Data Science

R 강의: 예측 모델의 최적화
caret package 를 이용한 parameter tuning

예측 모델에 영향을 주는 요인들

- A prediction model is the function of
  - Training dataset
  - Set of prediction variables
  - Parameters of the machine learning algorithm
  - 모델의 성능은 위의 요인들에 따라 좌우됨.
- Optimization of model – 모델 최적화
  - prediction model 이 실제 예측할 data 에 대하여 최고의 성능(정확도)을 발휘 하도록 요인들을 설정할 필요
  - 그러나, 실제 예측할 dataset 은 unknown 이므로 주어진 데이터 셋(표본)을 사용하여 모델 생성 및 검증
  - 통계적, 확률적인 접근 방법을 사용
  - 예측변수 선택 – Selection of prediction variables
  - 파라미터 튜닝 – Parameter tuning

Training dataset

Training dataset

- 실제 데이터를 반영할 수 있도록
  실제 데이터와 동일한 분포를 가지도록 설정
- 다다익선 – 클수록 좋다.
- 데이터 수집의 문제, 비용 등 – 충분한 양을 가지기 어렵다.
- 크기가 너무 작거나 분포가 편중된 경우 overfitting 의 우려

Cross-validation – 교차 검증

- training, testing dataset 부족에 문제 해결
- 모델 성능 측정을 정확하게 할 목적
- 방법: 다양한 데이터셋을 준비 – 복수의 모델 생성 및 평균적인 성능 측정
- 성능을 측정할 요인(변수집합, 파라미터) 외에
  다른 요인을 randomize(무작위화)하여
  평균적인 성능을 측정하고자 함
- 이외에도 다른 resampling 방법 – repeated holdout, bootstrapping 을 사용

모델의 최적화

- 모델이 최고의 성능을 가지도록
  - 변수집합, 파라미터 값들의 조합을 결정
  - 조합 검색의 문제
- 전역 검색
  - 요인들의 선택지 및 값 의 모든 조합에 대하여
  - 모델 생성 및 성능 측정하고
  - 최적의 값을 선택
  - 비효율적이나 계산가능한 경우 전역 최적값을 구할 수 있음.
- Heuristic
  - R에서의 대부분의 모델 생성 함수 에서는 parameter 들에 대하여 기본값을 자동으로 설정
- 고차원 적인 검색 방법
  - 수학적, 알고리즘에 의한 최적값 검색

Model training and tuning using caret package

파라미터 최적화를 자동화함

- 각 파라미터 조합에 대하여
  resampling 방식으로 복수의 model 생성 및 평균적인 성능 측정

```
Define sets of model parameter values to evaluate
for each parameter set do
    for each resampling iteration do
        Hold–out specific samples
        [Optional] Pre–process the data
        Fit the model on the remainder
        Predict the hold–out samples
    end
    Calculate the average performance across hold–out predictions
end
Determine the optimal parameter set
Fit the final model to all the training data using the optimal parameter set
```

← resampling (CV, repeated holdout, bootstrapping, etc)으로 tr, ts set 을 구분하여 모델 생성 및 성능시험

→https://topepo.github.io/caret/model-training-and-tuning.html#basic

Note: caret package 를 사용하지 않더라도 위의 절차로 파라미터 최적화 실행

Model training and tuning using caret package - Example

예제 → https://csantill.github.io/RTuningModelParameters/

- caret 이 지원하는 model 확인

  → rpart 가 아닌 rpart2

```
names(getModelInfo())
```

```
## [178] "rotationForestCp"    "rpart"               "rpart1SE"
## [181] "rpart2"              "rpartCost"           "rpartScore"
```

- rpart2 에서 최적화 할 수 있는 parameter 확인

  → maxdepth

```
modelLookup("rpart2")
```

```
##    model parameter          label forReg forClass probModel
## 1 rpart2  maxdepth Max Tree Depth   TRUE     TRUE      TRUE
```

Model training and tuning using caret package - Example

- train.control()
  - resampling, 검색 방법 설정
- train()
  - training 알고리즘 지정
  - resampling(bootstrap, CV) 을 기반으로 각 파라미터에 대한 성능 평가
  - optimal parameter 와 model 선택
  - training set 에 대하여 모델의 성능을 평가

Model training and tuning using caret package - Example

train.control

```
train.control <- trainControl(
                          method = "repeatedcv",
                          number = 10, ## 10-fold CV
                          repeats = 3,## repeated three times
                          # USE AUC
                          summaryFunction = twoClassSummary,
                          classProbs = TRUE
                          )
```

- resampling 은 CV,
- classProbs(class 예측 확률) 계산 을 선택

train

```
reset.seed()
system.time (rpartFit1 <- train(diabetes~., data=train.data,
                 method = "rpart2",
                 tuneLength = 6,
                 trControl = train.control,
                 metric = "ROC"
                ))
```

- method = learning algorithm
- tuneLength = 파라미터 조합의 수
- 최적화 metric = "ROC", "Accuracy", "Kappa" 등에서 선택

Model training and tuning using caret package - Example

- maxdepth 에 대하여 성능 측정 – maxdepth = 13 인 경우 ROC 가 최대

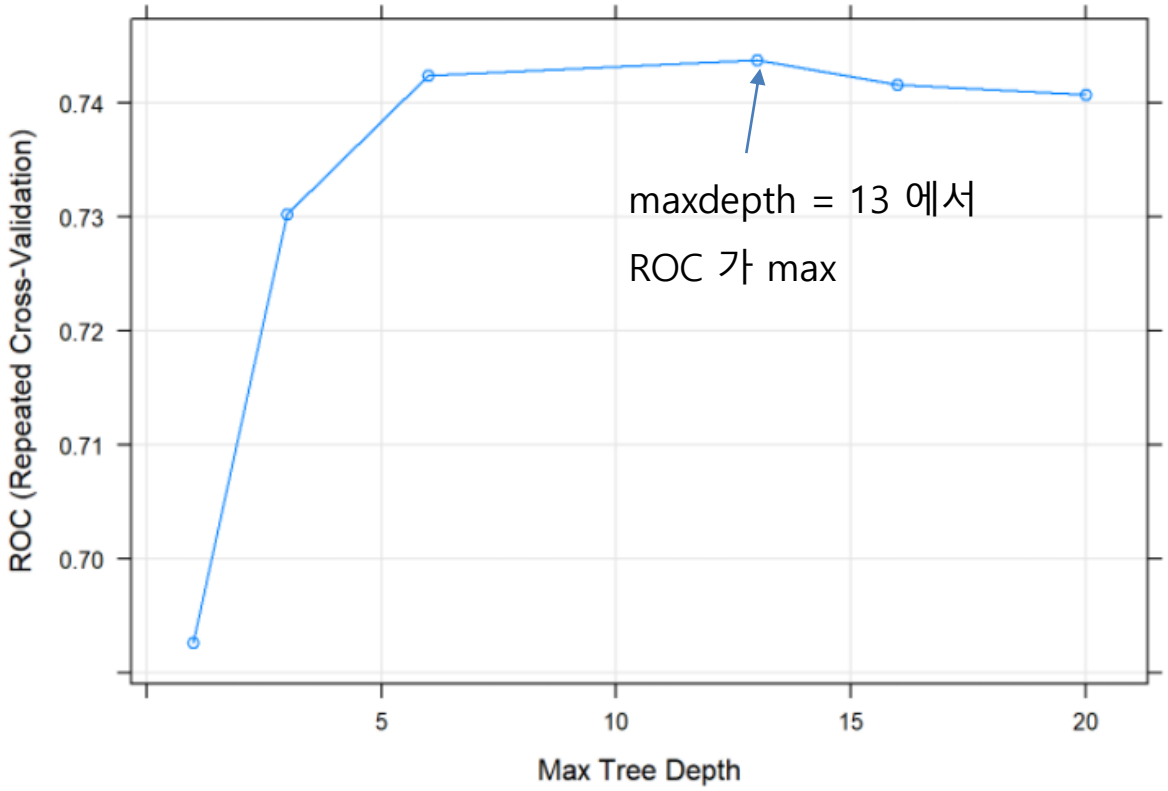
```
rpartFit1
```

```
## CART
##
## 538 samples
##   8 predictor
##   2 classes: 'neg', 'pos'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 485, 484, 484, 485, 484, 484, ...
## Resampling results across tuning parameters:
##
##   maxdepth  ROC        Sens       Spec
##    1        0.6926344  0.7714286  0.6138402
##    3        0.7302130  0.8352381  0.5603314
##    6        0.7423726  0.8390476  0.5498051
##   13        0.7437650  0.8285714  0.5552632
##   16        0.7416054  0.8266667  0.5551657
##   20        0.7407324  0.8266667  0.5533138
##
## ROC was used to select the optimal model using  the largest value.
## The final value used for the model was maxdepth = 13.
```

Model training and tuning using caret package - Example

- plot parameter vs ROC

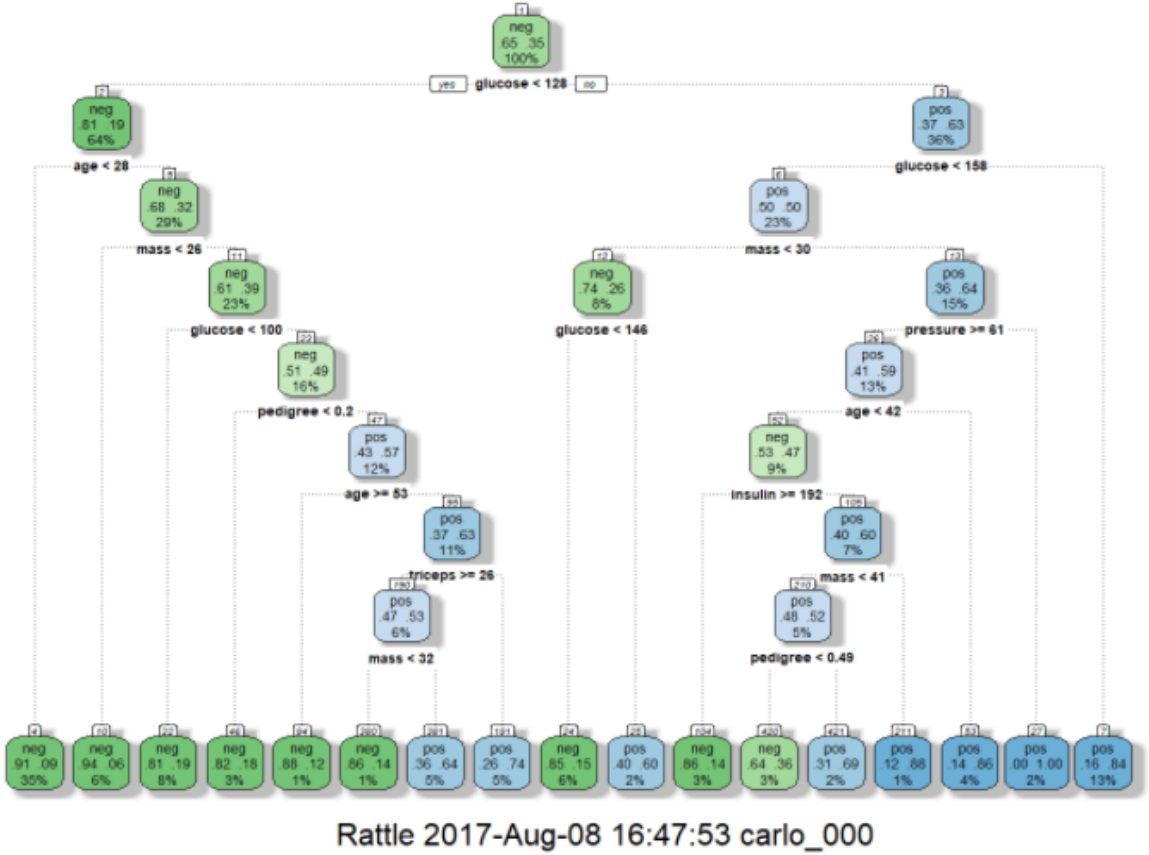
```
plot(rpartFit1)
```



Model training and tuning using caret package - Example

- Final model

```
fancyRpartPlot(rpartFit1$finalModel)
```



Model training and tuning using caret package - Example

Grid search

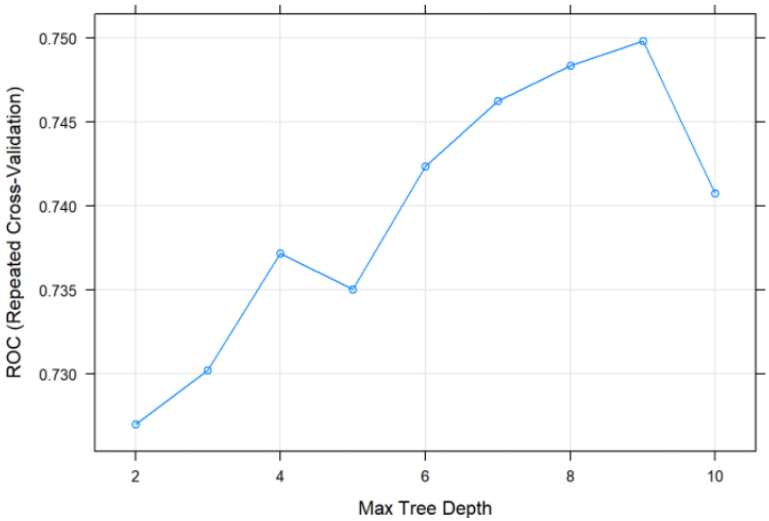
- 최적화할 parameter에 대하여 시험해 볼 값을 을 상세하게 수동으로 설정 가능

```
reset.seed()

tune.gridcart <- expand.grid(maxdepth = 2:10)

system.time (rpartFit2 <- train(diabetes~., data=train.data,
                  method = "rpart2",
                  tuneGrid =tune.gridcart,
                  trControl = train.control,
                  metric = "ROC"
                ))
```

```
plot(rpartFit2)
```



예제

- parameter tuning – rpart, caret
  - https://csantill.github.io/RTuningModelParameters/
  - https://rstudio-pubs-static.s3.amazonaws.com/246827_b9a4314244084f00ad5139144a1997d8.html
  - https://lovetoken.github.io/r/machinelearning/2017/04/23/caret_package.html