동국대학교
dongguk university

Logical Reasoning & Knowledge Based System

Practice3

실습 1. Python Library

- Numpy
  - Rank, Zeros, Ones, Identity, Random
  - Array Indexing
  - Data type
  - Array Math
- Pandas
  - Object creation
  - Viewing Data
  - Import csv dataset
  - Selection
  - Setting
  - Operation
  - merge
- Matplotlib
  - pyplot

Pandas Exercise

1. Load Packages

In [1]:

2. Import DataSet

In [2]:

Out[2]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/10 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/10 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/10 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/10 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/10 8:26	3.39	17850.0	United Kingdom

3. Create a bar graph with the 10 countries that have the most 'Quantity' ordered except UK

In [3]:

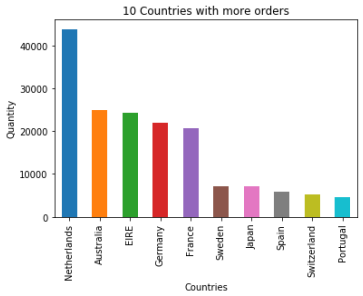
```
# group by the Country

# sort the value and get the first 10 after UK

# create the plot

# Set the title and labels

# show the plot
```



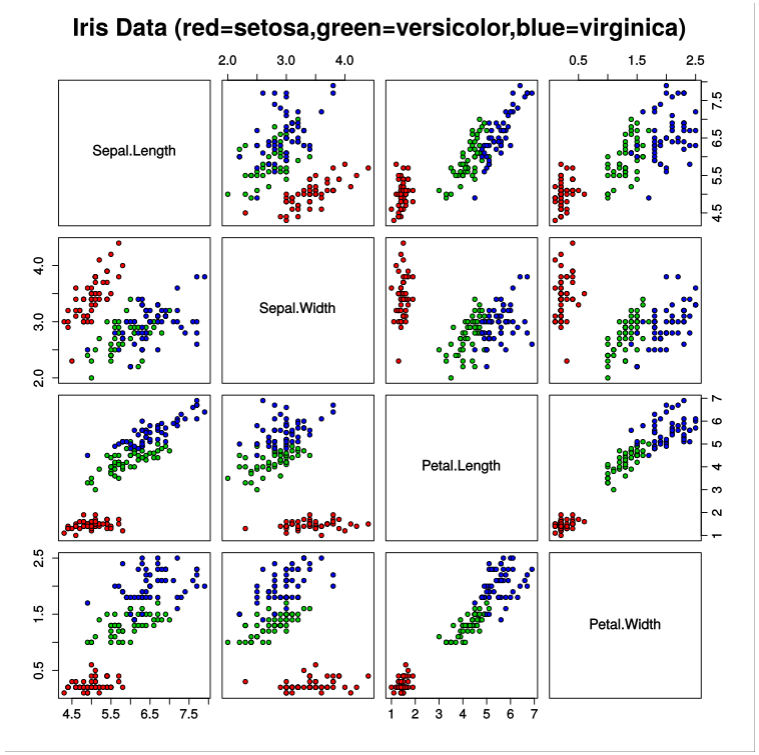
실습 2. KNN (scikit-learn)

- Iris data set classification

[setosa]

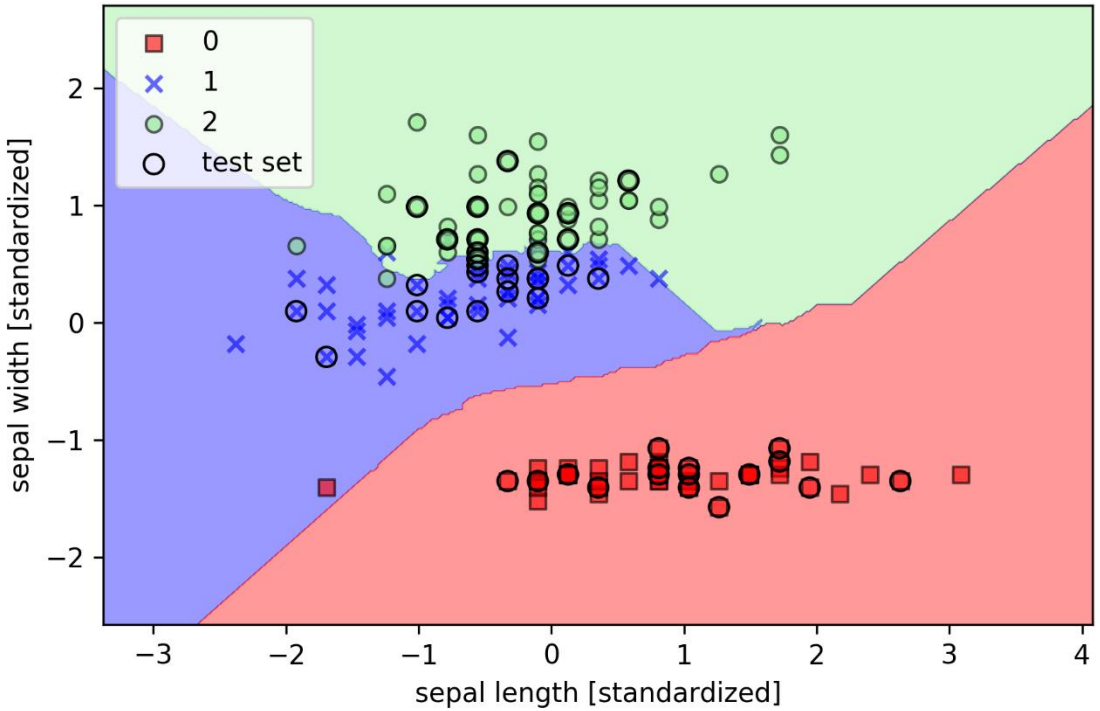
[versicolor]

[virginica]



실습 2. KNN (scikit-learn)

- Plot Decision Regions



실습 3. Weighted KNN Implementation

- k-Nearest Neighbor method
  1. Define distance functions
  - Example
    - $d_{sex}(A, B) = |A - B|$ (female:0, male:1)
    - $d_{age}(A, B) = |A - B|$ / max difference
    - $d = d_{sex} + d_{age}$

    - $d_{sum}(c5, c1) = |0 - 1| + |20 - 40| / 20 = 2.0$
    - $d_{sum}(c5, c2) = |0 - 1| + |20 - 20| / 20 = 1.0$
    - $d_{sum}(c5, c3) = |0 - 0| + |20 - 30| / 20 = 0.5$
    - $d_{sum}(c5, c4) = |0 - 1| + |20 - 30| / 20 = 1.5$

실습 3. Weighted KNN Implementation

2. Predict value from neighbors
   - Weighted average of neighbor values f_i (Y = +1, N = -1)

$$Let \quad w_i = \frac{1}{d_{ij}}, \quad W = \sum w_i$$

$$f_j = \sum f_i \cdot \frac{w_i}{W}$$

- Example
  - 3-NN → c3, c2, c4

    c3: f_3 = -1(N), d_{35} = 0.5, w_3 = 2.0

    c2: f_2 = +1(Y), d_{25} = 1.0, w_2 = 1.0

    c4: f_4 = +1(Y), d_{45} = 1.5, w_4 = 0.67
  - f_5 = [(-1*2.0) + (1*1.0) + (1*0.67)] / 3.67 = -0.09 → N

실습 3. Weighted KNN Implementation

```python
def getPredictionsWeightedKNN(trainingSet, testSet, k):
    predictions = []
    # for each instances(vectors) in testSet
    for i in range(len(testSet)):
        # get K neighbors from trainingSet
        neighbors = getNeighbors(trainingSet, testSet[i], k, distance=distance)

        # Weighted KNN
        # get W & # get f_j
        weights = []
        fi_wi = []
        for neighbor in neighbors:
            # set d_ij(distance)
            d_ij = neighbor[1]
            label = neighbor[2]
            f_i = function_i(label)

            # (1) Let w_i = (1 / d_ij)

            # (2) f_i * w_i

        # (3) W = Sum([w_i])

        # (4) f_j = Sum(fi_wi) / W

        # determine label by f_j
        if f_j <= 0:
            prediction = 0
        elif f_j > 0:
            prediction = 1

        # put result in the predictions
        predictions.append(prediction)

    return predictions
```