R Notebook

R Studio API Code

Libraries

```r
library(tidyverse)
library(haven)
library(caret)
library(RANN)
```

Data Import and Cleaning

```r
gss <- read_sav("../Data/GSS2006.sav") %>%
  select(BIG5A1,BIG5B1,BIG5C1,BIG5D1,BIG5E1,BIG5A2,BIG5B2,BIG5C2,BIG5D2,BIG5E2,HEALTH) %>%
  mutate_all(as.numeric)
gss_tbl <- gss[rowSums(is.na(gss[,1:10]))!=ncol(gss[,1:10]) & !is.na(gss[,11]),]
pander::pander(head(gss_tbl))
```

Table 1: Table continues below

BIG5A1	BIG5B1	BIG5C1	BIG5D1	BIG5E1	BIG5A2	BIG5B2	BIG5C2	BIG5D2
2	2	2	2	2	2	2	4	2
5	1	1	2	1	1	3	5	5
2	2	1	4	1	2	4	3	1
1	1	1	1	4	2	5	5	5
1	1	2	1	1	3	5	5	5
2	2	2	2	2	2	3	5	3

BIG5E2	HEALTH
4	3
5	1
4	1
5	3
5	3
3	2

I read in the 10 personality predictors and the health criterion, converted all to numeric variables, and removed rows where there was no data at all, rows where all 10 predictor variables were missing, and rows where the criterion was missing.

Analysis

```
pp <- preProcess(gss_tbl[,-11],
                 method=c("center","scale","zv","knnImpute"))
gss_preprocessed <- predict(pp,newdata=gss_tbl)

set.seed(2020)
rows <- sample(nrow(gss_preprocessed))
shuffled_gss <- gss_preprocessed[rows,]
gss_holdout <- shuffled_gss[1:250,]
gss_train <- shuffled_gss[251:nrow(shuffled_gss),]

ols_model <- train(
  HEALTH~.^2,
  gss_train,
  method="glm",
  trControl=trainControl(method="cv",number=10,verboseIter=F),
  na.action=na.pass
)
ols_val_train <- cor(predict(ols_model,gss_train,na.action=na.pass),gss_train$HEALTH) # correlation bet
ols_val_holdout <- cor(predict(ols_model,gss_holdout,na.action=na.pass),gss_holdout$HEALTH) # correlati

glmnet_model <- train(
  HEALTH~.^2,
  gss_train,
  method="glmnet",
  trControl=trainControl(method="cv",number=10,verboseIter=F),
  na.action=na.pass
)
glmnet_val_train <- cor(predict(glmnet_model,gss_train,na.action=na.pass),gss_train$HEALTH) # correlati
glmnet_val_holdout <- cor(predict(glmnet_model,gss_holdout,na.action=na.pass),gss_holdout$HEALTH) # cor

svm_model <- train(
  HEALTH~.^2,
  gss_train,
  method="svmLinear",
  trControl=trainControl(method="cv",number=10,verboseIter=F),
  na.action=na.pass
)
svm_val_train <- cor(predict(svm_model,gss_train,na.action=na.pass),gss_train$HEALTH) # correlation bet
svm_val_holdout <- cor(predict(svm_model,gss_holdout,na.action=na.pass),gss_holdout$HEALTH) # correlati

xgb_model <- train(
  HEALTH~.^2,
  gss_train,
  method="xgbTree",
  trControl=trainControl(method="cv",number=10,verboseIter=F),
  na.action=na.pass
```

```r
)
xgb_val_train <- cor(predict(xgb_model,gss_train,na.action=na.pass),gss_train$HEALTH) # correlation bet
xgb_val_holdout <- cor(predict(xgb_model,gss_holdout,na.action=na.pass),gss_holdout$HEALTH) # correlati

validities <- matrix(c(ols_val_train,ols_val_holdout,
                       glmnet_val_train,glmnet_val_holdout,
                       svm_val_train,svm_val_holdout,
                       xgb_val_train,xgb_val_holdout),
                     nrow=4,ncol=2,byrow=T,dimnames=list(c("ols","glmnet","svm","xgb"),
                                                         c("train","holdout")))
pander::pander(validities)
```

	train	holdout
ols	0.2936	0.164
glmnet	0.2541	0.2141
svm	0.2645	0.1711
xgb	0.3536	0.2622

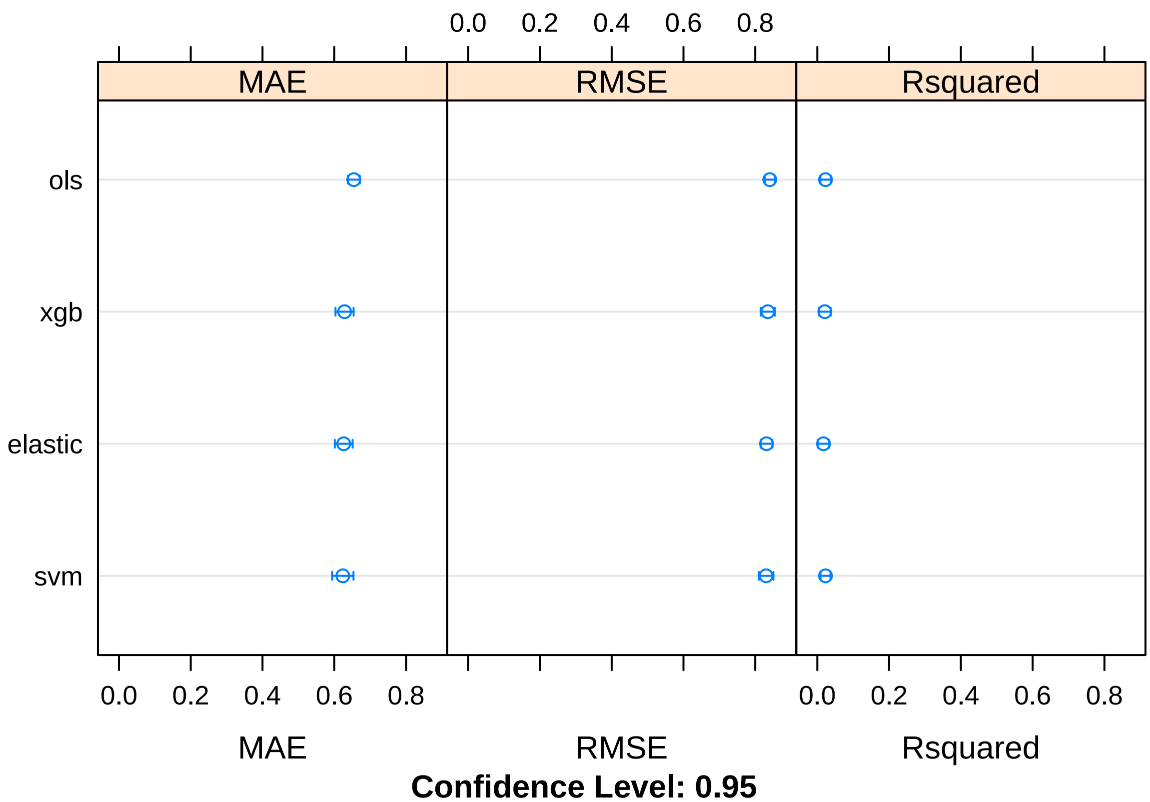
```r
summary(resamples(list("ols"=ols_model,"elastic"=glmnet_model,"svm"=svm_model,"xgb"=xgb_model)))
```

```
##
## Call:
## summary.resamples(object = resamples(list(ols = ols_model, elastic
##  = glmnet_model, svm = svm_model, xgb = xgb_model)))
##
## Models: ols, elastic, svm, xgb
## Number of resamples: 10
##
## MAE
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## ols     0.6089230 0.6405301 0.6604256 0.6543565 0.6662192 0.6828501    0
## elastic 0.5783609 0.6117698 0.6192162 0.6262211 0.6378564 0.6879304    0
## svm     0.5554573 0.5953747 0.6265282 0.6237493 0.6387097 0.7070409    0
## xgb     0.5678623 0.6155808 0.6361417 0.6286158 0.6486813 0.6782854    0
##
## RMSE
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## ols     0.8124804 0.8261625 0.8368205 0.8404638 0.8570135 0.8676765    0
## elastic 0.7954076 0.8180466 0.8291962 0.8314074 0.8405640 0.8694776    0
## svm     0.7807662 0.8159438 0.8303278 0.8304015 0.8444090 0.8809867    0
## xgb     0.7885402 0.8221633 0.8432825 0.8349607 0.8543988 0.8686232    0
##
## Rsquared
##                 Min.     1st Qu.      Median       Mean    3rd Qu.       Max.
## ols     0.0007916426 0.006473956 0.022388377 0.02288110 0.03070297 0.06722579
## elastic 0.0006341155 0.004432708 0.007341556 0.01716379 0.02299664 0.07491850
## svm     0.0001663978 0.009874320 0.020620584 0.02285823 0.02805948 0.05562887
## xgb     0.0004073072 0.001736327 0.018199538 0.02097668 0.02881553 0.07255770
##         NA's
## ols        0
## elastic    0
```

```
## svm         0
## xgb         0
```

Visualization

```r
dotplot(resamples(list("ols"=ols_model,"elastic"=glmnet_model,"svm"=svm_model,"xgb"=xgb_model)))
```



Because the final hypertuning parameters used for the elastic net model was $\alpha = 1$, $\lambda = .022$, the optimal model was a LASSO regression.

When evaluating different models with metrics of RMSE and R^2, elastic net regression (LASSO really) had the lowest RMSE and the highest R^2, and therefore is the best-performing model out of the four. However, when examining validities for both the training and the holdout sample, the results are somewhat discrepant. Extreme gradient boosted regression has the highest correlation with the criterion in the training sample (.36), as well the highest correlaion in the holdout sample (.25). Elastic net regression performed well in terms of RMSE and R^2, but had the lowest correlation in the training sample. However, its validity in the holdout sample is the second highest, showing good generalizability. Support vector regression and OLS models had poorer validities in the holdout sample. Therefore, I prefer the elastic net regression model overall.