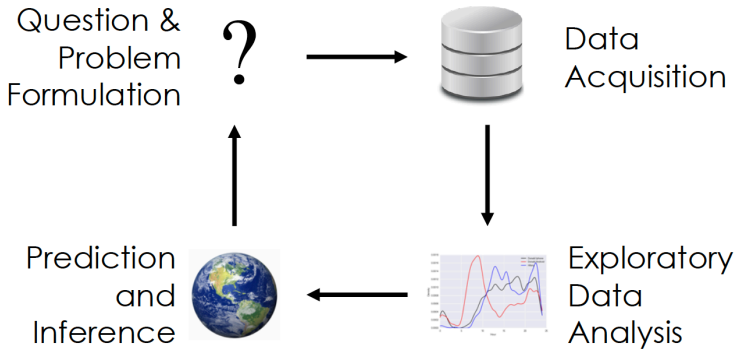
데이터 분석 과정

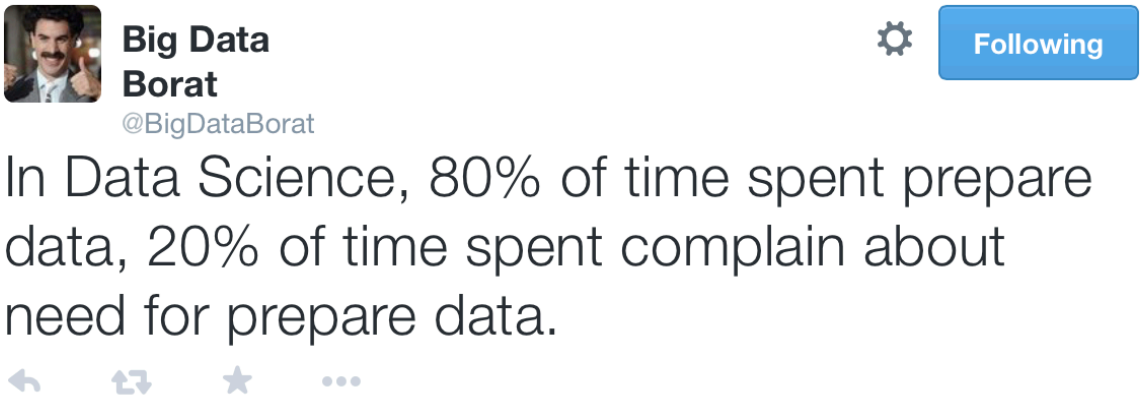


ESC 20기 조민주

EDA(Exploratory Data Analysis)

데이터를 변형하고, 시각화하고, 요약하는 과정

답이 없음

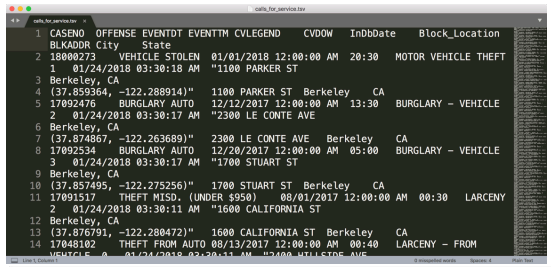


EDA에서 고려하는 주요 **데이터 특성**

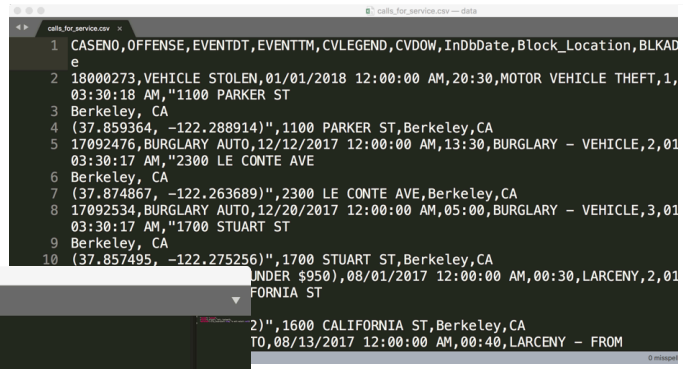
- **Structure** : 데이터 파일이 어떻게 생겼는지
- **Granularity** : 각 데이터가 얼마나 정교한지
- **Scope** : 데이터가 얼마나 (불)완전한지
- **Temporality** : 데이터와 시간
- **Faithfulness** : 데이터가 얼마나 믿을만한지

EDA에서 고려하는 주요 **데이터 특성**

- **Structure** : 데이터 파일이 어떻게 생겼는지

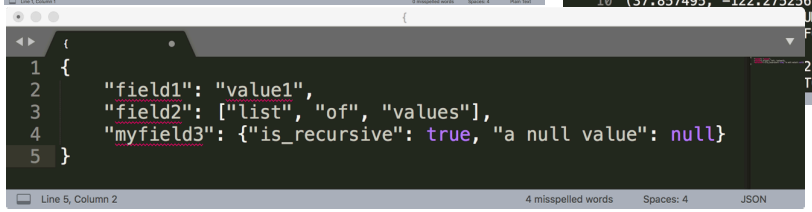


```
CASENO  OFFENSE EVENTDT EVENTTM CVLEGEND    CVDOW   InDbDate    Block_Location
BLKADDR City    State
18000273    VEHICLE STOLEN  01/01/2018 12:00:00 AM  20:30   MOTOR VEHICLE THEFT
1   01/24/2018 03:30:18 AM  "1100 PARKER ST
Berkeley, CA
(37.859364, -122.288914)"   1100 PARKER ST  Berkeley    CA
17092476    BURGLARY AUTO   12/12/2017 12:00:00 AM  13:30   BURGLARY - VEHICLE
2   01/24/2018 03:30:17 AM  "2300 LE CONTE AVE
Berkeley, CA
(37.874867, -122.263689)"   2300 LE CONTE AVE   Berkeley    CA
17092534    BURGLARY AUTO   12/20/2017 12:00:00 AM  05:00   BURGLARY - VEHICLE
3   01/24/2018 03:30:17 AM  "1700 STUART ST
Berkeley, CA
(37.857495, -122.275256)"   1700 STUART ST  Berkeley    CA
17091517    THEFT MISD. (UNDER $950)    08/01/2017 12:00:00 AM  00:30   LARCENY
2   01/24/2018 03:30:11 AM  "1600 CALIFORNIA ST
Berkeley, CA
(37.876791, -122.280472)"   1600 CALIFORNIA ST  Berkeley    CA
17048102    THEFT FROM AUTO 08/13/2017 12:00:00 AM  00:40   LARCENY - FROM
```



```
CASENO,OFFENSE,EVENTDT,EVENTTM,CVLEGEND,CVDOW,InDbDate,Block_Location,BLKAD
e
18000273,VEHICLE STOLEN,01/01/2018 12:00:00 AM,20:30,MOTOR VEHICLE THEFT,1,
03:30:18 AM,"1100 PARKER ST
Berkeley, CA
(37.859364, -122.288914)",1100 PARKER ST,Berkeley,CA
17092476,BURGLARY AUTO,12/12/2017 12:00:00 AM,13:30,BURGLARY - VEHICLE,2,01
03:30:17 AM,"2300 LE CONTE AVE
Berkeley, CA
(37.874867, -122.263689)",2300 LE CONTE AVE,Berkeley,CA
17092534,BURGLARY AUTO,12/20/2017 12:00:00 AM,05:00,BURGLARY - VEHICLE,3,01
03:30:17 AM,"1700 STUART ST
Berkeley, CA
(37.857495, -122.275256)",1700 STUART ST,Berkeley,CA
UNDER $950),08/01/2017 12:00:00 AM,00:30,LARCENY,2,01
FORNIA ST

2)",1600 CALIFORNIA ST,Berkeley,CA
TO,08/13/2017 12:00:00 AM,00:40,LARCENY - FROM
```



```
{
    "field1": "value1",
    "field2": ["list", "of", "values"],
    "myfield3": {"is_recursive": true, "a null value": null}
}
```

Line 5, Column 2 — 4 misspelled words — Spaces: 4 — JSON

```
169.237.46.168 - - [26/Jan/2014:10:47:58 -0800] "GET
/stat141/Winter04 HTTP/1.1" 301 328
"http://anson.ucdavis.edu/courses/"  "Mozilla/4.0 (compatible; MSIE
6.0; Windows NT 5.0; .NET CLR 1.1.4322)"
```

```
<catalog>
  <plant type='a'>
    <common>Bloodroot</common>
    <botanical>Sanguinaria canadensis</botanical>
    <zone>4</zone>
    <light>Mostly Shady</light>
    <price>2.44</price>
    <availability>03/15/2006</availability>
    <description>
        <color>white</color>
        <petals>true</petals>
    </description>
    <indoor>true</indoor>
  </plant>
…
</catalog>
```

EDA에서 고려하는 주요 **데이터 특성**

- **Structure** : 데이터 파일이 어떻게 생겼는지
- **Granularity** : 각 데이터가 얼마나 정교한지
- **Scope** : 데이터가 얼마나 (불)완전한지
- **Temporality** : 데이터와 시간
- **Faithfulness** : 데이터가 얼마나 믿을만한지

EDA에서 고려하는 주요 **데이터 특성**

- **Structure** : 데이터 파일이 어떻게 생겼는지
- **Granularity** : 각 데이터가 얼마나 정교한지
- **Scope** : 데이터가 얼마나 (불)완전한지
- **Temporality** : 데이터와 시간
- **Faithfulness** : 데이터가 얼마나 믿을만한지

EDA에서 고려하는 주요 **데이터 특성**

- **Structure** : 데이터 파일이 어떻게 생겼는지
- **Granularity** : 각 데이터가 얼마나 정교한지
- **Scope** : 데이터가 얼마나 (불)완전한지
- **Temporality** : 데이터와 시간 — datetime
- **Faithfulness** : 데이터가 얼마나 믿을만한지

1998-11-08
1900-01-01 ??
1970-01-01 ??
1996-08-11

EDA에서 고려하는 주요 **데이터 특성**

- Missing Values or default values
- 시간대가 맞지 않는 점
- 중복된 기록
- 철자 오류
- 단위가 나와있지 않거나 다른 값들
- 거짓 정보들

" ", 0, -1, 999, NaN, Null, 1970, 1900

1) 해당 레코드 삭제
2) Imputation
하지만! bias 체크!

- **Faithfulness** : 데이터가 얼마나 믿을만한지

EDA & Data Cleaning

EDA - 코드

```
for i in range(3):
    print(i, "\t", stops_json['data'][i])
```

```
0       ['row-xkgf-zr8a.6fpe', '00000000-0000-0000-4A5A-0E740119A894', 0, 1556687429, None, 1556687433, None, '{ }',
'2019-00004289', '2019-01-25T14:51:35', '2100 BLOCK GRANT ST', 'T', 'AM2TWN;', None, None, None, None, None, None, No
ne, None, None, None, None, None, None]
1       ['row-6s6i~mhea-aiq2', '00000000-0000-0000-2A6D-4576080468CA', 0, 1555712765, None, 1555712773, None, '{ }',
None, '2017-08-07T07:18:25', ' SAC/ASHB', 'T', 'M; ', None, None, None, None, None, None, None, None, None, None, Non
e, None, None]
```

```
# Load the data from JSON and assign column titles
stops = pd.DataFrame(
    stops_json['data'],
    columns=[c['name'] for c in stops_json['meta']['view']['columns']])

stops.head()
```

	sid	id	position	created_at	created_meta	updated_at	updated_meta	meta	Incident Number	Call Date/Time	...	Location 1 (city)	Location 1 (state)	Loca 1
0	row-xkgf-zr8a.6fpe	00000000-0000-0000-4A5A-0E740119A894	0	1556687429	None	1556687433	None	{}	2019-00004289	2019-01-25T14:51:35	...	None	None	N
1	row-6s6i~mhea-aiq2	00000000-0000-0000-2A6D-4576080468CA	0	1555712765	None	1555712773	None	{}	None	2017-08-07T07:18:25	...	None	None	N
2	row-zquh.a2sj_q4q3	00000000-0000-0000-97BB-D07AA3099E29	0	1556687429	None	1556687433	None	{}	2019-00016836	2019-03-29T21:37:10	...	None	None	N

```
for c in stops_json['meta']['view']['columns']:
    desc = ""
    if "description" in c:
        desc = c["description"]
    print(c["name"], ":", "\n\t", "\n\t".join(desc.split("\n")))
```

```
sid :

id :

position :

created_at :
```

import pandas as pd

pd.read_csv("판다.csv")
pd.read_json("판다.json")
...

내장 함수가 없는 경우도 있습니다

EDA - 코드

fillna() : na값을 채우는 함수

Ex) data["hi"].fillna((data["hi"].mean()), inplace = True)

dropna() : na값이 있는 행/열을 지우는 함수

Ex) data.dropna(inplace = True)

```
data.isna().sum()
```

```
Attr1     0
Attr2     0
Attr3     0
Attr4     0
Attr5     0
         ..
Attr61    0
Attr62    0
Attr63    0
Attr64    0
class     0
Length: 65, dtype: int64
```

```
np.sum((data.isna().sum()!=0))
```

```
0
```

Nan 값은 하나도 없다

?가 보이니 ? 개수를 세어봤더니 ? 값은 엄청 많다..

```
np.sum(data == "?").sort_values()[-15:]
```

```
C:\Users\MINJU\Anaconda3\lib\site-packages\pandas\core\ops\array_ops.py:253: FutureWarn
urning scalar instead, but in the future will perform elementwise comparison
  res_values = method(rvalues)
Attr33      28
Attr47      57
Attr52      60
Attr32      72
Attr21     112
Attr41     142
Attr24     149
Attr53     162
Attr54     162
Attr28     162
Attr64     162
Attr45     418
Attr60     420
Attr27     462
Attr37    3100
dtype: int64
```

Attr37은 6855개 중에 3100개가 ?값인데 과연 이걸 쓸 수 있을까..? (X37 (current assets - inventories) / long-term liabilities)

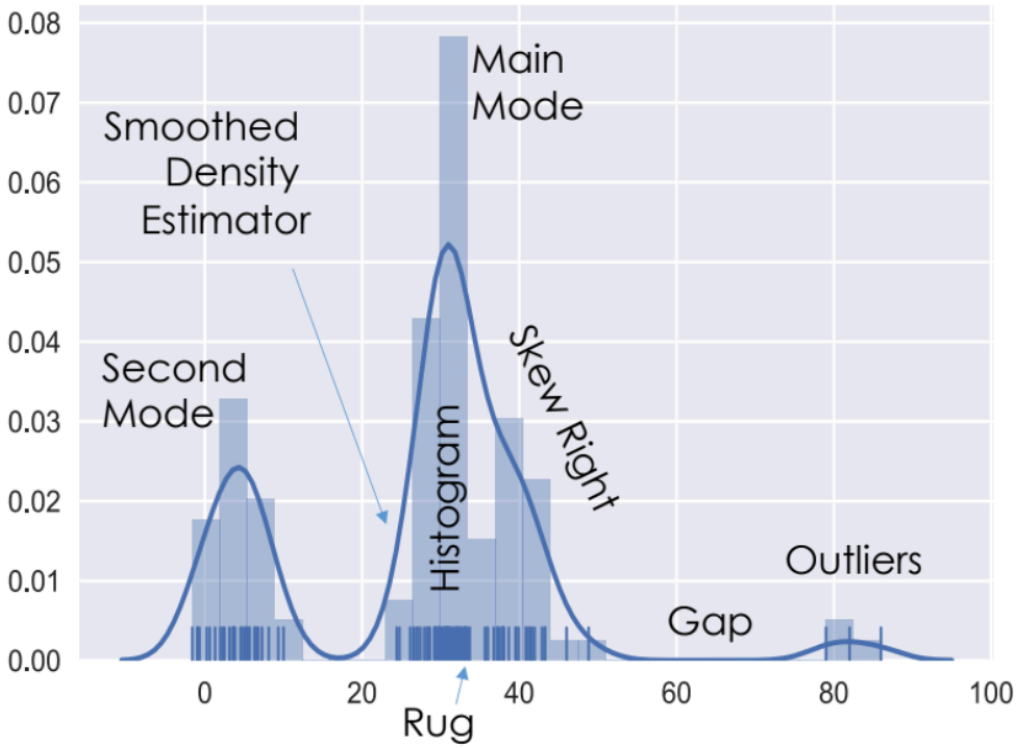
```
np.sum(data == "?", axis = 1).value_counts()
```

```
0     3334
1     2577
2      413
3      223
4       85
5       78
6       41
7       39
8       15
9        9
14       7
10       6
18       4
17       4
11       4
15       3
13       3
16       2
12       2
21       2
20       1
25       1
22       1
19       1
dtype: int64
```

- 행별로 ? 가 있는 것끼리 상관관계를 봤더니 아예 ?가 없는 행은 3334개밖에없다.
- 그러므로 그냥 ? 있는 행을 지워서는 안 된다..
- ?가 많은 행도 있지만 딱히 엄청 상관관계가 있어 보이진 않는다.
- ?가 많은 행의 target 변수들을 비교한 후 상관관계를 보고 ?가 너무 많은 행은 그냥 없
- 계산 식들이니까 혹시 다른 변수들을 이용해 ? 값을 구할 방법은 없을까?

EDA - 시각화

- **Modes**
  - Number
  - Location
  - Size
- **Symmetry**
  - Symmetric
  - Skewed left or right
- **Tails**
- **Gaps**
- **Outliers**



EDA - 시각화

- **Modes**
  - Number
  - Location
  - Size
- **Symmetry**
  - Symmetric
  - Skewed left or right
- **Tails**
- **Gaps**
- **Outliers**

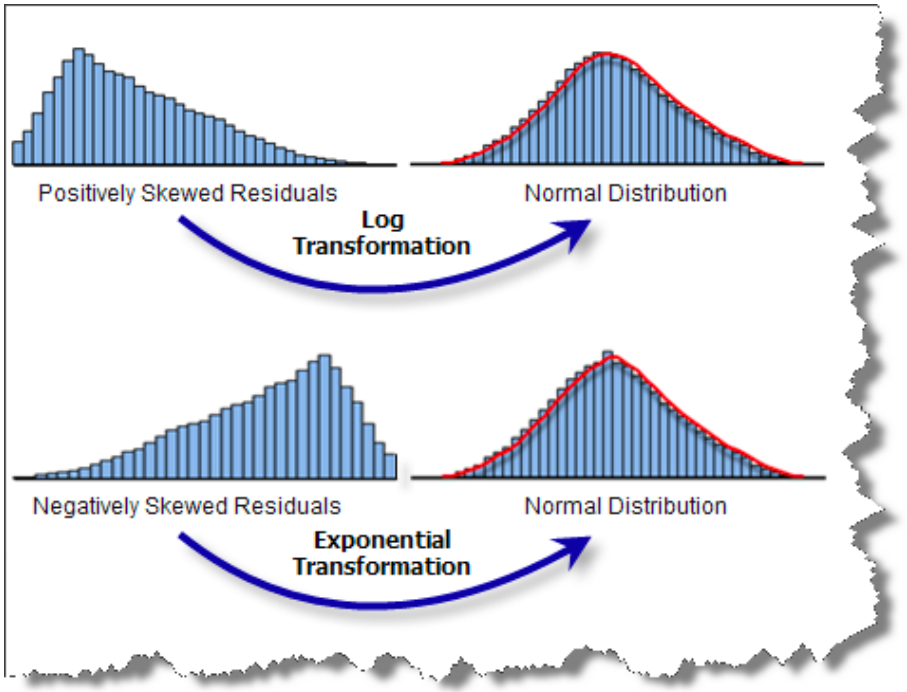
: Fisher-Pearson coefficient of skewness

$$g_1 = \frac{m_3}{m_2^{3/2}}$$

$$m_i = \frac{1}{N}\sum_{n=1}^{N}(x[n] - \bar{x})^i$$

```
>>> from scipy.stats import skew
>>> skew([1, 2, 3, 4, 5])
0.0
>>> skew([2, 8, 0, 4, 1, 9, 9, 0])
0.2650554122698573
```

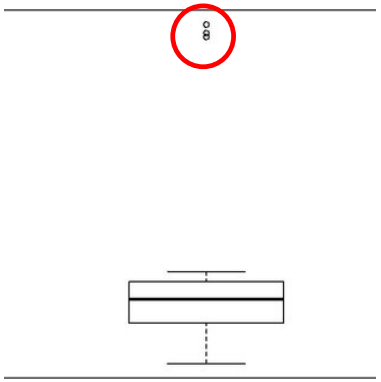
Transformation



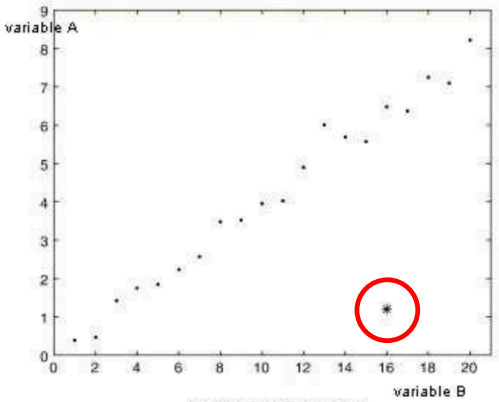
EDA - 시각화

- **Modes**
  - Number
  - Location
  - Size
- **Symmetry**
  - Symmetric
  - Skewed left or right
- **Tails**
- **Gaps**
- **Outliers**

Univariate



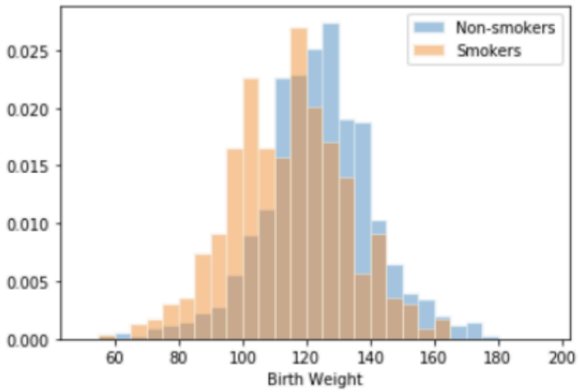
Multivariate

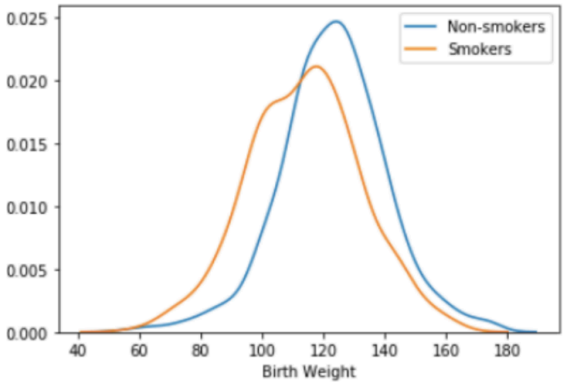


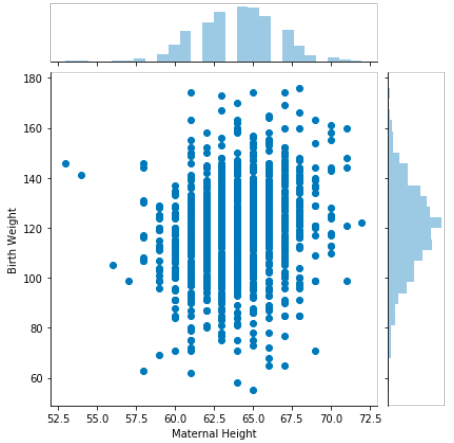
(Q1-1.5*IQR, Q3+1.5*IQR)
IQR = Q3 – Q1

EDA - 시각화

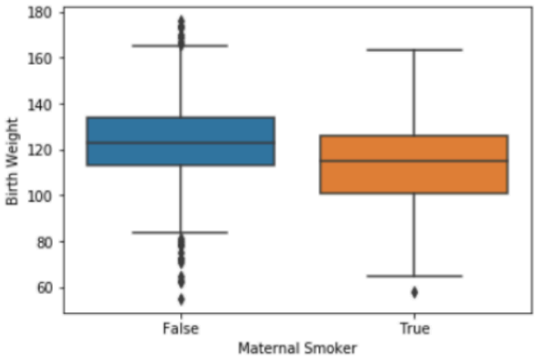
import matplotlib.pyplot as plt
import seaborn as sns

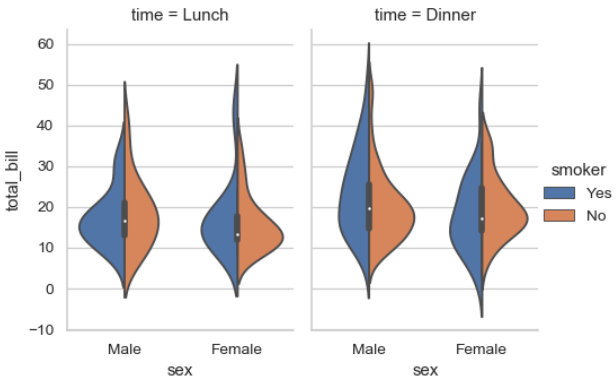


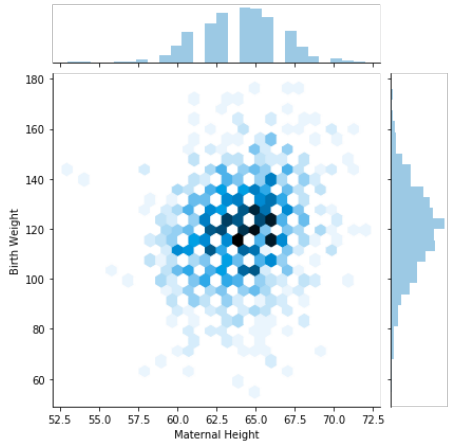




Side-by-side Boxplots







Feature Engineering

feature를 다른 새로운 feature로

uid	age	state	hasBought	review
0	32	NY	True	"Meh."
42	50	WA	True	"Worked out of the box ..."
57	16	CA	NULL	"Hella tots lit..."

Entirely **Quantitative** and **Transformed** Values

AK	...	NY	...	WY	age	age^2	hasBought missing
0	...	1	...	0	32	32^2	0
0	...	0	...	0	50	50^2	0
0	...	0	...	0	16	16^2	1

- **필요없는 변수** : 없애자
- **양적 변수** : 스케일링 — Standard / Minmax
- **질적 변수** : One-hot-Encode

	name	kind	age
0	Goldy	Fish	0.5
1	Scooby	Dog	7.0
2	Brian	Dog	3.0
3	Francine	Cat	10.0
4	Goldy	Dog	1.0

Pandas has a built in function to construct one-hot encodings called `get_dummies`

```
pd.get_dummies(df['kind'])
```

	Cat	Dog	Fish
0	0	0	1
1	0	1	0
2	0	1	0
3	1	0	0
4	0	1	0

```
pd.get_dummies(df)
```

	age	name_Brian	name_Francine	name_Goldy	name_Scooby	kind_Cat	kind_Dog	kind_Fish
0	0.5	0	0	1	0	0	0	1
1	7.0	0	0	0	1	0	1	0
2	3.0	1	0	0	0	0	1	0
3	10.0	0	1	0	0	1	0	0
4	1.0	0	0	1	0	0	1	0

```
: from sklearn.preprocessing import OneHotEncoder

oh_enc = OneHotEncoder()
```

Feature Engineering

- **Dimension Reduction**
  (차원 축소)

  - **Feature Selection**
    (변수 선택)

  - **Feature Extraction**
    (변수 추출)

Feature Engineering

차원 축소 - 변수 선택

1) 도메인 지식
2) Feature Importance (변수 중요도)
3) 상관관계 - 다중공선성

- Dimension Reduction
(차원 축소)

- Feature Selection
(변수 선택)

- Feature Extraction
(변수 추출)

Feature Engineering

차원 축소 - 변수 선택

1) 도메인 지식
2) Feature Importance (변수 중요도)
3) 상관관계 - 다중공선성

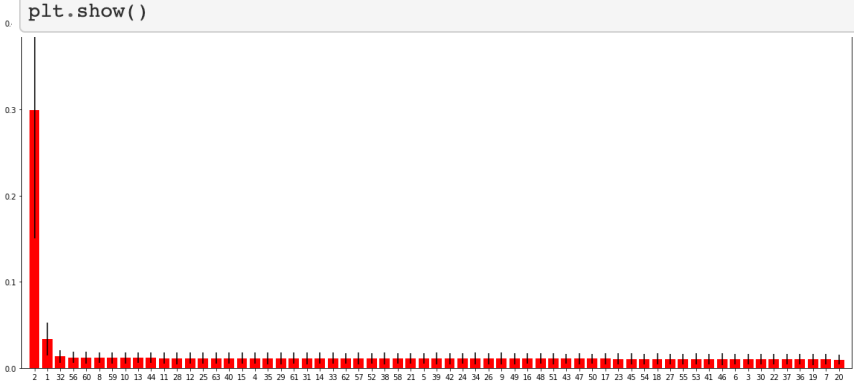
```
from sklearn.datasets import make_classification
from sklearn.ensemble import ExtraTreesClassifier
```

```
X, y = make_classification(n_samples=1000,
                           n_features=63,
                           n_informative=2,
                           n_redundant=0,
                           n_repeated=0,
                           n_classes=2,
                           random_state=24,
                           shuffle=False)
```

```
forest = ExtraTreesClassifier(n_estimators=250,
                              random_state=0)
```

```
forest.fit(X, y)
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]
```

```
# Plot the impurity-based feature importances of the forest
plt.figure(figsize=(20,10))
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices],
        color="r", yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), indices+1)
plt.xlim([-1, X.shape[1]])
plt.show()
```



Feature Engineering

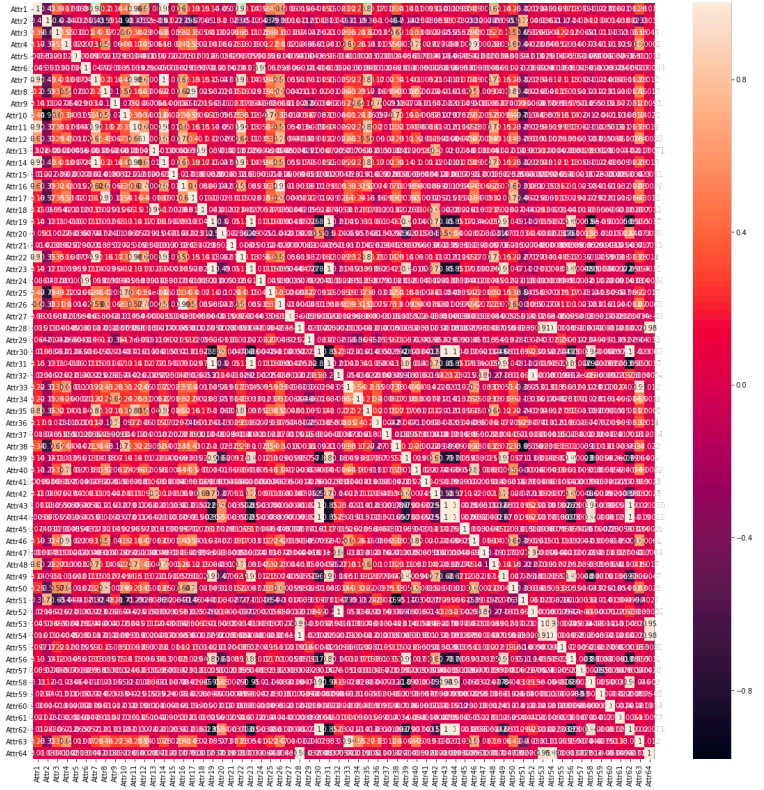
차원 축소 - 변수 선택

1) 도메인 지식
2) Feature Importance (변수 중요도)
3) 상관관계 - 다중공선성

- **Dimension Reduction**
  (차원 축소)

  - **Feature Selection**
    (변수 선택)

  - **Feature Extraction**
    (변수 추출)



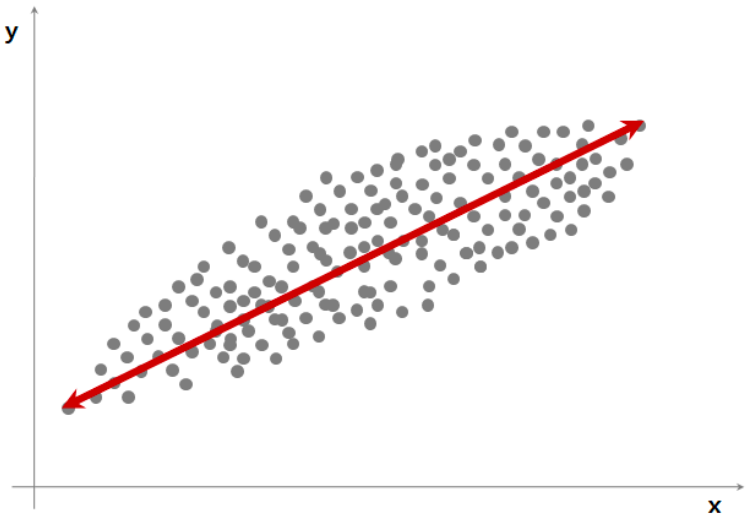
차원 축소 - 변수 추출

1) PCA(Principal Component Analysis)
2) FA
3) 파생변수

- **Dimension Reduction**
  (차원 축소)

  - **Feature Selection**
    (변수 선택)

  - **Feature Extraction**
    (변수 추출)



차원 축소 - 변수 추출

1) PCA
2) FA(Factor Analysis)
3) 파생변수

- **Dimension Reduction**
  (차원 축소)

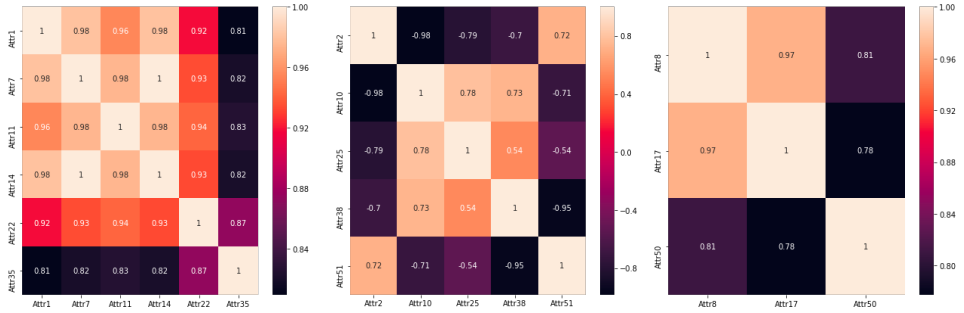
  - **Feature Selection**
    (변수 선택)

  - **Feature Extraction**
    (변수 추출)

```
for i in range(1, no_na_wo_class.shape[1]+1):
    print("High correlation with Attr", i, "is : ")
    print(list(no_na_wo_class.columns[(no_na_wo_class.corr()["Attr{}".format(i)]>=0.7) |
                                      (no_na_wo_class.corr()["Attr{}".format(i)]<=-0.7)]))
```

```
High correlation with Attr 1 is :
['Attr1', 'Attr7', 'Attr11', 'Attr14', 'Attr22', 'Attr35']
High correlation with Attr 2 is :
['Attr2', 'Attr10', 'Attr25', 'Attr38', 'Attr51']
High correlation with Attr 3 is :
['Attr3']
```

```
for i in [1,2,4,6,8,9,10,12,13,19,28,30,32,33]:
    idx_list = list(no_na_wo_class.columns[(no_na_wo_class.corr()["Attr{}".format(i)]>=0.7) |
                                           (no_na_wo_class.corr()["Attr{}".format(i)]<=-0.7)])
    f, axes = plt.subplots(nrows = 1, ncols = 1, figsize=(7,7))
    sns.heatmap(no_na_wo_class[idx_list].corr(), annot=True)
```



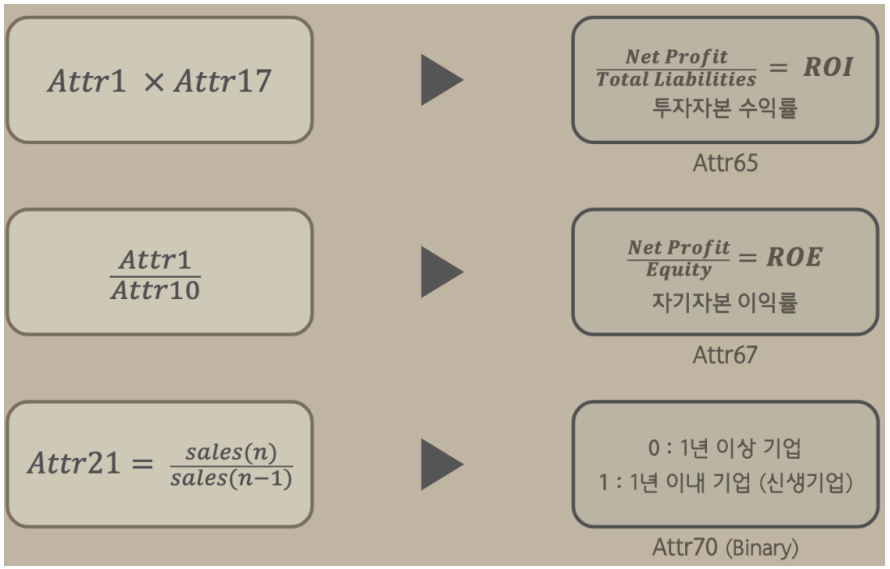
차원 축소 - 변수 추출

1) PCA
2) FA
3) 파생변수

- **Dimension Reduction**
  (차원 축소)

  - **Feature Selection**
    (변수 선택)

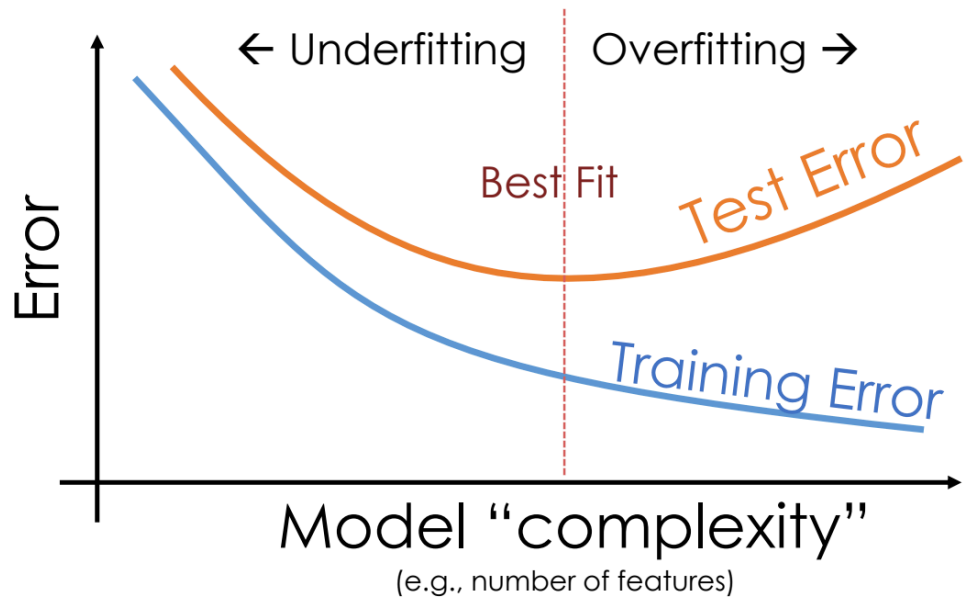
  - **Feature Extraction**
    (변수 추출)

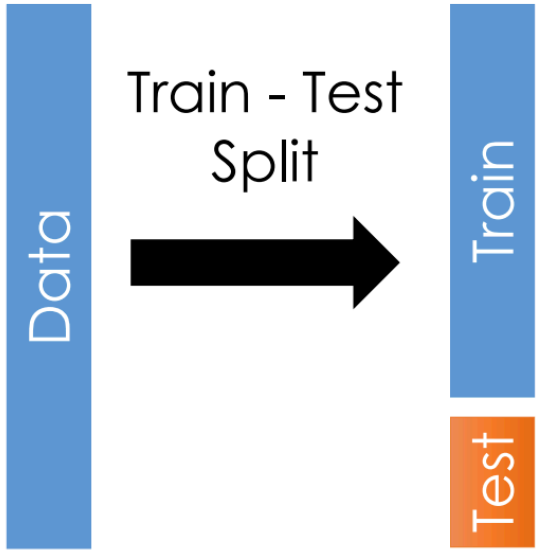


Train-Test Split

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

```
X_train, X_test, y_train, y_test = train_test_split(datamice.iloc[:, 0:(datamice.shape[1]-1)], datamice['class'], test_size=0.3, random_state= 730)
```



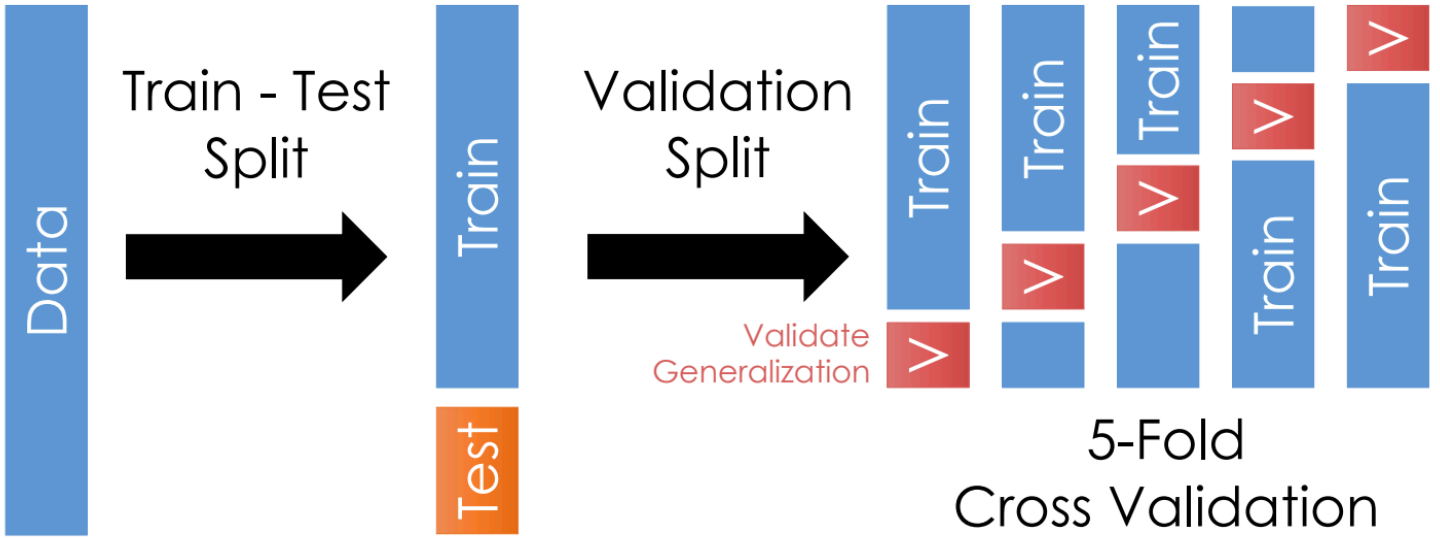


Validation Split(일반화) - CV

```
re_stf = RepeatedStratifiedKFold(n_splits = 10, n_repeats = 3, random_state = 2)
scores = cross_val_score(model, data_X, data_y, scoring = "f1_micro", cv = re_stf)
```

```
print("Mean CV F1 : %.3f"% np.mean(scores))
```

```
Mean CV F1 : 0.936
```

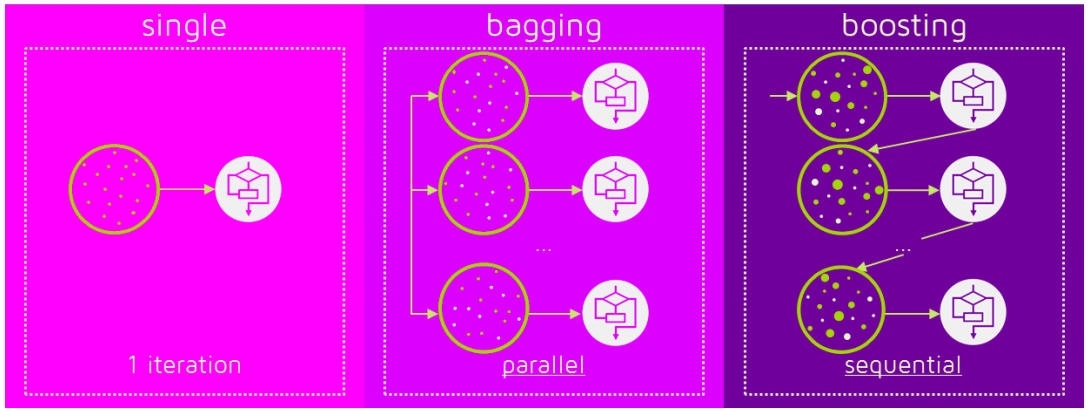


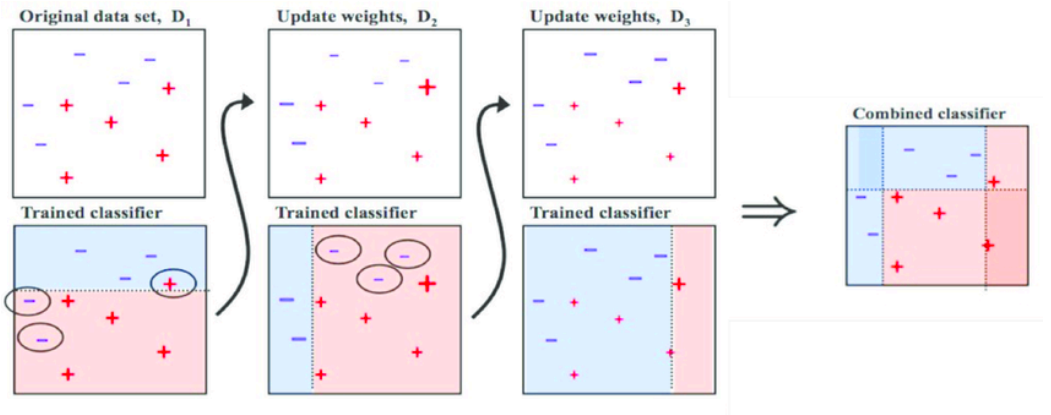
Modeling

- **회귀**
  - Ridge, Lasso
  - Logistic
- **앙상블**
  - Bagging
    - Random Forest
  - Boosting
    - XGBoost

Modeling

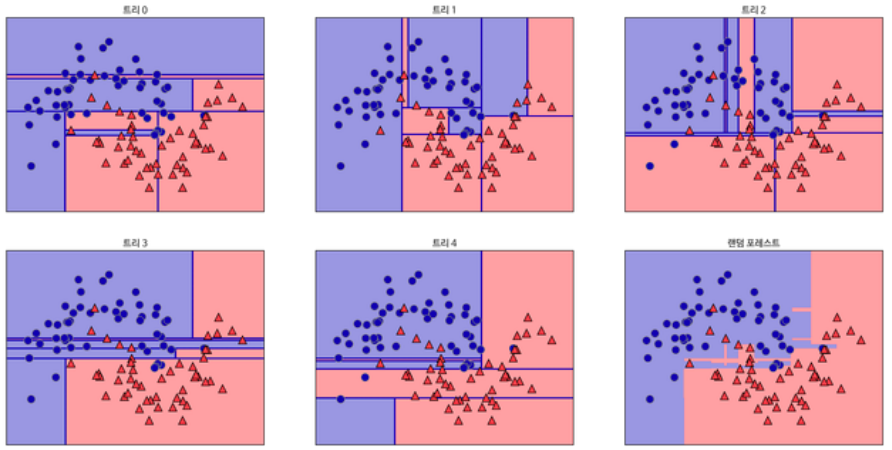
- 회귀
  - Ridge, Lasso
  - Logistic
- 앙상블
  - Bagging
    - Random Forest
  - Boosting
    - XGBoost





Modeling

- 회귀
  - Ridge, Lasso
  - Logistic
- 앙상블
  - Bagging
    - Random Forest
  - Boosting
    - XGBoost



Random Forest

```
from sklearn.ensemble import RandomForestClassifier
```

```
forest = RandomForestClassifier(n_estimators=500, random_state=24)
```

```
scores = cross_val_score(forest, X_train, Y_train, scoring = "f1_micro", cv = re_stf)
print("Mean CV F1 : %.3f"% np.mean(scores))
scores = cross_val_score(forest, X_train, Y_train, scoring = "roc_auc", cv = re_stf)
print("Mean CV ROC_AUC : %.3f"% np.mean(scores))
```

```
Mean CV F1 : 0.956
Mean CV ROC_AUC : 0.882
```

```
forest.fit(X_train, Y_train)
pred_y = forest.predict(X_test)
print("F1 : %.3f" % f1_score(Y_test, pred_y, average = 'micro'))
print("ROC AUC : %.3f" % roc_auc_score(Y_test, pred_y))
```

```
F1 : 0.955
ROC AUC : 0.612
```

Modeling

- 회귀
  - Ridge, Lasso
  - Logistic
- 앙상블
  - Bagging
    - Random Forest
  - Boosting
    - XGBoost

XGBOOST

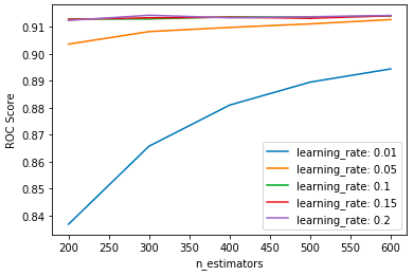
```
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV
```

```
n_estimators = [200, 300, 400, 500, 600]
learning_rate = [0.01, 0.05, 0.1, 0.15, 0.2]
param_grid = dict(learning_rate=learning_rate, n_estimators=n_estimators)
```

```
xgb_greedy_model = XGBClassifier()
grid_search = GridSearchCV(xgb_greedy_model, param_grid, scoring="roc_auc", cv=re_stf)
grid_result = grid_search.fit(X_train, Y_train)
```

```
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
# plot results
scores = np.array(means).reshape(len(learning_rate), len(n_estimators))
for i, value in enumerate(learning_rate):
    plt.plot(n_estimators, scores[i], label='learning_rate: ' + str(value))
plt.legend()
plt.xlabel('n_estimators')
plt.ylabel('ROC Score')
```

```
Best: 0.914288 using {'learning_rate': 0.2, 'n_estimators': 300}
0.836783 (0.045509) with: {'learning_rate': 0.01, 'n_estimators': 200}
0.865733 (0.039090) with: {'learning_rate': 0.01, 'n_estimators': 300}
0.880966 (0.033196) with: {'learning_rate': 0.01, 'n_estimators': 400}
0.889485 (0.030789) with: {'learning_rate': 0.01, 'n_estimators': 500}
0.894394 (0.030367) with: {'learning_rate': 0.01, 'n_estimators': 600}
0.903598 (0.029459) with: {'learning_rate': 0.05, 'n_estimators': 200}
0.908234 (0.030914) with: {'learning_rate': 0.05, 'n_estimators': 300}
0.909779 (0.030060) with: {'learning_rate': 0.05, 'n_estimators': 400}
0.911092 (0.029803) with: {'learning_rate': 0.05, 'n_estimators': 500}
0.912689 (0.030012) with: {'learning_rate': 0.05, 'n_estimators': 600}
0.912882 (0.029384) with: {'learning_rate': 0.1, 'n_estimators': 200}
0.912917 (0.029403) with: {'learning_rate': 0.1, 'n_estimators': 300}
0.913672 (0.030758) with: {'learning_rate': 0.1, 'n_estimators': 400}
```



Regex 문법

변수 이름은 그 변수를 잘 대표하도록

코드는 최대한 잘 보존해놓기

SQL

끝