MLPC_R_SVG

September 1, 2020

[17]:
```
#Importing libraries
from sklearn.neural_network import MLPClassifier
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_moons
from sklearn.datasets import load_breast_cancer
import mglearn
import matplotlib.pyplot as plt
import pandas as pd
```

[2]:
```
#importing cryography dataset
cryo_df = pd.read_csv("C:/Users/delld/Downloads/Cryotherapy.csv")
```

[3]:
```
#Details of Cryography dataset
cryo_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90 entries, 0 to 89
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   sex                  90 non-null     int64
 1   age                  90 non-null     int64
 2   Time                 90 non-null     float64
 3   Number_of_Warts      90 non-null     int64
 4   Type                 90 non-null     int64
 5   Area                 90 non-null     int64
 6   Result_of_Treatment  90 non-null     int64
dtypes: float64(1), int64(6)
memory usage: 5.0 KB
```

[4]:
```
X_features = cryo_df.columns
X_features
```

[4]:
```
Index(['sex', 'age', 'Time', 'Number_of_Warts', 'Type', 'Area',
       'Result_of_Treatment'],
      dtype='object')
```

[7]:
```
X = cryo_df[X_features]
Y = cryo_df['Result_of_Treatment']
X=X.drop(['Result_of_Treatment'],axis=1)
```

[8]:
```
#spliting into training and testing samples
X_train,X_test,y_train,y_test = train_test_split(X,Y,random_state=42)
```

[9]:
```
#MLP classifier with alpha = 0.0001, 100 hidden nodes, relu activation function
mlp = MLPClassifier(max_iter=5000,random_state=42)
mlp.fit(X_train,y_train)
mlp.score(X_test,y_test)
```

[9]: 0.8260869565217391

[11]:
```
#MLP classifier with alpha = 0.01, single layer 10 hidden nodes, relu␣
↪activation function
mlp = MLPClassifier(max_iter=5000,alpha=0.
↪01,random_state=42,hidden_layer_sizes=10)
mlp.fit(X_train,y_train)
mlp.score(X_test,y_test)
```

[11]: 0.782608695652174

[12]:
```
#MLP classifier with alpha = 0.01, 2 layers of each 10 hidden nodes, relu␣
↪activation function
mlp = MLPClassifier(max_iter=5000,alpha=0.
↪01,random_state=42,hidden_layer_sizes=[10,10])
mlp.fit(X_train,y_train)
mlp.score(X_test,y_test)
```

[12]: 0.7391304347826086

[13]:
```
#MLP classifier with alpha = 0.01, 3 layers of each 5 hidden nodes, relu␣
↪activation function
mlp = MLPClassifier(max_iter=5000,alpha=0.
↪01,random_state=42,hidden_layer_sizes=[5,5,5])
mlp.fit(X_train,y_train)
mlp.score(X_test,y_test)
```

[13]: 0.782608695652174

[16]:
```
#MLP classifier with alpha = 0.01, 3 layers of each 5 hidden nodes, tanh␣
↪activation function
mlp = MLPClassifier(max_iter=5000,alpha=0.
↪0001,activation='tanh',random_state=42)
mlp.fit(X_train,y_train)
```

```
mlp.score(X_test,y_test)
```

[16]:
```
0.8260869565217391
```

[18]:
```
mlp_r = MLPRegressor()
```

[19]:
```
mlp_r
```

[19]:
```
MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
             beta_2=0.999, early_stopping=False, epsilon=1e-08,
             hidden_layer_sizes=(100,), learning_rate='constant',
             learning_rate_init=0.001, max_fun=15000, max_iter=200,
             momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
             power_t=0.5, random_state=None, shuffle=True, solver='adam',
             tol=0.0001, validation_fraction=0.1, verbose=False,
             warm_start=False)
```

[20]:
```
#importing Concrete dataset
concrete_df = pd.read_csv("C:/Users/delld/Downloads/Concrete_Data.csv")
```

[21]:
```
concrete_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Cement               1030 non-null   float64
 1   Blast_Furnace_Slag   1030 non-null   float64
 2   Fly Ash              1030 non-null   float64
 3   Water                1030 non-null   float64
 4   Superplasticizer     1030 non-null   float64
 5   Coarse_Aggregate     1030 non-null   float64
 6   Fine_Aggregate       1030 non-null   float64
 7   Age                  1030 non-null   int64  
 8   Concrete_strength    1030 non-null   float64
dtypes: float64(8), int64(1)
memory usage: 72.5 KB
```

[22]:
```
X_features = concrete_df.columns
X_features
```

[22]:
```
Index(['Cement ', 'Blast_Furnace_Slag ', 'Fly Ash ', 'Water',
       'Superplasticizer ', 'Coarse_Aggregate ', 'Fine_Aggregate', 'Age ',
       'Concrete_strength'],
      dtype='object')
```

[23]:
```
X = concrete_df[X_features]
Y = concrete_df['Concrete_strength']
X=X.drop(['Concrete_strength'],axis=1)
```

[24]:
```
#spliting into training and testing samples
X_train,X_test,y_train,y_test = train_test_split(X,Y,random_state=42)
```

[25]:
```
#MLP Regressor with alpha = 0.0001, 100 hidden nodes, relu activation function
mlpr = MLPRegressor(max_iter=5000,random_state=42)
mlpr.fit(X_train,y_train)
mlpr.score(X_test,y_test)
```

[25]: 0.8229000437208142

[27]:
```
#MLP Regressor with alpha = 0.01, single layer 10 hidden nodes, relu activation␣
↪function
mlpr = MLPRegressor(max_iter=5000,alpha=0.
↪01,random_state=42,hidden_layer_sizes=10)
mlpr.fit(X_train,y_train)
mlpr.score(X_test,y_test)
```

[27]: 0.6169970051806134

[28]:
```
#MLP Regressor with alpha = 0.01, 2 layers of each 10 hidden nodes, relu␣
↪activation function
mlpr = MLPRegressor(max_iter=5000,alpha=0.
↪01,random_state=42,hidden_layer_sizes=[10,10])
mlpr.fit(X_train,y_train)
mlpr.score(X_test,y_test)
```

[28]: 0.8236199633496706

[29]:
```
#MLP Regressor with alpha = 0.01, 3 layers of each 5 hidden nodes, relu␣
↪activation function
mlpr = MLPRegressor(max_iter=5000,alpha=0.
↪01,random_state=42,hidden_layer_sizes=[5,5,5])
mlpr.fit(X_train,y_train)
mlpr.score(X_test,y_test)
```

[29]: 0.6178031052501342

[30]:
```
#MLP Regressor with alpha = 0.01, 3 layers of each 5 hidden nodes, tanh␣
↪activation function
mlpr = MLPRegressor(max_iter=5000,alpha=0.
↪0001,activation='tanh',random_state=42)
mlpr.fit(X_train,y_train)
```

```
mlpr.score(X_test,y_test)
```

[30]: 0.7534476692125461

[ ]: