DALC AI Study 2wk

DONGDUK AI LEARING CREW AI STUDY

TIME: PM 8~9

contents

Part1 | Intro to skit-learn

- Skit-learn library
- Data pre-processing
- Cross Validation

Part2 | regression

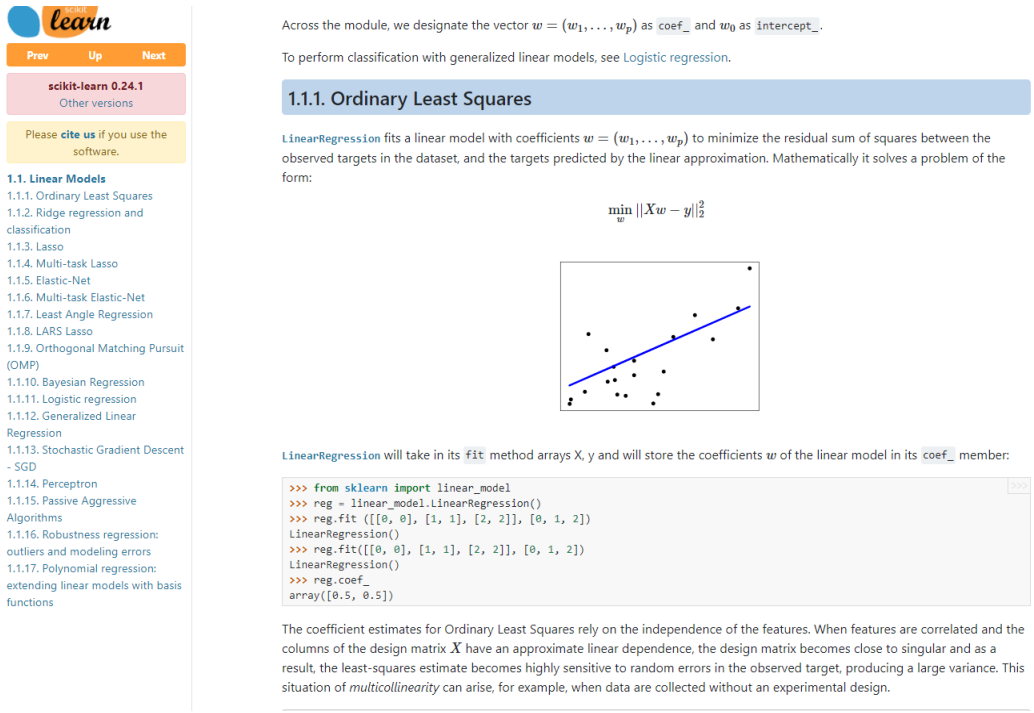
- Regression
- Linear regression, Lasso, Ridge , Elastic

Part1 | Intro to skit-learn

- Skit-learn library
- Data preprocessing
- Cross Validation

Scikit-learn 라이브러리 소개

머신러닝 에 가장 많이 사용되는 라이브러리



scikit-learn

Prev Up Next

scikit-learn 0.24.1
Other versions

Please cite us if you use the software.

1.1. Linear Models
1.1.1. Ordinary Least Squares
1.1.2. Ridge regression and classification
1.1.3. Lasso
1.1.4. Multi-task Lasso
1.1.5. Elastic-Net
1.1.6. Multi-task Elastic-Net
1.1.7. Least Angle Regression
1.1.8. LARS Lasso
1.1.9. Orthogonal Matching Pursuit (OMP)
1.1.10. Bayesian Regression
1.1.11. Logistic regression
1.1.12. Generalized Linear Regression
1.1.13. Stochastic Gradient Descent - SGD
1.1.14. Perceptron
1.1.15. Passive Aggressive Algorithms
1.1.16. Robustness regression: outliers and modeling errors
1.1.17. Polynomial regression: extending linear models with basis functions

Across the module, we designate the vector $w = (w_1, \ldots, w_p)$ as `coef_` and w_0 as `intercept_`.

To perform classification with generalized linear models, see Logistic regression.

1.1.1. Ordinary Least Squares

`LinearRegression` fits a linear model with coefficients $w = (w_1, \ldots, w_p)$ to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation. Mathematically it solves a problem of the form:

$$\min_w ||Xw - y||_2^2$$

`LinearRegression` will take in its `fit` method arrays X, y and will store the coefficients w of the linear model in its `coef_` member:

```
>>> from sklearn import linear_model
>>> reg = linear_model.LinearRegression()
>>> reg.fit ([[0, 0], [1, 1], [2, 2]], [0, 1, 2])
LinearRegression()
>>> reg.fit([[0, 0], [1, 1], [2, 2]], [0, 1, 2])
LinearRegression()
>>> reg.coef_
array([0.5, 0.5])
```

The coefficient estimates for Ordinary Least Squares rely on the independence of the features. When features are correlated and the columns of the design matrix X have an approximate linear dependence, the design matrix becomes close to singular and as a result, the least-squares estimate becomes highly sensitive to random errors in the observed target, producing a large variance. This situation of *multicollinearity* can arise, for example, when data are collected without an experimental design.

-> 해당 홈페이지

Conda install scikit-learn
Pip install scikit-learn

import sklearn

Scikit-learn 모듈

더 자세한 모듈들 소개 (사이킷럿 홈페이지)

분류	모듈명	설명
예제 데이터	sklearn.datasets	사이킷런에 내장되어 예제로 제공하는 데이터 세트
데이터 분리, 검증 & 파라미터 튜닝	sklearn.model_selection	교차 검증을 위한 학습용/테스트용 분리, 그리드 서치(Grid Search)로 최적 파라미터 추출 등의 API 제공
피처 처리	sklearn.preprocessing	데이터 전처리에 필요한 다양한 가공 기능 제공(문자열을 숫자형 코드 값으로 인코딩, 정규화, 스케일링 등)
	sklearn.feature_selection	알고리즘에 큰 영향을 미치는 피처를 우선순위 대로 셀렉션 작업을 수행하는 다양한 기능 제공
	sklearn.feature_extraction	텍스트 데이터나 이미지 데이터의 벡터화된 피처를 추출하는 데 사용됨. 예를 들어 텍스트 데이터에서 Count Vectorizer 나 Tf-Idf Vectorizer 등을 생성하는 기능 제공. 텍스트 데이터의 피처 추출은 sklearn.feature_extraction.text 모듈에, 이미지 데이터의 피처 추출은 sklearn.feature_extraction.image 모듈에 지원 API가 있음.
피처 처리 & 차원 축소	sklearn.decomposition	차원 축소와 관련한 알고리즘을 지원하는 모듈임. PCA, NMF, Truncated SVD 등을 통해 차원 축소 기능을 수행할 수 있음

분류	모듈명	설명
평가	sklearn.metrics	분류, 회귀, 클러스터링, 페어와이즈(Pairwise)에 대한 다양한 성능 측정 방법 제공 Accuracy, Precision, Recall, ROC-AUC, RMSE 등 제공
ML 알고리즘	sklearn.ensemble	앙상블 알고리즘 제공 랜덤 포레스트, 에이다 부스트, 그래디언트 부스팅 등을 제공
	sklearn.linear_model	주로 선형 회귀, 릿지(Ridge), 라쏘(Lasso) 및 로지스틱 회귀 등 회귀 관련 알고리즘을 지원. 또한 SGD(Stochastic Gradient Descent) 관련 알고리즘도 제공
	sklearn.naive_bayes	나이브 베이즈 알고리즘 제공. 가우시안 NB , 다항 분포 NB 등.
	sklearn.neighbors	최근접 이웃 알고리즘 제공. K-NN 등
	sklearn.svm	서포트 벡터 머신 알고리즘 제공
	sklearn.tree	의사 결정 트리 알고리즘 제공
	sklearn.cluster	비지도 클러스터링 알고리즘 제공 (K-평균, 계층형, DBSCAN 등)
유틸리티	sklearn.pipeline	피처 처리 등의 변환과 ML 알고리즘 학습, 예측 등을 함께 묶어서 실행할 수 있는 유틸리티 제공

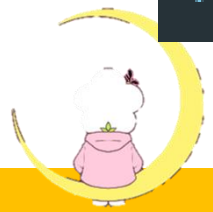
Scikit-learn 모듈 실행 예시

```
import numpy as np                                  ## 기초 수학 연산 및 행렬계산
import pandas as pd                                 ## 데이터프레임 사용
from sklearn import datasets                        ## iris와 같은 내장 데이터 사용
from sklearn.model_selection import train_test_split    ## train, test 데이터 분할

from sklearn.linear_model import LinearRegression       ## 선형 회귀분석
from sklearn.linear_model import LogisticRegression     ## 로지스틱 회귀분석
from sklearn.naive_bayes import GaussianNB              ## 나이브 베이즈
from sklearn import svm                                 ## 서포트 벡터 머신
from sklearn import tree                                ## 의사결정나무
from sklearn.ensemble import RandomForestClassifier     ## 랜덤포레스트

import matplotlib.pyplot as plt                     ## plot 그릴때 사용
```

```
# 분류용 가상 데이터 만들기
from sklearn.datasets import make_classification
```



Scikit-learn 연습용 데이터 셋

[연습용 데이터셋 종류 (사이킷런 홈페이지)]

```python
import pandas as pd
from sklearn.datasets import load_iris

iris = load_iris()

print("들어 있는 key들", iris.keys())

iris_data = iris.data
iris_label = iris.target

# 독특하게 내장 데이터셋이 df로 들어 있는게 아니라 ndarray로 들어있으며
# target, target_names, feature_names가 별도로 iris의 키로 들어 있음
print("data type", type(iris_data))
print("target 값", iris_label)
print("target 명", iris.target_names)

# 따라서 이를 df로 만들기 위해서 별도의 가공이 필요함
iris_df = pd.DataFrame(iris_data, columns=iris.feature_names)
iris_df['label'] = iris.target
iris_df.head()
```


import pandas as pd
from sklearn.datasets import load_iris

iris = load_iris() print("들어 있는 key들", iris.keys())
iris_data = iris.data iris_label = iris.target

독특하게 내장 데이터셋이 df로 들어 있는게 아니라 ndarray로 들어있으며
target, target_names, feature_names가 별도로 iris의 키로 들어 있음

print("data type", type(iris_data))
print("target 값", iris_label)
print("target 명", iris.target_names)
# 따라서 이를 df로 만들기 위해서 별도의 가공이 필요함

iris_df = pd.DataFrame(iris_data, columns=iris.feature_names)
iris_df['label'] = iris.target
iris_df.head()

결과:

```
들어 있는 key들 dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])
data type <class 'numpy.ndarray'>
target 값 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
target 명 ['setosa' 'versicolor' 'virginica']
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	label
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

Part2 | regression

- Regression
- Linear regression, Lasso, Ridge , Elastic

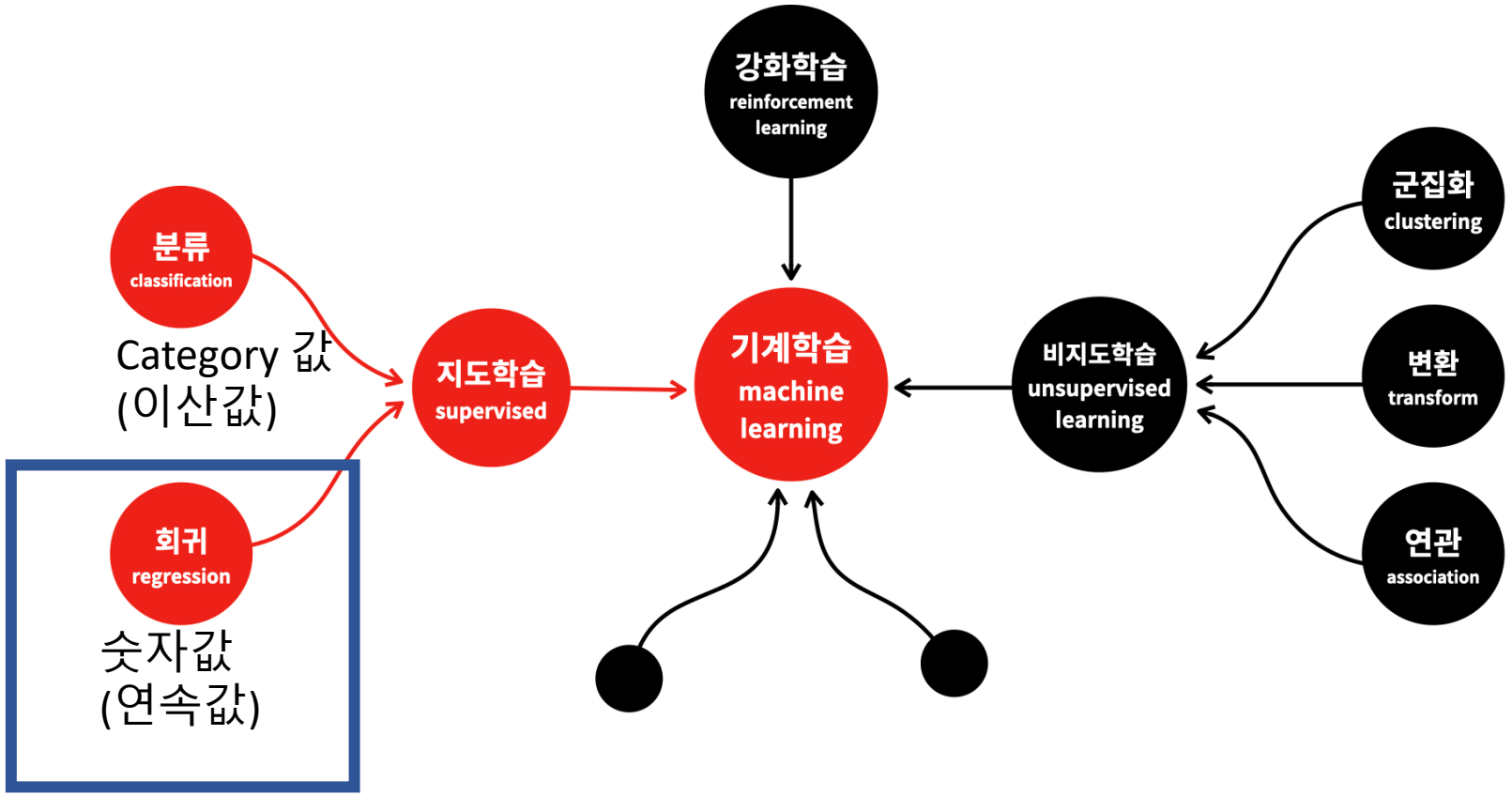
Regression

분류 구현 클래스

DesicionTreeClassifier
RandomForestClassifier
GradientBoostingClassifier
GaussianNB
SVC

회귀 구현 클래스

LinearRegreesion
Ridge
Lasso
RandomForestRegressor
GradientBoostingRegreesor



example

지하철 역과의 거리

일정 거리안의 관공서

마트

학군의 수

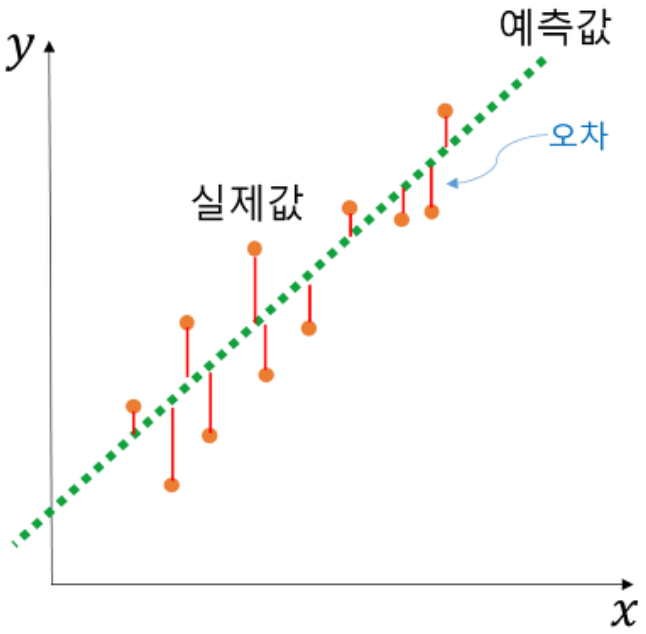
} 집 값

엄마 키

아빠 키

} 자식 키

선형회귀



1. 단순 선형 회귀

$$y = Wx + b$$

2. 다중 선형 회귀

$$y = W_1x_1 + W_2x_2 + \ldots W_nx_n + b$$

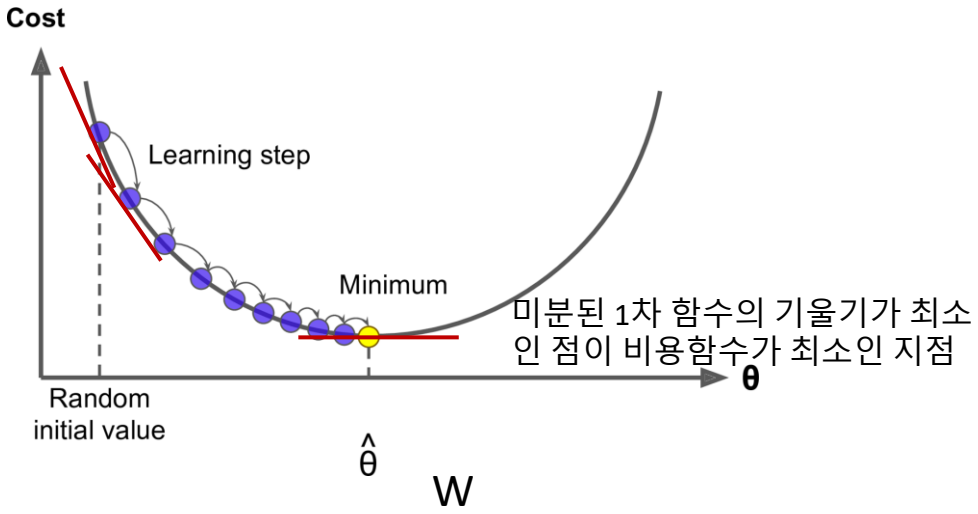
최적의 회귀 계수를 찾아내는 것이 핵심

최적의 회귀 계수 찾기

Residual Sum of Square

Cost Function(비용함수),
Loss function (손실함수)

$$\min \quad RSS(w_0, w_1) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (w_0 + w_1 * x_i))^2$$



미분된 1차 함수의 기울기가 최소인 점이 비용함수가 최소인 지점

사용된 idea

Ex. 2차 함수의 최저점 구하기>
2차함수의 미분 값인 1차 함수의 기울기(접선의 기울기)가 가장 작은 값을 찾는다.

변수가 여러 개? 편미분 사용

$$\frac{\partial R(w)}{\partial w_1} = \frac{2}{N}\sum_{i=1}^{N} -x_t * (y - (w_0 + w_1 x_i)) = -\frac{2}{N}\sum_{i=1}^{N} x_t * (\text{실제값}_i - \text{예측값}_i)$$

$$\frac{\partial R(w)}{\partial w_0} = \frac{2}{N}\sum_{i=1}^{N} -(y_i - (w_0 + w_i x_i) = -\frac{2}{N}\sum_{i=1}^{N} (\text{실제값}_i - \text{예측값}_i)$$

Gradient descent process

step1

w_1, w_0를 임의의 값으로 설정하고 첫 비용 함수의 값을 계산합니다.

step2

w_1을 $w_1 - \eta \frac{2}{N} \sum_{i=1}^{N} x_t * (\text{실제값}_i - \text{예측값}_i)$,

w_0을 $w_0 - \eta \frac{2}{N} \sum_{i=1}^{N} (\text{실제값}_i - \text{예측값}_i)$ 으로 업데이트한 후

다시 비용 함수의 값을 계산합니다.

step3

비용 함수의 값이 감소했으면 다시 Step 2를 반복합니다. 더 이상 비용 함수의 값이 감소하지 않으면 그때의 w_1, w_0를 구하고 반복을 중지합니다.

회귀 평가 지표

평가 방법	사이킷런 평가 지표 API	Scoring 함수 적용 값	수식
MAE	metrics.mean_absolute_error	'neg_mean_absolute_error'	$MAE = \frac{\sum \lvert y - \hat{y} \rvert}{n}$
MSE	metrics.mean_squared_error	'neg_mean_squared_error'	$MSE = \frac{\sum (y - \hat{y})^2}{n}$
R^2	metrics.r2_score	'r2'	$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$

사이킷 런은 RMSE를 제공 x

다항 회귀

```
from sklearn.preprocessing import PolynomialFeatures

X = [[2,3]]
print(X)
poly = PolynomialFeatures()
poly.fit(X)
print(poly.transform(X))

# fit과 transform 매서드를 하나로 붙인 매서드 fit_transform
poly.fit_transform(X)
```

```
[[2, 3]]
[[1. 2. 3. 4. 6. 9.]]
array([[1., 2., 3., 4., 6., 9.]])
```

$1, x_1, x_2, x_1^2, x_1x_2, x_2^2$

-다항 회귀도 선형회귀
: 선형/비선형 회귀를 나누는 기준은 회귀 계수가 선형/비선형인지에 따른 것이지 독립변수의 선형/ 비선형 여부와는 무관

규제

비용함수 목표

1. 학습데이터 잔차 최소화
2. 회귀 계수 크기 제어 (과적합 방지)

$$RSS(W) + alpha2 * ||W||_2^2 + alpha1 * ||W||_1$$

Ridge
(L1방식)

Lasso
(L2방식)

ElasticNet
(L1방식+L2방식)

규제

규제 (Regularization)

학습이 과대적합 되는 것을 방지하고자 일종의 **penalty**를 부여하는 것

L2 규제 (L2 Regularization)

- 각 가중치 제곱의 합에 규제 강도(Regularization Strength) λ를 곱한다.
- λ를 크게 하면 가중치가 더 많이 감소되고(규제를 중요시함), λ를 작게 하면 가중치가 증가한다(규제를 중요시하지 않음).

L1 규제 (L1 Regularization)

- 가중치의 제곱의 합이 아닌 **가중치의 합**을 더한 값에 규제 강도(Regularization Strength) λ를 곱하여 오차에 더한다.
- 어떤 가중치(w)는 실제로 0이 된다. 즉, 모델에서 완전히 제외되는 특성이 생기는 것이다.

L2 규제가 L1 규제에 비해 더 안정적이라 일반적으로는 L2규제가 더 많이 사용된다

Regression algorithm

Solver 라든지 다른 매개변수도 있지만 기본적인 것만 정리함

클래스명	매개변수		속성		특징
LinearRegression()	fit_intercept	defalut = True 절편을 계산할 것인지 결정	coef_	fit()후 회귀계수 값을 저장	lasso, ridge, elasticnet도 linearregression이 가진 매개변수 가지고있음.
	normalize	defalut = False 회귀를 수행하기 전 데이터 세트 정규화	intercept_	intercept	
PolynomialFeatures()	degree	단항식 피처를 degree에 해당하는 다항식 피처로 변환			
Ridge()	alpha	L2 규제 계수	coef_, intercept_		계수의 값을 작게해 과적합을 개선
Lasso()	alpha	L1규제 계수	coef_, intercept_		영향력이 크지 않은 계수의 값을 0으로 변환
ElasticNet()	alpha	a + b	coef_, intercept_		엘라스틱 넷 규제 = a * L1 + b* L2
	l1_ratio	a / (a+b)			
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet					

참고자료

https://teddylee777.github.io/scikit-learn/scikit-learn-linear-with-regularizations
파이썬 머신러닝 완벽가이드