homework1

September 13, 2019

NAME: Jacob Duvall

1 Homework 1

1.0.1 Objectives

- Basic numpy operations to access data
- Basic plotting of subsets of data
- Simple descriptive statistics
- Do not save work within the ml_practices folder
  - create a folder in your home directory for assignments, and copy the temples there

1.0.2 General References

- Sci-kit Learn Breast Cancer Dataset
- Numpy Reference
- Summary of matplotlib
  - Plot
  - Boxplots
  - Histograms
  - Scatter plots
  - Colormap Plots

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

2 LOAD BREAST CANCER DATA SET

[3]:
```python
"""
Load the dataset into the bc_dataset variable, by calling the
load_breast_cancer() function imported from sklearn.datasets.
Then display the bc_dataset object's list of keys. bc_dataset
is a dictionary object.
"""
```

```
bc_dataset = load_breast_cancer()
```

2.0.1 Dataset Details

The `bc_dataset` variable is a dictionary with multiple fields: * `data` : m by n numpy array of the n observed feature values, for each of the m samples
* `target` : m by 1 numpy array of samples' classification as either malignant (i.e. 0) or benign (i.e. 1)
* `target_names` : 2 by 1 numpy array of the possible tumor classifications
* `DESCR` : string containing a detailed description of the dataset
* `feature_names` : n by 1 numpy array of the names of the feature variables
* `filename` : string containing the absolute path to where the file containing all the data information is located on the local system

[4]:
```
""" TODO
Print out the description of the data, by accessing the
'DESCR' field
"""
print(bc_dataset.DESCR)
```

```
.. _breast_cancer_dataset:

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

**Data Set Characteristics:**

    :Number of Instances: 569

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

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

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

.. topic:: References

   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction
     for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
     Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
     San Jose, CA, 1993.
   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and
     prognosis via linear programming. Operations Research, 43(4), pages
570-577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning
techniques
     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77
(1994)
     163-171.
```

2.1 SETUP USEFUL VARIABLES

[75]:
```
"""
Store the names of the features and the names
of the target classes, into the variables
feature_names and target_names respectively.
"""
feature_names = bc_dataset.feature_names
target_names = bc_dataset.target_names

""" TODO
Print the list of feature names and target names
"""
print(feature_names)
print(target_names)
```

```
['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']
['malignant' 'benign']
```

[74]:
```
"""
Create variables for the feature and target data
The X variable is a numpy array containing the data measured
for each feature for each sample. Each column of X is a
different feature for all the samples. Each row of X is a
different sample with all its features.
The y variable is a numpy array containing the classification
for each sample. A sample tumor is either Benign or Malignant.
"""
X = bc_dataset['data']
y = bc_dataset['target']

""" TODO
Print the dimensions of the X and y variables respectively
"""
print(X.shape)
print(y.shape)
```

```
(569, 30)
(569,)
```

[7]:
```python
"""
Store the number of samples and the number of features, by
accessing the values from the shape of X
"""
nsamples = X.shape[0]
nfeatures = X.shape[1]

""" TODO
Print the print the number of samples and numberof features respectively
"""
print(nsamples)
print(nfeatures)
```

```
569
30
```

2.2 SELECT SUBSET OF FEATURES

Not all available data is necessary or useful for making predictions and classifying observations. There are numerous feature selection algorithms that exist, which will be discussed in more detail later within the cousre. For now we are going to arbitrarly select mean radius, mean area, mean concavity, and mean symmetry as our predictor variables. We will not yet be performing any predictions in this assignment; rather this term is used to conveniently distinuguish this subset of features from the full set of features.

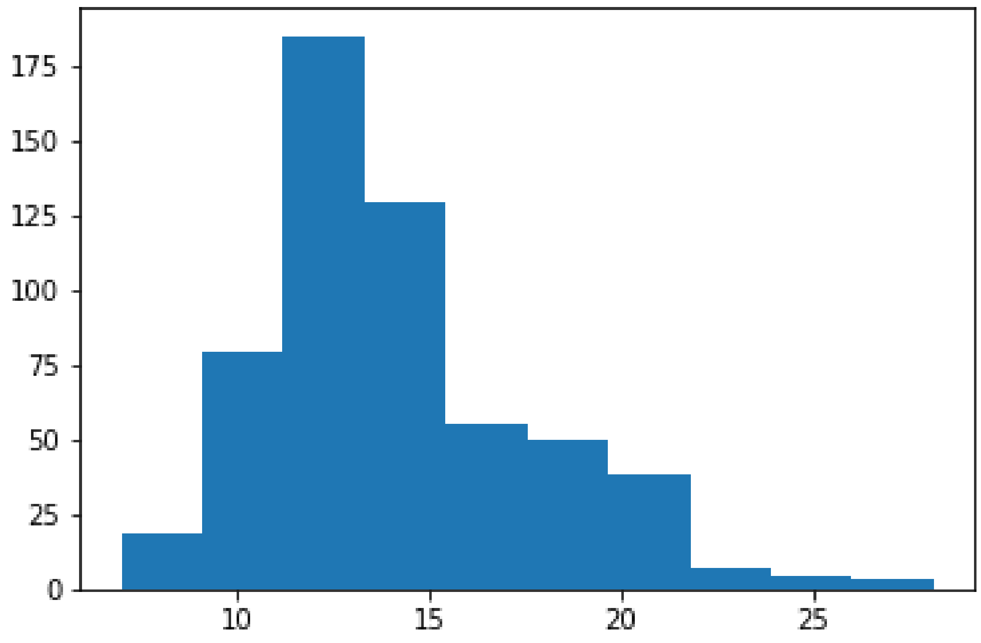
[8]:
```python
"""
Feature Column Indices
The values observed for each feature resides within a particular
column of the feature matrix, X. For example, column 0 contains the
values of the mean radius for each observation, the column at index
3 contains the values for the mean area, and so on.
"""
mean_radius_idx = 0
mean_area_idx = 3
mean_concavity_idx = 6
mean_symmetry_idx = 8

"""
Create a list of the select subset of features
"""
predictors = [mean_radius_idx, mean_area_idx, mean_concavity_idx,␣
 ↪mean_symmetry_idx]

"""
Create a variable, storing the number of predictors
"""
npredictors = len(predictors)
```

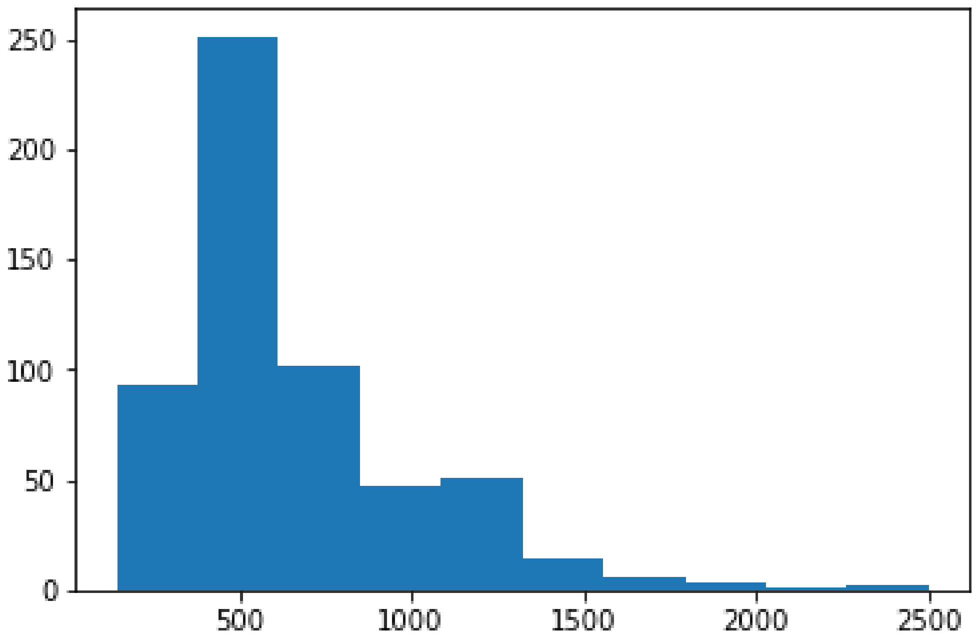
```
"""
Create a list of corresponding names for the selected set of features.
This is conveniently done using list comprehension
"""
pred_names = [bc_dataset['feature_names'][index] for index in predictors]
"""
Print the list of predictor names
"""
print("Predictor Feature Names:")
print(pred_names)
```

```
Predictor Feature Names:
['mean radius', 'mean area', 'mean concavity', 'mean symmetry']
```

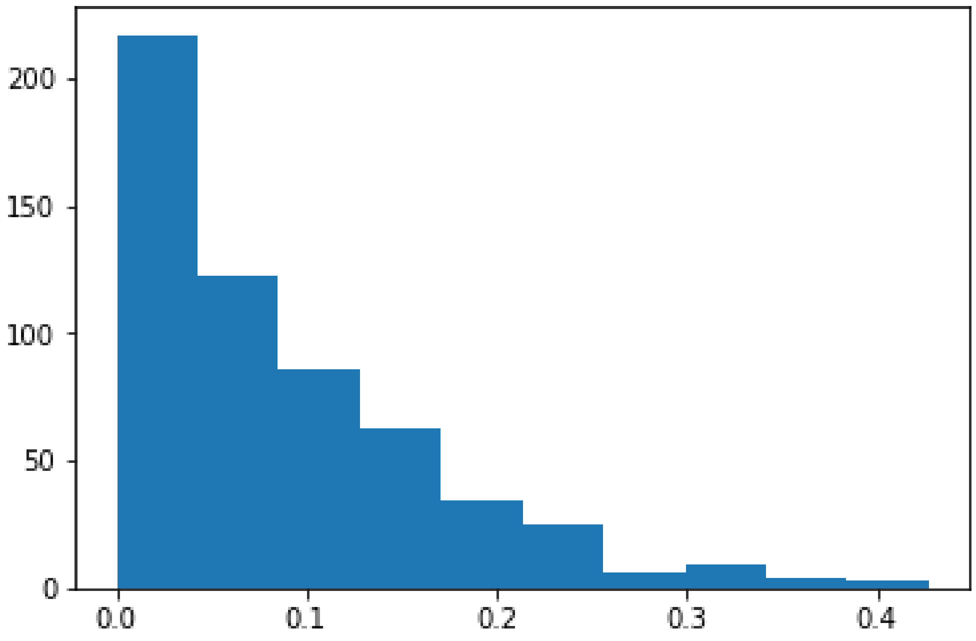
2.3 BASIC HISTOGRAMS OF FEATURES

[69]:
```
""" TODO
HISTOGRAMS OF THE CHOSEN PREDICTOR FEATURES
You may plot the histrograms within their
own figure or within their own subplot of
the same figure
"""
for pred in predictors:
    plt.hist(X[:,pred])
    plt.show()
```

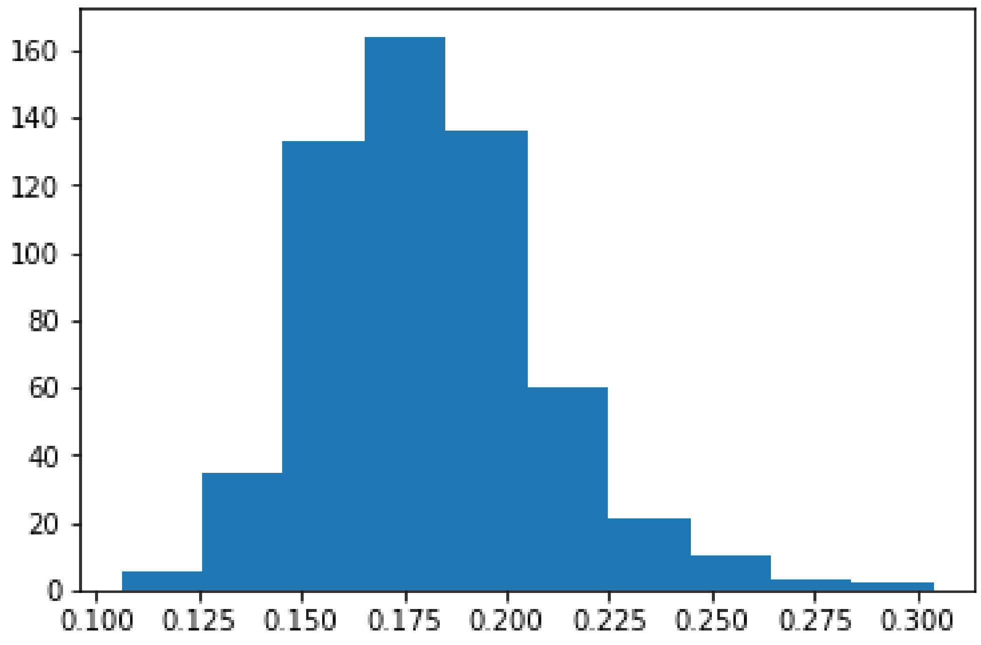




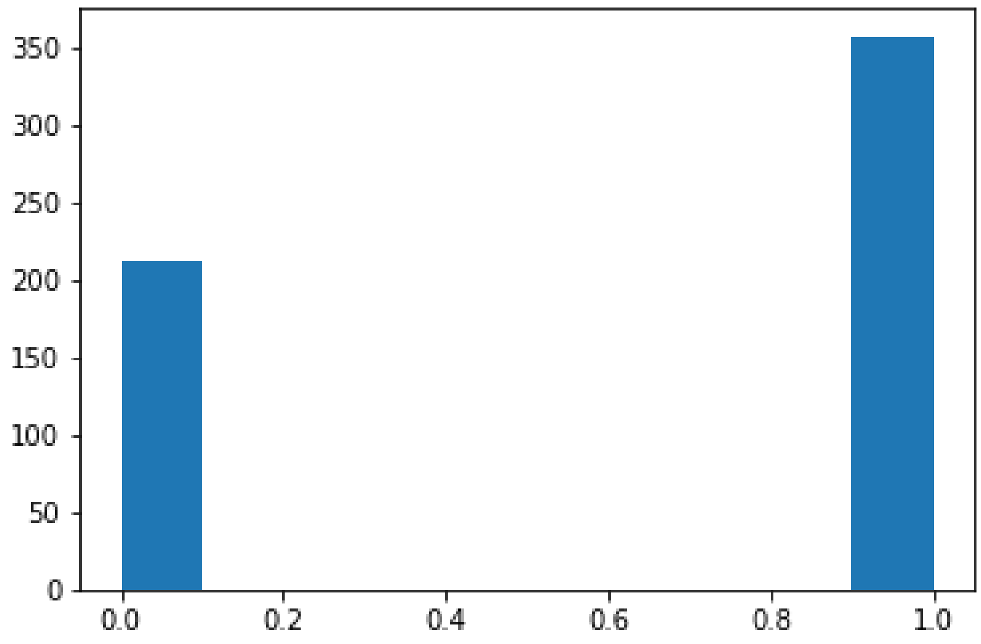
250
200
150
100
50
0
500
1000
1500
2000
2500



200
150
100
50
0
0.0
0.1
0.2
0.3
0.4



[73]:
```
""" TODO
Create a histogram or barplot for the counts
for each target class
"""
plt.hist(y)
plt.show()
```



2.4 BASIC BOXPLOTS OF FEATURES

Boxplots or box-and-whisker plots are used to obtain a perspective of the distribution of the data. The box within the figure displays the 25th percentile (Q1), the median, and the 75th percentile (Q3) of the data. The range between the 75th percentile value and the 25th percentile value is the interquartile range (IQR = Q3 - Q1). The end of bottom line is Q1 - 1.5 * IQR. The end of top line is Q3 + 1.5 * IQR. Anything beyond the lines, the circles, are suggested outliers.

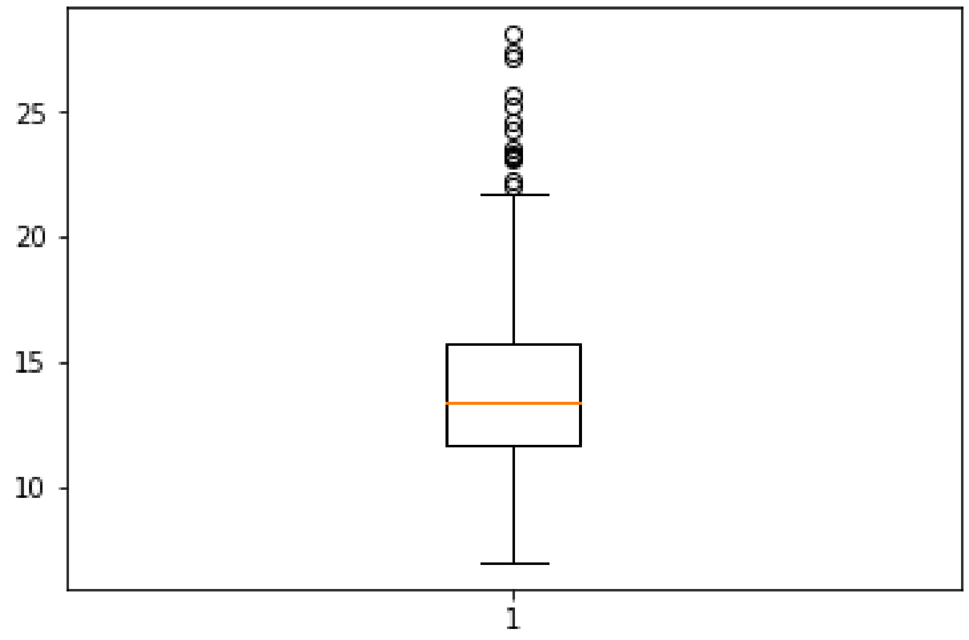
<

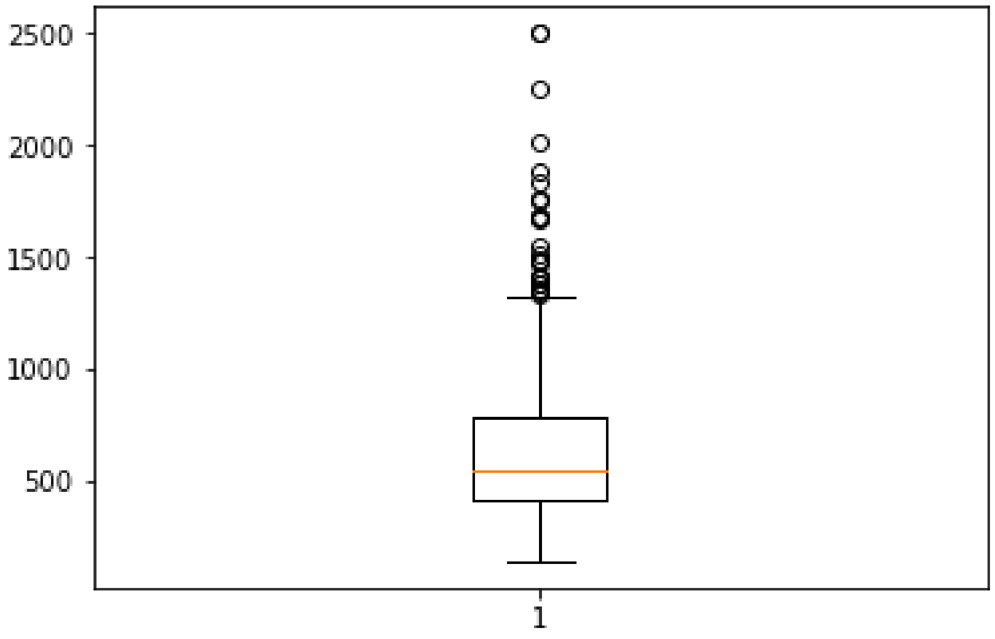
>

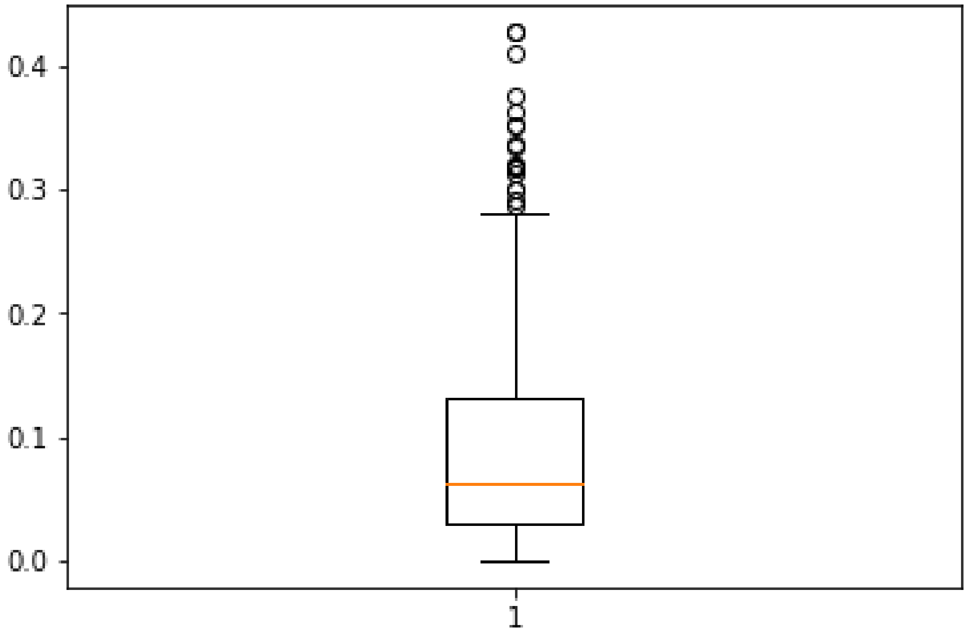
One can use the `boxplot(data_values, labels=[name])` to generate a boxplot. `data_values` would be the set of observed values for a paritucular feature and `labels` should be provided as a list, with the name of the feature, in place of `name`.

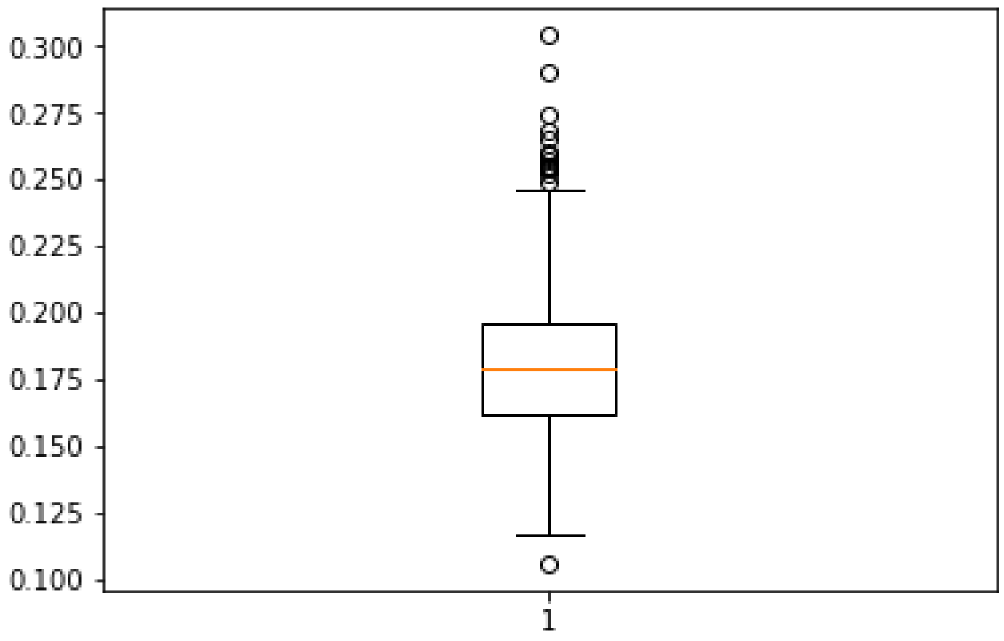
[66]:
```
""" TODO
BOXPLOTS OF THE CHOSEN PREDICTOR FEATURES
You may place the boxplots within their
own figure or within their own subplot of
the same figure
"""
print(len(X[0]))
for pred in predictors:
    plt.boxplot(X[:,pred])
    plt.show()
```

```
30
```




2500
2000
1500
1000
500
1


0.4
0.3
0.2
0.1
0.0
1



2.5 DESCRIPTIVE STATISTICS

[13]:
```
# Simply run this cell
"""
Create a separate variable of the data from the
predictors
"""
Xpreds = X[:, predictors]

"""
Check if any values are NaN (not a number)
"""
np.any(np.isnan(X[:, predictors]))
```

[13]:
```
False
```

[92]:
```
""" TODO
Compute the following descriptive statistics of the
features ignoring NaN values, using numpy:
mean, median, standard deviation, min, and max

Make sure to compute the statistics of the columns
of X (i.e. of each feature). You can specify this
by setting axis=0 for each of the functions

Compute and print the results
"""
```

```
for preds in range(4):
    mean = np.mean(Xpreds[:,preds], axis = 0)
    print("The mean of", pred_names[preds], ":", mean)
    median = np.median(Xpreds[:,preds],  axis = 0)
    print("The median of", pred_names[preds], ":", median)
    standard_deviation = np.std(Xpreds[:,preds], axis = 0)
    print("The standard deviation of", pred_names[preds], ":",␣
 ↪standard_deviation)
    min = np.min(Xpreds[:,preds], axis = 0)
    print("The min of", pred_names[preds], ":", min)
    max = np.max(Xpreds[:,preds], axis = 0)
    print("The max of", pred_names[preds], ":", max)
    print("\n")
```

```
The mean of mean radius : 14.127291739894552
The median of mean radius : 13.37
The standard deviation of mean radius : 3.520950760711062
The min of mean radius : 6.981
The max of mean radius : 28.11


The mean of mean area : 654.8891036906855
The median of mean area : 551.1
The standard deviation of mean area : 351.60475406323
The min of mean area : 143.5
The max of mean area : 2501.0


The mean of mean concavity : 0.0887993158172232
The median of mean concavity : 0.06154
The standard deviation of mean concavity : 0.07964972534603185
The min of mean concavity : 0.0
The max of mean concavity : 0.4268


The mean of mean symmetry : 0.18116186291739894
The median of mean symmetry : 0.1792
The standard deviation of mean symmetry : 0.027390180864268532
The min of mean symmetry : 0.106
The max of mean symmetry : 0.304
```

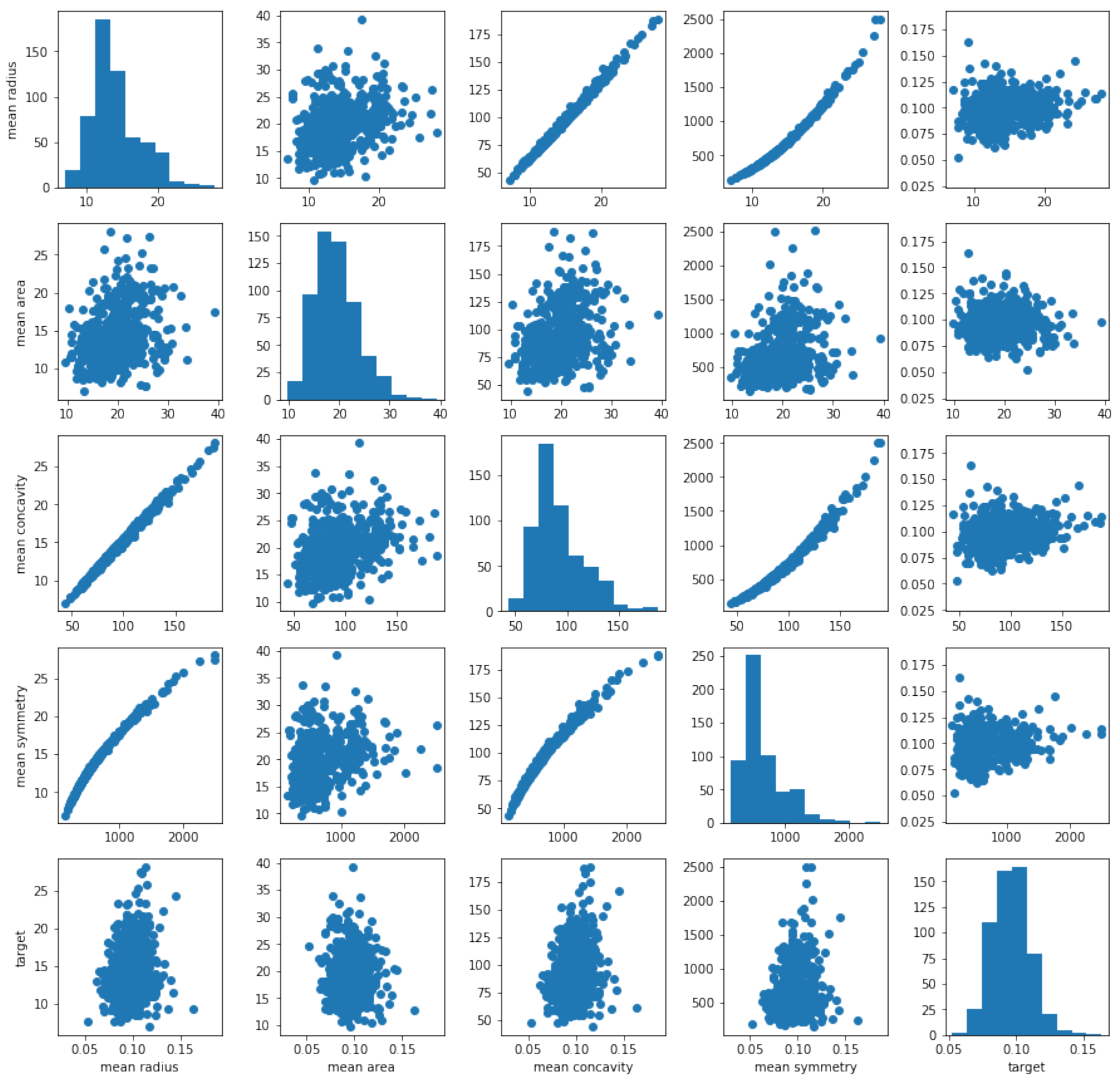
2.6 FEATURE CORRELATIONS

It's useful to know the correlation between various features, as well as each feature and the predicted label. Feature correlation is useful for feature selection and understanding the

relationship between multiple variables within a dataset. Correlation is either positive, negative, or zero. When two features increase simultaneously, they are positively correlated. When one feature increases while the other decreases, the features are negatively correlated. Zero correlation is when there is no relationship between the features. Correlation is on the range -1 (perfect negative correlation) and 1 (pefect positive correlation).
We can construct scatter plots of one feature versuses another to observe linear or nonlinear relationships.
Complete the following set of scatter plots:
<

>

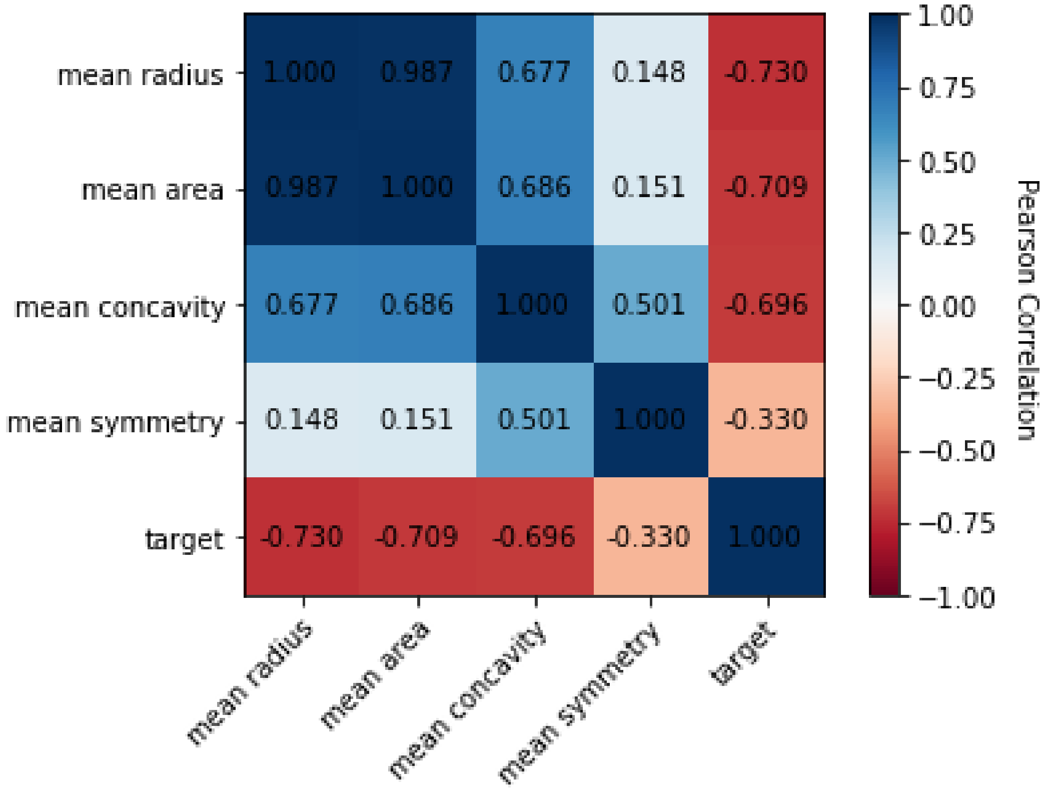
[68]:
```
"""
Using the scatter plot function, construct plots depicting the
correlation between all pairings of the selected predictor features
and between all predictors and the determined target.
The figure will contain r by r subplots, where r = npredictors + 1.
Where subplot(i,j) is a scatter plot of the feature i versus feature j.
When i == j, plot the histogram of feature i instead of a scatter plot.
We are also interested in the correlation between each of the features
and the target classification, thus we will combine the predictors matrix
and the target vector into one large matrix for convenience.
"""
# Append the y to the end of the matrix of predictors
Xycombo = np.append(Xpreds, y.reshape(-1, 1), axis=1)
# Append the name 'target' to the end of the list of predictor names
Xycolnames = pred_names + ['target']

# Create the scatter plots
fig, axs = plt.subplots(npredictors+1, npredictors+1, figsize=(15, 15))
fig.subplots_adjust(wspace=.35)
for f1 in range(npredictors+1):
    for f2 in range(npredictors+1):

        # v TODO ------------------------------------------------------------
        if f1 == f2:
            axs[f1, f2].hist(X[:,[f1]])
        else:
            axs[f1, f2].scatter(X[:,[f1]], X[:,[f2]])
        # ^ TODO ------------------------------------------------------------


        # include labels only when necessary
        if f1 == npredictors:
            axs[f1, f2].set_xlabel(Xycolnames[f2])
        if f2 == 0:
            axs[f1, f2].set_ylabel(Xycolnames[f1])
```



2.7 IMAGES AND COLORMAPS

Create a colormap plot of the correlations between the all the predictors and the target

[107]:
```
"""
Generate a figure that plots the a correlation matrix
as a colormap.
PARAMS:
    corrs: matrix of correlations between the features
    varnames: list of the names of each of the features
              (e.g. the column names)
"""
def correlationmap(corrs, varnames):
    nvars = corrs.shape[0]
    # create the figure and plot the correlation matrix
    fig, ax = plt.subplots()
    im = ax.imshow(corrs, cmap='RdBu', vmin=-1, vmax=1)
    cbar = ax.figure.colorbar(im, ax=ax)
    cbar.ax.set_ylabel("Pearson Correlation", rotation=-90, va="bottom")

    # Specify the row and column ticks and labels for the figure
    ax.set_xticks(range(nvars))
    ax.set_yticks(range(nvars))
    ax.set_xticklabels(varnames)
    ax.set_yticklabels(varnames)

    # Rotate the tick labels and set their alignment.
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",␣
 ↪rotation_mode="anchor")

    # Loop over data dimensions and create text annotations.
    for i in range(nvars):
        for j in range(nvars):
            text = ax.text(j, i, "%.3f" % corrs[i, j],
                           ha="center", va="center", color="k")
# END DEF correlationmap ----------------------------------------------


"""
Compute the Pearson correlation between the columns of Xycombo using
the numpy function corrcoef(). The corrcoef() function performs the
the pairwise correlation on the rows of a matrix, thus you will need to
transpose the input.
"""
Xytranspose = np.matrix.transpose(Xycombo)
Xycorrs = np.corrcoef(Xytranspose)
""" TODO
Call the function defined above, correlationmap(), to generate a
colormap plot of the correlations between columns of the Xycombo matrix.
Mke sure to read any provided code
"""
correlationmap(Xycorrs, Xycolnames)
```


mean radius
mean area
mean concavity
mean symmetry
target
1.000 0.987 0.677 0.148 -0.730
0.987 1.000 0.686 0.151 -0.709
0.677 0.686 1.000 0.501 -0.696
0.148 0.151 0.501 1.000 -0.330
-0.730 -0.709 -0.696 -0.330 1.000
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Pearson Correlation

[ ]: