**Chapter 2. End-to-End Machine Learning Project**

예제 프로젝트를 끝까지 살펴보기 주요 단계

1. Look at the big picture.
2. Get the data.
3. Discover and visualize the data to gain insights.
4. Prepare the data for Machine Learning algorithms.
5. Select a model and train it.
6. Fine-tune your model.
7. Present your solution.
8. Launch, monitor, and maintain your system.

Working with real data

인위적으로 만든 데이터 셋이 아닌 실제 데이터를 가지고 실험하는 것이 공부하는데 가장 좋은 방법.

우리가 사용할 수 있는 데이터

* Popular open data repositories:  - UC Irvine Machine Learning Repository  - Kaggle datasets  - Amazon’s AWS datasets
* Meta portals (they list open data repositories):  - http:// dataportals.org/  - http:// opendatamonitor.eu/  - http:// quandl.com/
* Other pages listing many popular open data repositories:  - Wikipedia’s list of Machine Learning datasets  - Quora.com question  - Datasets subreddit

이번 장에서 캘리포니아주택가격에 대한 데이터사용(1990년 조사 기반)

이 데이터가 학습을 위해 많은 양을 포함, 카테고리 추가



1. **Look at the big picture.**

수행해야 할 첫 번째 과제는 캘리포니아 인구 조사 데이터를 사용하여 캘리포니아 주거 가격 모델을 구축하는 것.

이 데이터에는 캘리포니아의 각 블록 그룹에 대한 인구, 중간 소득, 주택 가격 중앙값 등과 같은 측정 기준이 있다.(블록 그룹은 일반적으로 인구가 600-3,000 명입니다. 우리는 그걸 짧게 "구역"이라고 부를 것입니다.)

모델은 이 데이터로부터 학습하고 다른 모든 지표를 고려할 때 다른 구역의 주택 가격의 중간 값을 예측할 수 있어야합니다.

//////////////

첫 번째 일은 Machine Learning 프로젝트 점검표를 확인. 대부분의 기계 학습 프로젝트에 대해 합리적으로 잘 작동해야하지만 사용자의 요구에 맞게 조정

1. Frame the problem and look at the big picture.
2. Get the data.
3. Explore the data to gain insights.
4. Prepare the data to better expose the underlying data patterns to Machine Learning algorithms.
5. Explore many different models and short-list the best ones.
6. Fine-tune your models and combine them into a great solution.
7. Present your solution.
8. Launch, monitor, and maintain your system.
   1. **Frame the problem**

가장 먼저 해야할 질문은 ‘비즈니스 목표가 정확히 무엇인지’ 입니다.

모델을 만드는 것이 최종 목표가 아닐 수도 있습니다. 회사는 이 모델을 어떻게 사용하고 이익을 얻을까? 이는 문제의 틀을 정하는 방법, 선택할 알고리즘, 모델을 평가하는 데 사용할 성능 측정 방법 및 조정을 위해 얼마나 많은 노력을 기울여야하는지 판단하기 때문에 중요합니다.

귀하의 상사는 모델의 산출물이 다른 많은 기계 신호 시스템과 함께 다른 기계 학습 시스템에 공급 될 것이라고 대답합니다. 이 다운 스트림 시스템은 특정 지역에 투자 할 가치가 있는지 여부를 결정합니다. 이 권리는 수익에 직접적인 영향을 주기때문에 중요합니다.



다음으로 물어볼 질문으로 현재 솔루션이 어떻게 보이는지 (있는 경우)입니다. 종종 문제 해결 방법에 대한 통찰 뿐만 아니라 참조 실적을 제공합니다.

상사는 지구 주택 가격이 현재 전문가가 수동으로 추정하고 있다고 대답합니다. 팀이 구역에 대한 최신 정보를 수집하고 주택 가격의 중간 값을 얻을 수 없는 경우 복잡한 규칙을 사용하여 추정합니다.이것은 값 비싸고 시간이 많이 소요되며 예상치는 좋지 않습니다. 실제 주택 가격의 중앙값을 알아낼 수 있는 경우에는 추정치가 10 % 이상 떨어져 있음을 종종 깨닫게 됩니다.

따라서 해당 지구에 대한 다른 데이터를 고려해 지구의 주택 가격의 중간값을 예측하는 모델을 훈련하는 것이 유용 할 것이라고 회사는 생각합니다.

이 모든 정보를 통해 시스템 설계를 시작할 준비가 되었습니다. 먼저 문제의 프레임을 만들어야 합니다.

**Is it supervised, unsupervised, or Reinforcement Learning?**

**Is it a classification task, a regression task, or something else?**

**Should you use batch learning or online learning techniques?**

이것은 지도학습의 전형적인 과업이다. 또한 값을 예측 하라는 메시지가 표시되고, 시스템이 예측을하기 위해 여러 가지 기능을 사용할 것이기 때문에 이것은 다 변수(multivariate) 회귀 문제입니다 (이는 구역의 인구, 중간 소득 등을 사용합니다). 마지막으로, 시스템에 지속적인 데이터 흐름이 없으며, 변화하는 데이터에 빠르게 적응할 필요가 없으며, 데이터가 메모리에 들어가기에 충분히 작기 때문에 batch 학습이 적합하다.

* 파이프라인 일련의 데이터 처리 구성 요소를 데이터 파이프라인이라고 합니다. 파이프 라인은 기계 학습 시스템에서 매우 일반적으로 사용. 많은 데이터를 조작하고, 변환해야 하기 때문입니다.  구성 요소는 비동기 적으로 실행됩니다.  각 구성 요소는 많은 양의 데이터를 가져 와서 처리하고 결과를 다른 데이터 저장소에 저장 한 다음 나중에 파이프 라인의 다음 구성 요소가 이 데이터를 가져 와서 자체 출력을 내보내는 식으로 진행  각 구성 요소는 상당히 독립적 입니다. 구성 요소 간의 인터페이스는 단순히 데이터 저장소입니다. 이를 통해 시스템을 매우 쉽게 파악할 수 있으며 (데이터 흐름 그래프를 사용하여) 여러 팀이 서로 다른 구성 요소에 집중할 수 있습니다. 또한 구성 요소가 고장나면 다운 스트림 구성 요소가 끊어진 구성 요소의 마지막 출력을 사용하여 정상적으로 계속 실행될 수 있습니다
  1. **Select a performance measure**

다음 단계로는 성능 척도를 측정. 회귀문제의 일반적인 성능 측정으로 평균 제곱근 편차(RMSE)가 있다.



일반적으로 RMSE가 회귀 작업을 위한 성능 측정에 대해 선호하는 벙법 이지만 많은 이상치 districts가 있다고 가정하면 평균 절대값 오차(MAE)가 사용될 수 있다.



위에 두 식은 두 벡터 사이의 거리를 측정하는 방법이다;

RMSE는 유클리드 거리측정방식을 이용하고 MAE는 맨하탄 거리 측정 방식을 이용

* 1. **Check the assumptions**

지금까지 가정한 것을 나열하고 검증하는 것.

1. **Get the data**

**2.1) Download the data**

사용하고 있는 머신에 다양성을 두기 위해 os를 import하여 처리

pandas를 통해 데이터 불러오고 함수를 정의함으로써 새로운 데이터가 추가되도 편리하게 재사용 가능

fetch\_housing\_data로 데이터 호출, load\_housing\_data 로 불러와서 진행

**2.2) Take a quick look at the data structure**

총 20640개의 데이터와 10개의 feature(다변수)

housing = load\_housing\_data()

housint.head() - 데이터 표로 나타냄

[housing.info](http://housing.info)() - 데이터의 총 수, 칼럼에 대한 수, 타입



total\_bedrooms에 누락된 데이터 있음

ocean\_proximity을 제외한 나머지는 숫자 속성 object로 텍스트 속성을 가진다

housing.describe() - 각각의 feature에 대한 개수, 평균, 표준편차 등

housing['feature name'].value\_counts() - feature name의 구성값 확인

matplotlib로 데이터 히스토그램 만들기



median\_income이 미국 달러로 나타나지 않음

housing\_median\_age, median\_house\_value 모두 상한선이 있다.

다른 속성을 가진다.

왼쪽보다 중앙에서 오른쪽으로 많이 확장된다 - 알고리즘 패텀 감지 어려움

추후 이런 속성들을 종 모양의 분포를 가질 수 있도록 변형 할 것.

**2.3) Create a test set**

테스트 셋을 만드는 것은 이론적으로 간단하다. 몇 개의 inst들을 랜덤으로 추출하면 된다. 일반적으로 테스트 셋은 20%정도를 추출

train\_set, test\_set = split\_train\_test(housing, 0.2)

이 방법은 잘 돌아가지만 프로그램을 다시 돌린다면 다른 테스트 셋 생성

처음 돌렸을 때 데이터 저장하고 이를 불러오거나 seed값을 설정해서 항상 같은 shuffled된 값들을 생성

두가지 방법 모두 데이터셋 업데이트하고 fetch할 때 문제 발생 - 각각의 inst 식별자(identifier)를 사용하여 inst가 테스트 셋에 포함되어야 하는지를 결정 -> 데이터 셋을 새로 고치더라도 일관되게 유지

별도의 index 컬럼을 추가해서 다른 행과 겹치지 않기 때문에 유니크하다

sklearn에서는 더 간단하게 데이터 셋과 테스트 셋을 나눌 수 있다

train\_set, test\_set = train\_test\_split(housing, test\_size=0.2, random\_state=42)

중간 소득이 주택 가격의 중앙값을 예측하는 데 매우 중요한 속성이라고 말한 전문가와 이야기했다고 가정, 중간 소득에 대한 히스토그램을보다 자세히 살펴 보겠습니다



대부분 값이 2에서 5에 밀집되어 있고 몇몇은 6을 초과한다 각 데이터 집합에 편향되지 않은 인스턴스들을 갖는 것이 중요하다.

중간소득에 1.5로 나눈 값을 반올림 처리하고 5보다 큰 값은 5로 합친 것이 아래 표



이제 소득 카테고리를 기반으로 계층화 된 샘플링을 수행 할 준비가 되었습니다.

이를 위해 Scikit-Learn의 StratifiedShuffleSplit 를 사용할 수 있습니다.



이제 원상태로 돌아가기 위해 income\_cat 속성을 삭제

1. **Discover and visualize the data to gain insights**

지금까지 전반적인 데이터의 구조와 테스트셋의 구성에 대해 살펴보았다.

**3.1) Visualizing geographical data**

우리의 데이터는 지리적 정보(경도와 위도)가 있기 때문에 scatterplot으로 시각화 하는 것은 좋은 아이디어다



특정 패턴을 보기 어려워 알파값을 0.1로 주어서 쉽게 시각화

housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.1)



이제 주택 가격에 대해 살펴본다

각 원의 반지름은 해당 지역의 인구 (옵션 s)

색상은 가격을 나타냅니다 (옵션 c)

jet는 파란색(낮은 값) 에서 빨간색(높은 가격) 범위의 미리 정의 된 색상 맵 (옵션 cmap)을 사용





이는 주택 가격이 위치(바닷가 근처)와 인구밀도와 매우 관련이 있다

아마도 클러스터링 알고리즘을 사용하여 주요 클러스터를 탐지하고 클러스터 센터의 근접성을 측정하는 새로운 기능을 추가하는 것이 유용 할 것입니다

**3.2) Looking for correlations**

데이터 셋이 매우 크지 않기 때문에 corr()함수를 사용하여 모든 속성 쌍들 사이의 표준상관계수를 쉽게 계산할 수 있다. 그리고 각 feature들이 얼마나 주택중간가격(median house value)과의 상관관계가 있는지 확인할 수 있다.



상관계수 범위는 -1에서 1까지이다. 1과 가까울수록 양의 상관관계, -1에 가까울수록 강한 음의 상관관계 0에 가까울수록 두 데이터간의 상관관계는 적다.



상관 계수는 선형 상관 관계 만 측정합니다

두 번째 행은 상관 계수가 1 또는 -1 인 예제를 보여줍니다. 이것이 사면과는 아무런 관련이 없음을 주목하십시오. 높이가 인치 단위이면 피트와 나노 단위의 높이와 상관계수가 1이다

단 행의 모든 플롯이 축이 명확하게 독립적이지는 않지만 상관 계수가 제로와 같은 점에 유의하십시오. 이는 비선형 관계의 예입니다

속성들 간의 상관관계를 확인하기 위한 다른 방법으로 pandas의 scatter\_matrix 함수를 사용

9개의 feature 각각에 대해 상관분석을 한다



같은 속성에 대해서는 히스토그램으로 표현(대각선)

**3.3) Experimenting with attribute combinations**

데이터를 기계 학습 알고리즘에 보내기 전에 정리해야 할 몇 가지 데이터 단점을 확인하고 속성, 특히 대상 속성과의 흥미로운 상관 관계를 발견했습니다.

Machine Learning 알고리즘에 대한 데이터를 실제로 준비하기 전에 해야 할 일은 다양한 속성 조합을 시험해 보는 것입니다.

예를 들어 총 구역에 있는 방의 총 개수는 실제로 얼마나 많은 가구들이 존재하는지 모르면 불필요한 정보

총 침실 수 자체는 유용하지 않다

필요한건 방의 개수와 비교, 가구당 인구 또한 흥미로운 조합의 속성





새로 추가된 방 개수당 화장실 수 속성은 총 방의 개수나 총 침실 수보다 관련이 높다

침실의 낮은 비율은 집 값이 비싼거와 연관이 있고 가구당 방의 개수는 총 방의 개수보다 관련이 있다

신속하게 첫 번째 합리적으로 좋은 프로토 타입을 얻을 수있는 통찰력을 얻는 것입니다. 그러나 이것은 반복적 인 프로세스입니다. 일단 프로토 타입을 작성하고 실행하면 출력을 분석하여 더 많은 통찰력을 얻고이 탐색 단계로 돌아갈 수 있습니다

1. **Prepare the data for machine learning algorithms**

머신러닝 알고리즘을 위해 데이터를 준비하는 시간

그러나 먼저 (strat\_train\_set을 다시 복사하여) 깨끗한 트레이닝 세트로 되돌리고, 예측 인자와 레이블을 분리합시다



**4.1) Data cleaning**

대부분의 기계 학습 알고리즘은 누락 된 기능을 사용할 수 없으므로이를 처리 할 수있는 몇 가지 기능을 만들어 보겠습니다. 이전에 total\_bedrooms 속성에 누락 된 값이 있다는 것을 알았으므로 이를 수정합시다. 세 가지 옵션이 있습니다.

* Get rid of the corresponding districts.
* Get rid of the whole attribute.
* Set the values to some value (zero, the mean, the median, etc.).

DataFrame의 dropna(), drop(), and fillna() 함수를 사용하여 쉽게 해결

# option 1 housing.dropna( subset =[" total\_bedrooms"])

nan 값들이 있는 데이터 삭제

# option 2 housing.drop(" total\_bedrooms", axis = 1)

전체 속성의 값 삭제(total\_bedfooms 열을 삭제)

# option 3 median = housing[" total\_bedrooms"]. median()

housing[" total\_bedrooms"]. fillna( median, inplace = True)

nan 값들을 특정 선택하는 데이터로 채움(트레이닝 셋의 중간 값을 계산해줘야되며 트레이닝 셋의 빈 칸에 중간값으로 채워줘야된다)

sklearn의 Imputer이라는 클래스로 누락된 값을 처리

Imputer inst 생성하고 각 속성의 누락된 값을 해당 속성의 중앙값으로 대체



앞 2.2)에서 언급했던 ocean\_aproximity가 텍스트 속성 이기에 숫자 속성으로 변환 후 fit() 함수로 imputer inst 를 트레이닝 데이터에 맞출 수 있다.



imputer는 단순히 각 속성의 중앙값을 계산하고 그 결과를 해당 statistics\_ inst 변수에 저장했습니다. total\_bedrooms 속성만 누락 된 값을 갖지만 시스템이 실행 된 후에는 새 데이터에 누락 된 값이 없다는 것을 확신 할 수 없기 때문에 모든 수치 속성에 imputer를 적용하는 것이 더 안전합니다.

훈련된 imputer로 누락된 값들을 학습된 중앙값으로 트레이닝 셋을 변환할 수 있다

결과물 X는 변환된 feature들을 포함한 Numpy 배열이고

pandas DataFrame으로 다시 되돌리고 싶다면 다음과 같다



4.2)Handling text and categorical attributes

이전에 카테고리 속성 ocean\_proximity는 텍스트 속성 이므로 중간 값을 계산할 수 없었다.

대부분의 기계학습 알고리즘은 숫자로 처리하는 것을 선호하므로 카테고리를 텍스트에서 숫자로 변환



이를 위해 각 범주를 다른 정수로 매핑하는 Pandas의 factorize() 메서드를 사용할 수 있습니다

이 표현의 한 가지 문제는 ML 알고리즘이 인접한 두 개의 값이 두 개의 먼 값보다 더 유사하다고 가정합니다.



이 문제를 해결하기 위해 바이너리 속성을 부여한다

한 카테고리가 1일 때 나머지 속성은 0 다른 한 카테고리가 1일때 나머지는 0을 주는 방식

<1H OCEAN, INLAND, NEAR OCEAN, NEAR BAY, ISLAND 라는 5개의 카테고리에서 <1H OCEAN 이 1이면 10000, NEAR OCEAN이 1이면 00100 이런 식으로

Scikit-Learn은 정수 카테고리 값을 one-hot 벡터로 변환하는 OneHotEncoder를 제공한다. fit\_transform ()은 2차원 배열을 필요로하지만, housing\_cat\_encoded는 1차원 배열이므로 모양을 바꿔야합니다.



또한 출력은 NumPy 배열 대신 SciPy sparse 행렬입니다. 카테고리가 수천 가지인 경우 매우 유용하게 적용된다. 행렬은 하나의 1인 행을 제외하고는 0으로 채워집니다. 대부분 0을 저장하기 위해 많은 양의 메모리를 사용하는 것은 매우 낭비 일 것입니다. 따라서 sparse 행렬는 0이 아닌 요소의 위치만 저장합니다.

조밀 한 NumPy 배열로 변환하려면, toarray () 함수를 사용



CategoricalEncoder 클래스를 사용하여 한 번에 두 가지 변형 (텍스트 범주에서 정수 범주로, 그런 다음 정수 범주에서 단일 핫 벡터로)을 적용 할 수 있습니다

책에는 LabelBinarizer 함수인데 바뀜



**4.3) Custom transformers**

Scikit-Learn은 많은 유용한 transtormers를 제공하지만 사용자 정의 정리 작업이나 특정 속성 결합과 같은 작업을 위해 직접 작성해야합니다.

Scikit-Learn은 클래스를 생성하고 3개의 함수만 구현(fit(), transform(), fit\_transform())

BaseEstimator를 기본 클래스로 추가하면 자동 하이퍼 매개 변수 조정에 유용한 두 개의 추가 함수 (get\_params (), set\_params ())가 제공

트랜스포머는 하나의 하이퍼 매개 변수 add\_bedrooms\_per\_room을 기본적으로 True로 설정합니다

보다 일반적으로 하이퍼 매개 변수를 추가하여 100 % 확신 할 수 없는 모든 데이터 준비 단계를 게이트 할 수 있습니다. 이러한 데이터 준비 단계를 자동화하면 할수록 더 많은 조합을 자동으로 시도 할 수 있으므로 훌륭한 조합을 찾을 확률이 높아지고 많은 시간을 절약할 수 있다



**4.4) Feature scaling**

데이터에 적용해야하는 가장 중요한 변환 중 하나는 feature scaling입니다. 소수의 예외를 제외하고는 입력 숫자 속성이 매우 다른 스케일의 경우 기계 학습 알고리즘이 제대로 수행되지 않습니다.

모든 속성의 스케일을 동일하게 유지하는 두 가지 일반적인 방법이 있습니다. 최소 최대 스케일링과 표준화입니다.

* Min-max scaling(많은 사람들은 이를 정규화라고 부름)은 매우 간단합니다. 값은 0에서 1까지의 범위에서 끝나도록 이동 및 재조정됩니다. 최소값을 빼고 최대 마이너스 값으로 나눔으로써 값을 구합니다.
* Standardization(표준화)은 완전히 다릅니다. 먼저 평균 값을 뺍니다 (표준화 된 값은 항상 평균이 0입니다). 그런 다음 분산으로 나눕니다.  최소 최대 확장과 달리 표준화는 값을 특정 범위로 한정하지 않으며 일부 알고리즘에서는 문제가 될 수 있습니다(예로 neural network의 경우 input을 0에서 1까지로 한정) 그러나 표준화는 이상 치의 영향을 훨씬 덜 받습니다.(예를 들어, 실수로 한 구역에 중간 소득이 100 이라 하면 최소 최대 스케일링은 0-15에서 다른 모든 값을 0-1.15로 낮추지만 표준화는 그다지 영향을 받지 않습니다.)

Scikit-Learn은 표준화를 위해 StandardScaler를 제공합니다.

4.5)Transformation pipelines

보시다시피 올바른 순서로 실행해야하는 많은 데이터 변환 단계가 있습니다. 다행히도, Scikit-Learn는 이러한 일련의 변환을 돕기 위해 Pipeline 클래스를 제공합니다



파이프 라인 생성자는 일련의 단계를 정의하는 ('이름' / estimator) 쌍인 리스트를 사용합니다. 마지막 estimator를 제외한 모든 것은 변압기(transformers)여야 합니다(즉, fit\_transform () 함수가 있어야 함)

이름은 원하는대로 지정할 수 있습니다 (두 개의 밑줄 "\_\_"을 포함하지 않는 한)

파이프 라인의 fit () 함수를 호출하면 모든 호출기의 출력 값을 매개 변수로 전달하여 최종 호출기에 도달 할 때까지 모든 호출기의 fit\_transform ()함수를 순차적으로 호출하여 다음 호출에 전달합니다.

파이프 라인은 최종 견적 도구와 동일한 방법을 제공합니다. 이 예에서 마지막 estimator는 변환기인 StandardScaler이므로 파이프 라인에는 모든 변환을 데이터에 순서대로 적용하는 transform () 함수가 있습니다 (fit () 및 transform ()을 호출하는 대신 사용할 수 있는 fit\_transform 함수도 가지고 있습니다). ).

NumPy 배열로 숫자 열을 먼저 수동으로 추출하는 대신 숫자가 아닌 열을 포함하는 pandas 데이터 프레임을 직접 파이프 라인에 공급할 수 있다면 좋을 것입니다.(Scikit-Learn에는 Pandas DataFrames를 다루는 것이 없지만 이 작업을 위해 사용자 정의 변환기를 작성할 수 있습니다.)



DataFrameSelector는 원하는 속성을 선택하고, 나머지를 삭제하고, 결과 DataFrame을 NumPy 배열로 변환하여 데이터를 변환합니다.(one-hot 인코딩) 이를 통해 Pandas DataFrame을 사용하고 숫자 값만 처리하는 파이프 라인을 쉽게 작성할 수 있습니다. 파이프 라인은 수치 속성 만 선택하는 DataFrameSelector로 시작하고 앞에서 설명한 다른 사전 처리 단계를 따라야합니다.

또한 DataFrameSelector를 사용하여 범주 속성을 선택한 다음 CategoricalEncoder를 적용하여 범주 형 특성에 대한 다른 파이프 라인을 쉽게 작성할 수 있습니다.



그러나 이 두 파이프 라인을 어떻게 하나의 파이프 라인에 결합시킬 수 있습니까? 대답은 Scikit-Learn의 FeatureUnion 클래스를 사용하는 것입니다. 당신은 변압기 목록을 제공합니다



1. **Select and train a model**

문제를 골라 내고 데이터를 얻은 다음 그것을 탐색하고 훈련 세트와 테스트 세트를 샘플링 한 다음 변환 파이프 라인을 작성하여 기계 학습 알고리즘에 대한 데이터를 자동으로 정리하고 준비합니다.

이제 기계 학습 모델을 선택하고 교육 할 준비가 되었습니다.

**5.1) Training and Evaluating on the Training Set**

좋은 소식은 이러한 모든 이전 단계 덕분에 이제는 생각하는 것보다 훨씬 더 간단해질 것입니다. 이전 장에서 했던것처럼 Linear Regression 모델을 먼저 훈련한다.



이제 선형 선형 회귀 모델을 사용할 수 있습니다.



예상치가 정확하지는 않지만 (예 : 첫 번째 예상치가 40 % 가까이 떨어져 있음) 작동합니다. Scikit-Learn의 mean\_squared\_error 함수를 사용하여 전체 훈련 세트에서이 회귀 모델의 RMSE를 측정합시다.



대부분의 지역의 median\_housing\_values는 $ 120,000에서 $ 265,000 사이이므로 $ 68,628의 전형적인 예측 오차는 그다지 만족스럽지 않습니다. 이것은 훈련 데이터에 부합하지 않는 모델의 예입니다. 이것이 발생하면 기능이 좋은 예측을하기에 충분한 정보를 제공하지 못하거나 모델이 충분히 강력하지 않다는 것을 의미 할 수 있습니다.

DecisionTreeRegressor를 훈련 시키자. 이것은 강력한 모델로서 데이터에서 복잡한 비선형 관계를 찾을 수 있습니다



모델이 훈련되었으므로 훈련 세트에서 모델을 평가 해 봅시다.



0이 나왔다. 잘못 되었다.

물론 모델이 데이터에 지나치게 과장되어있을 가능성이 훨씬 큽니다. 앞에서 살펴 보았듯이 확신 할 수 있는 모델을 시작할 준비가 될 때까지 테스트 셋를 만지는 것을 원하지 않는다.

트레이닝 셋의 일부를 사용하고 그 중 모델 검증용으로 사용해야 합니다.(저번에 말한 Validation set)

**5.2) Better evaluation using cross-validation**

Decision Tree mode을 평가하는 한 가지 방법은 train\_test\_split 함수를 사용하여 훈련 세트를 더 작은 훈련 세트와 유효성 검증 세트로 분할 한 다음, 더 작은 훈련 세트에 대해 모델을 훈련시키고 유효성 검증 세트에 대해 평가하는 것입니다.

훌륭한 대안은 Scikit-Learn의 K-fold cross-validation을 사용하는 것입니다 훈련 세트를 무작위로 폴드라는 10 개의 하위 세트로 분할 한 다음 Decision Tree mode을 10 번 평가하고 평가하고 매번 다른 폴드를 선택하고 다른 폴드를 교육합니다



이제 Decision Tree가 이전처럼 훌륭하게 보이지 않습니다. 실제로 선형 선형 모델보다 성능이 떨어지는 것 같습니다! 교차 유효성 검사를 사용하면 모델 실적 추정치 뿐만 아니라 추정치의 정확성 (즉, 표준 편차)을 측정 할 수 있습니다. 방금 하나의 유효성 검증 세트를 사용한 경우에는 이 정보가 없습니다. 그러나 교차 검증은 모델을 여러 번 교육하는 비용을 필요로 하므로 항상 가능한 것은 아닙니다.

같은 방식으로 선형 회귀 모델에 적용



Decision Tree mode은 선형 회귀 모델보다 성능이 좋지 않을 만큼 너무 잘 맞지 않습니다.

마지막 모델 인 RandomForestRegressor를 사용해 봅시다.



좋은 결과를 얻었지만 아직 과대적합 되어있다.

RandomForestRegressor 기능의 무작위 부분 집합에 대해 많은 의사 결정 트리를 학습 한 다음 예상치를 평균화하여 작동합니다. 다른 많은 모델 위에 모델을 작성하는 것을 Ensemble Learning이라고 하며, 종종 ML 알고리즘을 더욱 발전시키는 좋은 방법입니다.

서로 다른 방향으로 과대적합된 트리를 많이 만들어내면 그 결과를 평균냄으로써 과대적합된 양을 줄일 수 있다는 것이다. 이렇게 하면 트리 모델의 예측 성능은 유지되면서 과대적합이 줄어드는 것이 수학적으로 증명 되었다.

1. **Fine-Tune your model**

이제 유망한 모델 리스트가 있다고 가정 해 봅시다. 이제 미세 조정해야 합니다.

이를 수행하는 한 가지 방법은 하이퍼 매개 변수 값의 훌륭한 조합을 찾을 때까지 수동으로 하이퍼 매개 변수를 사용하는 것입니다

**6.1) Grid search**

Scikit-Learn의 GridSearchCV를 사용하여 검색하십시오. 실험 할 하이퍼 매개 변수와 시도 할 값을 지정하고 교차 유효성 검사를 사용하여 하이퍼 매개 변수 값의 가능한 모든 조합을 평가합니다.

 GridSearchCV가 refit = True (기본값)로 초기화되면 교차 검증을 사용하여 최상의 추정량을 찾으면 전체 교육 세트에서 다시 계산합니다. 더 많은 데이터를 제공하면 성능이 향상 될 것이므로 일반적으로 좋은 아이디어입니다.

일부 데이터 준비 단계를 하이퍼 매개 변수로 처리 할 수 있다는 것을 잊지 마십시오. 예를 들어 그리드 검색은 확실하지 않은 기능을 추가할지 여부를 자동으로 찾습니다. 특이점, 누락 된 기능, 기능 선택 등을 처리하는 가장 좋은 방법을 자동으로 찾기 위해 유사하게 사용될 수 있습니다.

**6.2) Randomized search**

이전 예제에서와 같이 비교적 적은 수의 조합을 탐색 할 때 그리드 검색 방법이 좋지만 하이퍼 매개 변수 검색 공간이 큰 경우 RandomizedSearchCV를 대신 사용하는 것이 좋습니다. 이 클래스는 GridSearchCV 클래스와 거의 같은 방식으로 사용할 수 있지만 가능한 모든 조합을 시도하는 대신 모든 반복에서 각 하이퍼 매개 변수의 임의 값을 선택하여 주어진 임의의 조합 수를 계산합니다

* If you let the randomized search run for, say, 1,000 iterations, this approach will explore 1,000 different values for each hyperparameter (instead of just a few values per hyperparameter with the grid search approach).
* You have more control over the computing budget you want to allocate to hyperparameter search, simply by setting the number of iterations.

**6.3) Ensemble Methods**

시스템을 미세 조정하는 또 다른 방법은 성능이 가장 우수한 모델을 결합하는 것입니다. 그룹 (또는 "앙상블")은 특히 개별 모델이 매우 다른 유형의 오류를 만드는 경우 가장 적합한 개별 모델 (Random Forest가 자신이 사용하는 개별 의사 결정 트리보다 우수한 성능을 발휘하는 것처럼)보다 성능이 우수합니다

**6.4) Analyze the Best Models and Their Errors**

가장 좋은 모델을 검사하여 문제에 대한 좋은 통찰력을 얻을 수 있습니다. 예를 들어, RandomForestRegressor는 정확한 예측을 하기 위해 각 속성의 상대적 중요성을 나타낼 수 있습니다. 이 정보를 사용하면 덜 유용한 기능 중 일부를 삭제할 수 있습니다

또한 시스템에서 발생하는 특정 오류를 살펴본 후 오류의 원인과 해결할 수 있는 문제 (추가 기능 추가 또는 반대로 정보가 없는 오류 제거, 특이치 청소 등)를 시도해야 합니다.

**6.5) Evaluate Your System on the Test Set**

잠시 동안 모델을 조정 한 후에는 결국 충분히 잘 수행되는 시스템을 갖게됩니다. 이제 테스트 세트의 최종 모델을 평가할 시간입니다. 이 과정에는 특별한 것이 없습니다. 테스트 세트에서 예측 변수와 레이블을 가져와서 full\_pipeline을 실행하여 데이터를 변환합니다 (fit\_transform ()이 아닌 transform ()을 호출하십시오!), 테스트 세트의 최종 모델을 평가하십시오.

이제 프로젝트가 시작되는 단계입니다. 솔루션을 제시해야합니다. (배운 것을 강조하고, 성공한 것과 그렇지 않은 것을 가정하고, 가정과 시스템의 한계점을 강조하고) 모든 것을 문서화하고 명확한 시각화로 멋진 프레젠테이션을 만들어야합니다

1. **Launch, Monitor, and Maintain Your System**

프로덕션 입력 데이터 소스를 시스템에 연결하고 테스트를 작성하여 프로덕션을위한 솔루션을 준비해야합니다.

또한 정기적 인 간격으로 시스템의 실시간 성능을 확인하고 모니터가 떨어질 때 경고를 발생시키는 모니터링 코드를 작성해야 합니다

시스템의 성능을 평가하려면 시스템의 예측을 샘플링하고 평가해야 합니다. 이것은 일반적으로 인간의 분석이 필요합니다.

또한 시스템의 입력 데이터 품질을 평가해야 합니다. 품질이 좋지 않은 신호 (예 : 임의의 값을 보내는 오작동 센서 또는 다른 팀의 출력이 오래됨)로 인해 성능이 약간 저하 될 수 있지만 시스템의 성능이 저하되어 경고를 유발할 때까지 시간이 걸릴 수 있습니다.

마지막으로, 일반적으로 신선한 데이터를 사용하여 정기적으로 모델을 교육하고 싶습니다. 가능한 한이 프로세스를 자동화해야 합니다. 그렇지 않은 경우 6 개월 (기껏해야)마다 모델을 새로 고칠 가능성이 높으며 시스템 성능은 시간이 지남에 따라 심각하게 변동될 수 있습니다.