핸즈온 머신러닝

Chapter 3 : MNIST
~예시 싹 뺀 요약 노트~

MNIST 흐름 보기

1. 데이터 불러오고 살피기

2. 이진 분류기 훈련

   SGD Classifier

3. 성능 측정

   3.1 교차 검증을 사용한 정확도 측정
   3.2 오차 행렬
   3.3 정밀도와 재현율 (그리고 트레이드오프)
   3.4 ROC 곡선

4. 다중 분류

   OvR
   OvO
   Multinominal

5. 에러 분석

6. 다중 레이블 분류

7. 다중 출력 분류

1. 데이터 살펴보기

Mnist란 10만개의 정수 이미지로 구성되어있으며, 픽셀 데이터, 2차원 배열이다.

```
# 훈련셋 확인해서 스플릿~
X_train, X_test, y_train, y_test = X[:60000], X[60000:], y[:60000], y[60000:]

# 훈련 세트는 이미 섞여 있기 때문에 모든 교차 검증 폴드가 비슷해진다.
# 어떤 학습 알고리즘은 훈련 샘플 순서에 민감해서 많은 비슷한 샘플이 있을 경우 성능이 나빠진다.
# 데이터셋을 섞으면 이런 문제를 방지할 수 있다.
# 그러나 주식 가격 같은 시계열 데이터는 오히려 섞지 않는 것이 나을 수 있다.
```

2. 이진 분류기 훈련

숫자 '5' 가 있을 때, 5와 5가 아닌 두 개의 숫자 클래스를 구분할 수 있게 하는 것이 이진 분류기의 예이다. 즉, 5만 True 이고, 나머지는 False 값을 갖게 하는 것이다.

```
y_train_5 = (y_train == 5)      # 5만 true이고 나머지는 false (이진분류기 만들기~)
y_test_5 = (y_test == 5)
```

SGD Classifier
SGD : 확률적 경사 하강법 (Stochastic Gradient Descent)

- 매우 큰 데이터셋도 효율적으로 처리
- 한 번에 하나씩, 훈련 샘플의 독립적 처리
- 온라인 학습에 좋다.

핸즈온 머신러닝에서는 이 분류기로 시작한다.

3. 성능 측정

그래서 정확도를 분류기의 성능 지표로 선호하지 않아요. 불균형한 데이터셋을 다룰 때는 딱히 좋은 값이 아니어도 90% 등의 좋게 보이는 값이 나오니까요.

분류기를 사용했으니, 분류기의 성능을 측정해 보아야겠죠! 분류기를 평가하는 것은 회귀 모델보다 어렵기 때문에 상당한 분량 주의... 사용할 수 있는 성능 지표가 많습니다! 소제목으로 나누어서 살펴봅시다.

3.3.1 교차 검증을 사용한 정확도 측정!

StratifiedKFold:

- 클래스별 비율이 유지되도록 폴드를 만들기 위해 계층적 샘플링 수행
- 분류기 객체를 복제해서 훈련 폴드로 훈련 시키고
- 테스트 폴드로 예측을 만드는!
- 그 다음엔 예측(올바른)수를 세서 정확한 예측의 비율을 출력한다.

Cross_val_score:

- Scoring='accuracy' 를 사용해서 모든 교차 검증 폴드에 대한 정확도를 출력합니다.
- 95%

BaseEstimator

- 분류기를 베이스 에스티메이터를 사용해 만들고 (다 5가 아니라고 분류하는 분류기 만들어)
- Cross_val_score를 활용해 다시 정확도를 측정하면
- 90% (즉, 5는 전체 데이터의 10% 뿐....)

3. 성능 측정

3.3.2 오차 행렬!

: 클래스 A 의 샘플이 클래스 B로 분류된 횟수를 세는 것

분류기의 성능을 평가하는 더 좋은 방법이 오차 행렬을 조사하는 것이다. 분류기가 숫자 5를 3으로 잘못 분류한 횟수를 알고 싶다면, 오차 행렬의 5행 3열을 보면 되는 것...

Q. 오차 행렬을 만들려면?
A. 먼저 실제 타깃과 비교할 수 잇는 예측값을 만들어야 한다.
cross_val_predict() 를 사용할 수 있다.
테스트 세트로 예측값을 만들 수는 있으나 이 단계에서 사용은 금물!
프로젝트의 맨 마지막에 사용해야 하니까.

cross_val_predict()
: k-겹 교차 검증을 수행하나, 평가 점수를 반환하지 않고 테스트폴드에서 얻은 예측을 반환!
즉, 이 친구가 교차 검증을 오차 행렬을 반환하는 것이다.

	예측	
실제	5 아닌 값 잘 분류	5 아닌 값 잘못분류
	5인 값 잘 분류	5 인 값 잘못분류

		실제 값 (5가 맞냐)	
		맞음	아님
분류 결과	맞음	진짜 양성	가짜 양성
	아님	가짜 음성	진짜 음성

잘 분류되었다면 우측 상단과 좌측 하단(가짜 양성과 가짜 음성) 의 FP, FN이 0일 것이다.

이 때, 5라고 예측 분류된 값 중에서 '진짜'는 얼마나 잘 분류되었는지를 '정밀도'라고 하며, 식은 아래와 같다.

정밀도 = TP/(TP+FP)

반면, 실제 값이 5인 '진짜 5'들 중에서 5라고 잘 분류된 값이 얼마나 있는지는 '재현율' 이라고 하며, 식은 아래와 같다. (진짜 양성 비율, 정답률)

재현율 = TP/(TP+FN)

3.3.3 정밀도와 재현율!

: 클래스 A 의 샘플이 클래스 B로 분류된 횟수를 세는 것

분류기의 성능을 평가하는 더 좋은 방법이 오차 행렬을 조사하는 것이다. 분류기가 숫자 5를 3으로 잘못 분류한 횟수를 알고 싶다면, 오차 행렬의 5행 3열을 보면 되는 것...

Q. 오차 행렬을 만들려면?
A. 먼저 실제 타깃과 비교할 수 잇는 예측값을 만들어야 한다.

정밀도 : 5라고 예측 분류된 값 중에서 '진짜'는 얼마나 잘 분류되었는지
재현율 : 실제 값이 5인 '진짜 5'들 중에서 5라고 잘 분류된 값이 얼마나 있는지

3. 성능 측정

3.3.3 정밀도와 재현율!

: 클래스 A 의 샘플이 클래스 B로 분류된 횟수를 세는 것

① F1 점수: 정밀도와 재현율의 조화 평균

$$F1 = \frac{2}{\frac{1}{\text{정밀도}} + \frac{1}{\text{재현율}}} = 2\frac{\text{정밀도} * \text{재현율}}{\text{정밀도} + \text{재현율}} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

이렇게 복잡하게 구할 필요 없이 sklearn.metrics 에서 f1_score() 함수를 호출하면 좀 더 쉽다.

F1의 값은 정밀도와 재현율이 비슷한 분류기에서 값이 높다.
상황에 따라 더욱 중요한 값에 무게를 두고 분류기를 훈련시키면 된다.
즉, 정밀도와 재현율을 모두 얻을 수는 없으며, 이를 정밀도/재현율 트레이드 오프라고 한다.

② 정밀도/재현율 트레이드 오프

SGD Classifier가 어떻게 분류하는지 보면서 이해할 수 있다.
: **결정 함수**를 사용하여 각 샘플의 점수를 계산

- 샘플 점수 <= 임곗값 : 음성 클래스
- 샘플 점수 > 임곗값 : 양성 클래스

3.3.4 ROC 곡선!

: 수신기 조작 특성 곡선, 이진 분류에서 널리 사용.
: 정밀도/재현율 곡선과 비슷하나, 거짓양성비율에 대항 진짜양성비율의 곡선 (FPR/TPR)

- FPR : 양성으로 잘못 분류된 음성 샘플의 비율
- TPR : 재현율, 민감도
- TNR : 특이도, 진짜 음성 비율

1-TNR = FPR

① roc_curve()

여러 임곗값에서 TPR과 FPR 계산

② 맷플롯립으로 TPR에 대한 FPR곡선 나타내기

트레이드 오프: 재현율이 높을 수록 분류기가 만드는 거짓 양성 늘어나.

③ RandomForestClassifier를 훈련시켜, SGDClassifier의 ROC와 비교

ROC, AUC 점수 비교할 것.

- predict_proba() : 아래 표와 같은 배열을 반환 (어떤 이미지가 5일 확률 70%)

RandomForestClassifier

	클래스
샘플	샘플이 클래스에 속할 확률

- roc_curve() : 레이블과 점수를 기대하지만, 점수 대신 클래스 확률 전달도 가능

SGDClassifier

4. 다중 분류

이중 분류 : 두 개의 클래스 구별
다중 분류 : 둘 이상의 클래스 구별

- OvA : 이진 분류기 여러 개를 훈련시켜 클래스가 n개인 분류 시스템 만들기
- OvO : 각 조합 마다 이진 분류기를 훈련 (구별할 두 클래스 샘플만 필요)
  훈련 세트 크기에 민감한 알고리즘은 작은 훈련 세트에서 많은 분류기를 훈련시키는 얘 선호

① decision_function()
샘플 당 여러 개의 점수 반환 (클래스마다 1개 씩)

② OneVsOneClassifier, OnevsRestClassifier
원래는 자동 매치를 해주지만 방법을 강제하고 싶을 때.

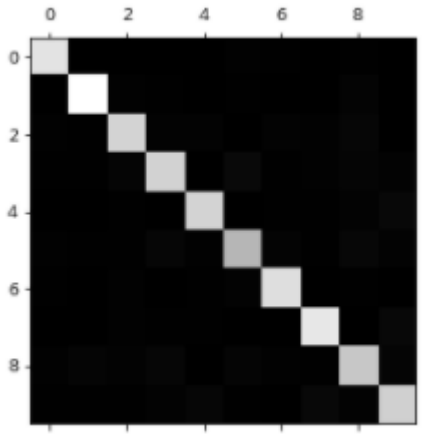
③ cross_val_score()
분류기 사용 시 일반적으로 교차 검증 사용, SGDClassifier의 정확도 평가.

④ 입력 스케일 조정
-> 정확도 높일 수 있다.

5. 에러 분석

① cross_val_predict(), confusion_matrix()
matshow()함수로 출력한 것을 이미지로 표현해보자

	예측 클래스
실제 클래스	

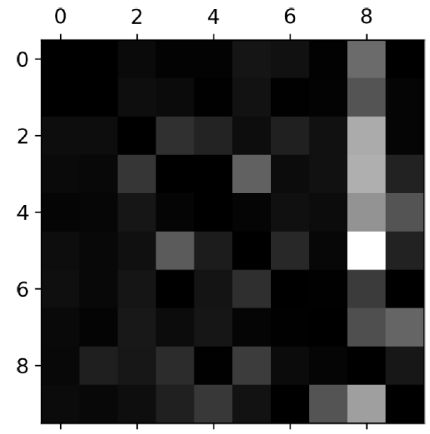


대부분의 이미지가 올바르게 분류 (주대각선)

에러 부분에 초점 ->

$$\frac{\text{오차행렬 값}}{\text{대응 클래스 이미지 개수}}$$

주대각선은 0으로 ->



5는 8로 많이 잘못 분류 (열)
8은 8로 잘 분류 (행)

② 개개의 에러 분석
분류기 뭐하고, 왜 잘못되었는지 알 수 있으나, 시간이 오래걸리고 더 복잡

③ 오차 행렬 분석
분류기 성능 향상 방안에 대한 통찰.
예) 8로 잘못 분류 되는거 개선 필요: 동심원 세기, 데이터 더 많이 모으기 등