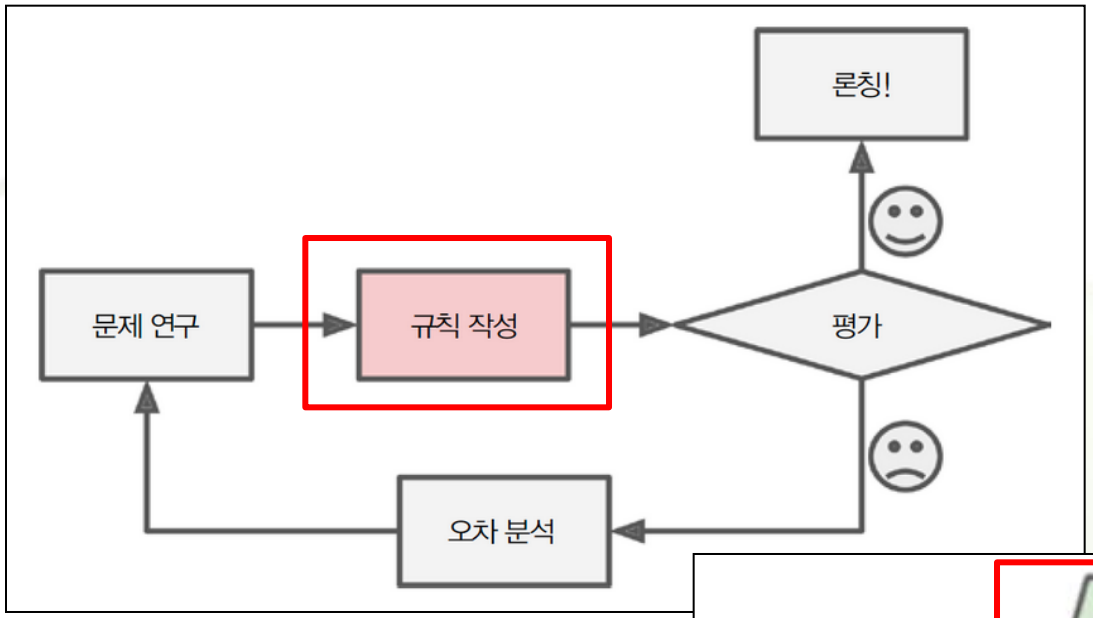
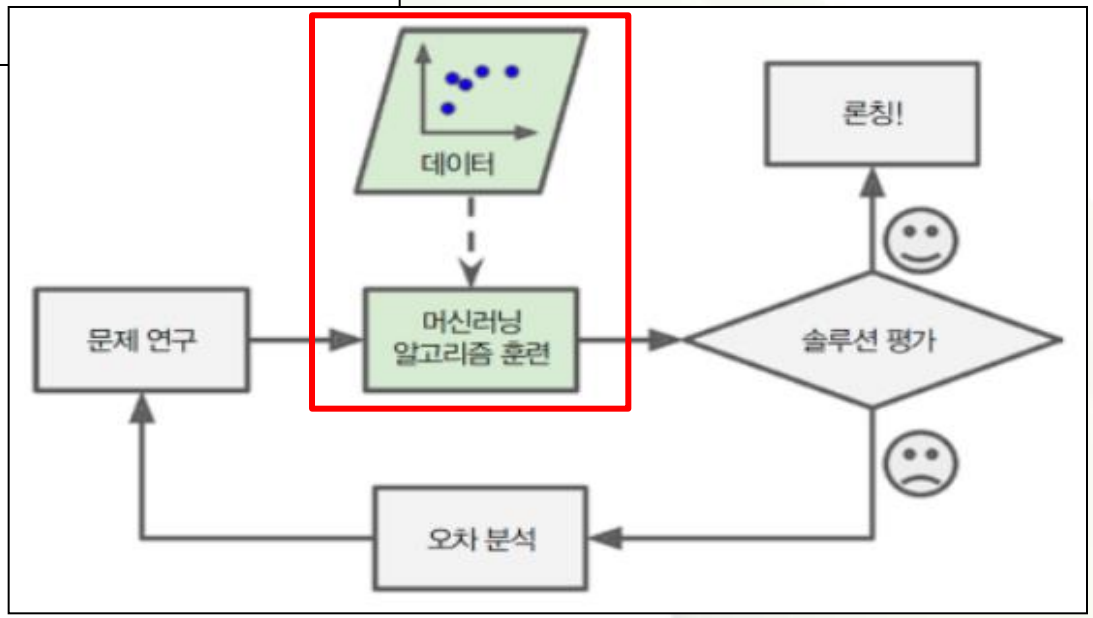
Introduction to Machine Learning

일반적인 프로그램과 머신 러닝



일반적인 프로그램



머신러닝 프로그램

머신 러닝 시스템 종류

- 학습하는 동안의 감독 형태나 정보량 기준

학습방법	설명
지도학습	▪ 알고리즘에 주입하는 데이터에 레이블이라는 답 포함 • 범주에 대한 분류와 수치를 예측하는 회귀
비지도학습	▪ 알고리즘에 주입하는 데이터에 레이블 없음
강화학습	▪ 환경을 관찰해서 행동을 실행하고 그 결과로 보상 (또는 벌칙) 부여 → 가장 큰 보상을 얻기 위해 [정책]이라는 최상의 전략 학습

머신 러닝 시스템 종류

- 점진적 학습 여부 기준

학습방법	설명
배치학습	▪ 점진적으로 학습할 수 없고 가용한 데이터를 모두 사용 ▪ 시간과 자원 소모량이 많아서 오프라인으로 학습 ▪ 먼저 시스템을 훈련시키고 제품 시스템에 적용하면 더 이상의 학습은 없음 ▪ 새로운 데이터를 적용하려면 전체 데이터로 다시 학습 후 적용
온라인학습	▪ 데이터를 순차적으로 한 개씩 또는 작은 묶음 단위로 주입하여 학습 ▪ 새로운 데이터를 즉시 적용할 수 있음

- 일반화 방법 기준

학습방법	설명
사례 기반 학습	▪ 시스템이 사례를 기억함으로써 학습 → 유사도 측정을 사용해서 새로운 데이터에 일반화
모델 기반 학습	▪ 샘플들의 모델을 만들어 예측에 사용

머신 러닝 알고리즘

- 주요 머신 러닝 알고리즘

학습방법	주요 알고리즘
지도학습	k-Nearest Neighbors, Linear Regression, Logistic Regression, Support Vector Machine (SVM), Decision Tree & Random Forests, Neural Networks
비지도학습	[Clustring] k-Means, Hierachical Cluster Analysis, Expectation Maximization [Visualization & Dimensionality Reduction] Principal Component Analysis (PCA), Kernel PCA, Locally-Linear Embedding, t-Distributed Stochastic Neighbor Embedding (t-SNE) [Association Rule Learning] Apriori, Eclat

Supervised Learning

분류와 회귀

- 지도학습의 두 종류.

- 분류
  - » 미리 정의된 가능성 있는 여러 클래스 레이블 중 하나를 예측하는 것
  - » 두 개의 클래스로 분류하는 이진 분류와 셋 이상의 클래스로 분류하는 다중 분류

- 회귀
  - » 연속적인 숫자 또는 부동소수점(실수) 데이터를 예측하는 것

- 출력 값의 연속성 여부가 두 기법을 구분하는 중요한 기준
  - » 일반적으로 연속성이 있으면 회귀, 없으면 분류
  - » 양적 데이터는 회귀, 범주형 데이터는 분류

일반화, 과대적합, 과소적합

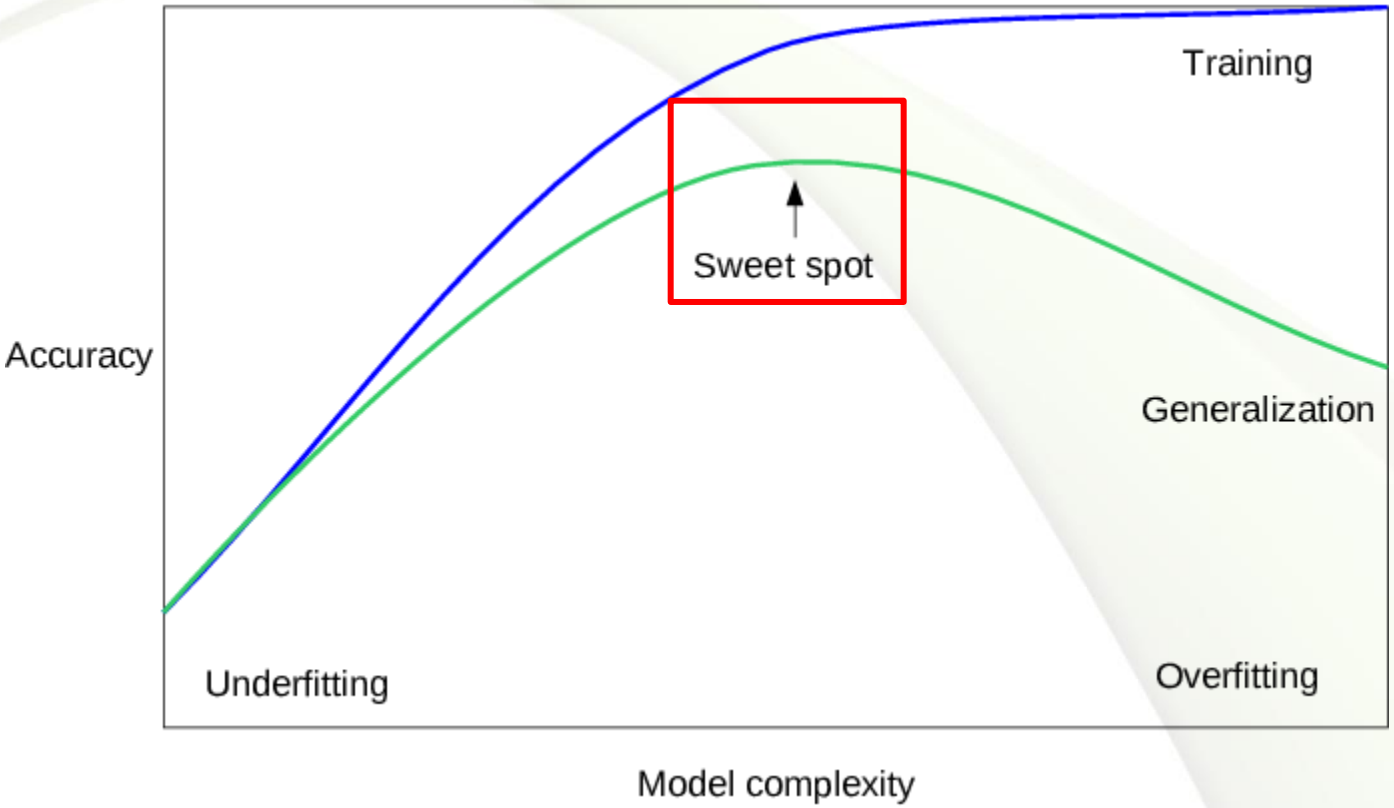
- 훈련 세트에서 테스트 세트로 일반화
  - » 모델이 처음 보는 데이터에 대해 정확하게 예측할 수 있게 되는 것
  - » 모델을 만들 때 가능한 정확하게 일반화하도록 구현해야 함

- 과대적합 (Overfitting)
  - » 모델이 훈련 세트에 너무 가깝게 맞춰져서 새로운 데이터에 일반화되기 어려운 경우
  - » 훈련 데이터는 잘 설명하지만 새로운 데이터에 대한 예측 정확도가 낮음

- 과소적합 (Underfitting)
  - » 모델을 지나치게 단순화해서 훈련 데이터와 테스트 데이터 모두에서 예측 정확도가 낮음

일반화, 과대적합, 과소적합

- 일반화 성능을 최대화 하는 모델을 찾는 것이 데이터 분석의 목표

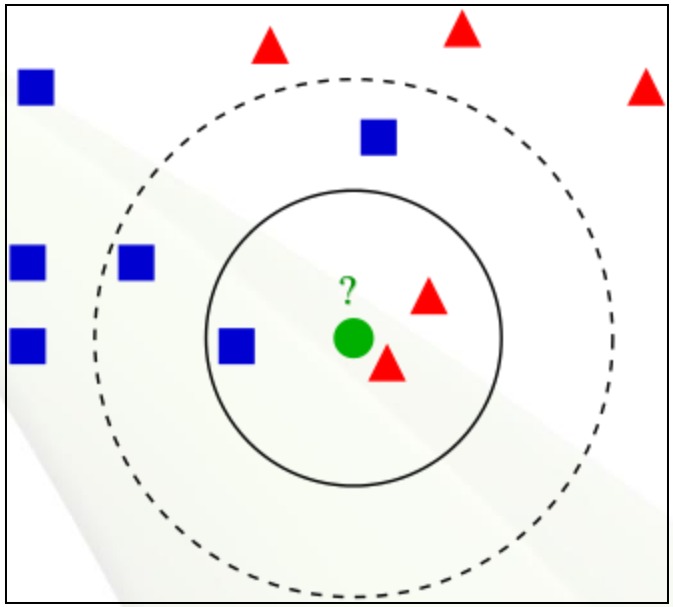


- 일반적으로 데이터가 많으면 다양성을 강화하기 때문에 큰 데이터 셋을 사용하면 과대적합 없이 복잡한 모델을 만드는 것 가능

KNN (K Nearest Neighbors)

- 가장 간단한 머신러닝 알고리즘으로 분류 및 회귀에 사용

- 새로운 데이터가 주어졌을 때 기존 데이터 가운데 가장 가까운 k개 이웃의 정보로 새로운 데이터를 예측하는 방법
  - 분류일 경우 이웃 데이터의 분류가 예측 값, 회귀일 경우 이웃 데이터의 종속 변수의 평균이 예측 값



- 게으른 모델
  - 모델을 별도로 구축하지 않고 새로운 데이터가 발생했을 때 거리를 계산하고 예측

KNN (K Nearest Neighbor)

- 이웃 데이터와의 거리 측정
  - » 유클리디안, 맨하탄 등 다양한 거리 측정 방법 사용
  - » 문제의 복잡도, 데이터 타입 등에 따라 선택
  - » 가장 흔하게 사용되는 거리 척도는 유클리드 거리
  - » 거리 측정 전에 반드시 변수 정규화 필요

- 탐색할 이웃 수 (K)
  - » k가 작을 경우 데이터의 지역적 특성을 과도하게 반영 (overfitting)
  - » k가 클 경우 과도하게 정규화 (underfitting)
  - » 최적의 k값을 찾기 위해 k값을 작은 값에서 시작해서 큰 값으로 변경하면서 실험

KNN (K Nearest Neighbor)

- 주요 매개 변수
  - » 데이터 포인트 사이의 거리 측정 방법 → 주로 유클리디안 거리 방식 사용
  - » 이웃의 수 → 3개 또는 5개에서 잘 동작하지만 상황에 따라 조정 필요

- 장점
  - » 이해하기 쉽고 빠르게 만들 수 있음
  - » 세밀하게 조정하지 않아도 비교적 좋은 성능 발휘 → 더 복잡한 알고리즘을 적용하기 전에 시도해 볼 수 있는 시작점으로 유용

- 단점
  - » 특성 또는 샘플의 개수가 많으면 예측이 느리게 처리됨
  - » 특성 값이 대부분 0인 데이터 세트에서는 잘 동작하지 않음
  - » 이런 이유로 현업에서는 활용도 낮음

선형 회귀

- 가장 간단하고 오래된 회귀용 선형 알고리즘으로 최소 제곱법으로도 불림

- 예측과 훈련 세트에 있는 목적 변수 y 사이의 평균제곱오차(MSE)를 최소화하는 파라미터 w와 b를 추적
  - » 평균제곱오차 → (예측 값과 목적 변수 값의 차이)^2 / 데이터 개수

$$MSE = 1/n \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$$

- 매개변수가 없는 것이 장점이지만 이로 인해 모델의 복잡도를 제어할 방법도 없음

- 모델이 과대 적합된 경우 복잡도를 제어할 수 있는 모델 필요
  - » 기본 선형 회귀 대신 릿지 회귀와 라소 회귀 모델을 널리 사용

릿지 회귀

- 최소 제곱법에서 사용한 예측 함수를 사용하지만 릿지 회귀에서 가중치 선택은 훈련 데이터를 잘 예측하는 것뿐만 아니라 추가 제약 조건을 만족시키기 위한 목적도 포함
  - » 가중치의 절대 값을 최대한 작게 만드는 것 → w의 모든 원소를 0에 가깝게 만드는 것 (기울기를 작게 만드는 것) → 모든 특성이 출력에 주는 영향을 최소화
  - » 이런 제약을 규제(regularization)라고 하며 릿지 회귀에 사용되는 규제를 L2 규제라고 함

$$Cost\,(W) = RSS(W) + \lambda * (sum\ of\ squares\ of\ weights)$$

$$= \sum_{i=1}^{N} \left\{ y_i - \sum_{j=0}^{M} w_j\, x_{ij} \right\}^2 + \lambda \sum_{j=0}^{M} w_j^2$$

- scikit-learn의 Ridge 사용
  - » 알파 값을 크게 해서 규제의 강도를 높이면 일반화 성능이 향상됨

라쏘(Lasso) 회귀

- 릿지 회귀와 같이 가중치 계수를 0에 가깝게 만드는 작업을 하지만 릿지 회귀와 다른 방식으로 처리 → L1 규제

- L1 규제의 결과로 어떤 가중치 계수는 실제 0이 되기도 함
  - » 모델에서 완전히 제외되는 특성이 발생
  - » 특성 선택이 자동으로 이루어지는 것으로 해석할 수 있음
  - » 일부 계수를 0으로 만들면 모델을 이해하기 쉬워지고 모델의 중요한 특성을 구분할 수 있음

$$Cost\,(W) = RSS(W) + \lambda * (sum\ of\ absolute\ value\ of\ weights)$$

$$= \sum_{i=1}^{N} \left\{ y_i - \sum_{j=0}^{M} w_j\, x_{ij} \right\}^2 + \lambda \sum_{j=0}^{M} |w_j|$$

- scikit-learn의 Lasso 사용

이진 분류용 선형 모델

- 주어진 속성의 선형결합을 바탕으로 분류 수행

- 이진 분류 선형 방정식

$$\hat{y} = w[0] \times x[0] + w[1] \times x[1] + \cdots + w[p] \times x[p] + b > 0$$

  » 결정 경계가 입력의 선형 함수 → 선, 평면, 초평면을 사용해서 두 개의 클래스를 구분하는 분류기

- 대표적인 선형 분류 알고리즘은
  » 로지스틱 회귀(Logistic Regression)와
  » 서포트 벡터 머신(Support Vector Machine, SVM)

- scikit-learn의 LogisticRegression과 LinearSVC 사용

다중 클래스 분류용 선형 모델

- 로지스틱 회귀만 제외하고 많은 선형 분류 모델은 태생적으로 이진 분류만 지원

- 이진 분류 알고리즘을 다중 클래스 알고리즘으로 확장하는 보편적인 방법은 일대다(one-vs-rest) 방법
  - » 각 클래스를 다른 모든 클래스와 구분하도록 이진 분류 모델 학습
  - » 클래스의 수 만큼 이진 분류 모델 생성
  - » 모든 이진 분류기 중에서 가장 높은 점수를 내는 분류기의 클래스를 예측 값으로 사용

나이브 베이즈를 사용한 분류

- 18세기 수학자 토마스 베이즈의 업적으로부터 유래

- LogisticRegression이나 LinearSVC 같은 선형 분류기보다 훈련 속도가 빠른 편이지만 일반화 성능은 다소 뒤지는 편

- 분류기 종류
  - » GuassianNB → 독립변수가 정규분포인 데이터에 적용
  - » BernoulliNB → 독립변수가 이항분포인 데이터에 적용
  - » MultinomialNB → 독립변수가 다항분포인 데이터에 적용

- BernoulliNB와 MultinomialNB는 대부분 텍스트 데이터 분석에 사용

나이브 베이즈를 사용한 분류

- 적용 사례
  - » 정크 이메일 필터링과 같은 문서 분류
  - » 침입자 검출 또는 컴퓨터 네트워크에서 이상 행동 검출
  - » 관찰된 증상을 고려한 질병 진찰

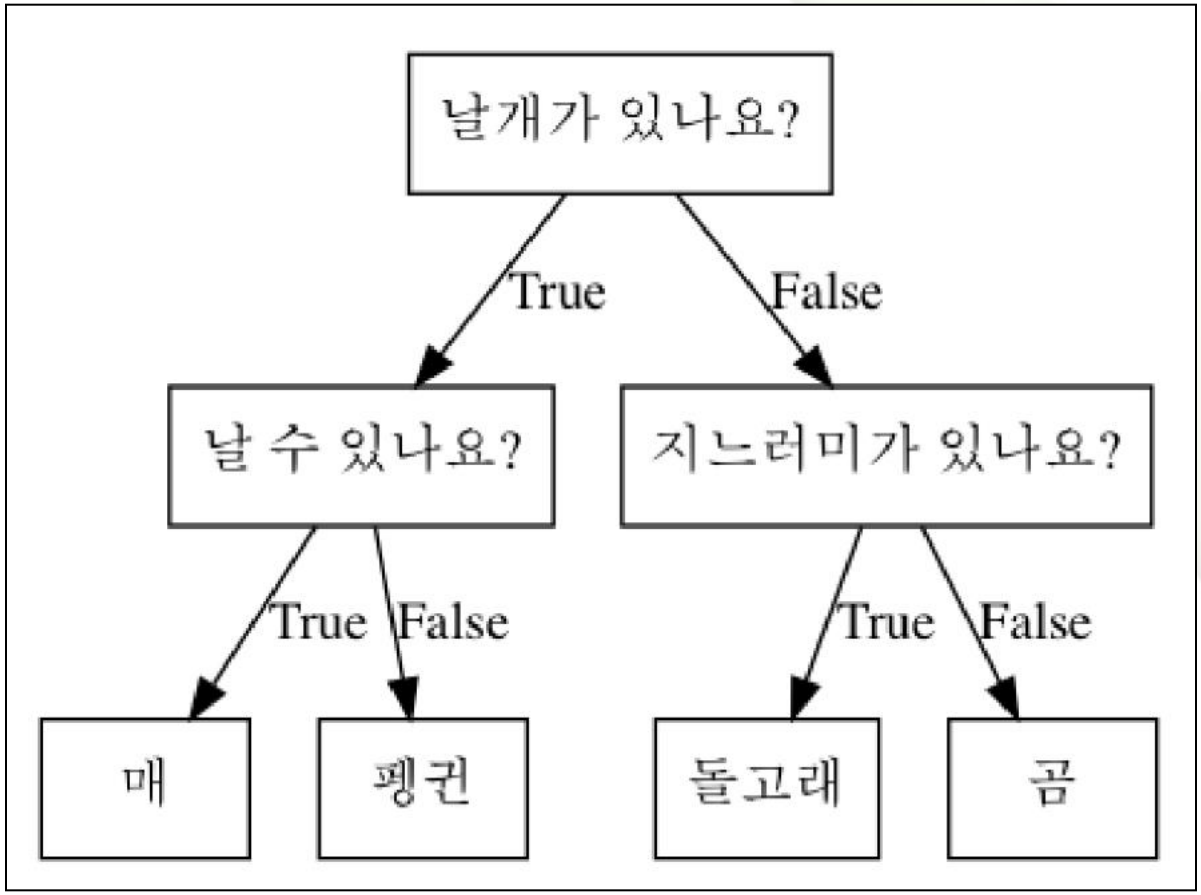
- 장단점
  - » 선형 모델의 장단점과 비슷

장점	단점
• 단순하고 빠르며 효과적 • 노이즈와 결측 데이터가 있어도 잘 수행됨 • 훈련 데이터의 양에 영향을 받지 않음(상대적으로 적은 사례 사용)	• 수치 속성으로 구성된 많은 데이터 세트에 대해 이상적이지 않음

결정 트리

- 분류와 회귀에 광범위하게 사용되는 모델

- 결정에 다다르기 위해 예/아니오 질문을 이어 나가면서 학습



트리 구조의 모델 형성

복잡도 제어

- 순수 노드 → 하나의 타겟 클래스로 구성된 노드

- 모든 리프 노드가 순수 노드가 될 때까지 진행하면 모델이 매우 복잡해지고 훈련 데이터에 과대적합됨 → 순수 노드로 이루어진 트리는 훈련 데이터에 100% 정확하게 맞는 모델

- 과대적합을 막는 방법은
  - » 트리 생성을 일찍 중단하기 (사전 가지치기)
  - » 데이터 포인트가 적은 노드를 삭제하거나 병합 (사후 가지치기)

장단점

- 장점
  - » 만들어진 모델을 쉽게 시각화할 수 있어서 비전문가도 이해하기 쉬움
  - » 데이터의 스케일에 구애받지 않음 → 정규화 또는 표준화 처리 불필요

- 단점
  - » 사전 가지치기를 사용해도 과대적합되는 경향이 강해서 일반화 성능이 좋지 않음 → 대안으로 앙상블 방법 사용

앙상블 (Ensemble)

- 여러 머신러닝 모델을 연결해서 더 강력한 모델을 만드는 기법

- 두 개의 모델이 분류와 회귀의 다양한 데이터 세트에서 효과적으로 동작
  - » 랜덤 포레스트 (Random Forest)
  - » 그레디언트 부스팅 결정 트리 (Gradient Boosting Decision Tree)

랜덤 포레스트 (Random Forest)

- 결정트리의 주요 단점인 훈련 데이터에 과대적합되는 경향을 회피하는 방법
- 조금씩 다른 여러 결정트리의 묶음
- 기본적으로 예측력이 좋으면서 서로 다른 방향으로 과대적합된 트리를 많이 만들어 그 결과를 평균 내면 과대적합된 양을 줄일 수 있다는 것이 수학적으로 검증됨
- 트리들이 서로 달라지도록 트리 생성시 무작위성 주입
  - » 데이터 포인트를 무작위로 선택하는 방법
  - » 분할 테스트에서 특성을 무작위로 선택하는 방법

랜덤 포레스트 (Random Forest)

- 장점
  - » 성능이 매우 뛰어나고
  - » 매개변수 튜닝을 많이 하지 않아도 잘 작동하며
  - » 데이터의 스케일을 맞출 필요도 없음

- 단점
  - » 텍스트 데이터와 같이 매우 차원이 높고 희소한 데이터에는 잘 작동하지 않음 → 선형 모델이 더 적합
  - » 선형 모델에 비해 많은 메모리를 사용하며 훈련과 예측이 느림

그래디언트 부스팅 회귀 트리

- 여러 개의 결정 트리를 묶어 강력한 모델을 만드는 방법 (랜덤 포레스트와 동일) → 회귀와 분류 모두에 사용 가능

- 이전 트리의 오차를 보완하는 방식으로 순차적으로 트리 생성 (랜덤 포레스트와 차이)
  - » 이전 트리의 오차를 얼마나 강하게 보정할 것인지 설정 (learning_rate)

- 무작위성 없음 → 대신 과적합화를 막기 위해 강력한 사전 가지치기 사용

- 1 ~ 5 정도의 낮은 트리를 사용하기 때문에 메모리 사용량이 적고 예측도 빠름

- 트리가 많이 추가될수록 예측 성능 향상됨

그레디언트 부스팅 결정 트리

- 장점
  - » 특성의 스케일 조정 필요 없음

- 단점
  - » 매개변수를 잘 조정해야 의미 있는 결과 도출
  - » 훈련 시간이 오래 걸림
  - » 희소한 고차원 데이터에 대해 잘 작동하지 않음

커널 서포트 벡터 머신

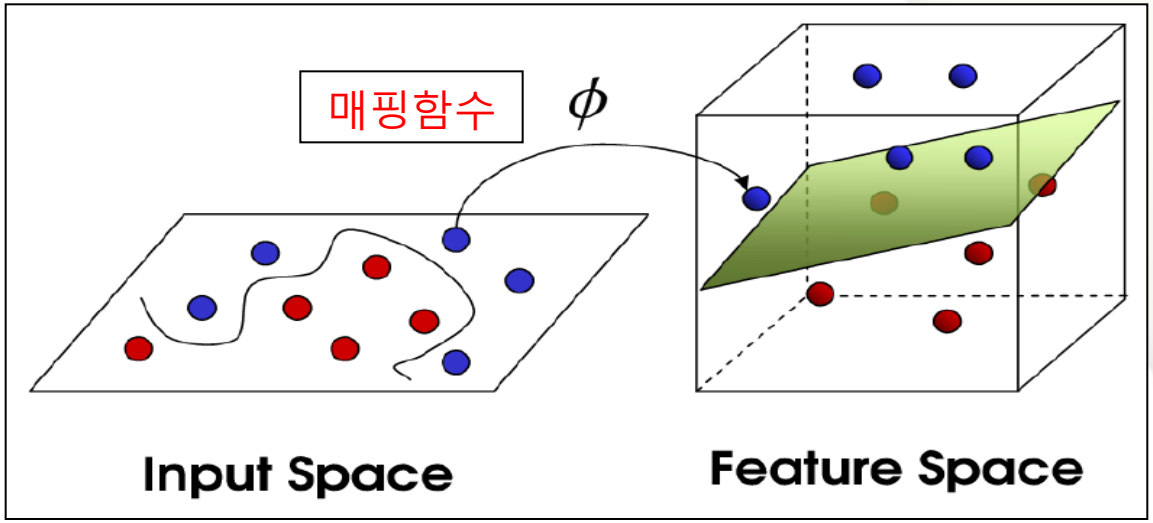
- 입력 데이터가 단순한 초평면(hyperplane)으로 정의되지 않는 더 복잡한 모델을 만들 수 있도록 확장

- 분류와 회귀에 적용 가능

선형 모델의 비선형 특성

- 직선과 초평면은 유연하지 못해서 저차원 데이터 세트에서 선형 모델이 매우 제한적

- 선형 모델을 유연하게 만들기 위해 특성끼리 곱하거나 거듭제곱 하는 방식으로 새로운 특성을 추가하는 방법 사용



커널 기법

- 비선형 특성을 추가해서 선형 모델을 강력하게 만드는 경우 추가할 특성을 선택하는 문제와 많은 특성을 추가했을 때 연산 비용 문제 발생

- 커널 기법을 사용하면 수학적 기교를 통해 새로운 특성을 많이 만들지 않고도 고차원에서 분류기 학습 가능

- 데이터를 고차원에 매핑할 때 사용하는 방법
  - » 다항식 커널
  - » RBF(Radial Basis Function) 커널

SVM 데이터 전처리

- 특성 스케일에 매우 민감해서 입력 특성의 범위를 비슷하게 만들어야 함
- 일반적으로 모든 특성 값을 0 ~ 1 범위에 맞추는 방법을 많이 사용

Spark Machine Learning

Machine Learning on Spark

- 스파크 기반 머신 러닝의 장점
  - 스파크의 분산 처리 능력을 이용해 매우 큰 규모의 데이터셋으로 비교적 빠른 속도의 머신 러닝 알고리즘을 훈련하고 적용할 수 있음
  - 머신 러닝 작업의 대부분을 한 곳에서 수행할 수 있는 통합 플랫폼 제공

- 주요 기능
  - 분류, 회귀, 클러스터링, 협업 필터링 등 주요 머신 러닝 알고리즘 지원
  - 특성 추출, 변환, 선택과 관련된 API 제공
  - 파이프라인 API 지원
  - 모델 및 파이프라인을 저장하고 불러오는 기능 지원
  - 선형대수, 통계, 데이터 처리 등의 유용한 함수 제공

스파크 머신 러닝 주요 라이브러리

- Spark MLlib
  - UC 버클리의 MLBase 프로젝트 기반 → 오픈 소스 커뮤니티 주도로 확장
  - 스파크 0.8 버전부터 지원
  - spark.mllib 패키지를 사용하는 RDD 기반 API
  - 스파크 3.0 이후 제외될 예정

- Spark ML
  - 스파크 1.2부터 지원되는 머신 러닝 API
  - spark.ml 패키지를 사용하는 데이터프레임 기반 API
  - 머신 러닝 파이프라인 기능 제공
    - › 머신 러닝과 관련된 모든 연산 작업을 시퀀스 하나로 모아 단일 연산처럼 처리

주요 알고리즘 구현

- 텍스트 데이터 처리 관련 도구

도구	설명
Tokenizer	공백 문자를 기준으로 입력 문자열을 개발 단어의 배열로 변환하고 이 배열을 값으로 하는 새로운 컬럼을 생성하는 트랜스포머
TF-IDF	여러 문서 집합에서 특정 단어가 특정 문서 내에서 가지는 중요도를 수치화한 값
StringIndexer	문자열 컬럼에 대응하는 숫자형 컬럼을 생성하는 평가자 문자열 컬럼의 각 문자의 노출 빈도에 따라 인덱스 부여
IndexToString	StringIndexer의 인코딩 결과를 원래의 문자열로 반환하는 트랜스포머 예측 완료된 최종 데이터셋에서 실제 문자열 레이블로 복원하는데 사용

주요 알고리즘 구현

- 회귀 알고리즘 구현 도구
  - 설명변수를 통해 연속형 변수를 예측하는 모델

도구	설명
LinearRegression	연속형 변수를 예측하는 선형 회귀 모델 구현 학습(fit)의 결과는 LinearRegressionModel 객체
GeneralizedLinearRegression	잔차 분포가 정규 분포를 따르지 않는 경우의 선형 회귀 모델 구현
DecisionTreeRegressor	결정 트리 기반 회귀 알고리즘 구현
RandomForestRegressor	다수의 결정 트리를 결합한 트리의 앙상블 알고리즘 구현
GBTRegressor	Stochastic Gradient Boosting 기반의 결정 트리 앙상블 알고리즘 구현

주요 알고리즘 구현

- 분류 알고리즘 구현 도구
  - 설명 변수를 통해 범주형 변수를 예측하는 모델

도구	설명
LogisticRegression	확률 기반 선형 분류 알고리즘 구현
DecisionTreeClassifier	결정 트리 기반의 분류 알고리즘 구현
RandomForestClassifier	트리 앙상블 기반의 분류 알고리즘 구현
GBTClassifier	Stochastic Gradient Boosting 트리 앙상블 분류 알고리즘 구현
MultilayerPerceptronClassifier	다층 신경망 기반 분류 알고리즘 구현
OneVsRest	이진 분류 도구를 기반으로 다중 분류 알고리즘 구현
NaiveBayes	조건부 확률에 관한 베이즈 이론 기반의 분류 알고리즘 구현 Multinomial Naive Bayes, Bernoulli Naive Bayes 모델 선택 가능

주요 알고리즘 구현

- 클러스트링
  - 데이터의 유사도를 기반으로 데이터의 그룹을 분류하는 모델
  - 레이블 없이 학습하는 비지도 학습 알고리즘

도구	설명
KMeans	확률 기반 선형 분류 알고리즘 구현
GaussianMixture	전체 데이터셋을 다수의 가우시안 분포 합으로 분류

- 협업 필터링
  - 사용자 선호의 유사도를 기반으로 사용자의 관심사를 예측
  - 상품 추천 등의 추천 시스템에 많이 사용

도구	설명
ALS	사용자와 상품 사이의 평점 정보로 구성된 거대 희소 행렬에서 비어 있는 값을 찾는 알고리즘 중 가장 많이 사용되는 Alternating Least Squares 알고리즘 구현