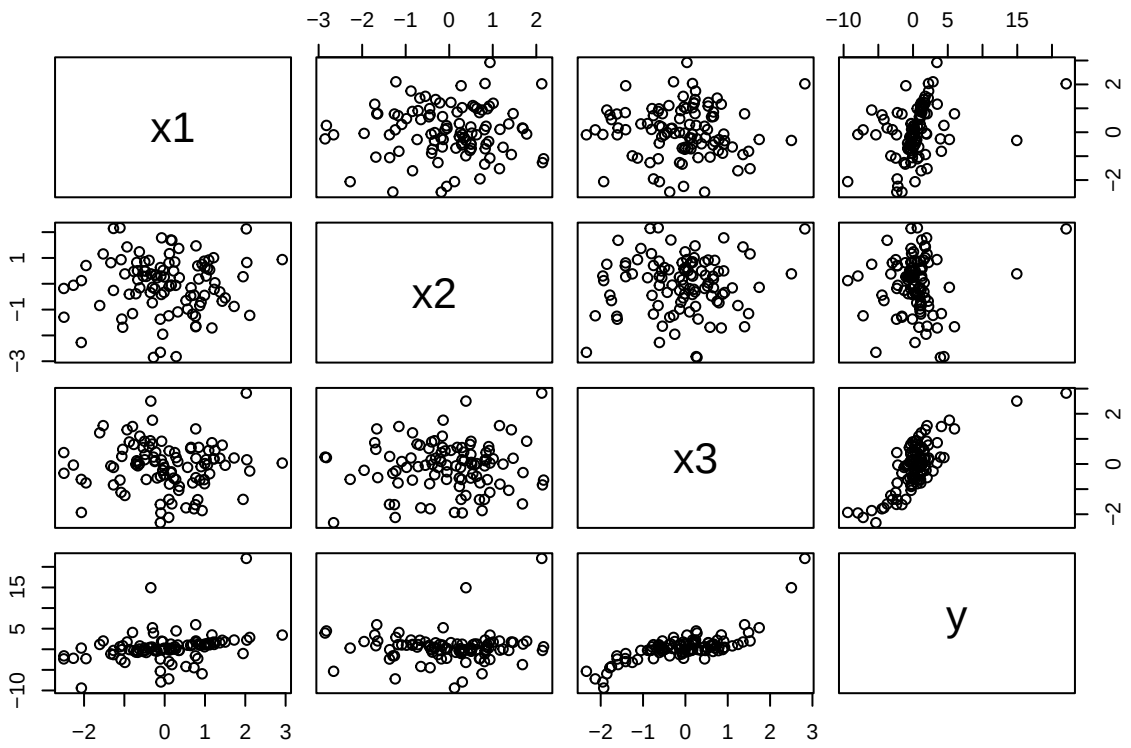
practical 2

qvns53

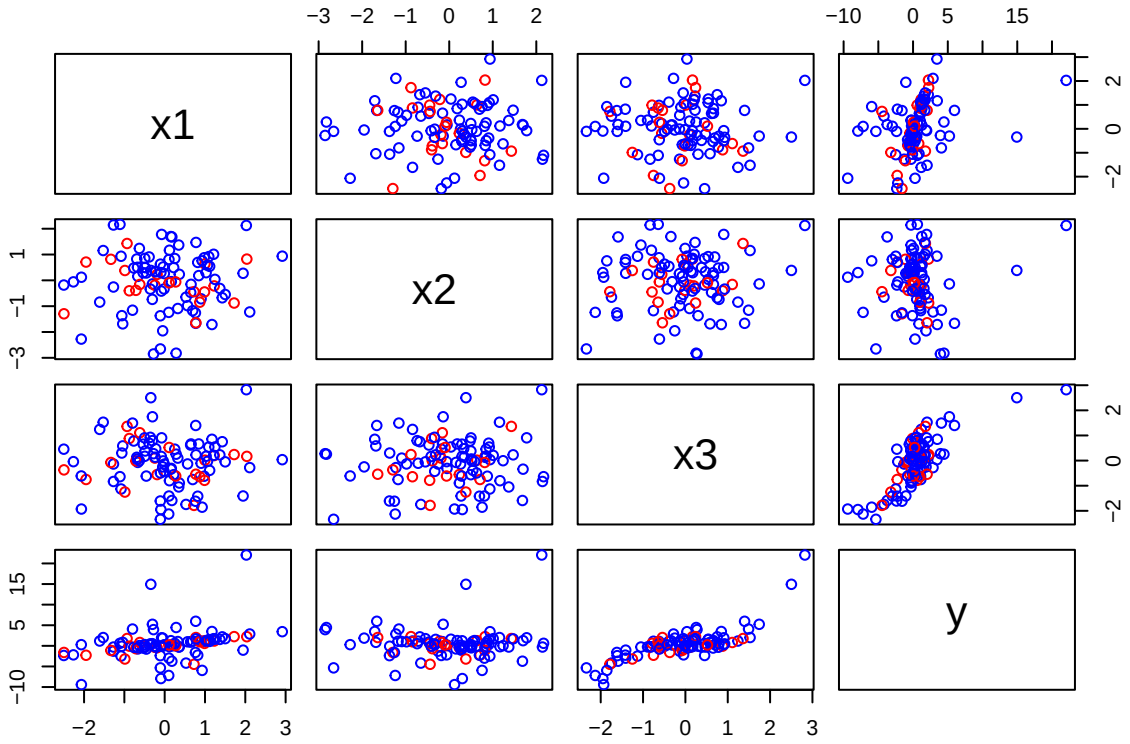
2024-11-07

```r
# Generate synthetic data
data <- data.frame(
  x1 = rnorm(100),
  x2 = rnorm(100),
  x3 = rnorm(100))
data$y = data$x1 + 0.5*data$x2^2 +
  (data$x3-0.1*data$x2)^3 + rnorm(100,0,0.1)

# Pairs plot
pairs(data)
```




```r
# Split the data
train_indices <- sample(1:nrow(data), 0.8*nrow(data))
```

```r
data_train <- data[train_indices, ]
data_test <- data[-train_indices, ]

# Pairs plot colouring by training/test
pairs(data,
      col = ifelse(1:nrow(data) %in% train_indices,
                   "blue", "red"))
```



Task 2

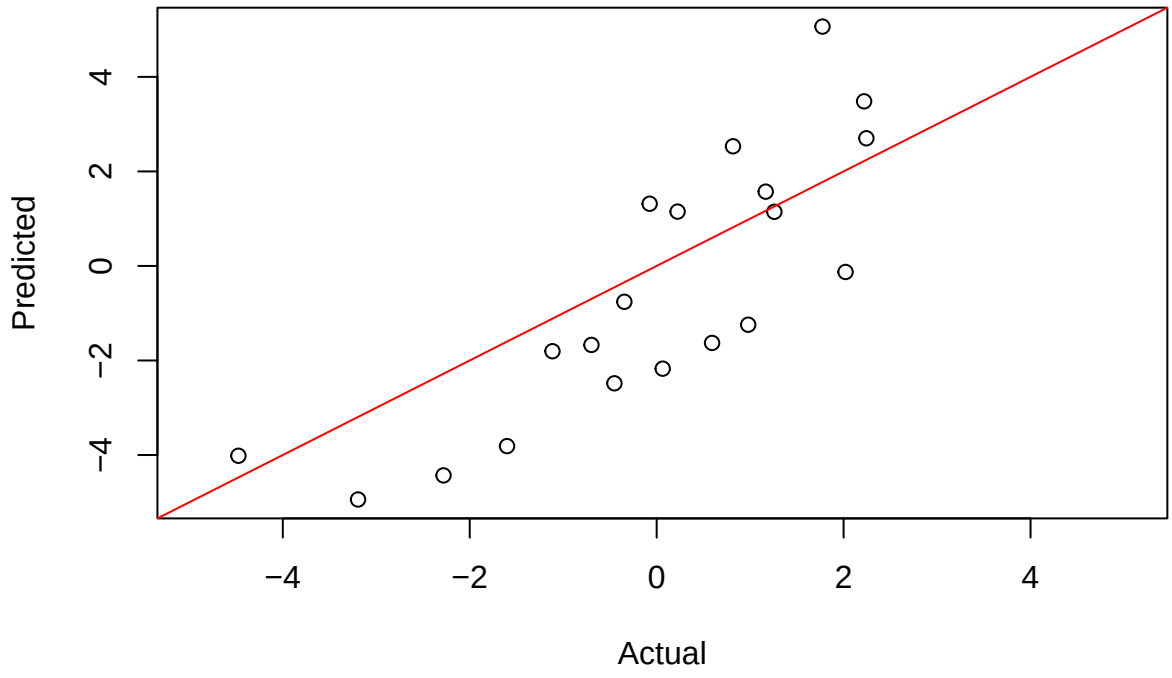
```r
# Fit a polynomial model
model <- lm(y ~ poly(x1, 2) + poly(x2, 2) + poly(x3, 2),
            data = data_train)

# Predict on the test data
predictions <- predict(model, newdata = data_test)

# Calculate MSE
mse <- mean((data_test$y - predictions)^2)
mse
```

```
## [1] 2.82631
```

```r
# Calculate adjusted R-squared
# (Hint it has been calculated in the model summary)
adj_r2 <- summary(model)$adj.r.squared
adj_r2
```

```
## [1] 0.8204106
```

```r
# Plot actual vs predictions
plot(data_test$y, predictions,
     xlab = "Actual",
     ylab = "Predicted",
     xlim = c(min(data_test$y, predictions),
              max(data_test$y, predictions)),
     ylim = c(min(data_test$y, predictions),
              max(data_test$y, predictions)))

abline(a = 0, b = 1, col = "red")
```



Task 3

```r
# Load the caret package
library(caret)
```

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

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

```
# Set up the cross-validation
control <- trainControl(method = "cv",
                        number = 5)

# Fit the model using cross-validation
model_cv <- train(y ~ poly(x1, 2) + poly(x2, 2) + poly(x3, 2),
                  data = data,
                  method = "lm",
                  trControl = control)

# List the elements
names(model_cv)
```

```
##  [1] "method"       "modelInfo"    "modelType"    "results"      "pred"
##  [6] "bestTune"     "call"         "dots"         "metric"       "control"
## [11] "finalModel"   "preProcess"   "trainingData" "ptype"        "resample"
## [16] "resampledCM"  "perfNames"    "maximize"     "yLimits"      "times"
## [21] "levels"       "terms"        "coefnames"    "xlevels"
```

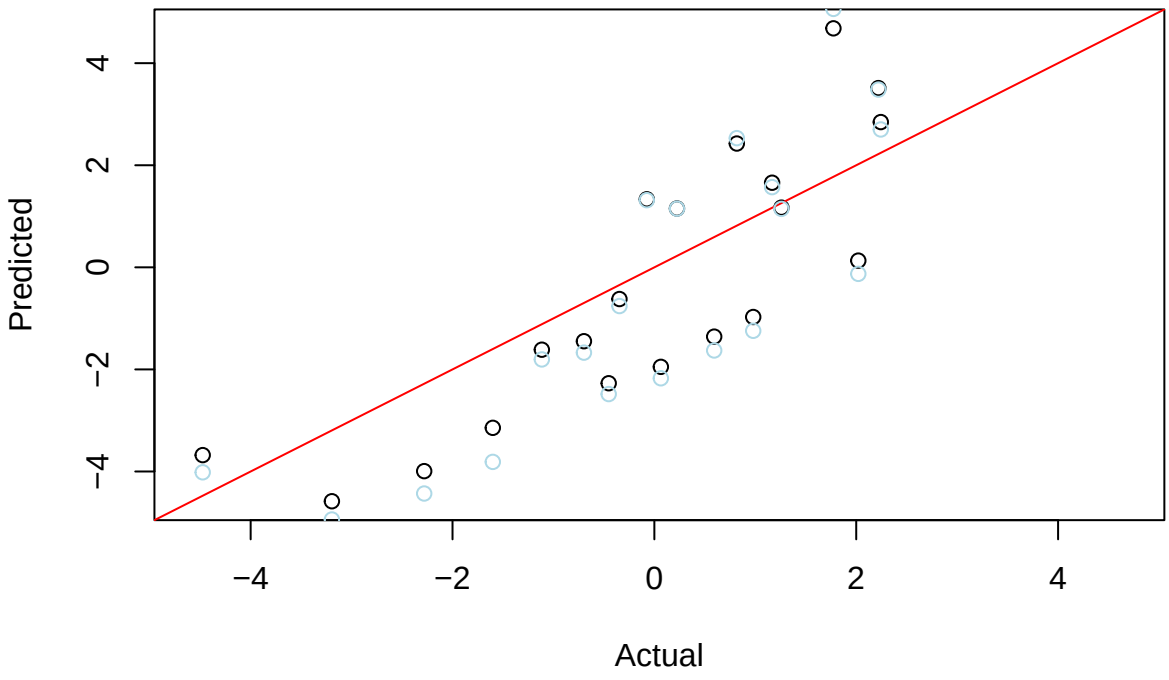
```
# Extract the results
results <- model_cv$results
results
```

```
##   intercept     RMSE  Rsquared      MAE    RMSESD RsquaredSD     MAESD
## 1      TRUE 1.844196 0.7256331 1.403947 0.4544831 0.09654733 0.2544728
```

```
# Hence, calculate the MSE
mse <- results$RMSE^2
mse
```

```
## [1] 3.401057
```

```
# Make predictions on the "test" data
predictions_cv <- predict(model_cv,
                          newdata = data_test)
# Plot actual vs predictions
plot(data_test$y, predictions_cv,
     xlab = "Actual",
     ylab = "Predicted",
     xlim = c(min(data_test$y, predictions_cv),
              max(data_test$y, predictions_cv)),
     ylim = c(min(data_test$y, predictions_cv),
              max(data_test$y, predictions_cv)))
abline(a = 0, b = 1, col = "red")

# Overlay points from initial model
points(data_test$y, predictions, col = "lightblue")
```



Task 4 - repeat with leave-one-out cross-validation

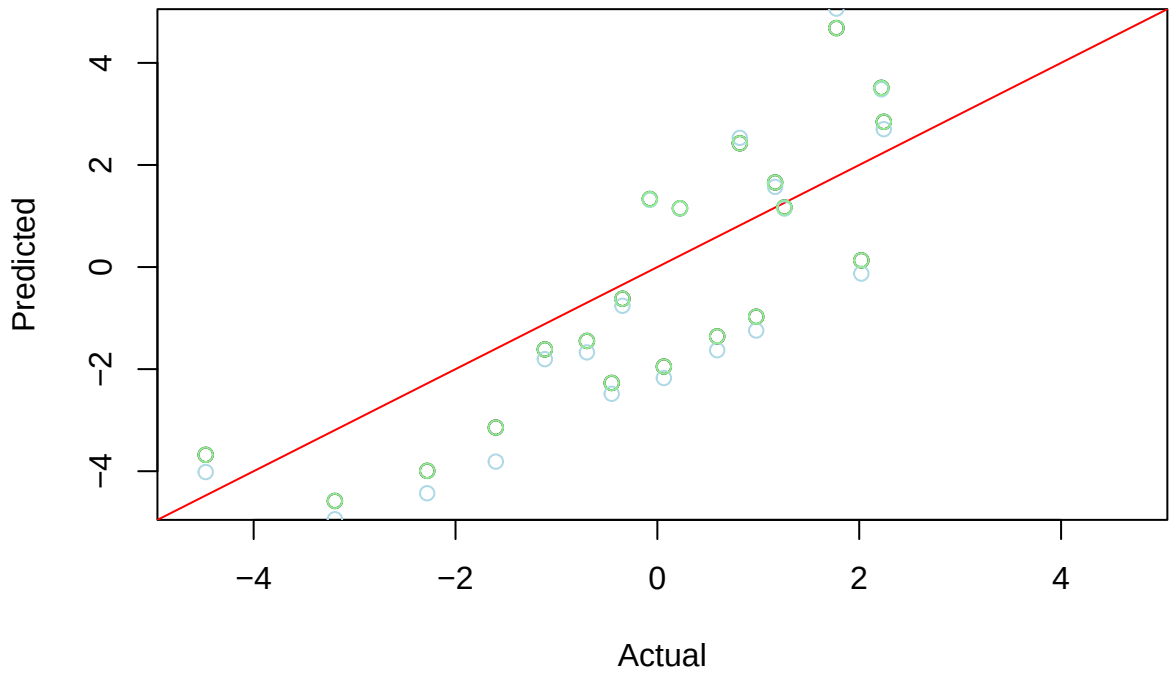
```
# Set up the cross-validation
control <- trainControl(method = "LOOCV")

# Fit the model using leave-one-out cross-validation
model_loocv <- train(y ~ poly(x1, 2) + poly(x2, 2) + poly(x3, 2),
                     data = data,
                     method = "lm",
                     trControl = control)

# Extract the results
model_loocv$results
```

```
##   intercept    RMSE  Rsquared     MAE
## 1      TRUE 2.00263 0.6909937 1.41606
```

```
# Calculate the MSE
mse <- model_loocv$results$RMSE^2
mse
```

```
## [1] 4.010527
```

```r
# Make predictions on the "test" data
predictions_loocv <- predict(model_loocv, data_test)


# Plot actual vs predictions
plot(data_test$y, predictions_loocv,
     xlab = "Actual",
     ylab = "Predicted",
     xlim = c(min(data_test$y, predictions_loocv),
              max(data_test$y, predictions_loocv)),
     ylim = c(min(data_test$y, predictions_loocv),
              max(data_test$y, predictions_loocv)))

abline(0, 1, col="red")


# Overlay points from initial model
points(data_test$y, predictions, col="lightblue")


# Overlay points from 5-fold cross-validation
points(data_test$y, predictions_cv, col="lightgreen")
```



Classification

Task 5 - kNN

```
# Load the data
weather_full <- read.csv("https://www.maths.dur.ac.uk/users/john.p.gosling/MATH3431_practicals/weather_

# Display the first few rows
head(weather_full)
```

```
##   Temperature Humidity Wind.Speed Precipitation....   Cloud.Cover
## 1          14       73        9.5                82 partly cloudy
## 2          39       96        8.5                71 partly cloudy
## 3          30       64        7.0                16         clear
## 4          38       83        1.5                82         clear
## 5          27       74       17.0                66      overcast
## 6          32       55        3.5                26      overcast
##   Atmospheric.Pressure UV.Index Season Visibility..km. Location Weather.Type
## 1              1010.82        2 Winter             3.5   inland        Rainy
## 2              1011.43        7 Spring            10.0   inland       Cloudy
## 3              1018.72        5 Spring             5.5 mountain        Sunny
## 4              1026.25        7 Spring             1.0  coastal        Sunny
## 5               990.67        1 Winter             2.5 mountain        Rainy
## 6              1010.03        2 Summer             5.0   inland       Cloudy
```

```
str(weather_full)
```

```
## 'data.frame':    13200 obs. of  11 variables:
##  $ Temperature         : num  14 39 30 38 27 32 -2 3 3 28 ...
##  $ Humidity            : int  73 96 64 83 74 55 97 85 83 74 ...
##  $ Wind.Speed          : num  9.5 8.5 7 1.5 17 3.5 8 6 6 8.5 ...
##  $ Precipitation....   : num  82 71 16 82 66 26 86 96 66 107 ...
##  $ Cloud.Cover         : chr  "partly cloudy" "partly cloudy" "clear" "clear" ...
##  $ Atmospheric.Pressure: num  1011 1011 1019 1026 991 ...
##  $ UV.Index            : int  2 7 5 7 1 2 1 1 0 8 ...
##  $ Season              : chr  "Winter" "Spring" "Spring" "Spring" ...
##  $ Visibility..km.     : num  3.5 10 5.5 1 2.5 5 4 3.5 1 7.5 ...
##  $ Location            : chr  "inland" "inland" "mountain" "coastal" ...
##  $ Weather.Type        : chr  "Rainy" "Cloudy" "Sunny" "Sunny" ...
```

```
summary(weather_full)
```

```
##   Temperature        Humidity        Wind.Speed     Precipitation....
##  Min.   :-25.00   Min.   : 20.00   Min.   : 0.000   Min.   :  0.00
##  1st Qu.:  4.00   1st Qu.: 57.00   1st Qu.: 5.000   1st Qu.: 19.00
##  Median : 21.00   Median : 70.00   Median : 9.000   Median : 58.00
##  Mean   : 19.13   Mean   : 68.71   Mean   : 9.832   Mean   : 53.64
##  3rd Qu.: 31.00   3rd Qu.: 84.00   3rd Qu.:13.500   3rd Qu.: 82.00
##  Max.   :109.00   Max.   :109.00   Max.   :48.500   Max.   :109.00
##  Cloud.Cover        Atmospheric.Pressure    UV.Index         Season
##  Length:13200       Min.   : 800.1       Min.   : 0.000   Length:13200
```

```
##  Class :character   1st Qu.: 994.8       1st Qu.: 1.000   Class :character
##  Mode  :character   Median :1007.6       Median : 3.000   Mode  :character
##                     Mean   :1005.8       Mean   : 4.006
##                     3rd Qu.:1016.8       3rd Qu.: 7.000
##                     Max.   :1199.2       Max.   :14.000
##  Visibility..km.    Location         Weather.Type
##  Min.   : 0.000   Length:13200       Length:13200
##  1st Qu.: 3.000   Class :character   Class :character
##  Median : 5.000   Mode  :character   Mode  :character
##  Mean   : 5.463
##  3rd Qu.: 7.500
##  Max.   :20.000
```

```
# Select the features of interest
weather <- weather_full[ , c(8,1,3,4,6)]

# Convert the season to a factor
weather$Season <- as.factor(weather$Season)

# Summarise the data
summary(weather)
```

```
##     Season      Temperature       Wind.Speed     Precipitation....
##  Autumn:2500   Min.   :-25.00   Min.   : 0.000   Min.   :  0.00
##  Spring:2598   1st Qu.:  4.00   1st Qu.: 5.000   1st Qu.: 19.00
##  Summer:2492   Median : 21.00   Median : 9.000   Median : 58.00
##  Winter:5610   Mean   : 19.13   Mean   : 9.832   Mean   : 53.64
##                3rd Qu.: 31.00   3rd Qu.:13.500   3rd Qu.: 82.00
##                Max.   :109.00   Max.   :48.500   Max.   :109.00
##  Atmospheric.Pressure
##  Min.   : 800.1
##  1st Qu.: 994.8
##  Median :1007.6
##  Mean   :1005.8
##  3rd Qu.:1016.8
##  Max.   :1199.2
```

```
str(weather)
```

```
## 'data.frame':    13200 obs. of  5 variables:
##  $ Season              : Factor w/ 4 levels "Autumn","Spring",..: 4 2 2 2 4 3 4 4 4 4 ...
##  $ Temperature         : num  14 39 30 38 27 32 -2 3 3 28 ...
##  $ Wind.Speed          : num  9.5 8.5 7 1.5 17 3.5 8 6 6 8.5 ...
##  $ Precipitation....   : num  82 71 16 82 66 26 86 96 66 107 ...
##  $ Atmospheric.Pressure: num  1011 1011 1019 1026 991 ...
```

```
head(weather)
```

```
##   Season Temperature Wind.Speed Precipitation.... Atmospheric.Pressure
## 1 Winter          14        9.5                82              1010.82
## 2 Spring          39        8.5                71              1011.43
## 3 Spring          30        7.0                16              1018.72
```

```
## 4 Spring          38        1.5                82             1026.25
## 5 Winter          27       17.0                66              990.67
## 6 Summer          32        3.5                26             1010.03
```

```r
set.seed(123)

# Split the data
train_indices <- sample(1:nrow(weather), 0.8*nrow(weather))
weather_train <- weather[train_indices, ]
weather_test <- weather[-train_indices, ]

# Fit a k-nearest neighbours model
model_knn <- train(Season ~ .,
                   data = weather_train,
                   method = "knn",
                   trControl = trainControl(method = "cv",
                                            number = 10))

# Extract the results
model_knn$results
```

```
##   k  Accuracy     Kappa  AccuracySD    KappaSD
## 1 5 0.4190361 0.1764632 0.013756183 0.01864285
## 2 7 0.4135425 0.1673106 0.012058863 0.01566726
## 3 9 0.4150555 0.1692315 0.009676837 0.01307511
```

```r
# Make predictions on the test data
predictions_knn_weather <- predict(model_knn, weather_test)

# Calculate the confusion matrix
confusionMatrix(predictions_knn_weather,
                weather_test$Season)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Autumn Spring Summer Winter
##     Autumn    121    121    123    123
##     Spring    117    148    136    115
##     Summer    109    143    101     98
##     Winter    150    155    128    752
##
## Overall Statistics
##
##                Accuracy : 0.425
##                  95% CI : (0.406, 0.4441)
##     No Information Rate : 0.4121
##     P-Value [Acc > NIR] : 0.09279
##
##                   Kappa : 0.1863
##
##  Mcnemar's Test P-Value : 0.03359
##
```

```
## Statistics by Class:
##
##                      Class: Autumn Class: Spring Class: Summer Class: Winter
## Sensitivity                0.24346       0.26102       0.20697        0.6912
## Specificity                0.82874       0.82248       0.83736        0.7210
## Pos Pred Value             0.24795       0.28682       0.22395        0.6346
## Neg Pred Value             0.82528       0.80273       0.82321        0.7691
## Prevalence                 0.18826       0.21477       0.18485        0.4121
## Detection Rate             0.04583       0.05606       0.03826        0.2848
## Detection Prevalence       0.18485       0.19545       0.17083        0.4489
## Balanced Accuracy          0.53610       0.54175       0.52216        0.7061
```

Task 6 - naive Bayes

```r
# Fit a naive Bayes model
model_nb <- train(Season ~ .,
                  data = weather_train,
                  method = "naive_bayes",
                  trControl = trainControl(method = "cv",
                                           number = 10))

# Extract the results
model_nb$results
```

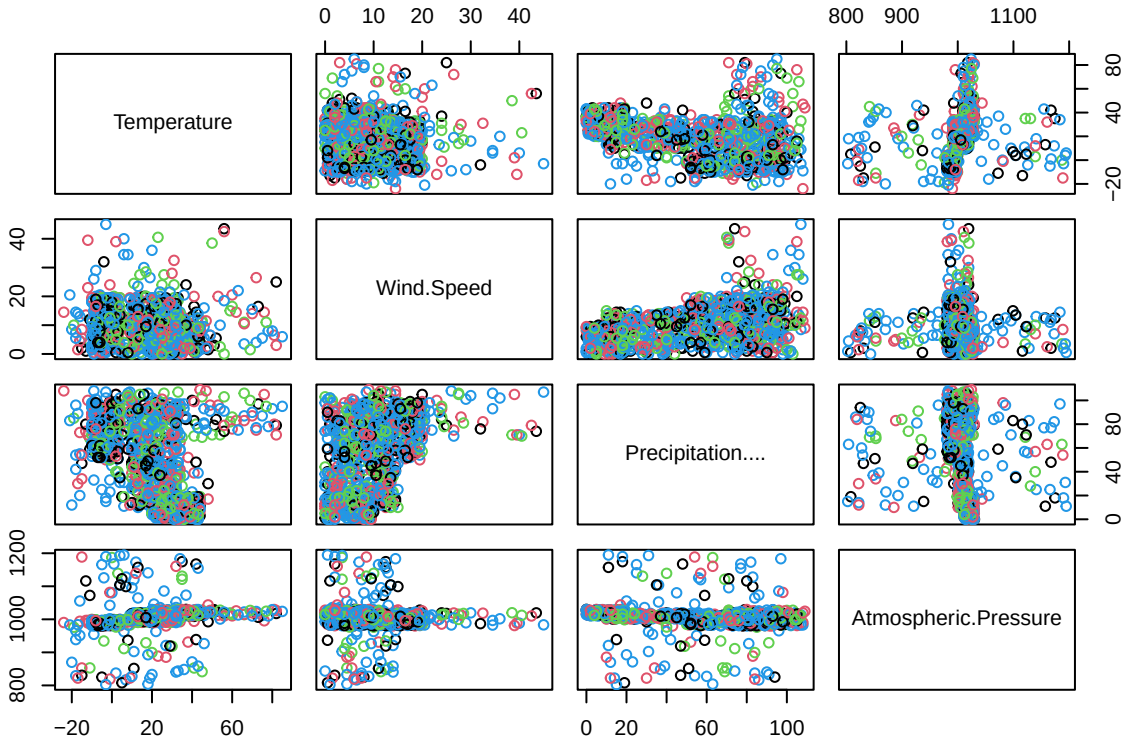
```
##   usekernel laplace adjust  Accuracy     Kappa  AccuracySD    KappaSD
## 1     FALSE       0      1 0.4261357 0.1233466 0.009247048 0.01448711
## 2      TRUE       0      1 0.4171392 0.1783374 0.014763637 0.02182833
```

```r
# Make predictions on the test data
predictions_nb <- predict(model_nb, weather_test)

# Calculate the confusion matrix
confusionMatrix(predictions_nb,
                weather_test$Season)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Autumn Spring Summer Winter
##     Autumn      1      0      0      0
##     Spring     32     49     43     41
##     Summer    193    194    180    170
##     Winter    271    324    265    877
##
## Overall Statistics
##
##                Accuracy : 0.4193
##                  95% CI : (0.4004, 0.4384)
##     No Information Rate : 0.4121
##     P-Value [Acc > NIR] : 0.2321
##
```

```
##                   Kappa : 0.1251
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: Autumn Class: Spring Class: Summer Class: Winter
## Sensitivity              0.0020121       0.08642       0.36885        0.8061
## Specificity              1.0000000       0.94404       0.74117        0.4459
## Pos Pred Value           1.0000000       0.29697       0.24423        0.5049
## Neg Pred Value           0.8120500       0.79071       0.83815        0.7663
## Prevalence               0.1882576       0.21477       0.18485        0.4121
## Detection Rate           0.0003788       0.01856       0.06818        0.3322
## Detection Prevalence     0.0003788       0.06250       0.27917        0.6580
## Balanced Accuracy        0.5010060       0.51523       0.55501        0.6260
```

```
# Pairs plot coloured by season
pairs(weather[sample(1:nrow(weather),1000),2:5],
      col = weather_train$Season)
```




```
# Also consider the names of the full set of variables
names(weather_full)
```

```
##  [1] "Temperature"          "Humidity"             "Wind.Speed"
##  [4] "Precipitation...."    "Cloud.Cover"          "Atmospheric.Pressure"
```

```
##  [7] "UV.Index"             "Season"               "Visibility..km."
## [10] "Location"             "Weather.Type"
```

Task 7 - Decision Trees

```r
# Load the data
mushroom <- read.csv("https://www.maths.dur.ac.uk/users/john.p.gosling/MATH3431_practicals/mushrooms.csv

# Display the first few rows
head(mushroom)
```

```
##   class cap.shape cap.surface cap.color bruises odor gill.attachment
## 1     p         x           s         n       t    p               f
## 2     e         x           s         y       t    a               f
## 3     e         b           s         w       t    l               f
## 4     p         x           y         w       t    p               f
## 5     e         x           s         g       f    n               f
## 6     e         x           y         y       t    a               f
##   gill.spacing gill.size gill.color stalk.shape stalk.root
## 1            c         n          k           e          e
## 2            c         b          k           e          c
## 3            c         b          n           e          c
## 4            c         n          n           e          e
## 5            w         b          k           t          e
## 6            c         b          n           e          c
##   stalk.surface.above.ring stalk.surface.below.ring stalk.color.above.ring
## 1                        s                        s                      w
## 2                        s                        s                      w
## 3                        s                        s                      w
## 4                        s                        s                      w
## 5                        s                        s                      w
## 6                        s                        s                      w
##   stalk.color.below.ring veil.type veil.color ring.number ring.type
## 1                      w         p          w           o         p
## 2                      w         p          w           o         p
## 3                      w         p          w           o         p
## 4                      w         p          w           o         p
## 5                      w         p          w           o         e
## 6                      w         p          w           o         p
##   spore.print.color population habitat
## 1                 k          s       u
## 2                 n          n       g
## 3                 n          n       m
## 4                 k          s       u
## 5                 n          a       g
## 6                 k          n       g
```

```r
# Convert all variables to factors and put back into the data frame
mushroom <- lapply(mushroom, as.factor)
mushroom <- as.data.frame(mushroom)

# Remove all but the first four predictors
```

```r
# to aid interpretation of the tree
mushroom <- mushroom[,1:5]

# Summarise the data
summary(mushroom)
```

```
##  class    cap.shape cap.surface   cap.color    bruises
##  e:4208   b: 452    f:2320      n      :2284   f:4748
##  p:3916   c:   4    g:   4      g      :1840   t:3376
##           f:3152    s:2556      e      :1500
##           k: 828    y:3244      y      :1072
##           s:  32                w      :1040
##           x:3656                b      : 168
##                                 (Other): 220
```

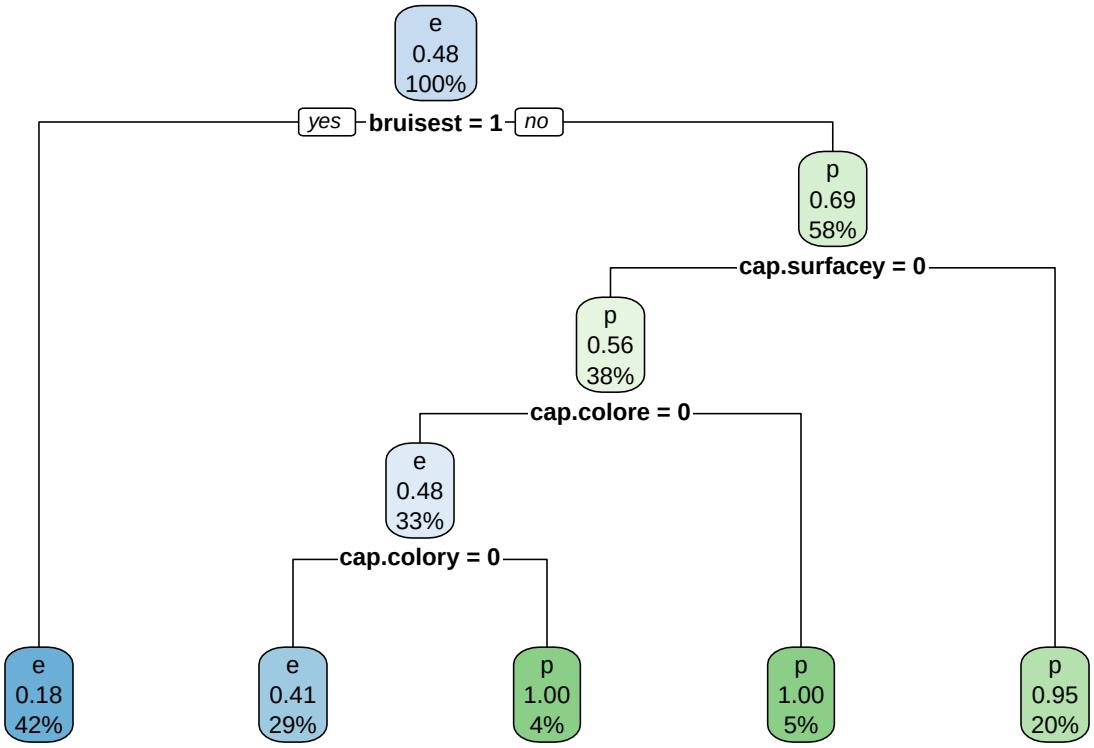
```r
str(mushroom)
```

```
## 'data.frame':    8124 obs. of  5 variables:
##  $ class      : Factor w/ 2 levels "e","p": 2 1 1 2 1 1 1 1 2 1 ...
##  $ cap.shape  : Factor w/ 6 levels "b","c","f","k",..: 6 6 1 6 6 6 1 1 6 1 ...
##  $ cap.surface: Factor w/ 4 levels "f","g","s","y": 3 3 3 4 3 4 3 4 4 3 ...
##  $ cap.color  : Factor w/ 10 levels "b","c","e","g",..: 5 10 9 9 4 10 9 9 9 10 ...
##  $ bruises    : Factor w/ 2 levels "f","t": 2 2 2 2 1 2 2 2 2 2 ...
```

```r
head(mushroom)
```

```
##   class cap.shape cap.surface cap.color bruises
## 1     p         x           s         n       t
## 2     e         x           s         y       t
## 3     e         b           s         w       t
## 4     p         x           y         w       t
## 5     e         x           s         g       f
## 6     e         x           y         y       t
```

```r
# Load the `rpart` package
library(rpart)

# Fit a decision tree using 5-fold cross-validation
model_tree <- train(class ~ .,
                    data = mushroom,
                    method = "rpart",
                    trControl = trainControl(method = "cv",
                                             number = 5))

# Load the `rpart.plot` package
library(rpart.plot)

# Plot the tree
rpart.plot(model_tree$finalModel)
```


e
0.48
100%
yes
bruisest = 1
no
p
0.69
58%
cap.surfacey = 0
p
0.56
38%
cap.colore = 0
e
0.48
33%
cap.colory = 0
e
0.18
42%
e
0.41
29%
p
1.00
4%
p
1.00
5%
p
0.95
20%