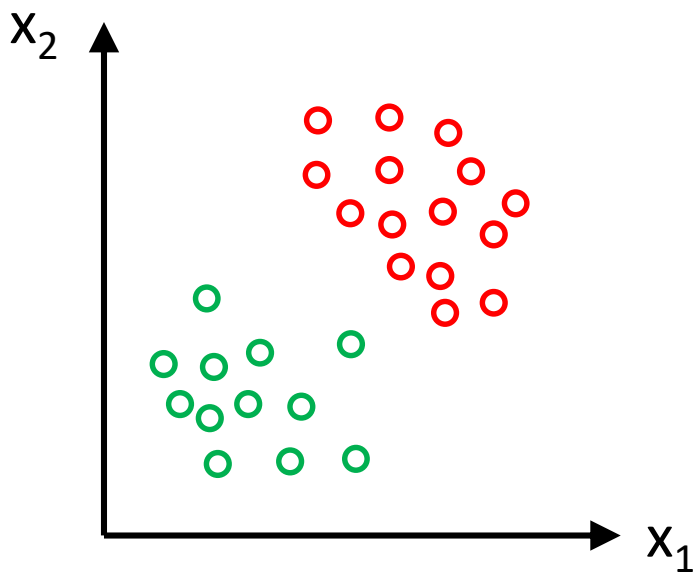
k-NN Classification

Hanwool Jeong

hwjeong@kw.ac.kr

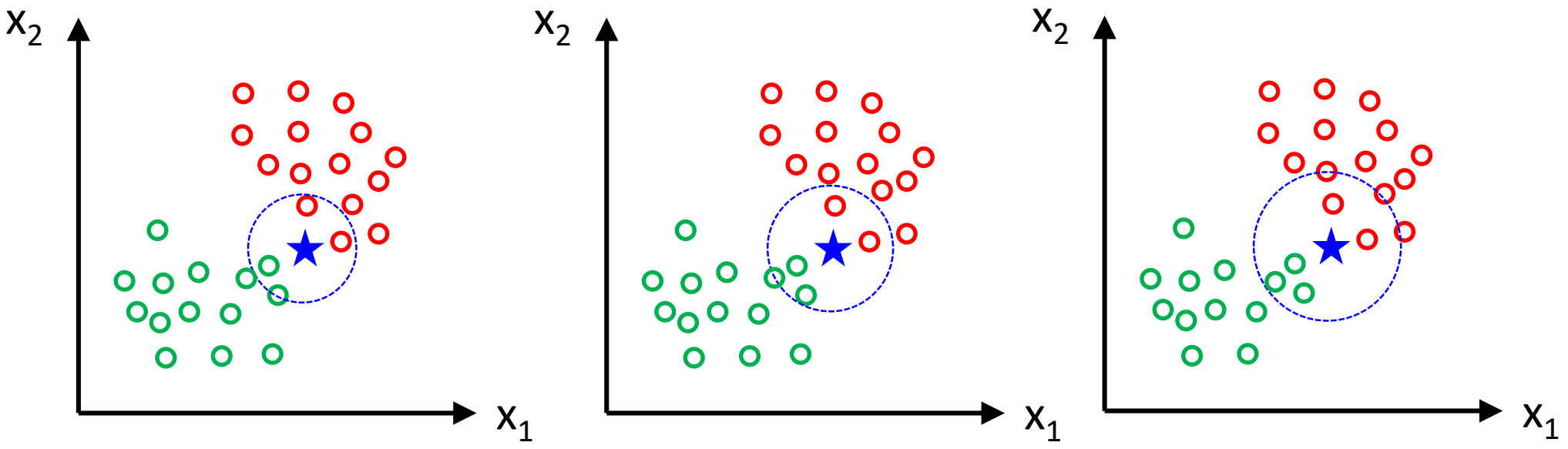
Classification based on Nearest Neighbors

- The simplest method to determine the class of new data is based on the nearest neighbor.
- Is there any problem?



Check k Nearest Neighbors (k-NN)

- k-NN is to perform classification based on comparing how k nearest neighbors are composed of.
- The result can be different according to k values
- Odd k is preferred. Why?



Steps for k-NN Algorithm

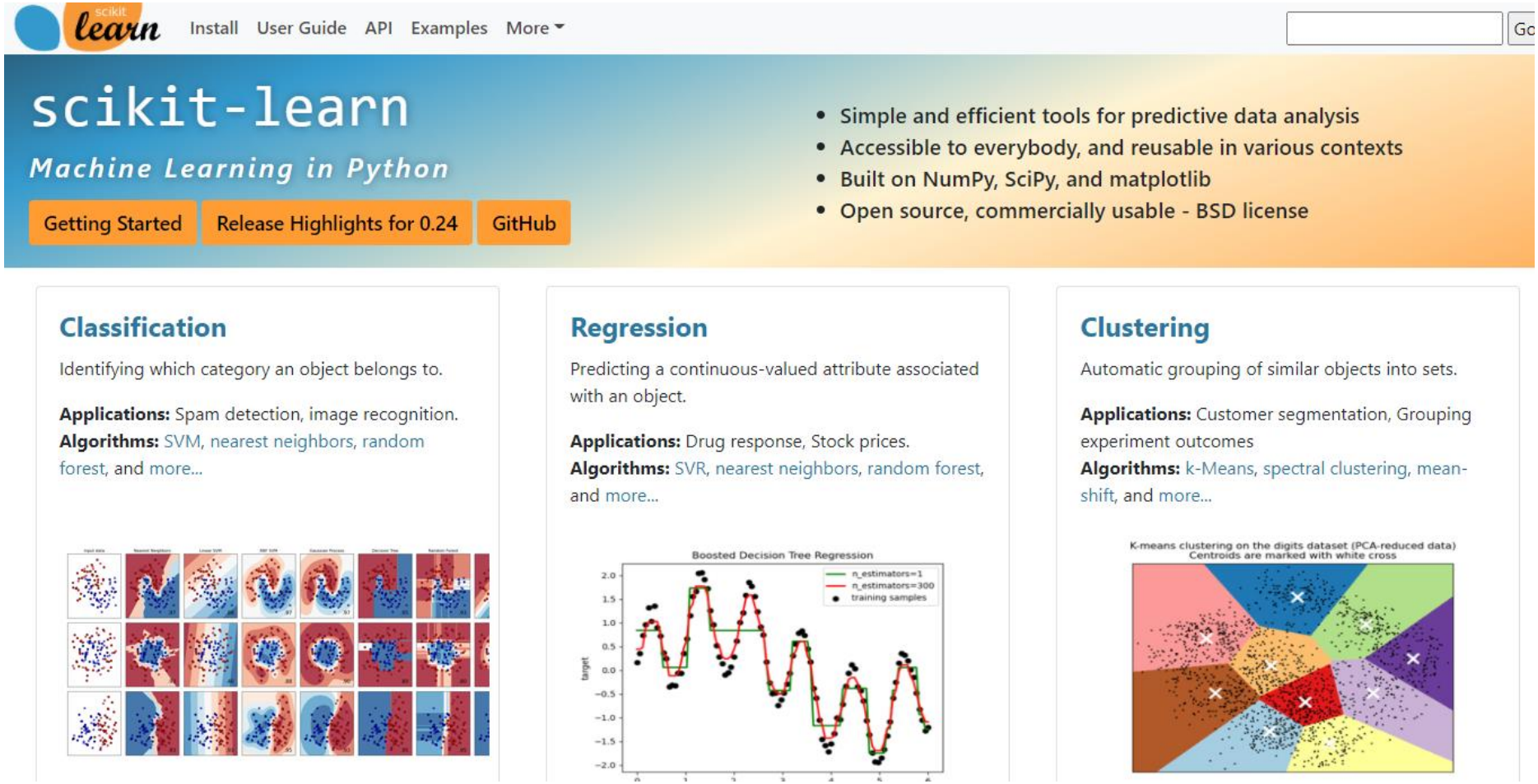
1) Calculating distance to training set data from the input
2) Examining the top “k”s nearest neighbors’ distance
3) Decide the class according to majority of the class.

Can you Refine k-NN?

- We can give more importance to the closer one!

Scikit-learn Library

- From now on, we will exploit scikit-learn library
- https://scikit-learn.org



Scikit-learn Provides Various Sample Datasets

	Explanation
load_boston([return_X_y])	Load and return the boston house-prices dataset (regression).
load_iris([return_X_y])	Load and return the iris dataset (classification).
load_diabetes([return_X_y])	Load and return the diabetes dataset (regression).
load_digits([n_class, return_X_y])	Load and return the digits dataset (classification).
load_linnerud([return_X_y])	Load and return the linnerud dataset (multivariate regression).
load_wine([return_X_y])	Load and return the wine dataset (classification).
load_breast_cancer([return_X_ y])	Load and return the breast cancer wisconsin datas et (classification).

Try This!

```
from sklearn.datasets import load_iris
iris = load_iris()

from sklearn.datasets import load_breast_cancer
bCancer = load_breast_cancer()

```

Split Train Data vs. Test Data

- After loading datasets, the data for training/test should be split as:

```
from sklearn.model_selection import train_test_split
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4, random_state = 42)
```

Perform k-NN!

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)
scores = metrics.accuracy_score(y_test, y_pred)
```