* sklearn.preprocessing.StandardScaler

<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler.fit_transform>

mean을 제거하고 unit variance을 scaling함으로써 feature들을 standardize한다.

샘플 x의 standard score : z = (x – u) / s 로 계산된다.

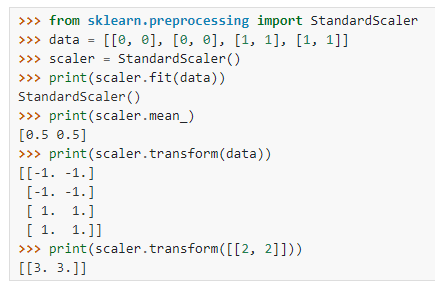
u : 트레이닝 샘플들의 평균(mean). (with\_mean=False일 때 0)

s: 트레이닝 샘플들의 standard deviation. (with\_std=False일 때 1)

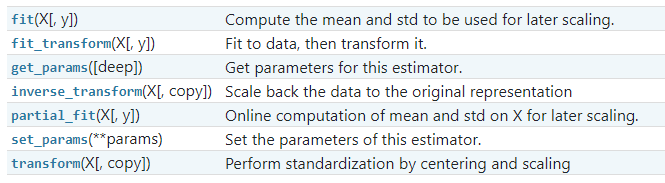
Centering과 Scaling은 각 feature들에 독립적으로 일어난다.

dataset의 Standardization은 많은 머신러닝 estimator들의 common requirement이다.

**[Examples]**



[Methods]



출처 : <https://datascienceschool.net/view-notebook/f43be7d6515b48c0beb909826993c856/>

**<scikit-learn의 전처리 기능>**

preprocessing 서브패키지에서 다양한 전처리 기능을 제공.

**-스케일링 (Scaling)**

스케일링은 자료 집합에 적용되는 전처리 과정. 모든 자료에 선형 변환을 적용하여 전체 자료의 분포를 평균 0, 분산 1이 되도록 만드는 과정.

스케일링은 자료의 overflow나 underflow를 방지하고 독립변수의 공분산 행렬의 조건수(condition number)를 감소시켜 최적화 과정에서의 안정성 및 수렴 속도를 향상시킴.

**제공하는 스케일링 클래스**

* StandardScaler(X) : 평균이 0과 표준편차가 1이 되도록 변환
* RobustScaler(X) : 중앙값(median)이 0, IQR(interquartile range)이 1이 되도록 변환
* MinMaxScaler(X) : 최대값이 각각 1, 최소값이 0이 되도록 변환
* MaxAbsScaler(X) : 0을 기준으로 절대값이 가장 큰 수가 1 또는 -1이 되도록 변환

**사용 방법**

1. 학습용 데이터의 분포 추정 : 학습용 데이터를 입력으로 하여 **fit** 메서드를 실행하면 분포 모수를 객체 내에 저장
2. 학습용 데이터 변환 : 학습용 데이터를 입력으로 하여 **transform** 메서드를 실행하면 학습용 데이터를 변환
3. 검증용 데이터 변환 : 검증용 데이터를 입력으로 하여 **transform** 메서드를 실행하면 검증용 데이터를 변환

1)번과 2)번의 과정을 합쳐서 **fit\_transform** 메서드를 사용할 수도 있다.

* fit\_transform(X[, y])
* sklearn.decomposition.PCA

<https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>

Principal component analysis(PCA).

[참고]

차원 축소 – PCA, 주성분 분석(1)

<https://excelsior-cjh.tistory.com/167>

출처 : <https://datascienceschool.net/view-notebook/f10aad8a34a4489697933f77c5d58e3a/>

데이터 간의 변이(variation)은 무작위가 아니라 특정한 규칙에 의해 만들어지는 경우가 있다.

데이터 간의 변이 규칙을 찾아낼 때 PCA를 이용한다.

PCA(Principal Component Analysis)는 주성분 분석이라고도 함. 고차원 데이터 집합이 주어졌을 때 원래의 고차원 데이터와 가장 비슷하면서 더 낮은 차원 데이터를 찾아내는 방법. 차원 축소(dimension reduction)이라고도 한다.

더 낮은 차원의 데이터값 변화가 더 높은 차원의 데이터값 변화를 설명할 수 있다는 것은 복잡해 보이는 고차원 데이터의 변이를 몇 가지 원인으로 설명할 수 있다는 뜻.

**사이킷런의 PCA 기능**

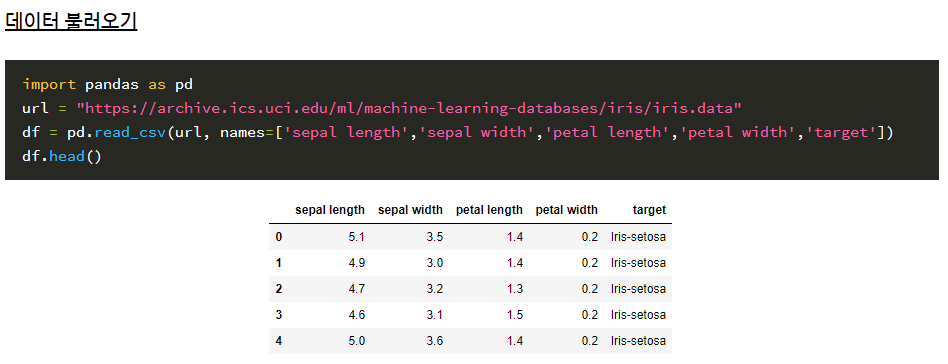
사이킷런의 decomposition 서브패키지는 PCA 분석을 위한 PCA 클래스를 제공.

**사용 방법**

* 입력 인수:
  + n\_componets : 정수 (주성분을 몇 개로 할지 결정)
* 메서드 :
  + fit\_transform() : 특징행렬을 낮은 차원의 근사행렬로 변환
  + inverse\_transform() : 변환된 근사행렬을 원래의 차원으로 복귀
* 속성 :
  + mean\_ : 평균 벡터
  + components\_ : 주성분 벡터

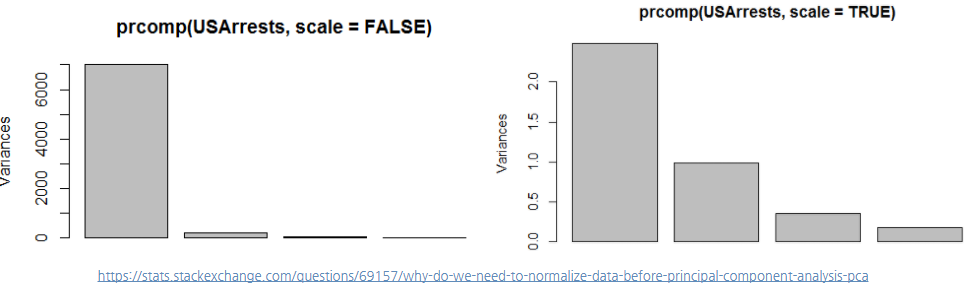
PCA(주성분 분석)\_Python 코드 포함

<https://m.blog.naver.com/PostView.nhn?blogId=tjdrud1323&logNo=221720259834&proxyReferer=https:%2F%2Fwww.google.com%2F>



**표준화**

pca를 하기 전에 데이터 스케일링을 하는 이유는 데이터의 스케일에 따라 주성분의 설명 가능한 분산량이 달라질 수 있기 때문이다.



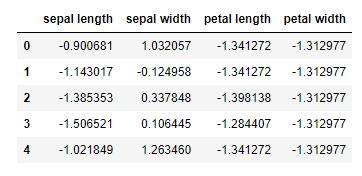
scale을 하기 전 후 설명 가능한 분산량이 다른 것을 알 수 있다.

* scale을 하지 않으면 변인이 가진 값의 크기에 따라 설명 가능한 분산량이 왜곡될 수 있기 때문에 반드시 표준화를 해주어야 함.

또한, 설명 가능한 분산량이 왜곡됨에 따라 모델 성능이 저하될 수 도 있음.

|  |
| --- |
| from sklearn.preprocessing import StandardScaler # 표준화 패키지 라이브러리  x = df.drop([‘target’], axis=1).values # 독립변인들의 value값만 추출  y = df[‘target’].values # 종속변인 추출  x = StandardScaler().fit\_transform(x) # x객체에 x를 표준화한 데이터를 저장  features = …  pd.DataFrame(x, columns=features).head() |

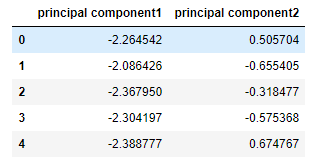
표준화된 데이터



**PCA 실행**

|  |
| --- |
| from sklearn.decomposition import PCA  pca = PCA(n\_components=2) # 주성분을 몇 개로 할지 결정  principalComponents = pca.fit\_transform(x)  principalDf = pd.DataFrame(data=principalComponents, columns = [‘component1’, ‘component2’]  # 주성분으로 이루어진 데이터프레임 구성 |

principalDf.head()



핵심) PCA의 파라미터 주성분 개수(n\_components)를 결정하여 개수에 맞는 원래 데이터에서 변환된 주성분을 추출하는 것.

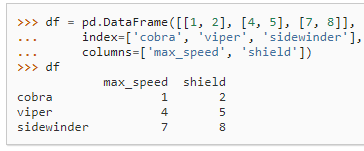
* 주성분 개수 결정의 근거는???
* pandas.DataFrame.loc

<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.loc.html>

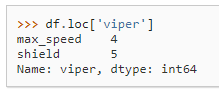
labeling

**Examples**

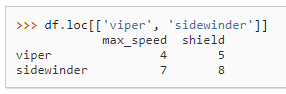
Getting values



row을 Series로 반환한다.



labels의 리스트. [[]] 은 DataFrame을 반환한다.



행과 열에 Single label



Slice with labels for row and single label for column. As mentioned above, note that both the start and stop of the slice are included.

행을 위한 라벨과 열을 위한 한 개의 라벨을 slice. slice의 시작과 stop이 모두 포함됨.

* (x, y)를 return 해주는 건가?

**데이터프레임 고급 인덱싱**

<https://datascienceschool.net/view-notebook/704731b41f794b8ea00768f5b0904512/>

데이터프레임에서 특정한 데이터만 골라내는 것을 indexing이라고 한다.

Pandas는 numpy행렬과 같이 쉼표를 사용한 (행 인덱스, 열 인덱스) 형식의 2차원 인덱싱을 지원하기 위해 다음과 같은 특별한 인덱서(indexer) 속성도 제공한다.

* loc : 라벨값 기반의 2차원 인덱싱
* iloc : 순서를 나타내는 정수 기반의 2차원 인덱싱
* at : 라벨값 기반의 2차원 인덱싱 (한 개의 스칼라 값만 찾음)
* iat : 순서를 나타내는 정수 기반의 2차원 인덱싱 (한 개의 스칼라 값만 찾음)

loc 인덱서

|  |
| --- |
| df.loc[행 인덱싱 값] |

또는

|  |
| --- |
| df.loc[행 인덱싱 값, 열 인덱싱 값 |

이때 인덱싱 값은 다음 중 하나이다.

행 인덱싱 값은 정수 또는 행 인덱스 데이터, 열 인덱싱 값은 라벨 문자열.

* 인덱스 데이터
* 인덱스 데이터 슬라이스
* 인덱스 데이터 리스트
* 같은 행 인덱스를 가지는 Boolean series (행 인덱싱의 경우)
* 또는 위의 값들을 반환하는 함수
* **EllipticEnvelope**

<https://scikit-learn.org/stable/modules/generated/sklearn.covariance.EllipticEnvelope.html>

가우시안 분포에서 outliers를 detecting하는 object

Anomaly Detection in Scikit-Learn

<https://sdsawtelle.github.io/blog/output/week9-anomaly-andrew-ng-machine-learning-with-python.html>

Novelty와 Outlier Detection

<https://flonelin.wordpress.com/2017/03/29/novelty%EC%99%80-outlier-detection/>

[Scikit-Learn] Preprocessing

<https://jjhwqqq.tistory.com/172>