Iris Flower

In [11]:

```
# Load libraries
import pandas
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

plt.rcParams['figure.figsize'] = [10, 5]
```

Loading the dataset:

In [12]:

```
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
# names of the columns
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset = pandas.read_csv(url, names=names)
```

Summarize the Dataset

1. Dimensions of the dataset
2. Peek at the data
3. Statistical summary of all attributes
4. Breakdown of the date by the class variable

Dimensions of dataset:

In [3]:

```
# shape
print(dataset.shape)
# (instances, attributes)
```

```
(150, 5)
```

Peek at the data

In [4]:

```
# head
print(dataset.head(20))
```

```
    sepal-length  sepal-width  petal-length  petal-width        class
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
5            5.4          3.9           1.7          0.4  Iris-setosa
6            4.6          3.4           1.4          0.3  Iris-setosa
```

```
7            5.0           3.4            1.5           0.2  Iris-setosa
8            4.4           2.9            1.4           0.2  Iris-setosa
9            4.9           3.1            1.5           0.1  Iris-setosa
10           5.4           3.7            1.5           0.2  Iris-setosa
11           4.8           3.4            1.6           0.2  Iris-setosa
12           4.8           3.0            1.4           0.1  Iris-setosa
13           4.3           3.0            1.1           0.1  Iris-setosa
14           5.8           4.0            1.2           0.2  Iris-setosa
15           5.7           4.4            1.5           0.4  Iris-setosa
16           5.4           3.9            1.3           0.4  Iris-setosa
17           5.1           3.5            1.4           0.3  Iris-setosa
18           5.7           3.8            1.7           0.3  Iris-setosa
19           5.1           3.8            1.5           0.3  Iris-setosa
```

Statistical Summary

In [5]:

```
# descriptions
print(dataset.describe())
```

```
       sepal-length  sepal-width  petal-length  petal-width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

Class Distribution

(There are 50 instances of each class of flower)

In [6]:

```
# class distribution
print(dataset.groupby('class').size())
```

```
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

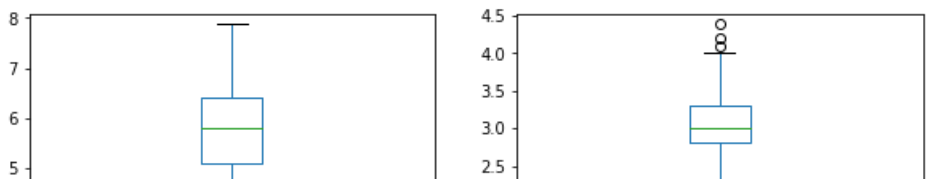
Data Visualization

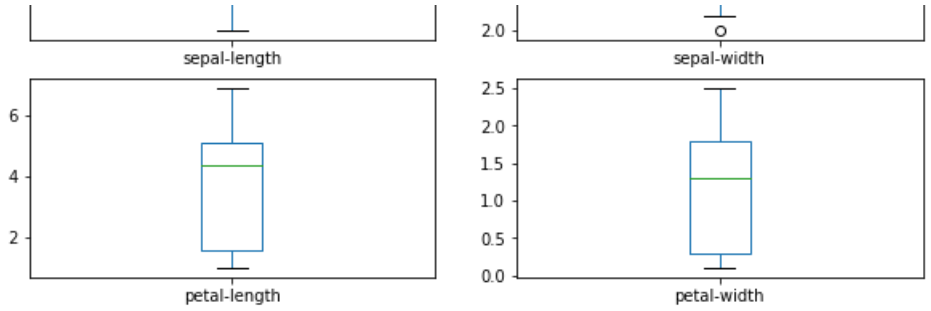
1. Univariate plots to better understand each attribute
2. Multivariate plots to better understand the relationship between attributes

Univariate Plots

In [13]:

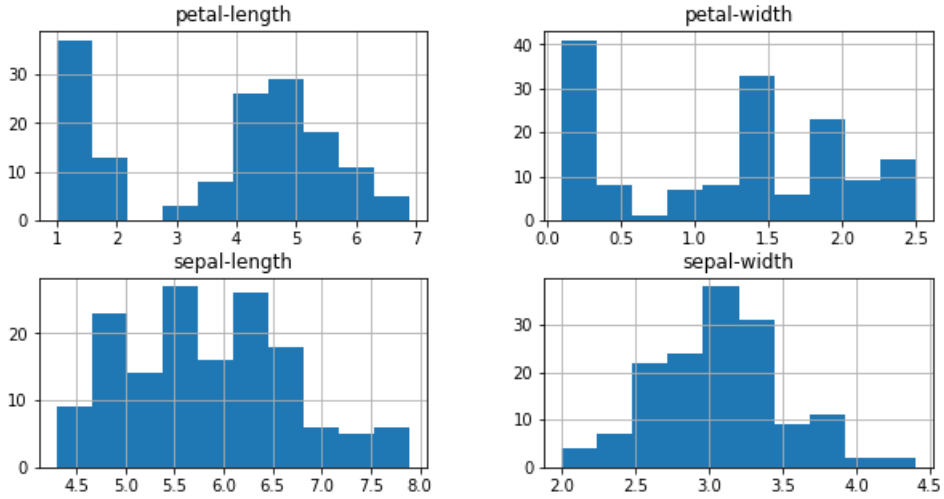
```
# box and whisker plots
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```





In [14]:

```
# histograms
dataset.hist()
plt.show()
```



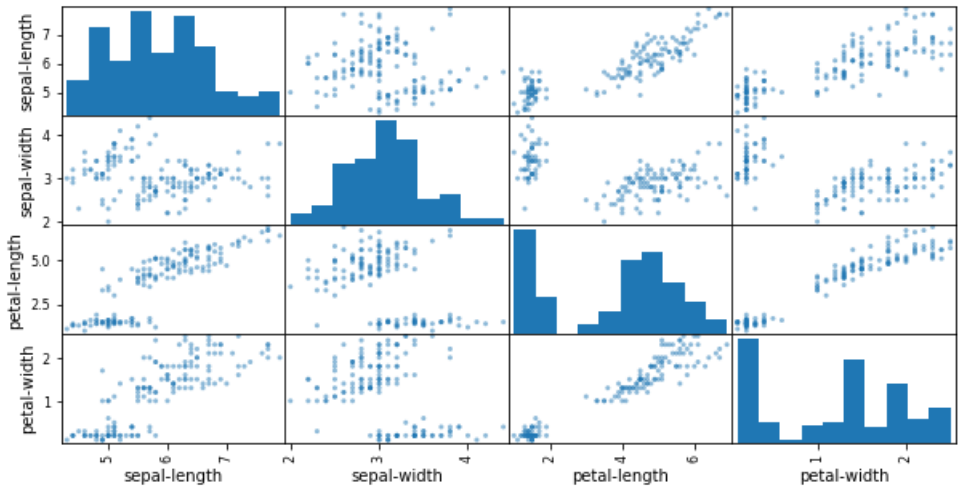
Looks like two of the input variables have a Gaussian distribution. This is useful ot note as we can use algorithms the can exploit this assumption

Multivariable Plots

Now we can look at the interactions between variables

In [15]:

```
# scatter plot matrix
scatter_matrix(dataset)
plt.show()
```



Note the diagonal grouping of some pairs of attributes. This suggests a high correlation and a predictable relationship.

Evaluate Some Algorithms

1. Separate out a validation dataset.
2. Set-up the test harness to use 10-fold cross validation
3. Build 5 different models to predict species from flower measurements
4. Select best model

Create a Validation Dataset

We need to know that the model we created is any good. We will use statistical methods to estimate the accuracy of the models that we create on unseen data. To do this we need to hold back some data that the alogrithms will not get to see. We will split the into two, 80% of which we will to train our models and 20% that we will hold back as a validation dataset.

In [10]:

```
# Split-out validation dataset
array = dataset.values
X = array[:,0:4]
Y = array[:,4]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y, test_size=val
idation_size, random_state=seed)
```

Test harness

We will use 10-fold cross validation to estimate accuracy. This will split out dataset into 10 parts, train on 9 and test on 1 and repeat for all combinations of train-test splits.

In [16]:

```
# Test options and evaluation metric
seed = 7
scoring = 'accuracy'
```

We are using the metric of ‘accuracy‘ to evaluate models. This is a ratio of the number of correctly predicted instances in divided by the total number of instances in the dataset multiplied by 100 to give a percentage (e.g. 95% accurate). We will be using the scoring variable when we run build and evaluate each model next.

Build Models

Let’s evaluate 6 different algorithms:

Logistic Regression (LR) Linear Discriminant Analysis (LDA) K-Nearest Neighbors (KNN). Classification and Regression Trees (CART). Gaussian Naive Bayes (NB). Support Vector Machines (SVM).

In [22]:

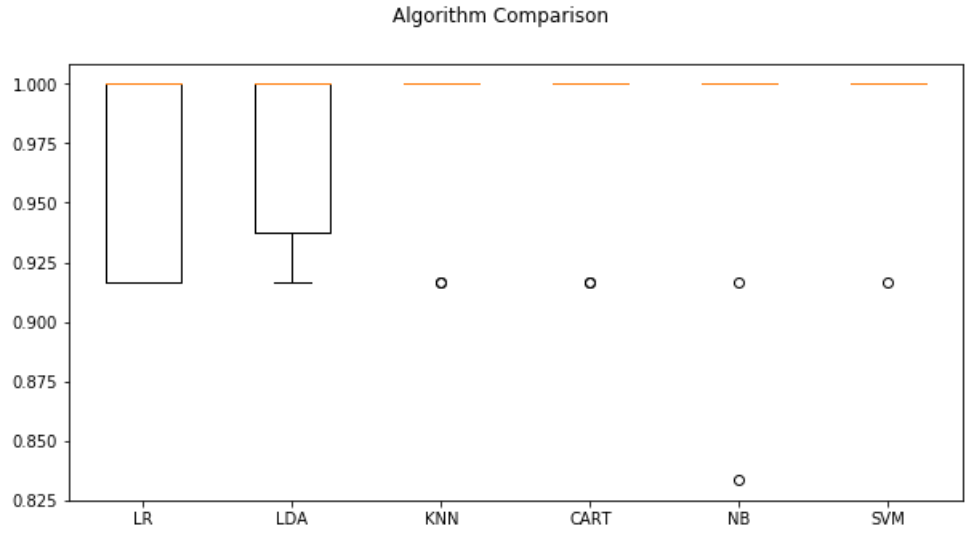
```
# Spot check Algorithms
models = []
models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring
)
    results.append(cv_results)
```

```
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.966667 (0.040825)
LDA: 0.975000 (0.038188)
KNN: 0.983333 (0.033333)
CART: 0.983333 (0.033333)
NB: 0.975000 (0.053359)
SVM: 0.991667 (0.025000)
```

In [24]:

```
# Compare Algorithms
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```



Make Predictions

In [27]:

```
# make predictions on validation dataset
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)
predictions = knn.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation,predictions))
```

```
0.9
[[ 7  0  0]
 [ 0 11  1]
 [ 0  2  9]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.85      0.92      0.88        12
 Iris-virginica       0.90      0.82      0.86        11

       accuracy                           0.90        30
      macro avg       0.92      0.91      0.91        30
   weighted avg       0.90      0.90      0.90        30
```

In [ ]: