# Psychoinformatics - Week 10 (Exercises)

by boyonglin (r10945002@ntu.edu.tw)

In [ ]:
```
import warnings, numpy as np
import xgboost
from matplotlib.pyplot import *
%matplotlib inline
warnings.simplefilter('ignore', DeprecationWarning)
from sklearn import *
import copy
```

# 1 執行並觀察以下的機器學習結果 (2分)

## 1.0 IRIS dataset & Ensemble model function

In [ ]:
```
iris = datasets.load_iris()
X=iris.data
Y=iris.target
```

In [ ]:
```
np.random.seed(0)
sss=model_selection.StratifiedShuffleSplit(n_splits=5,test_size=0.1)
def EnsembleModels(og_model, Max_n_estimators):
    accs=[] # mean cross-validation accuracies of the models w/ different n_estimat
    for n in range(1,Max_n_estimators+1):
        if n%10 == 0:
            print('▓',end=' ') # showing progress
        acc=[] # cross-validation accuracies of the ensemble model w/ n_estimators=
        for train_index, test_index in sss.split(X, Y): # 5-fold cross-validation o
            X_train, X_test = X[train_index], X[test_index]
            Y_train, Y_test = Y[train_index], Y[test_index]
            model=copy.deepcopy(og_model) # to avoid possible model re-training
            model.n_estimators=n
            model.fit(X_train[:,0:2],Y_train) #training
            acc.append(model.predict(X_test[:,0:2])==Y_test)
        accs.append(np.mean(acc)) # aggregating mean cross-validation accuracies ac
    return(accs)
```

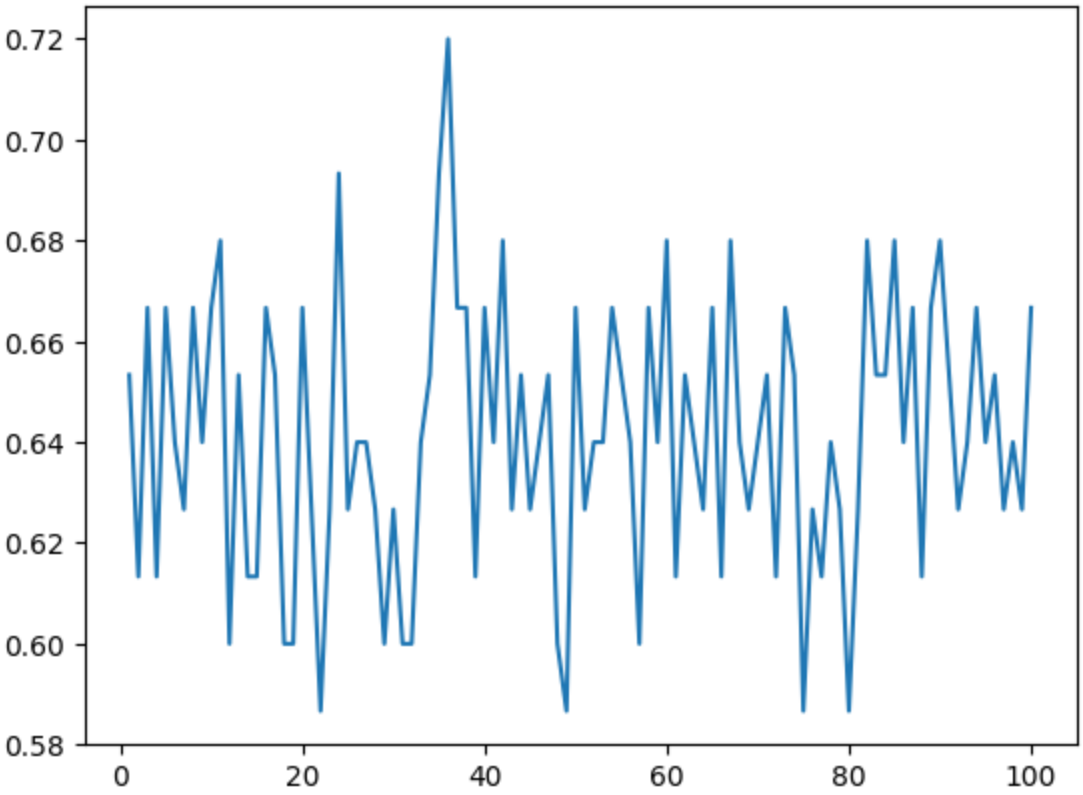
## 1.1 Bagging (Bootstrap Aggregating)

### 1.1.1 Tree max_depth = 1

In [ ]:
```
model=ensemble.BaggingClassifier(tree.DecisionTreeClassifier(max_depth=1))
accs=EnsembleModels(model,100)
```

```
plot(range(1,101),accs)
print('mean accuracy:',np.mean(accs))
```

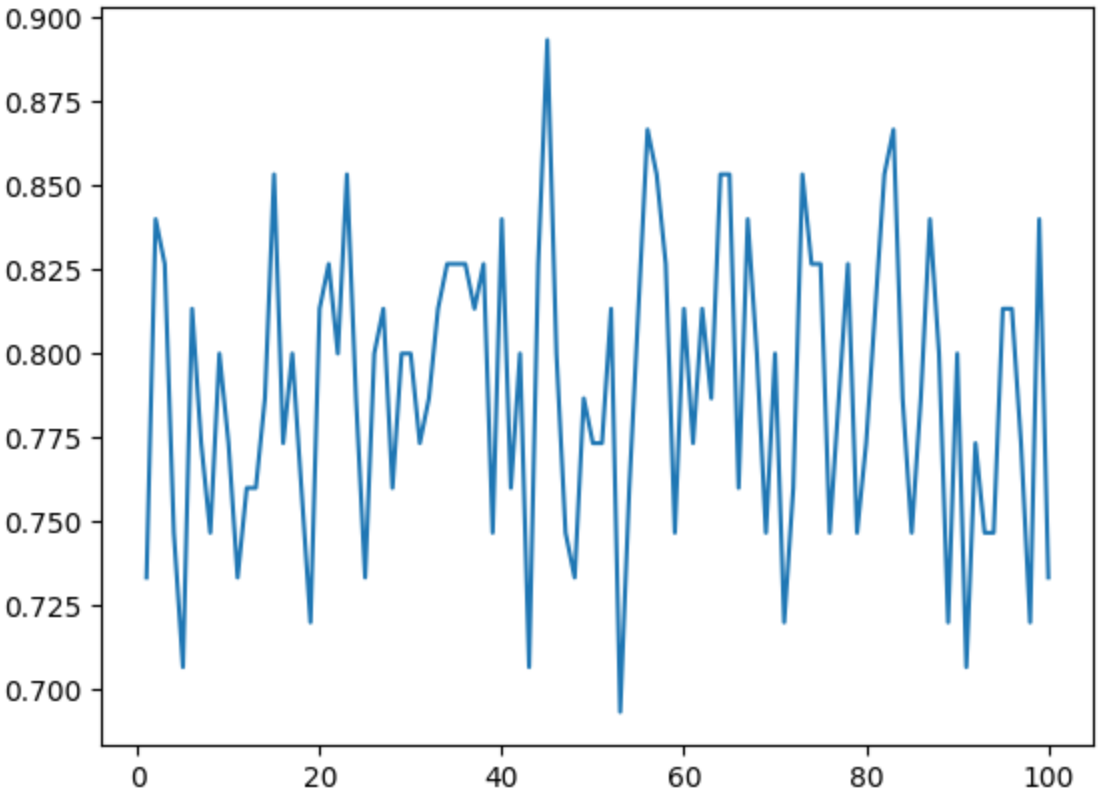
```
mean accuracy: 0.6410666666666667
```




### 1.1.2 Tree max_depth = 3

In [ ]:
```
model=ensemble.BaggingClassifier(tree.DecisionTreeClassifier(max_depth=3))
accs=EnsembleModels(model,100)
plot(range(1,101),accs)
print('mean accuracy:',np.mean(accs))
```

```
mean accuracy: 0.7893333333333333
```



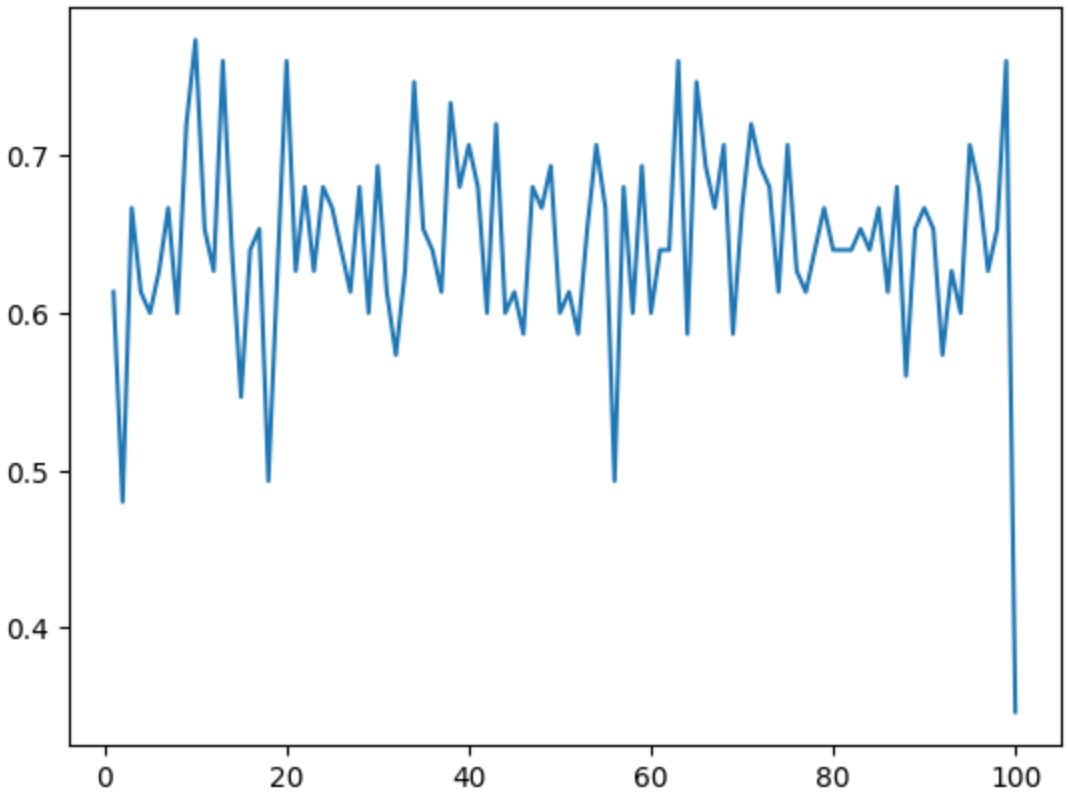

# 1.2 Boosting

## 1.2.1 AdaBoost

### 1.2.1.1 Tree max_depth = 1

In [ ]:
```
model=ensemble.AdaBoostClassifier(tree.DecisionTreeClassifier(max_depth=1))
accs=EnsembleModels(model,100)
plot(range(1,101),accs)
print('mean accuracy:',np.mean(accs))
```

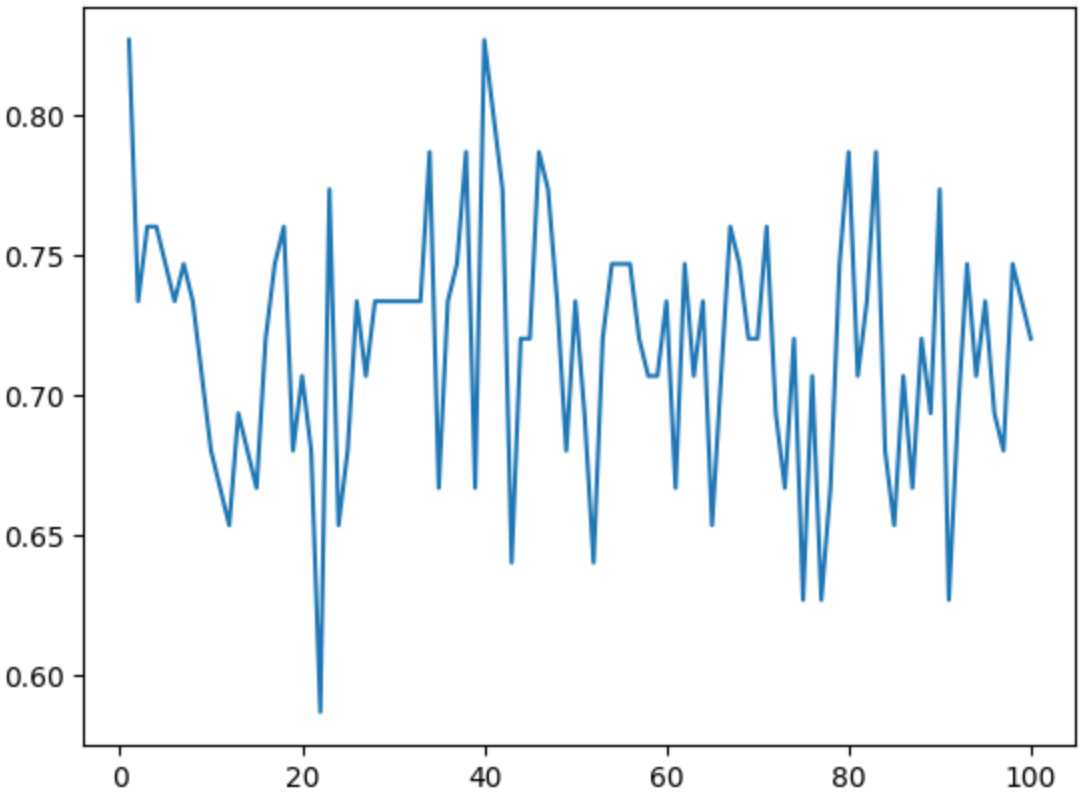
```
mean accuracy: 0.6452000000000001
```




#### 1.2.1.2 Tree max_depth = 3

In [ ]:
```
model=ensemble.AdaBoostClassifier(tree.DecisionTreeClassifier(max_depth=3))
accs=EnsembleModels(model,100)
plot(range(1,101),accs)
print('mean accuracy:',np.mean(accs))
```

```
mean accuracy: 0.7170666666666667
```




## 1.2.2 Gradient Boosting

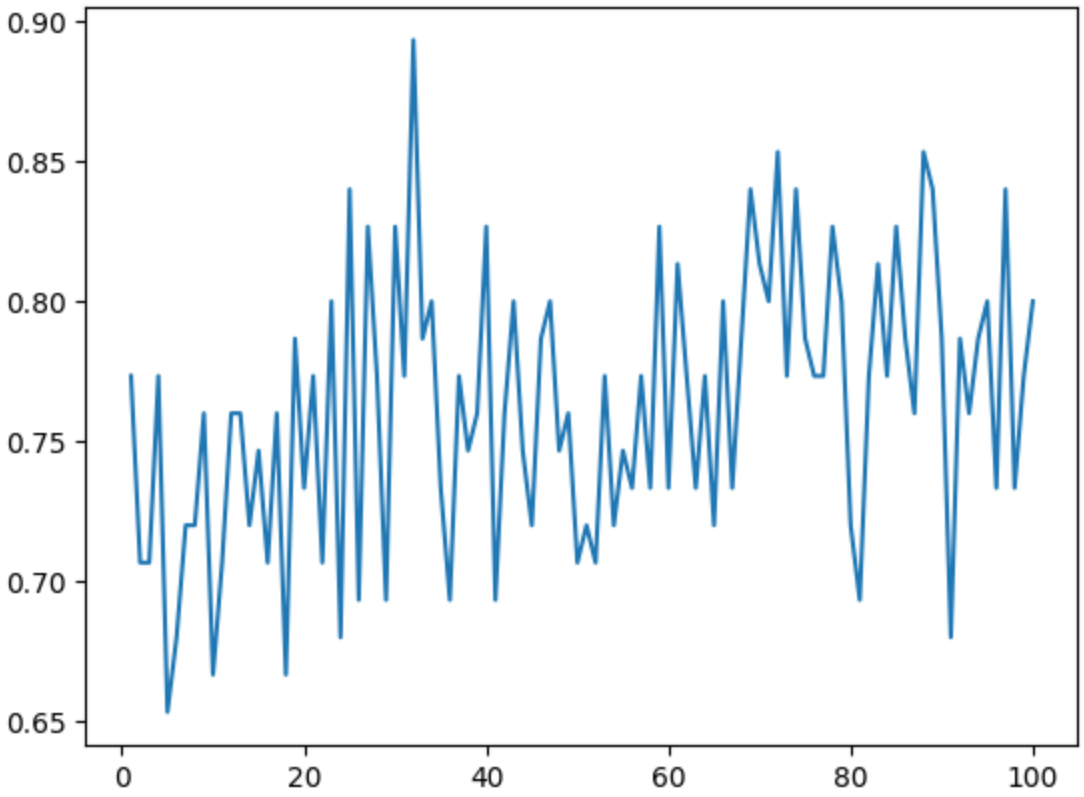
The following two implementations are conceptually identical but XGBoost is more resource-efficient and can be parallelized/distributed.

### 1.2.2.1 Scikit-learn's Gradient Tree Boosting

#### 1.2.2.1.1 Tree max_depth = 1

In [ ]:
```
model=ensemble.GradientBoostingClassifier(max_depth=1)
accs=EnsembleModels(model,100)
plot(range(1,101),accs)
print('mean accuracy:',np.mean(accs))
```

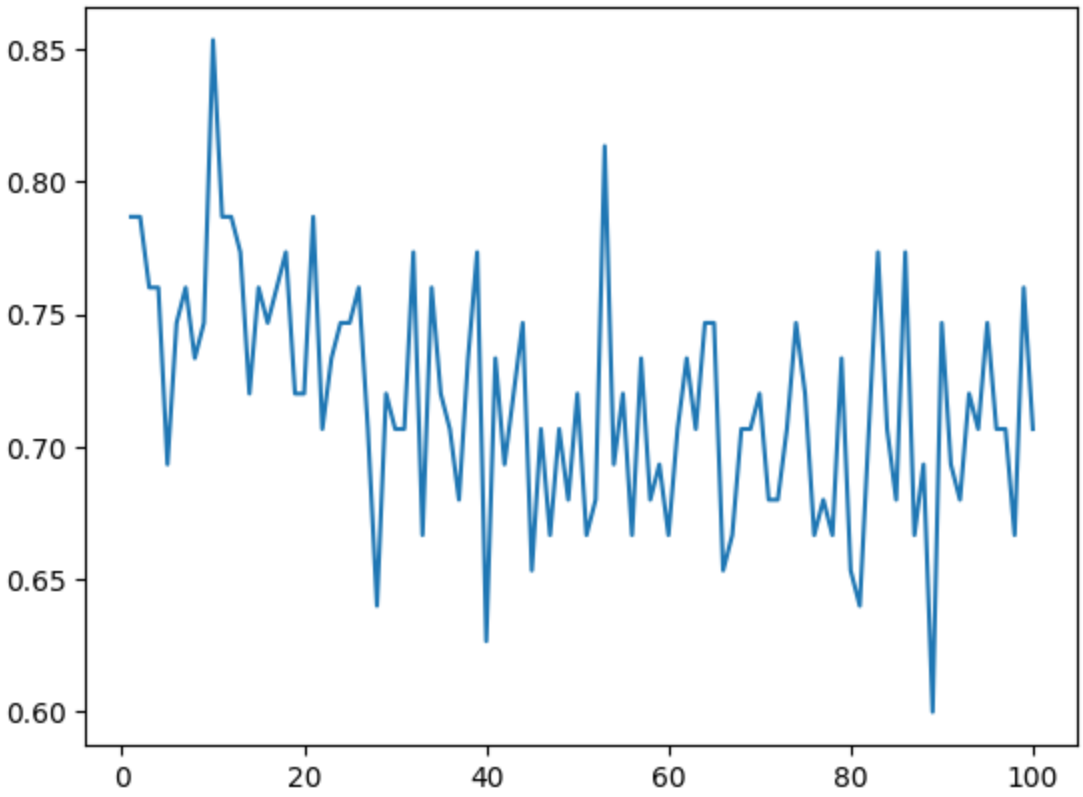
```
mean accuracy: 0.7622666666666666
```




#### 1.2.2.1.2 Tree max_depth = 3

In [ ]:
```
model=ensemble.GradientBoostingClassifier(max_depth=3)
accs=EnsembleModels(model,100)
plot(range(1,101),accs)
print('mean accuracy:',np.mean(accs))
```

```
mean accuracy: 0.7172
```

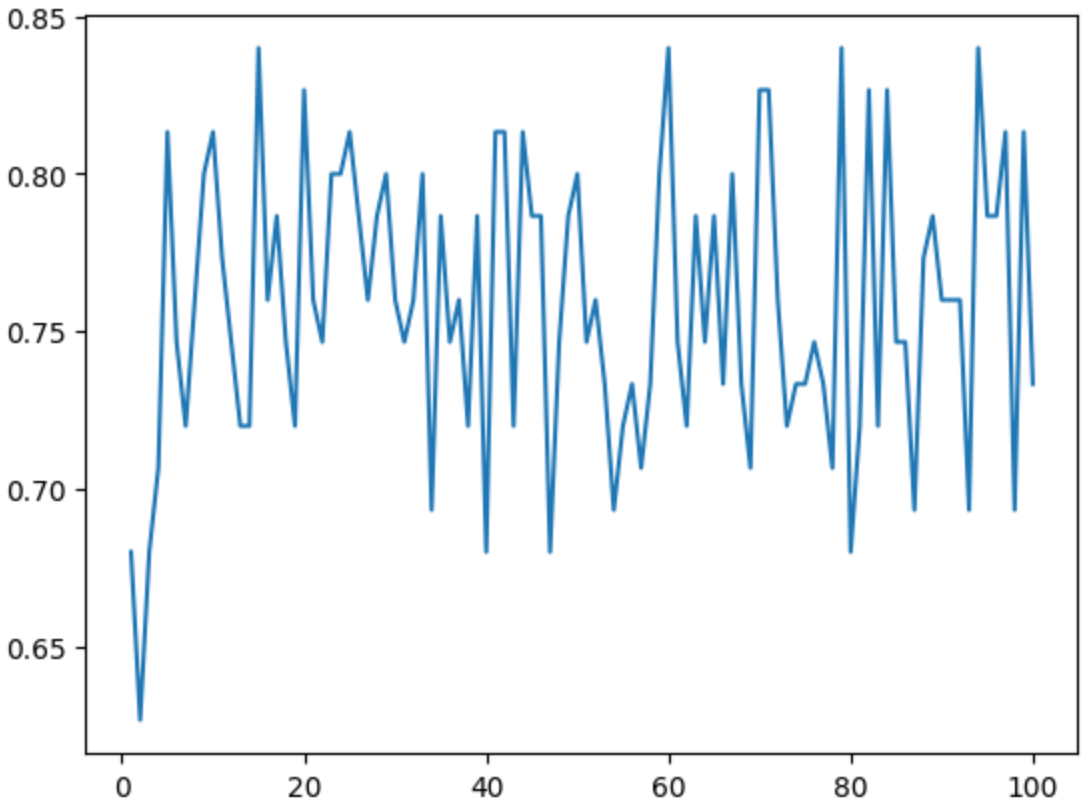



### 1.2.2.2 XGBoost (eXtreme Gradient Boosting)

#### 1.2.2.2.1 Tree max_depth = 1

In [ ]:
```
model=xgboost.XGBClassifier(max_depth=1)
accs=EnsembleModels(model,100)
plot(range(1,101),accs)
print('mean accuracy:',np.mean(accs))
```

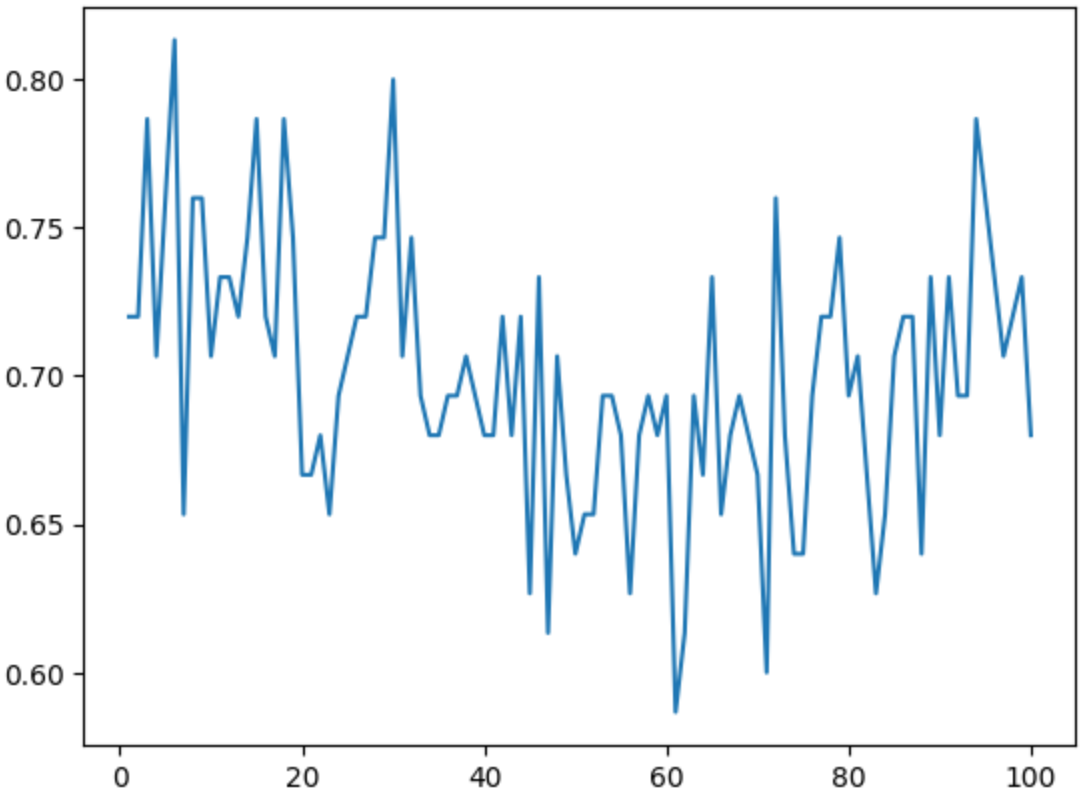
```
mean accuracy: 0.7583999999999999
```




1.2.2.2.2 Tree max_depth = 3

In [ ]:
```
model=xgboost.XGBClassifier(max_depth=3)
accs=EnsembleModels(model,100)
plot(range(1,101),accs)
print('mean accuracy:',np.mean(accs))
```

```
mean accuracy: 0.7002666666666667
```




# 2 根據以上的觀察回答以下的問題 (6 分)

## 2.1 在Bagging時, 1.1.2中複雜模型的正確率是否比1.1.1簡單模型的正確率好或差? 為什麼 (2分)

在 Bagging 中，複雜模型的正確率比簡單模型高。因為 Bagging 的核心思想是透過對訓練數據的隨機抽樣（Bootstrap 抽樣）生成多個不同的子樣本，然後分別在這些子樣本上訓練獨立，最後做預測並平均。
高複雜性的 Bagging 模型相對較容易在不同的子樣本上產生差異化的預測，從而增加集成模型的多樣性，與提高模型的穩定性。

## 2.2 在Boosting時, 1.2.1.2/1.2.2.1.2/1.2.2.2.2中複雜模型的正確率是否比1.2.1.1/1.2.2.1.1/1.2.2.2.1中相對應的簡單模型正確率好或差? 為什麼 (2分)

AdaBoost 複雜模型比簡單模型正確率高，因為在複雜 AdaBoost 中，當前一輪的弱學習器錯誤分類的樣本權重會增加，這樣這些樣本在下一輪的訓練中會變得更加重要。
Gradient Boosting 則相反，複雜模型比簡單模型正確率低，因為 Gradient Boosting 是通過迭代地擬合 residual 來進行模型訓練的，每個新的弱學習器都被設計來減小前一輪弱學習器的殘差，而不是像AdaBoost那樣專注於先前分類錯誤的樣本。

## 2.3 為何只有Boosting在簡單模型時(1.2.1.1/1.2.2.1.1/1.2.2.2.1)，正確率大致上會隨著n_estimators數目變多而增加，但Bagging和複雜的Boosting模型卻不是如此? (2分)

在簡單模型下，Boosting 的弱學習器之間的差異性較小，因此增加弱學習器的數量可以增加模型的多樣性，從而提高模型的準確性。

而 Bagging 和複雜的 Boosting 模型已經具有足夠的多樣性，增加弱學習器的數量對模型的準確性影響不大。此外，增加弱學習器的數量也會增加計算時間和內存消耗，因此需要在準確性和效率之間進行權衡。