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(['feature_names', 'target_names', 'DESCR', 'data', 'target'])
```

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 autograder
         # this function and compare the return value against the correct solution value
         def answer_zero():
             # This function returns the number of features
         #      of the breast cancer dataset, which is an integer.
             # The assignment question description will tell
         #      you the general format the autograder is expecting
             return len(cancer['feature_names'])

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

Out[3]: 30

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():

             # Your code here

             return # Return your answer


         answer_one()
```

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():
             cancerdf = answer_one()

             # Your code here

             return # Return your answer


         answer_two()
```

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 [ ]:  def answer_three():
             cancerdf = answer_one()

             # Your code here

             return X, y
```

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 [ ]:  from sklearn.model_selection import train_test_split

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

             # Your code here

             return X_train, X_test, y_train, y_test
```

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 [ ]:  from sklearn.neighbors import KNeighborsClassifier

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

             # Your code here

             return # Return your answer
```

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

             # Your code here

             return # Return your answer
```

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

            # Your code here

            return # Return your answer
```

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

            # Your code here

            return # Return your answer
```

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 [ ]:  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, b
             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, ben_train_y),
                       knn.score(mal_test_X, mal_test_y),
                       knn.score(ben_test_X, ben_test_y)]


             plt.figure()

             # Plot the scores as a bar chart
             bars = plt.bar(np.arange(4), scores, color=['#4c72b0','#4c72b0','#55a868','#5

             # 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:.{1}f}'.format(height, 2),
                               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',
                            labelleft='off', labelbottom='on')

             # 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',
                                   'Malignant\nTest', 'Benign\nTest'], alpha=0.8);
             plt.title('Training and Test Accuracies for Malignant and Benign Cells', alph
```

Uncomment the plotting function to see the visualization.

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

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

```
In [ ]:
```