Intro to Machine Learning:

Machine Learning: is the science or art of programming computers so they can learn from the data on their own.

Data Science: is basically an approach of handling complex problems using statistics, domain experience and a bit of ML algorithms.

Artificial Intelligence: is actually an appliaction designed to address a particular problem using ML algos.

A Data Science Life Cycle:

1. Understand the domain & problem statement.
2. Exploratory Data Analysis.
3. Train an ML algos & come up with an AI solution.
4. Evaluate your solution:
   4.1. if evaluation is falsified, go back to step 2 & 3
   4.2. if evaluation are confident enough, go ahead & launch your solution

Types of ML algos:

3 ways of categorizing diff ML algos,

- whether they are trained with some human supervision or not? (`Supervied` ML algo, `Unsupervised` ML algo, `Reinforcement` algo)
- whether or not your ML algos can learn incrementally or on the fly (`Online Learning algo` , `Batch Learning algo`)
- whether they work by simply comparing new data points to the already known data points (known because your ML algo saw it while training) or instead can detect patterns in the training data and build a predictive model out of it. (`Instance-based learning` , `Model-based learning`)

`Supervised Learning` :
some examples,

```
    - k-Nearest Neighbours
    - Linear Regression
    - Logistic Regression
    - Support Vector Machines
    - Decision Trees & Random Forests
    - Neural Networks
```

`Unsupervised Learning` :
some examples,

```
    - Clustering
        - k-Means
        - Hierarchical Clustering
        - Expectation Maximization
    - Dimensionality Reduction and Data Visualization
        - Principal Component Analysis
        - Kernel PCA
        - Locally Linear Embedding
        - t-distributed Stochastic Neighbor Embedding (t-SNE)
    - Association Rule
        - Apriori
        - FP Growth
        - Eclat
```

Classification

Types of Classification:

- Logistic Regression
- Naive Bayes Classification
- Decision Trees
- Random Forests
- Support Vector Machines (SVM)
- k-Nearest Neighbors (KNN)
- Stochastic Gradient Descent Classifier
- Neural Network (Deep Learning)

MNIST

In [1]:

```
# loading necessary libs
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
from sklearn.datasets import fetch_openml
X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
```

In [3]:

```
# looking at the first row of feature dataset X
X[0]
```

Out[3]:

```
array([  0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   3.,  18.,
        18.,  18., 126., 136., 175.,  26., 166., 255., 247., 127.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
        30.,  36.,  94., 154., 170., 253., 253., 253., 253., 253., 225.,
       172., 253., 242., 195.,  64.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,  49., 238., 253., 253., 253., 253.,
       253., 253., 253., 253., 251.,  93.,  82.,  82.,  56.,  39.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
        18., 219., 253., 253., 253., 253., 253., 198., 182., 247., 241.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,  80., 156., 107., 253.,
       253., 205.,  11.,   0.,  43., 154.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,  14.,   1., 154., 253.,  90.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
       139., 253., 190.,   2.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,  11., 190., 253.,  70.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,  35., 241., 225., 160., 108.,   1.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,  81., 240.,
       253., 253., 119.,  25.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,  45., 186., 253., 253., 150.,  27.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,  16.,  93., 252., 253., 187.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 249., 253.,
       249.,  64.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,  46., 130., 183., 253., 253., 207.,   2.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,  39., 148., 229., 253., 253., 253.,
       250., 182.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,  24., 114.,
       221., 253., 253., 253., 253., 201.,  78.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,  23.,  66., 213., 253., 253., 253., 253., 198.,  81.,
         2.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,  18., 171., 219., 253., 253.,
       253., 253., 195.,  80.,   9.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,  55.,
```

```
       172., 226., 253., 253., 253., 253., 244., 133.,  11.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0., 136., 253., 253., 253., 212., 135.,
       132.,  16.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.])
```

In [4]:

```
# looking at the first row of y-label 'y'
y[0]
```

Out[4]:

```
'5'
```

In [5]:

```
28*28 # its 784 attirbutes because 28 pixels x 28 pixels = 784 pixel values in an array
```

Out[5]:

```
784
```

In [6]:

```
type(X)
```

Out[6]:

```
numpy.ndarray
```

In [7]:

```
type(y)
```

Out[7]:

```
numpy.ndarray
```

In [8]:

```
X.shape
```

Out[8]:

```
(70000, 784)
```

In [9]:

```
y.shape
```

Out[9]:

```
(70000,)
```

In [10]:

```
some_digit = X[0]
some_digit = some_digit.reshape(28, 28)
```

In [11]:

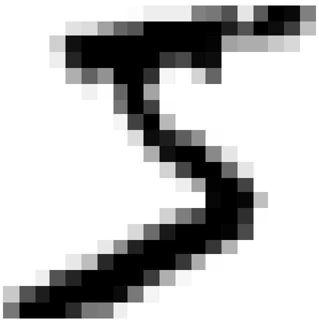
```
some_digit.shape
```

Out[11]:

```
(28, 28)
```

In [12]:

```
plt.imshow(some_digit, cmap = plt.cm.binary)
plt.axis('off')
plt.show()
```



In [13]:

```
y[0]
```

Out[13]:

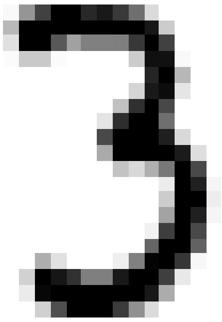
```
'5'
```

In [14]:

```
def plot_digit(data):
    image = data.reshape(28, 28)
    plt.imshow(image, cmap=plt.cm.binary)
    plt.axis('off')
    plt.show()
```

In [15]:

```
plot_digit(X[12312])
```



In [16]:

```
y[12312]
```

Out[16]:

```
'3'
```

How to load in the external data source: loading the MNIST dataset from Kaggle

datasource: [https://www.kaggle.com/c/digit-recognizer/data (https://www.kaggle.com/c/digit-recognizer/data)](https://www.kaggle.com/c/digit-recognizer/data)

In [17]:

```
mnist = pd.read_csv('dataset/digit-recognizer/train.csv')
```

In [18]:

```
type(mnist)
```

Out[18]:

```
pandas.core.frame.DataFrame
```

In [19]:

```
mnist.head()
```

Out[19]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pi
0	1	0	0	0	0	0	0	0	0	0	...	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	

5 rows × 785 columns

In [20]:

```
mnist.columns
```

Out[20]:

```
Index(['label', 'pixel0', 'pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixel
5',
       'pixel6', 'pixel7', 'pixel8',
       ...
       'pixel774', 'pixel775', 'pixel776', 'pixel777', 'pixel778', 'pixel7
79',
       'pixel780', 'pixel781', 'pixel782', 'pixel783'],
      dtype='object', length=785)
```

In [21]:

```
mnist.values
```

Out[21]:

```
array([[1, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       ...,
       [7, 0, 0, ..., 0, 0, 0],
       [6, 0, 0, ..., 0, 0, 0],
       [9, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [22]:

```
X = mnist.drop('label', axis=1) # dropping 'label' columns, keeping rest of fields
y = mnist[['label']] # slicing up & keeping just the 'label' column

print(X.shape)
print(y.shape)
```

```
(42000, 784)
(42000, 1)
```

In [23]:

```
y.values
```

Out[23]:

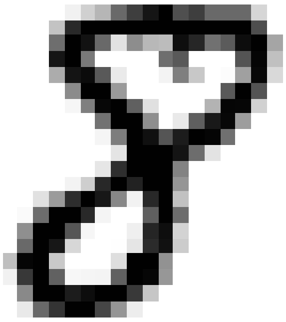
```
array([[1],
       [0],
       [1],
       ...,
       [7],
       [6],
       [9]], dtype=int64)
```

In [24]:

```
def plot_digit(data):
    image = data.values.reshape(28, 28)
    plt.imshow(image, cmap=plt.cm.binary)
    plt.axis('off')
    plt.show()
```

In [25]:

```
plot_digit(X.iloc[2100])
```



In [26]:

```
y.iloc[2100] # verifying
```

Out[26]:

```
label    8
Name: 2100, dtype: int64
```

Google Up: Read More on **Data Snooping Bias**

Training and Testing

[option 1] writing your own logic of splitting the train and test dataset

In [27]:

```
# writing your own logic for splitting train & test
def split_train_test(data, test_ratio):
    np.random.seed(29) # to generate random numbers but have the same random numbers ge
nerated everytime you run this code
    shuffled_idx = np.random.permutation(len(data))
    test_set_size = int(len(data) * test_ratio)
    test_idx = shuffled_idx[:test_set_size]
    train_idx = shuffled_idx[test_set_size:]

    return data.iloc[train_idx], data.iloc[test_idx]
```

In [28]:

```
train_set, test_set = split_train_test(X, 0.2)

print(len(train_set), "train + ", len(test_set), "test")
```

```
33600 train +  8400 test
```

In [29]:

```
train_set.shape
```

Out[29]:

```
(33600, 784)
```

In [30]:

```
test_set.shape
```

Out[30]:

```
(8400, 784)
```

other ways of splitting using scikit-learn's built in methods

[option 2] using Scikit-Learn's Random Sampling (pretty much the same logic laid down by sklearn with random seed like feature)

What is Random Sampling?: `basically randomly picking up some indices from the original dataset and keeping some portion (like 70%) of it into train and other into test set (rest 30%).`

```
from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(data, test_size=0.2, random_state=29)
```

[option 3] using Scikit-Learn's Stratified Sampling (keeping dataset stratified even after sampling)

What is Stratified Sampling?: `basically picking indices in such a way the proportion of different classes in the so called train & test data set now, is same as the proportion of different classes in the original dataset.`

```
from sklearn.model_selection import StratifiedShuffleSplit
for train_idx, test_idx in split.split(X, y):
    strat_train_set = data[train_idx]
    strat_test_set = data[test_idx]
```

Cross Validation (CV):
`k-fold Cross Validation`

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(your_model, your_predictors, your_target_labels, cv=10,
scoring='mean_squared_error')
model_mse_scores = np.mean(scores)
```

In [31]:

```
# Stratified Sampling split
from sklearn.model_selection import StratifiedShuffleSplit
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=29)

for train_idx, test_idx in split.split(mnist, mnist['label']):
    strat_train_set = mnist.loc[train_idx]
    strat_test_set = mnist.loc[test_idx]
```

In [32]:

```
strat_train_set.shape
```

Out[32]:

```
(33600, 785)
```

In [33]:

```
strat_test_set.shape
```

Out[33]:

```
(8400, 785)
```

In [34]:

```
mnist.shape
```

Out[34]:

```
(42000, 785)
```

In [35]:

```
mnist['label'].value_counts()
```

Out[35]:

```
1    4684
7    4401
3    4351
9    4188
2    4177
6    4137
0    4132
4    4072
8    4063
5    3795
Name: label, dtype: int64
```

In [36]:

```
100*(3795/42000) # proportion of class '5' in mnist
```

Out[36]:

```
9.035714285714286
```

In [37]:

```
strat_train_set['label'].value_counts()
```

Out[37]:

```
1    3747
7    3521
3    3481
9    3350
2    3342
6    3309
0    3306
4    3258
8    3250
5    3036
Name: label, dtype: int64
```

In [38]:

```
100*(3036/33600) # proportion of class '5' in stratified sampling
```

Out[38]:

```
9.035714285714286
```

as you can see, the proportions are maintained in stratified sampling split

In [39]:

```
# Random Sampling split
from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(mnist, test_size=0.2, random_state=29)
```

In [40]:

```
X_train, y_train = train_set.drop('label', axis=1), train_set[['label']]
X_test, y_test = test_set.drop('label', axis=1), test_set[['label']]
```

Binary Classifier

"5-Detector Classifier"

In [41]:

```
y_train_5 = (y_train == 5) # True for all 5s, false otherwise
y_test_5 = (y_test == 5)
```

In [42]:

```
# building a Stochastic Gradient Descent (SGD) Binary Classifier

from sklearn.linear_model import SGDClassifier

sgd_clf = SGDClassifier(random_state=29)
sgd_clf.fit(X_train, y_train_5)
```

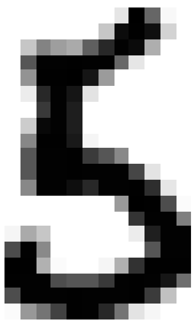
```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
1: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

Out[42]:

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=29, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False)
```

In [43]:

```
# Lets pick one example

some_digit = X.iloc[4500]
plot_digit(some_digit)
print(y.iloc[4500])
```



```
label    5
Name: 4500, dtype: int64
```

In [44]:

```
# predicting from our model
sgd_clf.predict([some_digit])
```

Out[44]:

```
array([ True])
```

In [45]:

```
# cross validation

from sklearn.model_selection import cross_val_score

cross_val_score(sgd_clf, X_train, y_train_5, scoring="accuracy", cv=3)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
1: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
1: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
1: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

Out[45]:

```
array([0.95419643, 0.94910714, 0.95133929])
```

above 95% accuracy on all cross-validation folds!

But beware!! `Accuracy is generally not a good measure of performance for a classifier`

Lets build a DUMB CLASSIFIER, it predicts every single image as "not-5" class

In [46]:

```
from sklearn.base import BaseEstimator

class Never5Classifier(BaseEstimator):
    def fit(self, X, y=None):
        pass # no training at all

    def predict(self, X):
        return np.zeros((len(X), 1), dtype=bool) # always predicting "False"
```

In [47]:

```
never_a_5_clf = Never5Classifier()
cross_val_score(never_a_5_clf, X_train, y_train_5, cv=3, scoring="accuracy")
```

Out[47]:

```
array([0.91285714, 0.906875  , 0.90973214])
```

Confusion Matrix

In [48]:

```
from sklearn.model_selection import cross_val_predict

y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
1: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
1: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
1: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

In [49]:

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_train_5, y_train_pred)
```

Out[49]:

```
array([[29783,   787],
       [  841,  2189]], dtype=int64)
```

Precision

A more concise metric to look at is to look at its 'accuracy' of all the positive preidctions, also called as **Precision** of the classifier

$$precision = \frac{TP}{TP + FP}$$

Recall

Along with Precision, another metric to consider is the **sensitivity** of the model, also called as **Recall** or **True Positive Rate**

$$recall = \frac{TP}{TP + FN}$$

In [50]:

```
from sklearn.metrics import precision_score, recall_score
```

In [51]:

```
precision_score(y_train_5, y_train_pred)
```

Out[51]:

```
0.7355510752688172
```

In [52]:

```
recall_score(y_train_5, y_train_pred)
```

Out[52]:

```
0.7224422442244225
```

correctly claiming to predict 73.55% of the time & only detecting about 72.24% of the 5s

F_1 Score

F_1 Score is a **harmonic mean** of Precision and recall. The reason it uses harmonic mean is because the harmonic means gives more weightage to low values. As a result, the F1 score is high only when both precision & recall are high

$$F_1 = \frac{2}{(\frac{1}{precision} + \frac{1}{recall})} = 2X\frac{precision X recall}{precision + recall} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

In [53]:

```
from sklearn.metrics import f1_score

f1_score(y_train_5, y_train_pred)
```

Out[53]:

```
0.728937728937729
```

Precision / Recall Tradeoff:

Tadeoff: `increasing the precision reduces the recall and vice-versa`

In [54]:

```
# calculating the decision scores

y_scores = sgd_clf.decision_function([some_digit])
y_scores
```

Out[54]:

```
array([103853.01219271])
```

In [55]:

```
threshold = 0
```

In [56]:

```
y_some_digit_pred = (y_scores > threshold)
y_some_digit_pred
```

Out[56]:

```
array([ True])
```

In [57]:

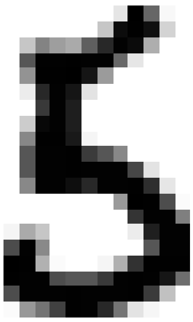
```
threshold = 200000

y_some_digit_pred = (y_scores > threshold)
y_some_digit_pred
```

Out[57]:

```
array([False])
```

In [58]:

```
plot_digit(some_digit)
```



In [59]:

```
y_scores = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3, method="decision_functi
on")
```

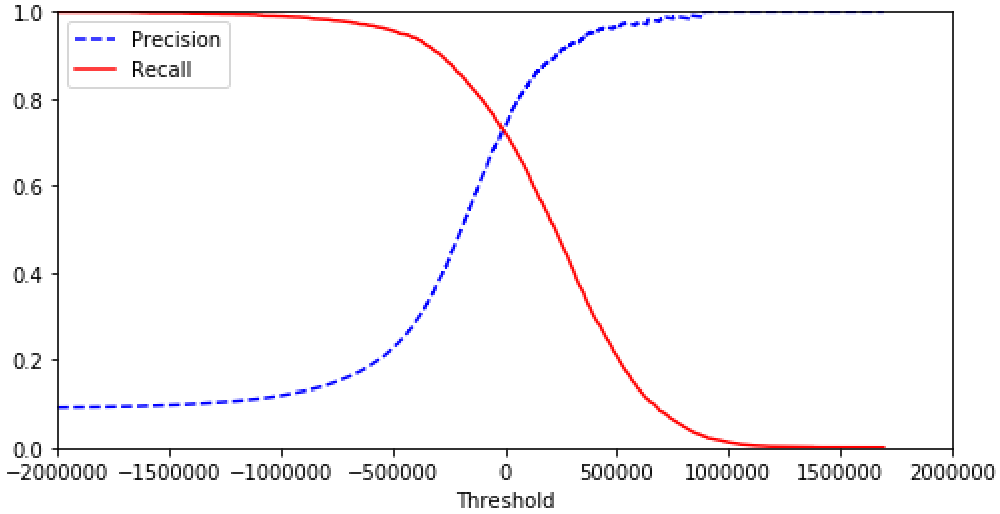
```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:
235: DataConversionWarning: A column-vector y was passed when a 1d array w
as expected. Please change the shape of y to (n_samples, ), for example us
ing ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
```

In [60]:

```
from sklearn.metrics import precision_recall_curve

precisions, recalls, thresholds = precision_recall_curve(y_train_5, y_scores)
```

In [61]:

```
def plot_precision_recall_vs_threshold(precisions, recalls, thresholds):
    plt.plot(thresholds, precisions[:-1], "b--", label="Precision")
    plt.plot(thresholds, recalls[:-1], "r-", label="Recall")
    plt.xlabel("Threshold")
    plt.legend(loc="upper left")
    plt.ylim([0,1])


plt.figure(figsize=(8,4))
plot_precision_recall_vs_threshold(precisions, recalls, thresholds)
plt.xlim([-2000000, 2000000])
plt.show()
```



In [62]:

```
from sklearn.metrics import precision_score, recall_score

y_train_2M = (y_scores > 200000)

precision_score(y_train_5, y_train_2M)
```

Out[62]:

0.8864653243847874

In [63]:

```
recall_score(y_train_5, y_train_2M)
```

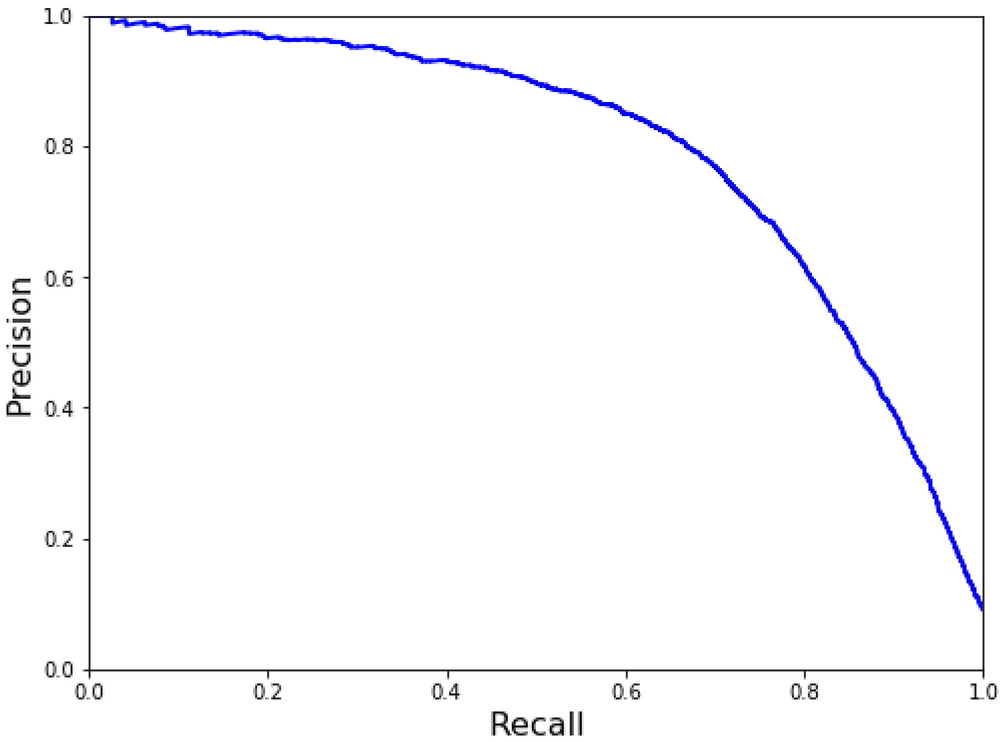
Out[63]:

0.523102310231023

another approach would be to plot precision directly against the model's recall

In [64]:

```
def plot_percision_vs_recall(predictions, recalls):
    plt.plot(recalls, precisions, "b-", linewidth=2)
    plt.xlabel("Recall", fontsize=16)
    plt.ylabel("Precision", fontsize=16)
    plt.axis([0,1,0,1])

plt.figure(figsize=(8,6))
plot_percision_vs_recall(precisions, recalls)
plt.show()
```



ROC (Receiver Operating Characteristics)

its basically a curve b/w True Positive Rate (i.e. Recall) and False Positive Rate.

The FPR is the ratio of -ve instances that were incorrectly classified as positive. It is equal to $1 - TNR$, which is the ratio of -ve instances that are correctly classified as negative.

TNR are aka **specificity**

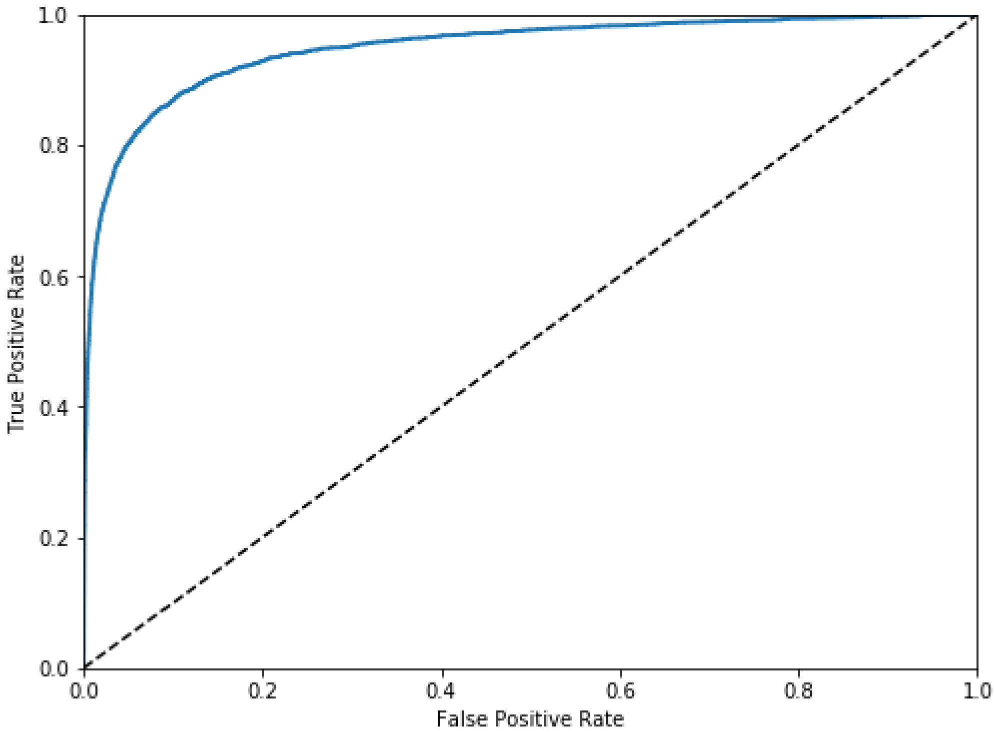
```
ROC Curve is basically b/w TPR (recall) versus 1-specificity
```

In [65]:

```
from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(y_train_5, y_scores)
```

In [66]:

```
def plot_roc_curve(fpr, tpr, label=None):
    plt.plot(fpr, tpr, linewidth=2, label=label)
    plt.plot([0,1],[0,1], 'k--')
    plt.axis([0,1,0,1])
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")


plt.figure(figsize=(8, 6))
plot_roc_curve(fpr, tpr)
plt.show()
```



ROC AUC (Area Under the Curve):

In [67]:

```
from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(y_train_5, y_scores)
```

In [68]:

```
from sklearn.metrics import roc_auc_score

roc_auc_score(y_train_5, y_scores)
```

Out[68]:

```
0.9483319892342521
```

ROC AUC when equal to 1, the classifier is a perfect classifier whereas a purely random classifier will have ROC AUC equal to 0.5

In [69]:

```
# lets try another classifier

from sklearn.ensemble import RandomForestClassifier

rf_clf = RandomForestClassifier(random_state=29)
```

In [70]:

```
y_probas_forest = cross_val_predict(rf_clf, X_train, y_train_5, cv=3, method="predict_p
roba")
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:
235: DataConversionWarning: A column-vector y was passed when a 1d array w
as expected. Please change the shape of y to (n_samples, ), for example us
ing ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in ve
rsion 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in ve
rsion 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in ve
rsion 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

In [71]:

```
y_probas_forest[:5]
```

Out[71]:

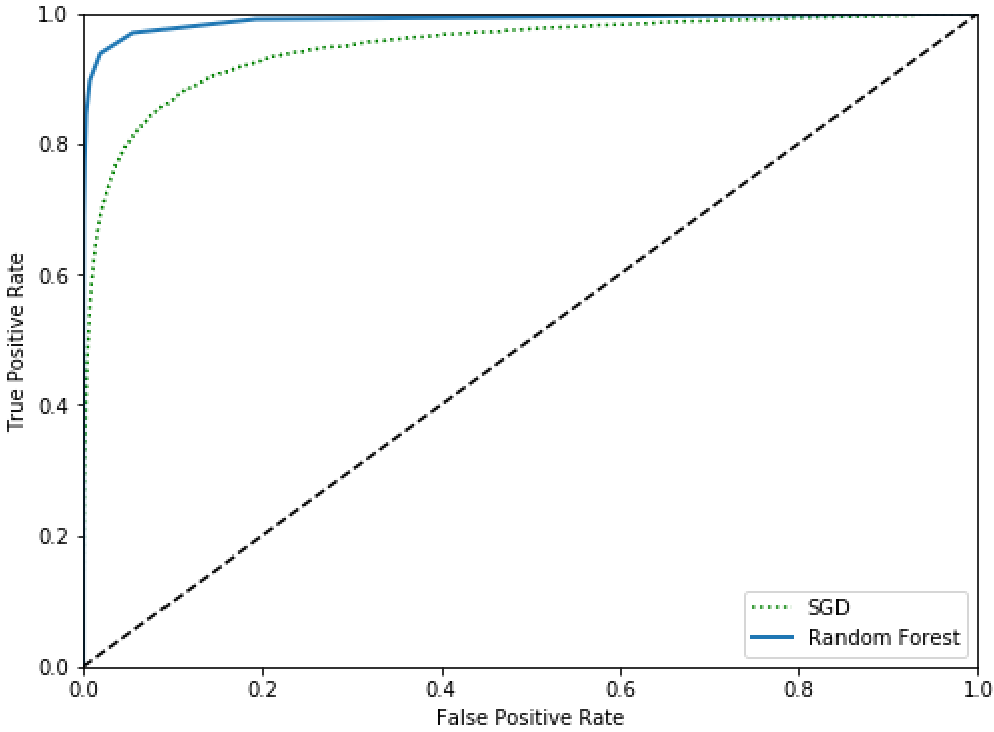
```
array([[0.9, 0.1],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.9, 0.1]])
```

In [72]:

```
# since ROC curve, needs scores and not probabilities, we'll only look at the probabili
tty score of all positive class
y_scores_forest = y_probas_forest[:,-1]
fpr_forest, tpr_forest, thresholds_forest = roc_curve(y_train_5, y_scores_forest)
```

In [73]:

```
plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, "g:", label="SGD")
plot_roc_curve(fpr_forest, tpr_forest, "Random Forest")
plt.legend(loc="lower right")
plt.show()
```



In [74]:

```
roc_auc_score(y_train_5, y_scores_forest)
```

Out[74]:

```
0.9896922768822515
```

Multiclass Classification

two approaches based on which all the classification algorithm work for multiclass classification problems:
OvA (One vs All) approach
OvO (One vs One) approach.

Almost all the linear models (like SVM, Logit) use OvA approach to do multiclass classification

In [76]:

```
sgd_clf_multiclass = SGDClassifier(random_state=29)
sgd_clf_multiclass.fit(X_train, y_train)
```

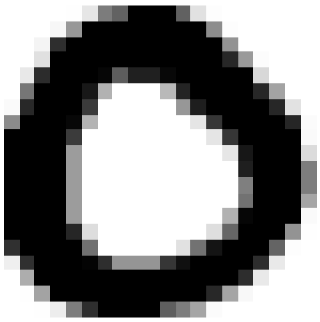
```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
1: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

Out[76]:

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=29, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False)
```

In [83]:

```
some_digit = X.iloc[4]
plot_digit(some_digit)

some_digit_scores  = sgd_clf_multiclass.decision_function([some_digit])
some_digit_scores
```



Out[83]:

```
array([[  802325.27201187, -2443588.60849563,  -925759.97107992,
         -813057.48713559, -1754991.27925447, -1510896.30585057,
        -1330687.76618897, -1708996.29203621,  -790423.48805796,
        -2088557.04923088]])
```

In [84]:

```
import numpy as np

np.argmax(some_digit_scores)
```

Out[84]:

```
0
```

In [85]:

```
# what classes actually exists in your y-label
sgd_clf_multiclass.classes_
```

Out[85]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], dtype=int64)
```

In [86]:

```
from sklearn.multiclass import OneVsOneClassifier

ovo_clf = OneVsOneClassifier(SGDClassifier(random_state=29))
ovo_clf.fit(X_train, y_train)
ovo_clf.predict([some_digit])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
1: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
```

```
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
```

```
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
```

```
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
```

```
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been add
ed in SGDClassifier in 0.19. If both are left unset, they default to max_i
ter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=100
0. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
```

Out[86]:

```
array([0], dtype=int64)
```

In [87]:

```
len(ovo_clf.estimators_)
```

Out[87]:

```
45
```

In [88]:

```
ovo_clf.estimators_
```

Out[88]:

```
(SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
```

```
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
```

```
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
```

```
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
```

```
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
```

```
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False),
 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
        early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
        l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=Non
e,
        n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
        power_t=0.5, random_state=29, shuffle=True, tol=None,
        validation_fraction=0.1, verbose=0, warm_start=False))
```

Multilabel Classification

In [90]:

```
y_train_large = (y_train >= 6)
y_train_odd = (y_train % 2 == 1)
y_multilabel = np.c_[y_train_large, y_train_odd]
```

In [91]:

```
y_multilabel
```

Out[91]:

```
array([[False, False],
       [ True, False],
       [False, False],
       ...,
       [False, False],
       [ True,  True],
       [False, False]])
```

In [92]:

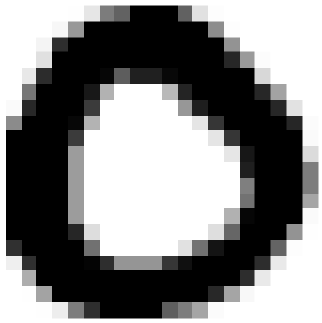
```
from sklearn.neighbors import KNeighborsClassifier

knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_multilabel)
```

Out[92]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='uniform')
```

In [94]:

```
plot_digit(some_digit)
```



In [93]:

```
knn_clf.predict([some_digit])
```

Out[93]:

```
array([[False, False]])
```

as 0 is less than 6 (False) & 0 is an odd digit (False)

In [ ]:

```

```