Support Vector Machines

Yifei Sun

Contents

```r
library(mlbench)
library(ISLR)

library(caret) #only allows implementation from kernel lab lib
library(e1071)
library(kernlab)

library(DALEX) #generic, can be applied to lots of models
```

Classification

We use the Pima Indians Diabetes Database for illustration. The data contain 768 observations and 9 variables. The outcome is a binary variable `diabetes`.

```r
data(PimaIndiansDiabetes)
dat <- PimaIndiansDiabetes
dat$diabetes <- factor(dat$diabetes, c("pos", "neg"))

set.seed(2021)
rowTrain <- createDataPartition(y = dat$diabetes,
                                p = 0.75,
                                list = FALSE)
```

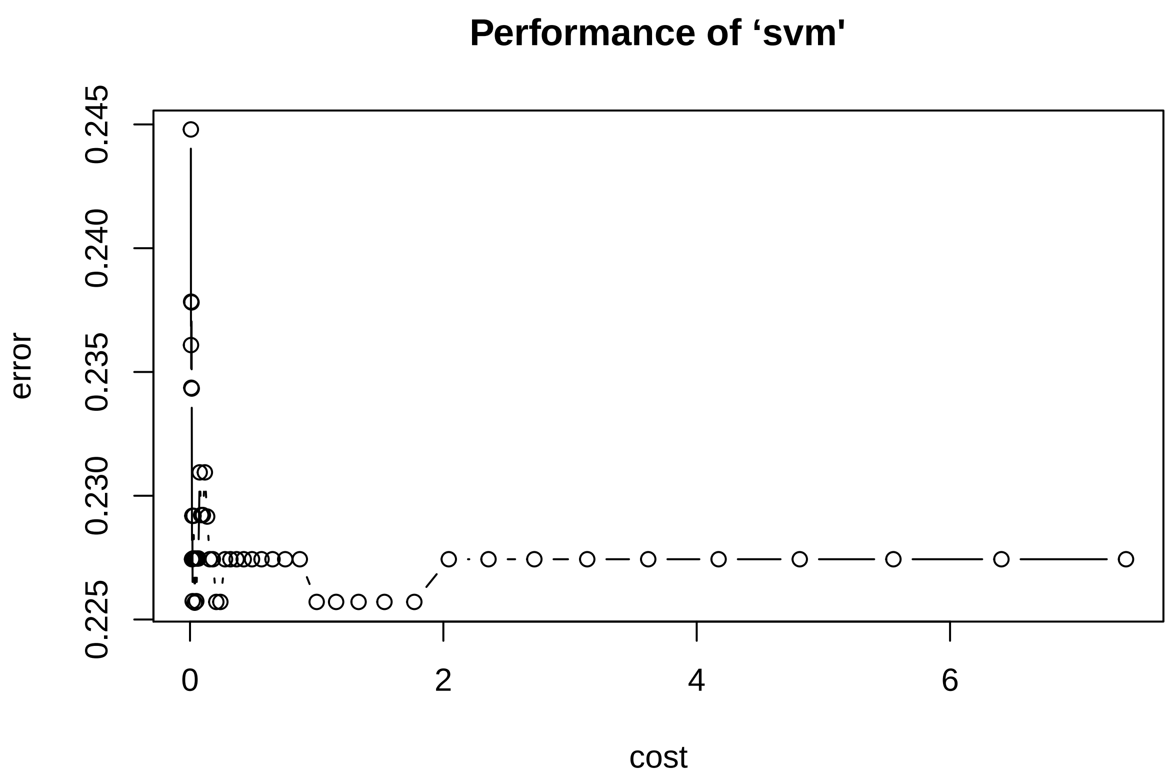
Using `e1071`

Check https://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf for more details.

Linear boundary

Most real data sets will not be fully separable by a linear boundary. Support vector classifiers with a tuning parameter `cost`, which quantifies the penalty associated with having an observation on the wrong side of the classification boundary, can be used to build a linear boundary.

```r
set.seed(1)
linear.tune <- tune.svm(diabetes ~ . ,
                        data = dat[rowTrain,],
                        kernel = "linear", #give us linear decision boundary.
                        cost = exp(seq(-5,2,len=50)),#tuning parameter
                        scale = TRUE) #recommend scaling for svm
# another argument: tunecontrol = tune.control (cross = 10)

plot(linear.tune)
```



```
# summary(linear.tune)
linear.tune$best.parameters #smallest cost => model is less flexible
```

```
##          cost
## 13 0.03741385
```

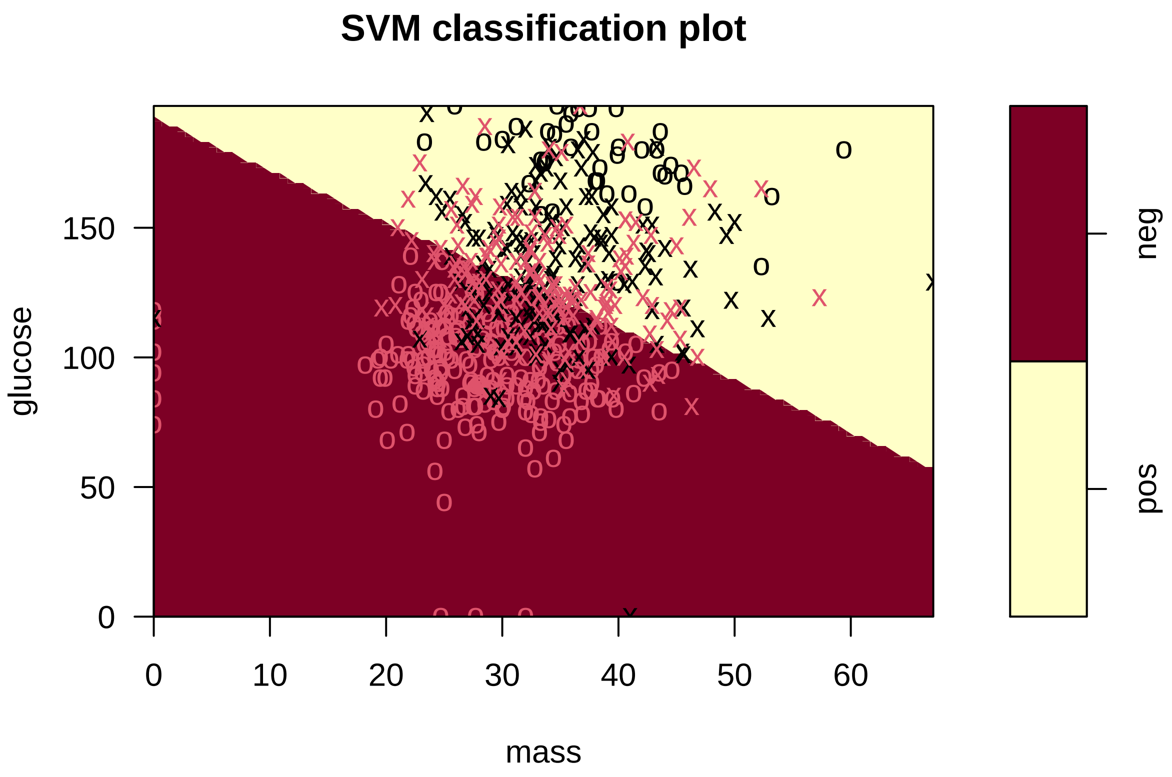
```
best.linear <- linear.tune$best.model
summary(best.linear)
```

```
##
## Call:
## best.svm(x = diabetes ~ ., data = dat[rowTrain, ], cost = exp(seq(-5,
##     2, len = 50)), kernel = "linear", scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.03741385
##
## Number of Support Vectors:  324
##
##  ( 160 164 )
##
##
## Number of Classes:  2
##
## Levels:
##  pos neg
```

```r
pred.linear <- predict(best.linear, newdata = dat[-rowTrain,])

confusionMatrix(data = pred.linear,
                reference = dat$diabetes[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction pos neg
##        pos  39  15
##        neg  28 110
##
##                Accuracy : 0.776
##                  95% CI : (0.7104, 0.8329)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 0.0001183
##
##                   Kappa : 0.4839
##
##  Mcnemar's Test P-Value : 0.0672525
##
##             Sensitivity : 0.5821
##             Specificity : 0.8800
##          Pos Pred Value : 0.7222
##          Neg Pred Value : 0.7971
##              Prevalence : 0.3490
##          Detection Rate : 0.2031
##    Detection Prevalence : 0.2812
##       Balanced Accuracy : 0.7310
##
##        'Positive' Class : pos
##
```

```r
# we have 8 predictors but can only plot 2 at once
plot(best.linear, dat[rowTrain,],
     glucose ~ mass,
     slice = list(pregnant = 5, triceps = 20,
                  insulin = 20, pressure = 75,
                  pedigree = 1, age = 50),
     grid = 100) #higher grid the more defined line
```

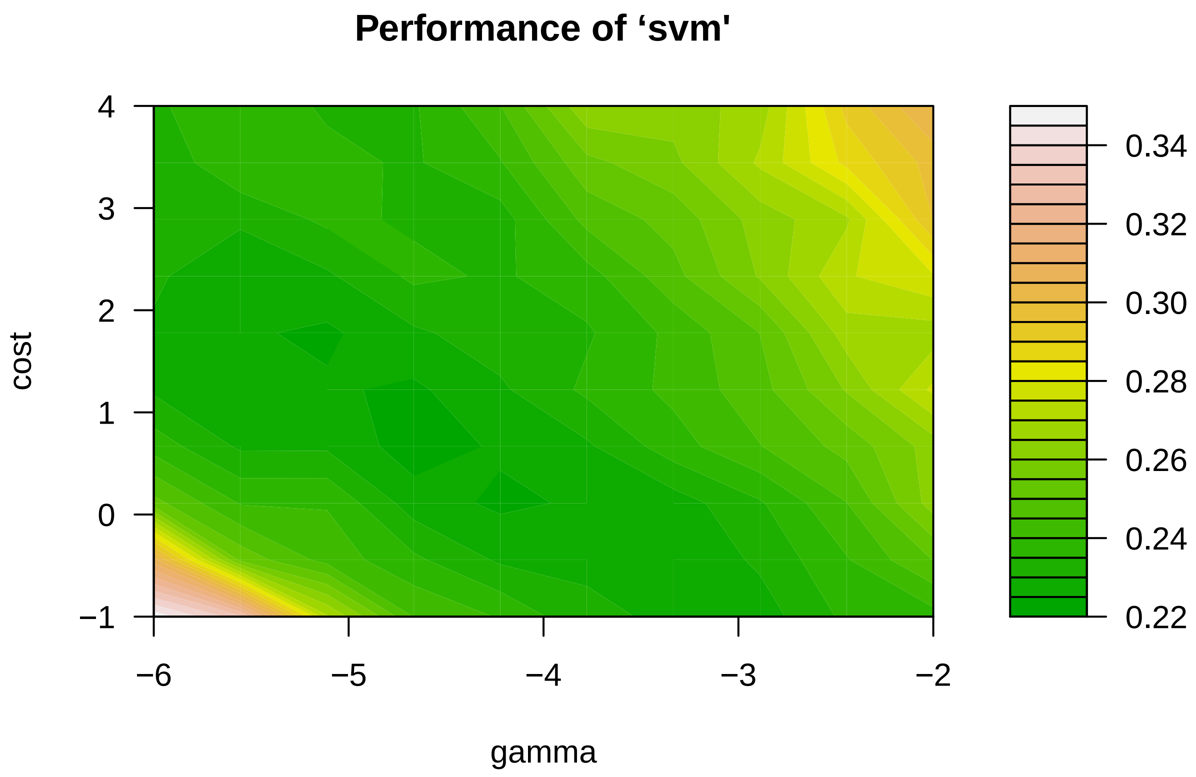


```
#x obs are support vectors => critical in decide decision boundary
```

Radial kernel RBF

Support vector machines can construct classification boundaries that are nonlinear in shape. We use the radial kernel.

```
set.seed(1)
radial.tune <- tune.svm(diabetes ~ . ,
                        data = dat[rowTrain,],
                        kernel = "radial",
                        cost = exp(seq(-1,4,len=10)),
                        gamma = exp(seq(-6,-2,len=10)))

plot(radial.tune, transform.y = log, transform.x = log,
     color.palette = terrain.colors) #green means misclassification is lower
```



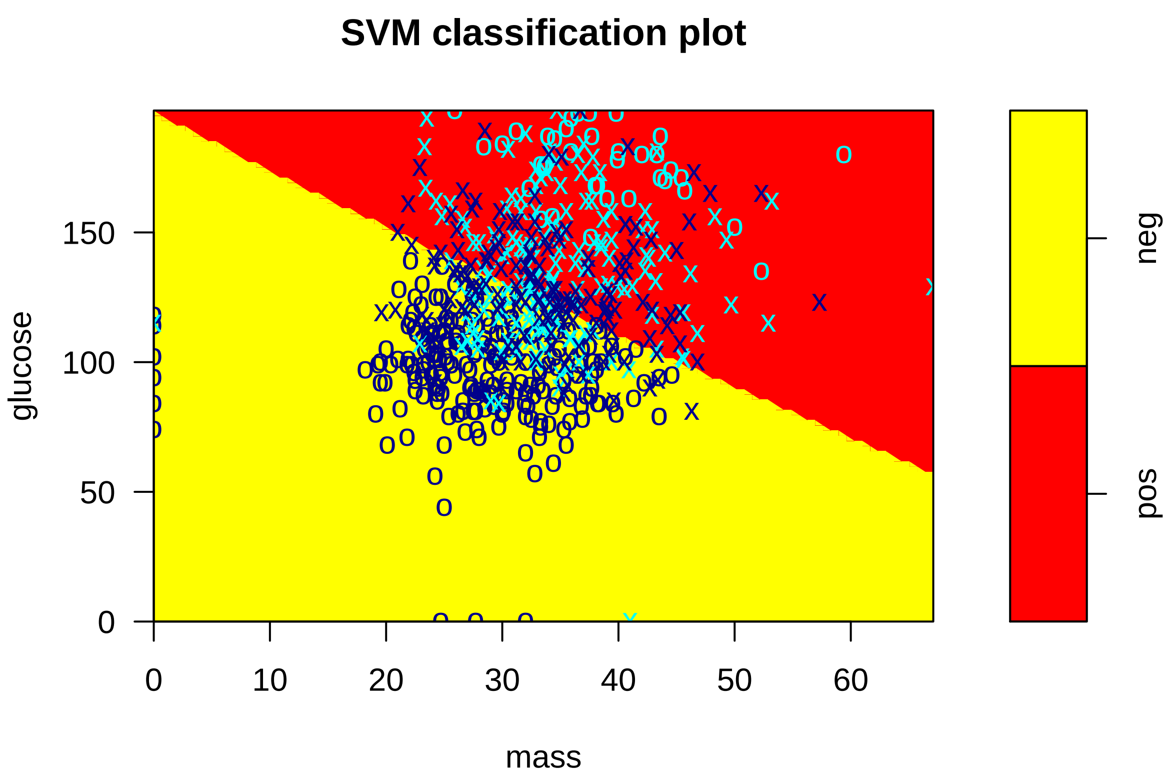
```r
# summary(radial.tune)

best.radial <- radial.tune$best.model
summary(best.radial)
```

```
##
## Call:
## best.svm(x = diabetes ~ ., data = dat[rowTrain, ], gamma = exp(seq(-6,
##     -2, len = 10)), cost = exp(seq(-1, 4, len = 10)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1.947734
##
## Number of Support Vectors:  327
##
##  ( 164 163 )
##
##
## Number of Classes:  2
##
## Levels:
##  pos neg
```

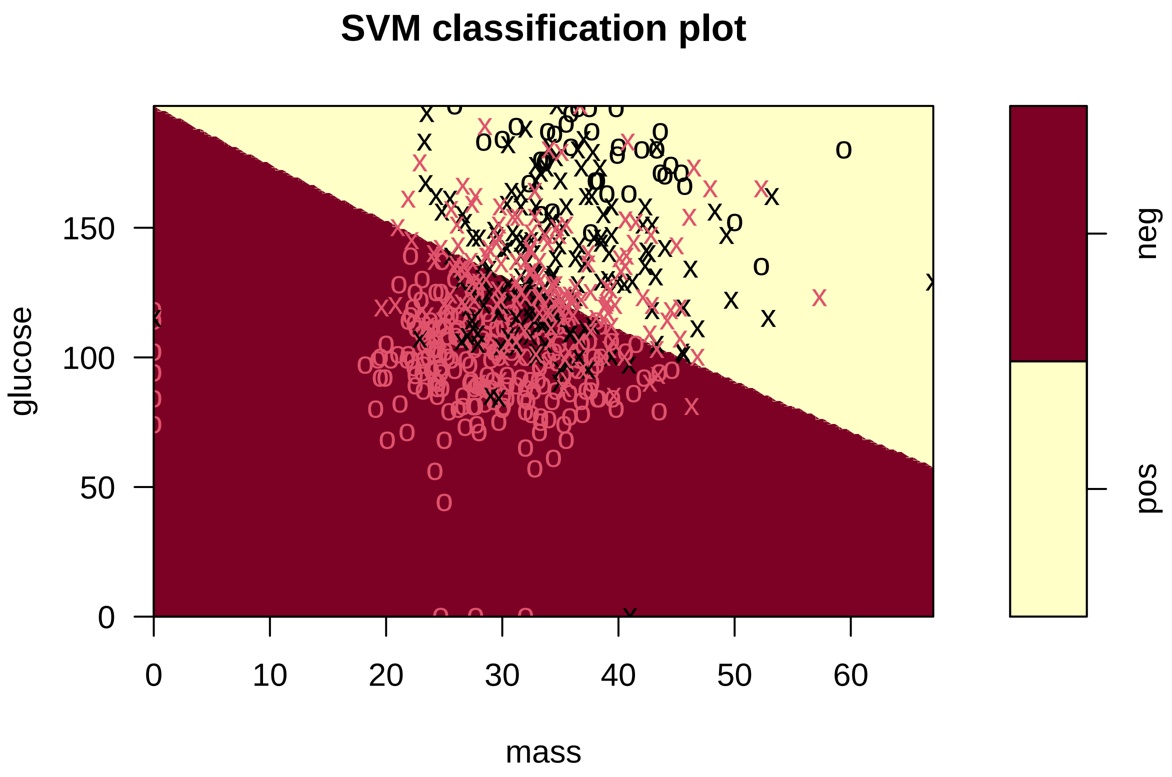
```r
pred.radial <- predict(best.radial, newdata = dat[-rowTrain,])

confusionMatrix(data = pred.radial,
                reference = dat$diabetes[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction pos neg
##        pos  39  16
##        neg  28 109
##
##                Accuracy : 0.7708
##                  95% CI : (0.7048, 0.8283)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 0.0002216
##
##                   Kappa : 0.4738
##
##  Mcnemar's Test P-Value : 0.0972544
##
##             Sensitivity : 0.5821
##             Specificity : 0.8720
##          Pos Pred Value : 0.7091
##          Neg Pred Value : 0.7956
##              Prevalence : 0.3490
##          Detection Rate : 0.2031
##    Detection Prevalence : 0.2865
##       Balanced Accuracy : 0.7270
##
##        'Positive' Class : pos
##
```

```
#visualize decision boundary
plot(best.radial, dat[rowTrain,],
     glucose ~ mass,
     slice = list(pregnant = 5, triceps = 20,
                  insulin = 20, pressure = 75,
                  pedigree = 1, age = 50),
     grid = 100,
     symbolPalette = c("cyan","darkblue"), #class labels
     color.palette = heat.colors) #background colors: rainbow, etc
```



```r
plot(best.radial, dat[rowTrain,],
     glucose ~ mass,
     slice = list(pregnant = 5, triceps = 20,
                  insulin = 20, pressure = 75,
                  pedigree = 1, age = 50),
     grid = 200)
```



Using `kernlab` - another commonly used library in R for SVM

Check https://cran.r-project.org/web/packages/kernlab/vignettes/kernlab.pdf for more details. This has only one tuning parameter to show the syntax, not recommended for model training.

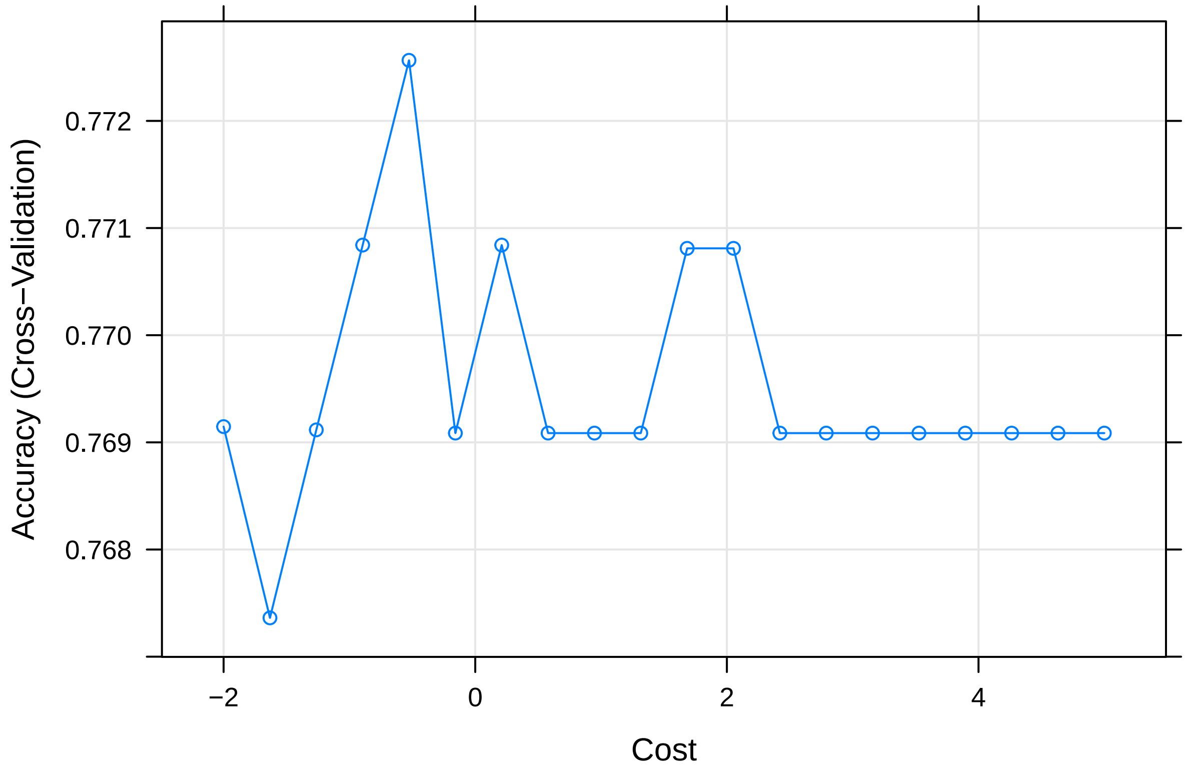
```
x_train <- as.matrix(dat[rowTrain, 1:8])
x_test <- as.matrix(dat[-rowTrain, 1:8])

linear <- ksvm(x = x_train,
               y = dat$diabetes[rowTrain],
               type = "C-svc",
               kernel = "vanilladot",
               C = 1,
               scaled = TRUE)
```

```
##  Setting default kernel parameters
```

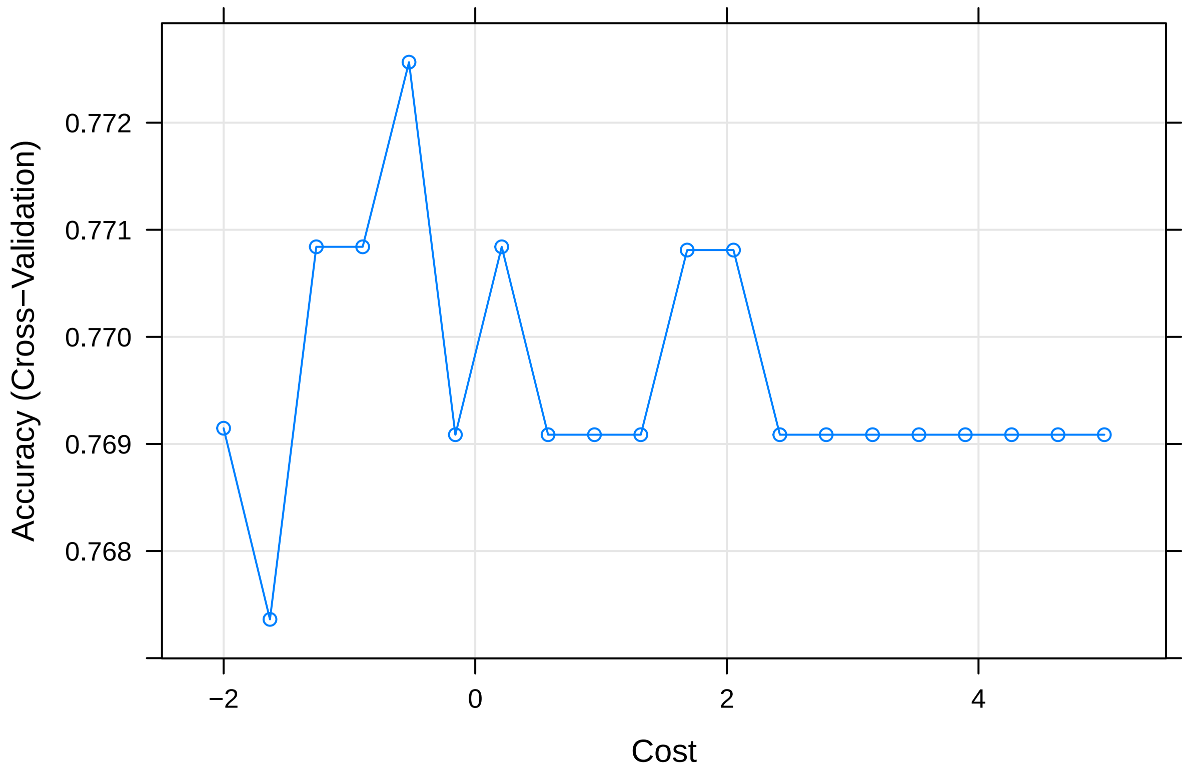
```
pred.linear2 <- predict(linear, newdata = x_test)


# "?dots" for definition of kernel functions

set.seed(1)
rbf <- ksvm(x = x_train,
            y = dat$diabetes[rowTrain],
            type = "C-svc",
            kernel = "rbfdot" ,
            kpar = "automatic",
            C = 1)

pred.rbf <- predict(rbf, newdata = x_test)
```

Using caret

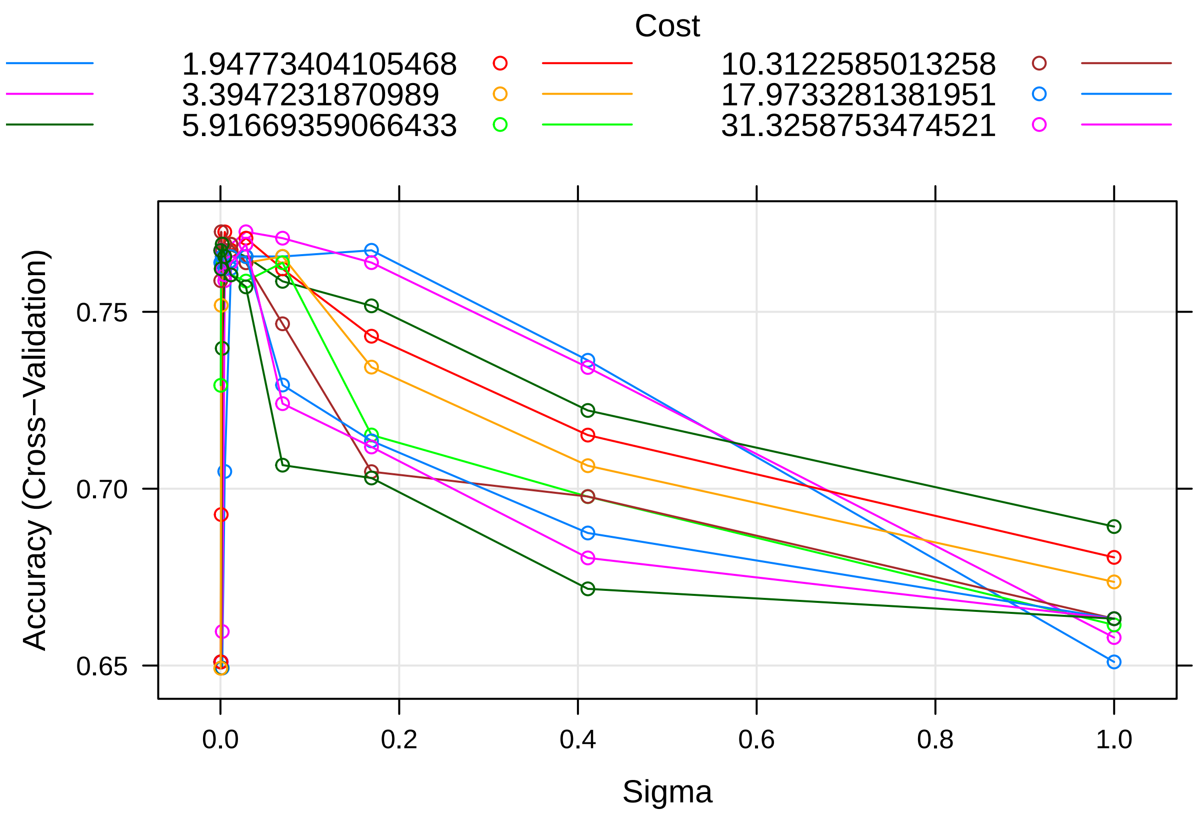
```r
ctrl <- trainControl(method = "cv")

# kernlab
set.seed(1)
svml.fit <- train(diabetes ~ . ,
                  data = dat[rowTrain,],
                  method = "svmLinear",
                  # preProcess = c("center", "scale"),
                  tuneGrid = data.frame(C = exp(seq(-2,5,len=20))),
                  trControl = ctrl)

plot(svml.fit, highlight = TRUE, xTrans = log)
```



```r
# e1071
set.seed(1)
svml.fit2 <- train(diabetes ~ . ,
                  data = dat[rowTrain,],
                  method = "svmLinear2",
                  tuneGrid = data.frame(cost = exp(seq(-2,5,len=20))),
                  trControl = ctrl)

plot(svml.fit2, highlight = TRUE, xTrans = log)
```

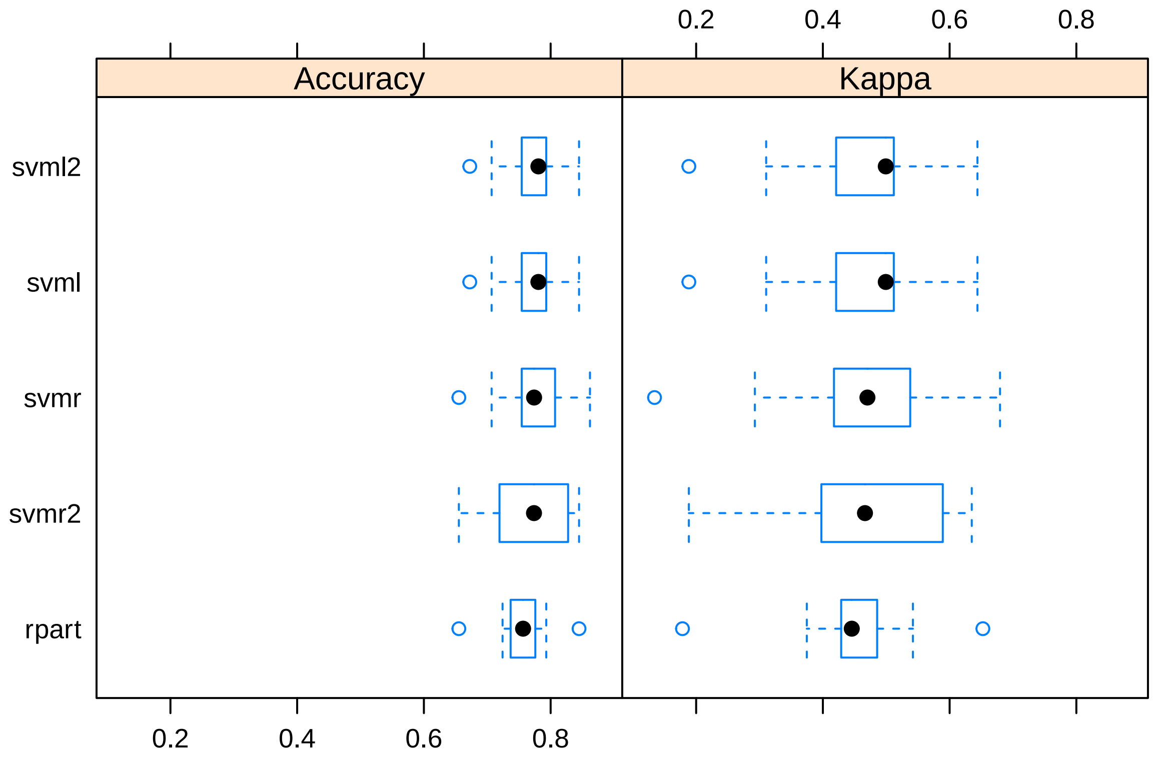


```
svmr.grid <- expand.grid(C = exp(seq(-1,4,len=10)),
                         sigma = exp(seq(-8,0,len=10)))

# tunes over both cost and sigma
set.seed(1)
svmr.fit <- train(diabetes ~ . , dat,
                  subset = rowTrain,
                  method = "svmRadialSigma",
                  preProcess = c("center", "scale"),
                  tuneGrid = svmr.grid,
                  trControl = ctrl)

plot(svmr.fit, highlight = TRUE)
```



```r
# tune over cost and uses a single value of sigma based on kernlab's sigest function
set.seed(1)
svmr.fit2 <- train(diabetes ~ . , dat,
                   subset = rowTrain,
                   method = "svmRadialCost",
                   preProcess = c("center", "scale"),
                   tuneGrid = data.frame(C = exp(seq(-4,2,len=10))),
                   trControl = ctrl)

# Platt's probabilistic outputs; use with caution
set.seed(1)
svmr.fit3 <- train(diabetes ~ . , dat,
                   subset = rowTrain,
                   method = "svmRadialCost",
                   preProcess = c("center", "scale"),
                   tuneGrid = data.frame(C = exp(seq(-4,2,len=10))),
                   trControl = ctrl,
                   prob.model = TRUE)
# predict(svmr.fit3, newdata = x_test, type = "prob")

set.seed(1)
rpart.fit <- train(diabetes ~ . , dat,
                   subset = rowTrain,
                   method = "rpart",
                   tuneLength = 50,
                   trControl = ctrl)
```

```r
resamp <- resamples(list(svmr = svmr.fit, svmr2 = svmr.fit2,
                         svml = svml.fit, svml2 = svml.fit2,
                             rpart = rpart.fit))
bwplot(resamp)
```



We finally look at the test data performance.

```
pred.svml <- predict(svml.fit, newdata = dat[-rowTrain,])
pred.svmr <- predict(svmr.fit, newdata = dat[-rowTrain,])

confusionMatrix(data = pred.svml,
                reference = dat$diabetes[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction pos neg
##        pos  40  17
##        neg  27 108
##
##                Accuracy : 0.7708
##                  95% CI : (0.7048, 0.8283)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 0.0002216
##
##                   Kappa : 0.4776
##
##  Mcnemar's Test P-Value : 0.1748444
##
##             Sensitivity : 0.5970
##             Specificity : 0.8640
##          Pos Pred Value : 0.7018
##          Neg Pred Value : 0.8000
##              Prevalence : 0.3490
##          Detection Rate : 0.2083
##    Detection Prevalence : 0.2969
##       Balanced Accuracy : 0.7305
##
##        'Positive' Class : pos
##
```

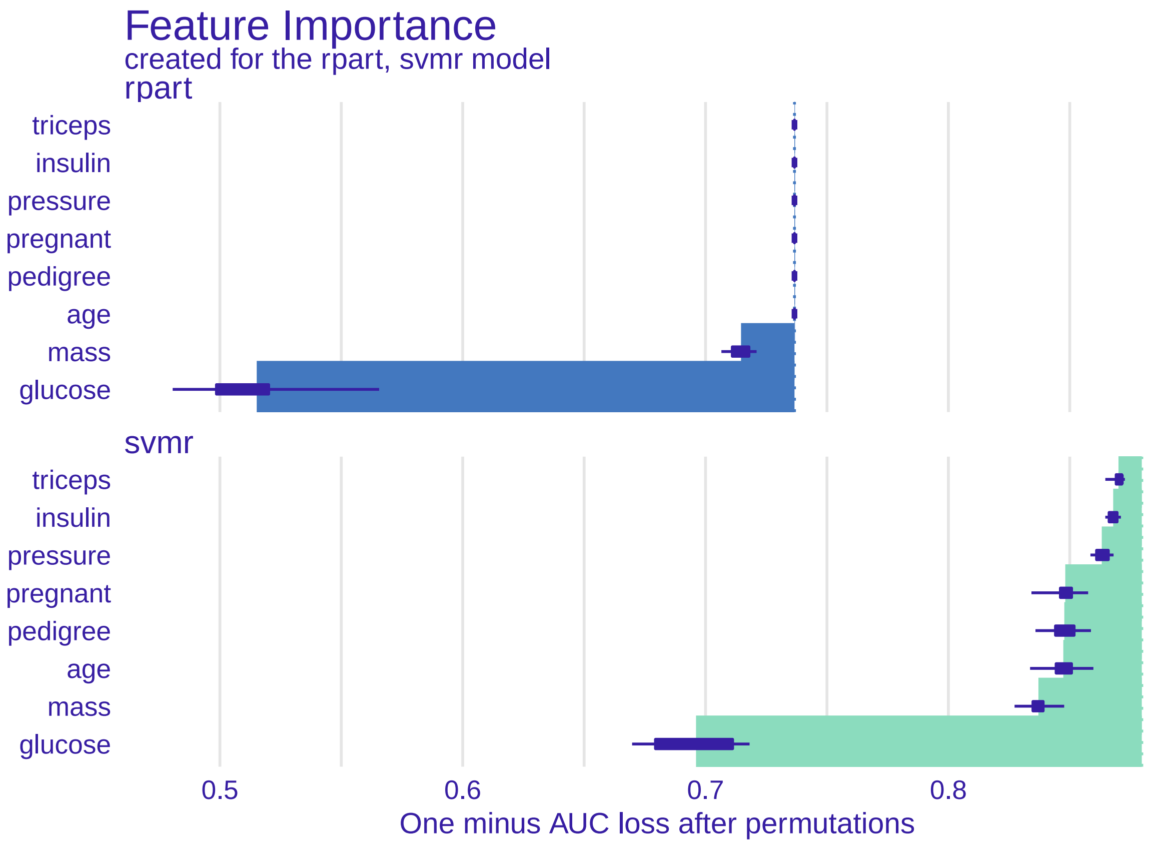
```
confusionMatrix(data = pred.svmr,
                reference = dat$diabetes[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction pos neg
##        pos  41  17
##        neg  26 108
##
##                Accuracy : 0.776
##                  95% CI : (0.7104, 0.8329)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 0.0001183
##
##                   Kappa : 0.4912
##
##  Mcnemar's Test P-Value : 0.2224692
##
##             Sensitivity : 0.6119
##             Specificity : 0.8640
##          Pos Pred Value : 0.7069
##          Neg Pred Value : 0.8060
##              Prevalence : 0.3490
##          Detection Rate : 0.2135
##    Detection Prevalence : 0.3021
##       Balanced Accuracy : 0.7380
##
##        'Positive' Class : pos
##
```

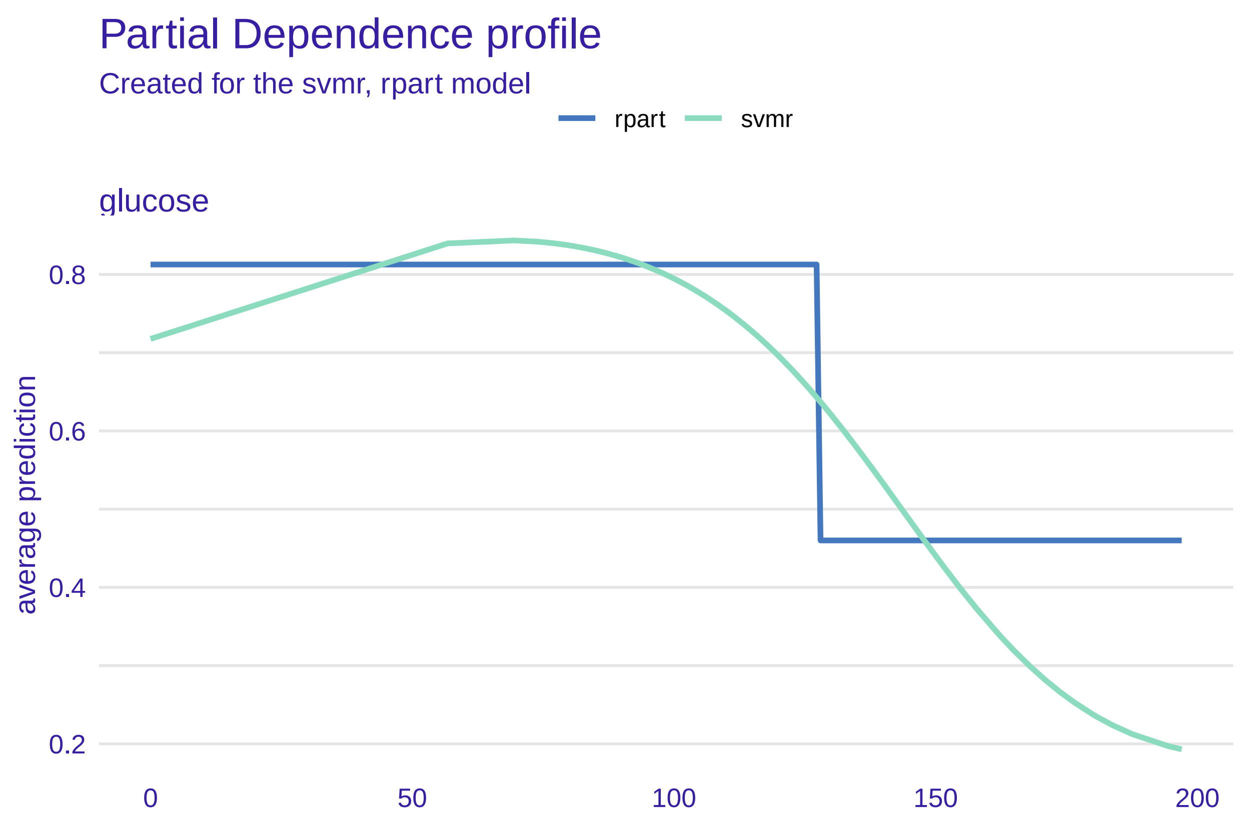
Understanding your models with DALEX

```
explainer_rpart <- explain(rpart.fit,
                           label = "rpart",
                           data = x_train,
                           y = as.numeric(dat$diabetes[rowTrain] == "pos"),
                           verbose = FALSE)

# SVM does not output probabilities, this is using Platt's output and is just for illustration of DALEX
explainer_svm <- explain(svmr.fit3,
                         label = "svmr",
                         data = x_train,
                         y = as.numeric(dat$diabetes[rowTrain] == "pos"),
                         verbose = FALSE)
```

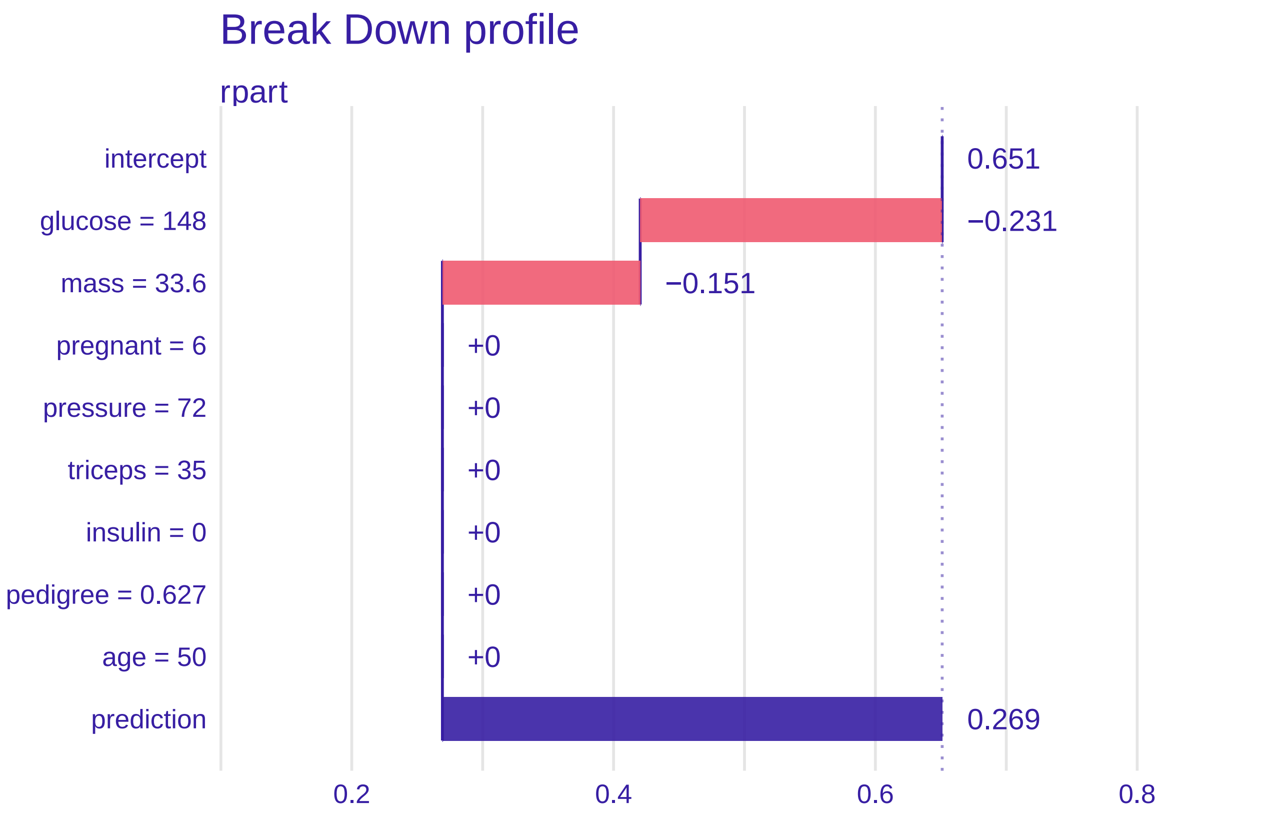
```
# variable importance
vi_rpart <- model_parts(explainer_rpart)
vi_svm <- model_parts(explainer_svm)

plot(vi_rpart, vi_svm)
```



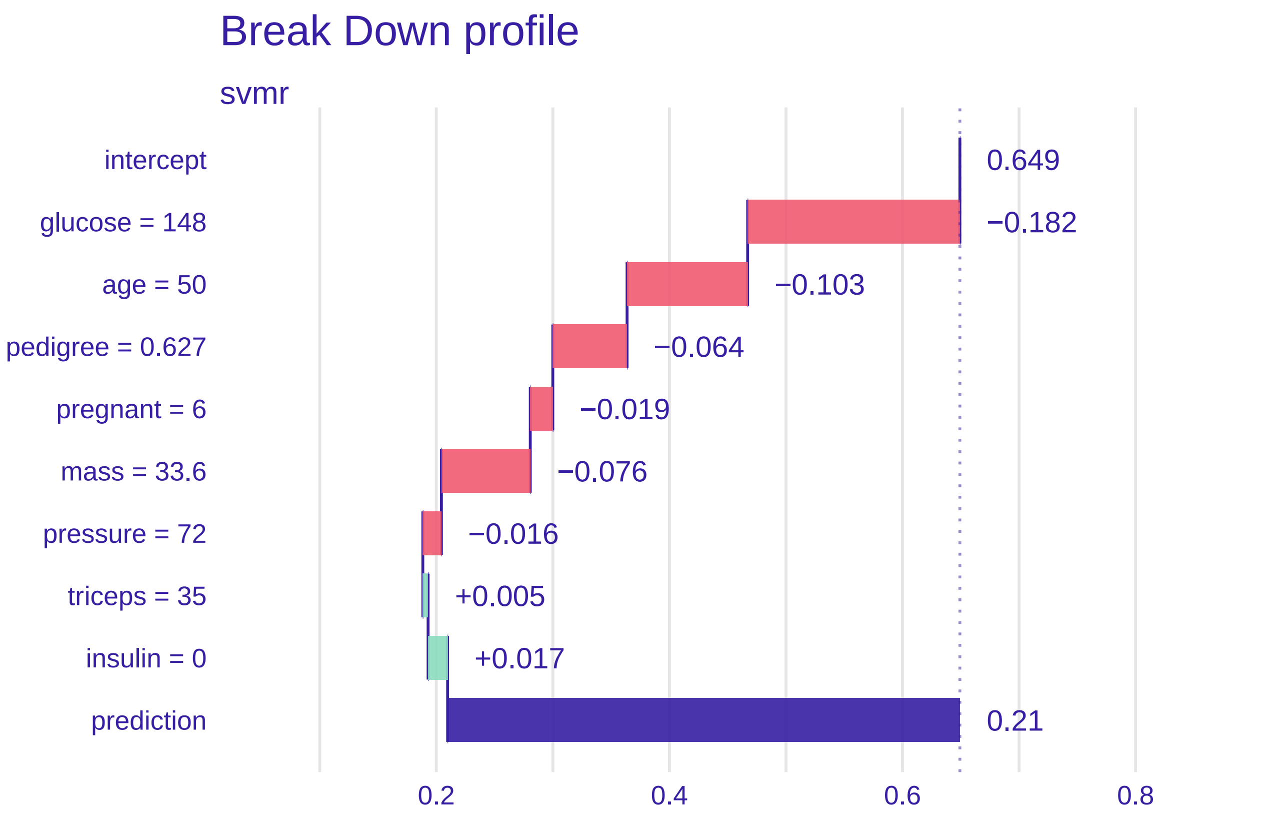
```
# PDP
pdp_svm <- model_profile(explainer_svm,
                         variable = "glucose",
                         type = "partial")
pdp_rpart <- model_profile(explainer_rpart,
                           variable = "glucose",
                           type = "partial")
plot(pdp_svm, pdp_rpart)
```



```r
# bread down
pb_svm <- predict_parts(explainer_svm,
                        new_observation = dat[1,],
                        type = "break_down")
pb_rpart <- predict_parts(explainer_rpart,
                          new_observation = dat[1,],
                          type = "break_down")
plot(pb_rpart)
```


Break Down profile
rpart
intercept
glucose = 148
mass = 33.6
pregnant = 6
pressure = 72
triceps = 35
insulin = 0
pedigree = 0.627
age = 50
prediction
0.651
−0.231
−0.151
+0
+0
+0
+0
+0
+0
0.269
0.2
0.4
0.6
0.8

```r
plot(pb_svm)
```


Break Down profile
svmr
intercept
glucose = 148
age = 50
pedigree = 0.627
pregnant = 6
mass = 33.6
pressure = 72
triceps = 35
insulin = 0
prediction
0.649
−0.182
−0.103
−0.064
−0.019
−0.076
−0.016
+0.005
+0.017
0.21
0.2
0.4
0.6
0.8

Regression

Predict a baseball player's salary on the basis of various statistics associated with performance in the previous year. Use `?Hitters` for more details.

```r
data(Hitters)
Hitters <- na.omit(Hitters)

set.seed(2021)
trRows <- createDataPartition(Hitters$Salary,
                              p = .75,
                              list = F)
```

eps-regression

```r
set.seed(1)
svml.fit <- tune.svm(Salary ~ . ,
                     data = Hitters[trRows,],
                     kernel = "linear",
                     epsilon = exp(seq(-5,0,len=20)))

set.seed(1)
svmr.fit <- tune.svm(Salary ~ . ,
                     data = Hitters[trRows,],
                     kernel = "radial",
                     epsilon = exp(seq(-5,0,len=10)),
                     gamma = exp(seq(-6,-2,len=10)))

svmr.pred <- predict(svmr.fit$best.model, newdata = Hitters[-trRows,])
RMSE(svmr.pred, Hitters$Salary[-trRows])
```

```
## [1] 292.2582
```