In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
from sklearn.datasets import load_iris
dataset = load_iris()
X = dataset.data
y = dataset.target
```

In [4]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rand
```

In [5]:
```
knn1 = KNeighborsClassifier(n_neighbors = 11,p = 1)
knn1.fit(X_train,y_train)
knn1.predict(X_test)
knn1.score(X_test, y_test)
```

Out[5]: 0.9777777777777777

In [6]:
```
## Naive Bayes
from sklearn.naive_bayes import GaussianNB
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
nb_model.predict(X_test)
nb_model.score(X_test,y_test)
```

Out[6]: 0.9777777777777777

In [7]:
```
## They have the same score. In that case, the performance of a model depends
## If the dataset is small, probably models will have same score.
```

In [8]:
```
## Decesion Tree
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
dt_model = tree.DecisionTreeClassifier()
dt_model.fit(X_train, y_train)
y_predict = dt_model.predict(X_test)

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_predict)
```

Out[8]: 0.9777777777777777

In [9]:
```
## SVM
from sklearn.svm import SVC # "Support vector classifier"
svm_model = SVC(kernel='linear', C=1E10)
svm_model.fit(X_train, y_train)
svm_model.predict(X_test)
svm_model.score(X_test,y_test)
```

Out[9]: 1.0

In [13]:
```
## Random Forest
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
rf1_model = RandomForestClassifier()
params_rf = {'n_estimators': [50, 100, 200]}
rf_gs = GridSearchCV(rf1_model, params_rf, cv=5)
rf_gs.fit(X_train, y_train)
#save best model
rf_model = rf_gs.best_estimator_
rf_model.score(X_test,y_test)
```

Out[13]: 0.9777777777777777

In [14]:
```
from sklearn.ensemble import VotingClassifier
#create a dictionary of our models
estimators=[('DT', dt_model), ('NB', nb_model), ('SVM', svm_model),('RF',rf_m
#create our voting classifier, inputting our models
ensemble = VotingClassifier(estimators, voting='hard') #Here, hard means majo

#fit model to training data
ensemble.fit(X_train, y_train)
#test our model on the test data
ensemble.score(X_test, y_test)
```

Out[14]: 0.9777777777777777

In [15]:
```
estimators=[('DT', dt_model), ('NB', nb_model), ('SVM', svm_model)]
#create our voting classifier, inputting our models
ensemble = VotingClassifier(estimators, voting='hard') #Here, hard means majo

#fit model to training data
ensemble.fit(X_train, y_train)
#test our model on the test data
ensemble.score(X_test, y_test)
```

Out[15]: 0.9777777777777777

In [16]:
```
estimators=[('DT', dt_model), ('NB', nb_model),('RF',rf_model)]
#create our voting classifier, inputting our models
ensemble = VotingClassifier(estimators, voting='hard') #Here, hard means majo

#fit model to training data
ensemble.fit(X_train, y_train)
#test our model on the test data
ensemble.score(X_test, y_test)
```

Out[16]: 0.9777777777777777

In [17]:
```
estimators=[('DT', dt_model), ('SVM', svm_model),('RF',rf_model)]
#create our voting classifier, inputting our models
ensemble = VotingClassifier(estimators, voting='hard') #Here, hard means majo

#fit model to training data
ensemble.fit(X_train, y_train)
#test our model on the test data
ensemble.score(X_test, y_test)
```

Out[17]: 0.9777777777777777

In [18]:
```
estimators=[('NB', nb_model), ('SVM', svm_model),('RF',rf_model)]
#create our voting classifier, inputting our models
ensemble = VotingClassifier(estimators, voting='hard') #Here, hard means majo

#fit model to training data
ensemble.fit(X_train, y_train)
#test our model on the test data
ensemble.score(X_test, y_test)
```

Out[18]: 0.9777777777777777

In [19]:
```
estimators=[('DT', dt_model), ('NB', nb_model)]
#create our voting classifier, inputting our models
ensemble = VotingClassifier(estimators, voting='hard') #Here, hard means majo

#fit model to training data
ensemble.fit(X_train, y_train)
#test our model on the test data
ensemble.score(X_test, y_test)
```

Out[19]: 0.9777777777777777

In [20]:
```
estimators=[('DT', dt_model), ('SVM', svm_model)]
#create our voting classifier, inputting our models
ensemble = VotingClassifier(estimators, voting='hard') #Here, hard means majo

#fit model to training data
ensemble.fit(X_train, y_train)
#test our model on the test data
ensemble.score(X_test, y_test)
```

Out[20]: 0.9777777777777777

In [21]:
```
estimators=[('DT', dt_model),('RF',rf_model)]
#create our voting classifier, inputting our models
ensemble = VotingClassifier(estimators, voting='hard') #Here, hard means majo

#fit model to training data
ensemble.fit(X_train, y_train)
#test our model on the test data
ensemble.score(X_test, y_test)
```

Out[21]: 0.9777777777777777

In [22]:
```
estimators=[('NB', nb_model), ('SVM', svm_model)]
#create our voting classifier, inputting our models
ensemble = VotingClassifier(estimators, voting='hard') #Here, hard means majo

#fit model to training data
ensemble.fit(X_train, y_train)
#test our model on the test data
ensemble.score(X_test, y_test)
```

Out[22]: 0.9777777777777777

In [23]:
```
estimators=[('NB', nb_model),('RF',rf_model)]
#create our voting classifier, inputting our models
ensemble = VotingClassifier(estimators, voting='hard') #Here, hard means majo

#fit model to training data
ensemble.fit(X_train, y_train)
#test our model on the test data
ensemble.score(X_test, y_test)
```

Out[23]: 0.9777777777777777

In [24]:
```
estimators=[ ('SVM', svm_model),('RF',rf_model)]
#create our voting classifier, inputting our models
ensemble = VotingClassifier(estimators, voting='hard') #Here, hard means majo

#fit model to training data
ensemble.fit(X_train, y_train)
#test our model on the test data
ensemble.score(X_test, y_test)
```

Out[24]:
```
0.9777777777777777
```

In [ ]:
```
## I can't determine which combination is the best because they all have the
```