



빅데이터와 인공지능의 이해

04.AI와 파이썬 라이브러리 활용

조창제

강의자료

2025.03.

목차

CONTENTS



I

빠르고 간단히 활용하는

라이브러리/패키지

II

원하는 형태로 데이터 변환

데이터 핸들링

III

디테일하고 아름답게

시각화



목차

CONTENTS

IV

보다 정밀하고 논리적으로

데이터분석





01. 개념

1. 라이브러리와 패키지

1) 라이브러리

- ① 특정 작업을 수행하기 위해 제공되는 재사용 가능한 코드 집합(함수, 클래스, 모듈 등)

2) 패키지

- ① 관련된 여러 모듈과 하위 라이브러리들의 집합

3) 파이썬

- 파이썬 공식 저장소 PyPI(Python Package Index) 저장소 기준 **530,000개 이상**의 라이브러리가 등록되어 있음
- 라이브러리를 완벽히 공부하기보다는 **필요한 라이브러리 위주로 모듈을 응용하는 방법을 익히는 것이 효과적**



Library
(공구 하나)



Package
(공구 집합)



1.정의

1) 파이썬의 고성능 수치 계산을 위한 라이브러리

① 다양한 자료타입과 자료 생성 함수를 지원

자료타입	설명
np.int64	정수
np.float64	실수
np.complex	복소수
np.bool	논리(True, False)
np.object	객체구조
np.string_	고정길이 문자열
np.unicode_	유니코드 문자 저장 가능

자료생성	설명
np.array	배열을 생성
np.zeros	0값을 가진 numpy 객체 생성
np.ones	1값을 가진 numpy 객체 생성
np.full	특정 값을 가진 numpy 객체 생성
np.eye	단위행렬 생성
np.linspace	start, end까지 n개의 선형적 값 생성
np.random.random	0~1사이 실수 생성
np.empty	메모리 상의 이전값들로 채워진 배열 생성



1.정의

1) 파이썬의 고성능 수치 계산을 위한 라이브러리

② 다양한 수학적 기능을 제공

- 집계함수 지원

함수	설명
np.add(a,b)	더하기
np.subtract(a, b)	빼기
np.multiply(a, b)	곱하기
np.divide(a, b)	나누기
np.exp(a)	지수연산
np.sqrt(a)	제곱근
np.log(a)	로그
np.sin(a), np.cos(a), np.tan(a)	삼각함수
a.dot(b), @	행렬곱

함수	설명
a.sum, a.cumsum	합계, 누적합
a.min, a.max	최소값, 최대값
a.mean, a.median, a.std	평균, 중앙값, 표준편차
a.corrcoef	상관계수
np.diag	주대각성분
np.linalg.det	행렬식



1.정의

1) 파이썬의 고성능 수치 계산을 위한 라이브러리

③ 다양한 배열의 처리에 특화

- 데이터 형태변환 및 자료 추출/결합 지원

함수	설명
np.sort	정렬
a[0:2], a[1,...], a[:1], a[...,-4:]	슬라이싱
a[2], a[1,2]	부분추출
a[a>2], a[[1,3,0],[1,2]]	인덱싱
a.flatten	객체구조
a.T, np.transpose(a)	전치
a.reshape	구조변형
np.concatenate	결합
np.hstack, np.vstack	수평/수직 결합
np.hsplit, np.vsplit	수평/수직분해

1.정의

1) 표 형식의 데이터를 다루는데 유용한 라이브러리

① 데이터 변환, 처리, 분석에 필요한 다양한 기능을 제공

- 다양한 데이터(CSV, Excel, SQL, JSON 등) 저장 및 불러오기 가능
- 데이터 탐색 및 필터링 가능, 데이터 변환 및 처리, 정렬 가능
- 결측값 처리 가능, 그룹화 및 집계 가능

Data Wrangling with pandas

Cheat Sheet

<http://pandas.pydata.org>

Syntax - Creating DataFrames

```
df = pd.DataFrame({
    "a": [4, 5, 6],
    "b": [7, 8, 9],
    "c": [10, 11, 12],
    index = [1, 2, 3]
})
```

Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```

Specify values for each row.

```
df = pd.DataFrame(
    {"a": [4, 5, 6],
     "b": [7, 8, 9],
     "c": [10, 11, 12]},
    index = pd.MultiIndex.from_tuples(
        [(1, 2), (1, 3), (2, 1), (2, 2)],
        names=['a', 'b'])
```

Create DataFrame with a MultiIndex

Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
      .rename(columns={
          "variable": "var",
          "value": "val"}))
      .query("val > 200")
```

Tidy Data - A foundation for wrangling in pandas

In a tidy data set:

- Each variable is saved in its own column
- Each observation is saved in its own row

Reshaping Data - Change the layout of a data set

pd.melt(df)
Gather columns into rows.

df.pivot(columns="var", values="val")
Spread rows into columns.

pd.concat([df1, df2])
Append rows of DataFrames

pd.concat([df1, df2], axis=1)
Append columns of DataFrames

Subset Observations (Rows)

```
df[df.length > 7]
```

Extract rows that meet logical criteria.

```
df.drop_duplicates()
```

Remove duplicate rows (only considers columns).

```
df.head(n)
```

Select first n rows.

```
df.tail(n)
```

Select last n rows.

Subset Variables (Columns)

```
df[["width", "length", "species"]]
```

Select multiple columns with specific names.

```
df[["width"] or df.width]
```

Select single column with specific name.

```
df.filter(regex="regex")
```

Select columns whose name matches regular expression regex.

Reshape (Regular Expressions) Examples

```
df[df["width"].str.contains("jail")]
```

Matches strings containing a "jail"

```
df[df["width"].str.contains("jail", na=False)]
```

Matches strings containing a "jail" (na=False)

```
df[df["width"].str.contains("jail", na=False, regex=True)]
```

Matches strings containing a "jail" (na=False, regex=True)

```
df[df["width"].str.contains("jail", na=False, regex=True, case=False)]
```

Matches strings containing a "jail" (na=False, regex=True, case=False)

```
df[df["width"].str.contains("jail", na=False, regex=True, case=False, flags=re.IGNORECASE)]
```

Matches strings containing a "jail" (na=False, regex=True, case=False, flags=re.IGNORECASE)

Summarize Data

```
df["var"].value_counts()
```

Count number of rows with each unique value of variable

```
len(df)
```

of rows in DataFrame.

```
df["var"].nunique()
```

of distinct values in a column.

```
df.describe()
```

Basic descriptive statistics for each column (or Groupby)

pandas provides a large set of summary functions that operate on different kinds of pandas objects (DataFrame columns, Series, Groupby, Expanding and Rolling (see below)) and produce single values for each of the groups. When applied to a DataFrame, the result is returned as a pandas Series for each column.

sum()
Sum values of each object.

count()
Count non-NA/null values of each object.

min()
Minimum value in each object.

max()
Maximum value in each object.

mean()
Mean value of each object.

median()
Median value of each object.

quantile()
Quantiles of each object.

std()
Standard deviation of each object.

apply()
Apply function to each object.

Group Data

```
df.groupby("by="col")
```

Return a Groupby object, grouped by values in column named "col".

```
df.groupby(level="ind")
```

Return a Groupby object, grouped by values in index level named "ind".

All of the summary functions listed above can be applied to a group.

agg()
Aggregate group using function.

Windows

```
df.rolling(window=5)
```

Return an Expanding object allowing summary functions to be applied cumulatively.

```
df.rolling(window=5, min_periods=1)
```

Return a Rolling object allowing summary functions to be applied to windows of length n.

Handling Missing Data

```
df.dropna()
```

Drop rows with any column having NA/null data.

```
df.fillna(value)
```

Replace all NA/null data with value.

Make New Columns

```
df.assign(new=lambda df: df.length*df.height)
```

Compute and append one or more new columns.

```
df["Volume"] = df.length*df.height*df.depth
```

Add single column.

```
pd.qcut(df.col, n, labels=False)
```

Bin column into n buckets.

pandas provides a large set of vector functions that operate on all columns of a DataFrame or a single selected column (a pandas Series). These functions produce vectors of values for each of the columns, or a single Series for the individual Series. Examples:

max()
Element-wise max.

clip(lower=-10, upper=10)
Element-wise min.

abs()
Absolute value.

The examples below can also be applied to groups. In this case, the function is applied on a per-group basis, and the returned vectors are of the length of the original DataFrame.

shift()
Copy with values shifted by 1.

rank(method="dense")
Ranks with no gaps.

rank(method="min")
Ranks. Ties get min rank.

rank(pct=True)
Ranks rescaled to interval [0, 1]

rank(method="first")
Ranks. Ties go to first value.

Plotting

```
df.plot.hist()
```

Histogram for each column

```
df.plot.scatter(x="var", y="var")
```

Scatter chart using pairs of points

Combine Data Sets

```
adf + bdf
```

Join matching rows from bdf to adf.

```
pd.merge(adf, bdf, how="left", on="x1")
```

Join matching rows from bdf to adf.

```
pd.merge(adf, bdf, how="right", on="x1")
```

Join matching rows from adf to bdf.

```
pd.merge(adf, bdf, how="inner", on="x1")
```

Join data. Retain only rows in both sets.

```
pd.merge(adf, bdf, how="outer", on="x1")
```

Join data. Retain all values, all rows.

Filtering Rows

```
adf[adf.x1.isin(bdf.x1)]
```

All rows in adf that have a match in bdf.

```
adf[~adf.x1.isin(bdf.x1)]
```

All rows in adf that do not have a match in bdf.

Set-like Operations

```
adf & bdf
```

Intersection.

```
adf | bdf
```

Union.

```
adf ^ bdf
```

Symmetric difference.



2. 활용

1) pandas 객체 추출

- ① `iloc(Integer location)`: 정수 위치 기반 인덱싱
- ② `loc(Label based location)`: 레이블 기반 인덱싱

객체 추출 관련 함수	설명
<code>df.sample</code>	데이터 프레임 내에서 랜덤 샘플 추출
<code>df.dropna</code>	결측 데이터를 제거
<code>df.fillna</code>	결측 데이터를 보간
<code>df.drop_duplicates</code>	중복 데이터를 제거
<code>df.head</code>	위에서부터 데이터 프레임 일부 추출
<code>df.tail</code>	아래에서부터 데이터 프레임 일부 추출

객체 변환 관련 함수	설명
<code>df.melt</code>	열을 행으로 변환
<code>df.pivot</code>	특정 열의 값을 새로운 열로 변환
<code>pd.concat</code>	데이터프레임을 행/열 기준으로 병합
<code>pd.merge</code>	공통 열 기준으로 데이터프레임 병합

집계 관련 함수	설명
<code>df[[columns]].value_counts</code>	컬럼들 기준으로 값의 개수를 카운팅
<code>df.groupby</code>	특정 열을 기준으로 그룹화



1.정의

1) 공간 데이터를 처리하고 분석하기 위한 라이브러리

- ① 공간 데이터를 생성하거나 저장 가능
- ② 공간 데이터에 대한 결합, 분할, 추출 등이 가능

지도자료 형식

벡터 형식

점, 선, 다각형 등의 형식의 자료



래스터 형식

격자 형식의 자료

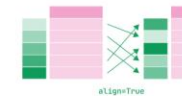


GeoPandas : : CHEAT SHEET

Geometric Confirmation

```
gp = geopandas.GeoSeries()
```

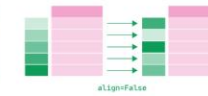
- gp.contains(other, align=True)** Returns a boolean of `True` for each aligned geometry that contains other.
- gp.contains(other, align=False)** Returns a boolean of `True` for each aligned geometry that contains other.
- gp.contains(other, align=True, level="point")** Returns a boolean of `True` for each aligned geometry that contains other.
- gp.contains(other, align=True, level="line")** Returns a boolean of `True` for each aligned geometry that contains other.
- gp.contains(other, align=True, level="polygon")** Returns a boolean of `True` for each aligned geometry that contains other.
- gp.contains(other, align=True, level="multipolygon")** Returns a boolean of `True` for each aligned geometry that contains other.



Geometric Operations

```
From shapely.ops import linemerge, polygonize
```

- gp.boundary** Returns a `GeometryArray` of the boundaries of each geometry.
- gp.buffer(distance, resolution=100)** Returns a `GeometryArray` of the buffers of each geometry.
- gp.intersection(other, align=True)** Returns a `GeometryArray` of the intersection of each geometry with other.
- gp.intersection(other, align=False)** Returns a `GeometryArray` of the intersection of each geometry with other.
- gp.intersection(other, align=True, level="point")** Returns a `GeometryArray` of the intersection of each geometry with other.
- gp.intersection(other, align=True, level="line")** Returns a `GeometryArray` of the intersection of each geometry with other.
- gp.intersection(other, align=True, level="polygon")** Returns a `GeometryArray` of the intersection of each geometry with other.
- gp.intersection(other, align=True, level="multipolygon")** Returns a `GeometryArray` of the intersection of each geometry with other.



GeoPandas : : CHEAT SHEET

Geometric Creation (shapely)

```
From shapely.geometry import Point, MultiPoint, LineString, Polygon, MultiPolygon
```

- Point(x, y)** Returns a `Point` object.
- MultiPoint(points)** Returns a `MultiPoint` object.
- LineString(points)** Returns a `LineString` object.
- Polygon(points)** Returns a `Polygon` object.
- MultiPolygon(polygons)** Returns a `MultiPolygon` object.

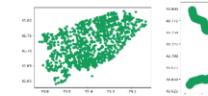


GeoPandas : : CHEAT SHEET

Geometry Operations

```
gp = geopandas.GeoSeries()
```

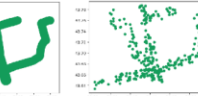
- gp.difference(other, align=True)** Returns a `GeometryArray` of the difference of each geometry with other.
- gp.difference(other, align=False)** Returns a `GeometryArray` of the difference of each geometry with other.
- gp.difference(other, align=True, level="point")** Returns a `GeometryArray` of the difference of each geometry with other.
- gp.difference(other, align=True, level="line")** Returns a `GeometryArray` of the difference of each geometry with other.
- gp.difference(other, align=True, level="polygon")** Returns a `GeometryArray` of the difference of each geometry with other.
- gp.difference(other, align=True, level="multipolygon")** Returns a `GeometryArray` of the difference of each geometry with other.



Misc. Operations

```
import geopandas as gpd
```

- gpd.GeoDataFrame.from_file(filename)** Returns a `GeoDataFrame` from a file.
- gpd.GeoDataFrame.to_file(filename)** Writes a `GeoDataFrame` to a file.
- gpd.GeoSeries.to_file(filename)** Writes a `GeoSeries` to a file.
- gpd.GeoSeries.to_crs(crs)** Returns a `GeoSeries` with a new coordinate reference system.
- gpd.GeoSeries.to_json()** Returns a `GeoSeries` with a new coordinate reference system.



GeoPandas : : CHEAT SHEET

CRS Examples

```
canada = gpd.read_file("canada.shp")
```

- canada.to_crs("EPSG:31436")** Returns a `GeoSeries` with a new coordinate reference system.
- canada.plot()** Returns a `GeoSeries` with a new coordinate reference system.



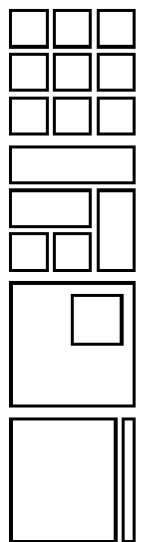
01.matplotlib

- 1) python의 데이터 시각화를 위한 라이브러리
- 2) 시각화한 자료를 다양한 확장자로 저장이 가능

10

2. 활용

- 1) 한 그림에 여러 그래프를 그릴 수 있음
- 2) 여러 그래프를 배열하여 그릴 수도 있음
- 3) 그림의 여백을 상세하게 수정할 수 있음

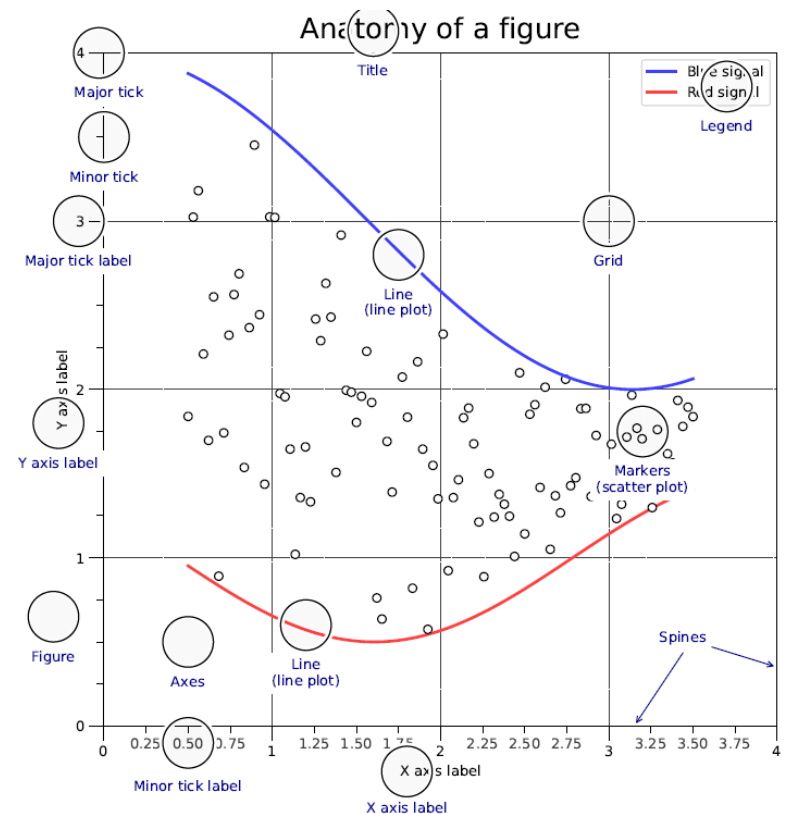
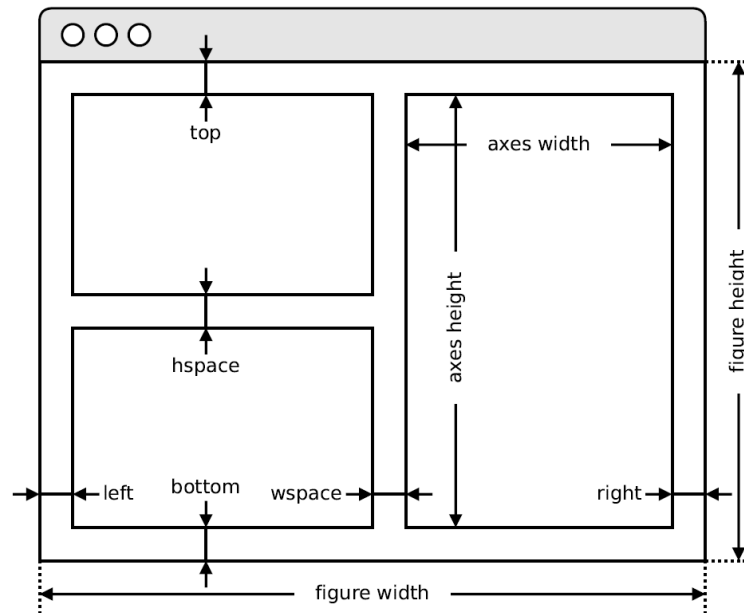


```
subplot[s](rows, cols, ...) API
fig, axs = plt.subplots(3, 3)
```

```
G = gridspec(rows, cols, ...) API
ax = G[0, :]
```

```
ax.inset_axes(extent) API
```

```
d=make_axes_locatable(ax) API
ax = d.new_horizontal('10%')
```





1.정의

1) python의 머신러닝 패키지 중 하나

① 데이터 전처리

- StandardScaler, MinMaxScaler, RobustScaler
- OneHotEncoder, LabelEncoder

② 데이터 분할

- train_test_split, Kfold, LeaveOneOut
- StratifiedShuffleSplit, StratifiedKFold: 특정 클래스 비율 유지
- GroupShuffleSplit, GroupKFold: 특정 그룹(환자ID, 사용자ID 등)을 기준으로 랜덤하게 분할





1.정의

1) python의 머신러닝 패키지 중 하나

① 다양한 모델 지원(지도, 비지도 학습)

- 지도학습 – 분류/회귀

- 트리기반 – DecisionTree, RandomForest, GradientBoosting, XGB, LGB, AdaBoost

- 그 외 – Logistic Regression, Linear Regression, Lasso, Ridge, SVM, LDA, QDA

- 비지도학습

- 차원축소 – PCA, SVD, tSNE, MDS

- 클러스터링 – Kmeans, DBSCAN 등

- 연관규칙 – Apriori

② 모델 학습 및 하이퍼 파라미터 튜닝

③ 모델 평가 기능 지원





Thank You

Email: qkdrk777777@naver.com