

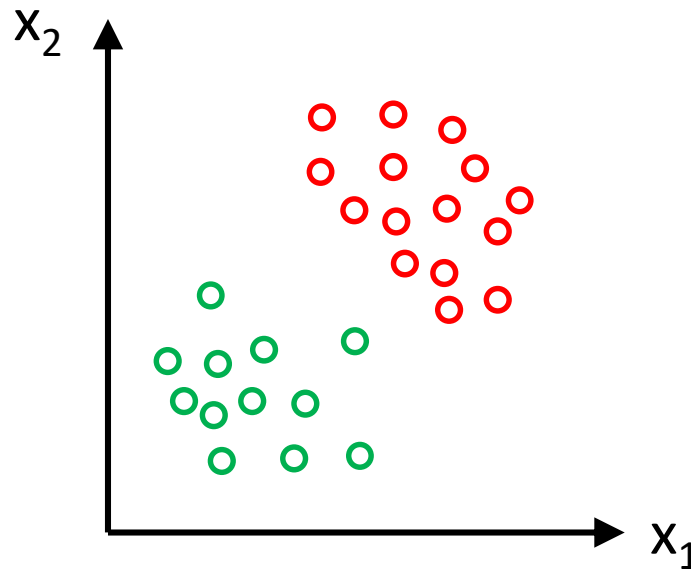
k-NN Classification

Hanwool Jeong

hwjeong@kw.ac.kr

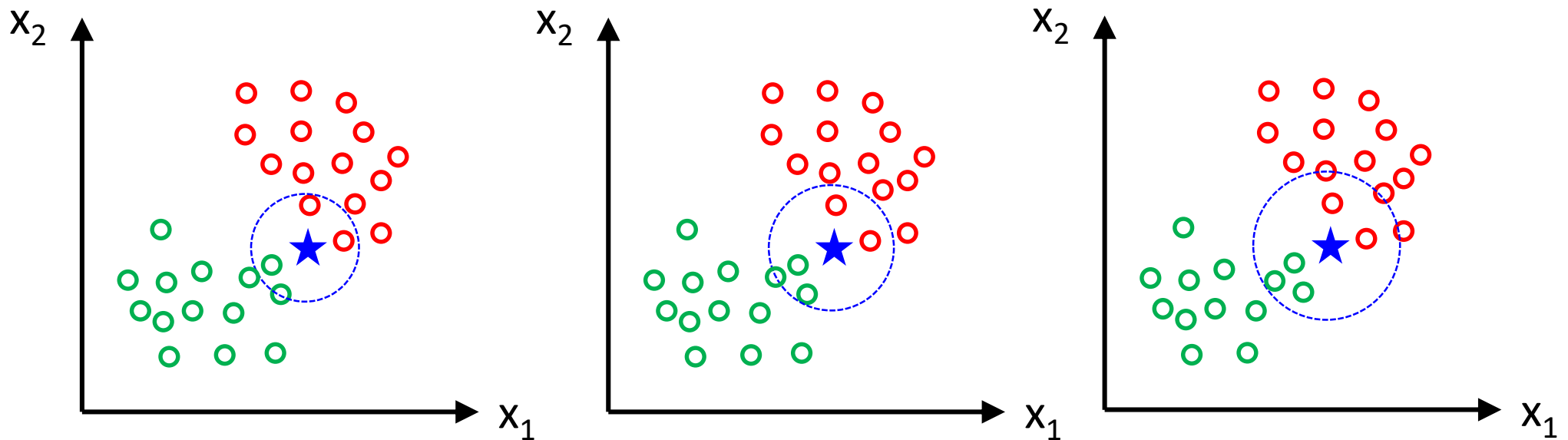
Classification based on Nearest Neighbors

- 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

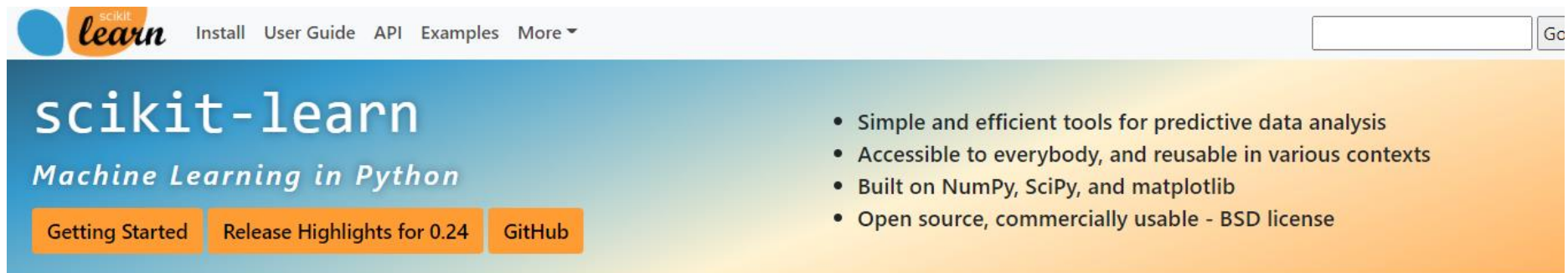
- 1) Calculating distance to training set data from the input
- 2) Examining the top “k”s nearest neighbors’ distance
- 3) Decide the class according to majority of the class.

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>



The screenshot shows the scikit-learn website homepage. At the top left is the scikit-learn logo. To its right are navigation links: "Install", "User Guide", "API", "Examples", and "More". A search bar is on the right. The main header features the text "scikit-learn" in a large font, with "Machine Learning in Python" below it. On the left, there are three orange buttons: "Getting Started", "Release Highlights for 0.24", and "GitHub". On the right, there is a list of four bullet points describing the library's features.

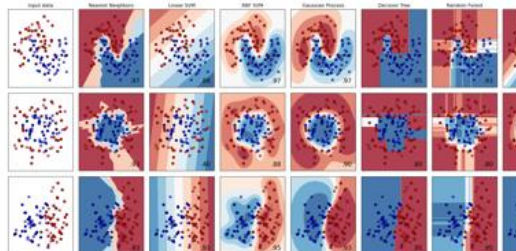
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...

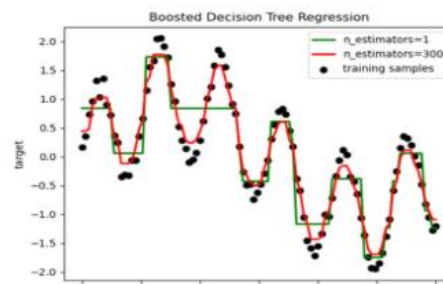


Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...

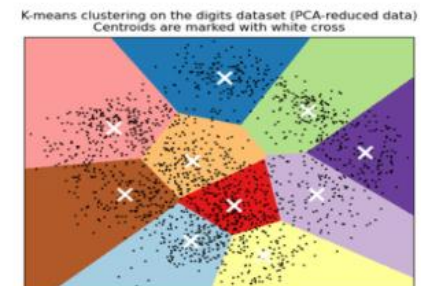


Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



Scikit-learn Provides Various Sample Datasets

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

Try This!

```
1  from sklearn.datasets import load_iris
2  iris = load_iris()
3
4  from sklearn.datasets import load_breast_cancer
5  bCancer = load_breast_cancer()
6
```


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)
```