DSII Midterm Project

Yiru Gong, yg2832; Yiwen Zhao, yz4187; Jiaqi Chen, jc5681

2022-05-10

Contents

```r
library(tidyverse)
library(summarytools)
library(corrplot)
library(caret)
library(MASS)
library(mlbench)
library(pROC) #ROCR
library(pdp)
library(vip)
library(AppliedPredictiveModeling) #for transparentTheme function
library(keras)
library(tfruns)
library(ISLR)
library(caret)
library(e1071)
library(kernlab)
```

Data Input

```r
data = read.csv('Covid19_vacc_predict_handout.csv')
data = data %>%
  na.omit() %>%
  dplyr::select(-id) %>%
  mutate(
    atlas_type_2015_mining_no = factor(atlas_type_2015_mining_no),
    covid_vaccination = factor(covid_vaccination),
    hum_region = factor(hum_region),
    sex_cd = factor(sex_cd),
    race_cd = factor(race_cd),
    lang_spoken_cd = factor(lang_spoken_cd),
    atlas_low_education_2015_update = factor(atlas_low_education_2015_update)
    )
dfSummary(data[,c(5,7,8,10,11,17,18)])
```

Data Frame Summary
Dimensions: 8308 x 7
Duplicates: 7802

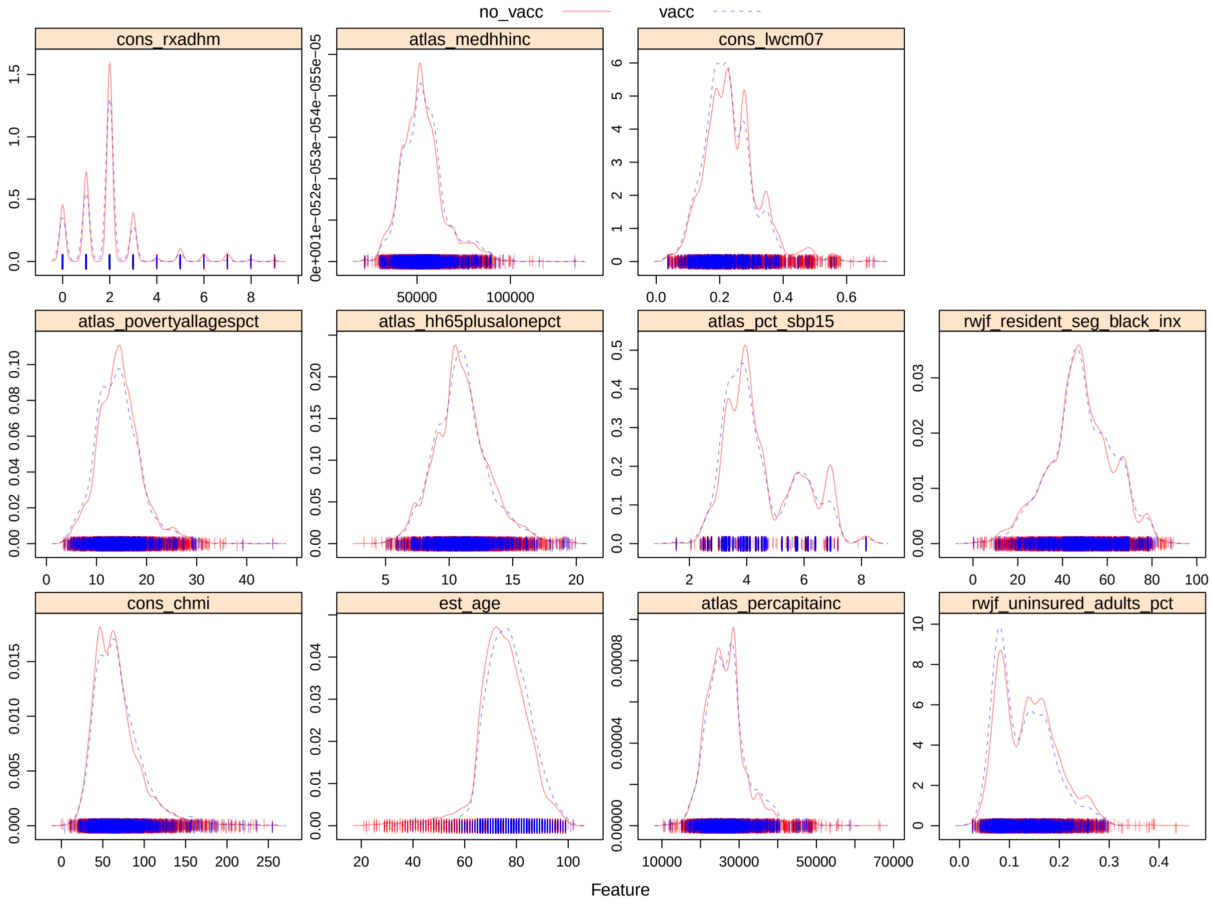
No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Valid	Missing
1	atlas_type_2015_mining_no [factor]	1. 0 2. 1	8177 (98.4%) 131 (1.6%)	IIIIIIIIIIIIIIIIIII	8308 (100.0%)	0 (0.0%)
2	covid_vaccination [factor]	1. no_vacc 2. vacc	6682 (80.4%) 1626 (19.6%)	IIIIIIIIIIIIIIII III	8308 (100.0%)	0 (0.0%)

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Valid	Missing
3	hum_region [factor]	1. CALIFORNIA/NEVADA 2. CENTRAL 3. CENTRAL WEST 4. EAST 5. EAST CENTRAL 6. FLORIDA 7. GREAT LAKES/CENTRAL NORTH 8. GULF STATES 9. INTERMOUNTAIN 10. MID-ATLANTIC/NORTH CAROLI [5 others]	299 (3.6%) 551 (6.6%) 238 (2.9%) 491 (5.9%) 1370 (16.5%) 607 (7.3%) 1111 (13.4%) 454 (5.5%) 220 (2.6%) 845 (10.2%) 2122 (25.5%)	I	8308 (100.0%)	0 (0.0%)
4	sex_cd [factor]	1. F 2. M	4527 (54.5%) 3781 (45.5%)	IIIIIIIIII IIIIIIIII	8308 (100.0%)	0 (0.0%)
5	lang_spoken_cd [factor]	1. * 2. CHI 3. CRE 4. ENG 5. KOR 6. OTH 7. SPA 8. VIE	10 (0.1%) 13 (0.2%) 4 (0.0%) 7957 (95.8%) 7 (0.1%) 34 (0.4%) 276 (3.3%) 7 (0.1%)		8308 (100.0%)	0 (0.0%)
6	atlas_low_education_2015_update [factor]	1. 0 2. 1	7769 (93.5%) 539 (6.5%)	IIIIIIIIIIIIIIIIII I	8308 (100.0%)	0 (0.0%)
7	race_cd [factor]	1. 0 2. 1 3. 2 4. 3 5. 4 6. 5 7. 6	160 (1.9%) 7317 (88.1%) 558 (6.7%) 80 (1.0%) 56 (0.7%) 129 (1.6%) 8 (0.1%)	IIIIIIIIIIIIIIII I	8308 (100.0%)	0 (0.0%)

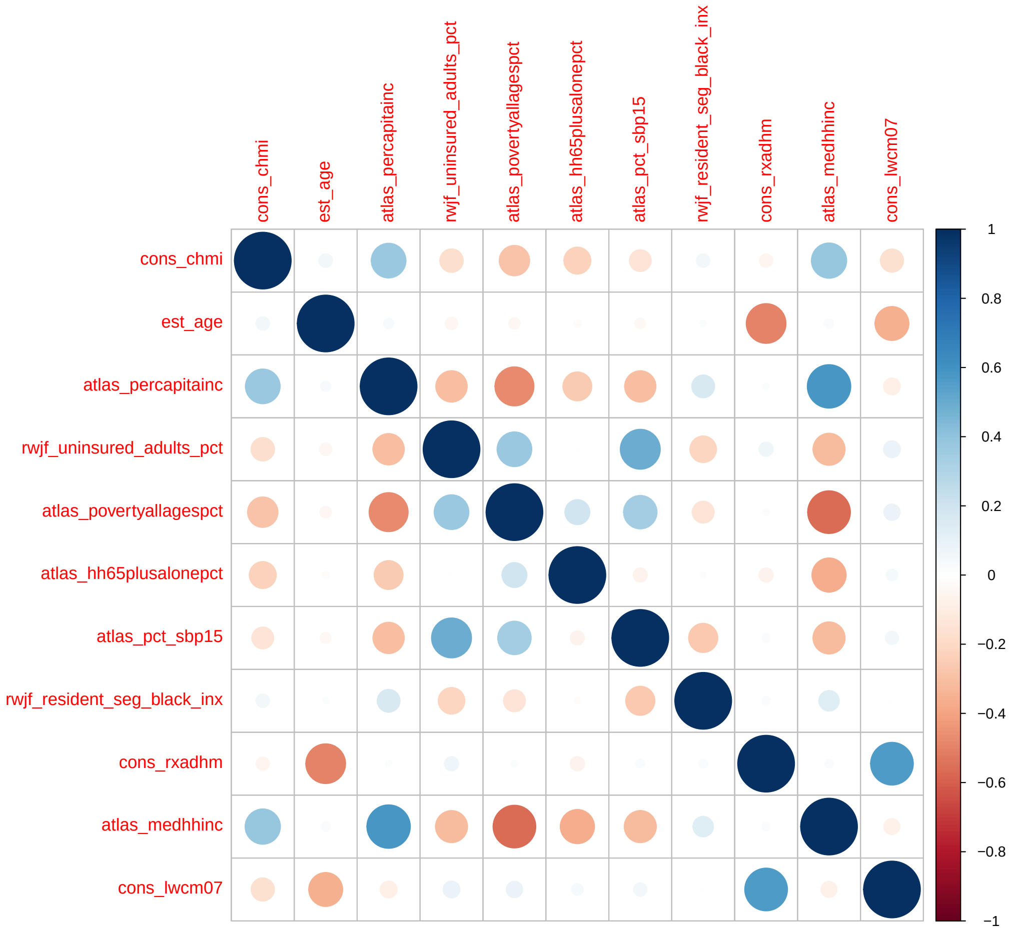
```
data2 = model.matrix(covid_vaccination ~ ., data)[ ,-1]
```

Exploratory analysis

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

#figure 1
featurePlot(x = data[,-c(5,7,8,10,11,17,18)],
            y = data$covid_vaccination,
            scales = list(x = list(relation = "free"),
                          y = list(relation = "free")),
            plot = "density", pch = "|",
            auto.key = list(columns = 2))
```



```
#correlation
corrplot(cor(data[,-c(5,7,8,10,11,17,18)]), method = "circle", type = "full")
```



Data split

```
set.seed(1)
rowTrain <- createDataPartition(y = data$covid_vaccination,
                                p = 0.7,
                                list = FALSE)
x = data2[rowTrain,]
y = data$covid_vaccination[rowTrain]
x2 = data2[-rowTrain,]
y2 = data$covid_vaccination[-rowTrain]

save(x,y,x2,y2,file = "split_data.Rdata")
```

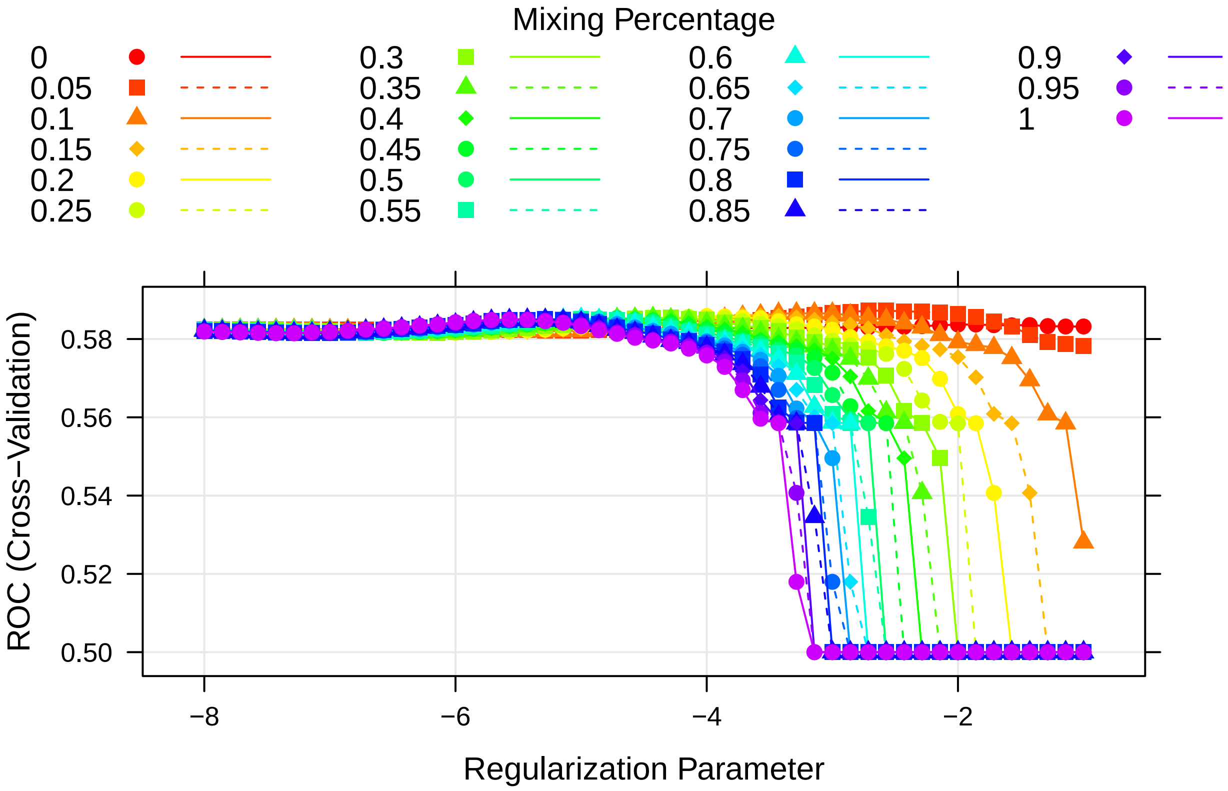
Model fitting

Penalized logistic regression (GLMNET)

```
ctrl <- trainControl(method = "cv",
                     summaryFunction = twoClassSummary,
                      classProbs = TRUE)

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

model.glmn$bestTune
```

```
##    alpha     lambda
## 89  0.05 0.07642629
```

```
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))
```

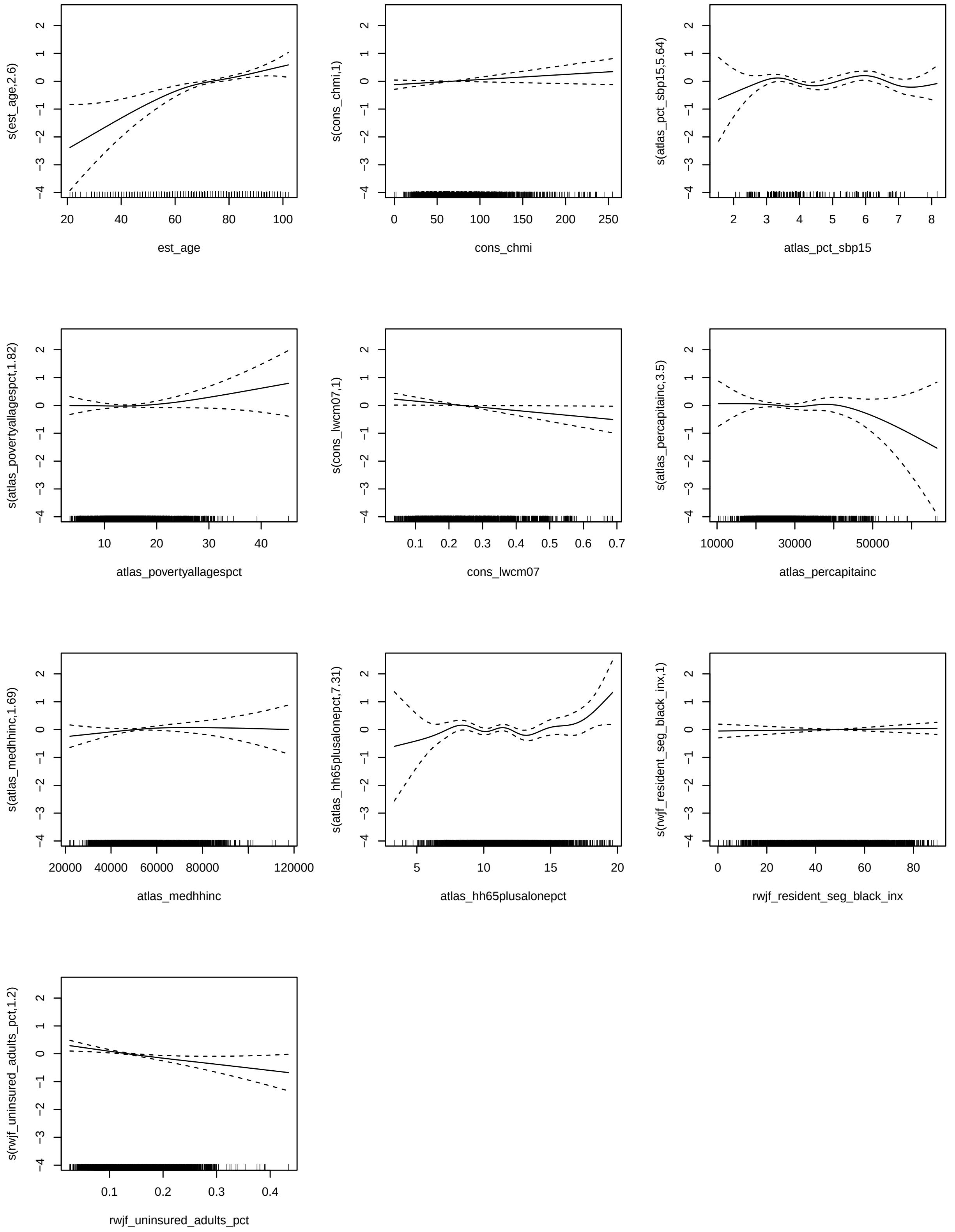


GAM

```
set.seed(1)
model.gam <- train(data[rowTrain,-c(7:8)], y,
                   method = "gam",
                   metric = "ROC",
                   trControl = ctrl)
### row 8: hum_region report error

model.gam$finalModel
```

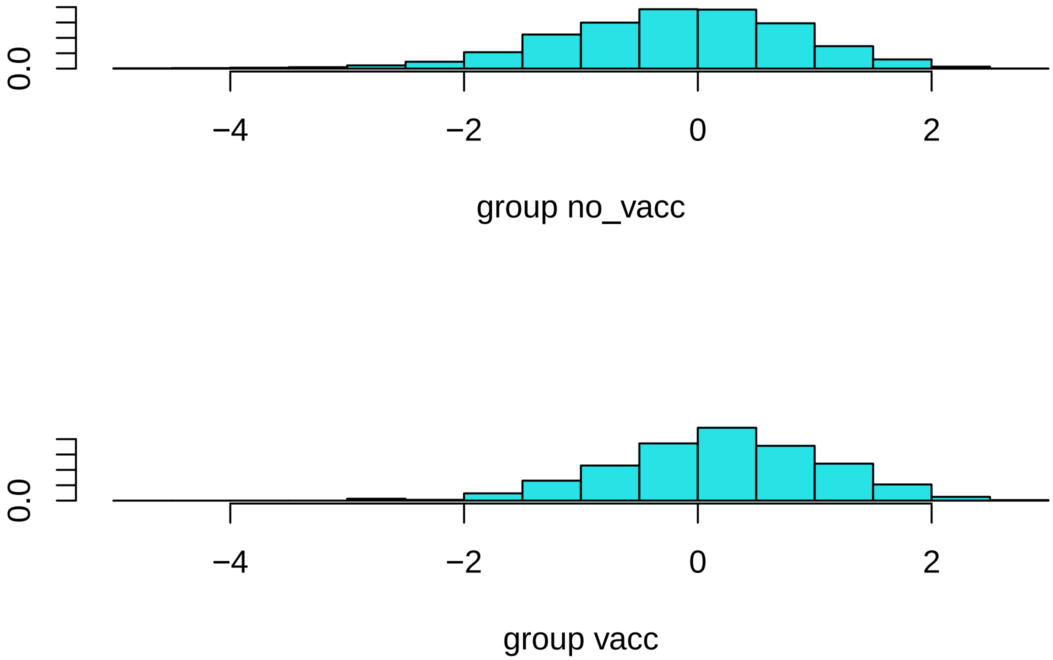
```
##
## Family: binomial
## Link function: logit
##
## Formula:
## .outcome ~ sex_cd + atlas_low_education_2015_update + race_cd +
##     cons_rxadhm + s(est_age) + s(cons_chmi) + s(atlas_pct_sbp15) +
##     s(atlas_povertyallagespct) + s(cons_lwcm07) + s(atlas_percapitainc) +
##     s(atlas_medhhinc) + s(atlas_hh65plusalonepct) + s(rwjf_resident_seg_black_inx) +
##     s(rwjf_uninsured_adults_pct)
##
## Estimated degrees of freedom:
## 2.60 1.00 5.64 1.82 1.00 3.50 1.69
## 7.31 1.00 1.20  total = 36.76
##
## UBRE score: -0.02449249
```

```
# fig 2
par(mfrow=c(4,3))
plot(model.gam$finalModel)
```


s(est_age,2.6)
2 1 0 −1 −2 −3 −4
20 40 60 80 100
est_age
s(cons_chmi,1)
2 1 0 −1 −2 −3 −4
0 50 100 150 200 250
cons_chmi
s(atlas_pct_sbp15,5.64)
2 1 0 −1 −2 −3 −4
2 3 4 5 6 7 8
atlas_pct_sbp15
s(atlas_povertyallagespct,1.82)
2 1 0 −1 −2 −3 −4
10 20 30 40
atlas_povertyallagespct
s(cons_lwcm07,1)
2 1 0 −1 −2 −3 −4
0.1 0.2 0.3 0.4 0.5 0.6 0.7
cons_lwcm07
s(atlas_percapitainc,3.5)
2 1 0 −1 −2 −3 −4
10000 30000 50000
atlas_percapitainc
s(atlas_medhhinc,1.69)
2 1 0 −1 −2 −3 −4
20000 40000 60000 80000 120000
atlas_medhhinc
s(atlas_hh65plusalonepct,7.31)
2 1 0 −1 −2 −3 −4
5 10 15 20
atlas_hh65plusalonepct
s(rwjf_resident_seg_black_inx,1)
2 1 0 −1 −2 −3 −4
0 20 40 60 80
rwjf_resident_seg_black_inx
s(rwjf_uninsured_adults_pct,1.2)
2 1 0 −1 −2 −3 −4
0.1 0.2 0.3 0.4
rwjf_uninsured_adults_pct

LDA

```
lda.fit <- lda(y~x)
plot(lda.fit)
```

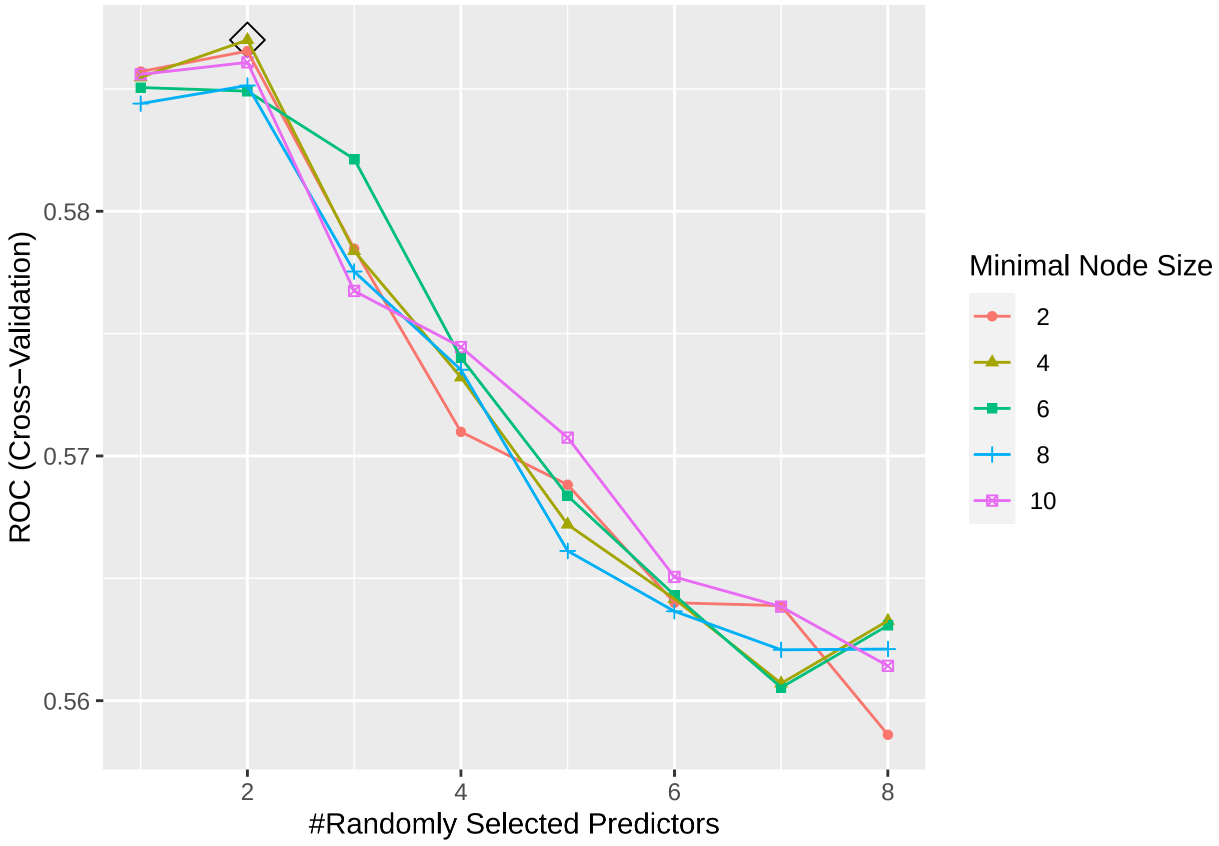


```
set.seed(1)
model.lda <- train(x, y,
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl)
```

Random Forest

```
# rf.ctrl <- trainControl(method = "cv",
#                       classProbs = TRUE,
#                       summaryFunction = twoClassSummary)

rf.grid <- expand.grid(mtry = 1:8,
                       splitrule = "gini",
                       min.node.size = seq(from = 2, to = 10,
                                           by = 2))
```

```
set.seed(1)
rf.fit <- train(covid_vaccination ~ . ,
                data,
                subset = rowTrain,
                method = "ranger",
                tuneGrid = rf.grid,
                metric = "ROC",
                trControl = ctrl)

ggplot(rf.fit, highlight = TRUE)
```



Support Vector Machine

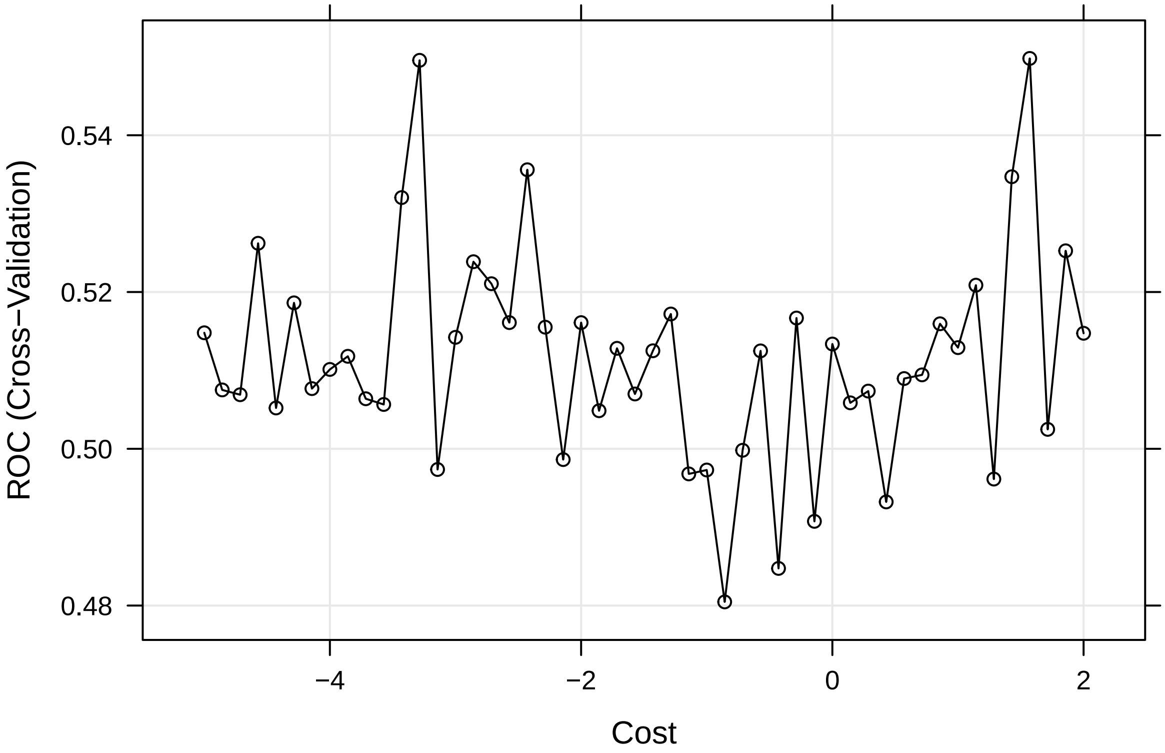
```
data$covid_vaccination <- factor(data$covid_vaccination, c("vacc", "no_vacc"))
dat <- data[-c(5,8,10,11,17,18)]
summary(dat)
```

```
##    cons_chmi         est_age       atlas_percapitainc rwjf_uninsured_adults_pct
##  Min.   :  0.00   Min.   : 21.00   Min.   :10399      Min.   :0.02616
##  1st Qu.: 47.00   1st Qu.: 70.00   1st Qu.:23056      1st Qu.:0.08593
##  Median : 62.00   Median : 75.00   Median :26132      Median :0.13357
##  Mean   : 67.21   Mean   : 75.18   Mean   :26685      Mean   :0.13645
##  3rd Qu.: 79.00   3rd Qu.: 81.00   3rd Qu.:28949      3rd Qu.:0.17323
##  Max.   :255.00   Max.   :102.00   Max.   :66522      Max.   :0.43395
##  atlas_povertyallagespct covid_vaccination atlas_hh65plusalonepct
##  Min.   : 3.40           vacc   :1626      Min.   : 3.309
##  1st Qu.:11.60           no_vacc:6682      1st Qu.: 9.626
##  Median :14.40                             Median :10.878
##  Mean   :14.61                             Mean   :10.993
##  3rd Qu.:17.00                             3rd Qu.:12.155
##  Max.   :45.20                             Max.   :19.960
##  atlas_pct_sbp15 rwjf_resident_seg_black_inx  cons_rxadhm    atlas_medhhinc
##  Min.   :1.546   Min.   : 0.2584             Min.   :0.000   Min.   : 22045
##  1st Qu.:3.525   1st Qu.:40.3734             1st Qu.:1.000   1st Qu.: 45813
##  Median :4.110   Median :47.8798             Median :2.000   Median : 51865
##  Mean   :4.559   Mean   :48.1327             Mean   :1.921   Mean   : 53259
##  3rd Qu.:5.710   3rd Qu.:57.8018             3rd Qu.:2.000   3rd Qu.: 58742
##  Max.   :8.160   Max.   :89.6102             Max.   :9.000   Max.   :134609
##   cons_lwcm07
##  Min.   :0.03724
##  1st Qu.:0.18190
##  Median :0.22509
##  Mean   :0.23641
##  3rd Qu.:0.28204
##  Max.   :0.68722
```

```
# SVM with Linear Kernal
# ctrl1 <- trainControl(method = "cv")
set.seed(1)
svml.fit <- train(covid_vaccination ~ . ,
                  data = dat[rowTrain,],
                  method = "svmLinear",
                  metric = "ROC",
                  # preProcess = c("center", "scale"),
                  tuneGrid = data.frame(C = exp(seq(-5,2,len=50))),
                  trControl = ctrl)
```

```
## maximum number of iterations reached 0.003289937 -0.003225867maximum number of iterations reached 0.0
```

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



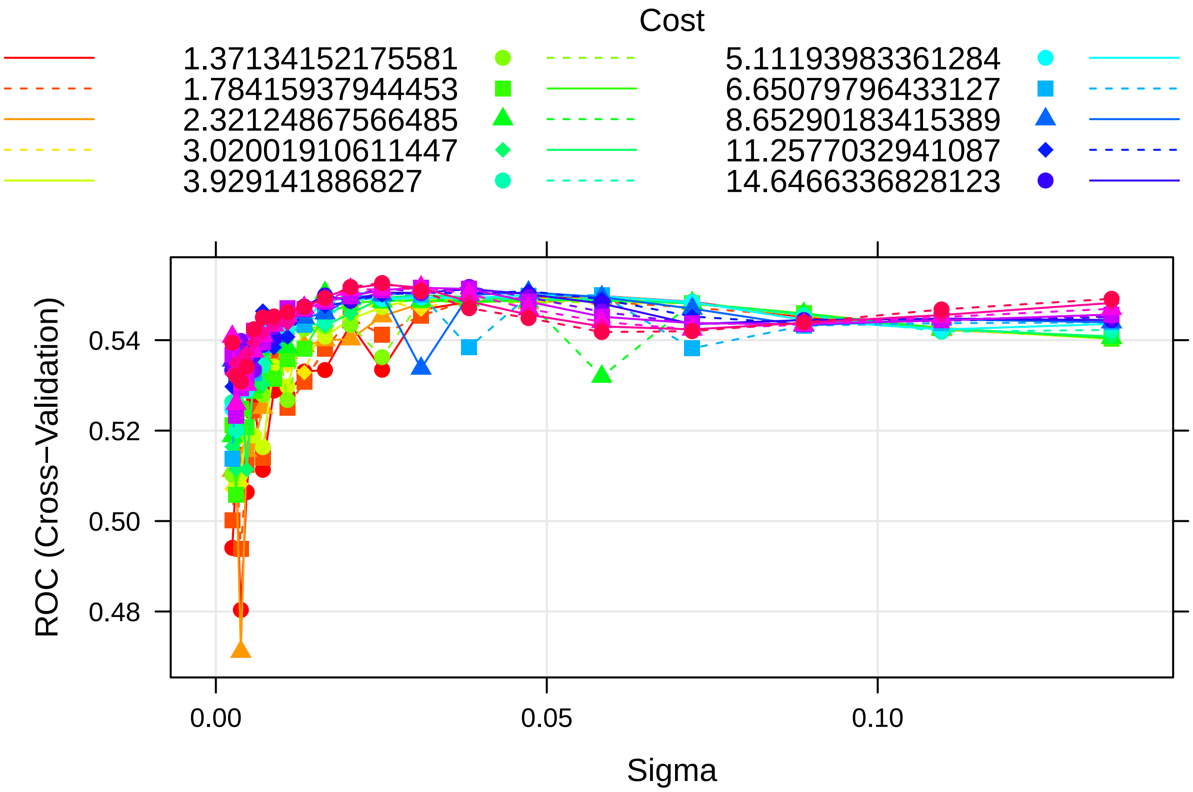
```
svml.fit$bestTune
```

```
##           C
## 47 4.81352
```

```
# SVM with Radial Kernel
svmr.grid <- expand.grid(C = exp(seq(-1,4,len=20)),
                         sigma = exp(seq(-6,-2,len=20)))

set.seed(1)
svmr.fit <- train(covid_vaccination ~ . , dat,
                  subset = rowTrain,
                  method = "svmRadialSigma",
                  metric = "ROC",
                  tuneGrid = svmr.grid,
                  trControl = ctrl)
```

```
## maximum number of iterations reached 0.001902067 -0.001852983maximum number of iterations reached 0.0
```

```
myCol <- rainbow(20)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))

plot(svmr.fit, highlight = TRUE, par.settings = myPar)
```



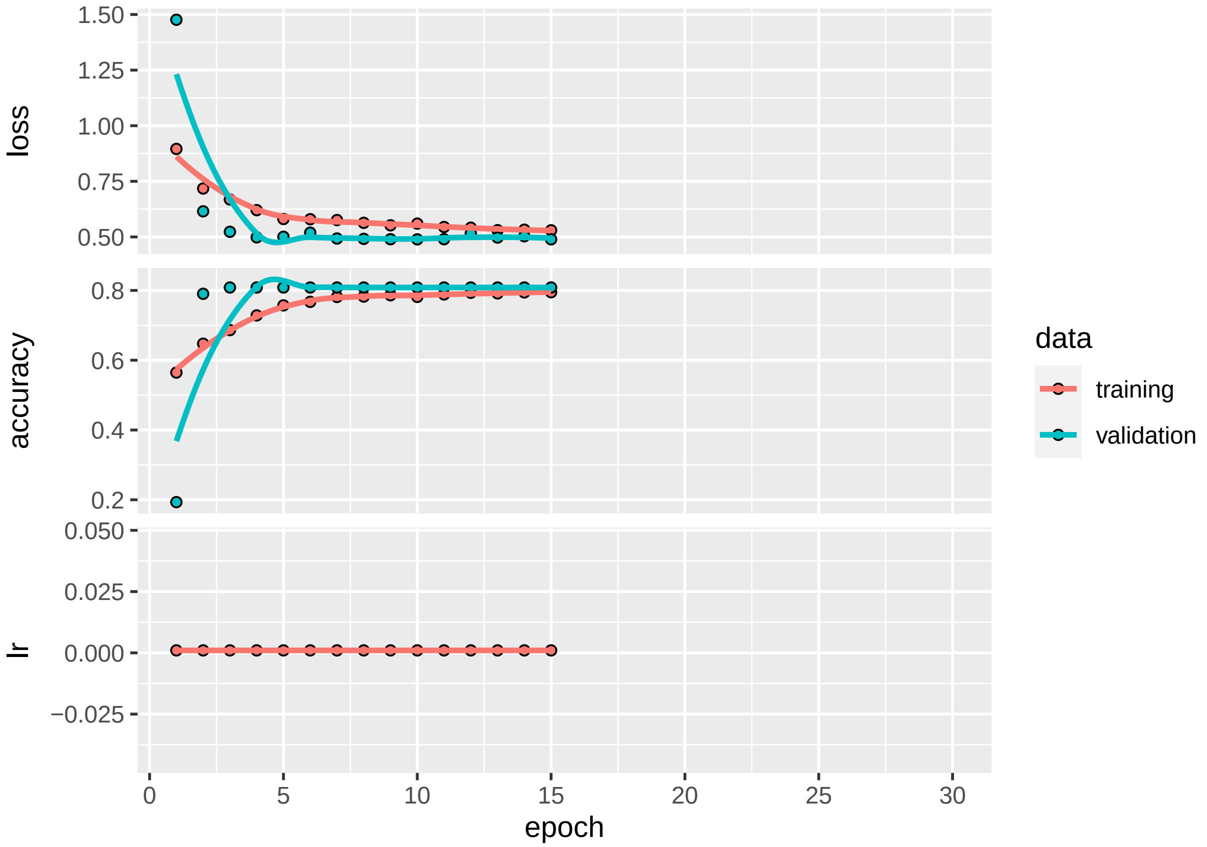
```
svmr.fit$bestTune
```

```
##        sigma        C
## 392 0.025117 54.59815
```

Neural Network

```
## tuning
set.seed(1)
runs <- tuning_run("keras_grid_search.R",
                   flags = list(
                   nodes_layer1 = c(64, 128, 256),
                   nodes_layer2 = c(64, 128, 256),
                   nodes_layer3 = c(64, 128, 256),
                   dropout_layer1 = c(0.2, 0.3, 0.4),
                   dropout_layer2 = c(0.2, 0.3, 0.4),
                   dropout_layer3 = c(0.2, 0.3, 0.4)),
                   confirm = FALSE,
                   echo = FALSE,
                   sample = 0.01) # try more after class

best = runs[which.max(runs$metric_val_accuracy),]
```

```
y_c = ifelse(y=="vacc",1,0)
y_c <- to_categorical(y_c, 2)
y2_c = ifelse(y2=="vacc",1,0)
y2_c <- to_categorical(y2_c, 2)

model.nn <- keras_model_sequential() %>%
  layer_dense(units = best$flag_nodes_layer1, activation = "relu", input_shape = ncol(x)) %>%
  layer_batch_normalization() %>%
  layer_dropout(rate = best$flag_dropout_layer1) %>%
  layer_dense(units = best$flag_nodes_layer2, activation = "relu") %>%
  layer_batch_normalization() %>%
  layer_dropout(rate = best$flag_dropout_layer2) %>%
  layer_dense(units = best$flag_nodes_layer3, activation = "relu") %>%
  layer_batch_normalization() %>%
  layer_dropout(rate = best$flag_dropout_layer3) %>%
  layer_dense(units = 2, activation = "sigmoid") %>%
  compile(loss = "categorical_crossentropy",
          optimizer = optimizer_rmsprop(),
          metrics = "accuracy")
fit.nn = model.nn %>%
  fit(x = x,
      y = y_c,
      epochs = 30,
      batch_size = 256,
      validation_split = 0.2,
      callbacks = list(callback_early_stopping(patience = 5),
                       callback_reduce_lr_on_plateau()),
      verbose = 2)
plot(fit.nn)
```



```
## testing and evaluation
score <- model.nn %>% evaluate(x2, y2_c)
score
```

```
##      loss  accuracy
## 0.5050931 0.8044962
```

Model Comparison

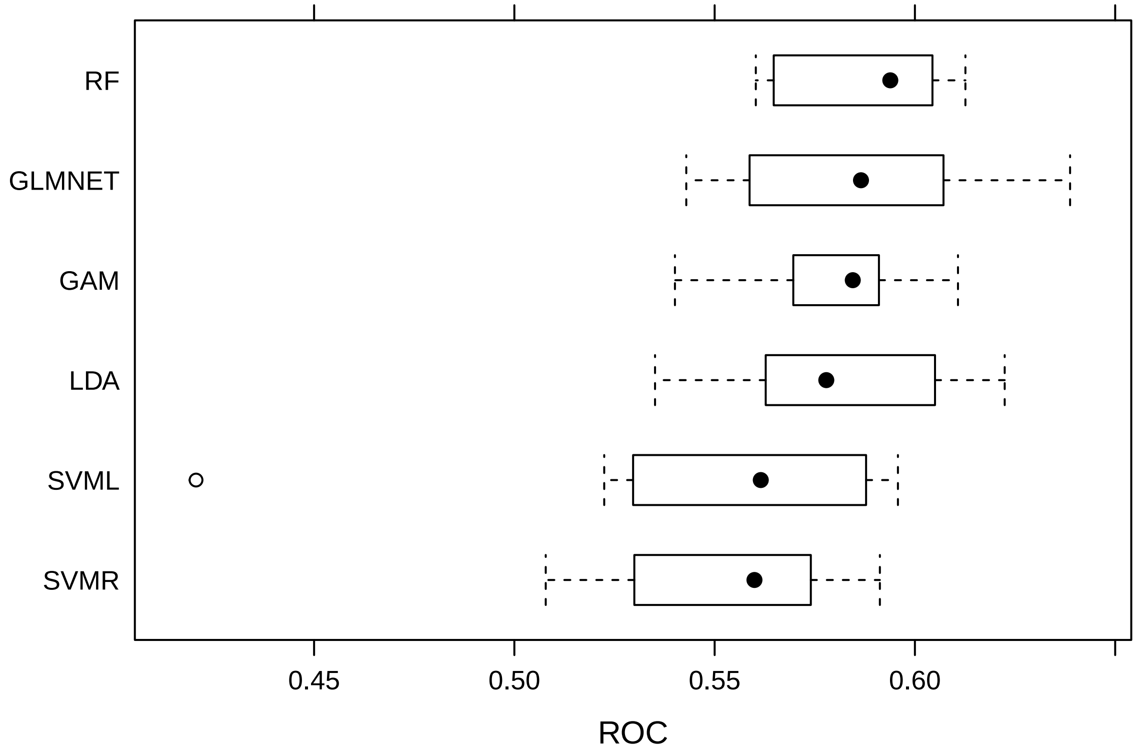
CV Compare

```
res <- resamples(list(GLMNET = model.glmn,
                      GAM = model.gam,
                      LDA = model.lda,
                      RF = rf.fit,
                      SVML = svml.fit,
                      SVMR = svmr.fit))

#KNN
summary(res)
```

```
##
```

```
## Call:
## summary.resamples(object = res)
##
## Models: GLMNET, GAM, LDA, RF, SVML, SVMR
## Number of resamples: 10
##
## ROC
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLMNET 0.5429205 0.5624063 0.5865405 0.5872412 0.6059043 0.6387389    0
## GAM    0.5400922 0.5726839 0.5844861 0.5816581 0.5903860 0.6107550    0
## LDA    0.5351253 0.5645383 0.5778804 0.5822242 0.6033940 0.6224134    0
## RF     0.5602789 0.5649677 0.5938578 0.5870003 0.6036325 0.6126106    0
## SVML   0.4205278 0.5348807 0.5615171 0.5498035 0.5852967 0.5957602    0
## SVMR   0.5078348 0.5310246 0.5599415 0.5526272 0.5734883 0.5912506    0
##
## Sens
##             Min. 1st Qu. Median      Mean 3rd Qu. Max. NA's
## GLMNET 1.0000000       1      1 1.0000000       1    1    0
## GAM    0.9978632       1      1 0.9995726       1    1    0
## LDA    1.0000000       1      1 1.0000000       1    1    0
## RF     1.0000000       1      1 1.0000000       1    1    0
## SVML   0.0000000       0      0 0.0000000       0    0    0
## SVMR   0.0000000       0      0 0.0000000       0    0    0
##
## Spec
##        Min. 1st Qu. Median        Mean 3rd Qu.       Max. NA's
## GLMNET    0       0      0 0.000000000       0 0.00000000    0
## GAM       0       0      0 0.000877193       0 0.00877193    0
## LDA       0       0      0 0.000000000       0 0.00000000    0
## RF        0       0      0 0.000000000       0 0.00000000    0
## SVML      1       1      1 1.000000000       1 1.00000000    0
## SVMR      1       1      1 1.000000000       1 1.00000000    0
```

```r
# figure 4
bwplot(res, metric = "ROC")
```

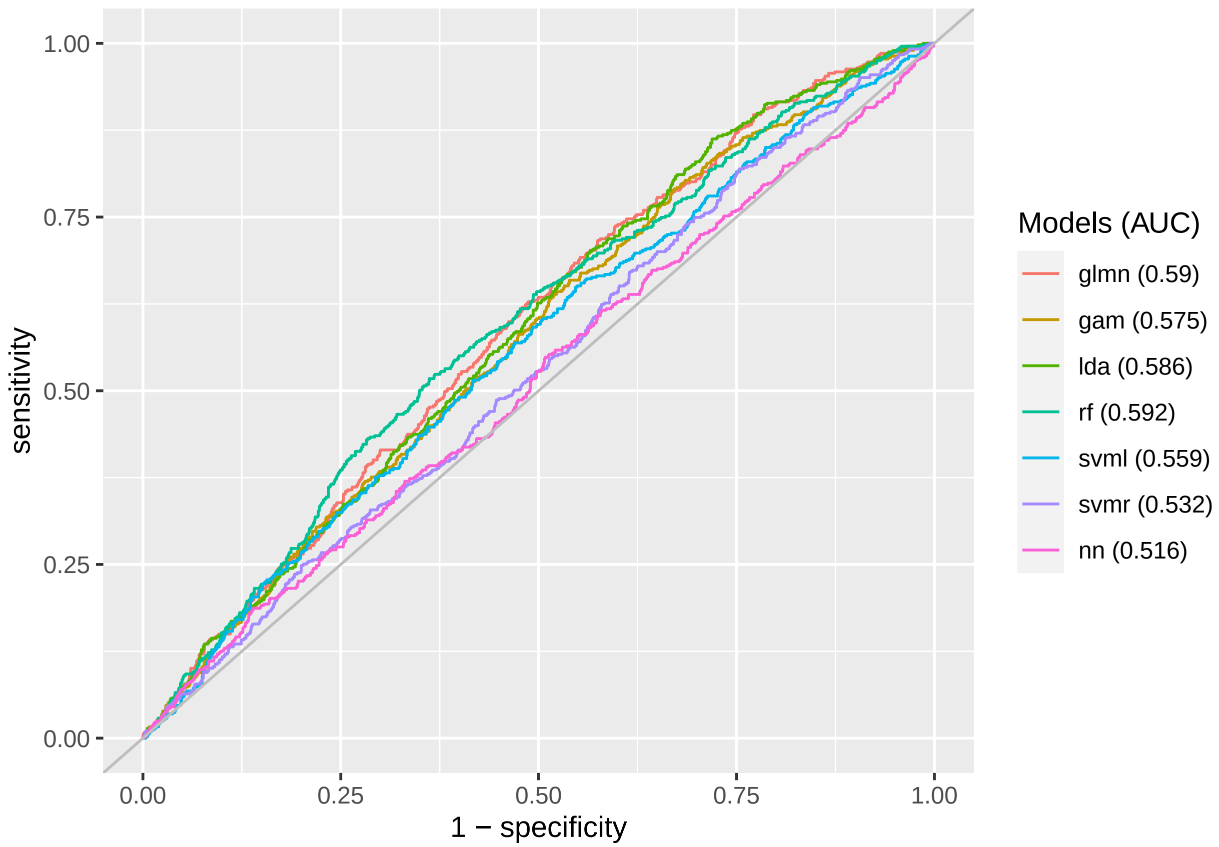


Test data performance

```
# # raw pred
# glmn.pred <- predict(model.glmn, newdata = x2, type = "raw")
# gam.pred <- predict(model.gam, newdata = data[-rowTrain,-c(7:8)], type = "raw")
# lda.pred <- predict(model.lda, newdata = x2, type = "raw")
# rf.pred <- predict(rf.fit, newdata = data[-rowTrain,], type = "raw")
#
# svml.pred <- predict(svml.fit, newdata = dat[-rowTrain,], type = "raw")
# svmr.pred <- predict(svmr.fit, newdata = dat[-rowTrain,], type = "raw")
#
# pred_test <- model.nn %>% predict(x2) %>% k_argmax() %>% as.matrix() %>% as.numeric()
# nn.pred = ifelse(pred_test==0,"no_vacc","vacc")
# nn.pred = factor(nn.pred,levels = c("no_vacc","vacc"))
#
# # Confusion Matrix
# cm.glmn = confusionMatrix(data = glmn.pred, reference = y2, positive = "vacc")$overall
# cm.gam = confusionMatrix(data = gam.pred, reference = y2, positive = "vacc")$overall
# cm.lda = confusionMatrix(data = lda.pred, reference = y2, positive = "vacc")$overall
# cm.rf = confusionMatrix(data = rf.pred, reference = y2, positive = "vacc")$overall
# cm.svml = confusionMatrix(data = svml.pred, reference = y2, positive = "vacc")$overall
# cm.svmr = confusionMatrix(data = svmr.pred, reference = y2, positive = "vacc")$overall
# cm.nn = confusionMatrix(data = nn.pred, reference = y2, positive = "vacc")$overall
#
```

```
# cm_df = data.frame(GLMN = cm.glmn, GAM = cm.gam, LDA = cm.lda, RF = cm.rf, SVML = cm.svml, SVMR = cm.
# knitr::kable(cm_df, digits = 4)
```

```
glmn.pred <- predict(model.glmn, newdata = x2, type = "prob")[,2]
gam.pred <- predict(model.gam, newdata = data[-rowTrain,-c(7:8)], type = "prob")[,2]
lda.pred <- predict(model.lda, newdata = x2, type = "prob")[,2]
rf.pred <- predict(rf.fit, newdata = data[-rowTrain,], type = "prob")[,2]
svml.pred <- predict(svml.fit, newdata = dat[-rowTrain,], type = "prob")[,2]
svmr.pred <- predict(svmr.fit, newdata = dat[-rowTrain,], type = "prob")[,2]

pred_test <- model.nn %>% predict(x2)
nn.pred = pred_test[,2]

roc.glmn <- roc(y2, glmn.pred)
roc.gam <- roc(y2, gam.pred)
roc.lda <- roc(y2, lda.pred)
roc.rf <- roc(y2, rf.pred)
roc.svml <- roc(y2,svml.pred)
roc.svmr <- roc(y2,svmr.pred)
roc.nn = roc(y2,nn.pred)

auc <- c(roc.glmn$auc[1], roc.gam$auc[1], roc.lda$auc[1],roc.rf$auc[1],roc.svml$auc[1],roc.svmr$auc[1],

modelNames <- c("glmn","gam","lda","rf","svml","svmr","nn")

# fig 5
ggroc(list(roc.glmn, roc.gam, roc.lda, roc.rf,roc.svml, roc.svmr, roc.nn), legacy.axes = TRUE) +
  scale_color_discrete(labels = paste0(modelNames, " (", round(auc,3),")"),
                       name = "Models (AUC)") +
  geom_abline(intercept = 0, slope = 1, color = "grey")
```


1.00
0.75
0.50
0.25
0.00
sensitivity
0.00
0.25
0.50
0.75
1.00
1 − specificity
Models (AUC)
glmn (0.59)
gam (0.575)
lda (0.586)
rf (0.592)
svml (0.559)
svmr (0.532)
nn (0.516)