새로운 특성(feature)를 선택하는 방법 ¶

학습 목표

- 새로운 특성을 여러가지 방법으로 선택하는 것에 대해 알아봅니다.

학습 내용

- 01 일변량 통계(univariate statistics)
- 02 모델 기반 선택(model-based selection)
- 03 반복적 선택(iterative selection)

목차

01. 일변량 통계

목차로 이동하기

- 개개의 특성과 타깃(목표변수) 사이에 중요한 통계적 관계가 있는지 계산
- 분류에서는 분산분석(ANOVA)라고 한다.
- 각 특성(feature)이 독립적으로 평가.
- 계산이 매우 빠르고 평가를 위한 모델을 만들 필요가 없음.
- SelectPercentile에서 특성을 선택하는 기준은 F-값. 값이 클수록 클래스 평균의 분산이 비교적 크다.
  - F-value는 표본 집단을 비교하기 위한 지표.
  - F = (표본평균간 퍼진 정도)/(표본내에서 퍼진 정도)

분류 - f_classif, 회귀 - f_regression

In [1]:

```
import warnings
warnings.filterwarnings(action='ignore')
# warnings.filterwarnings(action='default')
```

In [2]:

```
from sklearn.feature_selection import SelectPercentile, f_classif

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
import numpy as np
```

유방암 데이터에 노이즈를 데이터를 추가

In [3]:

```
cancer = load_breast_cancer()
print(cancer.data.shape)
```

(569, 30)

In [4]:

```
# 고정된 난수를 발생
rng = np.random.RandomState(42)
noise = rng.normal(size=(len(cancer.data), 90))
noise.shape
```

Out[4]:

(569, 90)

In [5]:

```
# 데이터 노이즈 특성 추가
# 30개는 원본 특성, 다음 90개는 노이즈
X_w_noise = np.hstack([cancer.data, noise])
X_w_noise.shape
```

Out[5]:

(569, 120)

In [6]:

```
X = X_w_noise  # 입력
y = cancer.target # 출력

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    random_state=0,
                                                    test_size=0.3)
```

전체 특성을 이용한 모델 학습 및 평가

In [8]:

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()
model.fit(X_train, y_train)
print("전체 특성 사용(학습) : {:.3f}".format(model.score(X_train, y_train)))
print("전체 특성 사용(테스트) : {:.3f}".format(model.score(X_test, y_test)))
```

전체 특성 사용(학습) : 1.000
전체 특성 사용(테스트) : 0.953

In [10]:

```
# 30%를 뽑는 것을 학습
select = SelectPercentile(score_func=f_classif, percentile=30)
select.fit(X_train, y_train)

## 학습 세트에 적용
X_train_selected = select.transform(X_train)

print( "X_train.shape:", X_train.shape)
print( "X_train_selected.shape", X_train_selected.shape)
```

```
X_train.shape: (398, 120)
X_train_selected.shape (398, 36)
```

- 결과를 통해 우리는 특징 개수가 120개에서 36 개로 줄어든 것을 확인할 수 있음.

In [11]:

```
import matplotlib.pyplot as plt
```

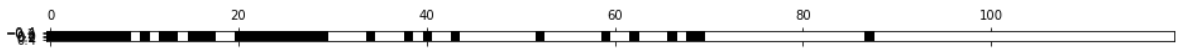
In [12]:

```
### 어떤 특성이 선택되었는지 확인
mask = select.get_support()
print(mask)
plt.matshow(mask.reshape(1, -1), cmap='gray_r')
```

```
[ True  True  True  True  True  True  True  True  True False  True False
  True  True False  True  True  True False False  True  True  True  True
  True  True  True  True  True  True False False False False  True False
 False False  True False  True False False  True False False False False
 False False False False  True False False False False False False  True
 False False  True False False False  True False  True  True False False
 False False False False False False False False False False False False
 False False False  True False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False]
```

Out[12]:

```
<matplotlib.image.AxesImage at 0x170cad52400>
```



36개의 원본 특성에서 많이 남고, 나머지 특성들은 대부분 제거됨.

일부 특성을 이용한 모델 학습 및 평가

In [13]:

```
# 테스트 데이터 변환
X_test_selected = X_test[:, mask]

print(X_test_selected.shape)
model.fit(X_train_selected, y_train)
print("일부 특성 사용(학습) : {:.3f}".format(model.score(X_train_selected, y_train)))
print("일부 특성 사용(테스트): {:.3f}".format(model.score(X_test_selected, y_test)))
```

```
(171, 36)
일부 특성 사용(학습) : 1.000
일부 특성 사용(테스트): 0.965
```

1-1-2 모델 기반 특성 선택

목차로 이동하기

- 지도 학습 머신러닝 모델을 사용하여 **특성의 중요도를 평가**해서 가장 중요한 특성들만 선택
- 특성 선택에 사용하는 지도 학습 모델은 최종적으로 사용할 지도학습 모델과 같을 필요는 없음.
- 결정트리와 유사한 모델은 feature_importance_ 속성을 제공함.
- 선형 모델의 절대값으로 특성의 중요도를 재는데 사용
- 모델 기반의 특성 선택은 SelectFromModel에 구현되어 있음.

In [14]:

```
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier
select = SelectFromModel(RandomForestClassifier(n_estimators=100,
                                                random_state=42),
                                                threshold="1.5 * median") # median : 특성중요도
```

- SelectFromModel은 지도학습 모델로 계산된 중요도가 임계치보다 큰 모든 특성을 선택
- 절반 가량의 특성이 선택될 수 있도록 중간값을 임계치로 사용.
- 트리 100개로 만든 랜덤 포레스트 분류기를 사용.

In [15]:

```
select.fit(X_train, y_train)
X_train_l1 = select.transform(X_train)
print("X_train.shape :" , X_train.shape)
print("X_train_l1.shape :", X_train_l1.shape)
```

```
X_train.shape : (398, 120)
X_train_l1.shape : (398, 36)
```

In [16]:

```
### 어떤 특성이 선택되었는지 확인
mask = select.get_support()
print(mask)
plt.matshow(mask.reshape(1, -1), cmap='gray_r')
plt.xlabel("특성 번호")
```

```
[ True  True  True  True  True  True  True  True  True  True  True False
  True  True False  True  True  True False  True  True  True  True  True
  True  True  True  True  True  True False False  True False False False
 False False False False False  True False False False False False False
 False False False False False False False False  True False  True False
 False False False False  True False False False False False False False
 False False False False False False False False False False False False
 False False False False False  True False False False False False False
 False False False False False False False False False False  True False
  True False False False False False False False  True False False False]
```

Out[16]:

```
Text(0.5, 0, '특성 번호')
```

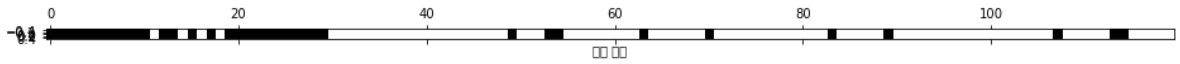
In [17]:

```
select.fit(X_test, y_test)
X_test_l1 = select.transform(X_test)

mask = select.get_support()
plt.matshow(mask.reshape(1, -1), cmap='gray_r')
plt.xlabel("특성 번호")
```

Out[17]:

```
Text(0.5, 0, '특성 번호')
```



- train과 test를 했을 때의 데이터가 다름. 하나로 기준을 정해서 진행.

In [18]:

```
select.fit(X_train, y_train)
X_train_l1 = select.transform(X_train)
mask = select.get_support()
```

In [19]:

```
X_test_l1 = X_test[:, mask]
```

In [20]:

```
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train_l1, y_train)
print("일부 특성 사용(학습) : {:.3f}".format(model.score(X_train_l1, y_train)))
print("일부 특성 사용(테스트) : {:.3f}".format(model.score(X_test_l1, y_test)))

# score = LogisticRegression().fit(X_train, y_train).score(X_test_l1, y_test)
```

```
일부 특성 사용(학습) : 1.000
일부 특성 사용(테스트) : 0.959
```

1-1-3 반복적 특성 선택

목차로 이동하기

- 일변량 모델은 통계를 이용. 모델을 사용하지 않음.(F값)
- 모델 기반 선택은 **하나의 모델**을 사용
- 반복적 특성 선택(iterative Feature Selection)에서는 특성의 수가 각기 다른 일련의 모델이 만들어짐. 두가지 방법
  - 하나, 특성을 하나도 선택하지 않은 상태로 시작해서 어떤 종료 조건까지 하나씩 추가
  - 둘, 모든 특성을 가지고 시작하여 어떤 종료 조건이 될때까지 특성을 하나씩 제거.
- 이 모델들은 앞서 소개한 방법들보다 계산 비용이 훨씬 많이 든다.

- 재귀적 특성 제거(RFE:recursive feature elimination)가 위의 방법 중 하나.

In [21]:

```
%%time

from sklearn.feature_selection import RFE
select = RFE(RandomForestClassifier(n_estimators=100, random_state=42),
             n_features_to_select=36)

select.fit(X_train, y_train)

# 선택된 특성을 표시합니다.
mask = select.get_support()
plt.matshow(mask.reshape(1,-1), cmap='gray_r')
plt.xlabel("특성 번호")
```

```
CPU times: total: 17.5 s
Wall time: 17.5 s
```

Out[21]:

```
Text(0.5, 0, '특성 번호')
```

- 일변량 분석이나 모델 기반 특성보다 특성 선택이 나아짐.
- 랜덤 포레스트 모델은 특성이 누락될때마다 다시 학습하므로 35번 실행.
- 이 코드를 실행하면 모델 기반 선택보다 훨씬 오래 걸림.

In [22]:

```
X_train_rfe = select.transform(X_train)
mask = select.get_support()

model = RandomForestClassifier(n_estimators=100,
                               random_state=42).fit(X_train_rfe, y_train)
score = model.score(X_train_rfe, y_train)
print("학습용 평가 점수 : {:.3f}".format(score))

X_test_rfe = X_test[:, mask]
score = model.score(X_test_rfe, y_test)
print("테스트용 평가 점수 : {:.3f}".format(score))
```

```
학습용 평가 점수 : 1.000
테스트용 평가 점수 : 0.965
```