

# Assignment 1

June 27, 2020

---

*You are currently looking at **version 1.3** 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).

```
In [36]: 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 Wisconsin (Diagnostic) Database  
=====

Notes

-----

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<sup>2</sup> / 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 largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 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/
```

#### 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.

```
In [37]: # Check the data of cancer
         print(cancer)
```

```
{'data': array([[ 1.79900000e+01,  1.03800000e+01,  1.22800000e+02, ...,
                  2.65400000e-01,  4.60100000e-01,  1.18900000e-01],
 [ 2.05700000e+01,  1.77700000e+01,  1.32900000e+02, ...,
                  1.86000000e-01,  2.75000000e-01,  8.90200000e-02],
 [ 1.96900000e+01,  2.12500000e+01,  1.30000000e+02, ...,
                  2.43000000e-01,  3.61300000e-01,  8.75800000e-02],
 ...,
 [ 1.66000000e+01,  2.80800000e+01,  1.08300000e+02, ...,
                  1.41800000e-01,  2.21800000e-01,  7.82000000e-02],
 [ 2.06000000e+01,  2.93300000e+01,  1.40100000e+02, ...,
                  2.65000000e-01,  4.08700000e-01,  1.24000000e-01],
 [ 7.76000000e+00,  2.45400000e+01,  4.79200000e+01, ...,
                  0.00000000e+00,  2.87100000e-01,  7.03900000e-02]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1,
1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1,
0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1,
0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1,
1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1]]), 'target_names': array(['0', '1'])}
```

```
dtype='<U9'), 'DESCR': 'Breast Cancer Wisconsin (Diagnostic) Database\n=====\nmean smoothness', 'mean compactness', 'mean concavity',\n'mean concave points', 'mean symmetry', 'mean fractal dimension',\n'radius error', 'texture error', 'perimeter error', 'area error',\n'smoothness error', 'compactness error', 'concavity error',\n'concave points error', 'symmetry error', 'fractal dimension error',\n'worst radius', 'worst texture', 'worst perimeter', 'worst area',\n'worst smoothness', 'worst compactness', 'worst concavity',\n'worst concave points', 'worst symmetry', 'worst fractal dimension'],\ndtype='<U23')}
```

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

```
In [38]: cancer.keys()
```

```
Out[38]: dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names'])
```

### 1.0.1 Question 0 (Example)

How many features does the breast cancer dataset have?

*This function should return an integer.*

```
In [39]: # You should write your whole answer within the function provided. The autograder
# this function and compare the return value against the correct solution
def answer_zero():
    # This function returns the number of features of the breast cancer dataset
    # The assignment question description will tell you the general format
    return len(cancer['feature_names'])

# You can examine what your function returns by calling it in the cell. If you
# about the assignment formats, check out the discussion forums for any FAQ
answer_zero()
```

```
Out[39]: 30
```

### 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)
```

```
In [40]: def answer_one():
```

```
    # Your code here
    # add the data and the colname
    df = pd.DataFrame(data=cancer.data, columns=cancer.feature_names)
    # add the target as another col
    df['target'] = cancer.target
    return df # Return your answer
```

```
answer_one()
```

```
Out[40]:
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness
0	17.990	10.38	122.80	1001.0	0.11840
1	20.570	17.77	132.90	1326.0	0.08474
2	19.690	21.25	130.00	1203.0	0.10960
3	11.420	20.38	77.58	386.1	0.14250
4	20.290	14.34	135.10	1297.0	0.10030
5	12.450	15.70	82.57	477.1	0.12780
6	18.250	19.98	119.60	1040.0	0.09463
7	13.710	20.83	90.20	577.9	0.11890
8	13.000	21.82	87.50	519.8	0.12730
9	12.460	24.04	83.97	475.9	0.11860
10	16.020	23.24	102.70	797.8	0.08200
11	15.780	17.89	103.60	781.0	0.09710
12	19.170	24.80	132.40	1123.0	0.09740
13	15.850	23.95	103.70	782.7	0.08401
14	13.730	22.61	93.60	578.3	0.11310
15	14.540	27.54	96.73	658.8	0.11390
16	14.680	20.13	94.74	684.5	0.09867
17	16.130	20.68	108.10	798.8	0.11700
18	19.810	22.15	130.00	1260.0	0.09831
19	13.540	14.36	87.46	566.3	0.09779
20	13.080	15.71	85.63	520.0	0.10750
21	9.504	12.44	60.34	273.9	0.10240
22	15.340	14.26	102.50	704.4	0.10730

23	21.160	23.04	137.20	1404.0	0.09428
24	16.650	21.38	110.00	904.6	0.11210
25	17.140	16.40	116.00	912.7	0.11860
26	14.580	21.53	97.41	644.8	0.10540
27	18.610	20.25	122.10	1094.0	0.09440
28	15.300	25.27	102.40	732.4	0.10820
29	17.570	15.05	115.00	955.1	0.09847
..	...	...	...	...	...
539	7.691	25.44	48.34	170.4	0.08668
540	11.540	14.44	74.65	402.9	0.09984
541	14.470	24.99	95.81	656.4	0.08837
542	14.740	25.42	94.70	668.6	0.08275
543	13.210	28.06	84.88	538.4	0.08671
544	13.870	20.70	89.77	584.8	0.09578
545	13.620	23.23	87.19	573.2	0.09246
546	10.320	16.35	65.31	324.9	0.09434
547	10.260	16.58	65.85	320.8	0.08877
548	9.683	19.34	61.05	285.7	0.08491
549	10.820	24.21	68.89	361.6	0.08192
550	10.860	21.48	68.51	360.5	0.07431
551	11.130	22.44	71.49	378.4	0.09566
552	12.770	29.43	81.35	507.9	0.08276
553	9.333	21.94	59.01	264.0	0.09240
554	12.880	28.92	82.50	514.3	0.08123
555	10.290	27.61	65.67	321.4	0.09030
556	10.160	19.59	64.73	311.7	0.10030
557	9.423	27.88	59.26	271.3	0.08123
558	14.590	22.68	96.39	657.1	0.08473
559	11.510	23.93	74.52	403.5	0.09261
560	14.050	27.15	91.38	600.4	0.09929
561	11.200	29.37	70.67	386.0	0.07449
562	15.220	30.62	103.40	716.9	0.10480
563	20.920	25.09	143.00	1347.0	0.10990
564	21.560	22.39	142.00	1479.0	0.11100
565	20.130	28.25	131.20	1261.0	0.09780
566	16.600	28.08	108.30	858.1	0.08455
567	20.600	29.33	140.10	1265.0	0.11780
568	7.760	24.54	47.92	181.0	0.05263

	mean compactness	mean concavity	mean concave points	mean symmetry
0	0.27760	0.300100	0.147100	0.2419
1	0.07864	0.086900	0.070170	0.1812
2	0.15990	0.197400	0.127900	0.2069
3	0.28390	0.241400	0.105200	0.2597
4	0.13280	0.198000	0.104300	0.1809
5	0.17000	0.157800	0.080890	0.2087
6	0.10900	0.112700	0.074000	0.1794
7	0.16450	0.093660	0.059850	0.2196

8	0.19320	0.185900	0.093530	0.2350
9	0.23960	0.227300	0.085430	0.2030
10	0.06669	0.032990	0.033230	0.1528
11	0.12920	0.099540	0.066060	0.1842
12	0.24580	0.206500	0.111800	0.2397
13	0.10020	0.099380	0.053640	0.1847
14	0.22930	0.212800	0.080250	0.2069
15	0.15950	0.163900	0.073640	0.2303
16	0.07200	0.073950	0.052590	0.1586
17	0.20220	0.172200	0.102800	0.2164
18	0.10270	0.147900	0.094980	0.1582
19	0.08129	0.066640	0.047810	0.1885
20	0.12700	0.045680	0.031100	0.1967
21	0.06492	0.029560	0.020760	0.1815
22	0.21350	0.207700	0.097560	0.2521
23	0.10220	0.109700	0.086320	0.1769
24	0.14570	0.152500	0.091700	0.1995
25	0.22760	0.222900	0.140100	0.3040
26	0.18680	0.142500	0.087830	0.2252
27	0.10660	0.149000	0.077310	0.1697
28	0.16970	0.168300	0.087510	0.1926
29	0.11570	0.098750	0.079530	0.1739
..	...	...	...	...
539	0.11990	0.092520	0.013640	0.2037
540	0.11200	0.067370	0.025940	0.1818
541	0.12300	0.100900	0.038900	0.1872
542	0.07214	0.041050	0.030270	0.1840
543	0.06877	0.029870	0.032750	0.1628
544	0.10180	0.036880	0.023690	0.1620
545	0.06747	0.029740	0.024430	0.1664
546	0.04994	0.010120	0.005495	0.1885
547	0.08066	0.043580	0.024380	0.1669
548	0.05030	0.023370	0.009615	0.1580
549	0.06602	0.015480	0.008160	0.1976
550	0.04227	0.000000	0.000000	0.1661
551	0.08194	0.048240	0.022570	0.2030
552	0.04234	0.019970	0.014990	0.1539
553	0.05605	0.039960	0.012820	0.1692
554	0.05824	0.061950	0.023430	0.1566
555	0.07658	0.059990	0.027380	0.1593
556	0.07504	0.005025	0.011160	0.1791
557	0.04971	0.000000	0.000000	0.1742
558	0.13300	0.102900	0.037360	0.1454
559	0.10210	0.111200	0.041050	0.1388
560	0.11260	0.044620	0.043040	0.1537
561	0.03558	0.000000	0.000000	0.1060
562	0.20870	0.255000	0.094290	0.2128
563	0.22360	0.317400	0.147400	0.2149



564	0.11590	0.243900	0.138900	0.1726
565	0.10340	0.144000	0.097910	0.1752
566	0.10230	0.092510	0.053020	0.1590
567	0.27700	0.351400	0.152000	0.2397
568	0.04362	0.000000	0.000000	0.1587

	mean fractal dimension	...	worst texture	worst perimeter	\
0	0.07871	...	17.33	184.60	
1	0.05667	...	23.41	158.80	
2	0.05999	...	25.53	152.50	
3	0.09744	...	26.50	98.87	
4	0.05883	...	16.67	152.20	
5	0.07613	...	23.75	103.40	
6	0.05742	...	27.66	153.20	
7	0.07451	...	28.14	110.60	
8	0.07389	...	30.73	106.20	
9	0.08243	...	40.68	97.65	
10	0.05697	...	33.88	123.80	
11	0.06082	...	27.28	136.50	
12	0.07800	...	29.94	151.70	
13	0.05338	...	27.66	112.00	
14	0.07682	...	32.01	108.80	
15	0.07077	...	37.13	124.10	
16	0.05922	...	30.88	123.40	
17	0.07356	...	31.48	136.80	
18	0.05395	...	30.88	186.80	
19	0.05766	...	19.26	99.70	
20	0.06811	...	20.49	96.09	
21	0.06905	...	15.66	65.13	
22	0.07032	...	19.08	125.10	
23	0.05278	...	35.59	188.00	
24	0.06330	...	31.56	177.00	
25	0.07413	...	21.40	152.40	
26	0.06924	...	33.21	122.40	
27	0.05699	...	27.26	139.90	
28	0.06540	...	36.71	149.30	
29	0.06149	...	19.52	134.90	
..	...	...	...	...	
539	0.07751	...	31.89	54.49	
540	0.06782	...	19.68	78.78	
541	0.06341	...	31.73	113.50	
542	0.05680	...	32.29	107.40	
543	0.05781	...	37.17	92.48	
544	0.06688	...	24.75	99.17	
545	0.05801	...	29.09	97.58	
546	0.06201	...	21.77	71.12	
547	0.06714	...	22.04	71.08	
548	0.06235	...	25.59	69.10	

549	0.06328	...	31.45	83.90
550	0.05948	...	24.77	74.08
551	0.06552	...	28.26	77.80
552	0.05637	...	36.00	88.10
553	0.06576	...	25.05	62.86
554	0.05708	...	35.74	88.84
555	0.06127	...	34.91	69.57
556	0.06331	...	22.88	67.88
557	0.06059	...	34.24	66.50
558	0.06147	...	27.27	105.90
559	0.06570	...	37.16	82.28
560	0.06171	...	33.17	100.20
561	0.05502	...	38.30	75.19
562	0.07152	...	42.79	128.70
563	0.06879	...	29.41	179.10
564	0.05623	...	26.40	166.10
565	0.05533	...	38.25	155.00
566	0.05648	...	34.12	126.70
567	0.07016	...	39.42	184.60
568	0.05884	...	30.37	59.16

	worst area	worst smoothness	worst compactness	worst concavity	\
0	2019.0	0.16220	0.66560	0.71190	
1	1956.0	0.12380	0.18660	0.24160	
2	1709.0	0.14440	0.42450	0.45040	
3	567.7	0.20980	0.86630	0.68690	
4	1575.0	0.13740	0.20500	0.40000	
5	741.6	0.17910	0.52490	0.53550	
6	1606.0	0.14420	0.25760	0.37840	
7	897.0	0.16540	0.36820	0.26780	
8	739.3	0.17030	0.54010	0.53900	
9	711.4	0.18530	1.05800	1.10500	
10	1150.0	0.11810	0.15510	0.14590	
11	1299.0	0.13960	0.56090	0.39650	
12	1332.0	0.10370	0.39030	0.36390	
13	876.5	0.11310	0.19240	0.23220	
14	697.7	0.16510	0.77250	0.69430	
15	943.2	0.16780	0.65770	0.70260	
16	1138.0	0.14640	0.18710	0.29140	
17	1315.0	0.17890	0.42330	0.47840	
18	2398.0	0.15120	0.31500	0.53720	
19	711.2	0.14400	0.17730	0.23900	
20	630.5	0.13120	0.27760	0.18900	
21	314.9	0.13240	0.11480	0.08867	
22	980.9	0.13900	0.59540	0.63050	
23	2615.0	0.14010	0.26000	0.31550	
24	2215.0	0.18050	0.35780	0.46950	
25	1461.0	0.15450	0.39490	0.38530	

26	896.9	0.15250	0.66430	0.55390
27	1403.0	0.13380	0.21170	0.34460
28	1269.0	0.16410	0.61100	0.63350
29	1227.0	0.12550	0.28120	0.24890
..	...	...	...	...
539	223.6	0.15960	0.30640	0.33930
540	457.8	0.13450	0.21180	0.17970
541	808.9	0.13400	0.42020	0.40400
542	826.4	0.10600	0.13760	0.16110
543	629.6	0.10720	0.13810	0.10620
544	688.6	0.12640	0.20370	0.13770
545	729.8	0.12160	0.15170	0.10490
546	384.9	0.12850	0.08842	0.04384
547	357.4	0.14610	0.22460	0.17830
548	364.2	0.11990	0.09546	0.09350
549	505.6	0.12040	0.16330	0.06194
550	412.3	0.10010	0.07348	0.00000
551	436.6	0.10870	0.17820	0.15640
552	594.7	0.12340	0.10640	0.08653
553	295.8	0.11030	0.08298	0.07993
554	595.7	0.12270	0.16200	0.24390
555	357.6	0.13840	0.17100	0.20000
556	347.3	0.12650	0.12000	0.01005
557	330.6	0.10730	0.07158	0.00000
558	733.5	0.10260	0.31710	0.36620
559	474.2	0.12980	0.25170	0.36300
560	706.7	0.12410	0.22640	0.13260
561	439.6	0.09267	0.05494	0.00000
562	915.0	0.14170	0.79170	1.17000
563	1819.0	0.14070	0.41860	0.65990
564	2027.0	0.14100	0.21130	0.41070
565	1731.0	0.11660	0.19220	0.32150
566	1124.0	0.11390	0.30940	0.34030
567	1821.0	0.16500	0.86810	0.93870
568	268.6	0.08996	0.06444	0.00000

	worst concave points	worst symmetry	worst fractal dimension	target
0	0.26540	0.4601	0.11890	0
1	0.18600	0.2750	0.08902	0
2	0.24300	0.3613	0.08758	0
3	0.25750	0.6638	0.17300	0
4	0.16250	0.2364	0.07678	0
5	0.17410	0.3985	0.12440	0
6	0.19320	0.3063	0.08368	0
7	0.15560	0.3196	0.11510	0
8	0.20600	0.4378	0.10720	0
9	0.22100	0.4366	0.20750	0
10	0.09975	0.2948	0.08452	0

11	0.18100	0.3792	0.10480	0
12	0.17670	0.3176	0.10230	0
13	0.11190	0.2809	0.06287	0
14	0.22080	0.3596	0.14310	0
15	0.17120	0.4218	0.13410	0
16	0.16090	0.3029	0.08216	0
17	0.20730	0.3706	0.11420	0
18	0.23880	0.2768	0.07615	0
19	0.12880	0.2977	0.07259	1
20	0.07283	0.3184	0.08183	1
21	0.06227	0.2450	0.07773	1
22	0.23930	0.4667	0.09946	0
23	0.20090	0.2822	0.07526	0
24	0.20950	0.3613	0.09564	0
25	0.25500	0.4066	0.10590	0
26	0.27010	0.4264	0.12750	0
27	0.14900	0.2341	0.07421	0
28	0.20240	0.4027	0.09876	0
29	0.14560	0.2756	0.07919	0
..	...	...	...	...
539	0.05000	0.2790	0.10660	1
540	0.06918	0.2329	0.08134	1
541	0.12050	0.3187	0.10230	1
542	0.10950	0.2722	0.06956	1
543	0.07958	0.2473	0.06443	1
544	0.06845	0.2249	0.08492	1
545	0.07174	0.2642	0.06953	1
546	0.02381	0.2681	0.07399	1
547	0.08333	0.2691	0.09479	1
548	0.03846	0.2552	0.07920	1
549	0.03264	0.3059	0.07626	1
550	0.00000	0.2458	0.06592	1
551	0.06413	0.3169	0.08032	1
552	0.06498	0.2407	0.06484	1
553	0.02564	0.2435	0.07393	1
554	0.06493	0.2372	0.07242	1
555	0.09127	0.2226	0.08283	1
556	0.02232	0.2262	0.06742	1
557	0.00000	0.2475	0.06969	1
558	0.11050	0.2258	0.08004	1
559	0.09653	0.2112	0.08732	1
560	0.10480	0.2250	0.08321	1
561	0.00000	0.1566	0.05905	1
562	0.23560	0.4089	0.14090	0
563	0.25420	0.2929	0.09873	0
564	0.22160	0.2060	0.07115	0
565	0.16280	0.2572	0.06637	0
566	0.14180	0.2218	0.07820	0

567	0.26500	0.4087	0.12400	0
568	0.00000	0.2871	0.07039	1

[569 rows x 31 columns]

### 1.0.3 Question 2

What is the class distribution? (i.e. how many instances of malignant (encoded 0) and how many benign (encoded 1)?)

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

```
In [41]: def answer_two():
         cancerdf = answer_one()

         # Your code here
         # 0 is mal and 1 is benine
         mal = len(cancerdf[cancerdf['target'] == 0])
         ben = len(cancerdf[cancerdf['target'] == 1 ])
         s = pd.Series([mal, ben], index = ['malignant', 'benign'])
         return s # Return your answer
```

answer\_two()

```
Out[41]: malignant    212
         benign       357
         dtype: int64
```

### 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, a pandas DataFrame, has shape (569, 30) \* y, a pandas Series, has shape (569,).*

```
In [42]: def answer_three():
         cancerdf = answer_one()

         # Your code here
         X = cancerdf.iloc[:,0:30]
         y = cancerdf.iloc[:,30]

         return X, y
         answer_three()
```

```
Out[42]: (    mean radius  mean texture  mean perimeter  mean area  mean smoothnes
         0         17.990         10.38             122.80      1001.0         0.1184
         1         20.570         17.77             132.90      1326.0         0.0847
         2         19.690         21.25             130.00      1203.0         0.1096)
```

3	11.420	20.38	77.58	386.1	0.1425
4	20.290	14.34	135.10	1297.0	0.1003
5	12.450	15.70	82.57	477.1	0.1278
6	18.250	19.98	119.60	1040.0	0.0946
7	13.710	20.83	90.20	577.9	0.1189
8	13.000	21.82	87.50	519.8	0.1273
9	12.460	24.04	83.97	475.9	0.1186
10	16.020	23.24	102.70	797.8	0.0820
11	15.780	17.89	103.60	781.0	0.0971
12	19.170	24.80	132.40	1123.0	0.0974
13	15.850	23.95	103.70	782.7	0.0840
14	13.730	22.61	93.60	578.3	0.1131
15	14.540	27.54	96.73	658.8	0.1139
16	14.680	20.13	94.74	684.5	0.0986
17	16.130	20.68	108.10	798.8	0.1170
18	19.810	22.15	130.00	1260.0	0.0983
19	13.540	14.36	87.46	566.3	0.0977
20	13.080	15.71	85.63	520.0	0.1075
21	9.504	12.44	60.34	273.9	0.1024
22	15.340	14.26	102.50	704.4	0.1073
23	21.160	23.04	137.20	1404.0	0.0942
24	16.650	21.38	110.00	904.6	0.1121
25	17.140	16.40	116.00	912.7	0.1186
26	14.580	21.53	97.41	644.8	0.1054
27	18.610	20.25	122.10	1094.0	0.0944
28	15.300	25.27	102.40	732.4	0.1082
29	17.570	15.05	115.00	955.1	0.0984
..	...	...	...	...	...
539	7.691	25.44	48.34	170.4	0.0866
540	11.540	14.44	74.65	402.9	0.0998
541	14.470	24.99	95.81	656.4	0.0883
542	14.740	25.42	94.70	668.6	0.0827
543	13.210	28.06	84.88	538.4	0.0867
544	13.870	20.70	89.77	584.8	0.0957
545	13.620	23.23	87.19	573.2	0.0924
546	10.320	16.35	65.31	324.9	0.0943
547	10.260	16.58	65.85	320.8	0.0887
548	9.683	19.34	61.05	285.7	0.0849
549	10.820	24.21	68.89	361.6	0.0819
550	10.860	21.48	68.51	360.5	0.0743
551	11.130	22.44	71.49	378.4	0.0956
552	12.770	29.43	81.35	507.9	0.0827
553	9.333	21.94	59.01	264.0	0.0924
554	12.880	28.92	82.50	514.3	0.0812
555	10.290	27.61	65.67	321.4	0.0903
556	10.160	19.59	64.73	311.7	0.1003
557	9.423	27.88	59.26	271.3	0.0812
558	14.590	22.68	96.39	657.1	0.0847

559	11.510	23.93	74.52	403.5	0.0920
560	14.050	27.15	91.38	600.4	0.0992
561	11.200	29.37	70.67	386.0	0.0744
562	15.220	30.62	103.40	716.9	0.1048
563	20.920	25.09	143.00	1347.0	0.1099
564	21.560	22.39	142.00	1479.0	0.1110
565	20.130	28.25	131.20	1261.0	0.0978
566	16.600	28.08	108.30	858.1	0.0845
567	20.600	29.33	140.10	1265.0	0.1178
568	7.760	24.54	47.92	181.0	0.0520

	mean compactness	mean concavity	mean concave points	mean symmetry
0	0.27760	0.300100	0.147100	0.2419
1	0.07864	0.086900	0.070170	0.1812
2	0.15990	0.197400	0.127900	0.2069
3	0.28390	0.241400	0.105200	0.2597
4	0.13280	0.198000	0.104300	0.1809
5	0.17000	0.157800	0.080890	0.2087
6	0.10900	0.112700	0.074000	0.1794
7	0.16450	0.093660	0.059850	0.2190
8	0.19320	0.185900	0.093530	0.2350
9	0.23960	0.227300	0.085430	0.2030
10	0.06669	0.032990	0.033230	0.1528
11	0.12920	0.099540	0.066060	0.1842
12	0.24580	0.206500	0.111800	0.2397
13	0.10020	0.099380	0.053640	0.1847
14	0.22930	0.212800	0.080250	0.2069
15	0.15950	0.163900	0.073640	0.2303
16	0.07200	0.073950	0.052590	0.1580
17	0.20220	0.172200	0.102800	0.2164
18	0.10270	0.147900	0.094980	0.1582
19	0.08129	0.066640	0.047810	0.1885
20	0.12700	0.045680	0.031100	0.1967
21	0.06492	0.029560	0.020760	0.1815
22	0.21350	0.207700	0.097560	0.2521
23	0.10220	0.109700	0.086320	0.1769
24	0.14570	0.152500	0.091700	0.1995
25	0.22760	0.222900	0.140100	0.3040
26	0.18680	0.142500	0.087830	0.2252
27	0.10660	0.149000	0.077310	0.1697
28	0.16970	0.168300	0.087510	0.1920
29	0.11570	0.098750	0.079530	0.1739
..	...	...	...	...
539	0.11990	0.092520	0.013640	0.2037
540	0.11200	0.067370	0.025940	0.1818
541	0.12300	0.100900	0.038900	0.1872
542	0.07214	0.041050	0.030270	0.1840
543	0.06877	0.029870	0.032750	0.1628

544	0.10180	0.036880	0.023690	0.1620
545	0.06747	0.029740	0.024430	0.1664
546	0.04994	0.010120	0.005495	0.1885
547	0.08066	0.043580	0.024380	0.1669
548	0.05030	0.023370	0.009615	0.1580
549	0.06602	0.015480	0.008160	0.1970
550	0.04227	0.000000	0.000000	0.1661
551	0.08194	0.048240	0.022570	0.2030
552	0.04234	0.019970	0.014990	0.1539
553	0.05605	0.039960	0.012820	0.1692
554	0.05824	0.061950	0.023430	0.1566
555	0.07658	0.059990	0.027380	0.1593
556	0.07504	0.005025	0.011160	0.1791
557	0.04971	0.000000	0.000000	0.1742
558	0.13300	0.102900	0.037360	0.1454
559	0.10210	0.111200	0.041050	0.1388
560	0.11260	0.044620	0.043040	0.1537
561	0.03558	0.000000	0.000000	0.1060
562	0.20870	0.255000	0.094290	0.2128
563	0.22360	0.317400	0.147400	0.2149
564	0.11590	0.243900	0.138900	0.1720
565	0.10340	0.144000	0.097910	0.1752
566	0.10230	0.092510	0.053020	0.1590
567	0.27700	0.351400	0.152000	0.2397
568	0.04362	0.000000	0.000000	0.1587

	mean fractal dimension	...	worst radius \
0	0.07871	...	25.380
1	0.05667	...	24.990
2	0.05999	...	23.570
3	0.09744	...	14.910
4	0.05883	...	22.540
5	0.07613	...	15.470
6	0.05742	...	22.880
7	0.07451	...	17.060
8	0.07389	...	15.490
9	0.08243	...	15.090
10	0.05697	...	19.190
11	0.06082	...	20.420
12	0.07800	...	20.960
13	0.05338	...	16.840
14	0.07682	...	15.030
15	0.07077	...	17.460
16	0.05922	...	19.070
17	0.07356	...	20.960
18	0.05395	...	27.320
19	0.05766	...	15.110
20	0.06811	...	14.500



21	0.06905	...	10.230
22	0.07032	...	18.070
23	0.05278	...	29.170
24	0.06330	...	26.460
25	0.07413	...	22.250
26	0.06924	...	17.620
27	0.05699	...	21.310
28	0.06540	...	20.270
29	0.06149	...	20.010
..	...	...	...
539	0.07751	...	8.678
540	0.06782	...	12.260
541	0.06341	...	16.220
542	0.05680	...	16.510
543	0.05781	...	14.370
544	0.06688	...	15.050
545	0.05801	...	15.350
546	0.06201	...	11.250
547	0.06714	...	10.830
548	0.06235	...	10.930
549	0.06328	...	13.030
550	0.05948	...	11.660
551	0.06552	...	12.020
552	0.05637	...	13.870
553	0.06576	...	9.845
554	0.05708	...	13.890
555	0.06127	...	10.840
556	0.06331	...	10.650
557	0.06059	...	10.490
558	0.06147	...	15.480
559	0.06570	...	12.480
560	0.06171	...	15.300
561	0.05502	...	11.920
562	0.07152	...	17.520
563	0.06879	...	24.290
564	0.05623	...	25.450
565	0.05533	...	23.690
566	0.05648	...	18.980
567	0.07016	...	25.740
568	0.05884	...	9.456

	worst texture	worst perimeter	worst area	worst smoothness \
0	17.33	184.60	2019.0	0.16220
1	23.41	158.80	1956.0	0.12380
2	25.53	152.50	1709.0	0.14440
3	26.50	98.87	567.7	0.20980
4	16.67	152.20	1575.0	0.13740
5	23.75	103.40	741.6	0.17910

6	27.66	153.20	1606.0	0.14420
7	28.14	110.60	897.0	0.16540
8	30.73	106.20	739.3	0.17030
9	40.68	97.65	711.4	0.18530
10	33.88	123.80	1150.0	0.11810
11	27.28	136.50	1299.0	0.13960
12	29.94	151.70	1332.0	0.10370
13	27.66	112.00	876.5	0.11310
14	32.01	108.80	697.7	0.16510
15	37.13	124.10	943.2	0.16780
16	30.88	123.40	1138.0	0.14640
17	31.48	136.80	1315.0	0.17890
18	30.88	186.80	2398.0	0.15120
19	19.26	99.70	711.2	0.14400
20	20.49	96.09	630.5	0.13120
21	15.66	65.13	314.9	0.13240
22	19.08	125.10	980.9	0.13900
23	35.59	188.00	2615.0	0.14010
24	31.56	177.00	2215.0	0.18050
25	21.40	152.40	1461.0	0.15450
26	33.21	122.40	896.9	0.15250
27	27.26	139.90	1403.0	0.13380
28	36.71	149.30	1269.0	0.16410
29	19.52	134.90	1227.0	0.12550
..	...	...	...	...
539	31.89	54.49	223.6	0.15960
540	19.68	78.78	457.8	0.13450
541	31.73	113.50	808.9	0.13400
542	32.29	107.40	826.4	0.10600
543	37.17	92.48	629.6	0.10720
544	24.75	99.17	688.6	0.12640
545	29.09	97.58	729.8	0.12160
546	21.77	71.12	384.9	0.12850
547	22.04	71.08	357.4	0.14610
548	25.59	69.10	364.2	0.11990
549	31.45	83.90	505.6	0.12040
550	24.77	74.08	412.3	0.10010
551	28.26	77.80	436.6	0.10870
552	36.00	88.10	594.7	0.12340
553	25.05	62.86	295.8	0.11030
554	35.74	88.84	595.7	0.12270
555	34.91	69.57	357.6	0.13840
556	22.88	67.88	347.3	0.12650
557	34.24	66.50	330.6	0.10730
558	27.27	105.90	733.5	0.10260
559	37.16	82.28	474.2	0.12980
560	33.17	100.20	706.7	0.12410
561	38.30	75.19	439.6	0.09267

562	42.79	128.70	915.0	0.14170
563	29.41	179.10	1819.0	0.14070
564	26.40	166.10	2027.0	0.14100
565	38.25	155.00	1731.0	0.11660
566	34.12	126.70	1124.0	0.11390
567	39.42	184.60	1821.0	0.16500
568	30.37	59.16	268.6	0.08996

	worst compactness	worst concavity	worst concave points	worst symm
0	0.66560	0.71190	0.26540	0.0
1	0.18660	0.24160	0.18600	0.0
2	0.42450	0.45040	0.24300	0.0
3	0.86630	0.68690	0.25750	0.0
4	0.20500	0.40000	0.16250	0.0
5	0.52490	0.53550	0.17410	0.0
6	0.25760	0.37840	0.19320	0.0
7	0.36820	0.26780	0.15560	0.0
8	0.54010	0.53900	0.20600	0.0
9	1.05800	1.10500	0.22100	0.0
10	0.15510	0.14590	0.09975	0.0
11	0.56090	0.39650	0.18100	0.0
12	0.39030	0.36390	0.17670	0.0
13	0.19240	0.23220	0.11190	0.0
14	0.77250	0.69430	0.22080	0.0
15	0.65770	0.70260	0.17120	0.0
16	0.18710	0.29140	0.16090	0.0
17	0.42330	0.47840	0.20730	0.0
18	0.31500	0.53720	0.23880	0.0
19	0.17730	0.23900	0.12880	0.0
20	0.27760	0.18900	0.07283	0.0
21	0.11480	0.08867	0.06227	0.0
22	0.59540	0.63050	0.23930	0.0
23	0.26000	0.31550	0.20090	0.0
24	0.35780	0.46950	0.20950	0.0
25	0.39490	0.38530	0.25500	0.0
26	0.66430	0.55390	0.27010	0.0
27	0.21170	0.34460	0.14900	0.0
28	0.61100	0.63350	0.20240	0.0
29	0.28120	0.24890	0.14560	0.0
...	...	...	...	...
539	0.30640	0.33930	0.05000	0.0
540	0.21180	0.17970	0.06918	0.0
541	0.42020	0.40400	0.12050	0.0
542	0.13760	0.16110	0.10950	0.0
543	0.13810	0.10620	0.07958	0.0
544	0.20370	0.13770	0.06845	0.0
545	0.15170	0.10490	0.07174	0.0
546	0.08842	0.04384	0.02381	0.0

547	0.22460	0.17830	0.08333	0.
548	0.09546	0.09350	0.03846	0.
549	0.16330	0.06194	0.03264	0.
550	0.07348	0.00000	0.00000	0.
551	0.17820	0.15640	0.06413	0.
552	0.10640	0.08653	0.06498	0.
553	0.08298	0.07993	0.02564	0.
554	0.16200	0.24390	0.06493	0.
555	0.17100	0.20000	0.09127	0.
556	0.12000	0.01005	0.02232	0.
557	0.07158	0.00000	0.00000	0.
558	0.31710	0.36620	0.11050	0.
559	0.25170	0.36300	0.09653	0.
560	0.22640	0.13260	0.10480	0.
561	0.05494	0.00000	0.00000	0.
562	0.79170	1.17000	0.23560	0.
563	0.41860	0.65990	0.25420	0.
564	0.21130	0.41070	0.22160	0.
565	0.19220	0.32150	0.16280	0.
566	0.30940	0.34030	0.14180	0.
567	0.86810	0.93870	0.26500	0.
568	0.06444	0.00000	0.00000	0.

	worst fractal dimension
0	0.11890
1	0.08902
2	0.08758
3	0.17300
4	0.07678
5	0.12440
6	0.08368
7	0.11510
8	0.10720
9	0.20750
10	0.08452
11	0.10480
12	0.10230
13	0.06287
14	0.14310
15	0.13410
16	0.08216
17	0.11420
18	0.07615
19	0.07259
20	0.08183
21	0.07773
22	0.09946
23	0.07526

24	0.09564
25	0.10590
26	0.12750
27	0.07421
28	0.09876
29	0.07919
..	...
539	0.10660
540	0.08134
541	0.10230
542	0.06956
543	0.06443
544	0.08492
545	0.06953
546	0.07399
547	0.09479
548	0.07920
549	0.07626
550	0.06592
551	0.08032
552	0.06484
553	0.07393
554	0.07242
555	0.08283
556	0.06742
557	0.06969
558	0.08004
559	0.08732
560	0.08321
561	0.05905
562	0.14090
563	0.09873
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
5	0
6	0
7	0
8	0
9	0

10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	1
20	1
21	1
22	0
23	0
24	0
25	0
26	0
27	0
28	0
29	0
	..
539	1
540	1
541	1
542	1
543	1
544	1
545	1
546	1
547	1
548	1
549	1
550	1
551	1
552	1
553	1
554	1
555	1
556	1
557	1
558	1
559	1
560	1
561	1
562	0
563	0
564	0
565	0

```

566     0
567     0
568     1
Name: target, dtype: int64)

```

### 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,)*

```
In [43]: from sklearn.model_selection import train_test_split
```

```

def answer_four():
    X, y = answer_three()
    # Line from notebook should work
    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
    # Your code here

    return X_train, X_test, y_train, y_test

```

### 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`.*

```
In [44]: from sklearn.neighbors import KNeighborsClassifier
```

```

def answer_five():
    X_train, X_test, y_train, y_test = answer_four()

    # Your code here
    # we use the high level functions
    #from sklearn.neighbors import KNeighborsClassifier
    knn = KNeighborsClassifier(n_neighbors = 1)

    #fit it
    knn.fit(X_train, y_train)

    return knn # Return your answer

```

### 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 predict method of `KNeighborsClassifier`).

*This function should return a numpy array either `array([ 0.])` or `array([ 1.])`*

```
In [45]: # this is a prediction, where the data set fictitious, say the mean of all
```

```
def answer_six():
    cancerdf = answer_one()

    # get the mean of each col and reshape as given
    means = cancerdf.mean()[:-1].values.reshape(1, -1)

    # Your code here
    # pass it to knn to predict
    knn = answer_five()

    pred = knn.predict(means)

    return pred # Return your answer

answer_six()

# answer was array([1]) - seems benign
```

```
Out[45]: array([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`.*

```
In [46]: def answer_seven():
    X_train, X_test, y_train, y_test = answer_four()
    knn = answer_five()

    # Your code here
    #this would be simple and run the prediction on the test array X-test
    pred = knn.predict(X_test)

    return pred # Return your answer

answer_seven()
```

```
Out[46]: 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, 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, 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, 0, 1, 1, 1, 1, 1, 1, 0
0, 1, 1, 1, 0])

```

### 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*

```

In [47]: def answer_eight():
        X_train, X_test, y_train, y_test = answer_four()
        knn = answer_five()

        # Your code here
        # this is to check the score
        score = knn.score(X_test, y_test)

        return score # Return your answer

answer_eight()
# 0.91608391608391604 was the answer - good enough

```

Out[47]: 0.91608391608391604

### 1.0.10 Optional plot

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

```

In [48]: def accuracy_plot():
        import matplotlib.pyplot as plt

        %matplotlib notebook

        X_train, X_test, y_train, y_test = answer_four()

        # Find the training and testing accuracies by target value (i.e. malignant/benign)
        mal_train_X = X_train[y_train==0]
        mal_train_y = y_train[y_train==0]
        ben_train_X = X_train[y_train==1]
        ben_train_y = y_train[y_train==1]

        mal_test_X = X_test[y_test==0]
        mal_test_y = y_test[y_test==0]
        ben_test_X = X_test[y_test==1]
        ben_test_y = y_test[y_test==1]

        knn = answer_five()

```

```

scores = [knn.score(mal_train_X, mal_train_y), knn.score(ben_train_X,
                knn.score(mal_test_X, mal_test_y), knn.score(ben_test_X, ben

plt.figure()

# Plot the scores as a bar chart
bars = plt.bar(np.arange(4), scores, color=['#4c72b0', '#4c72b0', '#55a8

# directly label the score onto the bars
for bar in bars:
    height = bar.get_height()
    plt.gca().text(bar.get_x() + bar.get_width()/2, height*.90, '{0:.1f}'
                    ha='center', color='w', fontsize=11)

# remove all the ticks (both axes), and tick labels on the Y axis
plt.tick_params(top='off', bottom='off', left='off', right='off', label

# remove the frame of the chart
for spine in plt.gca().spines.values():
    spine.set_visible(False)

plt.xticks([0,1,2,3], ['Malignant\nTraining', 'Benign\nTraining', 'Mal
plt.title('Training and Test Accuracies for Malignant and Benign Cells

```

Uncomment the plotting function to see the visualization.

**Comment out** the plotting function when submitting your notebook for grading.

```
In [49]: accuracy_plot()
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
In [ ]:
```