Handwritten Digit Recognition using SVM Classifier

1 — **Finding the Problem - Application**
- To Predict
- Handwritten Digit recognition

2 — **Collecting Dataset**

Classes	10
Samples per class	~180
Samples total	1797
Dimensionality	64
Features	integers 0-16

- Input: Pixel value of Input Image Output: 10 Digits (0 to 9)
- Scikit-learn Digit Dataset

3 — **Load & Summarize Dataset**
- Load Dataset from the directory & Summarize the details such as data shape & length
  - Data — Shape — (1797, 64)
  - Image — Shape — (1797, 8, 8)
  - Target — [0 1 2 ... 8 9 8]

4 — **Visualize the Data**
- Gray values to Image

```
array([[ 0.,  0.,  0., 12., 13.,  5.,  0.,  0.],
       [ 0.,  0.,  0., 11., 16.,  9.,  0.,  0.],
       [ 0.,  0.,  3., 15., 16.,  6.,  0.,  0.],
       [ 0.,  7., 15., 16., 16.,  2.,  0.,  0.],
       [ 0.,  0.,  1., 16., 16.,  3.,  0.,  0.],
       [ 0.,  0.,  1., 16., 16.,  6.,  0.,  0.],
       [ 0.,  0.,  1., 16., 16.,  6.,  0.,  0.],
       [ 0.,  0.,  0., 11., 16., 10.,  0.,  0.]])
```

5 — **Segregating Dataset into X & Y**
- Input: Image Pixels
  - dataset.images.reshape((imageDatalength,-1))
- Output: Class
  - dataset.target

6 — **Splitting Dataset to Train & Test**
- Useful for validation
- train_test_split(X, Y, test_size = 0.25, random_state = 0)

7 — **Algorithm** — **Support Vector Machine - Classifier**

- **Types**
  - Linear SVM
  - Non-Linear SVM

- **Main Components**
  - Hyperplane — It is a decision plane which separates between a set of objects
  - Support Vectors — Data point close to the Hyperplane, which defines the separating line better by calculating margins
  - Margin — It is a gap between the two lines on the closest class points

- **SVM Kernel** - Take data as input and transform it into the required form
  - Linear Kernel
    - K(x, xi) = sum(x * xi)
    - sum of the multiplication of each pair of input values
  - Radial Basis Function Kernel — Default
    - K(x,xi) = exp(-gamma * sum((x – xi^2))
    - It can map an input space in infinite dimensional space
    - Gamma range 0 to 1
    - Perfect Gamma Value (Default) -0.1
    - If Gamma increases - Overfitting will occur
  - Polynomial Kernel — curved or nonlinear input space

- **Other Tuning Parameters**
  - C - Regularization parameter — Float - default=1.0
    - The strength of the regularization is inversely proportional to C
  - Degree — Int -default=3
    - Degree of the polynomial kernel function ('poly')
  - Gamma — Float, default='scale' or 'auto'
    - Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

- **Choose Good C & Gamma**
  - GridSearchCV with C and gamma spaced exponentially far apart to choose good values

8 — **Training**
- Training our Model for Pre-processed Dataset
- model.fit(X_train, y_train)

9 — **Evaluate/Validation**
- Obtaining the accuracy of the Model
- accuracy_score

10 — **Prediction**
- Observing how our model is classifying our new data

11 — **Playing with Different Methods of SVC**
- To observe which is giving more accurate