assignment1

September 26, 2023

---

*You are currently looking at **version 0.1** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the Jupyter Notebook FAQ course resource.*

---

1 Assignment 1 - Introduction to Machine Learning

For this assignment, you will be using the Breast Cancer Wisconsin (Diagnostic) Database to create a classifier that can help diagnose patients. First, read through the description of the dataset (below).

[2]:
```python
import numpy as np
import pandas as pd
from sklearn.datasets import load_breast_cancer

cancer = load_breast_cancer()

print(cancer.DESCR) # Print the data set description
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
```

```
    - concavity (severity of concave portions of the contour)
    - concave points (number of concave portions of the contour)
    - symmetry
    - fractal dimension ("coastline approximation" - 1)

    The mean, standard error, and "worst" or largest (mean of the three
    worst/largest values) of these features were computed for each image,
    resulting in 30 features.  For instance, field 0 is Mean Radius, field
    10 is Radius SE, field 20 is Worst Radius.

    - class:
            - WDBC-Malignant
            - WDBC-Benign

:Summary Statistics:

===================================== ====== ======
                                       Min    Max
===================================== ====== ======
radius (mean):                        6.981  28.11
texture (mean):                       9.71   39.28
perimeter (mean):                     43.79  188.5
area (mean):                          143.5  2501.0
smoothness (mean):                    0.053  0.163
compactness (mean):                   0.019  0.345
concavity (mean):                     0.0    0.427
concave points (mean):                0.0    0.201
symmetry (mean):                      0.106  0.304
fractal dimension (mean):             0.05   0.097
radius (standard error):              0.112  2.873
texture (standard error):             0.36   4.885
perimeter (standard error):           0.757  21.98
area (standard error):                6.802  542.2
smoothness (standard error):          0.002  0.031
compactness (standard error):         0.002  0.135
concavity (standard error):           0.0    0.396
concave points (standard error):      0.0    0.053
symmetry (standard error):            0.008  0.079
fractal dimension (standard error):   0.001  0.03
radius (worst):                       7.93   36.04
texture (worst):                      12.02  49.54
perimeter (worst):                    50.41  251.2
area (worst):                         185.2  4254.0
smoothness (worst):                   0.071  0.223
compactness (worst):                  0.027  1.058
concavity (worst):                    0.0    1.252
concave points (worst):               0.0    0.291
symmetry (worst):                     0.156  0.664
```

```
    fractal dimension (worst):            0.055  0.208
    ===================================== ====== ======
```

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

:Creator: Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

:Donor: Nick Street

:Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992], a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes.

The actual linear program used to obtain the separating plane in the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. topic:: References

- W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.
- O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. Operations Research, 43(4), pages

```
570-577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning
techniques
     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77
(1994)
     163-171.
```

The object returned by `load_breast_cancer()` is a scikit-learn Bunch object, which is similar to a dictionary.

[3]:
```
cancer.keys()
```

[3]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names',
'filename', 'data_module'])
```

1.0.1 Question 0 (Example)

How many features does the breast cancer dataset have?

This function should return an integer.

[4]:
```
# You should write your whole answer within the function provided. The
↪autograder will call
# this function and compare the return value against the correct solution value
def answer_zero():
    # This function returns the number of features of the breast cancer
↪dataset, which is an integer.
    # The assignment question description will tell you the general format the
↪autograder is expecting
    return len(list(cancer['feature_names']))
    # YOUR CODE HERE
    raise NotImplementedError()

# You can examine what your function returns by calling it in the cell. If you
↪have questions
# about the assignment formats, check out the discussion forums for any FAQs
```

[ ]:
```

```

[5]:
```
answer_zero()
```

[5]: 30

[6]:
```
# len(list(cancer['feature_names']))
```

1.0.2 Question 1

Scikit-learn works with lists, numpy arrays, scipy-sparse matrices, and pandas DataFrames, so converting the dataset to a DataFrame is not necessary for training this model. Using a DataFrame does however help make many things easier such as munging data, so let's practice creating a classifier with a pandas DataFrame.

Convert the sklearn.dataset `cancer` to a DataFrame.

This function should return a `(569, 31)` DataFrame with

columns =

```
['mean radius', 'mean texture', 'mean perimeter', 'mean area',
'mean smoothness', 'mean compactness', 'mean concavity',
'mean concave points', 'mean symmetry', 'mean fractal dimension',
'radius error', 'texture error', 'perimeter error', 'area error',
'smoothness error', 'compactness error', 'concavity error',
'concave points error', 'symmetry error', 'fractal dimension error',
'worst radius', 'worst texture', 'worst perimeter', 'worst area',
'worst smoothness', 'worst compactness', 'worst concavity',
'worst concave points', 'worst symmetry', 'worst fractal dimension',
'target']
```

and index =

```
RangeIndex(start=0, stop=569, step=1)
```

[7]:
```python
def answer_one():
    # YOUR CODE HERE
    df = load_breast_cancer(as_frame=True)
    df = df.frame
    return df
    raise NotImplementedError()
```

[ ]:
```python

```

[8]:
```python
answer_one()
```

[8]:
```
     mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0          17.99         10.38          122.80     1001.0          0.11840
1          20.57         17.77          132.90     1326.0          0.08474
2          19.69         21.25          130.00     1203.0          0.10960
3          11.42         20.38           77.58      386.1          0.14250
4          20.29         14.34          135.10     1297.0          0.10030
..           …             …               …          …                …
564        21.56         22.39          142.00     1479.0          0.11100
565        20.13         28.25          131.20     1261.0          0.09780
566        16.60         28.08          108.30      858.1          0.08455
567        20.60         29.33          140.10     1265.0          0.11780
568         7.76         24.54           47.92      181.0          0.05263
```

```
     mean compactness  mean concavity  mean concave points  mean symmetry  \
0             0.27760         0.30010              0.14710         0.2419   
1             0.07864         0.08690              0.07017         0.1812   
2             0.15990         0.19740              0.12790         0.2069   
3             0.28390         0.24140              0.10520         0.2597   
4             0.13280         0.19800              0.10430         0.1809   
..                ...             ...                  ...            ...   
564           0.11590         0.24390              0.13890         0.1726   
565           0.10340         0.14400              0.09791         0.1752   
566           0.10230         0.09251              0.05302         0.1590   
567           0.27700         0.35140              0.15200         0.2397   
568           0.04362         0.00000              0.00000         0.1587   

     mean fractal dimension  ...  worst texture  worst perimeter  worst area  \
0                   0.07871  ...          17.33           184.60      2019.0   
1                   0.05667  ...          23.41           158.80      1956.0   
2                   0.05999  ...          25.53           152.50      1709.0   
3                   0.09744  ...          26.50            98.87       567.7   
4                   0.05883  ...          16.67           152.20      1575.0   
..                      ...  ...            ...              ...         ...   
564                 0.05623  ...          26.40           166.10      2027.0   
565                 0.05533  ...          38.25           155.00      1731.0   
566                 0.05648  ...          34.12           126.70      1124.0   
567                 0.07016  ...          39.42           184.60      1821.0   
568                 0.05884  ...          30.37            59.16       268.6   

     worst smoothness  worst compactness  worst concavity  \
0             0.16220            0.66560           0.7119   
1             0.12380            0.18660           0.2416   
2             0.14440            0.42450           0.4504   
3             0.20980            0.86630           0.6869   
4             0.13740            0.20500           0.4000   
..                ...                ...              ...   
564           0.14100            0.21130           0.4107   
565           0.11660            0.19220           0.3215   
566           0.11390            0.30940           0.3403   
567           0.16500            0.86810           0.9387   
568           0.08996            0.06444           0.0000   

     worst concave points  worst symmetry  worst fractal dimension  target  
0                  0.2654          0.4601                  0.11890       0  
1                  0.1860          0.2750                  0.08902       0  
2                  0.2430          0.3613                  0.08758       0  
3                  0.2575          0.6638                  0.17300       0  
4                  0.1625          0.2364                  0.07678       0  
..                    ...             ...                      ...     ...  
```

```
564                 0.2216          0.2060                    0.07115       0
565                 0.1628          0.2572                    0.06637       0
566                 0.1418          0.2218                    0.07820       0
567                 0.2650          0.4087                    0.12400       0
568                 0.0000          0.2871                    0.07039       1

[569 rows x 31 columns]
```

[9]:
```python
# converting the scikit-learn bunch object into a dataframe

# df = load_breast_cancer(as_frame=True)
# df = df.frame
```

1.0.3 Question 2

What is the class distribution? (i.e. how many instances of `malignant` and how many `benign`?)

This function should return a Series named **`target`** *of length 2 with integer values and index =* `['malignant', 'benign']`

[47]:
```python
def answer_two():

    # YOUR CODE HERE

    data = answer_one()

    target = data.target.value_counts()

    target.index = ['malignant', 'benign']

    return target

    raise NotImplementedError()
```

[ ]:
```python

```

[48]:
```python
answer_two()
```

[48]:
```
malignant    357
benign       212
Name: target, dtype: int64
```

[12]:
```python
# data = answer_one()
```

[13]:
```python
# count = data.target.value_counts()

# count
```

[14]:
```
# # ser = pd.Series(data, index=[10, 11, 12, 13, 14])

# # letters=["a","b","c","ab","abc","abcd","bc","d"]
# # series=pd.Series(letters)

# target = count

# target.index = ['malignant', 'benign']

# target
```

1.0.4 Question 3

Split the DataFrame into X (the data) and y (the labels).

This function should return a tuple of length 2: (X, y)*, where* * X *has shape* (569, 30) * y *has shape* (569,).

[15]:
```
def answer_three():
    # YOUR CODE HERE

    data = answer_one()

    features = data.columns[:-1]
    target = data.columns[-1]

    X = data[features]
    y = data[target]

    return (X, y)

    raise NotImplementedError()
```

[ ]:

[16]:
```
answer_three()
```

[16]:
```
(     mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
 0          17.99         10.38          122.80     1001.0          0.11840
 1          20.57         17.77          132.90     1326.0          0.08474
 2          19.69         21.25          130.00     1203.0          0.10960
 3          11.42         20.38           77.58      386.1          0.14250
 4          20.29         14.34          135.10     1297.0          0.10030
 ..           …             …              …          …               …
 564        21.56         22.39          142.00     1479.0          0.11100
 565        20.13         28.25          131.20     1261.0          0.09780
 566        16.60         28.08          108.30      858.1          0.08455
 567        20.60         29.33          140.10     1265.0          0.11780
```

568	7.76	24.54	47.92	181.0	0.05263

	mean compactness	mean concavity	mean concave points	mean symmetry	\
0	0.27760	0.30010	0.14710	0.2419	
1	0.07864	0.08690	0.07017	0.1812	
2	0.15990	0.19740	0.12790	0.2069	
3	0.28390	0.24140	0.10520	0.2597	
4	0.13280	0.19800	0.10430	0.1809	
..	…	…	…	…	
564	0.11590	0.24390	0.13890	0.1726	
565	0.10340	0.14400	0.09791	0.1752	
566	0.10230	0.09251	0.05302	0.1590	
567	0.27700	0.35140	0.15200	0.2397	
568	0.04362	0.00000	0.00000	0.1587	

	mean fractal dimension	…	worst radius	worst texture	\
0	0.07871	…	25.380	17.33	
1	0.05667	…	24.990	23.41	
2	0.05999	…	23.570	25.53	
3	0.09744	…	14.910	26.50	
4	0.05883	…	22.540	16.67	
..	…	…	…	…	
564	0.05623	…	25.450	26.40	
565	0.05533	…	23.690	38.25	
566	0.05648	…	18.980	34.12	
567	0.07016	…	25.740	39.42	
568	0.05884	…	9.456	30.37	

	worst perimeter	worst area	worst smoothness	worst compactness	\
0	184.60	2019.0	0.16220	0.66560	
1	158.80	1956.0	0.12380	0.18660	
2	152.50	1709.0	0.14440	0.42450	
3	98.87	567.7	0.20980	0.86630	
4	152.20	1575.0	0.13740	0.20500	
..	…	…	…	…	
564	166.10	2027.0	0.14100	0.21130	
565	155.00	1731.0	0.11660	0.19220	
566	126.70	1124.0	0.11390	0.30940	
567	184.60	1821.0	0.16500	0.86810	
568	59.16	268.6	0.08996	0.06444	

	worst concavity	worst concave points	worst symmetry	\
0	0.7119	0.2654	0.4601	
1	0.2416	0.1860	0.2750	
2	0.4504	0.2430	0.3613	
3	0.6869	0.2575	0.6638	
4	0.4000	0.1625	0.2364	

```
..              …                      …               …
564          0.4107                 0.2216          0.2060
565          0.3215                 0.1628          0.2572
566          0.3403                 0.1418          0.2218
567          0.9387                 0.2650          0.4087
568          0.0000                 0.0000          0.2871

     worst fractal dimension
0                    0.11890
1                    0.08902
2                    0.08758
3                    0.17300
4                    0.07678
..                       …
564                  0.07115
565                  0.06637
566                  0.07820
567                  0.12400
568                  0.07039

[569 rows x 30 columns],
0      0
1      0
2      0
3      0
4      0
      ..
564    0
565    0
566    0
567    0
568    1
Name: target, Length: 569, dtype: int64)
```

[17]:
```
# data = answer_one()

# data.head()
```

[18]:
```
# features = data.columns[:-1]
```

[19]:
```
# target = data.columns[-1]
```

[20]:
```
# data.columns
```

[21]:
```
# X = data[features]

# X
```

[22]:
```
# y = data[target]

# y
```

1.0.5 Question 4

Using `train_test_split`, split `X` and `y` into training and test sets (`X_train, X_test, y_train, and y_test`).

Set the random number generator state to 0 using `random_state=0` to make sure your results match the autograder!

This function should return a tuple of length 4: `(X_train, X_test, y_train, y_test)`, *where* * `X_train` *has shape* `(426, 30)` * `X_test` *has shape* `(143, 30)` * `y_train` *has shape* `(426,)` * `y_test` *has shape* `(143,)`

[23]:
```
from sklearn.model_selection import train_test_split

def answer_four():
    # YOUR CODE HERE

    return train_test_split(answer_three()[0], answer_three()[1], random_state␣
↪= 0)

    raise NotImplementedError()
```

[ ]:
```
```

[24]:
```
answer_four()
```

[24]:
```
[     mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
 293       11.850         17.46           75.54      432.7          0.08372
 332       11.220         19.86           71.94      387.3          0.10540
 565       20.130         28.25          131.20     1261.0          0.09780
 278       13.590         17.84           86.24      572.3          0.07948
 489       16.690         20.20          107.10      857.6          0.07497
 ..           …             …               …          …                …
 277       18.810         19.98          120.90     1102.0          0.08923
 9         12.460         24.04           83.97      475.9          0.11860
 359        9.436         18.32           59.82      278.6          0.10090
 192        9.720         18.22           60.73      288.1          0.06950
 559       11.510         23.93           74.52      403.5          0.09261

      mean compactness  mean concavity  mean concave points  mean symmetry  \
 293           0.05642        0.026880             0.022800         0.1875
 332           0.06779        0.005006             0.007583         0.1940
 565           0.10340        0.144000             0.097910         0.1752
 278           0.04052        0.019970             0.012380         0.1573
```

489	0.07112	0.036490	0.023070	0.1846
..	…	…	…	…
277	0.05884	0.080200	0.058430	0.1550
9	0.23960	0.227300	0.085430	0.2030
359	0.05956	0.027100	0.014060	0.1506
192	0.02344	0.000000	0.000000	0.1653
559	0.10210	0.111200	0.041050	0.1388

	mean fractal dimension	…	worst radius	worst texture	\
293	0.05715	…	13.060	25.75	
332	0.06028	…	11.980	25.78	
565	0.05533	…	23.690	38.25	
278	0.05520	…	15.500	26.10	
489	0.05325	…	19.180	26.56	
..	…	…	…	…	
277	0.04996	…	19.960	24.30	
9	0.08243	…	15.090	40.68	
359	0.06959	…	12.020	25.02	
192	0.06447	…	9.968	20.83	
559	0.06570	…	12.480	37.16	

	worst perimeter	worst area	worst smoothness	worst compactness	\
293	84.35	517.8	0.13690	0.17580	
332	76.91	436.1	0.14240	0.09669	
565	155.00	1731.0	0.11660	0.19220	
278	98.91	739.1	0.10500	0.07622	
489	127.30	1084.0	0.10090	0.29200	
..	…	…	…	…	
277	129.00	1236.0	0.12430	0.11600	
9	97.65	711.4	0.18530	1.05800	
359	75.79	439.6	0.13330	0.10490	
192	62.25	303.8	0.07117	0.02729	
559	82.28	474.2	0.12980	0.25170	

	worst concavity	worst concave points	worst symmetry	\
293	0.13160	0.09140	0.3101	
332	0.01335	0.02022	0.3292	
565	0.32150	0.16280	0.2572	
278	0.10600	0.05185	0.2335	
489	0.24770	0.08737	0.4677	
..	…	…	…	
277	0.22100	0.12940	0.2567	
9	1.10500	0.22100	0.4366	
359	0.11440	0.05052	0.2454	
192	0.00000	0.00000	0.1909	
559	0.36300	0.09653	0.2112	

```
     worst fractal dimension
293                  0.07007
332                  0.06522
565                  0.06637
278                  0.06263
489                  0.07623
..                       ...
277                  0.05737
9                    0.20750
359                  0.08136
192                  0.06559
559                  0.08732

[426 rows x 30 columns],
     mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
512        13.40         20.52           88.64      556.7          0.11060
457        13.21         25.25           84.10      537.9          0.08791
439        14.02         15.66           89.59      606.5          0.07966
298        14.26         18.17           91.22      633.1          0.06576
37         13.03         18.42           82.61      523.8          0.08983
..           ...           ...             ...        ...              ...
236        23.21         26.97          153.50     1670.0          0.09509
113        10.51         20.19           68.64      334.2          0.11220
527        12.34         12.27           78.94      468.5          0.09003
76         13.53         10.94           87.91      559.2          0.12910
162        19.59         18.15          130.70     1214.0          0.11200

     mean compactness  mean concavity  mean concave points  mean symmetry  \
512           0.14690         0.14450              0.08172         0.2116
457           0.05205         0.02772              0.02068         0.1619
439           0.05581         0.02087              0.02652         0.1589
298           0.05220         0.02475              0.01374         0.1635
37            0.03766         0.02562              0.02923         0.1467
..                ...             ...                  ...            ...
236           0.16820         0.19500              0.12370         0.1909
113           0.13030         0.06476              0.03068         0.1922
527           0.06307         0.02958              0.02647         0.1689
76            0.10470         0.06877              0.06556         0.2403
162           0.16660         0.25080              0.12860         0.2027

     mean fractal dimension  ...  worst radius  worst texture  \
512                 0.07325  ...         16.41          29.66
457                 0.05584  ...         14.35          34.23
439                 0.05586  ...         14.91          19.31
298                 0.05586  ...         16.22          25.26
37                  0.05863  ...         13.30          22.81
..                      ...  ...           ...            ...
```

```
236                0.06309  …         31.01          34.51
113                0.07782  …         11.16          22.75
527                0.05808  …         13.61          19.27
76                 0.06641  …         14.08          12.49
162                0.06082  …         26.73          26.39

     worst perimeter  worst area  worst smoothness  worst compactness  \
512           113.30       844.4           0.15740            0.38560
457            91.29       632.9           0.12890            0.10630
439            96.53       688.9           0.10340            0.10170
298           105.80       819.7           0.09445            0.21670
37             84.46       545.9           0.09701            0.04619
..               …           …                 …                 …
236           206.00      2944.0           0.14810            0.41260
113            72.62       374.4           0.13000            0.20490
527            87.22       564.9           0.12920            0.20740
76             91.36       605.5           0.14510            0.13790
162           174.90      2232.0           0.14380            0.38460

     worst concavity  worst concave points  worst symmetry  \
512          0.51060               0.20510          0.3585
457          0.13900               0.06005          0.2444
439          0.06260               0.08216          0.2136
298          0.15650               0.07530          0.2636
37           0.04833               0.05013          0.1987
..               …                    …              …
236          0.58200               0.25930          0.3103
113          0.12950               0.06136          0.2383
527          0.17910               0.10700          0.3110
76           0.08539               0.07407          0.2710
162          0.68100               0.22470          0.3643

     worst fractal dimension
512                  0.11090
457                  0.06788
439                  0.06710
298                  0.07676
37                   0.06169
..                       …
236                  0.08677
113                  0.09026
527                  0.07592
76                   0.07191
162                  0.09223

[143 rows x 30 columns],
293    1
```

```
332    1
565    0
278    1
489    0
      ..
277    0
9      0
359    1
192    1
559    1
Name: target, Length: 426, dtype: int64,
512    0
457    1
439    1
298    1
37     1
      ..
236    0
113    1
527    1
76     1
162    0
Name: target, Length: 143, dtype: int64]
```

[25]:
```
# from sklearn.model_selection import train_test_split

# train_test_split(answer_three()[0], answer_three()[1], random_state = 0)
```

1.0.6 Question 5

Using KNeighborsClassifier, fit a k-nearest neighbors (knn) classifier with `X_train`, `y_train` and using one nearest neighbor (`n_neighbors = 1`).

*This function should return a `sklearn.neighbors.classification.KNeighborsClassifier`.

[26]:
```
from sklearn.neighbors import KNeighborsClassifier

def answer_five():
    # YOUR CODE HERE

    from sklearn.neighbors import KNeighborsClassifier

    knn = KNeighborsClassifier(n_neighbors = 1)

    return knn.fit(answer_four()[0], answer_four()[2])

    raise NotImplementedError()
```

[ ]:

```

```

[27]:

```
answer_five()
```

[27]:

```
KNeighborsClassifier(n_neighbors=1)
```

1.0.7 Question 6

Using your knn classifier, predict the class label using the mean value for each feature.

Hint: You can use `cancerdf.mean()[:-1].values.reshape(1, -1)` which gets the mean value for each feature, ignores the target column, and reshapes the data from 1 dimension to 2 (necessary for the precict method of KNeighborsClassifier).

[41]:

```
def answer_six():
    # YOUR CODE HERE

    knn = answer_five()

    cancerdf = answer_one()

    return knn.predict(cancerdf.mean()[:-1].values.reshape(1, -1))

    raise NotImplementedError()
```

[ ]:

```

```

[42]:

```
answer_six()
```

[42]:

```
array([1])
```

[43]:

```
# knn = answer_five()

# cancerdf = answer_one()

# knn.predict(cancerdf.mean()[:-1].values.reshape(1, -1))
```

1.0.8 Question 7

Using your knn classifier, predict the class labels for the test set `X_test`.

This function should return a numpy array with shape `(143,)` and values either `0.0` or `1.0`.

[29]:

```
def answer_seven():
    # YOUR CODE HERE

    knn = answer_five()
```

```
    return knn.predict(answer_four()[1])

    raise NotImplementedError()
```

[ ]:

[30]:
```
answer_seven()
```

[30]:
```
array([1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1,
       0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1,
       0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1,
       1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0])
```

[31]:
```
# knn = answer_five()

# knn.score(answer_four()[1], answer_four()[3])
```

[32]:
```
# knn = answer_five()

# knn.predict(answer_four()[1])
```

1.0.9 Question 8

Find the score (mean accuracy) of your knn classifier using `X_test` and `y_test`.

This function should return a float between 0 and 1

[33]:
```
def answer_eight():
    # YOUR CODE HERE

    knn = answer_five()

    return knn.score(answer_four()[1], answer_four()[3])

    raise NotImplementedError()
```

[ ]:

[34]:
```
answer_eight()
```

[34]:
```
0.916083916083916
```

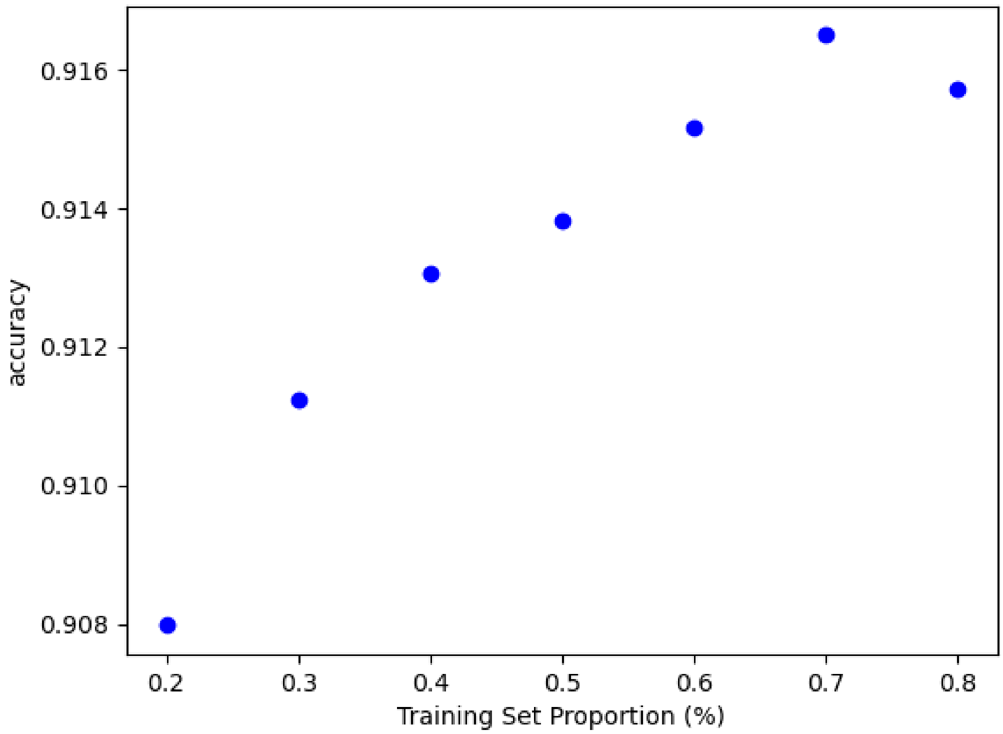
1.0.10 Optional plot

Try using the plotting function below to visualize the different predicition scores between train and test sets, as well as malignant and benign cells.

[35]:
```
def accuracy_plot():
    import matplotlib.pyplot as plt

    %matplotlib notebook




    # YOUR CODE HERE
    raise NotImplementedError()
```

[36]:
```
# Uncomment the plotting function to see the visualization,
# Comment out the plotting function when submitting your notebook for grading

# accuracy_plot()
```

[38]:
```
%matplotlib widget

import matplotlib.pyplot as plt

knn = answer_five()

t = [0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2]

plt.figure()

for s in t:
    scores = []
    for i in range(1,1000):
        X_train, X_test, y_train, y_test = train_test_split(answer_three()[0],␣
 ↪answer_three()[1], test_size = 1-s)
        knn.fit(X_train, y_train)
        scores.append(knn.score(X_test, y_test))
    plt.plot(s, np.mean(scores), 'bo')

plt.xlabel('Training Set Proportion (%)')
plt.ylabel('accuracy');
```



[ ]:
```python
# t = [0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2]

# knn = KNeighborsClassifier(n_neighbors = 5)

# plt.figure()

# for s in t:

#     scores = []
#     for i in range(1,1000):
#         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =␣
↪1-s)
#         knn.fit(X_train, y_train)
#         scores.append(knn.score(X_test, y_test))
#     plt.plot(s, np.mean(scores), 'bo')

# plt.xlabel('Training set proportion (%)')
# plt.ylabel('accuracy');
```