R_Recap_02

Felix Mueller

2/27/2017

Prepare your R session

```
setwd("~/Google_Drive/github/R/R_Recap/session_02")
library(dplyr)
library(ggplot2)
```

How to apply models in general

- Understanding the problem
- Understanding the data
- Preparing the data
- Split in Train & Test
- Predicting variable selection
- Resampling
- Model
- Tuning of the model
- Evaluation

=> ITERATIVE PROCESS! # Iris dataset {.smaller}

Loading the Iris dataset

```
glimpse(iris)
```

```
## Observations: 150
## Variables: 5
## $ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6,
## $ Sepal.Width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4,
## $ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4,
## $ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3,
## $ Species      <fctr> setosa, setosa, setosa, setosa, se
```

```
df = iris
df[,-5] = scale(df[,-5])
print(any(is.na(df)))
```

```
## [1] FALSE
```

Function: Split data

```
splitdf <- function(dataframe, seed=1,
                    percentage=0.8) {
  if (!is.null(seed)) set.seed(seed)
  index <- 1:nrow(dataframe)
  numTrainingSamples <- round(length(index)
                              * percentage)
  trainindex <- sample(index,
                       numTrainingSamples)
  trainset <- dataframe[trainindex, ]
  testset <- dataframe[-trainindex, ]
  list(trainset=trainset,testset=testset)}
```

resampling the iris data

```
df = df %>% filter(Species!= "setosa")
levels(df$Species) = list("virginica"="virginica",
      "versicolor"=c("versicolor","setosa"))
split <- splitdf(df,seed=1)
train <- split$trainset
test <- split$testset
library(ROSE)
print(sum(train$Species == "versicolor"))
```

```
## [1] 46
```

```
train <- ovun.sample(Species~.,data=train,
                     method="under",p=0.7)$data
print(sum(train$Species == "versicolor"))
```

```
## [1] 14
```

Building and tuning a tree

```
library(tree)
set.seed(37)
tree = tree(Species ~.,train)
tree_cv = cv.tree(tree,method="misclass")
print(tree_cv$size)
```

```
## [1] 3 2 1
```

```
print(tree_cv$dev)
```

```
## [1]  7  7 15
```

```
tree_pruned = prune.tree(tree,best=2)
```

Test with final test set

```
predictions = predict(tree,newdata=test)
predictions = as.factor(if_else(
  predictions[,2]>0.5,"virginica","versicolor"))
print(table(predictions,test$Species))
```

```
##
## predictions  virginica versicolor
##   versicolor         1          4
##   virginica         15          0
```

pruned prediction

```
predictions_pr = predict(tree_pruned,newdata=test)
predictions_pr = as.factor(
  if_else(predictions_pr[,2]>0.5,
          "virginica","versicolor"))
print(table(predictions_pr,test$Species))
```

```
##
## predictions_pr virginica versicolor
##     versicolor         1          4
##     virginica         15          0
```

Marketing Intelligence

```
#not in validation set --> exclude
df = df %>% dplyr::select(-iab_4)
names <- c('device','os','os_version','browser'
           ,'browser_version','prov','day','target')
df[,names] <- lapply(df[,names], factor)
```

Filtering

```
cols = sapply(df,
              function(x){is.integer(x)|is.numeric(x)})
cor_matrix = cor(df[,cols],as.numeric(df$target)
                 ,method="spearman")
#REALLY low threshold!
names_num <- rownames(subset(cor_matrix
                             ,abs(cor_matrix) >= 0.01))
df <- df %>% dplyr::select(one_of(names_num,names))
#Random Forest cannot handly NAs
df$prov <- NULL
```

Resample

```
splits = splitdf(df,percentage=0.9)
test = splits$testset
train = splits$trainset
train <- ovun.sample(formula=target~.,data=train
                     ,method="under", p=0.5)$data
rm(df)
rm(splits)
```

Random forest

```
library(randomForest)
model <- randomForest(target~., data=train,
                      importance=TRUE)
probs <- predict(model, type="prob", newdata = test)
predictions <- data.frame(target=test$target
                          , pred=probs[,2])
```

Threshold detection

```
library(pROC)
getThreshold <- function(predictions){
  myROC <- roc(formula = target ~ pred, predictions)
  optimalThreshold <- coords(myROC, "best"
                             , ret = "threshold")
}
threshold <- getThreshold(predictions)
tbl = table(predictions$target,
            predictions$pred > threshold)
```

Compare results

```
print(tbl)
```

```
##
##       FALSE   TRUE
##    0 133536  55952
##    1    123    235
```

```
auc = auc(predictions$target, predictions$pred)
accuracy <- (tbl[1,1] + tbl[2,2]) / sum(tbl)
F1 <- (2*(tbl[1,1]))/((2*(tbl[1,1]))+tbl[2,1]+tbl[1,2])
print(cat("Accuracy:",accuracy,"F1:",F1,"AUC:",auc,""))
```

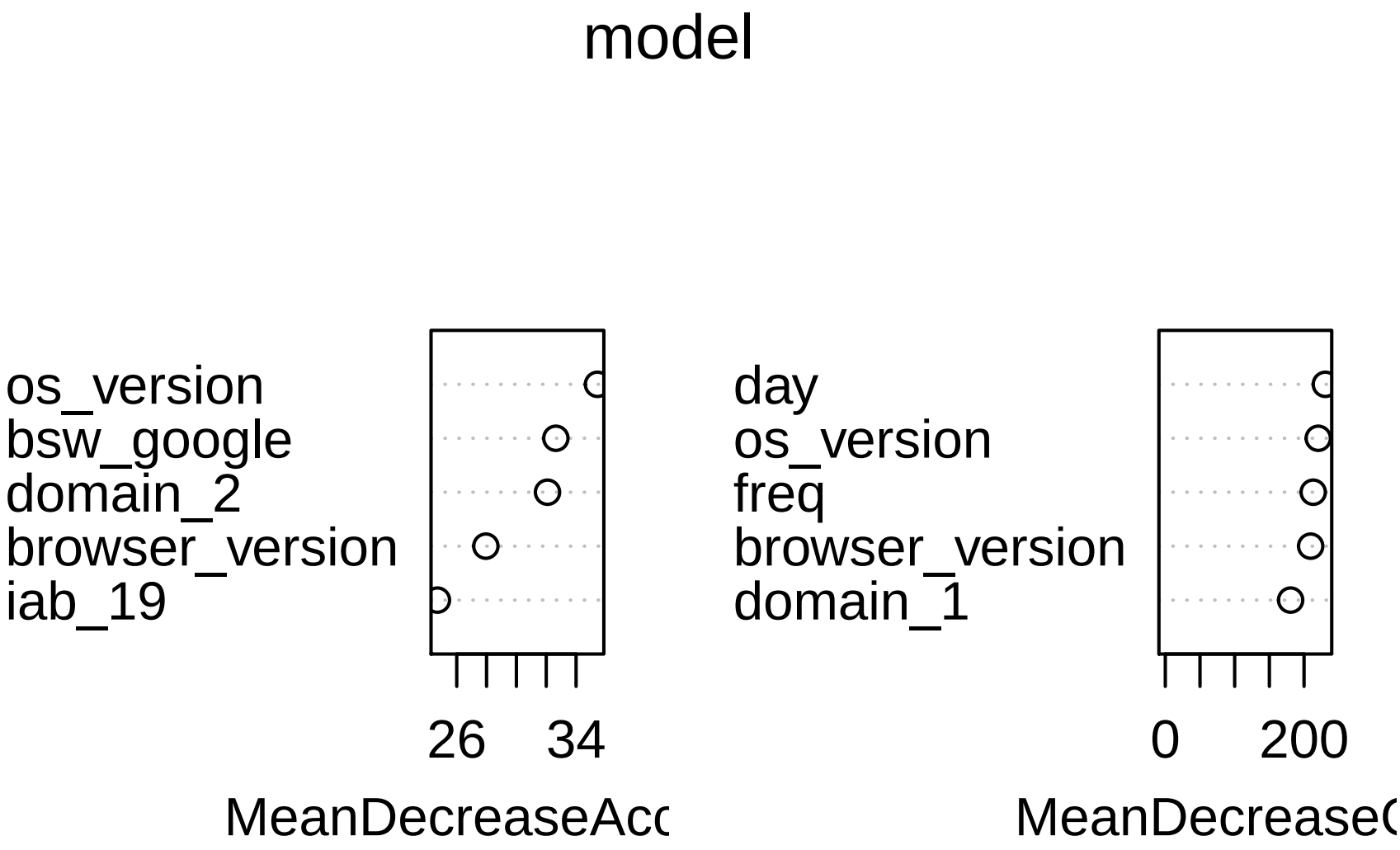
```
## Accuracy: 0.704629 F1: 0.8264722 AUC: 0.7253162 NULL
```

Variable importance

```
#high values wanted
print(head(importance(model)[,3:4]))
```

```
##                   MeanDecreaseAccuracy MeanDecreaseGini
## freq                           17.55651         213.18706
## adnexus                        22.43981         112.44327
## bsw_rubicon                    24.14906         150.20596
## bsw_google                     32.64226         152.25962
## bsw_adconductor                13.30819          77.42707
## bsw_pubmatic                   13.85020          59.55289
```

VarImpPlot


model
os_version
bsw_google
domain_2
browser_version
iab_19
26
34
MeanDecreaseAcc
day
os_version
freq
browser_version
domain_1
0
200
MeanDecreaseG

K-Fold CV

```
set.seed(1)
folds <- sample(rep(1:10,length=nrow(train)))
cv_error <- vector()
sequence = seq(1,10)
for(k in sequence){
  model <- randomForest(target~.,data=train[folds!=k,])
  pred <- predict(model,train[folds==k,],type="class")
  cv_error[k] <- mean(train[folds==k,]$target==pred)
}
```

ggplot2

```
q1 = qplot(y=cv_error,x=sequence,xlab="Iteration",
      ylab="CV Error",main="CV Error per iteration")+
  geom_line(colour=I("blue"))+
  geom_point(aes(x = which(cv_error==min(cv_error)),
                 y = min(cv_error)), color = I("red"))+
  geom_hline(aes(yintercept=max(cv_error)),
             color = I("red"),linetype = 2)
```

ggplot2 output

Other ideas to play with

1. mtry (default = sqrt(p)): number of variables per tree
2. ntree: number of trees