In [1]:

```
import pandas as pd
import numpy as np
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
```

In [2]:

```
filename = 'iris.csv'
```

In [3]:

```
df = pd.read_csv(filename)
```

In [6]:

```
df[[ 'SepalLengthCm' , 'SepalWidthCm' , 'PetalLengthCm' ,'PetalWidthCm' ]] = df[['SepalLeng
```

In [7]:

```
print (df.isnull().sum())
```

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [8]:

```
print(df.shape)
```

```
(150, 6)
```

In [9]:

```
print(df.dtypes)
```

```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [10]:

```
print(df.head(5))
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
```

In [11]:

```
print(df.describe())
```

```
               Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  150.000000     150.000000    150.000000     150.000000    150.000000
mean    75.500000       5.843333      3.054000       3.758667      1.198667
std     43.445368       0.828066      0.433594       1.764420      0.763161
min      1.000000       4.300000      2.000000       1.000000      0.100000
25%     38.250000       5.100000      2.800000       1.600000      0.300000
50%     75.500000       5.800000      3.000000       4.350000      1.300000
75%    112.750000       6.400000      3.300000       5.100000      1.800000
max    150.000000       7.900000      4.400000       6.900000      2.500000
```

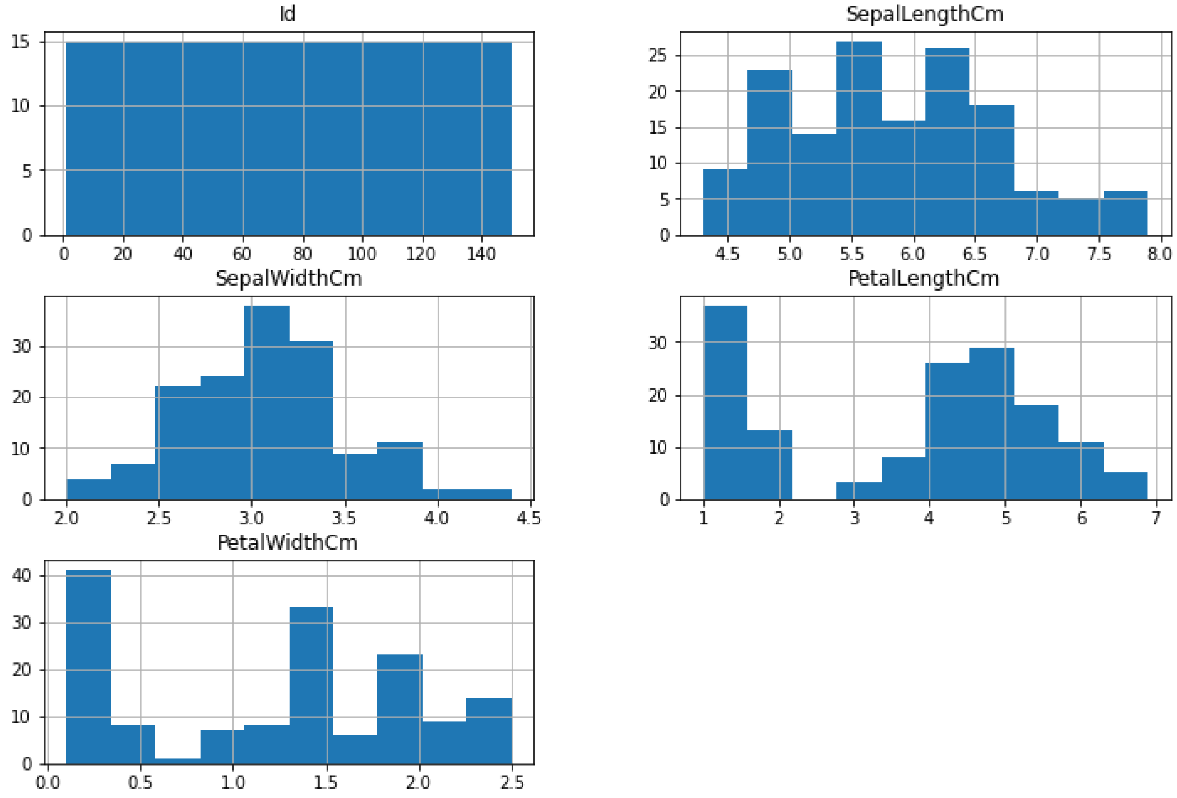
In [12]:

```
print(df.groupby('Species').size())
```

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

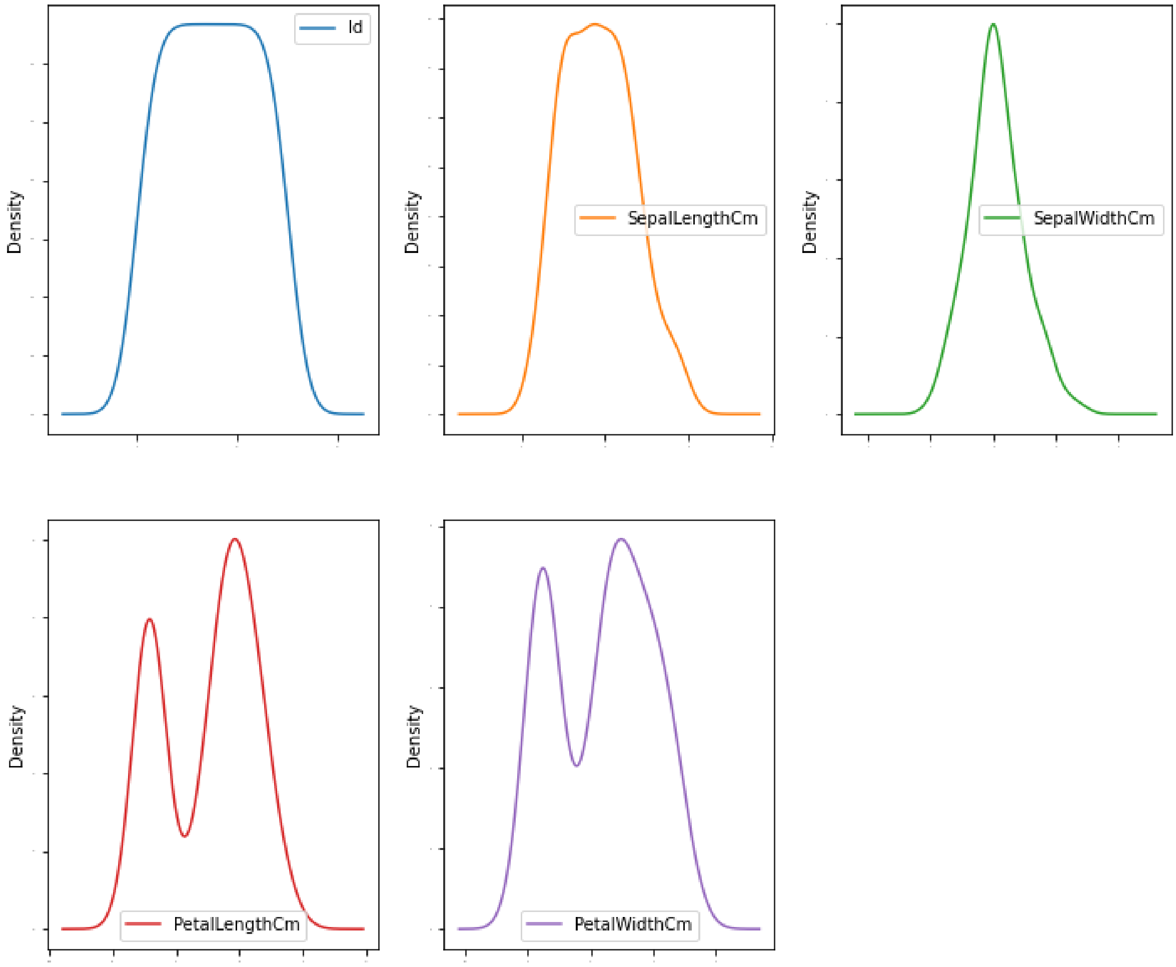
In [13]:

```
df.hist(figsize=(12, 8))
pyplot.show()
```



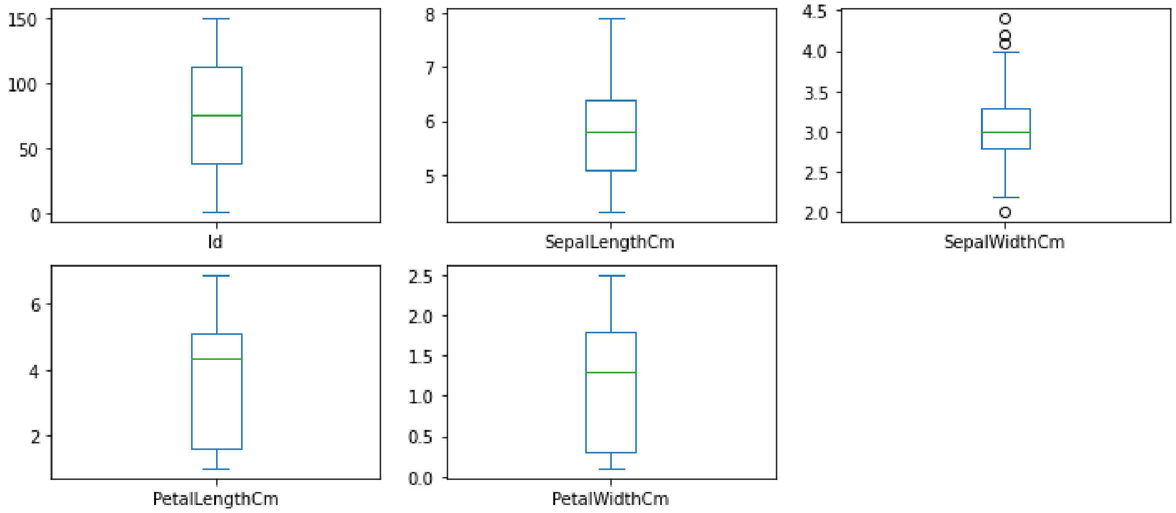
In [14]:

```
df.plot(kind='density', subplots=True, layout=(3, 3), sharex=False,
legend=True, fontsize=1,
figsize=(12, 16))
pyplot.show()
```



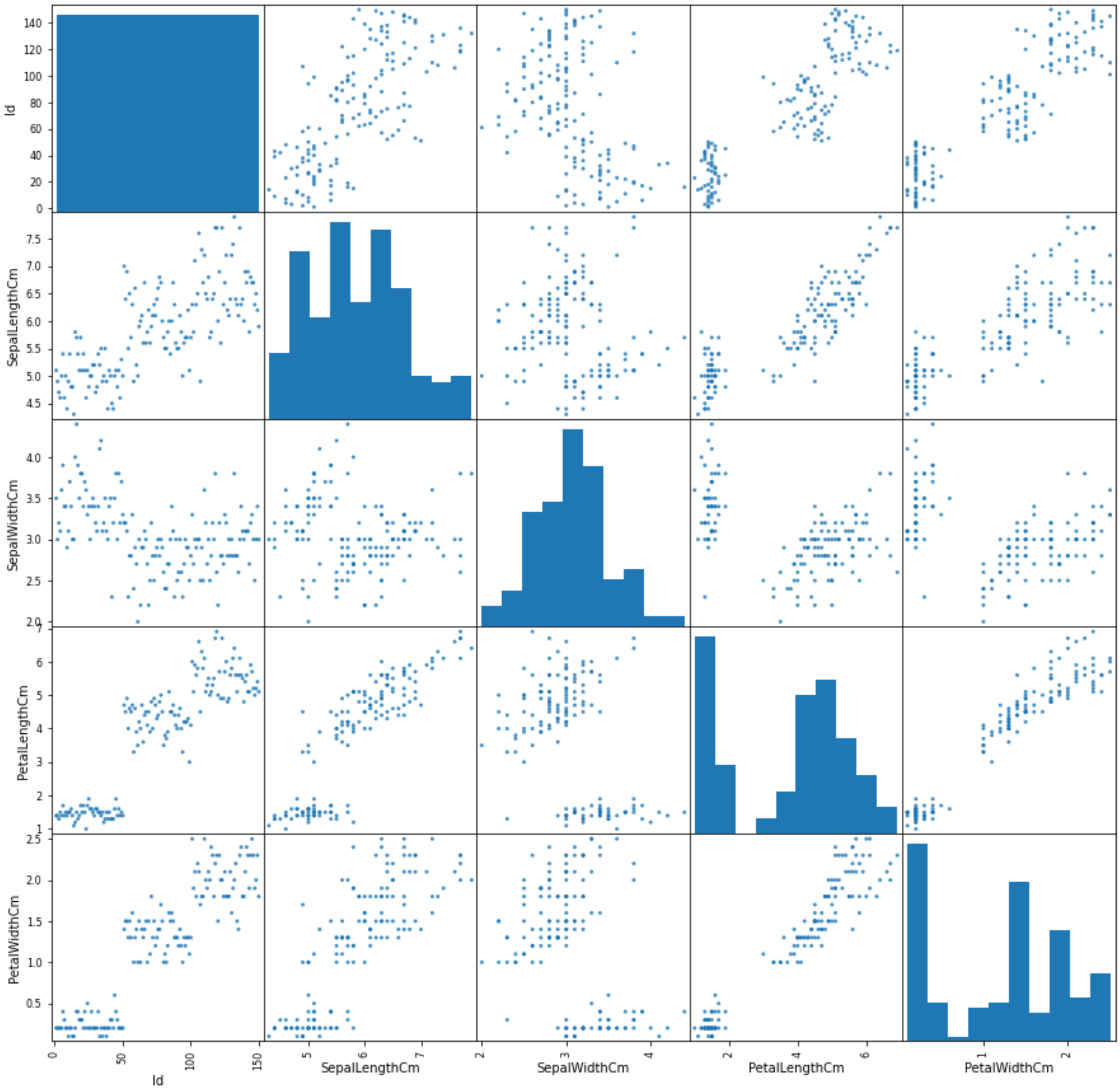
In [15]:

```
df.plot(kind='box', subplots=True, layout=(3,3), sharex=False,
figsize=(12,8))
pyplot.show()
```



In [16]:

```
scatter_matrix(df, alpha=0.8, figsize=(15, 15))
pyplot.show()
```



In [18]:

```
array = df.values
X = array[:,1:5]
Y = array[:,5]
```

In [19]:

```
test_size = 0.33
seed = 7
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=test_size, random_state=seed)
```

In [20]:

```
model = KNeighborsClassifier()
model.fit(X_train, Y_train)
predicted = model.predict(X_test)
report = classification_report(Y_test, predicted)
print("Classification Report: ", "\n", "\n", report)
```

```
Classification Report: 

                  precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       0.85      0.94      0.89        18
 Iris-virginica       0.94      0.83      0.88        18

       accuracy                           0.92        50
      macro avg       0.93      0.93      0.93        50
   weighted avg       0.92      0.92      0.92        50
```

In [21]:

```
result = model.score(X_test, Y_test)
print(("Accuracy: %.3f%%") % (result*100.0))
```

```
Accuracy: 92.000%
```

In [22]:

```
model.predict([[5.3, 3.0, 4.5, 1.5]])
```

Out[22]:

```
array(['Iris-versicolor'], dtype=object)
```

In [23]:

```
n_splits = 10
seed = 7
```

In [27]:

```
kfold = KFold(n_splits, random_state=seed, shuffle=True)
scoring = 'accuracy'
results = cross_val_score(model, X, Y, cv=kfold, scoring=scoring)
```

In [28]:

```
print("Accuracy: %.3f (%.3f)" % (results.mean(), results.std()))
```

```
Accuracy: 0.953 (0.052)
```

In [ ]: