Assignment 1

January 30, 2019

---

*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 [1]: 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
```

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

```
In [2]: cancer.keys()

Out[2]: 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 [3]: # You should write your whole answer within the function provided. The auto
        # this function and compare the return value against the correct solution w
        def answer_zero():
            # This function returns the number of features of the breast cancer dat
            # 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
    # about the assignment formats, check out the discussion forums for any FA
    answer_zero()
```

Out[3]: 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 [4]: def answer_one():
            """
            This function creates a pandas dataframe from a numpy array

            np.c_ is the numpy concatenate function
            which is used to concat cancer['data'] and cancer['target'] arrays
            for pandas column argument: concat cancer['feature_names'] list
            and string list (in this case one string)

            """
            data = np.c_[cancer.data, cancer.target]
            columns = np.append(cancer.feature_names, ['target'])
            df = pd.DataFrame(data = data, columns = columns)
            return df

        answer_one().head()
```

Out[4]:
```
    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

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   mean fractal dimension   ...    worst texture  worst perimeter  worst ar
0                 0.07871   ...            17.33           184.60      2019
1                 0.05667   ...            23.41           158.80      1956
2                 0.05999   ...            25.53           152.50      1709
3                 0.09744   ...            26.50            98.87       567
4                 0.05883   ...            16.67           152.20      1575

   worst smoothness  worst compactness  worst concavity  worst concave poin
0            0.1622             0.6656           0.7119                0.26
1            0.1238             0.1866           0.2416                0.18
2            0.1444             0.4245           0.4504                0.24
3            0.2098             0.8663           0.6869                0.25
4            0.1374             0.2050           0.4000                0.16

   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890     0.0
1          0.2750                  0.08902     0.0
2          0.3613                  0.08758     0.0
3          0.6638                  0.17300     0.0
4          0.2364                  0.07678     0.0

[5 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 [5]:
```
def answer_two():
    """
    This function determine the class distribution by categorizing 'target
    and benign (1).  Then a series is created by zipping the size of each c
```

```
    """

    cancerdf = answer_one()

    # Count malignants and benign as class distribution
    malignant = np.where(cancerdf['target']==0)
    benign = np.where(cancerdf['target']==1)

    # Group both as list.  This list will be used to create the series
    binary_list = [np.size(malignant), np.size(benign)]

    # Index the series by index = ['malignant', 'benign']
    index = ['malignant', 'benign']

    # Create series
    srs = pd.Series(binary_list, index =  index)
    return srs


answer_two()
```

```
Out[5]: 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 [6]: def answer_three():
            """
            This function splits the initial dataframe into dependent and independe
            Resulting X variable contains all the independent variables, while y is
            explained variable 'target'

            """



            cancerdf = answer_one()

            X = cancerdf.drop('target', axis = 1)
            y = cancerdf['target']
            return X, y

        answer_three()
```

```
Out[6]: (     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          0.08206
        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 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 symme
0              0.66560          0.71190               0.26540          0.4
1              0.18660          0.24160               0.18600          0.2
2              0.42450          0.45040               0.24300          0.3
3              0.86630          0.68690               0.25750          0.6
4              0.20500          0.40000               0.16250          0.2
5              0.52490          0.53550               0.17410          0.3
6              0.25760          0.37840               0.19320          0.3
7              0.36820          0.26780               0.15560          0.3
8              0.54010          0.53900               0.20600          0.4
9              1.05800          1.10500               0.22100          0.4
10             0.15510          0.14590               0.09975          0.2
11             0.56090          0.39650               0.18100          0.3
12             0.39030          0.36390               0.17670          0.3
13             0.19240          0.23220               0.11190          0.2
14             0.77250          0.69430               0.22080          0.3
15             0.65770          0.70260               0.17120          0.4
16             0.18710          0.29140               0.16090          0.3
17             0.42330          0.47840               0.20730          0.3
18             0.31500          0.53720               0.23880          0.2
19             0.17730          0.23900               0.12880          0.2
20             0.27760          0.18900               0.07283          0.3
21             0.11480          0.08867               0.06227          0.2
22             0.59540          0.63050               0.23930          0.4
23             0.26000          0.31550               0.20090          0.2
24             0.35780          0.46950               0.20950          0.3
25             0.39490          0.38530               0.25500          0.4
26             0.66430          0.55390               0.27010          0.4
27             0.21170          0.34460               0.14900          0.2
28             0.61100          0.63350               0.20240          0.4
29             0.28120          0.24890               0.14560          0.2
..                 ...              ...                   ...
539            0.30640          0.33930               0.05000          0.2
540            0.21180          0.17970               0.06918          0.2
541            0.42020          0.40400               0.12050          0.3
542            0.13760          0.16110               0.10950          0.2
```

543	0.13810	0.10620	0.07958	0.2
544	0.20370	0.13770	0.06845	0.2
545	0.15170	0.10490	0.07174	0.2
546	0.08842	0.04384	0.02381	0.2
547	0.22460	0.17830	0.08333	0.2
548	0.09546	0.09350	0.03846	0.2
549	0.16330	0.06194	0.03264	0.3
550	0.07348	0.00000	0.00000	0.2
551	0.17820	0.15640	0.06413	0.3
552	0.10640	0.08653	0.06498	0.2
553	0.08298	0.07993	0.02564	0.2
554	0.16200	0.24390	0.06493	0.2
555	0.17100	0.20000	0.09127	0.2
556	0.12000	0.01005	0.02232	0.2
557	0.07158	0.00000	0.00000	0.2
558	0.31710	0.36620	0.11050	0.2
559	0.25170	0.36300	0.09653	0.2
560	0.22640	0.13260	0.10480	0.2
561	0.05494	0.00000	0.00000	0.1
562	0.79170	1.17000	0.23560	0.4
563	0.41860	0.65990	0.25420	0.2
564	0.21130	0.41070	0.22160	0.2
565	0.19220	0.32150	0.16280	0.2
566	0.30940	0.34030	0.14180	0.2
567	0.86810	0.93870	0.26500	0.4
568	0.06444	0.00000	0.00000	0.2

	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      0.0
1      0.0
2      0.0
3      0.0
4      0.0
5      0.0
```

```
6       0.0
7       0.0
8       0.0
9       0.0
10      0.0
11      0.0
12      0.0
13      0.0
14      0.0
15      0.0
16      0.0
17      0.0
18      0.0
19      1.0
20      1.0
21      1.0
22      0.0
23      0.0
24      0.0
25      0.0
26      0.0
27      0.0
28      0.0
29      0.0
       ...
539     1.0
540     1.0
541     1.0
542     1.0
543     1.0
544     1.0
545     1.0
546     1.0
547     1.0
548     1.0
549     1.0
550     1.0
551     1.0
552     1.0
553     1.0
554     1.0
555     1.0
556     1.0
557     1.0
558     1.0
559     1.0
560     1.0
561     1.0
```

```
562    0.0
563    0.0
564    0.0
565    0.0
566    0.0
567    0.0
568    1.0
Name: target, dtype: float64)
```

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 [7]: from sklearn.model_selection import train_test_split

        def answer_four():
            """
            This function uses question 3's function and then split the data into t
            To costumize the size of each, train_test_split has "train_size" argume
            the proper number of observations/rows depending whether the new data b
            or test data.  The exercise allocates 426 observations in to the traini

            """
            X, y = answer_three()

            X_train, X_test, y_train, y_test = train_test_split(X, y, train_size =

            return X_train, X_test, y_train, y_test

        answer_four()
```

```
Out[7]: (     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
         346       12.060         18.90           76.66      445.3          0.08386
         357       13.870         16.21           88.52      593.7          0.08743
         355       12.560         19.07           81.92      485.8          0.08760
         112       14.260         19.65           97.83      629.9          0.07837
         68         9.029         17.33           58.79      250.5          0.10660
```

526	13.460	18.75	87.44	551.1	0.10750
206	9.876	17.27	62.92	295.4	0.10890
65	14.780	23.94	97.40	668.3	0.11720
437	14.040	15.98	89.78	611.2	0.08458
126	13.610	24.69	87.76	572.6	0.09258
429	12.720	17.67	80.98	501.3	0.07896
392	15.490	19.97	102.40	744.7	0.11600
343	19.680	21.68	129.90	1194.0	0.09797
334	12.300	19.02	77.88	464.4	0.08313
440	10.970	17.20	71.73	371.5	0.08915
441	17.270	25.42	112.40	928.8	0.08331
137	11.430	15.39	73.06	399.8	0.09639
230	17.050	19.08	113.40	895.0	0.11410
7	13.710	20.83	90.20	577.9	0.11890
408	17.990	20.66	117.80	991.7	0.10360
523	13.710	18.68	88.73	571.0	0.09916
361	13.300	21.57	85.24	546.1	0.08582
553	9.333	21.94	59.01	264.0	0.09240
478	11.490	14.59	73.99	404.9	0.10460
303	10.490	18.61	66.86	334.3	0.10680
..	...	...	...	...	...
459	9.755	28.20	61.68	290.9	0.07984
510	11.740	14.69	76.31	426.0	0.08099
151	8.219	20.70	53.27	203.9	0.09405
244	19.400	23.50	129.10	1155.0	0.10270
543	13.210	28.06	84.88	538.4	0.08671
544	13.870	20.70	89.77	584.8	0.09578
265	20.730	31.12	135.70	1419.0	0.09469
288	11.260	19.96	73.72	394.1	0.08020
423	13.660	19.13	89.46	575.3	0.09057
147	14.950	18.77	97.84	689.5	0.08138
177	16.460	20.11	109.30	832.9	0.09831
99	14.420	19.77	94.48	642.5	0.09752
448	14.530	19.34	94.25	659.7	0.08388
431	12.400	17.68	81.47	467.8	0.10540
115	11.930	21.53	76.53	438.6	0.09768
72	17.200	24.52	114.20	929.4	0.10710
537	11.690	24.44	76.37	406.4	0.12360
174	10.660	15.15	67.49	349.6	0.08792
87	19.020	24.59	122.00	1076.0	0.09029
551	11.130	22.44	71.49	378.4	0.09566
486	14.640	16.85	94.21	666.0	0.08641
314	8.597	18.60	54.09	221.2	0.10740
396	13.510	18.89	88.10	558.1	0.10590
472	14.920	14.93	96.45	686.9	0.08098
70	18.940	21.31	123.60	1130.0	0.09009
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	0.1009
192	9.720	18.22	60.73	288.1	0.0695
559	11.510	23.93	74.52	403.5	0.0926

	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
346	0.05794	0.007510	0.008488	0.1555
357	0.05492	0.015020	0.020880	0.1424
355	0.10380	0.103000	0.043910	0.1533
112	0.22330	0.300300	0.077980	0.1704
68	0.14130	0.313000	0.043750	0.2111
526	0.11380	0.042010	0.031520	0.1723
206	0.07232	0.017560	0.019520	0.1934
65	0.14790	0.126700	0.090290	0.1953
437	0.05895	0.035340	0.029440	0.1714
126	0.07862	0.052850	0.030850	0.1761
429	0.04522	0.014020	0.018350	0.1459
392	0.15620	0.189100	0.091130	0.1929
343	0.13390	0.186300	0.110300	0.2082
334	0.04202	0.007756	0.008535	0.1539
440	0.11130	0.094570	0.036130	0.1489
441	0.11090	0.120400	0.057360	0.1467
137	0.06889	0.035030	0.028750	0.1734
230	0.15720	0.191000	0.109000	0.2131
7	0.16450	0.093660	0.059850	0.2196
408	0.13040	0.120100	0.088240	0.1992
523	0.10700	0.053850	0.037830	0.1714
361	0.06373	0.033440	0.024240	0.1815
553	0.05605	0.039960	0.012820	0.1692
478	0.08228	0.053080	0.019690	0.1779
303	0.06678	0.022970	0.017800	0.1482
..	...	...	...	...
459	0.04626	0.015410	0.010430	0.1621
510	0.09661	0.067260	0.026390	0.1499
151	0.13050	0.132100	0.021680	0.2222
244	0.15580	0.204900	0.088860	0.1978
543	0.06877	0.029870	0.032750	0.1628
544	0.10180	0.036880	0.023690	0.1620
265	0.11430	0.136700	0.086460	0.1769
288	0.11810	0.092740	0.055880	0.2595
423	0.11470	0.096570	0.048120	0.1848
147	0.11670	0.090500	0.035620	0.1744
177	0.15560	0.179300	0.088660	0.1794
99	0.11410	0.093880	0.058390	0.1879

448	0.07800	0.088170	0.029250	0.1473
431	0.13160	0.077410	0.027990	0.1811
115	0.07849	0.033280	0.020080	0.1688
72	0.18300	0.169200	0.079440	0.1927
537	0.15520	0.045150	0.045310	0.2131
174	0.04302	0.000000	0.000000	0.1928
87	0.12060	0.146800	0.082710	0.1953
551	0.08194	0.048240	0.022570	0.2030
486	0.06698	0.051920	0.027910	0.1409
314	0.05847	0.000000	0.000000	0.2163
396	0.11470	0.085800	0.053810	0.1806
472	0.08549	0.055390	0.032210	0.1687
70	0.10290	0.108000	0.079510	0.1582
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 \
293	0.05715	...	13.060
332	0.06028	...	11.980
565	0.05533	...	23.690
278	0.05520	...	15.500
489	0.05325	...	19.180
346	0.06048	...	13.640
357	0.05883	...	15.110
355	0.06184	...	13.370
112	0.07769	...	15.300
68	0.08046	...	10.310
526	0.06317	...	15.350
206	0.06285	...	10.420
65	0.06654	...	17.310
437	0.05898	...	15.660
126	0.06130	...	16.890
429	0.05544	...	13.820
392	0.06744	...	21.200
343	0.05715	...	22.750
334	0.05945	...	13.350
440	0.06640	...	12.360
441	0.05407	...	20.380
137	0.05865	...	12.320
230	0.06325	...	19.590
7	0.07451	...	17.060
408	0.06069	...	21.080
523	0.06843	...	15.110
361	0.05696	...	14.200
553	0.06576	...	9.845

```
478               0.06574      ...            12.400
303               0.06600      ...            11.060
..                    ...      ...               ...
459               0.05952      ...            10.670
510               0.06758      ...            12.450
151               0.08261      ...             9.092
244               0.06000      ...            21.650
543               0.05781      ...            14.370
544               0.06688      ...            15.050
265               0.05674      ...            32.490
288               0.06233      ...            11.860
423               0.06181      ...            15.140
147               0.06493      ...            16.250
177               0.06323      ...            17.790
99                0.06390      ...            16.330
448               0.05746      ...            16.300
431               0.07102      ...            12.880
115               0.06194      ...            13.670
72                0.06487      ...            23.320
537               0.07405      ...            12.980
174               0.05975      ...            11.540
87                0.05629      ...            24.560
551               0.06552      ...            12.020
486               0.05355      ...            16.460
314               0.07359      ...             8.952
396               0.06079      ...            14.800
472               0.05669      ...            17.180
70                0.05461      ...            24.860
277               0.04996      ...            19.960
9                 0.08243      ...            15.090
359               0.06959      ...            12.020
192               0.06447      ...             9.968
559               0.06570      ...            12.480

     worst texture  worst perimeter  worst area  worst smoothness  \
293          25.75            84.35       517.8           0.13690
332          25.78            76.91       436.1           0.14240
565          38.25           155.00      1731.0           0.11660
278          26.10            98.91       739.1           0.10500
489          26.56           127.30      1084.0           0.10090
346          27.06            86.54       562.6           0.12890
357          25.58            96.74       694.4           0.11530
355          22.43            89.02       547.4           0.10960
112          23.73           107.00       709.0           0.08949
68           22.65            65.50       324.7           0.14820
526          25.16           101.90       719.8           0.16240
206          23.22            67.08       331.6           0.14150
65           33.39           114.60       925.1           0.16480
```

437	21.58	101.20	750.0	0.11950
126	35.64	113.20	848.7	0.14710
429	20.96	88.87	586.8	0.10680
392	29.41	142.10	1359.0	0.16810
343	34.66	157.60	1540.0	0.12180
334	28.46	84.53	544.3	0.12220
440	26.87	90.14	476.4	0.13910
441	35.46	132.80	1284.0	0.14360
137	22.02	79.93	462.0	0.11900
230	24.89	133.50	1189.0	0.17030
7	28.14	110.60	897.0	0.16540
408	25.41	138.10	1349.0	0.14820
523	25.63	99.43	701.9	0.14250
361	29.20	92.94	621.2	0.11400
553	25.05	62.86	295.8	0.11030
478	21.90	82.04	467.6	0.13520
303	24.54	70.76	375.4	0.14130
..	...	...	...	...
459	36.92	68.03	349.9	0.11100
510	17.60	81.25	473.8	0.10730
151	29.72	58.08	249.8	0.16300
244	30.53	144.90	1417.0	0.14630
543	37.17	92.48	629.6	0.10720
544	24.75	99.17	688.6	0.12640
265	47.16	214.00	3432.0	0.14010
288	22.33	78.27	437.6	0.10280
423	25.50	101.40	708.8	0.11470
147	25.47	107.10	809.7	0.09970
177	28.45	123.50	981.2	0.14150
99	30.86	109.50	826.4	0.14310
448	28.39	108.10	830.5	0.10890
431	22.91	89.61	515.8	0.14500
115	26.15	87.54	583.0	0.15000
72	33.82	151.60	1681.0	0.15850
537	32.19	86.12	487.7	0.17680
174	19.20	73.20	408.3	0.10760
87	30.41	152.90	1623.0	0.12490
551	28.26	77.80	436.6	0.10870
486	25.44	106.00	831.0	0.11420
314	22.44	56.65	240.1	0.13470
396	27.20	97.33	675.2	0.14280
472	18.22	112.00	906.6	0.10650
70	26.58	165.90	1866.0	0.11930
277	24.30	129.00	1236.0	0.12430
9	40.68	97.65	711.4	0.18530
359	25.02	75.79	439.6	0.13330
192	20.83	62.25	303.8	0.07117
559	37.16	82.28	474.2	0.12980

	worst compactness	worst concavity	worst concave points	worst symme
293	0.17580	0.13160	0.09140	0.3
332	0.09669	0.01335	0.02022	0.3
565	0.19220	0.32150	0.16280	0.2
278	0.07622	0.10600	0.05185	0.2
489	0.29200	0.24770	0.08737	0.4
346	0.13520	0.04506	0.05093	0.2
357	0.10080	0.05285	0.05556	0.2
355	0.20020	0.23880	0.09265	0.2
112	0.41930	0.67830	0.15050	0.2
68	0.43650	1.25200	0.17500	0.4
526	0.31240	0.26540	0.14270	0.3
206	0.12470	0.06213	0.05588	0.2
65	0.34160	0.30240	0.16140	0.3
437	0.12520	0.11170	0.07453	0.2
126	0.28840	0.37960	0.13290	0.3
429	0.09605	0.03469	0.03612	0.2
392	0.39130	0.55530	0.21210	0.3
343	0.34580	0.47340	0.22550	0.4
334	0.09052	0.03619	0.03983	0.2
440	0.40820	0.47790	0.15550	0.2
441	0.41220	0.50360	0.17390	0.2
137	0.16480	0.13990	0.08476	0.2
230	0.39340	0.50180	0.25430	0.3
7	0.36820	0.26780	0.15560	0.3
408	0.37350	0.33010	0.19740	0.3
523	0.25660	0.19350	0.12840	0.2
361	0.16670	0.12120	0.05614	0.2
553	0.08298	0.07993	0.02564	0.2
478	0.20100	0.25960	0.07431	0.2
303	0.10440	0.08423	0.06528	0.2
..	...	...	...	
459	0.11090	0.07190	0.04866	0.2
510	0.27930	0.26900	0.10560	0.2
151	0.43100	0.53810	0.07879	0.3
244	0.29680	0.34580	0.15640	0.2
543	0.13810	0.10620	0.07958	0.2
544	0.20370	0.13770	0.06845	0.2
265	0.26440	0.34420	0.16590	0.2
288	0.18430	0.15460	0.09314	0.2
423	0.31670	0.36600	0.14070	0.2
147	0.25210	0.25000	0.08405	0.2
177	0.46670	0.58620	0.20350	0.3
99	0.30260	0.31940	0.15650	0.2
448	0.26490	0.37790	0.09594	0.2
431	0.26290	0.24030	0.07370	0.2
115	0.23990	0.15030	0.07247	0.2

72	0.73940	0.65660	0.18990	0.3
537	0.32510	0.13950	0.13080	0.2
174	0.06791	0.00000	0.00000	0.2
87	0.32060	0.57550	0.19560	0.3
551	0.17820	0.15640	0.06413	0.3
486	0.20700	0.24370	0.07828	0.2
314	0.07767	0.00000	0.00000	0.3
396	0.25700	0.34380	0.14530	0.2
472	0.27910	0.31510	0.11470	0.2
70	0.23360	0.26870	0.17890	0.2
277	0.11600	0.22100	0.12940	0.2
9	1.05800	1.10500	0.22100	0.4
359	0.10490	0.11440	0.05052	0.2
192	0.02729	0.00000	0.00000	0.1
559	0.25170	0.36300	0.09653	0.2

	worst fractal dimension
293	0.07007
332	0.06522
565	0.06637
278	0.06263
489	0.07623
346	0.08083
357	0.07113
355	0.07188
112	0.10820
68	0.11750
526	0.08665
206	0.07380
65	0.08911
437	0.07234
126	0.07900
429	0.06025
392	0.10190
343	0.07918
334	0.07207
440	0.09532
441	0.07944
137	0.06765
230	0.09061
7	0.11510
408	0.08503
523	0.09031
361	0.06658
553	0.07393
478	0.09180
303	0.07842
..	...

```
459                   0.07211
510                   0.09879
151                   0.14860
244                   0.07614
543                   0.06443
544                   0.08492
265                   0.08218
288                   0.07009
423                   0.08839
147                   0.09218
177                   0.09519
99                    0.09353
448                   0.07463
431                   0.09359
115                   0.08541
72                    0.13390
537                   0.09970
174                   0.06164
87                    0.09288
551                   0.08032
486                   0.06596
314                   0.08116
396                   0.07686
472                   0.08273
70                    0.06589
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       13.400         20.52           88.64      556.7          0.11060
457       13.210         25.25           84.10      537.9          0.08791
439       14.020         15.66           89.59      606.5          0.07966
298       14.260         18.17           91.22      633.1          0.06576
37        13.030         18.42           82.61      523.8          0.08983
515       11.340         18.61           72.76      391.2          0.10490
382       12.050         22.72           78.75      447.8          0.06935
310       11.700         19.11           74.33      418.7          0.08814
538        7.729         25.49           47.98      178.8          0.08098
345       10.260         14.71           66.20      321.6          0.09882
421       14.690         13.98           98.22      656.1          0.10310
90        14.620         24.02           94.57      662.7          0.08974
412        9.397         21.68           59.75      268.8          0.07969
157       16.840         19.46          108.40      880.2          0.07445
89        14.640         15.24           95.77      651.9          0.11320
```

172	15.460	11.89	102.50	736.9	0.12570
318	9.042	18.90	60.07	244.5	0.09968
233	20.510	27.81	134.40	1319.0	0.09159
389	19.550	23.21	128.90	1174.0	0.10100
250	20.940	23.56	138.90	1364.0	0.10070
31	11.840	18.70	77.93	440.6	0.11090
283	16.240	18.77	108.80	805.1	0.10660
482	13.470	14.06	87.32	546.3	0.10710
211	11.840	18.94	75.51	428.0	0.08871
372	21.370	15.10	141.30	1386.0	0.10010
401	11.930	10.91	76.14	442.7	0.08872
159	10.900	12.96	68.69	366.8	0.07515
14	13.730	22.61	93.60	578.3	0.11310
364	13.400	16.95	85.48	552.4	0.07937
337	18.770	21.43	122.90	1092.0	0.09116
..	...	...	...	...	...
500	15.040	16.74	98.73	689.4	0.09883
338	10.050	17.53	64.41	310.8	0.10070
427	10.800	21.98	68.79	359.9	0.08801
406	16.140	14.86	104.30	800.0	0.09495
96	12.180	17.84	77.79	451.1	0.10450
490	12.250	22.44	78.18	466.5	0.08192
384	13.280	13.72	85.79	541.8	0.08363
281	11.740	14.02	74.24	427.3	0.07813
325	12.670	17.30	81.25	489.9	0.10280
190	14.220	23.12	94.37	609.9	0.10750
380	11.270	12.96	73.16	386.3	0.12370
366	20.200	26.83	133.70	1234.0	0.09905
469	11.620	18.18	76.38	408.8	0.11750
225	14.340	13.47	92.51	641.2	0.09906
271	11.290	13.04	72.23	388.0	0.09834
547	10.260	16.58	65.85	320.8	0.08877
550	10.860	21.48	68.51	360.5	0.07431
492	18.010	20.56	118.40	1007.0	0.10010
185	10.080	15.11	63.76	317.5	0.09267
306	13.200	15.82	84.07	537.3	0.08511
208	13.110	22.54	87.02	529.4	0.10020
242	11.300	18.19	73.93	389.4	0.09592
313	11.540	10.72	73.73	409.1	0.08597
542	14.740	25.42	94.70	668.6	0.08275
514	15.050	19.07	97.26	701.9	0.09215
236	23.210	26.97	153.50	1670.0	0.09509
113	10.510	20.19	68.64	334.2	0.11220
527	12.340	12.27	78.94	468.5	0.09003
76	13.530	10.94	87.91	559.2	0.12910
162	19.590	18.15	130.70	1214.0	0.11200

mean compactness mean concavity mean concave points mean symmetry

512	0.14690	0.144500	0.081720	0.2116
457	0.05205	0.027720	0.020680	0.1619
439	0.05581	0.020870	0.026520	0.1589
298	0.05220	0.024750	0.013740	0.1635
37	0.03766	0.025620	0.029230	0.1467
515	0.08499	0.043020	0.025940	0.1927
382	0.10730	0.079430	0.029780	0.1203
310	0.05253	0.015830	0.011480	0.1936
538	0.04878	0.000000	0.000000	0.1870
345	0.09159	0.035810	0.020370	0.1633
421	0.18360	0.145000	0.063000	0.2086
90	0.08606	0.031020	0.029570	0.1685
412	0.06053	0.037350	0.005128	0.1274
157	0.07223	0.051500	0.027710	0.1844
89	0.13390	0.099660	0.070640	0.2116
172	0.15550	0.203200	0.109700	0.1966
318	0.19720	0.197500	0.049080	0.2330
233	0.10740	0.155400	0.083400	0.1448
389	0.13180	0.185600	0.102100	0.1989
250	0.16060	0.271200	0.131000	0.2205
31	0.15160	0.121800	0.051820	0.2301
283	0.18020	0.194800	0.090520	0.1876
482	0.11550	0.057860	0.052660	0.1779
211	0.06900	0.026690	0.013930	0.1533
372	0.15150	0.193200	0.125500	0.1973
401	0.05242	0.026060	0.017960	0.1601
159	0.03718	0.003090	0.006588	0.1442
14	0.22930	0.212800	0.080250	0.2069
364	0.05696	0.021810	0.014730	0.1650
337	0.14020	0.106000	0.060900	0.1953
..	...	...	...	...
500	0.13640	0.077210	0.061420	0.1668
338	0.07326	0.025110	0.017750	0.1890
427	0.05743	0.036140	0.014040	0.2016
406	0.08501	0.055000	0.045280	0.1735
96	0.07057	0.024900	0.029410	0.1900
490	0.05200	0.017140	0.012610	0.1544
384	0.08575	0.050770	0.028640	0.1617
281	0.04340	0.022450	0.027630	0.2101
325	0.07664	0.031930	0.021070	0.1707
190	0.24130	0.198100	0.066180	0.2384
380	0.11110	0.079000	0.055500	0.2018
366	0.16690	0.164100	0.126500	0.1875
469	0.14830	0.102000	0.055640	0.1957
225	0.07624	0.057240	0.046030	0.2075
271	0.07608	0.032650	0.027550	0.1769
547	0.08066	0.043580	0.024380	0.1669
550	0.04227	0.000000	0.000000	0.1661

```
492        0.12890       0.117000           0.077620         0.2116
185        0.04695       0.001597           0.002404         0.1703
306        0.05251       0.001461           0.003261         0.1632
208        0.14830       0.087050           0.051020         0.1850
242        0.13250       0.154800           0.028540         0.2054
313        0.05969       0.013670           0.008907         0.1833
542        0.07214       0.041050           0.030270         0.1840
514        0.08597       0.074860           0.043350         0.1561
236        0.16820       0.195000           0.123700         0.1909
113        0.13030       0.064760           0.030680         0.1922
527        0.06307       0.029580           0.026470         0.1689
76         0.10470       0.068770           0.065560         0.2403
162        0.16660       0.250800           0.128600         0.2027

     mean fractal dimension           ...             worst radius  \
512                 0.07325           ...                   16.410
457                 0.05584           ...                   14.350
439                 0.05586           ...                   14.910
298                 0.05586           ...                   16.220
37                  0.05863           ...                   13.300
515                 0.06211           ...                   12.470
382                 0.06659           ...                   12.570
310                 0.06128           ...                   12.610
538                 0.07285           ...                    9.077
345                 0.07005           ...                   10.880
421                 0.07406           ...                   16.460
90                  0.05866           ...                   16.110
412                 0.06724           ...                    9.965
157                 0.05268           ...                   18.220
89                  0.06346           ...                   16.340
172                 0.07069           ...                   18.790
318                 0.08743           ...                   10.060
233                 0.05592           ...                   24.470
389                 0.05884           ...                   20.820
250                 0.05898           ...                   25.580
31                  0.07799           ...                   16.820
283                 0.06684           ...                   18.550
482                 0.06639           ...                   14.830
211                 0.06057           ...                   13.300
372                 0.06183           ...                   22.690
401                 0.05541           ...                   13.800
159                 0.05743           ...                   12.360
14                  0.07682           ...                   15.030
364                 0.05701           ...                   14.730
337                 0.06083           ...                   24.540
..                      ...           ...                      ...
500                 0.06869           ...                   16.760
338                 0.06331           ...                   11.160
```

427	0.05977	...	12.760
406	0.05875	...	17.710
96	0.06635	...	12.830
490	0.05976	...	14.170
384	0.05594	...	14.240
281	0.06113	...	13.310
325	0.05984	...	13.710
190	0.07542	...	15.740
380	0.06914	...	12.840
366	0.06020	...	24.190
469	0.07255	...	13.360
225	0.05448	...	16.770
271	0.06270	...	12.320
547	0.06714	...	10.830
550	0.05948	...	11.660
492	0.06077	...	21.530
185	0.06048	...	11.870
306	0.05894	...	14.410
208	0.07310	...	14.550
242	0.07669	...	12.580
313	0.06100	...	12.340
542	0.05680	...	16.510
514	0.05915	...	17.580
236	0.06309	...	31.010
113	0.07782	...	11.160
527	0.05808	...	13.610
76	0.06641	...	14.080
162	0.06082	...	26.730

	worst texture	worst perimeter	worst area	worst smoothness	\
512	29.66	113.30	844.4	0.15740	
457	34.23	91.29	632.9	0.12890	
439	19.31	96.53	688.9	0.10340	
298	25.26	105.80	819.7	0.09445	
37	22.81	84.46	545.9	0.09701	
515	23.03	79.15	478.6	0.14830	
382	28.71	87.36	488.4	0.08799	
310	26.55	80.92	483.1	0.12230	
538	30.92	57.17	248.0	0.12560	
345	19.48	70.89	357.1	0.13600	
421	18.34	114.10	809.2	0.13120	
90	29.11	102.90	803.7	0.11150	
412	27.99	66.61	301.0	0.10860	
157	28.07	120.30	1032.0	0.08774	
89	18.24	109.40	803.6	0.12770	
172	17.04	125.00	1102.0	0.15310	
318	23.40	68.62	297.1	0.12210	
233	37.38	162.70	1872.0	0.12230	

389	30.44	142.00	1313.0	0.12510
250	27.00	165.30	2010.0	0.12110
31	28.12	119.40	888.7	0.16370
283	25.09	126.90	1031.0	0.13650
482	18.32	94.94	660.2	0.13930
211	24.99	85.22	546.3	0.12800
372	21.84	152.10	1535.0	0.11920
401	20.14	87.64	589.5	0.13740
159	18.20	78.07	470.0	0.11710
14	32.01	108.80	697.7	0.16510
364	21.70	93.76	663.5	0.12130
337	34.37	161.10	1873.0	0.14980
..	...	...	...	...
500	20.43	109.70	856.9	0.11350
338	26.84	71.98	384.0	0.14020
427	32.04	83.69	489.5	0.13030
406	19.58	115.90	947.9	0.12060
96	20.92	82.14	495.2	0.11400
490	31.99	92.74	622.9	0.12560
384	17.37	96.59	623.7	0.11660
281	18.26	84.70	533.7	0.10360
325	21.10	88.70	574.4	0.13840
190	37.18	106.40	762.4	0.15330
380	20.53	84.93	476.1	0.16100
366	33.81	160.00	1671.0	0.12780
469	25.40	88.14	528.1	0.17800
225	16.90	110.40	873.2	0.12970
271	16.18	78.27	457.5	0.13580
547	22.04	71.08	357.4	0.14610
550	24.77	74.08	412.3	0.10010
492	26.06	143.40	1426.0	0.13090
185	21.18	75.39	437.0	0.15210
306	20.45	92.00	636.9	0.11280
208	29.16	99.48	639.3	0.13490
242	27.96	87.16	472.9	0.13470
313	12.87	81.23	467.8	0.10920
542	32.29	107.40	826.4	0.10600
514	28.06	113.80	967.0	0.12460
236	34.51	206.00	2944.0	0.14810
113	22.75	72.62	374.4	0.13000
527	19.27	87.22	564.9	0.12920
76	12.49	91.36	605.5	0.14510
162	26.39	174.90	2232.0	0.14380

	worst compactness	worst concavity	worst concave points	worst symme
512	0.38560	0.51060	0.20510	0.3
457	0.10630	0.13900	0.06005	0.2
439	0.10170	0.06260	0.08216	0.2

298	0.21670	0.15650	0.07530	0.2
37	0.04619	0.04833	0.05013	0.1
515	0.15740	0.16240	0.08542	0.3
382	0.32140	0.29120	0.10920	0.2
310	0.10870	0.07915	0.05741	0.3
538	0.08340	0.00000	0.00000	0.3
345	0.16360	0.07162	0.04074	0.2
421	0.36350	0.32190	0.11080	0.2
90	0.17660	0.09189	0.06946	0.2
412	0.18870	0.18680	0.02564	0.2
157	0.17100	0.18820	0.08436	0.2
89	0.30890	0.26040	0.13970	0.3
172	0.35830	0.58300	0.18270	0.3
318	0.37480	0.46090	0.11450	0.3
233	0.27610	0.41460	0.15630	0.2
389	0.24140	0.38290	0.18250	0.2
250	0.31720	0.69910	0.21050	0.3
31	0.57750	0.69560	0.15460	0.4
283	0.47060	0.50260	0.17320	0.2
482	0.24990	0.18480	0.13350	0.3
211	0.18800	0.14710	0.06913	0.2
372	0.28400	0.40240	0.19660	0.2
401	0.15750	0.15140	0.06876	0.2
159	0.08294	0.01854	0.03953	0.2
14	0.77250	0.69430	0.22080	0.3
364	0.16760	0.13640	0.06987	0.2
337	0.48270	0.46340	0.20480	0.3
..	...	...	...	
500	0.21760	0.18560	0.10180	0.2
338	0.14020	0.10550	0.06499	0.2
427	0.16960	0.19270	0.07485	0.2
406	0.17220	0.23100	0.11290	0.2
96	0.09358	0.04980	0.05882	0.2
490	0.18040	0.12300	0.06335	0.3
384	0.26850	0.28660	0.09173	0.2
281	0.08500	0.06735	0.08290	0.3
325	0.12120	0.10200	0.05602	0.2
190	0.93270	0.84880	0.17720	0.5
380	0.24290	0.22470	0.13180	0.3
366	0.34160	0.37030	0.21520	0.3
469	0.28780	0.31860	0.14160	0.2
225	0.15250	0.16320	0.10870	0.3
271	0.15070	0.12750	0.08750	0.2
547	0.22460	0.17830	0.08333	0.2
550	0.07348	0.00000	0.00000	0.2
492	0.23270	0.25440	0.14890	0.3
185	0.10190	0.00692	0.01042	0.2
306	0.13460	0.01120	0.02500	0.2

208	0.44020	0.31620	0.11260	0.4
242	0.48480	0.74360	0.12180	0.3
313	0.16260	0.08324	0.04715	0.3
542	0.13760	0.16110	0.10950	0.2
514	0.21010	0.28660	0.11200	0.2
236	0.41260	0.58200	0.25930	0.3
113	0.20490	0.12950	0.06136	0.2
527	0.20740	0.17910	0.10700	0.3
76	0.13790	0.08539	0.07407	0.2
162	0.38460	0.68100	0.22470	0.3

	worst fractal dimension
512	0.11090
457	0.06788
439	0.06710
298	0.07676
37	0.06169
515	0.06783
382	0.09349
310	0.06958
538	0.09938
345	0.08488
421	0.09208
90	0.07246
412	0.09206
157	0.05972
89	0.08473
172	0.10100
318	0.10550
233	0.08328
389	0.07602
250	0.07849
31	0.14020
283	0.10630
482	0.09326
211	0.07993
372	0.08666
401	0.07262
159	0.07685
14	0.14310
364	0.07582
337	0.09870
..	...
500	0.08549
338	0.07664
427	0.07662
406	0.07012
96	0.07376

```
490                    0.08203
384                    0.07320
281                    0.06688
325                    0.06888
190                    0.14460
380                    0.09215
366                    0.07632
469                    0.09270
225                    0.06072
271                    0.08022
547                    0.09479
550                    0.06592
492                    0.07625
185                    0.07697
306                    0.08385
208                    0.10760
242                    0.12970
313                    0.07434
542                    0.06956
514                    0.06954
236                    0.08677
113                    0.09026
527                    0.07592
76                     0.07191
162                    0.09223

[143 rows x 30 columns],
293    1.0
332    1.0
565    0.0
278    1.0
489    0.0
346    1.0
357    1.0
355    1.0
112    1.0
68     1.0
526    1.0
206    1.0
65     0.0
437    1.0
126    0.0
429    1.0
392    0.0
343    0.0
334    1.0
440    1.0
441    0.0
```

```
137    1.0
230    0.0
7      0.0
408    0.0
523    1.0
361    1.0
553    1.0
478    1.0
303    1.0
      ...
459    1.0
510    1.0
151    1.0
244    0.0
543    1.0
544    1.0
265    0.0
288    1.0
423    1.0
147    1.0
177    0.0
99     0.0
448    1.0
431    1.0
115    1.0
72     0.0
537    1.0
174    1.0
87     0.0
551    1.0
486    1.0
314    1.0
396    1.0
472    1.0
70     0.0
277    0.0
9      0.0
359    1.0
192    1.0
559    1.0
Name: target, dtype: float64,
512    0.0
457    1.0
439    1.0
298    1.0
37     1.0
515    1.0
382    1.0
```

```
310    1.0
538    1.0
345    1.0
421    1.0
90     1.0
412    1.0
157    1.0
89     1.0
172    0.0
318    1.0
233    0.0
389    0.0
250    0.0
31     0.0
283    0.0
482    1.0
211    1.0
372    0.0
401    1.0
159    1.0
14     0.0
364    1.0
337    0.0
      ...
500    1.0
338    1.0
427    1.0
406    1.0
96     1.0
490    1.0
384    1.0
281    1.0
325    1.0
190    0.0
380    1.0
366    0.0
469    1.0
225    1.0
271    1.0
547    1.0
550    1.0
492    0.0
185    1.0
306    1.0
208    1.0
242    1.0
313    1.0
542    1.0
```

```
514    0.0
236    0.0
113    1.0
527    1.0
76     1.0
162    0.0
Name: target, dtype: float64)
```

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 [8]: from sklearn.neighbors import KNeighborsClassifier

        def answer_five():
            """
            This function creates the model fit based on the prior X_train and y_tr

            """
            X_train, X_test, y_train, y_test = answer_four()

            knn = KNeighborsClassifier(n_neighbors = 1)
            outcome = knn.fit(X_train, y_train)

            return outcome
        answer_five()
```

```
Out[8]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                   metric_params=None, n_jobs=1, n_neighbors=1, p=2,
                   weights='uniform')
```

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).

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

```
In [9]: def answer_six():
            """
            This function takes the original dataframe and considered only the inde
            Then attributions' means are estimated.  The mean values estimated are
            from question 5 ...

                  KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='mi
                  metric_params=None, n_jobs=1, n_neighbors=1, p=2,
```

```
                 weights='uniform'))

            .  In this case the possible outcomes are either 1 (benign) or 0 (malig

            """
            cancerdf = answer_one()
            means = cancerdf.mean()[:-1].values.reshape(1, -1)

            result = answer_five().predict(means)

            return result# Return your answer

        answer_six()
```

```
Out[9]: 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 [10]: def answer_seven():
             """
             This function takes the original dataframe and considered only the ind
             Then attributions' values are used to predict outcomes on each value w
             the testing set.  Then the outcome from question 5 ...

                    KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='mi
                    metric_params=None, n_jobs=1, n_neighbors=1, p=2,
                    weights='uniform'))

             ... isused here to predict row by row.  In this case the possible outc

             """
             X_train, X_test, y_train, y_test = answer_four()
             knn = answer_five()

             result = knn.predict(X_test)

             return result

         answer_seven()
```

```
Out[10]: 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.])
```

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 [11]: def answer_eight():
             """
             This function scores the accuracy of the model in question.  We can do
              1.- By using the score()
              2.- By averaging the outcomes from the model vs the actuals

             """
             X_train, X_test, y_train, y_test = answer_four()
             knn = answer_five()

             outcome = knn.score(X_test, y_test)
             #outcome = np.mean(y_test == answer_seven())

             return outcome# Return your answer
         answer_eight()

Out[11]: 0.916083916083916O4
```

1.0.10 Optional plot

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

```
In [12]: 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. malig
             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, ber


    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:.{
                     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', labe

    # 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 [13]: accuracy_plot()
```

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


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

```
In [ ]:
```