In [11]:

```
#Load required packages
import pandas as pd
import numpy as np
from sklearn.externals import joblib
from sklearn import svm, datasets

from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier

from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

import warnings
warnings.filterwarnings('ignore')
```

In [12]:

```
iris = datasets.load_iris() # import data to play with
#To understand the data size, variables and class distribution

print ("Iris data set Description : ", iris['DESCR'])
```

Iris data set Description : .. _iris_dataset:

Iris plants dataset

Data Set Characteristics:

:Number of Instances: 150 (50 in each of three classes)
:Number of Attributes: 4 numeric, predictive attributes and the class
:Attribute Information:
- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- class:
    - Iris-Setosa
    - Iris-Versicolour
    - Iris-Virginica

:Summary Statistics:

	Min	Max	Mean	SD	Class Correlation
sepal length:	4.3	7.9	5.84	0.83	0.7826
sepal width:	2.0	4.4	3.05	0.43	-0.4194
petal length:	1.0	6.9	3.76	1.76	0.9490 (high!)
petal width:	0.1	2.5	1.20	0.76	0.9565 (high!)

:Missing Attribute Values: None
:Class Distribution: 33.3% for each of 3 classes.
:Creator: R.A. Fisher
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
:Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken from Fisher's paper. Note that it's the same as in R, but not as in the UCI Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

.. topic:: References

- Fisher, R.A. "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to Mathematical Statistics" (John Wiley, NY, 1950).
- Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis. (Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1. See page 218.
- Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System Structure and Classification Rule for Recognition in Partially Exposed Environments". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 1, 67-71.
- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions on Information Theory, May 1972, 431-433.
- See also: 1988 MLC Proceedings, 54-64. Cheeseman et al"s AUTOCLASS II conceptual clustering system finds 3 classes in the data.
- Many, many more ...

In [13]:
```
X = iris.data                    #Features
y = iris.target                   #Target variable
```

In [14]:
```
#split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
=10)
```

In [15]:
```
#Using SVM classifier
model = SVC(kernel='linear').fit(X_train, y_train)

#Calculate Test Prediction
y_pred = model.predict(X_test)
print(model.score(X_test,y_test.ravel()))
```

```
0.9666666666666667
```

In [16]:
```
# trying Random forest
model = RandomForestClassifier()
model.fit(X_train, y_train)
predictedvalues=model.predict(X_test)
print(accuracy_score(y_test,predictedvalues))
print(confusion_matrix(y_test, predictedvalues))
```

```
0.9666666666666667
[[10  0  0]
 [ 0 12  1]
 [ 0  0  7]]
```

In [17]:
```
# trying Adaboost
model = AdaBoostClassifier()
model.fit(X_train, y_train)
predictedvalues=model.predict(X_test)
print(accuracy_score(y_test,predictedvalues))
print(confusion_matrix(y_test, predictedvalues))
```

```
0.9666666666666667
[[10  0  0]
 [ 0 13  0]
 [ 0  1  6]]
```

In [18]:
```python
# trying XGboost
import xgboost as xgb
dtrain = xgb.DMatrix(X_train, label=y_train)
dtest = xgb.DMatrix(X_test, label=y_test)

from sklearn.datasets import dump_svmlight_file

dump_svmlight_file(X_train, y_train, 'dtrain.svm', zero_based=True)
dump_svmlight_file(X_test, y_test, 'dtest.svm', zero_based=True)
dtrain_svm = xgb.DMatrix('dtrain.svm')
dtest_svm = xgb.DMatrix('dtest.svm')

param = {
    'max_depth': 3,  # the maximum depth of each tree
    'eta': 0.3,  # the training step for each iteration
    'silent': 1,  # logging mode - quiet
    'objective': 'multi:softprob',  # error evaluation for multiclass training
    'num_class': 3}  # the number of classes that exist in this datset
num_round = 20  # the number of training iterations

bst = xgb.train(param, dtrain, num_round)
bst.dump_model('dump.raw.txt')

preds = bst.predict(dtest)

print(preds)
```

```
[10:35:28] 120x4 matrix with 480 entries loaded from dtrain.svm
[10:35:28] 30x4 matrix with 120 entries loaded from dtest.svm
[[0.00920194 0.97496223 0.01583584]
 [0.00293817 0.00471791 0.99234396]
 [0.990071   0.00668957 0.00323948]
 [0.00475869 0.98826444 0.00697682]
 [0.99074113 0.00601726 0.00324168]
 [0.01991933 0.9501693  0.02991132]
 [0.0142208  0.6243776  0.36140156]
 [0.00531657 0.98688865 0.00779475]
 [0.99074113 0.00601726 0.00324168]
 [0.00490279 0.98790914 0.0071881 ]
 [0.00438265 0.9824377  0.0131797 ]
 [0.00377201 0.02454171 0.97168636]
 [0.00854078 0.9767612  0.01469803]
 [0.9890463  0.00600697 0.00494677]
 [0.99074113 0.00601726 0.00324168]
 [0.00293917 0.00437896 0.9926819 ]
 [0.00437153 0.9904165  0.00521199]
 [0.9890463  0.00600697 0.00494677]
 [0.99074113 0.00601726 0.00324168]
 [0.99074113 0.00601726 0.00324168]
 [0.00293917 0.00437896 0.9926819 ]
 [0.00293817 0.00471791 0.99234396]
 [0.01137183 0.20873266 0.7798955 ]
 [0.99074113 0.00601726 0.00324168]
 [0.00438265 0.9824377  0.0131797 ]
 [0.99074113 0.00601726 0.00324168]
 [0.00441245 0.98911834 0.0064692 ]
 [0.00530938 0.98555356 0.00913703]
 [0.00653294 0.98156077 0.01190623]
 [0.00396643 0.00590945 0.99012417]]
```

In [19]:

```python
import numpy as np
best_preds = np.asarray([np.argmax(line) for line in preds])

mean_train=y_train.mean()
base_prediction=np.ones(y_test.shape)*mean_train

#compute MAE
from sklearn.metrics import mean_absolute_error
mae =mean_absolute_error(y_test ,base_prediction)
print("mae baseline:",mae)
```

```
mae baseline: 0.5799999999999998
```

In [23]:

```python
# save model
joblib.dump(model, 'model/iris_svm_model.pkl', compress=True)
```

Out[23]:

```
['model/iris_svm_model.pkl']
```