AI는 꽃을 구분할 수 있을까?

2024 AIML AI Winter School

Jehyeon Park,

Dept. of Artificial Intelligence Convergence

Hallym University, Republic of Korea,

Under Supervision of Prof. Dong-Ok Won

2024. 01. 09.

목차

- 데이터 분류와 분류 알고리즘
- 시각적으로 이해하는 데이터 분류(Artificial Neural Network)
- 데이터셋 불러오기 및 전처리
- 분류 모델에 따른 분류 성능 비교
- $+\alpha$

데이터 분류와 분류 알고리즘

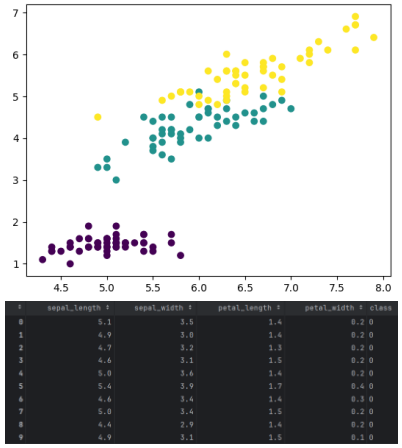
- **분류란?**
  - ➢ **머신러닝에서 분류란?**
    - ✓ 입력 데이터 X가 들어왔을 때 해당 데이터가 속한 클래스인 Y 를 예측하는 것
  - ➢ **어떻게 하면 컴퓨터가 데이터를 분류할 수 있을까?**
    - ✓ 이 과제를 해결하기 위한 다양한 머신 러닝 방법론들이 존재한다.

데이터 분류와 분류 알고리즘

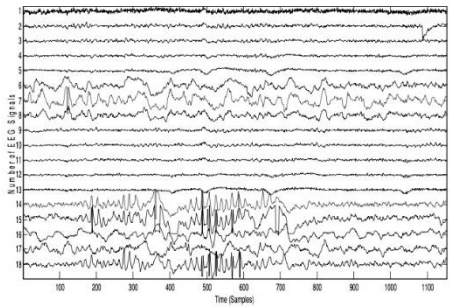
- 머신러닝을 활용한 데이터 분류

  - 데이터의 종류
    - ✓ 연속성이 없는 수치형 데이터 (ex. Iris Dataset, Boston Housing Dataset)
    - ✓ 연속성이 있는 시계열 데이터 (ex. 뇌파 신호, 주가 예측, 음성 데이터)
    - ✓ 영상 데이터 (ex. Image, Video)
    - ✓ Etc...

분류하고자 하는 데이터에 맞는 분류 모델을 선택해야 한다. 이 강의에서는 연속성이 없는 수치형 데이터를 활용해서 꽃을 분류하고자 한다.



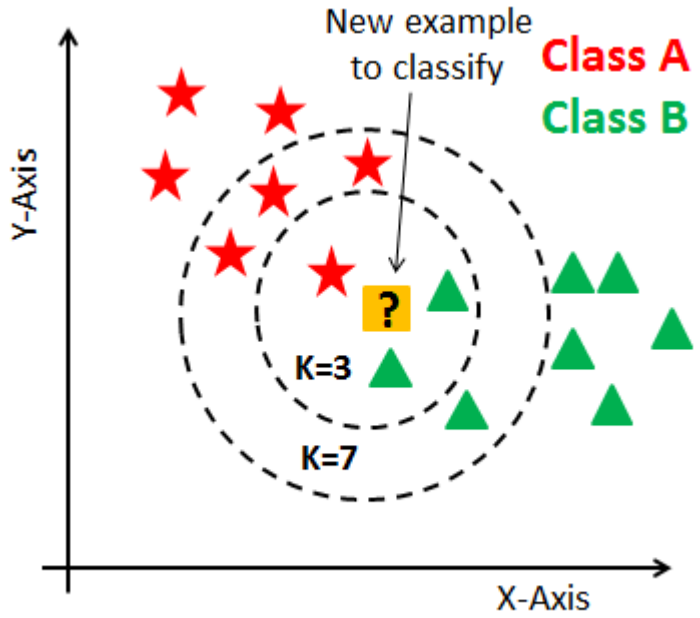
	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
7	5.0	3.4	1.5	0.2	0
8	4.4	2.9	1.4	0.2	0
9	4.9	3.1	1.5	0.1	0

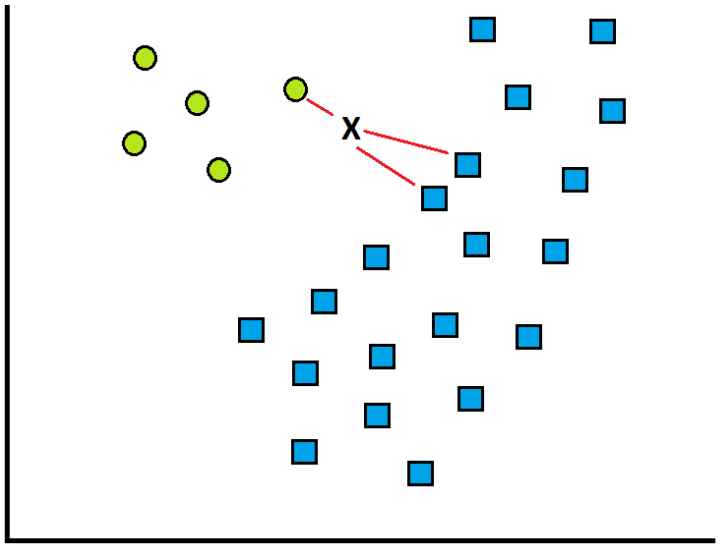


데이터 분류와 분류 알고리즘

- 데이터 분류를 위한 머신러닝 알고리즘

  - K-Nearest Neighbors
    - ✓ 자신과 가장 가까운 K개의 데이터 중 다수가 속한 레이블로 분류하는 방법론
    - ✓ 거리에 따른 가중치를 줘서 분류하는 Weighted-KNN과 같은 상위 모델 존재

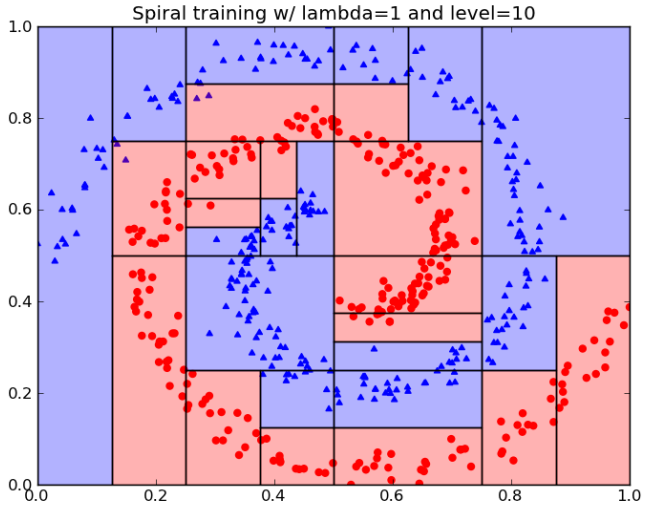


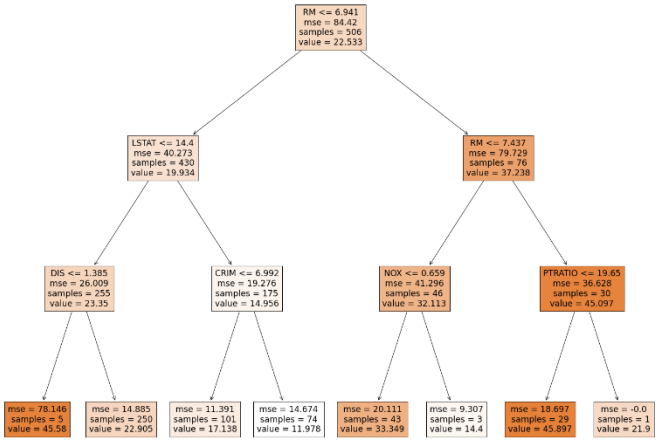


데이터 분류와 분류 알고리즘

- 데이터 분류를 위한 머신러닝 알고리즘

  - Decision Tree
    - ✓ Tree 방식을 데이터 분류에 활용
    - ✓ Entropy(불확실성)이 가장 적은 곳에서 분기를 나눔(Partitioning)
    - ✓ 후에 Tree가 너무 복잡해지지 않도록 가지치기(Pruning)



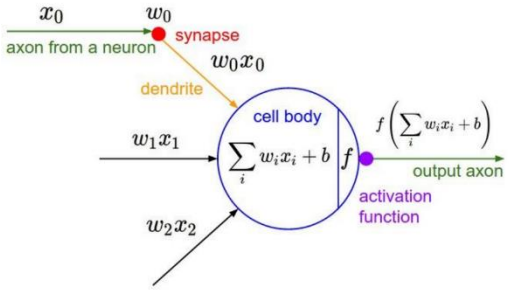


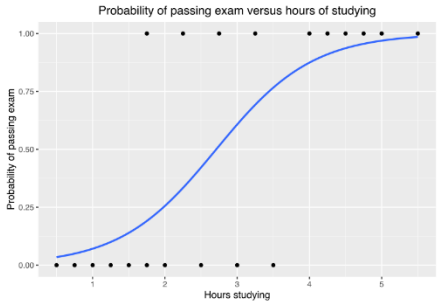
데이터 분류와 분류 알고리즘

- 데이터 분류를 위한 머신러닝 알고리즘

➢ Logistic Regression

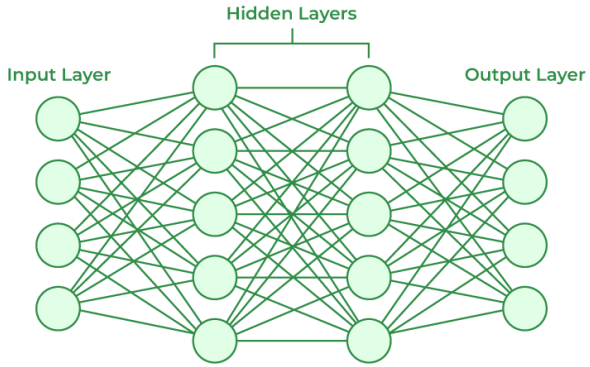
✓ 각 입력값과 가중치의 Weighted-Sum에 Sigmoid 함수를 적용하여 분류





➢ Artificial Neural Network

✓ Logistic Regression을 여러 겹 쌓은 형태

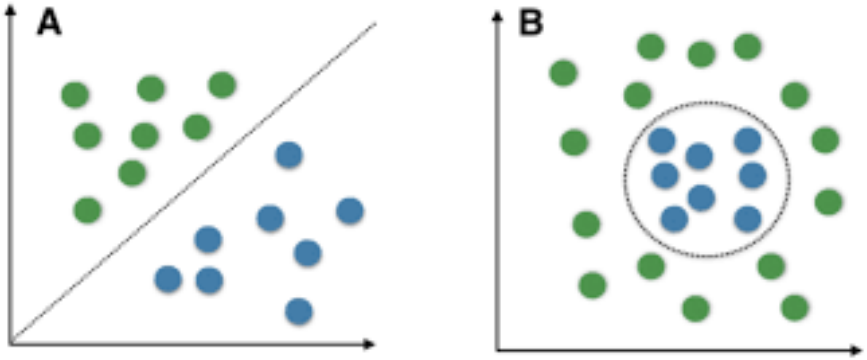


시각적으로 이해하는 데이터 분류(ANN)

- 뉴럴 네트워크 뜯어보기

  - 분류용 뉴럴 네트워크의 최종 목적
    - ✓ 데이터를 가장 잘 분류하는 하나의 직선을 찾는 것(2차원 기준)

  - 데이터를 하나의 직선으로 분류하기 위해서는 데이터를 Linearly Separable하게 만들어야 한다.



    - ✓ A의 경우 Linearly Separable하지만, B의 경우에는 어떻게 해결해야 할까?
    - ✓ => 비선형 활성화함수를 사용한다.

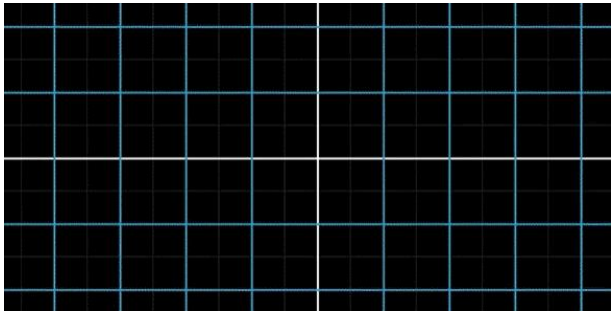
https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

시각적으로 이해하는 데이터 분류(ANN)

- 뉴럴 네트워크 뜯어보기

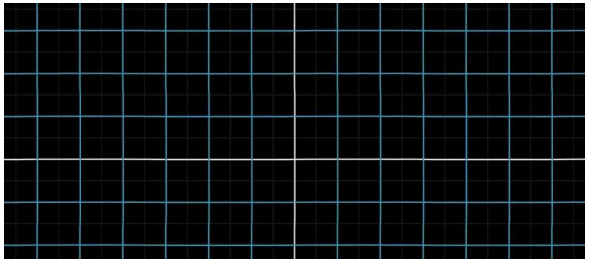
  - 만약 뉴럴 네트워크에 활성화함수가 없다면?
    - ✓ 비선형 활성화함수가 없는 뉴럴 네트워크는 단순 선형변환이다.
    - ✓ 더 정확하게는, W를 행렬곱하는 과정은 선형변환이다.

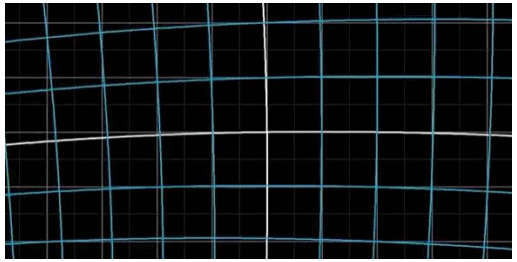
  - 선형 변환
    - ✓ 중점이 변하지 않는다.
    - ✓ 격자의 형태가 직선을 유지한다.
    - ✓ 격자와 격자 사이의 간격이 일정하다.



    - ✓ 따라서 선형 변환만으로는 원형 데이터를 Linearly Separable하게 만들 수 없다.

  - 비선형 변환
    - ✓ 비선형 변환의 종류는 너무 많기 때문에 두 개의 예시만 보여주고 넘어가겠다.

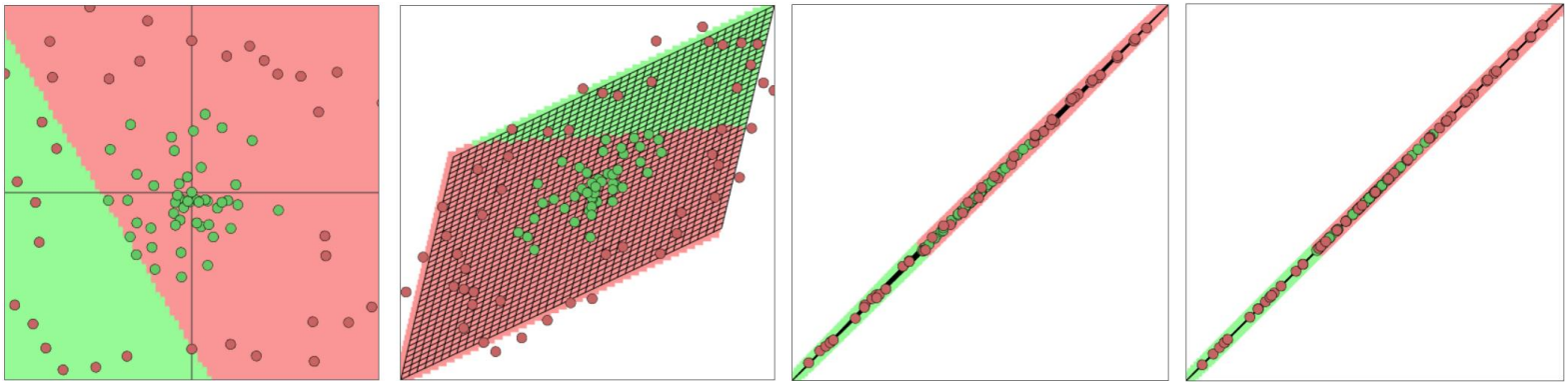




시각적으로 이해하는 데이터 분류(ANN)

- 뉴럴 네트워크 뜯어보기

  - 비선형 활성화함수가 없는 뉴럴 네트워크를 시각화해보자



Layer 0 Layer 1 Layer 2

  - ✓ 이전 페이지에서 언급한 대로 선형 변환만으로는 데이터를 Linearly Separable하게 변환할 수 없다.

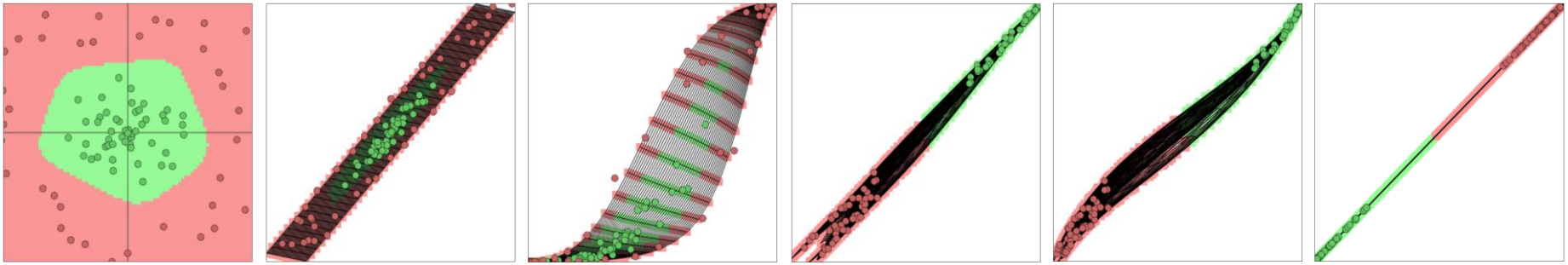
  - ✓ 따라서 모델에 비선형성을 추가해줘야 한다.

https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

시각적으로 이해하는 데이터 분류(ANN)

- 뉴럴 네트워크 뜯어보기

  - 비선형 활성화함수가 포함된 뉴럴 네트워크를 시각화해보자



  - ✓ Layer 2를 보면 모델이 데이터를 완벽하게 Linearly Separable하게 변환한 것을 확인할 수 있다.

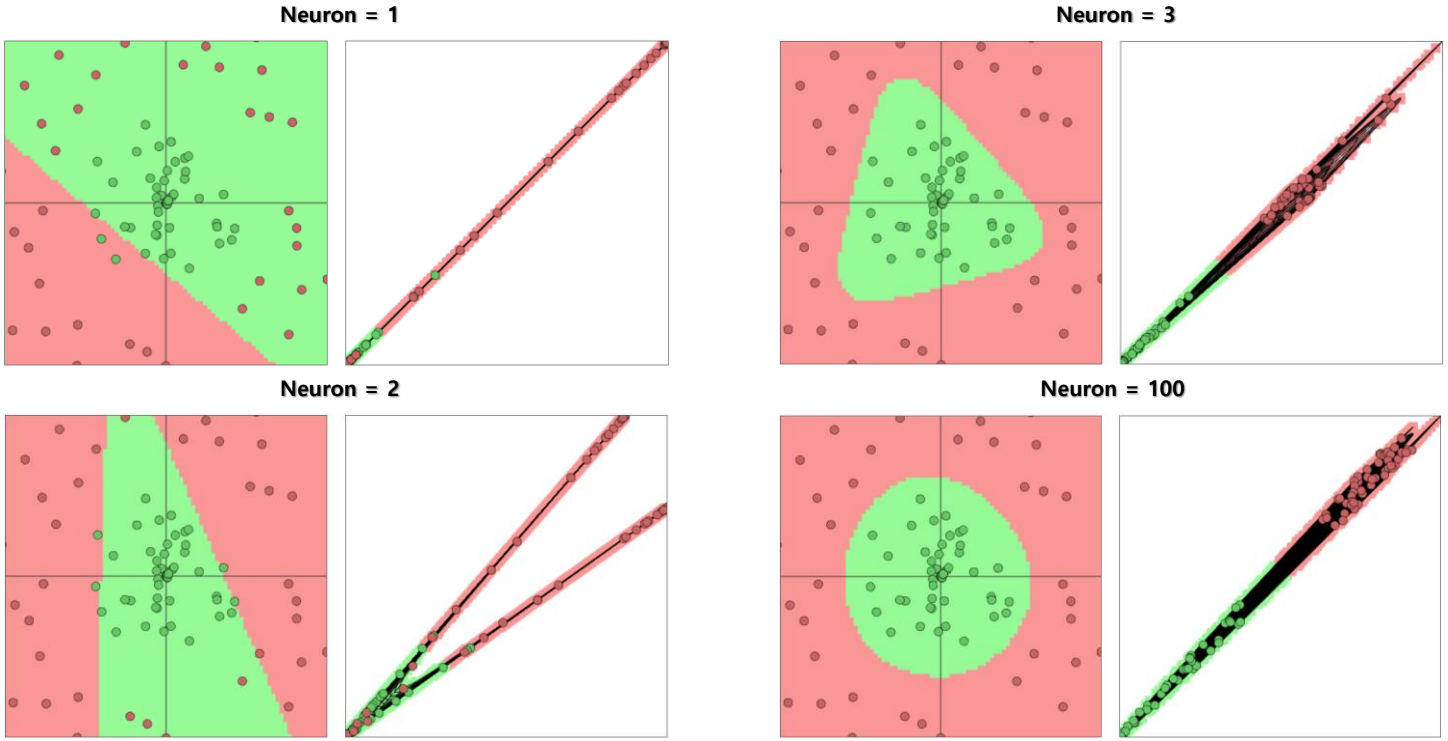
  - ✓ 모델에 비선형성을 추가하면 위와 같은 원형 데이터도 분류할 수 있게 된다.

https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

시각적으로 이해하는 데이터 분류(ANN)

- 뉴럴 네트워크 뜯어보기

  - 퍼셉트론의 수에 따른 분류 성능 변화



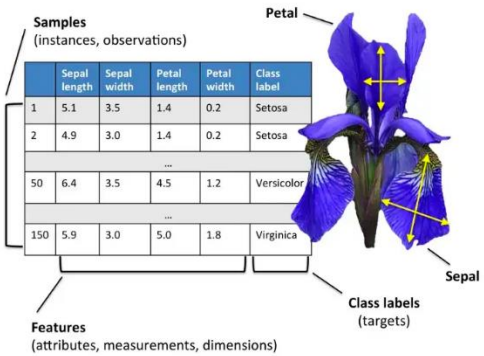
✓ 퍼셉트론의 수가 많아질 수록 데이터를 더 잘 표현할 수 있게 된다.

✓ 퍼셉트론의 수가 많아질 수록 더욱 고차원의 Decision Boundary를 만들 수 있게 된다.

https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

데이터 획득 및 전처리

- 활용할 데이터셋
  - Iris Dataset
    - ✓ 붓꽃의 꽃잎, 꽃받침의 길이, 너비 정보를 활용, 붓꽃의 종을 분류하기 위한 데이터셋
    - ✓ 일련번호, 꽃잎의 길이와 너비, 꽃받침의 길이와 너비, 종으로 구성
    - ✓ 총 150개의 데이터



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

- 데이터셋 불러오기

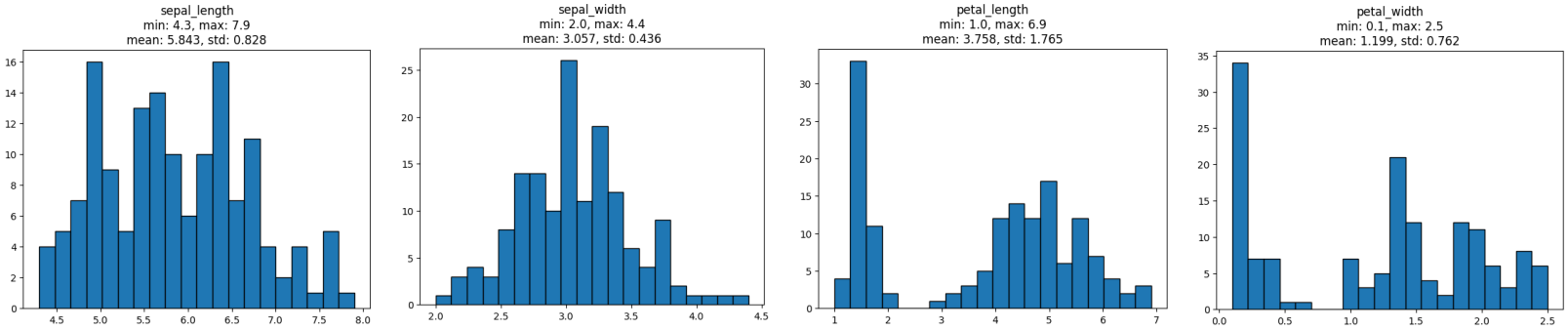
```
from sklearn.datasets import load_iris

iris = load_iris()
df = pd.DataFrame(iris.data, columns = iris.feature_names)
df['class'] = iris.target

df
```

1-10 150 rows × 5 columns pd.DataFrame

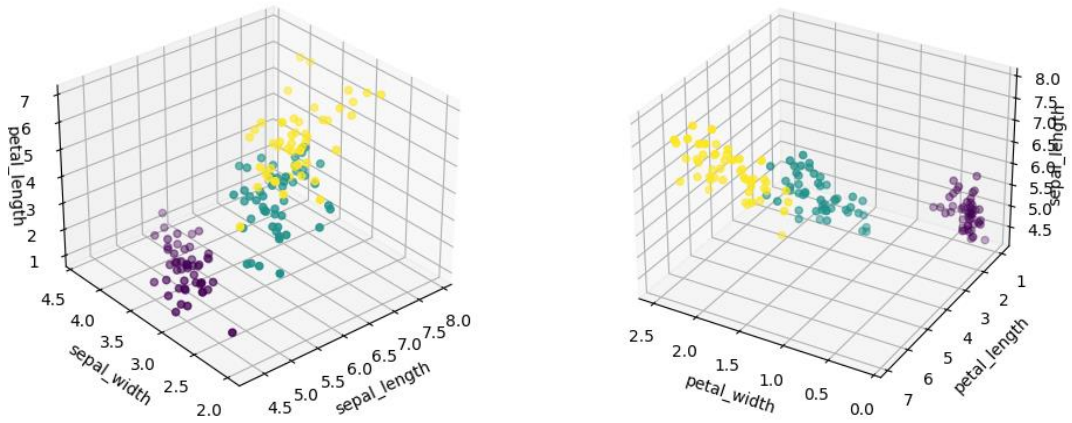
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
7	5.0	3.4	1.5	0.2	0
8	4.4	2.9	1.4	0.2	0
9	4.9	3.1	1.5	0.1	0

데이터 획득 및 전처리

- 데이터셋 살펴보기
  - 각 피쳐들의 분포 살펴보기



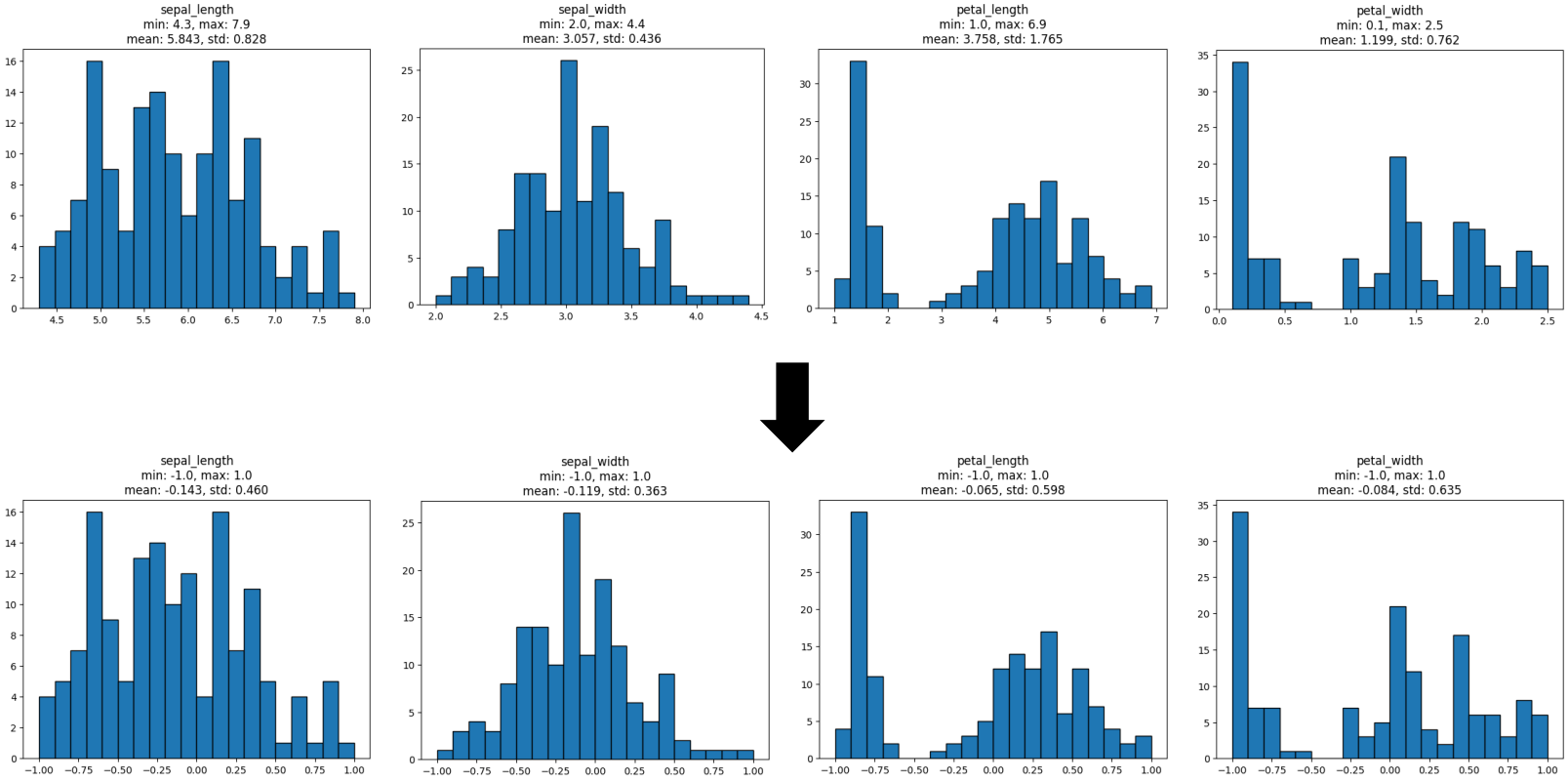
  - 데이터를 산점도로 보기



데이터 획득 및 전처리

- 데이터셋 전처리
  - ➢ 데이터를 -1, 1 사이로 정규화

```
df_max = df.max(axis = 0)
df_min = df.min(axis = 0)

df = (df - (df_max + df_min) / 2) # 데이터 중심 보정
df = df / df.max(axis = 0) # 데이터 정규화
```



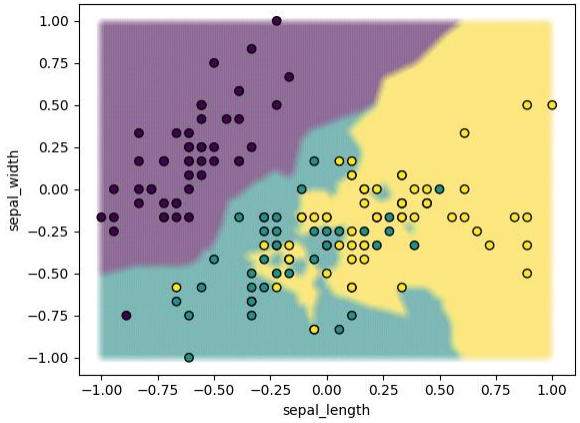
✓ 데이터의 통계적 특징을 유지한 채로 -1, 1 사이로 정규화

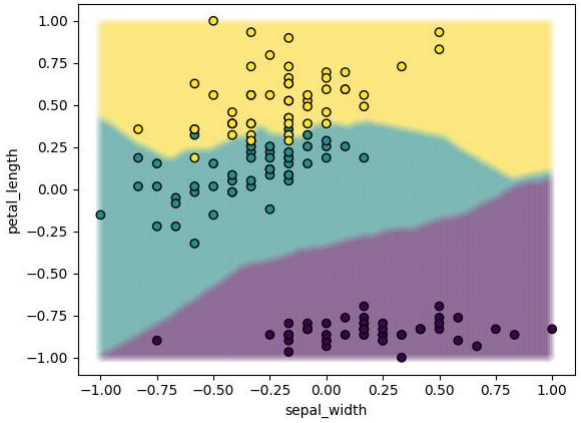
분류 모델에 따른 분류 성능 비교

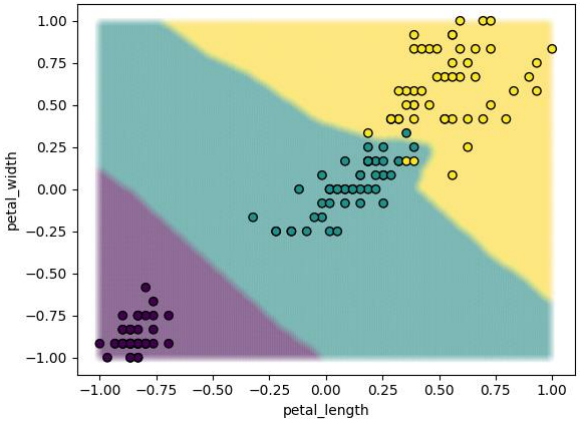
- **K-Nearest Neighbor**
  - ➢ 정확도: 95.5%

```
from sklearn.neighbors import KNeighborsClassifier

knn_classifier = KNeighborsClassifier(n_neighbors = 5)
knn_classifier.fit(df_train[feature_names], df_train['class'])

prediction = knn_classifier.predict(df_test[feature_names])

score = (prediction == df_test['class'].values).sum() / len(df_test)
score * 100
```

➢ **Decision Boundary**





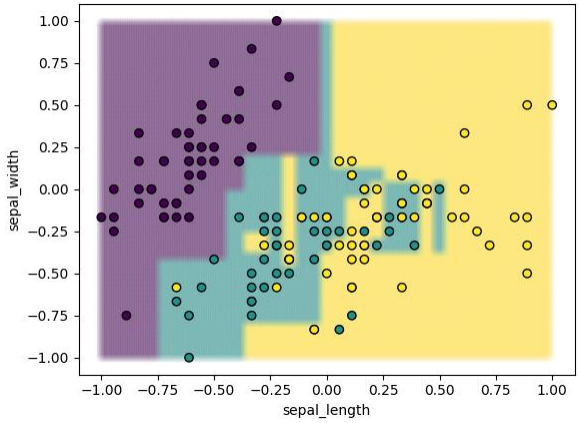


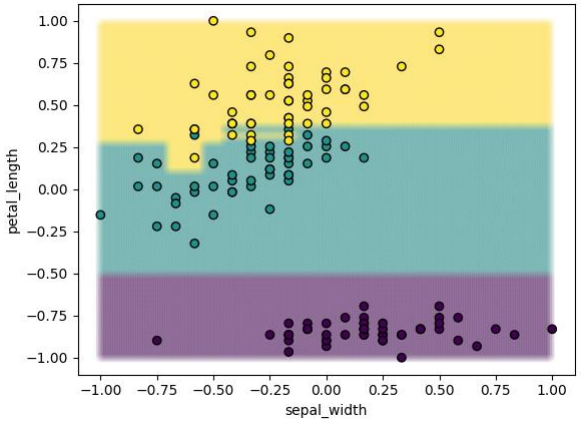
분류 모델에 따른 분류 성능 비교

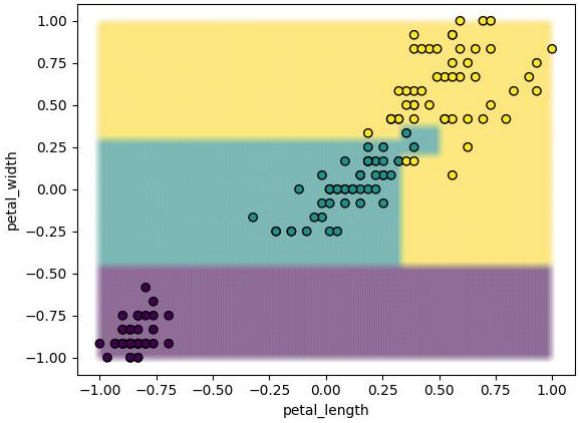
- **Decision Tree**
  - ➢ 정확도: 91.1%

```
from sklearn.tree import DecisionTreeClassifier

decision_tree_classifier = DecisionTreeClassifier()
decision_tree_classifier.fit(df_train[feature_names_full], df_train['class'])

prediction = decision_tree_classifier.predict(df_test[feature_names_full])

score = (prediction == df_test['class'].values).sum() / len(df_test)
score * 100
```

  - ➢ Decision Boundary





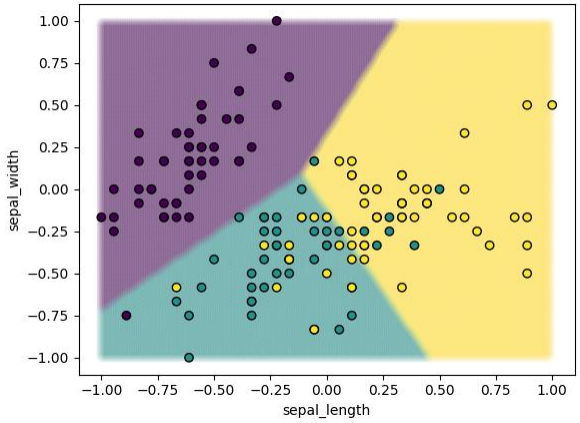


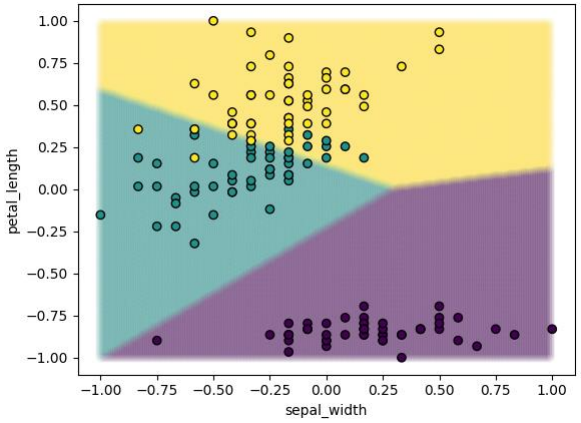
분류 모델에 따른 분류 성능 비교

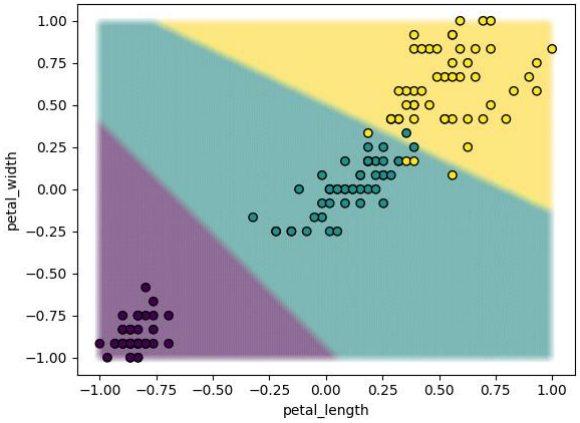
- **Logistic Regression**
  - ➢ 정확도: 95.5%

```
from sklearn.linear_model import LogisticRegression

logistic_regression = LogisticRegression()
logistic_regression.fit(df_train[feature_names_full], df_train['class'])

score = (prediction == df_test['class'].values).sum() / len(df_test)
score * 100
```

➢ **Decision Boundary**





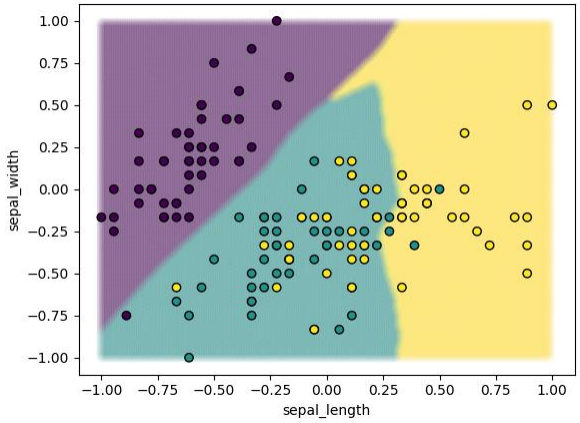


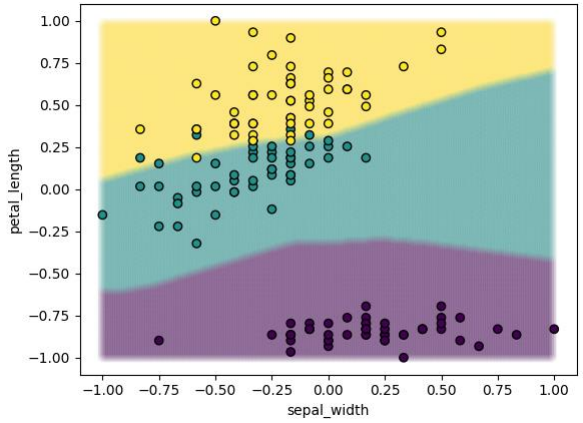
분류 모델에 따른 분류 성능 비교

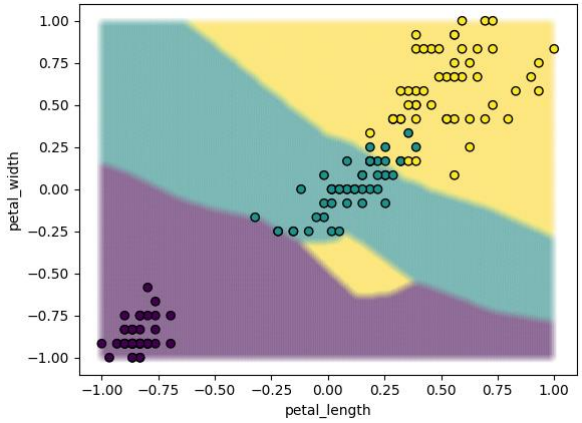
- **Artificial Neural Network**
  - ➢ 정확도: 97.7%

```python
class NN(nn.Module):
    def __init__(self, in_dim, out_dim):
        super(NN, self).__init__()

        self.fc1 = nn.Linear(4, 16)
        self.batch_norm1 = nn.BatchNorm1d(16)
        self.relu1 = nn.ReLU()

        self.fc2 = nn.Linear(16, 16)
        self.batch_norm2 = nn.BatchNorm1d(16)
        self.relu2 = nn.ReLU()

        self.fc3 = nn.Linear(16, 3)
        self.softmax = nn.Softmax()
```

  - ➢ Decision Boundary







$+\alpha$

- $+\alpha$
  - Ensemble
    - ✓ Hard Voting: 여러 모델들의 예측값 중 최빈값을 최종 예측값으로 정한다.
    - ✓ Soft Voting: 각 모델들의 예측 확률값을 취합하고 평균을 구해서 최종 예측값을 정한다.
  - 전의 머신러닝 모델들로 결과를 취합하여 Hard Voting한 결과
    - ✓ 정확도 95.5%
- 결론
  - AI는 꽃을 구분할 수 있다.

퀴즈

- 퀴즈
  - 뉴럴 네트워크에서 가중치 행렬과의 행렬곱은 무엇을 의미하는가
  - 뉴럴 네트워크에서 비선형 활성화함수를 사용하는 이유는 무엇인가
  - 데이터 분류에 사용하는 머신러닝 알고리즘을 3개 말하시오
  - 레이어의 너비와 Decision Boundary의 관계는 어떻게 되는가
  - 선형 변환의 성질을 3가지 말하시오