

강원지역혁신플랫폼

기계학습

Machine Learning



ROC 곡선 및 모델 평가 실습



▶ 학습목표

📁 ROC 곡선의 개념을 이해하고, 모델 평가를 구현할 수 있습니다.





01 | ML 폴더

◆ ML 폴더를 클릭하기

jupyter

QuitLogout

FilesRunningClusters

Select items to perform actions on them.

UploadNew↺

0 ▾ /

Name ▾Last ModifiedFile size

<input type="checkbox"/>	3D Objects	일 년 전	
<input type="checkbox"/>	anaconda3	7달 전	
<input type="checkbox"/>	Contacts	9달 전	
<input type="checkbox"/>	Desktop	4달 전	
<input type="checkbox"/>	Documents	6분 전	
<input type="checkbox"/>	Downloads	2시간 전	
<input type="checkbox"/>	Favorites	9달 전	
<input type="checkbox"/>	ML	22분 전	
<input type="checkbox"/>	Links	9달 전	
<input type="checkbox"/>	Music	9달 전	
<input type="checkbox"/>	OneDrive	일 년 전	
<input type="checkbox"/>	Pictures	9달 전	
<input type="checkbox"/>	Saved Games	9달 전	
<input type="checkbox"/>	scikit_learn_data	8달 전	
<input type="checkbox"/>	seaborn-data	3달 전	
<input type="checkbox"/>	Searches	3달 전	
<input type="checkbox"/>	Videos	9달 전	
<input type="checkbox"/>	Untitled.ipynb	4달 전	1.64 kB



02 | ch07 폴더

◆ ch07 폴더 클릭하기

jupyter

QuitLogout

FilesRunningClusters

Select items to perform actions on them.

UploadNew↺

<input type="checkbox"/> 0 ▾	▾ /	Name ▾	Last Modified	File size
<input type="checkbox"/>	ch00		9일 전	
<input type="checkbox"/>	ch03		5일 전	
<input type="checkbox"/>	ch04		4일 전	
<input type="checkbox"/>	ch05		2일 전	
<input type="checkbox"/>	ch06		몇 초 전	
<input type="checkbox"/>	ch07		몇 초 전	



03 | ch07_02_ROC곡선및모델평가실습.ipynb

★ ch07_02_ROC곡선및모델평가실습.ipynb 파일 클릭하기

Files

Running


OpenDownloadRenameDuplicateDelete

/ ch07 /

☐


Name

☐



ch07_01_모델일반화전략.ipynb

☐



ch07_02_ROC곡선및모델평가실습h.ipynb



04 | ROC 곡선

ROC 곡선(Receiver Operating Characteristic Curve)

△ ROC 곡선은 클래스 판별 기준 값의 변화에 따른 x축에 위양성율(fall-out), y축에 재현율(Recall)의 변화를 시각화한 것임

◆ ROC 곡선은 x축과 y축의 값으로 그래프를 그림

➤ x축 : FPR(False Positive Rate, 위양성율) $FP\ Rate = \frac{FP}{FP + TN}$

➤ y축 : TPR(True Positive Rate, 진양성율) $TP\ Rate = \frac{TP}{TP + FN}$

		예측 조건(Predicted condition)	
		예측 조건 긍정 (Predicted Condition positive)	예측 조건 부정 (Predicted Condition negative)
실제 조건 (True Condition)	조건 긍정 (Condition positive)	참 긍정 (True Positive, TP), 검정력(Power)	거짓 부정 (False Negative, FN), 제2종 오류(Type II Error)
	조건 부정 (Condition negative)	거짓 긍정 (False Positive, FP), 제1종 오류(Type I Error)	참 부정 (True Negative, TN)



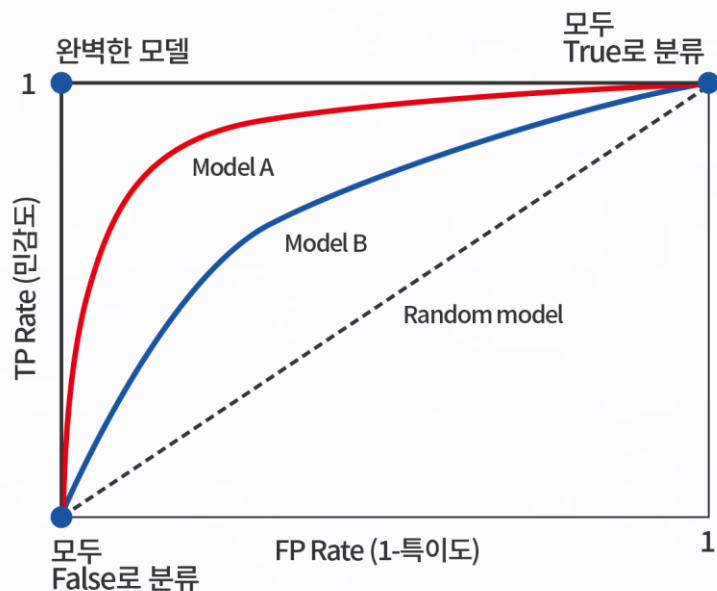
04 | ROC 곡선

△ FPR은 위양성율(fall-out), TPR은 재현율(Recall)을 의미함

◆ ROC 곡선은 아래와 같이 x축과 y축의 값으로 그래프를 그림

➤ x축 : FPR(False Positive Rate, 위양성율) $FP\ Rate = \frac{FP}{FP + TN}$

➤ y축 : TPR(True Positive Rate, 진양성율) $TP\ Rate = \frac{TP}{TP + FN}$



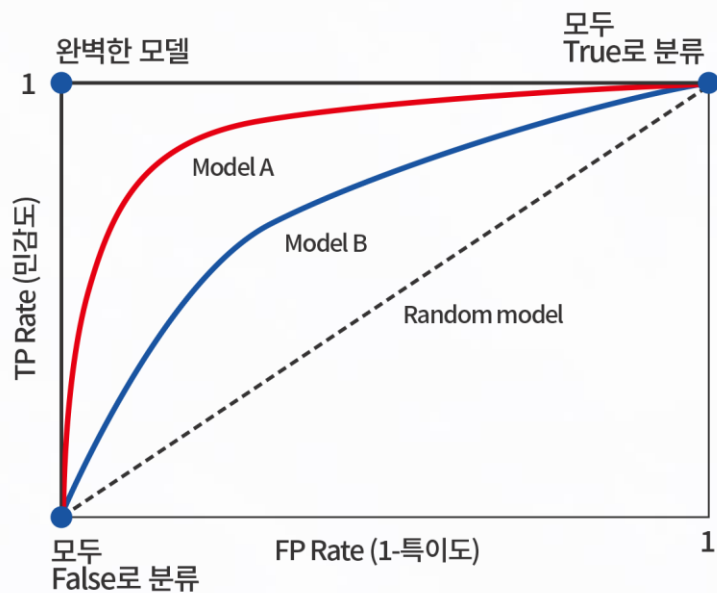
ROC 곡선



04 | ROC 곡선

△ ROC 곡선은 분류자(classifier)를 점으로 표시한 2차원 그래프임

- ◆ x축에는 FP Rate(1-특이도)
- ◆ y축에는 TP Rate(재현율 또는 민감도)
- ◆ 이 두 평가 값의 관계로 모델을 평가함



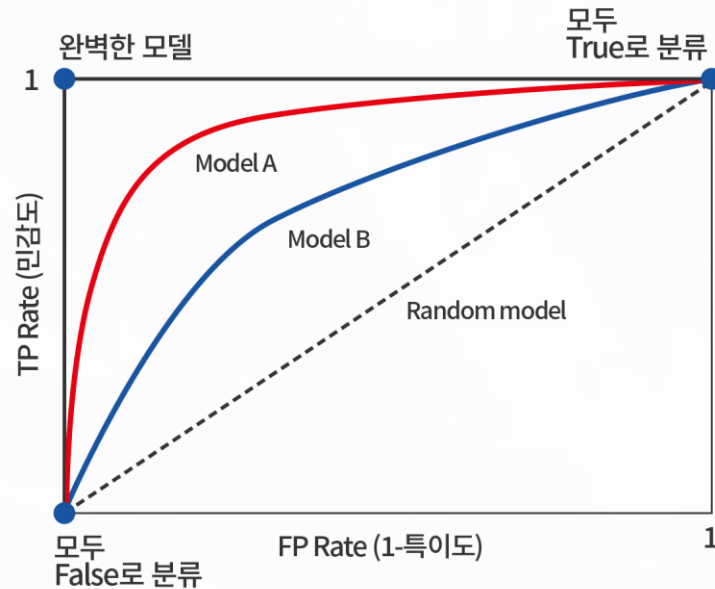
ROC 곡선



04 | ROC 곡선

△ ROC 곡선은 **모델의 분류, 계층 확률 추정, 점수화 성능**을 시각적으로 보여주기 위해 널리 **사용하는 도구**임

◆ ROC 곡선의 보면, **여러 분류자들의 수익**(진양성, TP)과 **비용**(위양성, FP)을 **한눈에 보고 장·단점을 비교**할 수 있음



ROC 곡선

$$\text{(수익)진양성 } TP Rate = \frac{TP}{TP+FN}$$

$$\text{(비용)위양성 } FP Rate = \frac{FP}{FP+TN}$$



04 | ROC 곡선

⚡ 클래스 판별 기준값이 낮을수록 양성(positive)으로 예측하는 수가 증가함

◆ 예를 들어 **setosa** 일 확률이 **0.6** (60%) 일 때를 생각해 보자.

‣ 기준 값을 **0.55** (55%)라고 하면 양성(positive)으로 예측함

‣ 기준 값을 **0.65** (65%)로 높이면 음성(negative)으로 예측이 바뀜



04 | ROC 곡선

△ 진양성율(재현율) **TPR**의 분모($TP+FN$)는 실제 양성(positive)인 표본 개수이므로 항상 고정임

$$TP Rate = \frac{TP}{TP + FN}$$

◆ 위양성율(1-특이도) **FPR**의 분모($FP+TN$)도 실제 음성(negative)인 표본 개수이므로 항상 고정임

$$FP Rate = \frac{FP}{FP + TN}$$

		예측 조건(Predicted condition)	
		예측 조건 긍정 (Predicted Condition positive)	예측 조건 부정 (Predicted Condition negative)
실제 조건 (True Condition)	조건 긍정 (Condition positive)	참 긍정 (True Positive, TP), 검정력(Power)	거짓 부정 (False Negative, FN), 제2종 오류(Type II Error)
	조건 부정 (Condition negative)	거짓 긍정 (False Positive, FP), 제1종 오류(Type I Error)	참 부정 (True Negative, TN)



04 | ROC 곡선

△ 양성 (positive) 예측은 다음과 같음

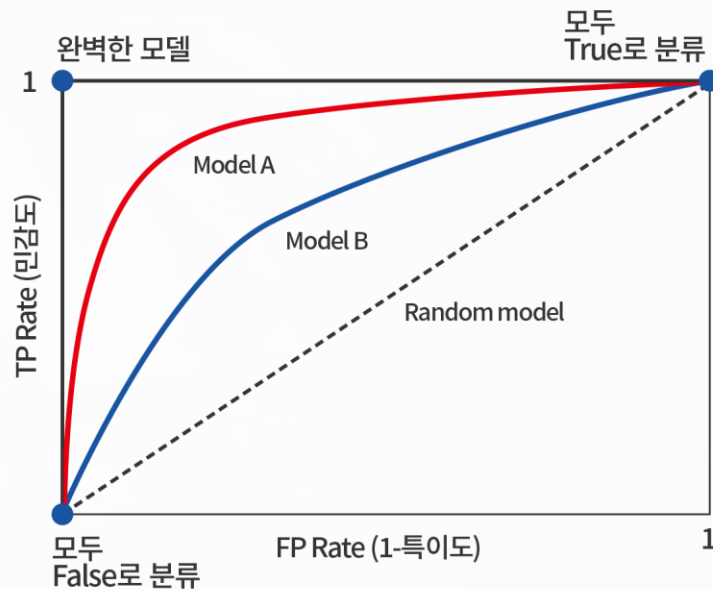
$$\text{양성 예측} = TP + FP$$

◆ 분모가 고정인 상태에서 양성(positive) 예측이 증가하면 일반적으로 TP와 FP도 동시에 증가함

➢ 진양성율(재현율)과 위양성율(1-특이도)은 양의 상관관계가 있음

$$TP\ Rate = \frac{TP}{TP + FN}$$

$$FP\ Rate = \frac{FP}{FP + TN}$$



ROC 곡선



05 | 모델 평가 실습 (1)



모델 평가 실습 (1)

△ 다음은 아이리스(iris) 데이터셋으로 모델 평가 실습을 진행해 보자.

◆ 아이리스 데이터 셋은 꽃잎의 각 부분의 너비와 길이 등을 측정한 데이터임

➢ 관측치는 150개, 속성은 6개로 구성되어 있음

➢ 아이리스 꽃은 아래 그림과 같음

열이름	설명
Caseno	일련번호
Sepal Length	꽃받침의 길이 정보
Sepal Width	꽃받침의 너비 정보
Petal Length	꽃잎의 길이 정보
Petal Width	꽃잎의 너비 정보
Species	꽃의 종류 정보 (setosa, versicolor, virginica)





05 | 모델 평가 실습 (1)

다음은 아이리스 데이터셋으로 의사결정나무 모델 성능 평가 결과를 혼동 행렬로 확인하는 코드이다.

실행결과에서 8개의 데이터가 잘못 분류된 것을 볼 수 있음

```
pred = model.predict(X_test)
confMatrix = confusion_matrix(y_test, pred)
print("Confusion Matrix : \n", confMatrix)
```

```
Confusion Matrix :
[[14  1  0]
 [ 0 13  2]
 [ 0  5 10]]
```

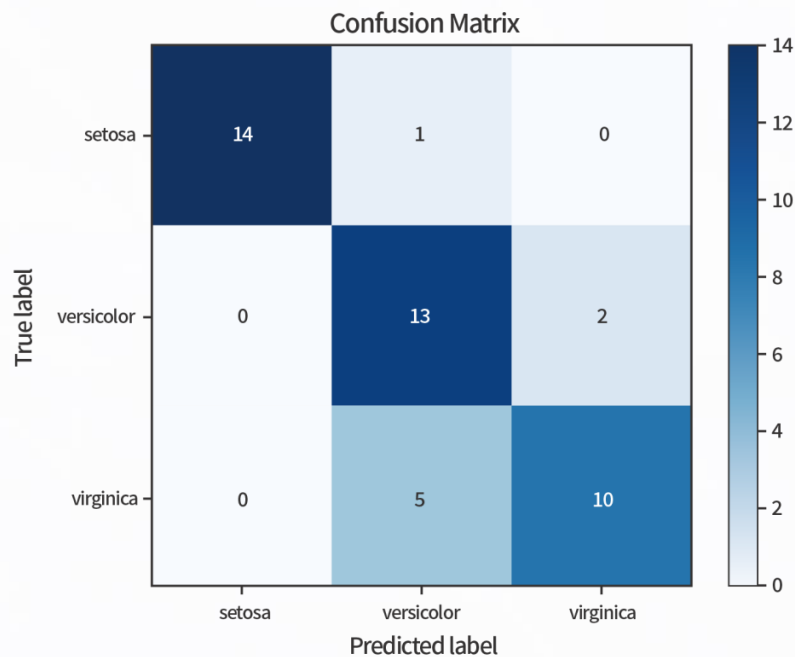


05 | 모델 평가 실습 (1)

다음은 아이리스 데이터셋으로 의사결정나무 모델 성능 평가 결과를 혼동 행렬 그래프로 출력하는 코드이다.

실행결과에서 8개의 데이터가 잘못 분류된 것을 볼 수 있음

```
pred = model.predict(X_test)
skplt.metrics.plot_confusion_matrix(y_test, pred)
plt.show()
```



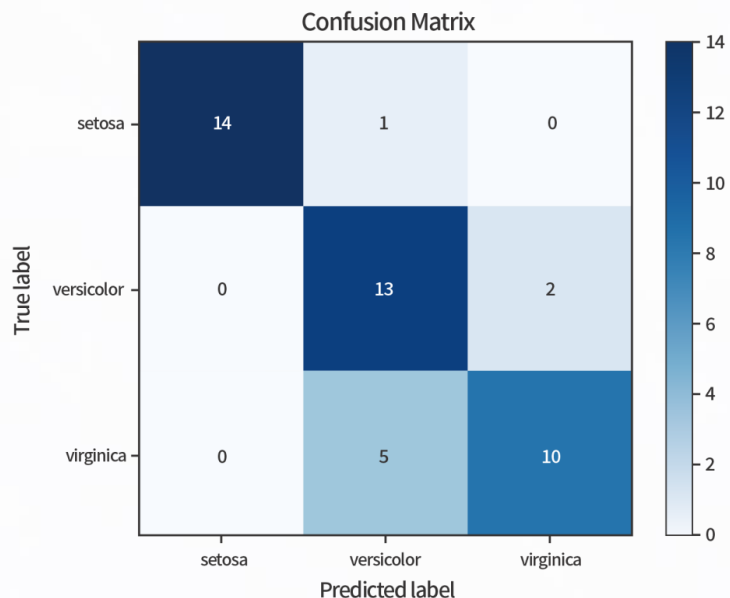


05 | 모델 평가 실습 (1)

다음은 아이리스 데이터 **훈련 데이터셋**으로 **의사결정나무 모델**을 **학습**하고,
테스트 데이터셋으로 **모델의 정확도**(accuracy)를 **계산**하는 코드이다.

◆ 실행결과에서 **정확도가 약 82.22%인 것**을 볼 수 있음

```
pred = model.predict(X_test)
acc = np.round(accuracy_score(y_test, pred), 4)
print("Accuracy : ", acc)                                # Accuracy : 0.8222
```



$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$Accuracy = \frac{14 + 13 + 10}{14 + 1 + 13 + 2 + 5 + 10} = \frac{37}{45} = 0.822$$



05 | 모델 평가 실습 (1)

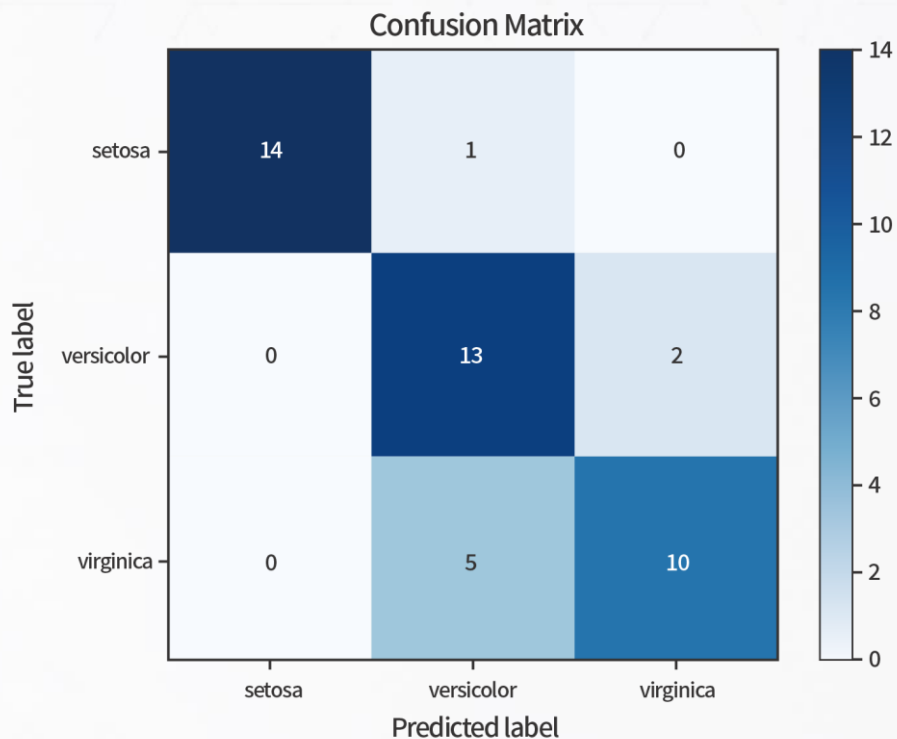
다음은 아이리스 데이터 **훈련 데이터셋**으로 **의사결정나무 모델**을 **학습**하고,
테스트 데이터셋으로 **모델의 정밀도**(precision)를 **계산**하는 코드이다.

◆ 실행결과에서 정밀도는 다음과 같음

➤ setosa: 1.0, versicolor: 0.684, virginica: 0.833

```
precisions = precision_score(y_test, model.predict(X_test), average=None)
for target, score in zip(iris_data.target_names, precisions):
    print(f"{target}의 정밀도: {score}")
# setosa의 정밀도: 1.0
# versicolor의 정밀도: 0.6842105263157895
# virginica의 정밀도: 0.8333333333333334
```

$$\begin{aligned} \text{Precision} &= \frac{TP}{TP + FP} & \text{versicolor} &= \frac{13}{1 + 13 + 5} = \frac{13}{19} = 0.684 \\ \text{setosa} &= \frac{14}{14} = 1.0 & \text{virginica} &= \frac{10}{10 + 2} = \frac{10}{12} = 0.833 \end{aligned}$$





05 | 모델 평가 실습 (1)

다음은 아이리스 데이터 **훈련 데이터셋**으로 **의사결정나무 모델**을 학습하고, **테스트 데이터셋**으로 **모델의 재현율**(recall)을 계산하는 코드이다.

◆ 실행결과에서 재현율은 다음과 같음

➤ setosa: 0.933, versicolor: 0.866, virginica: 0.666

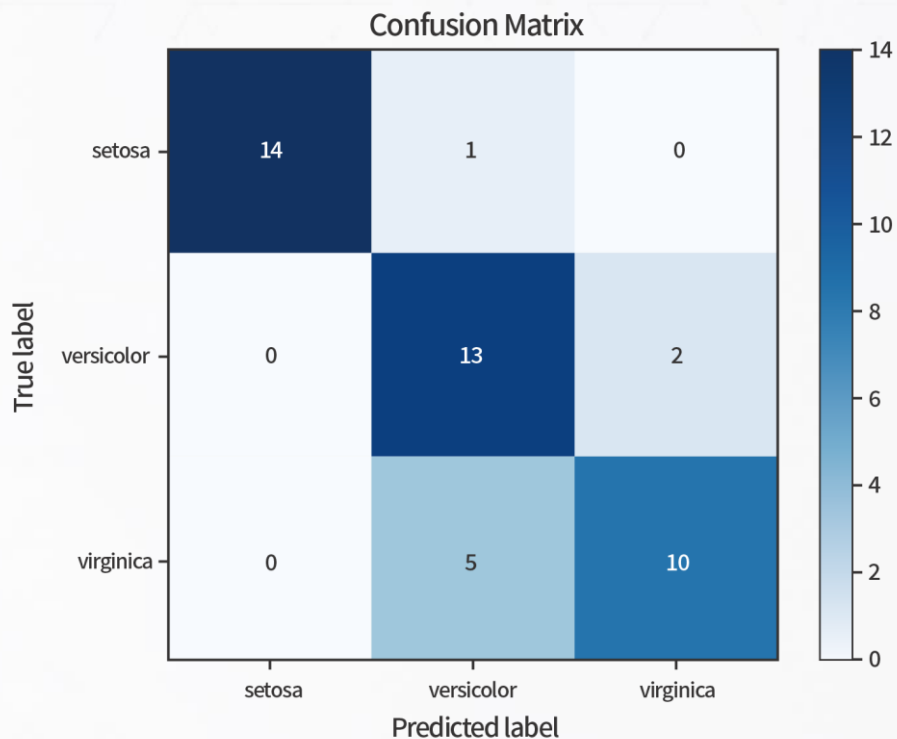
```
recalls = recall_score(y_test, model.predict(X_test), average=None)
for target, score in zip(iris_data.target_names, recalls):
    print(f"{target}의 재현율: {score}")
# setosa의 재현율: 0.9333333333333333
# versicolor의 재현율: 0.8666666666666667
# virginica의 재현율: 0.6666666666666666
```

$$TP\ Rate = \frac{TP}{TP + FN}$$

$$setosa = \frac{14}{14 + 1} = \frac{14}{15} = 0.933$$

$$versicolor = \frac{13}{13 + 2} = \frac{13}{15} = 0.866$$

$$virginica = \frac{10}{5 + 10} = \frac{10}{15} = 0.666$$





05 | 모델 평가 실습 (1)

다음은 아이리스 데이터 **훈련 데이터셋**으로 **의사결정나무 모델**을 **학습**하고,
테스트 데이터셋으로 **모델의 F1-점수**(f1-score)를 **계산**하는 코드이다.

◆ 실행결과에서 F1-점수는 다음과 같음

➤ setosa: 0.966, versicolor: 0.765, virginica: 0.741

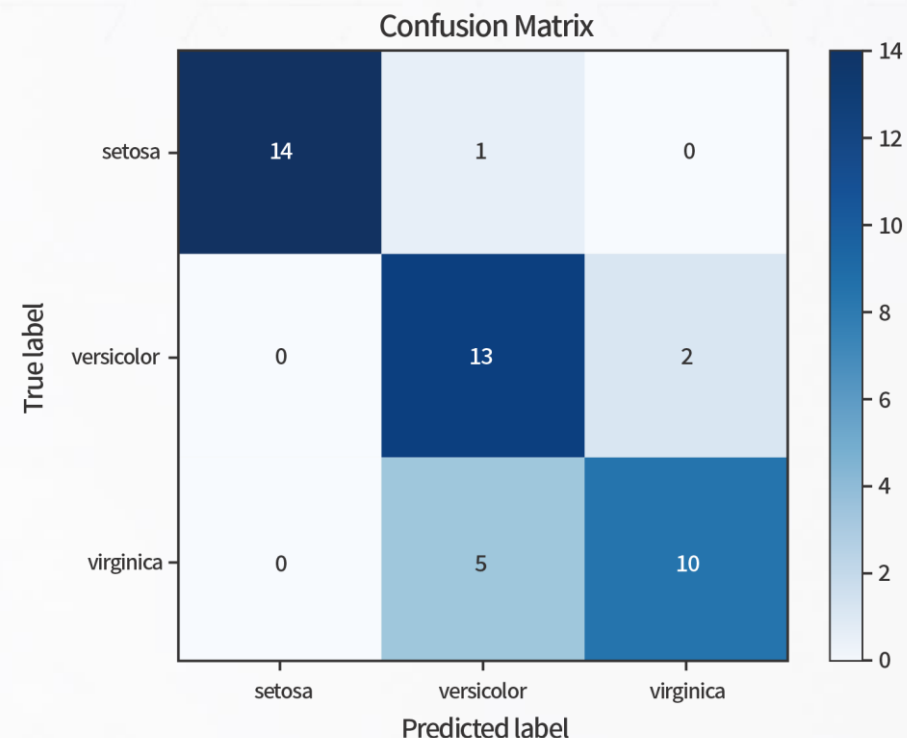
```
f1s = f1_score(y_test, model.predict(X_test), average=None)
for target, score in zip(iris_data.target_names, f1s):
    print(f"{target}의 f1-score: {score}")
# setosa의 f1-score: 0.9655172413793104
# versicolor의 f1-score: 0.7647058823529413
# virginica의 f1-score: 0.7407407407407408
```

$$F_1 - Score = 2 \times \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

setosa : 정밀도(precision)=1.0, 재현율(recall)=0.933

versicolor : 정밀도(precision)=0.684, 재현율(recall)=0.866

virginica : 정밀도(precision)=0.833, 재현율(recall)=0.666





05 | 모델 평가 실습 (1)

△ 다음은 아이리스 데이터 **훈련 데이터셋**으로 **의사결정나무 모델**을 **학습**하고,
테스트 데이터셋으로 **모델의 Classification Report**를 **출력**하는 코드이다.

✦ 실행결과에서 F1-점수, 정확도가 앞에서 계산한 결과와 **동일한 것**을 볼 수 있음

```
print(classification_report(y_test, model.predict(X_test), target_names=iris_data.target_names))
```

	precision	recall	f1-score	support
setosa	1.00	0.93	0.97	15
versicolor	0.68	0.87	0.76	15
virginica	0.83	0.67	0.74	15
accuracy			0.82	45
macro avg	0.84	0.82	0.82	45
weighted avg	0.84	0.82	0.82	45

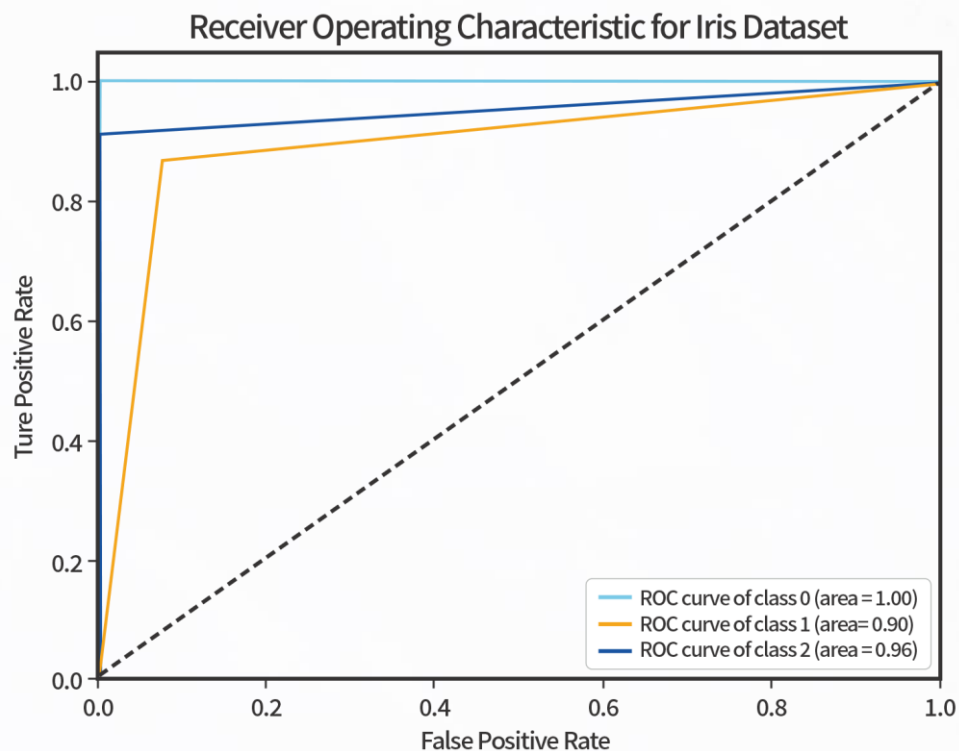


05 | 모델 평가 실습 (1)

△ 다음은 아이리스 데이터 **훈련 데이터셋**으로 **의사결정나무 모델**을 학습하고, **테스트 데이터셋**으로 **모델평가** 결과를 **ROC 곡선**으로 그리는 코드이다.

◆ 실행결과에서 **setosa(0)**, **virginica(2)**, **versicolor(1)**순으로 **분류 모델 성능**이 **좋은 것**을 볼 수 있음

➢ 아래의 ROC 곡선에서 알 수 있는 것처럼 이는 **다중 클래스 문제**임





05 | 모델 평가 실습 (1)

```
# 아이리스 데이터셋 로드
iris = datasets.load_iris()
X = iris.data
y = iris.target

# 데이터를 이진화 (One-vs-Rest)
y_binarized = label_binarize(y, classes=[0, 1, 2])
n_classes = y_binarized.shape[1]

# 데이터 분할 (학습용 데이터와 테스트용 데이터로 나누기)
X_train, X_test, y_train, y_test = train_test_split(X, y_binarized, test_size=0.5, random_state=42)

# 표준화 (StandardScaler)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# 분류기 설정
classifier = OneVsRestClassifier(DecisionTreeClassifier(random_state=42))

# 모델 학습 및 예측 확률 계산
y_score = classifier.fit(X_train, y_train).predict_proba(X_test)
```



05 | 모델 평가 실습 (1)

```
# ROC 곡선을 그리기 위한 설정
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# 전체 ROC 곡선 그리기
plt.figure()
colors = ['aqua', 'darkorange', 'cornflowerblue']
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic for Iris Dataset')
plt.legend(loc="lower right")
plt.show()
```




06 | 모델 평가 실습 (2)



모델 평가 실습 (2)

△ 다음은 **이진 분류 데이터셋**으로 **모델 평가 실습**을 진행해 보자.

◆ `make_classification` 함수로 변수가 2개인 이진 분류 데이터 4000개를 생성함

‣ 독립 변수 X 에는 **2개의 속성**이고, **데이터 형상**은 **(4000, 2)**로 구성함

‣ 종속 변수 y 의 **레이블**은 **0, 1**이고, **데이터 형상**은 **(4000,)**로 구성함



06 | 모델 평가 실습 (2)

△ 다음은 make_classification 함수로 변수가 2개인 이진 분류 데이터 4000개를 생성하는 코드이다.

- ✦ 실행결과 아래와 같이 독립 변수 x의 데이터 형상 (4000, 2), 종속 변수 y의 데이터 형상 (4000,) 인 것을 볼 수 있음

```
X, y = make_classification(n_samples=4000, n_features=2, n_informative=2, n_classes=2,
                           weights=[0.9, 0.1], n_redundant=0, random_state=0)
print(X.shape); print(y.shape); print(X[:10]); print(y[:10])
```

```
(4000, 2)
(4000,)
[[-1.68634397 -1.48712614]
 [-1.13876586 -0.011514 ]
 [-0.83921101 -0.97389947]
 [ 1.30473783 -1.72571204]
 [ 1.09798932 -0.9857081 ]
 [-1.63157569 -0.17534754]
 [-0.74394882 -0.85514724]
 [-0.3580264  -2.37929683]
 [-1.36898688 -1.18195672]
 [-1.0031273   1.01371182]]
[0 0 0 0 0 0 0 0 0 1]
```



06 | 모델 평가 실습 (2)

△ 다음은 이진 분류 데이터셋으로 **로지스틱회귀 모델 성능 평가 결과**를 **혼동 행렬로 확인**하는 코드이다.

✦ 실행결과에서 **137개**의 **데이터**가 **잘못 분류된 것**을 볼 수 있음

```
confusion_matrix(y_test, x_hat, labels=[0, 1])
```

```
array([[1744,  48],  
       [ 89, 119]], dtype=int64)
```

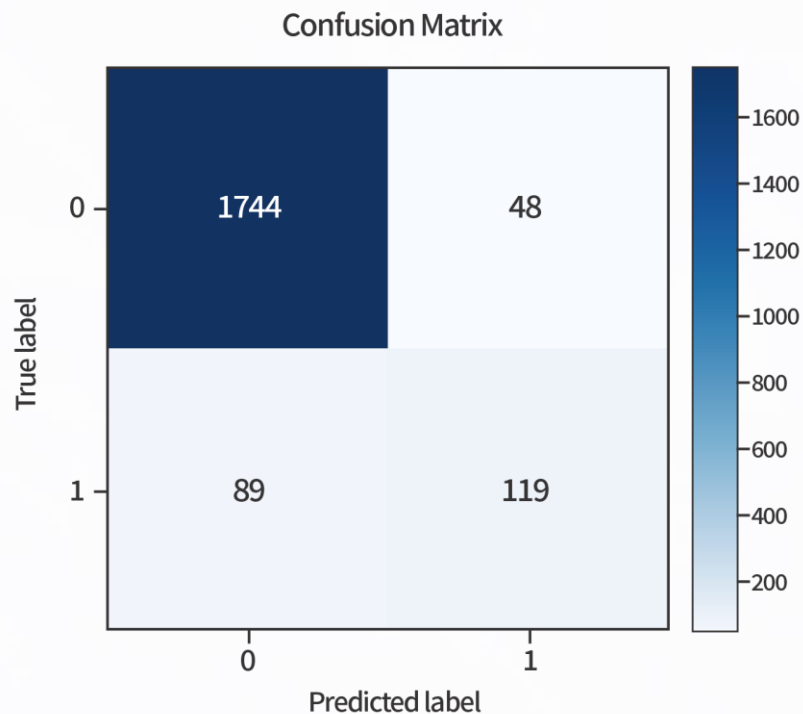


06 | 모델 평가 실습 (2)

다음은 이진 분류 데이터셋으로 로지스틱회귀 모델 성능 평가 결과를 혼동 행렬 그래프로 출력하는 코드이다.

실행결과에서 137개의 데이터가 잘못 분류된 것을 볼 수 있음

```
skplt.metrics.plot_confusion_matrix(y_test, x_hat)  
plt.show()
```





06 | 모델 평가 실습 (2)

△ 다음은 이진 분류 데이터셋으로 **로지스틱 회귀**(Logistic Regression) **모델 성능 평가 결과**로 Classification Report를 계산하는 코드이다.

◆ F1-점수의 경우 **0 레이블**은 **0.96**, **1 레이블**은 **0.63**인 것을 볼 수 있음

```
print(classification_report(y_test, model.predict(X_test)))
```

	precision	recall	f1-score	support
0	0.95	0.97	0.96	1792
1	0.71	0.57	0.63	208
accuracy			0.93	2000
macro avg	0.83	0.77	0.80	2000
weighted avg	0.93	0.93	0.93	2000



06 | 모델 평가 실습 (2)

△ 다음은 이진 분류 데이터셋으로 **서포트 벡터 머신(SVC)** 모델 성능 평가 결과로 Classification Report를 계산하는 코드이다.

◆ F1-점수의 경우 **0 레이블**은 **0.96**, **1 레이블**은 **0.61**인 것을 볼 수 있음

```
print(classification_report(y_test, model2.predict(X_test)))
```

	precision	recall	f1-score	support
0	0.95	0.98	0.96	1792
1	0.74	0.51	0.61	208
accuracy			0.93	2000
macro avg	0.84	0.75	0.79	2000
weighted avg	0.92	0.93	0.93	2000



06 | 모델 평가 실습 (2)

다음은 로지스틱 회귀(Logistic Regression)모델과 서포트 벡터 머신(SVC)모델의 Classification_Report를 비교 결과이다.

◆ 로지스틱 회귀 모델의 F1-점수는 0 레이블은 0.96, 1 레이블은 0.63인 것을 볼 수 있음

◆ 서포트 벡터 머신 모델의 F1-점수는 0 레이블은 0.96, 1 레이블은 0.61인 것을 볼 수 있음

	precision	recall	f1-score	support
0	0.95	0.97	0.96	1792
1	0.71	0.57	0.63	208
accuracy			0.93	2000
macro avg	0.83	0.77	0.80	2000
weighted avg	0.93	0.93	0.93	2000

로지스틱 회귀 모델

	precision	recall	f1-score	support
0	0.95	0.98	0.96	1792
1	0.74	0.51	0.61	208
accuracy			0.93	2000
macro avg	0.84	0.75	0.79	2000
weighted avg	0.92	0.93	0.93	2000

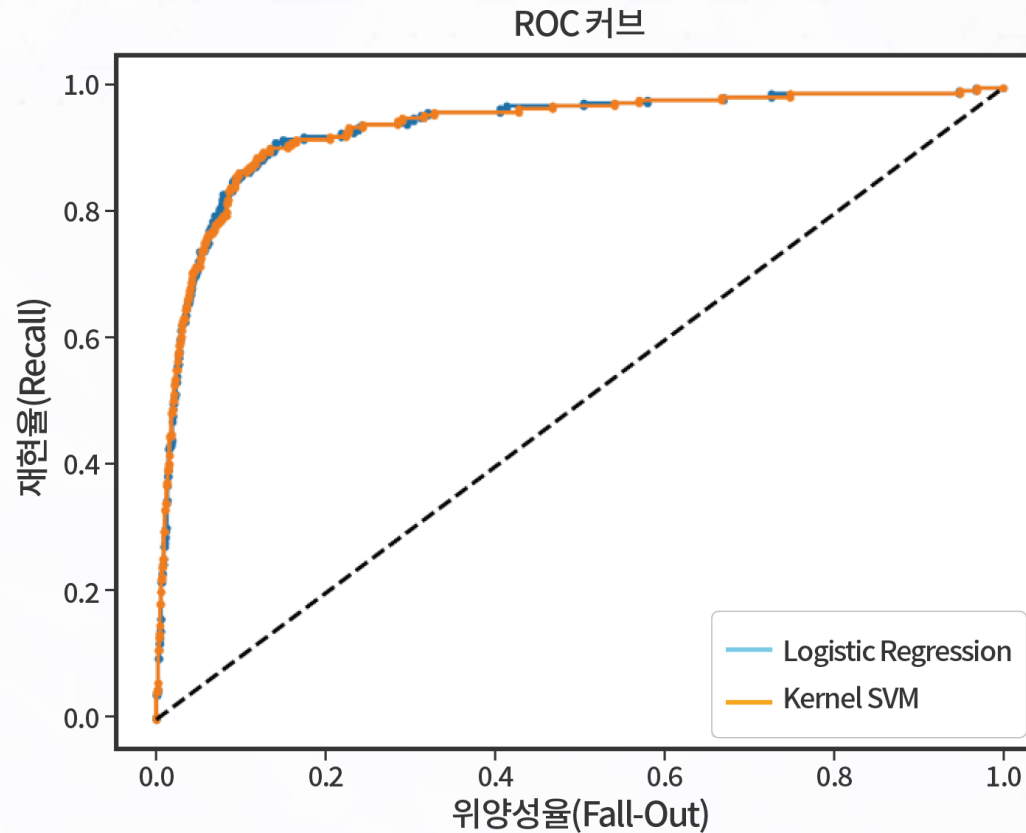
서포트 벡터 머신(SVC)모델



06 | 모델 평가 실습 (2)

△ 다음은 로지스틱 회귀(Logistic Regression)모델과 서포트 벡터 머신(SVC)모델의 ROC 곡선이다.

◆ 두 개 모델이 거의 전 구간에서 성능이 비슷하게 나타나는 것을 볼 수 있음





06 | 모델 평가 실습 (2)

```
fpr1, tpr1, thresholds1 = roc_curve(y_test, model.decision_function(X_test))
fpr2, tpr2, thresholds2 = roc_curve(y_test, model2.decision_function(X_test))

plt.rcParams["font.family"] = 'Malgun Gothic'
plt.plot(fpr1, tpr1, 'o-', ms=2, label="Logistic Regression")
plt.plot(fpr2, tpr2, 'o-', ms=2, label="Kernel SVM")
plt.legend()
plt.plot([0, 1], [0, 1], 'k--', label="random guess")
plt.xlabel('위양성율(Fall-Out)')
plt.ylabel('재현율(Recall)')
plt.title('ROC 커브')
plt.show()
```