Classification I

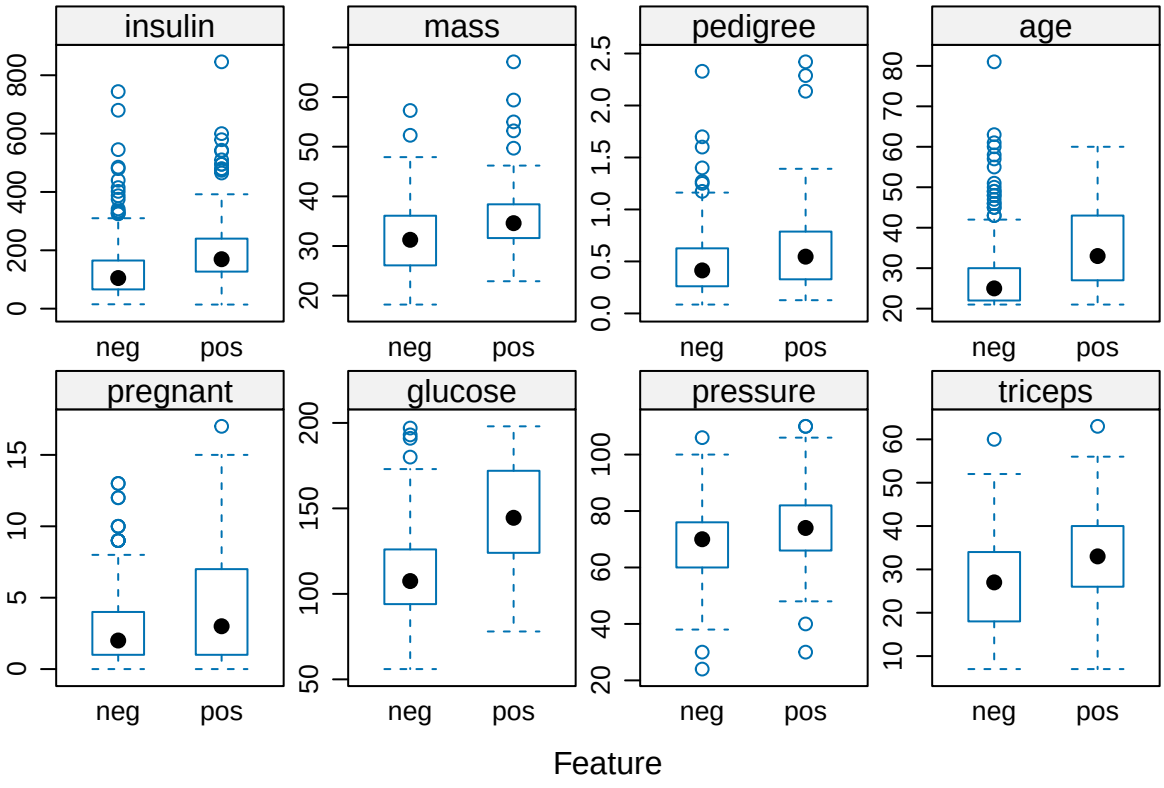
Yifei Sun, Runze Cui

Contents

```
library(caret)
library(glmnet)
library(tidymodels)
library(mlbench)
library(pROC)
library(pdp)
library(vip)
library(AppliedPredictiveModeling)
```

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

```
data(PimaIndiansDiabetes2)
dat <- na.omit(PimaIndiansDiabetes2)

featurePlot(x = dat[, 1:8],
            y = dat$diabetes,
            scales = list(x = list(relation = "free"),
                          y = list(relation = "free")),
            plot = "box")
```



The data is divided into two parts (training and test).

```
set.seed(1)
data_split <- initial_split(dat, prop = 0.75)

# Extract the training and test data
training_data <- training(data_split)
testing_data <- testing(data_split)
```

Logistic regression and its cousins

glm

```
contrasts(dat$diabetes)
```

```
##     pos
## neg   0
## pos   1
```

```
glm.fit <- glm(diabetes ~ .,
               data = training_data,
               family = binomial(link = "logit"))
```

We first consider the simple classifier with a cut-off of 0.5 and evaluate its performance on the test data.

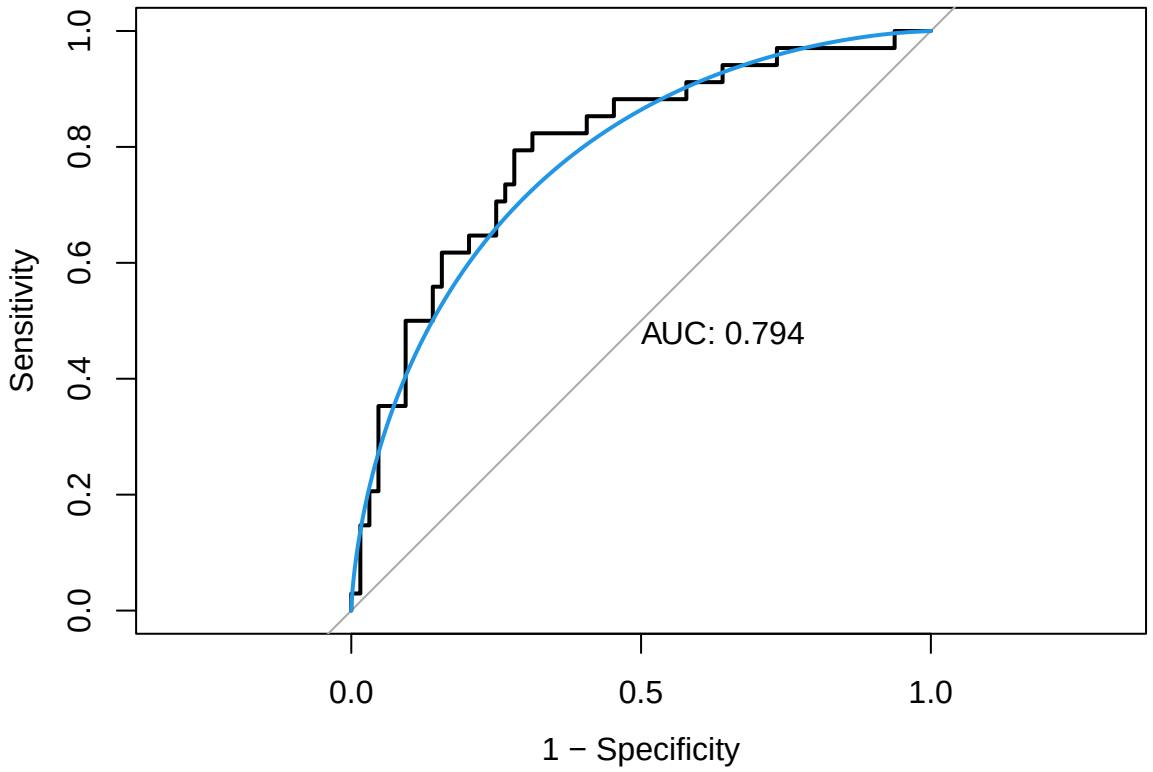
```
test.pred.prob <- predict(glm.fit, newdata = testing_data,
                          type = "response")
test.pred <- rep("neg", length(test.pred.prob))
test.pred[test.pred.prob > 0.5] <- "pos"

confusionMatrix(data = as.factor(test.pred),
                reference = testing_data$diabetes,
                positive = "pos")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  55  16
##        pos   9  18
##
##                Accuracy : 0.7449
##                  95% CI : (0.6469, 0.8276)
##     No Information Rate : 0.6531
##     P-Value [Acc > NIR] : 0.03334
##
##                   Kappa : 0.4085
##
##  Mcnemar's Test P-Value : 0.23014
##
##             Sensitivity : 0.5294
##             Specificity : 0.8594
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.7746
##              Prevalence : 0.3469
##          Detection Rate : 0.1837
##    Detection Prevalence : 0.2755
##       Balanced Accuracy : 0.6944
##
##        'Positive' Class : pos
##
```

We then plot the test ROC curve. You may also consider a smoothed ROC curve.

```
roc.glm <- roc(testing_data$diabetes, test.pred.prob)
plot(roc.glm, legacy.axes = TRUE, print.auc = TRUE)
```

```
plot(smooth(roc.glm), col = 4, add = TRUE)
```



We can also fit a logistic regression using caret. This is to compare the cross-validation performance with other models, rather than tuning the model.

```
# Using caret
ctrl <- trainControl(method = "cv", number = 10,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)
set.seed(1)
model.glm <- train(x = training_data[1:8],
                   y = training_data$diabetes,
                   method = "glm",
                   metric = "ROC",
                   trControl = ctrl)
```

Penalized logistic regression

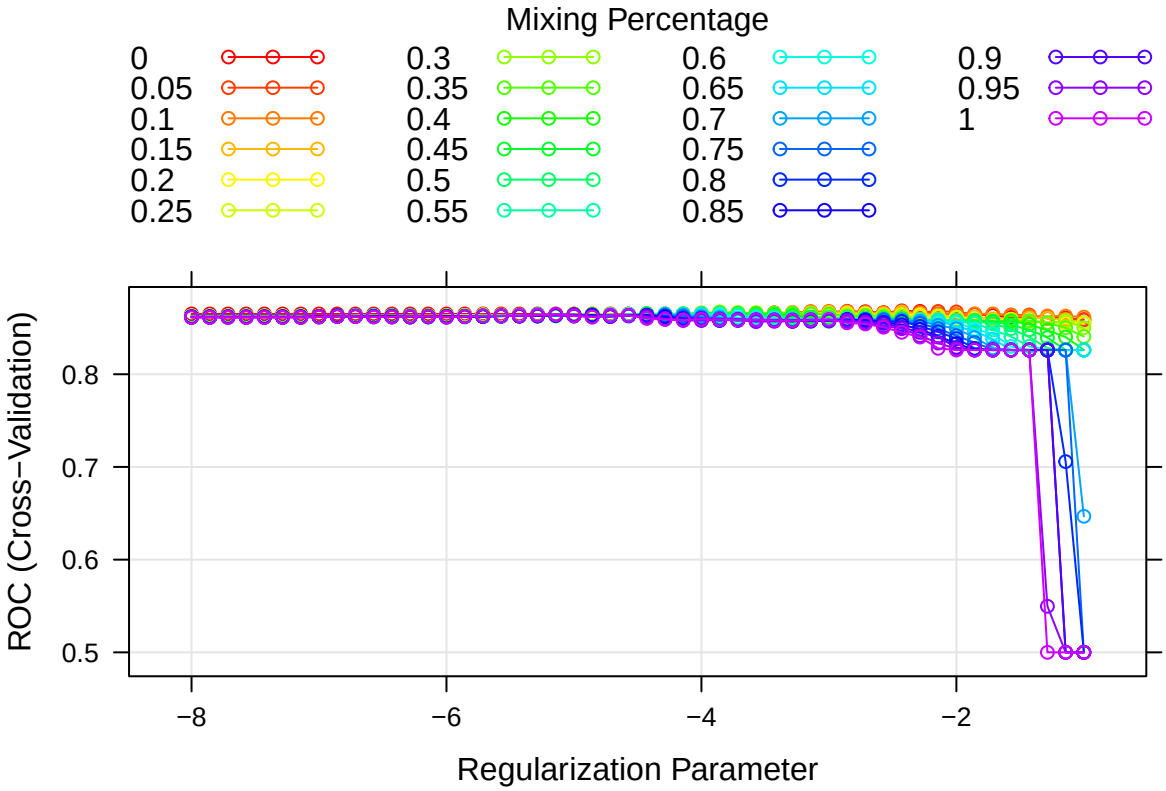
Penalized logistic regression can be fitted using `glmnet`. We use the `train` function to select the optimal tuning parameters.

```
glmnGrid <- expand.grid(.alpha = seq(0, 1, length = 21),
                        .lambda = exp(seq(-8, -1, length = 50)))
set.seed(1)
model.glmn <- train(x = training_data[1:8],
                    y = training_data$diabetes,
                    method = "glmnet",
                    tuneGrid = glmnGrid,
                    metric = "ROC",
                    trControl = ctrl)
```

```
model.glmn$bestTune
```

```
##    alpha     lambda
## 40     0 0.08816269
```

```
myCol <- rainbow(25)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))

plot(model.glmn, par.settings = myPar, xTrans = function(x) log(x))
```



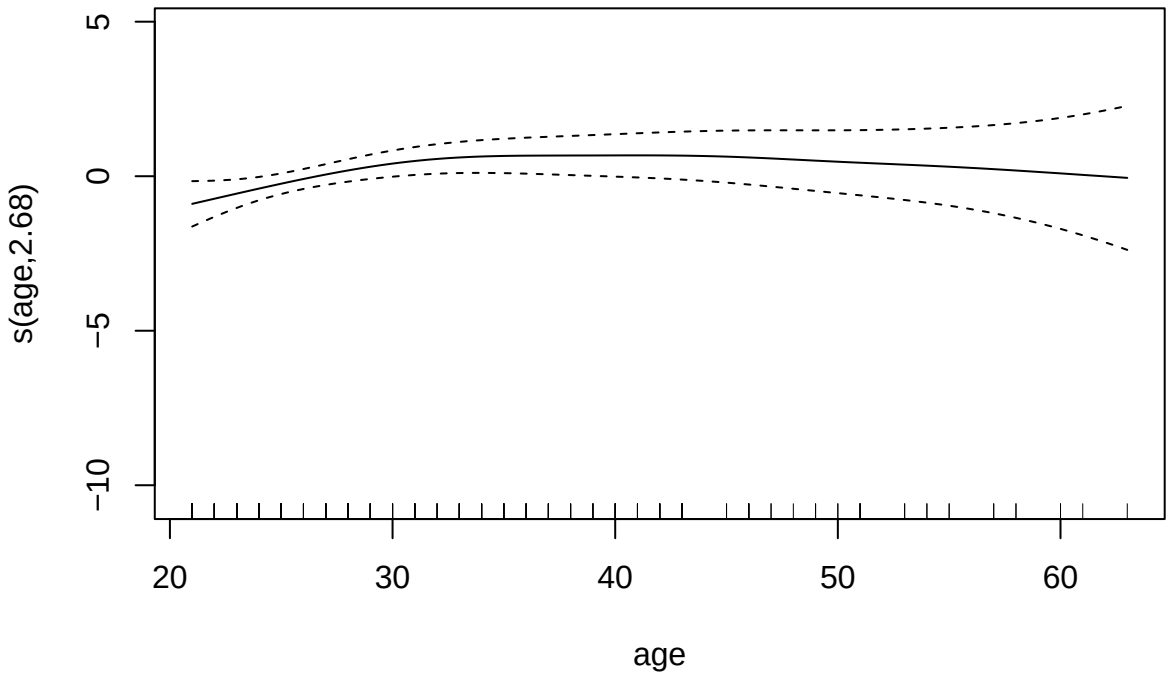
GAM

```
set.seed(1)
model.gam <- train(x = training_data[1:8],
                   y = training_data$diabetes,
                   method = "gam",
                   metric = "ROC",
                   trControl = ctrl)


model.gam$finalModel
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
```

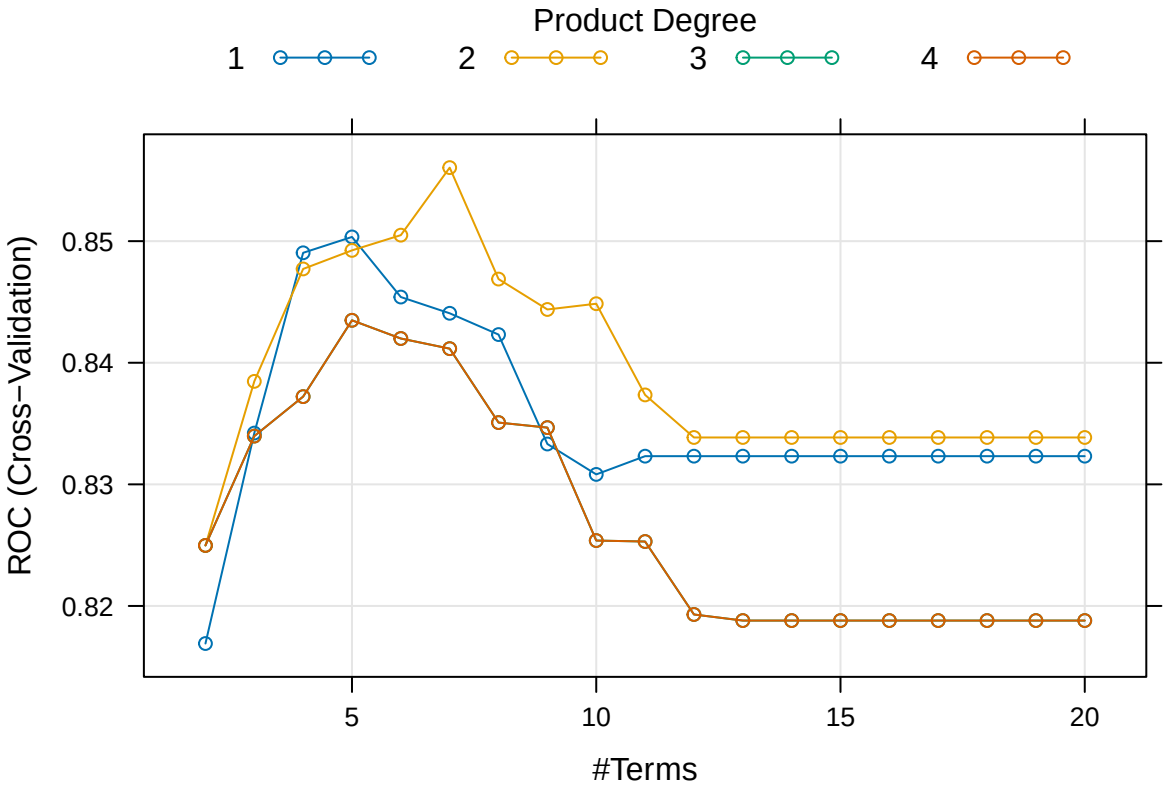
```
## .outcome ~ s(pregnant) + s(pressure) + s(age) + s(triceps) +
##     s(glucose) + s(insulin) + s(mass) + s(pedigree)
##
## Estimated degrees of freedom:
## 1.04 1.00 2.68 2.38 1.00 4.34 1.44
## 1.00  total = 15.89
##
## UBRE score: -0.1596863
```

```
plot(model.gam$finalModel, select = 3)
```



MARS

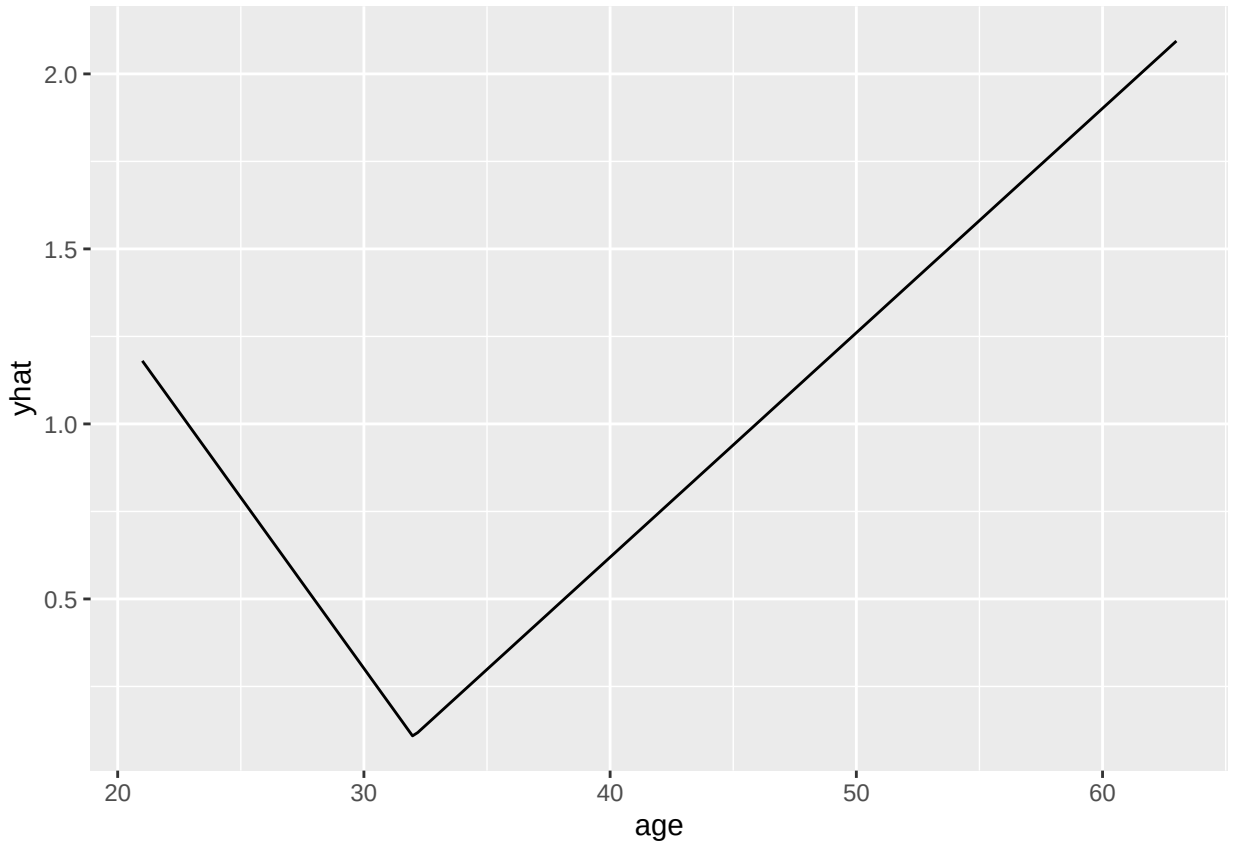
```
set.seed(1)
model.mars <- train(x = training_data[1:8],
                    y = training_data$diabetes,
                    method = "earth",
                    tuneGrid = expand.grid(degree = 1:4,
                                           nprune = 2:20),
                    metric = "ROC",
                    trControl = ctrl)

plot(model.mars)
```



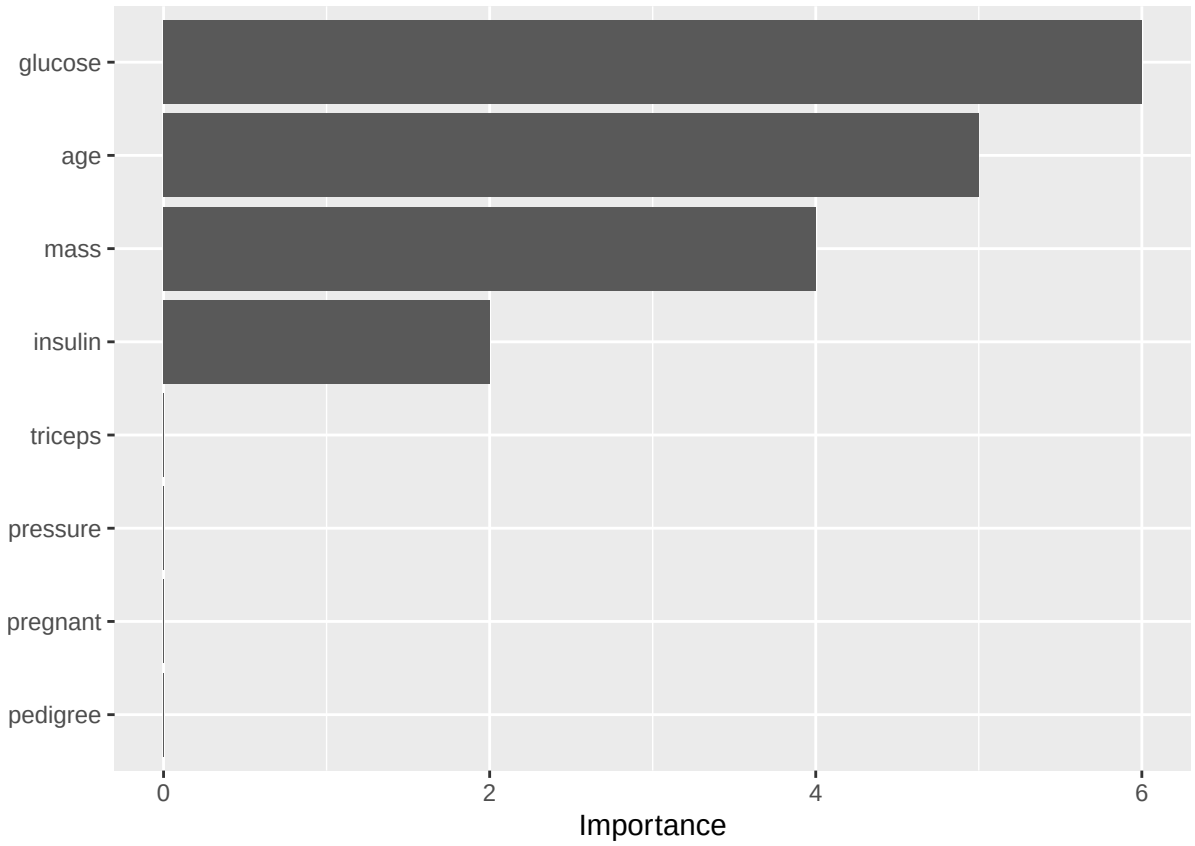

```
coef(model.mars$finalModel)
```

```
##                 (Intercept)             h(187-glucose)
##               3.7745918129              -0.0491408053
##                  h(32-age)               h(35.7-mass)
##              -0.2453779688              -0.1892759329
## h(146-glucose) * h(32-age) h(130-insulin) * h(age-32)
##               0.0017986652              -0.0051382452
##   h(glucose-187) * insulin
##              -0.0006838935
```

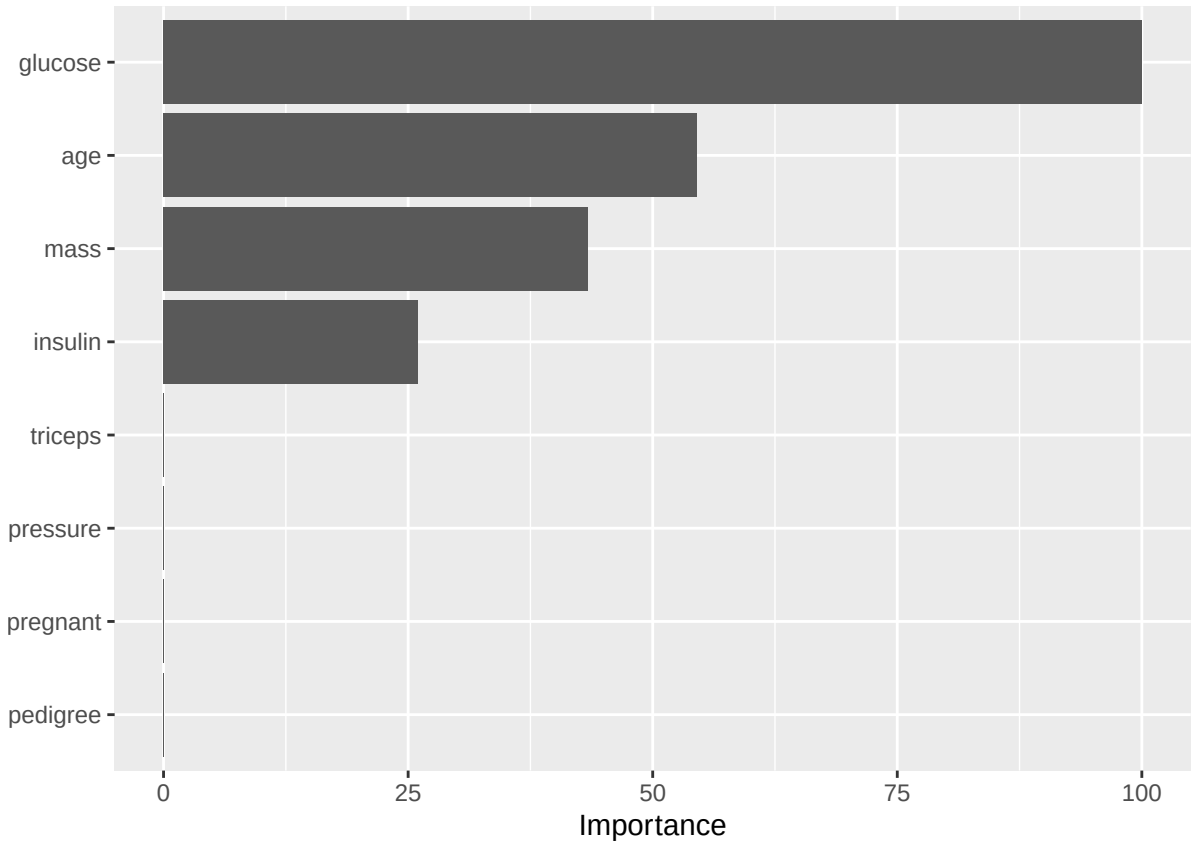
```
pdp::partial(model.mars, pred.var = c("age"), grid.resolution = 200) %>% autoplot()
```




```
vip(model.mars$finalModel, type = "nsubsets")
```




```
vip(model.mars$finalModel, type = "rss")
```



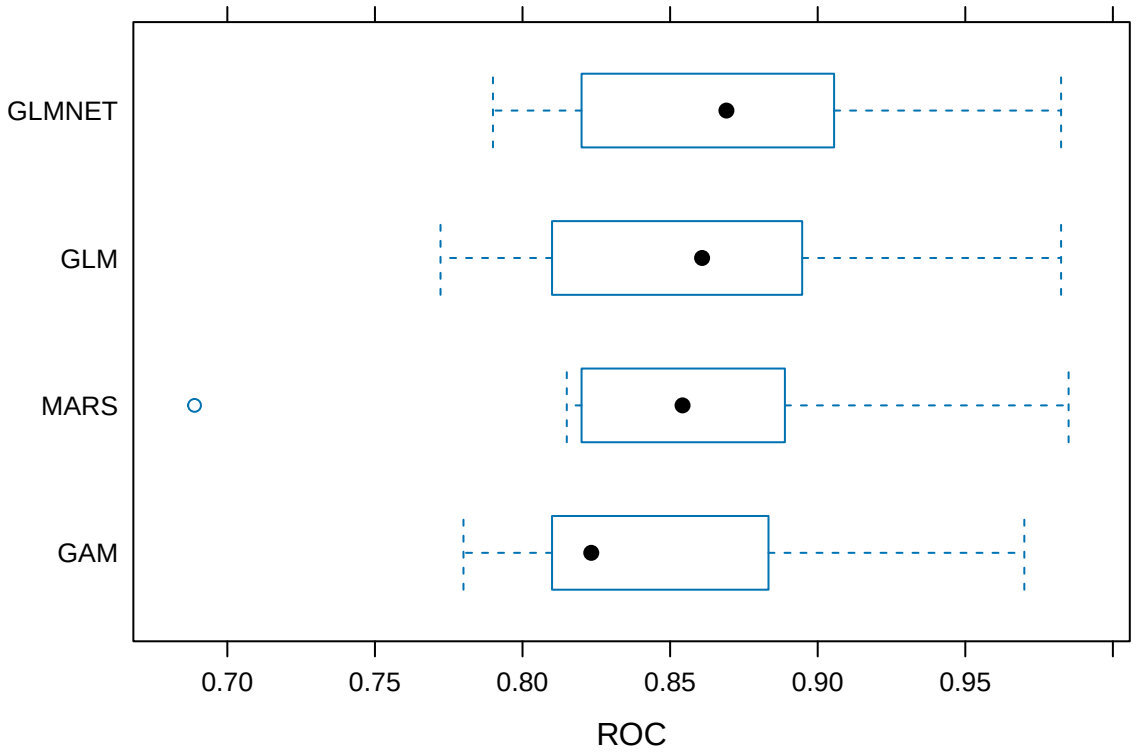

```
# see vi_model.earth for details on different types of variable importance of MARS
```

```
res <- resamples(list(GLM = model.glm,
                      GLMNET = model.glmn,
                      GAM = model.gam,
                      MARS = model.mars))
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: GLM, GLMNET, GAM, MARS
## Number of resamples: 10
##
## ROC
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.7722222 0.8187500 0.8608333 0.8628860 0.8904971 0.9824561    0
## GLMNET 0.7900000 0.8250000 0.8690789 0.8683392 0.8986111 0.9824561    0
## GAM    0.7800000 0.8102778 0.8232895 0.8522865 0.8819444 0.9700000    0
## MARS   0.6888889 0.8219444 0.8542105 0.8560497 0.8879167 0.9850000    0
##
## Sens
##             Min. 1st Qu.    Median      Mean   3rd Qu. Max. NA's
## GLM    0.8000000  0.8500 0.8750000 0.8892105 0.9000000    1    0
## GLMNET 0.8421053  0.8625 0.9000000 0.9092105 0.9375000    1    0
## GAM    0.6842105  0.8000 0.8250000 0.8428947 0.8986842    1    0
```

```
## MARS   0.7500000  0.8500 0.8973684 0.8734211 0.9000000    1    0
##
## Spec
##             Min. 1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.4444444   0.525 0.6500000 0.6355556 0.7000000 0.8888889    0
## GLMNET 0.3333333   0.500 0.6000000 0.5600000 0.6666667 0.7000000    0
## GAM    0.3333333   0.500 0.5777778 0.5944444 0.7000000 0.7777778    0
## MARS   0.3333333   0.600 0.6333333 0.6233333 0.7000000 0.7777778    0
```

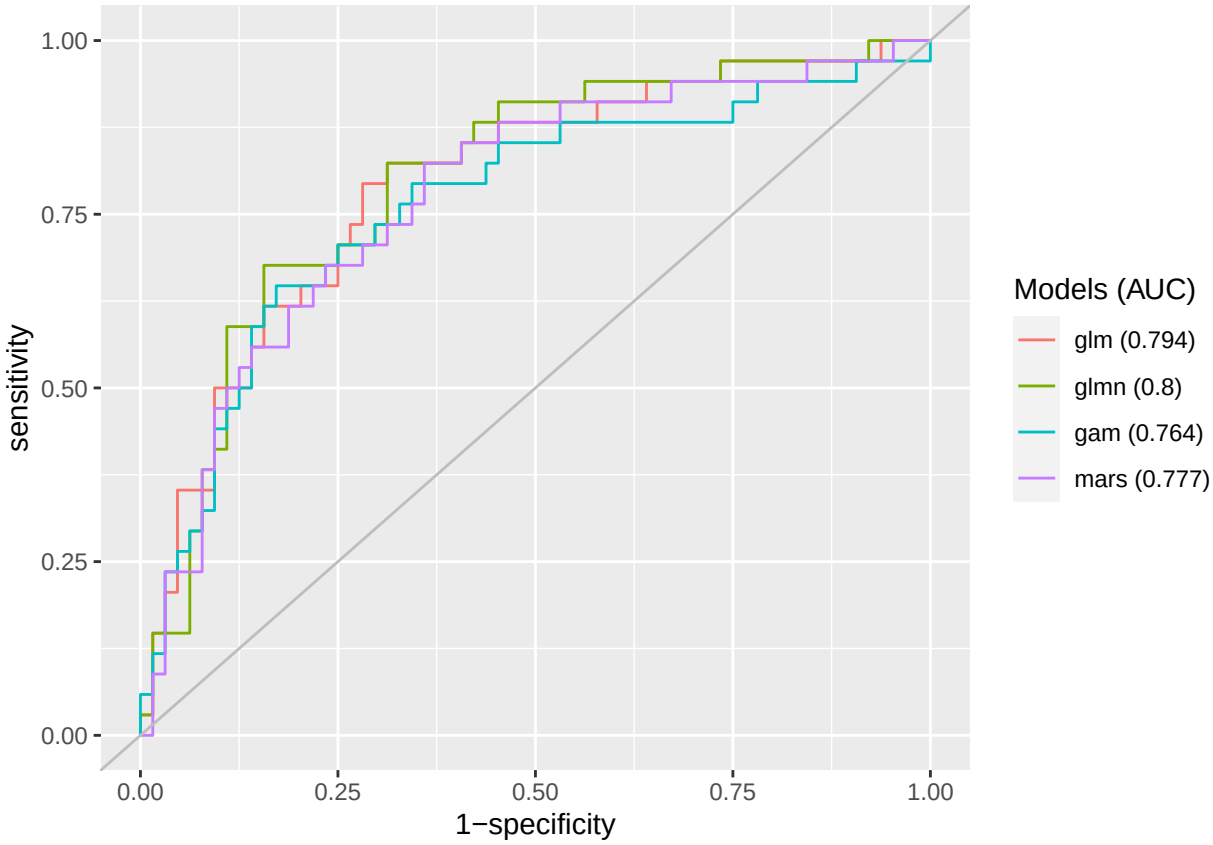
```
bwplot(res, metric = "ROC")
```



Now let's look at the test data performance.

```
glm.pred <- predict(model.glm, newdata = testing_data, type = "prob")[,2]
glmn.pred <- predict(model.glmn, newdata = testing_data, type = "prob")[,2]
gam.pred <- predict(model.gam, newdata = testing_data, type = "prob")[,2]
mars.pred <- predict(model.mars, newdata = testing_data, type = "prob")[,2]

roc.glm <- roc(testing_data$diabetes, glm.pred)
roc.glmn <- roc(testing_data$diabetes, glmn.pred)
roc.gam <- roc(testing_data$diabetes, gam.pred)
roc.mars <- roc(testing_data$diabetes, mars.pred)

auc <- c(roc.glm$auc[1], roc.glmn$auc[1],
         roc.gam$auc[1], roc.mars$auc[1])

modelNames <- c("glm","glmn","gam","mars")

ggroc(list(roc.glm, roc.glmn, roc.gam, roc.mars), legacy.axes = TRUE) +
  scale_color_discrete(labels = paste0(modelNames, " (", round(auc,3),")"),
                       name = "Models (AUC)") +
```

```
  geom_abline(intercept = 0, slope = 1, color = "grey")
```




```
## using plot.roc
# plot(roc.glm, legacy.axes = TRUE)
# plot(roc.glmn, col = 2, add = TRUE)
# plot(roc.gam, col = 3, add = TRUE)
# plot(roc.mars, col = 4, add = TRUE)
#
# legend("bottomright", legend = paste0(modelNames, ": ", round(auc,3)),
#        col = 1:4, lwd = 2)
```

tidymodels

```
set.seed(1)
cv_folds <- vfold_cv(training_data, v = 10)


# # Model specification for Penalized Logistic Regression
# glmnet_spec <- logistic_reg(penalty = tune(), mixture = tune()) %>%
#   set_engine("glmnet") %>%
#   set_mode("classification")
#
# # Define the workflow
# glmnet_grid_set <- parameters(penalty(range = c(-8, -1), trans = log_trans()),
#                               mixture(range = c(0, 1)))
```

```
# glmnet_grid <- grid_regular(glmnet_grid_set, levels = c(50, 21))
#
# # Set up the workflow
# glmnet_workflow <- workflow() %>%
#   add_model(glmnet_spec) %>%
#   add_formula(diabetes ~ .)
#
# # Tune the model
# glmnet_tune <- tune_grid(
#   glmnet_workflow,
#   resamples = cv_folds,
#   grid = glmnet_grid
# )
#
# # CV plot
# autoplot(glmnet_tune, metric = "roc_auc")
#
# glmnet_tune %>% show_best(metric = "roc_auc")
#
# glmnet_best <- select_best(glmnet_tune, metric = "roc_auc")
#
# final_glmnet_spec <- glmnet_spec %>%
#   update(penalty = glmnet_best$penalty,
#          mixture = glmnet_best$mixture)
#
# glmnet_fit <- fit(final_glmnet_spec, formula = diabetes ~ ., data = training_data)
#
# glmnet_model <- extract_fit_engine(glmnet_fit)
```

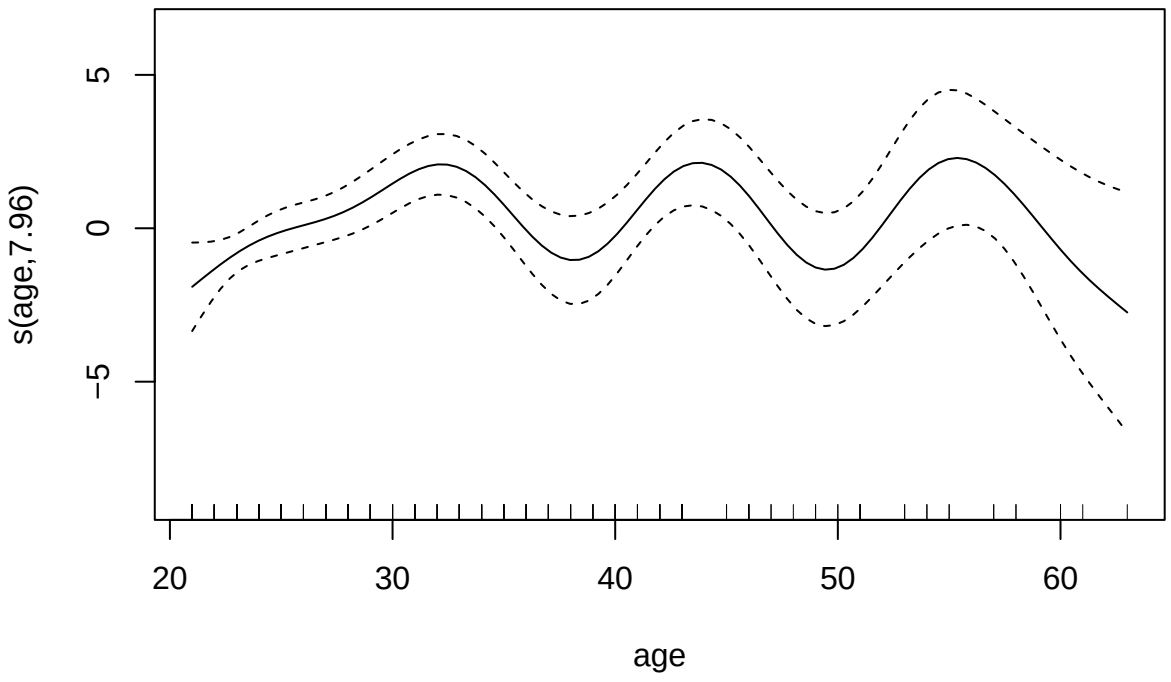
```
# Model specification for GAM
gam_spec <- gen_additive_mod(select_features = tune()) %>%
  set_engine("mgcv") %>%
  set_mode("classification")

# Set up the workflow
gam_workflow <-
  workflow() %>%
  add_model(gam_spec,
            formula = diabetes ~ s(pregnant) + s(glucose) + s(pressure) + s(triceps) +
                        s(insulin) + s(mass) + s(pedigree) + s(age)) %>%
  add_formula(diabetes ~ .)

gam_res <-
  gam_workflow %>% tune_grid(resamples = cv_folds)

show_best(gam_res, metric = "roc_auc")
```

```
## # A tibble: 2 x 7
##   select_features .metric .estimator  mean     n std_err .config
##   <lgl>           <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 TRUE            roc_auc binary     0.824    10  0.0303 Preprocessor1_Model2
## 2 FALSE           roc_auc binary     0.813    10  0.0347 Preprocessor1_Model1
```

```r
# Update the model specification
final_gam_spec <- gam_spec %>%
  update(select_features = TRUE)

gam_fit <- fit(final_gam_spec,
               formula = diabetes ~ s(pregnant) + s(glucose) + s(pressure) + s(triceps) +
                           s(insulin) + s(mass) + s(pedigree) + s(age),
               data = training_data)

gam_model <- extract_fit_engine(gam_fit)
gam_model
```

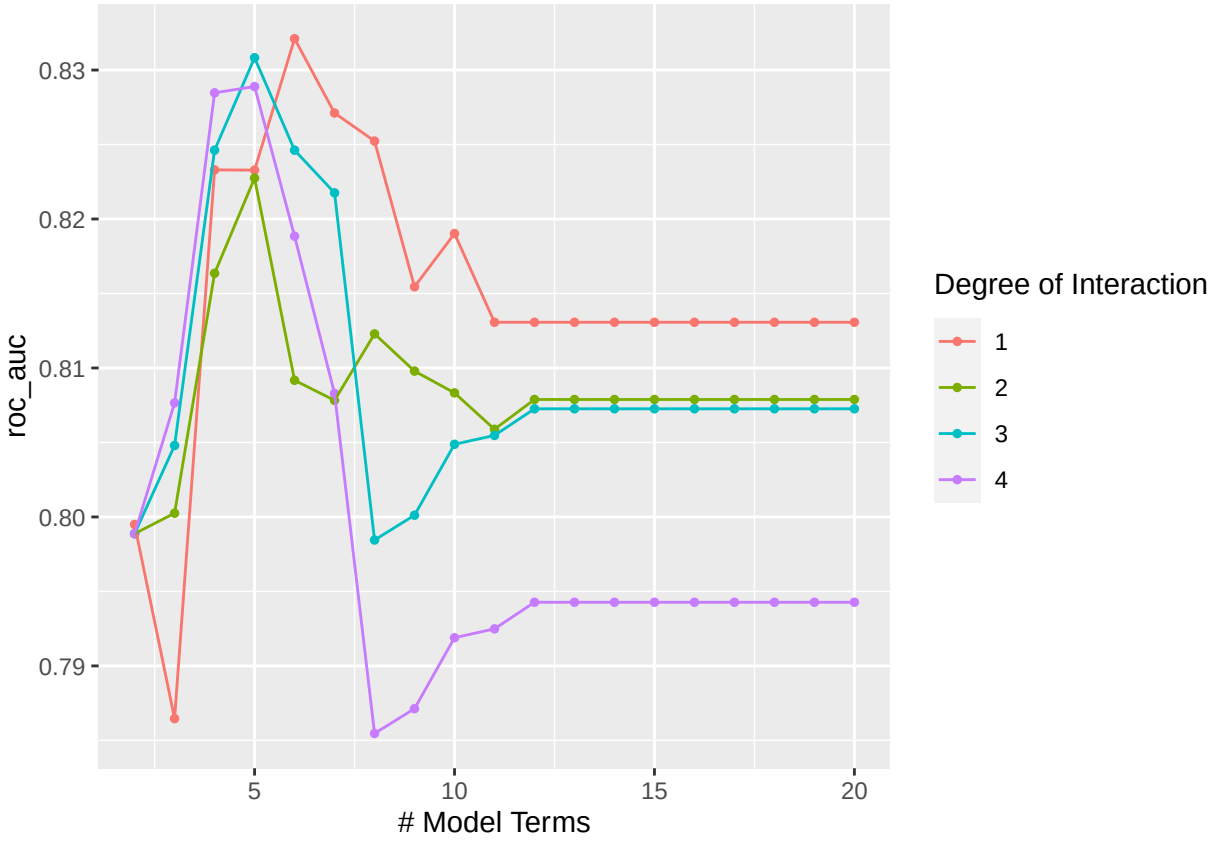
```
##
## Family: binomial
## Link function: logit
##
## Formula:
## diabetes ~ s(pregnant) + s(glucose) + s(pressure) + s(triceps) +
##     s(insulin) + s(mass) + s(pedigree) + s(age)
##
## Estimated degrees of freedom:
## 4.78 7.17 0.00 2.23 1.31 2.08 1.30
## 7.96  total = 27.83
##
## UBRE score: -0.2067444
```

```r
plot(gam_fit$fit, select = 8)
```




```r
# Model specification for MARS
mars_spec <- mars(num_terms = tune(),
                  prod_degree = tune()) %>%
  set_engine("earth") %>%
  set_mode("classification")
```

```r
# Grid of tuning parameters
mars_grid_set <- parameters(num_terms(range = c(2, 20)),
                            prod_degree(range = c(1, 4)))
mars_grid <- grid_regular(mars_grid_set, levels = c(19, 4))

# Set up the workflow
mars_workflow <- workflow() %>%
  add_model(mars_spec) %>%
  add_formula(diabetes ~ .)

# Model tuning
mars_tune <- tune_grid(
  mars_workflow,
  resamples = cv_folds,
  grid = mars_grid
)


autoplot(mars_tune, metric = "roc_auc")
```




```r
mars_tune %>% show_best()
```

```
## # A tibble: 5 x 8
##   num_terms prod_degree .metric .estimator  mean     n std_err .config
##       <int>       <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1         6           1 roc_auc binary     0.832    10  0.0245 Preprocessor1_Mo~
## 2         5           3 roc_auc binary     0.831    10  0.0211 Preprocessor1_Mo~
```

```
## 3         5           4 roc_auc binary     0.829    10  0.0226 Preprocessor1_Mo~
## 4         4           4 roc_auc binary     0.828    10  0.0278 Preprocessor1_Mo~
## 5         7           1 roc_auc binary     0.827    10  0.0252 Preprocessor1_Mo~
```

```
mars_best <- select_best(mars_tune, metric = "roc_auc")

final_mars_spec <- mars_spec %>%
  update(num_terms = mars_best$num_terms,
         prod_degree = mars_best$prod_degree)

mars_fit <- fit(final_mars_spec, formula = diabetes ~ ., data = training_data)

mars_model <- extract_fit_engine(mars_fit)
coef(mars_model)
```

```
##    (Intercept)      h(32-age)   h(35.7-mass)  h(pregnant-4) h(glucose-119)
##     0.17445366    -0.21630436    -0.18104446     1.65845335     0.05210362
##  h(pregnant-3)
##    -1.44048229
```

Test data performance

```
predict_mars <- predict(mars_fit, new_data = testing_data, type = "prob")

eval_data <- data.frame(truth = testing_data$diabetes, mars_prob = predict_mars$.pred_pos)

roc_auc(data = eval_data, truth = truth, mars_prob, event_level = "second")$.estimate
```

```
## [1] 0.7925092
```