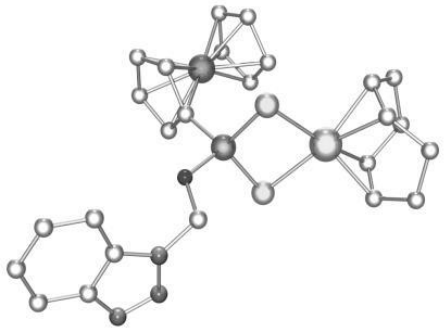


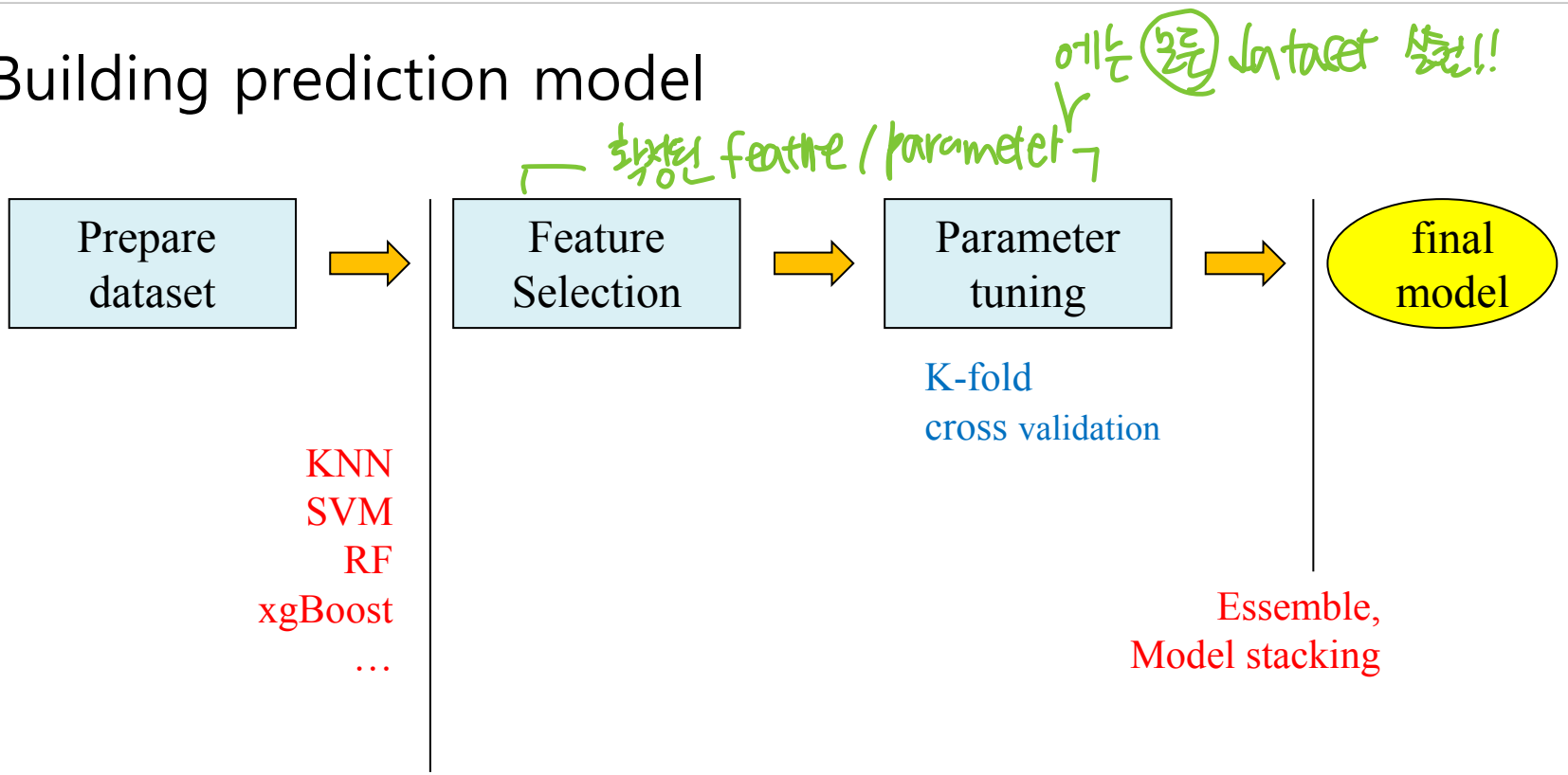
Feature Selection

Contents

- Feature selection
- How to evaluate
- How to select
- Simple FS algorithm

Summary

- Building prediction model



Feature Selection

- Classification 작업을 위한 전처리(preprocessing) 과정중의 하나
- 대상 dataset에 많은 수의 feature 가 포함되어 있는경우
  - Noise 가 포함되어 있을 수 있고
  - 중복된 feature 들이 있을 수 있다

- Feature selection 의 목표
  - Noise 와 중복을 제거하고
  - Classification accuracy 를 높여줄 수 있는 feature 들을 선별 한다.

Feature Selection

- Finding a feature subset that has the most discriminative information from the original feature space

thousands of features

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	0	88.233	1073.778	91.741	412.169	1253.27	108.913	10.882	82.505	72.125	275.624	3.411	935.091	506.61	764.704	347.897	3221.76	893.669	60.75	107.94
2	0	158.122	1319.648	38.94	357.442	1218.78	167.104	8.174	25.982	30.873	176.096	2.782	872.484	755.919	713.825	616.568	2404.192	963.634	169.422	89.305
3	0	159.443	1775.569	166.406	233.159	1161.329	96.989	14.17	124.004	30.002	257.097	4.946	778.163	778.327	743.13	509.338	2578.874	977.317	45.349	208.574
4	0	153.389	576.144	115.843	407.702	959.416	218.624	24.641	111.434	24.199	82.268	5.492	824.365	541.556	657.471	592.46	2231.719	704.34	45.932	260.728
5	0	67.095	1849.944	122.145	412.334	1089.363	78.337	12.565	99.277	21.579	251.408	2.821	587.29	644.925	733.738	597.25	2320.226	829.391	33.93	285.242
6	0	115.81	1787.291	51.875	295.794	1376.619	187.163	25.593	192.796	25.04	85.998	3.043	672.854	840.849	684.574	438.819	2146.98	1144.874	25.276	292.81
7	0	22.38	472.08	69.863	324.712	423.062	29.759	20.324	267.732	17.935	17.858	8.348	666.969	897.061	621.93	325.799	1510.115	424.769	45.439	148.96
8	0	195.234	1721.286	44.152	321.256	996.295	155.034	19.589	209.104	16.345	162.532	6.046	808.662	1137.29	951.609	666.563	3042.031	1071.724	164.425	98.655
9	0	49.727	429.134	104.177	249.983	515.067	87.265	9.082	20.304	7.056	6.593	3.329	460.959	528.484	729.029	360.421	2920.311	562.617	26.609	89.991
10	0	21.388	51.043	85.25	285.091	513.043	21.008	14.676	84.047	13.655	5.02	2.256	386.164	882.61	468.555	256.696	2499.596	151.973	29.331	59.136
11	0	211.443	1819.18	134.674	741.773	1221.95	99.301	17.402	46.112	22.712	134.902	10.598	796.441	1014.167	116.055	790.678	2881.319	1010.274	123.387	115.306
12	0	96.847	404.113	116.928	339.466	554.361	37.806	9.82	121.809	23.236	4.903	2.751	535.14	645.464	510.215	85.095	2080.999	444.301	49.85	42.68
13	0	39.629	2016.821	83.194	468.33	1938.943	82.717	14.681	146.35	11.535	195.427	2.359	599.625	591.332	765.142	572.608	2985.784	1215.491	63.216	81.907
14	0	47.856	257.848	329.572	672.674	50.842	198.015	66.276	50.264	34.278	15.449	7.85	932.977	243.047	983.763	179.568	104.245	42.254	65.561	95.313
15	0	63.966	1241.567	42.812	320.39	843.894	131.522	12.135	23.874	20.108	83.15	3.946	514.814	513.127	740.37	579.655	3007.358	1302.175	70.209	211.324
16	0	66.231	1341.094	38.388	452.191	843.749	144.431	14.147	140.765	30.254	17.296	4.663	360.942	327.566	554.683	517.91	2611.451	1070.009	38.515	165.508
17	0	76.883	2220.198	37.409	320.554	2002.628	98.65	11.926	86.213	8.788	134.632	3.008	525.598	664.076	748.957	555.856	2528.629	1044.925	125.549	172.597
18	0	116.288	1740.58	163.99	466.434	1131.244	172.678	18.557	118.9	32.535	211.57	3.985	956.67	709.275	743.686	432.731	2629.465	895.596	62.047	201.56
19	0	95.876	1208.355	87.969	189.802	1407.627	60.658	4.813	68.793	17.63	46.79	2.854	376.065	534.491	725.685	600.09	2590.157	1537.115	105.57	251.044
20	1	152.693	1489.308	112.843	378.559	1361.104	21.49	14.252	205.784	32.265	99.481	3.421	766.445	746.347	906.626	496.991	2117.147	908.547	108.659	115.797
21	1	73.313	1689.044	34.257	346.135	1403.109	24.225	11.477	247.829	23.638	184.243	2.575	996.667	313.226	730.595	537.606	1878.863	615.586	211.092	115.047
22	1	79.738	1082.284	53.81	215.853	1444.564	71.739	35.684	131.992	48.408	217.153	5.563	986.013	559.023	876.828	637.939	3605.016	754.428	109.18	80.444
23	1	92.809	1776.862	89.344	316.74	1712.537	130.069	8.891	83.499	10.182	223.336	3.013	809.63	588.67	684.392	510.608	2501.585	842.542	54.734	65.506
24	1	73.6	1343.897	82.311	1104.771	1581.533	31.039	283.218	618.272	30.237	262.392	3.119	1342.914	93.496	513.6	335.092	1633.52	550.721	3109.234	351.644
25	1	189.453	1701.781	69.025	570.107	1333.214	30.728	177.454	487.425	33.018	161.755	4.686	928.637	114.64	647.23	383.538	1603.914	576.533	1961.021	321.257
26	1	76.924	1838.192	123.977	368.19	1378.117	32.035	12.766	93.28	12.511	129.233	2.644	883.005	457.416	812.954	575.992	1640.058	734.809	113.455	190.224
27	1	139.589	2529.31	94.094	309.911	1608.247	70.06	14.975	141.988	52.1	109.752	4.086	748.119	653.128	788.104	619.724	1741.192	823.923	35.307	281.273

smaller good features

E	G	I	N	S
412.169	108.913	82.505	506.61	60.75
357.442	167.104	25.982	755.919	169.422
233.159	96.989	124.004	778.327	45.349
407.702	218.624	111.434	541.556	45.932
412.334	78.337	99.277	644.925	33.93
295.794	187.163	192.796	840.849	25.276
324.712	29.759	267.732	897.061	45.439
321.256	155.034	209.104	1137.29	164.425
249.983	87.265	20.304	528.484	26.609
285.091	21.008	84.047	882.61	29.331
741.773	99.301	46.112	1014.167	123.387
339.466	37.806	121.809	645.464	49.85
468.33	82.717	146.35	591.332	63.216
672.674	198.015	50.264	243.047	65.561
320.39	131.522	23.874	513.127	70.209
452.191	144.431	140.765	327.566	38.515
320.554	98.65	86.213	664.076	125.549
466.434	172.678	118.9	709.275	62.047
189.802	60.658	68.793	534.491	105.57
378.559	21.49	205.784	746.347	108.659
346.135	24.225	247.829	313.226	211.092
215.853	71.739	131.992	559.023	109.18
316.74	130.069	83.499	588.67	54.734
1104.771	31.039	618.272	93.496	3109.234
570.107	30.728	487.425	114.64	1961.021
368.19	32.035	93.28	457.416	113.455
309.911	70.06	141.988	653.128	35.307

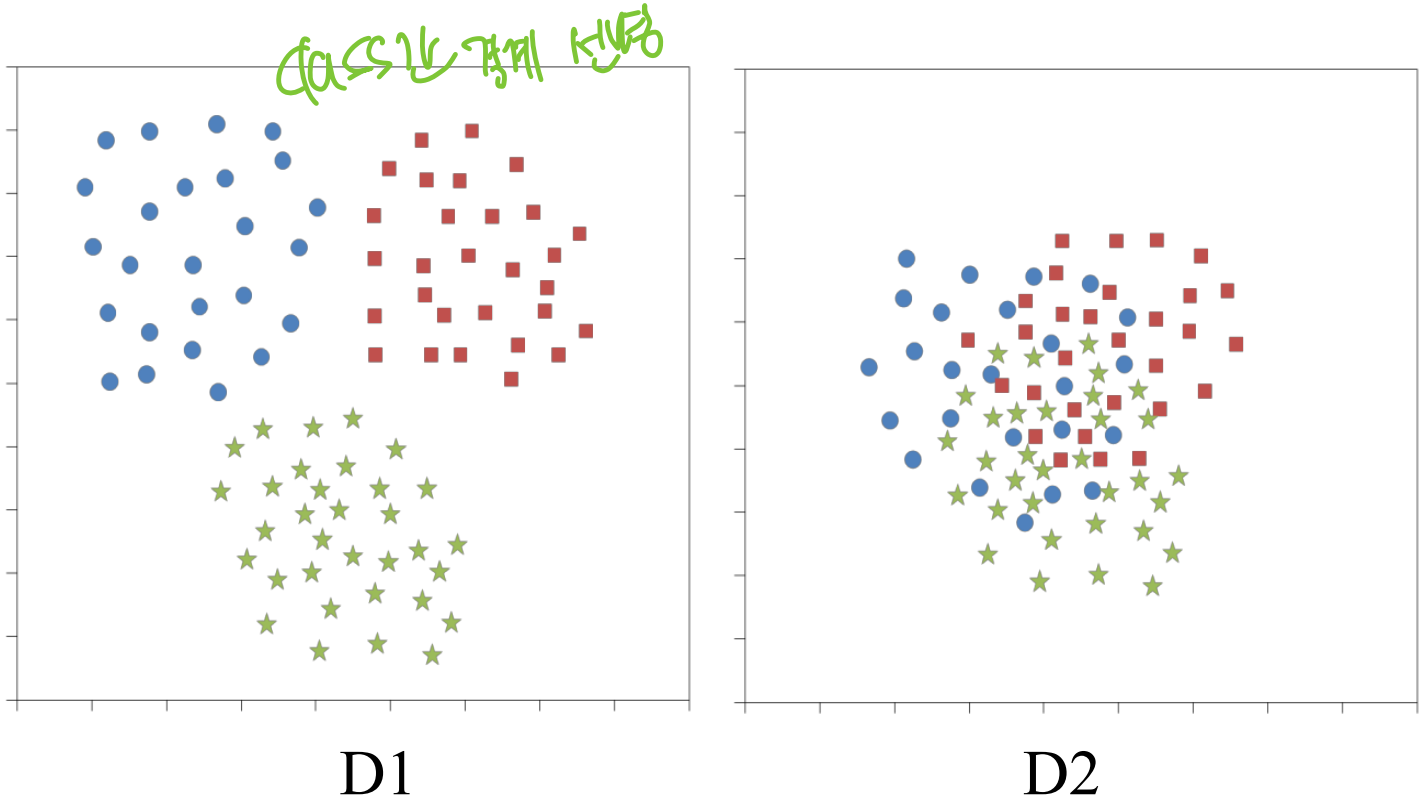
Feature Selection

- Objectives
  - To reduce dimensionality and remove noise
  - To improve mining performance
    - speed of learning
    - predictive accuracy
    - simplicity and comprehensibility of mined results

- 선별한 feature 자체가 의미 있는 경우도 있다.
  - Gene selection for microarray data

Feature Selection

- 어떤 feature 가 좋은 feature 인가
  - Class 간 경계가 clear 할수록 classification accuracy 가 높을 것이다.



Feature Selection

- 주어진 데이터셋으로 부터 항상 최고의 성능을 얻을 수 있는 feature selection 방법은?
  - 모든 feature 의 조합을 테스트
  - Greed search

모든 feature 조합 실험하긴 현실적 X

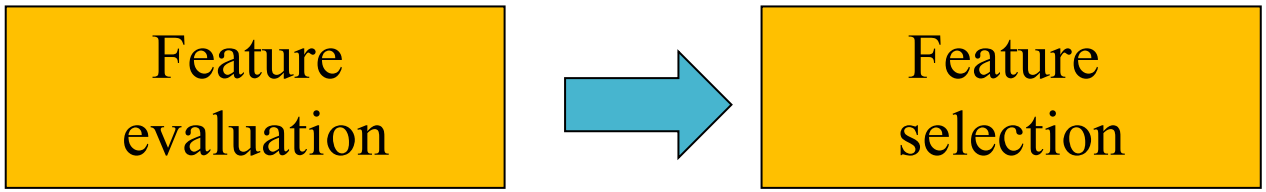
  - Feature 수가 20개일 때 모든 조합의 개수는?
    2^{20} = 1,048,576

  - greed search는 현실적으로 사용가능 하지 않음
  - Feature selection 이 필요한 이유

How to evaluate

- Dataset 에 포함된 각 feature 들을 어떻게 평가할 것인가
  - Feature evaluation function 이용
  - f(a) > f(b) 이면 a 가 b 보다 classification 에 유리함을 의미
  - 현재 수많은 evaluation function 이 발표 되었다.
  - FSDD, MRMR, Relief, RFS, CBFS, …

feature 평가 함수



How to evaluate

- 1. Filter Methods - Univariate
  - Consider one feature's contribution to the class at a time,
    - e.g. Information gain, chi-square
  - Advantages
    - Computationally efficient and parallelable
  - Disadvantages
    - May select low quality feature subsets

Feature들 간의 interaction 을 반영하지 못하기 때문에 성능이 좋지 않다

How to evaluate

- 1. Filter methods - Multivariate
  - Consider the contribution of a set of features to the class variable, e.g.
    - CFS (correlation feature selection) [M Hall, 2000]
    - FCBF (fast correlation-based filter) [Lei Yu, etc. 2003]
  - Advantages:
    - Computationally efficient
    - Select higher-quality feature subsets than univariate filters
  - Disadvantages:
    - Not optimized for a given classifier

How to evaluate

- 2. Wrapper methods 모델 알고리즘이 뽑은 feature select 함 !!
  - Select a feature subset by building classifiers e.g.
    - LASSO (least absolute shrinkage and selection operator) [R Tibshirani, 1996]
    - SVM-RFE (SVM with recursive feature elimination) [I Guyon, etc. 2002]
    - RF-RFE (random forest with recursive feature elimination) [R Uriarte, etc. 2006]
    - RRF (regularized random forest) [H Deng, etc. 2012]
  - Advantages:
    - Select high-quality feature subsets for a particular classifier
  - Disadvantages:
    - RFE methods are relatively computationally expensive.

How to evaluate

- 3. embedded method
  - Feature 평가 + 모델 구축이 통합되어 있음
  - Ex) Random Forest

How to select

- 방법1. 모든 feature 를 평가한 뒤에 평가점수가 높은 상위 *n* 개의 feature 를 선택한다. (with filter method)

- 방법2. Forward Search. (with wrapper method)
  - 평가 점수가 제일 높은 feature x 를 선택 집합에 넣는다.
  - x 와 나머지 feature 들과의 조합 중 평가점수가 x 보다 가장 높아지게 하는 feature y 를 찾아서 선택 집합에 넣는다.
  - 두번째 과정을 반복하되 모든 조합에 대해 이전 보다 평가 점수가 떨어지면 반복을 중단한다.

- 방법3. Backward elimination. (with wrapper method)
  - 일단 모든 feature 들을 선택 집합에 넣은 후 안 좋은 feature 들을 하나하나 제거해 가는 방법

FSelector package

- R packages for feature selection
  - FSelector
  - varSelRF
  - caret

FSelector 는 자바가 설치되고, 환경변수에 경로가 등록되어 있어야 함

FSelector package

- Filter method
  - chi.squared
  - linear.correlation
  - rank.correlation
  - information.gain
  - gain.ratio
  - symmetrical.uncertainty
  - oneR
  - random.forest.importance
  - relief
  - cfs

FSelector package

- Filter method

```
library(FSelector)
data(iris)

# evaluate each feature
weights <- relief(Species~., iris,
            neighbours.count = 5, sample.size = 20)
print(weights)
# choose best 2 features
subset <- cutoff.k(weights, 2)
# make new dataset
fsIris = cbind(iris[,subset], iris[,5])
```

FSelector package

- Filter method

```
> data(iris)
> weights <- relief(Species~., iris,
+                   neighbours.count = 5, sample.size = 20)
> print(weights)
             attr_importance
Sepal.Length       0.1409722
Sepal.Width        0.1566667
Petal.Length       0.3333051
Petal.Width        0.3456250
> # choose best 2 features
> subset <- cutoff.k(weights, 2)
> subset
[1] "Petal.Width"  "Petal.Length"
> fsIris = cbind(iris[,subset], iris[,5])
> head(fsIris)
  Petal.Width Petal.Length iris[, 5]
1         0.2          1.4    setosa
2         0.2          1.4    setosa
3         0.2          1.3    setosa
4         0.2          1.5    setosa
5         0.2          1.4    setosa
6         0.4          1.7    setosa
```

FSelector package

- Forward Search

```
library(FSelector)
data(iris)

library(e1071)
evaluator <- function(subset) {
  #5-fold cross validation
  k <- 5
  splits <- runif(nrow(iris))
  results = sapply(1:k, function(i) {
    test.idx <- (splits >= (i - 1) / k) & (splits < i / k)
    train.idx <- !test.idx
    test <- iris[test.idx, subset]
    train <- iris[train.idx, subset]
    model <- svm(train, iris$Species[train.idx])
    pred <- predict(model, test)
    acc = mean(pred==iris$Species[test.idx])
    return(acc)
  })
  print(subset)
  print(mean(results))
  return(mean(results))
}

subset <- forward.search(names(iris)[-5], evaluator)
```

FSelector package

```
> subset <- forward.search(names(iris)[-5], evaluator)
[1] "Sepal.Length"
[1] 0.7211232
[1] "Sepal.Width"
[1] 0.5051381
[1] "Petal.Length"
[1] 0.9418952
[1] "Petal.Width"
[1] 0.9596759
[1] "Sepal.Length" "Petal.Width"
[1] 0.9608791
[1] "Sepal.Width" "Petal.Width"
[1] 0.9534606
[1] "Petal.Length" "Petal.Width"
[1] 0.9595357
[1] "Sepal.Length" "Sepal.Width"  "Petal.Width"
[1] 0.9475613
[1] "Sepal.Length" "Petal.Length" "Petal.Width"
[1] 0.9583364
```

FSelector package

- Backward Elimination

```
library(FSelector)
data(iris)

library(e1071)
evaluator <- function(subset) {
  #5-fold cross validation
  k <- 5
  splits <- runif(nrow(iris))
  results = sapply(1:k, function(i) {
    test.idx <- (splits >= (i - 1) / k) & (splits < i / k)
    train.idx <- !test.idx
    test <- iris[test.idx, subset]
    train <- iris[train.idx, subset]
    model <- svm(train, iris$Species[train.idx])
    pred <- predict(model, test)
    acc = mean(pred==iris$Species[test.idx])
    return(acc)
  })
  print(subset)
  print(mean(results))
  return(mean(results))
}

subset <- backward.search(names(iris)[-5], evaluator)
```

FSelector package

```
> subset <- backward.search(names(iris)[-5], evaluator)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
[1] 0.9614951
[1] "Sepal.Width"  "Petal.Length" "Petal.Width"
[1] 0.9434332
[1] "Sepal.Length" "Petal.Length" "Petal.Width"
[1] 0.9495012
[1] "Sepal.Length" "Sepal.Width"  "Petal.Width"
[1] 0.9410104
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length"
[1] 0.9261937
```

Note.

- Feature selection 의 현실적 적용
  - 1) Filter method 로 개별 feature 평가 후 best *n* 을 찾는다
  - 2) forward, backward search 를 적용해 본다.
  - 3) Filter method 로 feature 수를 100개 미만으로 줄인 후 k 개로 구성된 feature 조합에 대해서 평가해 본다.
  - 4) 앞에서 시도한 방법중 가장 높은 성능을 보이는 feature 조합을 선택한다

[실습]

- mlbench 에 포함된 PimaIndiansDiabetes2 데이터셋에 대해 가장 높은 성능을 보이는 feature 의 조합을 찾아 보시오
  - 분류 알고리즘은 random forest 만 테스트
  - 가급적 많은 feature selection 을 테스트한다.