Classification I

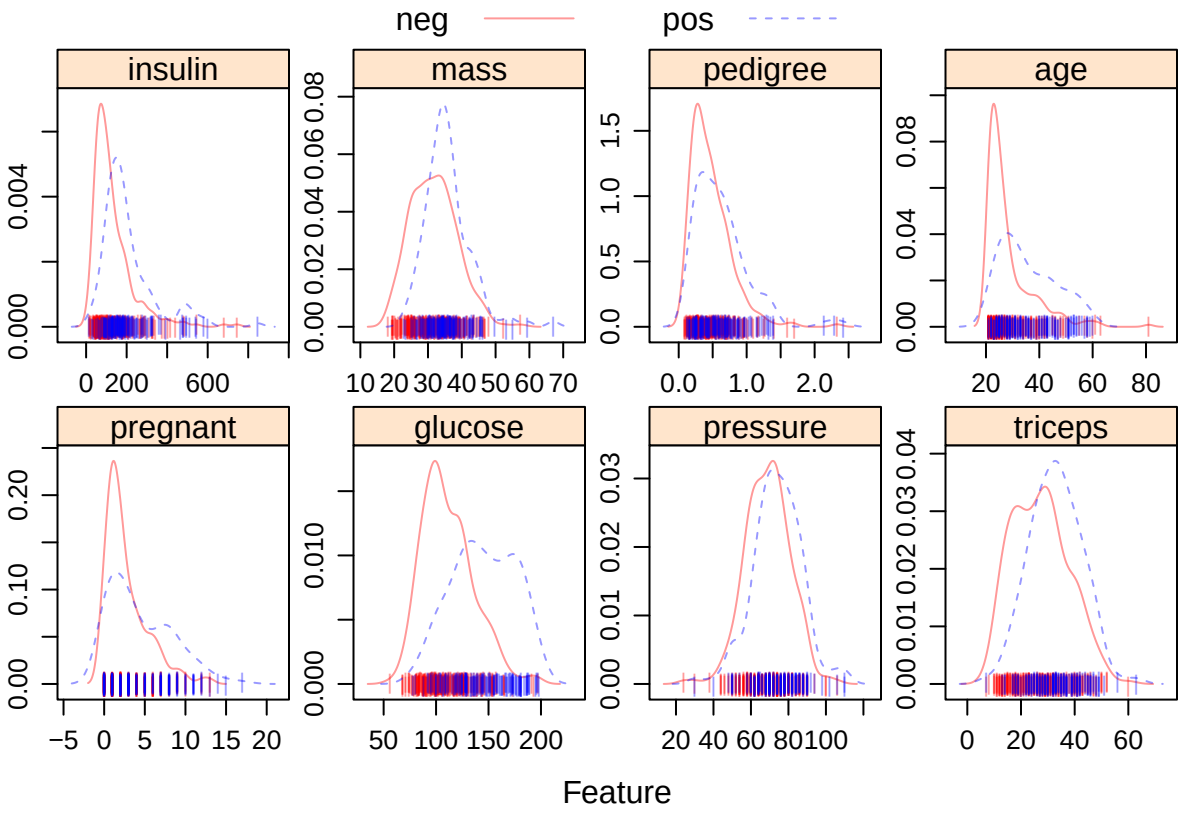
Yifei Sun

Contents

```
library(caret)
library(glmnet)
library(mlbench)
library(pROC)# another package rocr
library(pdp)
library(vip)# variance importance
library(AppliedPredictiveModeling)# one of text book only for theme
```

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)

theme1 <- transparentTheme(trans = .4)
trellis.par.set(theme1)

# density is proper for continuous but you should exclude categorical and binary
featurePlot(x = dat[, 1:8],
            y = dat$diabetes,
            scales = list(x = list(relation = "free"),
                          y = list(relation = "free")),
            plot = "density", pch = "|",
            auto.key = list(columns = 2))
```



```
# pch is only for dots shape
```

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

```
set.seed(1)
rowTrain <- createDataPartition(y = dat$diabetes,
                                p = 0.75,
                                list = FALSE)
```

Logistic regression and its cousins

glm

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

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

```
glm.fit <- glm(diabetes ~ .,
               data = dat,
               subset = rowTrain,
               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.

```
# type = "response" you get the predicted probabilities
test.pred.prob <- predict(glm.fit, newdata = dat[-rowTrain,],
                          type = "response")
test.pred <- rep("neg", length(test.pred.prob))
test.pred[test.pred.prob>0.5] <- "pos"

# data -> a factor of predicted classes , reference -> factor of true
confusionMatrix(data = as.factor(test.pred),
                reference = dat$diabetes[-rowTrain],
                positive = "pos")
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction neg pos
##        neg  60  11
##        pos   5  21
##
##                Accuracy : 0.8351
##                  95% CI : (0.746, 0.9027)
##     No Information Rate : 0.6701
##     P-Value [Acc > NIR] : 0.0002081
```

```
##
##                  Kappa : 0.6083
##
##  Mcnemar's Test P-Value : 0.2112995
##
##             Sensitivity : 0.6562
##             Specificity : 0.9231
##          Pos Pred Value : 0.8077
##          Neg Pred Value : 0.8451
##              Prevalence : 0.3299
##          Detection Rate : 0.2165
##    Detection Prevalence : 0.2680
##       Balanced Accuracy : 0.7897
##
##        'Positive' Class : pos
##
```

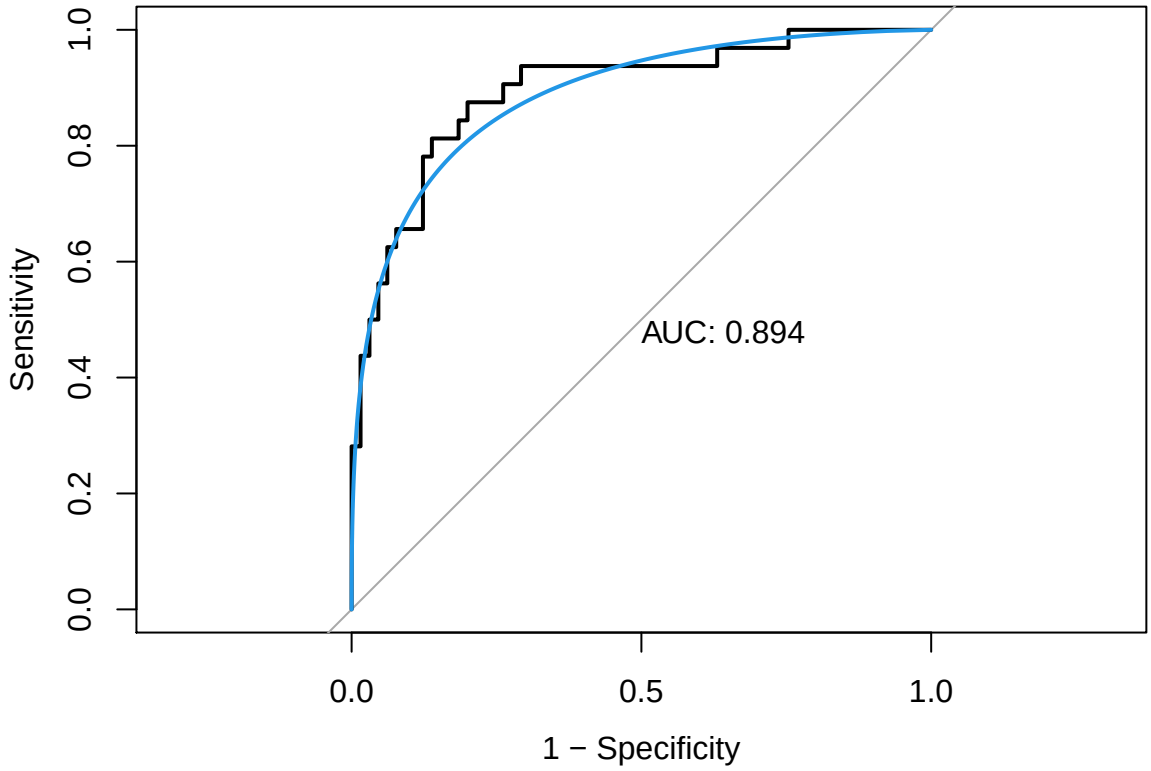
Confusion Matrix:

	Observed	
Predict	Negative	Positive
Negative	a	b
Positive	c	d

- Accuracy: $\frac{a+d}{n}$
- No Information Rate : $\max(\frac{a+c}{n}, \frac{b+d}{n})$
- Kappa: measures the agreement between classification and truth values
  - P_o: observed , accuracy $\frac{a+d}{n}$
  - P_e : probability of agreement by chance, random accuracy, probability that the labels produces by these two processes coincide by chance (assuming independence). $\frac{a+c}{n} \times \frac{a+b}{n} + \frac{b+d}{n} \times \frac{c+d}{n}$
  - $Kappa = \frac{P_o - P_e}{1-P_e}$
  - if perfect classifier $P_o = 1$, $Kappa = 1$; if classifier is same as agreement by chance, which means $P_o = P_e$, $Kappa = 0$. Also $Kappa$ can be negative, but usually -0.4-0.6 -0.6 -0.8+
- Mcnemar test: null hypotheses $P_b = P_c$. The null hypothesis of marginal homogeneity states that the two marginal probabilities for each outcome are the same, e.g.$P_a+P_b =P_a+P_c, P_c+P_d = P_b+P_d $
- Sensitivity: True Positive Rate, $\frac{d}{b+d}$
- Specificity: True Negative Rate, $\frac{a}{a+c}$
- PPV: Positive Predictive Value, $\frac{d}{/}c$
- NPV:
- Prevalence: $\frac{b+d}{n}$
- Detection Rate:
- Detection Prevalence:
- Balanced Accuracy: mean of sensitivity and specificity

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

```
roc.glm <- roc(dat$diabetes[-rowTrain], 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 = "repeatedcv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)
set.seed(1)
model.glm <- train(x = dat[rowTrain,1:8],
                   y = dat$diabetes[rowTrain],
                   method = "glm",
                   metric = "ROC",
                   trControl = ctrl)
# seed control the trainControl, the repeated cv procedure
```

Penalized logistic regression

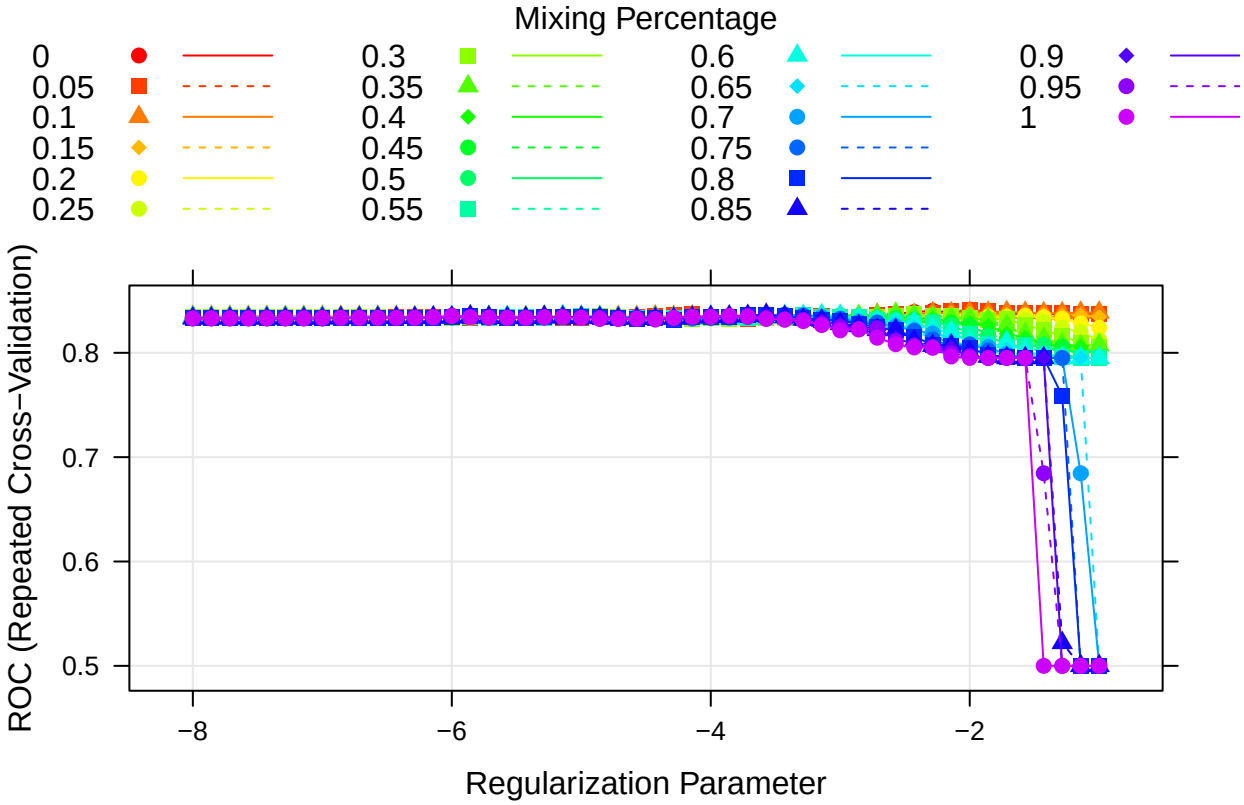
Penalized logistic regression can be fitted using `glmnet`. We use the `train` function to select the optimal tuning parameters.

```
glmnGrid <- expand.grid(.alpha = seq(0, 1, length = 21),
                        .lambda = exp(seq(-8, -1, length = 50)))
set.seed(1)
model.glmn <- train(x = dat[rowTrain,1:8],
                    y = dat$diabetes[rowTrain],
```

```
                    method = "glmnet",
                    tuneGrid = glmnGrid,
                    metric = "ROC",
                    trControl = ctrl)

model.glmn$bestTune
```

```
##    alpha    lambda
## 93  0.05 0.1353353
```

```
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))
```



Now we want to find the tuning parameters *maximizing* the function

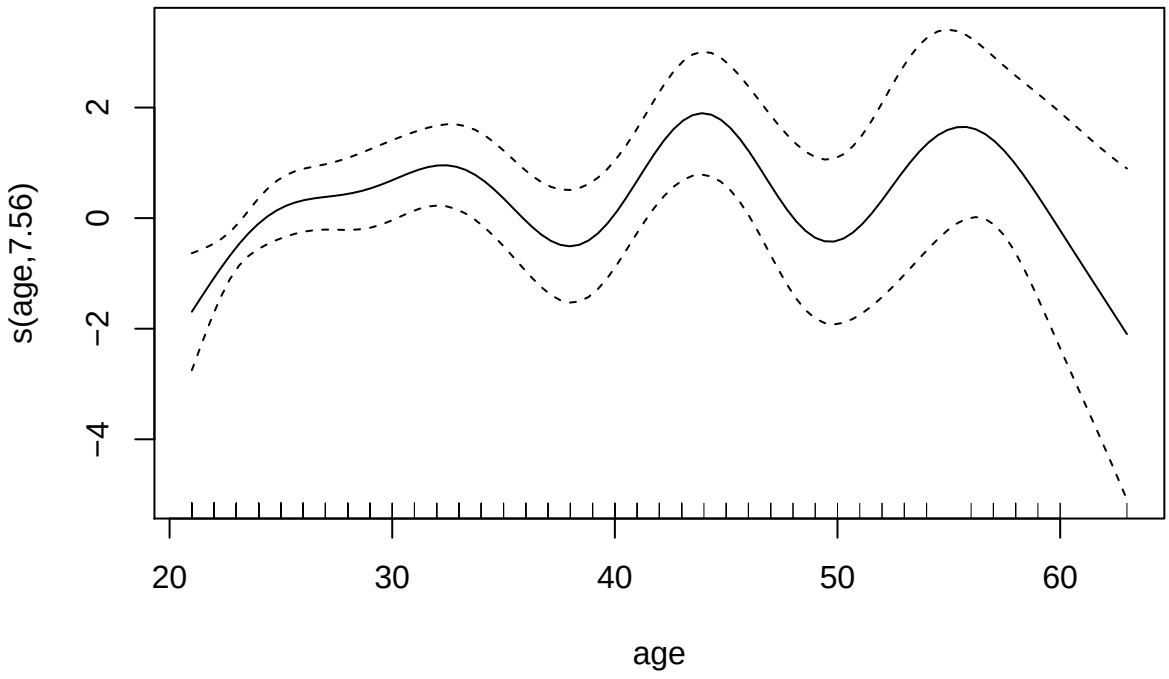
GAM

```
set.seed(1)
model.gam <- train(x = dat[rowTrain,1:8],
                   y = dat$diabetes[rowTrain],
```

```
                    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:
## 0.0001 0.0000 7.5614 1.3490 2.2830 0.0000 0.0000
## 1.6659  total = 13.86
##
## UBRE score: -0.0602217
```

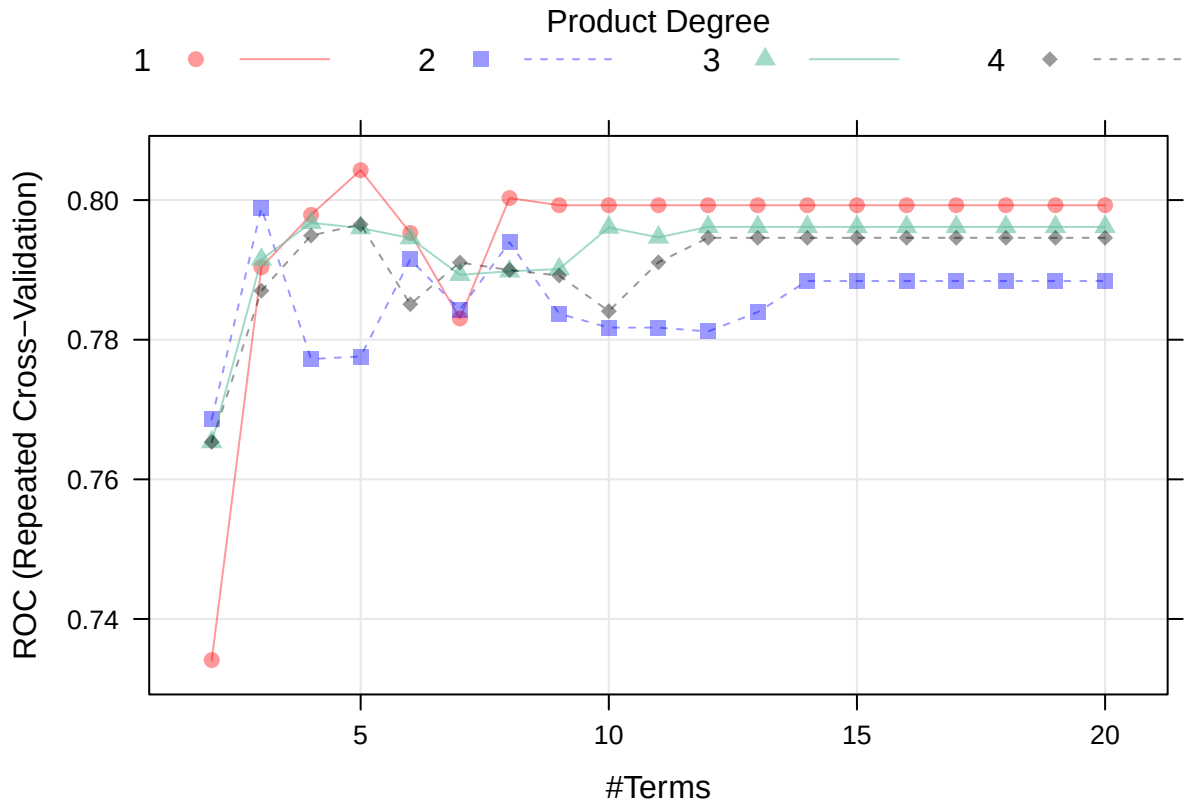
```
plot(model.gam$finalModel, select = 3)
```



we see edf 0.0001, means the model try to shrink this term towards zero

MARS

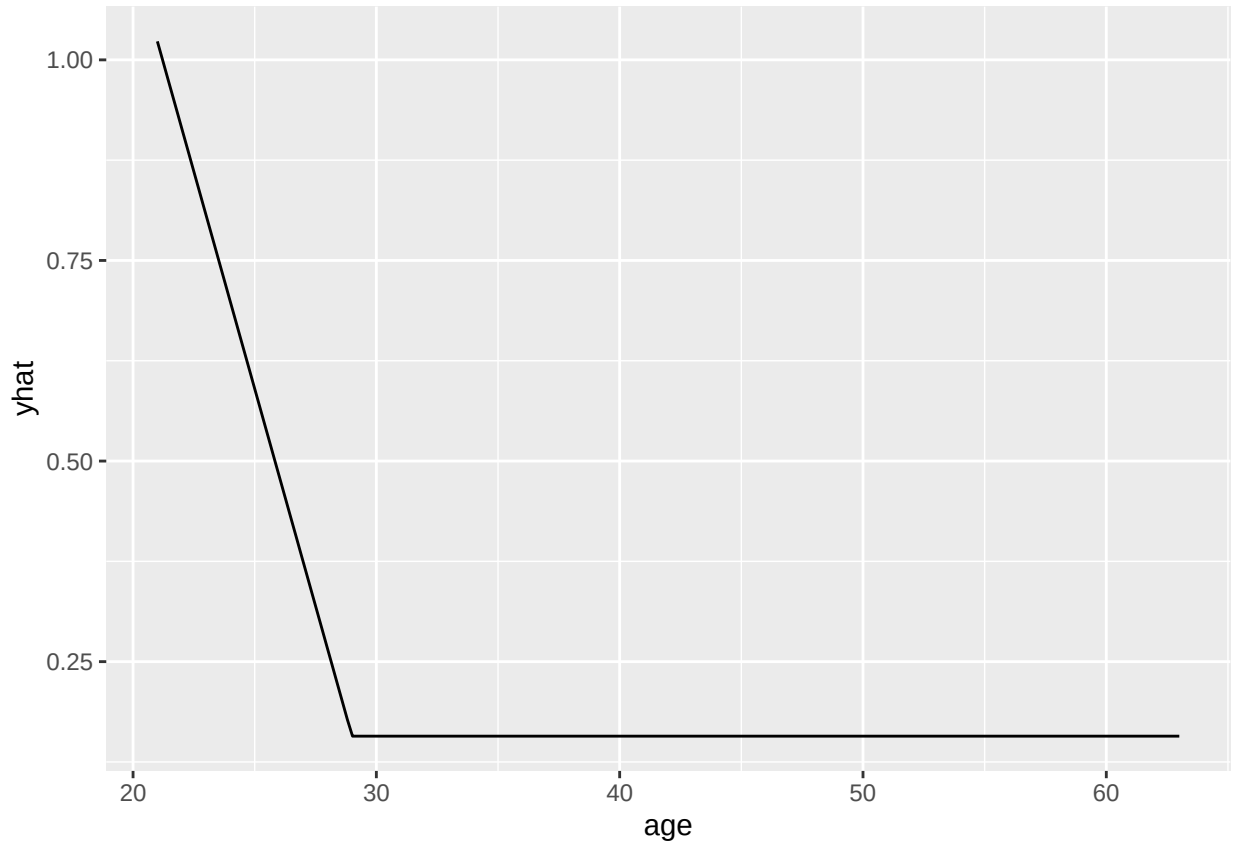
```
set.seed(1)
model.mars <- train(x = dat[rowTrain,1:8],
                    y = dat$diabetes[rowTrain],
                    method = "earth",
                    tuneGrid = expand.grid(degree = 1:4,
                                           nprune = 2:20),
                    metric = "ROC",
   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

plot(model.mars)
```



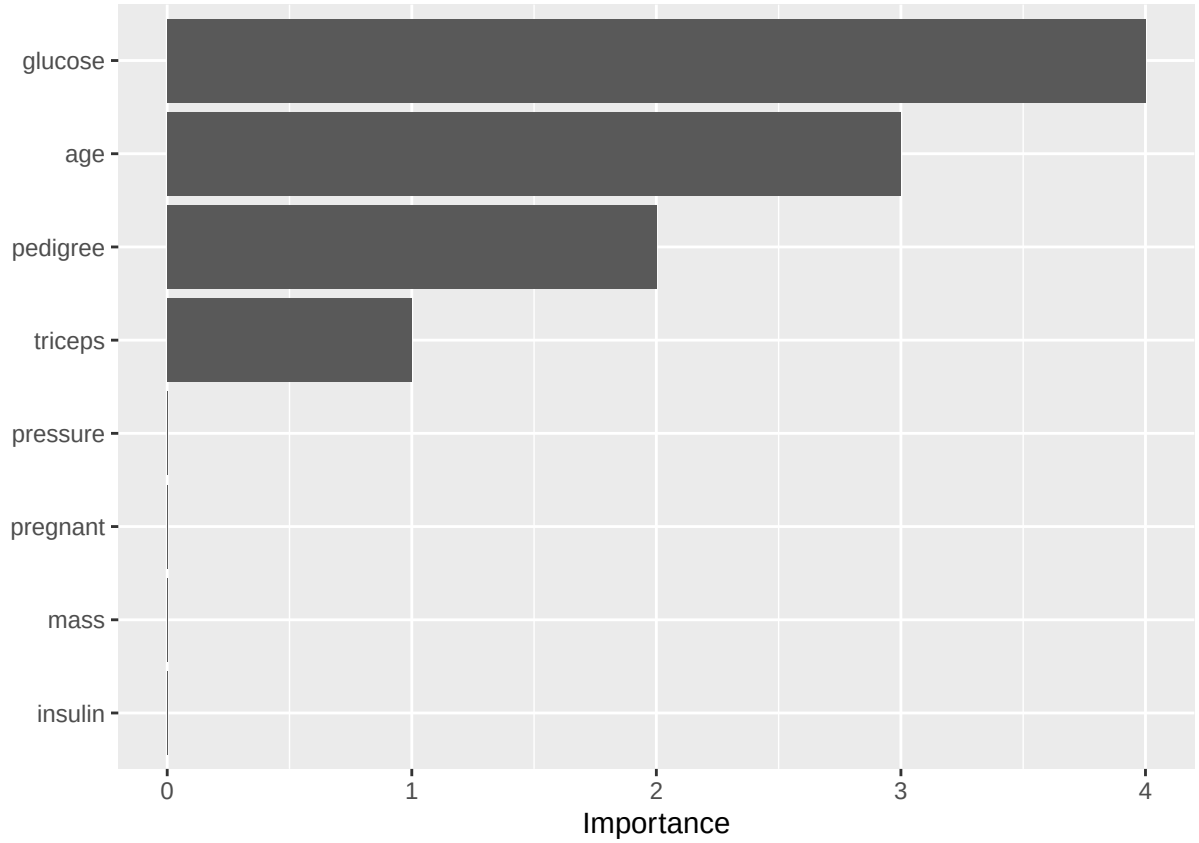

```
coef(model.mars$finalModel)
```

```
##       (Intercept)    h(glucose-117)        h(29-age) h(1.258-pedigree)
##         0.47846627        0.04364894       -0.21648937       -1.34468216
##     h(37-triceps)
##        -0.04894219
```

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




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



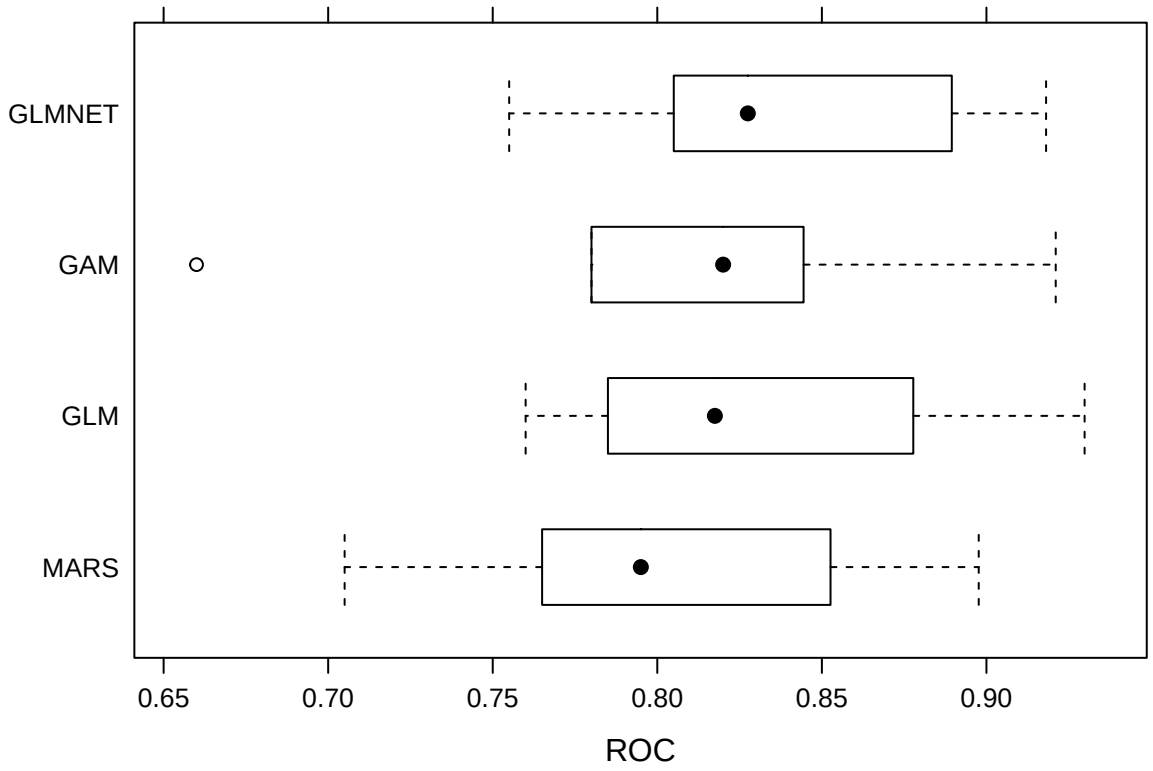
vip: variance importance in MARS, each term added to model, if no VIP, means they do not enter the model. Overall impact of variables

```
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    0.760 0.79125 0.8175 0.8329971 0.8754386 0.9298246    0
## GLMNET 0.755 0.80750 0.8275 0.8406813 0.8881579 0.9181287    0
## GAM    0.660 0.78625 0.8200 0.8174357 0.8412281 0.9210526    0
## MARS   0.705 0.76625 0.7950 0.8042836 0.8478070 0.8976608    0
##
## Sens
##        Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.75 0.8500000 0.8947368 0.8686842 0.9000000 0.9473684    0
```

```
## GLMNET 0.85 0.8625000 0.9000000 0.9092105 0.9473684 1.0000000    0
## GAM    0.75 0.8500000 0.9000000 0.8794737 0.9355263 1.0000000    0
## MARS   0.75 0.8421053 0.8500000 0.8531579 0.8875000 0.9473684    0
##
## Spec
##        Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM     0.3 0.4250000 0.5777778 0.5222222 0.6000000 0.7000000    0
## GLMNET  0.3 0.3500000 0.5000000 0.4822222 0.5888889 0.6666667    0
## GAM     0.3 0.4250000 0.5500000 0.5766667 0.7000000 0.8888889    0
## MARS    0.2 0.4111111 0.5000000 0.5022222 0.6000000 0.7777778    0
```

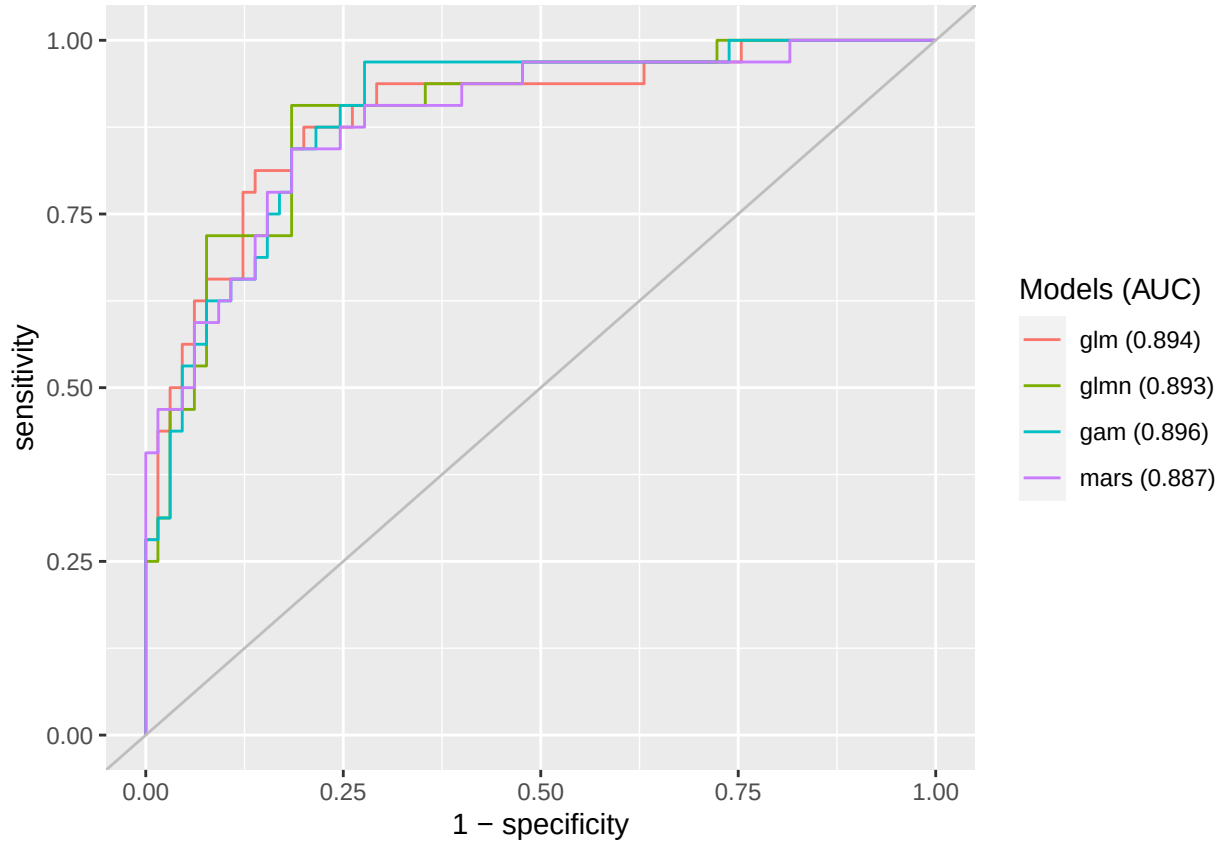
```
bwplot(res, metric = "ROC")
```



Now let's look at the test data performance.

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

roc.glm <- roc(dat$diabetes[-rowTrain], glm.pred)
roc.glmn <- roc(dat$diabetes[-rowTrain], glmn.pred)
roc.gam <- roc(dat$diabetes[-rowTrain], gam.pred)
roc.mars <- roc(dat$diabetes[-rowTrain], 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)
```