Assignment 1

January 31, 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 [2]: import numpy as np
        import pandas as pd
        from sklearn.datasets import load_breast_cancer

        cancer = load_breast_cancer()
        print(cancer.DESCR)
        #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^2 / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

The mean, standard error, and "worst" or largest (mean of the three 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.
```

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

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

```
Out[3]: 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 [4]: # You should write your whole answer within the function provided. The auto
        # this function and compare the return value against the correct solution v
        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 FAQ
        answer_zero()
```

```
Out[4]: 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 [5]: def answer_one():

            # Your code here

            index=pd.RangeIndex(start=0,stop=569,step=1)
            data=np.column_stack((cancer.data,cancer.target))
            columns=np.append(cancer.feature_names,'target');




            df=pd.DataFrame(data=data,index=index,columns=columns)

            return df.head()


        answer_one()
```

Out[5]:

	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 [11]: def answer_two():
             cancerdf = answer_one()
             index = ['malignant', 'benign']

             # Your code here
             maligant=np.where(cancerdf['target']==0)
             benign=np.where(cancerdf['target']==1)
             data=[np.size(maligant),np.size(benign)]
             print(data)
             h=pd.Series(data,index=index)
             return h


         answer_two()

[5, 0]


Out[11]: malignant    5
         benign       0
         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 [14]: def answer_three():
             cancerdf = answer_one()
             X=cancerdf.drop('target',axis=1)
             y=cancerdf.get('target')
             # x_train,x_test,y_train,y_test=train_test_split(x,y)


             # Your code here

             return X, y
         answer_three()
```

```
Out[14]: (   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 radius  \
          0                 0.07871           ...                    25.38
          1                 0.05667           ...                    24.99
          2                 0.05999           ...                    23.57
          3                 0.09744           ...                    14.91
          4                 0.05883           ...                    22.54

             worst texture  worst perimeter  worst area  worst smoothness  \
          0          17.33           184.60      2019.0            0.1622
          1          23.41           158.80      1956.0            0.1238
          2          25.53           152.50      1709.0            0.1444
          3          26.50            98.87       567.7            0.2098
          4          16.67           152.20      1575.0            0.1374

             worst compactness  worst concavity  worst concave points  worst symmet
          0             0.6656           0.7119                0.2654          0.46
          1             0.1866           0.2416                0.1860          0.27
```

```
2              0.4245           0.4504                0.2430          0.36
3              0.8663           0.6869                0.2575          0.66
4              0.2050           0.4000                0.1625          0.23

   worst fractal dimension
0                  0.11890
1                  0.08902
2                  0.08758
3                  0.17300
4                  0.07678

[5 rows x 30 columns], 0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
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 [17]:
```
from sklearn.model_selection import train_test_split

def answer_four():
    X, y = answer_three()
    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
answer_four()
```

Out[17]:
```
(   mean radius  mean texture  mean perimeter  mean area  mean smoothness
 1        20.57         17.77          132.90     1326.0          0.08474
 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
 1           0.07864          0.0869              0.07017         0.1812
 3           0.28390          0.2414              0.10520         0.2597
 4           0.13280          0.1980              0.10430         0.1809
```

```
   mean fractal dimension           ...             worst radius  \
1                 0.05667           ...                    24.99
3                 0.09744           ...                    14.91
4                 0.05883           ...                    22.54

   worst texture  worst perimeter  worst area  worst smoothness  \
1          23.41           158.80      1956.0            0.1238
3          26.50            98.87       567.7            0.2098
4          16.67           152.20      1575.0            0.1374

   worst compactness  worst concavity  worst concave points  worst symmet
1             0.1866           0.2416                0.1860          0.27
3             0.8663           0.6869                0.2575          0.66
4             0.2050           0.4000                0.1625          0.23

   worst fractal dimension
1                  0.08902
3                  0.17300
4                  0.07678

[3 rows x 30 columns],
   mean radius  mean texture  mean perimeter  mean area  mean smoothness
2        19.69         21.25           130.0     1203.0           0.1096
0        17.99         10.38           122.8     1001.0           0.1184

   mean compactness  mean concavity  mean concave points  mean symmetry
2            0.1599          0.1974               0.1279         0.2069
0            0.2776          0.3001               0.1471         0.2419

   mean fractal dimension           ...             worst radius  \
2                 0.05999           ...                    23.57
0                 0.07871           ...                    25.38

   worst texture  worst perimeter  worst area  worst smoothness  \
2          25.53            152.5      1709.0            0.1444
0          17.33            184.6      2019.0            0.1622

   worst compactness  worst concavity  worst concave points  worst symmet
2             0.4245           0.4504                0.2430          0.36
0             0.6656           0.7119                0.2654          0.46

   worst fractal dimension
2                  0.08758
0                  0.11890

[2 rows x 30 columns],
1    0.0
```

```
3    0.0
4    0.0
Name: target, dtype: float64,
2    0.0
0    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 [23]: from sklearn.neighbors import KNeighborsClassifier

         def answer_five():
             X_train, X_test, y_train, y_test = answer_four()
             knn=KNeighborsClassifier(n_neighbors=1)
             knn.fit(X_train,y_train)
             knn.score(X_test,y_test)

             # Your code here

             return knn
         answer_five()
```

```
Out[23]: 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 [30]: def answer_six():
             cancerdf = answer_one()
             means = cancerdf.mean()[:-1].values.reshape(1, -1)

             knn=answer_five()
             # Your code here

             return knn.predict(means)
         answer_six()
```

```
Out[30]: array([ 0.])
```

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 [33]: def answer_seven():
             X_train, X_test, y_train, y_test = answer_four()
             knn = answer_five()
             fruit_prediction = knn.predict(X_test)


             # Your code here

             return fruit_prediction
         answer_seven()
```

```
Out[33]: array([ 0.,  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 [34]: def answer_eight():
             X_train, X_test, y_train, y_test = answer_four()
             knn = answer_five()

             # Your code here


             return  knn.score(X_test,y_test)
         answer_eight()
```

```
Out[34]: 1.0
```

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 [35]: 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, 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:.{
                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 [ ]:
```
#accuracy_plot()
```

In [ ]: