**데이터 탐색과 시각화**

**1. 데이터프레임과 데이터 탐색**

head() : 데이터프레임에 포함된 처음 5개의 샘플 확인

info() : 데이터셋 정보 요약

-> 각 컬럼별 항목, 데이터 개수, 데이터 타입 등을 표시

value\_count() : 범주형 특성 탐색

describe() : 수치형 특성 탐색

수치형 특성별 히스토그램 – 폰트 크기, 이미지 사이즈 등을 설정 후 출력

**2. 훈련셋과 테스트셋**

train\_test\_split() : 훈련셋과 테스트셋을 구분

무작위로 데이터의 20% 정도를 테스트셋으로 지정할 수 있음.

계층 샘플링 : 전체 모집단을 “서로 겹치지 않는 하위 집단”으로 나눈 뒤, 각 계층에서 일정 수의 표본을 추출하는 방식.

-> 전체 모집단의 다양한 특성이 더 잘 반영되게 됨.

-> 기준이 되는 구간을 새로운 범주형 특성으로 추가, stratify=housing[“기준”] 키워드 인자 활용

계층 샘플링에 사용하기 위해 추가된 컬럼은 모델 훈련에서 필요하지 않기에 다시 삭제한다.

**3. 데이터 시각화**

훈련셋만을 대상으로 탐색과 시각화를 적용. 훈련셋 원본을 그대로 두고 복사해서 사용

지리적 데이터 시각화 : grid 키워드 인자 / alpha 키워드 인자(투명도 조절) / 색상 활용

corr() : 모든 수치형 특성 간의 표준 상관계수 계산

**데이터 준비: 정제와 전처리**

**데이터 정제**

결측치가 있는 샘플 확인 후 처리 방법 결정

- 결측치 특성 포함 샘플 삭제

- 결측치를 포함한 특성 삭제

- 결측치를 해당 특성의 중앙값/평균값 등으로 대체

**1. 사이킷런 API**

- 추정기 : fit() 지원. 일반적으로 변환기/예측기 둘 중 하나

- 변환기 : fit()(데이터 변환에 필요한 정보 계산), transform()(데이터 변환 실행), fit\_transform() 지원. 데이터 정제와 전처리 과정에서 주로 사용.

- 예측기 : fit()(모델의 훈련 관장), predict()(모델의 훈련이 종료된 후 실전에서 예측값을 계산할 활용), score()(predict() 메서드가 예측한 값의 성능을 측정) 지원. 일반적으로 모델이라 불림.

**2. SimpleImputer 변환기: 결측치 처리**

이를 사용하면 결측치를 해당 특성의 중앙값/평균값으로 대체하는 것을 쉽게 처리할 수 있다.

strategy 속성 : 결측치를 대체할 방식을 지정. 평균('mean'), 중앙값('median'), 최빈값('most\_frequent'), 상숫값('constant') 중 하나 선택

**3. 입력 데이터셋과 타깃 데이터셋**

- housing: 주택 중위가격 특성을 제외한 데이터를 지도 학습에 필요한 입력 데이터셋으로 사용하기 위해 변수가 가리키는 값을 변경

- housing\_labels: 주택 중위가격은 지도 학습의 타깃으로 활용

-> 주택 중위가격을 타깃값(레이블)으로 지정

**4. OneHotEncoder 변환기: 범주형 특성 전처리**

OneHotEncoder 클래스 : 범주 수 만큼의 새로운 특성 추가, 해당되는 범주와 관련된 특성값은 1, 나머지 특성값은 0.

-> 변환기의 반환값은 기본적으로 넘파이 어레이 자료형이다. 따라서 훈련에 반

드시 필요하지는 않지만 변환된 데이터셋을 특성명을 함께 표현하고 확인하고

싶을 때 데이터프레임으로 표현하면 좋다. 이를 위해 새로 생성된 특성명과 기존의 인덱스를 이용한다.

**5. MinMaxScaler와 StandardScaler 변환기: 수치형 특성 스케일링**

정규화 : min-max 스케일링

표준화

**6. FunctionTransformer 변환기**

FunctionTransformer 는 fit() 을 사용하지 않고 바로 transform() 을 적용하는 변환기를 생성할 때 활용한다.

로그 변환기 : 한쪽으로 치우친 특성에 로그함수를 적용한다.

비율 계산 변환기 : 비율과 관련된 새로운 특성을 생성할 때 사용한다.

**7. 군집 변환기 : 사용자 정의 변환기**

군집 유사도 적용 변환기는 직접 구현해야 한다. (단, 변환기를 정의할 때 BaseEstimator 클래스와 TransformerMixin 클래스를 상속하도록 함에 주의한다. 그래야 변환기의 기본 기능을 모두 갖추게 된다.)

**파이프라인**

**1. Pipeline 클래스**

여러 개의 전처리 작업과 모델을 하나의 객체로 묶어주는 도구. 교차 검증, 그리드 서치 등에 바로 사용 가능

make\_pipeline() 함수 활용(각 변환기의 이름은 자동 생성)

**2. ColumnTransformer 클래스**

특성별로 파이프라인을 지정할 수 있다.

make\_column\_transformer() 함수 활용 : 지정된 자료형을 사용하는 특성들만을 뽑아줌. 파이프라인을 적용할 특성을 선택하는 데 사용

**3. 데이터셋 변환 파이프라인**

비율 변환기, 로그 변환기, 군집 변환기, 기본 변환기 등을 종합

**모델 선택과 훈련**

**1. 모델 훈련과 평가**

모델을 정하여 훈련한 후 예측값의 오차 확인

-> RMSE가 많이 높을 경우 과소적합 발생

-> RMSE가 많이 낮을 경우 과대적합 발생

**2. 교차 검증**

교차 검증을 이용하여 훈련 과정 중의 모델 성능 평가를 진행한다. 교차 검증은 경우에 따라 많은 시간을 요한다.

-> 교차검증의 결과가 훈련셋에 대한 평가보다 안좋을 경우, 모델이 훈련셋에 너무 특화되어있음(과대적합)

**모델 미세 조정**

**1. 그리드 탐색**

여러 하이퍼파라미터 조합을 체계적으로 탐색해서 가장 성능 좋은 조합을 찾는 방법. 계산량이 많다.

-> 모델의 성능에 영향을 주는 하이퍼파라미터는 학습으로 자동 결정되지 않기 때문에 직접 선택해줘야함.

cv\_results\_ 속성 활용 : 그리드 탐색 과정에서 훈련된 모델 각각의 평가지표를 확인할 수 있다.

**2. 랜덤 탐색**

무작위로 선택한 n개의 하이퍼파라미터 조합에 대해 m-겹 교차 검증을 진행하기에 총 nxm번 훈련을 진행한다.

**3. 최적 모델 활용**

특성 이름을 추가한 후 특성별 상대적 중요도를 기준으로 내림차순으로 정렬하면 가장 중요한 특성을 확인할 수 있다.

**최적 모델 저장 및 활용**

**joblib 모듈**

최적의 모델을 이름을 지정하여 저장한다.

joblib.load() 함수를 이용하여 저장된 모델을 불러올 수 있다. 다만, 모델 정의에 필요해서 사용자가 직접 정의한 함수, 클래스 등을 모두 함께 불러와야 한다.