모델 평가 - 평가지표 알아보기

학습 내용

- 타이타닉 데이터 셋을 활용하여 평가지표에 대해 알아봅니다.
- 정밀도, 민감도 곡선을 그려봅니다.
- 평균 정밀도 값을 구해봅니다.
- ROC 커브와 AUC값을 구해봅니다.

목차

01. 데이터 준비 및 라이브러리 임포트

목차로 이동하기

한글 설정

In [33]:

```
import matplotlib
from matplotlib import font_manager, rc
import platform
import warnings
warnings.filterwarnings(action='ignore')
```

In [34]:

```
### 한글
path = "C:/Windows/Fonts/malgun.ttf"
if platform.system() == "Windows":
    font_name = font_manager.FontProperties(fname=path).get_name()
    rc('font', family=font_name)
elif platform.system()=="Darwin":
    rc('font', family='AppleGothic')
else:
    print("Unknown System")

matplotlib.rcParams['axes.unicode_minus'] = False
```

In [45]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

from sklearn.model_selection import train_test_split
```

In [46]:

```
train = pd.read_csv("./data/titanic/train.csv")
test = pd.read_csv("./data/titanic/test.csv")
sub = pd.read_csv("./data/titanic/gender_submission.csv")

train.shape, test.shape, sub.shape
```

Out[46]:

```
((891, 12), (418, 11), (418, 2))
```

데이터 준비

In [47]:

```
train.columns
```

Out[47]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [48]:

```
train.head()
```

Out[48]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

기본 데이터 전처리

In [49]:

```
train.isnull().sum(), test.isnull().sum()
```

Out[49]:

```
(PassengerId      0
 Survived         0
 Pclass           0
 Name             0
 Sex              0
 Age            177
 SibSp            0
 Parch            0
 Ticket           0
 Fare             0
 Cabin          687
 Embarked         2
 dtype: int64,
 PassengerId      0
 Pclass           0
 Name             0
 Sex              0
 Age             86
 SibSp            0
 Parch            0
 Ticket           0
 Fare             1
 Cabin          327
 Embarked         0
 dtype: int64)
```

In [50]:

```
print( train['Embarked'].value_counts() )
```

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

Fare, Embarked 처리

In [51]:

```
train.loc[ train['Embarked'].isnull(), 'Embarked' ] = 'S'
test['Fare'] = test['Fare'].fillna( test['Fare'].median()  )
```

In [52]:

```
train.isnull().sum(), test.isnull().sum()
```

Out[52]:

```
(PassengerId      0
 Survived         0
 Pclass           0
 Name             0
 Sex              0
 Age            177
 SibSp            0
 Parch            0
 Ticket           0
 Fare             0
 Cabin          687
 Embarked         0
 dtype: int64,
 PassengerId      0
 Pclass           0
 Name             0
 Sex              0
 Age             86
 SibSp            0
 Parch            0
 Ticket           0
 Fare             0
 Cabin          327
 Embarked         0
 dtype: int64)
```

레이블 인코딩

In [53]:

```
train.head()
```

Out[53]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

In [54]:

```
cat_Sex = {'male':1, 'female':2}
train['Sex'] = train['Sex'].map(cat_Sex).astype("int32")
test['Sex'] = test['Sex'].map(cat_Sex).astype("int32")
train.head()
```

Out[54]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cal
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	N
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	2	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	2	26.0	0	0	STON/O2. 3101282	7.9250	N
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	2	35.0	1	0	113803	53.1000	C1
4	5	0	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.0500	N

In [55]:

```
cat_Embarked = {'S':1, 'C':2, 'Q':3}
train['Embarked'] = train['Embarked'].map(cat_Embarked)
test['Embarked'] = test['Embarked'].map(cat_Embarked)

train.head()
```

Out[55]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cal
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	N
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	2	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	2	26.0	0	0	STON/O2. 3101282	7.9250	N
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	2	35.0	1	0	113803	53.1000	C1
4	5	0	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.0500	N

In [57]:

```
train.columns
```

Out[57]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [62]:

```
sel = ['Pclass', 'Sex', 'SibSp', 'Parch', 'Fare', 'Embarked']

X_tr = train[sel]
y_tr = train['Survived']

X_last_test = test[sel]

X_train, X_test, y_train , y_test = train_test_split(X_tr, y_tr, stratify=y_tr, random_state=0, tes
```

In [ ]:

```
```

02 각 모델별 정밀도,민감도, F1-score구해보기

목차로 이동하기

In [63]:

```
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

In [64]:

```
from sklearn.metrics import classification_report
```

In [65]:

```
svc = SVC(gamma=.05).fit(X_train, y_train)
tree = DecisionTreeClassifier().fit(X_train, y_train)
rf = RandomForestClassifier().fit(X_train, y_train)
```

In [66]:

```
pred = svc.predict(X_test)
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           0       0.76      0.81      0.78       110
           1       0.66      0.59      0.63        69

    accuracy                           0.73       179
   macro avg       0.71      0.70      0.71       179
weighted avg       0.72      0.73      0.72       179
```

In [67]:

```
pred = tree.predict(X_test)
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.81      0.80       110
           1       0.68      0.65      0.67        69

    accuracy                           0.75       179
   macro avg       0.73      0.73      0.73       179
weighted avg       0.75      0.75      0.75       179
```

In [68]:

```
pred = rf.predict(X_test)
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.77      0.78       110
           1       0.64      0.65      0.65        69

    accuracy                           0.73       179
   macro avg       0.71      0.71      0.71       179
weighted avg       0.73      0.73      0.73       179
```

- f1-score는 의사결정트리 모델이 0.67로 가장 좋다.

03 모델별 정밀도, 재현율 곡선을 그려보기

목차로 이동하기

In [90]:

```
from sklearn.metrics import precision_recall_curve
```

In [91]:

```
svc = SVC(gamma=.05).fit(X_train, y_train)
pred_svc = svc.decision_function(X_test)  #  0의 값을 기준으로 분포
print(pred[0:10])
```

```
[0.72       0.27640421 0.65       0.27640421 1.         0.
 0.28250037 0.84812157 0.2        0.09069573]
```

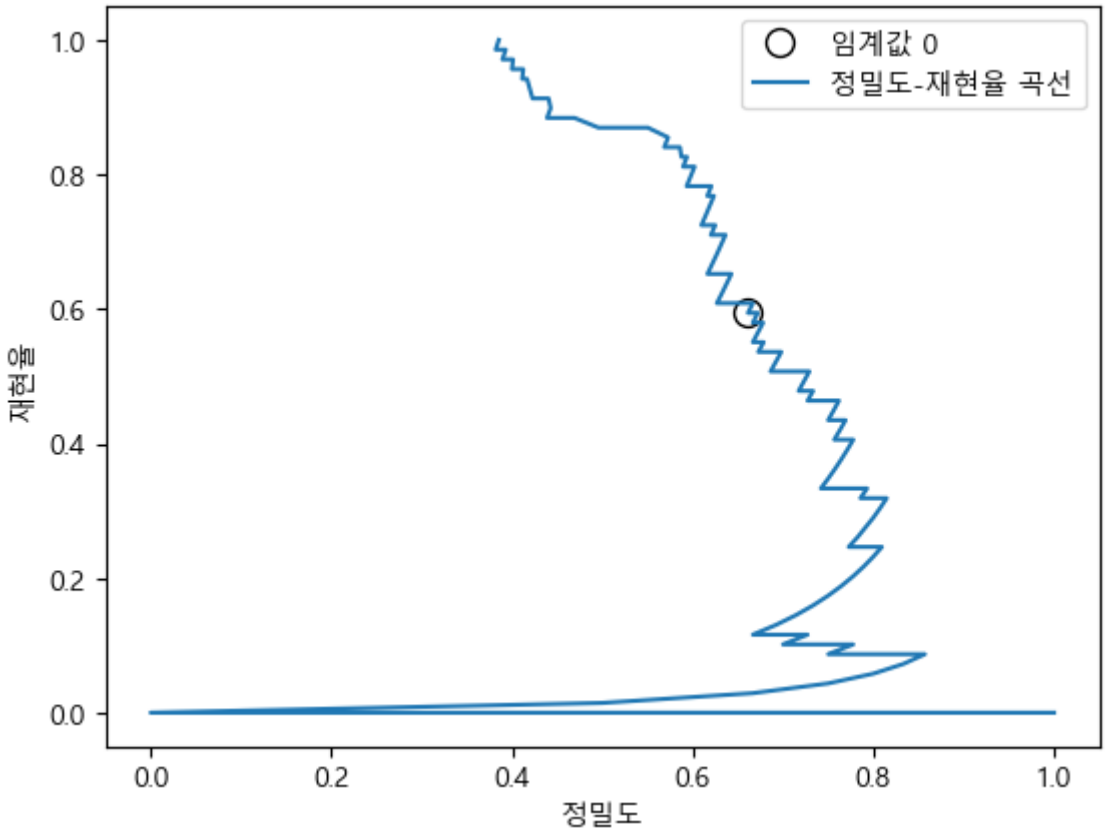
In [92]:

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

print("임계값 : ", thresholds.min(),  thresholds.max())

# 0에 가까운 임계값을 찾습니다
close_zero = np.argmin(np.abs(thresholds))  # thresholds의 절대값이 가장 작은 것(위치)
print(close_zero)

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

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

```
임계값 :  -1.1620575738476628 1.2594963931265546
74
```

Out[92]:

```
<matplotlib.legend.Legend at 0x2346e180880>
```



In [93]:

```
rf = RandomForestClassifier().fit(X_train, y_train)
pred_rf = rf.predict_proba(X_test)[:, 1]  #  0의 값을 기준으로 분포
print(pred[0:10])
```

```
[0.72       0.27640421 0.65       0.27640421 1.         0.
 0.28250037 0.84812157 0.2        0.09069573]
```

In [94]:

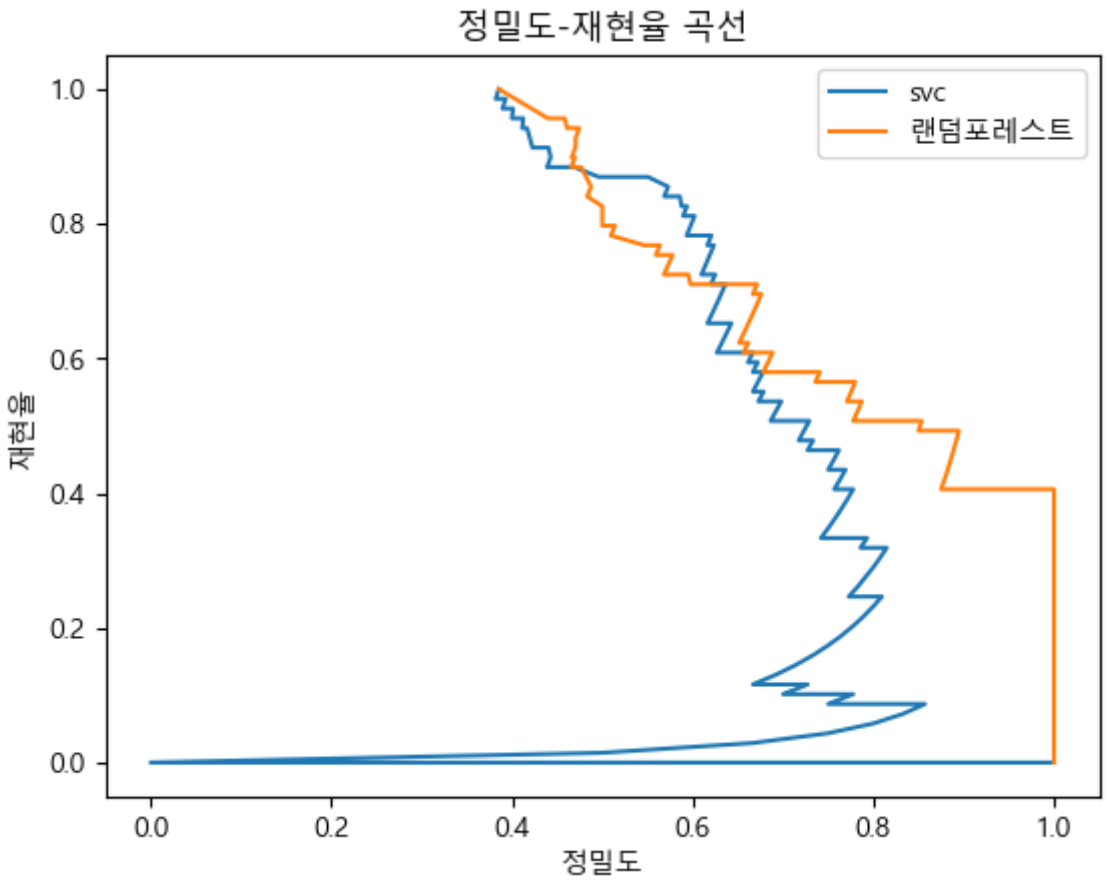
```
# precision, recall, thresholds = precision_recall_curve(y_test, pred_svc)
precision_rf, recall_rf, thresholds_rf = precision_recall_curve(y_test, pred_rf)
```

In [95]:

```
plt.plot(precision, recall, label="svc")
plt.plot(precision_rf, recall_rf, label="랜덤포레스트")

plt.title("정밀도-재현율 곡선")
plt.xlabel("정밀도")
plt.ylabel("재현율")
plt.legend(loc="best")
```

Out[95]:

```
<matplotlib.legend.Legend at 0x2346dfd9a60>
```



In [97]:

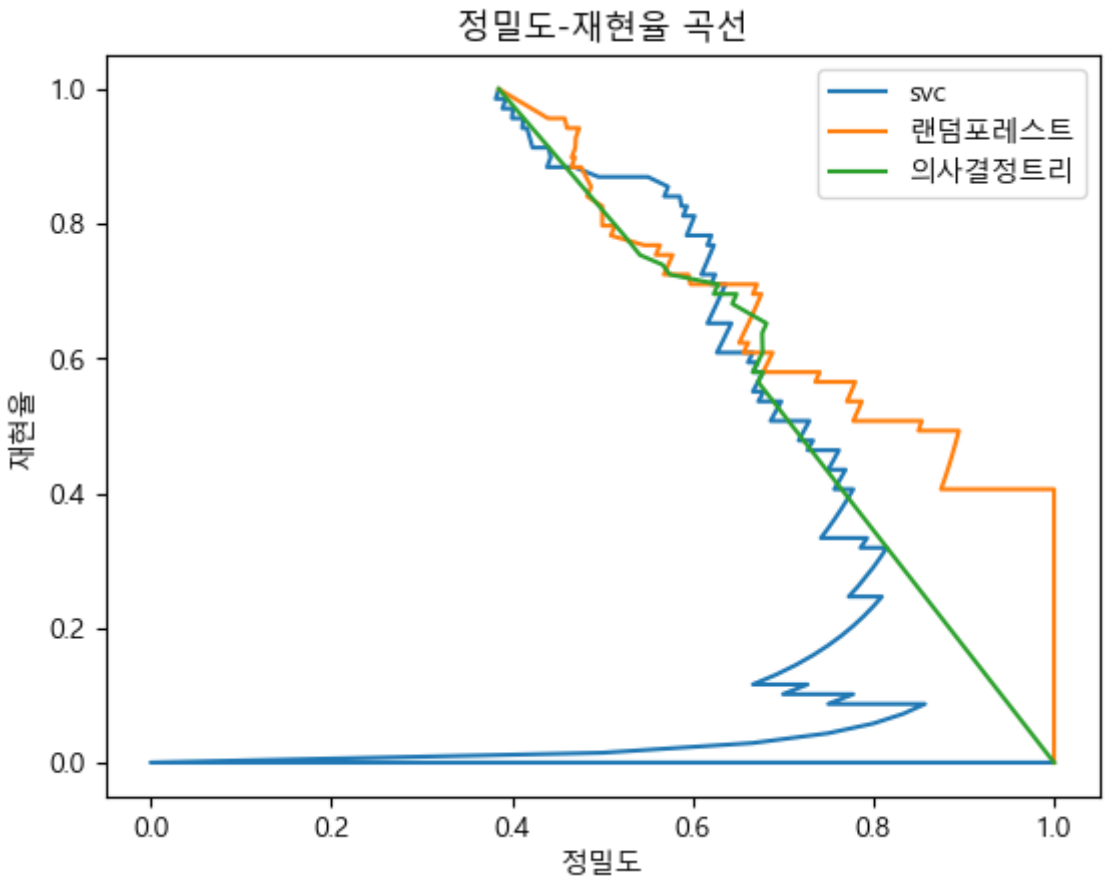
```
tree = DecisionTreeClassifier().fit(X_train, y_train)
pred_tree = tree.predict_proba(X_test)[:, 1]  #  0의 값을 기준으로 분포

precision_dt, recall_dt, thresholds_dt = precision_recall_curve(y_test, pred_tree)
```

In [98]:

```
plt.plot(precision, recall, label="svc")
plt.plot(precision_rf, recall_rf, label="랜덤포레스트")
plt.plot(precision_dt, recall_dt, label="의사결정트리")

plt.title("정밀도-재현율 곡선")
plt.xlabel("정밀도")
plt.ylabel("재현율")
plt.legend(loc="best")
```

Out[98]:

```
<matplotlib.legend.Legend at 0x2346e215880>
```



04 모델별 F1-score를 구하기

목차로 이동하기

In [99]:

```
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))
tree_f1score = f1_score(y_test, tree.predict(X_test))

print("랜덤 포레스트의 f1_score: {:.3f}".format(rf_f1score))
print("svc의 f1_score: {:.3f}".format(svc_f1score))
print("의사결정트리의 f1_score: {:.3f}".format(tree_f1score))
```

랜덤 포레스트의 f1_score: 0.657
svc의 f1_score: 0.626
의사결정트리의 f1_score: 0.667

05 평균 정밀도 구하기

목차로 이동하기

- 모델을 비교하기 위한 방법 중의 하나로 정밀도-재현율 곡선의 아랫부분 면적을 이용
  - 이 면적을 평균 정밀도(average precision)이라 한다.
  - 함수로 average_precision_score함수로 **평균 정밀도 계산**이 가능하다.

In [101]:

```
from sklearn.metrics import average_precision_score

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

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

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

랜덤 포레스트의 평균 정밀도: 0.780
의사결정트리의 평균 정밀도: 0.600
svc의 평균 정밀도: 0.669

06 ROC 커브 그려보기

목차로 이동하기

In [102]:

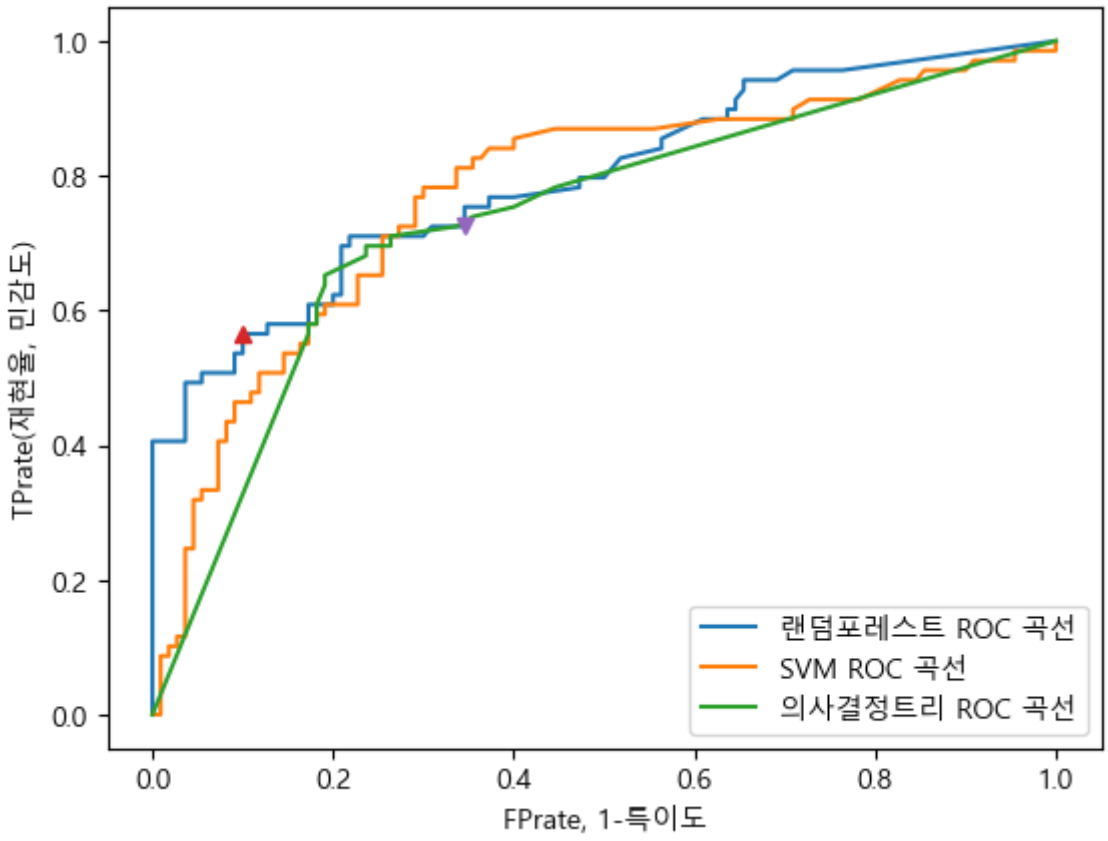
```
from sklearn.metrics import roc_curve

fpr_svc, tpr_svc, thresholds_svc = roc_curve(y_test, svc.decision_function(X_test)   )
fpr_rf, tpr_rf, thresholds_rf = roc_curve(y_test, rf.predict_proba(X_test)[:, 1]   )
fpr_dt, tpr_dt, thresholds_dt = roc_curve(y_test, tree.predict_proba(X_test)[:, 1]  )
```

In [114]:

```
plt.plot(fpr_rf, tpr_rf, label="랜덤포레스트 ROC 곡선")
plt.plot(fpr_svc, tpr_svc, label="SVM ROC 곡선")
plt.plot(fpr_dt, tpr_dt, label="의사결정트리 ROC 곡선")

plt.xlabel("FPrate, 1-특이도")
plt.ylabel("TPrate(재현율, 민감도)")
plt.legend(loc=4)

# 랜덤포레스트 임계값이 0.7일때의 위치
close_05_rf = np.argmin(np.abs(thresholds_rf - 0.7))
plt.plot(fpr_rf[close_05_rf], tpr_rf[close_05_rf], '^', label="RF 임계값 0.7")

# 랜덤포레스트 임계값이 0.3일때의 위치
close_05_rf = np.argmin(np.abs(thresholds_rf - 0.3))
plt.plot(fpr_rf[close_05_rf], tpr_rf[close_05_rf], 'v', label="RF 임계값 0.3")
```

Out[114]:

```
[<matplotlib.lines.Line2D at 0x2346f80b7f0>]
```



07 AUC 값 구하기

목차로 이동하기

In [116]:

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

```
랜덤 포레스트의 AUC: 0.796
의사결정트리의 AUC: 0.738
SVC의 AUC: 0.773
```

- AUC의 값이 가장 높은 모델은 랜덤 포레스트 모델입니다.

정리

- F1-score의 값이 가장 큰 모델은 0.667로 의사결정트리가 가장 좋습니다.
- 평균 정밀도 값은 랜덤 포레스트가 가장 좋습니다.
  - 랜덤 포레스트의 평균 정밀도: 0.780
  - 의사결정트리의 평균 정밀도: 0.600
  - svc의 평균 정밀도: 0.669
- AUC의 값이 1에 가장 가까운 모델은 0.797으로 랜덤 포레스트 모델입니다.
- 모든 경우에 좋다고 말할 수 없지만, 현재로서 가장 좋은 모델로 확인되는 것은 랜덤 포레스트 모델이라 말할 수 있습니다.