Support Vector Machines

Data Sets

Attrition

```
attrition <- attrition %>% mutate_if(is.ordered, factor, order = F)
attrition_h2o <- as.h2o(attrition)

churn <- initial_split(attrition, prop = .7, strata = "Attrition")

churn_train <- training(churn)
churn_test <- testing(churn)

rm(churn)
```

Ames, Iowa housing data.

```
set.seed(123)

ames <- AmesHousing::make_ames()
ames_h2o <- as.h2o(ames)

ames_split <- initial_split(ames, prop =.7, strata = "Sale_Price")

ames_train <- training(ames_split)
ames_test <- testing(ames_split)

rm(ames_split)

h2o.init(max_mem_size = "10g", strict_version_check = F)
```

```
 Connection successful!

R is connected to the H2O cluster:
    H2O cluster uptime:         3 seconds 33 milliseconds
    H2O cluster timezone:       America/New_York
    H2O data parsing timezone:  UTC
    H2O cluster version:        3.28.0.2
    H2O cluster version age:    14 days, 17 hours and 30 minutes
    H2O cluster name:           H2O_started_from_R_brandon_zuo483
    H2O cluster total nodes:    1
    H2O cluster total memory:   15.71 GB
    H2O cluster total cores:    16
    H2O cluster allowed cores:  16
    H2O cluster healthy:        TRUE
```

```
    H2O Connection ip:          localhost
    H2O Connection port:        54321
    H2O Connection proxy:       NA
    H2O Internal Security:      FALSE
    H2O API Extensions:         Amazon S3, XGBoost, Algos, AutoML, Core V3, TargetEncoder, Core
    R Version:                  R version 3.6.2 (2019-12-12)
```

```
train_h2o <- as.h2o(ames_train)
response <- "Sale_Price"
predictors <- setdiff(colnames(ames_train), response)
```

```
# Colors
dark2 <- RColorBrewer::brewer.pal(8, "Dark2")
set1 <- RColorBrewer::brewer.pal(9, "Set1")

# Plotting function; modified from svmpath::svmpath()
plot_svmpath <- function(x, step = max(x$Step), main = "") {

  # Extract model info
  object <- x
  f <- predict(object, lambda = object$lambda[step], type = "function")
  x <- object$x
  y <- object$y
  Elbow <- object$Elbow[[step]]
  alpha <- object$alpha[, step]
  alpha0 <- object$alpha0[step]
  lambda <- object$lambda[step]
  df <- as.data.frame(x[, 1L:2L])
  names(df) <- c("x1", "x2")
  df$y <- norm2d$y
  beta <- (alpha * y) %*% x

  # Construct plot
  ggplot(df, aes(x = x1, y = x2)) +
    geom_point(aes(shape = y, color = y), size = 3, alpha = 0.75) +
    xlab("Income (standardized)") +
    ylab("Lot size (standardized)") +
    xlim(-6, 6) +
    ylim(-6, 6) +
    coord_fixed() +
    theme(legend.position = "none") +
    theme_bw() +
    scale_shape_discrete(
      name = "Owns a riding\nmower?",
      breaks = c(1, 2),
```

```r
      labels = c("Yes", "No")
    ) +
    scale_color_brewer(
      name = "Owns a riding\nmower?",
      palette = "Dark2",
      breaks = c(1, 2),
      labels = c("Yes", "No")
    ) +
    geom_abline(intercept = -alpha0/beta[2], slope = -beta[1]/beta[2],
                color = "black") +
    geom_abline(intercept = lambda/beta[2] - alpha0/beta[2],
                slope = -beta[1]/beta[2],
                color = "black", linetype = 2) +
    geom_abline(intercept = -lambda/beta[2] - alpha0/beta[2],
                slope = -beta[1]/beta[2],
                color = "black", linetype = 2) +
    geom_point(data = df[Elbow, ], size = 3) +
    ggtitle(main)

}
```

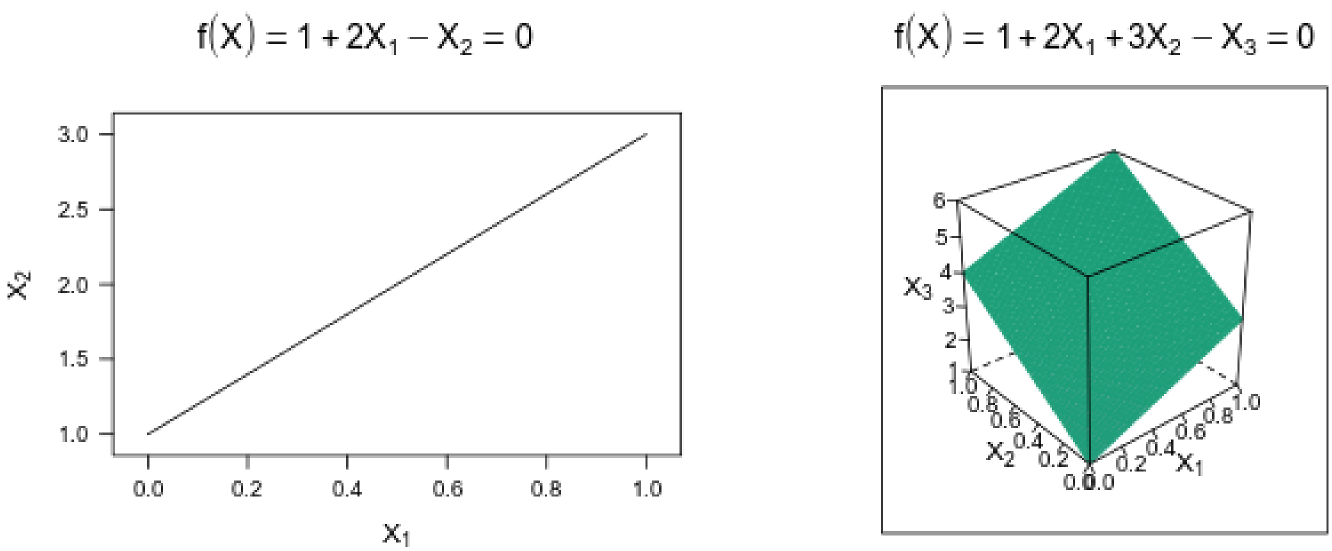
Support Vector Machines Overview

Support vector machines offer a direct approach to binary classification: try to find a hyperplane in some feature space that "best" separates the two classes.

Optimal Separating Hyperplanes

```r
# Construct data for plotting
x1 <- x2 <- seq(from = 0, to = 1, length = 100)
xgrid <- expand.grid(x1 = x1, x2 = x2)
y1 <- 1 + 2 * x1
y2 <- 1 + 2 * xgrid$x1 + 3 * xgrid$x2

# Hyperplane: p = 2
p1 <- lattice::xyplot(
  x = y1 ~ x1,
  type = "l",
  col = "black",
  xlab = expression(X[1]),
  ylab = expression(X[2]),
  main = expression({f(X)==1+2*X[1]-X[2]}==0),
```

```r
  scales = list(tck = c(1, 0))
)

# Hyperplane: p = 3
p2 <- lattice::wireframe(
  x = y2 ~ xgrid$x1 * xgrid$x2,
  xlab = expression(X[1]),
  ylab = expression(X[2]),
  zlab = expression(X[3]),
  main = expression({f(X)==1+2*X[1]+3*X[2]-X[3]}==0),
  drape = TRUE,
  colorkey = FALSE,
  col = dark2[1],
  scales = list(arrows = FALSE)
  # par.settings = list(axis.line = list(col = "transparent"))
)

# Display plots side by side
gridExtra::grid.arrange(p1, p2, nrow = 1)
```



Hard Margin Classifier

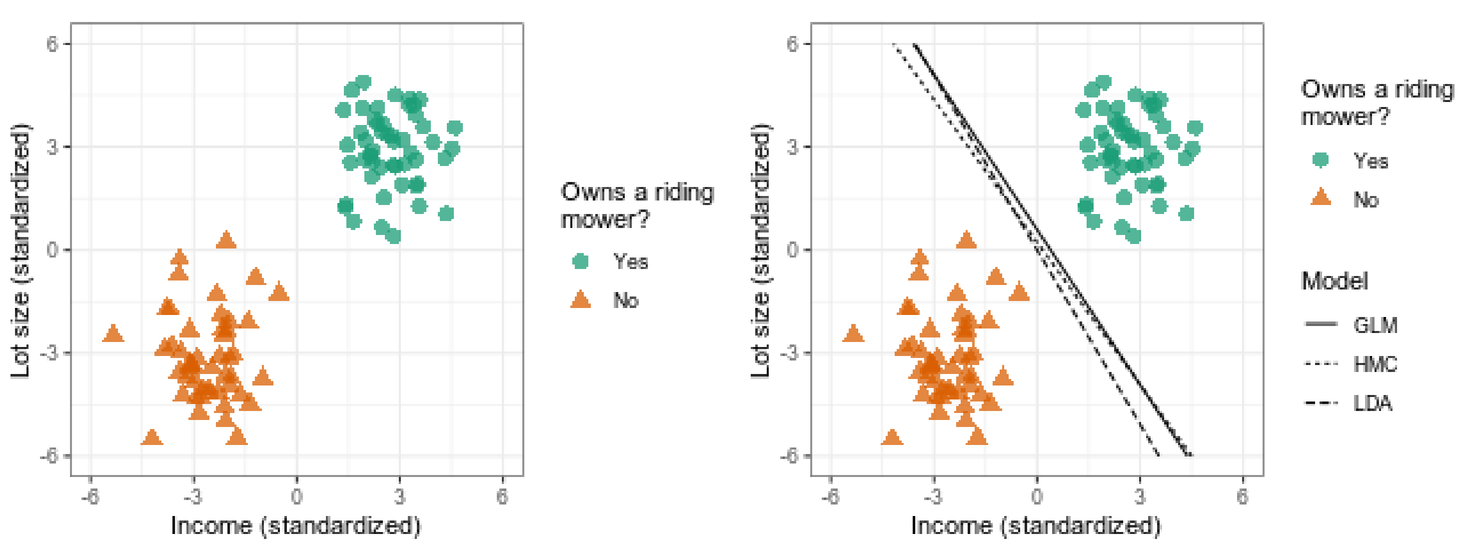
```r
# Simulate data
set.seed(805)
norm2d <- as.data.frame(mlbench::mlbench.2dnormals(
  n = 100,
  cl = 2,
```

```
  r = 4,
  sd = 1
))
names(norm2d) <- c("x1", "x2", "y")  # rename columns

# Scatterplot
p1 <- ggplot(norm2d, aes(x = x1, y = x2)) +
  geom_point(aes(shape = y, color = y), size = 3, alpha = 0.75) +
  xlab("Income (standardized)") +
  ylab("Lot size (standardized)") +
  xlim(-6, 6) +
  ylim(-6, 6) +
  coord_fixed() +
  theme(legend.position = "none") +
  theme_bw() +
  scale_shape_discrete(
    name = "Owns a riding\nmower?",
    breaks = c(1, 2),
    labels = c("Yes", "No")
  ) +
  scale_color_brewer(
    name = "Owns a riding\nmower?",
    palette = "Dark2",
    breaks = c(1, 2),
    labels = c("Yes", "No")
  )

# Fit a Logistic regression, linear discriminant analysis (LDA), and optimal
# separating hyperplane (OSH). Note: we sometimes refer to the OSH as the hard
# margin classifier
fit_glm <- glm(as.factor(y) ~ ., data = norm2d, family = binomial)
```

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

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

```
fit_lda <- MASS::lda(as.factor(y) ~ ., data = norm2d)
invisible(capture.output(fit_hmc <- ksvm(  # use ksvm() to find the OSH
  x = data.matrix(norm2d[c("x1", "x2")]),
  y = as.factor(norm2d$y),
  kernel = "vanilladot",  # no fancy kernel, just ordinary dot product
  C = Inf,                # to approximate hard margin classifier
  prob.model = TRUE       # needed to obtain predicted probabilities
)))
```

```r
# Grid over which to evaluate decision boundaries
npts <- 500
xgrid <- expand.grid(
  x1 = seq(from = -6, 6, length = npts),
  x2 = seq(from = -6, 6, length = npts)
)

# Predicted probabilities (as a two-column matrix)
prob_glm <- predict(fit_glm, newdata = xgrid, type = "response")
prob_glm <- cbind("1" = 1 - prob_glm, "2" = prob_glm)
prob_lda <- predict(fit_lda, newdata = xgrid)$posterior
prob_hmc <- predict(fit_hmc, newdata = xgrid, type = "probabilities")

# Add predicted class probabilities
xgrid2 <- xgrid %>%
  cbind("GLM" = prob_glm[, 1L],
        "LDA" = prob_lda[, 1L],
        "HMC" = prob_hmc[, 1L]) %>%
  tidyr::gather(Model, Prob, -x1, -x2)

# Scatterplot with decision boundaries
p2 <- p1 +
  stat_contour(data = xgrid2, aes(x = x1, y = x2, z = Prob, linetype = Model),
               breaks = 0.5, color = "black")

# Display plots side by side
gridExtra::grid.arrange(p1, p2, nrow = 1)
```

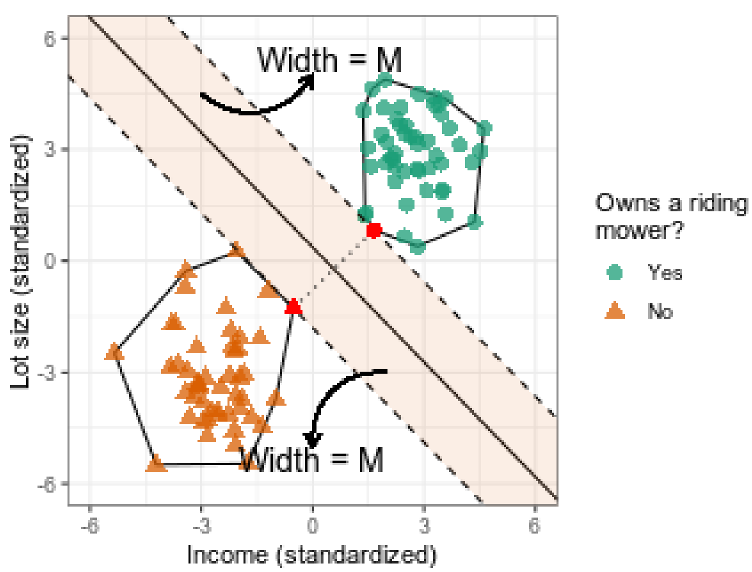


Outliers:

```r
# Compute convex hull for each class
hpts1 <- chull(norm2d[norm2d$y == 1, c("x1", "x2")])
hpts1 <- c(hpts1, hpts1[1L])
hpts2 <- chull(norm2d[norm2d$y == 2, c("x1", "x2")])
hpts2 <- c(hpts2, hpts2[1L])

# Support vectors
sv <- norm2d[fit_hmc@alphaindex[[1L]], c("x1", "x2")]  # 16-th and 97-th observations

# Compute the perpendicular bisector of the line segment joining the two support
# vectors
slope <- -1 / ((sv[2L, 2L] - sv[1L, 2L]) / (sv[2L, 1L] - sv[1L, 1L]))
midpoint <- apply(sv, 2, mean)

# Scatterplot with convex hulls, etc.
ggplot(norm2d, aes(x = x1, y = x2)) +

  # Convex hulls
  geom_polygon(
    data = norm2d[norm2d$y == 1, c("x1", "x2")][hpts1, c("x1", "x2")],
    color = "black",
    fill = "transparent"
  ) +
  geom_polygon(
    data = norm2d[norm2d$y == 2, c("x1", "x2")][hpts2, c("x1", "x2")],
    color = "black",
    fill = "transparent"
  ) +

  # Scatterplot
  geom_point(aes(shape = y, color = y), size = 3, alpha = 0.75) +
  xlab("Income (standardized)") +
  ylab("Lot size (standardized)") +
  xlim(-10, 10) +
  ylim(-10, 10) +
  # coord_fixed() +
  theme(legend.position = "none") +
  theme_bw() +
  scale_shape_discrete(
    name = "Owns a riding\nmower?",
    breaks = c(1, 2),
    labels = c("Yes", "No")
  ) +
  scale_color_brewer(
```

```
    name = "Owns a riding\nmower?",
    palette = "Dark2",
    breaks = c(1, 2),
    labels = c("Yes", "No")
  ) +

  # Decision boundary
  geom_abline(
    intercept = -slope * midpoint[1L] + midpoint[2L],
    slope = slope
  ) +

  # Margin boundaries (shaded in)
  geom_abline(
    intercept = -slope * sv[1L, 1L] + sv[1L, 2L],
    slope = slope,
    linetype = 2
  ) +
  geom_abline(
    intercept = -slope * sv[2L, 1L] + sv[2L, 2L],
    slope = slope,
    linetype = 2
  ) +
  annotate(
    geom = "polygon",
    x = c(-7, -7, 7, 7),
    y = c(-slope * sv[1L, 1L] + sv[1L, 2L] - 7 * slope,
          -slope * midpoint[1L] + midpoint[2L] - 7 * slope,
          -slope * midpoint[1L] + midpoint[2L] + 7 * slope,
          -slope * sv[1L, 1L] + sv[1L, 2L] + 7 * slope),
    alpha = 0.1,
    color = "transparent",
    fill = dark2[2]
  ) +
  annotate(
    geom = "polygon",
    x = c(-7, -7, 7, 7),
    y = c(-slope * sv[2L, 1L] + sv[2L, 2L] - 7 * slope,
          -slope * midpoint[1L] + midpoint[2L] - 7 * slope,
          -slope * midpoint[1L] + midpoint[2L] + 7 * slope,
          -slope * sv[2L, 1L] + sv[2L, 2L] + 7 * slope),
    alpha = 0.1,
    color = "transparent",
    fill = dark2[2]
```

```
) +

# Arrows, labels, etc.
annotate("segment",
  x = sv[1L, 1L], y = sv[1L, 2L], xend = sv[2L, 1L], yend = sv[2L, 2L],
  # alpha = 0.5,
  linetype = 3
  # arrow = arrow(length = unit(0.03, units = "npc"), ends = "both")
) +
geom_curve(x = -3, y = 4.5, xend = 0, yend = 5,
           arrow = arrow(length = unit(0.03, units = "npc"))) +
annotate("text", label = "Width = M", x = 0.45, y = 5.45, size = 5) +
geom_curve(x = 2, y = -3, xend = 0, yend = -5,
           arrow = arrow(length = unit(0.03, units = "npc"))) +
annotate("text", label = "Width = M", x = 0, y = -5.35, size = 5) +

# Support vectors
annotate("point", x = sv$x1[1], y = sv$x2[1], shape = 17, color = "red",
         size = 3) +
annotate("point", x = sv$x1[2], y = sv$x2[2], shape = 16, color = "red",
         size = 3) +
# geom_point(data = cbind(sv, y = c("2", "1")), aes(shape = y),
#            size = 4, color = "red") +

# Zoom in
coord_fixed(xlim = c(-6, 6), ylim = c(-6, 6))
```



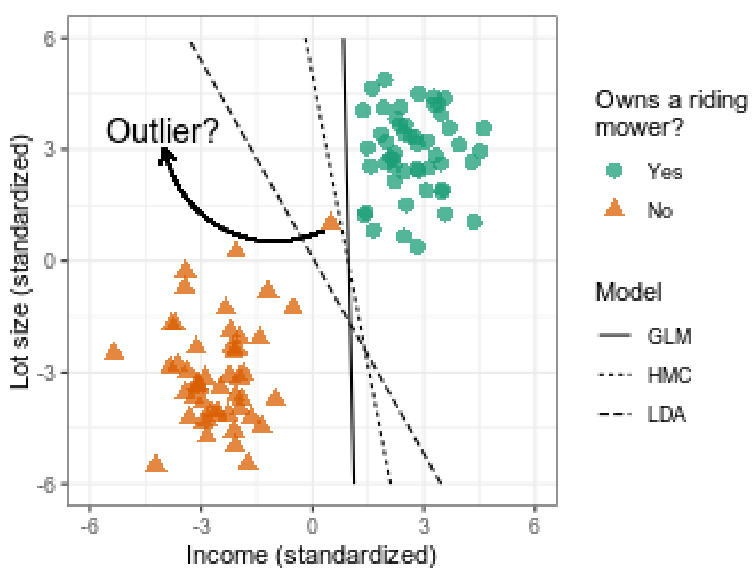
Soft Margin Classifier:

```r
# Add an outlier
norm2d <- rbind(norm2d, data.frame("x1" = 0.5, "x2" = 1, "y" = 2))

# Fit a Logistic regression, linear discriminant analysis (LDA), and optimal
# separating hyperplane (OSH)
#
# Note: we sometimes refer to the OSH as the hard margin classifier
fit_glm <- glm(as.factor(y) ~ ., data = norm2d, family = binomial)
```

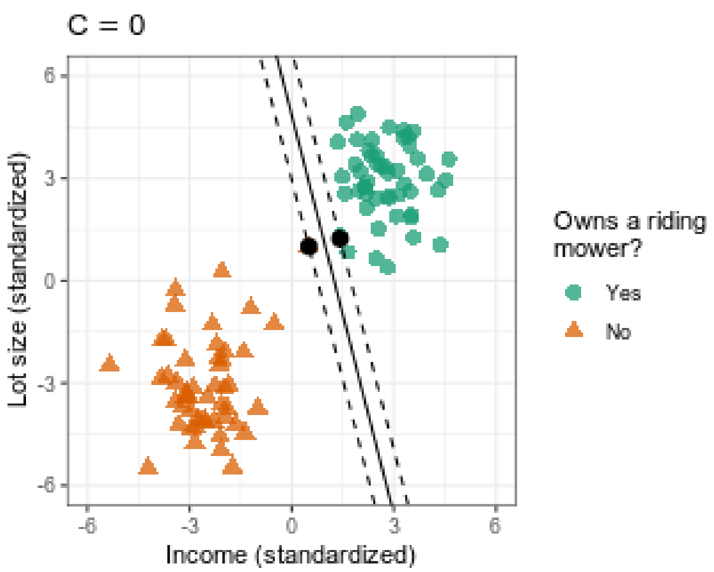
```
Warning: glm.fit: algorithm did not converge

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

```r
fit_lda <- MASS::lda(as.factor(y) ~ ., data = norm2d)
invisible(capture.output(fit_hmc <- ksvm(  # use ksvm() to find the OSH
  x = data.matrix(norm2d[c("x1", "x2")]),
  y = as.factor(norm2d$y),
  kernel = "vanilladot",  # no fancy kernel, just ordinary dot product
  C = Inf,                # to approximate maximal margin classifier
  prob.model = TRUE       # needed to obtain predicted probabilities
)))

# Grid over which to evaluate decision boundaries
npts <- 500
xgrid <- expand.grid(
  x1 = seq(from = -6, 6, length = npts),
  x2 = seq(from = -6, 6, length = npts)
)

# Predicted probabilities (as a two-column matrix)
prob_glm <- predict(fit_glm, newdata = xgrid, type = "response")
prob_glm <- cbind("1" = 1 - prob_glm, "2" = prob_glm)
prob_lda <- predict(fit_lda, newdata = xgrid)$posterior
prob_hmc <- predict(fit_hmc, newdata = xgrid, type = "probabilities")

# Add predicted class probabilities
xgrid2 <- xgrid %>%
  cbind("GLM" = prob_glm[, 1L],
        "LDA" = prob_lda[, 1L],
        "HMC" = prob_hmc[, 1L]) %>%
  tidyr::gather(Model, Prob, -x1, -x2)

# Scatterplot
ggplot(norm2d, aes(x = x1, y = x2)) +
```

```
  # Label outlier
  geom_curve(x = tail(norm2d, n = 1)$x1 - 0.2, y = tail(norm2d, n = 1)$x2 - 0.2,
             xend = -4, yend = 3, curvature = -0.5, angle = 90,
             arrow = arrow(length = unit(0.03, units = "npc"))) +
  annotate("text", label = "Outlier?", x = -4, y = 3.5, size = 5) +

  # Scatterplot, etc.
  geom_point(aes(shape = y, color = y), size = 3, alpha = 0.75) +
  xlab("Income (standardized)") +
  ylab("Lot size (standardized)") +
  xlim(-6, 6) +
  ylim(-6, 6) +
  coord_fixed() +
  theme(legend.position = "none") +
  theme_bw() +
  scale_shape_discrete(
    name = "Owns a riding\nmower?",
    breaks = c(1, 2),
    labels = c("Yes", "No")
  ) +
  scale_color_brewer(
    name = "Owns a riding\nmower?",
    palette = "Dark2",
    breaks = c(1, 2),
    labels = c("Yes", "No")
  ) +
  stat_contour(data = xgrid2, aes(x = x1, y = x2, z = Prob, linetype = Model),
               breaks = 0.5, color = "black")
```



```
# Fit the entire regularization path
fit_smc <- svmpath(
  x = data.matrix(norm2d[c("x1", "x2")]),
  y = ifelse(norm2d$y == 1, 1, -1)
)

# Plot both extremes
p1 <- plot_svmpath(fit_smc, step = max(fit_smc$Step), main = expression(C == 0))
p2 <- plot_svmpath(fit_smc, step = min(fit_smc$Step), main = expression(C == infinity))
gridExtra::grid.arrange(p1, p2, nrow = 1)
```



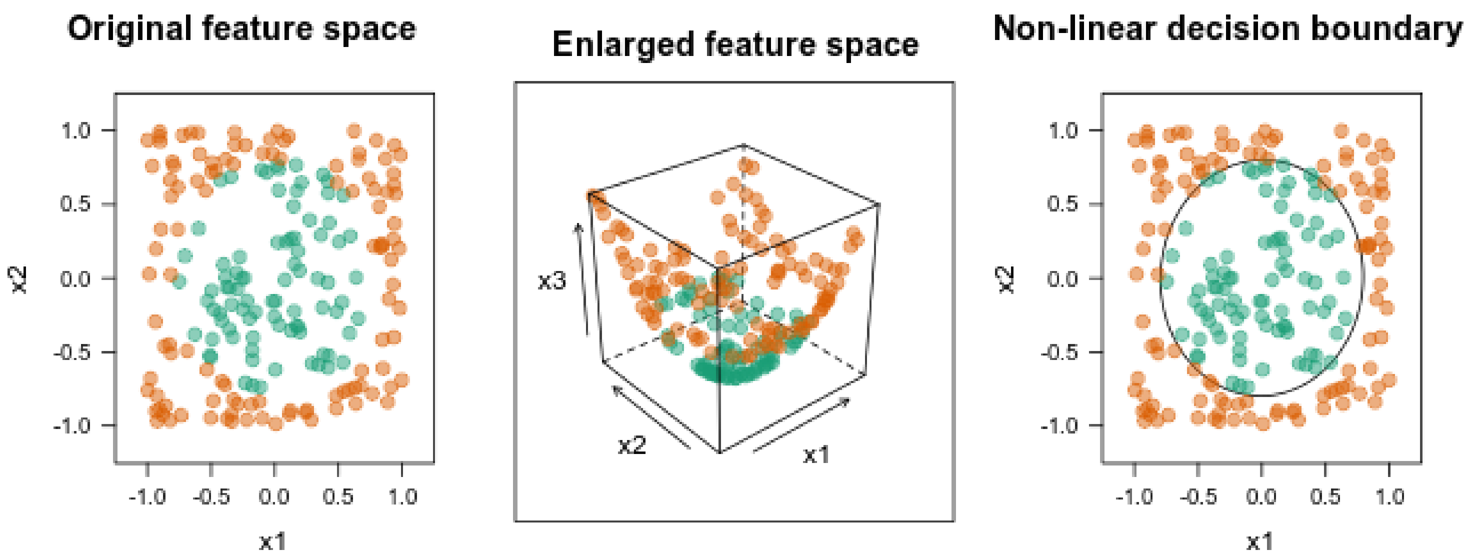
Support Vector Machine

```
# Load required packages
library(grid)
library(lattice)

# Simulate data
set.seed(1432)
circle <- as.data.frame(mlbench::mlbench.circle(
  n = 200,
  d = 2
))
names(circle) <- c("x1", "x2", "y")  # rename columns

# Fit a support vector machine (SVM)
fit_svm_poly <- ksvm(
```

```
  x = data.matrix(circle[c("x1", "x2")]),
  y = as.factor(circle$y),
  kernel = "polydot",       # polynomial kernel
  kpar = list(degree = 2),  # kernel parameters
  C = Inf,                  # to approximate maximal margin classifier
  prob.model = TRUE         # needed to obtain predicted probabilities
)

# Grid over which to evaluate decision boundaries
npts <- 500
xgrid <- expand.grid(
  x1 = seq(from = -1.25, 1.25, length = npts),
  x2 = seq(from = -1.25, 1.25, length = npts)
)

# Predicted probabilities (as a two-column matrix)
prob_svm_poly <- predict(fit_svm_poly, newdata = xgrid, type = "probabilities")

# Scatterplot
p1 <- contourplot(
  x = prob_svm_poly[, 1] ~ x1 * x2,
  data = xgrid,
  at = 0,
  labels = FALSE,
  scales = list(tck = c(1, 0)),
  xlab = "x1",
  ylab = "x2",
  main = "Original feature space",
  panel = function(x, y, z, ...) {
    panel.contourplot(x, y, z, ...)
    panel.xyplot(
      x = circle$x1,
      y = circle$x2,
      groups = circle$y,
      pch = 19,
      cex = 1,
      col = adjustcolor(dark2[1L:2L], alpha.f = 0.5),
      ...
    )
  }
)

# Enlarge feature space
circle_3d <- circle
```

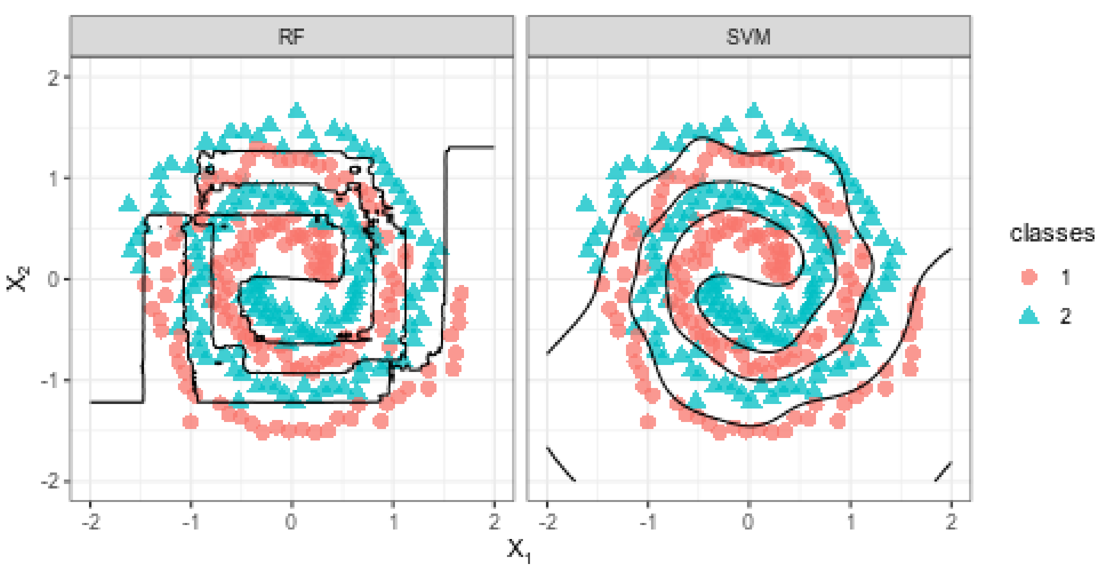
```r
circle_3d$x3 <- circle_3d$x1^2 + circle_3d$x2^2

# 3-D scatterplot
p2 <- cloud(
  x = x3 ~ x1 * x2,
  data = circle_3d,
  groups = y,
  main = "Enlarged feature space",
  par.settings = list(
    superpose.symbol = list(
      pch = 19,
      cex = 1,
      col = adjustcolor(dark2[1L:2L], alpha.f = 0.5)
    )
  )
)

# p2 <- scatterplot3d(
#   x = circle_3d[, -3],
#   pch = 19,
#   color = adjustcolor(dark2[1L:2L], alpha.f = 0.5)[circle_3d$y]
# )
# p2$plane3d(0.64, 0, 0, draw_polygon = TRUE)
# p2 <- recordPlot()

# Scatterplot with decision boundary
p3 <- contourplot(
  x = prob_svm_poly[, 1] ~ x1 * x2,
  data = xgrid,
  at = 0.5,
  labels = FALSE,
  scales = list(tck = c(1, 0)),
  xlab = "x1",
  ylab = "x2",
  main = "Non-linear decision boundary",
  panel = function(x, y, z, ...) {
    panel.contourplot(x, y, z, ...)
    panel.xyplot(
      x = circle$x1,
      y = circle$x2,
      groups = circle$y,
      pch = 19,
      cex = 1,
      col = adjustcolor(dark2[1L:2L], alpha.f = 0.5),
```

```
      ...
    )
  }
)

# Combine plots
gridExtra::grid.arrange(p1, p2, p3, nrow = 1)
```




```
# Load required packages
library(kernlab)  # for fitting SVMs
library(mlbench)  # for ML benchmark data sets

# Simulate train and test sets
set.seed(0841)
spirals <- as.data.frame(
  mlbench.spirals(300, cycles = 2, sd = 0.09)
)
names(spirals) <- c("x1", "x2", "classes")

# Fit an RF
set.seed(7256)
spirals_rfo <- ranger::ranger(classes ~ ., data = spirals, probability = TRUE)

# Fit an SVM using a radial basis function kernel
spirals_svm <- ksvm(classes ~ x1 + x2, data = spirals, kernel = "rbfdot",
                    C = 500, prob.model = TRUE)

# Grid over which to evaluate decision boundaries
npts <- 500
```

```r
xgrid <- expand.grid(
  x1 = seq(from = -2, 2, length = npts),
  x2 = seq(from = -2, 2, length = npts)
)

# Predicted probabilities (as a two-column matrix)
prob_rfo <- predict(spirals_rfo, data = xgrid)$predictions
prob_svm <- predict(spirals_svm, newdata = xgrid, type = "probabilities")

# Add predicted class probabilities
xgrid2 <- xgrid %>%
  cbind("RF" = prob_rfo[, 1L],
        "SVM" = prob_svm[, 1L]) %>%
  tidyr::gather(Model, Prob, -x1, -x2)

# Scatterplots with decision boundaries
ggplot(spirals, aes(x = x1, y = x2)) +
  geom_point(aes(shape = classes, color = classes), size = 3, alpha = 0.75) +
  xlab(expression(X[1])) +
  ylab(expression(X[2])) +
  xlim(-2, 2) +
  ylim(-2, 2) +
  coord_fixed() +
  theme(legend.position = "none") +
  theme_bw() +
  stat_contour(data = xgrid2, aes(x = x1, y = x2, z = Prob),
               breaks = 0.5, color = "black") +
  facet_wrap( ~ Model)
```



```r
# Linear (i.e., soft margin classifier)
caret::getModelInfo("svmLinear")$svmLinear$parameters
```

```
  parameter   class label
1         C numeric  Cost
```

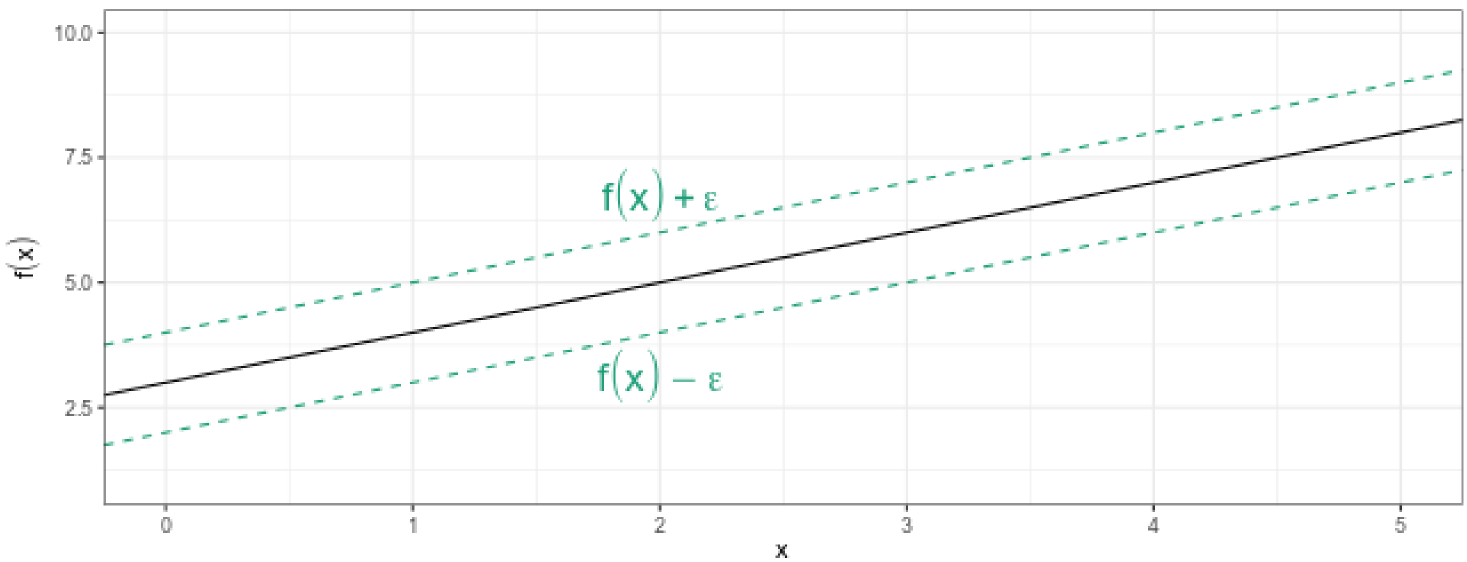
```r
# Polynomial kernel
caret::getModelInfo("svmPoly")$svmPoly$parameters
```

```
  parameter   class             label
1    degree numeric Polynomial Degree
2     scale numeric             Scale
3         C numeric              Cost
```

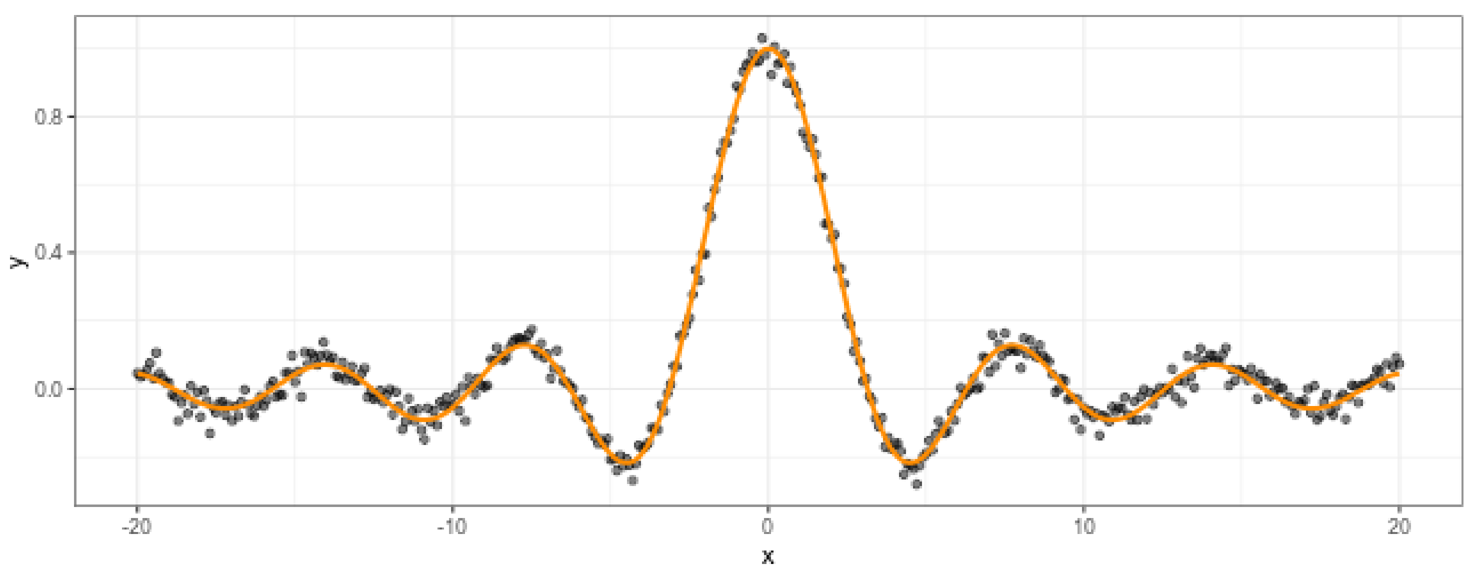
```r
# Radial basis kernel
caret::getModelInfo("svmRadial")$svmRadial$parameters
```

```
  parameter   class label
1     sigma numeric Sigma
2         C numeric  Cost
```

```r
ggplot() +
  geom_abline(intercept = 4, slope = 1, linetype = 2, color = dark2[1L]) +
  geom_abline(intercept = 3, slope = 1) +
  geom_abline(intercept = 2, slope = 1, linetype = 2, color = dark2[1L]) +
  xlim(0, 5) +
  ylim(1, 10) +
  xlab(expression(x)) +
  ylab(expression(f(x))) +
  theme_bw() +
  annotate("text", label = "f(x) + epsilon", parse = TRUE, x = 2, y = 6.75,
           size = 6, color = dark2[1L]) +
  annotate("text", label = "f(x) - epsilon", parse = TRUE, x = 2, y = 3.15,
           size = 6, color = dark2[1L])
```



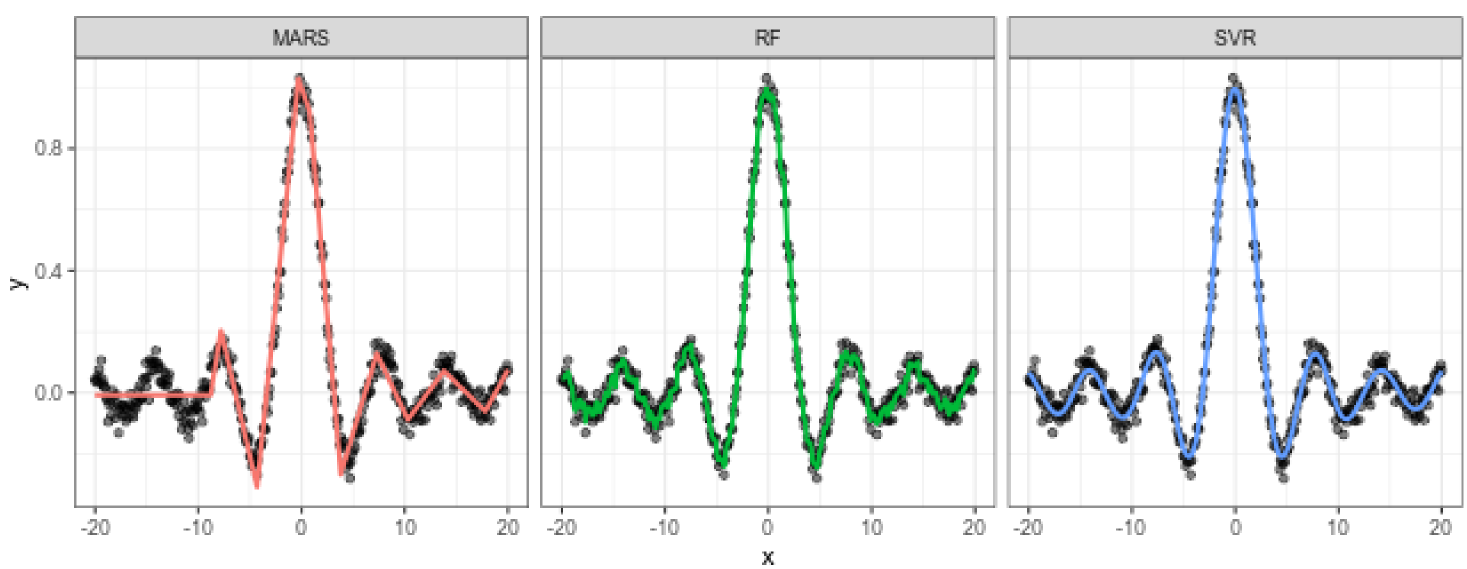

```r
# Simulate data
set.seed(1218)
x <- seq(from = -20, to = 20, by = 0.1)
y <- sin(x) / x + rnorm(length(x), sd = 0.03)
df <- na.omit(data.frame(x = x, y = y))

# Plot results
ggplot(df, aes(x = x, y = y)) +
  geom_point(alpha = 0.5) +
  geom_line(aes(x = x, y = sin(x) / x), size = 1, color = "darkorange") +
  theme_bw() +
  theme(legend.position = "none")
```



```r
# SVR model
set.seed(101)
svr <- kernlab::ksvm(y ~ x, data = df, kernel = "rbfdot", kpar = "automatic",
                     type = "eps-svr", epsilon = 0.1)

# MARS model
mars <- earth::earth(y ~ x, data = df)

# Random forest
set.seed(102)
rfo <- ranger::ranger(y ~ x, data = df)

# Gather predictions
df$SVR <- predict(svr, newdata = df)
df$MARS <- predict(mars, newdata = df)[, 1L, drop = TRUE]
df$RF <- predict(rfo, data = df)$predictions
df <- df %>% tidyr::gather(Model, Prediction, -x, -y)
```

```
Warning: attributes are not identical across measure variables;
they will be dropped
```

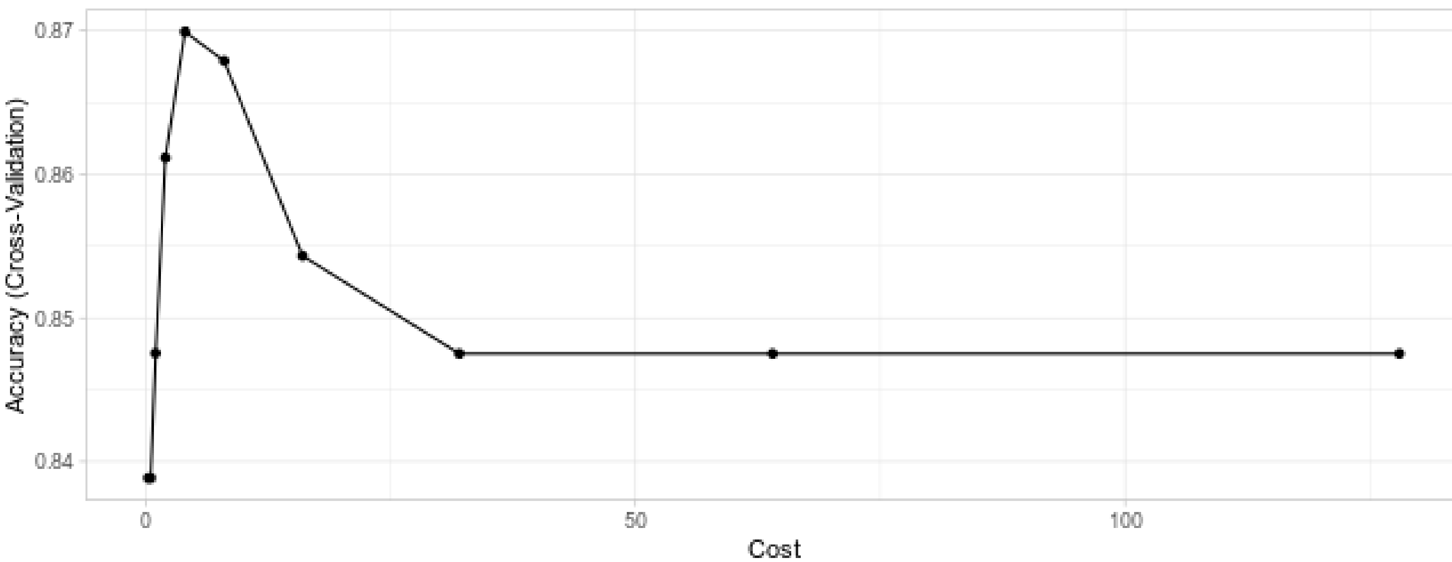
```r
# Plot results
ggplot(df, aes(x = x, y = y)) +
  geom_point(alpha = 0.5) +
  geom_line(aes(x = x, y = Prediction, color = Model), size = 1) +
  facet_wrap( ~ Model) +
  theme_bw() +
  theme(legend.position = "none")
```



Job Attrition Example:

```r
# Tune an SVM with radial basis kernel
set.seed(1854)  # for reproducibility
churn_svm <- train(
  Attrition ~ .,
  data = churn_train,
  method = "svmRadial",
  preProcess = c("center", "scale"),
  trControl = trainControl(method = "cv", number = 10),
  tuneLength = 10
)
```

```r
# Plot results
ggplot(churn_svm) + theme_light()
```




```r
# Print results
churn_svm$results
```

```
        sigma      C  Accuracy      Kappa  AccuracySD    KappaSD
1  0.01005408   0.25 0.8388542 0.00000000 0.004089627 0.00000000
2  0.01005408   0.50 0.8388542 0.00000000 0.004089627 0.00000000
3  0.01005408   1.00 0.8475454 0.09127657 0.005447000 0.07288227
4  0.01005408   2.00 0.8611572 0.28510729 0.011312180 0.07014162
5  0.01005408   4.00 0.8699240 0.41093848 0.029561058 0.12280243
6  0.01005408   8.00 0.8678973 0.41775948 0.031362598 0.13526394
7  0.01005408  16.00 0.8543230 0.37631668 0.028185467 0.11356374
8  0.01005408  32.00 0.8475170 0.34804689 0.026278875 0.10979794
9  0.01005408  64.00 0.8475170 0.34804689 0.026278875 0.10979794
10 0.01005408 128.00 0.8475170 0.34804689 0.026278875 0.10979794
```

Class Probabilities:

```r
# Control params for SVM
ctrl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = twoClassSummary  # also needed for AUC/ROC
)

# Tune an SVM
set.seed(5628)  # for reproducibility
churn_svm_auc <- train(
  Attrition ~ .,
  data = churn_train,
  method = "svmRadial",
  preProcess = c("center", "scale"),
  metric = "ROC",  # area under ROC curve (AUC)
  trControl = ctrl,
  tuneLength = 10
)

# Print results
churn_svm_auc$results
```

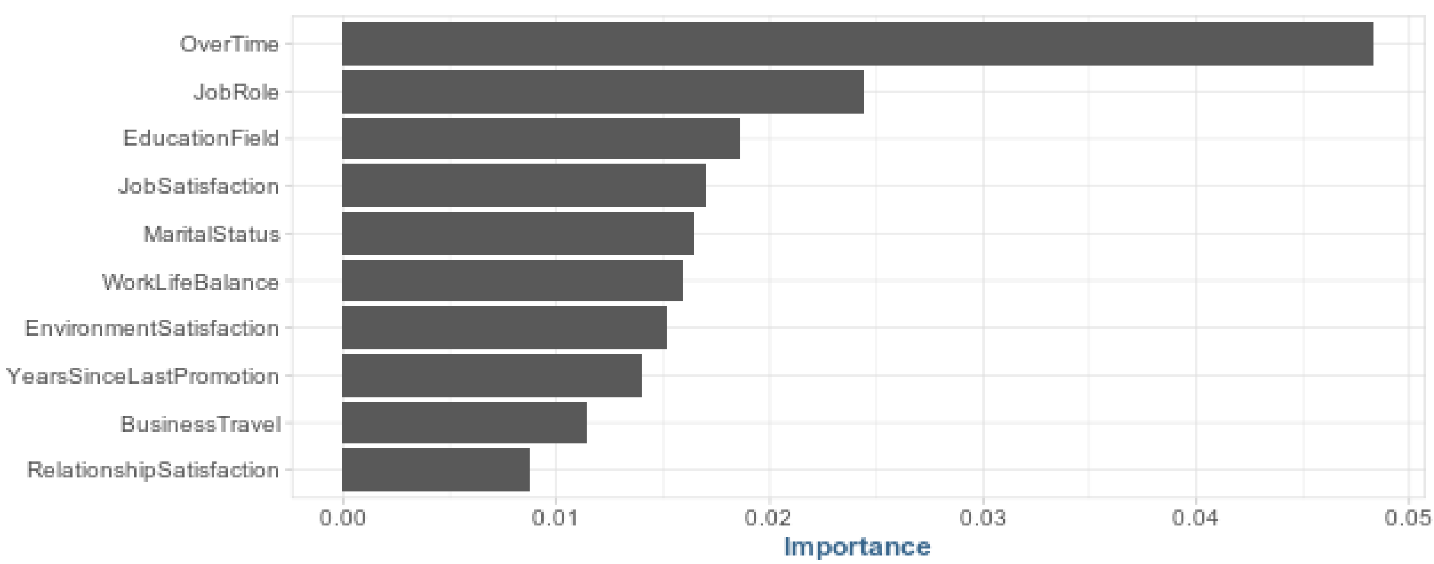
```
         sigma      C       ROC      Sens      Spec      ROCSD      SensSD
1  0.009768359   0.25 0.8036321 0.9733761 0.3327206 0.08290705 0.014551571
2  0.009768359   0.50 0.8037723 0.9676022 0.3448529 0.08292266 0.015260674
3  0.009768359   1.00 0.8038400 0.9721999 0.3268382 0.08291134 0.011283552
4  0.009768359   2.00 0.8033163 0.9757017 0.3202206 0.08161873 0.006547323
5  0.009768359   4.00 0.7945398 0.9849639 0.2841912 0.08476596 0.005549910
6  0.009768359   8.00 0.7801340 0.9849639 0.2371324 0.08180350 0.007756353
7  0.009768359  16.00 0.7611333 0.9861133 0.2058824 0.07492765 0.004868325
8  0.009768359  32.00 0.7502869 0.9849639 0.2113971 0.07433846 0.005549910
9  0.009768359  64.00 0.7501168 0.9838011 0.2058824 0.07354252 0.009777944
10 0.009768359 128.00 0.7501168 0.9826383 0.2176471 0.07354252 0.009864975
       SpecSD
1  0.07799453
2  0.10422864
3  0.10054174
4  0.09316063
5  0.11040368
6  0.13095267
7  0.09760935
8  0.06784060
9  0.09411573
10 0.07998237
```

```r
confusionMatrix(churn_svm_auc)
```

```
Cross-Validated (10 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction   No  Yes
       No  81.6 10.9
       Yes  2.3  5.2

 Accuracy (average) : 0.868
```

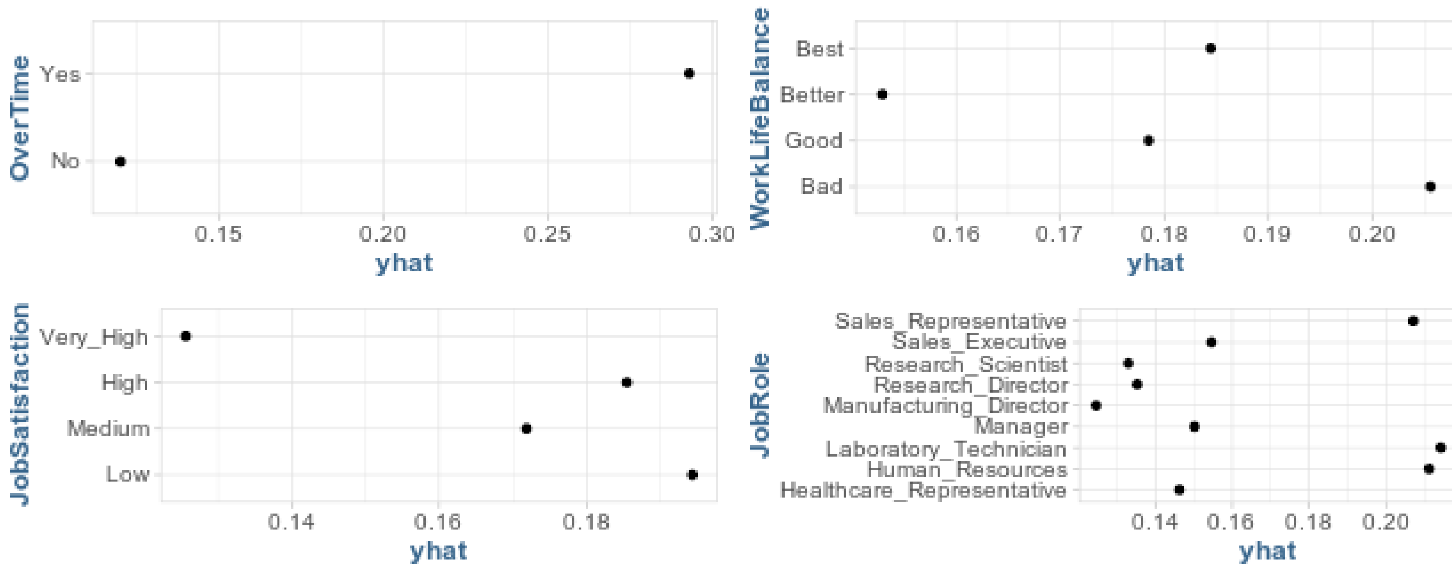
```r
prob_yes <- function(object, newdata) {
  predict(object, newdata = newdata, type = "prob")[, "Yes"]
}
```

```r
# Variable importance plot
set.seed(2827)  # for reproducibility
vip(churn_svm_auc, method = "permute", nsim = 5, train = churn_train,
    target = "Attrition", metric = "auc", reference_class = "Yes",
    pred_wrapper = prob_yes)
```




```r
features <- c("OverTime", "WorkLifeBalance",
              "JobSatisfaction", "JobRole")
pdps <- lapply(features, function(x) {
  pdp::partial(churn_svm_auc, pred.var = x, which.class = 2,
        prob = TRUE, plot = TRUE, plot.engine = "ggplot2") +
    coord_flip()
})
```

```r
grid.arrange(grobs = pdps,  ncol = 2)
```




```r
h2o.shutdown(prompt = FALSE)
```

```
[1] TRUE
```

```r
# clean up
rm(list = ls())
```