**1강 인공지능, 머신러닝 그리고 딥러닝이란 무엇인가?**

* 인공지능Ↄ 머신러닝 (소프트웨어에서 주로 사용됨)Ↄ딥러닝(=인공신경망)
* 본 강의는 scikit-learn을 활용한 머신러닝 & tensorflow를 활용한 딥러닝으로 이루어짐

**2강 코랩과 주피터 노트북으로 손코딩 준비**

* 딥러닝은 gpu를 사용함 – gpu란?

일종의 그래픽 처리 장치로 많은 양의 벡터연산이 필요한 딥러닝을 효과적으로 수행함

* 파이썬버전 3.12에서는 tensorflow와 연동이 안되므로 하위버전을 사용하도록
* 구글코랩의 실행과정: 구글 클라우드를 통해 실행 / 코랩을 통해 파일 편집 / 구글 드라이브에 저장­­­­

**3강 마켓과 머신러닝**

Ex) 사례: 길이와 무게를 활용하여 판매되는 생선의 종을 자동으로 인식할 수 있는 프로그램 제작

\*If fish length >=30과 같은 코드를 쓰지 않고 머신러닝을 활용하는 이유 : 이러한 규칙을 미리 규정하기 어려울 때가 존재함

* 머신러닝 관련 용어:
  + Class: 분류가 되는 것들 ex) 도미 & 방어
  + Classification: 분류과정
  + Binary classification: 두가지로 분류하는 과정
  + 샘플: 데이터 내의 각각의 수치로 이에 대응되는 수치는 특성이라고 함

Ex) 도미\_length = [25.4, 26.3]에서 25.4는 하나의 샘플

도미\_weight = [242.2, 290.0]에서 242.2는 위 샘플의 특성

* 산점도(scatterplot) 관련 코드:
  + *Import matplotlib.pyplot as plt* : 산점도 메써드가 포함된 library 불러오기
  + *Plt.scatter(x축데이터, y축데이터)* : 산점도 생성 \*축은 각 데이터에 자동으로 맞춰짐
    - Plt.scatter()를 두개 연속으로 쓰면 하나의 산점도에 두개의 그래프가 그려짐
  + *Plt*.xlabel(‘x축명칭)’ / plt.ylabel(‘y축명칭)’ : 레이블링
  + *Plt.show()* : 그래프 출력
* 연산자 오버라이딩: 리스트끼리의 덧셈은 두개의 리스트를 하나의 리스트를 병합해줌
* 사이킷런이 기대하는 데이터 형태는 2차원 배열 [[ \_\_ , \_\_ ]]
* *Fish\_data = [[1, w] for 1, w in zip(length, weight)]*에서 length는 1에, weight는 w에 들어가서 2차원 배열이 만들어진다.
* 이진분류의 경우 정답 형식: 정답개수만큼 1을, 오답개수만큼 0을 넣어 만든 리스트 제작
* K-최근접 이웃: 머신러닝 알고리즘의 일종
  + 알고리즘 원리:
    - 샘플 주의에 있는 클래스 중 가장 많은 클래스로 인식(default = 5)
      * Ex) 거리상 가장 가까운 5개의 클래스 중 3개가 1이면 1로 예측
  + 사용방법(이진분류):
    - *From sklearn.neighbors import KNeighborsClassifier*
    - *Kn = KNeighborsClassifier() :* Kn을 클래스의 객체로 지정
      * *KNneighborsClassifier(n\_neighbors=49)*라고 하면 49개의 값이 기준값이 됨
    - *Kn.fit(훈련세트, 훈련타겟)*
    - *Kn.score(테스트세트, 테스트타겟)*
    - *Kn.predict(새로운 샘플)* : 훈련을 토대로 샘플이 0인지 1인지 예측

**4강 훈련세트와 테스트 세트로 나누어 사용하기**

* 머신러닝의 종류
  + 지도학습: 가르침(타겟데이터)을 기반으로 하는 머신러닝 모델
  + 비지도학습: 가르침 없이 입력데이터만으로 학습이 이루어지는 머신러닝 모델
  + 강화학습: 모델이 행동을 수행한 다음에 주변의 환경에서 행동의 결과에 대한 피드백을 받아 지속적인 개선을 거치는 과정 -> 환경요인을 고려한 학습
* 훈련 세트와 테스트 세트의 생성
  + 주의할점 : 훈련세트와 테스트 세트를 나눌 때는 샘플링 편향이 일어나지 않도록 조심해야 한다. 🡪 테스트 세트와 훈련세트에 두 클래스가 골고루 섞여 들어가는 것
  + 넘파이 사용(많이 쓰이므로 익히는 것이 좋음)
    - 차원의 도메인에 따른 정의
      * 배열: 축의 개수
      * 벡터: 벡터의 개수
    - 주의할 점: 넘파이 배열에는 서로 다른 종류의 타입을 넣을 수 없음
    - Sklearn에 맞는 행렬형태로 데이터 변환하기
      * *Np.array()* : 를 이용해 쉽게 행렬 생성 가능
    - 데이터 골고루 섞기
      * 원리: 데이터의 개수만큼 index 리스트를 만들고 이를 셔플한한다. 그리고 나서 섞인 인덱스의 배열에 따라 train set과 test set를 나눈다.
        + *Index = np.arrange( : 샘플의 개수)* : 인덱스 리스트 생성
        + *Np.random.shuffle(index)* : 인덱스를 무작위로 섞는다

*Np.randomseed()* : 이후의 무작위 추출과정에서 재현가능 하도록 무작위의 결과를 동일하게 만드는 것.(실전에선 안 쓰임, 교육용)

* + - * + *train\_input = input\_arr[index[샘플의 개수 : ]]* : 배열 슬라이싱을 하여 인덱스가 담긴 배열을 넣어서 무작위 인덱스에 따라 다른 배열 안의 원소를 선택
  + 데이터 산점도로 확인
    - Plt.scatter(train\_input[모든 행 , 샘플 해당 열], train\_input[ 모든행, 특성 해당 열] : 에서
    - 테스트용
    - 레이블링
    - Plt.show()

**5강 정교한 결과 도출을 위한 데이터 전처리 알아보기**

* 전처리
  + *Np.Column\_stack(*샘플 데이터, 특성 데이터*)* : 두 배열이 각각 열의 형태로 병합해서 행렬을 이룸 \*np.row\_stack()도 가능
  + *Np.ones(중복 수)* : 1이 중복 수 만큼 생성됨
  + *Np*.concatenate(배열1, 배열2) : 두 배열을 이어 붙임
  + *Npfull((행,열),수)* : 수로 가득 채워진 행렬이 만들어짐
  + 사이킷런으로 데이터 나누기
  + *From sklearn.model\_selection import train\_test\_split* : apTjem 불러오기
  + *Train\_input, test\_input, train\_target, test\_target*

*= train\_test\_split(fish\_data, fish\_target, stratify=fish\_target, random\_state=42)*

* + - Stratify는 매개변수로 fish\_target을 정답으로 인지
    - Random\_state는 랜덤씨드
* 새 데이터 예측하기
  + *Distances, indexes = kn.kneighbors([[25. 150]])* : 새 데이터와 가장 가까운 샘플 5개의 거리랑 인덱스 반환
  + *Plt.scatter(25, 150, marker = ‘^’)* : [[25,150]] 데이터를 산점도 내에서 삼각형으로 표현
  + *Plt.scatter(tran\_input[indexes, 0], train\_input[indexes,1], markder = ‘D’)*  : 추출한 5개의 인덱스에 따라 점 대신 마름모를 생성
* K-최근접 알고리즘을 사용할 때 주의할 점: x축과 y축의 스케일 크기를 동일하게 해야 함

\* 스케일이 다르면 한 가지 요소에 따라서만 거리가 좌우되어 올바른 예측 불가능

* + Plt.xlim((0, 10000)): x축의 스케일을 0에서 1000으로 맞춤
  + 표준점수로 바꾸기
    - 표준점수 구하는 법: (훈련샘플(=train\_input) – 평균)/표준편차 = z정수(= 표준점수)
    - Mean = np.mean(train\_input, axis = 0) \*axis = 0은 열에 따라 평균계산한다는 의미
    - Std = np.std(train\_input, axis = 0)
    - Train\_scaled(z-score) = (train\_input – mean) / std

\*행렬끼리 뺴면 자동으로 위치상으로 대응되는 것들끼리 뺼셈 = 넘파이브로드캐스팅

* + - 전처리(z스코어) 후 훈련과정에 들어가야함, kn.fit(), kn.score()
    - 새로운 데이터를 예측할 때도 기준에 맞춰 변환한 후 예측

**6강 회귀 문제를 이해하고 k-최근접 이웃 알고리즘으로 풀어보기**

* 회귀모델: 수치를 예측하는 것 ex) 농어의 무게를 예측하라
  + 이전엔 target이 0,1이였지만 회귀(regression)에서는 임의의 숫자가 target임
* K-최근접 이웃 회귀 원리: 최근접 이웃에 따라 그들의 수치를 평균 내어 예측 값 도출

\*회귀 알고리즘 같은 경우엔 특성이 2개 이상일 때 산점도 말고 다른 그래프 활용

* 산점도에 그리기
  + 앞선 모델에 따라 생각해보면 height는 import data로 x 축에, weight는 타겟 데이터로 y축에 그림
* 훈련세트 준비:
  + Train\_test\_split()함수 적용 but 이번엔 length랑 weight를 따로 나누어 변수로 넣고 매개변수 stratify는 지정하지 않는다 \*정답 없음
* 현재 weight랑 length가 나뉘어져 있으니 행렬을 만들어도 열이 하나가 될 것이다.
  + *데이터변수이름.reshape(-1,1)* : -1은 남은 차원의 개수로 자동으로 구성되도록 하며 1은 열의 차원을 1개로 지정하는 것 🡪 행은 원소의 개수에 따라 구성됨
  + 타겟 데이터는 1차원 배열로 유지해도 괜찮음
* KN-최근접 회귀훈련모델 코드 작성법
  + *From sklearn.neighbors import KNeighborsRegressor*
  + *Knr = KNeighborsRegressor()*
  + *Knr.fit(train\_input, train\_target)*
  + *Knr.score(test\_input, test\_target)* : test input를 넣었을 떄 train input과 타겟 데이터에서 학습한 것을 토대로 예측을 하고 예측한 결과를 test\_target과 비교하여 정확도를 score로 출력
    - 비교방법: 친필, 폰트, 텍스트, 화이트이(가) 표시된 사진

      자동 생성된 설명 \*knr.score = R2(결정계수
      * 예측값이 평균과 일치한다면 결정계수가 0이 나올 것임 🡪 타겟값이 전체를 평균짓는 것만큼밖에 예측을 못하니 안 좋은 결과임
      * 예측값이 타깃과 동일하게 나올수록 R2의 값이 커짐 🡪 훈련 good
  + R2말고 다른 값을 기준으로 모델 평가 ex) mean\_absolute\_error(평균절대값 오차)
    - *Test\_prediction = Kdr.predict(test\_input)* : 예측한 결과 추출
    - *Mae = mean\_absolute\_error(test\_target, test\_prediction)* : 평균절대값 오차에 따라 test 타겟과 예측 결과 비교
    - *Mae가 결론적으로 평균절대값 오차가 되며 예측 데이터가 19정도만큼 오차가 난다고 볼 수 있음 \*절대값이므로 더 높은 지 낮은 지는 알 수 X*

**9강 로지스틱 회귀 알아보기**

* 로지스틱 회귀의 개념: z = a \* 무게 + b \* 길이 + c \* 대각선 + d \* 높이 + e \* 두께 + f
  + 선형함수를 학습하는 알고리즘
  + 이 상황에서 z의 값을 그대로 사용하면 회귀알고리즘이 되므로 확률형(0과 1 사이로 압축)으로 만들기 위해서는 시그모이드함수/로지스틱함수를 사용한다.
    - Z의 값이 무한대로 존재해도 0과 1사이로 그 출력값을 제한
    - 0.5보다 이하면 음성클래스, 0.5보다 크면 양성 클래스(=z가 0보다 작은 지 0보다 큰 지)
* 로지스틱 회귀(이진분류)