Assignment 6 Regression

In [21]:
```
%pylab inline

# (c) 2014 Reid Johnson and Everaldo Aguiar
#
# Functions to work with continuous data and linear regression models.

import matplotlib.pyplot as pl

def pairs(data):
    """Generates and shows a pairwise scatterplot of the dataset features.

    A figure with nxn scatterplots is generated, where n is the number of features. The features are
    defined as the all columns excluding the final column, which is defined as the class.

    Args:
      data (array): A dataset.

    """
    i = 1

    # Divide columns into features and class.
    features = list(data.columns)
    classes = features[-1] # create class column
    del features[-1] # delete class column from feature vector

    # Generate an nxn subplot figure, where n is the number of features.
    figure = pl.figure(figsize=(5*(len(data.columns)-1), 4*(len(data.columns)-1)))
    for col1 in data[features]:
        for col2 in data[features]:
            ax = pl.subplot(len(data.columns)-1, len(data.columns)-1, i)
            if col1 == col2:
                ax.text(2.5, 4.5, col1, style='normal', fontsize=20)
                ax.axis([0, 10, 0, 10])
                pl.xticks([]), pl.yticks([])
            else:
                for name in data[classes]:
                    cond = data[classes] == name
                    ax.plot(data[col2][cond], data[col1][cond], linestyle='none', marker='o', label=name
                #t = plt.title(name)
            i += 1

    pl.show()
```

```
Populating the interactive namespace from numpy and matplotlib
```

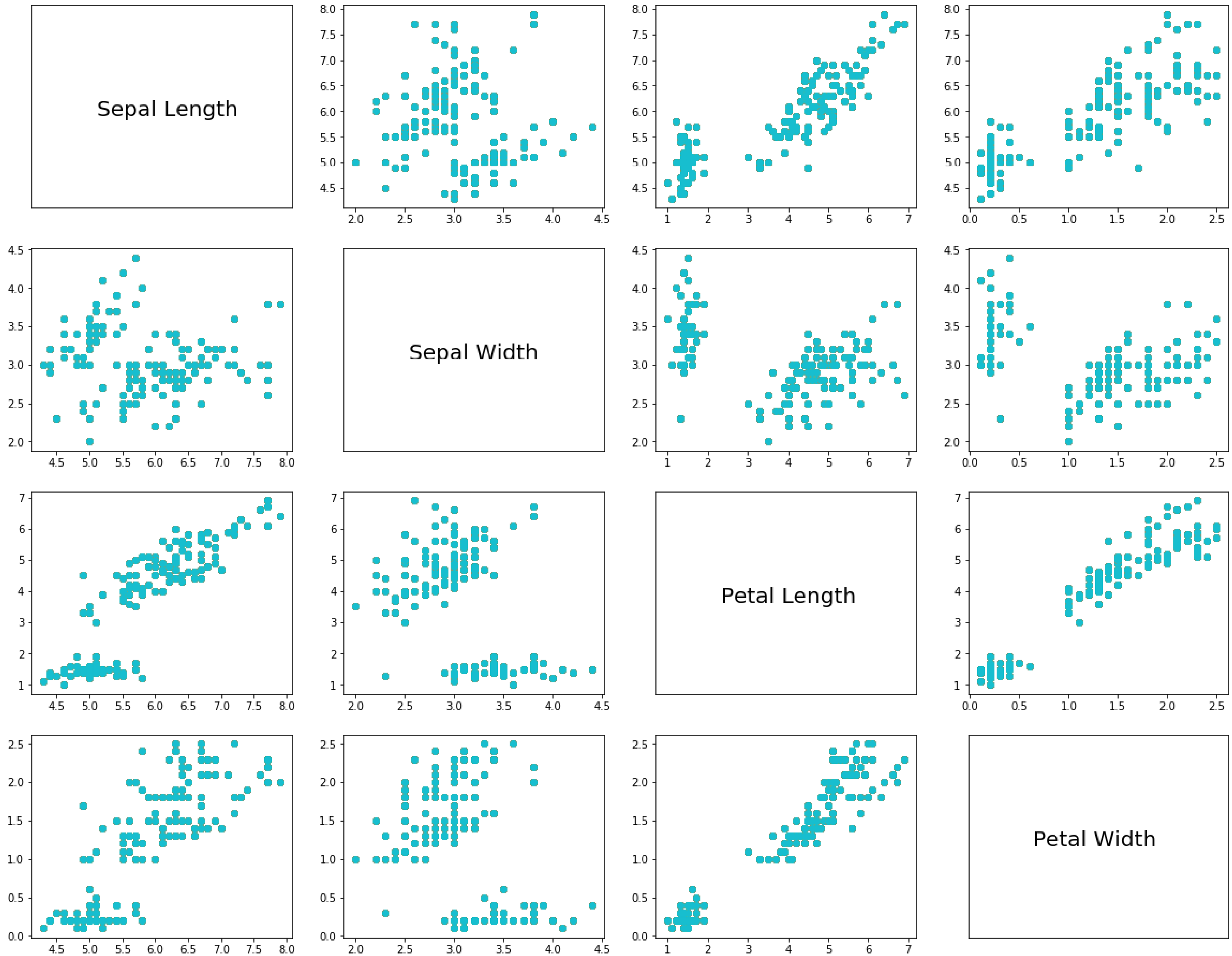
In [178]:
```
import pandas as pd

fileURL = 'http://archive.ics.uci.edu/ml/\
machine-learning-databases/iris/iris.data'
iris = pd.read_csv(fileURL, names=['Sepal Length', 'Sepal Width', \
                                   'Petal Length', 'Petal Width',
                                   'Species'], header=None)
iris = iris.dropna()
```

In [179]:
```
iris.head()
```

Out[179]:

	Sepal Length	Sepal Width	Petal Length	Petal Width	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

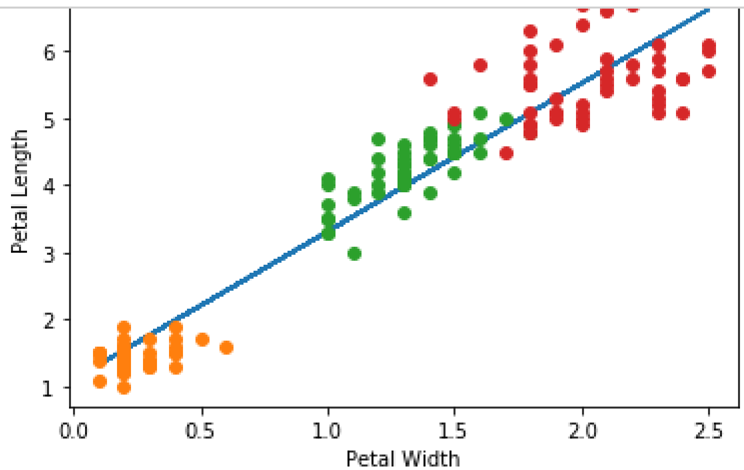
In [174]:
```
from  sklearn import  datasets
pairs(iris)
```



In [250]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
import itertools as iter

# Function to loop over loading data and running lreg
def lreg_stats(x_label, y_label):
    x_label = x_label[0]
    y_label = y_label[0]
    # Split into training and test data
    train, test = train_test_split(iris)
    x_train = train.iloc[:, x_label]
    y_train = train.iloc[:, y_label]
    x_test = test.iloc[:, x_label]
    y_test = test.iloc[:, y_label]

    # Make the linear model: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lin
    regr = LinearRegression()
    # Separate into x and y columns
    x_train = x_train.values.reshape(-1, 1)
    x_test = x_test.values.reshape(-1, 1)
    # Fit Lienear Model
    regr = regr.fit(x_train, y_train)

    # loop over species names and plot
    pl.plot(x_train, regr.predict(x_train))
    for name in set(iris['Species']):
        cond = iris['Species'] == name
        pl.plot(iris.iloc[:, x_label][cond], iris.iloc[:, y_label][cond], linestyle='none', marker='o')

    # Label the axes with the features
    pl.xlabel(iris.columns[x_label])
    pl.ylabel(iris.columns[y_label])
    pl.show()

    # The coefficient(s).
    print ("Coefficient(s): ", regr.coef_)

    # The mean square error.
    print ("Residual sum of squares: %.2f" % np.mean((regr.predict(x_test) - y_test) ** 2)))

    # Explained variance score (1 is perfect prediction).
```

```
    print ("Variance score: %.2f" % regr.score(x_test, y_test))

#Funcion to keep track of MSE while iterating over combinations of features
def get_mse(x_label, y_label):

    x_label = x_label[0]
    y_label = y_label[0]
    # Split into training and test data
    train, test = train_test_split(iris)
    x_train = train.iloc[:, x_label]
    y_train = train.iloc[:, y_label]
    x_test = test.iloc[:, x_label]
    y_test = test.iloc[:, y_label]

    # Make the linear model: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lin
    regr = LinearRegression()
    # Separate into x and y columns
    x_train = x_train.values.reshape(-1, 1)
    x_test = x_test.values.reshape(-1, 1)
    # Fit Lienear Model
    regr = regr.fit(x_train, y_train)
    # Get MSE only.
    mse = np.mean(((regr.predict(x_test) -y_test) ** 2))

    return (mse, (x_label, y_label))

# Remove 'species' from columns
cols = iris.columns[:-1]
cols = size(cols)
print(cols)

# y is independent variable, x is dependent
for y in range(cols):
    for x in range(cols):
        if y != x:
            # Print out all plots and corresponding stats
            lreg_stats((y,), (x,))

# Get combinations of features
# https://stackoverflow.com/questions/464864/how-to-get-all-possible-combinations-of-a-list-s-elements
def combinations(features):
    comb = (iter.combinations(features, l) for l in range(len(features) + 1))
    return list(iter.chain.from_iterable(comb))
```

```
# Initialize best mse and the best features to be updated
min_mse = 100
best_features = 0
for x in range(cols):
    features = []
    for feature in range(cols):
        if x != feature:
            features.append(feature)
    for f in combinations(features):
        if f:
            (curr_mse, (x_label, y_label)) = get_mse(f, (x, ))
            if curr_mse < min_mse:
                min_mse = curr_mse
                best_features = (x_label, y_label)
                print("Best Feature Combinations ",best_features[:])

# Get stats and plot of best feature combos
lreg_stats(best_features[0:], best_features[1:])
```




```
Coefficient(s):  [2.20946766]
Residual sum of squares: 0.15
Variance score: 0.95
Best Feature Combinations  (1, 0)
Best Feature Combinations  (2, 0)
```

1. Based upon the linear models you generated, which pair of features appear to be most predictive for one another? Note that you can answer this question based upon the output provided for the linear models.

The "Best Feature Combination" for being most predictive for one another is (2,3) which corresponds to Petal Length (2) and Petal Width (3), respectively.

2. Suppose you tried to generate a classification model on this dataset, but only after removing the feature that you were best able to predict based upon other features. How would removing this feature affect the classification performance?

The best feature to predict upon other features is Petal Length based on the results. If we were to remove Petal Length, then other features would combine to get similar results, although different, and have minimal effect on classification performance. However since these Petal Length is highly correlated to the other features, removing it may actually give us more reliable results (although not improved statistics) because linear models are susceptible to numerical instability due to multicollinearity (https://en.wikipedia.org/wiki/Multicollinearity).

In [ ]: