Machine learning 01

Byung Chang Chung

Gyeongsang National University

bcchung@gnu.ac.kr

What is machine learning (ML)?

- Field of study that gives computers the ability to learn without being explicitly programmed
  - Arthur Samuel 1959
- Machine learning is the study of computer algorithms that allow computer programs to automatically improve through experience
  - Tom Mitchell 1997

Why we use ML?

- Hard to maintenance
  - in traditional programming
    - long and complicated rules

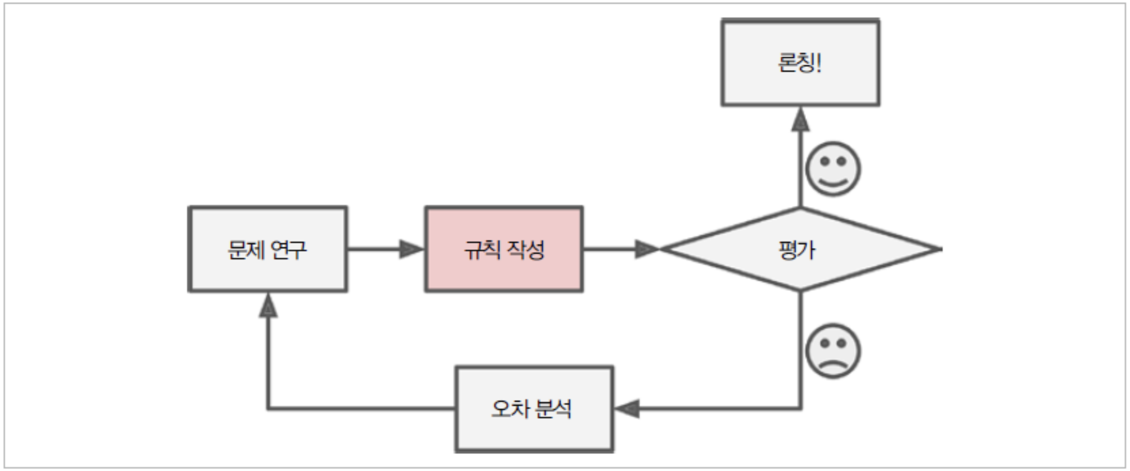


그림 1-1 전통적인 접근 방법

Why we use ML?

- Hard to maintenance
  - in machine learning
    - simplify code and perform better
    - adaptation to new data

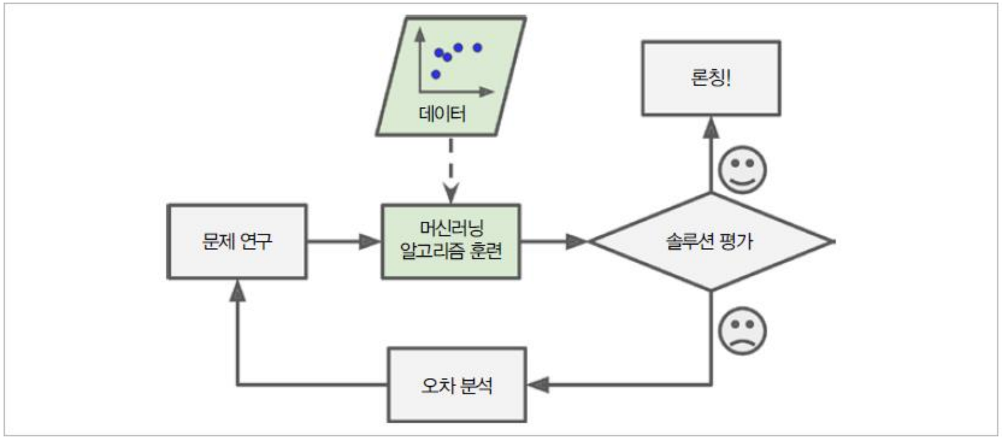


그림 1-2 머신러닝 접근 방법

Application examples

- Examples
  - image classification
  - disease detection
  - natural language processing
  - regression
  - voice recognition
  - anomaly detection

Category of ML

- Way of categorization
  - whether it's training under human supervision or not?
  - whether you're learning in real time or not?
  - similarity from previous experience or prediction from modelling?

Category of ML

- Supervised or not
  - Supervised learning
    - The training data injected into the algorithm includes the desired answer called label

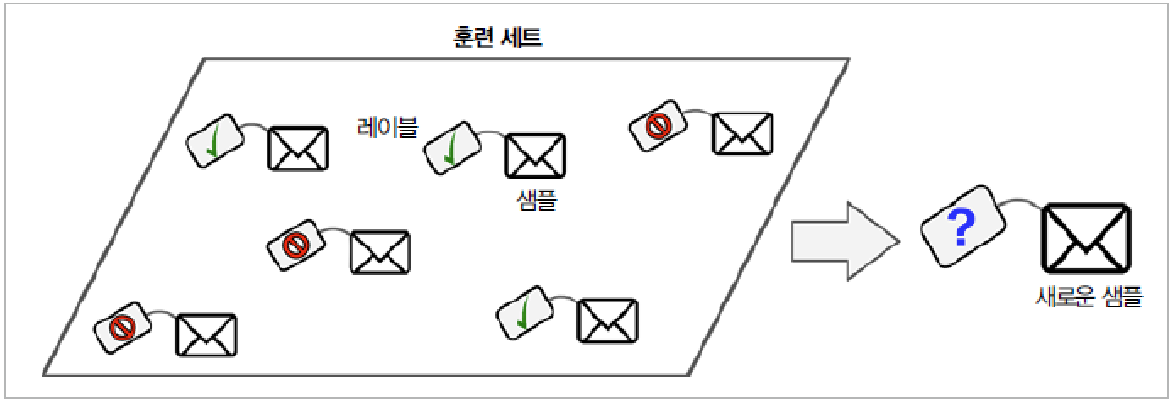


그림 1-5 스팸 분류를 위한 레이블된 훈련 세트(지도 학습의 예)

Category of ML

- Supervised or not
    - supervised learning
        - k-nearest neighbor
        - linear regression
        - logistic regression
        - support vector machine
        - decision tree and random forest
        - neural network

Category of ML

- Supervised or not
  - unsupervised learning
    - no label on the training data
    - the system must learn without any help

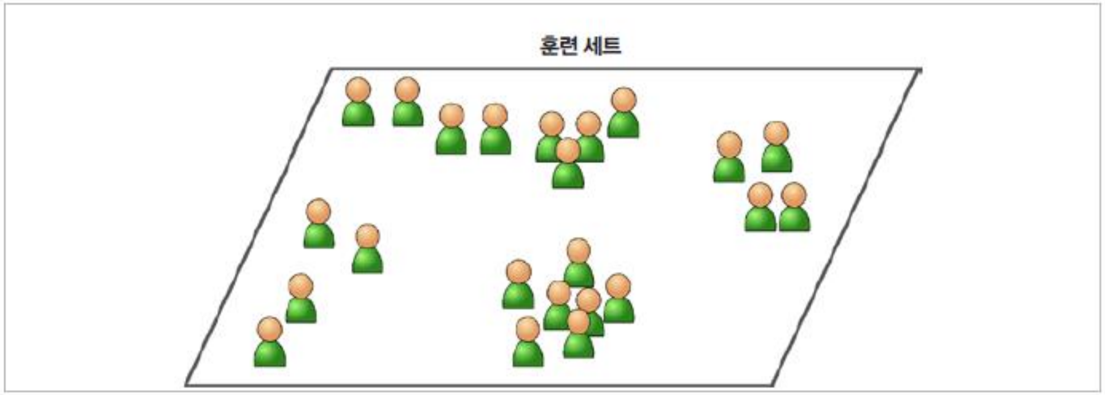


그림 1-7 비지도 학습에서 레이블 없는 훈련 세트

Category of ML

- Supervised or not
  - Unsupervised learning
    - clustering
    - visualization and dimensionality reduction
    - association rule learning

Category of ML

- real-time or not
    - batch learning
        - system can't learn gradually
    - online learning
        - system is trained by sequentially injecting data one by one
        - or in small batches called mini-batch

Category of ML

- real-time or not
  - online learning
    - system is trained by sequentially injecting data one by one
    - or in small batches called mini-batch

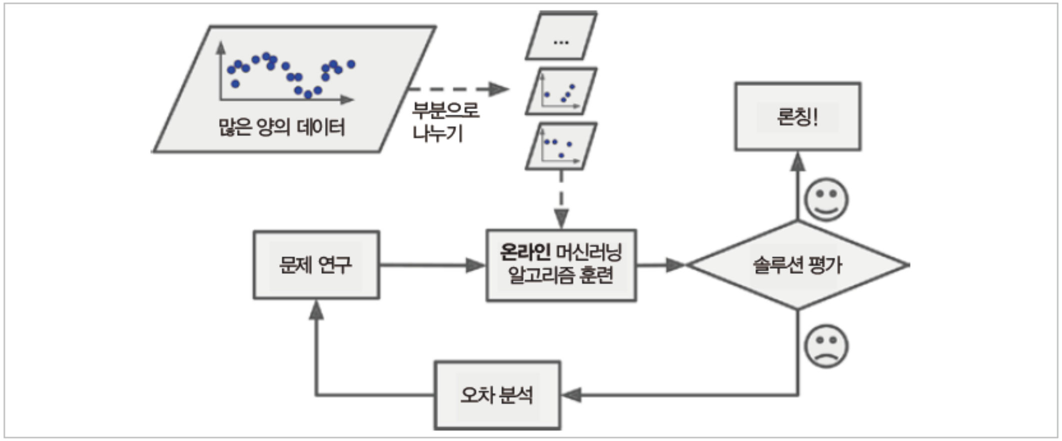


그림 1-14 온라인 학습을 사용한 대량의 데이터 처리

Category of ML

- similarity or modelling
  - instance-based learning
  - model-based learning

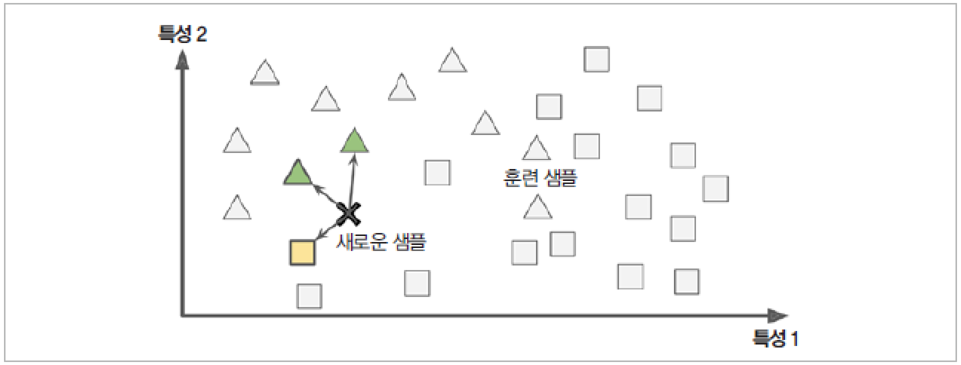


그림 1-15 사례 기반 학습

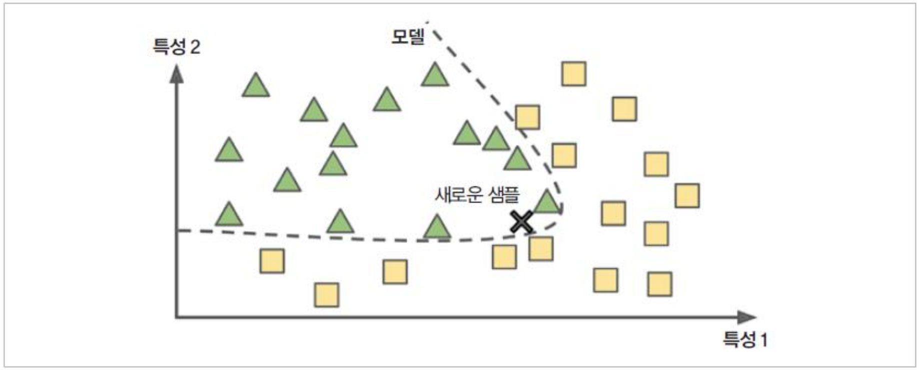
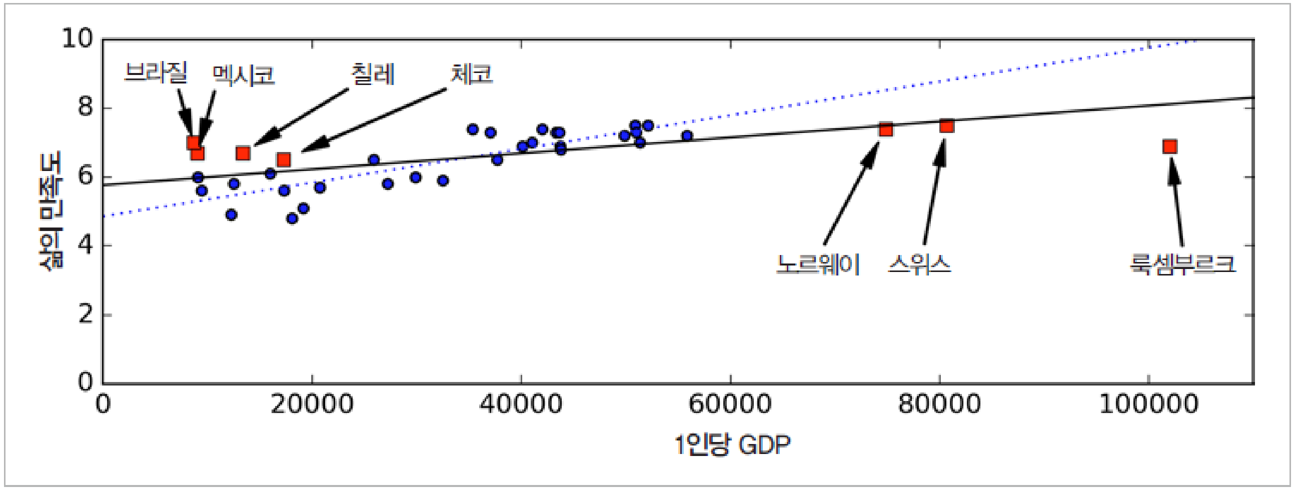


그림 1-16 모델 기반 학습

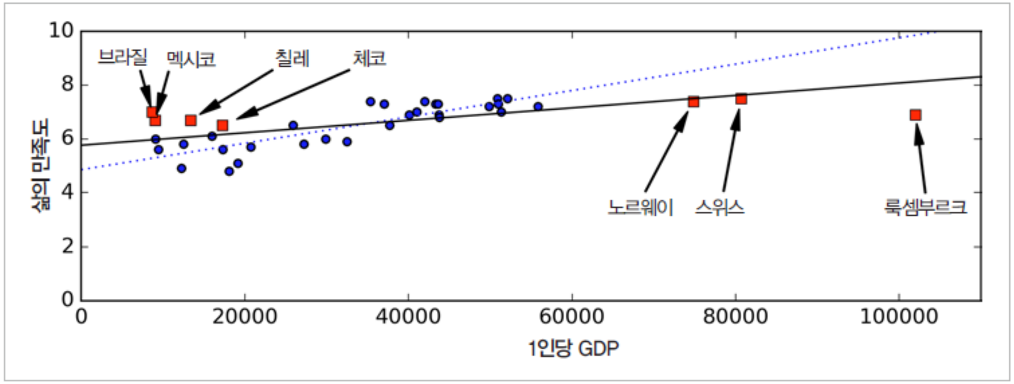
Major objectives

- Select a learning algorithm
- Train it on certain data



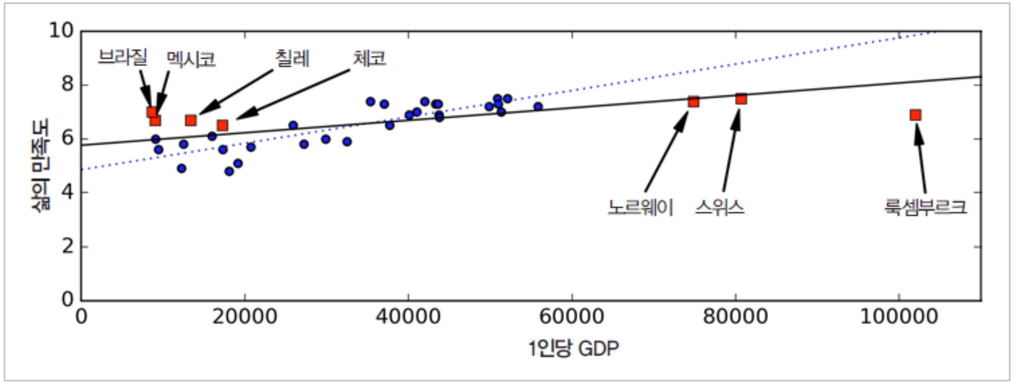
Major objectives

- Bad data
  - not enough training data
  - training data without representation
  - irrelevant characteristics



Major objectives

- Bad algorithm
  - overfitting
  - underfitting



Test and validation

- Dividing it into two sets: training set and a test set
  - 80% of the data is separated for training and 20% for testing
    - error rate to new sample: generalization error or external sample error
    - predicting how well the model will work for a new sample that has never been seen before

Training with real data

- How to obtain open dataset?
  - A famous public data store
    - UC Irvine Machine Learning Storage (http://archive.ics.uci.edu/ml)
    - Kaggle dataset (http://www.kaggle.com/datasets)
    - Amazon AWS dataset (https://registration.opendata.aws)
  - Listed public data stores
    - Data Portal Data Ports (http://dataportals.org)
    - Open Data Monitor Open Data Monitor (http://opendatamonitor.eu)
    - Quandl (http://quandl.com)

Training with real data

- California Housing Prices

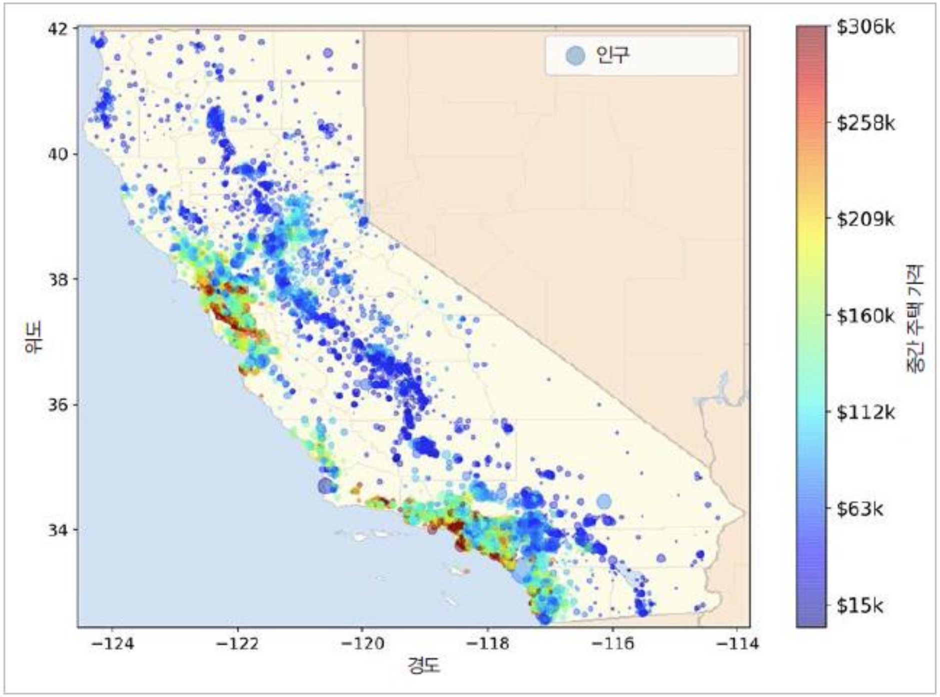
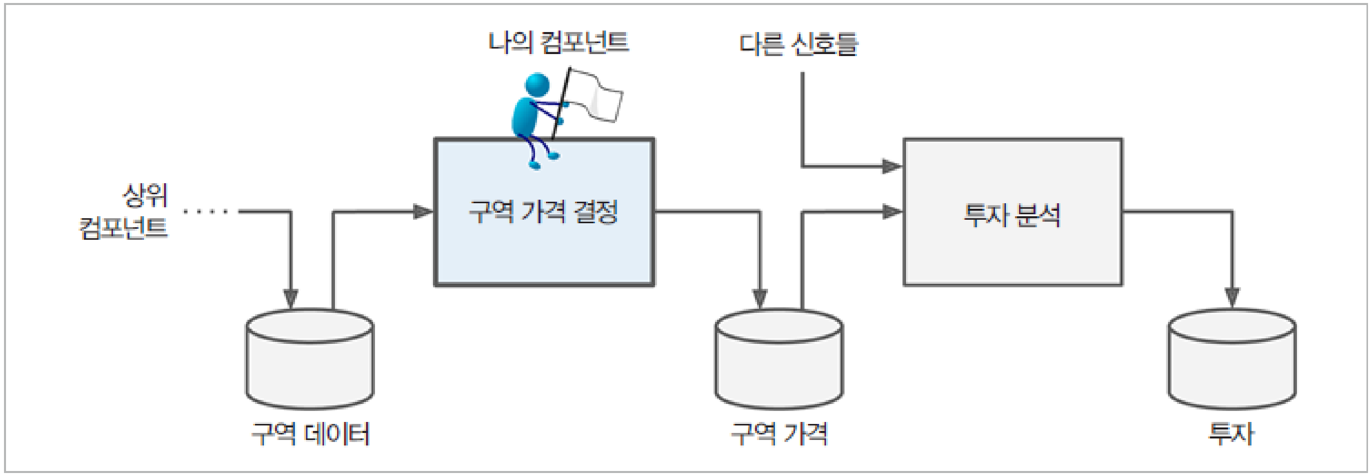


그림 2-1 캘리포니아 주택 가격

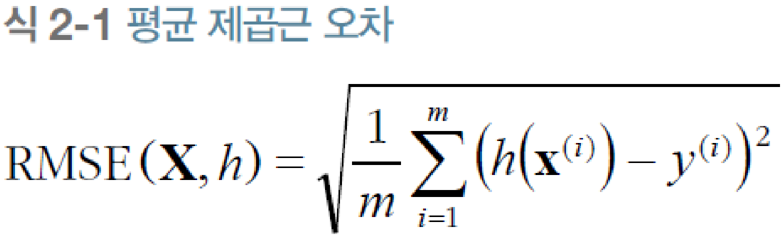
Problem statement

- California Housing Prices
  - creating a California housing price model
    - predicting the price of an intermediate house in an area



Problem statement

- California Housing Prices
  - creating a California housing price model
    - performance metric



식 2-1 평균 제곱근 오차

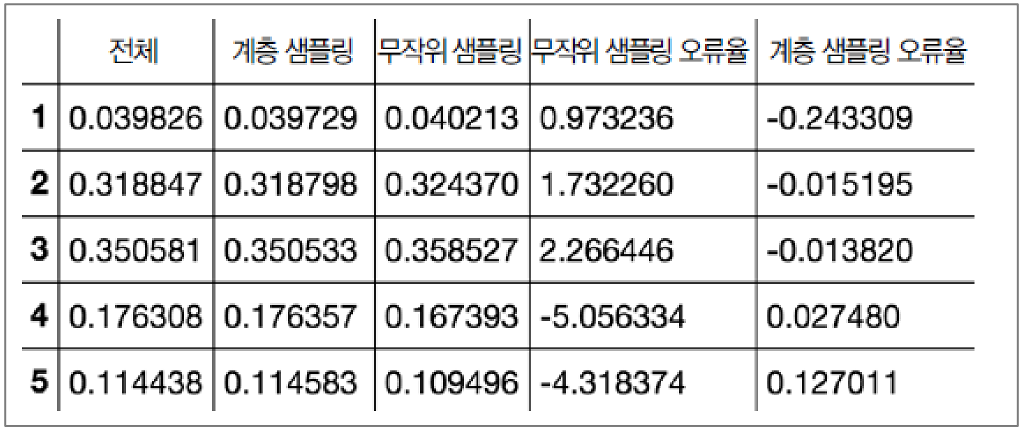
$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

Importing data

- California Housing Prices
  - example code on https://github.com/rickiepark/handson-ml2
  - installing tools: python, jupyter, numpy, pandas, matplotlib, scikit-learn
  - data download: housing.csv from StatLib
    - http://lib.stat.cmu.edu/datasets/
    - https://goo.gl/QgRbUL

Importing data

- Training-test split
  - stratified sampling
  - random sampling



	전체	계층 샘플링	무작위 샘플링	무작위 샘플링 오류율	계층 샘플링 오류율
1	0.039826	0.039729	0.040213	0.973236	-0.243309
2	0.318847	0.318798	0.324370	1.732260	-0.015195
3	0.350581	0.350533	0.358527	2.266446	-0.013820
4	0.176308	0.176357	0.167393	-5.056334	0.027480
5	0.114438	0.114583	0.109496	-4.318374	0.127011

Data visualization

- Plotting for understanding data properties

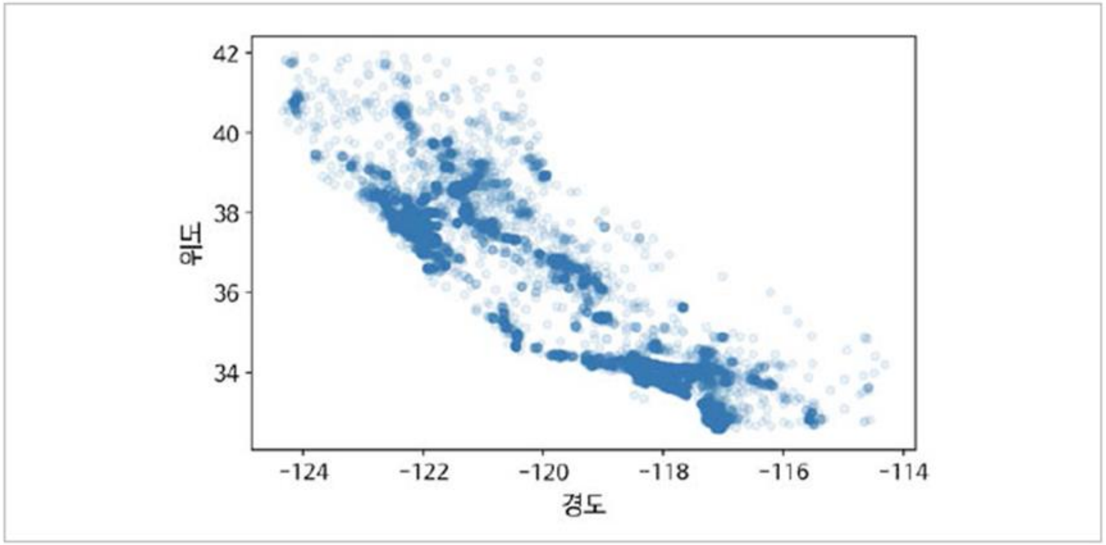


그림 2-12 밀집된 지역이 잘 부각된 산점도

Data visualization

- Plotting for understanding data properties

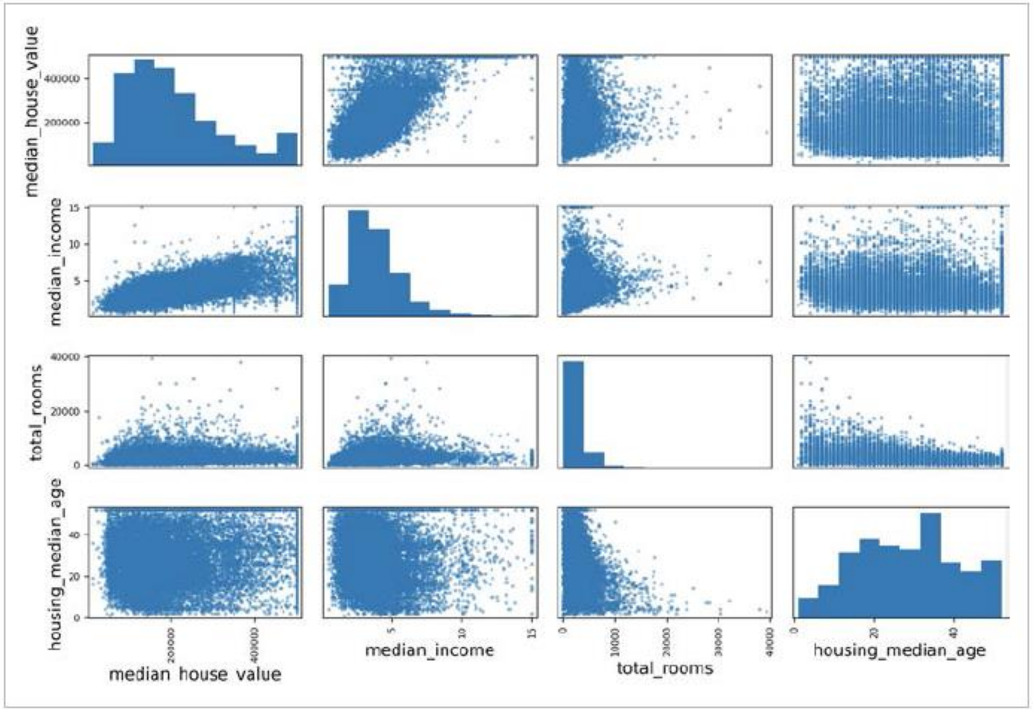


그림 2-15 이 산점도 행렬은 다른 수치형 특성에 대한 각 수치형 특성의 산점도와 각 수치형 특성의 히스토그램을 출력합니다.

Data visualization

- Plotting for understanding data properties

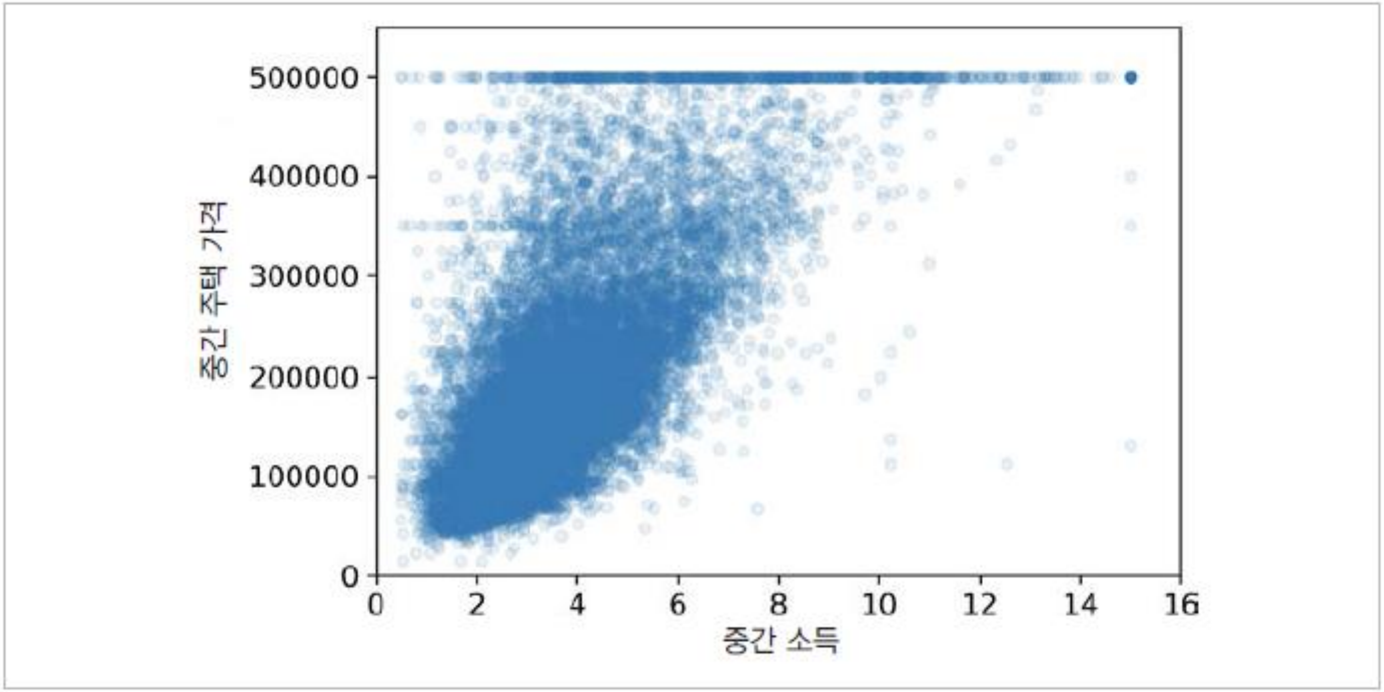


그림 2-16 중간 소득 대 중간 주택 가격

Data processing

- Automate data preparation by creating functions
  - data conversion is easily repeated for any dataset
  - gradually build transform libraries for future projects
  - easy to try different data conversions
  - convenient to see which combination is the best

Data processing

- Data purification
  - since most machine learning algorithms cannot deal with missing characteristics, a function capable of handling them is required

- Handling text and categorical characteristics

Data processing

- Feature scaling
  - min-max scaling
  - standardization

- Pipelining
  - if there are many conversion steps and it's complicated, use the pipeline

Model selection and training

- Selecting ML model
  - linear regression
  - decision tree
  - random forest model

- Taking cross validation
  - k-fold cross validation

Tuning ML model

- Grid search
  - Finding appropriate hyperparameter
  - Using GridSearchCV in scikit-learn

- Random search
  - Using RandomizedSearchCV

Tuning ML model

- Ensemble method
- Finding feature importance
- Evaluation with test dataset

Launching and maintenance

- Distribute models to commercial environments

- Write the monitoring code
  - check the real-time performance of the system at regular intervals
  - notify the alarm when the performance is poor

Launching and maintenance

- Update your dataset and retrain your model regularly
  - regularly collect new data and add labels
  - write a script that trains the model and automatically fine-tune the hyperparameters
  - run this script automatically daily or weekly depending on the task

Check the entire process

- Example code on

  https://github.com/rickiepark/handson-ml2

Classification

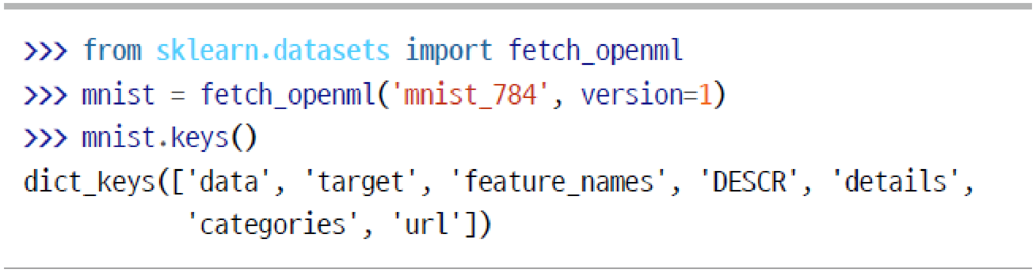
- Definition
  - a process related to categorization, the process in which ideas and objects are recognized, differentiated and understood

MNIST

- Modified National Institute of Standards and Technology database
  - a large database of handwritten digits that is commonly used for training various image processing systems

MNIST

- Downloading MNIST



```
>>> from sklearn.datasets import fetch_openml
>>> mnist = fetch_openml('mnist_784', version=1)
>>> mnist.keys()
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'details',
           'categories', 'url'])
```

  - DESCR key describes the dataset
  - data key with an array of samples consisting of one row and one column of characteristics
  - target key with label

MNIST

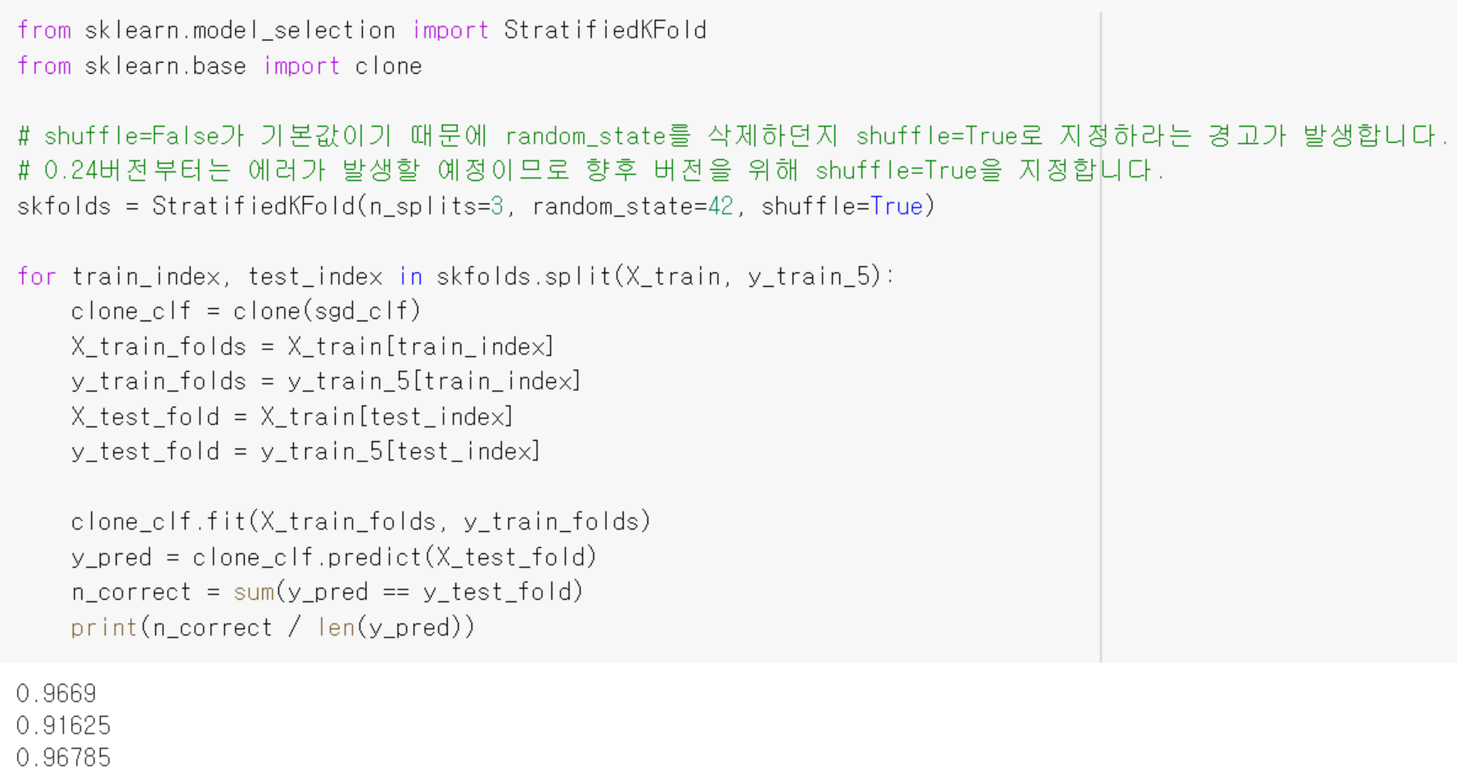
- MNIST keys
  - 70,000 images and each image has 784 characteristics (= 28 x 28 pixels)
  - Individual properties simply represent pixel intensities from 0 (white) to 255 (black)

Binary classifier

- Binary classifier
  - only identify the number 5
  - this "5-detector" separates the two classes, "5" and "not 5".
  - create a SGDClassifier model in scikit-learn
    - stochastic gradient descent (SGD)

Performance evaluation

- Cross validation

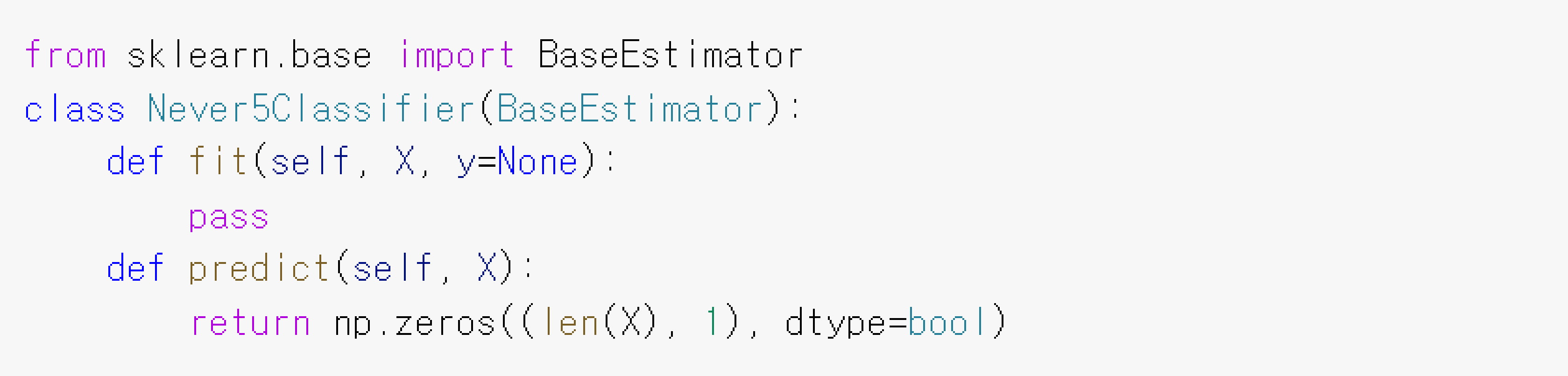


```
from sklearn.model_selection import StratifiedKFold
from sklearn.base import clone

# shuffle=False가 기본값이기 때문에 random_state를 삭제하던지 shuffle=True로 지정하라는 경고가 발생합니다.
# 0.24버전부터는 에러가 발생할 예정이므로 향후 버전을 위해 shuffle=True을 지정합니다.
skfolds = StratifiedKFold(n_splits=3, random_state=42, shuffle=True)

for train_index, test_index in skfolds.split(X_train, y_train_5):
    clone_clf = clone(sgd_clf)
    X_train_folds = X_train[train_index]
    y_train_folds = y_train_5[train_index]
    X_test_fold = X_train[test_index]
    y_test_fold = y_train_5[test_index]

    clone_clf.fit(X_train_folds, y_train_folds)
    y_pred = clone_clf.predict(X_test_fold)
    n_correct = sum(y_pred == y_test_fold)
    print(n_correct / len(y_pred))
```

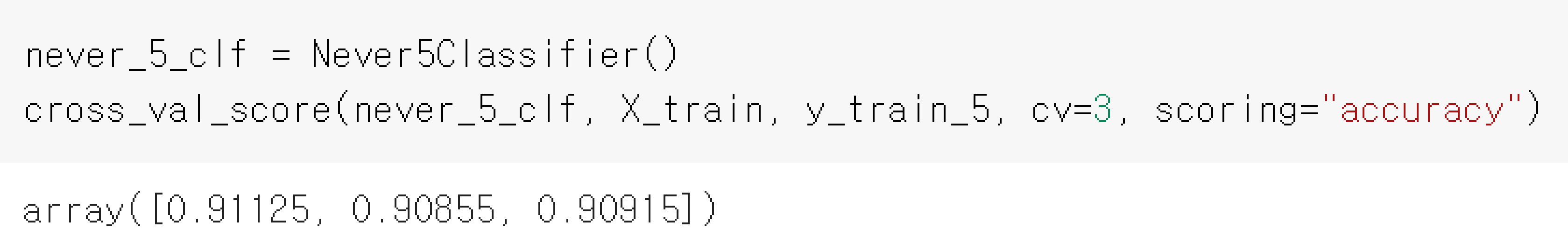
```
0.9669
0.91625
0.96785
```

Performance evaluation

- Cross validation



```python
from sklearn.base import BaseEstimator
class Never5Classifier(BaseEstimator):
    def fit(self, X, y=None):
        pass
    def predict(self, X):
        return np.zeros((len(X), 1), dtype=bool)
```

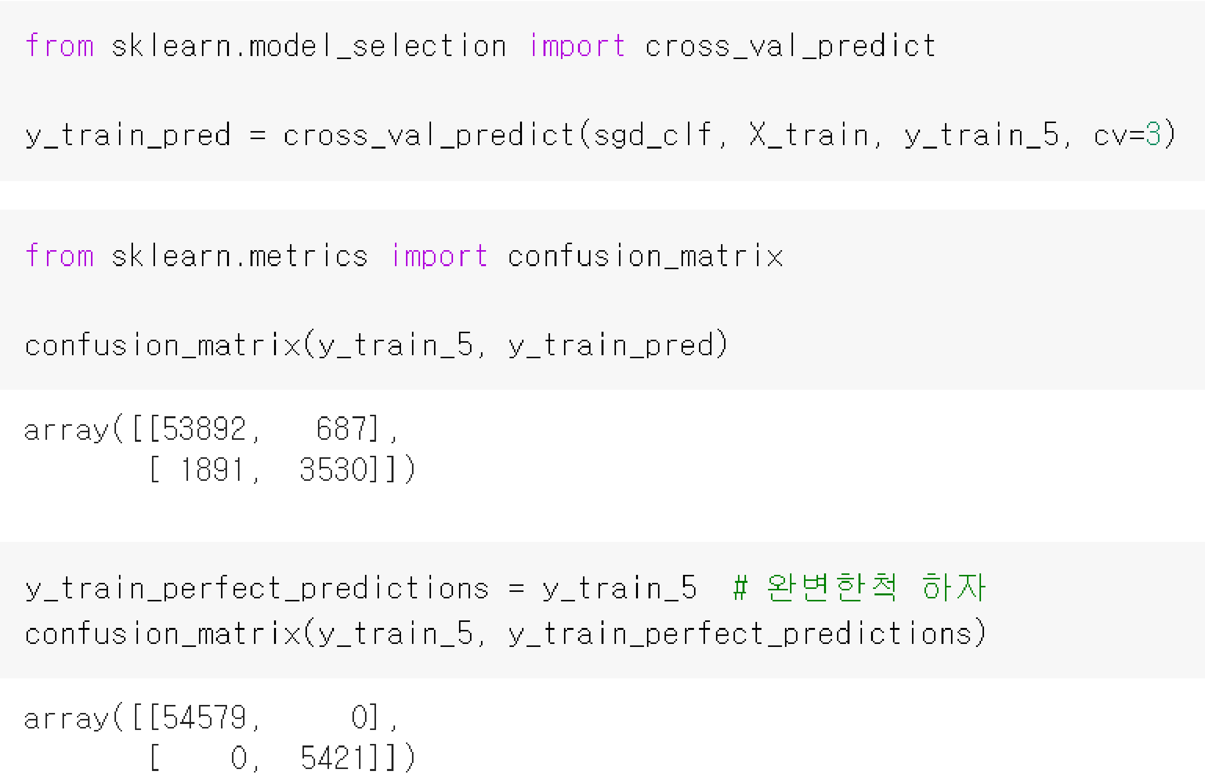


```python
never_5_clf = Never5Classifier()
cross_val_score(never_5_clf, X_train, y_train_5, cv=3, scoring="accuracy")
```

```
array([0.91125, 0.90855, 0.90915])
```

Performance evaluation

- Confusion matrix



```
from sklearn.model_selection import cross_val_predict

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

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_train_5, y_train_pred)
```

```
array([[53892,   687],
       [ 1891,  3530]])
```

```
y_train_perfect_predictions = y_train_5  # 완변한척 하자
confusion_matrix(y_train_5, y_train_perfect_predictions)
```

```
array([[54579,     0],
       [    0,  5421]])
```

Performance evaluation

- Confusion matrix

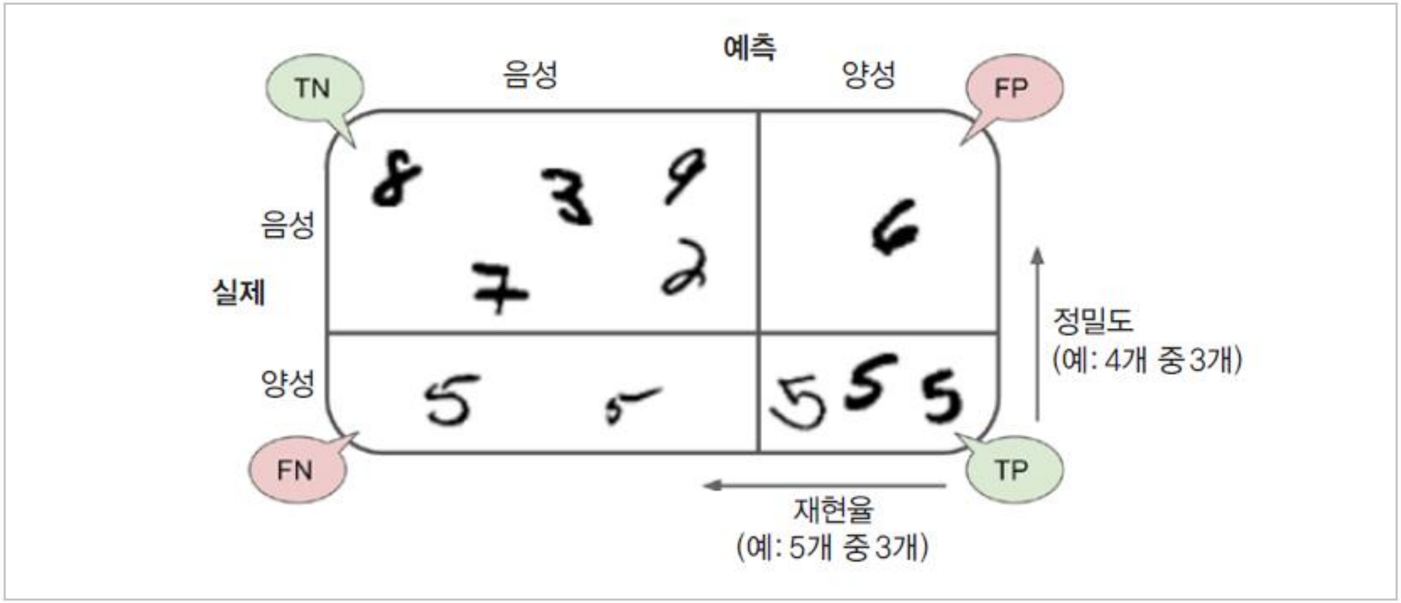


그림 3-2 이 오차 행렬 그림은 진짜 음성 샘플(왼쪽 위), 거짓 양성(오른쪽 위), 거짓 음성(왼쪽 아래), 진짜 양성(오른쪽 아래)를 보여줍니다.

Performance evaluation

- Precision and recall
  - precision: the fraction of relevant instances among the retrieved instances $$\frac{TP}{TP+FP}$$
  - recall: the fraction of relevant instances that were retrieved $$\frac{TP}{TP+FN}$$

Performance evaluation

- F1-score
  - harmonic mean of the precision and recall

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{tp}}{\mathrm{tp} + \frac{1}{2}(\mathrm{fp} + \mathrm{fn})}$$

Performance evaluation

- Precision-recall tradeoff

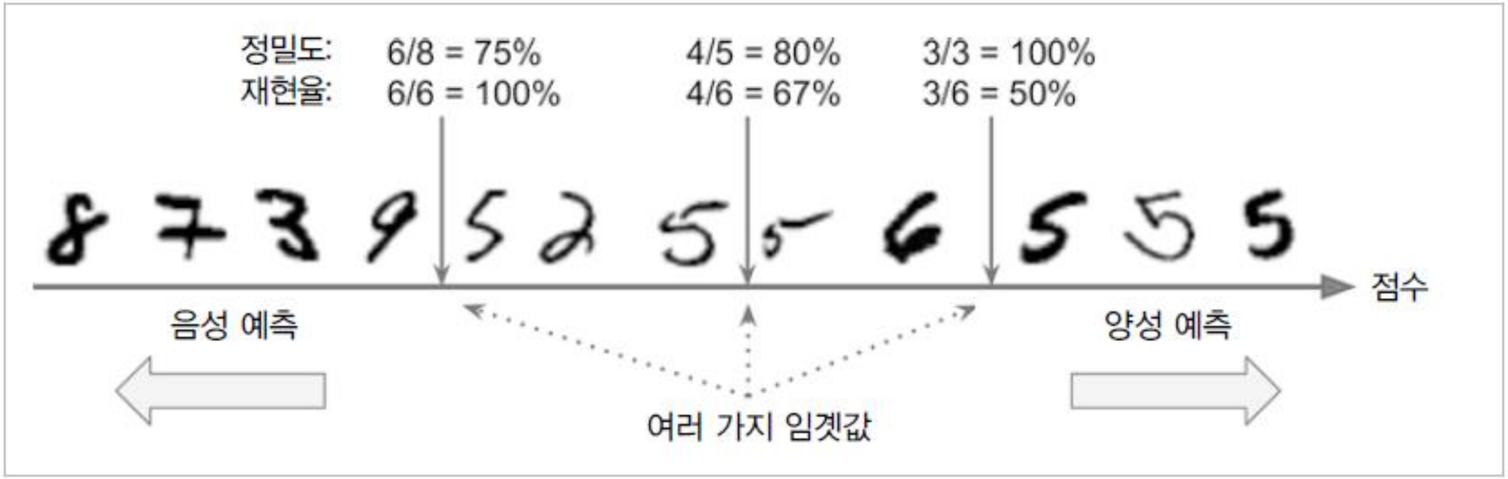
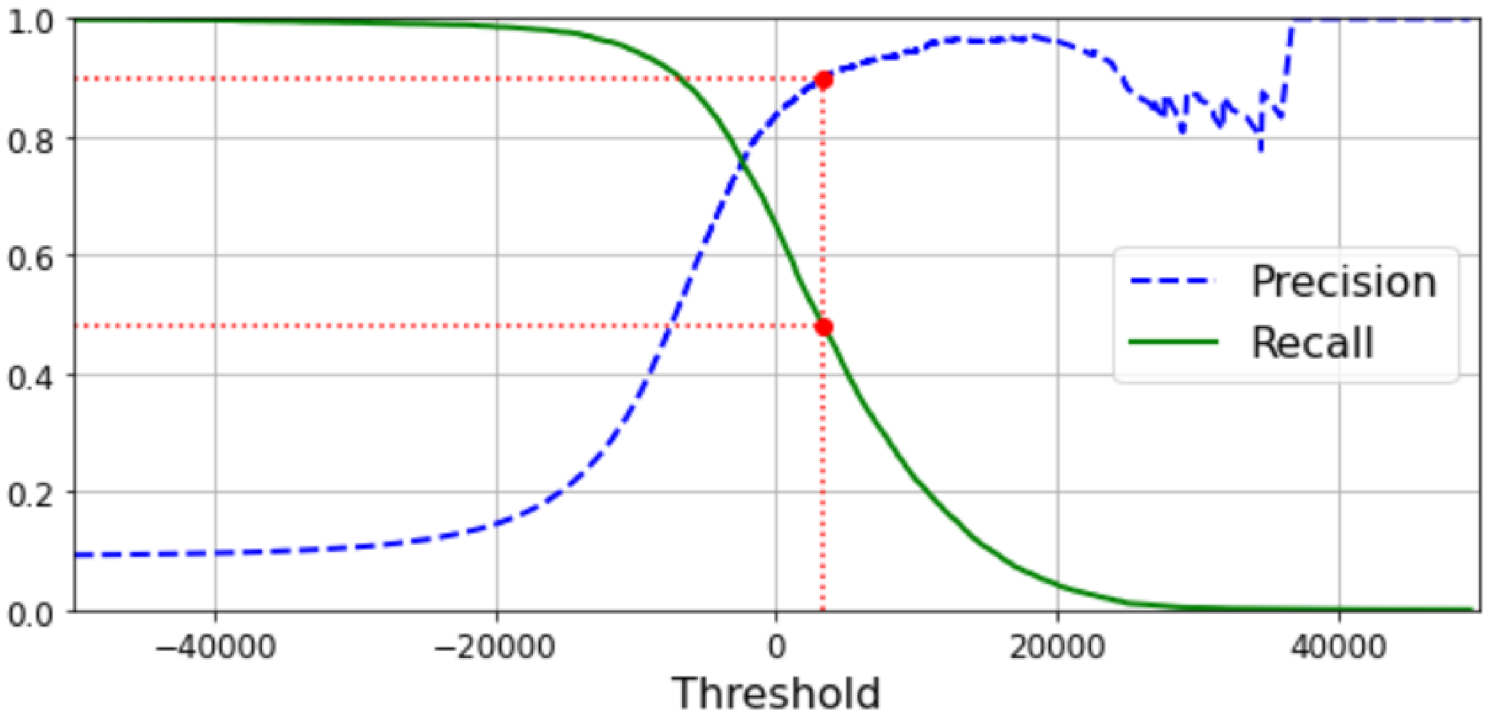


그림 3-3 이 정밀도/재현율 트레이드오프 이미지는 분류기가 만든 점수 순으로 나열되어 있습니다. 선택한 결정 임곗값 위의 것을 양성으로 판단합니다. 임곗값이 높을수록 재현율은 낮아지고 반대로 (보통) 정밀도는 높아집니다.

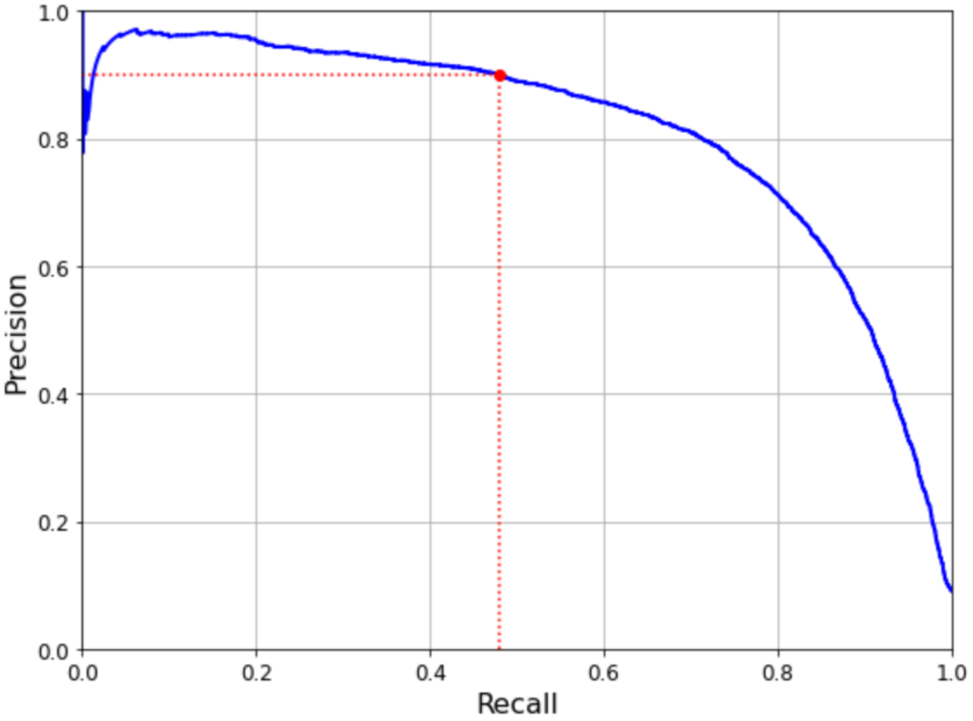
Performance evaluation

- Decision of precision-recall tradeoff



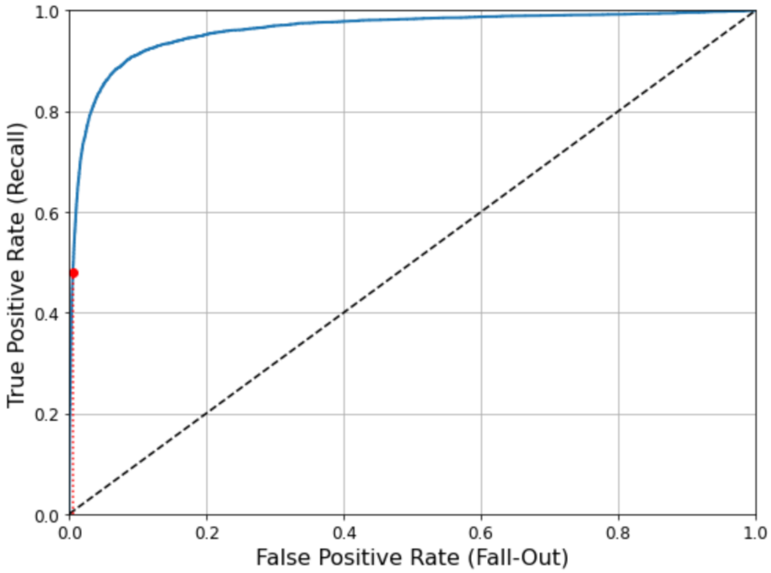
Performance evaluation

- Decision of precision-recall tradeoff



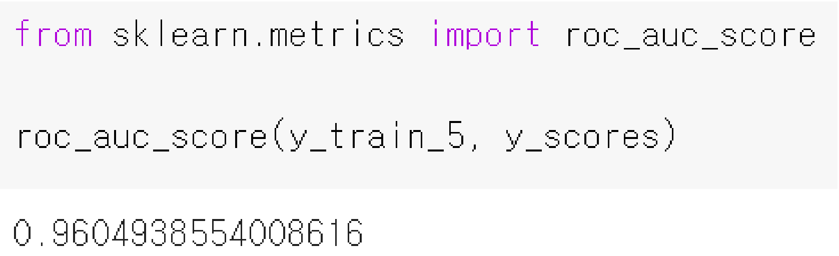
Performance evaluation

- ROC curve
  - receiver operating characteristic (ROC)
    - a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied



Performance evaluation

- ROC curve
  - area under the curve (AUC)
    - wider AUC → more accurate

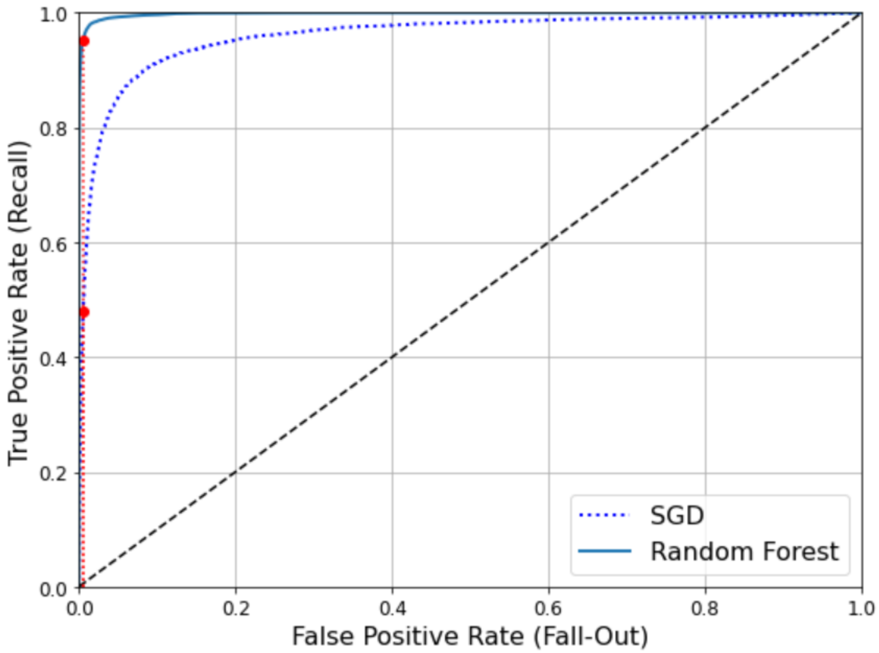


```python
from sklearn.metrics import roc_auc_score

roc_auc_score(y_train_5, y_scores)
```

```
0.9604938554008616
```

Performance evaluation

- ROC curve
  - AUC difference

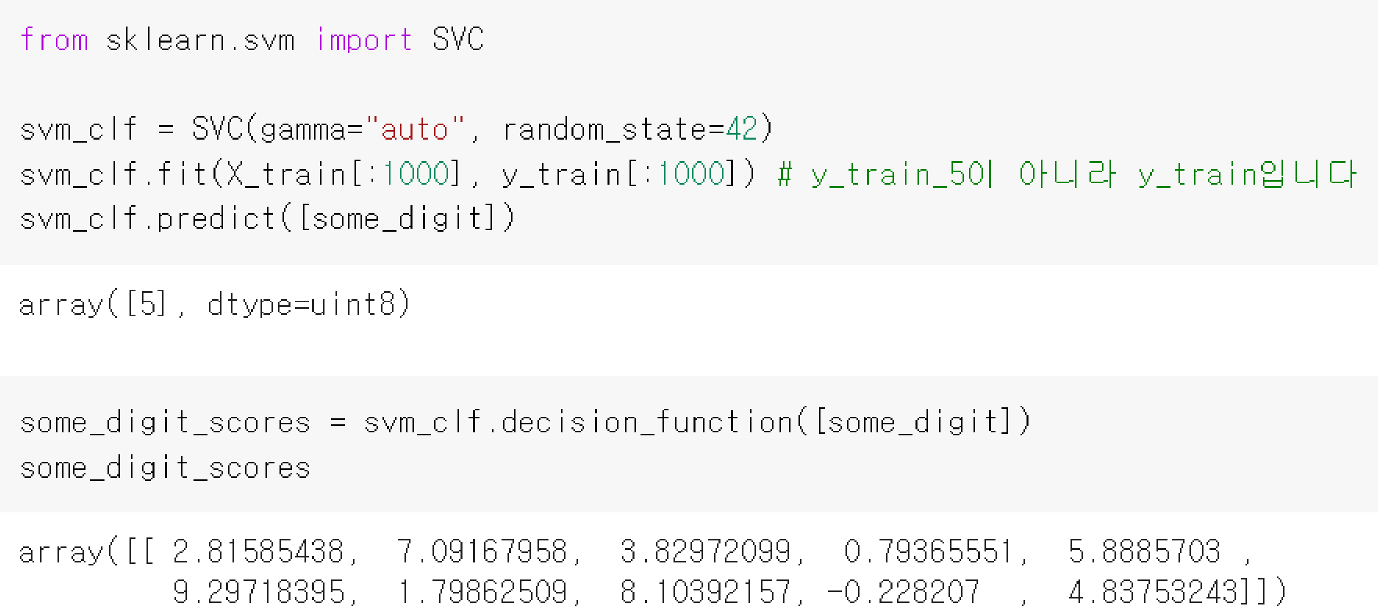


Multiclass classifier

- Classify more than 2 classes
  - SGD, Random forest, naïve Bayes classifier can do multiclass classification
  - logistic regression and support vector machine only do binary classification
    - one-versus-the-rest (OvR, OvA)
    - one-versus-one (OvO)

Multiclass classifier

- SVC example



```
from sklearn.svm import SVC

svm_clf = SVC(gamma="auto", random_state=42)
svm_clf.fit(X_train[:1000], y_train[:1000]) # y_train_5이 아니라 y_train입니다
svm_clf.predict([some_digit])
```

```
array([5], dtype=uint8)
```

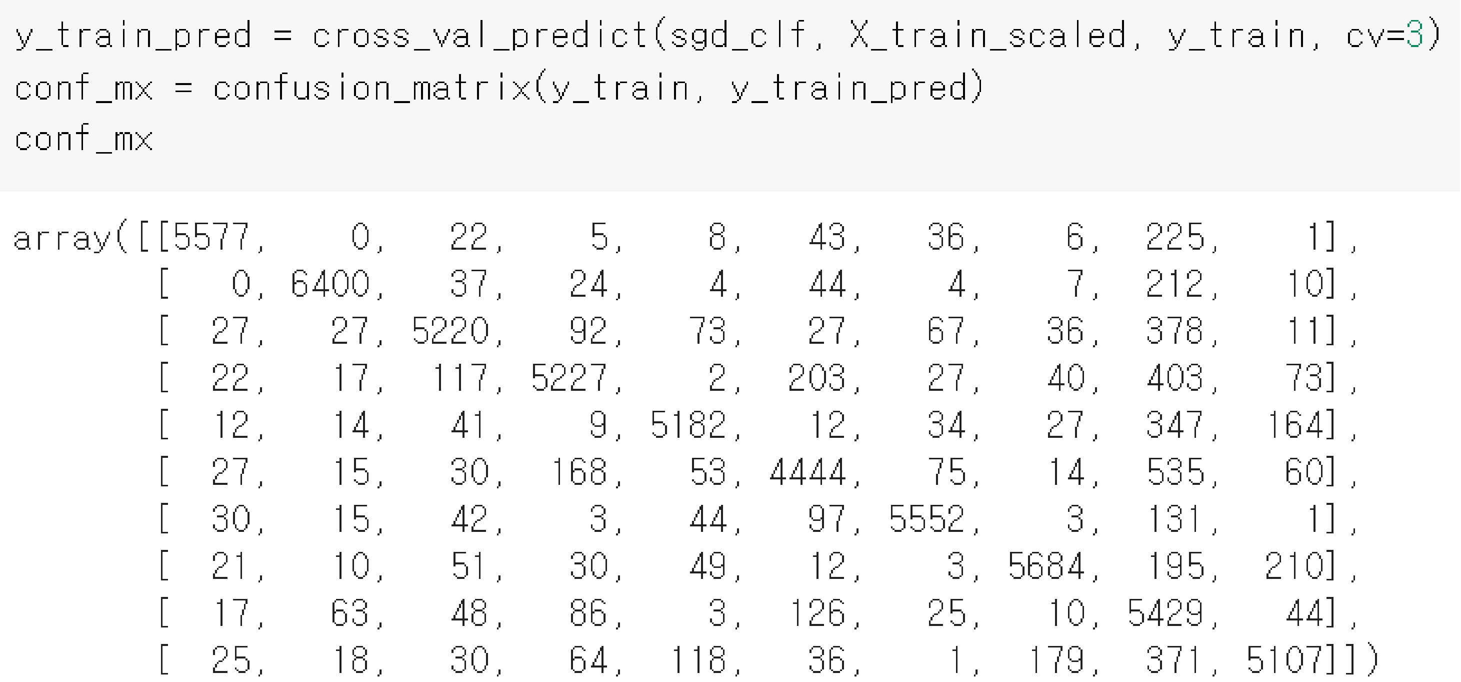
```
some_digit_scores = svm_clf.decision_function([some_digit])
some_digit_scores
```

```
array([[ 2.81585438,  7.09167958,  3.82972099,  0.79365551,  5.8885703 ,
         9.29718395,  1.79862509,  8.10392157, -0.228207  ,  4.83753243]])
```

경상국립대학교
GYEONGSANG NATIONAL UNIVERSITY

Error analysis

- Confusion matrix in multiclass classifier

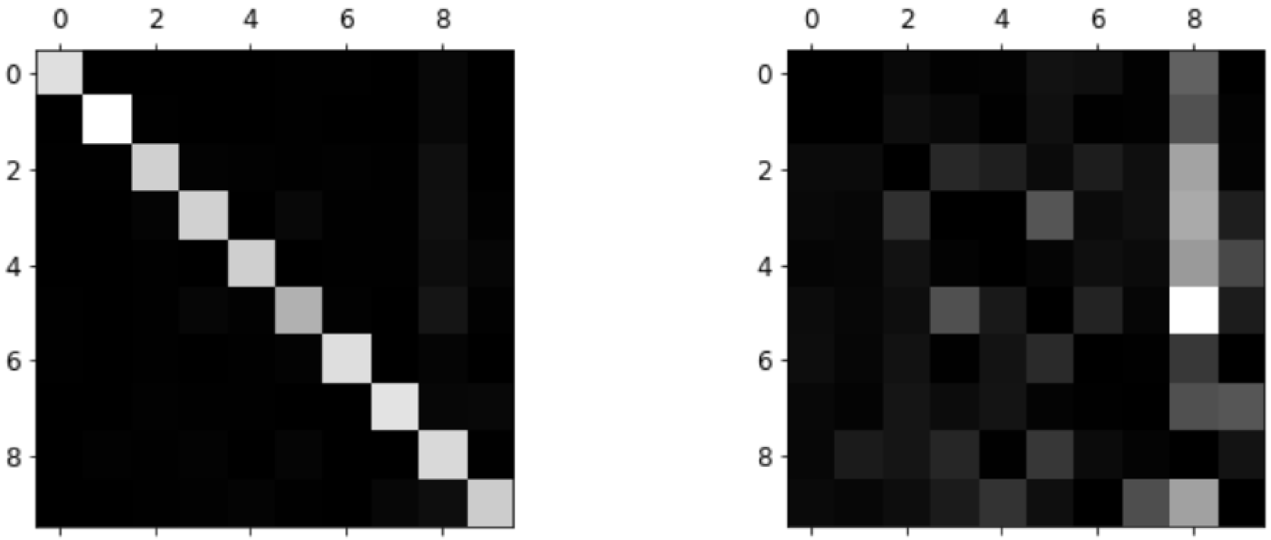


```
y_train_pred = cross_val_predict(sgd_clf, X_train_scaled, y_train, cv=3)
conf_mx = confusion_matrix(y_train, y_train_pred)
conf_mx
```

```
array([[5577,    0,   22,    5,    8,   43,   36,    6,  225,    1],
       [   0, 6400,   37,   24,    4,   44,    4,    7,  212,   10],
       [  27,   27, 5220,   92,   73,   27,   67,   36,  378,   11],
       [  22,   17,  117, 5227,    2,  203,   27,   40,  403,   73],
       [  12,   14,   41,    9, 5182,   12,   34,   27,  347,  164],
       [  27,   15,   30,  168,   53, 4444,   75,   14,  535,   60],
       [  30,   15,   42,    3,   44,   97, 5552,    3,  131,    1],
       [  21,   10,   51,   30,   49,   12,    3, 5684,  195,  210],
       [  17,   63,   48,   86,    3,  126,   25,   10, 5429,   44],
       [  25,   18,   30,   64,  118,   36,    1,  179,  371, 5107]])
```

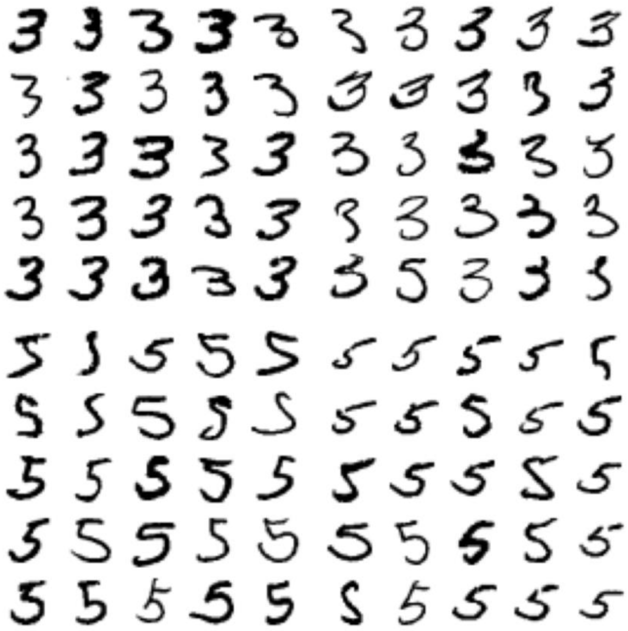
Error analysis

- Confusion matrix in multiclass classifier



Error analysis

- Check the confusion
  - left: classifier determines that it's 3.
  - right: classifier determines that it’s 5.

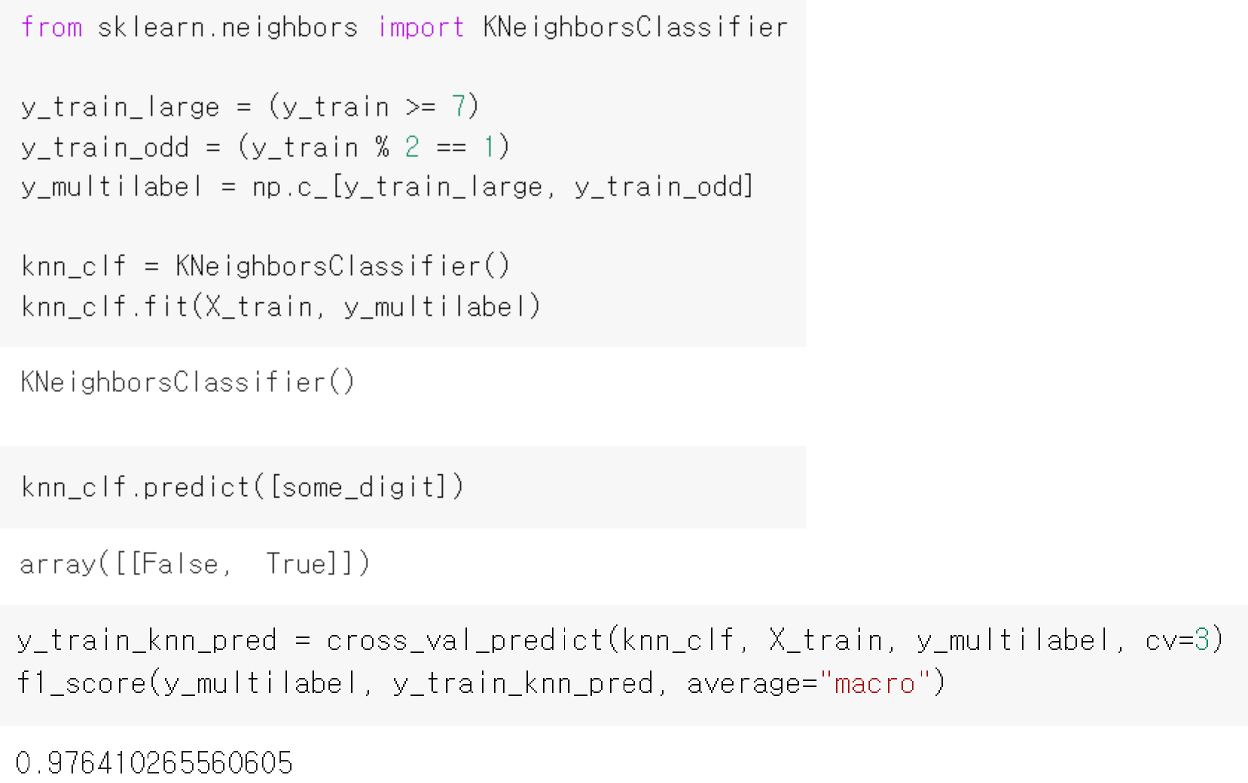


Multilabel classification

- Multi-label cases
  - for example,
    - if there are multiple people in one picture, each recognized person should be tagged
    - suppose the classifier is trained to recognize three faces: Alice, Bob, and Charlie
    - if the classifier sees a picture of Alice and Charlie, it will have to print [1, 0, 1] (i.e., 'with Alice, without Bob, with Charlie')

Multilabel classification

- How to evaluate?



```
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]])
```

```
y_train_knn_pred = cross_val_predict(knn_clf, X_train, y_multilabel, cv=3)
f1_score(y_multilabel, y_train_knn_pred, average="macro")
```

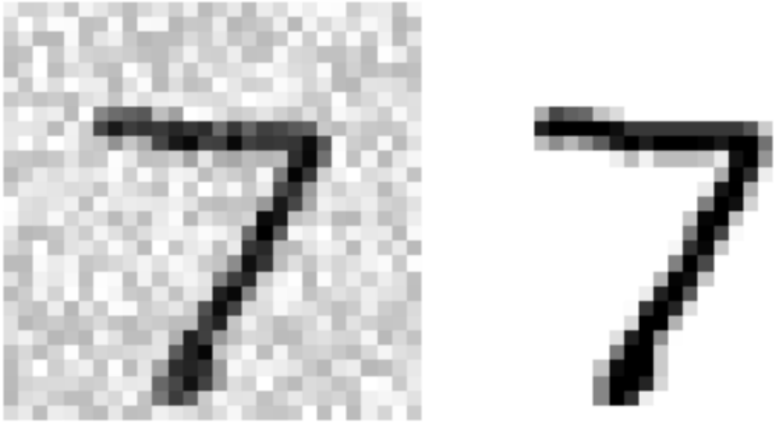
```
0.976410265560605
```

Multioutput classification

- Definition
  - one label may be generalized to be a multi-class, that is, two or more values may be allocated

Multioutput classification

- Example
  - noise reduction in MNIST
    - noisy image as an input, clean image as an output
    - output of the classifier is multiple labels (one label per pixel) and each label has various values (from 0 to 255)



Feel free to question

Through e-mail & LMS

본 자료의 연습문제는 수업의 본교재인
한빛미디어, Hands on Machine Learning(2판)에서 주로 발췌함