In [1]:

```
# Load necessary libs
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
```

In [2]:

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

In [4]:

```
dataset.shape
```

Out[4]:

```
(150, 5)
```

In [5]:

```
# Observe data
dataset.describe()
```

Out[5]:

	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

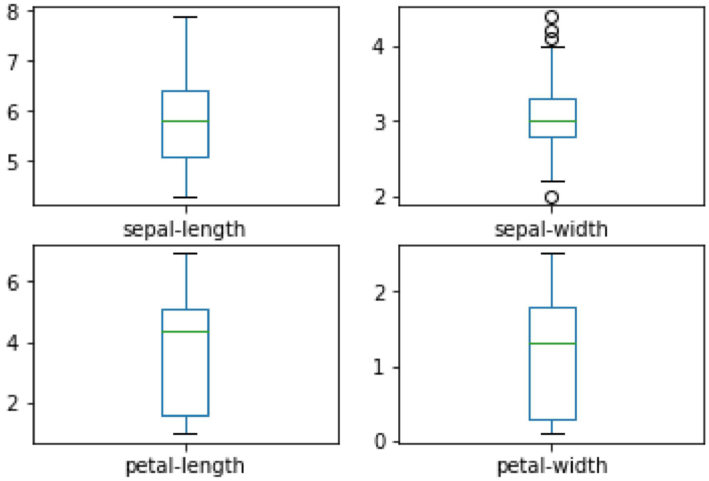
In [6]:

```
dataset.groupby('class').size()
```

Out[6]:

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

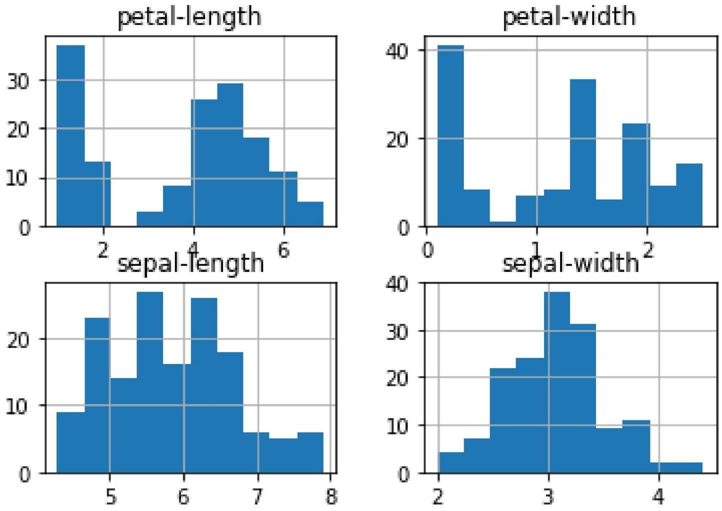
In [7]:
```
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=Fal
se)
plt.show()
```



In [8]:
```
dataset.hist()
```

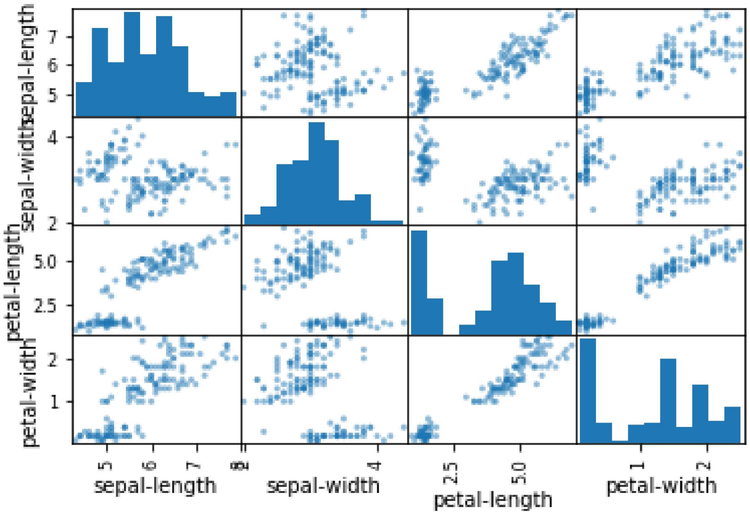
Out[8]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001EFB5684240>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EFB5DF0208
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001EFB5E2B278>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EFB5E63358
>]], dtype=object)
```

In [9]:
```
plt.show()
```



In [10]:

```
scatter_matrix(dataset)
plt.show()
```



In [11]:

```
# Prepare data
data = dataset.values
X = data[:,0:4]
Y = data[:,4]
X_train,X_validation,Y_train,Y_validation = model_selection.train_test_split(X
,Y,test_size=0.2,random_state=7)
```

In [12]:

```
#Choose models
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
```

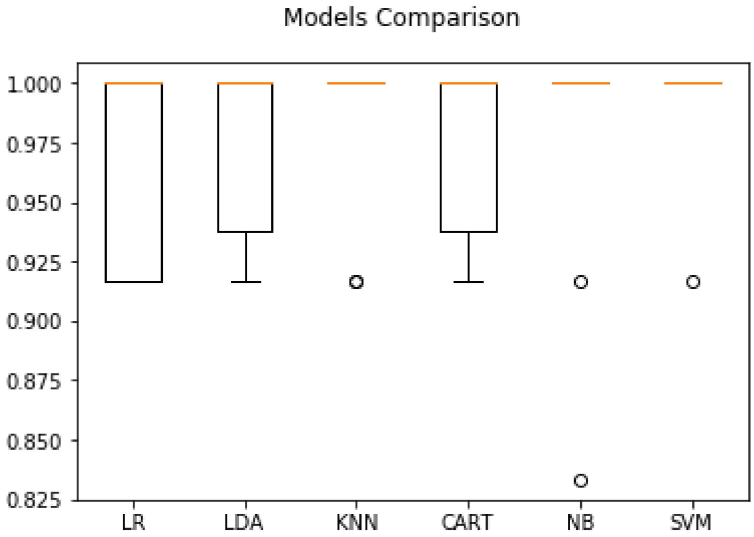
In [18]:

```
# Train and evaluate the models
kfold = model_selection.KFold(n_splits=10, random_state=7)
results = []
names = []
for name, model in models:
    result = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold
, scoring='accuracy')
    results.append(result)
    names.append(name)
    report = "%s : %f(%f)" % (name,result.mean(),result.std())
    print (report)
```

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

In [14]:

```
# Comparing the models
fig = plt.figure()
fig.suptitle('Models Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```



In [15]:

```
# making predictions with SVM
SVM = SVC()
SVM.fit(X_train, Y_train)
predictions = SVM.predict(X_validation)
print (accuracy_score(Y_validation,predictions))
print (confusion_matrix(Y_validation,predictions))
print (classification_report(Y_validation, predictions))
```

```
0.933333333333
[[ 7  0  0]
 [ 0 10  2]
 [ 0  0 11]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       1.00      0.83      0.91        12
 Iris-virginica       0.85      1.00      0.92        11

    avg / total       0.94      0.93      0.93        30
```

In [16]:

```
# making predictions with LR
LR = LogisticRegression()
LR.fit(X_train, Y_train)
predictions = LR.predict(X_validation)
print (accuracy_score(Y_validation,predictions))
print (confusion_matrix(Y_validation,predictions))
print (classification_report(Y_validation, predictions))
```

```
0.8
[[ 7  0  0]
 [ 0  7  5]
 [ 0  1 10]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.88      0.58      0.70        12
 Iris-virginica       0.67      0.91      0.77        11

    avg / total       0.83      0.80      0.80        30
```

In [17]:

```
# making predictions with LDA
LDA = LinearDiscriminantAnalysis()
LDA.fit(X_train, Y_train)
predictions = LDA.predict(X_validation)
print (accuracy_score(Y_validation,predictions))
print (confusion_matrix(Y_validation,predictions))
print (classification_report(Y_validation, predictions))
```

```
0.966666666667
[[ 7  0  0]
 [ 0 11  1]
 [ 0  0 11]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       1.00      0.92      0.96        12
 Iris-virginica       0.92      1.00      0.96        11

    avg / total       0.97      0.97      0.97        30
```

In [19]:

```
# making predictions with KNN
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

    avg / total       0.90      0.90      0.90        30
```

In [20]:

```
# making predictions with CART
CART = DecisionTreeClassifier()
CART.fit(X_train, Y_train)
predictions = CART.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

    avg / total       0.90      0.90      0.90        30
```

In [21]:

```
# making predictions with NB
NB = GaussianNB()
NB.fit(X_train, Y_train)
predictions = NB.predict(X_validation)
print (accuracy_score(Y_validation,predictions))
print (confusion_matrix(Y_validation,predictions))
print (classification_report(Y_validation, predictions))
```

```
0.833333333333
[[7 0 0]
 [0 9 3]
 [0 2 9]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.82      0.75      0.78        12
 Iris-virginica       0.75      0.82      0.78        11

    avg / total       0.84      0.83      0.83        30
```