In [2]:

```
import pandas as pd

data = pd.read_csv('6.vertebrate.csv',header = 'infer')
data
```

Out[2]:

	Name	Warm-blooded	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
0	human	1	1	0	0	1	0	mammals
1	python	0	0	0	0	0	1	reptiles
2	salmon	0	0	1	0	0	0	fishes
3	whale	1	1	1	0	0	0	mammals
4	frog	0	0	1	0	1	1	amphibians
5	komodo	0	0	0	0	1	0	reptiles
6	bat	1	1	0	1	1	1	mammals
7	pigeon	1	0	0	1	1	0	birds
8	cat	1	1	0	0	1	0	mammals
9	leopard shark	0	1	1	0	0	0	fishes
10	turtle	0	0	1	0	1	0	reptiles
11	penguin	1	0	1	0	1	0	birds
12	porcupine	1	1	0	0	1	1	mammals
13	eel	0	0	1	0	0	0	fishes
14	salamander	0	0	1	0	1	1	amphibians

In [3]:

```
import pandas as pd

data = pd.read_csv('6.vertebrate.csv',header = 'infer')
data['Class']=data['Class'].replace(['fishes','birds','amphibians','reptiles'],'non-mammals')
data
```

Out[3]:

	Name	Warm-blooded	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
0	human	1	1	0	0	1	0	mammals
1	python	0	0	0	0	0	1	non-mammals
2	salmon	0	0	1	0	0	0	non-mammals
3	whale	1	1	1	0	0	0	mammals
4	frog	0	0	1	0	1	1	non-mammals
5	komodo	0	0	0	0	1	0	non-mammals
6	bat	1	1	0	1	1	1	mammals
7	pigeon	1	0	0	1	1	0	non-mammals
8	cat	1	1	0	0	1	0	mammals
9	leopard shark	0	1	1	0	0	0	non-mammals
10	turtle	0	0	1	0	1	0	non-mammals
11	penguin	1	0	1	0	1	0	non-mammals
12	porcupine	1	1	0	0	1	1	mammals
13	eel	0	0	1	0	0	0	non-mammals
14	salamander	0	0	1	0	1	1	non-mammals

In [5]:

```
import pandas as pd

data = pd.read_csv('6.vertebrate.csv',header = 'infer')
data['Class']=data['Class'].replace(['fishes','birds','amphibians','reptiles'],'non-mammals')
pd.crosstab([data['Warm-blooded'],data['Gives Birth']],data['Class'])
```

Out[5]:

	Class	mammals	non-mammals
Warm-blooded	**Gives Birth**		
0	**0**	0	7
	1	0	1
1	**0**	0	2
	1	5	0

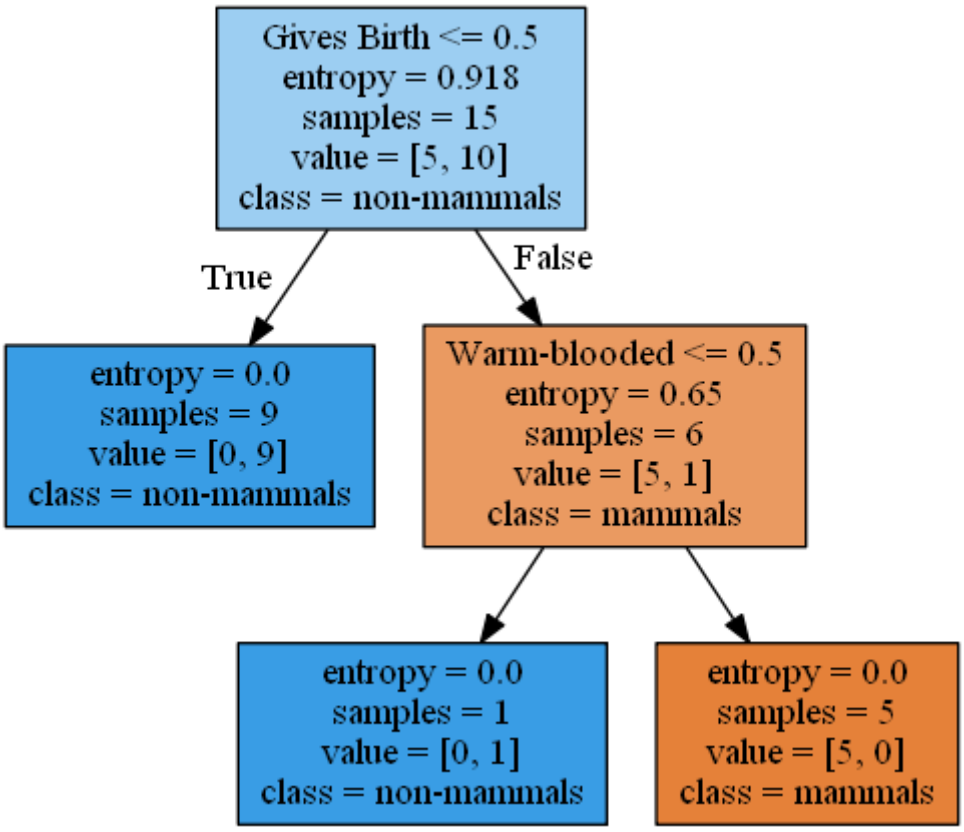
In [8]:

```
import pandas as pd
from sklearn import tree

data = pd.read_csv('6.vertebrate.csv',header = 'infer')
data['Class']=data['Class'].replace(['fishes','birds','amphibians','reptiles'],'non-mammals')

Y = data['Class']
X = data.drop(['Name','Class'],axis=1)

clf = tree.DecisionTreeClassifier(criterion='entropy',max_depth=3)
clf = clf.fit(X,Y)
```

In [1]:

```
import pandas as pd
from sklearn import tree
import pydotplus
from IPython.display import Image

data = pd.read_csv('6.vertebrate.csv',header = 'infer')
data['Class']=data['Class'].replace(['fishes','birds','amphibians','reptiles'],'non-mammals')

Y = data['Class']
X = data.drop(['Name','Class'],axis=1)

clf = tree.DecisionTreeClassifier(criterion='entropy',max_depth=3)
clf = clf.fit(X,Y)

dot_data = tree.export_graphviz(clf, feature_names=X.columns, class_names=['mammals','non-mammal
s'],filled=True, out_file=None)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

Out[1]:



In [3]:

```
import pandas as pd

data = pd.read_csv('6.vertebrate.csv',header = 'infer')
data['Class']=data['Class'].replace(['fishes','birds','amphibians','reptiles'],'non-mammals')

testData = [['gila monster',0,0,0,0,1,1,'non-mammals'],['platypus',1,0,0,0,1,1,'mammals'],
            ['owl',1,0,0,1,1,0,'non-mammals'],['dolphin',1,1,1,0,0,0,'mammals']]
testData = pd.DataFrame(testData,columns=data.columns)
testData
```

Out[3]:

	Name	Warm-blooded	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
0	gila monster	0	0	0	0	1	1	non-mammals
1	platypus	1	0	0	0	1	1	mammals
2	owl	1	0	0	1	1	0	non-mammals
3	dolphin	1	1	1	0	0	0	mammals

In [4]:

```
import pandas as pd

data = pd.read_csv('6.vertebrate.csv',header = 'infer')
data['Class']=data['Class'].replace(['fishes','birds','amphibians','reptiles'],'non-mammals')

testData = [['gila monster',0,0,0,0,1,1,'non-mammals'],['platypus',1,0,0,0,1,1,'mammals'],
            ['owl',1,0,0,1,1,0,'non-mammals'],['dolphin',1,1,1,0,0,0,'mammals']]
testData = pd.DataFrame(testData,columns=data.columns)

testY = testData['Class']
testX = testData.drop(['Name','Class'],axis=1)

predY = clf.predict(testX)
predictions = pd.concat([testData['Name'],pd.Series(predY,name='Predicted Class')],axis=1)
predictions
```

Out[4]:

	Name	Predicted Class
0	gila monster	non-mammals
1	platypus	non-mammals
2	owl	non-mammals
3	dolphin	mammals

In [5]:

```
import pandas as pd
from sklearn.metrics import accuracy_score

data = pd.read_csv('6.vertebrate.csv',header = 'infer')
data['Class']=data['Class'].replace(['fishes','birds','amphibians','reptiles'],'non-mammals')

testData = [['gila monster',0,0,0,0,1,1,'non-mammals'],['platypus',1,0,0,0,1,1,'mammals'],
           ['owl',1,0,0,1,1,0,'non-mammals'],['dolphin',1,1,1,0,0,0,'mammals']]
testData = pd.DataFrame(testData,columns=data.columns)

testY = testData['Class']
testX = testData.drop(['Name','Class'],axis=1)

predY = clf.predict(testX)
predictions = pd.concat([testData['Name'],pd.Series(predY,name='Predicted Class')],axis=1)

print('Accuracy on test data is %.2f' % (accuracy_score(testY,predY)))
```

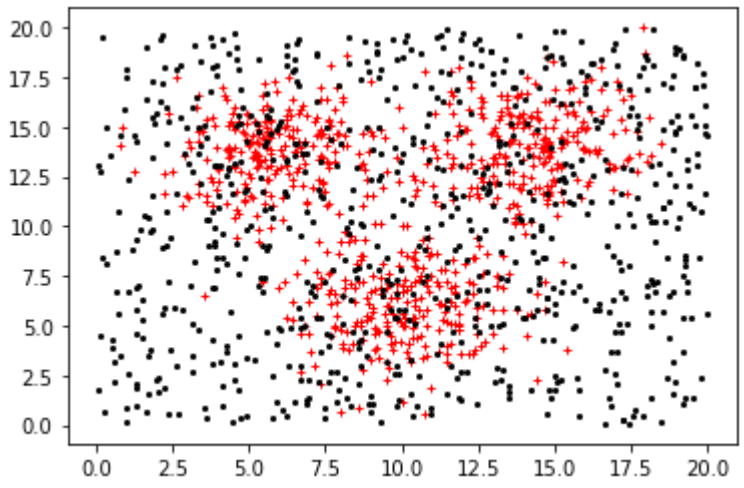
```
Accuracy on test data is 0.75
```

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.random import random

%matplotlib inline

N = 1500

mean1 = [6,14]
mean2 = [10,6]
mean3 = [14,14]
cov = [[3.5,0],[0,3.5]]  #diagonal covariance

np.random.seed(50)
X = np.random.multivariate_normal(mean1,cov,int(N/6))
X = np.concatenate((X,np.random.multivariate_normal(mean2,cov,int(N/6))))
X = np.concatenate((X,np.random.multivariate_normal(mean3,cov,int(N/6))))
X = np.concatenate((X,20*np.random.rand(int(N/2),2)))
Y = np.concatenate((np.ones(int(N/2)),np.zeros(int(N/2))))

plt.plot(X[:int(N/2),0],X[:int(N/2),1],'r+',X[int(N/2):,0],X[int(N/2):,1],'k.',ms=4)
```

Out[1]:

```
[<matplotlib.lines.Line2D at 0x1bf48cfd3c8>,
 <matplotlib.lines.Line2D at 0x1bf4def4ec8>]
```

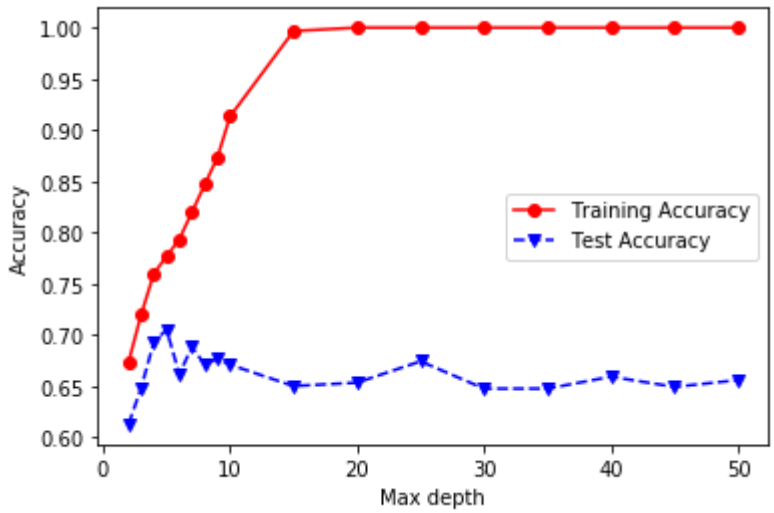


In [5]:

```
#####################################################
#Training and Test set creation
#####################################################

from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.8,random_state=1)

from sklearn import tree
from sklearn.metrics import accuracy_score

#####################################################
# Model fitting and evaluatio
#####################################################

maxdepths = [2,3,4,5,6,7,8,9,10,15,20,25,30,35,40,45,50]

trainAcc = np.zeros(len(maxdepths))
testAcc = np.zeros(len(maxdepths))

index = 0
for depth in maxdepths:
    clf = tree.DecisionTreeClassifier(max_depth=depth)
    clf = clf.fit(X_train,Y_train)
    Y_predTrain = clf.predict(X_train)
    Y_predTest = clf.predict(X_test)
    trainAcc[index] = accuracy_score(Y_train,Y_predTrain)
    testAcc[index] = accuracy_score(Y_test,Y_predTest)
    index+=1

#################################################
# Plot of training and test accuracies
#################################################

plt.plot(maxdepths,trainAcc,'ro-',maxdepths,testAcc,'bv--')
plt.legend(['Training Accuracy','Test Accuracy'])
plt.xlabel('Max depth')
plt.ylabel('Accuracy')
```

Out[5]:

```
Text(0, 0.5, 'Accuracy')
```

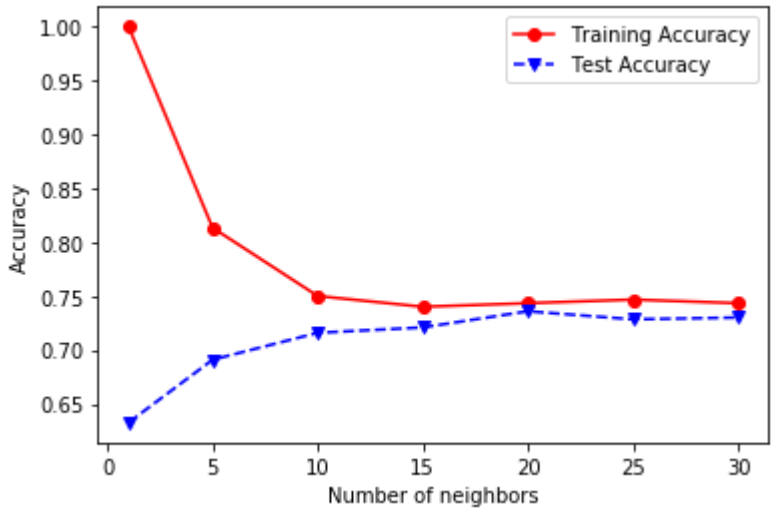


In [6]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.8,random_state=1)
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
%matplotlib inline

numNeighbors = [1,5,10,15,20,25,30]
trainAcc = []
testAcc = []

for k in numNeighbors:
    clf = KNeighborsClassifier(n_neighbors=k, metric='minkowski',p=2)
    clf.fit(X_train,Y_train)
    Y_predTrain = clf.predict(X_train)
    Y_predTest = clf.predict(X_test)
    trainAcc.append(accuracy_score(Y_train,Y_predTrain))
    testAcc.append(accuracy_score(Y_test,Y_predTest))

plt.plot(numNeighbors,trainAcc,'ro-',numNeighbors,testAcc,'bv--')
plt.legend(['Training Accuracy','Test Accuracy'])
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
```

Out[6]:

```
Text(0, 0.5, 'Accuracy')
```

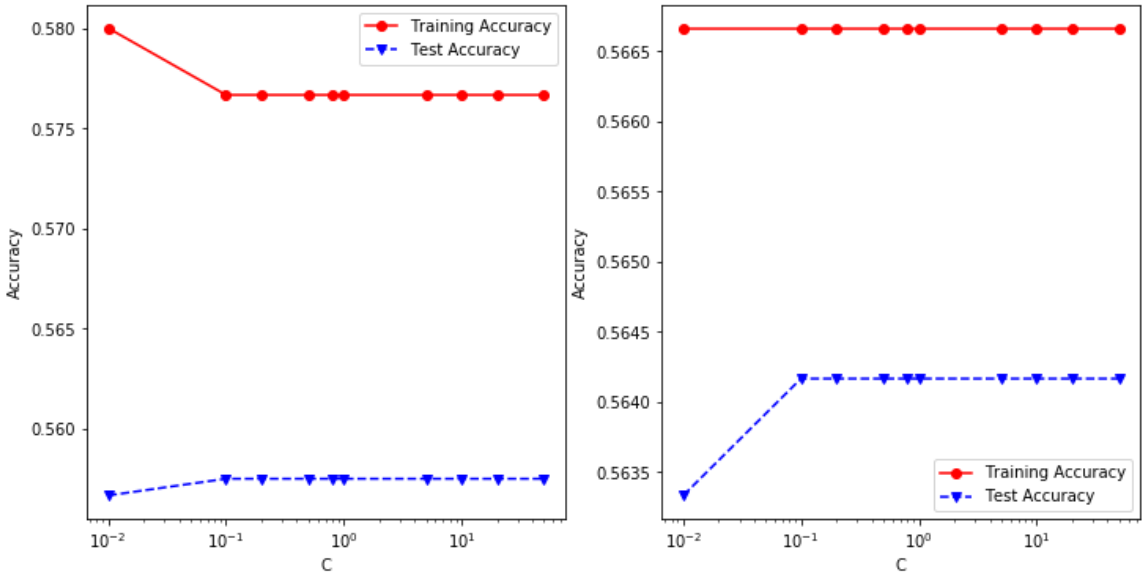


In [13]:

```
from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.8,random_state=1)

from sklearn import tree
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.svm import SVC
%matplotlib inline

C = [0.01,0.1,0.2,0.5,0.8,1,5,10,20,50]
LRtrainAcc = []
LRtestAcc = []
SVMtrainAcc = []
SVMtestAcc = []

for param in C:
    clf = linear_model.LogisticRegression(C=param)
    clf.fit(X_train,Y_train)
    Y_predTrain = clf.predict(X_train)
    Y_predTest = clf.predict(X_test)
    LRtrainAcc.append(accuracy_score(Y_train,Y_predTrain))
    LRtestAcc.append(accuracy_score(Y_test,Y_predTest))

    clf = SVC(C=param,kernel='linear')
    clf.fit(X_train,Y_train)
    Y_predTrain = clf.predict(X_train)
    Y_predTest = clf.predict(X_test)
    SVMtrainAcc.append(accuracy_score(Y_train,Y_predTrain))
    SVMtestAcc.append(accuracy_score(Y_test,Y_predTest))

fig, (ax1,ax2) = plt.subplots(1,2,figsize=(12,6))
ax1.plot(C,LRtrainAcc,'ro-',C,LRtestAcc,'bv--')
ax1.legend(['Training Accuracy','Test Accuracy'])
ax1.set_xlabel('C')
ax1.set_xscale('log')
ax1.set_ylabel('Accuracy')

ax2.plot(C,SVMtrainAcc,'ro-',C,SVMtestAcc,'bv--')
ax2.legend(['Training Accuracy','Test Accuracy'])
ax2.set_xlabel('C')
ax2.set_xscale('log')
ax2.set_ylabel('Accuracy')
```

Out[13]:

```
Text(0, 0.5, 'Accuracy')
```

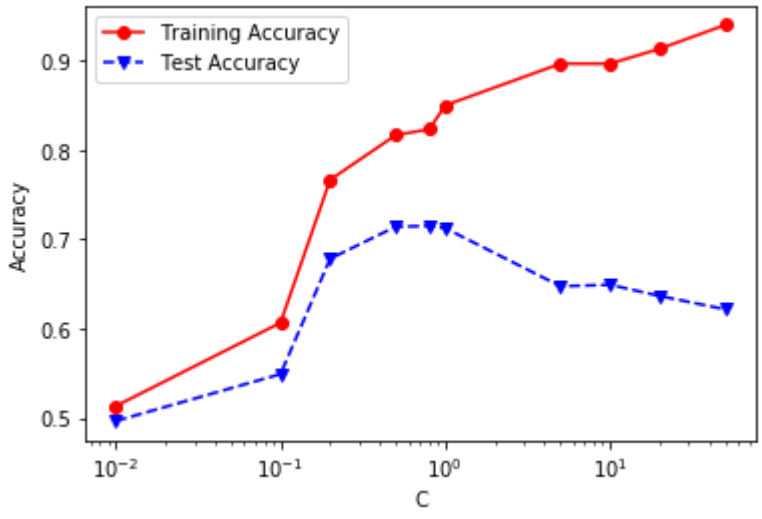


In [14]:

```
from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.8,random_state=1)

from sklearn import tree
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.svm import SVC
%matplotlib inline

C = [0.01,0.1,0.2,0.5,0.8,1,5,10,20,50]
SVMtrainAcc = []
SVMtestAcc = []

for param in C:
    clf = SVC(C=param,kernel='rbf',gamma='auto')
    clf.fit(X_train,Y_train)
    Y_predTrain = clf.predict(X_train)
    Y_predTest = clf.predict(X_test)
    SVMtrainAcc.append(accuracy_score(Y_train,Y_predTrain))
    SVMtestAcc.append(accuracy_score(Y_test,Y_predTest))

plt.plot(C,SVMtrainAcc,'ro-',C,SVMtestAcc,'bv--')
plt.legend(['Training Accuracy','Test Accuracy'])
plt.xlabel('C')
plt.xscale('log')
plt.ylabel('Accuracy')
```

Out[14]:

```
Text(0, 0.5, 'Accuracy')
```

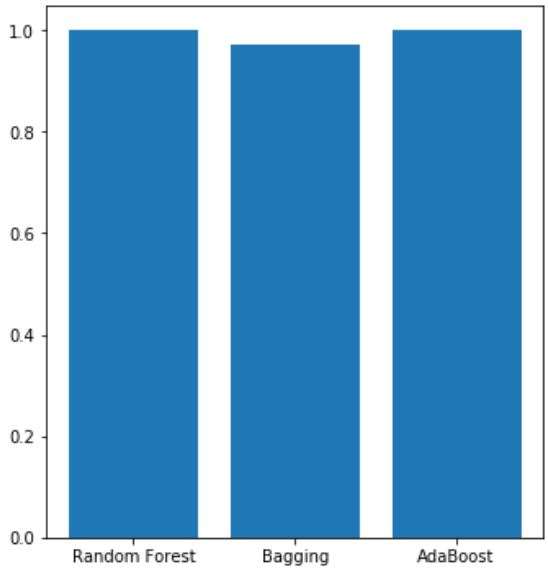


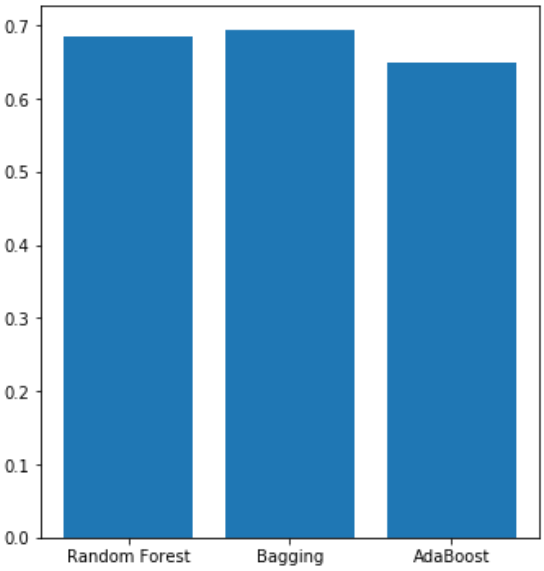
In [19]:

```
from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.8,random_state=1)

from sklearn import tree
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.svm import SVC
%matplotlib inline
from sklearn import ensemble
from sklearn.tree import DecisionTreeClassifier

numBaseClassifiers = 500
maxdepth = 10
trainAcc = []
testAcc = []

clf = ensemble.RandomForestClassifier(n_estimators=numBaseClassifiers)
clf.fit(X_train,Y_train)
Y_predTrain = clf.predict(X_train)
Y_predTest = clf.predict(X_test)
trainAcc.append(accuracy_score(Y_train,Y_predTrain))
testAcc.append(accuracy_score(Y_test,Y_predTest))

clf = ensemble.BaggingClassifier(DecisionTreeClassifier(max_depth=maxdepth),n_estimators=numBase
Classifiers)
clf.fit(X_train,Y_train)
Y_predTrain = clf.predict(X_train)
Y_predTest = clf.predict(X_test)
trainAcc.append(accuracy_score(Y_train,Y_predTrain))
testAcc.append(accuracy_score(Y_test,Y_predTest))

clf = ensemble.AdaBoostClassifier(DecisionTreeClassifier(max_depth=maxdepth),n_estimators=numBas
eClassifiers)
clf.fit(X_train,Y_train)
Y_predTrain = clf.predict(X_train)
Y_predTest = clf.predict(X_test)
trainAcc.append(accuracy_score(Y_train,Y_predTrain))
testAcc.append(accuracy_score(Y_test,Y_predTest))

methods = ['Random Forest','Bagging','AdaBoost']
fig,(ax1,ax2) = plt.subplots(1,2,figsize=(12,6))
ax1.bar([1.5,2.5,3.5],trainAcc)
ax1.set_xticks([1.5,2.5,3.5])
ax1.set_xticklabels(methods)
ax2.bar([1.5,2.5,3.5],testAcc)
ax2.set_xticks([1.5,2.5,3.5])
ax2.set_xticklabels(methods)
```

Out[19]:

```
[Text(0, 0, 'Random Forest'), Text(0, 0, 'Bagging'), Text(0, 0, 'AdaBoost')]
```





In [ ]: