In [2]:
```
import pandas as pd
import numpy as np
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.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn import datasets
```

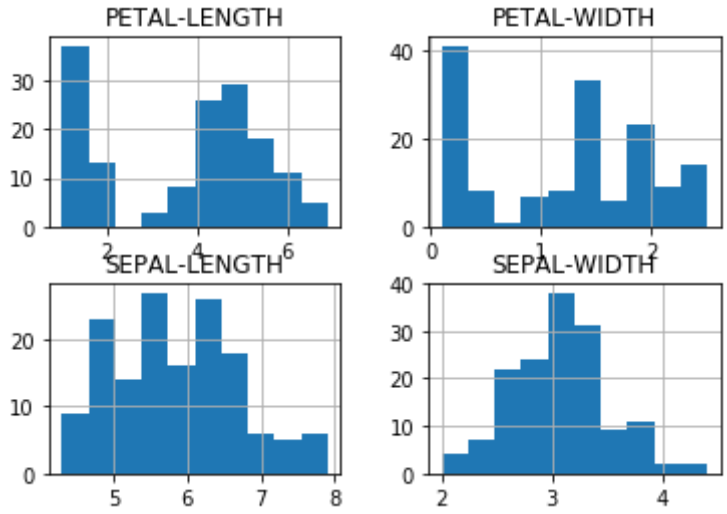
In [9]:
```
# Loading the dataset
path = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['SEPAL-LENGTH', 'SEPAL-WIDTH', 'PETAL-LENGTH', 'PETAL-WIDTH', 'CLASS']
data = pd.read_csv(path, names=names)
```

In [6]:
```
data.head(10)
```

Out[6]:

	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

In [10]:
```
data.hist()
plt.show()
```



In [12]:
```
# Converting into array and splitting
arr= data.values
x = arr[:,0:4]
y = arr[:,4]
validation_size=0.30
seed=11
x_train, x_test, y_train, y_test = model_selection.train_test_split(x, y
, test_size=validation_size, random_state=seed)
```

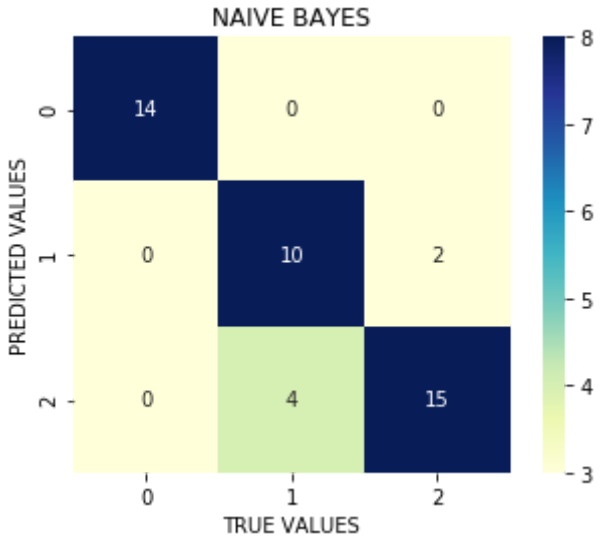
In [14]:
```
seed = 11
scoring = 'accuracy'
nw = []
nw.append(('Linear Discriminant Analysis', LinearDiscriminantAnalysis
()))
nw.append(('Naive Bayes', GaussianNB()))
# evaluate each model in turn
result = []
names = []
for name, model in nw:
 kfold = model_selection.KFold(n_splits=10, random_state=seed)
 cv_results = model_selection.cross_val_score(model, x_train, y_train, c
v=kfold, scoring=scoring)
 result.append(cv_results)
 names.append(name)
 msg = "%s: %f " % (name, cv_results.mean())
 print(msg)
```

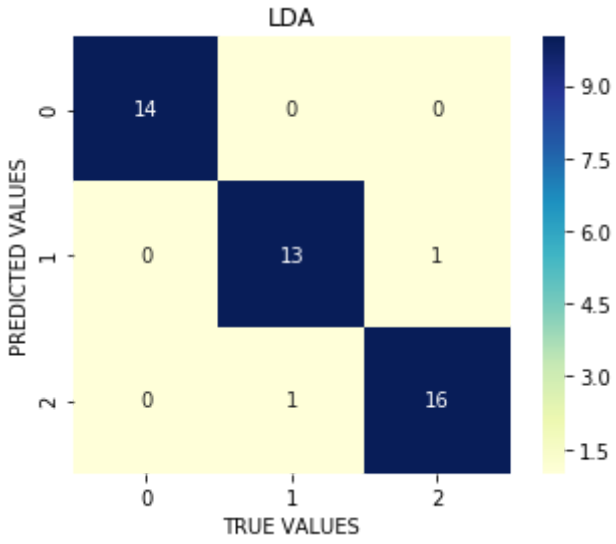
```
Linear Discriminant Analysis: 0.980909
Naive Bayes: 0.961818
```

In [18]:
```
# Test data
nw1 = GaussianNB()
nw1.fit(x_train, y_train)
predic = nw1.predict(x_test)
msg = "%s: %f " % ("Naive Bayes", accuracy_score(y_test, predic))
print(msg)
print(confusion_matrix(y_test, predic))
print(classification_report(y_test, predic))

lin= LinearDiscriminantAnalysis()
lin.fit(x_train, y_train)
predic1 = lin.predict(x_test)
msg = "%s: %f " % ("LDA", accuracy_score(y_test, predic1))
print(msg)
print(accuracy_score(y_test, predic1))
print(confusion_matrix(y_test, predic1))
print(classification_report(y_test, predic1))
```

```
Naive Bayes: 0.866667
[[14  0  0]
 [ 0 10  4]
 [ 0  2 15]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       0.83      0.71      0.77        14
 Iris-virginica       0.79      0.88      0.83        17

    avg / total       0.87      0.87      0.87        45

LDA: 0.955556
0.955555555556
[[14  0  0]
 [ 0 13  1]
 [ 0  1 16]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       0.93      0.93      0.93        14
 Iris-virginica       0.94      0.94      0.94        17

    avg / total       0.96      0.96      0.96        45
```

In [21]:
```
#Accuracy by confusion matrix

from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
cm = confusion_matrix(y_test, predic)
sns.heatmap(cm.T, square=True, annot=True, fmt='d', cbar=True, cmap="YlG
nBu",vmin=3,vmax=8)
plt.title("NAIVE BAYES")
plt.xlabel('TRUE VALUES')
plt.ylabel('PREDICTED VALUES')
plt.show()

cm = confusion_matrix(y_test, predic1)
sns.heatmap(cm.T, square=True, annot=True, fmt='d', cbar=True, cmap
="YlGnBu",vmin=1,vmax=10)
plt.title("LDA")
plt.xlabel('TRUE VALUES')
plt.ylabel('PREDICTED VALUES')
plt.show()
```





In [ ]:
```
#LDA performs better than Naive Bayes when clusters have large differnce
 between their means and small spreadout
#amongst themselves. LDA shows accuracy of 95.5% as compared to 86.6% in
 Naive Bayes.
```