|  |  |  |  |
| --- | --- | --- | --- |
|  | 장점 | 단점 |  |
| K-NN | 이해하기 매우 쉬운 모델이다. 그리고 많이 조정하지 않아도 자주 좋은 성능을 발휘한다 | 훈련 세트가 매우 크면 (특성의 수나 샘플의 수가 클 경우) 예측이 느려진다.  이해하긴 쉽지만, 예측이 느리고 많은 특성을 처리하는 능력이 부족해 현업에서는 잘 쓰지 않는다. | 따라서 주로 k-NN의 이러한 단점을 없앤 선형 모델을 사용하곤 한다. |
| 선형모델 | 학습 속도가 빠르고 예측도 빠르다. 매우 큰 데이터셋과 희소한 데이터셋에도 잘 작동한다. 또한 예측이 어떻게 만들어 지는지 수식 등을 통해 비교적 쉽게 이해할 수 있다. | 하지만 계수의 값들이 왜 그런지 명확하지 않을 때가 종종 있다. 특히 데이터셋의 특성들이 서로 깊게 연관되어 있을 때 그러하다. | 분류에도 널리 사용된다.  가장 널리 알려진 두 개의 선형 분류 알고리즘은 Logistic Regression과 support vector machine이다.  많은 선형 분류 모델은 태생적으로 이진 분류만을 지원한다. 즉 다중 클래스를 지원하지 않는다.  선형 모델은 샘플에 비해 특성이 많을 때 잘 작동한다. 그러나 저차원의 데이터셋에서는 다른 모델들의  일반화 성능이 더 좋다. |
| 나이브 베이즈(Naive Bayes)분류기 | Logistic Regression과 support vector machine같은 선형 분류기보다 훈련 속도가 빠른 편  훈련과 예측 속도는 빠르며 훈련 과정을 이해하기 쉽다. 희소한 고차원 데이터에서 잘 작동하며 비교적 매개변수에 민감하지 않다. 선형 모델로 학습 시간이 너무 오래 걸리는 매우 큰 데이터셋에는 나이브 베이즈 모델을 시도해볼 만하다 | 대신 일반화 성능이 조금 뒤진다. | 나이브 베이즈(Naive Bayes)분류기는 선형 모델과 매우 유사하다. |
| 결정 트리(decision tree) | 트리를 시각화하면 알고리즘의 예측이 어떻게 이뤄지는지 잘 이해할 수 있으며 비전문가에게 머신러닝 알고리즘을 설명하기에 좋다.  만들어진 모델을 쉽게 시각화할 수 있어서 이해하기 쉽다. 그리고 데이터의 스케일에 구애받지 않는다. 각 특성이 개별적으로 처리되기 때문이다. 그래서 결정 트리에서는 특성의 정규화나 표준화 같은 전처리 과정이 필요 없다. | 트리의 깊이가 4정도만 되어도 트리는 매우 장황해진다. 트리가 더욱 깊어지면 한눈에 보기가 힘들어진다.  트리 모델은 훈련 데이터 밖의 새로운 데이터를 예측할 능력이 없다. 이는 모든 트리 기반 모델의 공통된 단점이다.  트리 모델은 시계열 데이터엔 잘 맞지 않는다.  사전 가지치기를 사용함에도 불구하고 과대적합되는 경향이 있어 일반화 성능이 좋지 않다. 이 대안으로 앙상블 방법을 사용한다. | 분류와 회귀 문제에 널리 사용하는 모델  기본적으로 결정 트리는 결정에 다다르기 위해 예/아니오 질문을 이어 나가면서 학습한다. |
| 앙상블(ensemble) |  |  | 앙상블(ensemble)은 여러 머신러닝 모델을 연결하여 더 강력한 모델을 만드는 기법이다. 머신러닝의 여러 앙상블 기법 중 랜덤 포레스트(random forest)와 그래디언트 부스팅(gradient boosting)이 분류와 회귀 문제의 다양한 데이터셋에서 효과적으로 입증되어 있다. 이 두 모델 모두 모델을 구성하는 기본 요소로 결정 트리를 사용한다. |
| 앙상블(ensemble) - 랜덤 포레스트(random forest) | 결정 트리의 주요 단점은 훈련 데이터에 과대적합되는 경향이 있다는 것이다. 랜덤 포레스트는 이 문제를 회피할 수 있는 방법이다.  아무런 매개변수 튜닝 없이도 선형 모델이나 단일 결정 트리보다 높은 정확도를 낸다. 랜덤 포레스트는 하이퍼 파라미터 조정 없이 기본 설정으로도 좋은 결과를 만들어줄 때가 많다.  회귀와 분류에 있어서 랜덤 포레스트는 현재 가장 널리 사용되는 머신러닝 알고리즘이다.  랜덤 포레스트는 성능이 매우 뛰어나고 매개변수 튜닝을 많이 하지 않아도 잘 작동하며 데이터의 스케일을 맞출 필요도 없다. 기본적으로 랜덤 포레스트는 단일 트리의 단점을 보완하고 장점은 그대로 가지고 있다.  대량의 데이터셋에서 랜덤 포레스트를 만들 때 다소 시간이 걸릴 수 있지만 CPU 코어가 많다면 손쉽게 병렬 처리할 수 있다. | 랜덤 포레스트는 텍스트 데이터 같이 매우 차원이 높고 희소한 데이터에는 잘 작동하지 않는다. 이런 데이터에는 선형 모델이 더 적합하다.  랜덤 포레스트는 선형 모델보다 많은 메모리를 사용하며 훈련과 예측이 느리다. 속도와 메모리 사용에 제약이 있는 애플리케이션이라면 선형 모델이 적합할 수 있다.  더 많은 트리를 평균하면 과대적합을 줄여 더 안정적인 모델을 만든다. 하지만 더 많은 트리는 더 많은 메모리와 긴 훈련 시간으로 이어진다. | 랜덤 포레스트는 기본적으로 조금씩 다른 여러 결정 트리의 묶음이다. 랜덤 포레스트의 아이디어는 각 트리는 비교적 예측을 잘 할 수 있지만 데이터의 일부에 과대적합하는 경향을 가진다는 것에 기초한다. 예컨대 잘 작동하되 서로 다른 방향으로 과대적합된 트리를 많이 만들면 그 결과를 평균냄으로써 과대적합된 양을 줄일 수 있다. 이렇게 하면 트리 모델의 예측 성능이 유지되면서 과대적합이 줄어드는 것이 수학적으로 증명되었다. 이런 전략을 구현하려면 결정 트리를 많이 만들어야 한다. 각각의 트리는 타깃 예측을 잘 해야 하고 다른 트리와는 구별되어야 한다. 또한 랜덤 포레스트라는 이름에서 알 수 있듯이 트리 생성 시 무작위성을 주입한다 |
| 앙상블(ensemble) - 그래디언트 부스팅 회귀 트리 | 그래디언트 부스팅 트리는 보통 하나에서 다섯 정도의 깊지 않은 트리를 사용하므로 메모리를 적게 사용하고 예측도 빠르다.  그래디언트 부스팅 트리는 머신러닝 경연 대회에서 우승을 많이 차지하였고 업계에서도 널리 사용된다.  랜덤 포레스트보다는 배개변수 설정에 조금 더 민감하지만 잘 조정하면 더 높은 정확도를 제공한다.  그래디언트 부스팅 결정 트리는 지도 학습에서 가장 강력하고 널리 사용되는 모델 중 하나이다.  다른 트리 기반 모델처럼 특성의 스케일을 조정하지 않아도 되고 이진 특성이 연속적인 특성에서도 잘 동작한다. | 랜덤 포레스트보다는 배개변수 설정에 조금 더 민감하다.  가장 큰 단점은 매개변수를 잘 조정해야 한다는 것(--> 그래서 grid search 방식을 활용하여 적합한 매개변수를 찾아낸 것이 이 논문의 기여점이다. )과 훈련 시간이 길다는 것이다 | 그래디언트 부스팅 회귀 트리는 여러 개의 결정 트리를 묶어 강력한 모델을 만드는 또 다른 앙상블 방법이다. 이름이 회귀지만 이 모델은 회귀와 분류 모두에 사용할 수 있다. 랜덤 포레스트와 달리 그래디언트 부스팅은 이전 트리의 오차를 보완하는 방식으로 순차적으로 트리를 만든다. 기본적으로 그래디언트 부스팅 트리에는 무작위성이 없다. 대신 강력한 사전 가지치기가 사용된다.  그래디언트 부스팅의 근본 아이디어는 이런 얕은 트리(약한 학습기)를 많이 연결하는 것이다. 각각의 트리는 데이터의 일부에 대해서만 예측을 잘 수행할 수 있어서 트리가 많이 추가될수록 성능이 좋아진다 |
|  |  |  | 비슷한 종류의 데이터에서 그래디언트 부스팅과 랜덤 포레스트 둘 다 잘 작동한다. 하지만 보통 더 안정적인 랜덤 포레스트를 먼저 적용하곤 한다. 랜덤 포레스트가 잘 작동하더라도 예측 시간이 중요하거나 머신러닝 모델에서 마지막 성능까지 쥐어짜야 할 때 그래디언트 부스팅을 사용하면 도움이 된다. |
| 커널 기법(kernel trick) |  |  | 직선과 초평면은 유연하지 못하여(구부러지지 않아서) 저차원 데이터셋에서는 선형 모델에 매우 제한적이다. 선형 모델을 유연하게 만드는 한 가지 방법은 특성끼리 곱하거나 특성을 거듭 제곱하는 식으로 새로운 특성을 추가하는 것이다.  분류를 위한 선형 모델은 직선으로만 데이터 포인트를 나눌 수 있다.  데이터셋에 비선형 특성을 추가하여 선형 모델을 강력하게 만들 수 있다. 하지만 많은 경우 어떤 특성을 추가해야 할지 모르고 특성을 많이 추가하면 연산 비용은 커진다. 다행히 커널 기법(kernel trick)을 이용하면 실제로 데이터를 확장하지 않고 확장된 특성에 대한 데이터 포인트들의 거리(더 정확히는 스칼라 곱)을 계산한다. |
| 커널 기법(kernel trick) - SVM | 데이터의 특성이 몇 개 안 되더라도 작동한다.  저차원과 고차원의 데이터(즉 특성이 적을 때와 많을 때)에 모두 잘 작동 | 샘플이 많을 때는 잘 맞지 않는다. 10,000개의 샘플 정도면 SVM 모델이 잘 작동하겠지만 그 그 이상의 데이터셋에서는 속도와 메모리 관점에서 도전적인 과제이다.  데이터 전처리와 매개변수 설정에 신경을 많이 써야 한다. 그래서 요즘엔 사람들 대부분이 랜덤 포레스트나 그래디언트 부스팅 같은 (전처리가 거의 또는 전혀 필요 없는) 트리 기반 모델을 애플리케이션에 많이 사용한다.  더군다나 SVM 모델은 분석하기도 어렵다. 예측이 어떻게 결정되었는지 이해하기 어렵고 비전문가에게 모델을 설명하기가 난해하다. | SVM은 데이터의 특성이 몇 개 안 되더라도 복잡한 결정 경계를 만들 수 있다 |
|  |  | 하지만 모든 특성이 비슷한 단위이고 스케일이 비슷하면 SVM을 시도해볼만하기 때문에, 암호화폐 가격 데이터에 SVM을 적용하고자 하는 시도 또한 유의미하다고 생각한다. |  |