Week 4: Mini Project

This notebook will guide you through smaller portions of your final project. For this notebook, we will be using the Abalone dataset from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Abalone) (originating from the Marine Research Laboratories – Taroona). This dataset should already be in your folder (under `abalone.csv`) or you can download it at the above link.

Abalone

A Brief History of Abalones ¶

An abalone is a sea snail belonging to one of a range of 30 to 130 species (depending on which scientist you ask). It is commonly prized for its mother-of-pearl shell, pearls, and delicious flesh by a variety of cultures and has long been a valuable source of food in its native environments. Sadly, wild populations of abalone have been overfished and poached to the point where commercial farming supplies most of abalone flesh nowadays. It now sits on the list of current animals threatened by extinction.

Source: https://en.wikipedia.org/wiki/Abalone (https://en.wikipedia.org/wiki/Abalone)

Part 1: Familiarize Yourself With the Dataset

The purpose of this dataset is to predict the age of an abalone through physical characteristics, determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope -- a boring and time-consuming task. Good thing it's already been done for us!

Below is the dataset description from the UCI Machine Learning Repository.

Name	Data Type	Measure	Description
Sex	nominal		M, F, and I (infant)
Length	continuous	mm	Longest shell measurement
Diameter	continuous	mm	perpendicular to length
Height	continuous	mm	with meat in shell
Whole weight	continuous	grams	whole abalone
Shucked weight	continuous	grams	weight of meat
Viscera weight	continuous	grams	gut weight (after bleeding)
Shell weight	continuous	grams	after being dried
Rings	integer		+1.5 gives the age in years

Run the cells below to examine the dataset.

In [1]:
```
# Load Abalone dataset
# Remember to change the file location if needed
import csv
f = open("abalone.csv")
all_lines = csv.reader(f, delimiter = ',')

# We define a header ourselves since the dataset contains only the raw number
s.
dataset = []
header = ['Sex', 'Length', 'Diameter', 'Height', 'Whole Weight', 'Shucked Weig
ht', 'Viscera Weight',
          'Shell Weight', 'Rings']
for line in all_lines:
    # Aparea el encabezado con los datos
    d = dict(zip(header, line))
    # Se modifica el tipo de dato de cada caracteristica
    d['Length'] = float(d['Length'])
    d['Diameter'] = float(d['Diameter'])
    d['Height'] = float(d['Height'])
    d['Whole Weight'] = float(d['Whole Weight'])
    d['Shucked Weight'] = float(d['Shucked Weight'])
    d['Viscera Weight'] = float(d['Viscera Weight'])
    d['Shell Weight'] = float(d['Shell Weight'])
    d['Rings'] = int(d['Rings'])
    dataset.append(d)
```

In [2]:
```
# See first line of dataset
dataset[0]
```

Out[2]:
```
{'Sex': 'M',
 'Length': 0.455,
 'Diameter': 0.365,
 'Height': 0.095,
 'Whole Weight': 0.514,
 'Shucked Weight': 0.2245,
 'Viscera Weight': 0.101,
 'Shell Weight': 0.15,
 'Rings': 15}
```

Part 2: Simple Statistics

This dataset is already cleaned for us and relatively straightforward, without strings or time data. In your final project, you will have to take care of missing or tricky values yourself.

Fill in the following cells with the requested information about the dataset. The answers are given so you can check the output of your own code. For floating numbers, don't worry too much about the exact numbers as long as they are quite close -- different systems may have different rounding protocols.

Feel free to `import numpy` if you want more practice with it, or just use Python's native structures to play around with the numbers.

In [3]:
```
# Q: What is the total number of entries in the dataset?
# A: 4177
len(dataset)
```

Out[3]:
```
4177
```

In [4]:
```
# Q: What is the average length of an abalone?
# A: 0.5239920995930099 or 0.524
length = []
for value in dataset:
    length.append(value['Length'])
avLength = sum(length)/len(length)
avLength
```

Out[4]:
```
0.5239920995930099
```

In [5]:
```
# Q: What is the widest abalone in the dataset (diameter)?
# A: 0.65
diameters = []
for value in dataset:
    diameters.append(value['Diameter'])
max_diameter = max(diameters)
max_diameter
```

Out[5]:
```
0.65
```

In [6]:
```
rings = []
for value in dataset:
    rings.append(value['Rings'])
```

In [7]:
```
min(length),max(length),min(rings),max(rings)
```

Out[7]:
```
(0.075, 0.815, 1, 29)
```

In [8]:
```
len(rings),len(length)
```

Out[8]:
```
(4177, 4177)
```

PARSE RINGS VS LENGTH

In [9]:
```
# Se filtran en small los elementos cuya longitud es menor al promedio
# y en larger aquellos cuya longitud es mayor al promedio

from collections import defaultdict

small = defaultdict(list)
large = defaultdict(list)

# value = filas del dataset (diccionarios individuales)
for value in dataset:
    if value['Length'] <= 0.524:
        small[value['Rings']].append(value['Length'])
    else:
        large[value['Rings']].append(value['Length'])
```

In [10]:
```
# (valor * cantidad de elementos con ese valor) / cantidad total de elementos

ageSmall = 0
ageLarge = 0
number_ageSmall = 0
number_ageLarge = 0

for v in small:
    ageSmall += v*len(small[v])
    number_ageSmall += len(small[v])

for v in large:
    ageLarge += v*len(large[v])
    number_ageLarge += len(large[v])

ageSmall = ageSmall/number_ageSmall
ageLarge = ageLarge/number_ageLarge
```

In [11]:
```
# Q: What is the average number of rings of smaller abalones compared to that
 of larger abalones?
#    That is, do smaller abalones tend to be younger or older than larger abal
ones?
#    We will count small abalones as abalones with lengths less than or equal
 to the average length of
#    an abalone. The average length of an abalone is 0.524.
# A: Small Abalones have on average 8.315645514223196 rings.
#    Large Abalones have on average 11.192848020434228 rings.

# Change variable name if necessary
print('Small Abalones have on average', ageSmall, 'rings.')
print('Large Abalones have on average', ageLarge, 'rings.')
```

```
Small Abalones have on average 8.315645514223196 rings.
Large Abalones have on average 11.192848020434228 rings.
```

Part 3: Data Visualizations

In this course, we learned about [Matplotlib (https://matplotlib.org)](https://matplotlib.org), a "Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms". There are a [variety of plots and figures (https://matplotlib.org/gallery/index.html)](https://matplotlib.org/gallery/index.html) we can make with Matplotlib, and in conjunction with NumPy, becomes a powerful and versatile tool in your skillset.

In lectures, we covered the basics of line plots, histograms, scatter plots, bar plots, and box plots. Let's try out a few below.

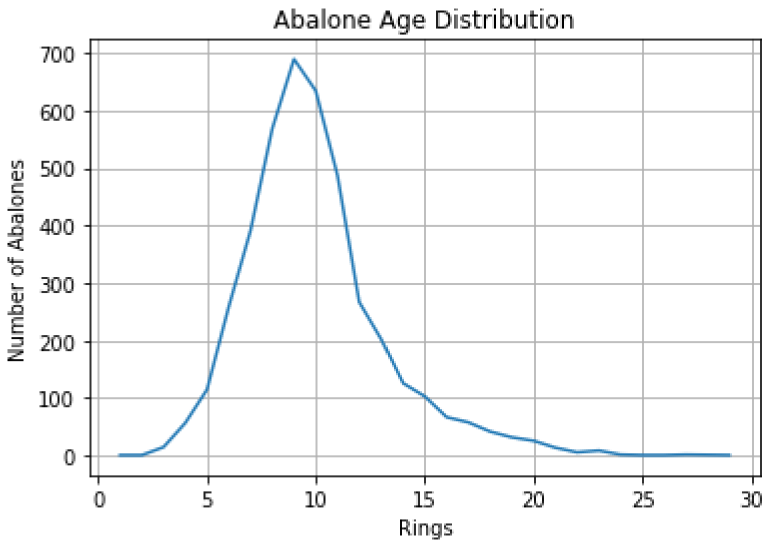
In [12]:
```
import matplotlib.pyplot as plt
from matplotlib import colors
import numpy
from collections import defaultdict
```

Line Plots

Line plots show the change in data over time. The example Line Plot below plots the change in density as abalones age (i.e. the distribution of rings). **Note that a line plot is not necessarily the best way to show this data since it doesn't deal with a trend!** Use a histogram (next step) to better showcase this data.

In [13]:
```
# Parse out Rings column from dataset
rings = [d['Rings'] for d in dataset]
rings.sort()

# Count number of abalones with each number of rings with defaultdict
abalone_rings = defaultdict(int)
for r in rings:
    abalone_rings[r] += 1
X = list(abalone_rings.keys())
Y = list(abalone_rings.values())

# Customize plot
plt.gca().set(xlabel='Rings', ylabel='Number of Abalones',
       title='Abalone Age Distribution')
plt.grid()

# Show the plot of Rings vs Number of Abalones
plt.plot(X, Y)
```

Out[13]:
```
[<matplotlib.lines.Line2D at 0x2664ad4b248>]
```



In [14]:
```
abalone_rings
```

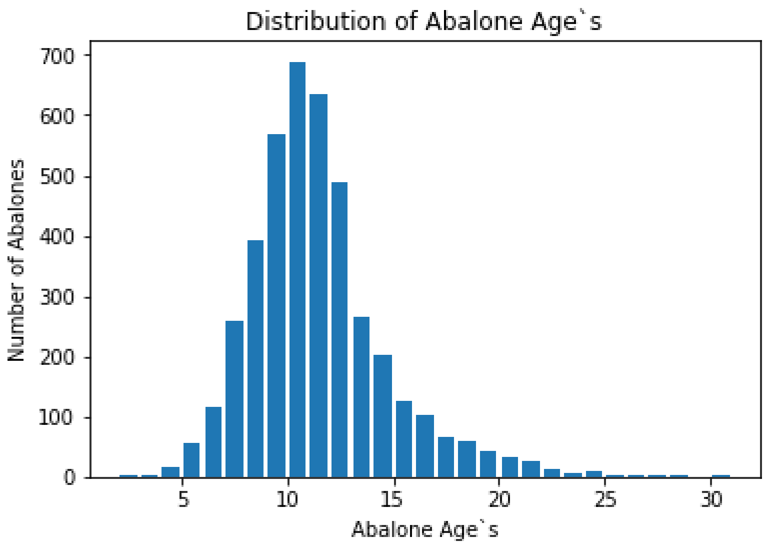
Out[14]:
```
defaultdict(int,
            {1: 1,
             2: 1,
             3: 15,
             4: 57,
             5: 115,
             6: 259,
             7: 391,
             8: 568,
             9: 689,
             10: 634,
             11: 487,
             12: 267,
             13: 203,
             14: 126,
             15: 103,
             16: 67,
             17: 58,
             18: 42,
             19: 32,
             20: 26,
             21: 14,
             22: 6,
             23: 9,
             24: 2,
             25: 1,
             26: 1,
             27: 2,
             29: 1})
```

Histograms

Histograms show the distribution of numeric continuous variables with central tendency and skewness. **Using the line plot data from above, plot a histogram showing the distribution of abalone age.** Feel free to explore matplotlib (https://matplotlib.org/gallery/index.html) on your own to customize your histogram and the following visualizations.

In [25]:
```
# Complete this cell with a histogram of abalone age distribution

# Flatten distribution list into frequency distribution
age_freq = []
for key in abalone_rings.keys():
    #print(abalone_rings.get(key))
    for i in range(0, abalone_rings.get(key)):
        # para la cantidad de valores asociados que tenga cada clave (anillo
s),
        # se agrega key (cantidad de anillos) a age_freq
        age_freq.append(key)
#print(age_freq[:30])

ages = defaultdict(int)
for age in age_freq:
    ages[age] += 1
#print(ages)

real_ages = []
for age in ages:
    real_ages.append(age + 1.5)

# Plot your histogram here
plt.bar(real_ages,ages.values())
plt.xlabel('Abalone Age`s')
plt.ylabel('Number of Abalones')
plt.title('Distribution of Abalone Age`s')
```

Out[25]:
```
Text(0.5, 1.0, 'Distribution of Abalone Age`s')
```



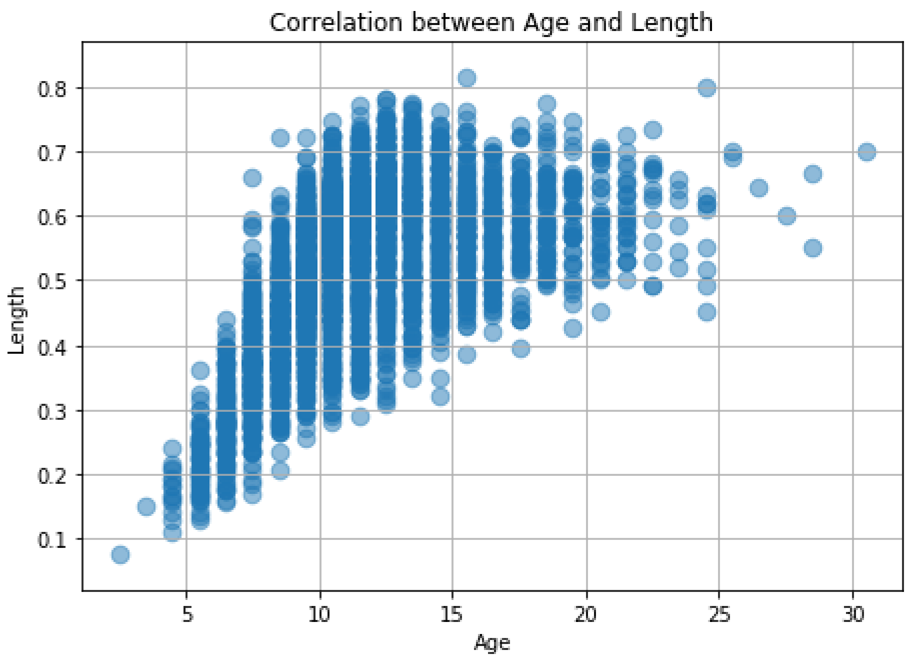
Scatter Plots

Scatter plots show the strength of a relationship between two variables (also known as correlations). From *Part 2: Simple Statistics*, we see that larger abalones tend to be larger, at least from a numbers perspective. **Let's see if this is actually true by creating a scatter plot showing the relationship between `Rings` and `Length`.**

On Your Own: Read up on `sciPy` and how you can calculate and graph the correlation as well.

In [55]:
```python
import matplotlib.pyplot as plt
# Complete this cell with a scatter plot of age vs length
rings = [d['Rings'] for d in dataset]
length = [d['Length'] for d in dataset]

ages = []
for ring in rings:
    ages.append(ring+1.5)

plt.axes([0.025, 0.025, 0.95, 0.95])
plt.scatter(ages, length, s=75, alpha=.5)

plt.xlabel('Age')
plt.ylabel('Length')
plt.grid(True)
plt.title('Correlation between Age and Length')
```

Out[55]:
```
Text(0.5, 1.0, 'Correlation between Age and Length')
```



Scatter plot parse:

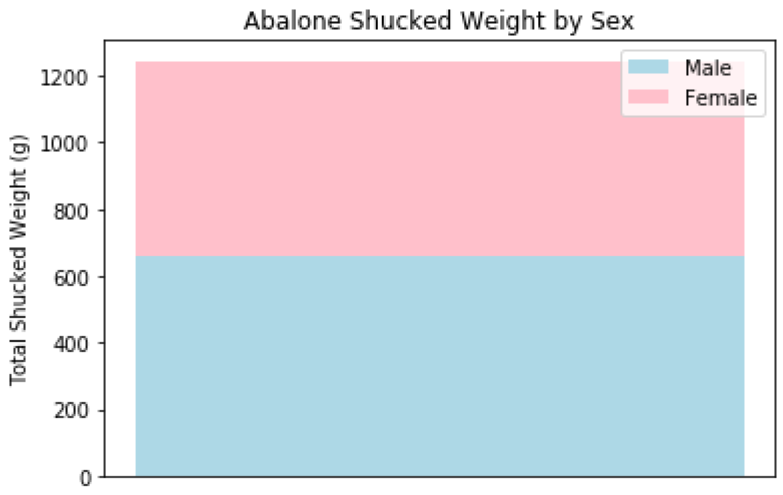
- Direction: Positive
- Form: Non-linear
- Strength: Strong
- Outliers: Very Few

Bar Plots

Bar plots are great for comparing categorical variables. There are a few subtypes of bar plots, such as the grouped bar chart or stacked bar chart. Since we have the `Sex` field to play with, we can compare data across `M` and `F` abalones. Below is a simple stacked bar chart comparing the `Sex` category with the `Shucked Weight` data. **Create a bar chart of your choice of data.**

You may refer to the cell below to parse out fields by sex.

In [58]:
```
# Example Stacked Bar Chart - Comparisons Between Sexes
# La suma de todos los pesos a partir de la cual estan los pesos
Mweight = sum([d['Shucked Weight'] for d in dataset if d['Sex'] is 'M'])
Fweight = sum([d['Shucked Weight'] for d in dataset if d['Sex'] is 'F'])
index = [1]

p1 = plt.bar(index, Mweight, color='lightblue')
p2 = plt.bar(index, Fweight, bottom=Mweight, color='pink')
plt.gca().set(title='Abalone Shucked Weight by Sex', ylabel='Total Shucked Wei
ght (g)');
plt.xticks([])

plt.legend((p1[0], p2[0]), ('Male', 'Female'))
plt.show()
```



```
In [71]: # Complete this cell with your choice of data
         Mweight,Fweight
```

```
Out[71]: (661.5415000000003, 583.1675)
```

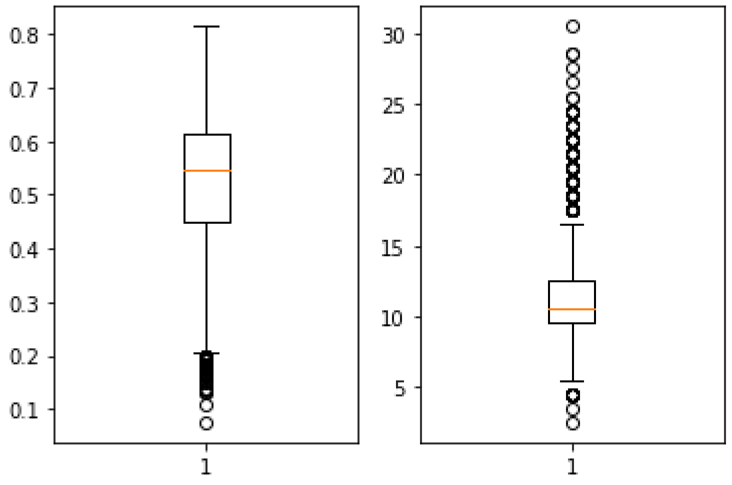
Box Plots

Box plots are useful for comparing distributions of data and are commonly found in research papers. The box portion of a box plot represents 50% of the data, and there are versions where you can mark outliers and other extremes. We have the distribution of rings already from the line plot example under the variable name `age_freq` , assuming you haven't modified it. **Find the distribution of another field of your choice and create one or more box plots with both of these fields.**

Hint: You can plot multiple box plots with the command `plt.boxplot([plot1, plot2, ..., plotn])` *or use* `subplots()` *to draw multiple separate plots at the same time. See [this matplotlib example (https://matplotlib.org/gallery/statistics/boxplot_demo.html#sphx-glr-gallery-statistics-boxplot-demo-py)](https://matplotlib.org/gallery/statistics/boxplot_demo.html#sphx-glr-gallery-statistics-boxplot-demo-py) for more.*

```
In [84]: # Complete this cell with multiple box plots
         plt.subplot(1,2,1)
         plt.boxplot(length)
         plt.subplot(1,2,2)
         plt.boxplot(ages)
```

```
Out[84]: {'whiskers': [<matplotlib.lines.Line2D at 0x2664d9bf8c8>,
           <matplotlib.lines.Line2D at 0x2664d9caa88>],
          'caps': [<matplotlib.lines.Line2D at 0x2664d9ca448>,
           <matplotlib.lines.Line2D at 0x2664d9d4988>],
          'boxes': [<matplotlib.lines.Line2D at 0x2664d9bf608>],
          'medians': [<matplotlib.lines.Line2D at 0x2664d9d4a88>],
          'fliers': [<matplotlib.lines.Line2D at 0x2664d9cca88>],
          'means': []}
```



Free Response (optional)

Experiment and create visualizations of your own here.