Data Pre-Processing for Machine Learning

Dr. Saerom Park
Statistical Learning and Computational Finance Lab.
Department of Industrial Engineering
psr6275@snu.ac.kr
http://slcf.snu.ac.kr

Table of Contents

- 목차

Reference

- **Reading:** [Raschka. (2017), chapter 4], [Müller. (2016), chapter 4].
- Githib
  - https://github.com/SLCFLAB/Fintech
- Slack
  - https://join.slack.com/t/fintech2018/shared_invite/enQtMzA2OTk5OTIyMjc3LTZmYmIzNDNiMWY0MzU4NDE1ZmY2NTg5OTU2Y2RhYTQyZjNhZjZmYjBmNjQ2MThmNzQ5YTY4NmE0OWJlNWRjNmU

Introduction

- 기계학습 모델을 위해서는 데이터 전처리가 필요
  - 수집되는 데이터들은 다양한 형태를 가지고 있음: 연속 vs 이산, 수치 vs 범주
    - 수치 데이터 (Numerical data): 숫자로 표현되는 데이터
      - 비율(Ratio) 데이터 (ex) 시험점수, 가격
      - 간격(Interval) 데이터: 숫자간 차이의 간격이 일정 (ex) 온도
    - Categorical 데이터: 숫자로 표현되지 않은 데이터
      - 순서(Ordinal) 데이터: 정렬할 수 있거나 순서를 가짐 (ex) XL > L > M
      - 명목(Nominal) 데이터: 순서를 가지지 않음 (ex) 색깔
  - 데이터를 어떻게 표현하는지가 machine learning 알고리즘의 성능에 매우 중요
  - 학습 모델들을 적용하기 전에 전처리 수행
    - 기계학습 모델 적용을 위한 수집된 데이터의 문제 해결
    - 기계학습 모델의 성능을 높이기 위한 전처리

SCIKIT-LEARN 시작

scikit-learn 시작

- scikit-learn
  - Python으로 구현된 기계학습 오픈 소스 라이브러리
  - 기계학습 관련 여러 기법들을 간단하게 사용할 수 있다.
  - 지도학습(Supervised learning), 비지도학습(Unsupervised learning), 모델 선택 및 평가(Model selection and Evaluation), 데이터 전처리 (Data preprocessing) 등에 사용

scikit-learn 시작

- scikit-learn
  - 기본 라이브러리
    - `from sklearn import datasets`
    - `from sklearn.preprocessing import StandardScaler`
    - `from sklearn.metrics import accuracy_score`
  - 분류 알고리즘
    - `from sklearn.linear_model import Perceptron`
    - `from sklearn.linear_model import LogisticRegression`
    - `from sklearn.svm import SVC`
    - `from sklearn.tree import DecisionTreeClassifier`
    - `from sklearn.ensemble import RandomForestClassifier`
    - `from sklearn.neighbors import KNeighborsClassifier`

DEALING WITH MISSING DATA

Dealing with Missing Data

- Missing Data
  - 실제로 수집되는 데이터는 다양한 이유로 missing data를 포함하고 있다.
    - 데이터 수집 단계의 에러, 설문 조사에서의 미응답 등
  - Missing value는 주로 빈 칸 , NaN, 또는 NULL으로 표시된다.
  - 대부분의 계산 도구는 이러한 missing value를 잘 다루지 못한다.
    - 기계 학습 모델을 적용하기 전에 missing data를 먼저 처리해야 한다.
- Dealing with missing data using pandas
  - Pandas.dataframe

Dealing with Missing Data

- Missing values in Tabular Data
  - CSV 파일을 이용한 간단한 예제
  - 더 큰 데이터에 대해서는, 빈 값을 일일이 확인하기 힘들다.
    - isnull 메서드는 각 값이 missing(True)인지 아닌지를 반환해준다.

pandas.isnull

pandas.**isnull**(*obj*) [source]

Detect missing values (NaN in numeric arrays, None/NaN in object arrays)

Parameters:	**arr** : *ndarray or object value* Object to check for null-ness
Returns:	**isna** : *array-like of bool or bool* Array or bool indicating whether an object is null or if an array is given which of the element is null.

- https://pandas.pydata.org/pandas-docs/stable/generated/pandas.isnull.html

Dealing with Missing Data

- Eliminating missing values
  - Missing value를 다루는 가장 간편한 방법은 제거하는 것이다.
    - dropna 메서드는 Missing value를 포함한 행 또는 열을 모두 제거한다.

pandas.DataFrame.dropna

`DataFrame.`**dropna**(*axis=0, how='any', thresh=None, subset=None, inplace=False*) [source]

Return object with labels on given axis omitted where alternately any or all of the data are missing

Parameters:

axis : *{0 or 'index', 1 or 'columns'}, or tuple/list thereof*
Pass tuple or list to drop on multiple axes

how : *{'any', 'all'}*
- any : if any NA values are present, drop that label
- all : if all values are NA, drop that label

thresh : *int, default None*
int value : require that many non-NA values

subset : *array-like*
Labels along other axis to consider, e.g. if you are dropping rows these would be a list of columns to include

inplace : *boolean, default False*
If True, do operation inplace and return None.

Returns:

dropped : *DataFrame*

  - https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.dropna.html

Dealing with Missing Data

- Eliminating missing values
  - Missing value를 다루는 가장 간편한 방법은 제거하는 것이다.
    - 파라미터를 통해서 missing value의 수를 조절할 수 있다.

pandas.DataFrame.dropna

`DataFrame.`**dropna**(*axis=0, how='any', thresh=None, subset=None, inplace=False*) [source]

Return object with labels on given axis omitted where alternately any or all of the data are missing

Parameters:

axis : *{0 or 'index', 1 or 'columns'}, or tuple/list thereof*
Pass tuple or list to drop on multiple axes

how : *{'any', 'all'}*
- any : if any NA values are present, drop that label
- all : if all values are NA, drop that label

thresh : *int, default None*
int value : require that many non-NA values

subset : *array-like*
Labels along other axis to consider, e.g. if you are dropping rows these would be a list of columns to include

inplace : *boolean, default False*
If True, do operation inplace and return None.

Returns:

dropped : *DataFrame*

Dealing with Missing Data

- Imputing missing values
  - Missing data를 모두 제거하면 너무 많은 정보를 잃을 수 있다.
    - 간단한 외삽법을 통해서 missing value의 값을 추정할 수 있다.
    - skelarn.preprocessing.Imputer
    - Mean imputation은 missing value의 값을 각 행 또는 열의 평균값으로 추정한다.

class sklearn.preprocessing. **Imputer** (*missing_values='NaN', strategy='mean', axis=0, verbose=0, copy=True*) [source]

Imputation transformer for completing missing values.

Read more in the User Guide.

Parameters:

missing_values : integer or "NaN", optional (default="NaN")

The placeholder for the missing values. All occurrences of missing_values will be imputed. For missing values encoded as np.nan, use the string value "NaN".

strategy : string, optional (default="mean")

The imputation strategy.

- If "mean", then replace missing values using the mean along the axis.
- If "median", then replace missing values using the median along the axis.
- If "most_frequent", then replace missing using the most frequent value along the axis.

  - http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Imputer.html

HANDLING CATEGORICAL DATA

Handling Categorical Data

- 적합한 수치 데이터로의 변환
  - 보통의 기계학습 방법들은 입력으로 수치 데이터를 요구
  - 범주형 데이터는 먼저 수치 벡터 표현으로 변환되어야 함
    - Mapping
    - One-hot encoding (dummy variable)
  - 분류 문제의 출력 데이터라도 라이브러리에 따라서 수치 encoding을 요구하는 경우가 있음

Handling Categorical Data

- Mapping ordinal features
  - Ordinal feature간의 수치적 차를 안다고 생각하자.
    - XL = L+1 = M+2
  - Ordinal value를 pandas의 map 메서드를 이용해서 mapping할 수 있다.

pandas.Series.map

Series.**map**(*arg*, *na_action=None*) [source]

Map values of Series using input correspondence (which can be a dict, Series, or function)

Parameters:	**arg** : *function, dict, or Series* **na_action** : *{None, 'ignore'}* If 'ignore', propagate NA values, without passing them to the mapping function
Returns:	**y** : *Series* same index as caller

  - https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.map.html

Handling Categorical Data

- Encoding class labels
  - 대부분의 머신 러닝 라이브러리를 사용할 때 클래스 label은 정수 값을 가져야 한다.
  - 앞에서와 마찬가지로 label을 mapping 할 수 있다.
    - Label 값은 ordinal이 아니므로 어떤 값을 가지든 상관 없다.
  - LabelEncoder 클래스를 활용해서 같은 작업을 더 간편하게 할 수 있다.
    - sklearn.preprocessing.LabelEncoder

`fit_transform(y)` [source]

Fit label encoder and return encoded labels

Parameters:	**y** : array-like of shape [n_samples] Target values.
Returns:	**y** : array-like of shape [n_samples]

  - http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

Handling Categorical Data

- Performing one-hot encoding on nominal features
  - LabelEncoder을 이용해서 nominal 값 역시 변환할 수 있다.
  - 하지만, 이런 방식으로 nominal 값을 다루었을 경우, 학습 알고리즘은 'red' > 'green' > 'blue'라고 인식하게 된다.
  - one-hot 인코딩을 통해 이 문제를 해결할 수 있다.
    - sklearn.preprocessing.OneHotEncoder
    - http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html
  - 'blue'는 'blue=1, green=0, red=0'과 같이 인코딩된다.

Handling Categorical Data

- Performing one-hot encoding on nominal features
  - Pandas의 get_dummies 메서드를 통해 같은 작업을 더 간편하게 할 수 있다.

pandas.get_dummies

pandas.**get_dummies**(*data, prefix=None, prefix_sep='_', dummy_na=False, columns=None, sparse=False, drop_first=False*) [source]

Convert categorical variable into dummy/indicator variables

Parameters:

data : *array-like, Series, or DataFrame*

prefix : *string, list of strings, or dict of strings, default None*
String to append DataFrame column names Pass a list with length equal to the number of columns when calling get_dummies on a DataFrame. Alternatively, *prefix* can be a dictionary mapping column names to prefixes.

prefix_sep : *string, default '_'*
If appending prefix, separator/delimiter to use. Or pass a list or dictionary as with *prefix*.

dummy_na : *bool, default False*
Add a column to indicate NaNs, if False NaNs are ignored.

columns : *list-like, default None*
Column names in the DataFrame to be encoded. If *columns* is None then all the columns with *object* or *category* dtype will be converted.

sparse : *bool, default False*
Whether the dummy columns should be sparse or not. Returns SparseDataFrame if *data* is a Series or if all columns are included. Otherwise returns a DataFrame with some SparseBlocks.

drop_first : *bool, default False*
Whether to get k-1 dummies out of k categorical levels by removing the first level.
New in version 0.18.0.

Returns

———-

dummies : *DataFrame or SparseDataFrame*

- https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html

Handling Categorical Data

- Performing one-hot encoding on nominal features
  - One-hot 인코딩을 하면 변수 간의 correlation이 높아진다.(Multi-collinearity)
  - 이를 해결하기 위해 one-hot 인코딩 된 열 중 하나를 제거할 수 있다.
    - get_dummies 메서드의 drop_first 파라미터를 통해 수행할 수 있다.
    - 정보의 양은 줄어들지 않는다는 점을 주목하자.

PARTITIONING DATASET

Partitioning dataset into training and test sets (1/3)

- 데이터 분리
  - 기계학습 데이터는 학습 데이터와 시험 데이터로 구성됨
    - 학습 데이터: 모델 학습 시 사용
    - 시험 데이터: 모델의 성능을 측정시 사용
  - 모델 선택 시 검증 데이터도 필요
    - Cross validation과 같은 방법은 데이터를 반복적으로 분리함

Partitioning dataset into training and test sets (2/3)

- Wine dataset
  - Open-source dataset with 178 wine samples with 13 numerical features (+1 class)
  - Class label : 1, 2, 3

```
Class labels [1 2 3]
```

	Class label	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735

Partitioning dataset into training and test sets (3/3)

- scikit-learn의 model_selection 모듈을 사용하면 트레이닝 셋과 테스트 셋을 간편하게 나눌 수 있다.
  - train_test_split 메서드를 통해 feature들과 label을 각각 트레이닝 셋, 테스트 셋으로 나눈다
    - Test_size=0.3은 테스트 셋의 비율이 각각 30%임을 의미한다.
    - 파라미터 stratify는 트레이닝 셋과 테스트 셋이 원래의 데이터 셋과 같은 클래스 비율을 가지도록 강제한다.

```
sklearn.model_selection. train_test_split (*arrays, **options)        [source]
```

  - http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

SCALING

Bringing features onto the same scale (1/3)

- 대부분의 기계 학습 모델에서 입력 변수간의 scale이 성능에 영향을 미침
  - 예외) Decision tree, random forest
  - 예) 거리 기반 방법
    - 두 가지 feature 중 첫 번째는 1에서 10까지, 두 번째는 1에서 100,000까지의 값을 가질 수 있다고 생각하자.
    - 유클리드 거리를 사용하는 KNN 알고리즘을 사용할 때, 데이터 간의 측정된 거리에서 첫 번째 feature 값은 거의 반영되지 않는다.
- Feature scaling이란 위와 같은 문제를 다루기 위해 feature들이 같은 스케일을 가지도록 처리함
  - 학습 데이터를 이용해서 scaling을 수행
  - 시험 데이터는 학습 데이터와 같이 변환함

Bringing features onto the same scale (2/3)

- Min-max scaling은 각각의 값을 다음과 같은 식을 통해 변환한다.

$$x_{norm}^{(i)} = \frac{x^{(i)} - x_{min}}{x_{max} - x_{min}}$$

- sklearn.preprocessing.MinMaxScaler

fit_transform (*X, y=None, **fit_params*) [source]

Fit to data, then transform it.

Fits transformer to X and y with optional parameters fit_params and returns a transformed version of X.

Parameters:	**X** : numpy array of shape [n_samples, n_features] Training set. **y** : numpy array of shape [n_samples] Target values.
Returns:	**X_new** : numpy array of shape [n_samples, n_features_new] Transformed array.

- http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

Bringing features onto the same scale (3/3)

- Scaling된 값이 특정 범위 안에 있기를 원할 때는 standardization이 더 나은 해결책일 수 있다.
- Standardization은 다음과 같은 식을 통해 각각의 값을 변환한다.

$$x_{std}^{(i)} = \frac{x^{(i)} - \mu_x}{\sigma_x}$$

- sklearn.preprocessing.StandardScaler
  - http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

FEATURE SELECTION

Selecting meaningful features

- 모든 feature들을 다 사용하는 것이 반드시 가장 좋은 성능을 가지지는 않음
  - Overfitting 문제 발생 가능
- Overfitting
  - Overfitting이란 모델이 트레이닝 셋에 지나치게 가깝게 피팅되어 새로운 데이터에 잘 적용되지 않는 것을 의미한다.
  - Overfitting은 모델이 지나치게 복잡하기 때문에 일어난다.
  - 해결 방법
    - Regularization을 통해 모델의 복잡도를 줄임
    - 더 적은 파라미터를 가지는 단순한 모델을 선택 (feature selection)

Selecting meaningful features: Regularization (1/4)

- L2 regularization은 weight가 큰 feature에 페널티를 가한다.

$$L2 : \| w \|_2^2 = \sum_{j=1}^{m} w_j^2$$

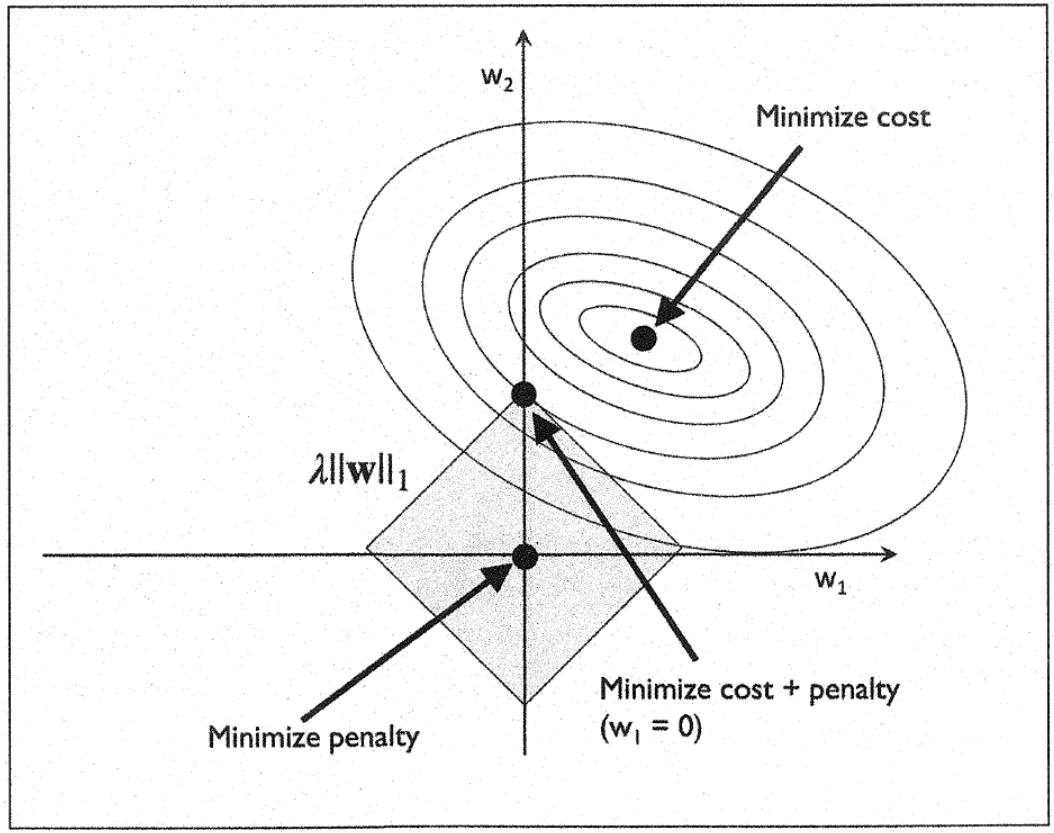
- 모델의 복잡도를 줄이는 다른 방법으로는 L1 regularization이 있다.

$$L1 : \| w \|_1 = \sum_{j=1}^{m} |w_j|$$

- L1 regularization은 대부분의 weight를 zero로 만든다(Sparsity)
  - 이러한 관점에서 L1 regularization을 feature selection의 일종으로 해석할 수 있다.

Selecting meaningful features: Regularization (2/4)

- Sparse solutions with L1 regularization



- L1 regularized system의 등고선이 뾰족한 모양을 가지기 때문에, 최적해가 축 위에 존재할 가능성이 높아진다.

Selecting meaningful features: Regularization (3/4)

- Sparse solutions with L1 regularization
  - Scikit-learn에서는 L1 regularization을 간단하게 사용할 수 있다.

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(penalty='l1', C=1.0)    # C가 작으면 강한 penalty
lr.fit(X_train_std, y_train)
print('Training accuracy:', lr.score(X_train_std, y_train))
print('Test_accuracy:', lr.score(X_test_std, y_test))
```

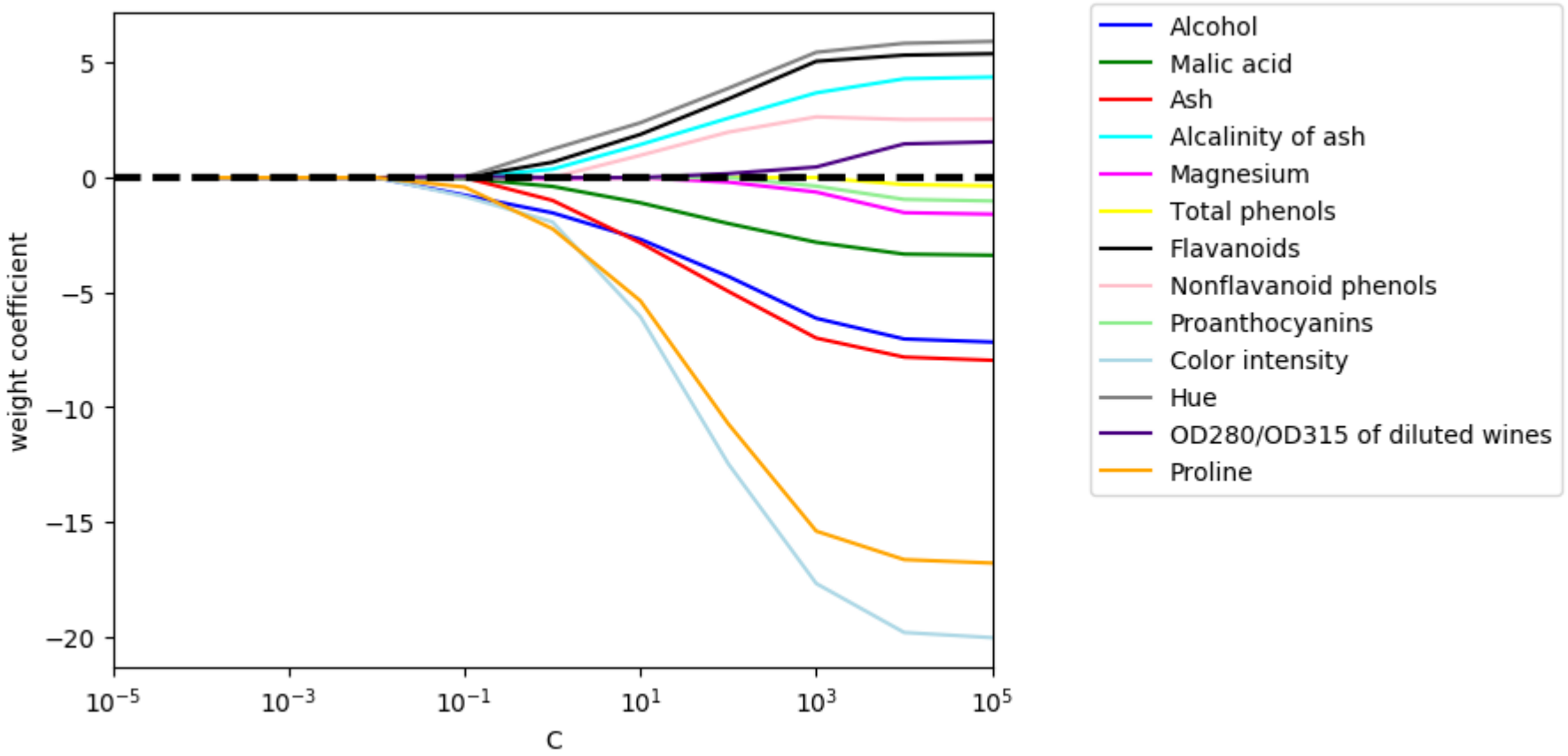
```
Training accuracy: 1.0
Test_accuracy: 1.0
```

```
lr.coef_    # weight 값 반환
```

```
array([[ 1.24542844,  0.18073132,  0.74625919, -1.16386231,  0.        ,
         0.        ,  1.16039206,  0.        ,  0.        ,  0.        ,
         0.        ,  0.55766468,  2.50924129],
       [-1.53720502, -0.38737731, -0.99546653,  0.36501425, -0.05975004,
         0.        ,  0.66806137,  0.        ,  0.        , -1.93404692,
         1.23421856,  0.        , -2.23167788],
       [ 0.13557485,  0.16843305,  0.35726521,  0.        ,  0.        ,
         0.        , -2.43788304,  0.        ,  0.        ,  1.56374042,
        -0.81879498, -0.49290116,  0.        ]])
```

Selecting meaningful features: Regularization (4/4)

- Regularization 강도를 높임으로써 sparsity를 높일 수 있다.
- Regularization 강도를 바꾸어 가면서 각 feature의 weight값 변화를 살펴보자.



Selecting meaningful features: Regularization

- L1 vs L2 reguarlization coefficient들의 weight 비교

- L2 regularization을 적용할 경우
  - C 값이 [10^-3, 10^-2, 10^-1,1,10,100,1000,10000,100000] 로 변할 때 각 변수의 weight 값이 어떻게 변하는지 plot 그리기

Selecting meaningful features: Feature Selection (1/4)

- 특성 자동 선택
  - Overfitting을 방지하기 위해 feature selection을 통해 차원을 축소
- 접근 방법
  - 일변량 통계
    - 일변량 통계에서는 각 입력 특성과 출력 특성 사이에 중요한 통계가 있는지 계산하여 깊게 관련된 특성을 판단함
    - 다른 특성과 깊게 연관된 특성은 선택되지 않음
  - 모델 기반 선택
    - 모델 기반 특성 선택은 지도 학습 모델을 사용하여 특성의 중요도를 평가
    - 특성 선택에 사용하는 모델은 최종적으로 사용할 지도 학습 모델과 같을 필요는 없음
  - 반복적 선택

Selecting meaningful features: Feature Selection (2/4)

- 일변량 통계
  - sklearn.feature_selection.SelectPercentile은 지정한 비율만큼 특성을 선택
    - Classification function에 대해서 default로는 ANOVA F-test로 변수를 선택함
    - http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectPercentile.html
  - sklearn.feature_selection.SelectKBest는 고정된 k개의 특성을 선택
    - http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

Selecting meaningful features: Feature Selection

- 랜덤하게 변수 추가한 경우, 선택되는 변수 수에 따라서 모델의 성능이 어떻게 바뀌는지 확인

Selecting meaningful features: Feature Selection (3/4)

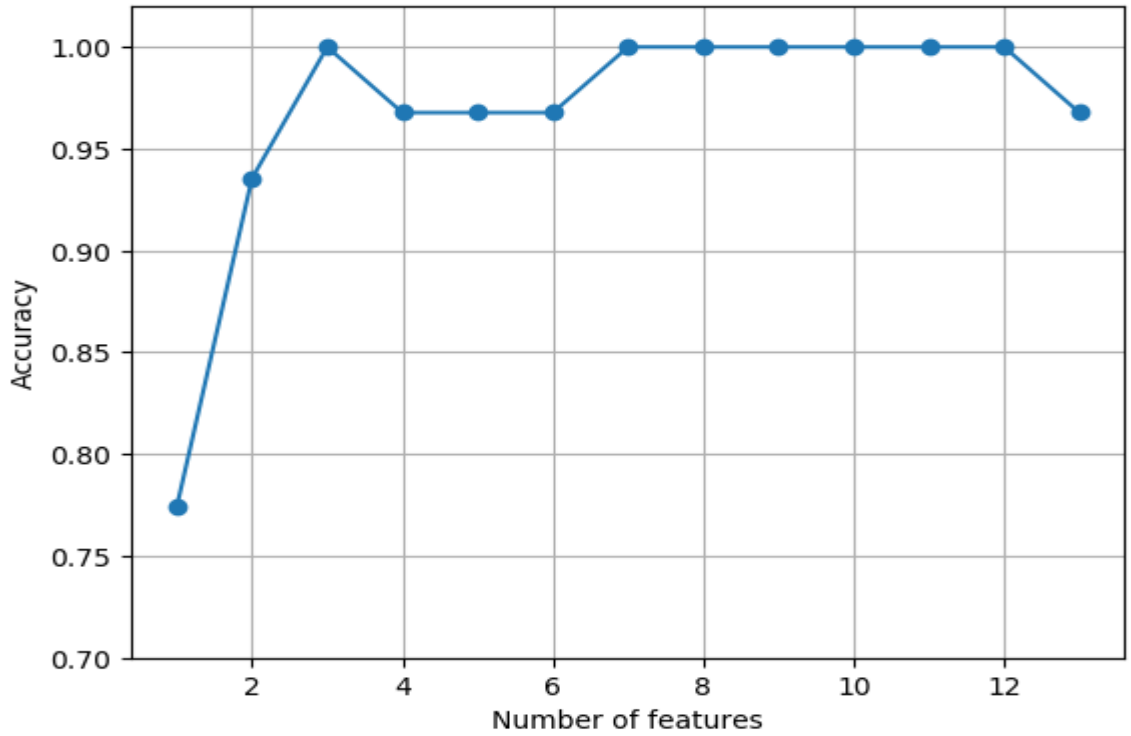
- 반복적 특성 선택
  - Sequential backward selection(SBS) 알고리즘은 greedy search 알고리즘의 일종으로, 분류기의 퍼포먼스 감소를 최소화하면서 계산 효율을 높이는 것을 목적으로 한다.
    - 특정 상황에서, SBS는 overfitting을 방지함으로써 모델의 예측력 또한 높일 수 있다.
  - SBS 알고리즘은 원하는 feature 수에 도달할 때까지 feature 의 수를 하나씩 줄여간다.
    - 각 단계에서, feature 하나를 제거했을 때 criterion J를 가장 크게 만드는 것을 제거한다.

$$x^{-} = argmax J(X_k - x)$$

  - SBS 알고리즘은 scikit-learn에 구현되어 있지 않기 때문에, 직접 구현해 보자.

Selecting meaningful features : Feature Selection (4/4)

- Sequential feature selection algorithms
  - KNN 분류기를 사용하는 SBS를 실행해보자.



  - KNN 분류기의 테스트 셋에서의 정확도가 향상함을 확인할 수 있다.

Reference

- [Raschka. (2017)] Raschka, Sebastian, and Vahid Mirjalili. *Python machine learning*. Packt Publishing Ltd, 2017.
- [Müller. (2016)] Müller, Andreas C., and Sarah Guido. *Introduction to machine learning with Python: a guide for data scientists*. 2016.
- [GÉRON. (2017)] GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* Sebastopol, CA: O, 2017.