임헌정
2017년 5월 19일

SK Machine Learning Course

1. Random Forest

   1.1. Source : 별도 첨부

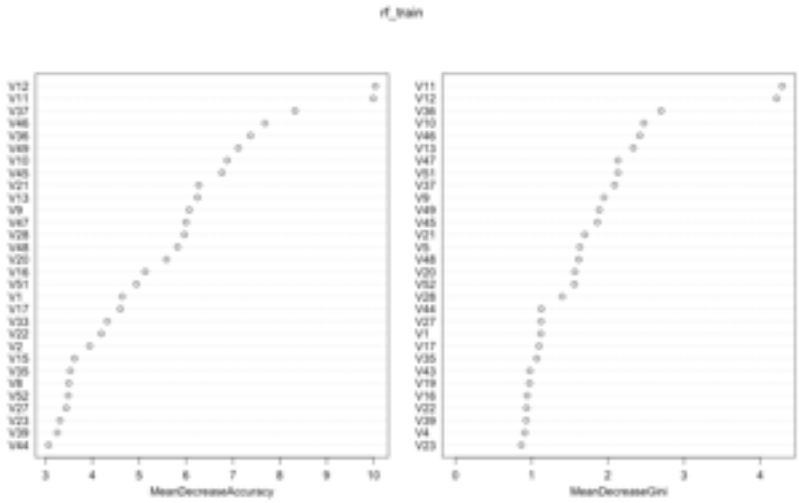
   1.2. Training

```
Call:
 randomForest(formula = V61 ~ ., data = train.sonar, mtry = 8,      importance = T)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 8

        OOB estimate of  error rate: 13.79%
Confusion matrix:
   M  R class.error
M 57 12   0.1739130
R  8 68   0.1052632
```

   1.3. 변수 중요도

      1.3.1. V11,V12 변수가 중요한 변수로 알려짐



   1.4. Test

      1.4.1. 약 83%의 정확성을 보임

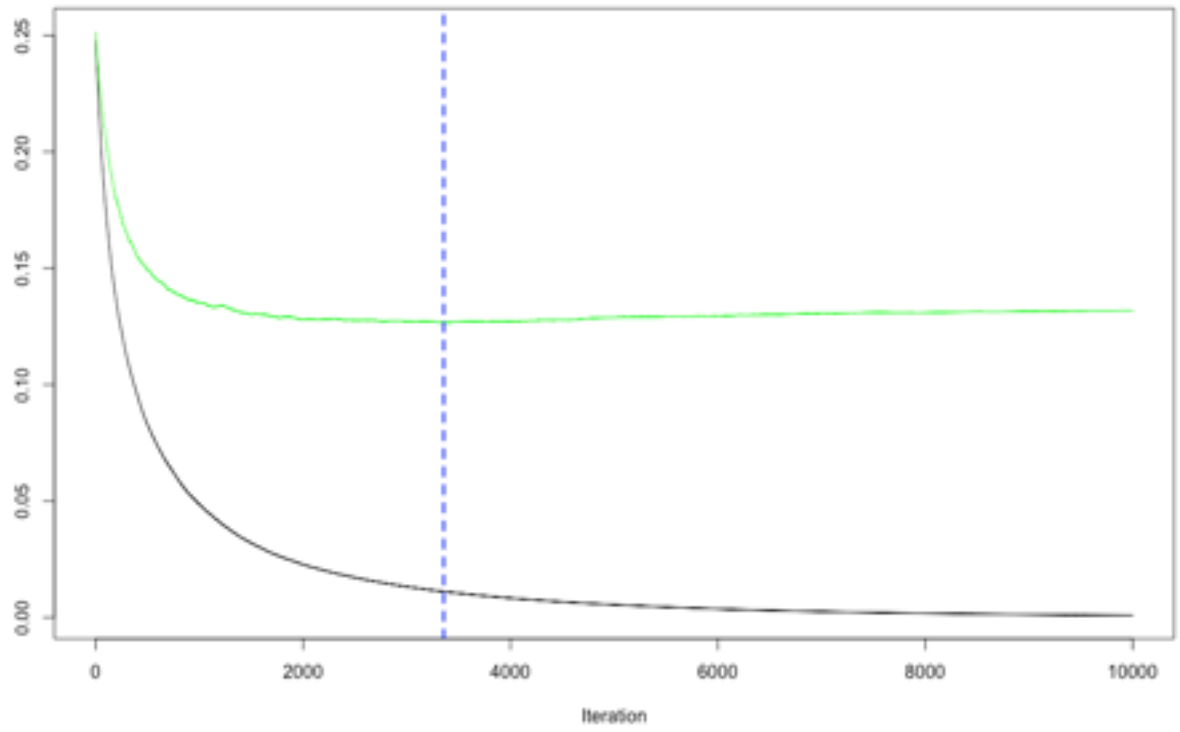
```
> err_rate
          test err
RF (m=8) 0.1746032
```

2. Boosting

2.1. Source : 별도 첨부

2.2. Training

2.2.1. Best Iteration



2.3. Test

2.3.1. 100% 확률로 분류

```
> pred.gbm
 [1] 1.4822471 1.0362588 2.0043180 1.9717670 1.0388258 1.8273315 1.3533269 1.0927541 1.1463959 1.2417211 1.2879686 1.0394538 1.9788242 1.1286568 0.9306855 1.1706131 1.0259688 1.3879583
[19] 0.8818339 1.0384837 0.9782552 1.0274728 0.9235187 0.7931878 1.3782676 1.0258385 1.2813419 1.7403353 1.3362361 1.9751183 1.9989686 1.5146415 1.7543161 1.7988094 1.6258841 1.7514423
[37] 1.0282225 1.9642556 1.8751520 1.9866832 1.7358516 1.9936185 1.9388998 1.4357415 1.6684357 1.4587499 1.6367015 1.0282706 1.4195896 1.0748847 1.8697519 1.0530234 1.6773479 1.9137541
[55] 2.1175523 2.2748841 1.9641576 1.9577667 1.9932306 2.1152853 2.0388133 1.0224754 1.6168715
> table(obs = test.sonar$V61,pred= ifelse(pred.gbm > 0.5,1,0))
   pred
obs  1
  M 28
  R 35
> 1- sum(ifelse(ifelse(pred.gbm>0.5,1,0)==test.sonar$V61,1,0))
[1] 1
```

3. SVM

3.1. Source : 별도 첨부

3.2. Training

```
Call:
svm(formula = V61 ~ ., data = train.sonar)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.01666667

Number of Support Vectors:  117

 ( 56 61 )


Number of Classes:  2

Levels:
 M R
```

3.3. Test

3.3.1. 약 83%의 정확성을 보임

```
> table(obs = test.sonar$V61,pred=pred.svm)
   pred
obs  M  R
  M 19  9
  R  2 33
> length(which(pred.svm != test.sonar$V61))/nrow(test.sonar)
[1] 0.1746032
```

4. DNN

4.1. Source

4.2. Train

4.2.1. Input & Hidden Layer

4.2.1.1. Activation Function : Relu

4.2.2. Output Layer

4.2.2.1. Activation Function : softmax

4.2.3.Complie

4.2.3.1. Loss Function : use

4.2.3.2. Optimizer : adam

4.2.3.3. metric : accuracy

4.2.4. Fitting

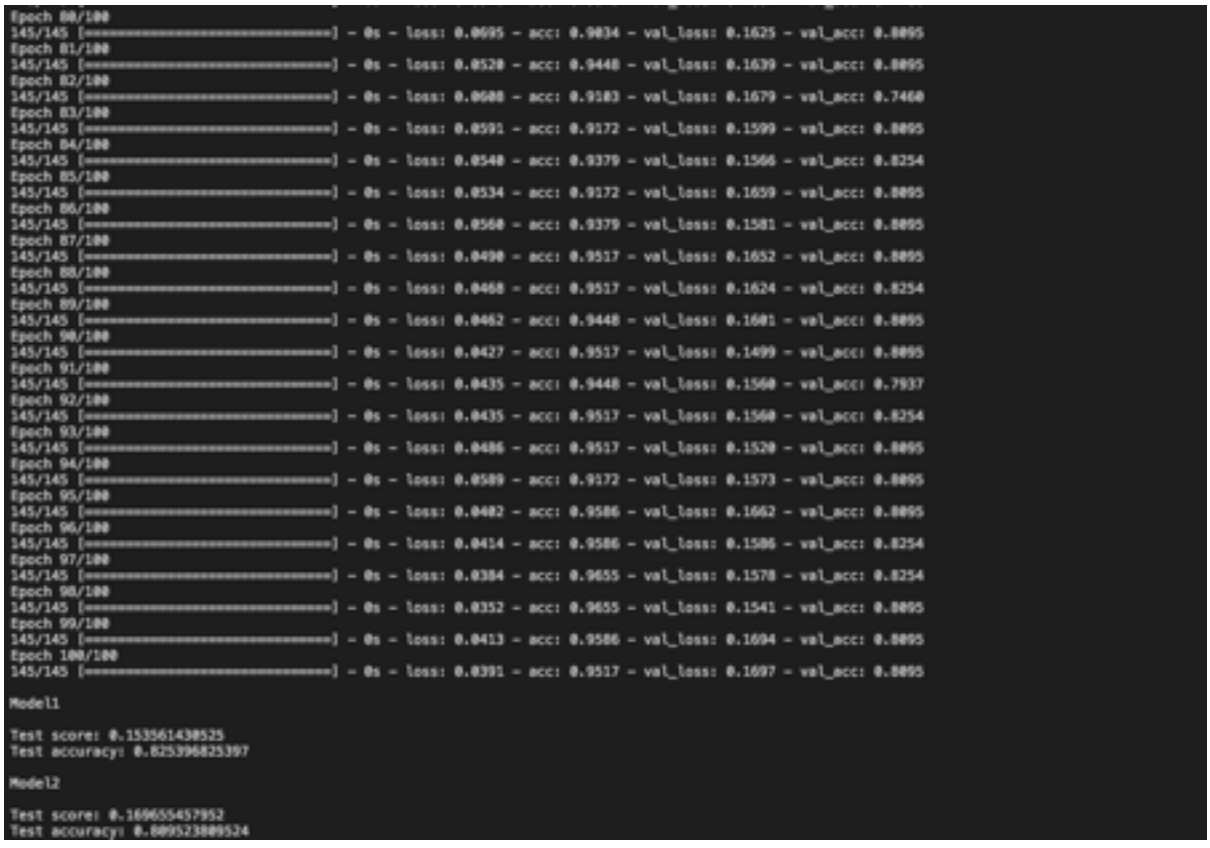
4.2.4.1. Epoch : 100

4.2.4.2. Batch Size : 10

4.3. Test

4.3.1. HiddenLayer(10*10) : 약 82%의 정확성을 보임

4.3.2. HiddenLayer(20*20) : 약 80%의 정확성을 보임



5. LASSO

5.1 Source : 별도 첨부

5.2 Training

5.2.1. best lambda search : lam =cv.out$lambda.min lam = 0.0198393

5.3 Test

5.3.1 : 약 76%의 정확도를 보임.

```
> table(obs = test.sonar$V61,pred= ifelse(pred_lasso ==1,"M","R"))
   pred
obs  M  R
  M 17 11
  R  4 31
>
> 1- sum(ifelse(ifelse(pred_lasso==1,"R","M")==test.sonar$V61,1,0)) /nrow(test.sonar)
[1] 0.7619048
>
```

* 종합 결과

  * 해당 데이터는 Boosting 알고리즘이 정확도가 가장 높다.

모델	정확도
Random Forest	83%
Boosting	**100%**
SVM	83%
DNN (10*10)	82%
DNN (20*20)	80%
LASSO	76%