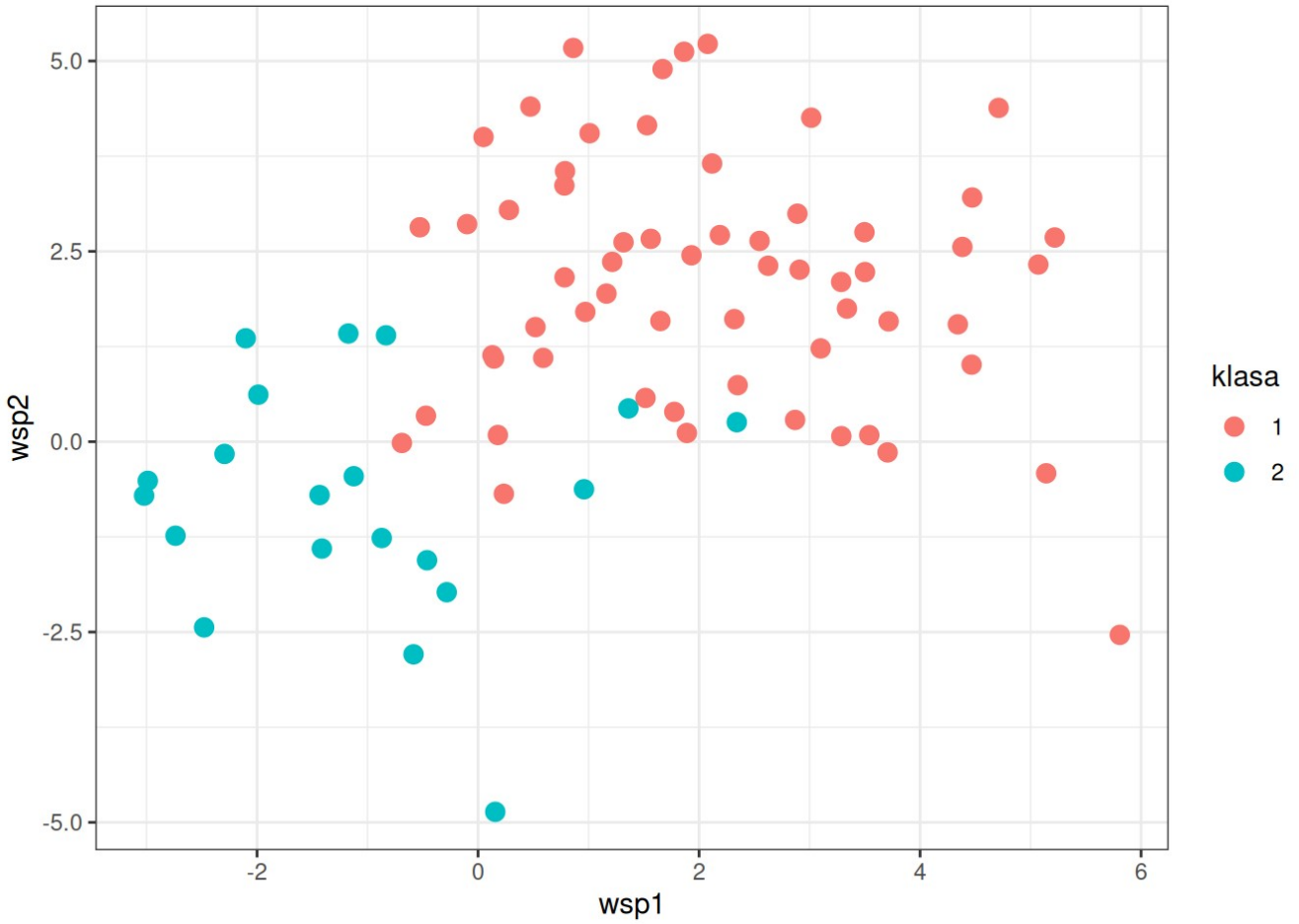
LABORATORIUM 11

- Liniowa Analiza Dyskryminacyjna
- Kwadratowa Analiza Dyskryminacyjna
- Maszyny Wektorów Nośnych

Liniowa Analiza Dyskryminacyjna

Prawdopodobnie najprostszą metodą klasyfikacji pod nadzorem jest **LDA** (*Linear Discriminant Analysis*). U jej podstaw leży założenie, że rozkłady zmiennych w każdej klasie są opisane wielowymiarowym rozkładem Gaussa. Ponadto, macierze kowariancji wszystkich klas są takie same. Można wtedy udowodnić, że klasy najlepiej rozdziela hiperpłaszczyzna (czyli prosta dla dwóch wymiarów). Zaczynamy od prostego przykładu dwóch klas, każda z nich jest reprezntowana przez punkty wylosowane z dwuwymiarowego rozkładu normalnego (funkcja `mvrnorm(liczba_danych, wartosc_oczekiwana, macierz_kowariancji)` z biblioteki `MASS`)

```
library(MASS)
library(ggplot2)

S <- matrix(c(3,0,0,3),2,2)
m1 <- c(2,2)
m2 <- c(-1,-1)

n1 <- 60
n2 <- 20
n <- n1 + n2

x1 <- mvrnorm(n1, m1, S)
x2 <- mvrnorm(n2, m2, S)

dane <- data.frame (wsp1 = c(x1[,1],x2[,1]),
                    wsp2 = c(x1[,2],x2[,2]),
                    klasa = as.factor (rep (c("1","2"), c(n1,n2))))

theme_set(theme_bw())
ggplot(dane) + geom_point(aes(x = wsp1, y = wsp2, color = klasa), shape = 19, size = 3)
```

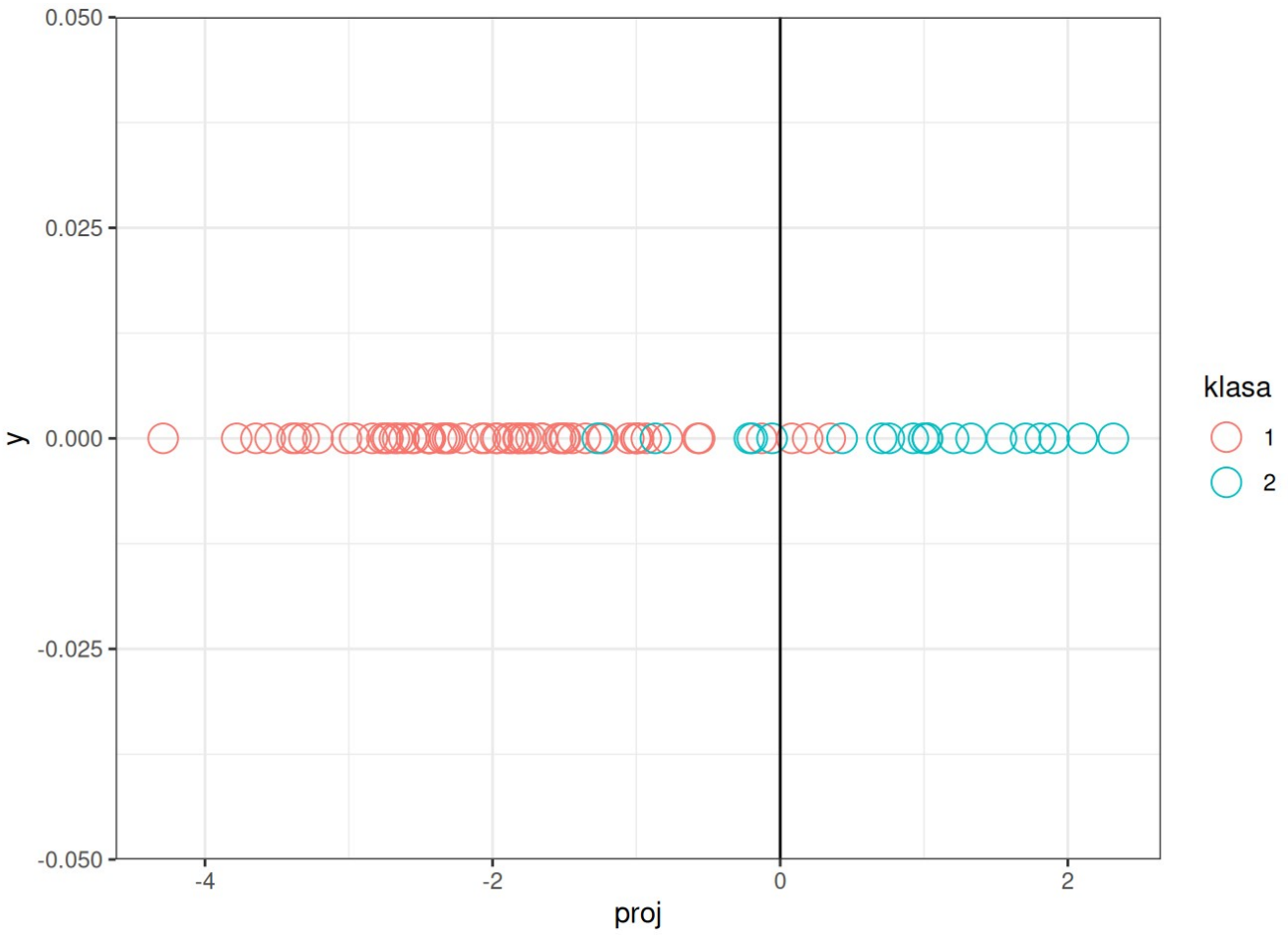


W pakiecie R LDA jest realizowane przez funkcję `lda()`, gdzie jako argumenty podajemy zależność pomiędzy polami ramki danych (np. `z ~ x + y`). W efekcie otrzymamy wartości prawdopodobieństw a priori (częstości klas), wartości oczekiwane punktów należących do klas a także kierunek, na który należy rzutować obserwacje tak, aby dokonać najlepszego rozdzielenia klas. Jeżeli wartość tak dokonanego rzutowania danej (punktu) jest większa od zera, przypisujemy go do klasy “1”, w przeciwnym wypadku do klasy “2”.

```
dane.lda <- lda(klasa ~ wsp1 + wsp2, data = dane)
dane.lda
```

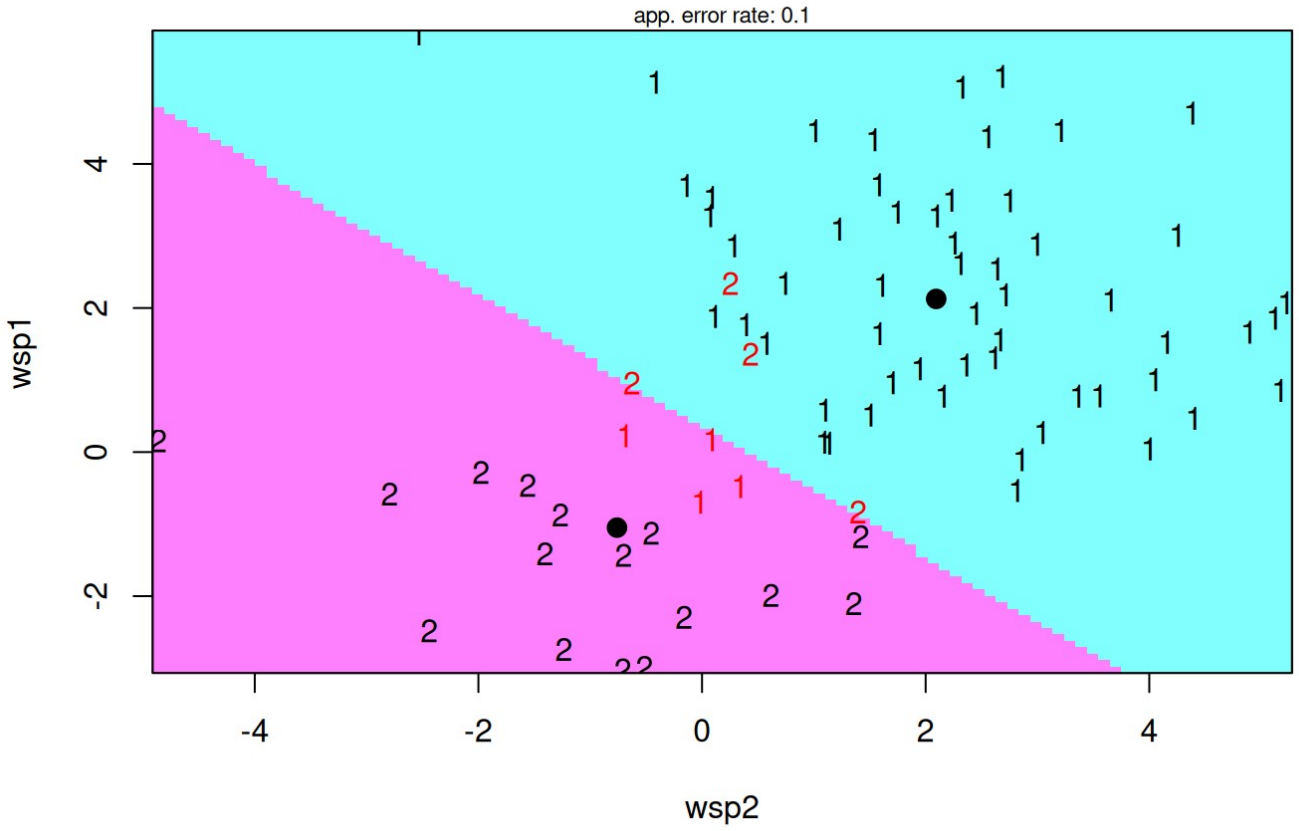
```
## Call:
## lda(klasa ~ wsp1 + wsp2, data = dane)
##
## Prior probabilities of groups:
##    1    2
## 0.75 0.25
##
## Group means:
##        wsp1       wsp2
## 1  2.126517  2.0942449
## 2 -1.048631 -0.7603175
##
## Coefficients of linear discriminants:
##             LD1
## wsp1 -0.4940120
## wsp2 -0.4477963
```

```
proj <- as.matrix(dane[,1:2]) %*% dane.lda$scaling
dane$proj <- proj[,1]
ggplot(dane) + geom_point(aes(x = proj, y = 0, color = klasa), shape=21, size=5) + geom_vl
         ine(xintercept=0)
```



Oczywiście “najlepsze” rozdzielenie klas nie oznacza, że otrzymamy podział, w którym wszystkie punkty z klasy “1” faktycznie zostaną sklasyfikowane jako należące do tej grupy - widać to na powyższym rysunku. Za pomocą biblioteki `klaR` i zaimplementowanej w niej funkcji `partimat()` można otrzymać explicite prostą rozdzielającą klasy, a także szybko zweryfikować, które punkty zostały błędnie sklasyfikowane. Ilościowej oceny dokładności klasyfikacji dokonamy przy użyciu macierzy pomyłek.

```
library(klaR)
partimat(klasa ~ wsp1 + wsp2, data = dane, method="lda")
```




```
library(caret)
```

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

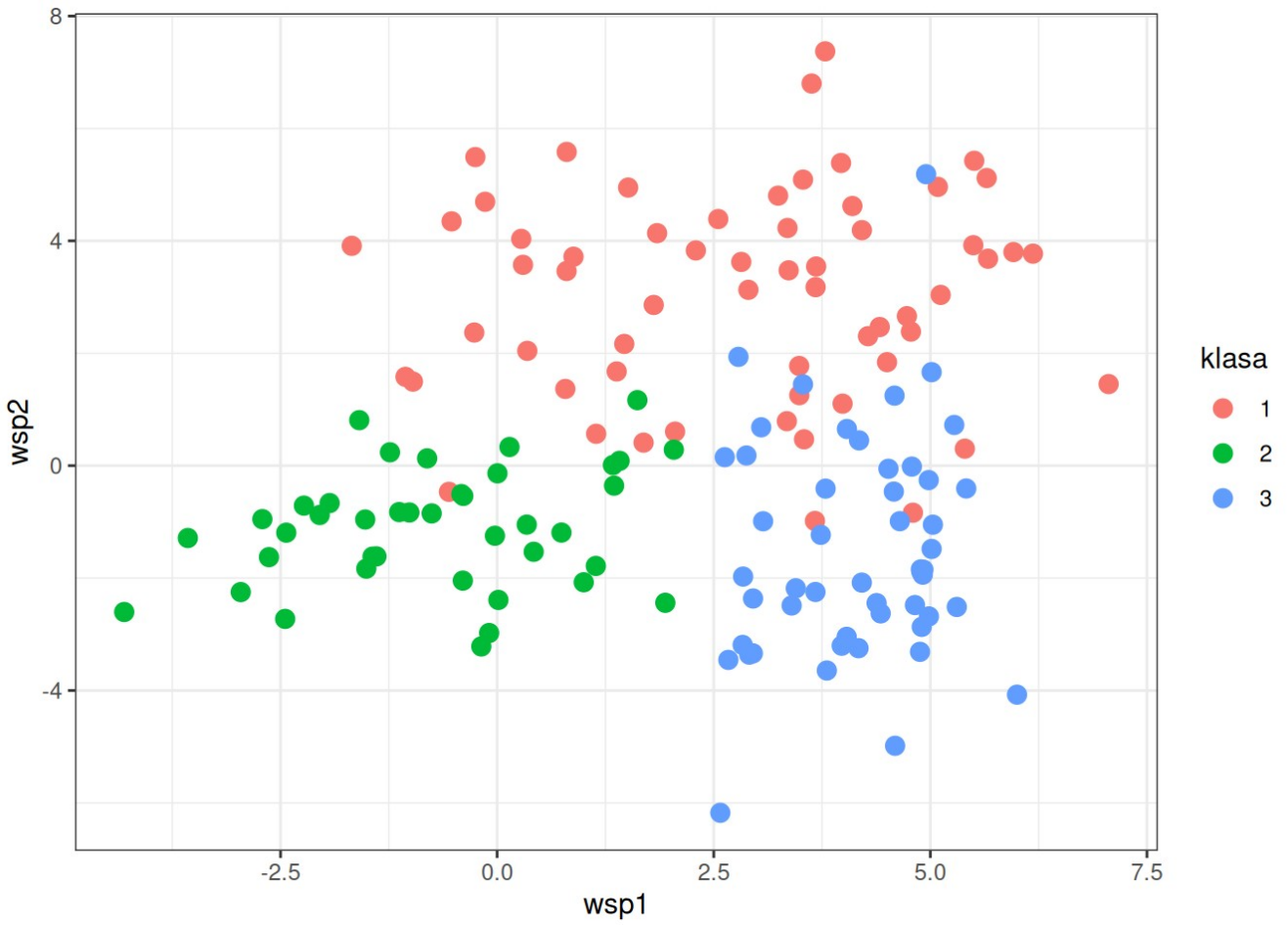
```
dane.pred <- predict(dane.lda, dane)
dane.conf <- confusionMatrix(dane.pred$class,dane$klasa, positive = "1")
dane.conf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 56  4
##          2  4 16
##
##                Accuracy : 0.9
##                  95% CI : (0.8124, 0.9558)
##     No Information Rate : 0.75
##     P-Value [Acc > NIR] : 0.0006477
##
##                   Kappa : 0.7333
##
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.9333
##             Specificity : 0.8000
##          Pos Pred Value : 0.9333
##          Neg Pred Value : 0.8000
##              Prevalence : 0.7500
##          Detection Rate : 0.7000
##    Detection Prevalence : 0.7500
##       Balanced Accuracy : 0.8667
##
##        'Positive' Class : 1
##
```

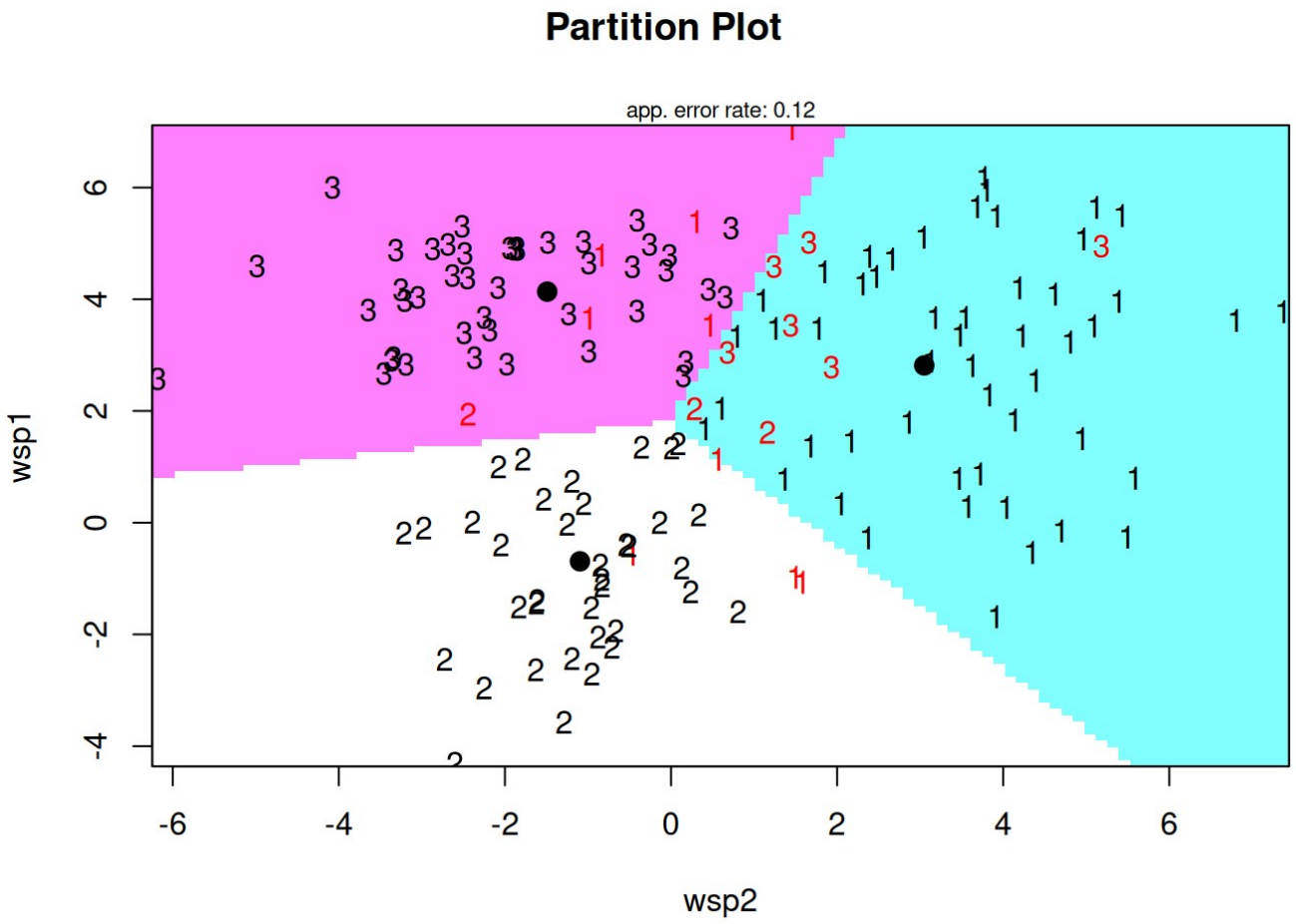
Kwadratowa Analiza Dyskryminacyjna

W sytuacji gdy nie jest spełnione założenie o równości macierzy kowariancji we wszystkich grupach, skuteczność liniowej analizy dyskryminacyjnej jest dużo mniejsza i dlatego lepiej skorzystać z kwadratowej analizy dyskryminacyjnej **QDA** (*Quadratic Discriminant Analysis*). Porównamy obie metody na sztucznym zbiorze zawierającym obserwacje z trzech grup o różnych macierzach kowariancji.

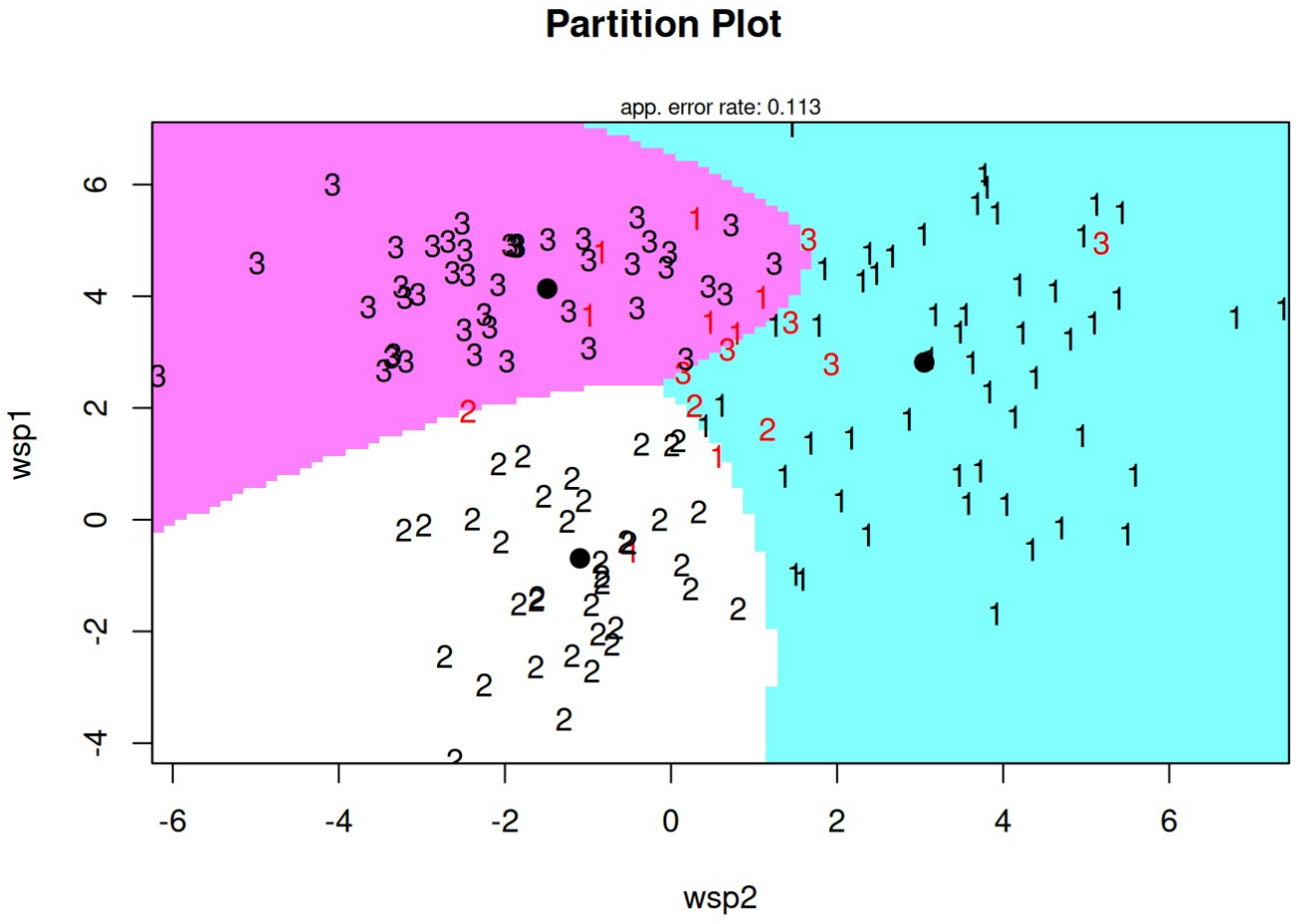
```
S1 <- matrix(c(4,0,0,4),2,2)
S2 <- matrix(c(2,0,0,1),2,2)
S3 <- matrix(c(1,0,0,4),2,2)
m1 <- c(3,3)
m2 <- c(-1,-1)
m3 <- c(4,-2)

n1 <- 60
n2 <- 40
n3 <- 50
n <- n1 + n2 + n3

x1 <- mvrnorm(n1, m1, S1)
x2 <- mvrnorm(n2, m2, S2)
x3 <- mvrnorm(n3, m3, S3)

dane <- data.frame (wsp1 = c(x1[,1], x2[,1], x3[,1]),
                    wsp2 = c(x1[,2], x2[,2], x3[,2]),
                    klasa = as.factor (rep (c("1", "2", "3"), c(n1, n2, n3))))

ggplot(dane) + geom_point(aes(x = wsp1, y = wsp2, color = klasa), shape = 19, size = 3)
```




```
partimat(klasa ~ wsp1 + wsp2, data = dane, method="lda")
```




```
partimat(klasa ~ wsp1 + wsp2, data = dane, method="qda")
```



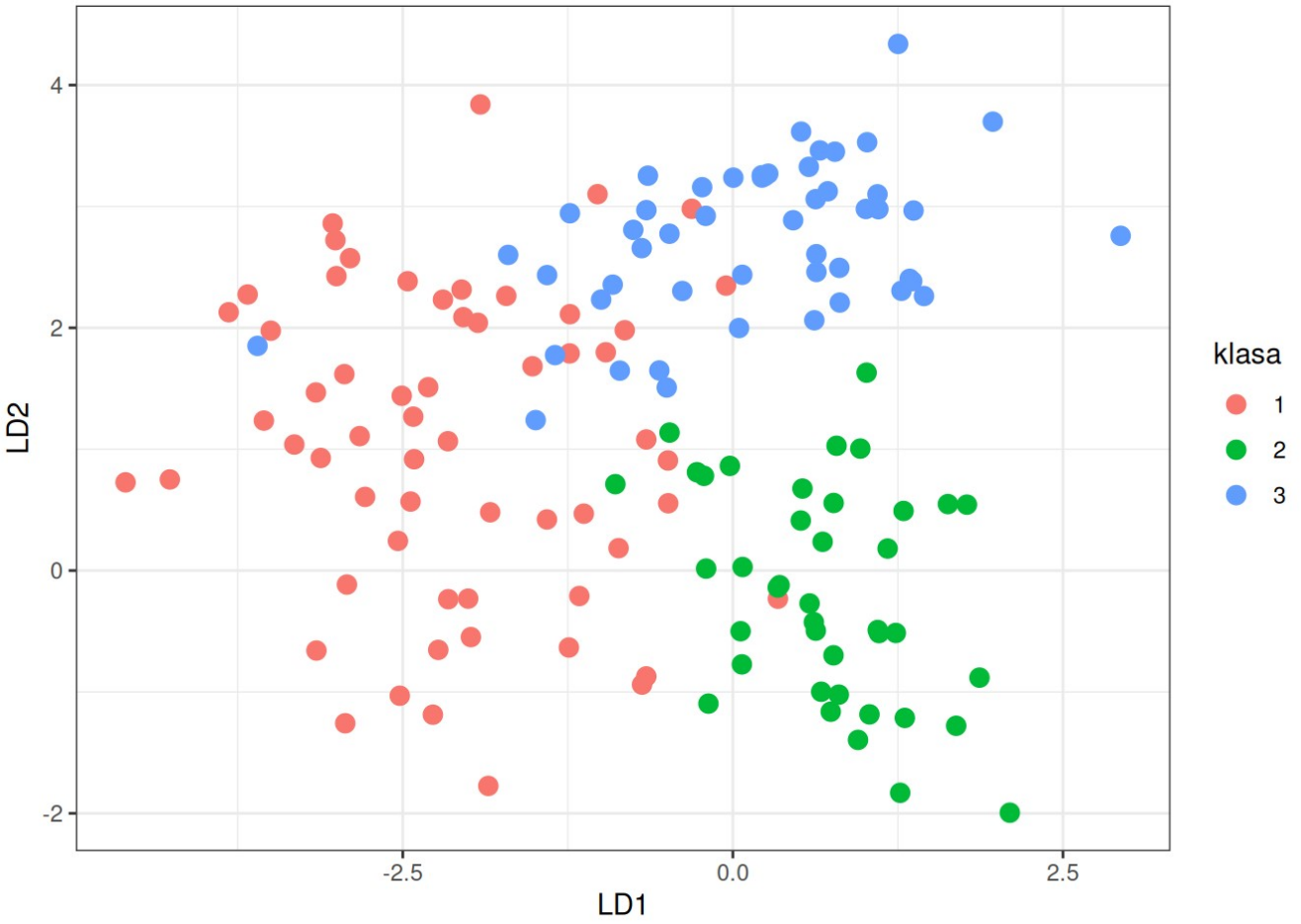
```
# Trenujemy klasyfikator LDA i testujemy go przy użyciu 10-krotnej walidacji krzyżowej
dane.lda.cv <- train(klasa ~ wsp1 + wsp2, data = dane, method="lda", trControl = trainCont
        rol(method = "cv", number=10))
# Trenujemy klasyfikator QDA i testujemy go przy użyciu 10-krotnej walidacji krzyżowej
dane.qda.cv <- train(klasa ~ wsp1 + wsp2, data = dane, method="qda", trControl = trainCont
        rol(method = "cv", number=10))
dane.lda.cv
```

```
## Linear Discriminant Analysis
##
## 150 samples
##   2 predictor
##   3 classes: '1', '2', '3'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 135, 135, 135, 135, 135, 135, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8866667  0.8286468
```

```
dane.qda.cv
```

```
## Quadratic Discriminant Analysis
##
## 150 samples
##   2 predictor
##   3 classes: '1', '2', '3'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 135, 135, 135, 135, 135, 135, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8866667  0.827803
```

```
# Obserwacje zrzutowane na proste ortogonalne do prostych rozdzielających
dane.lda <- lda(klasa ~ wsp1 + wsp2, data = dane)
proj <- as.matrix(dane[,1:2]) %*% dane.lda$scaling
dane.proj <- data.frame(proj, klasa=dane$klasa)
g <- ggplot(dane.proj, aes(x=LD1, y=LD2))
g + geom_point(aes(color=klasa), size=3)
```



Maszyny Wektorów Nośnych

Maszyny wektorów nośnych (podpierających) (*Support Vector Machines -SVM*) są dość współczesną (z lat 90-tych XX w.) metodą uczenia pod nadzorem. Ich podstawową cechą jest nowe spojrzenie na problem hiperpłaszczyzny rozdzielającej: zadanie polega na znalezieniu jak największego możliwego marginesu, rodzielającego punkty należące do różnych klas. Same hiperpłaszczyzny marginesów muszą przechodzić przez konkretne elementy próby uczącej i są nazywane właśnie wektorami podpierającymi. Jeśli przez $\mathbf{x_i}$ oznaczymy wektor obserwacji, a przez $y_i = \{-1, 1\}$ jego klasę, to wyrażenie na wektor podpierający można zapisać jako

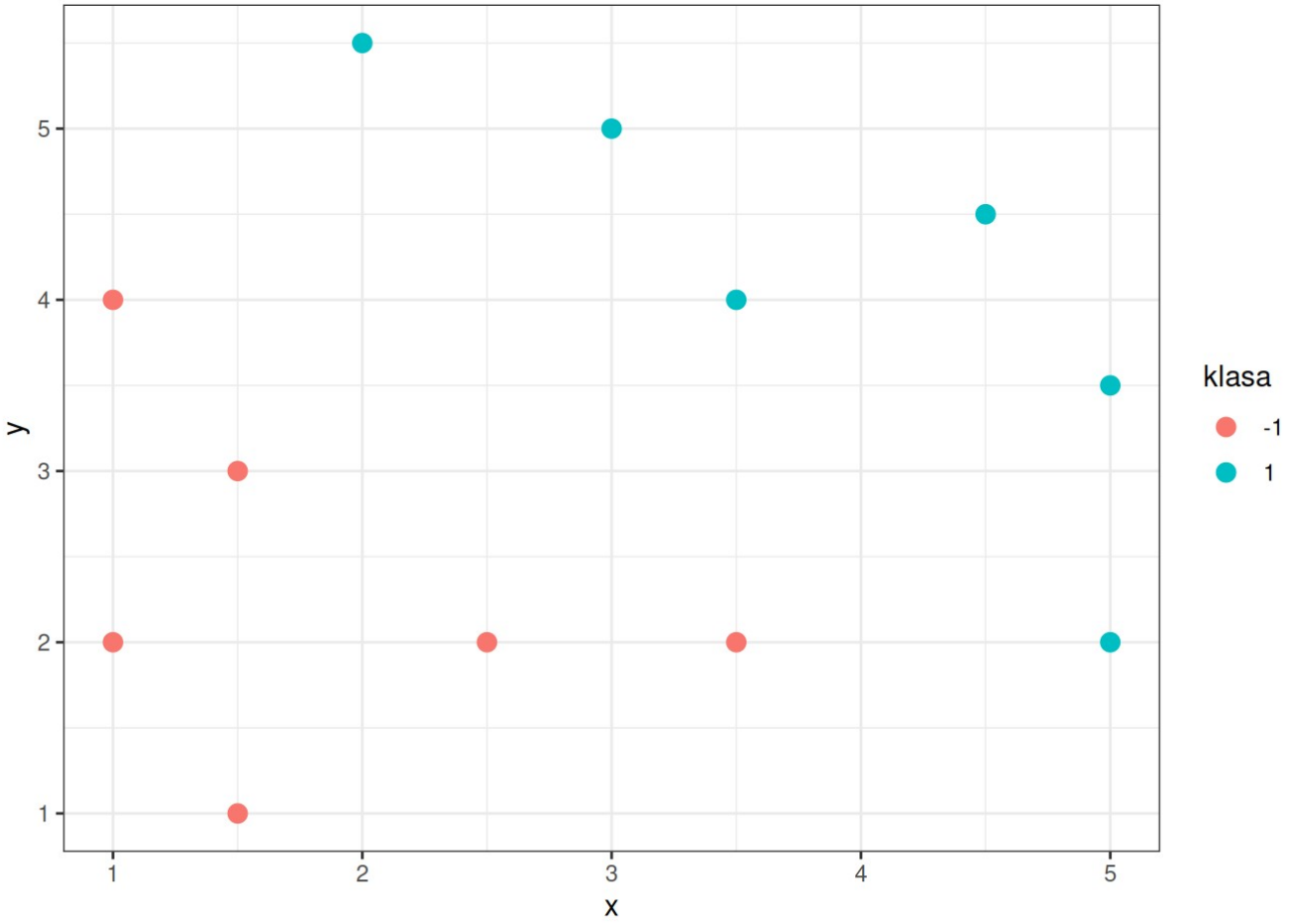
$$\mathbf{w} = \sum_{i=1}^{i=n} y_i \alpha_i x_i,$$

gdzie współczynniki α_i są niezerowe jedynie dla wektorów nośnych. Wyraz wolny jest równy

$$b = \frac{1}{2}(\mathbf{w} * \mathbf{x}_1^* + \mathbf{w} * \mathbf{x}_{-1}^*),$$

gdzie $\mathbf{x}_1, \mathbf{x}_{-1}$ to punkty przez które przechodzą wektory nośne. Jako ilustrację wykorzystamy prosty przykład z 12 obserwacjami z dwóch klas.

```
x <- c(1.0, 1.5, 1.0, 1.5, 2.5, 3.5, 2.0, 3.0, 3.5, 4.5, 5.0, 5.0)
y <- c(2.0, 1.0, 4.0, 3.0, 2.0, 2.0, 5.5, 5.0, 4.0, 4.5, 2.0, 3.5)

dane <- data.frame (x = x,
                    y = y,
                    klasa = as.factor (rep (c("-1", "1"), each=6)))

g <- ggplot(dane) + geom_point (aes (x = x, y = y, color = klasa), shape = 19, size = 3)
g
```



Do klasyfikacji użyjemy funkcji `svm()` z biblioteki `e1071`. Posiada ona wiele parametrów, m.in.:

- `type` - rodzaj maszyny, której chcemy użyć, jest do wyboru pięć opcji, trzy dotyczą klasyfikacji a dwie regresji,
- `kernel` - jądro użyte do trenowania i predykcji, do wyboru liniowe `linear`, wielomianowe `polynomial`, radialne `radial` oraz sigmoidalne `sigmoid`,
- `scale` - zmienna logiczna decydująca czy dane powinny być przeskalowane (odjęcie średniej i podzielenie przez wariancję), domyślnie ma wartość `TRUE`,
- `cost` - parametr regulujący karę za przekroczenie granicy marginesów.

```
library(e1071)

data.svm <- svm(klasa ~ x + y,
                data = dane,
                type = "C-classification",
                kernel = "linear",
                cost = 10,
                scale = FALSE)

# Wektory nośne
print(data.svm$SV)
```

```
##      x y.1
## 6  3.5 2.0
## 7  2.0 5.5
## 11 5.0 2.0
```

```
# Wartości współczynników alfa
print(data.svm$coefs)
```

```
##            [,1]
## [1,]  1.5416840
## [2,] -0.3264949
## [3,] -1.2151892
```

Jak widać, jedynie trzy punkty stanowią SV (supporting vectors - wektory nośne), przy czym wartości współczynników α kodują od razu klasę poprzez znaku przed współczynnikiem. Wyznaczymy wektor $\mathbf{w}$ i dzięki niemu naniesiemy na wykres hiperpłaszczyznę rodzielająca oraz hiperpłaszczyzny marginesów.

```
# Wyznaczenie wektora w
w <- t(data.svm$SV) %*% data.svm$coefs
w
```

```
##          [,1]
## x   -1.333041
## y.1 -1.142732
```

```
b <- data.svm$rho
b
```

```
## [1] -7.950963
```

```
xx <- 0:6

hiper <- data.frame (xx = xx,
                     r = -w[1]/w[2]*xx + b/w[2],
                     m1 = -w[1]/w[2]*xx + (b+1)/w[2],
                     m2 = -w[1]/w[2]*xx + (b-1)/w[2])

# Hiperpłaszczyzna rodzielająca
r <- geom_line(data = hiper, aes (x=xx, y=r), color = "red")

# Hiperpłaszczyzny marginesów
m1 <- geom_line(data = hiper, aes (x=xx, y=m1))
m2 <- geom_line(data = hiper, aes (x=xx, y=m2))

g + r + m1 + m2
```