assignment1-1

April 13, 2025

[12]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

0.0.1 Load Dataset

[13]:
```
url = "https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.
↪csv"
df = pd.read_csv(url)
df.head(10)
```

[13]:
```
      crim    zn  indus  chas    nox     rm    age     dis  rad  tax  ptratio  \
0  0.00632  18.0   2.31     0  0.538  6.575   65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07     0  0.469  6.421   78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07     0  0.469  7.185   61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18     0  0.458  6.998   45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18     0  0.458  7.147   54.2  6.0622    3  222     18.7
5  0.02985   0.0   2.18     0  0.458  6.430   58.7  6.0622    3  222     18.7
6  0.08829  12.5   7.87     0  0.524  6.012   66.6  5.5605    5  311     15.2
7  0.14455  12.5   7.87     0  0.524  6.172   96.1  5.9505    5  311     15.2
8  0.21124  12.5   7.87     0  0.524  5.631  100.0  6.0821    5  311     15.2
9  0.17004  12.5   7.87     0  0.524  6.004   85.9  6.5921    5  311     15.2

        b  lstat  medv
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90   5.33  36.2
5  394.12   5.21  28.7
6  395.60  12.43  22.9
7  396.90  19.15  27.1
8  386.63  29.93  16.5
9  386.71  17.10  18.9
```

Checking for null values

[14]:
```
df.isnull().sum()
```

[14]:
```
crim       0
zn         0
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
ptratio    0
b          0
lstat      0
medv       0
dtype: int64
```

[15]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64
 9   tax      506 non-null    int64
 10  ptratio  506 non-null    float64
 11  b        506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

[16]:
```
df.describe()
```

[16]:
```
             crim          zn       indus        chas         nox          rm  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     3.613524   11.363636   11.136779    0.069170    0.554695    6.284634
std      8.601545   23.322453    6.860353    0.253994    0.115878    0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
```

```
25%     0.082045    0.000000   5.190000   0.000000   0.449000   5.885500
50%     0.256510    0.000000   9.690000   0.000000   0.538000   6.208500
75%     3.677083   12.500000  18.100000   0.000000   0.624000   6.623500
max    88.976200  100.000000  27.740000   1.000000   0.871000   8.780000

              age         dis         rad         tax     ptratio           b  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.574901    3.795043    9.549407  408.237154   18.455534  356.674032
std     28.148861    2.105710    8.707259  168.537116    2.164946   91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%     45.025000    2.100175    4.000000  279.000000   17.400000  375.377500
50%     77.500000    3.207450    5.000000  330.000000   19.050000  391.440000
75%     94.075000    5.188425   24.000000  666.000000   20.200000  396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            lstat        medv
count  506.000000  506.000000
mean    12.653063   22.532806
std      7.141062    9.197104
min      1.730000    5.000000
25%      6.950000   17.025000
50%     11.360000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

Checking correlation with target variable MEDV

[18]:
```
df.corr()['medv'].sort_values()
```

[18]:
```
lstat     -0.737663
ptratio   -0.507787
indus     -0.483725
tax       -0.468536
nox       -0.427321
crim      -0.388305
rad       -0.381626
age       -0.376955
chas       0.175260
dis        0.249929
b          0.333461
zn         0.360445
rm         0.695360
medv       1.000000
Name: medv, dtype: float64
```

[20]:
```
X = df.loc[:,['lstat','ptratio','rm']]
Y = df.loc[:,"medv"]
```

```
X.shape,Y.shape
```

[20]: ((506, 3), (506,))

0.0.2 Preparing training and testing data set

[21]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.
 ↪25,random_state=10)
```

0.0.3 Normalizing training and testing dataset

[22]:
```
from sklearn.preprocessing import StandardScaler
```

[23]:
```
scaler = StandardScaler()
```

[24]:
```
scaler.fit(x_train)
```

[24]: StandardScaler()

[25]:
```
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
```

0.0.4 Preparing model

[26]:
```
from keras.models import Sequential
from keras.layers import Dense
```

[27]:
```
model = Sequential()
```

[28]:
```
model.add(Dense(128,input_shape=(3,),activation='relu',name='input'))
model.add(Dense(64,activation='relu',name='layer_1'))
model.add(Dense(1,activation='linear',name='output'))
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential"

```
 Layer (type)                      Output Shape                 Param #
```

```
 input (Dense)                    (None, 128)                 512

 layer_1 (Dense)                  (None, 64)                8,256

 output (Dense)                   (None, 1)                    65
```

Total params: 8,833 (34.50 KB)

Trainable params: 8,833 (34.50 KB)

Non-trainable params: 0 (0.00 B)

[29]:
```
model.fit(x_train,y_train,epochs=100,validation_split=0.05)
```

```
Epoch 1/100
12/12                 2s 23ms/step -
loss: 558.1187 - mae: 21.8508 - val_loss: 700.8205 - val_mae: 23.7638
Epoch 2/100
12/12                 0s 8ms/step - loss:
492.1137 - mae: 20.6388 - val_loss: 656.5317 - val_mae: 22.8078
Epoch 3/100
12/12                 0s 8ms/step - loss:
469.1971 - mae: 19.9330 - val_loss: 594.3977 - val_mae: 21.4058
Epoch 4/100
12/12                 0s 8ms/step - loss:
408.2853 - mae: 18.2776 - val_loss: 512.5362 - val_mae: 19.4592
Epoch 5/100
12/12                 0s 7ms/step - loss:
331.6889 - mae: 16.3889 - val_loss: 413.3873 - val_mae: 17.1842
Epoch 6/100
12/12                 0s 9ms/step - loss:
232.7548 - mae: 13.6839 - val_loss: 304.6852 - val_mae: 14.2029
Epoch 7/100
12/12                 0s 12ms/step -
loss: 142.1357 - mae: 10.6499 - val_loss: 207.0097 - val_mae: 11.0110
Epoch 8/100
12/12                 0s 8ms/step - loss:
72.1980 - mae: 7.3695 - val_loss: 142.7724 - val_mae: 8.6148
Epoch 9/100
12/12                 0s 8ms/step - loss:
41.6749 - mae: 5.1260 - val_loss: 111.3794 - val_mae: 7.1523
Epoch 10/100
12/12                 0s 11ms/step -
```

```
loss: 34.2609 - mae: 4.4484 - val_loss: 102.3676 - val_mae: 6.8814
Epoch 11/100
12/12            0s 8ms/step - loss:
34.1951 - mae: 4.5410 - val_loss: 96.8025 - val_mae: 6.6428
Epoch 12/100
12/12            0s 11ms/step -
loss: 31.5430 - mae: 4.1328 - val_loss: 93.4553 - val_mae: 6.5237
Epoch 13/100
12/12            0s 8ms/step - loss:
25.0297 - mae: 3.6735 - val_loss: 88.1755 - val_mae: 6.3738
Epoch 14/100
12/12            0s 8ms/step - loss:
26.2185 - mae: 3.8751 - val_loss: 85.6054 - val_mae: 6.3215
Epoch 15/100
12/12            0s 8ms/step - loss:
28.2955 - mae: 3.8190 - val_loss: 85.0852 - val_mae: 6.3433
Epoch 16/100
12/12            0s 8ms/step - loss:
22.4831 - mae: 3.5878 - val_loss: 82.7764 - val_mae: 6.2427
Epoch 17/100
12/12            0s 8ms/step - loss:
20.3968 - mae: 3.2710 - val_loss: 82.5676 - val_mae: 6.2068
Epoch 18/100
12/12            0s 8ms/step - loss:
22.6890 - mae: 3.4112 - val_loss: 82.7573 - val_mae: 6.1786
Epoch 19/100
12/12            0s 8ms/step - loss:
20.3038 - mae: 3.3030 - val_loss: 82.6876 - val_mae: 6.1875
Epoch 20/100
12/12            0s 12ms/step -
loss: 18.1940 - mae: 3.1957 - val_loss: 82.3712 - val_mae: 6.1761
Epoch 21/100
12/12            0s 8ms/step - loss:
21.1167 - mae: 3.3282 - val_loss: 84.2112 - val_mae: 6.1819
Epoch 22/100
12/12            0s 8ms/step - loss:
17.5337 - mae: 3.1397 - val_loss: 84.3052 - val_mae: 6.1591
Epoch 23/100
12/12            0s 8ms/step - loss:
18.7228 - mae: 3.1415 - val_loss: 82.2697 - val_mae: 6.0732
Epoch 24/100
12/12            0s 7ms/step - loss:
16.6766 - mae: 3.0005 - val_loss: 80.8897 - val_mae: 5.9858
Epoch 25/100
12/12            0s 7ms/step - loss:
16.6134 - mae: 3.0370 - val_loss: 81.4060 - val_mae: 5.9545
Epoch 26/100
12/12            0s 13ms/step -
```

```
loss: 16.7044 - mae: 3.0033 - val_loss: 80.2639 - val_mae: 5.8787
Epoch 27/100
12/12            0s 7ms/step - loss:
15.2913 - mae: 2.8955 - val_loss: 81.5878 - val_mae: 5.8762
Epoch 28/100
12/12            0s 8ms/step - loss:
16.1182 - mae: 3.0439 - val_loss: 79.0019 - val_mae: 5.8053
Epoch 29/100
12/12            0s 8ms/step - loss:
16.8150 - mae: 3.0625 - val_loss: 79.0107 - val_mae: 5.7505
Epoch 30/100
12/12            0s 8ms/step - loss:
17.0178 - mae: 2.9969 - val_loss: 78.6634 - val_mae: 5.7050
Epoch 31/100
12/12            0s 8ms/step - loss:
15.9267 - mae: 3.0086 - val_loss: 78.9061 - val_mae: 5.6831
Epoch 32/100
12/12            0s 7ms/step - loss:
15.9151 - mae: 2.9278 - val_loss: 80.2465 - val_mae: 5.7070
Epoch 33/100
12/12            0s 7ms/step - loss:
15.2599 - mae: 2.9143 - val_loss: 78.9737 - val_mae: 5.6547
Epoch 34/100
12/12            0s 8ms/step - loss:
18.4289 - mae: 3.0328 - val_loss: 79.6162 - val_mae: 5.6304
Epoch 35/100
12/12            0s 8ms/step - loss:
12.4532 - mae: 2.6375 - val_loss: 80.3411 - val_mae: 5.6226
Epoch 36/100
12/12            0s 8ms/step - loss:
15.1316 - mae: 2.8000 - val_loss: 78.5153 - val_mae: 5.5424
Epoch 37/100
12/12            0s 7ms/step - loss:
14.6459 - mae: 2.8143 - val_loss: 79.2373 - val_mae: 5.5485
Epoch 38/100
12/12            0s 7ms/step - loss:
13.7781 - mae: 2.7703 - val_loss: 81.3237 - val_mae: 5.5947
Epoch 39/100
12/12            0s 14ms/step -
loss: 13.8437 - mae: 2.7731 - val_loss: 79.8047 - val_mae: 5.5260
Epoch 40/100
12/12            0s 17ms/step -
loss: 13.4989 - mae: 2.7144 - val_loss: 80.6228 - val_mae: 5.5429
Epoch 41/100
12/12            0s 13ms/step -
loss: 14.2210 - mae: 2.7939 - val_loss: 79.4670 - val_mae: 5.4792
Epoch 42/100
12/12            0s 10ms/step -
```

```
loss: 14.6390 - mae: 2.8089 - val_loss: 79.1659 - val_mae: 5.4624
Epoch 43/100
12/12            0s 11ms/step -
loss: 15.7153 - mae: 2.8017 - val_loss: 79.1090 - val_mae: 5.4569
Epoch 44/100
12/12            0s 14ms/step -
loss: 12.6706 - mae: 2.6450 - val_loss: 81.5202 - val_mae: 5.4822
Epoch 45/100
12/12            0s 15ms/step -
loss: 15.8215 - mae: 2.8039 - val_loss: 77.5840 - val_mae: 5.3745
Epoch 46/100
12/12            0s 12ms/step -
loss: 14.3290 - mae: 2.7255 - val_loss: 78.9811 - val_mae: 5.4156
Epoch 47/100
12/12            0s 7ms/step - loss:
12.9319 - mae: 2.7115 - val_loss: 78.3627 - val_mae: 5.3878
Epoch 48/100
12/12            0s 8ms/step - loss:
14.2423 - mae: 2.7802 - val_loss: 78.9930 - val_mae: 5.3941
Epoch 49/100
12/12            0s 8ms/step - loss:
12.0062 - mae: 2.5779 - val_loss: 83.2168 - val_mae: 5.4871
Epoch 50/100
12/12            0s 8ms/step - loss:
17.9151 - mae: 2.8245 - val_loss: 79.7980 - val_mae: 5.3917
Epoch 51/100
12/12            0s 8ms/step - loss:
14.7325 - mae: 2.7219 - val_loss: 80.3248 - val_mae: 5.4057
Epoch 52/100
12/12            0s 7ms/step - loss:
13.0966 - mae: 2.5347 - val_loss: 80.3539 - val_mae: 5.3950
Epoch 53/100
12/12            0s 7ms/step - loss:
13.2195 - mae: 2.7199 - val_loss: 78.6914 - val_mae: 5.3392
Epoch 54/100
12/12            0s 8ms/step - loss:
11.7513 - mae: 2.6655 - val_loss: 80.4472 - val_mae: 5.3702
Epoch 55/100
12/12            0s 8ms/step - loss:
12.6565 - mae: 2.5602 - val_loss: 80.7330 - val_mae: 5.3987
Epoch 56/100
12/12            0s 8ms/step - loss:
11.2585 - mae: 2.4854 - val_loss: 78.8829 - val_mae: 5.3340
Epoch 57/100
12/12            0s 8ms/step - loss:
13.1163 - mae: 2.5514 - val_loss: 81.9629 - val_mae: 5.4354
Epoch 58/100
12/12            0s 8ms/step - loss:
```

```
12.8630 - mae: 2.5923 - val_loss: 79.1199 - val_mae: 5.3440
Epoch 59/100
12/12            0s 7ms/step - loss:
13.8013 - mae: 2.7128 - val_loss: 78.4612 - val_mae: 5.2757
Epoch 60/100
12/12            0s 8ms/step - loss:
11.1373 - mae: 2.4907 - val_loss: 81.3893 - val_mae: 5.3249
Epoch 61/100
12/12            0s 8ms/step - loss:
12.0066 - mae: 2.5024 - val_loss: 81.4757 - val_mae: 5.3353
Epoch 62/100
12/12            0s 8ms/step - loss:
10.8098 - mae: 2.4688 - val_loss: 79.1132 - val_mae: 5.2593
Epoch 63/100
12/12            0s 8ms/step - loss:
11.0650 - mae: 2.5097 - val_loss: 81.1247 - val_mae: 5.3110
Epoch 64/100
12/12            0s 7ms/step - loss:
12.7146 - mae: 2.5578 - val_loss: 81.1450 - val_mae: 5.3239
Epoch 65/100
12/12            0s 8ms/step - loss:
13.3256 - mae: 2.6110 - val_loss: 79.8177 - val_mae: 5.3016
Epoch 66/100
12/12            0s 8ms/step - loss:
12.9838 - mae: 2.6001 - val_loss: 82.4926 - val_mae: 5.3245
Epoch 67/100
12/12            0s 8ms/step - loss:
11.5377 - mae: 2.4639 - val_loss: 81.5450 - val_mae: 5.2797
Epoch 68/100
12/12            0s 7ms/step - loss:
11.8785 - mae: 2.4991 - val_loss: 79.1794 - val_mae: 5.2233
Epoch 69/100
12/12            0s 7ms/step - loss:
12.7058 - mae: 2.5602 - val_loss: 82.5923 - val_mae: 5.3161
Epoch 70/100
12/12            0s 7ms/step - loss:
12.2985 - mae: 2.5662 - val_loss: 82.3204 - val_mae: 5.2990
Epoch 71/100
12/12            0s 8ms/step - loss:
14.3597 - mae: 2.6181 - val_loss: 79.5691 - val_mae: 5.2133
Epoch 72/100
12/12            0s 12ms/step -
loss: 11.6386 - mae: 2.4731 - val_loss: 80.2985 - val_mae: 5.2331
Epoch 73/100
12/12            0s 7ms/step - loss:
13.7512 - mae: 2.6286 - val_loss: 81.7844 - val_mae: 5.2861
Epoch 74/100
12/12            0s 7ms/step - loss:
```

```
12.6466 - mae: 2.5278 - val_loss: 80.9901 - val_mae: 5.2580
Epoch 75/100
12/12            0s 7ms/step - loss:
11.1071 - mae: 2.3967 - val_loss: 80.5174 - val_mae: 5.1992
Epoch 76/100
12/12            0s 7ms/step - loss:
12.8600 - mae: 2.5432 - val_loss: 81.5574 - val_mae: 5.2612
Epoch 77/100
12/12            0s 7ms/step - loss:
11.4877 - mae: 2.4095 - val_loss: 82.2853 - val_mae: 5.2791
Epoch 78/100
12/12            0s 7ms/step - loss:
10.6604 - mae: 2.3342 - val_loss: 78.6105 - val_mae: 5.1563
Epoch 79/100
12/12            0s 8ms/step - loss:
11.1722 - mae: 2.4850 - val_loss: 83.1291 - val_mae: 5.2623
Epoch 80/100
12/12            0s 9ms/step - loss:
12.6258 - mae: 2.5957 - val_loss: 83.0356 - val_mae: 5.2622
Epoch 81/100
12/12            0s 8ms/step - loss:
11.1563 - mae: 2.4442 - val_loss: 81.5902 - val_mae: 5.1938
Epoch 82/100
12/12            0s 7ms/step - loss:
14.3285 - mae: 2.5540 - val_loss: 81.1910 - val_mae: 5.1913
Epoch 83/100
12/12            0s 7ms/step - loss:
9.6534 - mae: 2.2786 - val_loss: 84.0884 - val_mae: 5.2746
Epoch 84/100
12/12            0s 7ms/step - loss:
9.7331 - mae: 2.2831 - val_loss: 80.4482 - val_mae: 5.1360
Epoch 85/100
12/12            0s 7ms/step - loss:
12.1052 - mae: 2.4137 - val_loss: 85.2986 - val_mae: 5.2826
Epoch 86/100
12/12            0s 7ms/step - loss:
10.5723 - mae: 2.3271 - val_loss: 81.0709 - val_mae: 5.1158
Epoch 87/100
12/12            0s 9ms/step - loss:
10.7697 - mae: 2.3541 - val_loss: 81.7043 - val_mae: 5.1856
Epoch 88/100
12/12            0s 8ms/step - loss:
11.3862 - mae: 2.3504 - val_loss: 83.6433 - val_mae: 5.2358
Epoch 89/100
12/12            0s 8ms/step - loss:
10.4779 - mae: 2.3559 - val_loss: 82.3865 - val_mae: 5.1840
Epoch 90/100
12/12            0s 7ms/step - loss:
```

```
13.4379 - mae: 2.4991 - val_loss: 81.7888 - val_mae: 5.2069
Epoch 91/100
12/12                  0s 7ms/step - loss:
9.4054 - mae: 2.2485 - val_loss: 81.8825 - val_mae: 5.1963
Epoch 92/100
12/12                  0s 7ms/step - loss:
10.2854 - mae: 2.3362 - val_loss: 81.0412 - val_mae: 5.1106
Epoch 93/100
12/12                  0s 7ms/step - loss:
10.6862 - mae: 2.4048 - val_loss: 83.2576 - val_mae: 5.2124
Epoch 94/100
12/12                  0s 7ms/step - loss:
9.6060 - mae: 2.2149 - val_loss: 81.4294 - val_mae: 5.0666
Epoch 95/100
12/12                  0s 8ms/step - loss:
9.5531 - mae: 2.2138 - val_loss: 88.7827 - val_mae: 5.3905
Epoch 96/100
12/12                  0s 8ms/step - loss:
9.9675 - mae: 2.3416 - val_loss: 82.0919 - val_mae: 5.0625
Epoch 97/100
12/12                  0s 8ms/step - loss:
10.0462 - mae: 2.3966 - val_loss: 83.0600 - val_mae: 5.1738
Epoch 98/100
12/12                  0s 8ms/step - loss:
9.0970 - mae: 2.2369 - val_loss: 88.1156 - val_mae: 5.3620
Epoch 99/100
12/12                  0s 8ms/step - loss:
11.5422 - mae: 2.4479 - val_loss: 83.5703 - val_mae: 5.1216
Epoch 100/100
12/12                  0s 8ms/step - loss:
10.2971 - mae: 2.3681 - val_loss: 84.1505 - val_mae: 5.2347
```

[29]:
```
<keras.src.callbacks.history.History at 0x7fd5fe6c44d0>
```

[30]:
```
output = model.evaluate(x_test,y_test)
```

```
4/4                0s 11ms/step - loss:
20.8191 - mae: 3.0977
```

[31]:
```
print(f"Mean Squared Error: {output[0]}"
      ,f"Mean Absolute Error: {output[1]}",sep="\n")
```

```
Mean Squared Error: 22.908056259155273
Mean Absolute Error: 3.1313369274139404
```

[32]:
```
y_pred = model.predict(x=x_test)
```

```
4/4                0s 26ms/step
```

[33]:
```
print(*zip(y_pred,y_test))
```

```
(array([24.398563], dtype=float32), 28.4) (array([30.642723], dtype=float32),
31.1) (array([26.176365], dtype=float32), 23.5) (array([27.232159],
dtype=float32), 26.6) (array([19.975266], dtype=float32), 19.6)
(array([16.61432], dtype=float32), 14.3) (array([41.910316], dtype=float32),
50.0) (array([14.907179], dtype=float32), 14.3) (array([19.684042],
dtype=float32), 20.7) (array([42.172394], dtype=float32), 37.6)
(array([18.18777], dtype=float32), 20.4) (array([26.123325], dtype=float32),
27.5) (array([22.385643], dtype=float32), 36.2) (array([32.33351],
dtype=float32), 32.0) (array([30.99995], dtype=float32), 33.1)
(array([51.13626], dtype=float32), 48.8) (array([25.69979], dtype=float32),
24.6) (array([19.75602], dtype=float32), 26.4) (array([21.144825],
dtype=float32), 23.2) (array([19.387259], dtype=float32), 17.0)
(array([33.214046], dtype=float32), 41.3) (array([15.868946], dtype=float32),
14.9) (array([22.070572], dtype=float32), 18.5) (array([24.727432],
dtype=float32), 25.0) (array([36.854473], dtype=float32), 36.4)
(array([21.067219], dtype=float32), 19.5) (array([18.155659], dtype=float32),
27.1) (array([16.818901], dtype=float32), 14.9) (array([42.500572],
dtype=float32), 46.0) (array([10.98995], dtype=float32), 17.9)
(array([34.53775], dtype=float32), 30.3) (array([31.79146], dtype=float32),
31.6) (array([26.373589], dtype=float32), 23.1) (array([24.002913],
dtype=float32), 24.7) (array([15.220239], dtype=float32), 16.7)
(array([19.823938], dtype=float32), 18.3) (array([8.75378], dtype=float32), 8.4)
(array([31.637682], dtype=float32), 37.3) (array([24.803143], dtype=float32),
22.1) (array([24.12807], dtype=float32), 22.0) (array([38.498226],
dtype=float32), 46.7) (array([26.023102], dtype=float32), 30.1)
(array([13.792907], dtype=float32), 12.1) (array([29.333698], dtype=float32),
29.1) (array([17.089588], dtype=float32), 16.6) (array([27.06791],
dtype=float32), 23.9) (array([17.87679], dtype=float32), 19.9)
(array([18.509321], dtype=float32), 21.4) (array([43.958275], dtype=float32),
45.4) (array([16.591486], dtype=float32), 15.6) (array([20.114769],
dtype=float32), 22.7) (array([14.622158], dtype=float32), 12.5)
(array([20.296051], dtype=float32), 24.3) (array([38.532288], dtype=float32),
43.8) (array([24.152489], dtype=float32), 22.0) (array([34.399048],
dtype=float32), 33.8) (array([19.507032], dtype=float32), 19.3)
(array([18.91164], dtype=float32), 22.6) (array([22.18611], dtype=float32),
16.1) (array([21.447088], dtype=float32), 15.0) (array([18.676918],
dtype=float32), 19.6) (array([20.76945], dtype=float32), 21.2)
(array([50.898426], dtype=float32), 50.0) (array([55.270145], dtype=float32),
50.0) (array([27.642078], dtype=float32), 29.4) (array([15.435604],
dtype=float32), 17.8) (array([24.540203], dtype=float32), 22.8)
(array([11.377972], dtype=float32), 8.8) (array([27.366789], dtype=float32),
32.5) (array([39.37468], dtype=float32), 42.8) (array([16.866665],
dtype=float32), 12.6) (array([28.097702], dtype=float32), 28.6)
(array([17.903156], dtype=float32), 19.1) (array([21.560253], dtype=float32),
50.0) (array([21.379505], dtype=float32), 27.5) (array([11.579738],
```

dtype=float32), 23.7) (array([47.035942], dtype=float32), 50.0) (array([9.9491825], dtype=float32), 7.2) (array([20.152653], dtype=float32), 18.7) (array([32.45575], dtype=float32), 37.0) (array([19.778667], dtype=float32), 22.9) (array([25.007435], dtype=float32), 22.9) (array([19.67302], dtype=float32), 17.1) (array([24.248756], dtype=float32), 22.0) (array([30.96221], dtype=float32), 23.6) (array([25.785095], dtype=float32), 23.9) (array([25.849838], dtype=float32), 27.1) (array([34.126625], dtype=float32), 29.0) (array([23.998955], dtype=float32), 22.2) (array([11.075442], dtype=float32), 7.0) (array([23.129673], dtype=float32), 20.7) (array([21.060366], dtype=float32), 18.5) (array([23.529358], dtype=float32), 21.6) (array([24.402624], dtype=float32), 23.0) (array([18.886717], dtype=float32), 16.0) (array([18.79022], dtype=float32), 15.0) (array([25.424364], dtype=float32), 23.9) (array([19.069595], dtype=float32), 24.4) (array([21.107948], dtype=float32), 22.6) (array([18.68443], dtype=float32), 19.8) (array([21.469868], dtype=float32), 22.2) (array([19.680819], dtype=float32), 18.6) (array([19.106083], dtype=float32), 19.7) (array([23.26202], dtype=float32), 23.1) (array([13.743537], dtype=float32), 13.5) (array([20.69305], dtype=float32), 21.2) (array([19.181442], dtype=float32), 23.1) (array([15.548941], dtype=float32), 13.6) (array([28.404343], dtype=float32), 22.8) (array([23.594297], dtype=float32), 18.2) (array([11.3621235], dtype=float32), 13.1) (array([17.689964], dtype=float32), 23.2) (array([24.323505], dtype=float32), 22.8) (array([24.362988], dtype=float32), 25.1) (array([21.5598], dtype=float32), 18.9) (array([13.411655], dtype=float32), 10.9) (array([13.9047365], dtype=float32), 19.3) (array([20.843494], dtype=float32), 17.4) (array([18.694859], dtype=float32), 15.6) (array([20.088127], dtype=float32), 20.6) (array([29.624937], dtype=float32), 50.0) (array([35.08461], dtype=float32), 32.7) (array([20.963634], dtype=float32), 21.8) (array([16.399689], dtype=float32), 13.4) (array([17.689772], dtype=float32), 16.6) (array([23.295284], dtype=float32), 23.6) (array([12.578907], dtype=float32), 11.0)