모델 평가

- 정밀도-재현율 곡선과 ROC 곡선

한글 사전 설정

In [3]:

```
### 한글
import matplotlib
from matplotlib import font_manager, rc
font_loc = "C:/Windows/Fonts/malgunbd.ttf"
font_name = font_manager.FontProperties(fname=font_loc).get_name()
matplotlib.rc('font', family=font_name)
```

정밀도 재현율 곡선을 이용하여 성능을 판단해 보기

In [4]:

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
```

정밀도(x)와 재현율(y) - ROC 커브 확인해 보기

- precision_recall_curve() 메서드 이용

In [5]:

```
from mglearn.datasets import make_blobs
X, y = make_blobs(n_samples=(400, 50),
                  centers=2, cluster_std=[7.0, 2],
                  random_state=22)

print(X.shape, y.shape)
```

```
(450, 2) (450,)

C:₩Users₩front₩anaconda3₩lib₩site-packages₩sklearn₩utils₩deprecation.py:86: FutureWa
rning: Function make_blobs is deprecated; Please import make_blobs directly from sci
kit-learn
  warnings.warn(msg, category=FutureWarning)
```

In [6]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    random_state=0)
svc = SVC(gamma=.05).fit(X_train, y_train)
```

In [7]:

```
pred = svc.predict(X_test)
from sklearn.metrics import classification_report

print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.89      0.93       104
           1       0.35      0.67      0.46         9

    accuracy                           0.88       113
   macro avg       0.66      0.78      0.70       113
weighted avg       0.92      0.88      0.89       113
```

In [8]:

```
from sklearn.metrics import precision_recall_curve

precision, recall, thresholds = precision_recall_curve(
                          y_test, svc.decision_function(X_test))
```

In [9]:

```
# 부드러운 곡선을 위해 데이터 포인트 수를 늘립니다
X, y = make_blobs(n_samples=(4000, 500),
                  centers=2,
                  cluster_std=[7.0, 2],
                  random_state=22)

print(X.shape, y.shape)

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

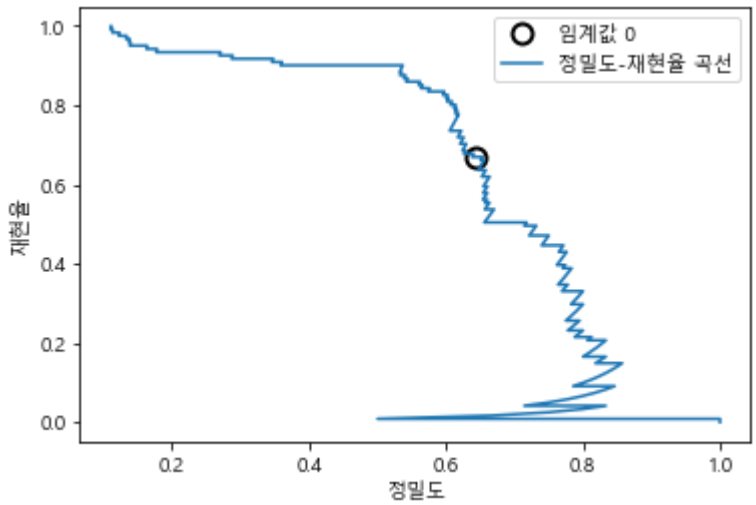
```
(4500, 2) (4500,)

C:₩Users₩front₩anaconda3₩lib₩site-packages₩sklearn₩utils₩deprecation.py:86: FutureWa
rning: Function make_blobs is deprecated; Please import make_blobs directly from sci
kit-learn
  warnings.warn(msg, category=FutureWarning)
```

In [10]:

```
svc = SVC(gamma=.05).fit(X_train, y_train)

pred = svc.decision_function(X_test)

precision, recall, thresholds = precision_recall_curve(
                                            y_test, pred)

# 0에 가까운 임계값을 찾습니다
close_zero = np.argmin(np.abs(thresholds))
plt.plot(precision[close_zero], recall[close_zero], 'o', markersize=10,
         label="임계값 0", fillstyle="none", c='k', mew=2)

plt.plot(precision, recall, label="정밀도-재현율 곡선")
plt.xlabel("정밀도")
plt.ylabel("재현율")
plt.legend(loc="best")
```

Out[10]:

```
<matplotlib.legend.Legend at 0x1ae8986fd00>
```



- 재현율(recall, sensitivity-민감도)
  - 실제 양성 데이터를 양성으로 잘 예측
  - TP/(TP + FN)
- ======
- FPRate => 1-특이도(TN/(FP + TN))
  - FP/(FP + TN)

정밀도

$$\text{정밀도(precision)} = \frac{\text{잘 예측(TP)}}{\text{예측을 양성으로 한 것 전체(TP+FP)}}$$

재현율(recall, 민감도, TPR)

$$\text{민감도(recall, 재현율)} = \frac{\text{잘 예측(TP)}}{\text{실제 값이 양성인것 전체(TP+FN)}}$$

랜덤 포레스트를 이용한 정밀도-재현율의 커브

In [11]:

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=100, random_state=0, max_features=2)
rf.fit(X_train, y_train)
pred =  rf.predict_proba(X_test)[:, 1]
pred
```

Out[11]:

```
array([0.  , 0.35, 0.7 , ..., 0.  , 0.  , 0.  ])
```

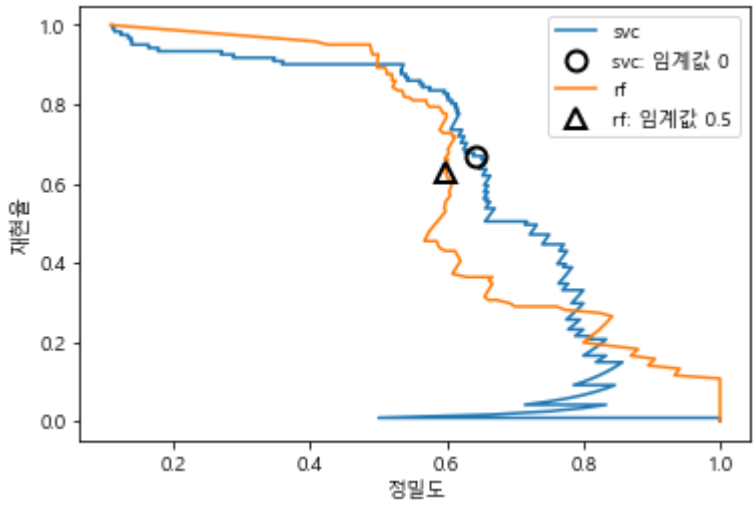
In [12]:

```
# RandomForestClassifier는 decision_function 대신 predict_proba를 제공합니다.
precision_rf, recall_rf, thresholds_rf = precision_recall_curve(
                                        y_test, pred)

# SVC모델 그래프
plt.plot(precision, recall, label="svc")

plt.plot(precision[close_zero],
         recall[close_zero], 'o',
         markersize=10,
         label="svc: 임계값 0",
         fillstyle="none",
         c='k',
         mew=2)

# 랜덤포레스트 그래프
plt.plot(precision_rf, recall_rf, label="rf")

close_default_rf = np.argmin(  np.abs(thresholds_rf - 0.5)  )
print(close_default_rf)

plt.plot(precision_rf[close_default_rf], recall_rf[close_default_rf], '^', c='k',
         markersize=10, label="rf: 임계값 0.5", fillstyle="none", mew=2)
plt.xlabel("정밀도")
plt.ylabel("재현율")
plt.legend(loc="best")
```

47

Out[12]:

<matplotlib.legend.Legend at 0x1ae8ac9fe80>



- 극단적인 부분, 재현율이 매우 높거나, 정밀도가 매우 높을 때는 랜덤포레스트가 더 낫다.
- 정밀도 0.7부분에서는 SVC가 좋음

In [13]:

```
from sklearn.metrics import f1_score

print("랜덤 포레스트의 f1_score: {:.3f}".format(f1_score(y_test, rf.predict(X_test))))
print("svc의 f1_score: {:.3f}".format(f1_score(y_test, svc.predict(X_test))))
```

```
랜덤 포레스트의 f1_score: 0.610
svc의 f1_score: 0.656
```

In [14]:

```
from sklearn.metrics import average_precision_score
ap_rf = average_precision_score(y_test, rf.predict_proba(X_test)[:, 1])
ap_svc = average_precision_score(y_test, svc.decision_function(X_test))

print("랜덤 포레스트의 평균 정밀도: {:.3f}".format(ap_rf))
print("svc의 평균 정밀도: {:.3f}".format(ap_svc))
```

```
랜덤 포레스트의 평균 정밀도: 0.660
svc의 평균 정밀도: 0.666
```

ROC 곡선

- ROC 곡선은 여러 임계값에서 분류기의 특성을 분석하는데 널리 사용되는 도구.
- ROC 곡선은 분류기의 모든 임계값을 고려
- 앞의 그래프의 x는 정밀도, y가 재현율(TPR)이었다면
  - ROC곡선은 x는 (False Positive rate), y를 재현율을(True Positive rate)로 한것.

ROC 와 *AUC*

- FPrate는 1-특이도와 같다
- FPrate는 실제 음성인 데이터 중에 양성으로 예측하여 틀린 것의 비율

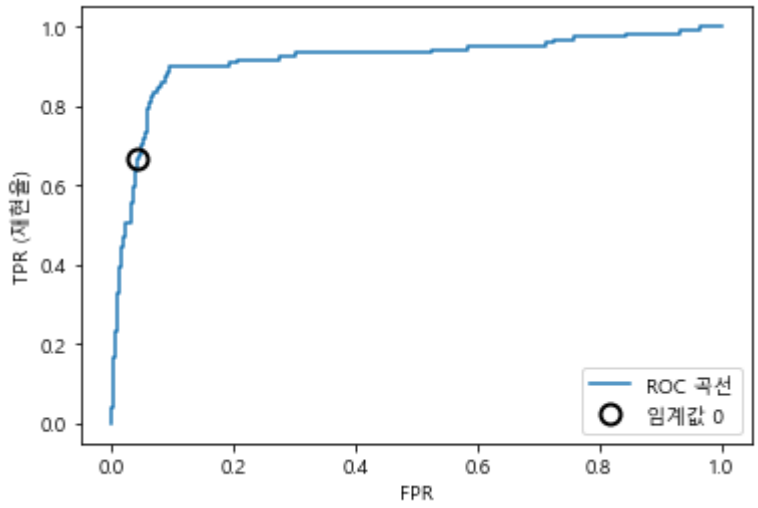
$$\text{FPRate} = \frac{\text{틀린 예측(FP)}}{\text{실제 값이 음성인것 전체(FP + TN)}}$$

In [15]:

```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, svc.decision_function(X_test))

plt.plot(fpr, tpr, label="ROC 곡선")
plt.xlabel("FPR")
plt.ylabel("TPR (재현율)")
# 0 근처의 임계값을 찾습니다
close_zero = np.argmin(np.abs(thresholds))
plt.plot(fpr[close_zero], tpr[close_zero], 'o', markersize=10,
         label="임계값 0", fillstyle="none", c='k', mew=2)
plt.legend(loc=4)
```

Out[15]:

```
<matplotlib.legend.Legend at 0x1ae8ace0e50>
```

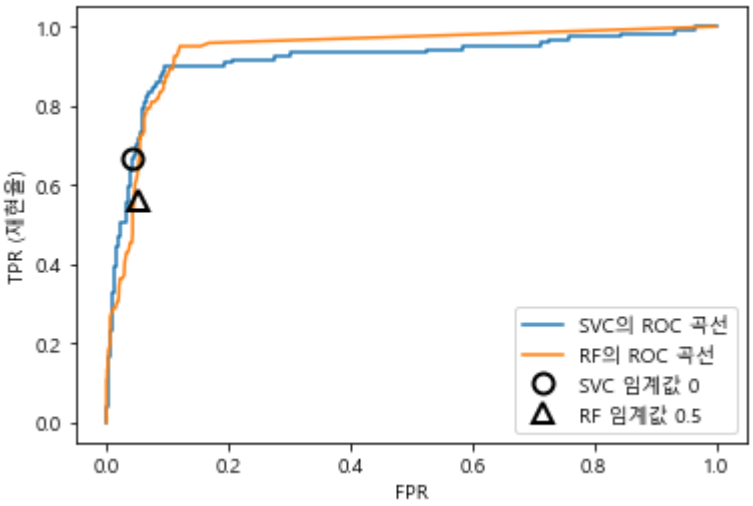


In [16]:

```
from sklearn.metrics import roc_curve
fpr_rf, tpr_rf, thresholds_rf = roc_curve(y_test, rf.predict_proba(X_test)[:, 1])

plt.plot(fpr, tpr, label="SVC의 ROC 곡선")
plt.plot(fpr_rf, tpr_rf, label="RF의 ROC 곡선")

plt.xlabel("FPR")
plt.ylabel("TPR (재현율)")
plt.plot(fpr[close_zero], tpr[close_zero], 'o', markersize=10,
         label="SVC 임계값 0", fillstyle="none", c='k', mew=2)
close_default_rf = np.argmin(np.abs(thresholds_rf - 0.5))
plt.plot(fpr_rf[close_default_rf], tpr[close_default_rf], '^', markersize=10,
         label="RF 임계값 0.5", fillstyle="none", c='k', mew=2)

plt.legend(loc=4)
```

Out[16]:

```
<matplotlib.legend.Legend at 0x1ae840be460>
```



In [17]:

```
from sklearn.metrics import roc_auc_score
rf_auc = roc_auc_score(y_test, rf.predict_proba(X_test)[:, 1])
svc_auc = roc_auc_score(y_test, svc.decision_function(X_test))
print("랜덤 포레스트의 AUC: {:.3f}".format(rf_auc))
print("SVC의 AUC: {:.3f}".format(svc_auc))
```

```
랜덤 포레스트의 AUC: 0.937
SVC의 AUC: 0.916
```

In [ ]: