Introduction to classification

STAT5003

Dr. Justin Wishart

Semester 2, 2022

Learning objective

The aim in this lecture is to learn four classification algorithms

- kNN,
- LDA
- Logistic regression
- Support vector machines

The dataset that we will be using for demonstration is included in the R package `mlbench`.

A simulated dataset is used for the demonstration of SVMs using the `e1071::svm` function.

Libraries to load

```
library(ggplot2)
library(tidyverse)
library(mlbench)
library(gridExtra)
library(grid)
library(MASS)
library(e1071)
library(caret)
```

Exploring the breast cancer dataset

This dataset contains 699 breast cancer samples with 9 features that have been classified as benign or malignant.

```
data("BreastCancer")
head(BreastCancer)
```

```
##        Id Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size
## 1 1000025            5         1          1             1            2
## 2 1002945            5         4          4             5            7
## 3 1015425            3         1          1             1            2
## 4 1016277            6         8          8             1            3
## 5 1017023            4         1          1             3            2
## 6 1017122            8        10         10             8            7
##   Bare.nuclei Bl.cromatin Normal.nucleoli Mitoses     Class
## 1           1           3               1       1    benign
## 2          10           3               2       1    benign
## 3           2           3               1       1    benign
## 4           4           3               7       1    benign
## 5           1           3               1       1    benign
## 6          10           9               7       1 malignant
```

```
# Checking how many samples there are in this dataset
dim(BreastCancer)
```

```
## [1] 699  11
```

```
#Finding the levels of target class
levels(BreastCancer$Class)
```

```
## [1] "benign"    "malignant"
```

```
# Let's just remove samples with missing values
BreastCancer.complete <- BreastCancer[complete.cases(BreastCancer),]
dim(BreastCancer.complete)
```

```
## [1] 683  11
```

```
# Or alternatively use `tidyr::drop_na`
BreastCancer <- BreastCancer %>% drop_na
dim(BreastCancer)
```

```
## [1] 683  11
```

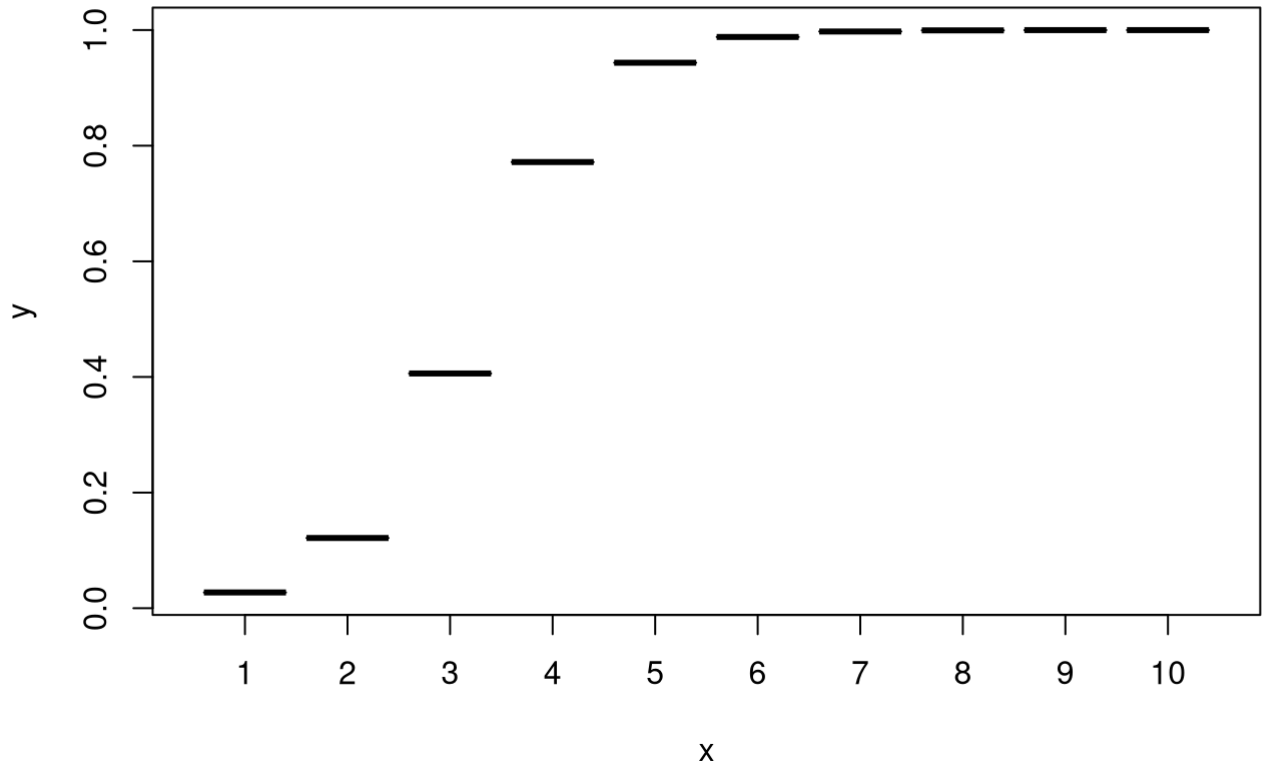
Logistic regression

To perform logistic regression in R, use the `glm()` function which stands for generalised linear model. It differs from the normal linear regression model `lm()` in that you have to define a link function that transforms the response variable. By passing in the argument `family = binomial(link = 'logit')`, we effectively tell `glm` to perform logistic regression.

```
# Perform logistic regression on one variable Cell.size
BreastCancer$Class_num <- as.numeric(BreastCancer$Class) - 1
BreastCancer$Cell_size_num <- as.numeric(BreastCancer$Cell.size)
# Now run logistic regression model
logistic.model <- glm(Class ~ as.numeric(Cell.size),
                      data = BreastCancer,
                      family = binomial(link = 'logit'))

summary(logistic.model)
```

```
##
## Call:
## glm(formula = Class ~ as.numeric(Cell.size), family = binomial(link = "logit"),
##     data = BreastCancer)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.2914  -0.2349  -0.2349   0.0142   2.6848
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -5.1745     0.3879  -13.34   <2e-16 ***
## as.numeric(Cell.size)   1.5980     0.1335   11.97   <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: 884.35  on 682  degrees of freedom
## Residual deviance: 254.76  on 681  degrees of freedom
## AIC: 258.76
##
## Number of Fisher Scoring iterations: 7
```

```
# Visualize the results of the logistic regression, see how the
# fitted values are between 0 and 1
plot(BreastCancer$Cell.size, logistic.model$fitted.values)
```



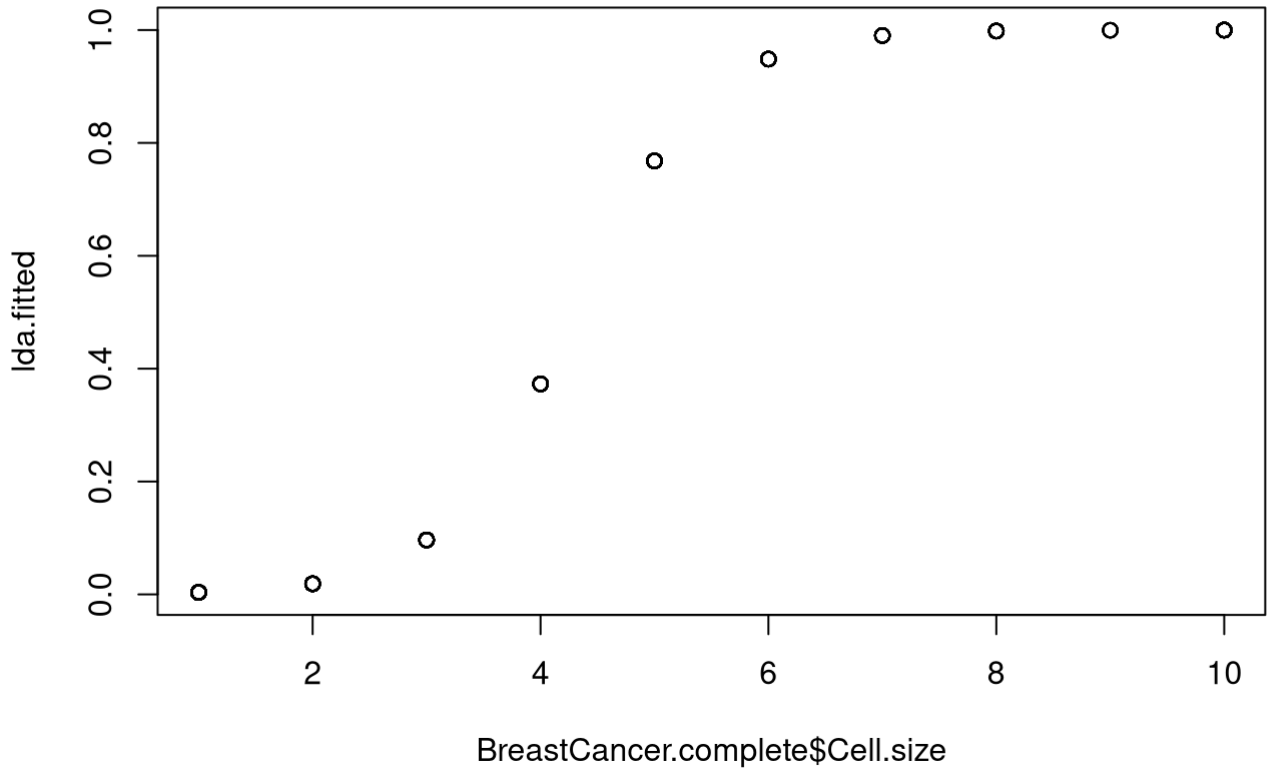

```
pred.classes <- ifelse(predict(logistic.model) > 0, "malignant", "benign")
# calculate classification accuracy (in percentage %)
mean(pred.classes == BreastCancer$Class) * 100
```

```
## [1] 92.97218
```

LDA classification

Now let's repeat the classification using the LDA algorithm. Let's first try the `lda()` function provided by the `MASS` package.

```
library(MASS)
# Train the lda model
BreastCancer.complete$Cell.size <- as.numeric(BreastCancer.complete$Cell.size)
lda.model <- MASS::lda(Class ~ Cell.size, data=BreastCancer.complete)
lda.fitted <- predict(lda.model, BreastCancer.complete)$posterior[, "malignant"]

# plot fitted values from LDA model
plot(BreastCancer.complete$Cell.size, lda.fitted)
```




```
# use fitted value to classify samples
lda.decision <- ifelse(lda.fitted > 0.5, 1, 0)
# calculate classification accuracy (in percentage %)
sum(lda.decision == BreastCancer.complete$Class_num) / nrow(BreastCancer.complete) *
100
```

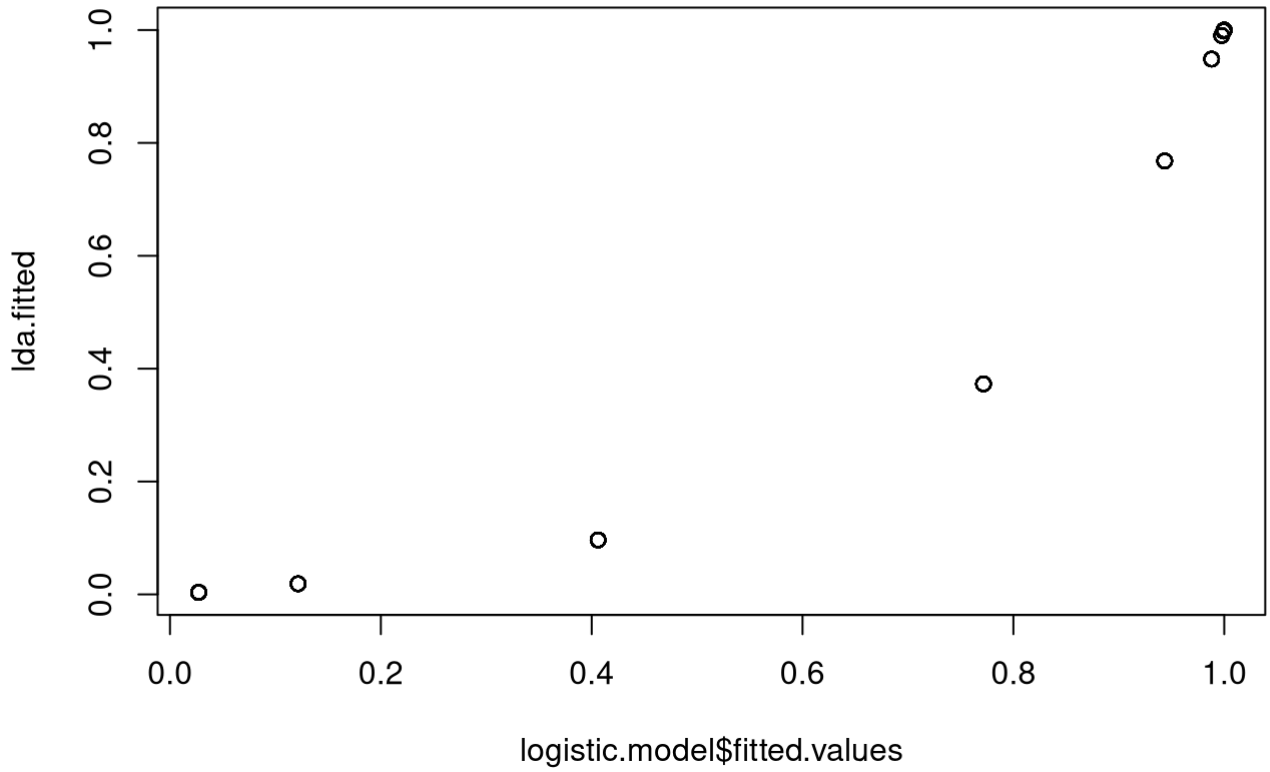
```
## [1] 0
```

```
mean(lda.decision == BreastCancer.complete$Class_num)
```

```
## [1] NaN
```

Compare Logistic regression with LDA

```
plot(logistic.model$fitted.values, lda.fitted)
```



Repeat logistic regression, LDA and kNN classification with the caret package

```
set.seed(123)
library(caret)
inTrain <- createDataPartition(BreastCancer.complete$Class, p = .8)[[1]]
breastCancerTrain <- BreastCancer.complete[ inTrain, ]
breastCancerTest  <- BreastCancer.complete[-inTrain, ]

# Here we create a logistic regression model using the train() function. For the "met
hod" parameter, "glm" stands for "generalized linear model" and using "glm" correspon
d to calling the "glm" package. This call will produce a logistic regression model.
# The trControl parameter gives control of what methods to be used for evaluating and
selecting model and how many times such procedure will be repeated. We will introduce
model evaluation and selection in later lectures.
set.seed(123)
logisticReg1 <- train(Class ~ as.numeric(Cell.size),
                      data = breastCancerTrain,
                      method = "glm", family = "binomial",
                      trControl = trainControl(method = "repeatedcv",
                                               repeats = 5))

## Print diagnostic and summary information and statistics fo the model
logisticReg1
```

```
## Generalized Linear Model
##
## 548 samples
##   1 predictor
##   2 classes: 'benign', 'malignant'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 493, 493, 494, 493, 493, 494, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.926924  0.8344079
```

```
# Let's try to include 3 more features in the model
logisticReg2 <- train(Class ~ as.numeric(Cell.size) +  as.numeric(Marg.adhesion) + a
s.numeric(Epith.c.size),
                      data = breastCancerTrain,
                      method = "glm", family = "binomial",
                      trControl = trainControl(method = "repeatedcv",
                                               repeats = 5))
logisticReg2
```

```
## Generalized Linear Model
##
## 548 samples
##   3 predictor
##   2 classes: 'benign', 'malignant'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 493, 494, 492, 493, 493, 494, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9375962  0.8607061
```

```
summary(logisticReg2)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.9543  -0.1970  -0.1970   0.0164   2.8113
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -6.4656     0.5895 -10.969  < 2e-16 ***
## `as.numeric(Cell.size)`       1.0638     0.1583   6.721 1.81e-11 ***
## `as.numeric(Marg.adhesion)`   0.4142     0.1085   3.818 0.000134 ***
## `as.numeric(Epith.c.size)`    0.5276     0.1494   3.531 0.000415 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 709.85  on 547  degrees of freedom
## Residual deviance: 167.10  on 544  degrees of freedom
## AIC: 175.1
##
## Number of Fisher Scoring iterations: 7
```

```
# # Let's try to train the model with LDA
lda <- train(Class ~ Cell.size + as.numeric(Marg.adhesion) + as.numeric(Epith.c.siz
e),
                     data = breastCancerTrain,
                     method = "lda",
                     trControl = trainControl(method = "repeatedcv",
                                              repeats = 5))
lda
```

```
## Linear Discriminant Analysis
##
## 548 samples
##   3 predictor
##   2 classes: 'benign', 'malignant'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 494, 492, 493, 492, 493, 493, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9160991  0.806508
```

```
# We can now use the model created on the test data set
lda.pred <- predict(lda, newdata = breastCancerTest)
head(lda.pred)
```

```
## [1] benign    benign    benign    malignant benign    benign
## Levels: benign malignant
```

```
confusionMatrix(data = lda.pred, reference = breastCancerTest$Class)
```

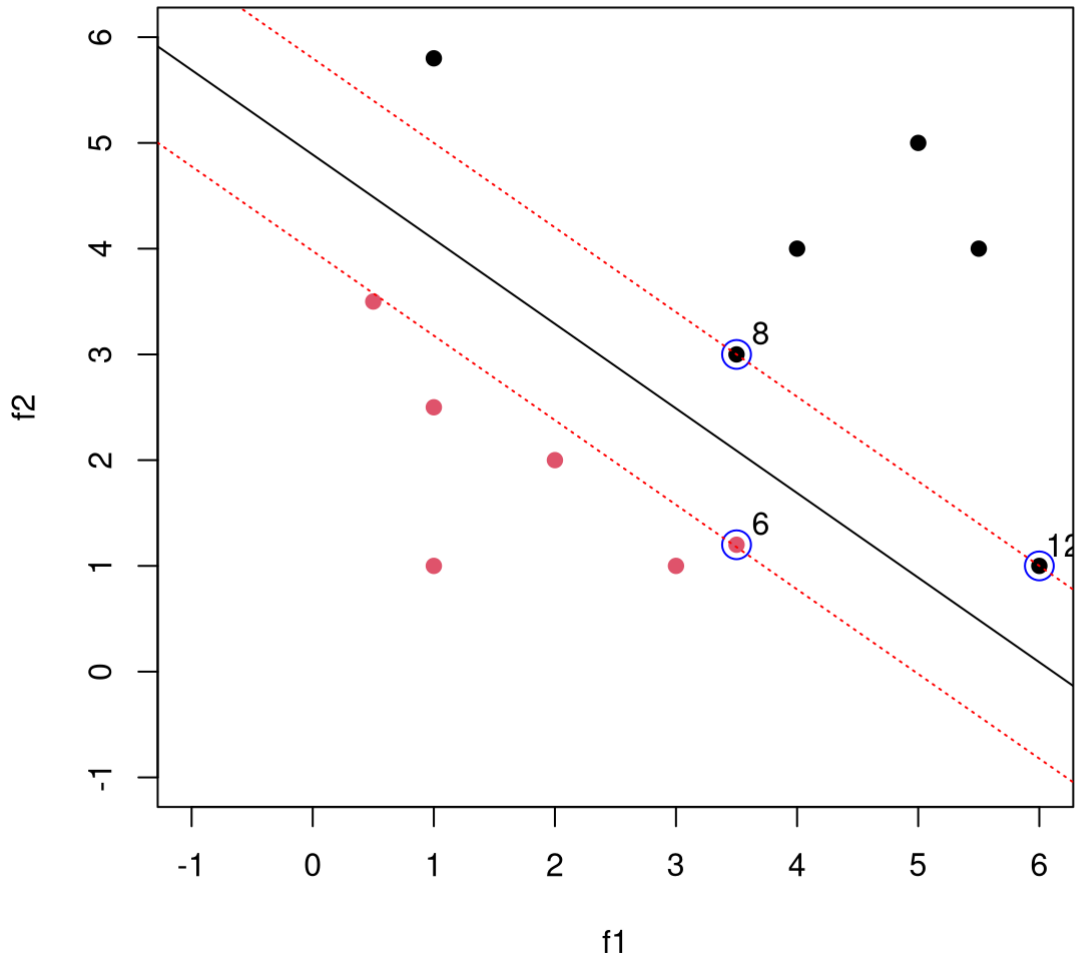
```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign        87        11
##   malignant      1        36
##
##                Accuracy : 0.9111
##                  95% CI : (0.8499, 0.9532)
##     No Information Rate : 0.6519
##     P-Value [Acc > NIR] : 2.463e-12
##
##                   Kappa : 0.7939
##
##  Mcnemar's Test P-Value : 0.009375
##
##             Sensitivity : 0.9886
##             Specificity : 0.7660
##          Pos Pred Value : 0.8878
##          Neg Pred Value : 0.9730
##              Prevalence : 0.6519
##          Detection Rate : 0.6444
##    Detection Prevalence : 0.7259
##       Balanced Accuracy : 0.8773
##
##        'Positive' Class : benign
##
```

```
# # Let's try to train the model with KNN
knn <- train(Class ~ Cell.size +  Marg.adhesion + Epith.c.size,
                     data = breastCancerTrain,
                     method = "knn",
                     trControl = trainControl(method = "repeatedcv",
                                              repeats = 5))

knn
```

```
## k-Nearest Neighbors
##
## 548 samples
##   3 predictor
##   2 classes: 'benign', 'malignant'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 493, 493, 493, 494, 493, 493, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.9416546  0.8710306
##   7  0.9365702  0.8591942
##   9  0.9365770  0.8590966
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

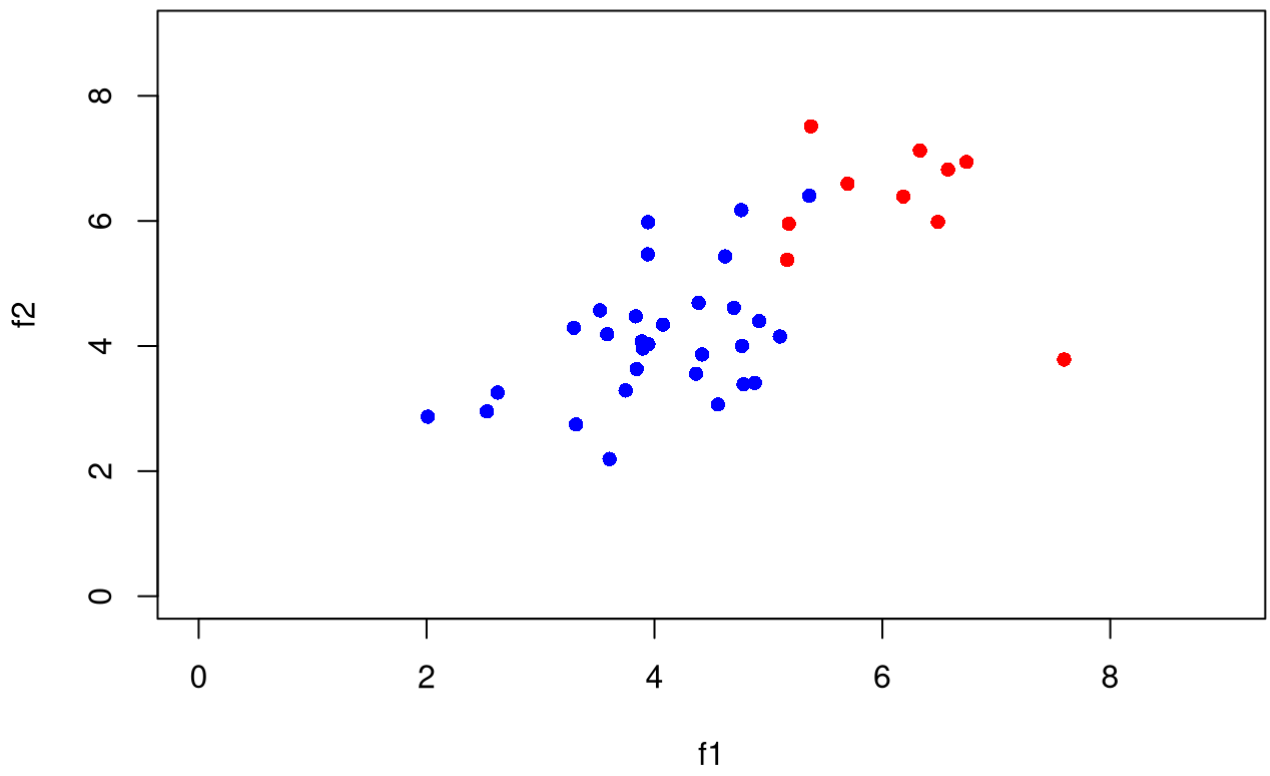
Demonstrate Support Vector Classifier

```
# create example data
f1 <- c(.5, 1, 1, 2, 3, 3.5,     1, 3.5, 4, 5, 5.5, 6)
f2 <- c(3.5, 1, 2.5, 2, 1, 1.2,  5.8, 3, 4, 5, 4, 1)
cls <- c(rep(+1, 6), rep(-1, 6))
dat <- cbind(f1, f2)

# plot all points from the two classes
plot(dat, col=(cls + 3)/2, pch = 19, xlim = c(-1, 6), ylim = c(-1, 6))

# train a maximal margin classifier
svm.model <- svm(dat, y = cls, kernel = "linear", type = "C-classification", scale =
FALSE)


# plot support vectors
points(svm.model$SV, col = "blue",cex = 2)
text(svm.model$SV + 0.2, labels = row.names(svm.model$SV)) # Could use labels here wi
th svm.model$index

# coefs: estimated betas
# svm.model$coefs gives the alpha weights
# svm.model$SV gives the support vectors.
# The hyperplane f(x) = beta_0 + \sum_{i in S} alpha_i <x, support_vector_i>
# The syntax %*% is to denote matrix multiplication
# alpha = svm.model$coefs
# support_vector = svm.model$SV
# Compute the weights
w <- t(svm.model$coefs) %*% svm.model$SV
# rho: the negative intercept of decision boundary
# Extract the beta_0
beta_0 <- -svm.model$rho
# Remap plane, i.e. equation currently in form f(x) = 0 = a + bx + cy
# where w = (b, c) and a = beta_0
# We want y = -a/c - b/c * x to use abline
# plot decision boundary
abline(a = -beta_0 / w[1, 2], b = -w[1, 1] / w[1, 2], col = "black", lty = 1)
# plot margins
abline(a = (-beta_0-1) / w[1, 2], b = -w[1, 1] / w[1, 2], col = "red", lty = 3)
abline(a = (-beta_0+1) / w[1, 2], b = -w[1, 1] / w[1, 2], col = "red", lty = 3)
```



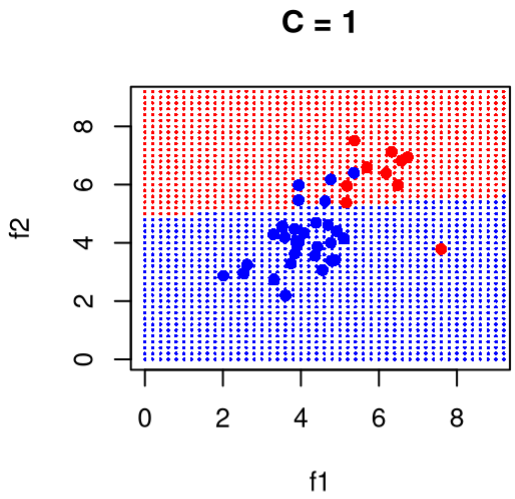
Create simulation dataset

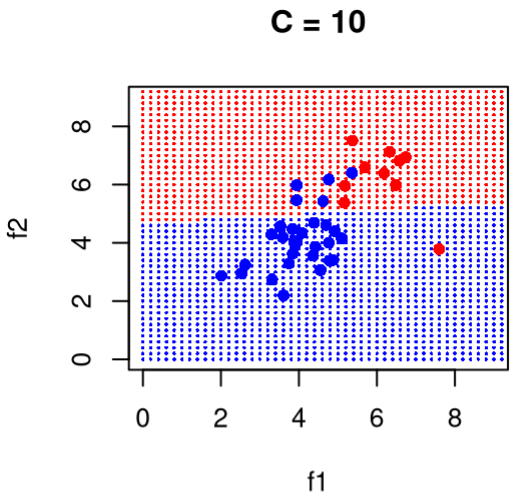
```
# create positive class sample with 2 descriptive features
set.seed(1)
f1 <- rnorm(10, mean = 6, sd = 1)
f2 <- rnorm(10, mean = 6, sd = 1)
old.P.data <- cbind(f1, f2)

# Alternatively
set.seed(1)
P.data <- replicate(2, rnorm(10, mean = 6, sd = 1))
colnames(P.data) <- c("f1", "f2")

# create negative class sample with 2 descriptive features
N.data <- replicate(2, rnorm(30, mean = 4, sd = 1))

# combine all samples
data.mat <- data.frame(rbind(P.data, N.data),
                       Class = rep(c("P", "N"), atime = c(nrow(P.data), nrow(N.dat
a))), stringsAsFactors = TRUE)
rownames(data.mat) <- paste("s", 1:(nrow(P.data)+nrow(N.data)), sep="")

# plot data
plot(P.data, col = "red", pch = 16, ylim = c(0, 9), xlim = c(0, 9))
points(N.data, col = "blue", pch = 16)
```

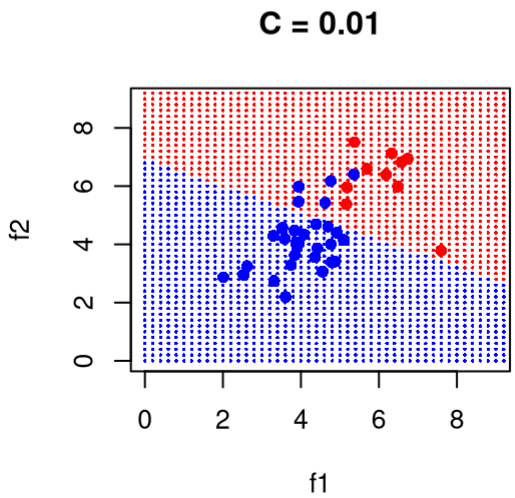


Train a support vector classifier

```
library(e1071)
# Try it with 3 different cost values
svm.model1 <- svm(x = data.mat[, -3], y = data.mat[, 3],
                  kernel = "linear", type = "C-classification", cost = 1)
svm.model2 <- svm(x = data.mat[, -3], y = data.mat[, 3],
                  kernel = "linear", type = "C-classification", cost = 10)
svm.model3 <- svm(x = data.mat[, -3], y = data.mat[, 3],
                  kernel = "linear", type = "C-classification", cost = 0.01)

par(mfrow = c(2, 2))
# mapping decision boundary for model1
## set up the plot without plotting points
plot(P.data, ylim = c(0, 9), xlim = c(0, 9), type = "n", main = "C = 1")

# Use the SVM model we've trained, predict a grid
# of points
for (x in seq(0, 10, by = 0.2)){
  for (y in seq(0, 10, by = 0.2)){
    t <- cbind(x, y)
    colnames(t) <- c("f1", "f2")
    if (predict(svm.model1, t) == 'N') {
      points(x, y, col = "blue", cex = 0.3, pch = 16)
    } else {
      points(x, y, col = "red", cex = 0.3, pch = 16)
    }
  }
}

points(P.data, col = "red", pch = 16)
points(N.data, col = "blue", pch = 16)

# mapping decision boundary for model2
plot(P.data, ylim = c(0, 9), xlim = c(0, 9), type = "n", main = "C = 10")
dat <- expand.grid(f1 = seq(0, 10, by = 0.2), f2 = seq(0, 10, by = 0.2))
predictions <- predict(svm.model2, dat)
points(dat, col = ifelse(predictions == "P", "red", "blue"), cex = 0.3, pch = 16)
points(P.data, col = "red", pch = 16)
points(N.data, col = "blue", pch = 16)


# mapping decision boundary for model3
plot(P.data, ylim = c(0, 9), xlim = c(0, 9), type = "n", main = "C = 0.01")
predictions <- predict(svm.model3, dat)
points(dat, col = ifelse(predictions == "P", "red", "blue"), cex = 0.3, pch = 16)
points(P.data, col = "red", pch = 16)
points(N.data, col = "blue", pch = 16)
```



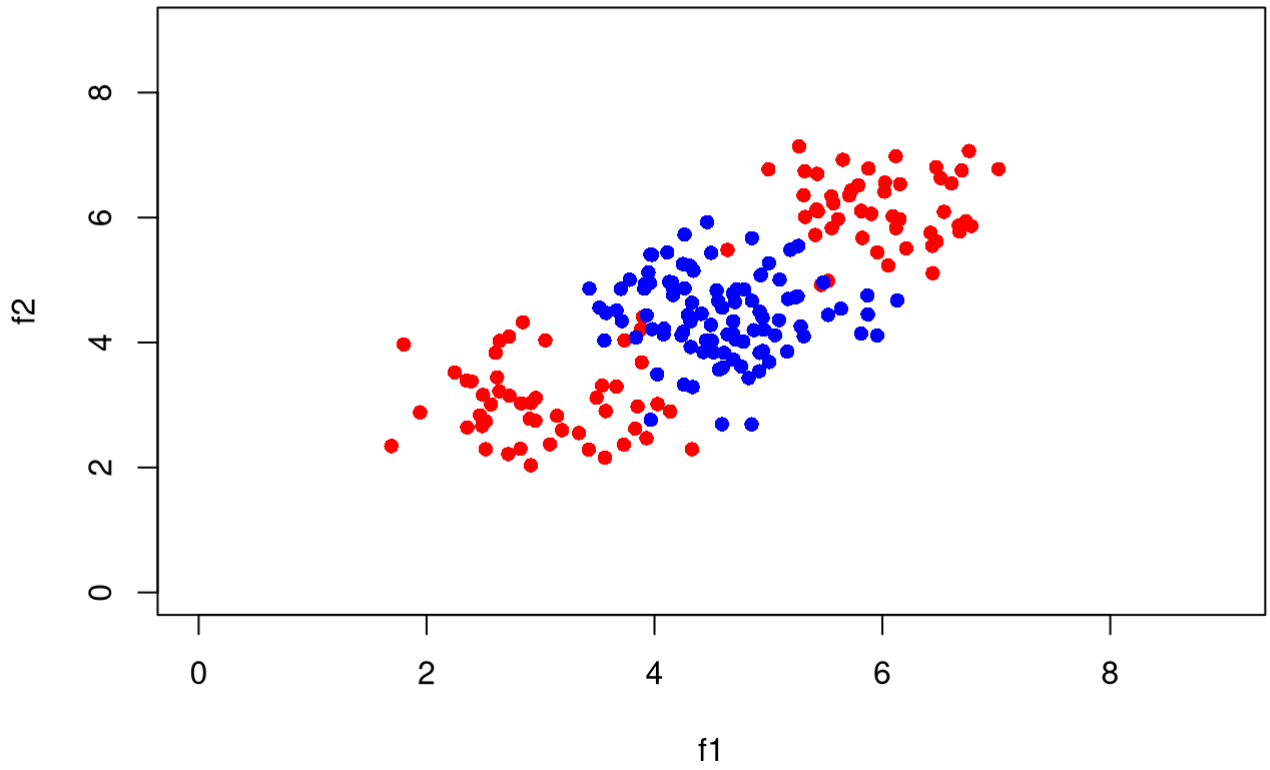




Demonstrate Support Vector Machines

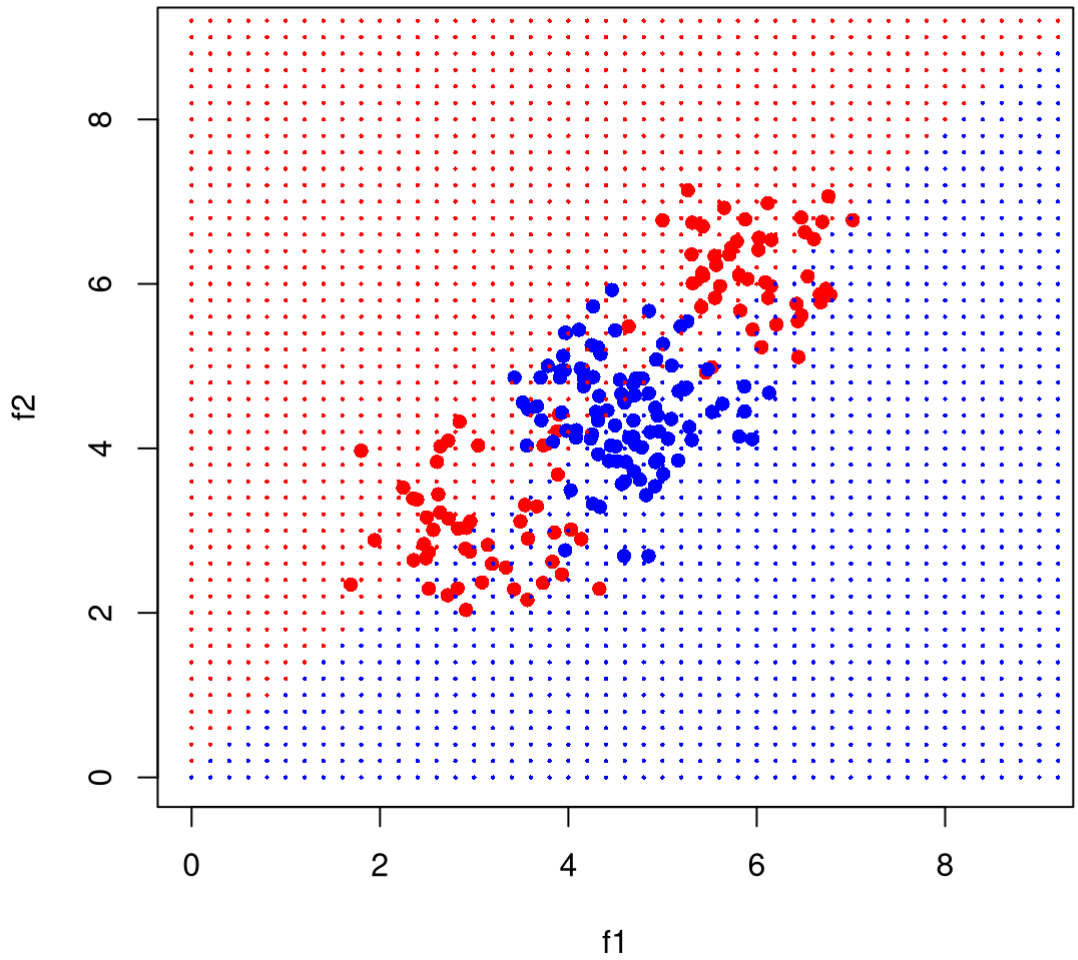
Create linearly non-seperable data

```
# create positive class sample with 2 descriptive features
set.seed(3)
f1 <- rnorm(50, mean = 6, sd = 0.6)
set.seed(4)
f2 <- rnorm(50, mean = 6, sd = 0.6)
P1.data <- cbind(f1, f2)

set.seed(5)
f1 <- rnorm(50, mean = 3, sd = 0.6)
set.seed(6)
f2 <- rnorm(50, mean = 3, sd = 0.6)
P2.data <- cbind(f1, f2)
P.data <- rbind(P1.data, P2.data)

# create positive class sample with 2 descriptive features
set.seed(7)
f1 <- rnorm(100, mean = 4.5, sd = 0.6)
set.seed(8)
f2 <- rnorm(100, mean = 4.5, sd = 0.6)
N.data <- cbind(f1, f2)

# combine all samples
data.mat <- data.frame(rbind(P.data, N.data),
                       Class = rep(c("P", "N"), time = c(nrow(P.data), nrow(N.dat
a))),
                       stringsAsFactors = TRUE)
rownames(data.mat) <- paste("s", 1:(nrow(P.data) + nrow(N.data)), sep = "")

# plot data
plot(P.data, col = "red", pch = 16, ylim = c(0, 9), xlim = c(0, 9))
points(N.data, col = "blue", pch = 16)
```



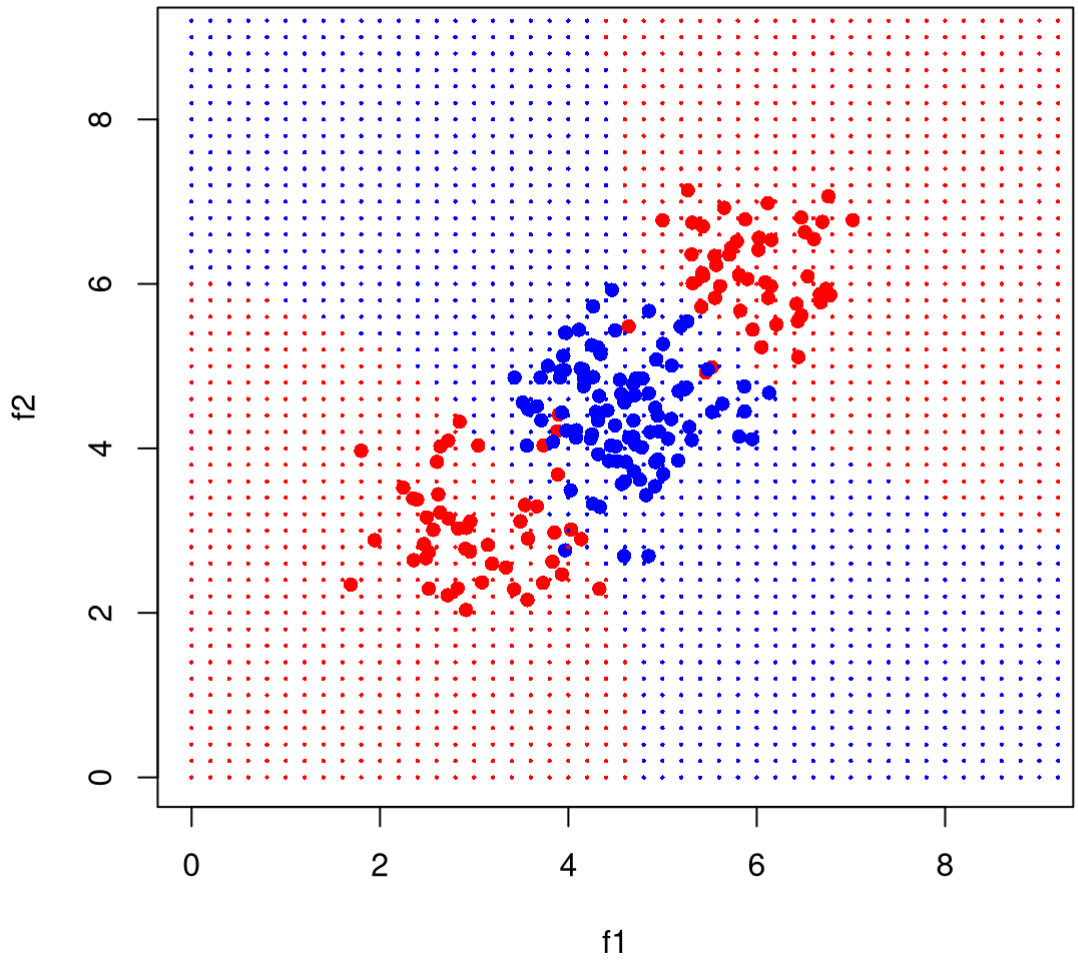
Using a support vector classifier to classify such a linearly non-seperable data

```
svm.model <- svm(x = data.mat[, -3], y = data.mat[, 3], kernel = "linear", type = "C-
classification")

# plot data
plot(P.data, col = "red", pch = 16, ylim = c(0, 9), xlim = c(0, 9))
points(N.data, col = "blue", pch = 16)

# mapping decision boundary
predictions <- predict(svm.model, dat)
points(dat, col = ifelse(predictions == "P", "red", "blue"), cex = 0.3, pch = 16)
```

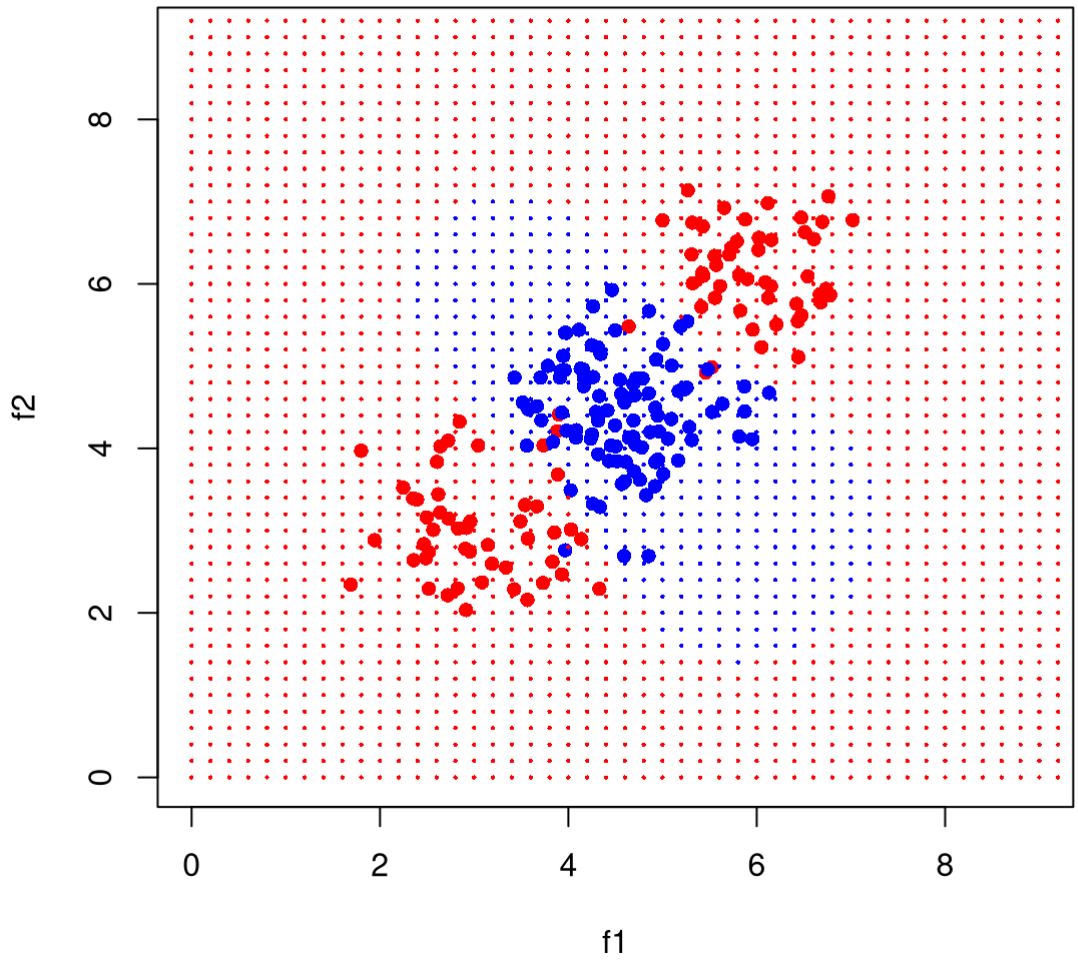


Using a support vector machine to classify such linearly non-seperable data

```
## polynomial kernel
svm.model <- svm(x = data.mat[, -3], y = data.mat[, 3], kernel = "polynomial", degree
= 6, type = "C-classification")

# plot data
plot(P.data, col = "red", pch = 16, ylim = c(0, 9), xlim = c(0, 9))
points(N.data, col = "blue", pch = 16)

# mapping decision boundary
predictions <- predict(svm.model, dat)
points(dat, col = ifelse(predictions == "P", "red", "blue"), cex = 0.3, pch = 16)
```




```
## radial basis function as kernel
svm.model <- svm(x = data.mat[, -3], y = data.mat[,3], kernel = "radial", type = "C-c
lassification")

# plot data
plot(P.data, col = "red", pch = 16, ylim = c(0, 9), xlim = c(0, 9))
points(N.data, col = "blue", pch = 16)

# mapping decision boundary
predictions <- predict(svm.model, dat)
points(dat, col = ifelse(predictions == "P", "red", "blue"), cex = 0.3, pch = 16)
```


f2
0
2
4
6
8
f1
0
2
4
6
8