Assignment_4_Instructions_and_solution

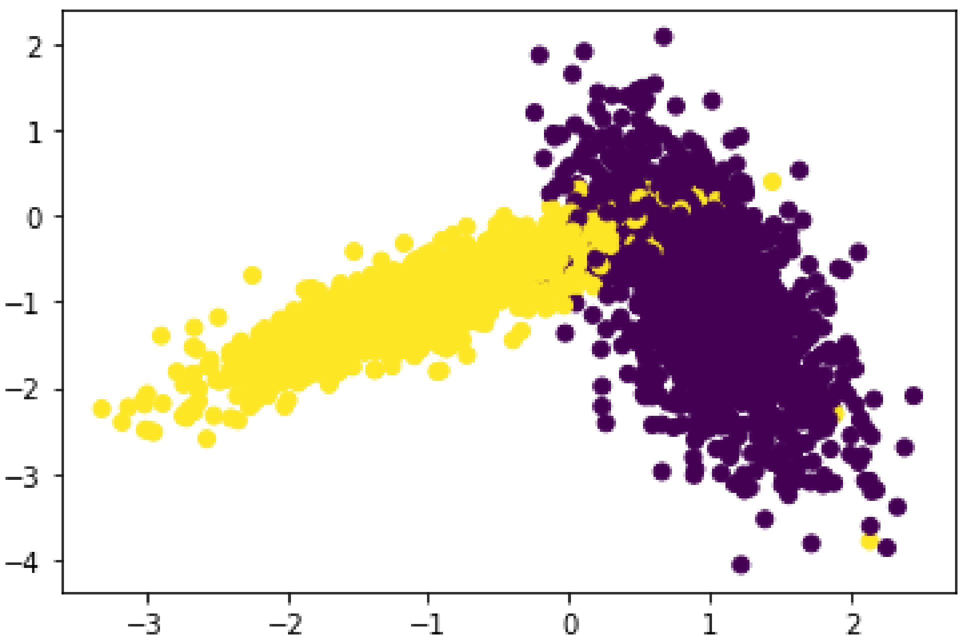
January 14, 2024

[1]:

```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import numpy
from tqdm import tqdm
import numpy as np
from sklearn.metrics.pairwise import euclidean_distances


x,y = make_classification(n_samples=10000, n_features=2, n_informative=2, n_redundant= 0, n_clusters_per_class=1, random_state=60)
X_train, X_test, y_train, y_test = train_test_split(x,y,stratify=y,random_state=42)

# del X_train,X_test
```

[2]:

```
%matplotlib inline
import matplotlib.pyplot as plt
colors = {0:'red', 1:'blue'}
plt.scatter(X_test[:,0], X_test[:,1],c=y_test)
plt.show()
```



1 Implementing Custom RandomSearchCV

[3]:
```
def RandomSearch(x_train, y_train, classifier, params, folds):
    trainscores, testscores = [], []
    for k in tqdm(params['k_neighbours']):
        trainscores_folds, testscores_folds = list(), list()
        for i in range(0, folds):
            # taking test indices first by using length of train data for ex:␣
 ↪for first i = 0 it would take data from 5k to 7.5k
            test_indices = list(range(len(x_train) - (len(x_train) // folds) *␣
 ↪(i + 1), len(x_train) - (len(x_train) // folds) * i))
            train_indices  = list(set(list(range(0, len(x_train)))) -␣
 ↪set(test_indices))

            # Dividing test and train data using test and train indices
            X_train, Y_train, X_test, Y_test = x_train[train_indices],␣
 ↪y_train[train_indices], x_train[test_indices], y_train[test_indices]
            # taking value of k and training the classifier using that.
            classifier.n_neighbors = k
            classifier.fit(X_train,Y_train)

            Y_predicted = classifier.predict(X_test)
            testscores_folds.append(accuracy_score(Y_test, Y_predicted))
            Y_predicted = classifier.predict(X_train)
            trainscores_folds.append(accuracy_score(Y_train, Y_predicted))

        # appending accuracy values to trainscores and testscores
        trainscores.append(np.mean(np.array(trainscores_folds)))
        testscores.append(np.mean(np.array(testscores_folds)))
    return trainscores, testscores
```

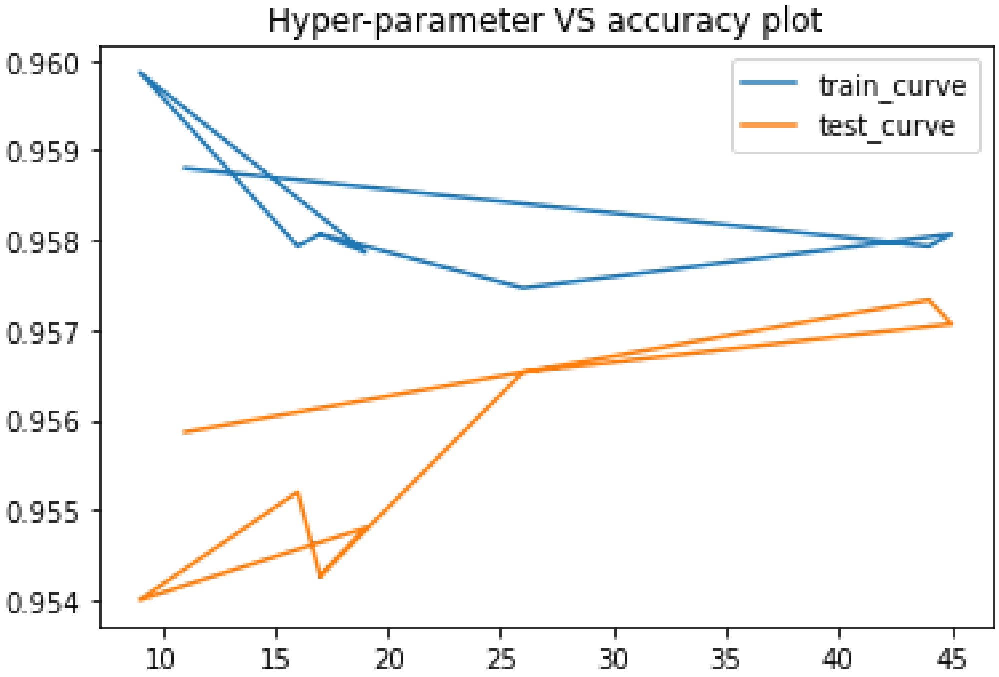
[4]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import random
import warnings
import numpy as np
warnings.filterwarnings('ignore')

neigh = KNeighborsClassifier()
param_range = (1, 50)
params = {'k_neighbours':np.random.uniform(param_range[0], param_range[1], 10).
 ↪astype('int')}
folds = 3

trainscores, testscores = RandomSearch(X_train, y_train, neigh, params, folds)
```

```
100%|
    | 10/10 [00:04<00:00,  2.00it/s]
```

[5]:

```
plt.plot(params['k_neighbours'], trainscores, label='train_curve')
plt.plot(params['k_neighbours'], testscores, label='test_curve')
plt.title('Hyper-parameter VS accuracy plot')
plt.legend()
plt.show()
```


Hyper-parameter VS accuracy plot
train_curve
test_curve
0.960
0.959
0.958
0.957
0.956
0.955
0.954
10
15
20
25
30
35
40
45

```
[ ]:
```