Pandas II

Function Application

- 다른 라이브러리의 기능을 Pandas 객체에 적용하려면 세 가지 중요한 방법을 알고 있어야함.
- 적절한 방법은 함수가 전체 DataFrame, 행 또는 열 또는 요소별로 작동하는지 여부에 따라 다름

  - 표 기능 응용 프로그램 : pipe()
  - 행 또는 열 Wise 함수 응용 프로그램 : apply()
  - 요소 함수 응용 프로그램 : applymap()

Reindexing

- DataFrame의 행 레이블과 열 레이블을 변경함.
- 다시 색인하려면 특정 축을 따라 주어진 레이블 세트와 일치하도록 데이터를 준수하는 것을 의미함.
- 여러 작업은 다음과 같은 인덱싱을 통해 수행 할 수 있음

  - 새로운 레이블 집합과 일치하도록 기존 데이터를 재정렬
  - 레이블 데이터가 없는 레이블 위치에 누락 값(NA)를 삽입

Interation

- Pandas 객체에 대한 기본 반복 동작은 유형에 따라 다름
- 시리즈를 반복 할 때, 그것은 배열과 같은 것으로 간주되며, 기본 반복은 값을 생성함
- DataFrame 및 Panel과 같은 다른 데이터 구조는 객체의 **키** 를 반복 하는 **dict-like** 규칙을 따름

  - **Series** – 값
  - **DataFrame** – 열 레이블
  - **Panel** – 항목 레이블

Sorting

- 두 가지 종류의 정렬을 제공함
  - 레이블 별
    - **sort_index ()** 메서드를 사용하면 축 인수와 정렬 순서를 전달하여 DataFrame을 정렬 할 수 있습니다. 기본적으로 정렬은 행 레이블에서 오름차순으로 수행됨
  - 실제 값 기준
    - 인덱스 정렬과 마찬가지로 **sort_values ()** 는 값을 기준으로 정렬하는 메서드입니다. 값을 정렬 할 DataFrame의 열 이름을 사용할 'by'인수를 사용함

텍스트 데이터 사용하기

- 문자열 데이터를 쉽게 조작할 수 있는 문자열 함수 세트를 제공함.
- 가장 중요한 점은 이러한 함수가 누락 된 / NaN 값을 무시 (또는 제외)한다는 것임.
- 거의 모든 이들 메서드는 파이썬 문자열 함수와 함께 작동함. 따라서 Series Object를 String Object로 변환 한 다음 작업을 수행이 필요함.

- lower(), upper(), len(), strip(), split(‘ ‘), cat(sep=‘ ‘), get_dummies(), contains(pattern), replace(a, b), repeat(value)
- count(pattern), startswith(pattern), endswith(pattern), find(pattern), findall(pattern), swapcase, islower(), isupper(), isnumeric()

옵션 및 사용자 정의

- 동작의 일부 측면을 맞춤 설정할 수있는 API를 제공함.
- API는 5 개의 관련 기능으로 구성됨
  - get_option()
  - set_option()
  - reset_option()
  - describe_option()
  - option_context()

데이터 인덱싱 및 선택

- 파이썬과 NumPy 색인 연산자 "[]"와 속성 연산자 "." 여러가지 상황에서 pandas의 구조를 빠르고 쉽게 접근하게 해줌
- 액세스 할 데이터의 유형을 미리 알지 못하기 때문에 표준 연산자를 직접 사용하면 몇 가지 최적화 제한이 있음.
- 프로덕션 코드의 경우이 장에서 설명한 최적화 된 팬더 데이터 액세스 방법을 활용하는 것이 좋음

Indexing	설명
.loc ()	라벨 기반
.iloc ()	정수 기반
.ix ()	레이블과 정수 모두 기반

통계 함수

- 통계 방법은 데이터의 동작을 이해하고 분석하는 데 도움을 줌.
  - Percent_change
  - Convariance
  - Correlation
  - Data Ranking

원도우 함수

- 수치 데이터 작업의 경우 롤링, 확대 및 지수 통계적으로 이동하는 가중치와 같은 몇 가지 변형을 제공함.
- **합계, 평균, 중앙값, 분산, 공분산, 상관 관계** 등이 있음

- .rolling()
- .expanding ()
- .wm()

집계

- 롤링, 확장 및 **ewm** 개체가 만들어 지면 데이터에서 집계를 수행하는 데 사용할 수있는 여러 가지 방법이 있음

- DataFrame에 집계 적용
  - 전체 데이터 프레임에 집계 적용
  - 데이터 프레임의 단일 열에 집계 적용
  - 데이터 프레임의 여러 열에 집계적용
  - 데이터 프레임의 단일 열에 여러 함수 적용
  - 데이터 프레임의 여러 열에 여러 함수 적용

누락된 데이터

- 누락 된 데이터는 실제 시나리오에서 항상 문제가 발생함.
- 기계 학습 및 데이터 마이닝과 같은 영역에서는 누락 된 값으로 인해 데이터 품질이 낮아 모델 예측의 정확성에 심각한 문제가 있음.
- 이러한 영역에서 모델을보다 정확하고 타당하게 만드는 주요 초점 포인트임

- 언제 그리고 왜 데이터를 놓쳤는가?
- 누락된 데이터 정리 / 채우기
- NaN을 스칼라 값으로 바꾸기
- NA값을 전진 및 후진 채우기
- 누락 된 값 삭제
- 누락 된 (또는) 일반 값 바꾸기

GroupBy

- 모든 **groupby**에는 원래 오브젝트에 대해 다음 선택중 하나가 포함됨
  - 개체 **분할(Splitting)**
  - 함수 **적용** 하기**(Applying)**
  - 결과 **결합(Combining)**

- 대부분의 경우 데이터를 세트로 분할하고 각 하위 세트에 일부 기능을 적용함. 적용 기능에서 다음 작업을 수행할 수 있음.
  - **집계(Aggregation)** - 요약 통계 계산
  - **변환(Transformation)** - 그룹 특정 작업 수행
  - **여과(Filtration)** - 일부 조건에서 데이터 삭제

Merging & Joining

- Pandas는 SQL과 같은 관계형 데이터베이스와 매우 유사한 모든 기능을 갖춘 고성능인 메모리 조인 연산을 사용함.
- Pandas는 DataFrame 객체 간의 모든 표준 데이터베이스 조인 작업의 진입 점으로 단일 함수 **merge를** 제공함.

- pd.merge(left, right, how='inner', on=None, left_on=None, right_on=None, left_index=False, right_index=False, sort=True)

  - **left** - DataFrame 객체임.
  - **Right** - 다른 DataFrame 객체임.
  - **On** - 가입 할 열 (이름). 왼쪽 및 오른쪽 DataFrame 개체 모두에 있어야함.
  - **Left_on** - 키로 사용할 왼쪽 DataFrame의 열임. DataFrame의 길이와 길이가 같은 열 이름이나 배열이 될 수 있음.

Merging & Joining

- **right_on** - 오른쪽 DataFrame에서 키로 사용할 열임. DataFrame의 길이와 길이가 같은 열 이름이나 배열이 될 수 있음.
- **left_index** - **True이면** 왼쪽 DataFrame의 색인 (행 레이블)을 조인 키로 사용. MultiIndex (계층 구조)가있는 DataFrame의 경우 레벨 수는 올바른 DataFrame의 조인 키 수와 일치해야함.
- **Right_index** - 올바른 DataFrame의 **left_index** 와 같은 사용법.
- **how** - '왼쪽', '오른쪽', '외부', '내부'중 하나임. 기본값은 inner입니다. 각 방법은 아래에 설명되어 있음.
- **Sort** - 사전 순으로 조인 키로 결과 DataFrame을 정렬함. True로 설정된 경우 False로 설정하면 많은 경우 성능이 크게 향상됨.

Concatenation

- Pandas는 **Series, DataFrame** 및 **Panel** 객체 를 쉽게 결합 할 수있는 다양한 기능을 제공함.
- pd.concat(objs,axis=0,join='outer',join_axes=None,ignore_index=False)

  - **objs** - Series, DataFrame 또는 Panel 객체의 시퀀스 또는 매핑함.
  - **axis** - {0, 1, ...}, 디폴트는 0임. 이것은 연결하는 축임.
  - **join조인** - { ' 내부 ' , ' 외부 ' }, 기본 ' 외부 ' . 다른 축 (들)에서 인덱스를 처리하는 방법. 노조 외부 및 교차로 내부.
  - **ignore_index** - 부울 값, 기본값은 False임. True이면 연결 축에서 인덱스 값을 사용하지 않아야함. 결과 축에는 0, ..., n – 1로 레이블이 지정됨.
  - **Join_axes** - Index 객체 목록임. 내외부 세트 로직을 수행하는 대신 다른 (n-1) 축에 사용할 특정 인덱스.

날짜기능

- 시계열을 확장하면 날짜 기능이 재무 데이터 분석에서 중요한 역할을함. Date 데이터로 작업하는 동안 다음과 같은 항목을 자주 접하게됨.
  - 날짜의 순서 생성
  - 날짜 계열을 다른 빈도로 변환

Timedelta

- 시간표(Timedelta)는 시간의 차이로, 예를 들어 일, 시간, 분, 초와 같은 차이 단위로 표현됨. 그들은 긍정적이거나 부정적일 수 있음.
- 다양한 인수를 사용하여 Timedelta 객체를 생성 할 수 있음.

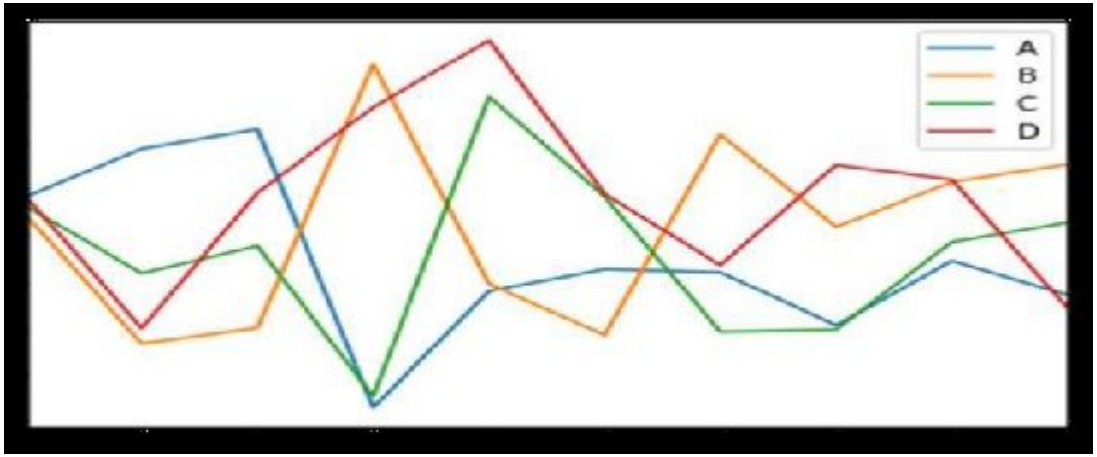
  - String
  - Integer
  - Data Offsets
  - to_timedelta()

범주형 데이터

- 종종 실시간으로 데이터에는 반복되는 텍스트 열이 포함됨. 성별, 국가 및 코드와 같은 기능은 항상 반복적임.
- 범주 형 변수는 제한된 수의 가능한 값만 취할 수 있음.
- 고정 길이 외에도 범주 형 데이터에는 순서가있을 수 있지만 숫자로 연산 할 수는 없음. 범주형은 pandas 데이터 형식임.
- 범주형 데이터 유형은 다음과 같은 경우에 유용함.
  - 몇 가지 다른 값으로 구성된 문자열 변수. 이러한 문자열 변수를 범주 형 변수로 변환하면 메모리가 절약됨.
  - 변수의 어휘 순서는 논리적 순서 (" 1 " , " 2 " , " 3 ")와 동일하지 않음. 범주로 변환하고 범주에 대한 순서를 지정하면 정렬 및 최소 / 최대는 어휘 순서 대신 논리적 순서를 사용함.
  - 다른 python 라이브러리에 대한 신호로서이 열은 범주 형 변수로 취급되어야함 (예 : 적절한 통계 방법 또는 플롯 유형 사용).

시각화

- Series와 DataFrame의이 기능은 **matplotlib 라이브러리 plot()** 메소드를 둘러싼 단순한 래퍼일뿐



시각화

- 인덱스가 날짜로 구성되어 있으면 위의 그림과 같이 **gct().autofmt_xdate ()** 를 호출하여 x축의 **서식**을 지정함
- **x** 와 **y** 키워드를 사용하여 한 열을 다른 열에 대해 플롯 할 수 있음.
- 플로팅 방법은 기본 선 그림 이외의 몇 가지 플롯 스타일을 허용함.
- 이 메소드는 **plot()에** 대한 kind 키워드 인수로 제공될 수 있음.

  - bar 또는 bar plots
  - 히스토그램의 히스토(hist for histogram)
  - boxplot 상자(box for boxplat)
  - 면적을 나타내는 '면적'('area' for area plots)
  - 산점도 'scatter'(for scatter plots)

IO도구

- **pandas I/O API를 사용하여 pd.read_csv()를 사용함(csv포멧인경우)**
- 텍스트 파일 (또는 flat 파일)을 읽는 두 가지 기능은 **read_csv()** 및 **read_table()**임.
- 모두 동일한 구문 분석 코드를 사용하여 표 형식의 데이터를 **DataFrame** 개체 로 지능적으로 변환함.