

## 3. Classification 모델 만들기

Supervised Learning 중의 하나인 Classification 모델링에  
대해 배워보자

# 목차

1. Classification 모델링이란?
2. Classification 모델 종류
3. Classification 모델 성능 평가
4. scikit-learn 사용법 학습
5. Classification 모델 실습 #1
6. Classification 모델 실습 #2



# Classification 모델링이란?

Classification 모델에 대해 학습해보자

Classification 모델링이란?

## Classification 모델링 정의

- 분류 모델은 데이터를 다양한 클래스로 분류하는데 사용
- 예: 이메일 스팸 감지, 질병 진단, 이미지 인식 등등

# Classification 모델의 종류

- 이진 분류 (Binary Classification)
  - 두 개의 클래스로 분류. 예: 스팸 이메일 vs 일반 이메일.
- 다중 클래스 분류 (Multiclass Classification)
  - 3+ 클래스로 분류. 예: 손글씨 숫자 인식 (0-9).
- 다중 레이블 분류 (Multilabel Classification)
  - 한 레코드가 여러 클래스에 할당될 수 있음.
  - 예: 뉴스 기사가 여러 카테고리에 속함

# Classification 알고리즘의 종류

- Logistic Regression
  - 이진 분류 문제에 자주 사용.
- Decision Tree
  - 직관적이고 시각화하기 쉬움.
- Random Forests
- Support Vector Machines
  - 복잡한 분류 문제에 효과적.
- Deep Learning
  - 고급 분류 문제, 특히 이미지 및 음성 인식에 사용



# Classification 모델 종류

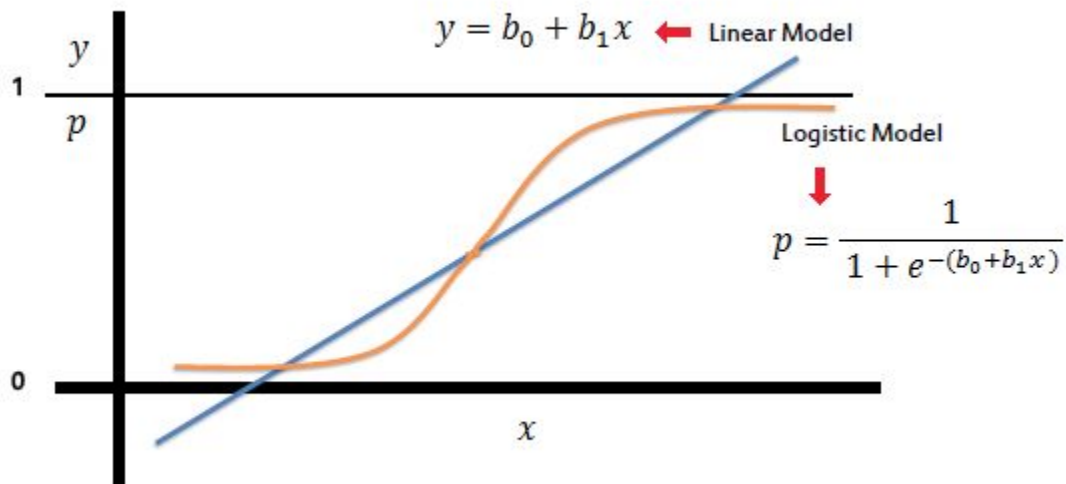
Classification 알고리즘에 대해 알아보자

# Logistic Regression vs. Softmax Regression

	Linear Regression		Logistic Regression		Softmax Regression	
types of supervised learning	regression		binary classification		multi-label classification	
predicting $x = 5, y = ?$	x (hours)	y (score)	x (hours)	y (pass/fail)	x (hours)	y (grade)
	10	90	10	P	10	A
	9	80	9	P	9	B
	3	50	3	F	3	D
	2	30	2	F	2	F
할 수 있는 것	공부 시간에 따른 시험 점수 예측		공부 시간에 따른 시험 합격 예측		공부 시간에 따른 학점 예측	

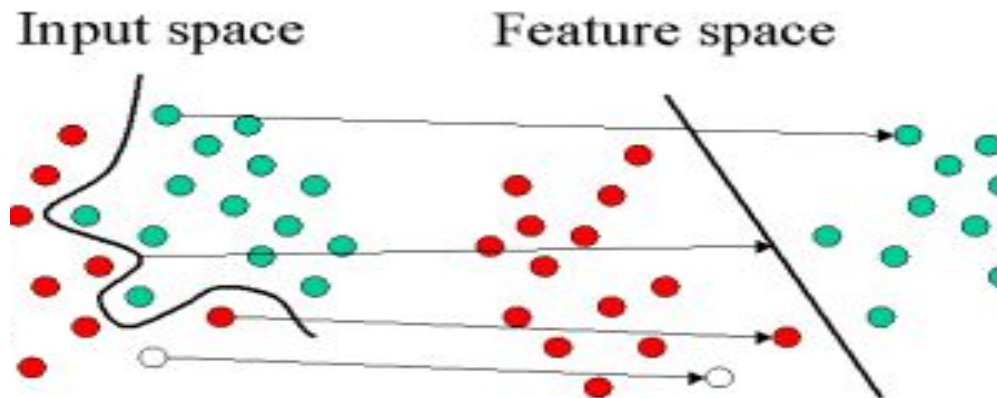


# Logistic Regression



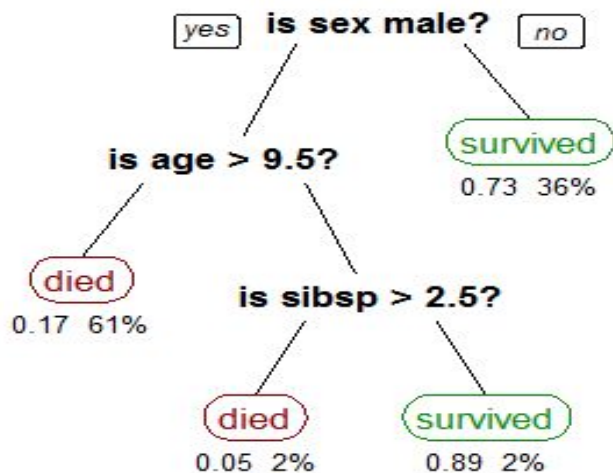
# Support Vector Machine

- Feature space를 변환하여 분류 작업 수행



# Decision Tree

- Decision Tree는 Regression으로도 사용 가능
- Feature의 중요도가 나오고 동작방식을 상대적으로 이해하고 설명 가능





# Classification 모델 성능 평가

Classification 모델의 성능 평가 방법을 알아보자

# Confusion Matrix

Truth \ Prediction	Spam	Non-Spam
Spam	True Positive	False Negative
Non-Spam	False Positive	True Negative

- $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$
- $\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$
- $\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$

Precision: spam으로 예측된 리뷰 중 얼마나 실제로 spam인가?

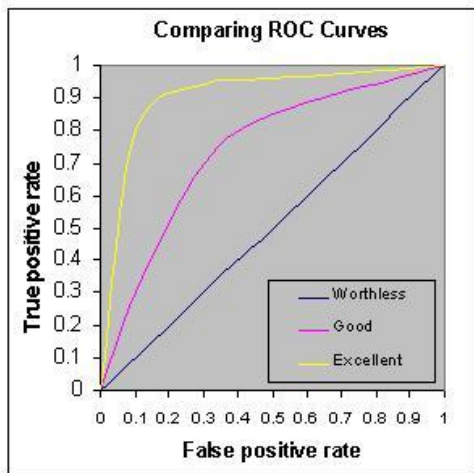
Recall: 실제 spam중의 얼마나 spam으로 예측되었나?

# True Positives & False Positives

- 스팸리뷰 Classification의 경우
  - True Positive: 스팸리뷰를 스팸리뷰라 판정하는 경우
    - $TPR: TP/(TP+FN)$
  - False Positive: 정상리뷰를 스팸리뷰라 판정하는 경우
    - $FPR: FP/(FP+TN)$
- 모든 리뷰를 스팸이라고 판정하면?
  - TPR은 100%이지만 FPR도 100%
- 모든 리뷰를 정상이라고 판정한다면?
  - FPR은 0%가 되지만 TPR도 0%
- Binary Classifier는 확률을 리턴
  - 어떤 threshold를 사용할지 결정하는데 사용하는 것이 ROC커브와 AUC

# ROC 커브

- ROC: Receiver Operating Characteristic
  - Binary classifier의 정확도를 보기위한 용도로 사용
- True Positive Rate을 Y축, False Positive Rate을 X축으로 그려진 그래프
  - TPR: 스팸 판정했을 때 맞을 확률, FPR: 스팸이 아닌 것들을 스팸 판정하는 확률
  - 이 그래프를 통해 TPR(높게)과 FPR(낮게)을 최적의 위치에서 조정



이 그래프에서 ROC 커브  
밑의 면적을 계산한 것이  
바로 AUC (Area Under the  
Curve)

# AUC (Area Under the Curve)

- AUC는 0부터 1 사이의 값을 리턴
  - 1에 가까우면 ML 모델이 굉장히 정확함을 의미
  - 0.5는 랜덤 추정에 가까움을 나타냄 (동전 던지기)
  - 0에 가까우면 데이터에 무엇인가 문제가 있음을 나타냄
- AUC 값을 보고 binary classification의 경우에는 threshold를 결정





# scikit-learn 학습

scikit-learn의 데이터 전처리 기능에 대해 알아보자 (Feature Engineering)

## 큰 흐름 (Hold Out 사용시)

- 먼저 머신러닝 알고리즘과 모델 성과 지표 결정
- 데이터셋 로딩
- 데이터셋을 훈련 데이터와 테스트 데이터로 분리
- 훈련 데이터 전처리
- 훈련 데이터로 모델 빌딩
- 테스트 데이터 전처리
- 레이블 정보 제외 테스트 데이터를 모델에 입력으로 지정
- 위 결과물과 정답 레이블을 비교하여 성과계산

# 피쳐 추출과 변환

- 피쳐 값들을 모델 훈련에 적합한 형태로 바꾸는 것을 지칭
- Data Transformations
  - Feature extraction
  - Preprocessing data
  - Imputation of missing values
  - Unsupervised dimensionality reduction
  - Transforming the prediction target (y)
  - ...

## 피쳐 추출과 변환: Feature extraction

- Loading features from dicts
- Feature hashing
- Text feature extraction
- Image feature extraction

# 피쳐 추출과 변환: Preprocessing data

- 숫자 필드 값의 범위 표준화
  - 숫자 필드라고 해도 가능한 값의 범위를 특정 범위(0부터 1)로 변환해야 함
  - 이를 피쳐 스케일링 (Feature Scaling) 혹은 정규화 (Normalization)

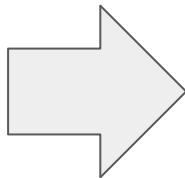
## 피쳐 추출과 변환: Imputation of missing values

- 비어있는 필드들의 값을 어떻게 채울 것인가?
  - `Imputer`. 앞서 타이타닉 승객 생존 분류기에 써봤음

## 피쳐 변환: 카테고리를 숫자로 변환 (1)

- 아래 왼쪽과 같은 값을 갖는 **Color**라는 이름의 피쳐가 존재한다면
- 이를 오른쪽과 같은 숫자로 변환해주는 것이 피쳐변환의 목적

변환 전
Red
Blue
Orange
White
Black



변환 후 (1)	변환 후 (2)
1	[ 1, 0, 0, 0, 0 ]
2	[ 0, 1, 0, 0, 0 ]
3	[ 0, 0, 1, 0, 0 ]
4	[ 0, 0, 0, 1, 0 ]
5	[ 0, 0, 0, 0, 1 ]

## 피쳐 변환: 카테고리를 숫자로 변환 (2)

- `sklearn.preprocessing` 모듈 아래 여러 인코더 존재
  - `OneHotEncoder`
  - `LabelEncoder`
  - `OrdinalEncoder`
  - ...
- 이 세 **Encoder**들간의 차이점은?
  - `OneHotEncoder`: 서로 관계없는 카테고리들을 인코딩하는 경우
    - City: Seoul, Tokyo, Beijing, ...
  - `LabelEncoder`: 레이블 필드를 인코딩하는 경우
  - `OrdinalEncoder`: 순서가 있는 카테고리들을 인코딩하는 경우
    - Rating: bad, average, good
    - Education level: high school, bachelor, master



## 피쳐 변환: 카테고리를 숫자로 변환 (3)

- Encoder를 사용하는 일반적인 방법

```
from sklearn.preprocessing import XxxEncoder  
X = [['seoul'], ['tokyo'], ['beijing']]
```

- 방법 1

```
enc = XxxEncoder()  
enc.fit(X)  
X_enc = enc.transform(X)
```

- 방법 2

```
X_enc = XxxEncoder().fit_transform(X)
```

## 피쳐 변환: 카테고리를 숫자로 변환 (4)

- OrdinalEncoder vs. OneHotEncoder

```
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder
X = [['seoul'], ['tokyo'], ['beijing']]
```

```
enc = OrdinalEncoder()
enc.fit(X)
enc.transform(X)
```

array([[1.], [2.], [0.]])

----

```
enc = OneHotEncoder()
enc.fit(X)
enc.transform(X)
```

array([[0., 1., 0.],  
 [0., 0., 1.],  
 [1., 0., 0.]])

## 피쳐 변환: 카테고리를 숫자로 변환 (5)

- LabelEncoder

```
from sklearn.preprocessing import LabelEncoder
```

```
y = ['recurrence-events', 'no-recurrence-events', 'recurrence-events']
```

```
enc = LabelEncoder()
```

```
enc.fit(y)
```

```
y_labeled = enc.transform(y)
```

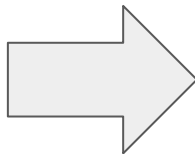
```
print(y_labeled)
```

[1, 0, 1]

# 피쳐 변환: 숫자 필드값의 범위 표준화 (1)

- 숫자 필드 값의 범위를 특정 범위(예를 들면 0부터 1)로 변환하는 것
- 피쳐 스케일링 (Feature Scaling) 혹은 정규화 (Normalization)라 부름

변환 전
-20
100
40
25
15



변환 후
0
1
0.5
0.375
0.125

## 피쳐 변환: 숫자 필드값의 범위 표준화 (2)

- `sklearn.preprocessing` 모듈 아래 두 개의 스케일러 존재
  - `StandardScaler`
  - `MinMaxScaler`
  - `RobustScaler`
  - `MaxAbsScaler`
- **StandardScaler**
  - 각 값에서 평균을 빼고 이를 표준편차로 나눔. 값의 분포를 정규분포를 따르게 변환
- **MinMaxScaler**
  - 모든 값을 0과 1사이로 스케일. 각 값에서 최소값을 빼고 (최대값-최소값)으로 나눔

## 피쳐 변환: 숫자 필드값의 범위 표준화 (3)

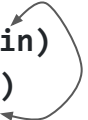
```
from sklearn.preprocessing import StandardScaler  
import numpy as np
```

```
X_train = np.array([[ 1., -1.,  2.],  
                    [ 2.,  0.,  0.],  
                    [ 0.,  1., -1.]])
```

```
scaler = StandardScaler().fit(X_train)
```

```
X_scaled = scaler.transform(X_train)
```

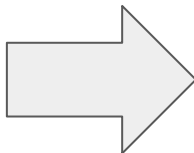
```
X_scaled= StandardScaler().fit_transform(X_train)
```



## 피쳐 변환: 값이 없는 필드 채우기 (1)

- 값이 존재하지 않는 레코드들이 존재하는 필드들의 경우 기본값을 정해서 채우는 것. **Impute**한다고 부름

변환 전
10
20
30
40



변환 후
10
<b>25</b>
20
30
40

## 피쳐 변환: 값이 없는 필드 채우기 (2)

- `sklearn.preprocessing` 모듈 아래 2개의 `Imputer` 존재
  - `SimpleImputer`
  - `IterativeImputer`
- **`SimpleImputer`**
  - 간단한 통계(평균, 중앙값, 최다 빈도 또는 상수)를 지정하여 누락된 값을 대체
- **`IterativeImputer`**
  - 결측값이 있는 각 피쳐를 다른 피쳐들의 함수로 지정



## 피쳐 변환: 값이 없는 필드 채우기 (3)

```
import numpy as np
from sklearn.impute import SimpleImputer

# Example dataset with missing values
data = np.array([
    [1, np.nan, 3],
    [4, 3, np.nan],
    [np.nan, 6, 9],
    [8, 5, 2]
])

imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputed_data = imputer.fit_transform(data)
print(imputed_data)
```

## 피쳐 변환: 값이 없는 필드 채우기 (4)

```
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

data = np.array([
    [1, np.nan, 3],
    [4, 3, np.nan],
    [np.nan, 6, 9],
    [8, 5, 2]
])

iterative_imputer = IterativeImputer(max_iter=10, random_state=0)
imputed_data = iterative_imputer.fit_transform(data)
print(imputed_data)
```

# 모델 테스트

- Hold out 테스트: `train_test_split`
- Cross Validation (X-Fold) 테스트: `cross_validate`

```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate
```

# 파이프라인 사용해보기

```
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
```

```
X, y = make_classification(random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

```
pipe = make_pipeline(StandardScaler(), LogisticRegression())
```

```
pipe.fit(X_train, y_train)
pipe.score(X_test, y_test)
```

# Classification 리포트 생성

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

```
classificationReport = classification_report(expected, predicted)
```

정답값

예측값

	precision	recall	f1-score	support
0	1.00	0.93	0.97	15
1	0.91	1.00	0.95	10
accuracy			0.96	25
macro avg	0.95	0.97	0.96	25
weighted avg	0.96	0.96	0.96	25

# 실습해보기

- 구글 Colab 상에서 실습 진행
- [Scikit-Learn Preprocessing 실습](#)

# Classification 모델 실습 #1

타이타닉 승객 생존 예측 Classification 모델을  
만들어보자

# 분류 (Classification) 머신 러닝

1. 지도 학습 (Supervised Learning)의 일부로 예측 대상이 유한한 경우
  - a. 예를 들면 타이타닉 승객 생존 예측처럼 예측값이 두 가지인 경우 (생존 혹은 비생존)
    - i. 캐글에서 유명한 이진 분류 문제
  - b. 이진분류 학습의 경우 AUC 혹은 F1을 성능지표로 사용함
2. 예측 대상이 무한한 경우 (연속되는 값) 이를 회귀 (Regression) 머신 러닝이라 부름
  - a. 예를 들면 주식가격 예측



# 타이타닉 승객 생존 예측

- 머신러닝의 Hello World라고 할 수 있는 굉장히 유명한 데이터셋
  - 2015년 캐글에서 “Titanic - Machine Learning from Disaster”라는 이름의 튜토리얼로 시작됨: <https://www.kaggle.com/c/titanic>
- Binary Classification 알고리즘 사용 예정
  - 생존 혹은 비생존을 예측하는 것이라 Binary Classification을 사용
    - 정확히는 Binomial Logistic Regression을 사용 (2개 클래스 분류기)
  - AUC (Area Under the Curve)의 값이 중요한 성능 지표가 됨
    - True Positive Rate: 생존한 경우를 얼마나 맞게 예측했나? 흔히 Recall이라고 부르기도함
    - False Positive Rate: 생존하지 못한 경우를 생존한다고 얼마나 예측했나?

# 타이타닉 승객 생존 예측

- 총 892개의 레코드로 구성되며 11개의 피쳐와 레이블 필드(생존여부)로 구성
  - 2번째 필드(Survived) 바로 예측해야하는 승객 생존 여부

필드 이름	설명
1. PassengerId	승객에게 주어진 일련번호
2. <b>Survived</b>	생존여부를 나타내는 레이블 정보
3. Pclass	티켓클래스. 1 = 1st, 2 = 2nd, 3 = 3rd
4. Name	승객의 이름
5. Gender	승객의 성별
6. Age	승객의 나이

필드 이름	설명
7. SibSp	같이 승선한 형제/자매와 배우자의 수
8. Parch	같이 승선한 부모와 자녀의 수
9. Ticket	티켓 번호
10. Fare	운임의 값
11. Cabin	숙소 번호
12. Embarked	승선한 항구. C = Cherbourg, Q = Queenstown, S = Southampton

# 타이타닉 승객 생존 예측

survived	pclass	name	sex	age	sibsp	parch	ticket	fare	embarked
1	2	Mary D Kingcome	F	55	0	0	248706	16	S
0	3	Rice, Master. Eugene	M	2	4	1	382652	29.125	Q



예측해야하는 값 - 보통  
레이블이라고 부르며 Y로  
표기



레이블을 예측하는데 사용하는 힌트들 - 피쳐(Feature)  
라고 부름. 흔히 X라고 표기

# 실습

1. [구글 Colab](#) 상에서 실습 진행



# 속제

오늘 배운 것을 정리해보도록 하자

## 3장 숙제

- 실습 #1에 다음 2가지 기능 추가하기
  - Age 필드의 값을 평균으로 채우는 일을 SimpleImputer로 해보기
  - Embarked 필드의 경우 아래 해보기
    - 비워진 값의 경우 Embarked 값 중 가장 흔한 값을 사용해서 SimpleImputer로 채우기
    - Embarked를 OneHotEncoder를 사용해서 변환후 그걸 학습하는 필드 중의 하나로 추가해보기
  - 최종 classification\_report의 실행 화면을 제출할 것!
- 숙제 리뷰는 4장 도입부에서 할 예정

# Classification 모델 실습 #2

당뇨병 여부 예측 Classification 모델을 만들어보자

# 당뇨병 예측 문제 소개

- 데이터 셋

- UCI 대학에서 제공한 미국 피마 인디언 여성의 당뇨병 데이터(21세 이상)로 '피마 인디언'은 절반이 당뇨병 환자 (생활 습관상의 문제).
- 총 767개의 레코드가 제공되고 각 레코드에는 모델의 훈련을 위해 사용할 수 있는 8개의 피쳐와 최종 결과 (당뇨병 여부)가 레이블로 제공됨

- 만들려는 모델

- 당뇨병 여부를 예측하는 모델
- Supervised 머신 러닝 문제에 해당: 이진 분류 (binary classification) 문제에 해당



## 당뇨병 여부 예측 문제 피쳐 소개 (1)

- **Pregnancies:** 임신 횟수
- **Glucose:** 75g의 구강 혈당이 주어지고 2시간 후에 측정된 포도당 값
- **BloodPressure:** 혈압이며 그 중에서도 이완시 혈압이다 (mm Hg)
- **SkinThickness:** 삼두근 피부주름의 두께 (mm). 체지방(fat) 값
- **Insulin:** 혈당 투여 2시간 후 인슐린 레벨 (mu U/ml)
- **BMI:** 체질량 지수
- **DiabetesPedigreeFunction:** 가족 병력 기반으로 계산된 당뇨병에 걸릴 확률
- **Age:** 나이
- **Outcome:** 당뇨병 여부를 나타내며 (1이면 당뇨) 모델의 예측 대상

# F1 지표란?

- 이진 분류의 정확도를 측정하기 위한 지표로 Precision과 Recall의 조화평균
- $F1 = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$ 
  - $\text{Precision} = \text{True Positive} / (\text{True Positive} + \text{False Positive})$
  - $\text{Recall} = \text{True Positive} / (\text{True Positive} + \text{False Negative})$

	가입 정답	미가입 정답
가입 예측	True Positive	False Positive
미가입 예측	False Negative	True Negative

# 실습

1. [구글 Colab](#) 상에서 실습 진행
  - a. EDA에 이어 ML 모델 빌딩



# Q & A

오늘 강의에 대해서 궁금한 부분이 있으면 알려주세요!