인공지능 (머신러닝)

* **빅데이터**란
* 거대한 규모(volume),빠른 속도(velocity),높은 다양성(variety) 특징을 가진 방대한 데이터

+ value, veracity….

\* 데이터

- 정형데이터(Structured data) : 정제되어 사용할 수 있는 데이터 = 수치화해서 알고리즘에 넣을수 있는 데이터

- 비정형데이터(Unstructured data) : 정제되지않은 데이터 = 알고리즘에 넣을수 없는 데이터

- 반정형데이터(Semi-structured data): 크롤링된 데이터 등등

데이터를 정형화 시킬 때는 목적성에 맞게 수집, 처리 해야한다.

작동방식 – 빅데이터 수집, 빅데이터 저장, 빅데이터 분석

* **AI 산업화 단계**

1단계 – 업무 생산성 향상

2단계 – 새로운 통찰력으로 문제 해결

3단계 – 의사결정 향상

4단계 – 새로운 신규 비즈니스 창출

* **AI**

연산능력(단순 계산, 제어)->판단능력(정해진 것 중 선택)->추론능력(데이터들로부터 예측/추론- 머신러닝)-> 학습능력(학습을 통해 진화-딥러닝)-> 창의력(새로운 것을 만듦, 초인공지능)

#척도 – 어떤 변수를 넣을때 어떤 기준으로 넣느냐,

명목척도, 서열척도, 등간척도(+-가능), 비율척도(\*/가능)

* **인공지능의 종류**
* 약한 인공지능 : 학습을 통해 문제를 해결 (알파고,시리,구글번역 등)
* 강한 인공지능 : 사고를 통해 문제를 해결 (사람과 같은 지능)
* 초 인공지능 : 창의력을 통해 문제 해결 – 불가능

머신러닝 – 컴퓨터가 스스로 학습하여 인공지능의 성능을 향상 시키는 기술 방법

딥러닝 – 인간의 뉴런과 비슷한 인공신경망 방식으로 정보를 처리

* **머신러닝**
* Supervised Learning(지도 학습) : 인간이 프로그램에 Label을 지정해준 학습
* Unsupervised Learning(비지도 학습): Label을 주지 않고 프로그램이 직접 학습

#인풋 데이터를 통해서 그 데이터의 규칙성을 찾아내고 결과 값을 예측 하는 것

* 강화학습
* 오토인코더
* **지도학습**

데이터에 대한 명시적인 답이 주어진상태에서 컴퓨터를 학습 시키는 방법

분류와 회귀로 나뉘어 진다.

분류모델 – 경우의 수가 적을때(범주형 변수),맞냐 아니냐로 구분

회귀모델 – 경우의 수가 많을때, 평가기준:결과값이 정답과 근사치냐(100 =99.89)로 구분

회귀: 연속적인 숫자를 예측하는것, 분류에비해 예측값의 미묘한 차이가 크게 중요하지 않음

* **비지도학습**

데이터에 대한 명시적인 답이 없는 상태에서 컴퓨터를 학습 시키는 방법

데이터의 숨겨진 특징,구조,패턴 등을 파악하는데 사용.

데이터를 비슷한 특성끼리 묶는 클러스터링과 차원 축소 등이 있음

기하적 방법 - 데이터들을 좌표평면으로 나누었을때 거리를 중심으로 비슷한 데이터끼리 묶는 것

확률적 방법 – 등장 확률을 통해서 구분

* **강화학습**

지도학습과 비슷하지만 완전한 답을 제공하지 않음

기계는 더 많은 보상을 얻을 수 있는 방향으로 행동을 학습해 나감

주로 게임이나 로봇을 학습시키는데 많이 사용

데이터가 부족한 상황에서 좋은 성능을 발휘

* **오토인코더**

인풋데이터 그 자체가 Label로 활용

사람 그림을 받아 행렬로 치환하는 A알고리즘(인코더),

행렬을 그림으로 치환하는 B알고리즘(디코더).B알고리즘이 원본 그림과 같아질 때까지 학습.

**#머신러닝의 단점**

Garbage In Garbage Out 혹은 좋은 데이터로 안좋은 모델을 쓰면 쓰레기가 나옴

* **머신러닝 학습과정 7단계**

**1. 문제 정의**

**2. 데이터 수집**

**3. 데이터 전처리**

- 결측치 및 이상치 처리(삭제 or 대체)

- 특성 공학 (Feature Engineering)

Scaling: 단위변환

Encoding: 문자형 데이터 -> 수치형 데이터, Binning: 수치형 -> 문자형

Normalization: 정규분포화

1. 특성 선택 (Feature Selection)
2. 특성 결합 (Feature Combination)
3. 특성 추출 (Feature Extraction)

**4. 탐색적 데이터 분석(EDA)**

**-** 기술 통계, 상관관계 분석

- 시각화(pandas, matplotlib, seaborn)

Histogram(빈도수), Boxplot(평균,중간값,사분위수 등)

Scatter plot(산점도, 수치+상관관계), Plot(선 그래프)

- 특성선택

**5. 모델 선택**

**6. 모델 학습**

**-** model.fit(X\_train,y\_train)

- train데이터와 test 데이터를 7:3 혹은 7.5:2.5정도로 분리

- train 데이터의 문제와 정답으로 모델 학습

- model.predict(X\_test)

- test 데이터의 문제를 넣고 정답을 예측

**7. 모델 평가(model Evaluation)\_ 분류**

- test 데이터의 문제를 넣어서 예측한 정답과 실제 정답을 비교

- 정확도, 재현율, 정밀도, F1 Score

|  |  |
| --- | --- |
| TP | FN |
| FP | TN |

* 분류 모델 평가지표

Confusion Matrix(오차행렬)

TP: True Positive FP: False Positive FN: False Negative TN: True Negative

Accuracy(정확도) – TP+TN/TP+TN+FP+FN

Precision(정밀도) = TP/TP+FP => 불균형데이터일때 AI가 밀어버리지 못하게

Recall(재현도) = TP/TP+FN

정밀도와 재현도는 반비례 관계에 있다. (리스크가 있다면 재현도가 높아지는게 좋다.)

정확도의 한계점 = AI가 불균형데이터를 처리할 때 한쪽으로 몰아버려 정확도를 올리려고 하는 경우가 있다. 리스크가 있을 때..

F1Score(조화평균) 2\*Precision\*Recall/ Precision + Recall = 2/(1/Precision)+(1/Recall)

Sensitivity(민감도) = TN/TP+FN, Specificity(특이도)=FP/FP+TN : ROC커브를 위해

**지도학습 알고리즘**

**과대적합(Overfitting)**

* train데이터에 너무 과도하게 학습되어 있어 train데이터에만 잘 동작하고 test에서는 예측성능이 저하되는 현상

Bias가 커지게 된다.

Train score는 높지만, test score 는 낮아진다

**과소적합(Underfitting)**

* train데이터를 충분히 반영하지 못해 train, test 데이터 모두에서 예측성능이

저하되는 현상(학습을 제대로 하지 못한 것)

Variance가 크게 나타난다.

Test,train score 둘다 낮음

**일반화(Generalization)** <-목표

* 모델이 처음보는 데이터에 대해 정확하게 예측할 수 있는 상태

데이터가 적을 경우, 과대적합 가능성이 높아진다.

**#**Bias: 평균으로부터 멀어져 있는 정도 = 예측값 – 측정값

**#**Variance: 각 데이터간 떨어진 정도 = 측정값의 평균 – 측정값

Bias-Variance trade-off : 학습량이 많아질수록 Bias는 커지고 Variance는 줄어듦

\* 과대,과소 적합 방지 Ridge Lasso

1. 규제(Regulation) 테크닉: 가중치에 한계를 줌

2. Drop Out 테크닉: 학습시킬때 일부를 빼고 학습시키는 방법(의존성 감소)

**1.분류**

**-** 미리 정의된, 가능성 있는 여러 클래스 레이블 중 하나를 예측하는 것

- 이전분류(binary classification) : 두 개의 클래스만으로 분류하는 것

- 다중분류(multiclass classification) : 셋 이상의 클래스로 분류하는 것

**KNN모델(Kth Nearest Neighbor)**

k번째 최근접 이웃 알고리즘

* 새로운 데이터 포인트와 가장 가까운 훈련 데이터셋의 데이터 포인트를 찾아 예측
* k값에 따라 가까운 이웃의 수를 결정

# k값은 Label의 배수가 되면 안됀다.

@ KNN에서 거리를 측정하는 방법

1. 유클리디안 거리(Euclidean Distance)

2. 맨해튼 거리(Manhattan Distance)

# **주요 매개변수(Hyperparameter)** : 인간이 정해준 값

# 매개변수(parameter) : 컴퓨터가 정해준 값

(Scikit-learn의 경우)

Ex) KneighborsClassifier(n\_neighbors = 이웃의 수(= Hyperparameter))

@ 변수 값 범위 재조정 방법

* Min max normalization : Z = (X -min(X))/(max(X)-min(X))
* Z-score standardization : Z=(X-평균)/표준편차

@ 장/단점

* 이해하기 매우 쉽고 큰 조정없이 나쁘지 않은 성능을 발휘하는 기초모델
* 훈련데이터 세트의 크기가 크면 예측이 느려짐(학습자체는 빠름)
* 계산 시간이 오래걸린다.
* 거리를 측정하기에 데이터의 스케일 조정이 필요할 수있음
* 직접적인 예측보단 데이터를 파악하기 위한 용도로 사용

**2. 선형 회귀(Linear Regression)**

- 학습데이터에는 없는 미지의 데이터에 대한 값을 예측할때,

- 데이터의 분포를 가장 잘 표현할 수 있는 직선을 그려서 값을 예측하는 방법

(#통계에서 선형회귀는 정규화를 제외한 데이터,모델조작을 허용하지 않음.

머신러닝에서는 상관없음.)

-데이터의 분포가 선형일때 사용

- mse(최소제곱오차 : (측정값-예측값)제곱)가 낮아지면 낮아질 수록 잘 예측한 모델

- 단순 선형 회귀(Simple Linear Regression)

y= Wx +b

- 다중 선형 회귀(Multiple Linear Regression)

y= W1x1 + W2x2 +… + Wnxn+b

@ 경사하강법(Gradient Descent Algorithm)

* 평균제곱 오차가 최소가 되게하는 최적의 w, b값을 찾는 방법론
* 기계가 스스로 학습한다는 머신, 딥러닝의 개념을 있게 한 핵심 알고리즘
* 비용함수의 기울기를 구하여 기울기가 낮은 쪽으로 계속 이동하여 값을 최적화 시키는 방법

= 최저점 수렴이 안정적, 속도가 느리고 메모리가 많이 필요

@ 확률적 경사하강법(Stochastic Gradient Descent)

* 큰 데이터셋에서 일반 경사하강법의 느린 단점을 보완하기 위한 방식
* 전체 데이터가 아닌 일부데이터만으로 w, b 값을 업데이트

-> 따라서 항상 좋은 방향으로만 업데이트가 일어나지는 않음

-> 일부데이터로 판단하기 때문에 속도가 빠름(데이터 수가 많을 때 유리)

= 최저점 수렴이 불안정, 속도가 빠름

(# 미니 배치 확률적 경사하강법 : 경사하강법과 확률적 경사하강법의 절충안, 전체데이터를 batch\_size개씩 나눠 학습)

1. 선형회귀는 지도학습이다. (회귀모델)
2. 선형회귀의 평가지표는 MSE다 (MSE는 작을수록 좋다)
3. MSE를 미분 / 경사하강법(데이터양or특성 많을수록유리)
4. GD/SGD(가장 보편적)/MGD
5. 학습률은 경사하강법의 속도,정교함을 결정 (학습률 업 = 정교함 다운,속도 업)

장/단점

직관적, 학습이 빠름

변수 설정을 할 수 없음, 선형성 전제를 요구

규제 모델의 비용함수(Cost Function)

* mse에 과도하게 적합한 값(Beta)에 alpha를 더해주는 규제.
* > 선형회귀의 과대적합을 피하기위해 사용
* Hyperparameter = Alpha(규제 강도)

1. L1규제: Lasso | mse + |B| 절대값으로 규제 : 특정항을 지워버릴때 사용
2. L2규제: Ridge | mse + B^2 으로 규제 : 특정항을 그래도 살려둘때 사용
3. L1+L2 : L1규제로 변수를 줄이고 L2규제로 남은 변수들의 영향도를 줄임

**로지스틱 회귀(Logistic Regression, logit)**

분류,기하 모델

Hyper Parameter

- C(규제강도: 결정경계를 통제할 수도 있음),

- max\_iter(경사하강법 사용을 위한 횟수 지정)

장/단점

* 회귀모델과 호환이 가능
* 직관적
* 이진분류만 가능

**의사결정나무 알고리즘(Decision Tree)**

**Tree 를 만들기 위해 예/아니오 질문을 반복하며 학습(스무고개와 유사)**

**지니 불순도(Gini Impurity 손실함수)**

#Entropy: 불확실성

#손실함수: 최적의 파라미터를 판단하는 기준

* 해당 범주안에 서로 다른 데이터가 얼마나 섞여있는지를 뜻함
* 결정 트리 모델의 노드 분할 기준
* 각 질문들이 얼마나 좋은 질문인지 수치로 파악 할 수 있음
* 0~0.5 사이 값을 범위로 가짐
* 불순도가 0에 가까울수록 잘 분류된것. (좋은 질문)
* 불순도가 0.5라면, 데이터가 5:5 비율로 섞여서 분류된것 (좋지 않은 질문)

**주요 매개변수**

* 트리의 최대 깊이 : max\_depth (값이 클수록 모델의 복잡도가 올라간다. 무한분류 방지)
* 리프 노드의 최대 개수 : max\_leaf\_nodes (복잡도 상승 방지)
* 리프 노드를 구성하는 최소 샘플의 개수: min\_samples\_leaf(리프를 생성할 최소데이터 개수)

**결정 트리 과대적합 제어**

트리의 깊이가 매우 깊고 복잡해지면, 해당 조건에만 잘 맞춰진 모델이 됨.

즉, **과대적합**의 위험성이 증가

* 트리의 최대깊이, 리프 노드의 최대 개수를 지정(작을수록 복잡도 하강)
* 리프노드를 구성하는 최소데이터의 개수를 지정(클수록 복잡도 하강)

1)가지치기(drop out)

적절한 수준에서 트리의 Nodes 일부를 자르거나 합쳐 줌으로서 과대적합을 제어

**장/단점**

* 만들어진 모델을 쉽게 시각화 할 수있어 이해하기 쉬움
* 거리 기반이 아니기 때문에 데이터 스케일에 영향을 받지 않아 데이터 전처리가 거의 필요 없어짐
* 트리구성시 각 특성의 중요도를 계산하기 때문에 특성선택에 활용될 수있음
* 복잡도가 올라갈수록 과대적합이 되기 쉬워 일반화 성능이 좋지 않음
* 회귀분석, 분류분석이 전부 가능

**랜덤 포레스트(Random Forest)**

서로 다른 특정, 노드 배치, 하이퍼파라미터를 랜덤으로 설정한 의사결정나무를 여러 생성해서

과대적합의 위험성을 줄인 방식

* 다수의 tree의 의견이 통합되지 않는다면 투표에의한 다수결의 원칙을 따름 – 앙상블방법

**주요 매개변수**

* 트리의 개수 : n\_estimators
* 선택할 특징의 최대 수: max\_features (1로 하면, 특성을 고려X, 큰값이면 DT와 비슷해짐)
* 선택할 데이터의 시드 : random\_state

**특징**

결정 트리 모델의 과대적합을 통계적 방법으로 해소

트리모델처럼 쉽고 직관적임

부스팅 방식에 비해 빠른 수행속도

모델튜닝을 위한 시간이 많이 필요

큰데이터 세트에도 잘 동작하지만, 트리의 개수가 많아지면 느려짐

**앙상블**

**보팅Voting**

**배깅Bagging**

**부스팅Boosting**

**1) 보팅**

-여러 개의 다른 모델이 예측한 결과를 투표 혹은 평균을 통해 최종 예측결과를 선정

1. 하드보팅(Hard voting): 다수결

2. 소프트 보팅(Soft voting): 각 확률의 평균

회귀모델은 각 모델들이 예측한 수치값의 평균을 통해서 최종 예측

**2) 배깅**

같은 종류의 모델을 결합(데이터 샘플링을 다르게, 중첩 허용)

과대적합 방지

**3) 부스팅**

여러개의 모델이 순차적으로 학습-예측하며 잘못 예측한 데이터에 가중치를 부여해,

오류를 개선해 나가면서 학습하는 방식(결정 트리 모형을 베이스로 사용)’

과소적합 방지

**그래디언트 부스팅**- 경사하강법을 적용한 학습법

**XG 부스팅(eXtreme Gradient Boosting)**: 그래디언트 부스팅의 과대적합을 막기위해 ,

규제를 포함한 부스팅방법, 병렬로 빠른 학습 가능,