Python 기초

3. Pandas

SK 네트웍스

Data Insight

개요

Pandas Series

Pandas Dataframe

Pandas with SQL Server

개요

- ✓ NumPy의 한계와 Pandas의 장점
- ✓ 목표

NumPy의 한계와 Pandas의 장점

- ✓ NumPy
  - 오직 하나의 data type
- ✓ Pandas
  - Column별 data type 지정 ➔ Table 형태(Dataframe)의 데이터 구조
  - Missing Value 처리
  - Relational operations
  - Time series 기능

목표

Pandas의 모든 것을 다루지 않습니다. Machine Learning 또는 Deep Learning을 공부하기 위해 최소한 알아야 하는 범위만을 다룹니다.

1. csv 파일 읽는법.
2. dataframe을 만드는 법.
3. dataframe을 다루기.
4. head, tail
5. 정렬
6. numpy array와 변환,
7. 다양한 dataframe 연산.

Pandas Series

- ✓ Pandas Series 생성 & Indexing
- ✓ Index 이름 변경

Pandas Series 생성 및 Indexing

✓ Series란?
- 1차원 Array 자료 구조
- index와 value로 구성
- SeriesName[#] 혹은 [#:#]로 값 조회

```
1    92400
2    92100
3    94300
4    92300
dtype: int64
```

✓ Numpy.array, list와 비교

Python

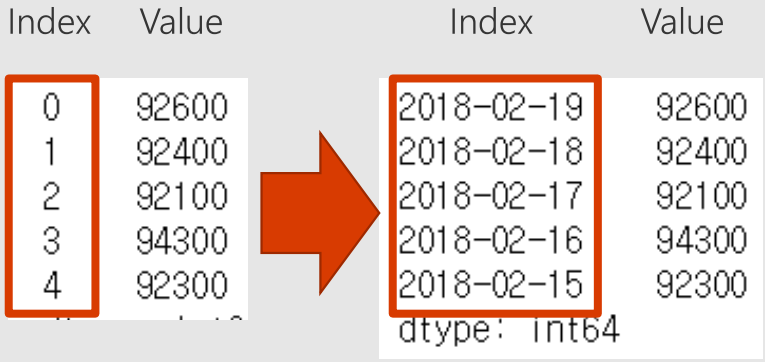
```
import pandas as pd
import numpy as np

# Pandas Series
stock = pd.Series([92600, 92400, 92100, 94300, 92300])
print(stock1[1:])

# Numpy rank 1 array
np_stock = np.array([92600, 92400, 92100, 94300,
92300])
print(np_stock[1:])

# python List
list_stock = [92600, 92400, 92100, 94300, 92300]
print(list_stock[1:] )
```

Index 이름 변경

✓ Series의 Index 이름을 변경할 수 있다.

Python

```
stock = pd.Series([92600, 92400, 92100, 94300, 92300]
                  , index=['2018-02-19', '2018-02-18'
                           , '2018-02-17', '2018-02-16'
                           , '2018-02-15'])
print(stock)
```



Index 이름 기준 연산

✓ Index 위치가 달라도, Index 값을 기준으로 연산 수행.

```
bill      165
daniel    175
james     179
dtype: int64
```

Python

```
# 인덱스 위치가 달라도, 인덱스 값을 기준으로 연산 수행.
score_math1 = pd.Series([90,80,85]
                        , index = ['james', 'daniel','bill'])
score_math2 = pd.Series([80,89,95]
                        , index = ['bill', 'james','daniel'])

print(score_math1 + score_math2)
```

실습 #1 : Pandas Series

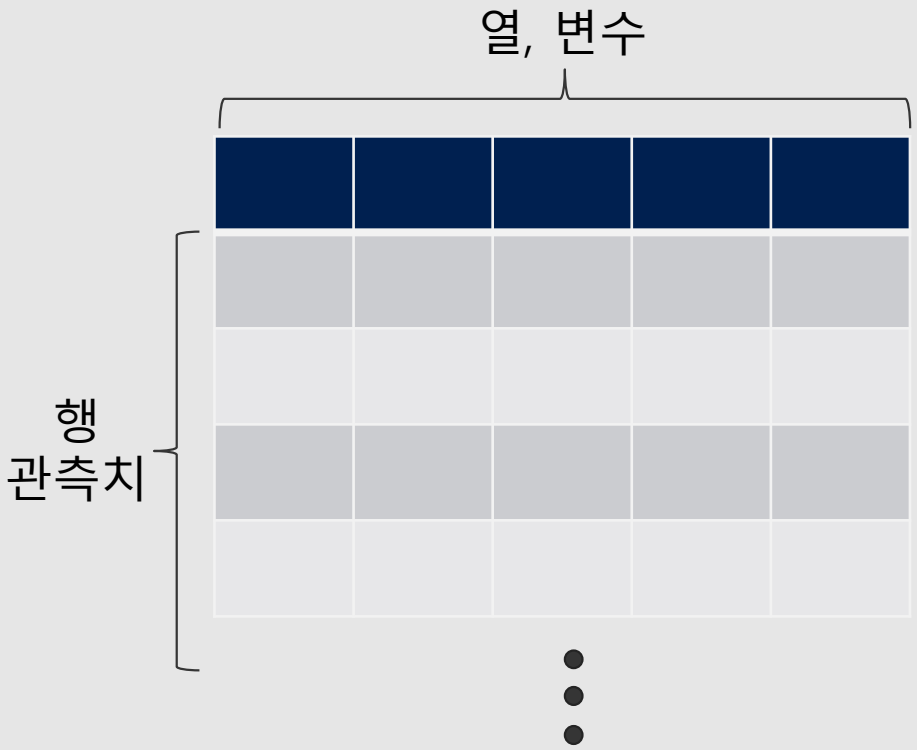
Pandas Dataframe

- ✓ Dataframe 생성
- ✓ CSV 파일에서 데이터 불러오기
- ✓ 데이터 미리보기
- ✓ 원하는 데이터 조회하기
- ✓ Dataframe 수정하기
- ✓ Group by, Join and Rolling
- ✓ NaN 찾고, 다른 값으로 바꾸기

Dataframe 생성

✓Dataframe이란?

- 데이터 분석에서 가장 중요한 데이터구조
- RDBMS에서의 테이블 형태
- 변수들의 집합
  → 각 열을 변수라고 부른다!
- 행렬과 다른 점은?



Dataframe 생성

✓ dp.Dataframe(dirctionary형태)

Python

```
# Dataframes 생성
d = {
  'col1': ['Item0', 'Item0', 'Item1', 'Item1']
 ,'col2': ['Gold', 'Bronze', 'Gold', 'Silver']
 ,'col3': [1, 2, 3, 4]
}
df = pd.DataFrame(d)
print(df)
```

CSV파일에서 데이터 불러오기

- ✓ 데이터를 가져오고자 할 때
  - Database에서 직접 가져오거나
  - CSV 파일에서 데이터를 불러온다.
- ✓ pd.read_csv( )
- ✓ pd.to_csv( )

Python

```
# Loading CSV files
df = pd.read_csv('Graduate_apply.csv', sep=',')

print(df.head())

df = pd.read_csv('Graduate_apply.csv'
                 , sep=','
                 , skipinitialspace=True)
print(df.head())

# to_csv
df.to_csv('./file.csv', header=True, index=False
          , encoding='utf-8')
```

Pandas DataType

Pandas dtype	Python type	NumPy type	Usage
object	str	string_, unicode_	Text
int64	int	int_, int8, int16, int32, int64, uint8, uint16, uint32, uint64	Integer numbers
float64	float	float_, float16, float32, float64	Floating point numbers
bool	bool	bool_	True/False values
datetime64	NA	datetime64[ns]	Date and time values
timedelta[ns]	NA	NA	Differences between two datetimes
category	NA	NA	Finite list of text values

데이터 살펴 보기①

- ✓상위, 하위 데이터 조회
  - df.head(#) : #값이 없으면 default는 5
  - df.tail(#) : #값이 없으면 default는 5
- ✓데이터프레임 모양 확인
  - df.shape
- ✓칼럼명들 조회
  - Df.columns

Python

```
# 첫 5개 행의 데이터를 보여줍니다.
df.head()

# 마지막 3개 행의 데이터를 보여줍니다.
df.tail(3)

# 데이터 프레임 모양 확인
df.shape

# 칼럼명 출력
Print(df.columns)
```

데이터 살펴 보기②

- ✓ 기초통계량
  - df.describe()
- ✓ Sorting
  - df.sort_values(), df.sort_index()

Python

```
# 간단한 통계 정보, 기초통계량,
df.describe()

# index로 정렬
df.sort_index(axis=0, ascending=False).head()

# 특정 컬럼의 값으로 정렬
df.sort_values(by=['admit', 'gpa’]
                          , ascending=False).head()
```

실습 #2 : csv 파일 불러와서 살펴 보기

원하는 데이터 조회하기 ①

✓ 칼럼명으로 조회

- df['칼럼명'] : Series로 결과 출력
- df[['칼럼명']] : Dataframe으로 결과 출력

Python

```
# 칼럼명으로 조회 1
df['gre'].head()
df['gre'].unique()

# 칼럼명으로 조회 2
df[['gre']].head()

# 두 개의 칼럼 동시 조회
df[['gpa', 'gre']].head()
```

원하는 데이터 조회하기 ②

✓Index로 조회

- df.iloc[row, column]

Python

```
# 행번호 1~3 조회
df.iloc[1:3]

# 0~4 rows & 0~2 columns
df.iloc[0:4, 0:2]
```

원하는 데이터 조회하기 ③

✓조건으로 조회

- df[조건]
- ==, !=, >=, <=, >, <
- .isin([val1, val2, ...])
- &(and), |(or)
- .str.contains(문자열)

Python

```
# Query by a single column value
df[df['gpa'] > 3.0].head()

# in 연산자
df[df['rank'].isin([1, 2])].head()

# &(and), |(or) 연산
df[(df['gpa'] > 3.0) & (df['rank'] == 3)].head()

df[(df['gpa'] > 3.0) | (df['rank'] == 3)].head()

# 문자열 포함하는 행 조회
df1[df1.col2.str.contains('ilver')]
```

실습 #3 : 원하는 데이터 조회하기

Dataframe 수정하기

- ✓ 인덱스, 칼럼명으로 수정
  - .loc[행이름, 칼럼명] 으로 조회
  - .iloc[행번호, 열번호] 으로 조회

Python

```
# 행이름(혹은 행번호)과 칼럼명으로 조회 & 수정
df1.loc[1, 'col2'] = 'Bronze and Gold'

# 행번호, 열번호로 조회 & 수정
df1.iloc[1, 1] = 'Bronze again'

# Replaces the column with the array. It could be
a numpy array or a simple list.
# Could also be used to create new columns
df1.loc[1:3, 'col3'] = ['Unknown0'] * 3
df1
```

Dataframe 수정하기

✓ Dummy Variable
- 범주형 데이터를 숫자로 변환하기

계절	
봄	1
여름	2
가을	3
겨울	4

봄	여름	가을	겨울
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

✓ 열 단위로 합치기
- 범주형 데이터를 숫자로 변환하기

Python

```
# 특정 칼럼의 Dummy Variable을 얻기
df_rank = pd.get_dummies(df['rank'])
print(df_rank.head())
print("------")

# Dummy 데이터를 원래 데이터와 합치기
df_new = pd.concat([df, df_rank], axis=1)
print(df_new.head())
print("------")
```

Dataframe 열 제거하기

- ✓ df.drop()
  - axis=1 : 칼럼을 삭제
  - inplace=True : df에 직접 삭제

Python

```
# 특정 칼럼 제거하기
df_new.drop('rank', axis=1, inplace=True)
print(df_new.head())

# 여러 칼럼 동시 제거하기
df_new.drop(['gre', 'gpa'], axis=1, inplace=True)
print(df_new.head())
```

실습 #4 : 데이터프레임 수정하기

Group by, Join, Rolling

✓ Group by
- 특정 열 기준으로 연속형 값 집계

Python

```
# rank별 평균 gre 값을 구하시오.
df.groupby(by=['rank'], as_index = False)['gre'].mean()

#as_index = True1이면 결과가 series로 나옴.
df.groupby(by = ['rank', 'admit']
                        , as_index=False)['gre', 'gpa'].mean()
```

실습 #5 : Group by

Group by, Join, Rolling

✓ Join : pd.merge()

- 특정 열(key) 기준으로 두 데이터 프레임 합치기
- 공통된 칼럼이 있으면 그 칼럼을 key로 사용

Python

```
mean_by_rank = df.groupby(by = ['rank'], as_index=False)['gre', 'gpa'].mean()
mean_by_rank.rename(columns = {'gre':'gre_mean', 'gpa':'gpa_mean'}, inplace = True)

std_by_rank = df.groupby(by = ['rank'], as_index=False)['gre', 'gpa'].std()
std_by_rank.rename(columns = {'gre':'gre_std', 'gpa':'gpa_std'}, inplace = True)

pd.merge(mean_by_rank, std_by_rank)
pd.merge(mean_by_rank, std_by_rank, on = 'rank', how = 'outer')
```

실습 #6 : Join(Merge)

Group by, Join, Rolling

✓ Rolling
- Sliding window로 이동하며 값 계산

Python

```
stock['Close_MA_3'] = stock['Close'].rolling(3).mean()
stock['Close_MA_3_lag1'] = stock['Close_MA_3'].shift()
stock['Close_MA_3(2)'] = stock['Close'].rolling(3, min_periods=1).mean()
stock['Close_MM_3'] = stock['Close'].rolling(3).max()
```

			최근 3 일간 평균	기간이 부족할땐 최소 1일	최근 3 일간 최대값
	Date	Close	Close_MA_3	Close_MA_3_lag1	Close_MA_3(2)
0	2000/3/27	4.125000	NaN	NaN	4.125000
1	2000/3/28	4.015625	NaN	NaN	4.070312
2	2000/3/29	4.000000	4.046875	NaN	4.046875
3	2000/3/30	3.843750	3.953125	4.046875	3.953125
4	2000/3/31	3.390625	3.744792	3.953125	3.744792
5	2000/4/3	3.437500	3.557292	3.744792	3.557292
6	2000/4/4	3.500000	3.442708	3.557292	3.442708
7	2000/4/5	3.484375	3.473958	3.442708	3.473958
8	2000/4/6	3.578125	3.520833	3.473958	3.520833
9	2000/4/7	3.609375	3.557292	3.520833	3.557292

실습 #7 : Rolling, Shift

Pandas with SQL Server

SQL Server 접속하기

✓ pyodbc 패키지

```
import pandas as pd
```

✓ 접속하기

- DB 연결설정하기

```
sql_conn = pyodbc.connect('DRIVER={ODBC Driver 13 for SQL Server}
                          ;SERVER=(local)
                          ;DATABASE=Retail
                          ;Trusted_Connection=yes')
```

- SQL 문 작성하고 실행하기

```
sql_sales = "select * from sales"
df_sales = pd.read_sql(sql_sales, sql_conn)
```