

digitdata

February 4, 2023

```
[31]: from sklearn.svm import SVC
      from sklearn.ensemble import RandomForestClassifier
      from sklearn.linear_model import LogisticRegression
      from sklearn.naive_bayes import GaussianNB
      from sklearn.naive_bayes import MultinomialNB
      from sklearn.tree import DecisionTreeClassifier
      from sklearn.model_selection import GridSearchCV
      from sklearn.model_selection import RandomizedSearchCV
      from sklearn.datasets import load_digits
      import pandas as pd
```

```
[32]: digit_data=load_digits()
```

1 Data Description

```
[33]: print(digit_data.DESCR)
```

```
.. _digits_dataset:
```

```
Optical recognition of handwritten digits dataset
```

```
-----
```

```
**Data Set Characteristics:**
```

```
:Number of Instances: 1797
:Number of Attributes: 64
:Attribute Information: 8x8 image of integer pixels in the range 0..16.
:Missing Attribute Values: None
:Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)
:Date: July; 1998
```

This is a copy of the test set of the UCI ML hand-written digits datasets
<https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits>

The data set contains images of hand-written digits: 10 classes where each class refers to a digit.

Preprocessing programs made available by NIST were used to extract normalized bitmaps of handwritten digits from a preprinted form. From a total of 43 people, 30 contributed to the training set and different 13 to the test set. 32x32 bitmaps are divided into nonoverlapping blocks of 4x4 and the number of on pixels are counted in each block. This generates an input matrix of 8x8 where each element is an integer in the range 0..16. This reduces dimensionality and gives invariance to small distortions.

For info on NIST preprocessing routines, see M. D. Garriss, J. L. Blue, G. T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C. L. Wilson, NIST Form-Based Handprint Recognition System, NISTIR 5469, 1994.

.. topic:: References

- C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their Applications to Handwritten Digit Recognition, MSc Thesis, Institute of Graduate Studies in Science and Engineering, Bogazici University.
- E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.
- Ken Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin. Linear dimensionality reduction using relevance weighted LDA. School of Electrical and Electronic Engineering Nanyang Technological University. 2005.
- Claudio Gentile. A New Approximate Maximal Margin Classification Algorithm. NIPS. 2000.

2 Attributes in Digit Dataset

```
[34]: dir(digit_data)
```

```
[34]: ['DESCR', 'data', 'feature_names', 'frame', 'images', 'target', 'target_names']
```

3 Image Data

```
[35]: digit_data.data
```

```
[35]: array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
          [ 0.,  0.,  0., ..., 10.,  0.,  0.],
          [ 0.,  0.,  0., ..., 16.,  9.,  0.],
          ...,
          [ 0.,  0.,  1., ...,  6.,  0.,  0.],
          [ 0.,  0.,  2., ..., 12.,  0.,  0.],
          [ 0.,  0., 10., ..., 12.,  1.,  0.]])
```

```
[36]: digit_data.data.shape
```

```
[36]: (1797, 64)
```

```
[37]: digit_data.images.shape # Image are of 8x8 shape
```

```
[37]: (1797, 8, 8)
```

4 Target Data

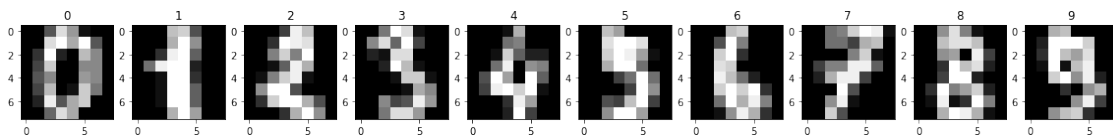
```
[38]: digit_data.target_names
```

```
[38]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

5 Target Images

```
[39]: import matplotlib.pyplot as plt
print("Target Images")
fig,ax = plt.subplots(1,10,figsize = (20,20))
ax = ax.ravel()
for i in range(10):
    image = digit_data.images[i]
    ax[i].imshow(image,cmap = 'gray')
    ax[i].set_title(digit_data.target[i])
plt.show()
```

Target Images



6 Preprocessing Data with MinMaxScaler

```
[40]: from sklearn.preprocessing import MinMaxScaler
mn=MinMaxScaler()
X=mn.fit_transform(digit_data.data)
X
```

```
[40]: array([[0.    , 0.    , 0.3125, ..., 0.    , 0.    , 0.    ],
        [0.    , 0.    , 0.    , ..., 0.625 , 0.    , 0.    ],
```

```
[0.      , 0.      , 0.      , ..., 1.      , 0.5625, 0.      ],
...,
[0.      , 0.      , 0.0625, ..., 0.375 , 0.      , 0.      ],
[0.      , 0.      , 0.125 , ..., 0.75  , 0.      , 0.      ],
[0.      , 0.      , 0.625 , ..., 0.75  , 0.0625, 0.      ]])
```

7 Train Test Split

```
[78]: from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,digit_data.target,test_size=0.
↪3,random_state=10)
```

```
[42]: X_train.shape
```

```
[42]: (1257, 64)
```

```
[43]: y_train.shape
```

```
[43]: (1257,)
```

8 Model And Parameters Dictionary To Select From

```
[44]: model_params={
    'svm':{
        'model':SVC(gamma='auto'),
        'params':{
            'C':list(range(1,30,5)),
            'kernel':['linear','rbf']
        }
    },
    'random_forest':{
        'model':RandomForestClassifier(),
        'params':{
            'n_estimators':list(range(1,100,5))
        }
    },
    'logistic_regression':{
        'model':LogisticRegression(solver='liblinear',multi_class='auto'),
        'params':{
            'C':list(range(1,30,5))
        }
    },
    'GaussianNB':{
        'model':GaussianNB(),
        'params':{}}
```

```

},
'MultinomialNB':{
    'model':MultinomialNB(),
    'params':{}}
},
'DecisionTree':{
    'model':DecisionTreeClassifier(),
    'params':{
        'criterion' : ["gini", "entropy", "log_loss"],
    }
}
}

```

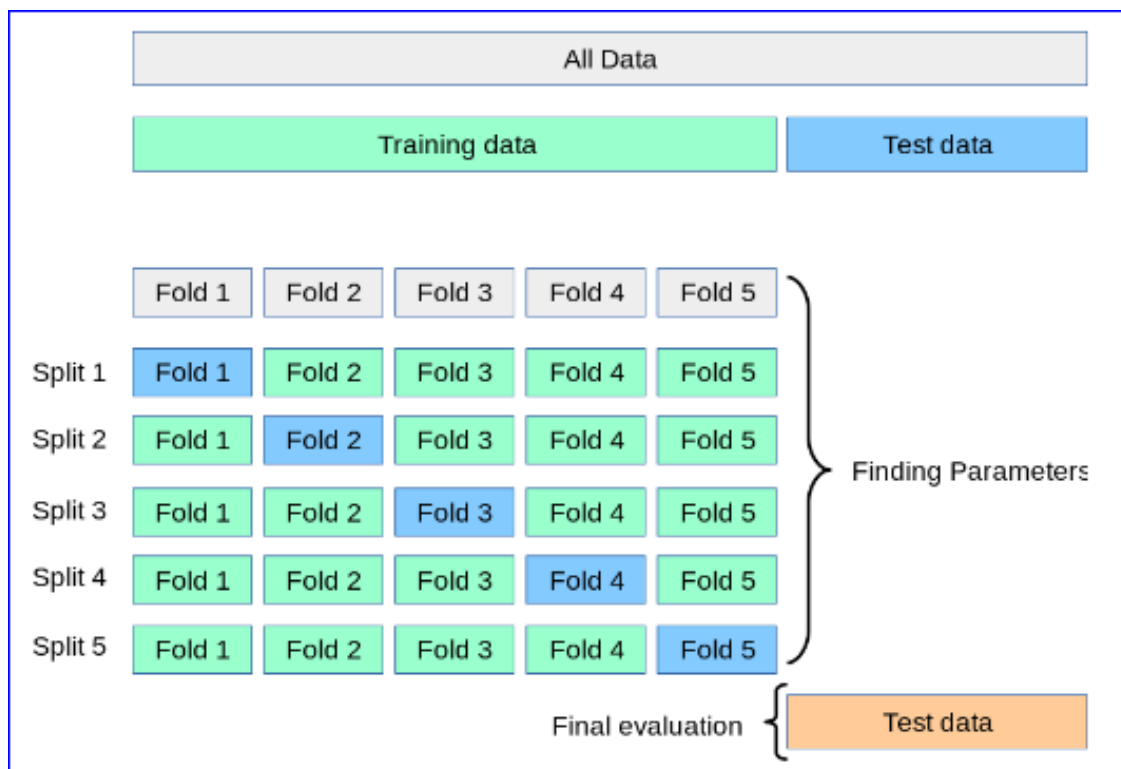
- Cross Validation On Each Step in GridSearchCV is shown below

```

[86]: from IPython import display
display.Image("D:\\Machine Learning Practise\\15) HyperParameter Tuning\\Exercise\\image.png",width=400,height=400)

```

[86]:



9 GridSearchCV

```
[46]: score=[]
      for model,mp in model_params.items():
          clf=GridSearchCV(mp['model'],mp['params'],cv=6,return_train_score=False)
          clf.fit(X,digit_data.target)
          score.append({
              'model':model,
              'best_score':clf.best_score_,
              'best_params':clf.best_params_
          })
```

10 GridSearchCV Report

```
[47]: result=pd.DataFrame(score)
      result
```

```
[47]:
```

	model	best_score	best_params
0	svm	0.963276	{'C': 6, 'kernel': 'rbf'}
1	random_forest	0.951039	{'n_estimators': 61}
2	logistic_regression	0.937679	{'C': 16}
3	GaussianNB	0.804705	{}
4	MultinomialNB	0.874812	{}
5	DecisionTree	0.822467	{'criterion': 'entropy'}

11 RandomizedSearchCV

```
[60]: score_rand=[]
      for model,mp in model_params.items():
          ␣
          ↪clf=RandomizedSearchCV(mp['model'],mp['params'],cv=6,return_train_score=False,n_iter=1)
          clf.fit(X,digit_data.target)
          score_rand.append({
              'model':model,
              'best_score':clf.best_score_,
              'best_params':clf.best_params_
          })
```

12 RandomizedSearchCV Report

```
[61]: result_rand=pd.DataFrame(score_rand)
      result_rand
```

```
[61]:
```

	model	best_score	best_params
0	svm	0.963276	{'kernel': 'rbf', 'C': 6}
1	random_forest	0.724539	{'n_estimators': 1}
2	logistic_regression	0.937672	{'C': 1}
3	GaussianNB	0.804705	{}
4	MultinomialNB	0.874812	{}
5	DecisionTree	0.821355	{'criterion': 'log_loss'}

13 Now Using SVC

- which is selected as best model from both above results

```
[79]: model=SVC(C=6,kernel='rbf')
      model.fit(X_train,y_train)
```

```
[79]: SVC(C=6)
```

14 Checkin For Overfitting

```
[80]: from sklearn.metrics import
      ↪confusion_matrix,classification_report,accuracy_score
      print("Training accuracy: ",accuracy_score(y_train,model.predict(X_train)))
      print("Testing accuracy: ",accuracy_score(y_test,model.predict(X_test)))
```

```
Training accuracy:  1.0
Testing accuracy:  0.987037037037037
```

15 Classification Visulization Uing Confusion Matrix

```
[82]: import seaborn as sns

      plt.figure(figsize=(8,8))
      sns.heatmap(confusion_matrix(y_test,model.predict(X_test)),annot=True)
      plt.xlabel("Predicted Digits",fontdict={"size":20})
      plt.ylabel("True Digits",fontdict={"size":20})
```

```
[82]: Text(51.0, 0.5, 'True Digits')
```

