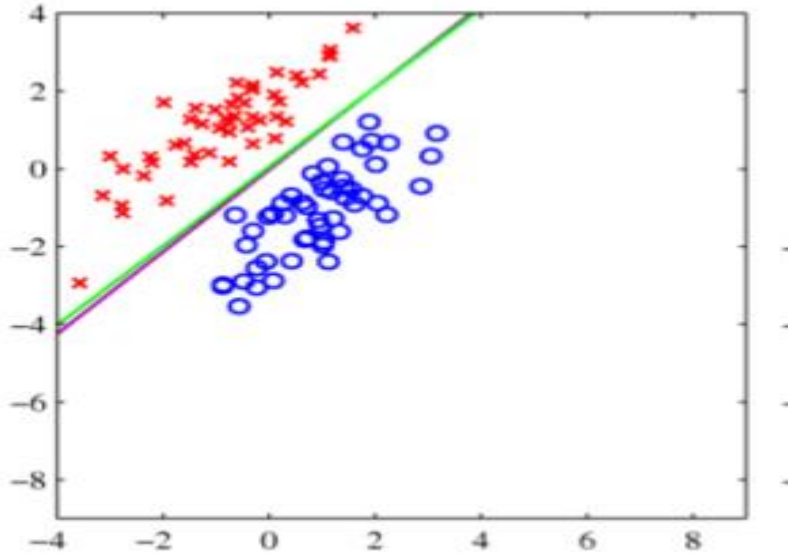
Algorithm Study

2. Linear Classification (선형 분류)

- 선형 분류 모델은 특성에 대한 가중치 합을 사용하며 예측한 값을 임계치 0과 비교한다. (이진분류)

$$\hat{y} = w[i] * x[i] + b > 0$$

- 일반화된 이진분류를 위한 함수는 위의 수식과 같이 표현된다.
- 함수에서 계산된 값이 0보다 작고 크고에 따라 클래스를 구분한다.
- 분류형 선형 모델에서는 결정 경계가 입력의 선형 함수이다. (선, 평면, 초평면을 이용해 클래스 구분)
- 선형 모델 학습 시, 특정 계수 절편의 조합이 Train data에 적합한지 측정하는 방법과 사용 가능한 규제가 있는지, 있다면 어떤 방식인지의 방법으로 구분할 수 있다.



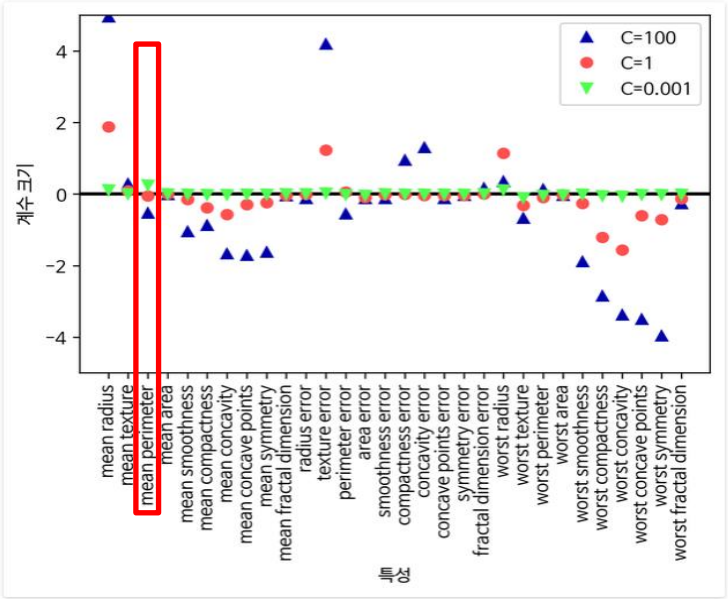
- 알고리즘 마다 훈련 세트 학습 정도를 측정하는 방법이 상이 하기 때문에 오분류의 수를 최소화하기 위한 w와 b를 조정하는 것은 불가능하다.
- 많은 어플리케이션에서 손실함수에 대한 차이는 중요하지 않다.

Algorithm Study

2. Linear Classification (선형 분류)

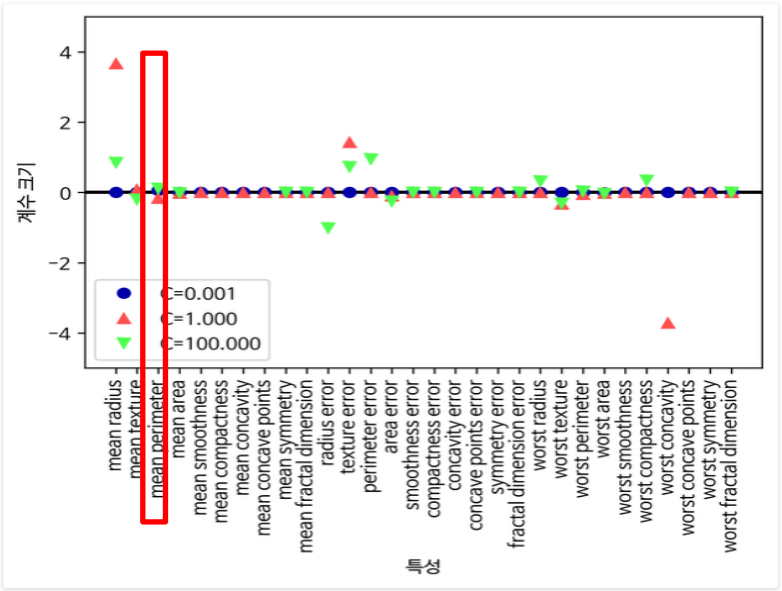
1) 로지스틱 회귀(Logistic Regression) / 서포트 벡터 머신 (svm.LinearSVC)

- 로지스틱회귀와 SVC에서 규제의 강도를 결정하는 매개변수는 C
  (C가 높아지면 규제가 감소) -> c값이 높으면 훈련 세트에 최대한 맞추고, 낮으면 w가 0에 가까워진다.
- 회귀와 비슷하게 분류에서의 선형 모델은 낮은 차원의 데이터에서는 결정 경계가 직선 혹은 평면이어서 제한적인 것처럼 보이지만, 고차원에서는 매우 강력해지며 특성이 많아지면 과대적합 방지의 중요성이 커짐.
- Logistic Regression은 기본적으로 L2 규제를 적용
- C값에 따라 각 피쳐들은 양수, 음수가 되기도 하며 양성, 악성 피쳐의 신호가 될 수 있다.
  -> 따라서 선형 모델의 피쳐는 의심하고 조심히 해석해야 한다.
- 더 이해하기 쉬운 모델을 원한다면(제한된 수의 피쳐 사용) L1규제를 이용하는 것이 좋다.



L2

L1



Algorithm Study

2. Linear Classification (선형 분류)
2) 다중 클래스 분류용 선형 모델

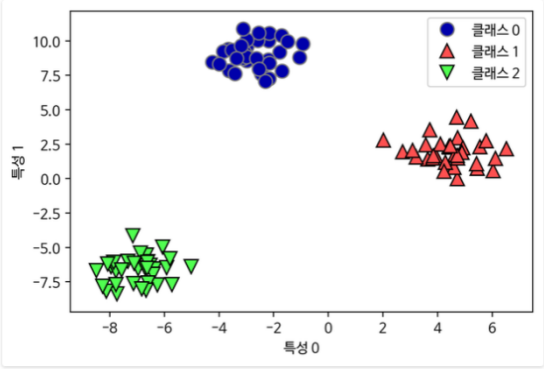
- 로지스틱 회귀(소프트맥스 함수를 지원)를 제외하고 많은 선형 분류 모델은 이진 분류만을 지원한다.
- 이진 분류 알고리즘을 다중 클래스 분류 알고리즘으로 확장하는 보편적 기법은 일대다 방법
  (각 클래스를 다른 모든 클래스와 구분하도록 이진 분류모델 학습 후 가장 높은 점수를 내는 분류기를 선택)

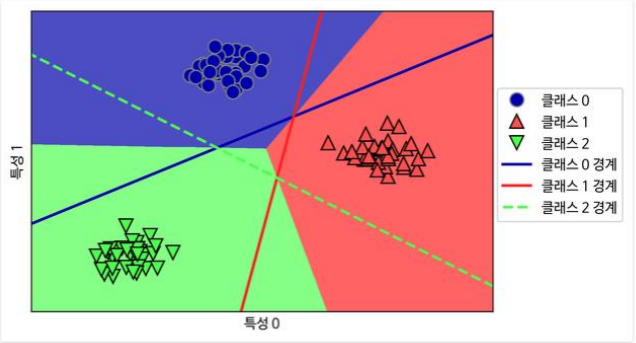
$$w[0] * x[0] + w[1] * x[1] + w[2] * x[2] + w[p] * x[p] + b$$

- 각 클래스의 w와 b를 하나씩 갖고, 다음 공식의 결괏값이 가장 높은 클래스가 해당 데이터의 클래스 레이블로 할당 (예측방법은 동일)

$$\Pr(Y_i = c) = \frac{e^{W_c * X_i}}{\sum_{k=1}^{K} e^{W_k * X_i}}$$

- 다중 클래스 로지스틱 회귀를 위한 공식. (i번째 데이터 포인트 Xi의 출력 Yi 클래스가 c일 확률 $\Pr(Y_i = c)$ 는 K개의 클래스에 대한 각각의 계수 W를 데이터 포인트에 곱하여 지수함수를 적용한 합)
- 보통 소프트 맥수 함수의 표현에서 b는 w에 포함 되어있는 것으로 나타냄.





Algorithm Study

2. Linear Classification (선형 분류)
2) 다중 클래스 분류용 선형 모델

- 선형 모델의 주요 매개변수는 회귀 모델에서는 alpha, 분류 모델에서는 C이다.
- Alpha값이 클수록, C 값이 작을 수록 모델이 단순해진다.
- 이러한 파라미터값은 로그 스케일로 최적치를 정하고 L1, L2규제 중 어떤 것을 사용할지 정해야 한다.
  -로그 스케일(자릿 수가 바뀌게 10배씩 변경 0.01, 0.1, 1, 10)
  -중요한 특성이 많지 않다면 L1규제 많으면 L2규제
- 선형 모델은 학습 속도, 예측 모두 빠르고 매우 큰 데이터 셋, 희소 데이터 셋에서도 잘 작동한다.
- 선형 모델의 대용량 처리 버전으로 구현된 확률적 경사 하강법(Stochastic Gradient Descent) 이용 가능.

Algorithm Study

2. Linear Classification (선형 분류)
3) 나이브 베이즈 분류기(Naïve Bayes)

- 일반적인 분류모델 (Logistc, SVC) 보다 훈련 속도가 빠르지만, 일반화 성능이 뒤쳐진다.

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

- Scikit-learn에 구현된 NB는 GaussianNB(연속적 데이터 적용-고차원), BernoulliNB(이진데이터), MultinomialNB(특성이 어떤 지를 헤아린 정수 카운트 데이터 – 예로 문장에 나타난 단어 횟수)가 있다.

- BernoulliNB, MultinomialNB(특성이 많은) 는 주로 텍스트 데이터를 분류할때 이용.
  ->예측 공식은 선형 모델과 형태가 같지만 coef_(계수)는 w가 아니다.
  -> coef_ = 특성 카운트 수를 로그 변환한 형태 / intercept_ = 클래스 카운트 수를 로그 변환 한 것.
  -> 예측할 때, 선형 함수와 같이 데이터 포인트에 coef_를 곱하고 intercept_를 더하여 클래스에 속할 확률 계산

- GaussianNB는 클래스별로 각 특성이 표준편차와 평균을 저장, MultinomialNB는 클래스별로 특성의 평균을 계산
  ->예측시, 데이터 포인트를 클래스의 통계 값과 비교해 가장 잘 맞는 클래스를 예측값으로 한다.

- Alpha 매개변수를 가지고 있으며 모든 특성에 양의 값을 가진 가상의 데이터 포인트를 alpha 개수 만큼 추가
  -> 통계 데이터를 완만하게 만들어주고 모델의 복잡도를 낮춘다.
  -> 성능 변동은 크지 않아 성능 향상에 기여하는 부분은 적지만, 정확도는 높일 수 있다.

Algorithm Study

2. Linear Classification (선형 분류)
4) 결정 트리 (Decision Tree)

- 분류와 회귀 문제에 널리 사용하는 모델로 기본적으로 결정에 다다르기 위해 예/아니오 질문을 이어나가며 학습.
- 노드는 질문이나 정답을 담은 상자(마지막 노드는 리프, 맨 위 노드는 루트노드, 엣지는 질문의 답과 다음 질문을 연결)
- 보통 연속된 특성으로 구성된 데이터에 적용할 테스트는 '특성 i는 값 a보다 큰가'의 형태를 띈다.
- 결정 트리는 계층적으로 영역을 분할해가는 알고리즘이라고 할 수 있다.
  ->데이터를 분할하는 것은 각 분할된 영역이 한 개의 타깃값과 하나의 회귀 분석결과를 가질때 까지 반복
  ->타깃 하나로만 이뤄진 리프 노드를 순수노드라고 한다.
- 루트 노드에서 시작해 테스트의 결과에 따라 왼쪽 또는 오른쪽으로 트리를 탐색해나가는 식으로 영역 탐색
- 회귀 문제에도 예측을 위해 각 노드의 테스트 결과에 따라 트리를 탐색하고, 새로운 데이터 포인트에 해당하는 리프노드를 탐색하고, 탐색한 리프노드의 훈련 데이터 평균값이 출력이 된다.

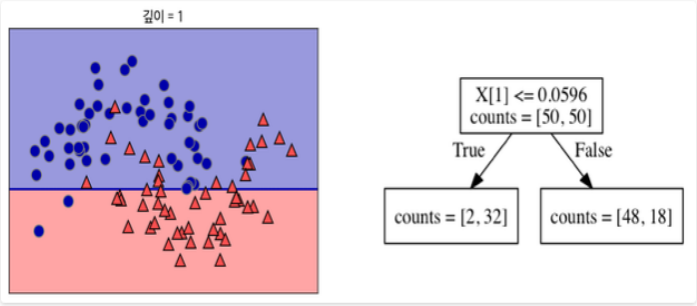


그림 2-24 깊이 1인 결정 트리(오른쪽)가 만든 결정 경계(왼쪽)

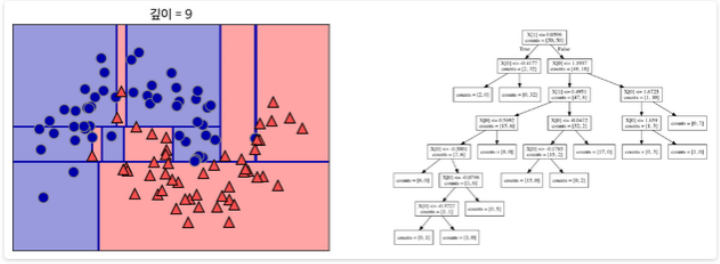


그림 2-26 깊이 9인 결정 트리의 일부(오른쪽)와 이 트리가 만든 결정 경계(왼쪽)(전체 트리는 너무 커서 일부만 표시했습니다.) [2]

Algorithm Study

2. Linear Classification (선형 분류)
4) 결정 트리 (Decision Tree)

- 모든 리프 노드가 순수 노드가 될 때까지 진행하면, 모델이 복잡해지고 과대적합이 발생된다.
  (결정 경계가 이상치 하나에 민감하다.)

- 과대적합을 막기 위해 트리 생성을 일찍 중단하는 사전 가지치기, 트리를 만든 후 데이터 포인트가 적은 노드를 삭제하거나 병합하는 사후 가지치기가 있다.
  -> 사전 가지치기는 트리의 최대 깊이, 리프의 최대 개수 제한, 분할 위한 포인트의 최소 개수 지정
  (max_depth, max_leaf_nodes, min_sample_leaf 지정)
  -> Scikit-learn에서는 사전 가지치기만 지원.

- 트리 모듈의 export_graphviz 함수를 이용해 트리를 시각화 가능.
  -> 트리의 알고리즘 예측이 어떻게 이뤄지는지 쉽게 파악하고, 양성, 악성샘플 파악 가능.

- 특성 중요도(tree.feature_importances_ 함수)를 통해 결정 트리 모델에서 각 특성이 얼마나 중요한지 확인 가능.

- 회귀 분석을 실시할 때에는 훈련 데이터 범위 밖의 포인트에 대한 예측은 불가능하다.
  (마지막 포인트를 이용해 예측하는 것이 끝 – 모든 트리 기반 모델의 공통된 단점)

- 결정 트리는 모델 시각화를 통해 이해하기 쉽고, 데이터 스케일에 구애 받지 않는다.
  -> 정규화나 표준화 같은 전처리 과정이 필요없다.

- 사전 가지치기를 이용하여도 과대적합되는 경향이 있어 일반화 성능이 좋지 않다.