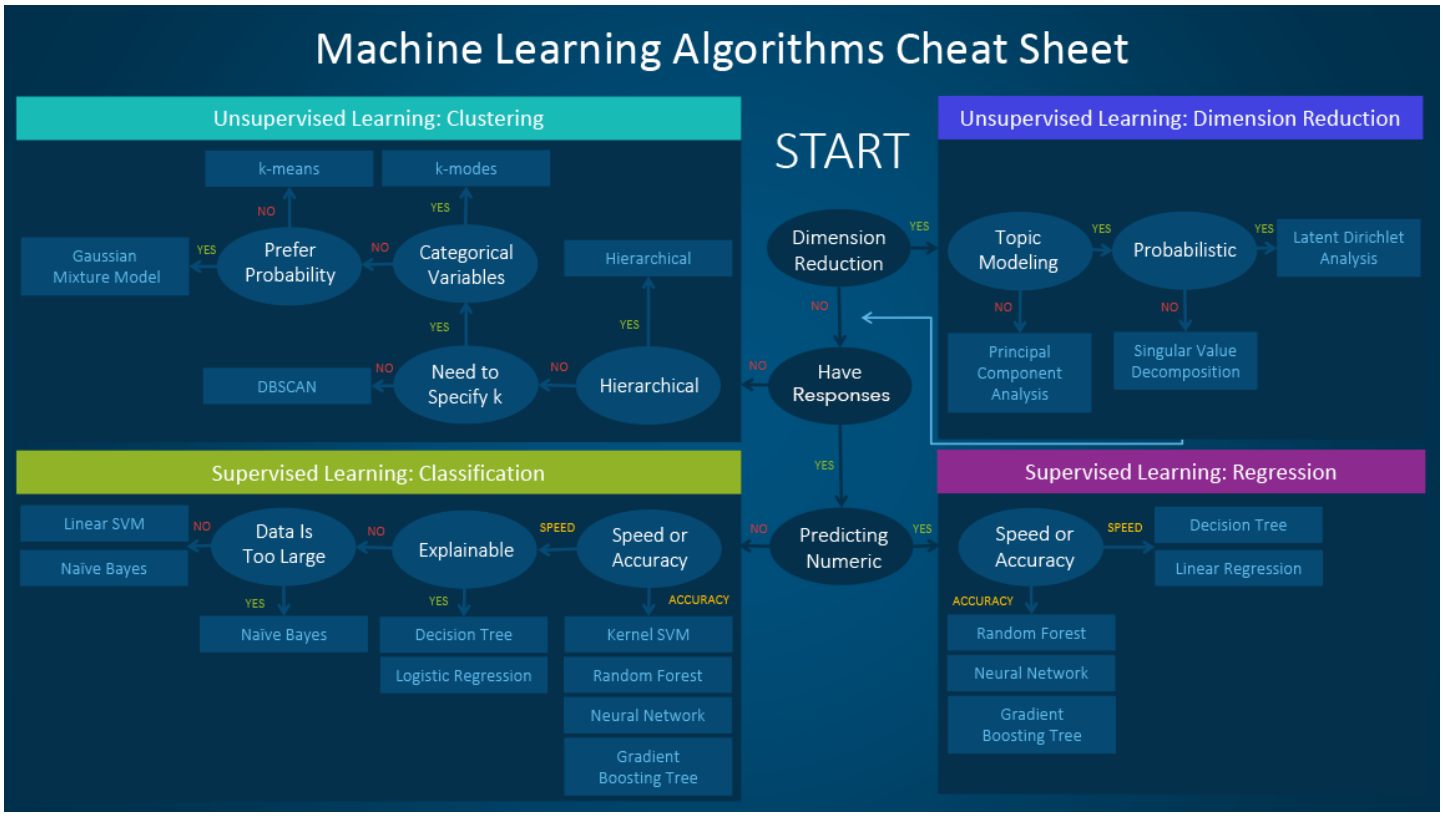
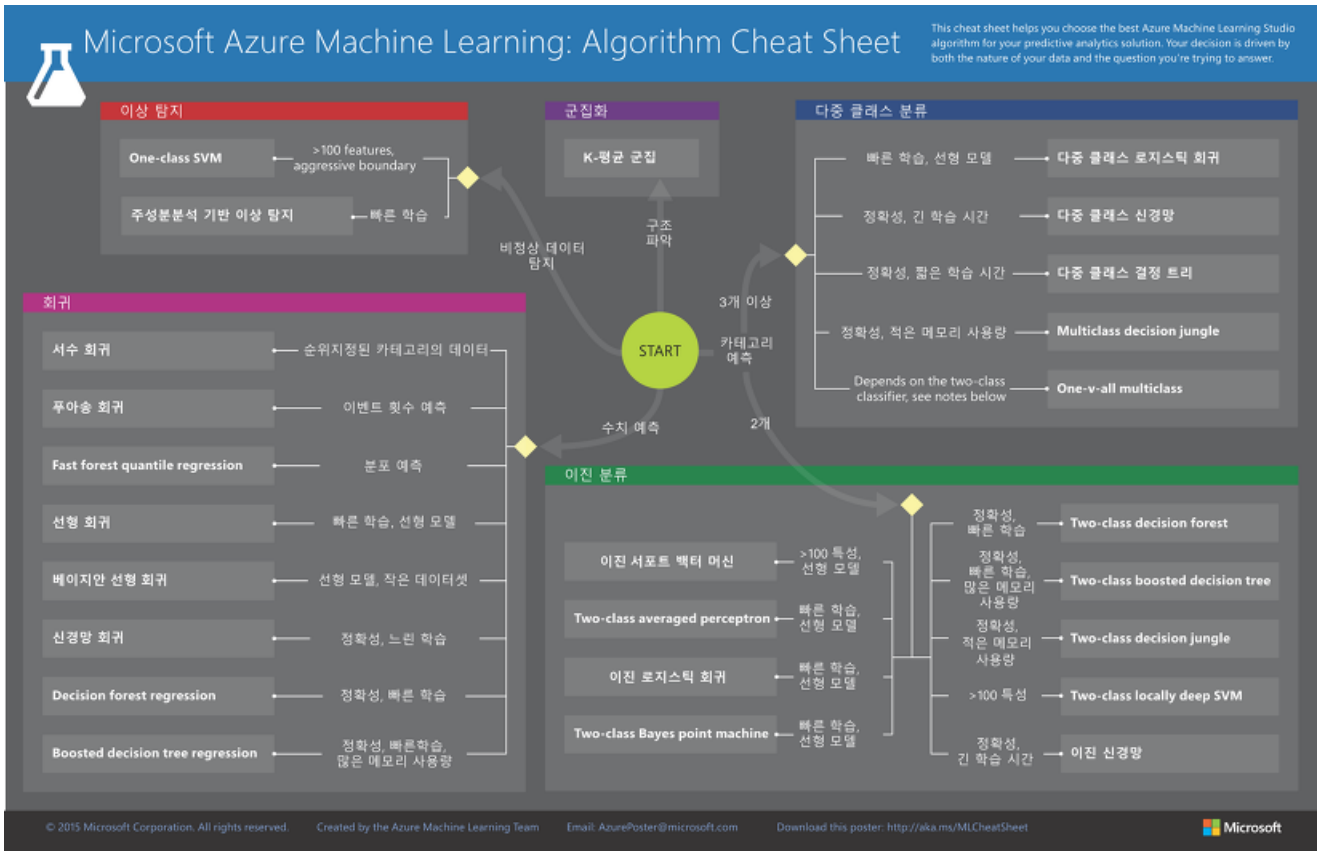
머신러닝 Algorithm Cheat Sheet



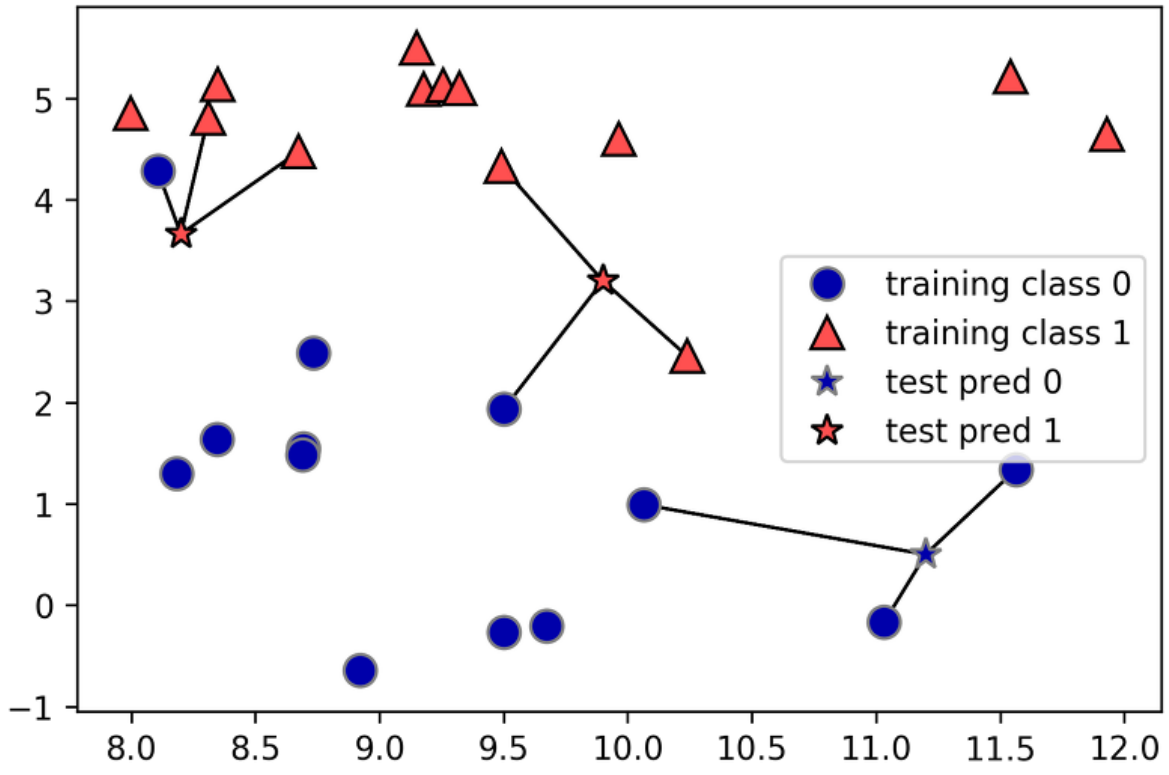


지도 학습 알고리즘

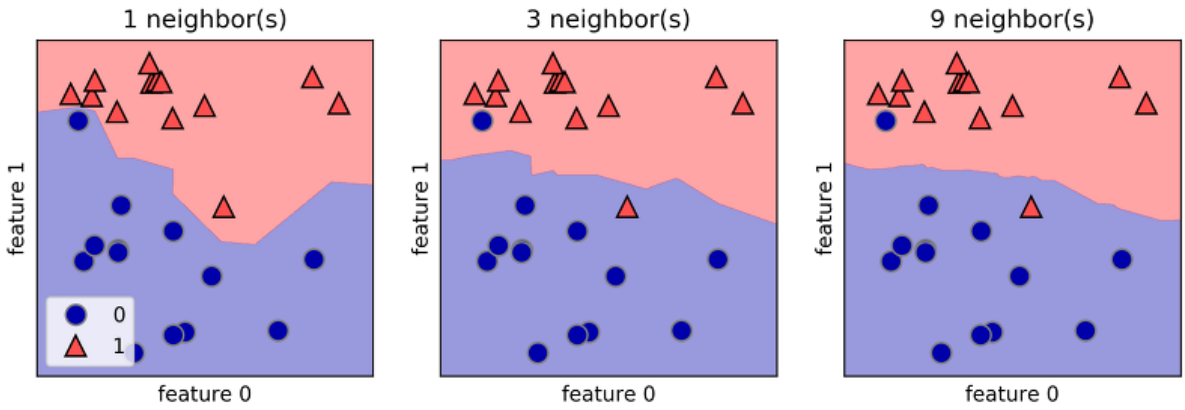
k-최근접 이웃(k-Nearest Neighbors) 분류 알고리즘

- 가장 간단한 머신러닝 알고리즘으로써, 훈련 데이터 세트를 저장하는 것이 모델을 만드는 과정의 전부
- 새로운 데이터 포인트가 주어지면 알고리즘이 훈련 데이터 세트에서 가장 가까운 데이터 포인트, ‘최근접 이웃'을 찾는다. 이 때 k는 찾고자 하는 최근접 이웃의 개수를 뜻한다.
- k-최근접 이웃 분류
    - 알고리즘이 가장 간단하도록 k가 1인 경우, 가장 가까운 훈련 데이터 포인트 하나를 최근접 이웃으로 찾아 출력 예측에 사용한다.
    - k가 2 이상일 때는 더 많은 이웃을 가지고 있는 클래스가 출력된다.

    - k가 3일때의 k-NN 분류 알고리즘 적용 모델

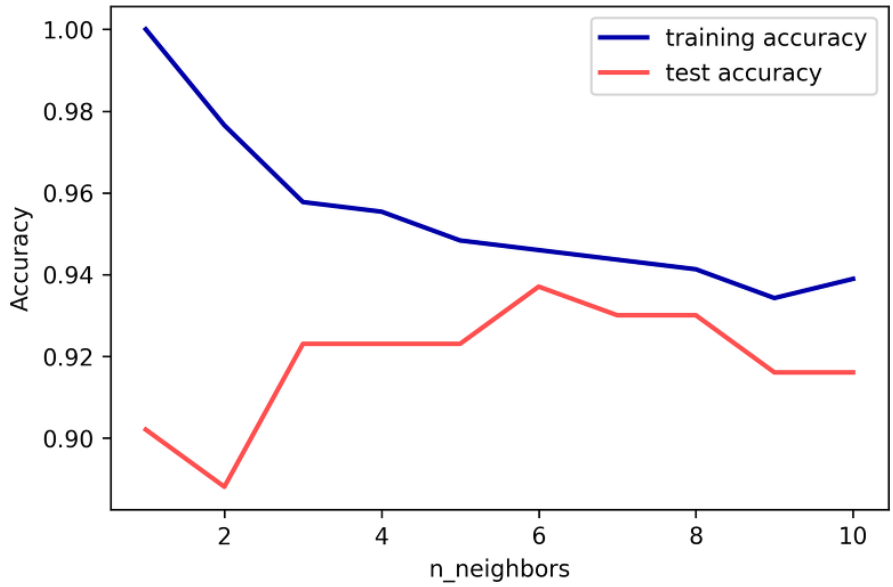

training class 0
training class 1
test pred 0
test pred 1
5
4
3
2
1
0
−1
8.0
8.5
9.0
9.5
10.0
10.5
11.0
11.5
12.0

- 결정 경계(Decision Boundary)
    - 알고리즘이 각각의 클래스로 지정한 영역을 나누는 경계. k가 1일 때는 결정 경계가 훈련 데이터에 가깝게 따라간다. k가 증가하면 결정 경계는 부드워지면서 더욱 단순한 모델이 된다 .
    - k가 각각 1, 3, 9일 때 모델들의 결정 경계



- k값이 작을 때는 훈련 데이터에 대한 정확도는 높지만 테스트 데이터에 대한 정확도는 낮다.
- 이것은 모델이 훈련 데이터에 딱 맞게 과대적합(Overfitting) 되어 있다는 것을 말한다.
- 반대로 k값이 크면 모델이 너무 단순하여 훈련 데이터 세트와 테스트 데이터 세트 모두에 대하여 정확도가 더 나빠진다. 이것은 과소적합(Underfitting)되어 있는 것이다.
- 따라서 k의 개수(최근접 이웃의 수)에는 최적점이 존재한다.

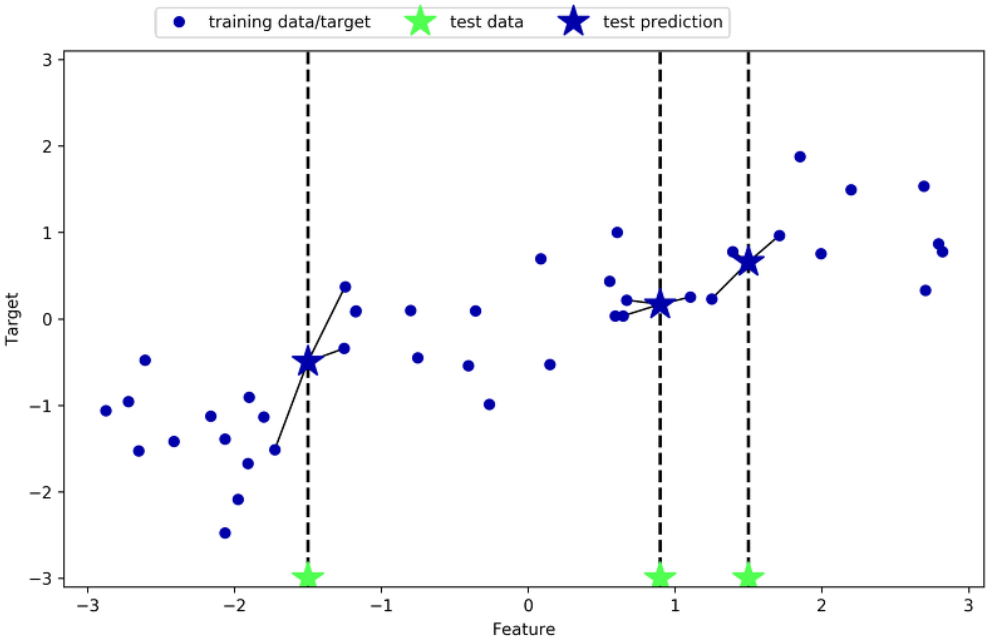
- k갑의 증가에 따른 모델 예측의 정확도



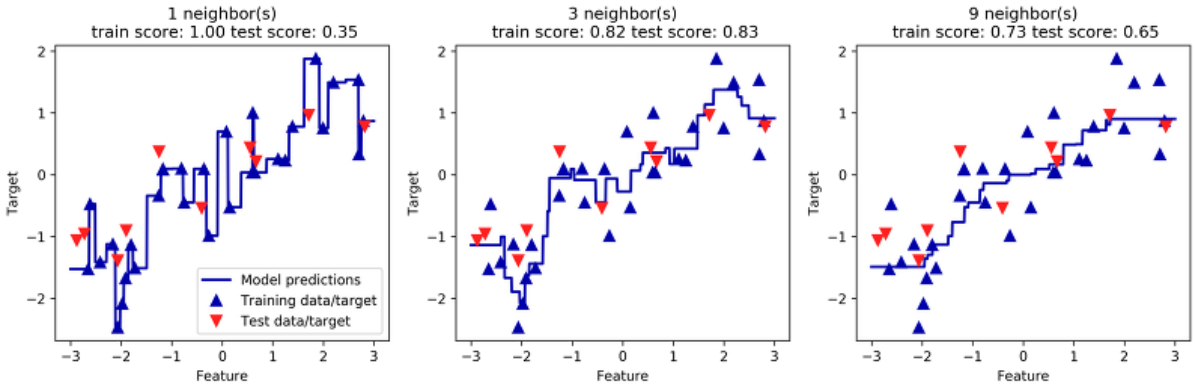
k-최근접 이웃 회귀 알고리즘

- k값이 작을 때의 모델은 가장 가까운 이웃의 값을 그대로 출력한다.
- k값이 여러 개이면, 모델은 최근접 이웃들의 평균을 출력한다.

- k가 3일때 k-NN 회귀 알고리즘 적용 모델



- k 값이 각각 1, 3, 9일 때 모델들의 예측 비교



- k값이 작을 때는 훈련 데이터 하나하나가 출력에 주는 영향이 크기 때문에 모델의 예측값이 훈련 데이터 포인트를 모두 지나간다. k 값이 증가하면 훈련 데이터에는 잘 맞지 않지만 더 안정적인 모델이 된다.

k-최근접 알고리즘 평가

- 장점
  - 이해하기 매우 쉬운 모델이다.
  - 많이 조정하지 않아도 좋은 성능을 발휘한다.
  - 매우 빠르게 만들 수 있어서 더 복잡한 알고리즘을 적용해보기 전에 시도해 볼 수 있다.

- 단점
  - 훈련 세트가 매우 크면 예측이 느려진다.
  - 많은 특성을 가진 데이터 세트에는 잘 동작하지 않는다.

- 특성 값이 대부분 0인 희소한 데이터 세트에는 특히 잘 작동하지 않는다.
- 예측이 느리고 많은 특성을 처리하는 능력이 부족하여 현업에서는 잘 사용하지 않는다.

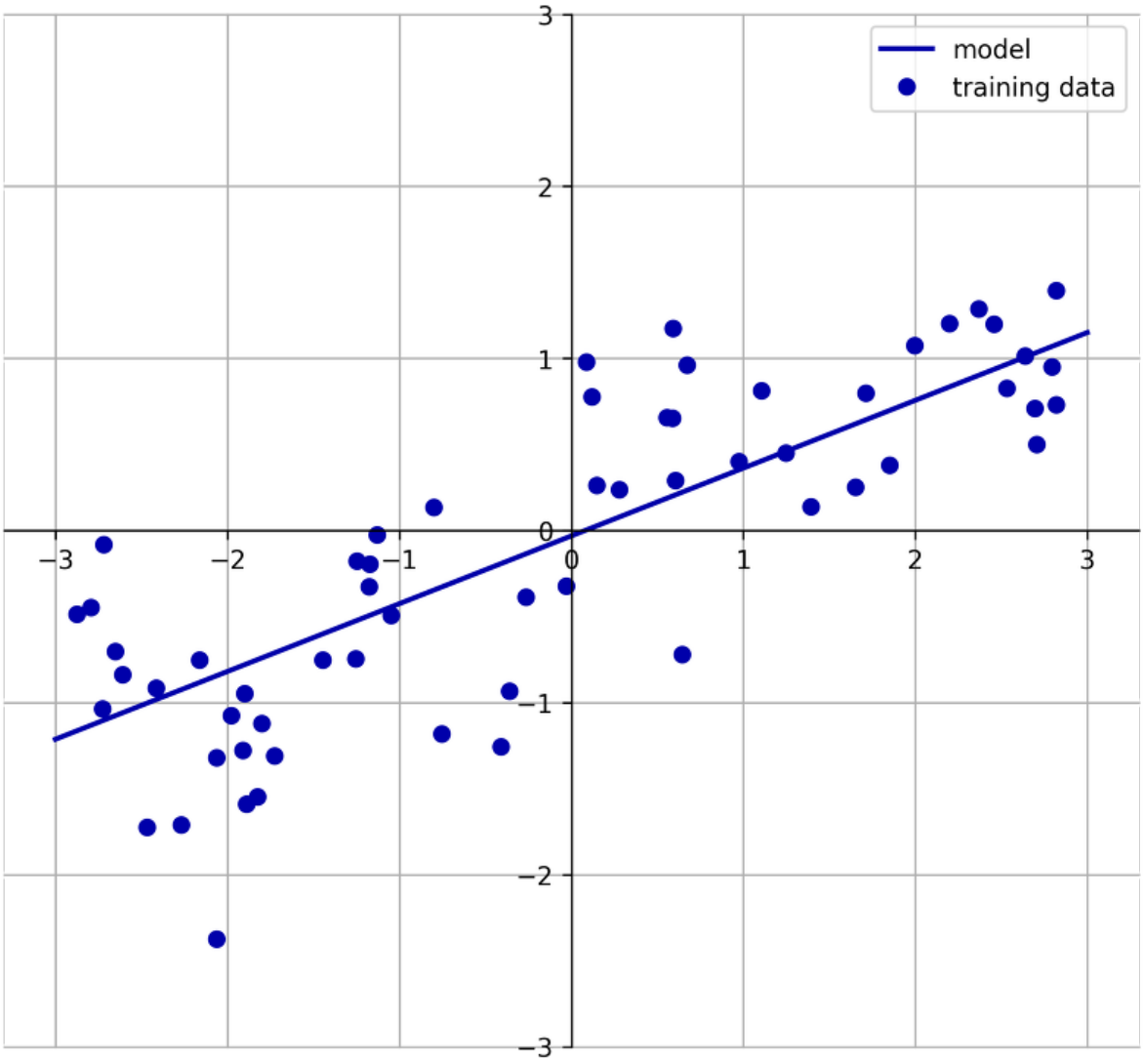
선형 회귀 모델 알고리즘

- 입력 특성에 대한 선형 함수를 만들어 출력을 예측한다.

회귀 선형 모델

- 일반화된 예측 함수 : y = wx + b(w : weight, 가중치, b: bias, 편향)
- 회귀를 위한 선형 모델은 특성이 하나일 때는 직선, 두 개일 때는 평면이 된다.

- 선형 모델의 예측



- 예측 값 y가 각 특성들의 선형 조합이라는 것은 비현실적인 가정일 수 있다. 그러나 이것은 1차원 데이터에서 생각할 수 있는 편견이다.
- 특성이 많은 데이터 세트, 특히나 훈련 데이터보다 특성이 더 많은 경우에는 어떤 예측 값 y도 완벽하게 선형 함수로 모델링할 수 있다.

- 회귀를 위한 선형 모델은 다양하다. 각 모델들은 훈련 데이터로부터 매개 변수 w와 b를 학습하는 방법과 모델의 복잡도를 제어하는 방법에서 차이가 있다.

선형 회귀(Linear Regression, 최고 제곱법)

- 가장 간단하고 오래된 회귀용 선형 알고리즘이다.
- 예측 값과 훈련 세트에 있는 y 사이의 평균 제곱 오차를 최소화하는 매개변수를 찾는다.
- 선형 회귀는 매개변수가 없는 것이 장점이지만, 그로 인해 모델의 복잡도를 제어할 방법 또한 없다. (**최소가 되는 오차에 따라 매개변수의 값이 정해진다. 따라서 매개변수의 변화에 따라 복잡도를 조절할 수 있는 다른 모델과는 달리, 데이터 세트가 주어지고 학습이 끝나면 매개변수가 고정되므로 복잡도를 제어할 방법이 없는 것이다.**)

- 1차원 데이터 세트에서는 모델이 매우 단순하고 제한적이므로 과대적합(Overfitting)이 되지 않는다.
- 특성이 많은 고차원 데이터 세트에서는 선형 모델의 성능이 매우 높아져 과대적합될 가능성이 높다.
- 따라서 기본적인 선형 회귀 방정식 대신 일반적으로 릿지 회귀를 사용한다.

릿지 회귀(Ridge)

- 회귀를 위한 선형 모델로써, 최소제곱법에서 사용한 것과 같은 예측 함수를 사용한다.
- 릿지 회귀에서의 가중치(w) 선택은 훈련 데이터를 잘 예측하기 위해서 뿐만 아니라 추가적인 제약 조건을 만족시키기 위한 목적도 있다.

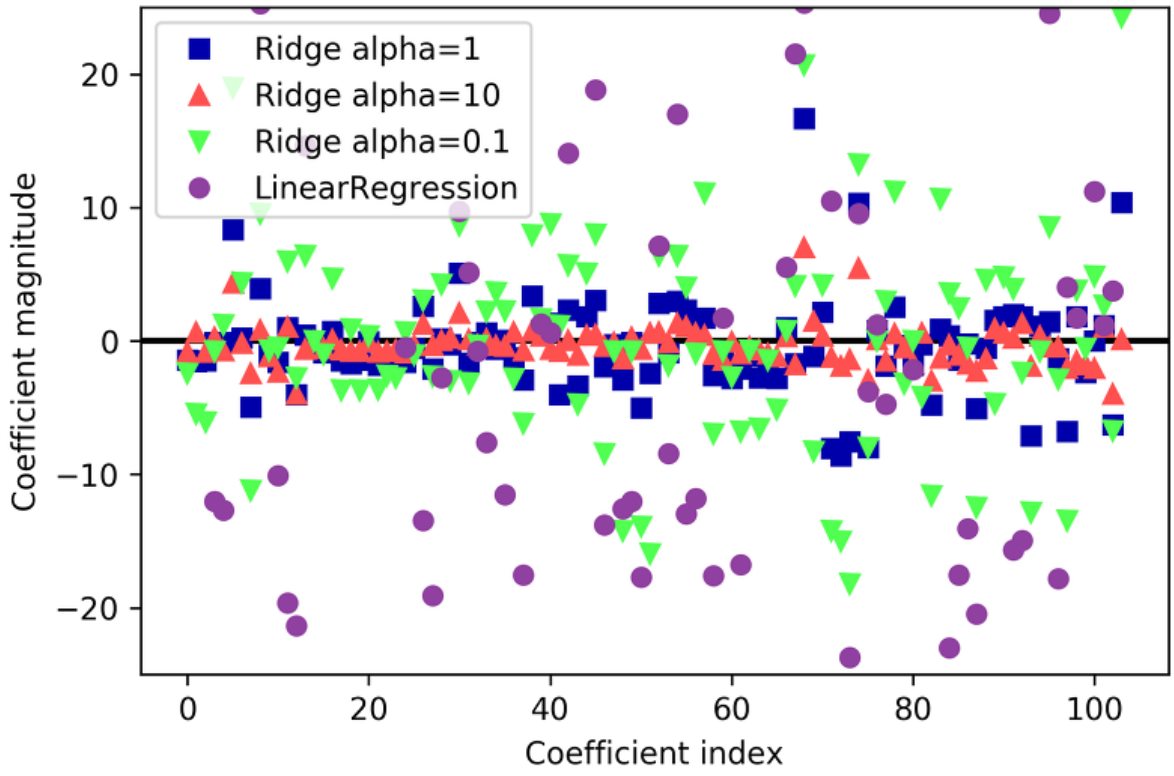
- 규제(Regularization)
    - 가중치의 절댓값을 가능한 작게 만든다. 즉, 가중치의 모든 원소를 0에 가깝게 하여 모든 특성이 출력에 주는 영향을 최소한으로 만든다.
    - (기울기를 작게 만든다) 규제란 과대적합이 되지 않도록 모델을 강제로 제한한다는 의미이다.
    - 릿지 회귀에 사용하는 규제 방식을 L2규제라 한다.

- L2 규제
    - 각 가중치 제곱의 합에 규제 강도(Regularization Strength) α를 곱한다. 그 값을 평균 제곱 오차식에 더한다. α를 크게 하면 가중치가 더 많이 감소되고(규제를 중요시함), α를 작게 하면 가중치가 증가한다(규제를 중요시하지 않음).
    - 가중치를 계산할 때, 이전의 가중치에서 미분한 오차(cost function, 비용함수)에 학습율을 곱한 값을 빼서 계산한다. 따라서 규제 강도 α가 증가하면 오차가 커지고 가중치가 감소되는 폭

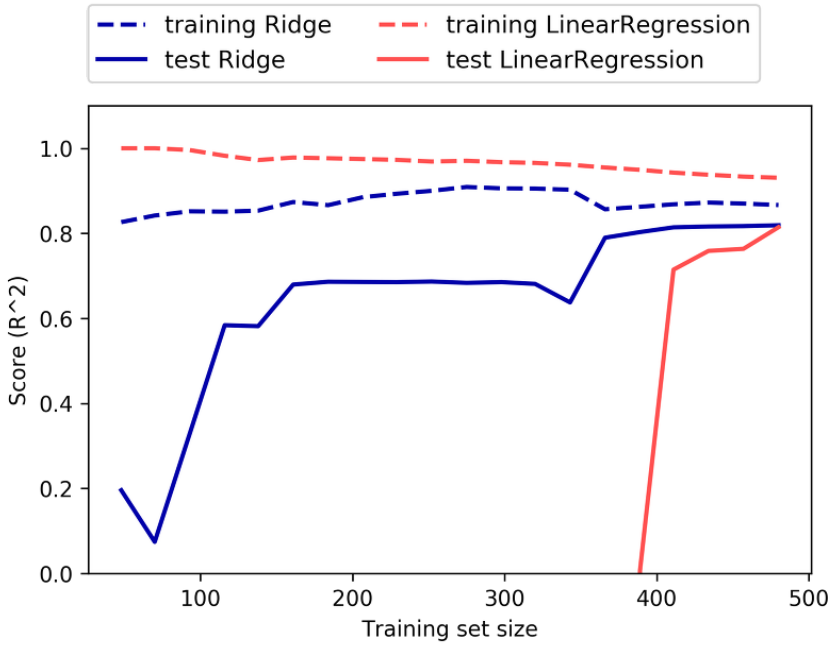
또한 커지게 된다. 즉, 규제 강도를 통해 가중치를 제어할 수 있는 것이다.

- α값이 커지면 w는 0에 가까워진다. 따라서 데이터의 특성을 더 많이 무시하기 때문에 성능(정확도)은 나빠지지만 복잡도가 낮아져 더 일반화된 모델이 된다. 반대로 α값이 작아지면 선형 회귀 모델과 거의 같아진다.

- 릿지 α값에 따른 가중치 w의 크기 변화



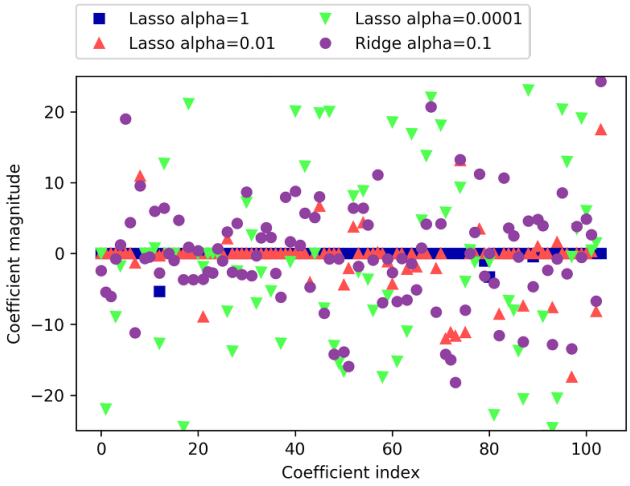
- 릿지 회귀와 선형 회귀의 학습 곡선



- 데이터 세트의 크기에 따른 모델의 성능 변화를 나타낸 그래프를 학습 곡선이라 한다.
    - 릿지 회귀에는 규제가 적용되므로 릿지 회귀의 훈련 데이터 점수가 선형 회귀의 훈련 데이터 점수보다 전체적으로 낮다.
    - 선형 회귀의 경우 데이터의 크기가 커질수록 훈련 데이터 점수가 낮아지는데, 이것은 데이터의 크기가 커지면 모델이 데이터를 기억하거나 과대적합(Overfitting)하기 어려워지기 때문이다.
    - 테스트 데이터의 경우 릿지 회귀의 점수가 더 높으며 데이터 세트의 크기가 작을수록 더 그렇다. 그러나 데이터가 충분하다면, 릿지 회귀와 선형 회귀의 성능은 같아진다.

라쏘(Lasso)

- 선형 회귀에 규제를 적용하는 데에 있어 릿지(Ridge) 회귀의 대안이 된다. 릿지 회귀에서와 같이 라쏘도 계수를 0에 가깝게 만들려고 한다. 그러나 릿지는 L2 규제를 사용하지만, 라쏘는 L1 규제를 사용한다는 점에서 차이가 있다.
- L1 규제
    - L1 규제는 가중치의 제곱의 합이 아닌 가중치의 합을 더한 값에 α를 곱하여 오차에 더한다. 이렇게 하면 L2 규제와는 달리 어떤 가중치 w는 실제로 0이 된다.
    - 즉, 모델에서 완전히 제외되는 특성이 생기는 것이다. 일부 계수를 0으로 만듦으로써 모델을 이해하기 쉬워지고, 모델의 가장 중요한 특성이 무엇인지 드러나는 것이다.



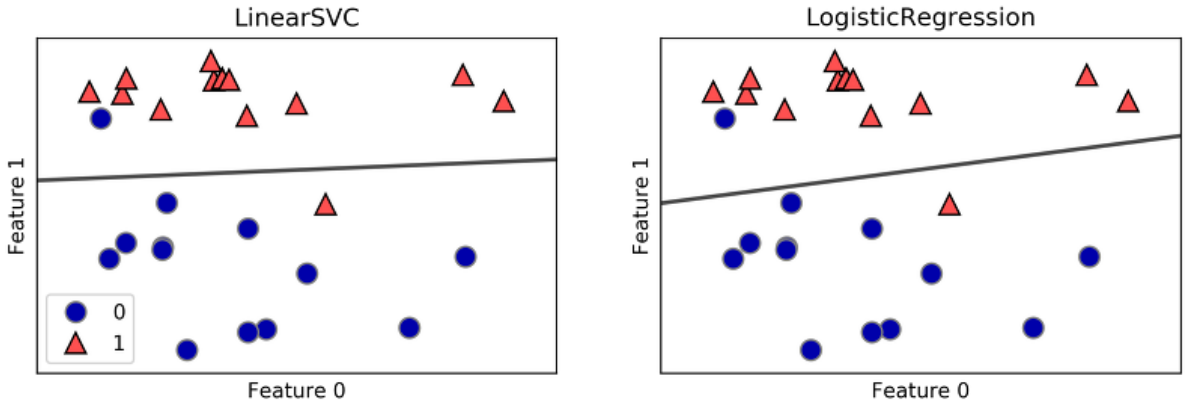
- 릿지 회귀와는 달리 라쏘 α값이 증가하면 어떤 계수들은 실제로 0이 된다.
    - 두 모델 중에서 보통은 릿지 회귀를 선호하지만, 특성이 많고 그 중 일부분만 중요하면 라쏘가 더 좋은 선택이다.

ElasticNet 모델

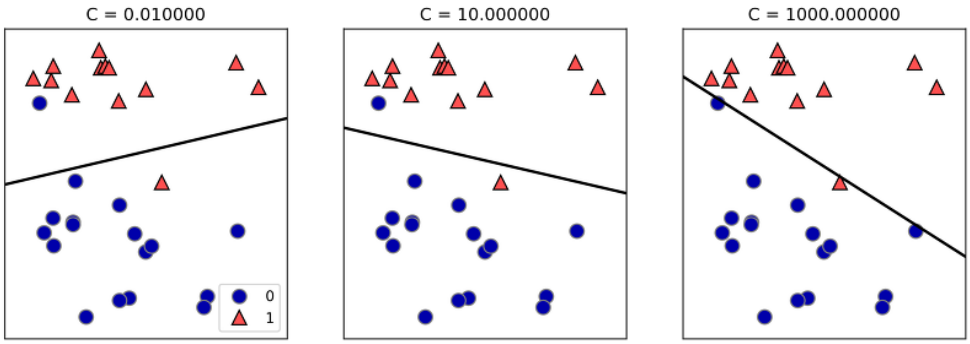
- 라쏘와 릿지를 결합한 모델. L1과 L2 규제를 위한 매개변수 두 개를 조정해야 한다.

선형 분류 모델

- y = wx+b의 함수에서 계산한 값이 0보다 작은지 큰지를 판단하여 클래스를 예측한다.
- 회귀용 선형 모델에서는 출력이 선형 함수이지만, **분류용 선형 모델에서는 결정 경계가 입력의 선형 함수이다.**
- 선형 분류 알고리즘에는 로지스틱 회귀(Logistic Regression, 이름에 회귀가 들어가지만 분류 알고리즘이다), 선형 서포트 벡터 머신(Support Vector Machine : SVC)이 가장 널리 알려져 있다.



- 릿지(Ridge) 회귀와 마찬가지로, 선형 SVC와 로지스틱 회귀 둘 모두 L2 규제를 사용한다. 따라서 둘은 비슷한 결정 경계를 그리고, 그림에서도 동일한 데이터 포인트 하나를 잘못 분류한 것을 볼 수 있다.
- 로지스틱 회귀와 선형 SVC에서 규제의 강도를 결정하는 매개변수는 C이다. 회귀에서는 α와는 반대로, C의 값이 커지면 규제는 오히려 감소한다.



- 매개변수 C가 크면 규제가 거의 없어 선형 분류 알고리즘은 훈련 데이터에 최대한 맞추려고 한다. 그러면 가중치 w는 일반적인 선형 모델과 거의 같아지고, 정확도는 올라가지만 모델은 복잡해져서 일반화가 잘 되지않아 과대적합(Overfitting)이 일어날 가능성이 생긴다.
- 반대로 매개변수 C가 작아지면 규제는 심해져서 데이터 각각의 특성이 많이 무시된다. 가중치 w는 0에 가깝게 수렴하고, 정확도는 떨어지지만 덜 복잡하고, 더 일반화된 모델이 생성된다.

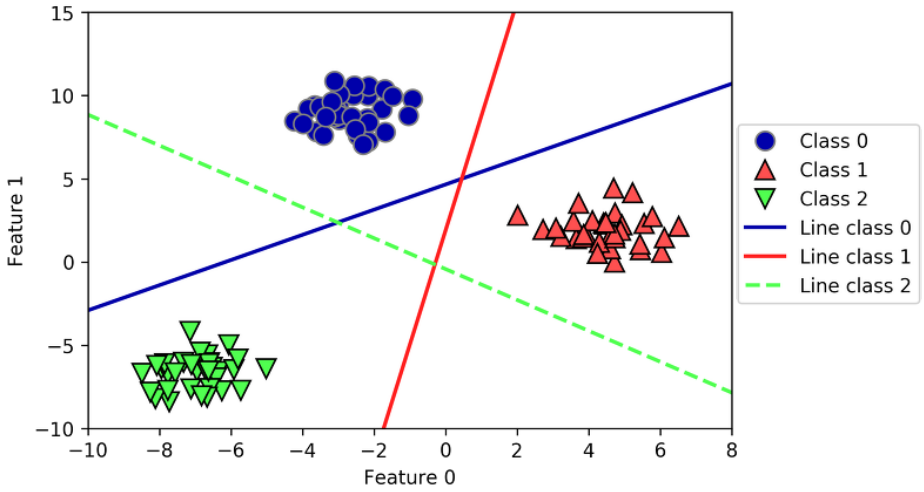
- 회귀와 비슷하게, 분류에서의 선형 모델 또한 낮은 차원의 데이터에서 결정 경계가 직선 혹은 평면이어서 매우 제한적인 것처럼 보인다. 그러나 고차원으로 갈수록 분류에 대한 선형 모델이 매우 강력해지며, 특성이 많아지면 과대적합되지 않도록 하는 것이 매우 중요하다.

다중 클래스 분류용 선형 모델

- 로지스틱 회귀(Logistic Regression)을 제외하고는, 많은 선형 분류 모델은 이진 분류만을 지원하지 않고, 다중 클래스를 지원하지 않는다.

- 이진 분류 알고리즘을 다중 클래스 분류 알고리즘으로 확장하기 위해 주로 일대다(one-vs-rest : ovr) 방법을 사용한다.

- 일대다(ovr) 방법 : 일대다 방법은 각 클래스를 모든 클래스와 구분하도록 이진 분류 모델을 학습시킨다. 그렇게 하면 결국에는 클래스의 수만큼 이진 분류 모델이 만들어지고, 예측 시에 만들어진 모든 이진 분류기가 작동하여 가장 높은 점수를 내는 분류기의 클래스를 예측값으로 선택하는 방법이다.



- 각각의 선이 이진 분류 모델을 뜻한다. 클래스의 수만큼 이진 분류 모델이 만들어진 것을 볼 수 있다. 이를 토대로 결정 경계가 그려진다.
- 이진 분류 모델 사이의 중앙 삼각형 영역 내부에 데이터 포인트가 들어올 경우, 각각의 모델이 평가한 값 중에서 가장 큰 쪽으로 분류되게 된다. 즉, 가장 가까운 직선의 클래스가 된다.

- 매개변수 선정
    - 매개변수 : 회귀모델에서는 α, 분류 모델에서는 C이다. α가

클수록, C는 작을수록 가중치는 0에 가까워지고 모델은 단순해진다. 보통 C와 α는 로그 스케일로 최적치를 정한다. 특별히 회귀 모델에서 매개변수를 조정하는 일이 매우 중요하다.

- 규제 : L1 규제와 L2 규제 중 어떤 규제를 사용할지를 정해야 한다. 중요한 특성이 많지 않다고 생각하거나 해당 모델의 중요한 특성이 무엇인지 해석할 때는 L1 규제를 사용하지만, 그렇지 않으면 기본적으로 L2 규제를 사용해야 한다.

선형 모델 평가

- 장점
    - 학습 속도가 빠르고 예측이 빠르다.
    - 매우 큰 데이터 세트나 희소한 데이터 세트에서도 잘 작동한다.
    - 공식이 존재하여 예측이 어떻게 만들어지는지 비교적 쉽게 이해할 수 있다.

- 단점
    - 데이터 세트의 특성들이 서로 깊게 연관되어 있을 때, 가중치의 값이 명확하지 않아 분석하기 매우 어려울 수 있다.
    - 데이터 세트가 클 때나 고차원에서 잘 작동하나 저차원의 데이터 세트에서는 다른 모델들의 일반화 성능이 더 좋다.

나이브 베이즈(Naive Bayes) 분류기

- 데이터의 각 특성을 개별로 취급하여 매개변수를 학습하고, 각 특성에서 클래스 별로 통계를 단순하게 취합한다.
- 선형 모델과 매우 유사하나 로지스틱 회귀(Logistic Regression)나 선형 SVM보다 훈련 속도가 빠르고, 일반화 성능은 조금 뒤쳐진다.
- 데이터의 특성들이 각각 독립적이고 동등하게 중요하다는 'Naive(순진한)' 가정이 들어간다. 그러나 실제 문제에서 모든 특성이 동등하게 중요하지 않는 경우가 많다.

- Ex) 어떠한 병에 걸릴 확률이 1%, 걸리지 않을 확률이 99%, 그 병에 대한 검사를 할 확률이 20%, 검사하지 않을 확률이 80%라고 생각한다. 각각의 경우가 독립적이라고 가정하면, 다음과 같이 동시확률분포(Joint Probability)를 계산할 수 있다.

	검사 O	검사 X	합계
병 O	0.2 X 0.01 = 0.002(0.2%)	0.8 X 0.01 = 0.008(0.8%)	0.01(1%)
병 X	0.2 X 0.99 = 0.198(19.8 %)	0.8 X 0.99 = 0.792(79.2%)	0.99(99%)
합계	0.2(20%)	0.8(80%)	1(100%)

- 계산된 통계를 취합하여 각 클래스별로 특성의 평균을 계산한다. 새로운 데이터 포인트가 주어지면, 클래스의 통계 값과 비교하여 가장 잘 맞는 클래스를 예측값으로 사용한다.

- 주로 사용되는 나이브 베이즈 분류기에는 가우시안 나이브 베이즈(GaussianNB), 베르누이 나이브 베이즈(BernoulliNB), 다항분포 나이브 베이즈(MultinomialNB)가 주로 사용된다.
- 가우시안 나이브 베이즈는 연속적인 모든 데이터에 적용할 수 있고, 베르누이 나이브 베이즈는 이진 데이터를, 다항분포 나이브 베이즈는 카운트 데이터(ex. 문장 속에 사용된 단어 횟수)에 적용된다. 베르누이와 다항분포 나이브 베이즈는 주로 텍스트 데이터를 분류할 때 사용된다.

- 가우시간 나이브 베이즈는 대부분 매우 고차원인 데이터 세트에 사용하고, 다른 두 나이브 베이즈 모델은 텍스트 같은 희소한 데이터를 카운트하는데 사용한다. 다항분포 나이브 베이즈는 보통 0이 아닌 특성이 비교적 많은 데이터 세트(큰 문서)에서 베르누이 나이브 베이즈보다 성능이 높다.

- 다항분포 나이브 베이즈는 클래스별로 특성의 평균을 계산하는데 반해, 가우시안 나이브 베이즈는 클래스별로 각 특성의 표준편차와 평균을 저장하여 이를 데이터 포인트와 비교해 예측 값을 출력한다.

- 다항분포와 베르누이 나이브 베이즈의 예측 공식은 선형 모델(y=wx+b)과 형태가 같으나, 나이브 베이즈 모델의 계수 w는 기울기가 아니라서 선형 모델과는 의미가 다르다.

- 나이브 베이즈 가우시간 분류 예제 : 주어진 특성으로 남성과 여성 구분 (예제 : https://ko.wikipedia.org/wiki/나이브_베이즈_분류)

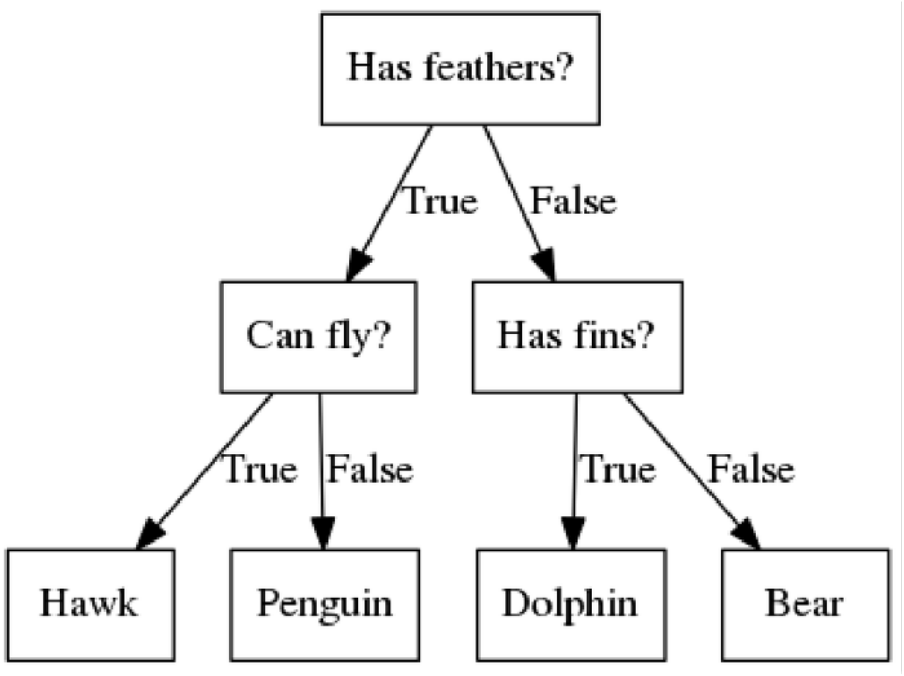
- 매개변수
    - 다항분포와 베르누이 나이브 베이즈는 모델의 복잡도를 조절하는 매개변수 α를 가지고 있다. α가 주어지면, 알고리즘이 모든 특성에 양의 값을 가진 가상의 데이터 포인트를 α의 수만큼 추가한다. 이로 인해 통계 데이터를 완만하게 만들어준다.
    - α가 크면 추가되는 양의 데이터가 많아 통계 데이터가 더 완만해지고 모델의 복잡도는 낮아진다. 다만, α에 따른 알고리즘의 성능 변동은 비교적 크지 않아서, α 값이 성능 향상에 크게 기여하지는 않는다. 그러나 α를 조절하면 어느 정도 정확도를 높일 수 있다.

나이브 베이즈 모델 평가

- 나이브 베이즈 모델과 선형 모델은 장단점이 비슷하다.
- 훈련과 예측 속도가 빠르고, 훈련 과정을 이해하기 쉽다.
- 희소한 고차원 데이터에서 잘 작동하며, 비교적 매개변수에 민감하지 않다.
- 선형 모델로는 학습 시간이 너무 오래 걸리는 매우 큰 데이터 세트에는 나이브 베이즈 모델을 시도해볼만하며, 종종 사용된다.

결정 트리(Decision Tree) 알고리즘

- 의사 결정 규칙과 그에 대한 결과들을 트리 구조로 도시화한 의사 결정 지원 도구의 일종이다.
- 의사를 결정하는데 있어서 질문을 던지고 그 질문에 대한 답들이 트리 구조를 이루고 있는 형태이다.
- 결정 트리는 분류와 회귀 문제에 널리 사용되는 모델이다.
- 기본적으로 결정 트리는 최종적으로 결정에 도달하기 위하여 예/아니오의 이진 분류에 대한 질문을 이어나가면서 학습한다.

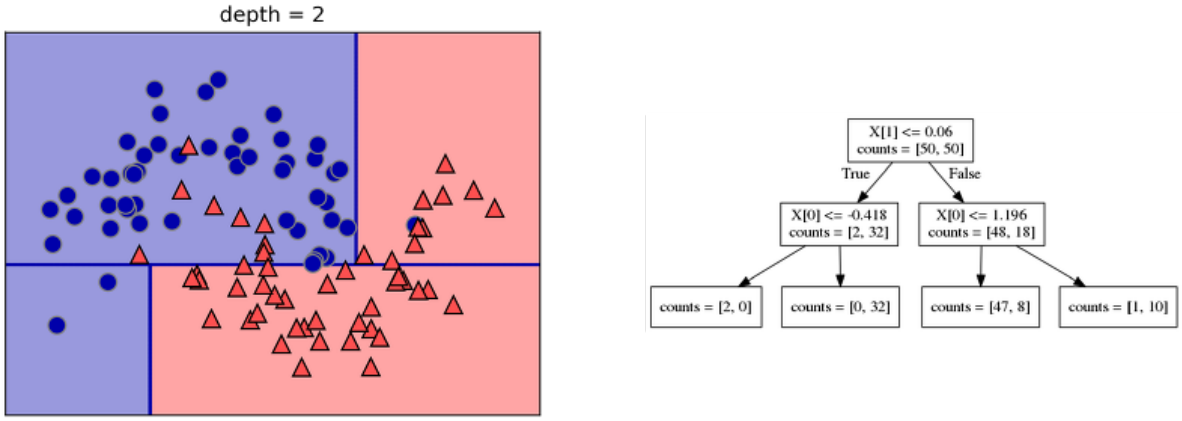


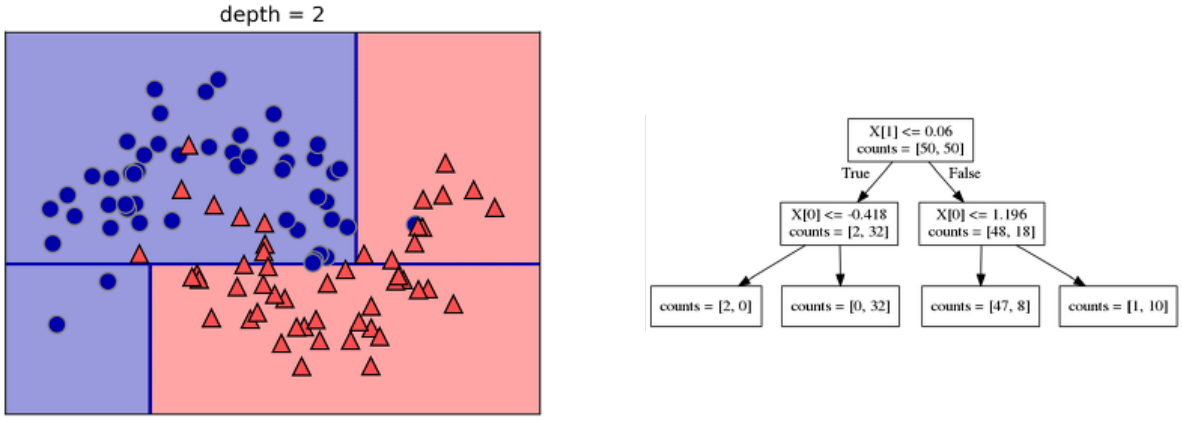
- 트리 구조
    - 트리 구조는 그래프 구조의 일종이다.

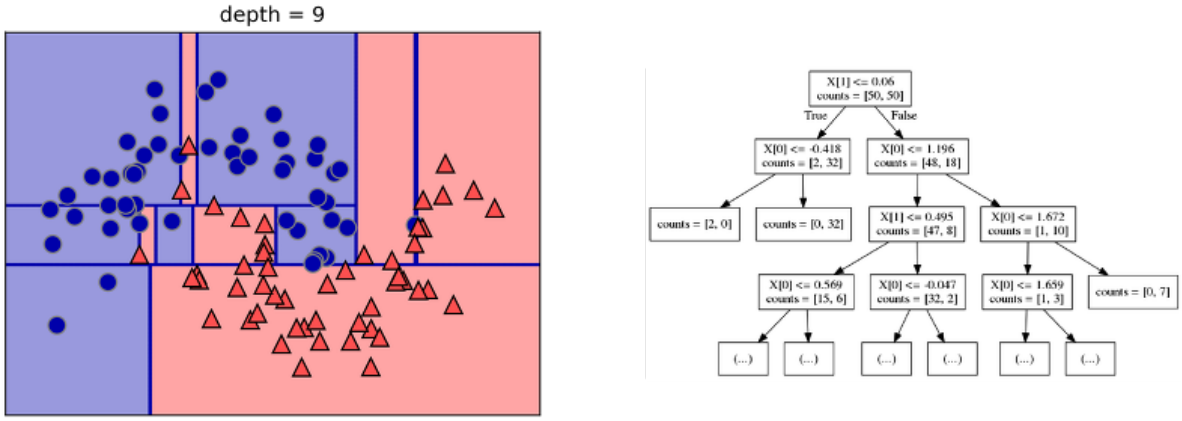
- 그래프 구조는 하나의 노드가 여러 노드와 연결될 수 있지만, 트리 구조에서는 하나의 노드는 최대 2개의 노드와 연결될 수 있다.
- 회로 없이 서로 다른 두 노드를 잇는 길이 하나뿐인 그래프를 트리하고 부른다.
- 트리에서 최상위 노드를 루트 노드(root node, 뿌리)라고 하며, 노드 A가 노드 B를 가리킬 때 A 노드를 B 노드의 부모 노드(parent node), B를 A의 자식 노드(child node)라고 한다.
- 자식 노드가 없는 최하위의 노드를 리프 노드(leaf node, 잎) 라고 하며, 리프 노드가 아닌 자식 노드를 가지고 있는 노드를 내부 노드(internal node)라고 한다.
- 리프 노드중에서도 하나의 타겟 값만을 가지는 노드를 순수 노드(pure node)라고 한다.
- 노드 사이를 연결하는 선은 엣지(Edge)라 하고, 노드가 분기함으로써 생성되는 층을 깊이(depth)라고 한다.

- 결정 트리 만들기
  - 결정 트리를 학습한다는 것은 정답에 가장 빨리 도달하는 질문의 목록을 학습한다는 뜻이다.
  - 머신러닝에서 이런 질문들을 테스트라고 한다.
  - 보통의 테스터는 예/아니오의 형태 특성으로 구성되지 않고, 2차원 데이터 세트와 같이 연속된 특성으로 구성된다. 그렇기 때문에 데이터에 적용할 테스트(질문)는 '해당 특성이 특정한 값보다 큰가?'와 같은 형태를 띈다.

- 트리를 만들 때의 알고리즘은 가능한 모든 테스트에서 타겟 값에 대하여 가장 많은 정보를 가진 것을 고른다.
- 즉, 거의 모든 테스트(질문)에서 데이터를 가장 잘 나누는 테스트를 찾는 것이다.







- 데이터를 나누는 과정을 반복하면 각 노드가 테스트(질문) 하나씩을 가진 이진 결정 트리를 만든다.
- 각 테스트는 하나의 축을 따라 데이터를 둘로 나누는 것으로 생각할 수 있다.
- 이는 계층적으로 영역을 분할해가는 알고리즘이라 할 수 있다 .
- 각 테스트는 하나의 특성에 대하여만 이루어지므로 나누어진 영역은 항상 축과 평행하다.

- 데이터를 분할하는 것은 각 분할된 영역이 분류에서는 하나의 클래스, 회귀에서는 하나의 분석 결과를 가질 때까지 반복된다.
- 분할이 완료되어 모델이 만들어진 후에 새로운 데이터가 주어지면, 그 데이터가 특성을 분할한 영역들 중에서 어디에 위치하는지를 확인하면 그 입력 데이터의 출력을 알 수 있다.
- 도달한 리프 노드의 영역 내에 한 종류의 타겟 값만 존재하면(순수 노드) 그 값이 출력이 되고, 여러 개의 타겟 값이 존재하면 그 중 다수인 것이 출력 된다.

- 분류와 같이 회귀 문제에서도 트리를 사용할 수 있다. 각 노드의 테스트(질문) 결과에 따라 트리를 탐색해나가고 새로운 데이터에 해당되는 리프 노드를 찾는다.
- 찾은 리프 노드의 훈련 데이터 평균값이 주어진 입력 데이터에 따른 출력이 된다.

- 복잡도 제어
  - 일반적으로 트리를 만들 때 모든 리프 노드가 순수 노드가 될 때까지 진행하면, 즉 각 리프 노드가 하나의 타겟 값만을 가지면 모델이 매우 복잡해지고 훈련 데이터에 과대적합(Overfitting)된다.
  - 순수 노드로 이루어진 트리에서, 훈련 데이터는 정확한 클래스의 리프 노드에 위치하지만 새로운 데이터가 주어지면 정확도가 떨어질 수 있다는 것다.
  - **트리의 깊이가 깊어진다는 것은 영역을 더 세밀하게 분할해나가는 것인데, 각 영역 내에 정확하게 하나의 클래스 타겟 값만 위치하도록 영역을 분할하면 모델이 너무 복잡하게 생성된다는 것이다.**
  - 결정 결계가 값 하나하나에 너무 민감하기 때문이다.

  - 한계를 주지 않으면 트리는 무한정 깊어지고 복잡해진다.
  - 모델이 너무 복잡해져 과대적합되는 것을 방지하는 방법은 크게 두가지가 있다.
    - 트리 생성을 일찍 중단하는 **사전 가지치기(Pre-Pruning)** 전략. 트리의 최대 깊이 혹은 리프 노드의 최대 개수를 제한하거나 노드가 분할하기 위한 포인트의 최소 한계를 지정해 주는 것이다.
    - 트리를 만든 후에 데이터 포인트가 적은 노드를 삭제하거나 병합하는 **사후 가지치기(Post-Pruning) 혹은 가지치기(Pruning)** 전략

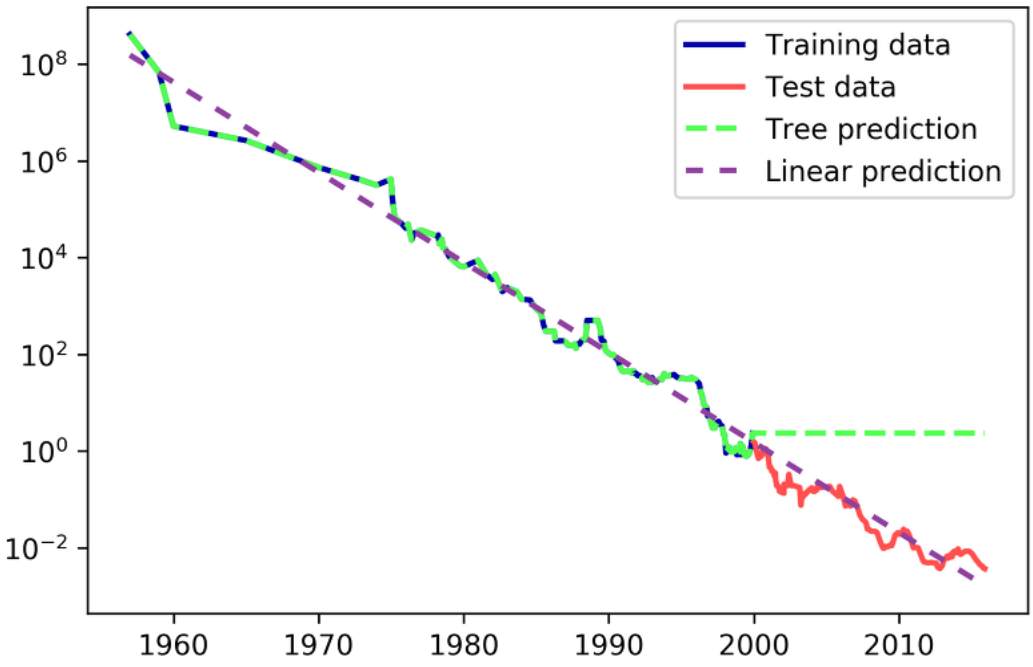
- 특성 중요도(Feature Importance)
    - 전체 트리를 살펴보는 것은 어려울 수 있으니, 트리가 어떻게 작동하는지 요약하는 속성들을 사용할 수 있다.
    - 특성 중요도도 그 중의 하나로, 트리를 만드는 결정에 각 특성에 각 특성이 얼마나 중요한지를 평가하는 것이다.
    - 이 값은 0과 1사이의 숫자로, 각 특성에 대해 0은 전혀 사용되지 않았다는 뜻이고 1은 완벽하게 타겟 클래스를 예측했다는 뜻이다.
    - 특성 중요도 전체의 합은 1이다.
    - 즉, 특성 중요도가 높으면 트리 구조에서 높은 층에 해당하는 노드의 테스트(질문)라는 것이고 특성 중요도가 낮으면 낮은 층에 해다하는, 세세하게 클래스를 분류하는 노드의 테스트(질문)이거나 혹은 테스트(질문)에 사용되닌 않는 특성이라는 것이다.

    - 그러나 특성 중요도가 낮다고 하여 그 특성이 유용하지 않다는 것은 아니다.
    - 단지 트리가 그 특성을 선택하지 않았을 뿐이며, 다른 특성이 동일한 정보를 지니고있어서 일 수 있다.
    - 음수가 될 수 있고 특성의 중요도를 나타내는 선형 모델의 가중치(계수)와는 달리, 특성 중요도는 항상 양수이며 특성이 어떤 클래스를 지지하는지는 알 수 없다.

회귀 결정 트리 알고리즘

- 회귀 결정 트리 알고리즘
  - 회귀 결정 트리에서도 이런 특성은 동일하게 적용되나 분류와는 달리 회귀를 위한 트리 모델에는 중요한 속성이 하나 있다.
  - 모든 트리 기반 회귀 모델은 외삽(Extrapolation), 훈련 데이터의 범위 밖에 위치한 포인트에 대해서는 예측을 할 수 없다는 것이다.
  - **선형 모델은 전체 데이터의 경향성을 토대로 모델을 만들기 때문에 훈련 데이터의 범위 밖이더라도 예측할 수 있지만, 훈련 데이터가 위치한 영역을 분할하여 예측하는 결정 트리는 범위 밖에 위치한 데이터에 대해서는 영역이 분할되어 있지 않기 때문에 모델이 예측할 수 없는 것이다.**

- 램(RAM) 가격 동향



- 2000년 전까지의 데이터로부터 2000년 후의 가격을 예측한 그래프이다.
- 선형 회귀 모델의 경우 직선으로 데이터를 근사하였고, 2000년 이후의 테스트 데이터를 거의 정확히 예측한다. 2000년 이전의 가격 동향을 토대로 생성된 모델이기 때문이다.
- 그러나 결정 트리의 경우 2000년 이전의 훈련 데이터에 대해서는 거의 완벽하게 예측을 하지만, 2000년 이후의 테스트 데이터에 대해서는 아예 예측을 하지 못한다. 그저 마지막 데이터 값을 그대로 출력할 뿐이다.
- 즉, 트리 모델은 훈련 데이터 밖의 새로운 데이터를 예측할 능력이 없다. 이는 모든 트리 기반 모델의 공통된 단점이다.

- 매개변수 선정
  - 결정 트리에서 모델의 복잡도를 조절하는 매개변수는 트리가 완전히 만들어지기 전에 멈추도록 하는 사전 가지치기 매개변수이다.
  - 보통은 사전 가지치기 방법 중에서 최대 깊이(Max depth), 최대 리프 노드의 개수(Max leaf nodes), 노드가 분할하기 위한 포인트의 최소 한도(Min samples leaf) 중에서 하나만 지정해주어도 과대적합(Overfitting)을 막는 데는 충분하다.

결정 트리 알고리즘 평가

- 장점
    - 만들어진 모델을 쉽게 시각화할 수 있어서 비전문가도 이해하기 쉽다.
    - 데이터의 스케일에 구애받지 않는다는 것이다. 각 특성이 개별적으로 처리되어 데이터를 분할하기 때문에, 데이터 스케일의 영향을 받지 않으므로 결정 트리에서는 데이터가 특정 범위 안에 들어오도록 하는 정규화(Normalization)나 표준화(Standardization)같은 데이터 전처리 과정이 필요가 없다. 특히나 특성의 스케일이 서로 다르거나 이진 특성과 연속적인 특성이 혼합되어 있을 때에도 잘 작동한다.
    - 학습이 끝난 결정 트리의 작업 속도가 매우 빠르다는 것 또한 장점이다.
    - 결정 트리는 모델이 어떻게 훈련되었는지 경로의 해석이 가능하다는 것이다.

- 단점
    - 사전 가지치기를 사용함에도 과대적합(Overfitting)되는 경향이 있어 모델의 일반화 성능이 좋지 않다는 것이다.
    - 단일 결정 트리의 이러한 단점을 보완한 것이 결정 트리의 앙상블 방법이다.

결정 트리 앙상블

- 앙상블(Ensemble)
    - 앙상블은 여러 머신러닝 모델을 연결하여 더 강력한 모델을 만드는 기법이다.
    - 큰 데이터 세트가 있을 때, 데이터 세트를 나누어 독립적으로 운영되는 각각의 머신러닝을 통해 학습시킨 뒤 모델을 합쳐 전체 데이터의 결과를 산출하는 방법이다.
    - 단일 머신러닝 모델로 학습하였을 때 보다 2%에서 5%정도 상승된 결과를 얻을 수 있다.

    - 머신러닝에는 여러 종류의 앙상블 모델이 많지만, 그 중에서 모델을 구성하는 기본 요소로 결정 트리를 사용하는 **랜덤 포레스트(Random Forest)**와 **그래디언트 부스팅(Gradient Boosting)** 이 두 앙상블 모델이 분류와 회귀 문제의 다양한 데이터 세트에서 효과적이라고 입증되었다.

랜덤 포레스트(Random Forest)

- 랜덤 포레스트는 기본적으로 조금씩 다른 여러 결정 트리의 묶음으로, 훈련 과정에서 구성한 다수의 결정 트리로부터 분류 또는 회귀 분석 결과를 출력함으로써 동작한다.
- 랜덤 포레스트를 사용하면 훈련 데이터에 과대적합되는 경향이 있는 단일 결정 트리 학습의 단점을 회피할 수 있다.

- 랜덤 포레스트의 아이디어는 각각의 단일 결정 트리는 비교적 예측은 잘 하지만, 데이터 일부에 과대적합(Overfitting)되는 경향을 가진다는데 기초한다.
- 즉, 작동 자체는 잘하므로 서로 다른 방향으로 과대적합된 트리를 많이 만들고 앙상블시켜 그 결과를 평균낸다면 과대적합된 양을 줄일 수 있다는 것이다.
- 이렇게 하면 트리 모델의 예측 성능은 유지되면서 과대적합이 줄어들 수 있다는 것이 증명되었다.

- 랜덤 포레스트 구축
    - 랜덤 포레스트를 구현하려면 일단 결정 트리를 많이 만들어야 한다.
    - 각각의 트리는 타겟에 대한 예측을 잘 수행해야 하고, 다른 트리와는 구별되어야 한다.
    - 랜덤 포레스트라는 이름에서 알 수 있듯이 구현할 때 트리들이 달라지도록 트리 생성 시에 무작위성을 주입한다.
    - 랜덤 포레스트에서 트리를 랜덤하게 만드는 방법은 두 가지이다.
        - ■ 트리를 만들 때 사용하는 데이터 포인트를 무작위로 선택하는 방법
        - ■ 노드가 분기하는 테스트(질문)에서 사용하는 특성을 무작위로 선택하는 방법
    - 랜덤 포레스트 모델을 만들려면 생성할 트리의 개수를 정해야 한다.
    - 트리들은 완전히 독립적으로 만들어져야 하므로 알고리즘은 각 트리가 고유하게 만들어지도록 무작위한 선택을 한다.
    - 따라서 트리를 만들기 전에 데이터의 **부트스트랩 샘플(Bootstrap Sample)**, 전체 데이터 포인트 중에서 무작위로 데이터를 전체 데이터의 개수만큼 중복 및 반복 추출하여 만든 샘플을 만든다.
    - 부트스트랩 샘플은 원래 데이터 세트와 크기는 같지만, 어떤 데이터 포인트는 누락될 수 있고, 어떤 데이터 포인트는 중복될 수도 있다. 이 샘플 데이터를 토대로 결정 트리를

만든다.

- 기존의 단일 결정 트리 알고리즘과는 조금 다르게 랜덤 포레스트 결정 트리 알고리즘을 만든다.
- 기존의 결정 트리는 각 노드에서 전체 특성을 대상으로 최선의 테스트(질문)를 찾는 것인데, 랜덤 포레스트에서는 알고리즘이 각 노드에서 후보 특성을 무작위로 선택한 후에 그 후보들 중에서 최선의 테스트를 찾는 것이다.
- 이러한 후보 특성을 고르는 것은 매 노드마다 반복되므로 트리의 각 노드는 다른 후보 특성들을 사용하여 테스트를 만들게 된다.
- 즉 결정 트리를 만들기 전에 데이터 세트 자체도 무작위로 추출해놓고, 그것을 토대로 결정 트리를 만들 때에도 노드가 분기하는(영역을 분할하는) 테스트(질문)의 후보들도 무작위로 추출하는 것이다.
- 이 두 매커니즘이 합쳐져서 랜덤 포레스트의 모든 트리가 서로 달라지도록 만든다.

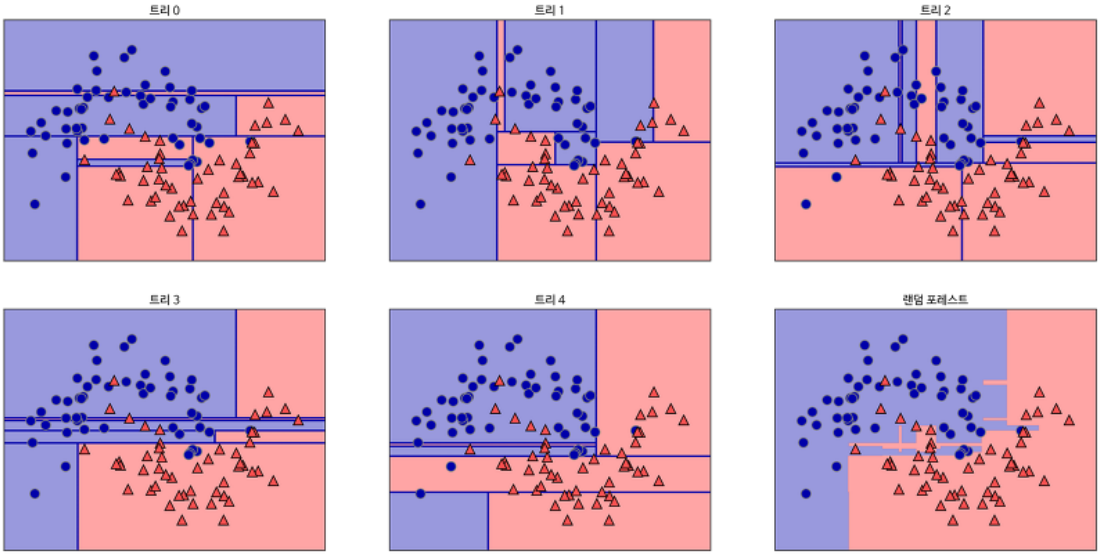
- 이러한 방식에서 핵심적인 매개변수는 **후보 특성의 최대 추출 개수(Max feature)**이다.
- 후보 특성의 최대 추출 개수를 전체 특성 개수로 설정하면 트리의 각 분기에서 결국 모든 특성을 고려하므로 특성 선택에 있어서 무작위성이 들어가지 않는다.
- 만약 후보 특성의 최대 추출 개수를 1로 둔다면 트리의 분기는 테스트(질문)할 특성을 고를 필요 없이 그냥 무작위로 선택한

특성의 임계치를 찾기만 하면된다.

- **결국 후보 특성 최대 추출 개수를 크게하면 랜덤 포레스트의 각 트리들은 서로 비슷해지고, 가장 두드러진 특성을 이용하여 데이터에 잘 맞춰질 것이다. 반대로 최대 추출 개수를 작게 하면 랜덤 포레스트의 각 트리들은 많이 달라지고, 데이터에 맞추기 위해 깊이가 깊어지게 된다.**

- 랜덤 포레스트로 예측을 할 때는 먼저 알고리즘이 모델에 있는 모든 트리의 예측을 만든다.
- 회귀의 경우에는 이 예측들의 평균으로 최종 예측을 만든다.
- 분류의 경우는 먼저 랜덤 포레스트를 구성하는 각각의 트리 알고리즘이 가능성 있는 출력 레이블의 확률을 제공함으로써 간접적인 예측을 한다. 각 트리들이 예측한 확률을 클래스별로 평균내어 가장 높은 확률을 가진 클래스가 예측값이 된다.

- 랜덤 포레스트 분석



- 다섯 개의 트리가 만든 결정 경계가 서로 확연하게 다르다.
- 부트스트랩 샘플로 인해 한쪽 트리에서 나타나는 훈련 데이터가 다른 트리에는 포함되지 않을 수 있기 때문에, 각 트리는 불완전하다.
- 다섯개의 트리가 합쳐져 생성된 최종적인 랜덤 포레스트 모델은 개개의 트리보다는 덜 과대적합(Overfitting)되고 훨씬 좋은 결정 경계를 만들어준다.
- 실제 사용의 경우 수백, 수천 개의 매우 많은 트리를 사용하기 때문에 더 부드러운 결정 경계가 만들어진다.

- 기본적으로 랜덤 포레스트는 아무런 매개변수 튜닝 없이도 선형 모델이나 단일 결정 트리보다 높은 정확도를 보인다.
- 단일 결정 트리에서 한 것 처럼 매개변수를 조정하거나 사전 가지치기를 할 수도 있으나, 랜덤 포레스트는 기본 설정만으로도 좋은 결과를 만들어줄 때가 많다.

- 결정 트리처럼 랜덤 포레스트 또한 특성 중요도를 제공하는데, 각 트리의 특성 중요도를 취합하여 계산한 것이다.
- 일반적으로 랜덤 포레스트에서 제공하는 특성 중요도가 하나의 트리에서 제공하는 것보다 더 신뢰할 만하다.
- 랜덤 포레스트를 만드는 무작위성은 알고리즘이 가능성 있는 많은 경우를 고려할 수 있도록 하므로, 그 결과로 랜덤 포레스트가 단일 트리보다 더 넓은 시각으로 데이터를 바라볼 수 있기 때문이다.

랜덤 포레스트 평가

- 장점
    - 랜덤 포레스트는 회귀와 분류에 있어서 현재 가장 널리 사용되는 머신러닝 알고리즘이다.
    - 기본적으로 랜덤 포레스트는 단일 트리의 단점을 보완하고 장점을 그대로 가지고 있다.
    - 성능이 매우 뛰어나 매개변수 튜닝을 하지 않아도 잘 작동하여 데이터의 스케일을 맞출 필요도 없다.

- 단점
    - 수십, 수백 개의 트리를 가지고 있는 랜덤 포레스트는 자세히 분석하기 어렵고, 특성의 일부만을 사용하므로 단일 결정 트리보다 더 깊어지는 경향이 또한 존재한다.
    - 비전문가에게 예측 과정을 시각적으로 보여주기 어려운 것이 단점이다.
    - 랜덤 포레스트는 텍스트 데이터 같은 차원이 높고 희고한

데이터에는 잘 작동하지 않는다.

- 대량의 데이터 세트에서 잘 작동하나 랜덤 포레스트 모델을 생성하기에 선형 모델보다 만은 메모리를 사용하여 훈련과 예측이 느려 시간이 다소 오래 걸린다.
- 항상 무작위적이기 때문에 특정 변수(Random state)를 지정해주지 않으면 항상 조금씩 다른 결과를 만들어 낸다.
- 트리가 많아질수록 변동은 적어진다.

그래디언트 부스팅(Gradient Boosting)

- 그래디언트 부스팅 회귀 트리(Gradient Boosting Regression Tree)
    - 여러 개의 결정 트리를 묶어 강력한 모델을 만드는 또 다른 앙상블 방법이다.
    - 선형 분류 모델의 한 종류인 로지스틱 회귀(Logistic Regression)와 마찬가지로 이름에 회귀가 들어가지만 이 모델은 회귀와 분류 모두에 사용할 수 있다.
    - 무작위의 데이터로 한꺼번에 다른 트리들을 만들어 결합하는 랜덤 포레스트와는 달리, 그래디언트 부스팅은 이전 트리의 오차를 보완하는 방식으로 작동하기에 결정 트리를 순차적으로 만들어나간다.
    - 이전에 만든 트리의 예측과 타겟 값 사이의 오차를 줄이는 방향으로 새로운 트리를 추가하는 알고리즘이다.
    - 이를 위해 손실 함수(Cost Function)를 정의하고 경사 하강법(Gradient Descent)을 사용하여 다음에 추가될 트리가 예측해야 할 값을 보정해나간다.

    - 기본적으로, 그래디언트 부스팅 회귀 트리에는 무작위성이 없다. 대신 사전 가지치기가 매우 강력하게 사용된다.
    - 그래디언트 부스팅 트리는 얕은 트리같은 간단한 모델(약한 학습기 : Weak Learner)들을 많이 연결하여 하나의 앙상블 모델을 만든 것이다.
    - 그래디언트 부스팅 트리에는 보통 1에서 5정도의 깊지 않은

트리들을 사용하여 메모리를 적게 사용하고 예측도 빠르다.

- 각 트리의 깊이가 깊지 않아 각각의 트리는 데이터의 일부에 대해서만 예측을 잘 수행할 수 있기 때문에 트리가 많이 추가될수록 성능이 좋아진다.

- 그래디언트 부스팅 트리는 랜덤 포레스트보다는 매개변수 설정에 조금 더 민감하지만 잘 조정만 한다면 더 높은 정확도를 제공해준다.
- 앙상블 방식에 있는 사전 가지치기나 트리의 개수 외에도 그래디언트 부스팅에서 중요한 매개변수는 이전 트리의 오차를 얼마나 강하게 보정할 것인지를 제어하는 **학습률(Learning Rate)**이다.
- 학습률이 크면 트리는 보정을 강하게 하기 때문에 복잡한 모델을 만든다.
- 또한 앙상블에 트리의 개수를 많이 추가하면 모델의 복잡도가 커지고 훈련 세트에서의 실수를 바로잡을 기회가 더 많아진다.(일반화 성능은 떨어진다)

- 그래디언트 부스팅 트리 또한 다른 결정 트리 기반의 모델처럼 특성 중요도가 존재한다.
- 다만 그래디언트 부스팅 트리는 일부 특성을 완전히 무시한다는 점에서 전체적인 특성을 대부분 사용하는 랜덤 포레스트 방식과는 차이를 보인다.
- 따라서 비슷한 종류의 데이터라면 그래디언트 부스팅과 랜덤 포레스트 모두 잘 작동하지만, 일반적으로 더 안정적인 랜덤

포레스트를 먼저 적용한다.

- 랜덤 포레스트가 잘 작동한다 하더라도 예측 시간이 중요하거나 머신러닝 모델에서 마지막 성능까지 필요할 때 그래디언트 부스팅을 사용하면 도움이 된다.
- 또한 랜덤 포레스트는 과대적합된 모델들을 이용하기 때문에 트리의 개수가 많을 수록 좋지만, 그래디언트 부스팅은 트리의 개수가 많아지면 과대적합되기 때문에 적절한 변수 조정이 필요하다.

그래디언트 부스팅 트리 평가

- 장점
  - 그래디언트 부스팅 결정 트리는 다른 트리 기반 모델처럼 특성의 스케일을 조정하지 않아도 되고, 이진 특성이 연속적인 특성에서도 잘 동작한다.

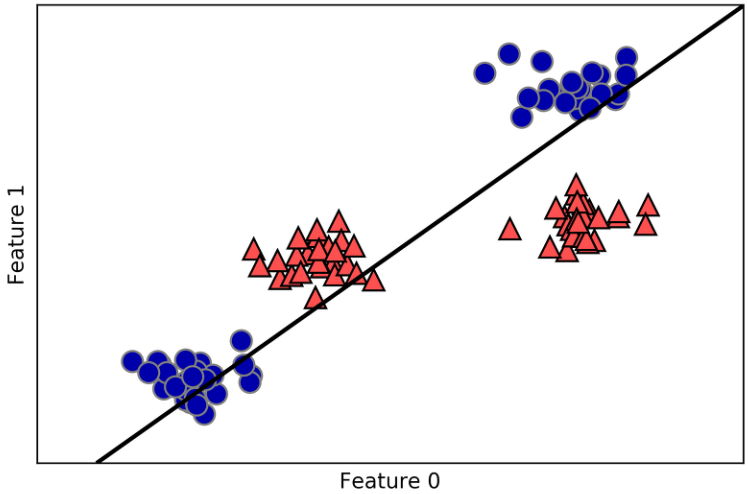
- 단점
  - 매개변수를 잘 조정해야 한다는 것과 훈련 시간이 길다는 것이다.
  - 트리 기반 모델의 특성상 희소한 고차원 데이터에는 잘 작동하지 않는 다는 것이 단점이다.

서포트 벡터 머신(Support Vector Machine, SVM) 알고리즘

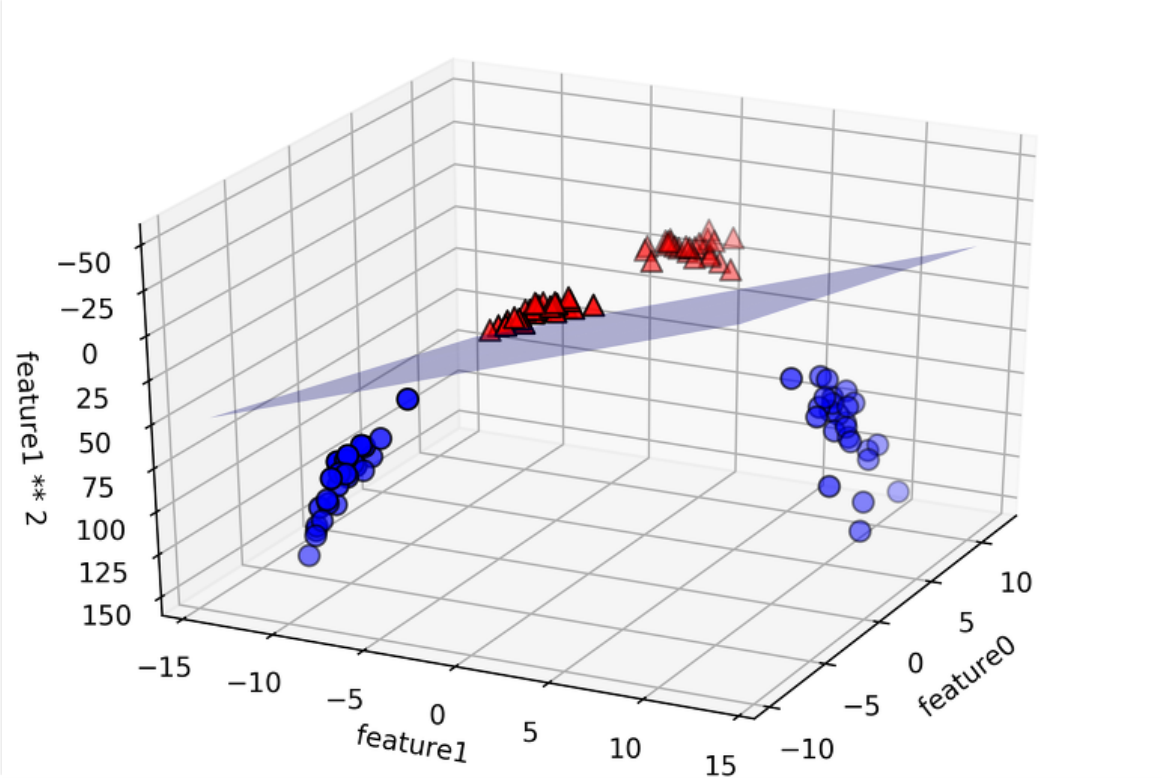
- 학습이 진행되는 동안 각 훈련 데이터 포인트가 두 클래스 사이의 결정 경계를 구분하는 데 얼마나 중요한지를 학습한다.
- 일반적으로 훈련 데이터의 일부, 두 클래스 사이의 경계에 위치한 데이터 포인트들만이 결정 경계를 만드는데 영향을 준다.
- 이런 데이터 포인트를 **서포트 벡터(Support Vector)**라 하며, 여기서 서포트 벡터 머신이라는 이름이 유래하였다.

- 새로운 데이터 포인트에 대해 예측할 때는, 데이터 포인트와 각 서포트 벡터와의 거리를 측정한다.
- 분류에 대한 결정은 서포트 벡터까지의 거리에 기반하며 서포트 벡터의 중요도는 훈련 과정에서 학습한다.
- 서포트 벡터 머신은 서로 다른 클래스를 지닌 데이터 사이의 간격이 최대가 되는 선이나 평면을 찾아 이를 기준으로 각 데이터들을 분류하는 모델이다.
- 데이터 사이에 존재하는 여백(margin)을 최대화하여 일반화하는 성능을 극대화한 모델이다.

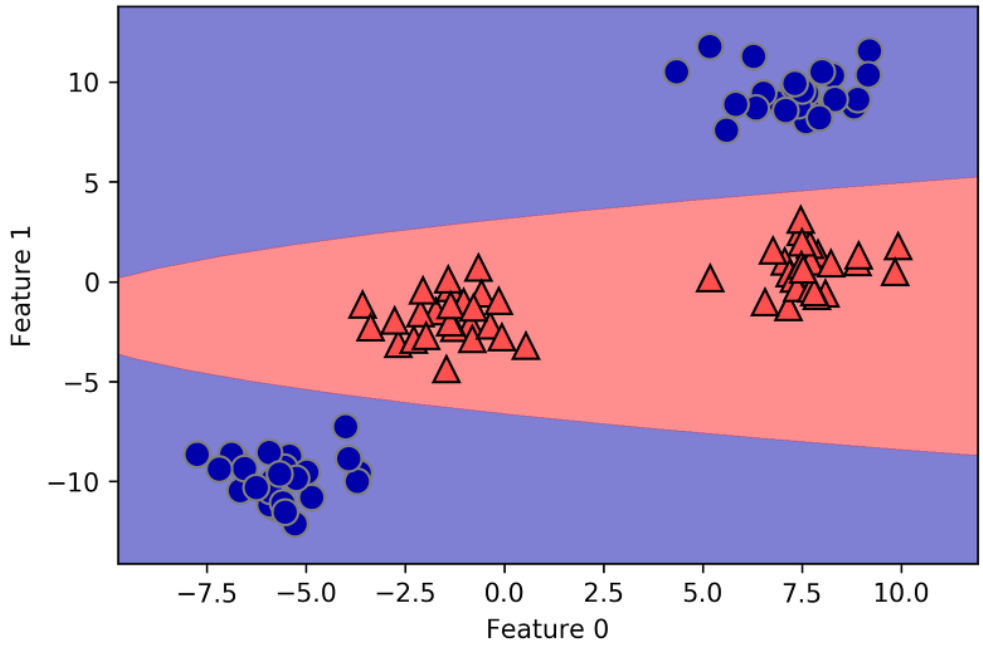
- 커널 서포트 벡터 머신(Kernelized Support Vector Machine : KSVM)
    - 커널 서포트 벡터 머신은 분류용 선형 서포트 벡터 머신과는 다르게 선형적으로 분류되지 않는 클래스를 구분하도록 비선형적 특성을 추가한 알고리즘이다. 즉 비선형 서포트 벡터 머신이다.



- 직선과 평면, 초평면은 유연하지 못하여 저차원 데이터 세트에서 선형 모델은 매우 제한적이다.
- 분류를 위한 선형 모델은 직선으로만 데이터 포인트를 나눌 수 있어서 위와 같은 데이터 세트에는 잘 들어맞지 않는다.
- 이러한 선형 모델을 유연하게 만드는 방법은 특성끼리 곱하거나 특성을 거듭제곱하는 식으로 새로운 특성을 추가하는 것이다. 그러면 더 고차원의 데이터가 생성된다.



- 새롭게 생성된 고차원의 데이터 세트에서는 선형 모델과 3 차원 공간의 평면을 사용해 두 클래스를 구분할 수 있다. 그러나 이것을 원래 특성으로 투영해보면, 이 선형 SVM 모델은 더 이상 선형이 아닌 비선형의 모습을 확인할 수 있다.



- 데이터 세트에 비선형적 특성을 추가하여 선형 모델을 강력하게 만든것이다.
- 하지만 대부분의 경우에 어떠한 특성을 추가해야 하는지 모르고, 또한 특성을 많이 추가하면 연산 비용이 커진다.
- 이러한 상황에서 수학적 기교를 사용하여 새로운 특성을 많이 만들지 않고도 고차원에서 분류기를 학습시킬 수 있는데, 이것을 **커널 기법(Kernel Trick)**이라 하며 실제로 데이터를 확장하지 않고도 확장된 특성에 대한 데이터 포인트들의 거리를 계산한다.
- 커널 기법에는 앞서 이야기한 특성을 추가하여 고차원에서 모델을 학습시키는 다항식 커널과 차원이 무한한 특성 공간에 매핑하는 가우시안 커널(RBF 커널) 두 가지가 있다.

서포트 벡터 머신 평가

- 장점
  - 커널 서포트 벡터 머신은 강력한 모델이며 다양한 데이터 세트에서 잘 작동한다.
  - 데이터의 특성이 몇 개 되지 않더라도 복잡한 결정 경계를 만들 수 있다.
  - 저차원과 고차원의 데이터(특성이 적을 때와 많을 때)에서 잘 작동한다.

- 단점
  - 샘플이 많을수록 속도가 느리고 메모리 할당이 크며, 성능이 잘 나오지 않는다.
  - 데이터 전처리와 매개변수 설정에 따라 정확도가 다르기 때문에 신경을 많이 써야한다.
  - 예측이 어떻게 결정되었는지 이해하기 어렵고 모델을 분석하기도 어렵다.