AI 프로그래밍 - KNN

Fly-AI in No.4

2023.12.27
대전대학교 PhD. 조 영복
ybcho@dju.ac.kr

머신러닝 데이터 전처리

❖ 데이터 전처리

- 레이블 인코딩
- 원핫인코딩

머신러닝 데이터 전처리

❖ 레이블 인코딩

```
from sklearn.preprocessing import LabelEncoder

items = ["tv", "냉장고", "컴퓨터", "전자레인지", "믹서", "선풍기", "믹서"]
encoder = LabelEncoder()
encoder.fit(items)
labels = encoder.transform(items)

print('labels:', labels)
print('encoder.classes:', encoder.classes_)
print('encoder.inverse_transform :',encoder.inverse_transform([4, 5, 3, 2, 4, 2, 0, 1, 1]))
```

```
labels: [0 1 5 4 2 3 2]
encoder.classes: ['tv' '냉장고' '믹서' '선풍기' '전자레인지' '컴퓨터']
encoder.inverse_transform : ['전자레인지' '컴퓨터' '선풍기' '믹서' '전자레인지' '믹서' 'tv' '냉장고' '냉장고']
```

머신러닝 데이터 전처리

```
    items
0      tv
1      냉장고
2      믹서
3      선풍기
4      전자레인지
5      컴퓨터
6      믹서
```

```
   items_tv  items_냉장고  items_믹서  items_선풍기  items_전자레인지  items_컴퓨터
0         1            0          0            0             0            0
1         0            1          0            0             0            0
2         0            0          0            0             0            1
3         0            0          0            0             1            0
4         0            0          1            0             0            0
5         0            0          0            1             0            0
6         0            0          1            0             0            0
```

```
print(df)

print(pd.get_dummies(df))
```

데이터 스케일링

- breast_cancer()
  - StandardScaler()
  - MinMaxScaler()
  - RobustScaler()
  - Normalizer()

SVM

❖ 데이터 스케일링

- 데이터 전처리
- 피처(feature)들마다 데이터 값의 범위가 다르기 때문에 범위 차이가 클경우 데이터를 갖고 모델링할때 0으로 수렴하거나 무한으로 발산
- 모든 피처들의 데이터 분포나 범위를 동일하게 조정
  - StandardScaler()
  - MinMaxScaler()
  - MaxAbsScaler()
  - RobustScaler()
  - Normalizer()

SVM-유방암 데이터 셋 실습

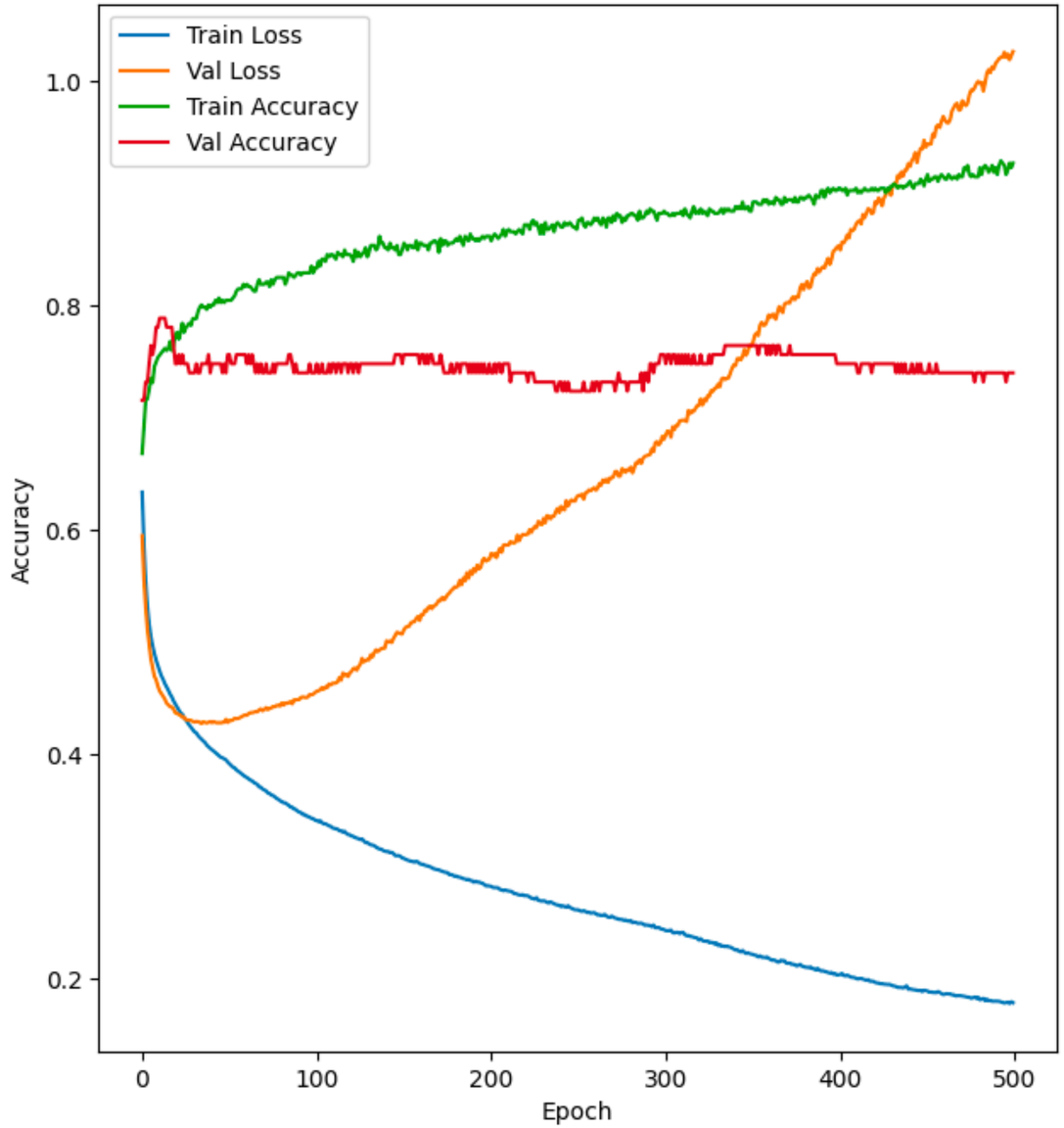
```
import sklearn
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_breast_cancer

cancer=load_breast_cancer()

cancer_df = pd.DataFrame(data=cancer.data, columns=cancer.feature_names)
cancer_df['target'] = cancer.target
cancer_df.head()
```

KNN

❖ Pandas

- Pandas
  - 데이터 처리와 분석을 위한 라이브러리
  - 행과 열로 이루어진 데이터 객체를 만들어 다룰 수 있음
  - 대용량의 데이터들을 처리하는데 매우 편리
  - pandas 자료구조
    - Series : 1차원
    - DataFrame : 2차원
    - Panel : 3차원
    - Pandas 로딩

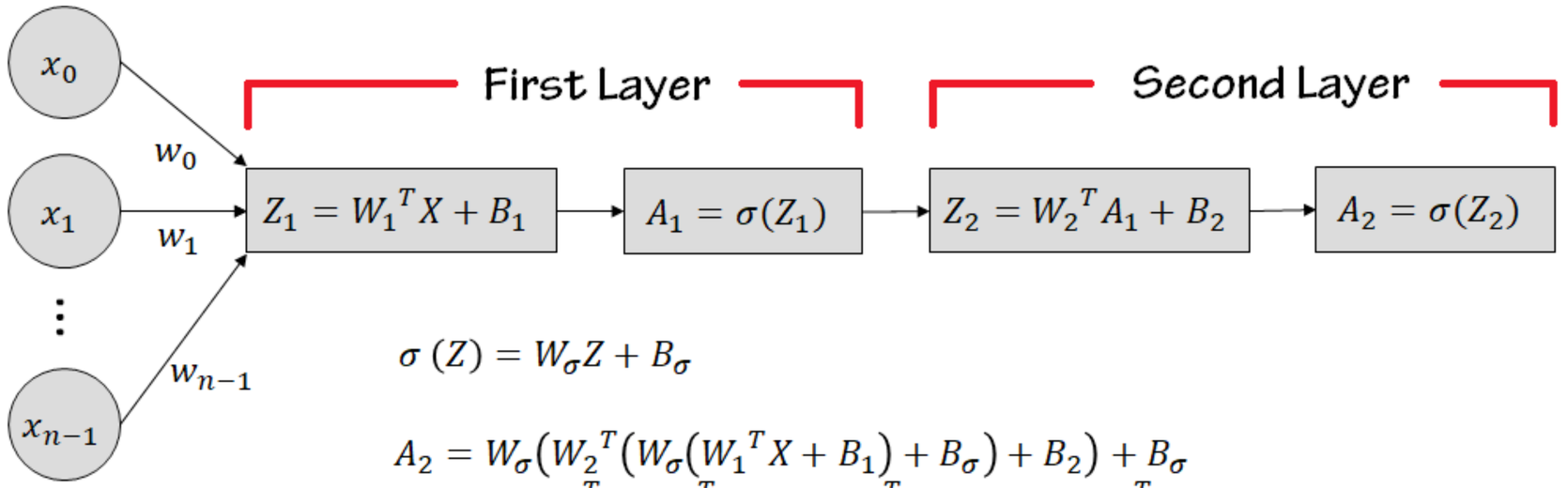
KNN-피마인디언 당뇨병 데이터



활성함수

❖뉴런이 다음 뉴런으로 신호를 보낼 때 입력신호가 일정 기준 이상이면 보내고 기준에 달하지 못하면 보내지 않을 수도 있다. 그 신호를 결정 해주는 것이 활성화 함수 (Activation Function)

- 선형함수
- 계단함수
- 시그모이드함수
- 탄젠트 함수
- ReLU 함수

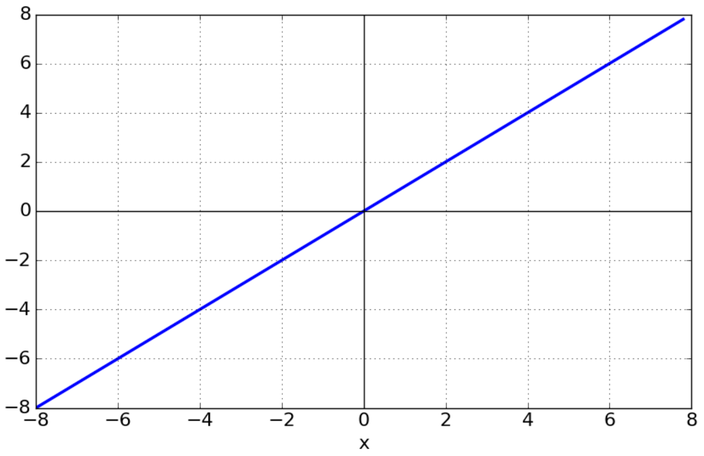


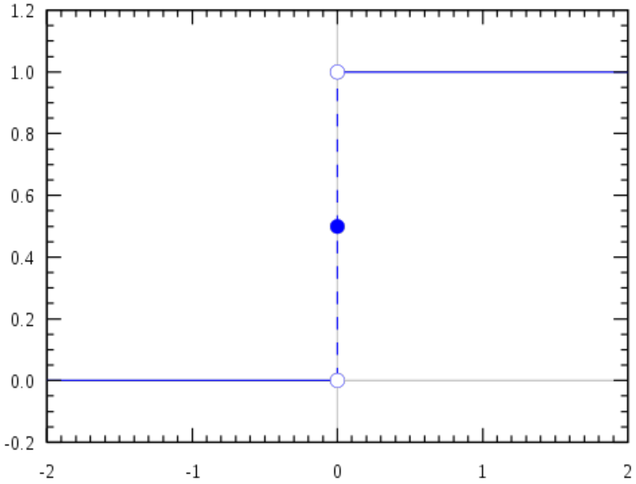
$$\sigma(Z) = W_\sigma Z + B_\sigma$$

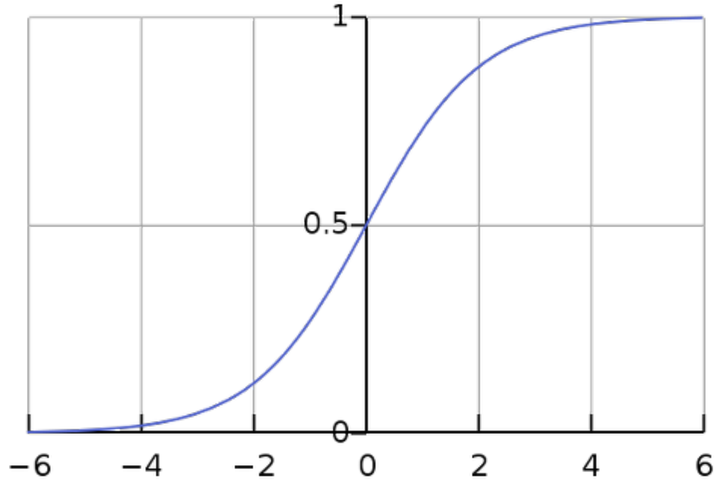
$$\begin{aligned} A_2 &= W_\sigma\left(W_2{}^T\left(W_\sigma\left(W_1{}^T X + B_1\right) + B_\sigma\right) + B_2\right) + B_\sigma \\ &= \underbrace{W_\sigma W_2{}^T W_\sigma W_1{}^T}_{W} X + \underbrace{W_\sigma W_2{}^T W_\sigma B_1 + W_\sigma W_2{}^T B_\sigma + W_\sigma B_2 + B_\sigma}_{B} \\ &= WX + B \end{aligned}$$

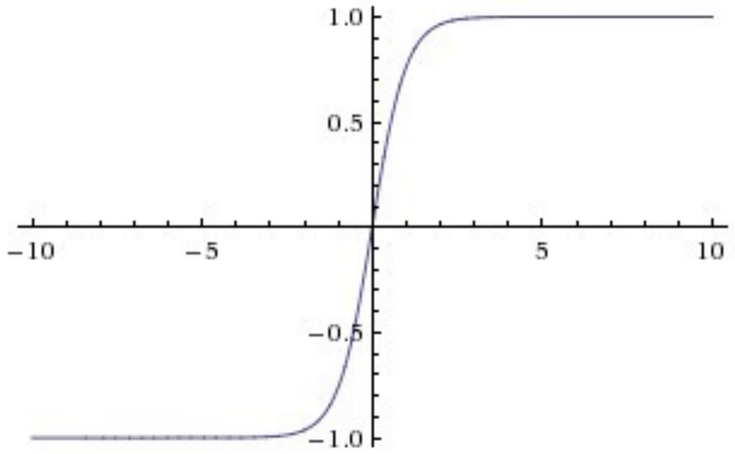
활성함수

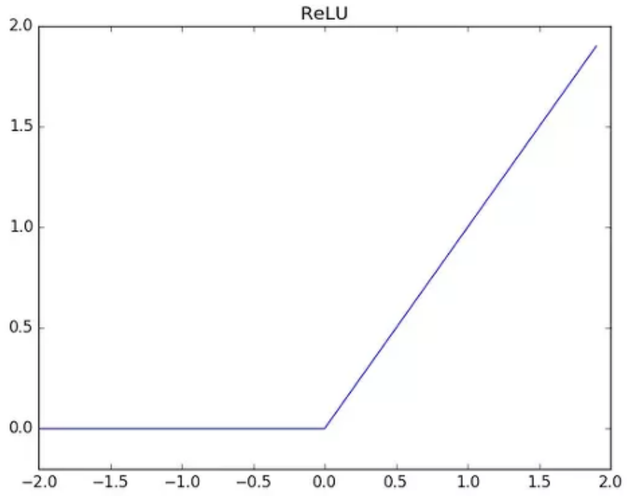
- 선형함수
- 계단함수
- 시그모이드함수
- 탄젠트 함수
- ReLU 함수











손실함수

❖ 손실 함수(loss function)란 머신러닝 혹은 딥러닝 모델의 출력값과 사용자가 원하는 출력값의 오차를 의미

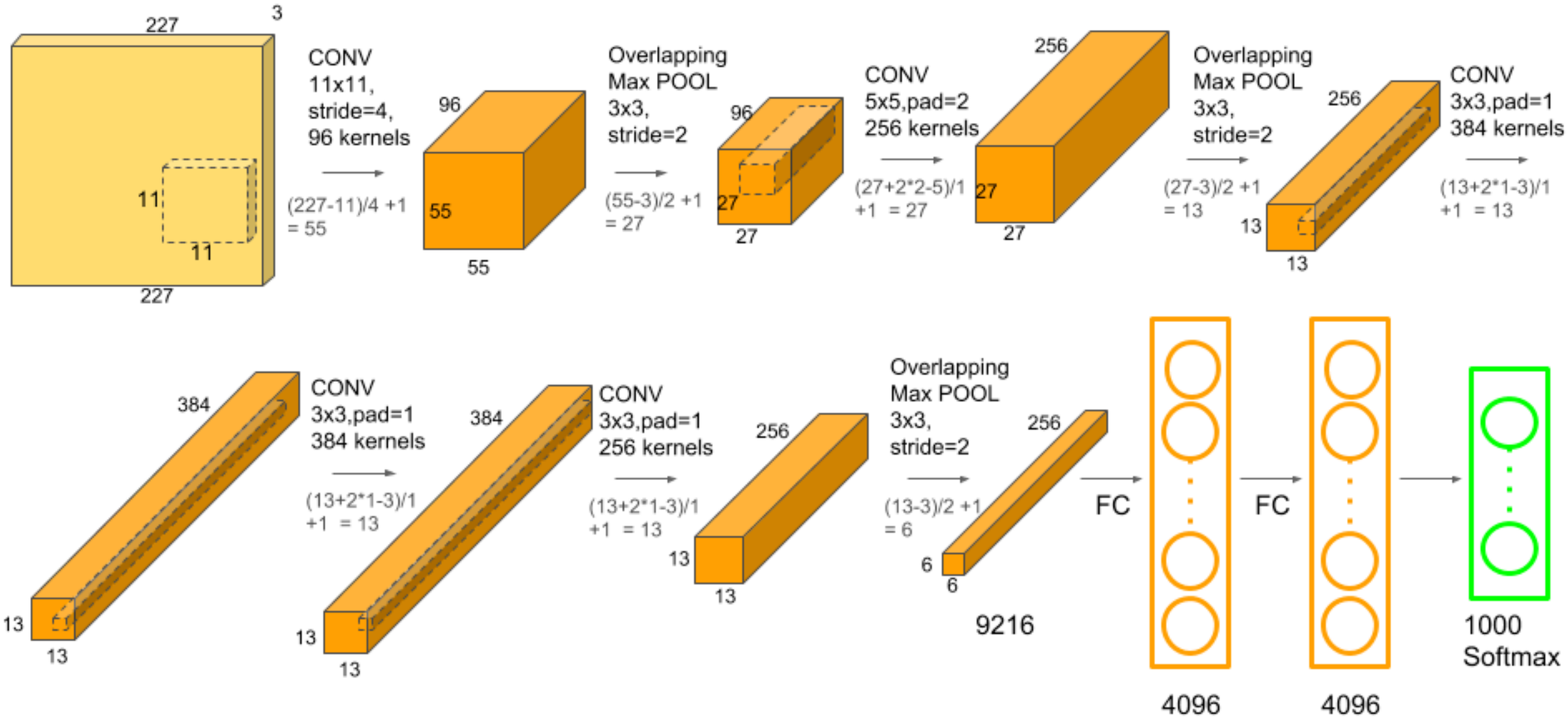
- 손실함수는 정답(y)와 예측(^y)를 입력으로 받아 실숫값 점수를 만드는데, 이 점수가 높을수록 모델이 안좋은 것
- 손실함수의 함수값이 최소화 되도록 하는 가중치(weight)와 편향(bias)를 찾는 것이 딥러닝 학습의 목표
- MSE
- RMSE
- Binary Crossentropy
- Categorical Crossentropy
- Sparse categorical crossentropy
- Focla loss

손실함수

❖ 손실 함수(loss function)란 머신러닝 혹은 딥러닝 모델의 출력값과 사용자가 원하는 출력값의 오차를 의미

- 손실함수는 정답(y)와 예측(^y)를 입력으로 받아 실숫값 점수를 만드는데, 이 점수가 높을수록 모델이 안좋은 것
- 손실함수의 함수값이 최소화 되도록 하는 가중치(weight)와 편향(bias)를 찾는 것이 딥러닝 학습의 목표
  - MSE
  - RMSE
  - Binary Crossentropy
  - Categorical Crossentropy
  - Sparse categorical crossentropy
  - Focla loss

CNN의 parameter 개수와 tensor 사이즈 계산



air-quality_no2_long

- air_quality를 타겟으로 분류 기준 설정하고 속성 분류보기
-

금융상품 갱신 여부 예측하는 인공신경망 구성하기

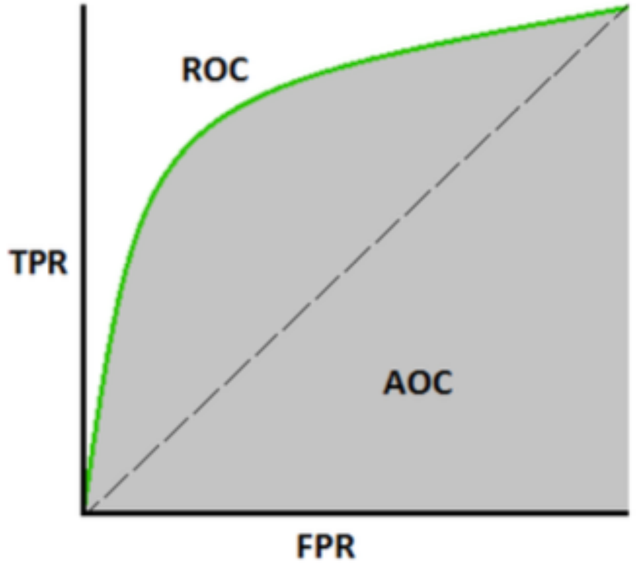
```
# 라이브러리 호출
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense

# 데이터 확인
df = pd.read_csv('./fly-AI/Churn_Modelling.csv')
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	E
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	

Confusion Matrix

❖ 예측값

- 정확도 (accurcy)
  - 전체의 경우 중에 맞은 경우에 대한 확률, (TN+TP) / (전체)
- 정밀도 (precision)
  - 맞다고 예측한 애들 중에 실제로 맞은것의 비율, TP / (FP+TP)
- 재현율 (recall)
  - 실제로 맞은 사람들 중에 맞았다고 예측한 애들의 비율, TP / (FN+TP)



- F1 스코어
  - 정밀도와 재현율은 서로 트레이드오프 관계, 둘 다 높은 점수를 맞아야 좋은 모델이라 할 수 있음
  - F1 스코어는 정밀도와 재현율의 조화 평균, 2 / [(1/recall) + (1/precision)]
- ROC, AUC
  - 맞출 때마다 TPR(수직) 방향으로 한 칸씩 올라갑니다. 그리고 틀리면 FPR(수평) 방향으로 이동

Confusion Matrix

❖ 예측값

- 정확도 (accurcy)
  - from sklearn.metrics import accuracy_score
  - accuracy_score(정답, 예측)
- 정밀도 (precision)
  - from sklearn.metrics import precision_score
  - precision_score(정답, 예측)
- 재현율 (recall)
  - from sklearn.metrics import recall_score
  - recall_score(정답, 예측)
- F1 스코어
  - from sklearn.metrics import f1_score
  - f1_score(정답, 예측)
- ROC, AUC
  - from sklearn.metrics import roc_auc_score
  - roc_auc_score(정답, 예측)

Confusion Matrix

❖ 데이터 스케일링 라이브러리

- from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler,RobustScaler, Normalizer
  - StandardScaler()
    - 평균이 0이고 분산이 1인 정규 분포로 만드는 것
    - 표준편차란 평균으로부터 얼마나 떨어져있는지를 구한 것
  - MinMaxScaler()
    - 최대값은 1 최소값은 0으로 최소-최대 정규화 Min-Max Normalization
    - 이상치에 취약.
  - MaxAbsScaler()
    - MaxAbsScaler는 MinMaxScaler와 비슷
    - 방법의 이름에서도 알 수 있듯이 모든 피처들의 절댓값이 0과 1 사이에 놓이도록 만들어줌.
    - 즉, 0을 기준으로 절댓값이 가장 큰 수가 1또는 -1의 값을 가짐.
    - 마찬가지로, 이상치의 영향을 크게 받기 때문에 이상치가 존재할 경우 이 방법은 적절하지 않음.

Confusion Matrix

❖ 데이터 스케일링 라이브러리

- from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler,RobustScaler, Normalizer
  - RobustScaler()
    - StandardScaler는 평균과 분산을 사용했지만 RobustScaler는 중간값(median)과 사분위값(quartile)을 사용
    - 이상치의 영향을 최소화
    - 표준화 후 데이터가 더 넓게 분포
  - Normalizer()
    - 열(Columns)을 대상으로 함
    - 한 행의 모든 피처들 사이의 유클리드 거리가 1이 되도록 데이터값 생성
    - 좀 더 빠르게 학습할 수 있고 과대적합 확률을 낮출 수 있다

결정트리(Decision Tree)

❖ Decision Tree

- 의사결정 규칙을 나무 구조로 나타내어 전체 자료를 몇 개의 작은 집단으로 나누어서 분석하는 기법
- SVM 처럼 결정트리(decision tree)는 분류 및 회귀가 가능한 머신러닝 알고리즘
- 매우 복잡한 데이터셋도 학습할 수 있는 강력한 알고리즘
- 최근에 많이 사용하는 랜덤 포레스트의 기본 구성 요소
  - Root Node : 깊이가 0인 꼭대가 노드
  - Leaf Node : 자식 노드가 없는 마지막 노드
  - Gini Impurity : 한 노드의 모든 샘플이 같은 클래스에 속해있으면, 해당 노드는 순수(gini=0)
- **결정 트리의 장점** : 스케일이나 평균을 원점에 맞추는 것과 같은 **데이터 전처리가 거의 필요하지 않음**.
- **사이킷런**: 이진 트리(자식 노드의 수가 2개 이하)만 만드는 CART 알고리즘을 사용.
- **결정 트리**: 위와 같이 매우 직관적이고 이해하기 쉽고, 해석력이 아주 좋다. ('화이트 박스')

결정트리(Decision Tree)

❖ 규제 매개변수

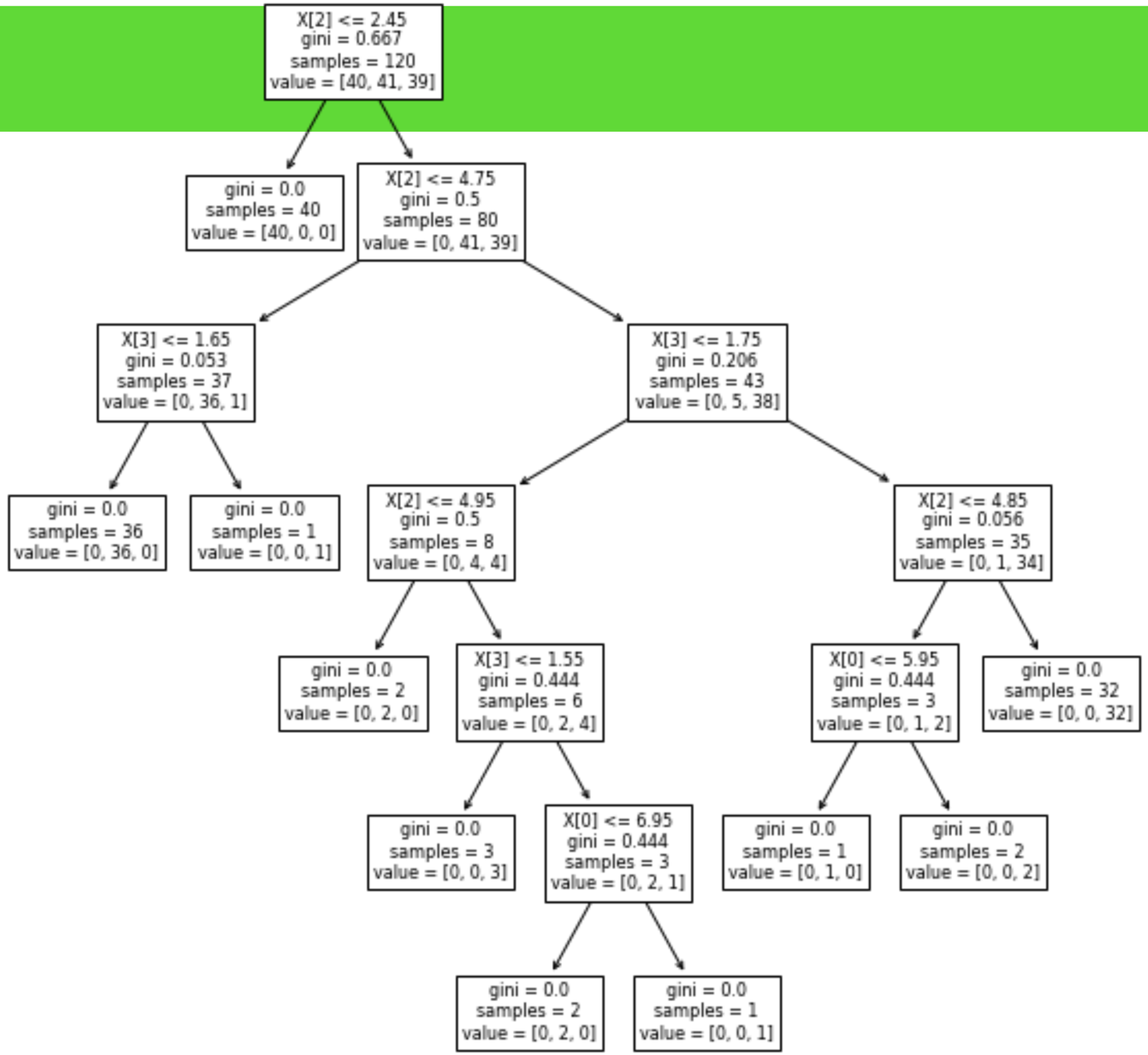
- 결정 트리의 경우 훈련 데이터에 대한 가정을 보통 두지 않음(선형모델은 데이터가 선형일거라 가정)
- 훈련 데이터에 대한 일반적인 가정을 두지 않는다면, 모델 자체가 훈련 데이터와 아주 가깝게 만들려고 해서 과접합이 발생할 수 있음
- 결정 트리 모델의 경우에는 일반적인 선형 모델과 같이 파라미터 모델이 아니다
- 결정 트리 모델은 훈련되기 전에 파라미터의 수를 지정할 수 없는 비파라미터 모델(non-parametricmodel).
- 비파라미터 모델은 훈련 데이터에 맞춰지기 때문에 모델 구조가 고정되지 않고 자유롭다.
- 파라미터 모델은 미리 파라미터를 정할 수 있기 때문에 제한적이지만 과대적합의 위험을 조절할 수있음.
- 결정 트리는 보통 최대 깊이(max_depth)로 모델의 과적합을 규제할 수 있음.
- max_depth가 낮아지면 과대적합의 위험이 감소

결정트리(Decision Tree)

❖ 결정 트리의 규제 매개변수 종류

- **max_depth** : 트리 최대 깊이 : max_depth 감소 → 모델 규제 증가 →**과적합 감소**
- **min_samples_split** : 분할되기 위해 노드가 가져야 하는 최소 샘플 수
  - min_samples_split 증가 → 모델 규제 증가 → 과적합 감소
- **min_samples_leaf** : leaf node가 가지고 있어야할 최소 샘플 수
  - min_samples_leaf 증가 → 모델 규제 증가 → 과적합 감소
- **min_weight_fraction_leaf** : min_samples_leaf와 비슷, 가중치가 부여된 전체 샘플 수에서의 비
  - min_weight_fraction_leaf 증가 → 모델 규제 증가 → 과적합 감소
- **max_leaf_nodes** : leaf node의 총 최대 개수
  - max_leaf_nodes 감소 → 모델 규제 증가 → 과적합 감소
- **max_features** : 최상의 분할을 찾을 때 고려할 기능의 수
  - max_features 감소 → 모델 규제 증가 → 과적합 감소

Iris data set



회귀분석

❖ 회귀분석이란

- 독립변수(x)로 종속변수(y)를 예측하는 것
  - 독립변수: 변수의 변화 원인이 모형 **밖**에 있는 변수
  - 종속변수: 변수의 변화 원인이 모형 **안**에 있을 변수

체중이 식사량에 따라 달라진다고 가정하면,
- 식사량이 많아지면 체중도 증가하고,
- 식사량이 감소하면 체중도 감소할 것
- 그러면 체중은 식사량에 종속되었다(체중은 종속변수).

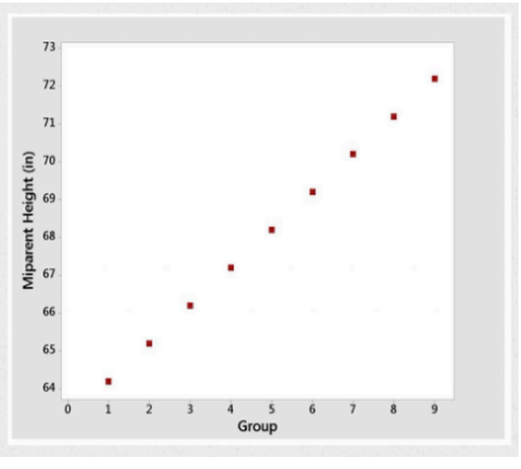
회귀분석

❖ 선형회귀분석이란

- 독립변수와 종속변수 사이에 직**선**적인 **형**태의 관계가 있다고 가정
- 직선적인 형태란
  - 독립변수가 일정하게 증가하면, 종속변수도 그에 비례해서 증가하거나 또는 감소하는 형태
  - 선형 모형에서 x와 y의 관계를 수식

밥을 한 그릇 더 먹으면 체중이 정확히 100g 증가한다면 직선적인 형태

- 밥을 한 그릇, 두 그릇, 세 그릇, .. 먹으면
- 체중이 100g, 200g, 300g으로 증가
- 이것을 그림으로 그려보면 직선



회귀분석

❖ 회귀분석을 통해 우리는 무엇을 알수 있을까?

- **모형적합도**: 모형이 데이터에 얼마나 잘 맞는가?
  - 예) 식사량과 체중의 관계가 데이터에 잘 맞는지 검증해볼 수 있다.
- **회귀계수**: 독립변수의 변화가 종속변수를 얼마나 변화시키는가?
  - 예) 식사량이 증가하면 체중이 얼마나 증가하는지 알 수 있다.

앙상블 보팅

❖앙상블 학습

- 보팅, 배깅, 부스팅
- 보팅은 기본적으로 여러 개의 분류기를 사용하여 각각의 분류기 결과를 투표하여 예측
- 보팅(Voting)
  - 서로 다른 알고리즘을 가진 분류기의 조합, 다른 알고리즘을 사용하여 분류기를 조합
  - 하드 보팅과 소프트 보팅
  - 하드 보팅이란, 각각 분류기의 결괏값 중 가장 많은 걸 따른다.
  - 소프트 보팅이란, 분류기의 확률을 더하고 각각 평균을 내서 확률이 제일 높은 값으로 결괏값을 선정

앙상블 보팅

❖앙상블 학습

- 배깅(Bagging)
  - 보팅과 다르게 서로 같은 알고리즘의 분류기 조합
  - 분류기 1의 데이터는 [0,0,3,4,5,5] 분류기 2의 데이터는 [0,1,2,3,4,5] 분류기 3의 데이터는 [0,1,1,2,4,5] 이런 식으로 개별 데이터의 중첩을 허락