6장* {caret} 패키지 소개

충북대학교 정보통계학과 나 종 화
(cherin@cbnu.ac.kr)

CONTENTS

6.1 서론

6.2 사용 절차: PLS 회귀 예제

6.3 {caret}의 적용 범위(모형)

6.4 {caret}을 이용한 변수선택

6.4.1 변수의 중요도

6.4.2 변수 선택

6.1 서론

- R의 {caret} 패키지는, “classification and regression training”의 약어로, 복잡한 회귀와 분류 문제에 대한 모형 훈련(training)과 조절(tuning) 과정을 간소화하는 함수를 포함한다.
- 이 패키지는 훈련 데이터의 전처리, 변수의 중요성 계산 및 모형 시각화를 위한 방법을 포함한다. 또한, 이 패키지는 많은 R 패키지를 사용하지만 패키지 시작 시에 그들을 로드하지 않는다.
- 이 패키지의 “suggest" 필드에는 28개의 패키지가 들어 있다. {caret}은 필요에 따라 패키지를 로드하고 패키지가 설치되어 있다고 가정한다.

6.2 사용 절차: PLS 회귀 예제

- {caret}은 변수선택과 기타 기능 뿐 아니라 모형 개발과 평가 과정을 간소화하기 위한 몇 가지 함수를 가진다.
- 패키지의 기본 도구중 하나는 train() 함수로 다음의 기능을 수행한다.
  - 재표본(resampling)을 사용하여 모형의 조절모수가 성능에 미치는 영향을 평가하고,
  - 이들 모수에서 "최적"의 모형을 선택하고,
  - 훈련 집합에서 모형 성능을 추정한다.

6.2 사용 절차: PLS 회귀 예제

- 이 과정을 보다 자세히 소개하면 [표 6.1]과 같다.

[표 6.1] train() 함수의 수행 절차

1	평가할 모형의 모수 집합을 정의
2	for 각 모수 집합에 대해 수행 do
3	for 각 재표본 반복에 대해 수행 do
4	· 특정 표본을 예비표본으로 남김
5	· [선택적] 데이터에 대한 전처리 수행
6	· 남겨진 표본으로 모형 적합
7	· 예비표본에 대해 예측 수행
8	end
9	예비표본 예측에 대한 평균 성능 계산
10	end
11	최적의 모수집합 결정
12	최적의 모수집합을 사용하여 모든 훈련 자료에 대한 최종모형을 적합

6.2 사용 절차: PLS 회귀 예제

- 이 프로세스의 거의 모든 단계를 맞춤 설정할 수 있는 옵션이 있다(예: 재표본 기법, 최적의 모수 선택 등).
- 이 기능을 설명하기 위해 수중 음파탐지기로부터의 신호자료인 Sonar{mlbench} 자료를 사용한다. 이 자료는 60개의 예측변수에서 수집된 208개의 자료 점으로 구성되었다. 분석의 목적은 Class 변수(M: 금속 실린더, R: 바위)를 예측하는 것이다.
- 먼저, 데이터를 두 개의 그룹으로 나눈다. 즉, 훈련용 자료 집합(training set)과 검증용 자료 집합(test set)으로 나누며, 이를 위해 createDataPartition() 함수를 이용한다.

6.2 사용 절차: PLS 회귀 예제

```
> library(caret)
> library(mlbench)
> data(Sonar)
> set.seed(107)
> inTrain <- createDataPartition(y = Sonar$Class,    # 반응치 자료가 필요
      p = .75,    # 훈련용 데이터의 비율
      list = FALSE)    # 결과의 형식 지정
```

6.2 사용 절차: PLS 회귀 예제

```
> ## 훈련용 자료에 포함될 자료의 행 번호가 출력됨
> str(inTrain)
int [1:157, 1] 1 2 3 6 7 9 10 11 12 13 ...
- attr(*, "dimnames")=List of 2
..$ : NULL
..$ : chr "Resample1"
```

해 석 디폴트로 creatDataPartition() 함수는 층화 확률분할을 실시한다.

6.2 사용 절차: PLS 회귀 예제

```
> # 훈련용과 검증용 자료를 저장
> training <- Sonar[ inTrain,]
> testing <- Sonar[-inTrain,]
> nrow(training)
 [1] 157
> nrow(testing)
 [1] 51
```

6.2 사용 절차: PLS 회귀 예제

- 알고리즘을 사용하여 모형을 조절하기 위해 train() 함수를 사용한다. 부분최소제곱 판별분석(PLSDA) 모형은 유지되어야 하는 PLS 구성요소의 수에 대해 조절된다. 가장 기본적인 구문은 다음과 같다:

```
> plsFit <- train(Class ~ .,
                  data = training,
                  method = "pls",
                  preProc = c("center", "scale"))
> # 훈련용 자료와 모든 미래 표본의 예측변수에 대해 중심화(center)와 척도화(scale)를 수행
```

6.2 사용 절차: PLS 회귀 예제

- 그러나 다음과 같이 몇 가지 방법으로 사용자 주문이 가능하다.
  - 함수가 평가하는 PLS 모형의 집합을 확장한다. 기본적으로 이 함수는 각 조절 모수의 세 가지 값을 조절한다.
  - 사용되는 재표본의 유형. 디폴트로 단순 붓스트랩이 사용된다. 우리는 이 함수에 10-중첩 교차타당 검증의 세 번 반복을 사용할 것이다.
  - 성능의 측정 방법. 디폴트로 전체 정확도와 카파 통계량(Kappa statistic)이 계산된다. 회귀 모델의 경우 평균제곱오차와 R^2가 계산된다. 여기에서 ROC 곡선 아래의 면적, 민감도 및 특이성을 추정하기 위해 함수가 변경된다.

6.2 사용 절차: PLS 회귀 예제

- 조율모수의 후보 값을 변경하려면 tuneLength= 또는 tuneGrid= 옵션 중 하나를 사용할 수 있다.
- train() 함수는 조율모수의 후보 집합을 생성 할 수 있고 tuneLength= 옵션은 평가되는 수를 제어한다. PLS의 경우 이 함수는 1에서 tuneLength 까지의 정수 수열을 사용한다. 1에서 15 사이의 모든 정수를 평가하려면 tuneLength=15로 설정하면 된다.
- tuneGrid= 옵션은 특정 값이 필요할 때 사용한다. 데이터 프레임은 각 행이 조율모수 설정이고 각 열이 조율모수인 경우에 사용된다. 이를 설명하기 위한 예제는 다음과 같다.

6.2 사용 절차: PLS 회귀 예제

```
> ctrl <- trainControl(method = "repeatedcv", repeats = 3)
> plsFit <- train(Class ~ ., data = training,
                  method = "pls",
                  tuneLength = 15,
                  trControl = ctrl,
                  preProc = c("center", "scale"))
```

6.2 사용 절차: PLS 회귀 예제

- 마지막으로, 다른 성능 평가 측도를 선택하려면 trainControl() 함수에 추가 옵션을 사용한다. summaryFunction= 옵션은 관측값과 예측값을 취하여 성능 측도를 추정하는 함수에서 사용된다.
- 두 개의 함수 defaultSummary와 twoClassSummary가 이미 패키지에 포함되어 있다. 후자는 ROC 곡선 아래의 면적, 민감도 및 특이도와 같은 2-집단(class) 문제에 대한 측도를 계산한다.
- ROC 곡선은 예측된 클래스 확률(자동으로 계산되지 않음)을 기반으로 하므로 다른 옵션이 필요하다. classProbs=TRUE 옵션은 이러한 계산을 포함하는 데 사용된다.

6.2 사용 절차: PLS 회귀 예제

- 끝으로, 이 함수는 최상의 결과를 가지는 조절모수를 선택한다. 맞춤형 성능 측도를 사용하고 있으므로 최적화해야 하는 기준도 지정해야 한다. train을 호출 할 때, metric="ROC"를 사용하여 이를 수행 할 수 있다.

```
> set.seed(123)
> ctrl <- trainControl(method = "repeatedcv",
                       repeats = 3,
                       classProbs = TRUE,
                       summaryFunction = twoClassSummary)
```

6.2 사용 절차: PLS 회귀 예제

```
> plsFit <- train(Class ~ .,
                  data = training,
                  method = "pls",
                  tuneLength = 15,
                  trControl = ctrl,
                  metric = "ROC",
                  preProc = c("center", "scale"))
> #  metric= 옵션에는 "accuracy", "Kappa", "RMSE", "Rsquared"기 있다. 이 옵션은 최종모형의 선택에 사용되는 목적함수를 지정함
```

6.2 사용 절차: PLS 회귀 예제

```
> plsFit
 Partial Least Squares
       157 samples
       60 predictor
       2 classes: 'M', 'R'

Pre-processing: centered (60), scaled (60)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 142, 141, 141, 141, 142, 142, ...
Resampling results across tuning parameters:
       ncomp   ROC            Sens           Spec
       1       0.8135582      0.7120370      0.7172619
       2       0.8713211      0.7782407      0.8130952
       3       0.8660962      0.7699074      0.8339286
       …
       15      0.8150463      0.7337963      0.7744048
ROC was used to select the optimal model using the largest value.
The final value used for the model was ncomp = 2.
```

6.2 사용 절차: PLS 회귀 예제

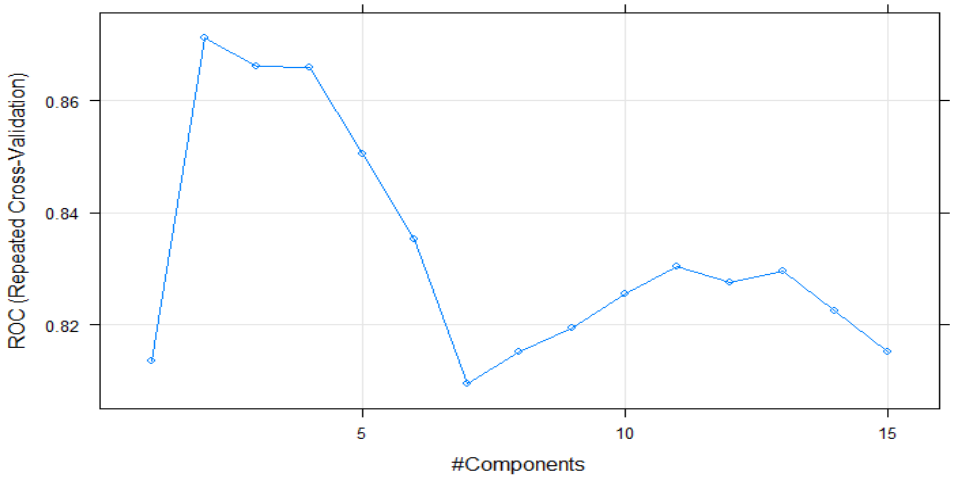
해 석 이 출력에서 격자 결과는 성능의 평균 재표본된 추정치이다. 하단의 메모는 ROC가 가장 큰 값이 최적 모형의 선택에 사용되었음을 말하며, 그 모형의 최종값은 ncomp=2 이다.

이 값을 바탕으로 최종 PLS 모형은 이 지정을 사용하는 전체 자료에 대한 것이고, 이것은 미래의 자료를 예측하는데 사용 된다.

6.2 사용 절차: PLS 회귀 예제

- 이 패키지는 결과를 시각화하는 몇 가지 함수가 있다. 이 작업을 수행하는 한 가지 방법은 train 객체에 대해 plot() 함수를 수행하는 것이다.

```
> plot(plsFit)
```



해 석 PLS 성분의 수와 재표본된 AUC(area under the ROC curve) 추정값의 관계를 보여준다.

6.2 사용 절차: PLS 회귀 예제

- 새로운 표본의 예측에는 predict.train이 사용될 수 있다. 분류모형의 경우, 기본 동작은 예측된 집단을 계산하는 것이다. type="prob"옵션을 사용하여 모형으로부터 집단 확률을 계산할 수 있다.

```
> plsClasses <- predict(plsFit, newdata = testing)
> str(plsClasses)
 Factor w/ 2 levels "M","R": 2 1 1 2 1 2 2 2 2 2 ...
> plsProbs <- predict(plsFit, newdata = testing, type = "prob")
> head(plsProbs)
          M         R
 4 0.3762529 0.6237471
 5 0.5229047 0.4770953
 8 0.5839468 0.4160532
16 0.3660142 0.6339858
20 0.7351013 0.2648987
25 0.2135788 0.7864212
```

6.2 사용 절차: PLS 회귀 예제

- {caret}은 모형 적합에 대한 정오분류행렬(confusion matrix)과 관련 통계를 계산하는 함수를 포함한다.

```
> confusionMatrix(data = plsClasses, testing$Class)    # 정오분류행렬
 Confusion Matrix and Statistics
           Reference
 Prediction  M      R
       M    20      7
       R     7     17

              Accuracy : 0.7255
                95% CI : (0.5826, 0.8411)
   No Information Rate : 0.5294
   P-Value [Acc > NIR] : 0.003347
                 Kappa : 0.4491
Mcnemar's Test P-Value : 1.000000
                                  (…)
```

6.2 사용 절차: PLS 회귀 예제

```
                                          (…)
                     Sensitivity : 0.7407
                     Specificity : 0.7083
                  Pos Pred Value : 0.7407
                  Neg Pred Value : 0.7083
                     Prevalence  : 0.5294
                  Detection Rate : 0.3922
            Detection Prevalence : 0.5294
               Balanced Accuracy : 0.7245
                'Positive' Class : M
```

6.2 사용 절차: PLS 회귀 예제

- 자료에 대해 다른 모형을 적합하기 위해, 최소한의 변경만으로 train() 함수를 호출할 수 있다.

  예를 들어, 이 자료에 일반화(regularized) 판별모형을 적합하는 예는 다음과 같다.

```
> ##  설명을 위해 custom grid가 사용됨
> rdaGrid = data.frame(gamma = (0:4)/4, lambda = 3/4)
> set.seed(123)
> rdaFit <- train(Class ~ .,
                  data = training,
                  method = "rda",
                  tuneGrid = rdaGrid,     #  조율모수의 특정 격자(grid)를 지정
                  trControl = ctrl,
                  metric = "ROC")
```

6.2 사용 절차: PLS 회귀 예제

```
> rdaFit
Regularized Discriminant Analysis

157 samples
 60 predictor
  2 classes: 'M', 'R'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 142, 141, 141, 141, 142, 142, ...
Resampling results across tuning parameters:

  gamma  ROC        Sens       Spec
  0.00   0.8448826  0.7884259  0.7625000
  0.25   0.8860119  0.8060185  0.8035714
  0.50   0.8851190  0.8097222  0.7666667
  0.75   0.8685847  0.7745370  0.7529762
  1.00   0.7563823  0.6615741  0.6803571

Tuning parameter 'lambda' was held constant at a value of 0.75
ROC was used to select the optimal model using  the largest value.
The final values used for the model were gamma = 0.25 and lambda = 0.75.
```

6.2 사용 절차: PLS 회귀 예제

```
> rdaClasses <- predict(rdaFit, newdata = testing)
> confusionMatrix(rdaClasses, testing$Class)
Confusion Matrix and Statistics

          Reference
Prediction  M  R
         M 22  5
         R  5 19

               Accuracy : 0.8039
                 95% CI : (0.6688, 0.9018)
    No Information Rate : 0.5294
    P-Value [Acc > NIR] : 4.341e-05

                  Kappa : 0.6065
 Mcnemar's Test P-Value : 1
                                  (…)
```

6.2 사용 절차: PLS 회귀 예제

```
                                    (…)
            Sensitivity : 0.8148
            Specificity : 0.7917
         Pos Pred Value : 0.8148
         Neg Pred Value : 0.7917
             Prevalence : 0.5294
         Detection Rate : 0.4314
   Detection Prevalence : 0.5294
      Balanced Accuracy : 0.8032

       'Positive' Class : M
```

6.2 사용 절차: PLS 회귀 예제

- 이 모형들은 재표본결과와 어떻게 비교되는가? 재표본함수는 재표본결과를 수집, 요약 및 대비(contrast)하는 데 사용될 수 있다. train() 함수의 호출에 앞서 난수가 동일한 값으로 초기화 된 경우, 각 모형에 동일한 중첩(folds)이 사용된다

```
> resamps <- resamples(list(pls = plsFit, rda = rdaFit))
> summary(resamps)

Call:
summary.resamples(object = resamps)

Models: pls, rda
Number of resamples: 30

                                    (…)
```

6.2 사용 절차: PLS 회귀 예제

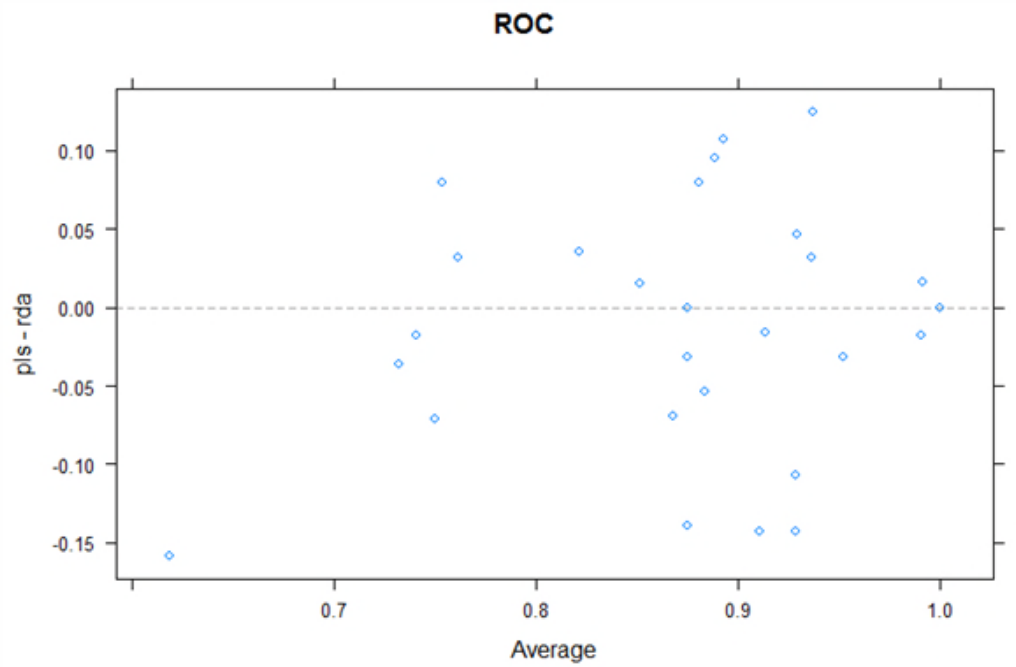
```
                                  (…)
ROC
      Min. 1st Qu. Median   Mean 3rd Qu. Max. NA's
pls 0.5397  0.8333 0.8672 0.8713  0.9509    1    0
rda 0.6984  0.8398 0.9028 0.8860  0.9787    1    0

Sens
      Min. 1st Qu. Median   Mean 3rd Qu. Max. NA's
pls 0.3333  0.7500 0.7778 0.7782  0.8750    1    0
rda 0.4444  0.6875 0.8750 0.8060  0.8889    1    0

Spec
      Min. 1st Qu. Median   Mean 3rd Qu. Max. NA's
pls 0.5000  0.7143 0.8571 0.8131  0.9688    1    0
rda 0.1429  0.7232 0.8571 0.8036  0.8571    1    0
```

6.2 사용 절차: PLS 회귀 예제

- 이러한 결과를 시각화하는 몇 가지 함수가 있다. 예를 들어, Bland Altman 유형 플롯은 xyplot(resamps, what="BlandAltman")을 사용한다. 그 결과는 비슷하다.
- 각 재표본에 대해 쌍을 이루는 결과가 있으므로 평균 재표본된 AUC에 차이가 있는지를 평가하는데 대응 t-검정이 사용될 수 있다.

```
> xyplot(resamps, what = "BlandAltman")
```

6.2 사용 절차: PLS 회귀 예제



참 고 재표본된 ROC 값의 Bland-Altman 그림이다. x축은 두 방법의 평균값이고, y축은 두 방법의 차이(pls-rda)를 나타낸다.

6.2 사용 절차: PLS 회귀 예제

```
> diffs <- diff(resamps)
> summary(diffs)

Call:
summary.diff.resamples(object = diffs)

p-value adjustment: bonferroni
Upper diagonal: estimates of the difference
Lower diagonal: p-value for H0: difference = 0

ROC
    pls    rda
pls        -0.01469
rda 0.2975

                                    (…)
```

6.2 사용 절차: PLS 회귀 예제

```
                                        (…)
Sens
    pls    rda
pls        -0.02778
rda 0.125

Spec
    pls     rda
pls         0.009524
rda 0.7348
```

참 고 이 분석을 바탕으로 모형 간의 차이는 -0.015 ROC 단위이며(RDA 모형이 약간 큼), 차이에 대한 양측 검정의 p-값은 0.29749이다.

6.2 사용 절차: PLS 회귀 예제

- 이 절의 내용은 Kuhn(2016)의 "A short introduction to the caret package"를 옮긴 것이다. {caret} 패키지에 대한 더 자세한 도움말은 아래의 사이트를 참조하기 바란다.

http://caret.r-forge.r-project.org/

6.3 {caret}의 적용 범위(모형)

- 다음의 [표 6.2]는 {caret} 패키지의 train() 함수에서 사용되는 모형을 정리한 것이다. 이 함수들은 method= 옵션에서 지정된다.

[표 6.2] train() 함수에 사용되는 모형

모형	method=옵션	패키지명	조율(tuning)모수
Recursive partitioning	rpart	rpart*	maxdepth
	ctree	party	mimcriterion
Boosted trees	gbm	gbm*	interaction.depth, n.trees, shrinkage
	blackboost	mboost	maxdepth, mstop
	ada	ada	maxdepth, iter, nu
Other boosted models	glmboost	mboost	mstop
	gamboost	mboost	mstop
	logitboost	caTools	nIter

(…)

6.3 {caret}의 적용 범위(모형)

[표 6.2] 부분최소제곱회귀의 수행원리

모형	method=옵션	패키지명	조율(tuning)모수
Random forests	rf	randomForest*	mtry
	cforest	party	mtry
Bagged trees	treebag	ipred	None
Neural Networks	nnet	nnet	decay, size
Partial least squares	pls*	pls, caret	ncomp
Support vector machines (RBF kernel)	svmRadial	kernlab	sigma, C
Support vector machines (polynomial kernel)	svmPoly	kernlab	scale, degree, C
Gaussian process (RBF kernel)	gausssprRadial	kernlab	sigma
Gaussian process (polynomial kernel)	gausssprPoly	kernlab	scale, degree

6.3 {caret}의 적용 범위(모형)

[표 6.2] 부분최소제곱회귀의 수행원리

모형	method=옵션	패키지명	조율(tuning)모수
Linear least squares	lm*	stats	None
Multivariate adaptive regression splines	earth, mars	earth	degree, nprune
Bagged MARS	bagEarth*	caret, earth	degree, nprune
Elastic net	enet	elasticnet	lambda, fraction
The lasso	lasso	elasticnet	fraction
Relevance vector machines (RBF kernel)	rvmRadial	kernlab	sigma
Relevance vector machines (polynomial kernel)	rvmPoly	kernlab	scale, degree

6.3 {caret}의 적용 범위(모형)

[표 6.2] 부분최소제곱회귀의 수행원리

모형	method=옵션	패키지명	조율(tuning)모수
Linear discriminant analysis	lda	MASS	None
Stepwise diagonal discriminant analysis	sddaLDA, sddaQDA	SDDA	None
Logistic/multinomial regression	multinom	nnet	decay
Regularized discriminant analysis	rda	klaR	lambda, gamma
Flexible discriminant analysis (MARS basis)	fda*	mda, earth	degree, nprune

6.3 {caret}의 적용 범위(모형)

[표 6.2] 부분최소제곱회귀의 수행원리

모형	method=옵션	패키지명	조율(tuning)모수
Bagged FDA	bagFDA*	caret, earth	degree, nprune
Least squares support vector machines (RBF kernel)	lssvmRadial	kernlab	sigma
k nearest neighbors	knn3	caret	k
Nearent shrunken centroids	pam*	pamr	threshold
Naive Bayes	nb	klaR	usekernel
Generalized partial least squares	gpls	gpls	K.prov
Learned vector quantization	lvq	class	k

(* 표시는 모형-특정적 변수 중요도 방법을 사용할 수 있음)

6.4 {caret}을 이용한 변수선택

6.4.1 변수의 중요도

- 변수의 중요도(importance)는 모형을 작성하여 데이터로부터 추정할 수 있다. 의사결정나무와 같은 일부 방법에는 변수 중요도를 알려주는 메커니즘이 내장되어 있다. 다른 알고리즘의 경우 중요도는 각 변수에 대해 수행된 ROC 곡선 분석을 사용하여 추정될 수 있다.
- R의 varImp{caret} 함수는 train 객체 또는 특정 방법(회귀 또는 분류 모형)에 의한 결과 객체에 대한 변수 중요도를 계산해 준다. 이 함수의 일반 형식은 다음과 같다.

```
varImp(object, scale = TRUE, ...)
```

  - object는 train 객체 또는 특정 모형의 결과 객체("earth", "fda", "rpart", "randomForest", "gbm", "classbagg", "regbagg", "pamrtrained", "lm", "mvr", "bagEarth", "bagFDA", "RandomForest", "rfe", "dsa", "multinom", "cubist", "JRip", "PART", "C5.0", "nnet", "glmnet", "plsda")
  - scale=TRUE(디폴트)는 변수 중요도를 0~100 사이의 값으로 제공함

6.4 {caret}을 이용한 변수선택

예제 1 PimaIndiansDiabetes{mlbench} 자료에 대해 학습벡터양자화(learning vector quantization, LVQ) 모형을 구축한다. varImp() 함수를 사용하여 변수 중요도를 추정하고 이를 출력한다. glucose, mass와 age가 이 데이터 셋에서 가장 중요한 3가지 변수이며 insulin 변수가 가장 덜 중요함을 보여 준다.

```
> set.seed(100)
> library(mlbench)
> library(caret)
> data(PimaIndiansDiabetes)    # 768개 관측치와 9개 변수
```

6.4 {caret}을 이용한 변수선택

```
> # 훈련에 대한 설정
> control <- trainControl(method="repeatedcv", number=10, repeats=3)
```

```
> # 모형훈련
> model <- train(diabetes~., data=PimaIndiansDiabetes, method="lvq",
                 preProcess="scale", trControl=control)
Loading required package: class
```

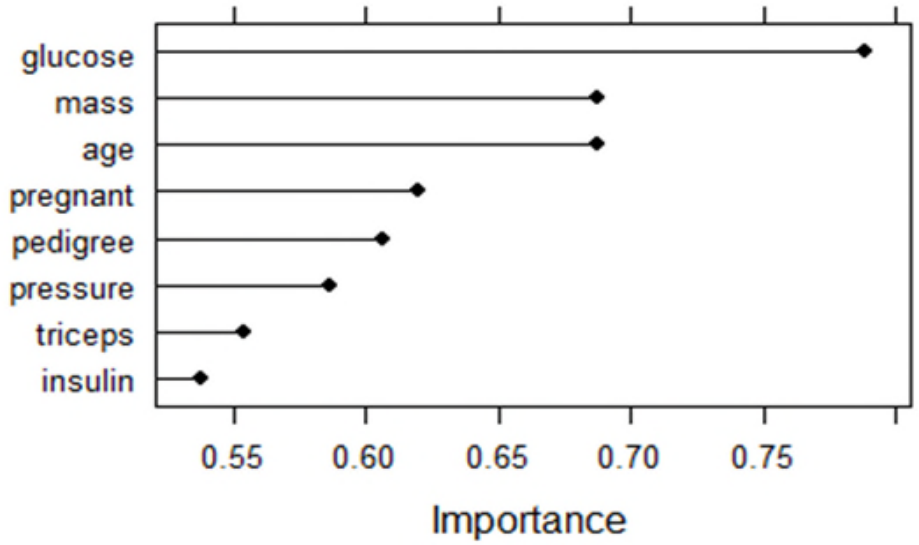
```
> # 변수 중요도 추정
> importance <- varImp(model, scale=FALSE)
```

6.4 {caret}을 이용한 변수선택

```
> print(importance)
ROC curve variable importance

         Importance
glucose      0.7881
mass         0.6876
age          0.6869
pregnant     0.6195
pedigree     0.6062
pressure     0.5865
triceps      0.5536
insulin      0.5379
```

6.4 {caret}을 이용한 변수선택

```
> plot(importance)
```



6.4 {caret}을 이용한 변수선택

6.4.2 변수 선택

- 자동 변수선택방법은 데이터 셋의 서로 다른 부집합에 대해 여러 모형을 적합하고 정확한 모형을 구축하는 데 필요하고 필요하지 않은 변수를 식별할 수 있다.
- {caret} 패키지에서 제공되는 가장 널리 사용되는 자동 변수선택방법은 후진제거(backwards elimination)법으로도 알려진 재귀적 변수제거(recursive feature elimination, 이하 RFE) 방법으로, 예측변수의 중요도에 따라 하나씩 제거해 나간다.

6.4 {caret}을 이용한 변수선택

- R의 rfe{caret} 함수는 RFE 알고리즘을 수행한다. rfe() 함수의 일반 형식은 다음과 같다.

```
rfe(x, y,
    sizes = 2^(2:4),
    metric = ifelse(is.factor(y), "Accuracy", "RMSE"),
    maximize = ifelse(metric == "RMSE", FALSE, TRUE),
    rfeControl = rfeControl(),
    ...)
```

  - x는 모형훈령을 위한 예측변수(행렬 또는 데이터프레임)
  - y는 훈련용 자료의 반응변수 벡터(수치형 또는 요인)
  - sizes= 고려해야 할 변수의 수

6.4 {caret}을 이용한 변수선택

예제 2 PimaIndiansDiabetes 자료에 대해 RFE 방법을 적용한다. 모형을 평가하기 위해 매 반복마다 랜덤포리스트 알고리즘이 사용되었다. 알고리즘은 변수의 모든 가능한 부집합을 탐색하도록 구성되었다. 이 예에서는 8개의 변수가 모두 선택되었다.

```
> set.seed(50)
> library(mlbench)
> library(caret)
> data(PimaIndiansDiabetes)
> # 랜덤포리스트 선택함수(functions=rfFuncs)를 사용하여 RFE를 조율
> # method= 재표본의 방법 지정: "boot", "cv", "LOOCV", "LGOCV"
> # number= 중첩의 수 또는 재표본 반복의 수를 지정
> control <- rfeControl(functions=rfFuncs, method="cv", number=10)
```

6.4 {caret}을 이용한 변수선택

```
> # RFE 알고리즘을 수행
> results <- rfe(PimaIndiansDiabetes[,1:8], PimaIndiansDiabetes[,9],
                 sizes=c(1:8), rfeControl=control)
> # size= 모형선택을 위해 고려할 변수의 수 지정
> print(results)

Recursive feature selection
Outer resampling method: Cross-Validated (10 fold)
Resampling performance over subset size:
 Variables Accuracy  Kappa AccuracySD KappaSD Selected
         1   0.6927 0.2681    0.03248 0.07499
         2   0.7369 0.4062    0.04125 0.09603
         3   0.7423 0.4199    0.04503 0.09986
         4   0.7579 0.4596    0.03835 0.09166
         5   0.7539 0.4477    0.04532 0.10436
         6   0.7513 0.4364    0.05051 0.11237
         7   0.7565 0.4492    0.05989 0.13187
         8   0.7696 0.4748    0.05432 0.12490        *

The top 5 variables (out of 8):
   glucose, mass, age, pregnant, insulin
```

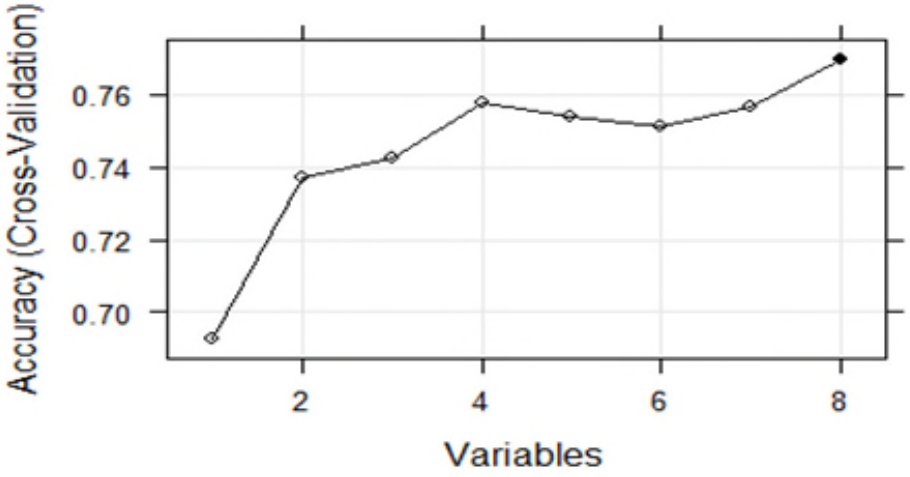
해석

교차타당법을 이용한 평가 결과 모형에 포함될 최적의 변수 크기는 8임.

6.4 {caret}을 이용한 변수선택

```
> predictors(results)
[1] "glucose"  "mass"     "age"      "pregnant" "insulin"
[6] "pedigree" "triceps"  "pressure"
```

```
> plot(results, type=c("g", "o"))
```



6.4 {caret}을 이용한 변수선택

용 어 **Regularization(제약 또는 일반화)**

통계학 분야에서 “regularization”은 과적합(overfitting)의 문제를 피하기 위해 추가적인 정보를 도입하는 과정이다. 이 정보는 주로 복잡성에 대한 벌점(penalty)의 형태로 주어진다.

Regularization의 예로는 라소나 능형회귀에서 손실함수에 벌점항(penalty term)을 추가하는 과정, 판별분석에서 손실함수에 제약항(regularization term or regularizer)을 추가하는 과정, 신경망에서 비용함수에 제약항을 추가하는 과정(“weight decay” 방법) 등이 있다.

6.4 {caret}을 이용한 변수선택

용 어 Regularization(제약 또는 일반화)

아울러, 기계학습(machine learning)의 관점에서 통계적 모형은 오차가 포함된 관측값을 학습하므로 오차를 같이 학습하려는 경향을 가진다. 따라서 적합모형이 모형적합에 사용되지 않은 새로운 자료에 대해서는 예측력이 떨어지는 경우가 발생한다.

즉, 모형의 일반화(generalization)가 잘 되지 않는다. 따라서 단순히 표본내의 오차만을 최소화하는 것이 아니라 특정 제약항을 추가함으로써 일반화의 성질을 좋게 하는 목적으로 사용할 수 있다.