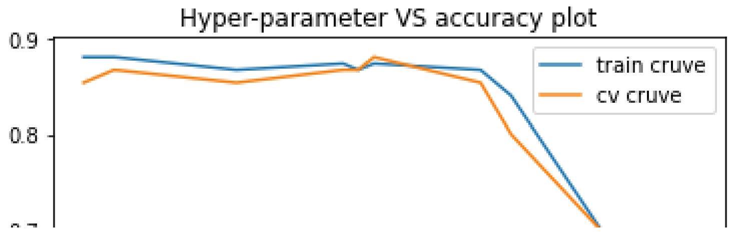
WE CAN WRITE OUR OWN RANDOM GRID SEARCH CV. IT IS A CROSS VALIDATION SPLIT TECHNIQUE IN ORDER TO FIND THE GOOD HYPERPARAMATERS.

Double-click (or enter) to edit

```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
import random
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import numpy as np
import warnings
warnings.filterwarnings("ignore")

#################################Random Search cv definition##############################
def RandomSearchCV(x_train,y_train,classifier,k_val,folds):
    X_train=[];Y_train=[];cvscores=[];trainscores=[];testscores=[]
    #######################DIVIDE x_train into k fold###############################

    #The idea is simple. If we have 5 folds that implies we divide the the data into 5 par

    #x_train[j*int((len(x_train)/folds)):(j+1)*int((len(x_train)/folds))] -->It takes sele
                  # over these 5 parts, we assign a variable j and move it part by part. P

    for j in range(0,folds):
        if ((j+1)*int((len(x_train)/folds)))<= int(len(x_train)) and ((j+1)*int((len(y_tra
                       X_train.append(x_train[j*int((len(x_train)/folds)):(j+1)*int((len(x
                       Y_train.append(y_train[j*int((len(y_train)/folds)):(j+1)*int((len(y

      # So every part is being appended in X_train and Y_train. Seperate steps are given b

      #Now we have data divided into parts. We will then take a k neighbour and apply trai

      #What is cv Data? Simple for 5 parts of your train every 4 parts goes to train and r

      #Start with K values now
    for k in (k_val):       ################### for every k neighbor value################

        trainscores_folds = [];testscores_folds = [] # these will handle the mean accuracy

        for j in range(0, folds):       #####################for every fold/part#########
        #####selecting  data points accordingly, also X_train,Y_train,Xtrain,Ytrain:list #
                    Xtrain=[];Ytrain=[];Xtest=[];Ytest=[]
                    for t in range(0,folds):
                      #the below eqn has RHS which tells the part in the data taken, ex:j=
                      if folds-1-t != j:   ######### when this value becomes equal then da

                                  Xtrain.append(X_train[t]);Ytrain.append(Y_train[t])
                      else:
                                  Xtest.append(X_train[folds-1-j]);Ytest.append(Y_train[fo
                    #### fitting the k neighbors for the jth part being cv and the remaini
                    classifier.n_neighbors = k

                    #Note: Please fit the x data in 2-d array, y data(class label) will be
                    classifier.fit(np.array(Xtrain).reshape((folds-1)*len(Xtrain[0]),2),np

                    #### prediction and accuracy scores for test##########################
                    Y_predicted = classifier.predict(np.array(Xtest).reshape(len(Xtest[0])
                    testscores_folds.append(accuracy_score(np.array(Ytest).reshape(len(Ytr

                    #### prediction and accuracy scores for train ########################
                    Y_predicted = classifier.predict(np.array(Xtrain).reshape((folds-1)*le
                    trainscores_folds.append(accuracy_score(np.array(Ytrain).reshape((fold

        #Mean scores for every k neighbour value
        testscores.append(np.mean(np.array(testscores_folds)))# average CV score for all t
        trainscores.append(np.mean(np.array(trainscores_folds)))
    return trainscores,testscores

sample_size=int(input('please enter number of samples in the dataset : '))
x,y = make_classification(n_samples=sample_size, n_features=2, n_informative=2,n_redundant
x_train, x_test, y_train, y_test = train_test_split(x,y,stratify=y,random_state=42)

####################################GET # OF FOLDS########################################
folds=int(input('please enter number of folds you need < 10 :   '))

low_lim=int(input('define k range: please enter the lower limit:   '))
hig_lim=int(input('define k range: please enter the high limit > 10:    '))


###################################GET 10 UNIQUE K VALUES#################################
k_val=random.sample(range(low_lim,hig_lim),10);k_val=sorted(k_val,reverse=False)

#Considering simple k neighbours algorithm
neigh = KNeighborsClassifier()
trainscores,cvscores=RandomSearchCV(x_train,y_train,neigh,k_val,folds)


################################## plotting the accuracy plot#############################
plt.plot(k_val,trainscores, label='train cruve')
plt.plot(k_val,cvscores, label='cv cruve')
plt.title('Hyper-parameter VS accuracy plot')
plt.legend()
plt.show()
```

```
please enter number of samples in the dataset : 100
please enter number of folds you need < 10 :   3
define k range: please enter the lower limit:   1
define k range: please enter the high limit > 10:    50
```



AS THE BEST K-NEIGHBOUR IS OBSERVED AT 26, LETS FIND THE ACCURACY SCORE FOR IT

```
neigh=KNeighborsClassifier(n_neighbors=26)
neigh.fit(x_train,y_train)
y_pred=neigh.predict(x_train)
accuracy_score(y_pred.reshape(len(y_train)),y_train)
```

```
0.8533333333333334
```

✓ 0s completed at 5:57 PM