머신러닝에서 하나의 개체 or 행 : 샘플 or 데이터 포인트, 샘플의 속성 or 열 : 특성

Scipy : 과학 계산용 함수를 모아놓은 파이썬 패키지

matplotlib : 파이썬의 대표적 과학 계산용 그래프 라이브러리

pandas : 데이터의 처리와 분석을 위한 파이썬 라이브러리

데이터 배열의 크기 : 샘플의 수 \* 특성의 수

과대적합 : 모델이 훈련세트의 각 샘플에 너무 가깝게 맞춰져서 새로운 데이터에 일반화되기 어려울 때이다. (복잡)

붓꽃 종류 : 클래스, 붓꽃 하나 : 데이터 포인트 하나, 품종 : 레이블

점 하나 : 데이터 포인트, 점의 색과 모양 : 데이터 포인트가 속한 클래스

Chapter2 : 지도학습

지도학습 – 분류 : 가능성 있는 여러 클래스 레이블 중 하나를 예측, 회귀 : 연속적인 숫자를 예측

과대적합 : 너무 복잡한 모델을 만드는 것, 과소적합 : 너무 간단한 모델이 선택

2.3.2 k-최근접 이웃

K-최근접 이웃 : 새로운 데이터 포인트에 대해 예측할 때 훈련 데이터셋에서 가장 가까운 훈련 데이터 포인트를 최근접 이웃으로 찾아 예측으로 사용

데이터 포인트 3개 추가 -> 추가한 각 데이터 포인트에서 가장 가까운 훈련 데이터 포인트 연결

임의의 k개를 선택 시 -> 테스트 포인트 하나에 대해 클래스 0에 속한 이웃이 몇 개 인지, 클래스 1에 속한 이웃이 몇 개 인지 센 후 이웃이 더 많은 클래스를 레이블로 지정

Scikit-learn(사이킷런) -> 훈련 세트와 테스트 세트로 나눈다, KNeighborsClassifier를 import 하고 객체를 만든다. 데이터를 저장(학습) – clf.fit(X\_train, y\_train), predict 메서드를 호출하여 예측, 일반화가 잘 되었는지 score 메서드로 평가

KNeighborsClassifier 분석

각 데이터 포인트가 속한 클래스에 따라 xy평면에 색을 칠한다.

클래스 0, 클래스1로 지정한 영역으로 나뉜다. -> 결정경계

이웃 수를 늘릴수록 결정 경계는 더 부드러워진다 -> 더 단순

유방암 데이터셋 - 훈련세트와 테스트 세트로 나눈다. 이웃의 수를 달리하여 훈련세트, 테스트세트 성능 평가 -> 이웃의 수가 늘어나면 모델은 단순, 정확도 감소

정확도가 가장 좋을 때 -> 이웃의 수가 중간 정도인 여섯 개 사용시

k-최근접 이웃 회귀

wave 데이터셋을 이용해서 이웃이 하나인 최근접 이웃 사용 - x축에 세 개의 테스트 데이터를 흐린 별 모양으로 표시 – 가장 가까운 이웃의 타깃값

이웃을 둘 이상 – 이웃간의 평균이 예측

Scikit-learn -> import KNeightborsRegressor, wave 데이터셋을 훈련 세트와 테스트 세트로 나눈다, 이웃의 수를 3으로 하여 모델의 객체를 만든다, 모델 학습(fit), 예측, 평가(score – 회귀일때 R^2(R제곱) 반환)

KNeighborsRegressor 분석

-3과 3 사이 1000개의 데이터 포인트를 만든다, 1, 3, 9 이웃을 사용한 예측

이웃이 1개 – 훈련 세트의 각 데이터 포인트가 예측에 주는 영향이 커서 예측값이 훈련 데이터 포인트를 모두 지나감. 이웃을 많이 사용할수록 더 안정된 예측을 얻는다.

KNeighbors 분류기에 중요한 매개변수 – 데이터 포인트 사이의 거리를 재는 방법, 이웃 수

k-NN의 장점은 이해하기 쉽다. 단점은 예측이 느리고 많은 특성을 처리하는 능력이 부족

2.3.3 선형 모델

k-NN의 단점이 없는 알고리즘 – 선형모델

선형모델 : 입력 특성에 대한 선형 함수를 만들어 예측을 수행함.

회귀의 선형 모델을 위한 일반화된 예측 함수 w[0]\*x[0]+w[1]\*x[1]+…+w[p]\*x[p]+b

x[0]부터 x[p]까지는 하나의 데이터 포인트에 대한 특성(특성의 개수는 p+1), w와 b는 모델이 학습할 파라미터, y는 모델이 만들어낸 예측값, 직선의 방정식과 유사

회귀를 위한 선형 모델은 특성 1개 – 직선, 특성 2개 – 평면, 그 이상 – 초평면

선형회귀(최소제곱법) : 예측과 훈련 세트에 있는 타깃 y사이의 평균제곱오차를 최소화하는 파라미터 w와 b를 찾는다. 평균제곱오차 -> 예측값과 타깃값의 차이를 제곱하여 더한 후 샘플의 개수로 나눈 것

Import LinearRegreesion, 기울기 파라미터(w)는 lr객체의 coef\_속성에 저장, 절편 파라미터(b)는 intercept\_속성에 저장

R제곱이 0.66 훈련세트 & 테스트 세트 점수가 매우 비슷 – 과소적합

보스턴 주택가격 데이터 셋인 경우 – 샘풀 : 506개, 특성 : 105개

결과 -> 훈련세트는 예측이 정확, 테스트 세트 R제곱값이 매우 낮다. – 복잡도를 제어하기 어렵

훈련 데이터와 테스트 데이터 사이의 성능 차이 -> 과대적합

릿지 회귀 : 가중치(w)의 값을 가능한 작게 만들기.(w의 모든 원소가 0에 가깝게 만들기-기울기 작게) ->이러한 제약을 규제라고 한다.

규제란 과대적합(복잡, 모델이 훈련 세트의 각 샘플에 너무 가깝게 맞춰져서 새로운 데이터에 일반화되기 어려울 때) 이 되지 않도록 모델을 강제로 제한하는 것이다.

릿지 회귀에서 사용하는 규제 : L2규제

보스턴 주택가격 데이터 셋을 릿지 회귀에 적용시키면 선형 회귀보다 훈련 세트 점수는 더 낮지만 테스트 세트 점수는 더 높다. (선형 모델일 경우는 과대적합이지만, 릿지 회귀를 적용시키면 과대적합이 적어진다.) 모델의 복잡도가 낮으면 더 일반화된 모델

Alpha 매개변수로 훈련 세트의 성능 대비 모델을 얼마나 단순화할지 지정 가능

alpha값을 높이면 계수를 0에 가깝게 만들고 일반화에 도움을 준다.

높은 alpha 값은 coef\_(coefficient 계수)의 절대값 크기(기울기)가 작다.

릿지, 선형 회귀 모두 훈련 세트의 점수가 테스트 세트 점수보다 높다.

테스트 데이터에서는 릿지의 점수가 높다. 두 모델의 성능은 데이터가 많아질수록 좋아진다.

* 데이터를 충분히 주면 규제 항은 덜 중요해져 릿지 회귀와 선형 회귀의 성능은 같아진다.

라쏘 : 릿지 회귀와 같이 계수를 0에 가깝게 만든다. 라쏘는 L1 규제

L1 규제의 결과로 모델에서 완전히 제외되는 특성(어떤 계수는 정말로 0이 된다.)이 생긴다.

라쏘 – 훈련세트, 테스트 세트 모두 결과가 안좋다. (과소적합)

과소적합을 줄이기 위해 alpha 값을 줄이고, max\_iter(반복 실행하는 최대 횟수)의 기본값을 늘린다.

alpha값을 낮추면 복잡도 증가, 성능이 좋아진다.

특성이 많고 일부분만 중요, 분석 쉽다. – 라쏘

분류용 선형 모델 : 예측한 값을 임계치 0 과 비교 / 계산한 값이 0보다 작으면 클래스 : -1, 계산한 값이 0보다 크면 클래스 : +1

결정 경계가 입력의 선형 함수이다.(이진 선형 분류기는 선, 평면, 초평면을 사용해서 두 개의 클래스를 구분하는 분류기)

두개의 선형 분류 알고리즘 – linear\_model.LogisticRegression에 구현된 로지스틱 회귀, svm.LinearSVC에 구현된 선형 서포트 벡터 머신이다.

결정 경계가 직선으로 표현되고, 위쪽은 클래스1, 아래쪽은 클래스 0으로 나뉜다.

두 모델은 L2규제 사용

LogisticRegression, LinearSVC에서 규제 강도를 결정하는 매개변수 : C

C의 값 높 – 규제 감소(훈련 세트에 가능한 최대로 맞추려 함) (개개의 데이터 포인트를 정확히 분류하려고 노력함.)

C의 값 낮 – 계수 벡터(w)가 0에 가까워진다. (데이터 포인트 중 다수에 맞추려 함.)

C값이 커질수록 결정 경계가 기울면서 거의 모든 데이터 포인트를 올바르게 분류함.

유방암 데이터 셋을 이용해 LogisticRegression을 분석 – 95%의 정확도

C=100 으로 증가 – 훈련 세트, 테스트 세트 점수 정확도 증가

C=0.01로 감소 – 훈련 세트와 테스트 세트의 정확도는 낮아짐.

L2규제 모델 사용, 규제를 강하게 할수록 계수들을 0에 더 가깝게 만든다.

L1규제 : 일부 특성만 사용, 더 이해하기 쉽다.

L2규제 : 전체 특성을 모두 사용

다중 클래스 분류용 선형 모델

많은 선형 모델은 이진 분류만을 지원한다. – 다중 클래스를 지원하지 않는다.

이진 분류 알고리즘을 다중 클래스 분류 알고리즘으로 확장하는 방법 : 일대다 방법

일대다 방식은 각 클래스를 다른 모든 클래스와 구분하도록 이진 분류 모델을 학습시킨다.

결국 각 클래스의 수만큼 이진 분류 모델이 만들어짐.

예측 시, 모든 이진 분류기가 작동하여 가장 높은 점수를 내는 분류기의 클래스를 예측값으로 선택. 클래스별 이진 분류기를 만들면 각 클래스가 계수 벡터(w)와 절편(b)을 하나씩 갖게 된다.

w[0]\*x[0]+w[1]\*x[1]+…+w[p]\*x[p]+b 공식의 결과값이 가장 높은 클래스가 해당 데이터의 클래스 레이블로 할당된다.

세 개의 클래스를 가진 간단한 데이터셋에 일대다 방식 적용 - 그림 2-19 세 개의 클래스를 가진 2차원 데이터셋

coef\_배열의 크기는 (3, 2)이다. Coef\_의 행은 세 개의 클래스에 각각 대응하는 계수 벡터를 담고 있고, 열은 각 특성에 따른 계수 값을 가지고 있다.

Intercept\_는 각 클래스의 절편을 담은 1차원 벡터이다.

훈련 데이터의 클래스 0에 속한 모든 포인트는 클래스 0지역에, 클래스 0에 속한 포인트는 클래스 2를 구분하는 직선 위, 클래스 1을 구분하는 직선 왼쪽에 있다.

이 영역의 최종 분류기는 클래스 0으로 분류한다.

중앙의 삼각형 영역의 데이터 포인트는 분류 공식의 결과가 가장 높은 클래스로 분류된다.

즉, 가장 가까운 직선의 클래스가 된다.

alpha값이 클수록 C값이 작을수록 모델이 단순해진다.

로그 스케일로 C와 alpha를 정하고, L1규제를 사용할지 L2 규제를 사용할지 정한다.

선형 모델의 장점은 학습 속도가 빠르고 예측도 빠르다, 쉽게 이해가 가능하다.

선형 모델은 샘플에 비해 특성이 많을 때 잘 작동한다.

2.3.4 나이브 베이즈 분류기(분류만 가능, 선형 모델보다 훨씬 빠름)

선형 분류기보다 훈련 속도는 빠르지만, 대신 일반화 성능이 안좋다.

각 특성을 개별로 취급해 파라미터를 학습하고 각 특성에서 클래스별 통계를 단순하게 취합한다.

GaussianNB : 연속적인 어떤 데이터에 적용 가능, 클래스 별로 각 특성의 표준편차와 평균을 저장

BernoulliNB : 이진 데이터, MultinomialNB : 클래스 별로 특성의 평균 계산, 카운트 데이터에 적용 => 대부분 텍스트 데이터 분류 시 사용, alpha 매개변수o

Alpha가 크면 더 완만, 복잡도 낮

GaussianNB – 매우 고차원인 데이터셋에 사용, 다른 두 모델은 텍스트 같은 희소한 데이터에 사용

나이브 베이즈 모델의 장점 : 훈련과 예측 속도가 빠르며 훈련 과정을 이해하기 쉽다.

선형모델로 학습 시간이 오래 걸리는 매우 큰 데이터셋을 나이브 베이즈 모델을 사용해서 시도해볼만하다.

2.3.5 결정 트리 : 분류와 회귀 문제에 널리 사용 / 예 아니오 질문을 이어가며 학습

결정 트리를 학습한다 : 정답에 가장 빨리 도달하는 예/아니오 질문 목록을 학습

질문 -> 테스트(테스트 시, 사용하는 데이터)

트리를 만들 때 가능한 모든 테스트에서 타깃값에 대해 가장 많은 정보를 가진 것으로 고른다.

데이터를 분할 하는 것은 각 분할된 영역이 한 개의 타깃값을 가질때까지 반복

타깃 하나로만 이뤄진 리프 노드 – 순수노드

훈련 세트의 모든 데이터 포인트는 정확한 클래스의 리프 노드에 있다.

과대적합을 막는 방법 1) 트리 생성을 일찍 중단(사전 가지치기)

2) 트리를 만든 후 데이터 포인트가 적은 노드를 삭제하거나 병합하는 전략(사후 가지치기)

사전 가지치기 -> 트리의 최대 깊이나 리프의 최대 개수를 제한하거나 노드가 분할하기 위한 포인트의 최소 개수를 지정

Scikit-learn -> DecisionTreeRegressor, DesicionTreeClassifier 사전 가지치기만 지원

사전 가지치기로 max\_depth = 4라고 옵션을 주면 연속된 질문을 최대 4개로 제한 -> 과대적합 감소, 훈련 세트의 정확도는 감소하지만 테스트 세트의 성능 개선

결정 트리 분석

Export\_graphviz 함수로 트리 시각화

Samples – 각 노드에 있는 샘플의 수, value – 클래스 당 샘플의 수

대부분의 양성 샘플은 왼쪽에서 두번째 노드에 할당

특성 중요도 : 트리를 만드는 결정에 각 특성이 얼마나 중요한지를 평가

0~1사이 숫자

특성 중요도는 항상 양수

회귀 결정 트리 ->DecisionTreeRegressor은 외삽 – 훈련 데이터의 범위 밖의 포인트에 대해 예측할 수 없다.

선형 모델 – 테스트 데이터를 꽤 정확하게 예측

트리 모델 – 훈련 데이터를 완벽하게 예측- 트리 복잡도에 제한을 두지 않아 전체 데이터 셋을 기억

모든 트리 기반 모델들은 모델이 가진 데이터 범위 밖으로 나가면 단순히 마지막 포인트를 이용해 예측(밖의 새로운 데이터 예측x)

결정 트리 장점 - 만들어진 모델을 쉽게 시각화할 수 있어 비전문가도 이해하기 쉽다. 데이터 스케일에 구애x, 전처리 과정 필요x, 특성의 스케일이 서로 다르거나 특성이 혼합되어 있는 경우도 잘 작동

단점 – 일반화 성능이 좋지x

2.3.6 결정 트리의 앙상블

앙상블 – 여러 머신러닝 모델을 연결하여 더 강력한 모델을 만들기

랜덤 포레스트 – 훈련 데이터에 과대적합되는 경향을 회피o, 조금씩 다른 여러 결정 트리의 묶음

각 트리는 비교적 예측을 잘할 수 있지만, 데이터의 일부에 과대적합하는 경향을 가진다.

서로 다른 방향으로 과대적합된 트리를 많이 만들면 그 결과를 평균내서 과대적합의 양을 줄일 수 있다.

랜덤 포레스트는 트리를 랜덤하게 만들기

방법 1) 트리를 만들 때 사용하는 데이터 포인트를 무작위로 선택하는 방법

2) 분할 테스트에서 특성을 무작위로 선택하는 방법

랜덤 포레스트 구축

생성할 트리의 개수를 정하기, 부스트스트랩 샘플을 생성 n\_samples개의 데이터 포인트 중에서 무작위 데이터를 n\_samples 횟수만큼 반복 추출

데이터 셋으로 결정 트리 만들기

알고리즘이 각 노드에서 후보 특성을 무작위로 선택한 후 이 후보들 중에서 최선의 테스트를 찾는다. , 특성의 개수 max\_features 매개변수로 조정

Max\_features 값을 크게 하면 랜덤 포레스트의 트리들은 매우 비슷, 데이터에 잘 맞춰진다.

낮게 하면 랜덤 포레스트 트리는 많이 달라지고 각 트리는 데이터에 맞추기 위해 깊이가 깊어진다.

회귀 ) 예측 시 먼저 알고리즘이 모델에 있는 모든 트리의 예측을 만든다.

분류 ) 알고리즘이 가능성 있는 출력 레이블의 확률을 제공함으로써 간접적인 예측, 예측한 확률을 평균 내어 가장 높은 확률을 가진 클래스가 예측값 estimator\_속성에 트리 저장

랜덤 포레스트는 선형 모델이나 단일 결정트리보다 높은 97% 정확도

장점 ) 성능이 매우 뛰어나고 매개변수 튜닝을 많이 하지 않아도 잘 작동, 데이터 스케일을 맞출 필요x, 간소하게 표현해야 하면 단일 트리 사용o

랜덤 포레스트 트리가 많 -> random\_state 값의 변화에 따른 변동이 적

랜덤 포레스트는 희소한 데이터에는 작동x

단점 ) 선형 모델보다 많은 메모리 사용시 훈련과 예측이 느리다

중요매개변수 -> n\_estimators(클수록 좋다), max\_features(각 트리가 얼마나 무작위가 될지 결정)

분류 ->max\_features=sqrt(n\_features), 회귀 -> max\_featrues = n\_features

그래디언트 부스팅 회귀 트리- 여러 개의 결정 트리를 묶어 강력한 모델을 만드는 또 다른 앙상블 방법 / 회귀, 분류 모두 사용 o

이전 트리의 오차를 보완하는 방식으로 순차적으로 트리를 만든다. , 무작위성x, 강력한 사전 가지치기o, 메모리 적게 사용, 예측 빠

간단한 모델을 많이 연결, 트리가 많이 추가될수록 성능이 좋

Learning\_rate : 이전 트리의 오차를 얼마나 강하게 보정할 것인지 제어

학습률이 크면 트리는 보정을 강하게 복잡한 모델, n\_estimators값을 키우면 앙상블에 트리가 많이 추가, 복잡도 커짐.

트리의 최대깊이를 낮추는 것- 모델 성능 향상에 기여

예측 시간이 중요하거나 마지막 성능까지 쥐어짤 때 랜덤 포레스트 보단 그래디언트 부스팅

단점 ) 매개변수를 잘 조정해야한다. 훈련시간이 길다.

장점 ) 특성의 스케일을 조정하지 않아도 되고 이진 특성이 연속적인 특성에서도 잘 동작

Learing\_rate를 낮 – 더 많은 트리 추가, n\_estimators(트리 개수 지정) 클 – 모델이 복잡, 과대적합 가능성o

가용한 시간과 메모리 한도에서 n\_estimators를 맞추고 적절한 learning\_rate 찾는것이 중요

Max\_depth (트리의 복잡도 낮추기) – 매우 작게 설정

2.3.7 커널 서포트 벡터 머신(SVM) : 입력 데이터에서 단순한 초평면으로 정의되지 않는 더 복잡한 모델을 만들 수 있도록 확장

저차원 데이터셋-선형 모델이 제한적 / 선형 모델을 유연하게 만드는 방법 : 특성끼리 곱 or 특성을 거듭제곱

X\_new = np.hstack([X, X[:, 1:] \*\*2]) 3차원 데이터 포인트로 표현

커널 기법 : 수학적 기교를 사용해서 새로운 특성을 많이 만들지 않고 고차원에서 분류기를 학습

* 실제로 데이터를 확장하기 않고 확장된 특성에 대한 데이터 포인트들의 거리를 계산

데이터를 고차원 공간에 매핑하는데 많이 사용하는 방법 1) 원래 특성의 가능한 조합을 지정된 차수까지 모두 계산 – 다항식 커널, 2) RBF 커널 (가우시안 커널) : 차원이 무한한 특성 공간에 매핑하는 것

훈련 데이터의 일부만 결정 경계를 만드는 데 영향을 준다.

두 클래스 사이의 경계에 위치한 데이터 포인트인데 데이터 포인트 -> 서포트 벡터라 한다.

gamma 매개변수 : 가우시안 커널 폭의 역수, 하나의 훈련 샘플이 미치는 영향의 범위를 지정

작은 값은 넓은 영역, 카우시안 커널의 반경이 클수록 훈련 샘플의 영향 범위도 커진다.

C매개변수 : 각 포인트의 중요도를 제한

Gamma 값이 작으면 모델의 복잡도 낮춘다.

C값이 커질수록 포인트들이 모델이 큰 영향, 정확히 분류

각 특성의 최소값과 최대값을 로그 스케일로 나타내면 데이터셋의 특성은 자릿수 자체가 완전히 다르다.- 해결방법 : 특성 값의 범위가 비슷해지도록 조정하는 것 (데이터 스케일 조정),

C나 gamma값 증가

SVM의 장점 : 데이터 특성이 몇 개 안되더라도 복잡한 결정 경계 만들 수 o

단점 : 샘플이 많을땐 안맞을수도.., 데이터 전처리와 매개변수 설정에 신경을 많이 써야한다.

2.3.8 신경망(딥러닝)

입력 특성과 예측은 노드로, 계수는 노드 사이 연결로 나타낸 로지스틱 회귀

은닉 유닛을 100->10개로 줄이면 결정 경계가 날카로움

매끄러운 결정 경계 ->은닉층 추가 or tanh함수

은닉 유닛, alpha 매개변수 / 무작위로 다른 초기값을 주되 같은 매개변수로 학습한 결정 경계

모든 은닉 유닛에서 작은 가중치를 가진 특성은 모델에 덜 중요하다고 추론 가능

장점 : 대량의 데이터에 내재된 정보를 잡아내고 매우 복잡한 모델을 만들 수 o

단점 : 종종 학습이 오래걸림. 데이터 전처리에 주의해야함.

신경망에서 가장 중요한 매개변수 : 은닉층의 개수, 은닉층의 유닛 수

층의 개수, 층당 유닛 개수, 규제, 비선형성을 사용해 모델 정의

Solver 매개변수를 사용해 학습 – 옵션 1) adam, 2)lbfgs(안정적이지만 규모가 큰 모델이나 대량의 데이터 셋에서 시간이 오래 걸린다.) 3)sgd

2.4 불확실성 추정

결정 함수 - decision\_function

반환값의 크기 : n\_samples, 각 샘플이 하나의 실수 반환

예측 확률 – predict\_proba

값의 크기는 이진 분류에서 항상(n\_samples, 2)

불확실성과 모델의 정확도가 동등하면 보정되었다고 한다.

Predict\_proba, decision\_function의 결과값 크기는 항상 (n\_samples, n\_classes)

과소적합 : 훈련 데이터의 다양한 특징을 반영하지 못하는 모델

과대적합 : 훈련 데이터에 너무 맞춰져 있어 새로운 데이터에 일반적이지 못한 모델

요약 및 정리

최근접 이웃 : 작은 데이터셋일 경우, 기본 모델로서 좋고, 설명하기 쉬움.

선형 모델 : 대용량 데이터셋 가능, 고차원 데이터에 가능

나이브 베이즈 : 분류만 가능, 선형 모델보다 빠름, 대용량 데이터셋과 고차원 데이터에 가능, 선형모델보다 덜 정확

결정 트리 : 매우빠름. 데이터 스케일 조정 필요 없음. 시각화하기 좋고 설명하기 쉬움.

랜덤 포레스트 : 결정 트리 하나보다 거의 항상 좋은 성능을 냄. 매우 안정적, 강력함. 데이터 스케일 조정 필요 없음. 고차원 최소 데이터에 잘 안맞음.

그래디언트 부스팅 결정 트리 : 랜덤 포레스트보다 조금 더 성능 좋음. 랜덤 포레스트보다 학습은 느리나 예측 빠름. 메모리 조금 사용. 랜덤 포레스트보다 매개변수 튜닝 많이 필요.

서포트 벡터 머신 : 비슷한 의미의 특성으로 이뤄진 중간 규모 데이터셋에 잘 맞음. 데이터 스케일 조정 필요. 매개변수 민감

신경망 : 대용량 데이터셋에서 복잡한 모델 만들기 가능. 매개변수 선택과 데이터 스케일에 민감. 큰 모델은 학습이 오래걸림.