Machine Learning Report

02

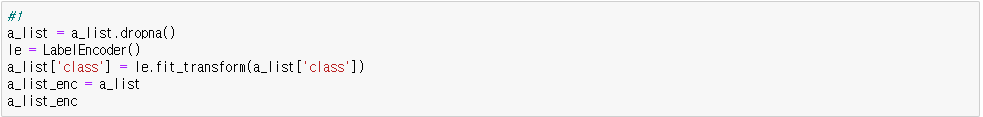
학번: 2017112200

이름: 신현호

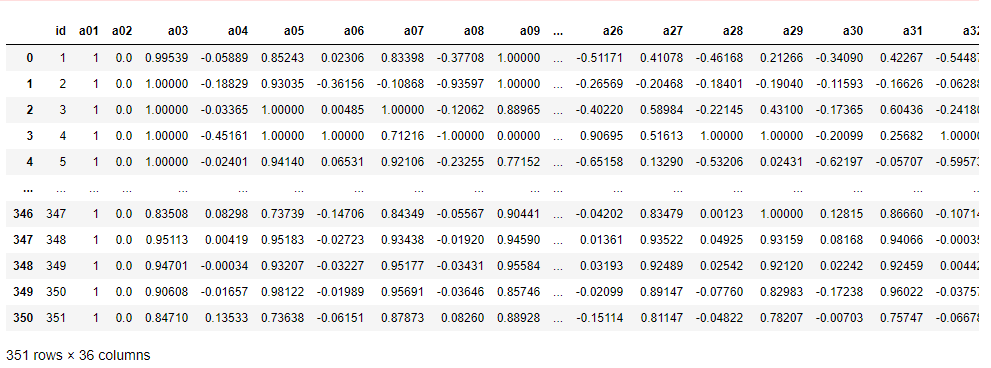
1. Label Encoding scikit library (like most of ML packages) handles numbers only. That means: if some attributes contain symbols/strings (categorical attributes), you must change them to integer numbers. For example, suppose an attribute ‘temperature’ contains 3 values: ‘high’, ‘medium’, ‘low’, you have to change them to 0, 1, 2, respectively. (the integer value ought to begin with 0, and, for now, the order of the value doesn’t matter) Label encoding MUST be done for EVERY categorical attributes (including target attributes).

1) scikit learn provides a built-in function ‘LabelEncoder’ for this purpose. ‘LabelEncoder’ changes symbols to integers from 0 to (no\_of\_values – 1). Refer to the following example and implement Label Encoding using ‘LabelEncoder’

<Code>



<Result>

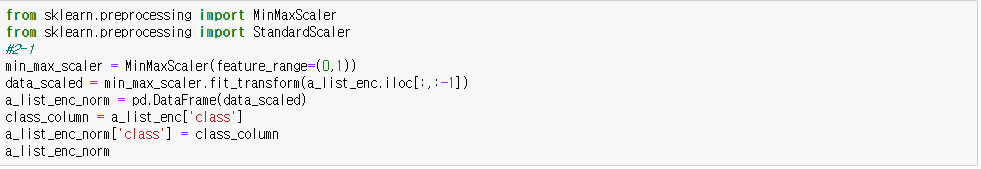


2. Normalize

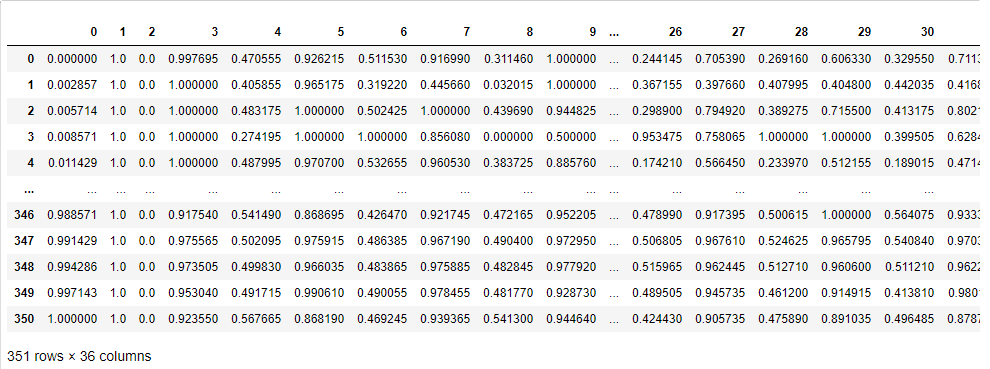
Using 'a\_list\_enc' above, we are going to normalize the values of each attributes.

1. For every value x of a certain attribute, change it to |x - min|/|max-min| where ‘min’/‘max’ means the minimum/maximum value within the attribute, respectively. scikit learn has a function ‘MinMaxScaler’ which does the same functionality.

<Code>

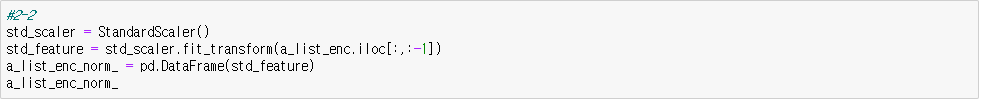


<Result>

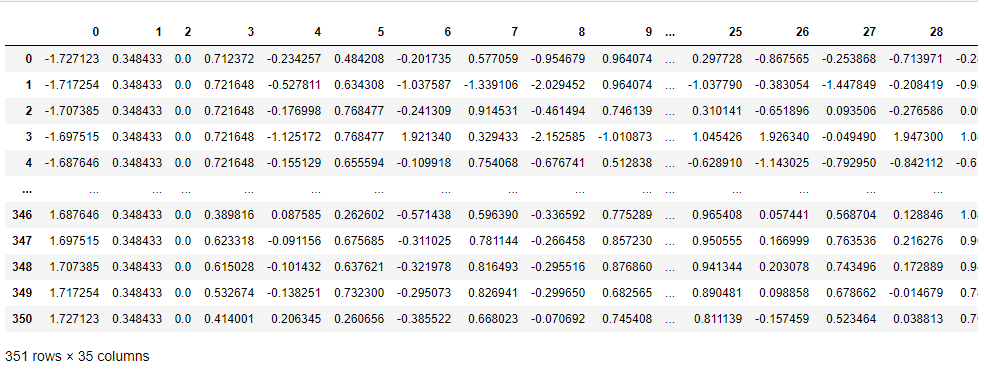


1. Instead of ‘MinMaxScaler’, now you use ‘StandardScaler’

<Code>



<Result>

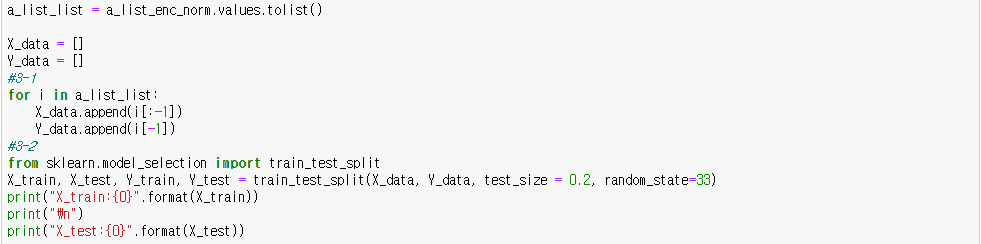


3. Divide\_train\_test

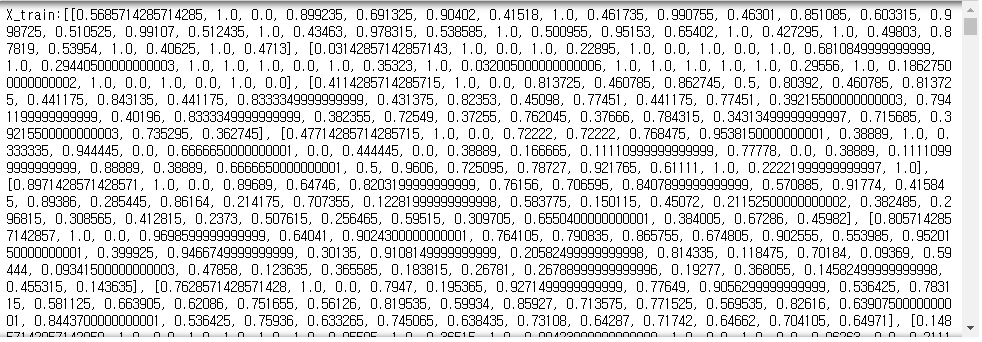
From a file ‘a\_list’, ‘X\_data’ is the attribute values except target attribute and ‘Y\_data’ is the list of target values. For example, suppose a\_list=[[1,0,2,3,1], [0,1,1,2,0], [0,1,0,1,1], [0,0,2,3,1]], X\_data = [[1,0,2,3], [0,1,1,2], [0,1,0,1], [0,0,2,3]] and Y\_data = [1,0,1,1]

1. Write a program which splits ‘a\_list\_enc\_norm’ file into ‘X\_data’ and ‘Y\_data’.
2. Split ‘X\_data’ and ‘Y\_data’ into X\_train, X\_test, Y\_train, Y\_test as follows: X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X\_data, Y\_data, test\_size=0.2, random\_state=33)

<Code>



<Result>



4. Running Neural Network

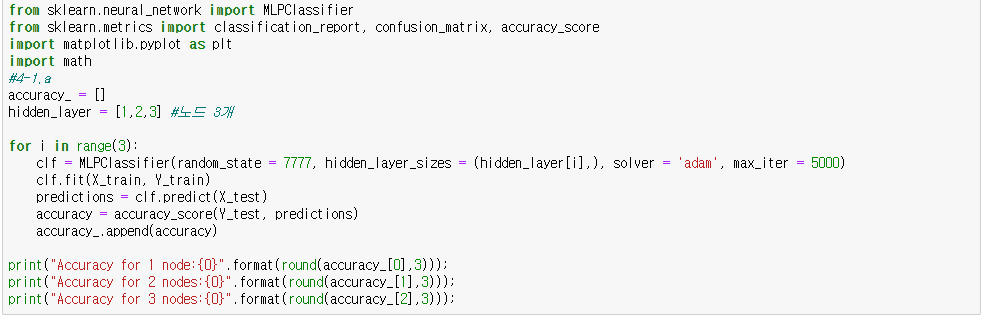
1) different hidden layer/node hidden\_layer\_sizes=(20,) means one hidden layer with 20 hidden nodes hidden\_layer\_sizes=(15,15,15) means 3 hidden layers with 15 hidden nodes each. run neural network with

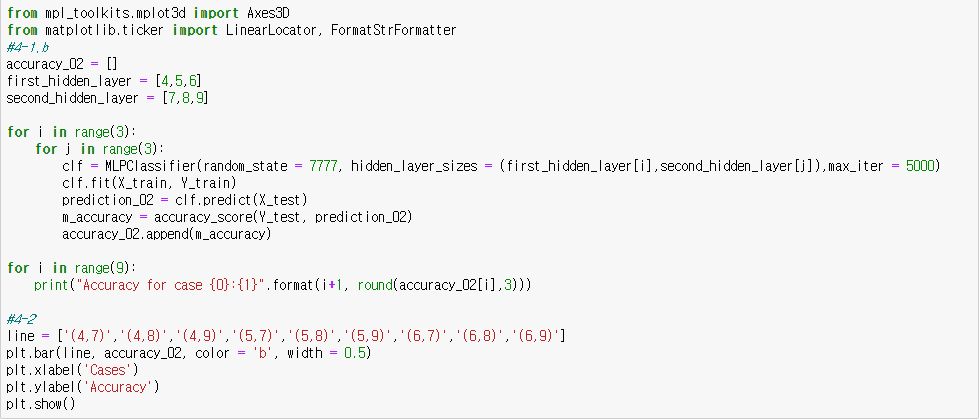
a. one hidden layer with 3 different hidden nodes (on your own choice)

b. two hidden layers with 3 different hidden node configurations. (on your own choice

2) based on the result of 1), plot the relationship between accuracy and hidden layer/nodes.

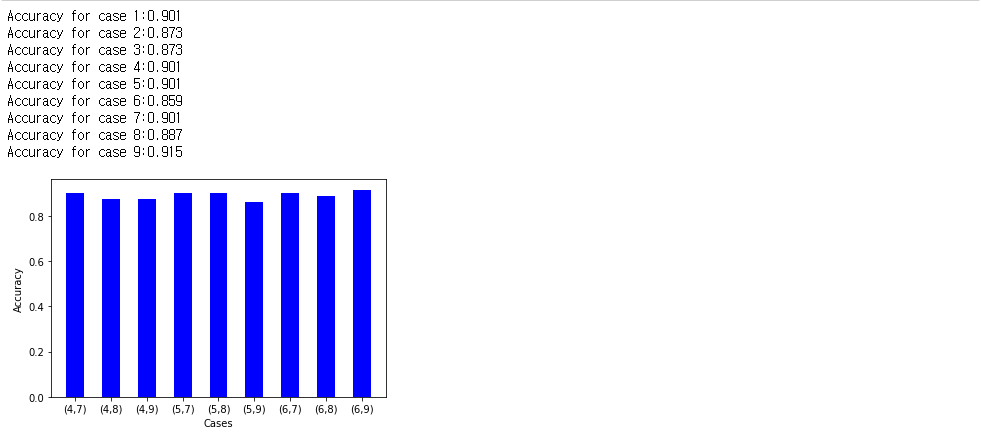
<Code>





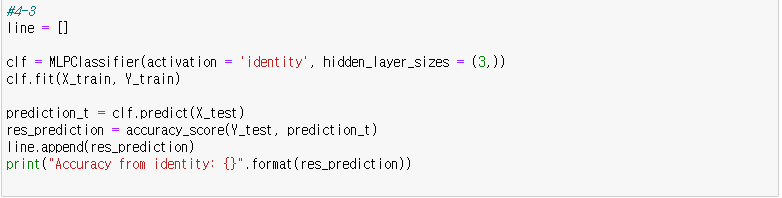
<Result>

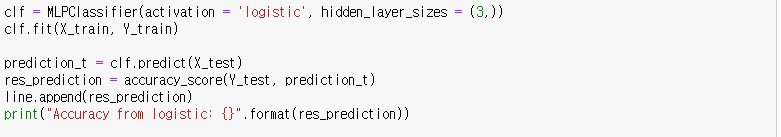


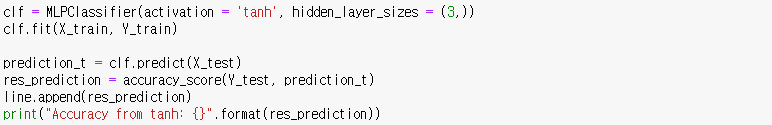


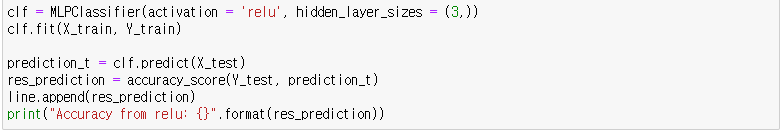
1. run neural network by changing 'activation function' {‘identity’, ‘logistic’, ‘tanh’, ‘relu’}, Compare the results of these activation functions by plotting

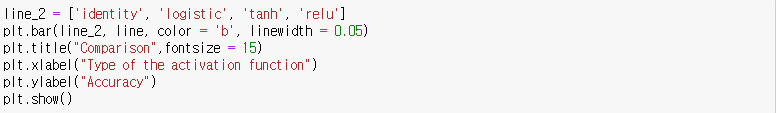
<Code>











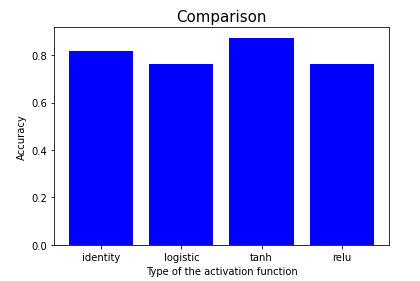
<Result>





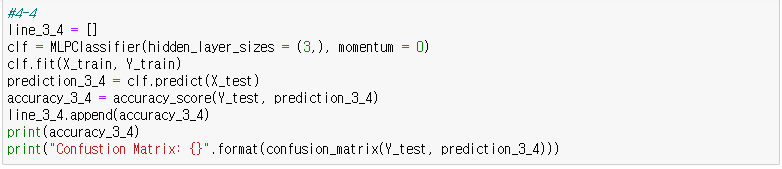


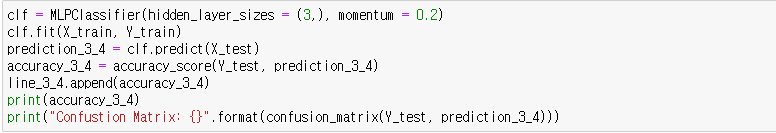


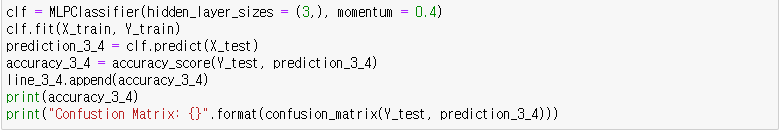


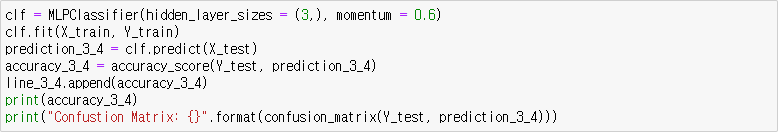
1. run neural network with different 'momentum' values of (0, 0.2, 0.4, 0.6, 0.8). Compare the results of these parameter values by plotting.

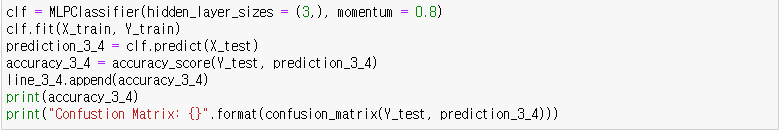
<Code>

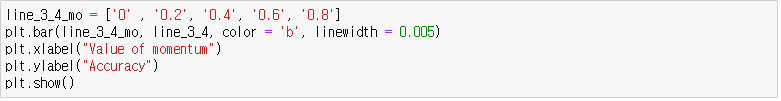








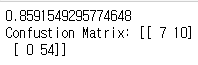




<Result>

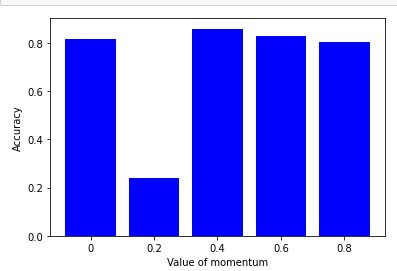






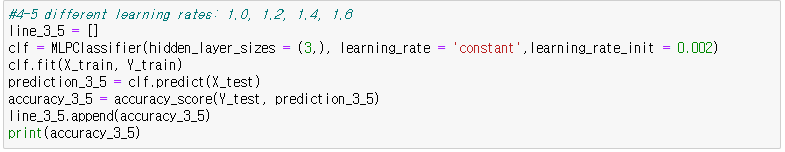


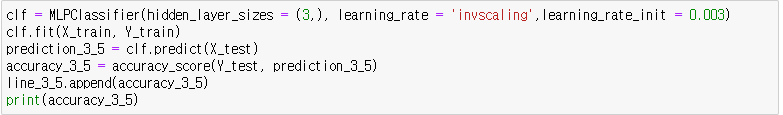


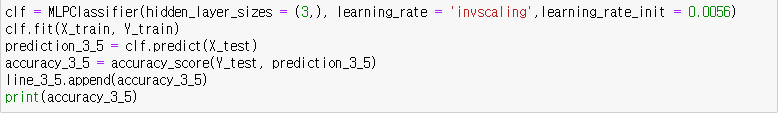


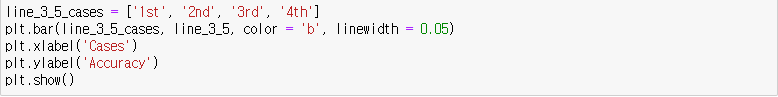
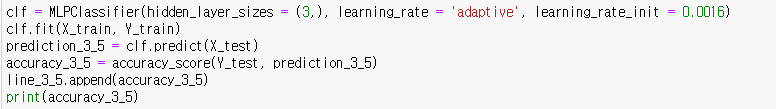
1. run neural network with 4 different learning rate (on your choice). You have to change 'learning\_rate' parameter and (sometimes) 'learning\_rate\_init' parameter. Compare the results of these parameter values by plotting.

<Code>









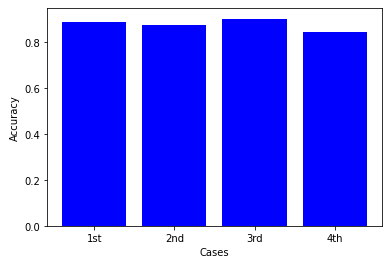
<Result>











5. For every numeric(continuous) attribute, we are going to discretize it, meaning numeric attribute is converted in a number of intervals (bins). In this exercise, we use equal distance discretization. You can also use scikit library to discretize values.

Referring to the following example code,

from sklearn.preprocessing import KBinsDiscretizer

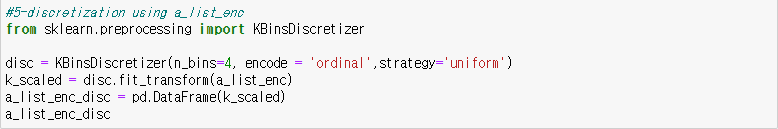
disc = KBinsDiscretizer(n\_bins=3, encode='uniform', strategy='uniform')

disc.fit\_transform(X)

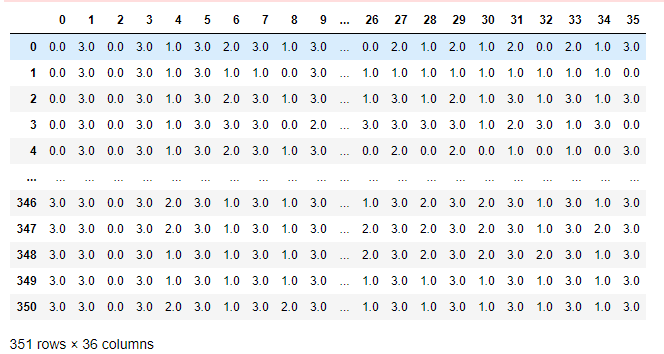
Using 'a\_list\_enc' from Q. 1 above, do the following.

1. For every numeric attribute in ‘a\_list\_enc’, discretize the attribute with N=4 intervals. (you can change N value if you want)

<Code>



<Result>



6. Running Decision Tree

1) Using Q. 3 method, split 'a\_list\_enc\_disc' into X\_train, X\_test, Y\_train, Y\_test From Q. 4 (neural network), replace the following line

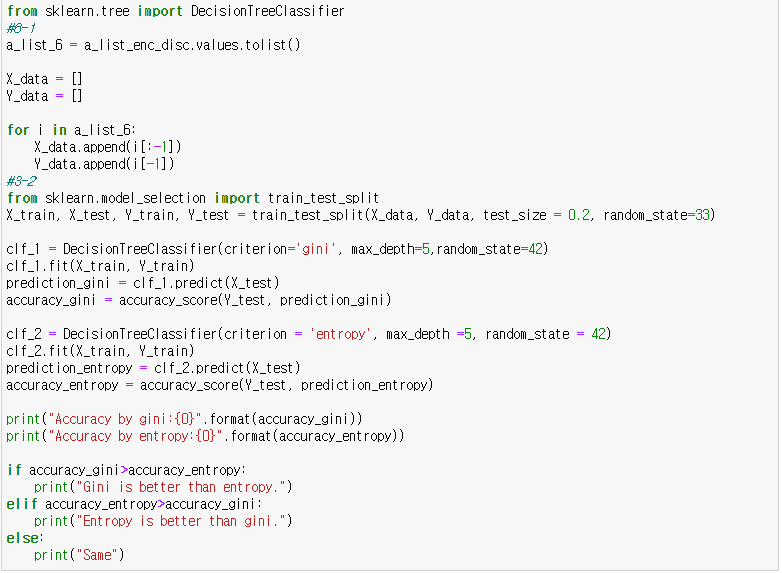
clf = MLPClassifier(hidden\_layer\_sizes=(20,))

by

clf = DecisionTreeClassifier()

Run this decision tree TWICE by changing criterion=’gini’ or ‘entropy’. Compare the results. For your dataset, which model is better ? gini or entropy ? (Maybe your ‘max\_depth’ value should be 5 or higher.)

<Code>

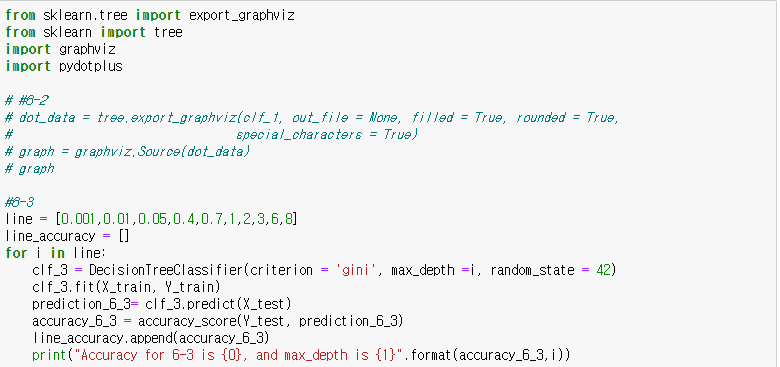


<Result>

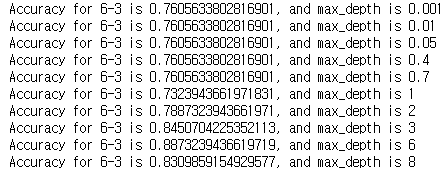


1. show the diagram of one decision tree using graphviz (refer to lab class)
2. run decision tree TEN TIMES by changing 'max\_depth' values. What if the max\_depth value is very large or very small? You need to compare the two values(very small and very large values).

<Code>

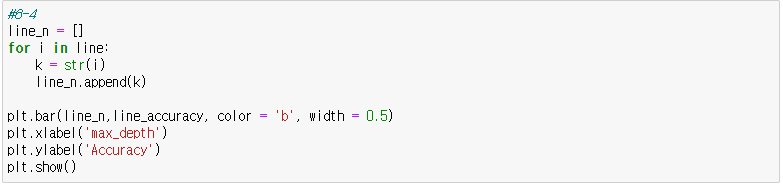


<Result>

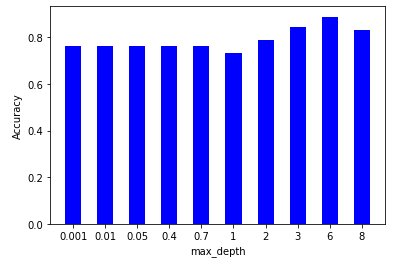


1. Plot the graph of Q 3) and find the optimal ‘max\_depth’.

<Code>



<Result>



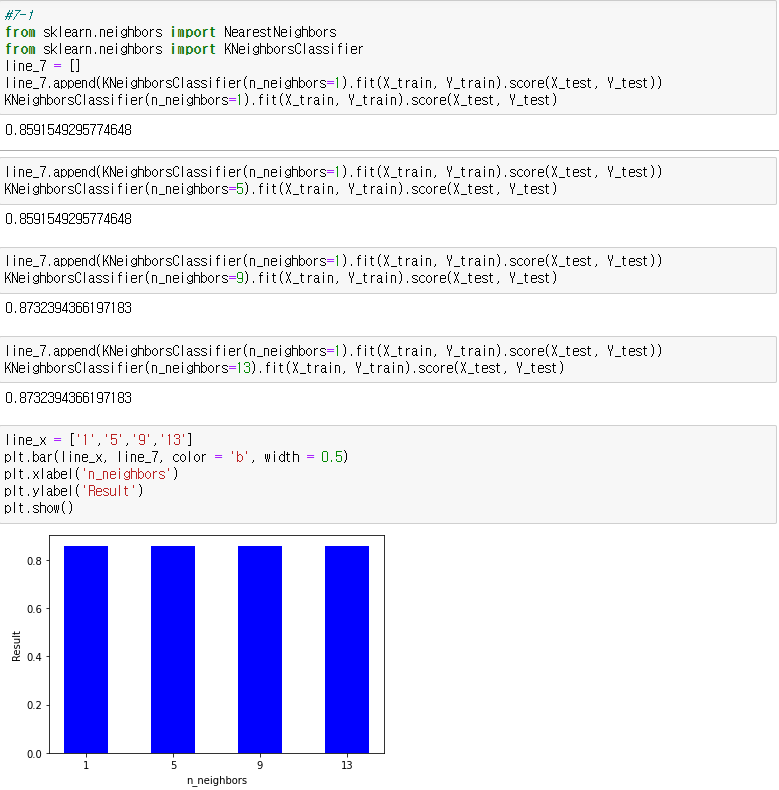
1. Explain why the ‘max\_depth’ is the optimal value in relation to the Q 4) graph.

* 정확도의 값이 서서히 올라가다가 6에서 가장 좋은 성능을 보이고, 8에서는 꺾이는 것을 볼 수 있다. max\_depth의 값이 너무 커지면 과적합 현상이 발생하기 때문에 고려하지 않아야 할 것들까지 고려함으로써 정확도를 계산하므로 성능을 저해시키기 때문에 8이 6보다 성능이 좋지 않다.

7. Running IBL (K nearest neighbor)

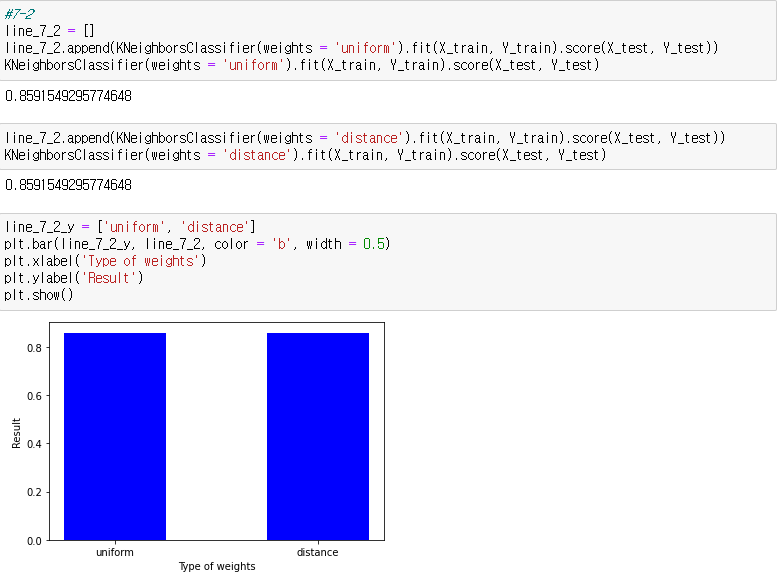
1) run the algorithm by changing the value of n\_neighbors (default=5) to 1, 5, 9, and 13, respectively. Draw a graph showing the relationship between accuracy and n\_neighbors.

<Code & Result>(이 부분은 한 번에 확인하시는 것이 편할 것 같아서 코드와 결과를 한번에 첨부했습니다.)



2) run the algorithm by changing the value of weights to ‘uniform’ and ‘distance’, respectively. Compare the results by plotting bar graph and explain the meaning of the results.

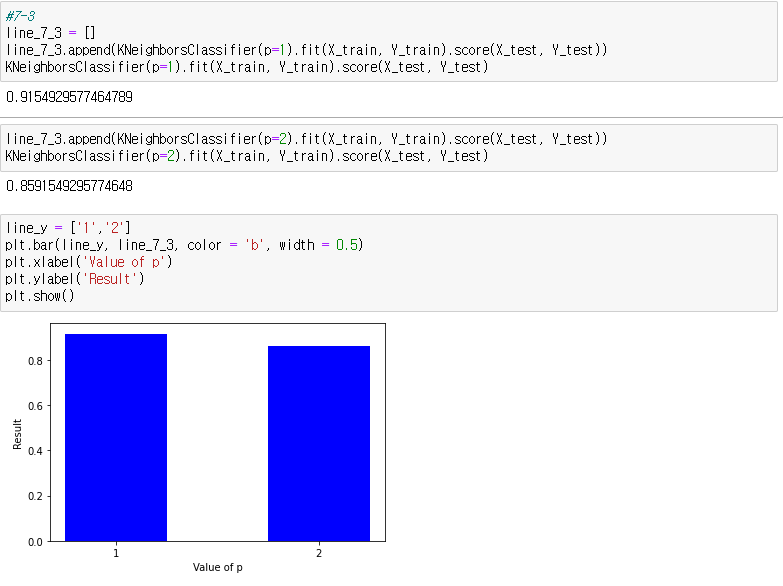
<Code & Result>



* 매개변수 weights가 uniform일 때는 단순 데이터들의 평균을 계산하는 반면, distance일 때는 데이터 분포도 상에서의 데이터 사이의 거리를 고려하여 가중치 평균을 계산한다. 코드의 결과는 똑같은 정확도를 보였다.

1. run the algorithm by changing the value of p to 1 (Manhattan) and 2 (Euclidean). Compare the results by plotting bar graph and explain the meaning of the results.

<Code & Result>



* p 값은 minkowski의 매개변수이며, 1일 때는 맨허튼 거리 공식, 2일 때는 유클라디안 거리 공식을 사용한다. 결과를 보아, 맨해튼 거리 공식을 썼을 때가 정확도가 더 높았다.