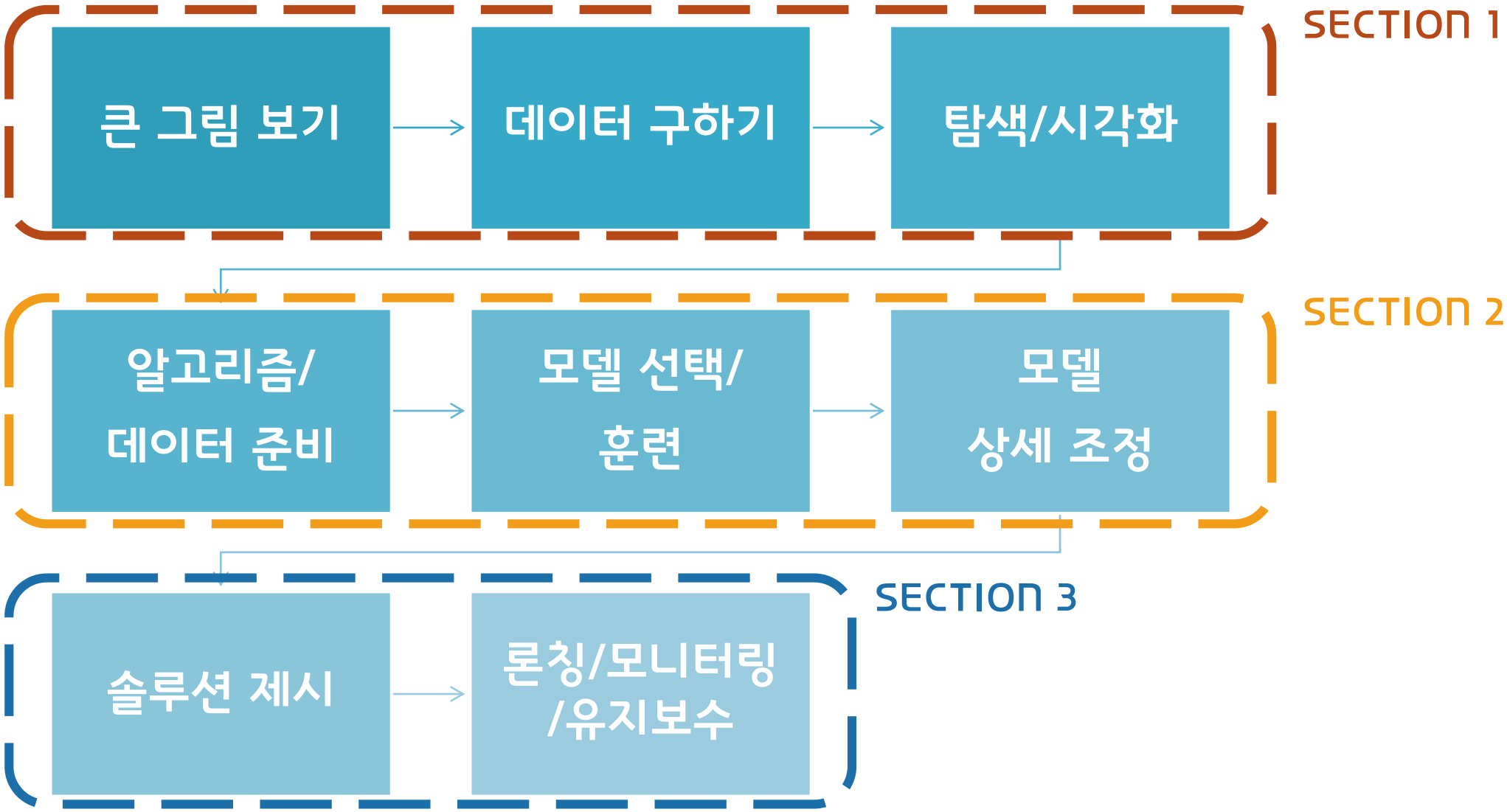
Hands On Machine Learning2

Chapter 2

전체 프로세스


큰 그림 보기
데이터 구하기
탐색/시각화
SECTION 1
알고리즘/
데이터 준비
모델 선택/
훈련
모델
상세 조정
SECTION 2
솔루션 제시
론칭/모니터링
/유지보수
SECTION 3

SECTION 1

큰 그림 보기

- 문제 정의, 사용 목적
- 성능 측정 지표 선택
- 가정 검사

데이터 구하기

- 작업 환경 만들기
- 데이터 다운로드
- 데이터 구조 훑어보기
- 테스트 세트 만들기

탐색/시각화

- 지리 데이터 시각화
- 상관관계 조사
- 특성 조합 실험

SECTION 1

큰 그림 보기

- 문제 정의, 사용 목적
- 성능 측정 지표 선택
- 가정 검사

데이터 구하기

- 작업 환경 만들기
- 데이터 다운로드
- 데이터 구조 훑어보기
- 테스트 세트 만들기

탐색/시각화

- 지리 데이터 시각화
- 상관관계 조사
- 특성 조합 실험

train_test_split vs. split_train_test

난수 초깃값 지정하는 random_state 매개변수는 train_test_split에 있다.

train_test_split 는 행의 개수가 같은 여러 개의 데이터셋을 넘겨서 같은 기반으로 나눌 수 있다. (무작위 샘플링)

=> 데이터 프레임이 레이블에 따라 여러 개로 나뉘어 있을 때 유용

Stratified Sampling (계층적 샘플링)

계층은 무작위 샘플링보다 모델에 적합한 샘플을 추출할 수 있다는 점에서 자주 이용하는 방법이다.

그러나 너무 많은 계층으로나눈다면 계층의 중요도를 추정하는데에 있어 편향이 발생할 것이다.

pd.cut()함수를 사용해 카테고리 특성을 만들 수 있다. (카테고리 1은 0-1.5까지의 범위)

SECTION 2

알고리즘/ 데이터 준비

- **데이터 정제**
- **텍스트/범주형 특성**
- 변환기
- 특성 스케일링
- 변환 파이프라인

모델 선택/ 훈련

- 훈련세트 훈련, 평가
- 교차 검증 평가

모델 상세 조정

- 그리드 탐색
- 랜덤 탐색
- 앙상블
- 최상 모델/오차 분석
- 테스트 세트로 시스템 평가

- **SimpleImputer (정제)**

누락된 값을 다뤄준다.
수치형 데이터만 중간 값을 다루며 imputer 객체의 fit() 메소드로 적용이 가능하다.

- **Imputer**

특성의 중간값 계산, 결과를 객체의 statistics속성에 저장한다.

- **OrdinalEncoder (텍변특)**

Text->num으로 바꿀 때. Categories_인스턴스 변수 사용, 배열 리스트 반환, 카테고리 목록 얻기
문제: 가까이 있는게 멀리 있는 친구보다 비슷하다고 여길 수 있다
순서가 있는 변수라면 ㄱ ㅊ

- **OneHotEncoder (텍변특)**

더미 특성. 출력을 희소 행렬로 해 줌

- **인코딩** : 더미를 만드는 것이라고 생각하면 쉬움

- **=> 희..소..행..렬..?**

SciPy Sparse Matrix (사이파이 희소행렬)
사이파이 희소행렬은 카테고리가 있는 범주형 특성이 수천 개 있을 대 효율적이다. 이를 원핫인코딩하면 열이 수천 개인 행렬로 변하며, 각 행은 1이 하나 뿐이고 이외에는 0으로 채워진다.

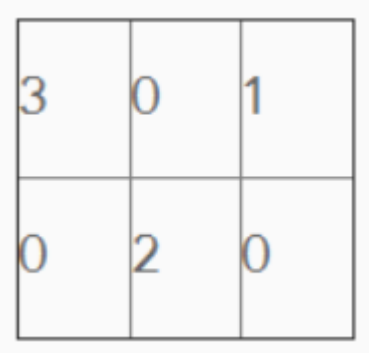
0을 모두 메모리에 저장하는 것은 낭비이므로 희소 행렬은 0이 아닌 원소의 위치만 저장한다. 이 행렬을 거의 일반적인 2차원 배열처럼 사용할 수 있으나 넘파이로 바꾸려면 toarray() 메소드를 소환하기만 하면 된다.

https://ko.wikipedia.org/wiki/%ED%9D%AC%EC%86%8C%ED%96%89%EB%A0%AC

희소행렬이란?
대부분의 값이 0으로 채워진 행렬을 희소 행렬이라고 하며, 대부분 0이 아닌 값으로 채워진 행렬은 밀집 행렬이라고 한다(Dense Matrix).

BOW형태를 가진 언어모델의 피처 벡터화는 대부분 희소 행렬을 만든다.

희소 행렬은 불필요한 0 값이 많기 때문에 메모리 낭비가 심하며, 행렬 크기가 커서 연산 시 시간도 많이 소모된다. 따라서 이런 희소 행렬의 메모리 낭비를 막기 위해 변환하는 대표적인 방식이 COO와 CSR형식이다.



COO 형식
COO(Coordinate: 좌표) 형식은 0이 아닌 데이터만 별도의 배열에 저장하고, 그 데이터가 가리키는 행과 열의 위치를 별도의 배열에 저장하는 방식이다. 예를 들어 2 x 3 행렬이 있다고 할 때, COO 형식은 0이 아닌 값, 행 위치 값, 열 위치 값에 대한 배열로 표현한다.

예를 들어 위와 같은 행렬이 있다고 할 때, 0이 아닌 값은 [3, 1, 2]이며 3의 행과 열의위치는 (0, 0), 2는 (1, 1), 1은 (0, 2) 이다.

COO 형식은 0이 아닌 값, 행 위치 값, 열 위치 값에 대한 배열로 표현하며, 이 세 개의 배열만 저장해도 이를 통해 원본의 행렬을 구할 수 있게 된다. 이런 방법으로 원본형식을 다 저장하지 않아도 되기 때문에 메모리 낭비를 지양한다.

CSR 형식

CSR형식은 행과 열의 위치를 나타내기 위해 반복적으로 위치데이터를 상요하는 방식이다.

0	0	1	0	0	5
1	4	0	3	2	5
0	6	0	3	0	0
2	0	0	0	0	0
0	0	0	7	0	8
1	0	0	0	0	0

위와 같은 행렬이 있을 때, COO를 만들려면 [1, 5, 1, 4, 3, 2, 5, 6, 3, 2, 7, 8, 1]를 추출해야 하고, 0이 아닌 값의 행과 열의 위치를 나타내는 행렬을 구해보면

행 위치 배열 = [0, 0, 1, 1, 1, 1, 1, 2, 2, 3, 4, 4, 5],

열 위치 배열 = [2, 5, 0, 1, 3, 4, 5, 1, 3, 0, 3, 5, 0]이다.

행 위치 배열을 보면 순차적으로 반복적인 값이 나타나며 순차적으로 증가한다는 특성을 고려할 수 있다. 이렇게 하면 행 위치 배열의 고유한 값의 사작 위치만 표기하고 반복을 제거할 수 있다.

즉, 0의 시작 인덱스 0, 1의 시작 인덱스 2, 2의 시작 인덱스 7 이렇게 저장한다.

따라서 행 위치 배열을 시작 위치 배열인 [0, 2, 7, 9, 10, 12]로 변환하면 반복도 줄이고 메모리도 적게 사용이 가능하다. 마지막에는 총 항목 개수를 추가해주면 되며, 총 항목 13개를 적용한 최종 배열인 [0, 2, 7, 9, 10, 12, 13]이 도출된다.

즉, COO의 단점을 보완한 방식이 CSR방식이며, 사이킷 런의 CountVectorizer, TfidfVectorizer 클래스로 피처 벡터화된 행렬은 모두 CSR 형식이다.

변환기

사이킷 런은 상속이 아니라 덕 타이핑(상속/인터페이스)이 아니라, 객체의 속성이나 메소드가 객체 유형을 결정하는 방식)을 지원하므로
- fit() (self반환)
- transform()
- fit_transform()

메소드를 구현한 클래스를 만들면 된다.
마지막 메소드는 TransformerMixin을 상속하면 자동으로 생성되며 BaseEstimator를 사용하면 튜닝에 필요한 두 메소드 (get_params()와 set_params())를 얻을 수 있다.

스케일링

모든 특성의 범위를 같도록 만들어주는 방법
- min-max 스케일링
- 표준화

min-max 스케일링
: 정규화
MinMaxScaler 변환기를 통해 0-1사이 범위에 들게 스케일을 조정하거나 feature-range매개변수를 지정해 범위를 변경할 수 있다.

표준화
먼저 평균을 빼고 표준편차로 나눠서 결과 분포의 분산이 1이 되도록 한다. 표준화는 상한 하한이 없어 어떤 알고리즘에서는 문제가 되기도 한다. 그러나 표준화는 이상치에 영향을 덜 받는다.
StandardScaler를 통회 변환할 수 있따.

**주의
모든 변환기에서 스케일링은 전체 데이터가 아닌 train 세트에만 fit()메소드를 적용해야 하며, 이 다음 train, test, 그리고 새 데이터에 대해 transform()메소드를 적용한다.

- **Min-max scaling (정규화/스케일링)**
- **StandardScaler (스케일링)**

각 feature 평균 0, 분산 1. 모든 특성 같은 스케일~

- **RobustScaler (스케일링)**

이상치에 좀 둔감한 친구, 평균 분산 대신 분위랑 중위값 사용함. 모든 특성이 같은 크기 가지는 점은 위에 친구랑 비슷

- **Normalizer (스케일링)**

위 세 친구가 columns의 통계치 이용할 때, 노멀라이저는 행마다 각각 정규화가 되는 식. 유클리드 거리 1이 되게 데이터 조절.

모델 선택/훈련 교차 검증 평가

<결정 트리 모델을 평가하는 방법>

1. train_test_split을 활용, 훈련세트를 다시 나누고 더 작은 쪽에서 모델 훈련 후 큰 쪽에서 평가
2. k-겹 교차 검증

: 순련 세트를 폴드(fold)라고 불리는 10개의 서브셋으로 무작위 분할. 결정 트리 모델을 10번 훈련 후 평가하며 매번 다른 폴드를 선택해 평가에 사용하고, 나머지 9개는 훈련에 사용한다. 10개의 평가 점수가 담긴 배열이 결과가 됩니다.