Weekly Summary Template

Brady Miller

Table of contents

Tuesday, March 21

> **!** TIL
>
> Include a *very brief* summary of what you learnt in this class here.
> Today, I learnt the following concepts in class:
>
> 1. Creating decision boundaries
> 2. Creating and interpreting a confusion matrix
> 3. Multinomial Class Logistic Regression

Decision Boundaries

```
library(mlbench)
```

```
Warning: package 'mlbench' was built under R version 4.2.3
```

```
library(tidyverse)
```

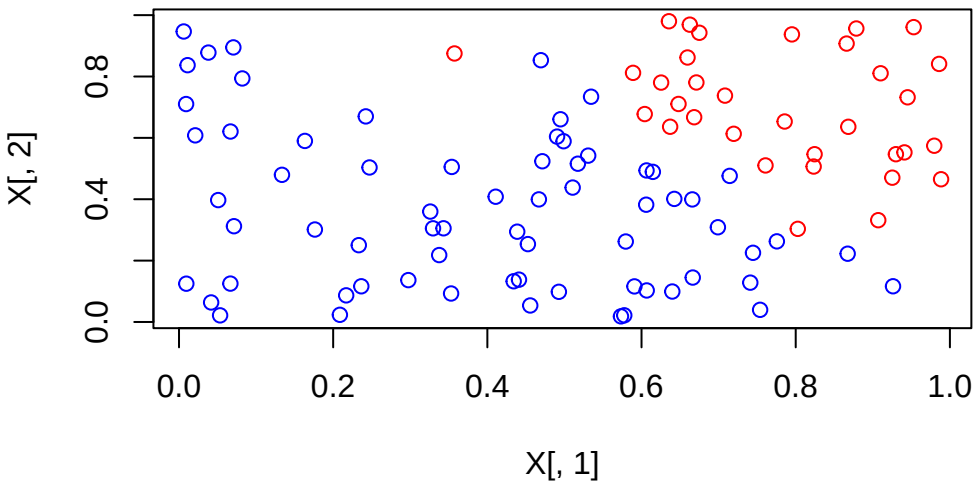
```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.0     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.1     v tibble    3.2.0
v lubridate 1.9.2     v tidyr     1.3.0
v purrr     1.0.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```r
library(nnet)
```

```
Warning: package 'nnet' was built under R version 4.2.3
```

```r
library(class)
```

```r
# Creating random scatterplot to create boundary on
X <- t(replicate(100,runif(2)))
y <- ifelse(apply(X,1,\(x)sum(x^1.5)) + 0.1 * rnorm(1000) <= 1, 0, 1) %>% as.factor()
col <- ifelse( y == 0, "blue", "red")

plot(X[,1], X[,2], col=col)
```



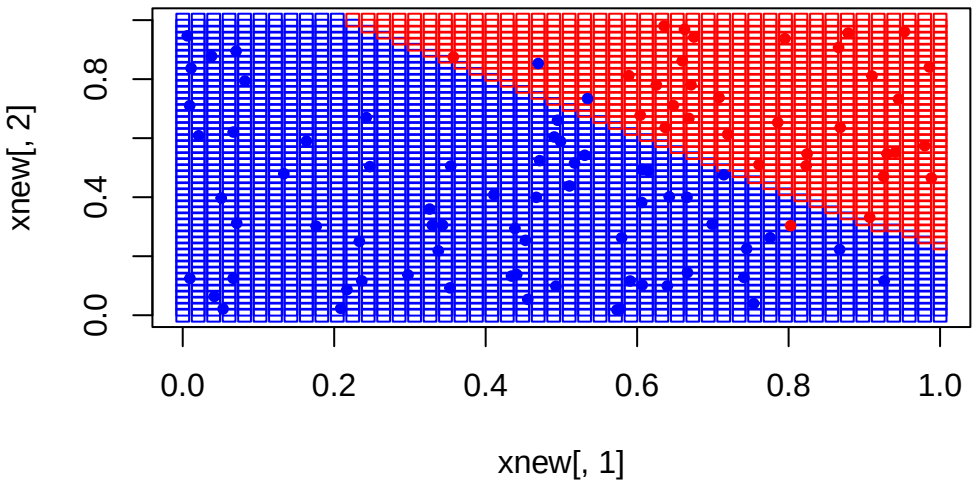

```r
df <- data.frame(y=y, x1 = X[,1], x2 = X[,2])
model <- glm(y~., df, family = binomial())
summary(model)
```

```
Call:
glm(formula = y ~ ., family = binomial(), data = df)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.16495  -0.08142  -0.00758   0.04797   2.97871

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -20.211      1.657  -12.20   <2e-16 ***
x1            16.485      1.486   11.10   <2e-16 ***
x2            16.568      1.388   11.94   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1262.6  on 999  degrees of freedom
```

```
Residual deviance:  256.8  on 997  degrees of freedom
AIC: 262.8

Number of Fisher Scoring iterations: 8
```

The summary of the model indicates that the slopes and intercepts are highly significant

```r
# creating a model to show for the logistic regression creates a boundary
# separates the 2 classes and shows what it predicts at each spot
xnew <- data.frame(
  x1 = rep(seq(0,1,length.out=50),50),
  x2 = rep(seq(0,1,length.out=50),each = 50)
)

prob <- predict(model, xnew, type = 'response')
decision <- ifelse(prob<0.5, 'blue', 'red')

plot(xnew[,1], xnew[,2], col=decision, pch=22)
points(X[,1], X[,2], col=col, pch=20)
```



The decision boundary model is pretty accurate in classifying the points. There are a few points that are misclassified but that is to be expected, as overfitting the decision boundary

may decrease accuracy on test data

Confusion matrix

```
idx <- sample(1:nrow(df), 100)
train <- df[-idx,]
test<- df[idx,]
model <- glm(y~., train, family = binomial())
probs <- predict(model, test, type = 'response')

predicted <- ifelse(probs<0.5, 0, 1)
expected <- test$y


table(predicted, expected)
```

```
         expected
predicted  0  1
        0 61  1
        1  4 34
```

How model performs on unseen data set

- 2x2 table for binary classification problem
- columns are ground truth
- first element = out of 69 people with y value 0 in test, model was able to correctly predict 65 of them
- out of 31 people in class 1, 29 were correctly predicted to be in class 1
- take into consideration how many false negatives/false positives your model gives when determining its goodness of fit
- want 0's in the bottom left and top right columns to indicate that the model predicted the label of each data sample in the test set with 100% accuracy

```
caret::confusionMatrix(data = as.factor(predicted), reference = as.factor(expected))
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 61  1
```

```
         1  4 34

               Accuracy : 0.95
                 95% CI : (0.8872, 0.9836)
    No Information Rate : 0.65
    P-Value [Acc > NIR] : 7.363e-13

                  Kappa : 0.8922

 Mcnemar's Test P-Value : 0.3711

            Sensitivity : 0.9385
            Specificity : 0.9714
         Pos Pred Value : 0.9839
         Neg Pred Value : 0.8947
             Prevalence : 0.6500
         Detection Rate : 0.6100
   Detection Prevalence : 0.6200
      Balanced Accuracy : 0.9549

       'Positive' Class : 0
```

- gives confidence interval, p-value and other statistics that we can look at

1. Below is the elements of the confusion matrix used to calculate sensitivity and specificity

- sensitivity = [1,1] / [1,1] + [2,1]
- specificity = [2,2] / [1,2] + [2,2]

Multinomial Class Logistic Regression

- generalization of logistic regression for more than two 'classes'
- pick base category

    1. constructs k-1 different logistic regression models for k categories of a response variable

- use softmax function for multinomial logistic regression

1. exponentiating every element and then normalizing it to be between 0 and 1 by dividing it by sum of all the exponentiated classes
2. weighted summation
3. sum of all exponentiated elements is 1

4. select max value to determine which class a point belongs to
5. 0 <= exponentiated values <= 1
6. one-hot encoding argMax(x1,x2,…,xi,…,xk) = (0,0,…,1,…,0)

- softmax(0,1) = (1/(1+exp(x), exp(x)/(1+exp(x))) = (1-sigmoid(x), sigmoid(x))

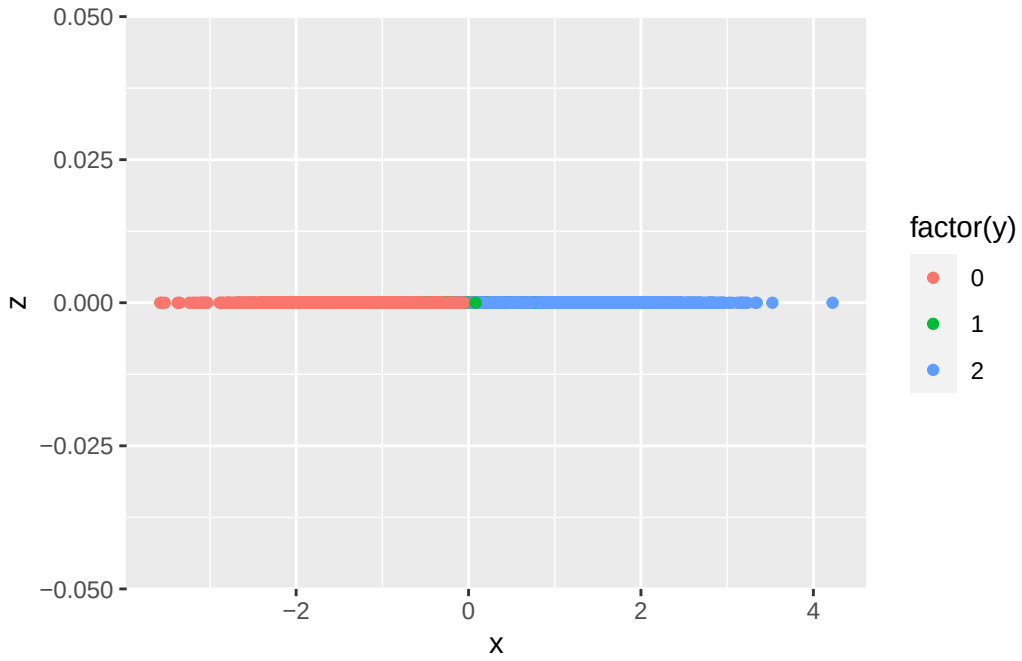
```
# create a multinomial classification problem
# want to classify a flower as one of 3 species based on petal width and petal length

# fix one level as reference level and then make 2 different models

sample(1:3, size = 1000, replace = TRUE, prob=c(0.8,0.1,0.1))
```

```
   [1] 1 1 1 1 1 1 2 1 1 3 1 1 1 1 1 2 3 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 3 1 1 2
  [38] 1 3 1 1 1 3 3 1 1 1 1 1 1 1 1 1 3 1 1 3 3 1 1 2 1 1 1 1 1 1 1 1 1 3 2 1
  [75] 1 1 1 2 1 1 1 1 2 1 1 1 3 3 1 1 3 1 1 1 1 1 1 2 1 1 1 1 1 1 1 3 1 2 1 3 2
 [112] 2 1 1 2 1 1 2 2 1 3 1 3 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 2 3 1 3 3 1 2
 [149] 3 1 1 1 3 1 1 1 1 1 1 1 3 1 3 2 3 1 1 2 1 1 1 1 1 1 1 1 3 1 1 1 1 1 3 1 2
 [186] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 3 1 1 1 2 1 1 1 1 1 1
 [223] 2 1 1 1 1 3 1 3 2 2 3 1 1 3 1 3 2 3 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 3 2 1 1
 [260] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 3 3 3 1 1 1 1 3
 [297] 1 1 3 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 3 1 1 2 2 1 2 1 1 1 1 1 1 1 2
 [334] 1 1 1 2 1 2 1 1 1 1 1 1 1 2 1 3 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [371] 1 1 1 3 3 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 3 1 1 3 2 1 1 1 1 1 1 1 1 1
 [408] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 2 1 3 1 1 1 3 1 1 1 1 1 1 1 2 1 1
 [445] 1 1 1 1 1 2 2 1 2 1 1 2 1 2 1 1 3 1 1 1 2 1 1 2 1 1 1 1 3 1 1 3 1 1 1 2 1
 [482] 2 1 2 1 1 1 1 1 1 1 1 1 2 1 1 3 3 1 1 1 3 1 3 1 1 1 1 1 3 2 1 1 1 1 1 1 2
 [519] 3 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 3 1 1 2 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1
 [556] 1 1 1 1 1 2 1 2 1 1 3 2 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 3
 [593] 1 3 1 1 2 1 2 1 1 3 2 3 1 1 2 3 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 3 1 1 1 3 2
 [630] 3 1 1 2 1 1 1 1 1 1 1 1 3 1 1 1 1 1 2 1 1 1 1 2 2 1 1 1 1 1 3 1 3 1 1 2 1
 [667] 1 1 1 1 1 1 1 1 2 1 2 2 2 3 3 1 1 1 2 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 2
 [704] 3 1 1 1 3 1 2 1 1 1 1 1 1 1 1 2 1 3 1 2 1 1 1 1 1 1 1 1 1 1 2 2 3 1 3 1 1
 [741] 1 3 2 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 3 1 1 2 1 1 3 1 1 1 3 1 1 1 1 1 1
 [778] 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [815] 1 1 1 1 1 1 1 3 1 1 1 1 1 2 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 2 1 1 2 3 3
 [852] 2 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 3 1 1 1 2 1 1 1 1 1 1 1 1 1 3 2 1 1 1
 [889] 1 1 1 1 3 1 1 1 1 1 3 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1
 [926] 1 2 1 1 3 1 1 3 1 3 2 1 2 1 1 1 1 3 3 1 1 1 1 1 1 1 3 1 1 1 1 3 3 2 3 1 3
 [963] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 3 3 3 1 2 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1
[1000] 1
```

```r
b <- c(-5,0,5)
prob_function = \(x) exp(b*x) / sum(exp(b*x))

x <- rnorm(10000)
y <- c()

# sampling class labels from the data set
for (i in 1:length(x)){
  y[i] <- sample(0:2, 1, prob = prob_function(x[i]))
}

cbind(x, y) %>% head
```

```
              x y
[1,] -0.05157392 2
[2,]  0.90188319 2
[3,] -1.10706508 0
[4,]  1.19844797 2
[5,] -1.08419024 0
[6,]  0.50291134 2
```

```r
data.frame(x=x, z = rep(0, length(x)), y=y) %>% ggplot(aes(x=x, y = z, color=factor(y))) +
```




```
# setting level 1 as the reference level
# will compare this individually against the other class levels
df <- data.frame(x=x, y=as.factor(y))
df$y <- relevel(df$y, ref = '1')
```

```
model <- nnet::multinom(y ~ x, df)
```

```
# weights:  9 (4 variable)
initial  value 10986.122887
iter  10 value 3285.392682
final  value 3285.156339
converged
```

```
summary(model)
```

```
Call:
nnet::multinom(formula = y ~ x, data = df)

Coefficients:
  (Intercept)         x
```

```
0  0.03288802 -4.892449
2  0.01532239  4.936018

Std. Errors:
  (Intercept)         x
0  0.05413537 0.1662174
2  0.05435110 0.1674048

Residual Deviance: 6570.313
AIC: 6578.313
```

- coefficients in the first block are telling us that the intercept and slope for the regression model of 0 vs 1 and 1 vs 2.
- slope will flip signs when you switch the class level that is being used as reference level

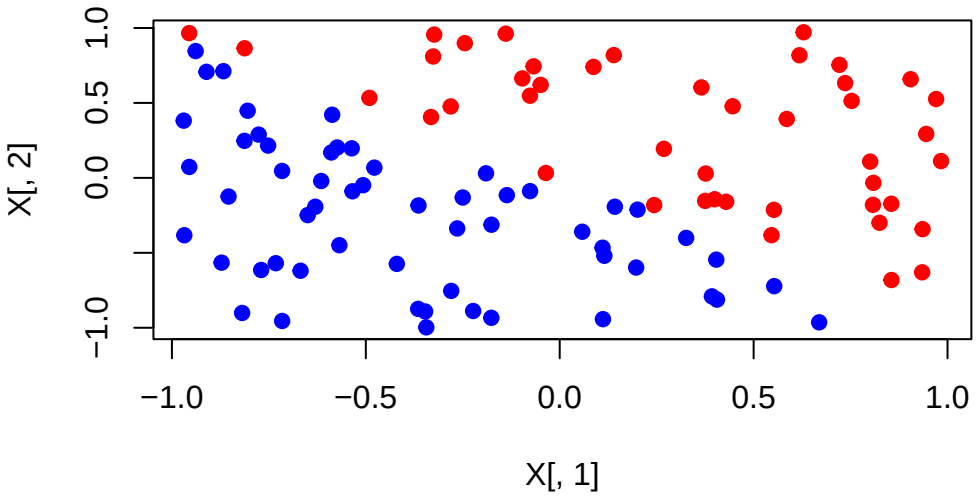
Thursday, March 23

> **!** TIL
>
> Include a *very brief* summary of what you learnt in this class here.
> Today, I learnt the following concepts in class:
>
> 1. More with creating decision boundaries
> 2. Creating and interpreting classification(decision) trees
> 3. Support vector machines

Creating more decision boundaries

- logistic regression is useful when you want to see how a covariate is affecting a response variable

```r
X <- t(replicate(100,2*runif(2)-1))
y <- ifelse(apply(X,1,\(x)(sum(x+0.01*rnorm(2)))) <= 0,0,1)
col <- ifelse(y == 0, "blue", "red")

plot(X[,1], X[,2], col=col, pch = 19)
```




```
df <- data.frame(y=y, x1=X[,1], x2=X[,2])
model <- glm(y ~ x1 + x2, df, family = binomial())
```

```
Warning: glm.fit: algorithm did not converge
```

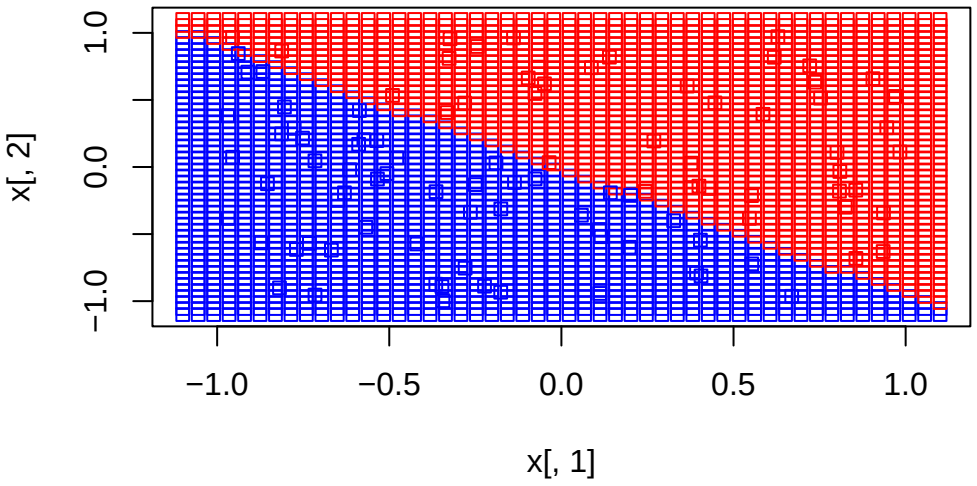
```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
f_logistic = \(x) predict(model, data.frame(x1=x[,1], x2 = x[,2]), type= 'response')
```

```
xnew <- cbind(
  x1 = rep(seq(-1.1,1.1,length.out=50),50),
  x2 = rep(seq(-1.1,1.1,length.out=50),each = 50)
)
```

```
plt <- function(f, x) {
  plot(x[,1], x[,2], col=ifelse(f(x) < 0.5, 'blue', 'red'), pch =22)
  points(df$x1, df$x2, col=ifelse(y=='0', 'blue', 'red'), pch =22)
}

overview <- function(f){
```

```
  predicted <- ifelse(f(df[,-1])<0.5, 0, 1)
  actual <- df[,1]
  table(predicted, actual)
}

plt(f_logistic, xnew)
```



Classification (Decision) Tree

- tries to find splits in categories
- hierarchical model that recursively partitions the data into smaller subsets based on the most informative features, and each partition is associated with a class label

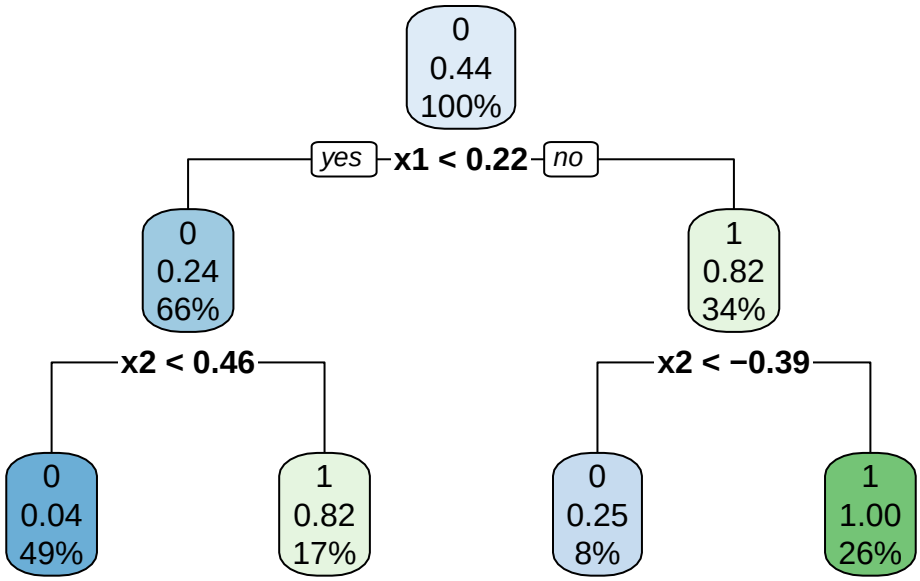
```
library(rpart)
```

```
Warning: package 'rpart' was built under R version 4.2.3
```

```
library(rpart.plot)
```

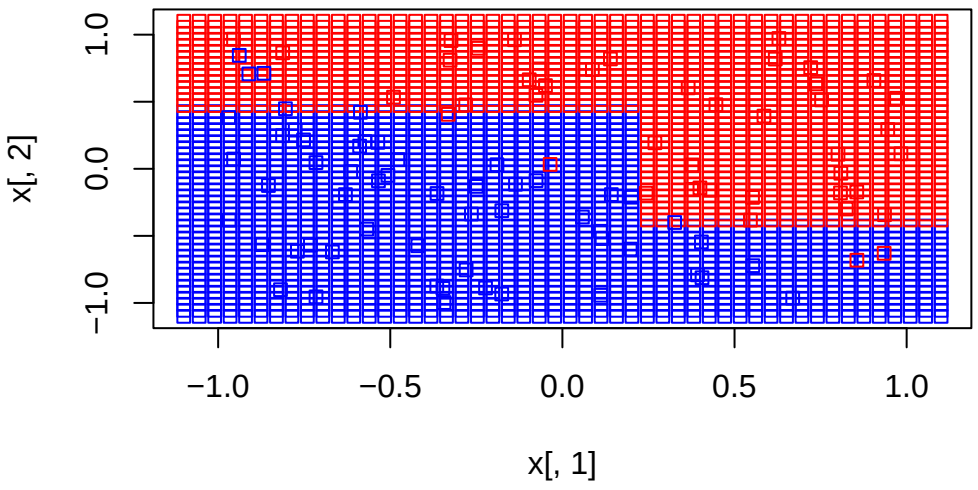
```
Warning: package 'rpart.plot' was built under R version 4.2.3
```

```
dtree <- rpart(y ~ x1 + x2, df, method = 'class')
rpart.plot(dtree)
```



- Each split involves a question which further divides the data into smaller groups
- Once the group is pure (all data samples in that group are the same) it is able to classify all those by the label those data samples possess

```
f_dtree <- \(x) as.numeric(predict(dtree, data.frame(x1 = x[,1], x2=x[,2]), type = 'class'
plt(f_dtree, xnew)
```



Creates multiple linear decision boundaries to split up the data based on what it predicts

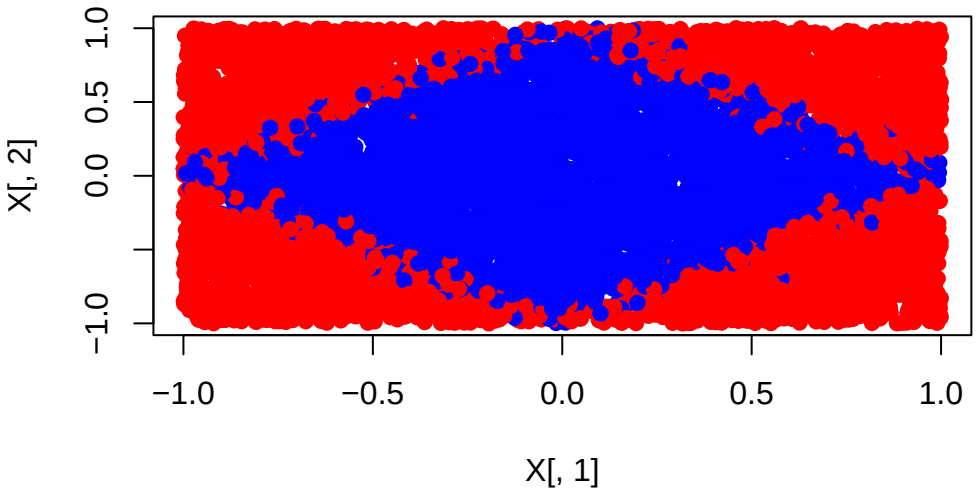
```
overview(f_dtree)
```

```
         actual
predicted  0  1
        0 53  4
        1  3 40
```

From the confusion matrix, we can see that it is pretty accurate, but has some errors as 6 are classified as red when it should be blue, and 3 are classified as blue when it should be red.

Support Vector Machine

```
n <- 7500
X <- t(replicate(n,2 * runif(2)-1))
y <- ifelse(apply(X,1, \(x) sum(abs(x))) + 0.1 * rnorm(n) <= 1, 0, 1) %>% as.factor()
col <- ifelse(y==0, 'blue', 'red')
df <- data.frame(y=y, x1=X[,1], x2=X[,2])

plot(X[,1], X[,2], col=col, pch=19)
```




```
library(e1071)
```

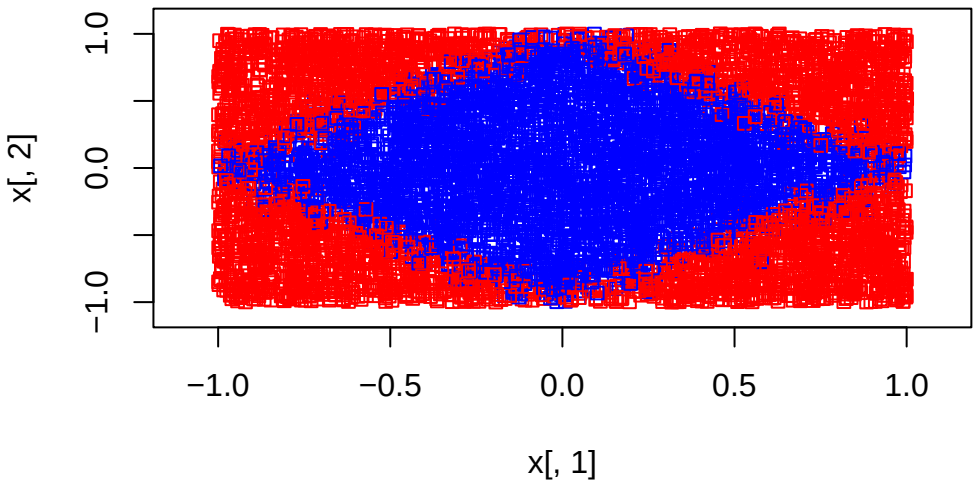
```
svm_model <- svm(y~x1+x2, df, kernel = 'radial')
summary(svm_model)
```

```
Call:
svm(formula = y ~ x1 + x2, data = df, kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  1629

 ( 813 816 )


Number of Classes:  2
```

```
Levels:
 0 1
```

```
f_svm <- \(x) predict(svm_model, x) %>% as.factor() - 1
plt(f_svm, xnew)
```

```
Warning in Ops.factor(predict(svm_model, x) %>% as.factor(), 1): '-' not
meaningful for factors
```



Shows the model that predicts what each data sample would be labeled at each spot on the graph. You can see that it is pretty similar to the original graph we created with the assigned red and blue points in the diamond formation.