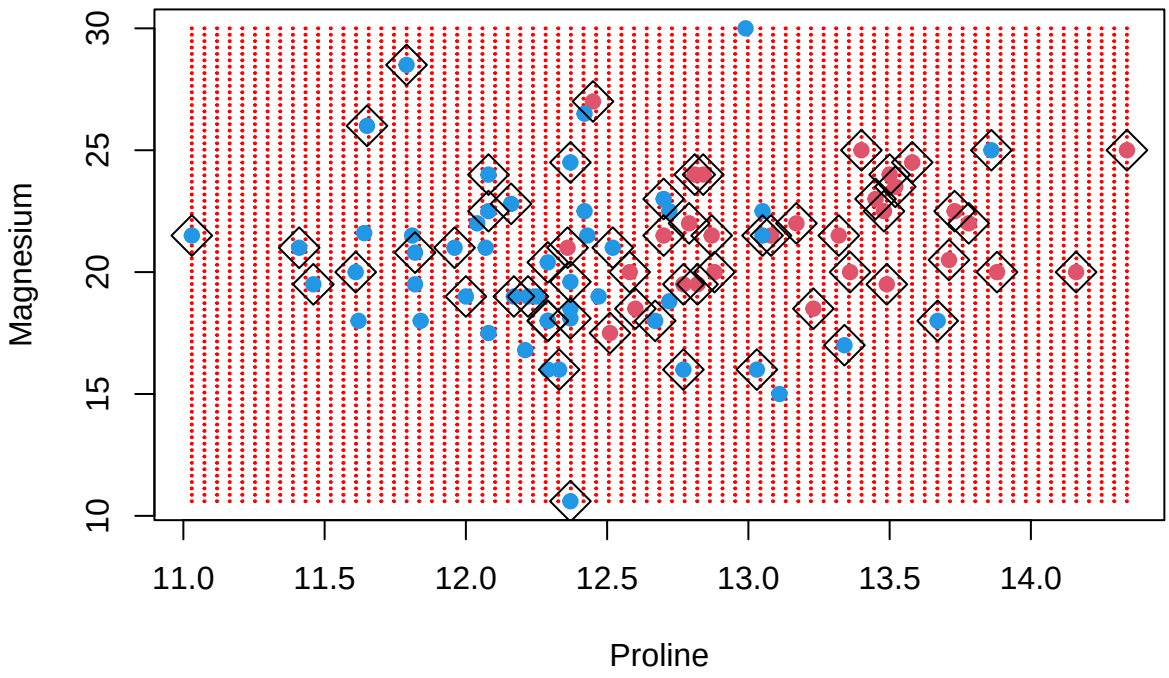
```
##          gamma     cost
## 1 1.033976e-25 0.015625
```

```
##
## Call:
## svm(formula = class ~ Proline + Magnesium, data = train, kernel = "polynomial",
##     gamma = 1.033976e-25, cost = 0.015625)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  0.015625
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  62
```

```
##       Predicted
## Actual  2  3
##      2 52  0
##      3 31  0
```

```
##   precision_3 recall_3      f1_3
## 2    0.626506        1 0.7703704
## 3         NaN        0       NaN
```

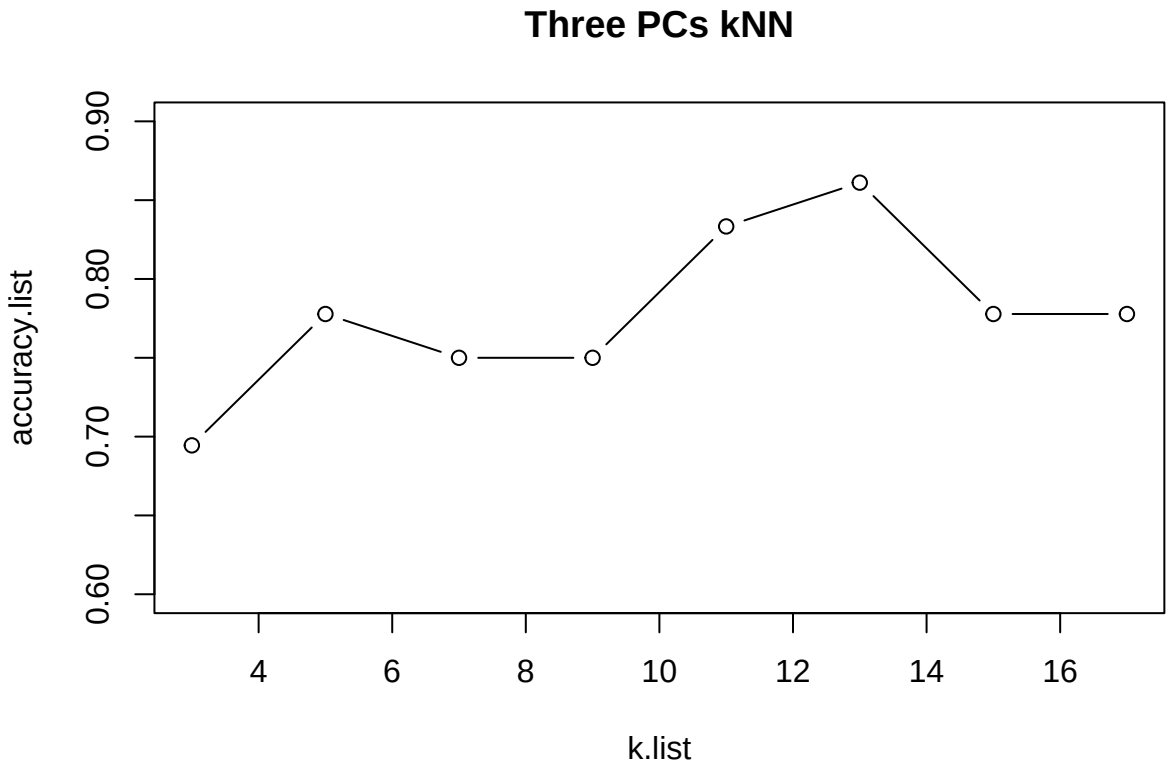
```
##    Proline Magnesium
## 1 11.03000      10.6
## 2 11.07473      10.6
## 3 11.11946      10.6
## 4 11.16419      10.6
## 5 11.20892      10.6
## 6 11.25365      10.6
## 7 11.29838      10.6
## 8 11.34311      10.6
```




```
##   precision_1 precision_2 precision_3 recall_1   recall_2 recall_3      f1_1
## 2    0.626506   0.6250000    0.626506        1 0.96153846        1 0.7703704
## 3         NaN   0.3333333         NaN        0 0.03225806        0       NaN
##         f1_2      f1_3
## 2 0.75757576 0.7703704
## 3 0.05882353       NaN
```

Choose another classification method (kNN, NaiveBayes, etc.) and train a classifier based on the same features.

```
## [1] 8
```




```
## [1] 0.6944444 0.7777778 0.7500000 0.7500000 0.8333333 0.8611111 0.7777778
## [8] 0.7777778

## k is maximum at  13

##          actual
## predicted  2  3
##         2 16  2
##         3  3 15

## [1] 0.8611111

##       Predicted
## Actual  2  3
##      2 16  3
##      3  2 15

## [1] 0.8611111

##    recall_4 precision_4      f1_4
## 2 0.8421053   0.8888889 0.8648649
## 3 0.8823529   0.8333333 0.8571429
```

Compare the performance of the 2 models (Precision, Recall, F1)

The second SVM has the best preformance.

```
##   precision_2 precision_4   recall_2  recall_4       f1_2      f1_4
## 2   0.6250000   0.8888889 0.96153846 0.8421053 0.75757576 0.8648649
## 3   0.3333333   0.8333333 0.03225806 0.8823529 0.05882353 0.8571429
```