Recognizing Handwritten Digits with scikit-learn

My name is Aarav Chauhan. I am undertaking B.Tech (a pre-final year student) from NIET Greater Noida. Currently, I am working as Intern at Suven Consultant Pvt. Ltd.

Today, I am here to explain my project i.e., “RECOGNISING HANDWRITTEN DIGITS WITH SCIKIT-LEARN”.

Recognizing handwritten text is a problem that can be traced back to the first automatic machines that needed to recognize individual characters in handwritten documents. Think about, for example, the ZIP codes on letters at the post office and the automation needed to recognize these five digits. Perfect recognition of these codes is necessary in order to sort mail automatically and efficiently. Included among the other applications that may come to mind is OCR (Optical Character Recognition) software. OCR software must read the handwritten text, or pages of printed books, for general electronic documents in which each character is well defined

> **Hypothesis :**
>
>

The Digits data set of the Scikit-learn library provides numerous data-sets that are useful for testing many problems of data analysis and prediction of the results. Some Scientist claims that it predicts the digit accurately 95% of the times. Perform data Analysis to accept or reject this Hypothesis.

> **Prerequists :**
>
>

Sklearn

Matplotlib

Basics of Machine learning

Dataset :

In this project, we are using the Handwritten Digits dataset which is already ready in the sklearn library. we can import the dataset using the below code.

```
from sklearn import datasets
digits = datasets.load_digits()
```

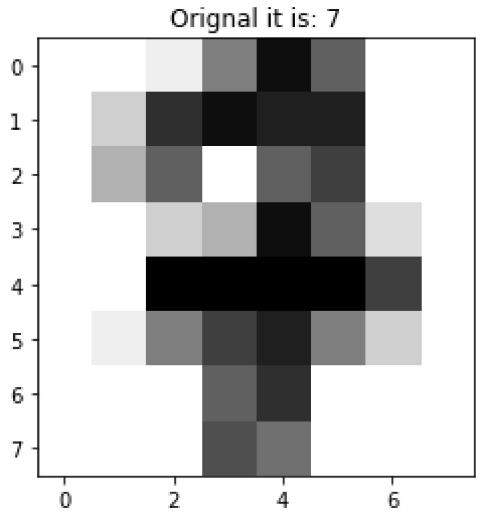
Digits dataset is a dictionary that contains data, targets, images, features names, description of the dataset, target names, etc.

We focus mainly on data and targets. We extract both on different variables.

```
main_data = digits['data']
targets = digits['target']
```

Now we can see our data look.

```
def view_digit(index):
    plt.imshow(digits.images[index] , cmap = plt.cm.gray_r ,
interpolation = 'nearest')
    plt.title('Orignal it is: '+ str(digits.target[index]))
    plt.show()view_digit(17)
```



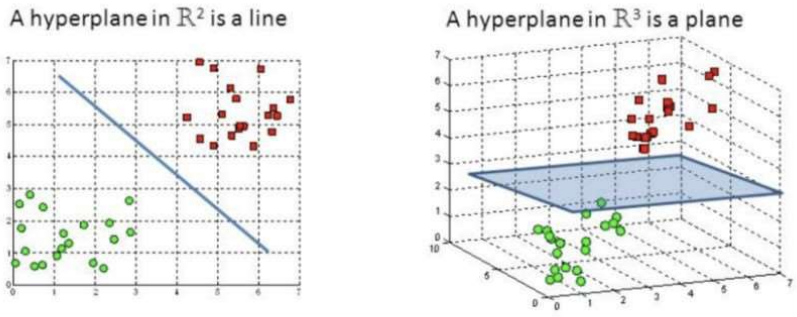
Handwritten Digit Dataset

Model Planning:

To see how different models work on different data sizes we are using 3 models Support vector Classifier, Decision Tree Classifier, Random Forest Classifier.

1. **Support Vector Classifier :**

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N—the number of features) that distinctly classifies the data points. More...



Hyperplanes in 2D and 3D feature space

Credits:- https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Code :

```
# import the SVC
from sklearn import svm
svc = svm.SVC(gamma=0.001 , C = 100.)
# gamma and C are hyperparameters# Training data = 1790 ,
Validation data = 6
svc.fit(main_data[:1790] , targets[:1790])# predict on test
data
predictions = svc.predict(main_data[1791:])# check the
result
predictions , targets[1791:]
```

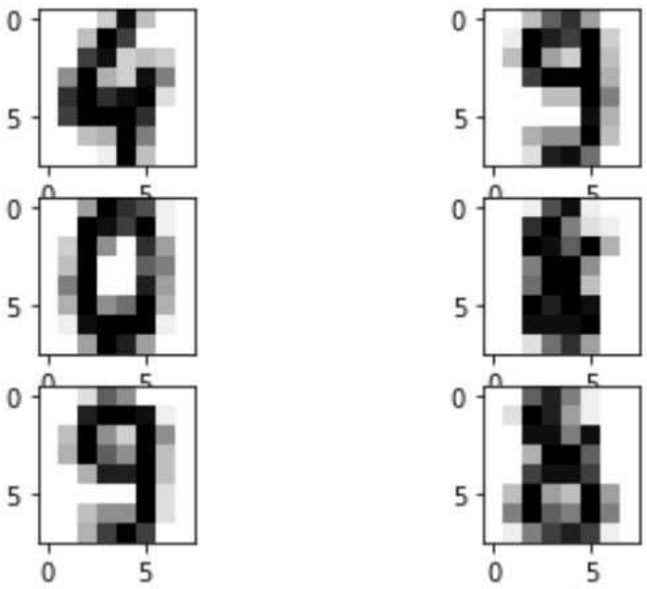
```
predictions , targets[1791:]
```

```
(array([4, 9, 0, 8, 9, 8]), array([4, 9, 0, 8, 9, 8]))
```

SVC Output

As we can see we use very high data for training and very little data for training and the Support Vector Classifier do a very good job on data and we get 100 % accuracy on test data.

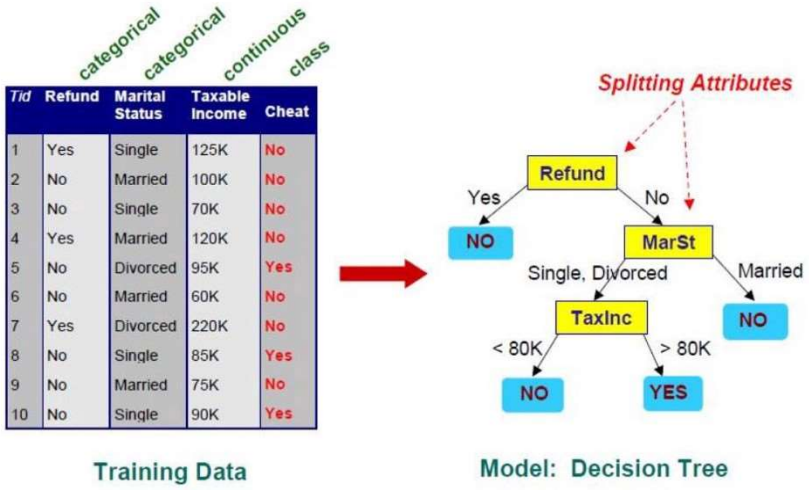
```
<matplotlib.image.AxesImage at 0x1fe8ee88fa0>
```



2. Decision Tree Classifier :

Decision Tree Classifier is a simple and widely used classification technique. It applies a straightforward idea to solve the classification problem. Decision Tree Classifier poses a series of carefully crafted questions about the attributes of the test record. Each time it receives an answer, a follow-up question is asked until a conclusion about the class label of the record is reached.

More Details on Decision Tree...



Credits:- http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/lguo/decisionTree.html

Code :

```
# import the Classifier
from sklearn.tree import DecisionTreeClassifier# Instanciate
Model
# we can also use criterion = 'entropy' both lead us to
nearly same
# resultdt = DecisionTreeClassifier(criterion = 'gini') #
fit the data on model
# Training Set = 1600 , Validation Set = 197
dt.fit(main_data[:1600] , targets[:1600])# prediction on
test data
predictions2 = dt.predict(main_data[1601:])# We use
classification materics as accuracy_score
# import accuracy_score
from sklearn.metrics import
accuracy_scoreaccuracy_score(targets[1601:] , predictions2)
```

```
accuracy_score(targets[1601:] , predictions2)
```

```
0.7857142857142857
```

Accuracy of Decision Tree Classifier

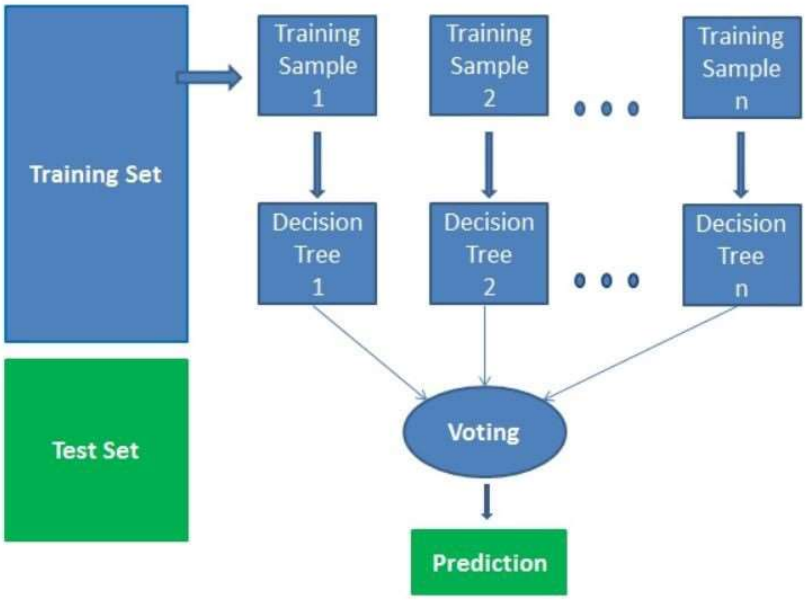
Now for this time, we use different sizes of training data and validation data. As we can see Decision Tree classifier performs poor performance on the data. we can increase the accuracy by fine-tuning the hyperparameters of DTC.

Hyperparameters of Decision Tree Classifier

3. Random Forest Classifier :

Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy-to-use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests create decision trees on randomly selected data samples, get a prediction from each tree, and selects the best solution by means of voting. It also provides a pretty good indicator of the feature's importance.

More on Random Forest...



Credits:- https://www.datacamp.com/community/tutorials/random-forests-classifier-python

Code :

```
from sklearn.ensemble import RandomForestClassifier#
n_estimators hyperparameters( default 100 )
rc = RandomForestClassifier(n_estimators = 150)# Training
Data = 1500 , Validation data = 297
rc.fit(main_data[:1500] , targets[:1500])predictions3 =
rc.predict(main_data[1501:])accuracy_score(targets[1501:] ,
predictions3)
```

```
accuracy_score(targets[1501:] , predictions3)
```

```
0.9222972972972973
```

Accuracy score Random Forest

As we can see Random Forest performs excellently with fewer data compare to both the Decision tree and Support vector. We get a 92 % Accuracy score with Random Forest Classifier.

Conclusion :

As per our hypothesis, we can say with hyperparameter tunning with different machine learning models or using more data we can achieve near 95% accuracy on the handwritten dataset. But make sure we also have a good amount of test data otherwise the model will get overfit.