dl-ex-2

March 8, 2023

[3]:
```
# importing packages
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import sklearn.datasets
```

[49]:
```
# reading Breast Cancer Dataset
breast_cancer = sklearn.datasets.load_breast_cancer()
```

[5]:
```
x = breast_cancer.data
y = breast_cancer.target
```

[50]:
```
print(x)
```

```
[0 1 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 0 0 0 0 0]
```

[51]:
```
print(y)
```

```
1
```

[52]:
```
data = pd.DataFrame(breast_cancer.data, columns=breast_cancer.feature_names)
```

[9]:
```
c=breast_cancer.feature_names
c.shape
```

[9]:
```
(30,)
```

[53]:
```
data['class'] = breast_cancer.target
```

[11]:
```
data.describe()
```

[11]:

	mean radius	mean texture	mean perimeter	mean area	\
count	569.000000	569.000000	569.000000	569.000000	
mean	14.127292	19.289649	91.969033	654.889104	
std	3.524049	4.301036	24.298981	351.914129	
min	6.981000	9.710000	43.790000	143.500000	
25%	11.700000	16.170000	75.170000	420.300000	
50%	13.370000	18.840000	86.240000	551.100000	
75%	15.780000	21.800000	104.100000	782.700000	

```
max      28.110000     39.280000      188.500000  2501.000000

       mean smoothness  mean compactness  mean concavity  mean concave points  \
count       569.000000        569.000000      569.000000           569.000000
mean          0.096360          0.104341        0.088799             0.048919
std           0.014064          0.052813        0.079720             0.038803
min           0.052630          0.019380        0.000000             0.000000
25%           0.086370          0.064920        0.029560             0.020310
50%           0.095870          0.092630        0.061540             0.033500
75%           0.105300          0.130400        0.130700             0.074000
max           0.163400          0.345400        0.426800             0.201200

       mean symmetry  mean fractal dimension  …  worst texture  \
count     569.000000              569.000000  …     569.000000
mean        0.181162                0.062798  …      25.677223
std         0.027414                0.007060  …       6.146258
min         0.106000                0.049960  …      12.020000
25%         0.161900                0.057700  …      21.080000
50%         0.179200                0.061540  …      25.410000
75%         0.195700                0.066120  …      29.720000
max         0.304000                0.097440  …      49.540000

       worst perimeter   worst area  worst smoothness  worst compactness  \
count       569.000000   569.000000        569.000000         569.000000
mean        107.261213   880.583128          0.132369           0.254265
std          33.602542   569.356993          0.022832           0.157336
min          50.410000   185.200000          0.071170           0.027290
25%          84.110000   515.300000          0.116600           0.147200
50%          97.660000   686.500000          0.131300           0.211900
75%         125.400000  1084.000000          0.146000           0.339100
max         251.200000  4254.000000          0.222600           1.058000

       worst concavity  worst concave points  worst symmetry  \
count       569.000000            569.000000      569.000000
mean          0.272188              0.114606        0.290076
std           0.208624              0.065732        0.061867
min           0.000000              0.000000        0.156500
25%           0.114500              0.064930        0.250400
50%           0.226700              0.099930        0.282200
75%           0.382900              0.161400        0.317900
max           1.252000              0.291000        0.663800

       worst fractal dimension       class
count               569.000000  569.000000
mean                  0.083946    0.627417
std                   0.018061    0.483918
min                   0.055040    0.000000
```

```
25%                    0.071460    0.000000
50%                    0.080040    1.000000
75%                    0.092080    1.000000
max                    0.207500    1.000000

[8 rows x 31 columns]
```

[12]:
```
print(data['class'].value_counts())
```

```
1    357
0    212
Name: class, dtype: int64
```

[13]:
```
print(breast_cancer.target_names)
```

```
['malignant' 'benign']
```

[14]:
```
data.groupby('class').mean()
```

[14]:

class	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	…	worst radius	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension
0	17.462830	21.604906	115.365377	978.376415	0.102898	0.145188	0.160775	0.087990	0.192909	0.062680	…	21.134811	29.318208	141.370330	1422.286321	0.144845	0.374824	0.450606	0.182237	0.323468	0.091530
1	12.146524	17.914762	78.075406	462.790196	0.092478	0.080085	0.046058	0.025717	0.174186	0.062867	…	13.379801	23.515070	87.005938	558.899440	0.124959	0.182673	0.166238	0.074444	0.270246	

```
1                    0.079442

[2 rows x 30 columns]
```

[15]:
```
from sklearn.model_selection import train_test_split
```

[54]:
```
x = data.drop('class', axis=1)
y=data['class']
```

[17]:
```
type(x)
```

[17]:
```
pandas.core.frame.DataFrame
```

[18]:
```
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.1)
```

[19]:
```
print(x.shape, X_train.shape, X_test.shape)
```

```
(569, 30) (512, 30) (57, 30)
```

[20]:
```
print(y.shape, Y_train.shape, Y_test.shape)
```

```
(569,) (512,) (57,)
```

[21]:
```
print(y.mean(), Y_train.mean(), Y_test.mean())
```

```
0.6274165202108963 0.626953125 0.631578947368421
```

[22]:
```
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.1,␣
↪stratify = y)
```

[23]:
```
print(y.mean(), Y_train.mean(), Y_test.mean()) # now ratio is maintained of␣
↪malignant and benign
```

```
0.6274165202108963 0.626953125 0.631578947368421
```

[24]:
```
print(x.mean(), X_train.mean(), X_test.mean())
```

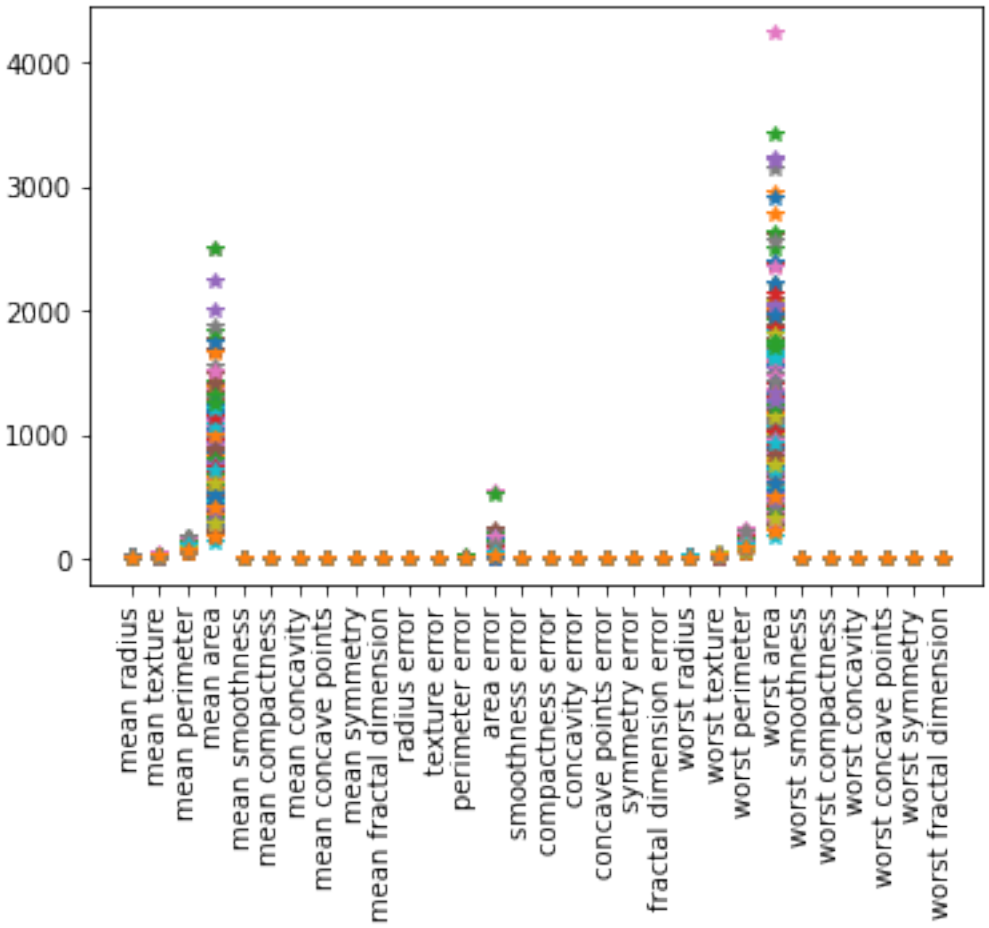
```
mean radius                14.127292
mean texture               19.289649
mean perimeter             91.969033
mean area                 654.889104
mean smoothness             0.096360
mean compactness            0.104341
mean concavity              0.088799
mean concave points         0.048919
mean symmetry               0.181162
mean fractal dimension      0.062798
radius error                0.405172
```

```
texture error                1.216853
perimeter error              2.866059
area error                  40.337079
smoothness error             0.007041
compactness error            0.025478
concavity error              0.031894
concave points error         0.011796
symmetry error               0.020542
fractal dimension error      0.003795
worst radius                16.269190
worst texture               25.677223
worst perimeter            107.261213
worst area                 880.583128
worst smoothness             0.132369
worst compactness            0.254265
worst concavity              0.272188
worst concave points         0.114606
worst symmetry               0.290076
worst fractal dimension      0.083946
dtype: float64 mean radius                  14.148947
mean texture                19.260781
mean perimeter              92.144883
mean area                  657.147461
mean smoothness              0.096602
mean compactness             0.105335
mean concavity               0.089602
mean concave points          0.049290
mean symmetry                0.181593
mean fractal dimension       0.062894
radius error                 0.407755
texture error                1.213243
perimeter error              2.883710
area error                  40.726930
smoothness error             0.007019
compactness error            0.025872
concavity error              0.032301
concave points error         0.011803
symmetry error               0.020651
fractal dimension error      0.003822
worst radius                16.296998
worst texture               25.622910
worst perimeter            107.482012
worst area                 883.696680
worst smoothness             0.132448
worst compactness            0.255780
worst concavity              0.273197
worst concave points         0.114995
worst symmetry               0.290359
```

```
worst fractal dimension      0.084003
dtype: float64 mean radius                 13.932772
mean texture                19.548947
mean perimeter              90.389474
mean area                  634.603509
mean smoothness              0.094188
mean compactness             0.095414
mean concavity               0.081590
mean concave points          0.045589
mean symmetry                0.177286
mean fractal dimension       0.061932
radius error                 0.381972
texture error                1.249286
perimeter error              2.707509
area error                  36.835263
smoothness error             0.007238
compactness error            0.021943
concavity error              0.028233
concave points error         0.011738
symmetry error               0.019563
fractal dimension error      0.003551
worst radius                16.019404
worst texture               26.165088
worst perimeter            105.277895
worst area                 852.615789
worst smoothness             0.131656
worst compactness            0.240655
worst concavity              0.263127
worst concave points         0.111118
worst symmetry               0.287530
worst fractal dimension      0.083433
dtype: float64
```

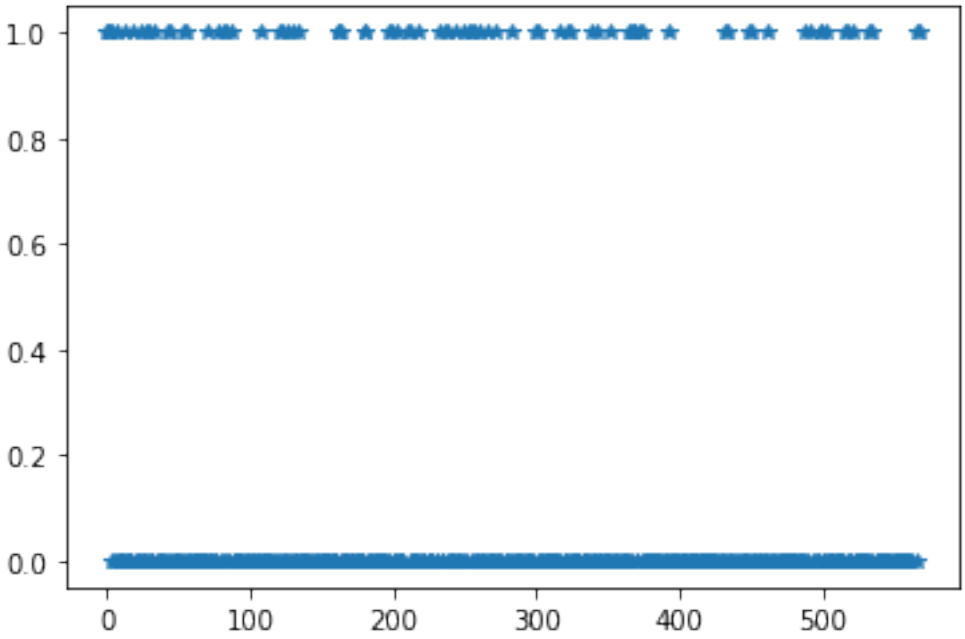
[25]:
```
import matplotlib.pyplot as plt
```

[26]:
```
plt.plot(X_train.T, '*')
plt.xticks(rotation='vertical')
plt.show()
```

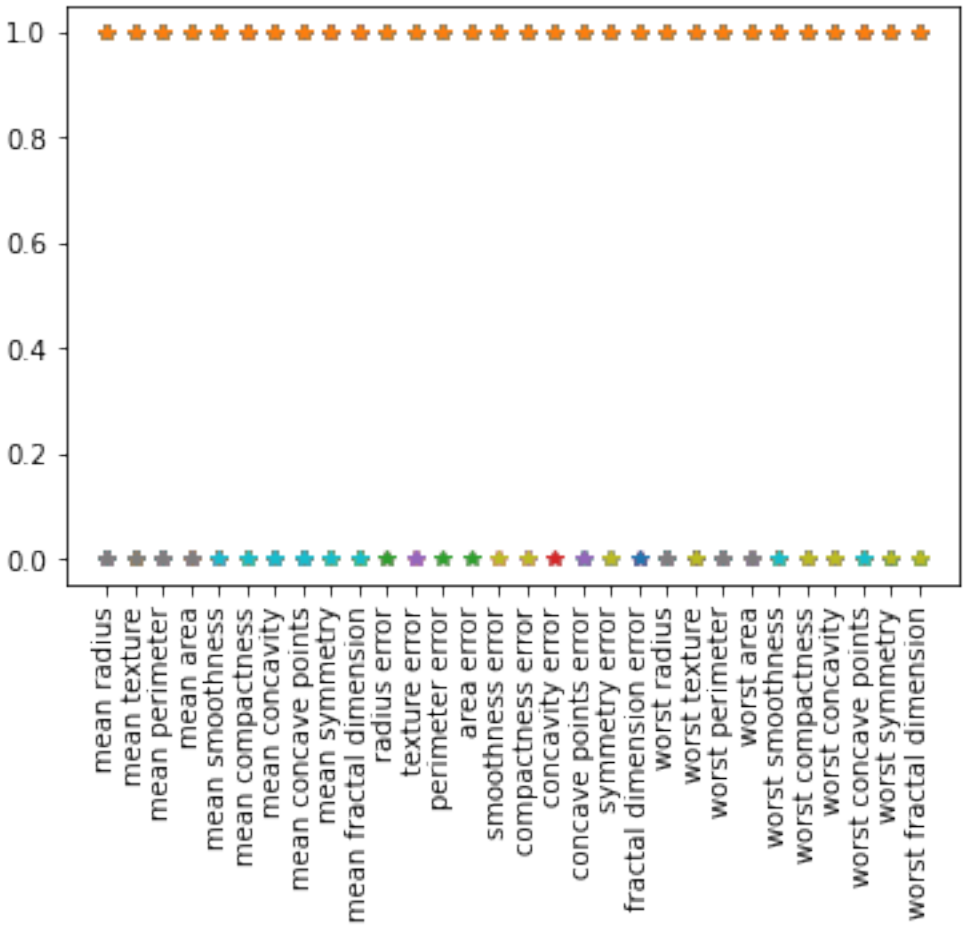


[28]:
```
x_binarised_3_train = X_train['mean area'].map(lambda x: 0 if x < 1000 else 1)
#x_binarised_3_train
plt.plot(x_binarised_3_train, '*')
```

[28]: [<matplotlib.lines.Line2D at 0x7ff1f9cf60a0>]



[30]:
```
x_binarised_train = X_train.apply(pd.cut, bins=2, labels=[1,0])
#x_binarised_train
plt.plot(x_binarised_train.T, '*')
plt.xticks(rotation='vertical')
plt.show()
```



[32]:
```
x_binarised_test = X_test.apply(pd.cut, bins=2, labels=[1,0])
#x_binarised_test
type(x_binarised_test)
```

[32]:
```
pandas.core.frame.DataFrame
```

[33]:
```
x_binarised_test = x_binarised_test.values
x_binarised_train = x_binarised_train.values
x_binarised_train
x_binarised_test
```

[33]:
```
array([[1, 1, 1, …, 1, 1, 1],
       [1, 1, 1, …, 1, 1, 1],
       [0, 1, 0, …, 0, 1, 1],
       …,
       [1, 1, 1, …, 1, 1, 1],
       [1, 0, 1, …, 1, 1, 1],
       [0, 1, 0, …, 0, 0, 0]], dtype=object)
```

[34]:
```
type(x_binarised_train)
```

[34]:
```
numpy.ndarray
```

MP Neuron Model

[35]:
```
from random import randint
```

[40]:
```
b = 3

i = randint(0, x_binarised_train.shape[0])

print("For row", i)

if (np.sum(x_binarised_train[100, :]) >= b):
    print("MP Neuron inference is malignant")
else:
    print("MP Neuron inference is benign")

if (Y_train[i] == 1):
    print("Ground Truth is malignant")
else:
    print("Ground truth is benign")
```

```
For row 38
MP Neuron inference is malignant
Ground truth is benign
```

[41]:
```
b = 3

Y_pred_train = []
accurate_rows = 0

for x, y in zip(x_binarised_train, Y_train):
    y_pred = (np.sum(x) >= b)
    Y_pred_train.append(y_pred)
    accurate_rows += (y == y_pred)

print(accurate_rows, accurate_rows/x_binarised_train.shape[0])
```

```
321 0.626953125
```

[42]:
```
for b in range(x_binarised_train.shape[1] + 1):
    Y_pred_train = []
    accurate_rows = 0

    for x, y in zip(x_binarised_train, Y_train):
        y_pred = (np.sum(x) >= b)
```

```
        Y_pred_train.append(y_pred)
        accurate_rows += (y == y_pred)

    print(b, accurate_rows, accurate_rows/x_binarised_train.shape[0])
```

```
0 321 0.626953125
1 321 0.626953125
2 321 0.626953125
3 321 0.626953125
4 321 0.626953125
5 321 0.626953125
6 321 0.626953125
7 321 0.626953125
8 321 0.626953125
9 321 0.626953125
10 321 0.626953125
11 321 0.626953125
12 321 0.626953125
13 321 0.626953125
14 324 0.6328125
15 326 0.63671875
16 330 0.64453125
17 335 0.654296875
18 337 0.658203125
19 341 0.666015625
20 345 0.673828125
21 354 0.69140625
22 362 0.70703125
23 372 0.7265625
24 392 0.765625
25 407 0.794921875
26 422 0.82421875
27 436 0.8515625
28 435 0.849609375
29 426 0.83203125
30 399 0.779296875
```

[43]:
```
from sklearn.metrics import accuracy_score
```

[44]:
```
b = 28

Y_pred_test = []

for x in x_binarised_test:
    y_pred = (np.sum(x) >= b)
    Y_pred_test.append(y_pred)
```

```
accuracy = accuracy_score(Y_pred_test, Y_test)
print(b,accuracy)
```

```
28 0.5614035087719298
```

MP Neuron Class

[45]:
```
class MPNeuron:

    def __init__(self):
        self.b = None

    def model(self, x):
        return(sum(x) >= self.b)

    def predict(self, X):
        Y = []
        for x in X:
            result = self.model(x)
            Y.append(result)
        return np.array(Y)

    def fit(self, X, Y):
        accuracy = {}

        for b in range(X.shape[1] + 1):
            self.b = b
            Y_pred = self.predict(X)
            accuracy[b] = accuracy_score(Y_pred, Y)

        best_b = max(accuracy,key = accuracy.get)
        self.b = best_b

        print('Optimal Value of  is', best_b)
        print('Highest accuracy is',accuracy[best_b])
```

[46]:
```
mp_neuron = MPNeuron()
mp_neuron.fit(x_binarised_train, Y_train)
```

```
Optimal Value of  is 27
Highest accuracy is 0.8515625
```

[47]:
```
Y_test_pred = mp_neuron.predict(x_binarised_test)
accuracy_test = accuracy_score(Y_test_pred, Y_test)
```

[48]:
```
print(accuracy_test)
```

```
0.631578947368421
```