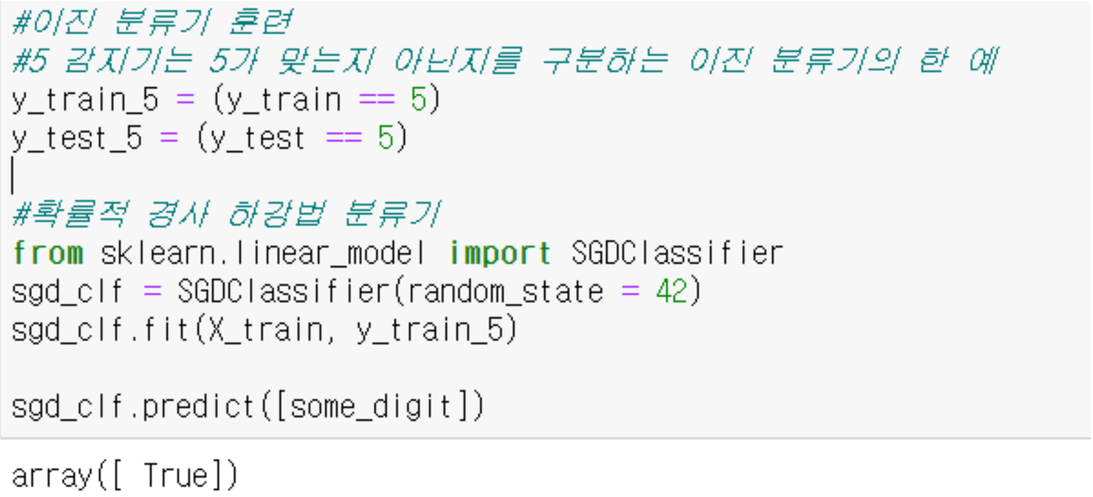
Classification

Data Set: MNIST

- Train vs test = 6:1
- binary classifier (이진 분류기)
- SGDClassifier



```
#이진 분류기 훈련
#5 감지기는 5가 맞는지 아닌지를 구분하는 이진 분류기의 한 예
y_train_5 = (y_train == 5)
y_test_5 = (y_test == 5)

#확률적 경사 하강법 분류기
from sklearn.linear_model import SGDClassifier
sgd_clf = SGDClassifier(random_state = 42)
sgd_clf.fit(X_train, y_train_5)

sgd_clf.predict([some_digit])
```

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

Classification evaluation

K fold cross validation

- Divide into K-fold and predict the result

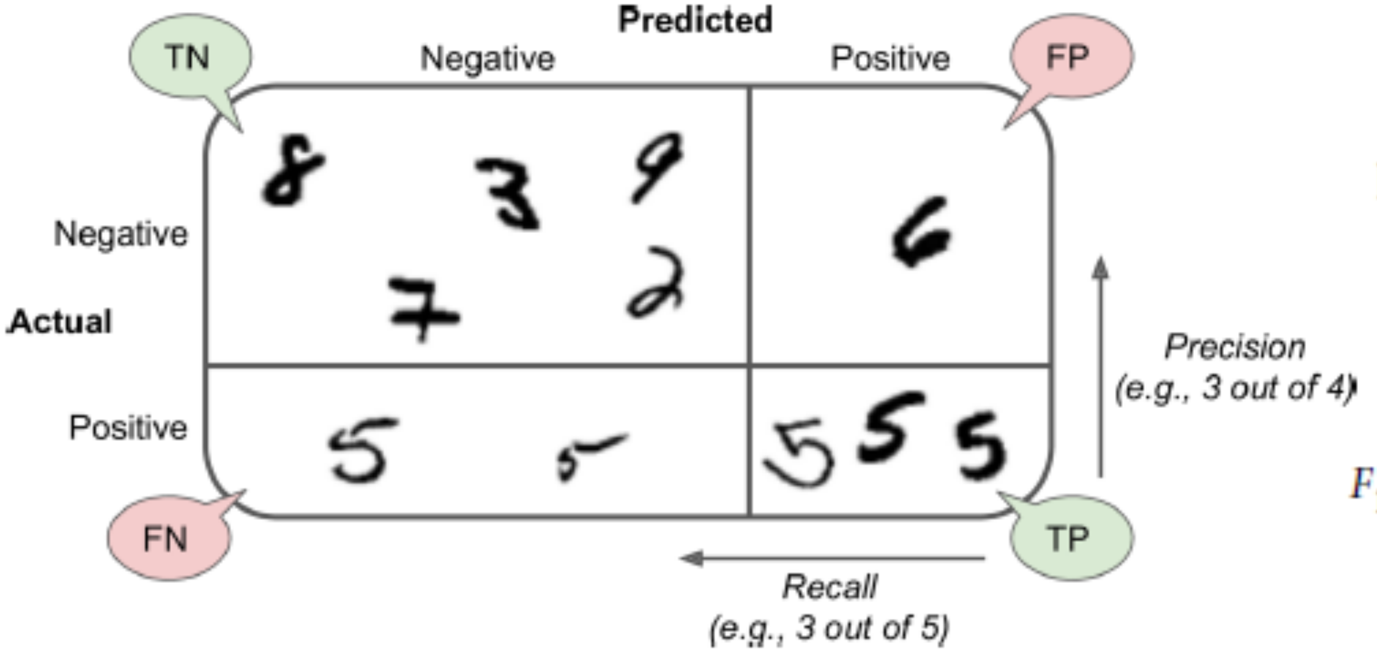
- Not good to use unbalanced data set

vs

confusion matrix

- Num of times A samples are classified as Class B

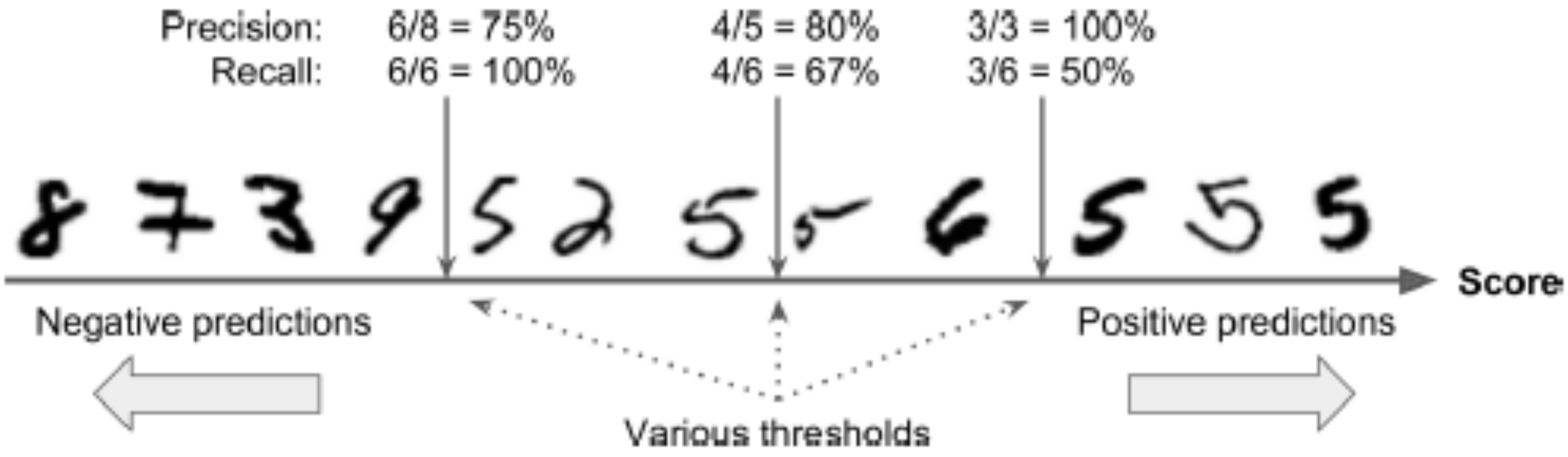
Confusion Matrix



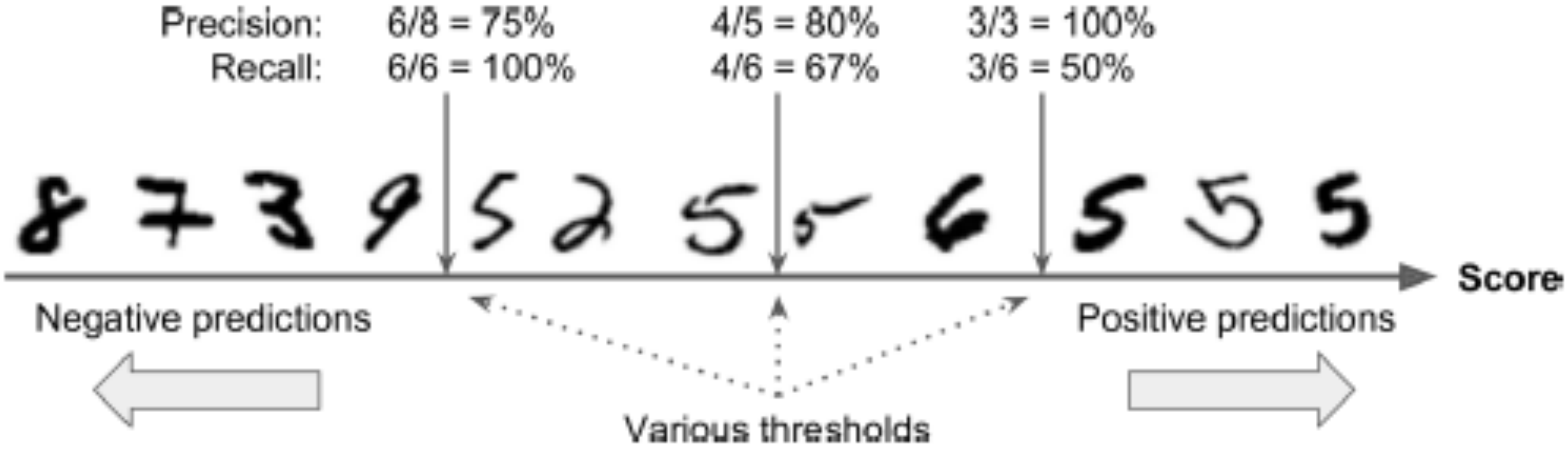
$$\text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = \frac{TP}{TP + FN}$$

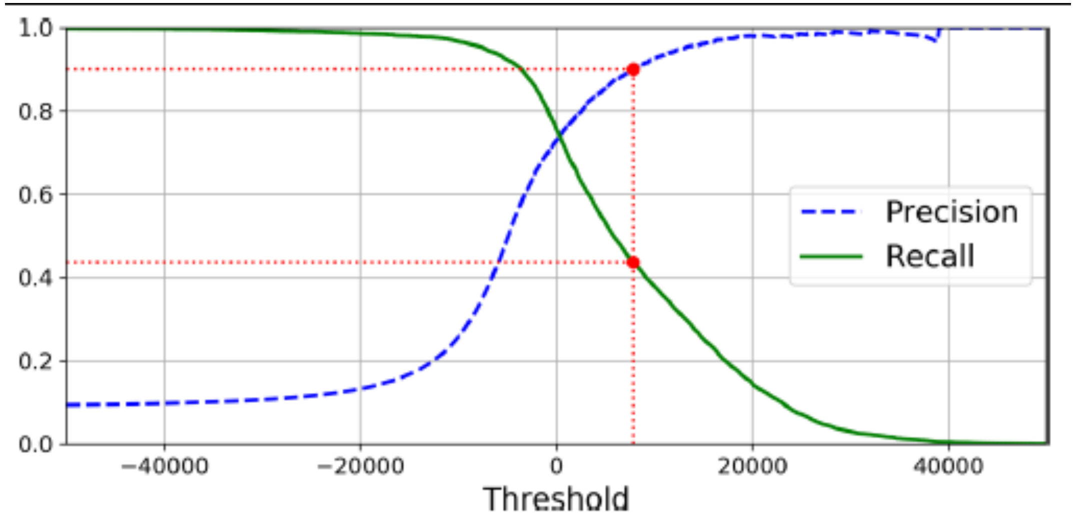
$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

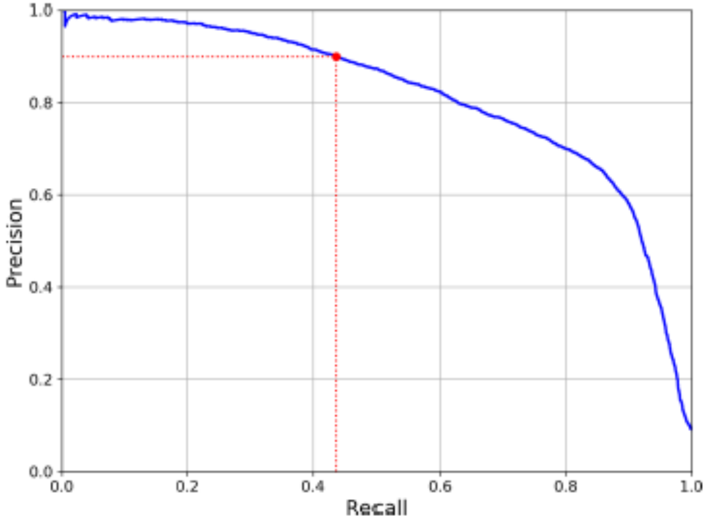
Precision/Recall tradeoff



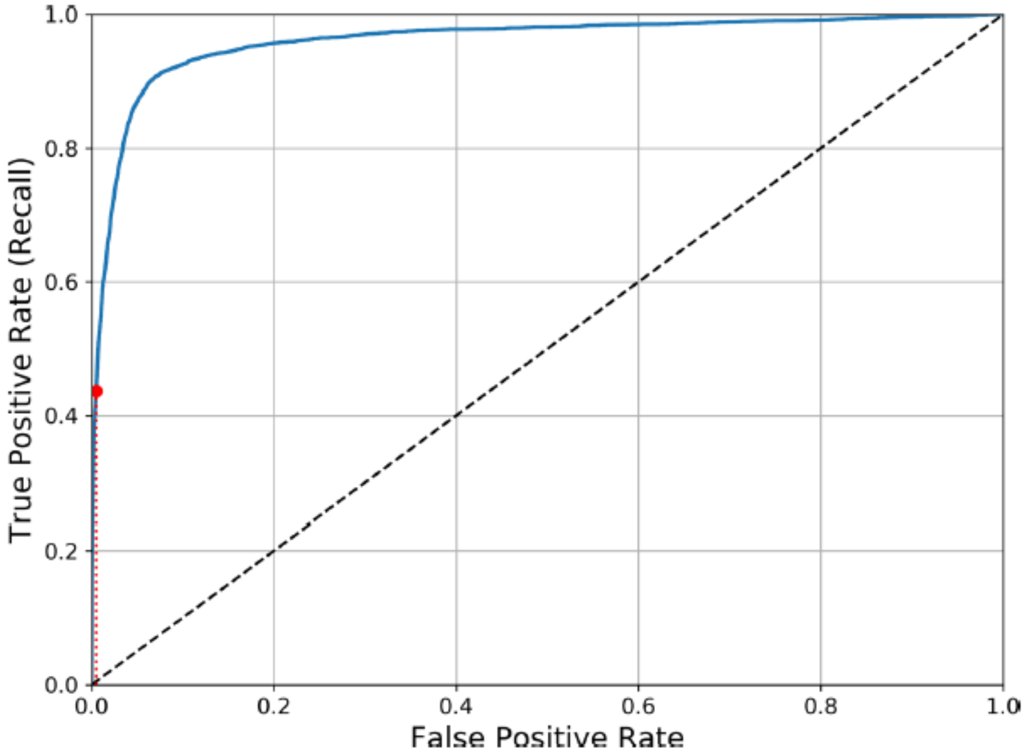
Precision/Recall tradeoff







ROC Curve

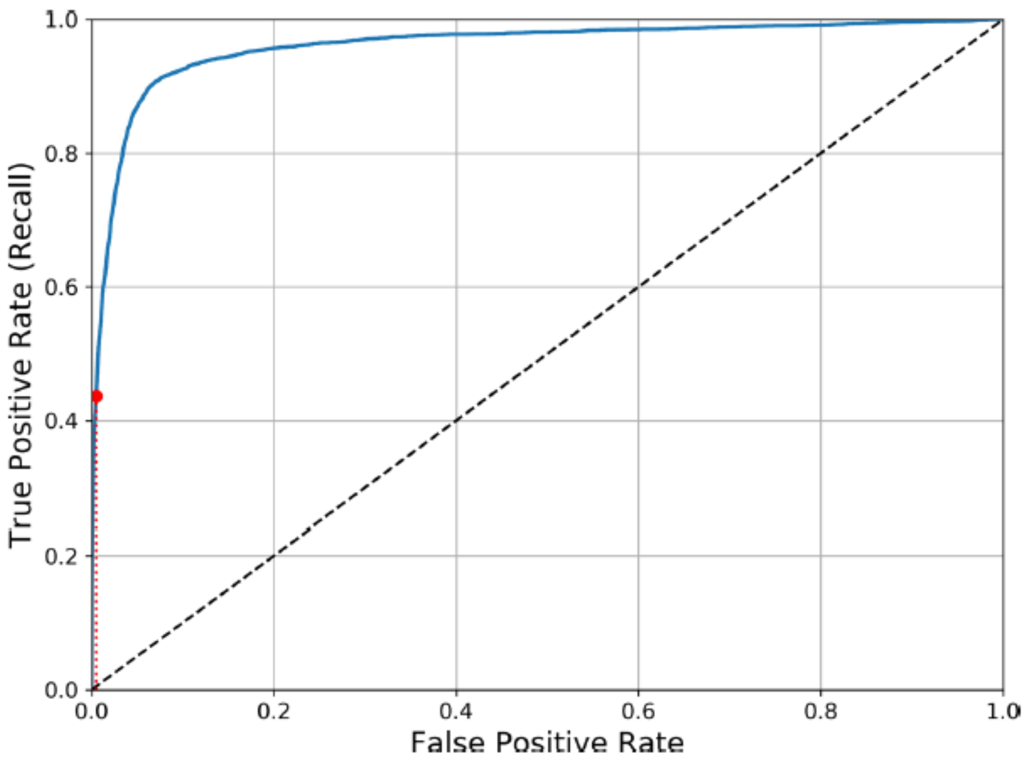


= 1- 진짜 음성 비율 (TNR)

가운데 점선 – Random Classifier

Measure area under the curve(AUC) to compare classifier

ROC Curve



= 1- 진짜 음성 비율
(TNR)

가운데 점선 – Random Classifier

Measure area under the curve(AUC)

to compare classifier

Multiclass classifier (다중 분류)

OvR(one-verus-the-rest)

-Training multiple binary classifiers

- Training 10 binary classifier
- if some has a high decision score

-> then select that model

OvO(one-verus-one)

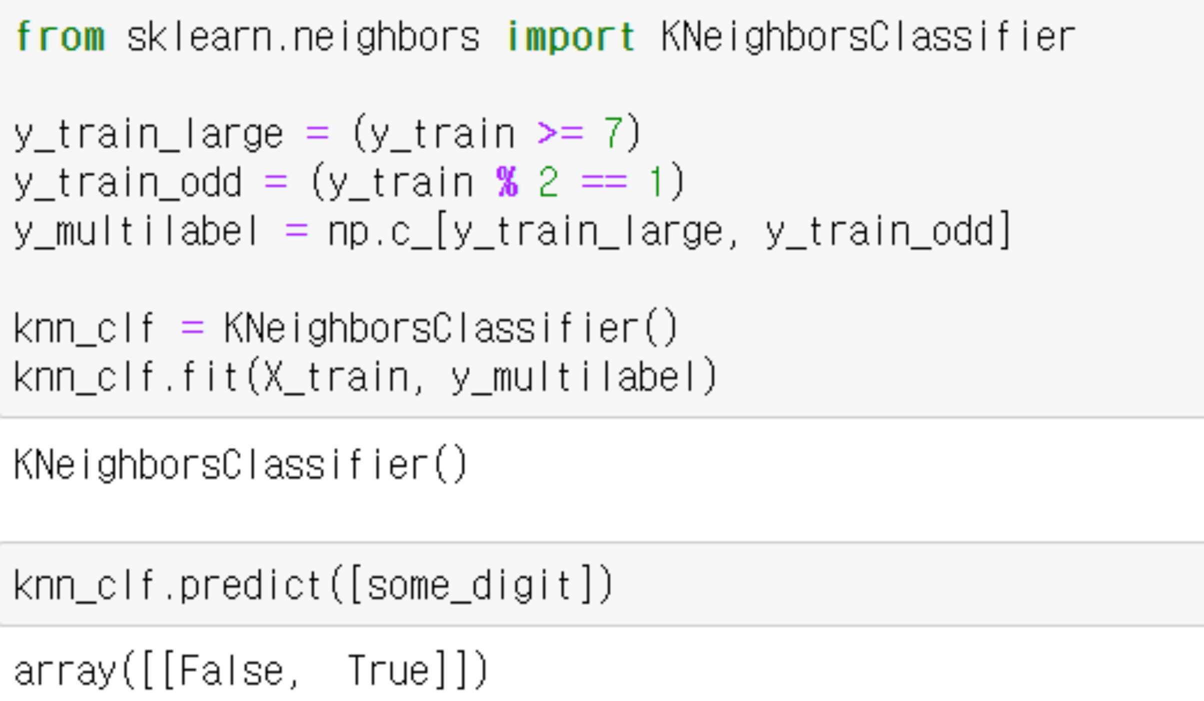
- Training a binary classifier for each combination of numbers

- Training 45 classifier
- if some has a high score

-> then select that model

Multilabel classification (다중 레이블 분류)

If there is a picture with many people, how can we do?



```
from sklearn.neighbors import KNeighborsClassifier

y_train_large = (y_train >= 7)
y_train_odd = (y_train % 2 == 1)
y_multilabel = np.c_[y_train_large, y_train_odd]

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

```
KNeighborsClassifier()
```

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

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