

머신러닝(Machine Learning)

1. Scikit-learn

- AI는 코딩하지 않은 것도 추론하여 결과를 도출해냄
- 설치: tensorflow, keras

1.1 머신러닝(Machine Learning)과 딥러닝(Deep Learning) 비교

1.1.1 머신러닝

- 규칙을 사람이 만들어 학습
- 기계한테 어떤것을 학습할 지 지정해주어야 함,
- 적은양의 데이터로 학습 가능

1.1.2 딥러닝

- 종단간 기계학습(데이터로부터 학습)
- 패턴을 스스로 찾아내어 학습 --> 왜 그러한 결과와 나왔는지 알수 없음(블랙박스)
- 수억개의 신경망(Neural Network)이 연결되어 학습
- ex) 알파고

1.2 지도학습(supervised)과 비지도학습(unsupervised)

- Model
 - Neural network

1) 지도학습(supervised)

- 데이터를 집어넣고 정답을 알려줌, 사전에 피쳐들을 넣어놓고 학습
- 1. 종류(Model)
 - 회귀(regression) --> 모델이 어려움
 - Model
 - ridge
 - 분류(classification) --> 평가가 어려움

- 이진분류 : 두개의 클래스로 분류
- 다중 분류 : 셋 이상의 클래스로 분류
- Model
 - NN : 나와 주위의 이웃을 비교, KNN(이웃에 따라 나의 분류가 달라짐, 저급모델)
 - Naive Bayes
 - Tree : 분류에 많이 사용, 분산값 이용
 - Ensemble : 저급의 모델을 모아 강력한 모델을 만들어 냄, 신뢰도가 높은 쪽에 가중치
- Feature selection : 모델을 학습시킬때 주로 어떤 피처를 가지고 학습시킬것인지 선택
- Semi-supervised : 정답이 없거나 imbalancing한 데이터를 정답 라인에 가깝게 끌어내려 분류
- 2. 공통점과 차이점
 - 공통점
 - 답이있음
 - 차이점
 - 회귀 : 연속된 수치, 특정한 수치로 표현됨 ex. 날씨, 주식가격, 집값
 - 분류 : 이산형(수치가 산발적), 여러 레이블 중 하나를 예측 ex. grade, 상증하
 - 타깃이 명확해야 두개를 명확히 구분 가능

2) 비지도학습(unsupervised)

- 답이 없음, 샘플을가지고 전체데이터에서 유추
- 1. 종류(Model)
 - 비지도변환(unsupervised Transformation)
 - 데이터를 짧게 표현해 원래 데이터보다 쉽게 해석할 수 있도록 만드는 알고리즘
 - 차원축소 : 데이터의 일부를 잘라내 대표값으로 축소, n회 반복
 - 이미지에 많이 사용(픽셀)
 - 군집(clustering) --> 현업에서 많이 사용
 - 분류와 군집의 차이는 정답이 있느냐, 없느냐
 - 정확한 정답있으면 분류
 - 정확한 정답이 없으면 군집
 - Density : 거리(뉴클리드 거리)기반

1.3 검증(Validation)

- 내 평가 모델의 당위성을 확보하기 위한 필수적인 단계
- 어떤 데이터인가에 따라 적합한 모델이 다름
 - 어떤 모델이 적합한지는 돌려보아야 한다.
 - 모델을 학습하고 난 후 반드시 검증 단계를 거쳐야 한다

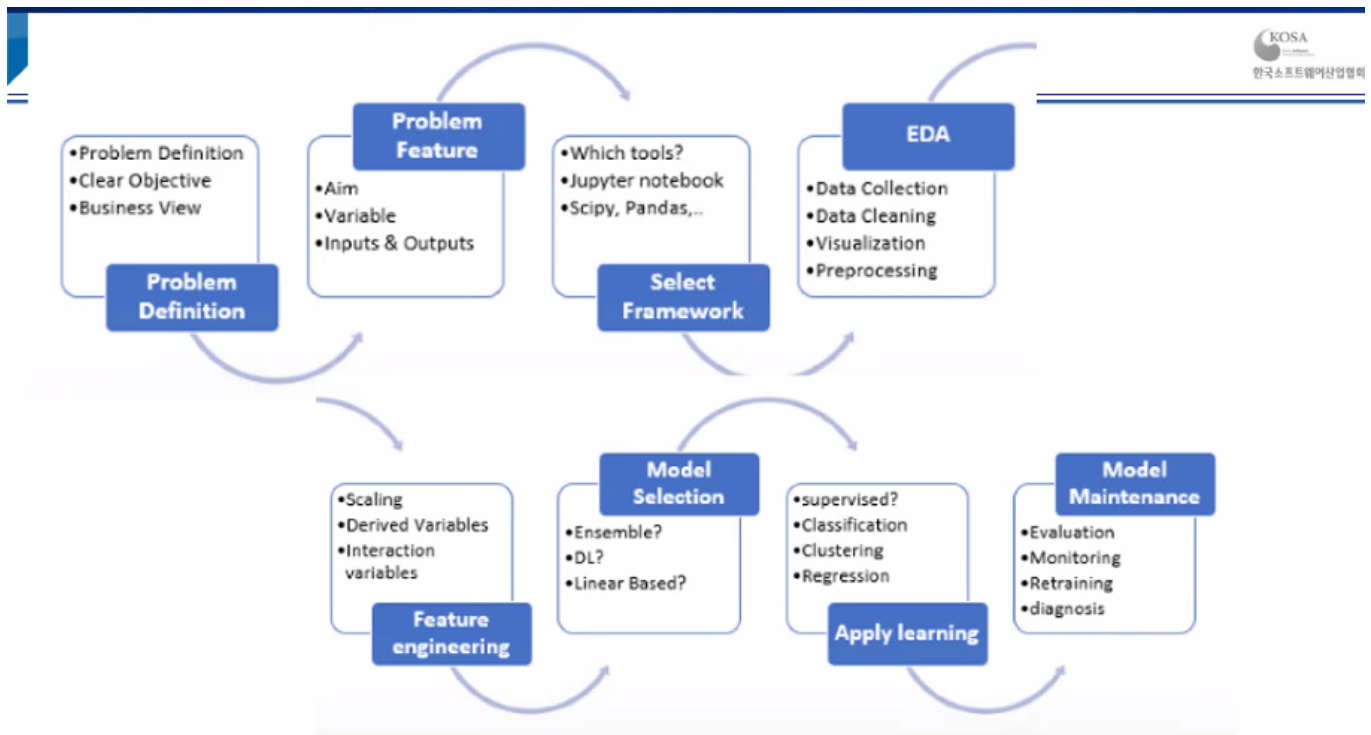
1.4 Data Transformation

- (class)Imputation : 결측치를 채움
- Pipeline : 데이터 가공하는 모든 단계를 묶어놓은 것, Pipeline거친 데이터를 볼 수 있음(Optional)

1.5 학습 종류

- 과대적합(OverFitting)
 - 완벽히 적합한 답안을 찾음 , 예측도를 현저히 낮춤-->답을 찾기 어려움
 - 머신러닝 정답률이 80~90%인 경우
 - 주어진 데이터에 맞는 경우만 학습, 다른 데이터가 들어오면 학습 불가능 , 일반화 불가능
- ★일반화(Generalization)
 - 평균으로부터 편차가 얼마나 떨어져 있는지(정규분포도)
 - 특이한 데이터(Outlier)가 들어오면 해당 데이터를 걷어내야 함
 - 이론적으로는 80~90% , 현업에서는 48~50%의 확률
- 과소적합(UnderFitting) : 너무 lose -->학습이 제대로 되지 않음

1.6 학습 단계



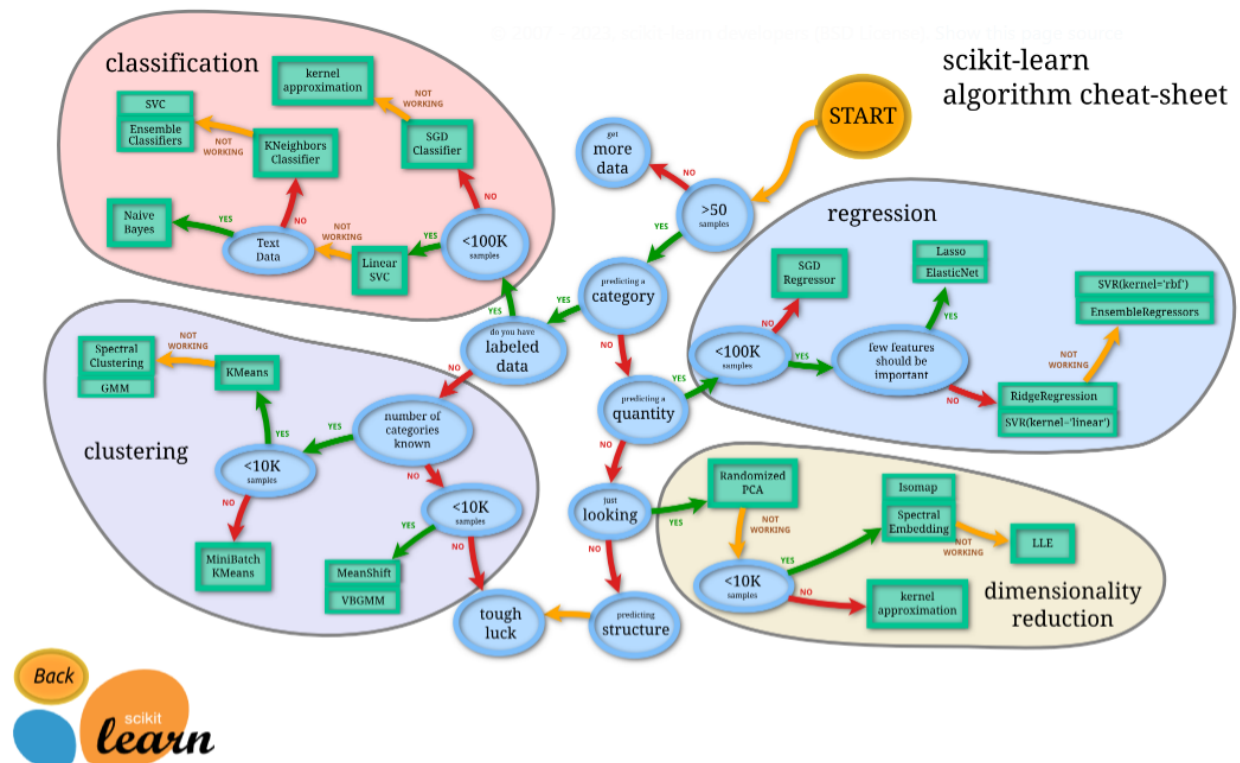
- 1. 문제정의(Problem Definition)
- 2. 피쳐선정(Problem Feature)

- 3. 개발 툴 선정(Select Framework)
- 4. EDA : 데이터 수집,가공,시각화, 전처리 --> 눈으로 보기
 - Data Collection
 - Data Preprocessing
 - 데이터 셋 확인
 - 유형(범주형/연속형)
 - 데이터 타입
 - 타겟확인(valancing/unvalancing)
 - 결측값 처리(Missing value treatment)
 - 삭제(행/컬럼)
 - 충분한 논의를 거쳐 삭제할 것인지 남겨둘 것인지 결정
 - 다른값으로 대체
 - 평균(mean), 최빈값(mode), 중간값(median) --> 모델 왜곡 높아짐
 - 예측값 삽입 --> 현업에서 가장 많이 사용
 - 회귀, KNN(결측채우기에 가장 효과적) 가장 많이 사용
 - 결측채우는 모델을 별도로 만들어 예측치 사용
 - 일시적으로 결측이 있는 데이터를 타겟으로 선정하여 예측
 - 이상값 처리(Outlier treatment)
 - 특이치 : 이상값 중에서도 별도로 심하게 outlier 되는 값
 - 이상값 찾아내기
 - histogram, scatter(*), bow plot, IQR 사용하여 이상치 측정
 - 이상값 : 단순한 오타, 특이한 값 등 --> 다른값으로 대체
 - 변수화
 - 이상치가 있다면 바로 삭제하는 것이 아니라 세부적인 파악 필요
 - 리샘플링 : 이상값을 분리하여 모델을 만듦
 - 케이스를 분리하여 분석
 - 이상값을 포함하여 모델링: 이상값 제외 모델을 모두 만들고 각 모델에 대해 설명
 - 정상, 비정상 데이터에 해당하는 모델을 각각 만든 후 두개의 모델의 차이를 사용
 - Data Cleaning
 - Transforming Features(가공) : 데이터에 변형을 가하는 작업
 - Feature Encoding
 - 분류의 경우 Encoding 중요하지 않음
 - 회귀의 경우 매우 중요, 들어오는 값에 따라 결과가 확연히 달라짐 --> OneHotEncoding 처리
 - Visualization
- 5. 피쳐 가공(Feature Engineering) : 피쳐 가공 --> 데이터 원본에 변형을 가함 ex.삭제, 결측채우기 등
 - Scaling : 단위변경, 편향된 분포 변경(표준화, 정규화)
 - 회귀 : log Scaling

- Binning
 - 구간화(연속형 -> 범주형)
- Dummy
 - Binning과 반대(범주형 -> 연속형)
 - 원핫인코딩
- Transform
 - 변수의 성질을 이용해 다른 변수를 만드는 것
 - FC
 - PCA : 이미지분석에 많이 사용, 편차를 가장 크게하는 값의 분포도를 이용하여 모든 데이터를 담을 수 있는 차원을 만드는 것
- 6. 모델 선정(Model Selection) : 어느 모델이 적합한지는 돌려보아야 함
- 7. 모델 학습(Apply Learning) : 데이터를 모델에 집어넣어보기
- 8. 유지보수(Model Maintenance) : 이전데이터와 새로운 데이터 학습 ex. 평가, 검증
 - 데이터 유형, 순서 등이 달라지는 경우 모델이 달라질 수 있음 --> 끊임없이 반복
 - 학습이 한번 끝날때마다 해당 결과를 저장 --> 다음 학습을 시킬때 시간 단축 가능

1.7 Choosing the right estimator

- 초기모델(현재는 많이 변화됨)
- label : 정답



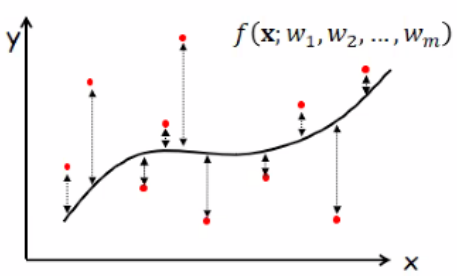
1.8 머신러닝 단계

- accuracy(정확도) : 정답과 일치하는 개수 --> 일반적으로 정확도만 보는 모델은 드물다
- loss : 정답과의 차이값 --> loss 값이 작은 모델 선택
- 학습 알고리즘이 비용함수(loss, $f(x)$)를 최소화하는 모델의 파라미터(가중치(기울기), b(절편)) 찾기
 - $y = f(x; w_1, w_2, \dots, w_n)$
 - ★가중치(기울기)가 크면 클수록 중요한 피쳐
 - 함수 f 가 주어진 데이터 x 에 가장 잘 부합하는 w 값을 구한다
 - $\text{Error} = (\text{실제값} - \text{예측치})^2$ 의 값의 총합 --> 최소화하는 함수(f , 기울기) 찾기

Goal of Machine Learning

• 주어진 데이터를 가장 잘 설명하는 함수란?

- 주어진 데이터와 오류를 최소화하는 함수
- $$\text{Error} = \sum_{(x,y) \in \text{Data}} (y - f(x; w_1, w_2, \dots, w_m))^2$$
- 즉 Error를 최소화하는 w_1, w_2, \dots, w_m 값을 갖는 함수
- $$E(w_1, w_2, \dots, w_m) = \sum_{(x,y) \in \text{Data}} (y - f(x; w_1, w_2, \dots, w_m))^2$$



- 모델에 신규데이터(y) 적용해 예측(추론, inference)
- 모델의 정확도(일반화) 확인
 - 데이터 : 데이터가 너무 적거나 다양하지 못함 --> 편향되게 학습, overfitting
 - 모델 : 다차원, 모델이 지나치게 복잡 --> overfitting

1.9 Fitting and Predicting

- fitting : 학습시키다
- predict : 예측, 맞춰봐
-

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(random_state=0)
X = [[ 1,  2,  3], # 2 samples, 3 features
      [11, 12, 13]]
y = [0, 1] # classes of each sample
clf.fit(X, y)
```

```
clf.predict(X) # predict classes of the training data
clf.predict([[4, 5, 6], [14, 15, 16]]) # predict classes of new data
```

- train 파일을 test용으로 나눔 : train_test_split --> 보통 8:2, 7:3 비율로 나눔
- random_state : random seed값 고정, 설정하지 않으면 데이터가 랜덤으로 계속 바뀜
- 하이퍼 파라미터를 튜닝하여 최적의 점수를 내게 되어있음
 - 파라미터는 점수에 굉장히 큰 영향을 미침
 - 파라미터 보정은 굉장히 어려운일, 보통은 있는 값을 사용
 - 최적의 파라미터를 찾아주는 패키지 존재함

```
from sklearn.model_selection import train_test_split
# X: dataset['data'], y:dataset['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

#분석을 위한 알고리즘 선택
from sklearn.____ import OOOalgorithmOOO
OOO = OOOalgorithmOOO(파라미터=보정값) ←분석에 중요한 영향을 미침

#train 문제/답 훈련
OOO.fit(X_train, y_train)
#test 문제
pred = OOO.predict(X_test)
#test 문제에 대한 정답률 : 일반화 정확도
OOO.score(y_test, pred)
```

1.10 연습

```

# -----
# 1. X(문제지), y(답안지) 분리
# 2. 위예것을 8:2 비율로 train, test용으로 분리
# 문제지80, 문제지20, 답안지80, 답안지20 = train_test_split(문제지, 답안지)
# 3. 모델선택
# 모델.fit(문제지80, 답안지80)
# 예측답안20 = 모델.predict(문제지20)
# 4. 점수확인 : *score(답안지20, 예측답안20)
# -----
# 데이터프레임 : object X, 결측 X
# X,y분리 → 8:2 → 모델 → fit → predict → score → 평가검증
# -----

```

X: 문제지 / y: 답안지
 X_train : 원래 제공되는 문제지
 X_test : 원래 제공되는 답안지
 -----모델.fit() : AI 학습시키고
 y_train : 주최측에서 제공하는 문제지 --> 내가 학습시킨 AI 적용 ---> 모델.predict() : 예측치를 구해 답안 작성
 y_test : 주최측에서 제공하는 짤 답안지 --> accuracy_score 사용해서 내가 작성한 답안지와 비교하여 점수를 냄

```

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

y= train['survived'] # target #1차는 소문자, list, series
X = train.drop('survived',axis=1) # target 제외한 나머지 #2차는 대문자, dataframe

X_train,X_test, y_train, y_test= train_test_split(X,y,test_size=0.2,random_state=1212)

dtc= DecisionTreeClassifier()
dtc.fit(X_train,y_train) # fit : train(80%)로 학습, return 없음
y_pred=dtc.predict(X_test) # predict : train(20%)로 예측

accuracy = accuracy_score(y_test,y_pred)
print(accuracy)

```