모델 평가

- 정밀도-재현율 곡선과 ROC 곡선

한글 사전 설정

In [1]:

```
### 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 [2]:

```
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 [3]:

```
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 [4]:

```
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 [5]:

```
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
```

정밀도 재현율 곡선 확인

In [9]:

```
from sklearn.metrics import precision_recall_curve

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

In [10]:

```
# 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)
```

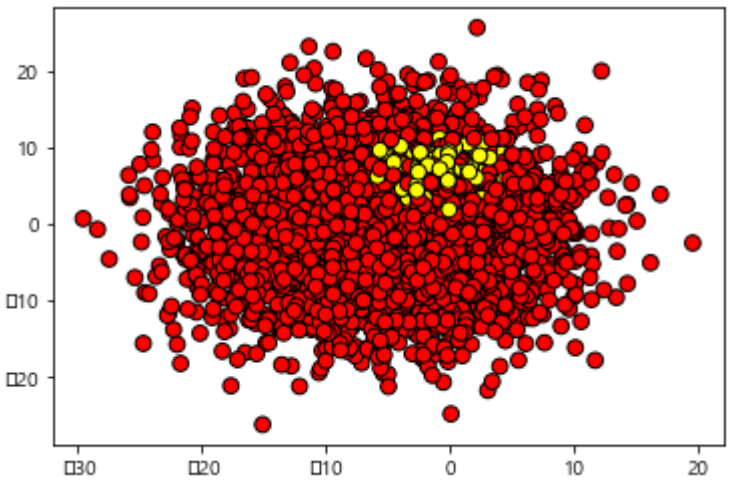
```
(4500, 2) (4500,)

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 [24]:

```
plt.scatter(X[:,0], X[:,1],
            c=y,
            cmap=plt.cm.autumn, s=60, edgecolors='k')  # plt.cm.Blues, RdYlGn, BrBG, Greens, RdGy, Y
```

Out[24]:

```
<matplotlib.collections.PathCollection at 0x1c99995bc40>
```



In [27]:

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

pred = svc.decision_function(X_test)  #  0의 값을 기준으로 분포
print(pred[0:10])
```

```
[-1.12256809  0.86782231 -0.14655591 -1.11330437 -1.11378355 -1.14388862
 -0.89066641 -0.99969575 -1.21851181 -1.25254411]
```

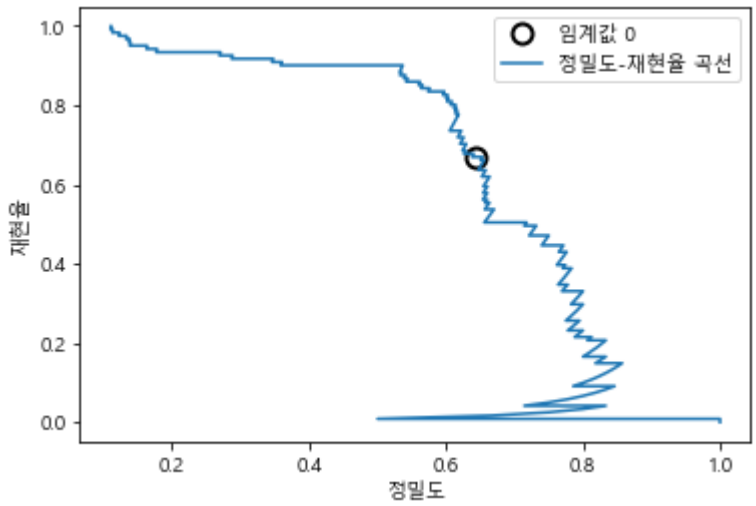
In [28]:

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

# 0에 가까운 임계값을 찾습니다
close_zero = np.argmin(np.abs(thresholds))
print(close_zero)

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")
```

964

Out[28]:

<matplotlib.legend.Legend at 0x1c9999edd60>



- 재현율(recall, sensitivity-민감도, Tprate)
  - 실제 양성 데이터를 양성으로 잘 예측
  - TP/(TP + FN)
- ==========

- FPRate
  - 실제 음성(0) 데이터를 음성으로 잘 예측
  - 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 [29]:

```
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[29]:

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

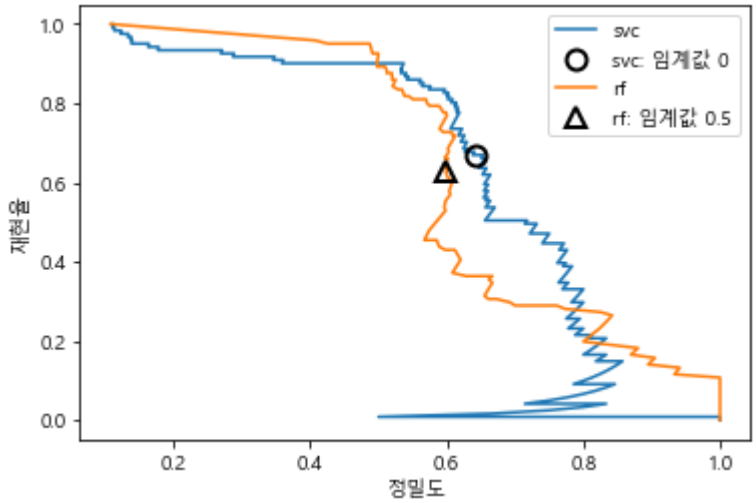
In [30]:

```
# 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[30]:

<matplotlib.legend.Legend at 0x1c999af2b50>



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

In [32]:

```
from sklearn.metrics import f1_score

rf_f1score = f1_score(y_test, rf.predict(X_test) )
svc_f1score = f1_score(y_test, svc.predict(X_test))

print("랜덤 포레스트의 f1_score: {:.3f}".format(rf_f1score))
print("svc의 f1_score: {:.3f}".format(svc_f1score))
```

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

In [33]:

```
from sklearn.metrics import average_precision_score

## 확률 예측
rf_pro = rf.predict_proba(X_test)[:, 1]
svc_dcfun = svc.decision_function(X_test)

ap_rf = average_precision_score(y_test, rf_pro)
ap_svc = average_precision_score(y_test, svc_dcfun)

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*

- y축은 Tprate(재현율, 민감도), x축은 Fprate라고 한다.
- 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 [35]:

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

임계값에 따른 각각의 Fprate, Tprate를 구하기

In [39]:

```
fpr.shape, tpr.shape, thresholds
```

Out[39]:

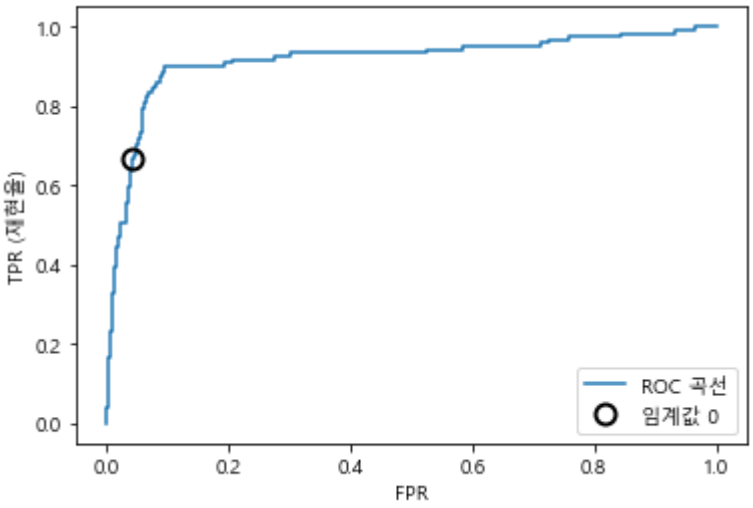
```
((121,),
 (121,),
 array([ 2.4675732 ,  1.4675732 ,  1.44543522,  1.3883083 ,  1.36399692,
         1.32045274,  1.30793345,  1.27362292,  1.26639366,  1.26549139,
         1.26218681,  1.23639476,  1.23272763,  1.23268814,  1.21274253,
         1.18867389,  1.17585471,  1.15089351,  1.14644476,  1.11697474,
         1.11164916,  1.07039562,  1.0618427 ,  1.02737422,  1.0221831 ,
         0.92622847,  0.92061278,  0.91871812,  0.91025922,  0.82556742,
         0.80058895,  0.78826384,  0.76431489,  0.73658061,  0.70361806,
         0.63202742,  0.61868639,  0.58699872,  0.47424583,  0.43174875,
         0.42161182,  0.41674789,  0.41170002,  0.41068324,  0.40404247,
         0.39698761,  0.37952465,  0.37283557,  0.36963527,  0.23779958,
         0.210375  ,  0.19734035,  0.16642743,  0.08952996,  0.06954494,
         0.0431748 , -0.04779208, -0.06239381, -0.08749885, -0.09744136,
        -0.11308646, -0.13793376, -0.14655591, -0.16681464, -0.23766011,
        -0.26421748, -0.28671312, -0.35207398, -0.35960512, -0.36357768,
        -0.38686615, -0.42239029, -0.43253037, -0.44822521, -0.50217567,
        -0.50567587, -0.5180301 , -0.5297265 , -0.53771063, -0.54150651,
        -0.58261387, -0.60168198, -0.64087279, -0.66798219, -0.72601863,
        -0.74522837, -0.76985051, -0.7837385 , -0.89066641, -0.90555299,
        -0.91428783, -0.92168496, -0.94995579, -0.96549528, -0.96757492,
        -0.97822999, -1.02946706, -1.02966988, -1.03764999, -1.03768168,
        -1.08420087, -1.08576068, -1.1047158 , -1.10500358, -1.12097238,
        -1.12114269, -1.13095237, -1.13145208, -1.16822306, -1.16860214,
        -1.17301962, -1.17324742, -1.19024128, -1.19055089, -1.22841838,
        -1.2288246 , -1.28439408, -1.28869488, -1.41801331, -1.42644669,
        -1.95625342]))
```

In [41]:

```
plt.plot(fpr, tpr, label="ROC 곡선")
plt.xlabel("FPR")
plt.ylabel("TPR (재현율)")

# 임계값이 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[41]:

```
<matplotlib.legend.Legend at 0x1c999b86e80>
```

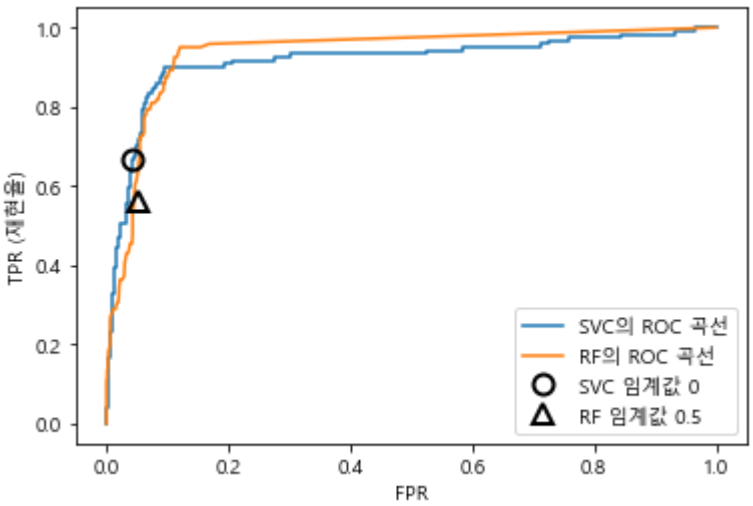


In [42]:

```
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[42]:

```
<matplotlib.legend.Legend at 0x1c999cbe100>
```



In [43]:

```
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
```

REF

- plt.cm.___ : https://chrisalbon.com/python/basics/set_the_color_of_a_matplotlib/ (https://chrisalbon.com/python/basics/set_the_color_of_a_matplotlib/)

In [ ]: