파이썬 머신러닝 기초

with  python™

01. scikit-learn 소개

◆ Python 머신러닝의 대표적인 라이브러리

✓ 출처 : https://scikit-learn.org/stable/

Classification

Regression

Clustering

Dimensionality Reduction

Model Selection

Preprocessing

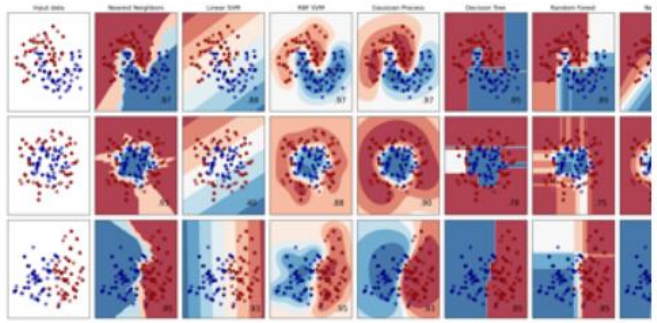
01. scikit-learn 소개

◆ Python 머신러닝의 대표적인 라이브러리

Classification

Identifying which category an object belongs to.

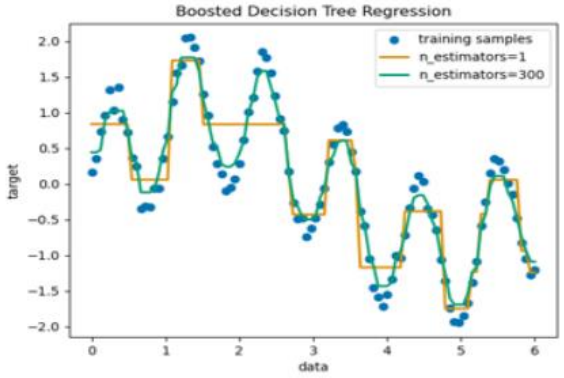
Applications: Spam detection, image recognition.
Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...



Regression

Predicting a continuous-valued attribute associated with an object.

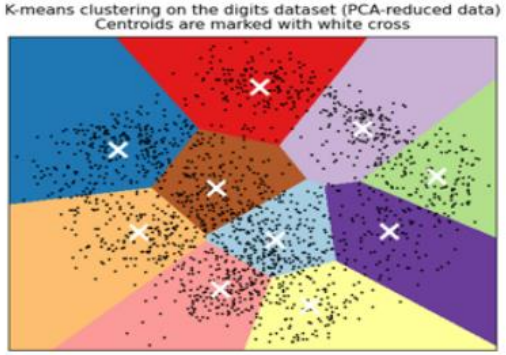
Applications: Drug response, Stock prices.
Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...



Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more...



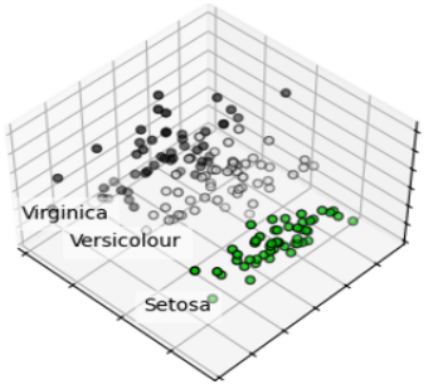
01. scikit-learn 소개

◆ Python 머신러닝의 대표적인 라이브러리

Dimensionality reduction

Reducing the number of random variables to consider.

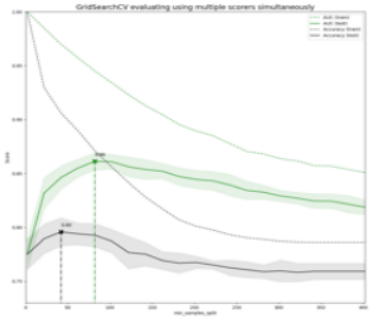
Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization, and more...



Model selection

Comparing, validating and choosing parameters and models.

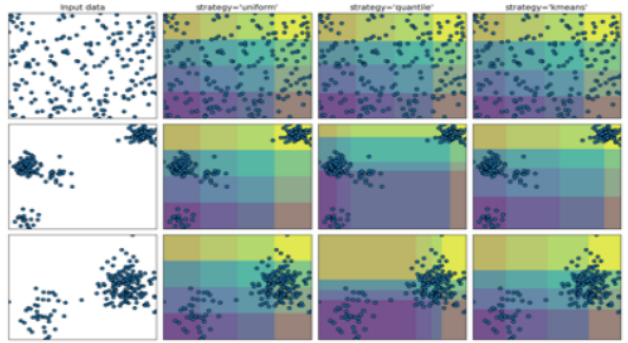
Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...



Preprocessing

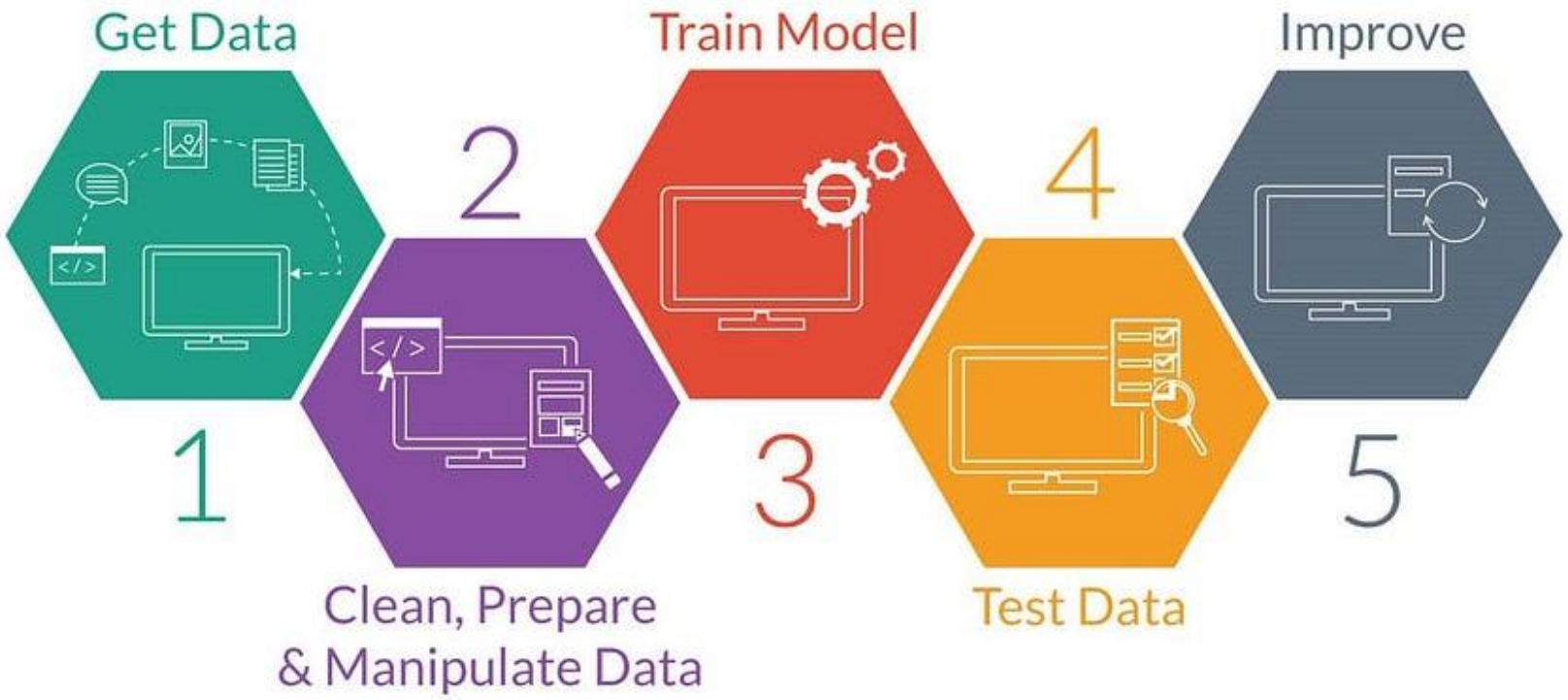
Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...



02. 머신러닝 프로세스

◆ 데이터 수집 > 데이터 가공 > 모델 학습 > 모델 평가 > 모델 성능 개선



출처 : https://shorturl.at/zAEM4

03. scikit-learn 주요 모듈

◆ 개요

✓ 머신러닝 분석 수행할 때 가장 유용하게 사용할 수 있는 파이썬 라이브러리

주요 업무	주요 모듈	설명
데이터 분리	sklearn.model_selection	• 훈련, 검증, 테스트 데이터로 데이터를 분리하기 위해 활용 • train_test_split 메서드 주로 활용
데이터 처리	sklearn.preprocessing	• Feature Scaling - StandardSacler : 평균 0, 분산 1로 기준으로 변경 - MinMaxScaler : 0과 1 사이에 위치하도록 데이터 재조정 • Binarization : 수치 데이터를 0과 1로 변환 • Encoding : 문자 데이터를 수치로 변환 • Imputer : 결측치를 채움
데이터 축소	sklearn.decomposition	• PCA 등을 통해 차원축소 지원 가능
모형 학습	sklearn.linear_model	• 선형회귀, 로지스틱 회귀 등의 알고리즘 지원
	sklearn.tree	• 트리 알고리즘 제공
	sklearn.ensemble	• 랜덤포레스트 알고리즘 제공
모형 평가	sklearn.metrics	• 분류, 회귀, 클러스터링 등에 관한 모형 평가 지표 제공

03. scikit-learn 주요 모듈

◆ 머신러닝 모형 학습 전, 데이터 전처리 이유

✓ 머신러닝 알고리즘 적용 전, 결측치 미 허용 → 다른 고정 값 입력 필요

✓ 문자열 값을 허용하지 않는다. → 숫자형으로 변환

✓ 서로 다른 변수의 값 범위를 일정한 수준으로 변환함

종류	모듈	설명
표준화Standardization	RobustScaler	• 중간값을 제거하고, 사분위수 범위에 따라 데이터의 배율 조정 • 1사분위 3사분위 사이의 범위 • 이상치 제거에 좋음 • quantile_range 파라미터(default [0.25, 0.75])에서 조정 가능
	StandardScaler	• 평균이 0이고 분산이 1이 정규분포를 가진 값으로 변환 • SVM, 선형회귀, 로지스틱 모델 사용 시, 필수 • 회귀보다는 분류분석에 유용
정규화Normalization	MinMaxScaler	• 모든 수치 데이터를 [0, 1], 음수가 존재하면 [-1, 1] 사이 변환 • 분류보다는 회귀에 유용
	MaxAbsScaler	• 최대절댓값과, 0이 각각 1, 0이 되도록 스케일링 하는 정규화 • 모든 값은 -1과 1사이에 표현 • 이상치에 매우 민감, • 분류보다는 회귀에 유용

03. scikit-learn 주요 모듈

◆ 머신러닝 모형 학습 전, 데이터 전처리 이유

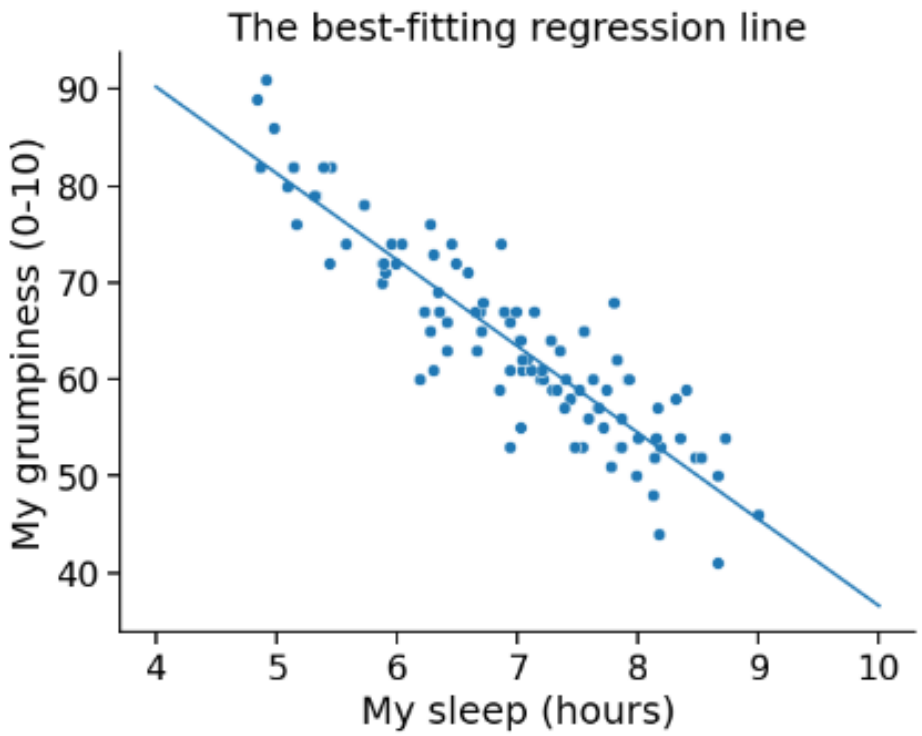
- ✓ 머신러닝 알고리즘 적용 전, 결측치 미 허용 → 다른 고정 값 입력 필요
- ✓ 문자열 값을 허용하지 않는다. → 숫자형으로 변환
- ✓ 서로 다른 변수의 값 범위를 일정한 수준으로 변환함

종류	모듈	설명
Ordinal Encoding	OrdinalEncoder	• 서열(Ordinal)척도를 숫자로 변환함 (독립변수만)
Label Encoding	LabelEncoder	• 서열(Nominal)척도를 숫자로 변환함 (종속변수만)
One-Hot Encoding	OneHotEncoder	• 명목(Nominal)척도를 숫자로 변환함 (독립변수만)

04. 머신러닝 성능평가

◆ 성능평가 – 회귀 모형 오차의 개념

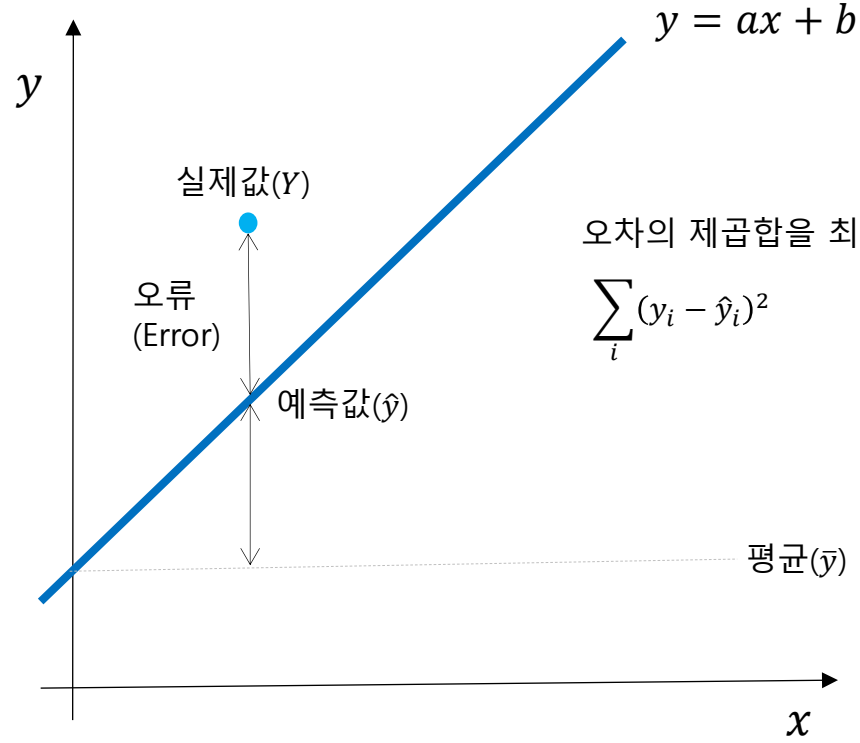
- ✓ 오차는 실제값과 예측값의 차이를 말함, 양(+)의 값과 음(-)의 값 발생
- ✓ 양과 음의 값 한산 시, 오차는 0에 가까워짐, 따라서, 제곱 또는 절댓값을 취함
- ✓ 회귀 모형은 오차의 제곱 혹은 절댓값의 합이 최소화되는 라인을 찾는 것이 목표임



04. 머신러닝 성능평가

◆ 성능평가 – 회귀 모형 오차의 개념

- ✓ 오차는 실제값과 예측값의 차이를 말함, 양(+)의 값과 음(-)의 값 발생
- ✓ 양과 음의 값 한산 시, 오차는 0에 가까워짐, 따라서, 제곱 또는 절댓값을 취함
- ✓ 회귀 모형은 오차의 제곱 혹은 절댓값의 합이 최소화되는 라인을 찾는 것이 목표임



$y = ax + b$

- a : 기울기(회귀계수, Coefficient), 최소제곱법 추정
- b : 절편(intercept)

오차의 제곱합을 최소화하는 라인

$$\sum_i (y_i - \hat{y}_i)^2$$

04. 머신러닝 성능평가

◆ 성능평가 – 회귀 : MAE(Mean Absolute Error)

✓ 실젯값과 예측값의 차이 활용

정 의	• 실젯값과 예측값의 차이를 절댓값으로 변환해 평균한 것
수 식	$MAE = \frac{1}{n}\sum_{i=1}^{n}\left\|Y_i - \hat{Y}_i\right\|$
특 징	• 에러의 크기가 그대로 반영 • 이상치에 영향 받음

◆ Python Code

```
1 :  from sklearn.metrics import mean_absolute_error
2 :  mae = mean_absolute_error(y_test, y_pred)
     0.00
```

04. 머신러닝 성능평가

◆ 성능평가 – 회귀 : MSE(Mean Squared Error)

✓ 실젯값과 예측값의 차이 활용

정 의	• 실젯값과 예측값의 차이를 제곱해 평균한 것
수 식	$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$
특 징	• 실젯값과 예측값 차이의 면적 합을 의미 • 특이값이 존재하면 수치가 증가

◆ Python Code

```
1  :  from sklearn.metrics import mean_squared_error
2  :  mae = mean_squared_error (y_test, y_pred)
      0.00
```

04. 머신러닝 성능평가

◆ 성능평가 – 회귀 : RMSE(Root Mean Squared Error)

✓ 실젯값과 예측값의 차이 활용

정 의	• 실젯값과 예측값의 차이를 제곱해 평균한 것에 루트를 씌운 것
수 식	$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$
특 징	• 오차에 제곱을 하면 오차가 클수록 그에 따른 가중치가 높게 반영 • 이 때, 손실이 기하급수적으로 증가하는 상황에서 실제 오류값의 평균보다 값이 더 커지지 않도록 상쇄하기 위해 사용

◆ Python Code

```
1 :  from sklearn.metrics import mean_squared_error
2 :  import numpy as np
3 :  mse = mean_squared_error(y_test, y_pred)
4 :  rmse = np.sqrt(mse)
```

04. 머신러닝 성능평가

◆ 성능평가 – 회귀 : MSLE(Mean Squared Log Error)

✓ 실젯값과 예측값의 차이 활용

정 의	• 실젯값과 예측값의 차이를 제곱해 평균한 것에 로그를 적용
수 식	$$MSLE = \log\left(\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2\right)$$
특 징	• RMSE와 같이 오차가 기하급수적으로 증가하는 상황에서 실제 오차의 평균보다 값이 더 커지지 않도록 상쇄

◆ Python Code

```
1 :  from sklearn.metrics import mean_squared_log_error
2 :  msle = mean_squared_log_error(y_test, y_pred)
3 :  msle
     0.00
```

04. 머신러닝 성능평가

◆ 성능평가 – 회귀 : MAPE(Mean Absolute Error)

✓ 실젯값과 예측값의 차이 활용

정 의	• MAE를 퍼센트로 반환한 것
수 식	$MAPE = \frac{n}{100}\sum_{i=1}^{n}\left\|\frac{y_i - \hat{f}(x_i)}{y_i}\right\|$
특 징	• 오차가 예측값에서 차지하는 정도를 나타냄

◆ Python Code

```
1  :  import numpy as np
2  :  def MAPE(y_test, y_pred):
3  :      mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100
4  :      return mape
5  :  mape = MAPE(y_test, y_pred)
6  :  mape
```

04. 머신러닝 성능평가

◆ 성능평가 – 회귀 : 결정계수(R^2)

✓ 실젯값과 예측값의 차이 활용

정 의	• 회귀 모형이 선형인 경우에는 결정계수를 평가 지표로 활용 • 결정계수는 0에서 1사이의 값을 가지며 1에 가까울수록 모형의 설명력이 좋음
수 식	$R^2 = \frac{SSR}{SST} = \frac{SST - SSE}{SST} = 1 - \frac{SSE}{SST}$

지표	설명	수식
SST (Total Sum of Squared)	• 전체제곱합 • 실제 관측치(y_i)와 y값들의 평균($\bar{y}$)의 차이를 제곱하여 합한 값 • y가 가지는 전체 변동	$\sum_{i-1}^{n}(y_i - \bar{y})^2$
SSR (Regression Sum of Squared)	• 회귀제곱합 • 모형 예측치($\hat{y}_i$)와 y값들의 평균($\bar{y}$)의 차이를 제곱하여 합한 값 • y가 가지는 전체 변동성 중 회귀 모형으로 설명할 수 있는 변동	$\sum_{i-1}^{n}(\hat{y}_i - \bar{y})^2$
SSE (Error Sum of Squared)	• 오차제곱합 • 실제 관측치(y_i)와 모형 예측치($\hat{y}_i$)의 차이를 제곱하여 합한 값 • y가 가지는 전체 변동성 중 회귀 모형으로 설명할 수 없는 변동	$\sum_{i-1}^{n}(y_i - \hat{y}_i)^2$

04. 머신러닝 성능평가

◆ 성능평가 – 회귀 : 결정계수(R^2)

✓ 실젯값과 예측값의 차이 활용

정 의	• 회귀 모형이 선형인 경우에는 결정계수를 평가 지표로 활용 • 결정계수는 0에서 1사이의 값을 가지며 1에 가까울수록 모형의 설명력이 좋음
수 식	$R^2 = \frac{SSR}{SST} = \frac{SST - SSE}{SST} = 1 - \frac{SSE}{SST}$

◆ Python Code

```
1 :  from sklearn.metrics import r2_score
2 :  r2 = r2_score(y_test, y_pred)
3 :  r2
     0.92
```

04. 머신러닝 성능평가

◆ 성능평가 – 분류 : 혼동행렬

		예측 결과	
		TRUE	FALSE
실제 정답	TRUE	TP (True Positive)	FN (False Negative)
	FALSE	FP (False Positive)	TN (True Negative)

◆ 용어 정리

- TN(True Negative) : 음성을 음성이라고 예측
- FP(False Positive) : 음성을 양성이라고 예측
- FN(False Negative): 양성을 음성이라고 예측
- TP(True Positive) : 양성을 양성이라고 예측

04. 머신러닝 성능평가

◆ 성능평가 – 분류 : 혼동행렬

✓ 정확도의 한계점 보완하기 위해 혼동행렬 활용

정 의	• 모델의 성능을 평가할 때 사용되는 지표, 예측값과, 실제값을 보여주는 행렬
특 징	• 4분면 행렬에서 실제 레이블 클래스 값과 예측 레이블 클래스 값이 어떤 유형을 가지고 매핑되는지 나타남

◆ Python Code

```
1 :  from sklearn.metrics import confusion_matrix
2 :  cm = confusion_matrix(y_test, y_pred)
3 :  cm
```

04. 머신러닝 성능평가

◆ 성능평가 – 분류 : 정확도(Accuracy)

✓ 실제분류와 예측분류가 얼마나 일치했는가를 기반으로 알고리즘의 성능을 평가

정 의	• 실제 데이터에서 예측 데이터가 얼마나 같은지 판단하는 지표
수 식	$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}$
특 징	• 데이터 구성에 따라 머신러닝 모델의 성능을 왜곡할 가능성 있음

◆ Python Code

```
1 : from sklearn.metrics import accuracy_score
2 : acc = accuracy_score(y_test, y_pred)
3 : acc
```

		예측 결과	
		TRUE	FALSE
실제 정답	TRUE	TP (True Positive)	FN (False Negative)
	FALSE	FP (False Positive)	TN (True Negative)

04. 머신러닝 성능평가

◆ 성능평가 – 분류 : 정밀도(Precision)

✓ Positive 데이터 예측에 집중한 평가지표

정 의	• Positive로 예측한 것들 중 실제로도 Positive인 것들의 비율
수 식	$Precision = \frac{TP}{FP + TP}$
특 징	• Positive 예측성능을 더욱 정밀하게 측정하기 위한 평가지표 • 양성 예측도라 불리움 • 정밀도가 상대적인 중요성을 가지는 경우 - 실제 Negative인 데이터를 Positive로 잘못 예측했을 때 업무상 큰 영향 발생 시

◆ Python Code

```
1 : from sklearn.metrics import precision_score
2 : acc = precision_score(y_test, y_pred)
3 : acc
```

		예측 결과	
		TRUE	FALSE
실제 정답	TRUE	TP (True Positive)	FN (False Negative)
	FALSE	FP (False Positive)	TN (True Negative)

04. 머신러닝 성능평가

◆ 성능평가 – 분류 : 재현율(Recall)

✓ Positive 데이터 예측에 집중한 평가지표

정 의	• 실제 Positive인 것들 중 Positive로 예측한 것들의 비율
수 식	$Recall = \frac{TP}{FN + TP}$
특 징	• 민감도(Sensitivity) 또는 TPR(True Positive Rate)라고 불림 • 재현율이 상대적인 중요성을 가지는 경우 - 실제 Positive인 데이터를 Negative로 잘못 예측 했을 때 업무상 큰 영향이 발생할 때

◆ Python Code

```
1 : from sklearn.metrics import recall_score
2 : recall = recall_score(y_test, y_pred)
3 : recall
```

		예측 결과	
		TRUE	FALSE
실제 정답	TRUE	TP (True Positive)	FN (False Negative)
	FALSE	FP (False Positive)	TN (True Negative)

04. 머신러닝 성능평가

◆ 성능평가 – 분류 : F1 스코어

✓ 정밀도와 재현율을 결합한 성능지표

정 의	• 실제 Positive인 것들 중 Positive로 예측한 것들의 비율
수 식	$F1\ score = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} = 2 \times \frac{precision \times recall}{precision + recall}$
특 징	• 정밀도와 재현율이 어느 한쪽으로 치우치지 않고 적절한 조화를 이룰 때 상대적으로 높은 수치를 나타냄

◆ Python Code

```
1 : from sklearn.metrics import f1_score
2 : f1 = f1_score(y_test, y_pred)
3 : f1
```

		예측 결과	
		TRUE	FALSE
실제 정답	TRUE	TP (True Positive)	FN (False Negative)
	FALSE	FP (False Positive)	TN (True Negative)

04. 머신러닝 성능평가

◆ 성능평가 – 분류 : ROC-Curve

✓ ROC 곡선과 이를 기반으로 하는 AUC 스코어는 이진 분류모델의 주요 성능평가지표임

정 의	• FPR(False Positive Rate)이 변할 때, TPR(True Positive Rate)이 변하는 것을 나타내는 곡선(ROC)
수 식	$$TNR = \frac{TN}{FP + TN}$$ $$FPR = 1 - TNR = \frac{FP}{FP + TN}$$
특 징	• TPR을 y축으로, FPR을 x축으로 하는 그래프 • 분류 결정 임곗값을 조절하면서 FPR이 0부터 1까지 변할 때, TPR의 변화값을 그래프에 나타냄 • 우상향 그래프로 그려짐

		예측 결과	
		TRUE	FALSE
실제 정답	TRUE	TP (True Positive)	FN (False Negative)
	FALSE	FP (False Positive)	TN (True Negative)

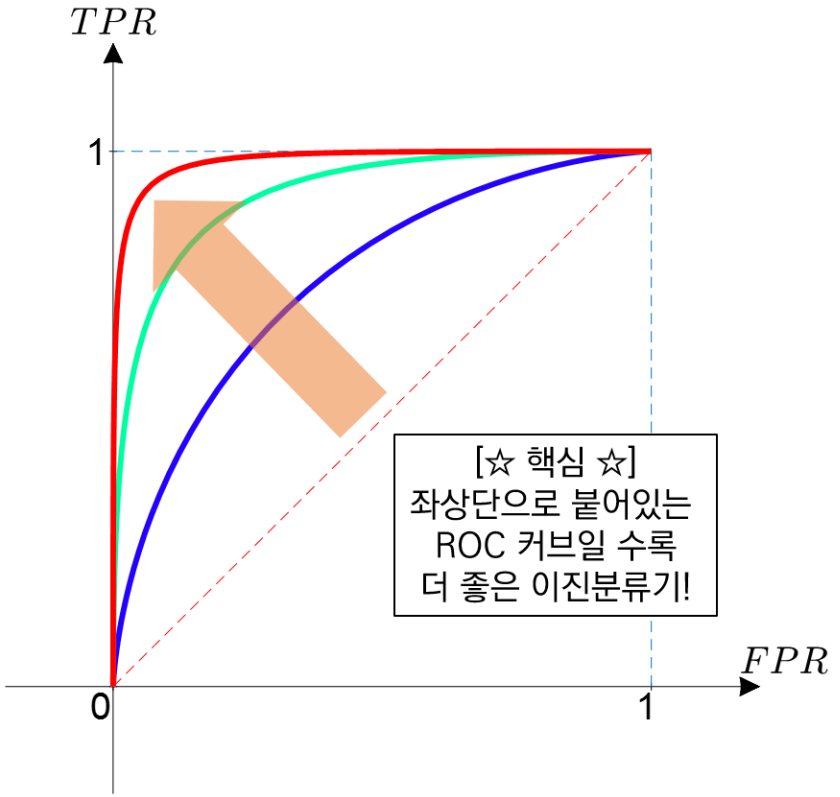
04. 머신러닝 성능평가

◆ 성능평가 – 분류 : ROC-Curve

✓ ROC 곡선과 이를 기반으로 하는 AUC 스코어는 이진 분류모델의 주요 성능평가지표임

◆ Python Code

```
1 : from sklearn.metrics import roc_curve
2 : import matplotlib.pyplot as plt
3 : fpr, tpr, thres = roc_curve(y_test, y_pred, pos_label = 1)
4 : plt.plot(fpr, tpr)
5 : plt.show()
```



04. 머신러닝 성능평가

◆ 성능평가 – 분류 : AUC 스코어

✓ ROC 곡선 아래의 면적 값을 분류 성능지표로 사용

정 의	• Area Under the ROC Curve • ROC 곡선 아래의 면적 • 1에 가까울수록 예측성능이 우수하다고 판단
특 징	• AUC 값이 커지려면 FPR이 작을 때 TPR 값이 커야 함 • 우상향 직선에서 멀어지고 왼쪽 상단의 모서리 쪽으로 가파르게 곡선이 이동할수록 AUC가 1에 가까워짐 • 랜덤 수준의 AUC값은 0.5

◆ Python Code

```
1  :  from sklearn.metrics import roc_curve, auc
2  :  fpr, tpr, thres = roc_curve(y_test, y_pred, pos_label = 1)
3  :  auc = auc(fpr, tpr)
5  :  auc
```

05. 하이퍼파라미터 튜닝

◆ 하이퍼 파라미터 튜닝

✓ 모델 최적화를 위해 여러가지 옵션을 조합하는 방법

Max Depth	Estimators	min Split	Accuracy	Precision	Recall	F-Score
10	200	2	45.23	50.12	54.23	56.12
20	300	5	48.12	49.23	50.12	58.12
30	400	10	60	56.22	50.12	53.23
40	500	2	54.2	57.23	49.11	50.23
50	600	5	60	61	62	60
60	700	10	51.8875	53.2	50.895	54.425
70	800	2	55.58	55.92	52.8375	55.395
80	900	5	56.52188	56.9125	53.03125	54.47125
90	1000	10	55.41688	56.8375	53.71063	55.0125

05. 하이퍼파라미터 튜닝

◆ 하이퍼 파라미터 튜닝

✓ Grid Search vs Random Search

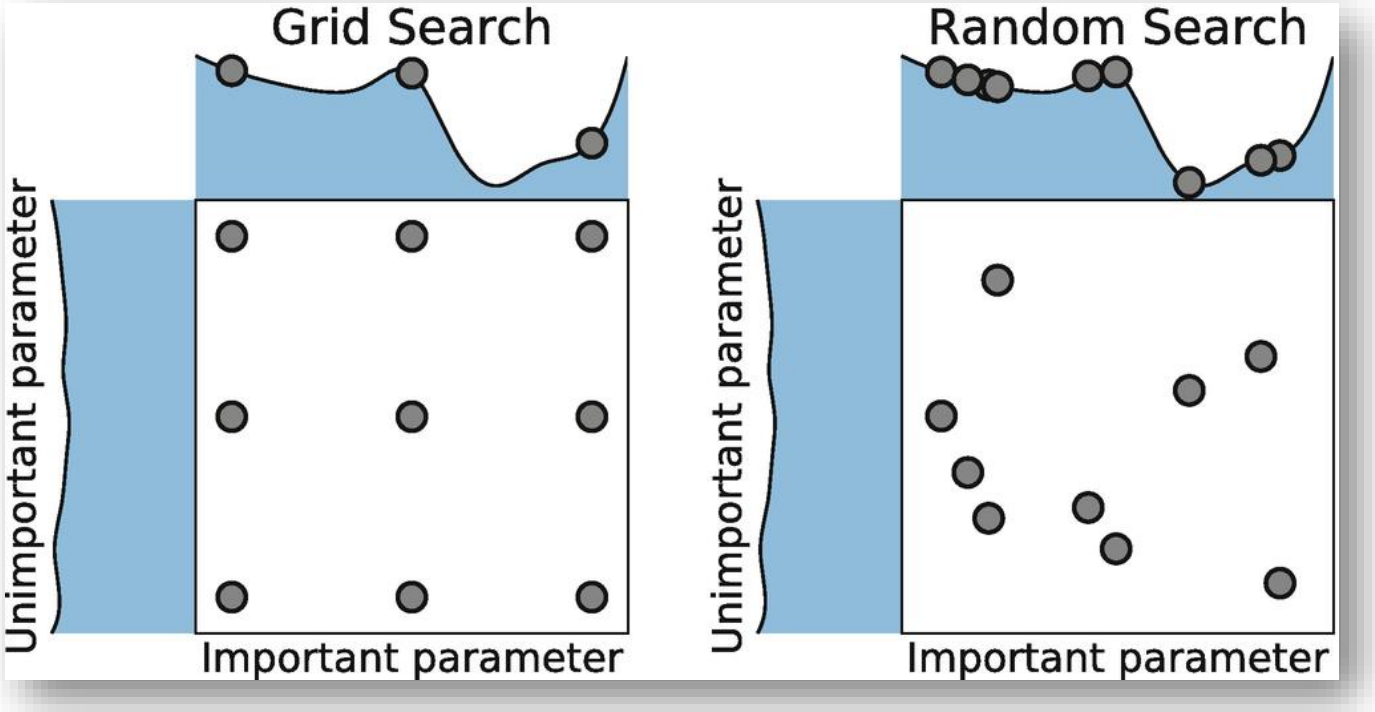
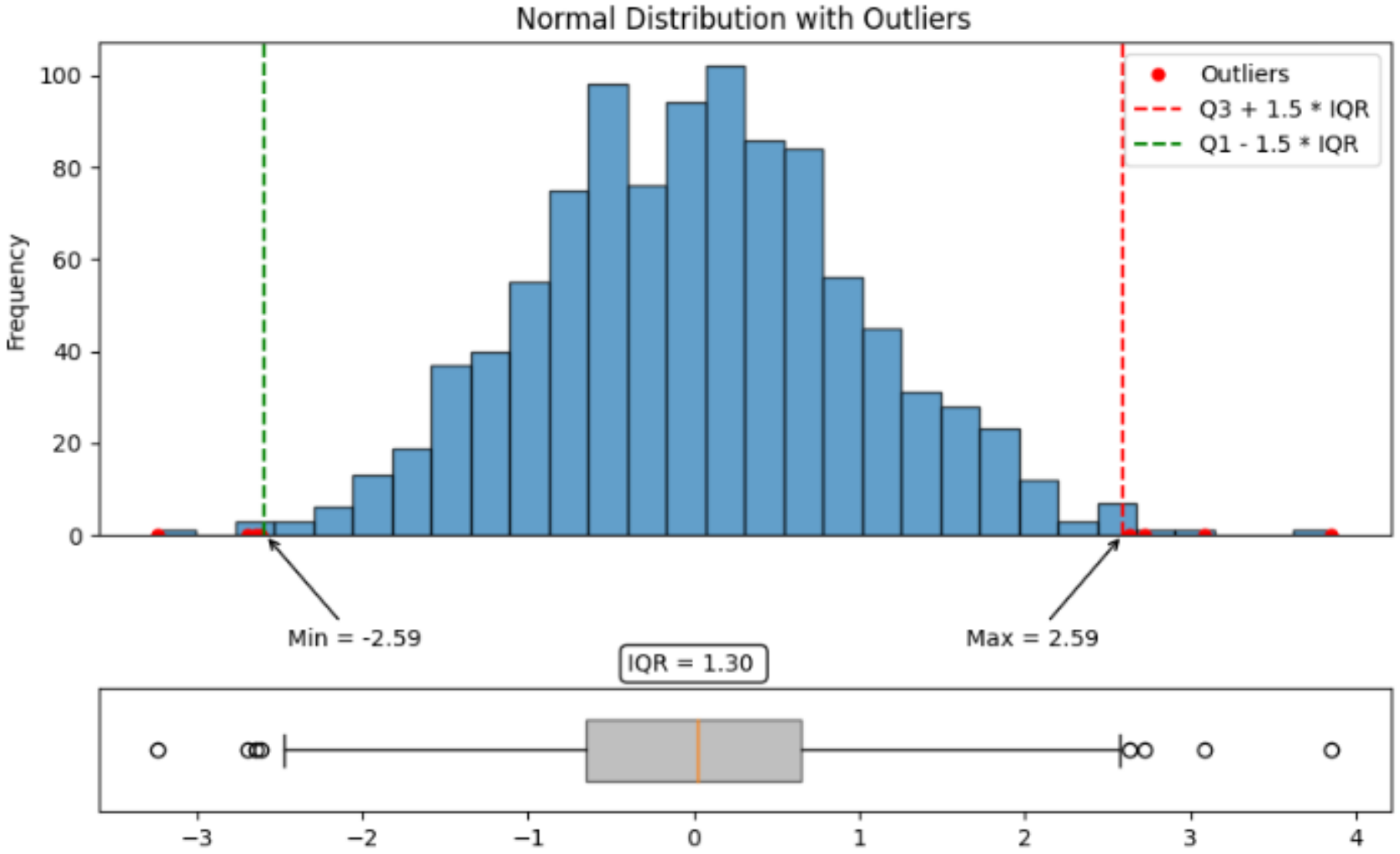


Image Source:
Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research 13, 281–305 (2012)

06. 머신러닝 수행 시 고려사항

- ◆ 이상치의 의미
  - ✓ 값이 크게 차이가 나는 경우
- ◆ 이상치 발생하는 주요 원인
  - ✓ 결측치로 표시할 때 (예: error, 9999)
  - ✓ 자료 수집의 오류 시, (실내 온도 측정 예: 40000)
  - ✓ 실제 관측치, (월급 예: 100만원, 1억원)
- ◆ 이상치 처리
  - ✓ 실무에서 표준화된 정답은 없음
  - ✓ 통계적인 방법 : IQR(Inter Quantile Range) 방식

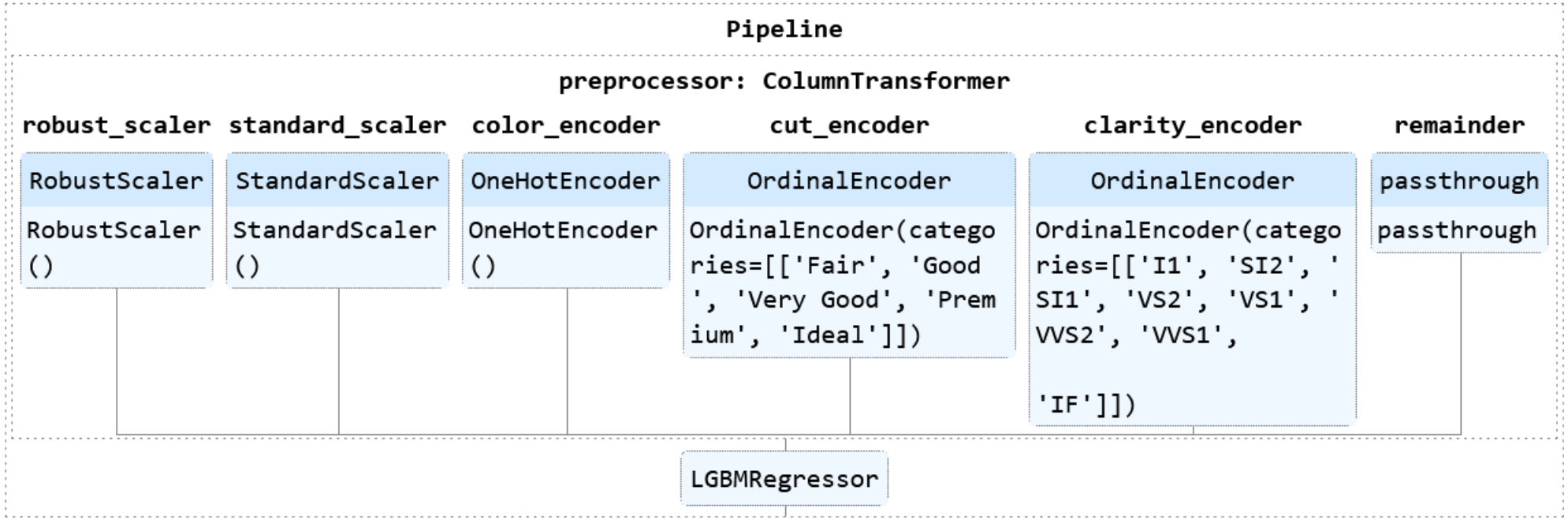
06. 머신러닝 수행 시 고려사항



06. 머신러닝 시 고려사항

◆ 모형 자동화

✓ sklearn.pipeline 과 sklearn.compose 모듈을 적극 활용



시계열 분석 모델

with  python™

01. 실습을 위한 시계열의 주요 개념

◆ ACF와 PACF

- ✓ 자기 상관은 자기자신의 다른시점과의 상관관계를 말하는 것
- ✓ 시차를 적용한 상관관계를 의미
- ✓ ARIMA 모델의 p와 q를 결정하는 데 활용함

ACF(Autocorrelation Function)	PACF(Partial Autocorrelation Function)
• 시차에 따른 일련의 자기상관 • 시차가 커질수록 ACF는 0에 가까워짐 • 시계열의 정상성을 평가할 때 유용 ✓ 정상 시계열 : ACF는 빠르게 0으로 접근 ✓ 비정상 시계열 : ACF는 천천히 0으로 접근	• 시차가 다른 두 시계열 데이터 간의 순수한 상호 연관성 • 두 시점 사이에 포함된 모든 영향은 제거됨

01. 실습을 위한 시계열의 주요 개념

◆ ACF와 PACF

✓ 시계열을 가장 잘 설명하는 모델의 결정

Model	ACF	PACF
AR(p)	점차 감소하여 0에 접근	시차 q 이후에 빠르게 감소
MA(q)	시차 q 이후에 빠르게 감소	점차 감소하여 0에 접근
ARMA(p, q)	가파른 절단 없음	가파른 절단 없음

◆ ADF 검정 (Augmented Dickey Fuller)

✓ 시계열을 가장 잘 설명하는 모델의 결정

✓ 단위근 (Unit root test) 검정을 통해 시계열의 정상성 여부를 판정

✓ 귀무가설 : 시계열에 단위근이 존재함 (비정상성 시계열)

✓ 대립가설 : 시계열이 정상성을 만족한다. (p-value < 0.05)

02. 시계열 데이터 개요

◆ 시계열 데이터(time series data)는 일정한 시간 간격으로 순차적으로 기록된 데이터의 집합

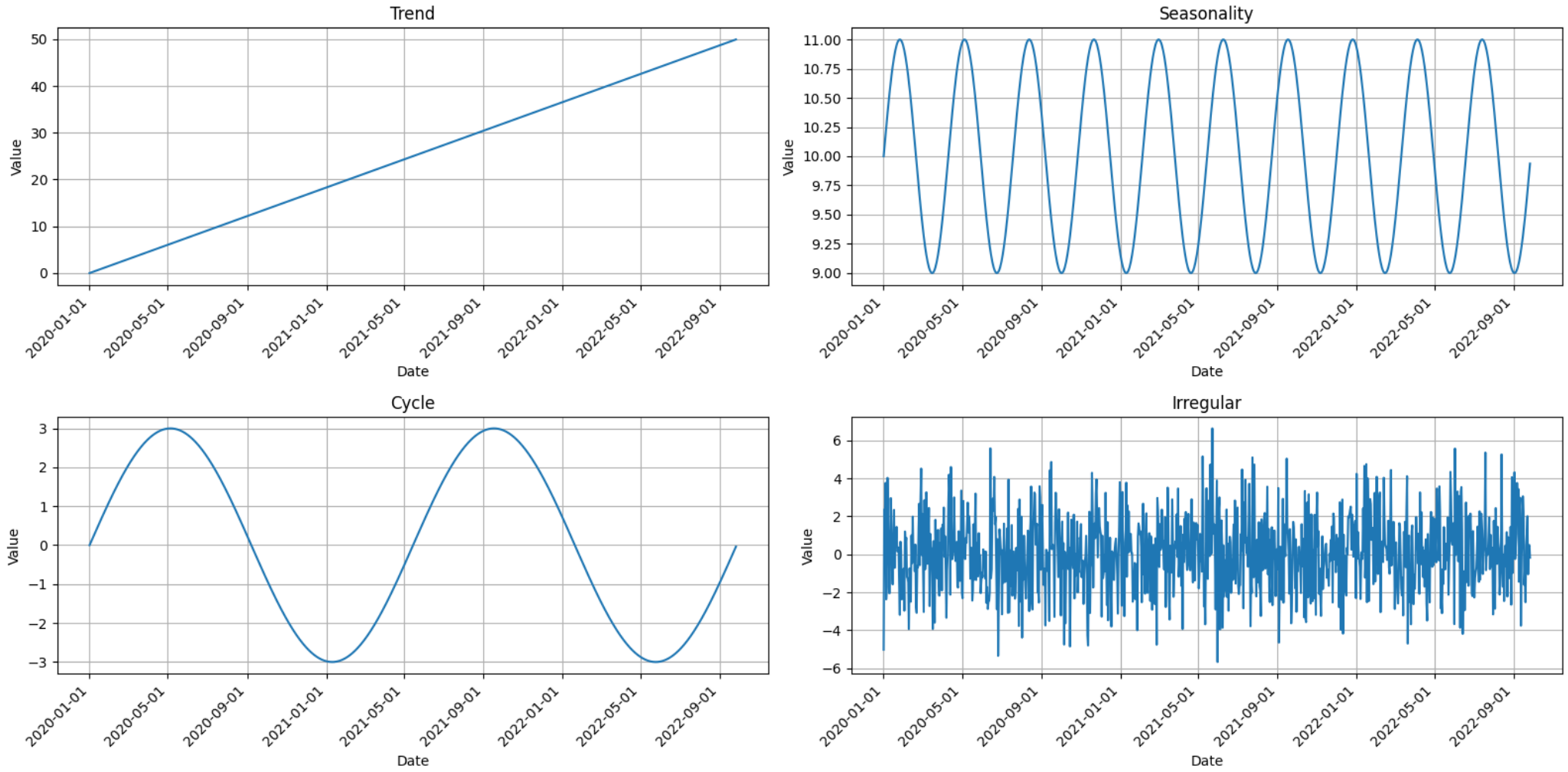
✓ 각 데이터 포인트들은 시간 순서대로 연결되어 있음

✓ 과거의 데이터가 미래의 데이터에 영향을 미칠 수 있음

주요 특성	설명
계절요인 (Seasonal Factor)	• 연간, 월간, 주간 등 일정한 주기를 가지고 반복되는 패턴이 있을 수 있음 • 예: 추석, 설, 크리스마스
추세요인 (Trend Factor)	• 장기적인 시간 동안 증가, 감소 또는 안정적인 경향을 보일 수 있음 • 예: GDP, 인구증가율, 출산율
순환요인 (Cycle Factor)	• 특정 주기 혹은 수년 간의 간격으로 발생하는 주기적인 패턴 • 예: 경기 변동성
불규칙요인 (Irregular Factor)	• 예측할 수 없는 또는 설명할 수 없는 요인에 의한 우연한 패턴(예측 불가), noise으로 간주됨 • 예: 전쟁, 홍수, 전염병

02. 시계열 데이터 개요

◆ 시계열 데이터(time series data)는 일정한 시간 간격으로 순차적으로 기록된 데이터의 집합



02. 시계열 데이터 개요

◆ 정상성(Stationary)

✓ **시계열분석을 하기 위해서는 기본적으로 평균, 분산, 공분산 및 기타 모든 분포적 특성이 일정한 성질인 정상성 만족**

구분	설명
기본 조건	• 평균(Mean)은 시점에 관계 없이 일정 • 분산(Variance)은 시점에 관계 없이 일정 • 공분산(Covariance)은 시계열 내의 특정 시점에 의존하지 않음, 다만 시차에만 의존 • 계절성 또는 주기성이 없음
정상성 가정 효과	• 모델링 단순화, ARIMA 모델은 정상성을 가정함. 즉 정상성을 만족하지 못하면 ARIMA 모델 사용 불가 • 신뢰할 수 있는 통계적 추론, 모델에 의해 추정된 파라미터들이 시간에 따라 일관된 신뢰 보장
정상성 검정	• Augmented Dickey-Fuller(ADF) 검정 : 대표적인 정상성 검정
정상성 변환방법	• 추세가 안 보이거나 평균이 일정하지 않을 시: 차분(Difference)을 통해서 가공 • 분산이 일정하지 않은 경우: 로그 변환, 제곱근 변환, 멱 변환 등을 통해 시계열 가공 • 추세 제거 : 데이터에서 추세 성분을 제거 • 분해 : 시계열에서 계절성과 추세 성분을 분리하고 제거

02. 시계열 데이터 개요

◆ 시계열 데이터 예측 주요 알고리즘 : 알고리즘 코드 적용 하기

< 전통 시계열 >

ARIMA

< Tree-Boosting >

LightGBM

< Facebook >

Prophet

03. 시계열 데이터 예측 – ARIMA

◆ ARIMA(AutoRegressive Integrated Moving Average)

- ✓ AR(자기회귀), I(차분), MA(이동평균)의 합성, 기존 AR, MA, ARMA 모델의 경우 데이터가 정상성이어야 함
- ✓ 올바른 AR 및 MA 모델의 차수, I(차분)의 횟수 결정하는 것이 ARIMA 모델의 핵심
- ✓ 모델 평가 : Akaike 정보 기준 활용
- ✓ sktime 라이브러리 : 각 차수 및 횟수를 자동으로 찾아줌
  - https://github.com/sktime/sktime

Welcome to sktime

> A unified interface for machine learning with time series

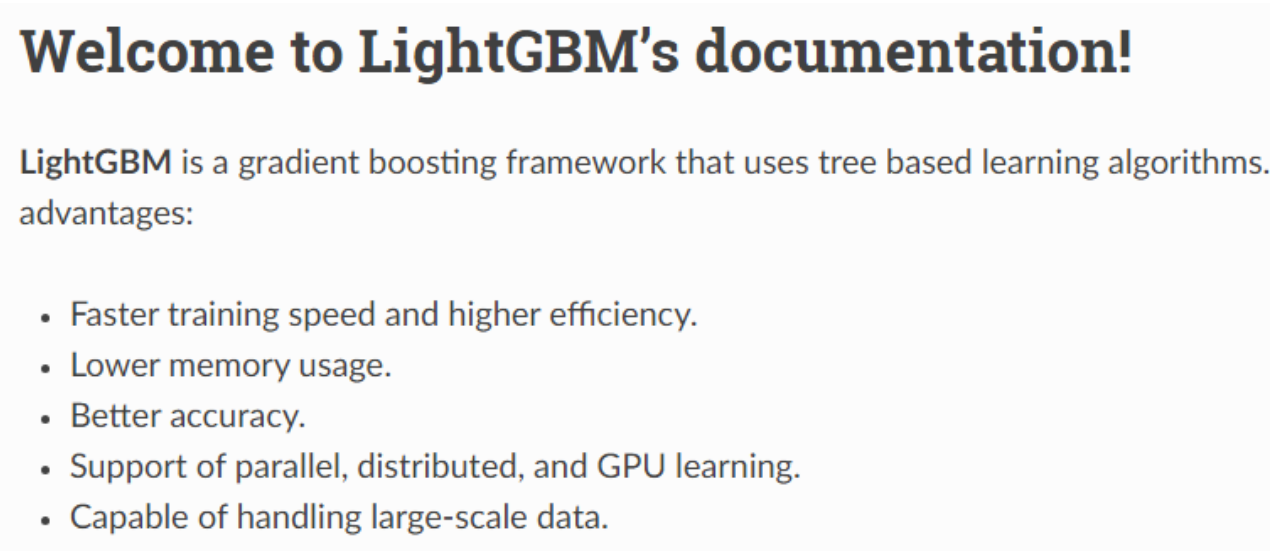
🚀 **Version 0.24.1 out now!** Check out the release notes here.

sktime is a library for time series analysis in Python. It provides a unified interface for multiple time series learning tasks. Currently, this includes time series classification, regression, clustering, annotation, and forecasting. It comes with time series algorithms and scikit-learn compatible tools to build, tune and validate time series models.

03. 시계열 데이터 예측 - LightGBM

◆ LightGBM(Light Gradient Boosting Machine)

- ✓ MS에서 개발한 Gradient Boosting Framework, 대규모 데이터셋을 빠르고 효율적으로 처리
- ✓ 기존 Boosting 모델보다 메모리 사용량이 적고, 실행 속도가 빠르며, 더 높은 정확도를 달성함
- ✓ 분류 및 다중분류 문제, 수치 예측 문제에 효과적
- ✓ 소규모 데이터셋에서의 과적합(Overfitting), 파라미터 튜닝 설계 시, 복잡할 수 있음
- ✓ 그 외 상세 내용 : https://lightgbm.readthedocs.io/en/stable/



Welcome to LightGBM's documentation!

LightGBM is a gradient boosting framework that uses tree based learning algorithms.
advantages:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel, distributed, and GPU learning.
- Capable of handling large-scale data.

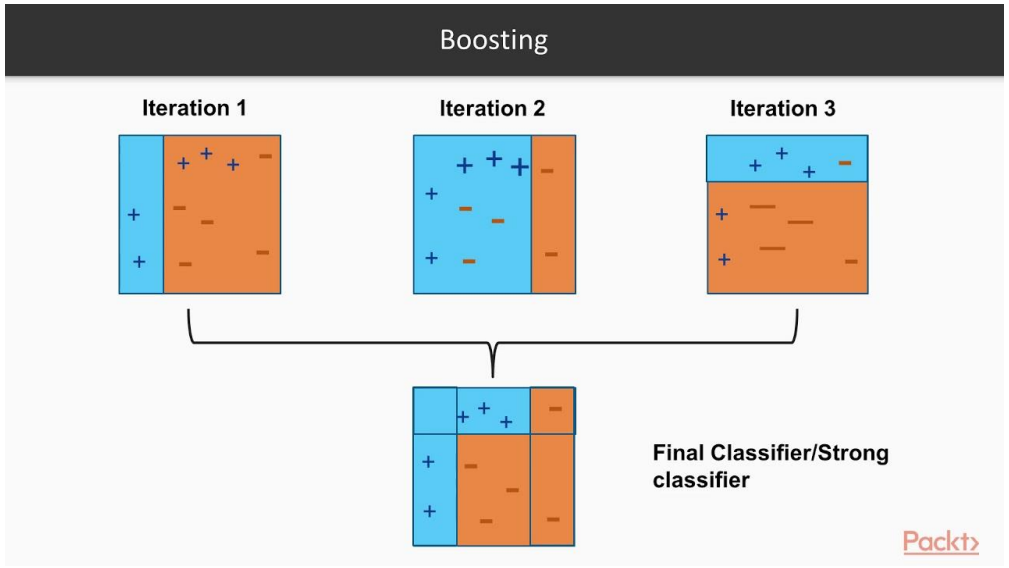
LightGBM

03. 시계열 데이터 예측 - LightGBM

◆ 머신러닝 모형 : LightGBM(Light Gradient Boosting Machine)

◆ GBM의 기본 원리

- ✓ 첫번째 단계의 Iteration 1을 활용하여 Y 예측
- ✓ 발생한 잔차(residual)을 다시 Iteration 2의 input으로 넣어주고 다시 예측
- ✓ 발생한 잔차(residual)을 다시 Iteration 3의 input으로 넣어주고 다시 예측
- ✓ residual은 점차 계속 작아질 것임

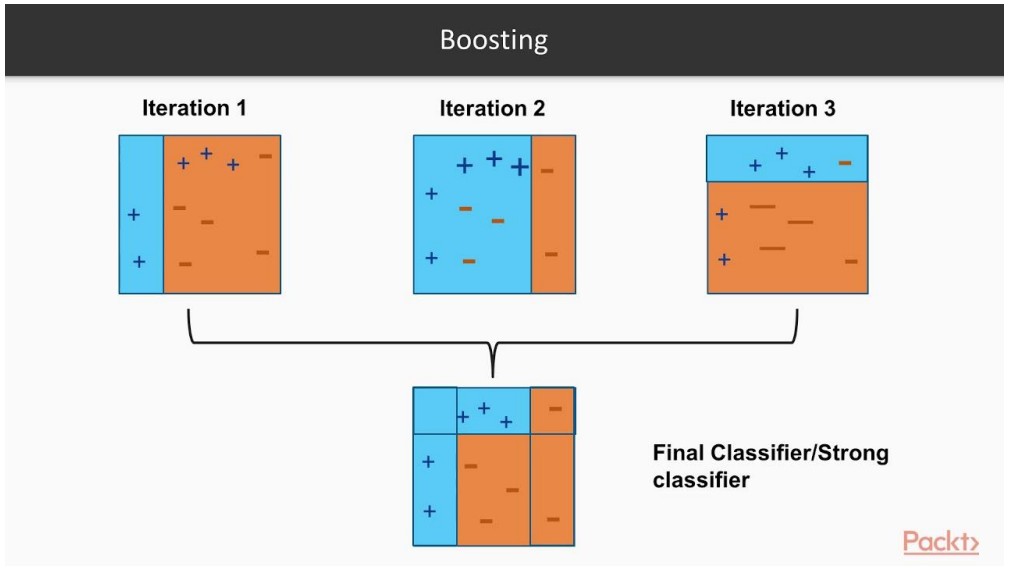


03. 시계열 데이터 예측 - LightGBM

◆ 머신러닝 모형 : LightGBM(Light Gradient Boosting Machine)

◆ GBM의 기본 원리

- ✓ 첫번째 단계의 Iteration 1을 활용하여 Y 예측
- ✓ 발생한 잔차(residual)을 다시 Iteration 2의 input으로 넣어주고 다시 예측
- ✓ 발생한 잔차(residual)을 다시 Iteration 3의 input으로 넣어주고 다시 예측
- ✓ residual은 점차 계속 작아질 것임
- ✓ 이렇게 만들어진 Iteration 1 + Iteration 2 + Iteration 3이 우리의 GBM이 됨



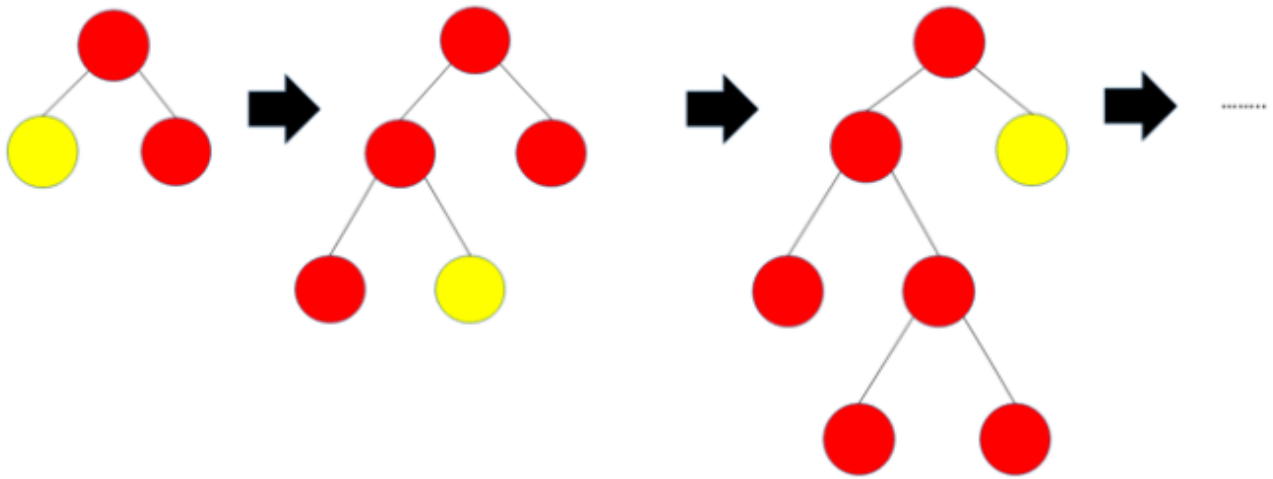
03. scikit-learn

◆ 머신러닝 모형 : LightGBM(Light Gradient Boosting Machine)

◆ 기본원리

✓ 높은 성능 : LightGBM은 대용량 데이터셋에서도 빠른 학습과 예측 제공, 적은 메모리 사용

✓ Leaf-Wise 트리 분할 : 트리의 균형을 유지하지 않고 최대손실을 갖는 리프 노드 우선 분할하여 더 정확한 예측 가능



Leaf-wise tree growth

03. 시계열 데이터 예측 - LightGBM

◆ 머신러닝 모형 : LightGBM(Light Gradient Boosting Machine)

◆ 하이퍼 파라미터 튜닝 주요 매개변수

주요 매개변수	설명
learing_rate	• 데이터 타입(기본값) : float(default = 0.1), • 각 부스팅의 반복 시, 분류기에 적용되는 가중치
n_estimators	• 데이터 타입(기본값) : int(default=100) • 설명 : LightGBM에서 부스팅 반복 횟수.
max_depth	• 데이터 타입(기본값) : int(default=-1) • 트리 모델의 최대 깊이를 제한. 이는 데이터가 작을 때 과적합을 처리하는 데 사용됨.
num_leaves	• 데이터 타입(기본값) : int(default=31) • 한 트리의 최대 잎 수
min_split_gain	• 데이터 타입(기본값) : double(default = 0.0) • 분할을 수행하기 위한 최소 이득 / 훈련 속도를 높이는데 사용
subsample	• 데이터 타입(기본값) : double(default = 1.0) • 리샘플링 없이 데이터의 일부를 무작위로 선택 • 모형학습 시간 및 과적합을 제어하는데 사용함
random_state	• 데이터 타입(기본값) : int(default=None) • 추정량의 임의성을 제어함

03. 시계열 데이터 예측 - Prophet

◆ Prophet

- ✓ Facebook에서 개발한 오픈 소스 시계열 예측 라이브러리
- ✓ 계절성을 갖는 비즈니스 시계열 데이터에 대한 예측 수행 시 강점을 가지고 있음
- ✓ 모델의 주요 구성 요소 : Trend, Seasonality, Holiday

주요 특징	설명
사용의 용이성	Prophet은 복잡한 시계열 모델링에 대한 전문 지식이 없는 사용자도 쉽게 고품질의 예측 설계 가능
계절성 인식	연간, 주간, 일간 계절성을 자동으로 감지하고 모델링
휴일 및 이벤트 처리	휴일과 특별 이벤트의 영향을 모델에 포함시킬 수 있음
추세 변화 포인트 감지	자동으로 시계열 데이터 내의 추세 변화 포인트를 감지하고 이를 모델에 반영
결과 해석 용이성	Prophet의 결과는 해석하기 쉽게 설계되어 있어, 비전문가도 결과를 이해하고 비즈니스 결정에 활용

03. 시계열 데이터 예측 - Prophet

◆ Prophet

- ✓ Facebook에서 개발한 오픈 소스 시계열 예측 라이브러리
- ✓ 계절성을 갖는 비즈니스 시계열 데이터에 대한 예측 수행 시 강점을 가지고 있음
- ✓ 모델의 주요 구성 요소 : Trend, Seasonality, Holiday
- ✓ 그 외 상세 내용 : https://facebook.github.io/prophet/