Predictive modelling

George, John & William

11/17/2021

Data Preparation and cleaning

Getting the missing values percentages

```
##   Number_Missing Missing_Rate        Variable
## 1           7087     56.53318      enrollment
## 2           7087     56.53318      employment
## 3           9448     75.36694 employment_type
## 4           9448     75.36694 weekly_work_hrs
## 5           7361     58.71889       ethnicity
## 6           7383     58.89438          gender
```

```
## [1] 4998   22
```

missing value treatment

Method I

Treating imbalance classification

```r
library(ROSE)
```

```
## Loaded ROSE 0.0-4
```

```r
data_pad_balance<-ovun.sample(FI_q30 ~ ., data = data_pad, method = "both", p=0.5,
dim(data_pad_balance)
```

```
## [1] 4998   22
```

Converting predictors to category

Partitioning data set

```
## [1] 3350   22
```

```
## [1] 1648   22
```

The training data has 76 observations with 1887 now (old =1057 when compared) variables. The testing data has 32 observation with 1887 now (old= 1057 when compared) variables.

Model fitting

```r
#------ Model building -----------
# Create a wrapper function to abstract away the common aspects of model fitting
```

```r
formula<- FI_q30~.
fit.model <- function(method, tunegrid="", data=NULL, formula=NULL) {

  data <- training
  if(is.null(formula)) formula<- FI_q30~.

  # Train the model
   train(
           formula,
           data = data,
           method = method,
           trControl = trainControl(method = "cv", 5),
           preProcess = c("center","scale"),
           tuneGrid = tunegrid)

}
```

```r
# Logistic Regression
log <-train(formula,
                 data=training,
                 method="glm",
                 family = binomial(link = "logit"),
                trControl = trainControl(method = "cv", 5),
                preProcess = c("center", "scale"))
```

```r
# LDA
lda <- train(formula,
                 data=training,
                 method="lda",
                trControl = trainControl(method = "cv", 5),
                preProcess = c("center", "scale"))
```

```r
#-------------- Elastic Net Models -------------------
# fit a LASSO model
lasso <- fit.model("glmnet", expand.grid(.alpha=1, .lambda=seq(0,0.1,0.01)))

# Fit a Ridge regression model
ridge <- fit.model("glmnet", expand.grid(.alpha=0, .lambda=seq(0,0.1,0.01)))
```

```r
# Bagging
# bag <- fit.model("rf", data.frame(mtry=11))
bag <- train(formula,
                 data=training,
                 method="rf",
                trControl = trainControl(method = "cv", 5),
                preProcess = c("center", "scale"),
                tuneGrid = data.frame(mtry=11),
                ntree = 1000)
```

```r
# Random Forest
# rf <- fit.model("rf", data.frame(mtry=1:10))
# rf <- train(formula,
#                  data=training,
#                  method="rf",
#                 trControl = trainControl(method = "cv", 5),
```

```
#                  preProcess = c("center", "scale"),
#                  tuneGrid = data.frame(mtry=1:10),
#                  ntree = 1000)
```

```
#---------------

# Support Vector Machine with linear kernel
set.seed(125)
trctrl <- trainControl(method = "cv", number=5)
svc <- train(formula, data = training, method = "svmLinear",
                    trControl=trctrl, prob.model=T,
                    tuneLength = 10)
```

```
# Support Vector Machine with radial kernel
set.seed(125)
trctrl <- trainControl(method = "cv", number=5)
svmR <- train(formula , data = training, method = "svmRadial",
                    trControl=trctrl, prob.model=T,
                    tuneLength = 10)
```

Making predictions

```
# pred <- function(model){
#   model <- lasso
#     pred.test<- predict(model, testing)
#     misscal<- round(mean(pred.test != testing$FI_q30),digits = 2)
#
#    #  test_pred_1 <- predict(model, newdata = testing, type= "prob")
#    #  ROC_SR <- roc(testing$FI_q30, predictor = test_pred_1[,2])
#    #  # # plot(ROC_SR, col="brown")
#    #  AUC<-round(ROC_SR$auc, digits=4)*100  # AUC
#    #  # text(x=0.4, y=0.25, paste("Area Under Curve = ", AUC, sep=""), col="blue", cex=1.2)
#    # AUC  = 0
#   return(list(missclass=misscal))
#}
```

Metrics

```
library(mlr3measures)
```

```
## In order to avoid name clashes, do not attach 'mlr3measures'. Instead, only load the namespace with

##
## Attaching package: 'mlr3measures'

## The following object is masked from 'package:pROC':
##
##     auc

## The following objects are masked from 'package:caret':
##
##     precision, recall, sensitivity, specificity

## The following object is masked from 'package:MASS':
##
##     fbeta
```

```r
# Create a custom confusion matrix with performance metrics
metrics <- function(model_object, response="", test_data=NULL) {
  # response = "FI_q30"
  # model_object <- log
  #
  if(is.null(test_data)) test_data <- testing

  # make predictions
  prediction <- predict(model_object, test_data)

  target <- test_data[, response]

  cmat <- confusionMatrix(prediction, target, mode = "prec_recall")

  misscal<- round(mean(prediction != target),digits = 2)

 # Returned outputs
 return(list(
   accuracy = (1-misscal),
   mcr = misscal,
   sens = round(cmat$byClass[1],2),
   spec = round(cmat$byClass[2],2),
   fbeta = round(cmat$byClass[7],2)
 ))

}
```

```r
metric_log <- metrics(lasso, response = "FI_q30")
#------- Compute performance metrics for the full models ---------------
log.metric <- metrics(log, response = "FI_q30")
lda.metric <- metrics(lda, response = "FI_q30")
# knn.metric <- metrics(knn)
lasso.metric <- metrics(lasso, response = "FI_q30")
ridge.metric <- metrics(ridge, response = "FI_q30")
bag.metric <- metrics(bag, response = "FI_q30")
# rf.metric <- metrics(rf)
svc.metric <- metrics(svc, response = "FI_q30")
# svmP.metric <- metrics(svmP)
svmR.metric <- metrics(svmR, response = "FI_q30")

mod.sum <- data.frame(rbind(
                          c("Logistic", log.metric$mcr, log.metric$accuracy, log.metric$sens, log.metri
                          c("LDA",  lda.metric$mcr, lda.metric$accuracy, lda.metric$sens, lda.metric$spe
                          c("LASSO", lasso.metric$mcr, lasso.metric$accuracy, lasso.metric$sens, lasso.m
                          c("Ridge", ridge.metric$mcr, ridge.metric$accuracy, ridge.metric$sens, ridge.m
                          c("Bagging", bag.metric$mcr, bag.metric$accuracy, bag.metric$sens, bag.metric$
                          c("SVC",  svc.metric$mcr, svc.metric$accuracy, svc.metric$sens, svc.metric$spe
                          c("SVM (Radial Kernel)", svmR.metric$mcr, svmR.metric$accuracy, svmR.metric$se

names(mod.sum) <- c("Model", "Misclassification Rate", "Accuracy", "Sensitivity", "Specificity", "fbeta

kable(mod.sum, align = "lccccc", caption = "Table : Evaluation metrics for Housing Insecurity with Perma
  kable_paper("hover", full_width = F)%>%
       kable_styling(font_size = 12)
```

Table 1: Table : Evaluation metrics for Housing Insecurity with Permanent Address as a response

Model	Misclassification Rate	Accuracy	Sensitivity	Specificity	fbeta
Logistic	0.11	0.89	0.86	0.91	0.88
LDA	0.11	0.89	0.85	0.92	0.88
LASSO	0.11	0.89	0.88	0.9	0.89
Ridge	0.11	0.89	0.86	0.92	0.88
Bagging	0.05	0.95	0.96	0.93	0.95
SVC	0.11	0.89	0.88	0.9	0.89
SVM (Radial Kernel)	0.07	0.93	0.93	0.92	0.93

Base on our table of results SVM with radial basis function is the best

```
Var <- varImp(svmR, scale = FALSE)
plot(Var)
```