K-Nearest Neighbor (KNN) Classifier and Regressor

Liang Liang

Categories of Machine Learning

- **Unsupervised Learning**

Clustering: k-means, GMM

Dimensionality reduction (representation learning): PCA, isomap, etc to learn a meaningful representation in a lower dimensional space

Probability Density Estimation: GMM, KDE

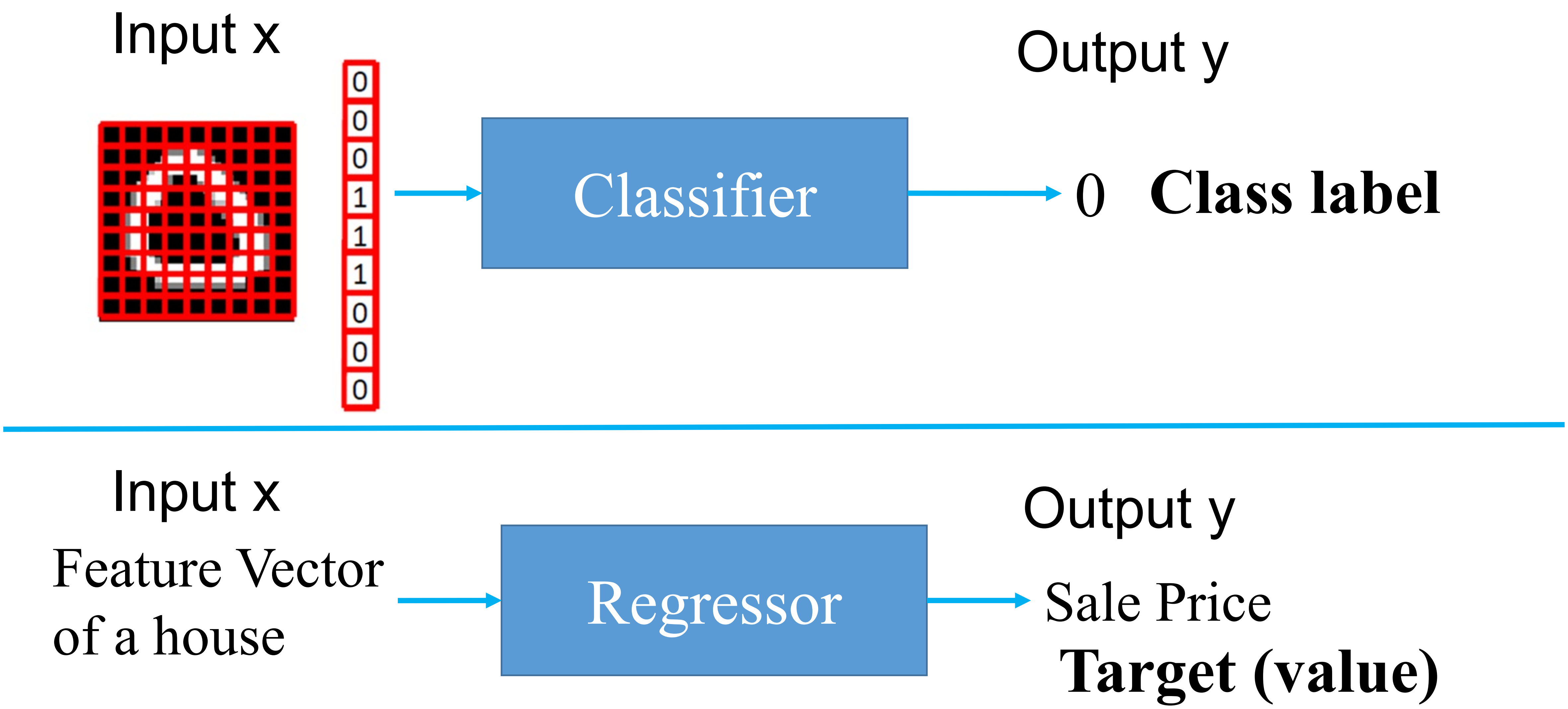
- **Supervised Learning**

to model the relationship between measured features of data and some labels associated with the data

- Reinforcement Learning

the goal is to develop a model (agent) that improves its performance based on interactions with the environment

Supervised Learning: classification and regression



Supervised Learning: classification and regression

Input x

Machine Learning Model

Output y

Dataset:

input-output pairs, $(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_N, y_N)$

Binary Classification

- Data points are from two classes. A data point only belongs to one class.

Classifier

label of the data point x

$y = 0$ male

$y = 1$ female

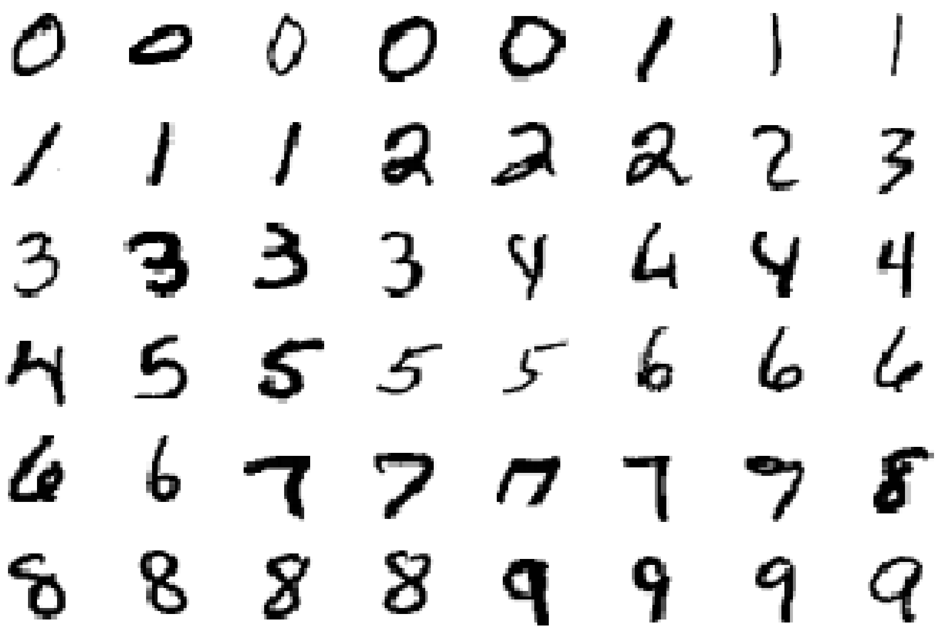
or

$y = -1$ male

$y = 1$ female

Multiclass Classification

- Data points are from many classes.
- A data point only belongs to one class.



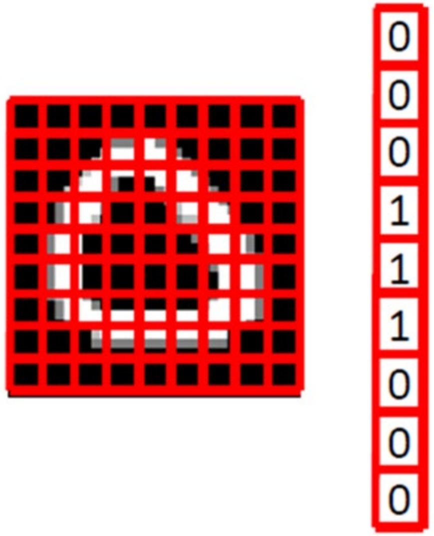
Classifier

label of the data point x
label = ?
10 possible labels:
(0, 1, 2, 3, 4, 5 ,6, 7, 8, 9)

Multiclass Classification

$$y = g(x)$$

$$x = \begin{pmatrix} x_{(1)} \\ x_{(2)} \\ \dots \\ x_{(M)} \end{pmatrix}$$



y is the *class label* of the data point x

$y = 0$

Multiclass Classification: one-hot-encoding

- Data points are from many classes. A data point only belongs to one class.

one-hot encoding: the output from a classifier is a vector, length= # of classes

$$y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \end{bmatrix} \qquad \begin{array}{c} \text{label} = 0 \\ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array} \qquad \begin{array}{c} \text{label} = 1 \\ \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array} \qquad \begin{array}{c} \text{label} = 2 \\ \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array} \qquad \begin{array}{c} \text{label} = 9 \\ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \end{array}$$

Output of a classifier could be real numbers

Example:

Classifier

soft label $\rightarrow$ convert to binary $\rightarrow$ hard label

$$\begin{bmatrix} 0.8 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.1 \\ 0.1 \end{bmatrix} \xrightarrow{\text{convert to binary}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \text{class label} = 0$$

10 possible labels:
(0, 1, 2, 3, 4, 5 ,6, 7, 8, 9)

"0.8" is usually interpreted as the "confidence" of the classifier about $\{y_{out} = 1\}$

Multi-Label Classification

- The data points are from many classes.
- A data point may belong to more than one class.

$$y_0 = 1\text{, it is a cat}$$
$$y_0 = 0\text{, it is not a cat}$$
$$y_1 = 1,\ \text{ it is cute}$$
$$y_1 = 0,\ \text{ it is not cute}$$

Classifier

$$y = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = \begin{array}{l} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{array} \begin{array}{l} \text{cat} \\ \text{cute} \end{array}$$

It is a cute cat

classifiers

- Many types of classifiers:
  KNN classifier (K-Nearest Neighbor)
  Naïve Bayes classifier
  Decision Tree classifier
  Random Forest classifier
  SVM classifier (Support Vector Machine)
  Neural Network classifier

- Each type is associated with specific Learning Algorithms

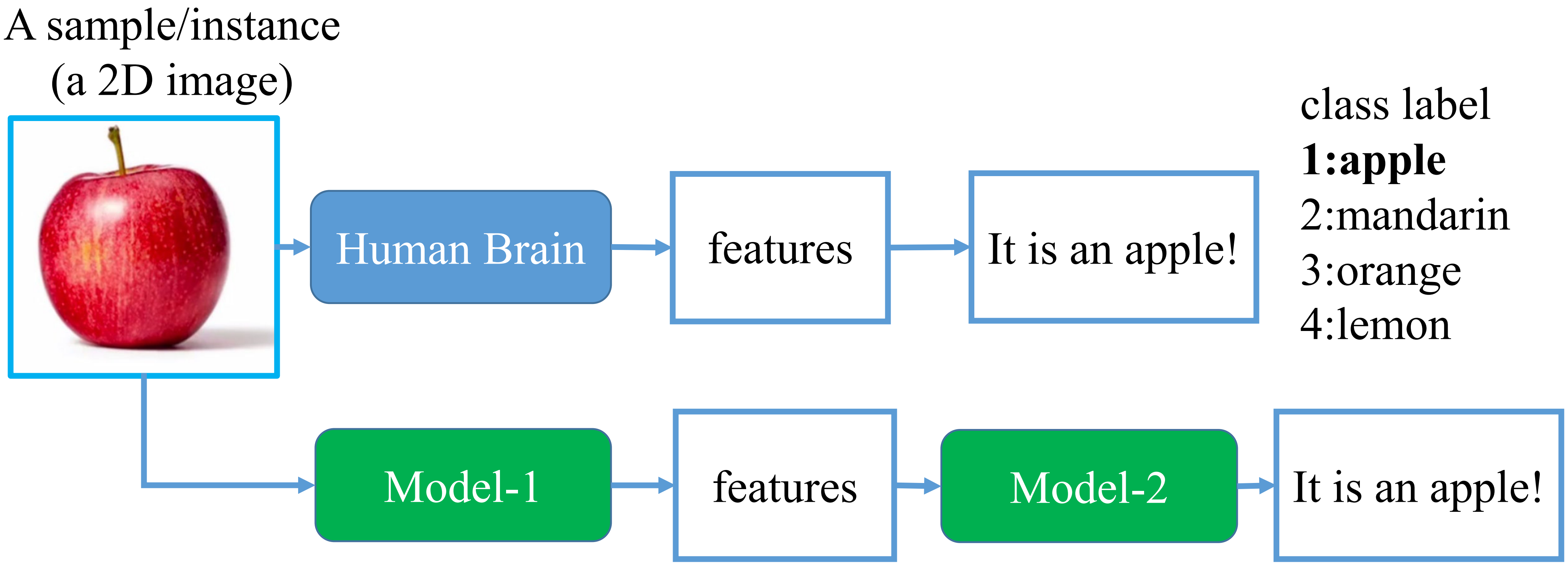
A Classification Task

- A simple task: classify a fruit into four classes/categories {1:apple, 2:mandarin, 3:orange, 4:lemon},
  note: class-3 contains oranges that are not mandarin oranges

A sample/instance

perform classification

class/fruit label
1:apple
2:mandarin
3:orange
4:lemon



It is not easy to develop Model-1 for feature extraction

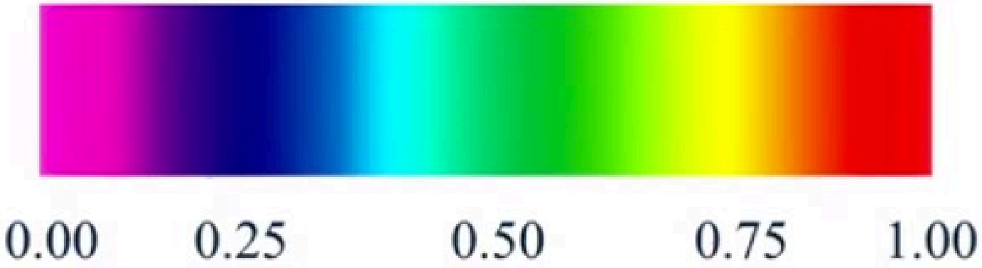
It is relatively easy to develop Model-2 for classification, given the features of the sample.

Now, let's develop Model-2 for classification.

1:apple
2:mandarin
3:orange
4:lemon

The feature vector of a fruit sample: [width, height, color_score]

color_score is a number (0~1) to describe the color



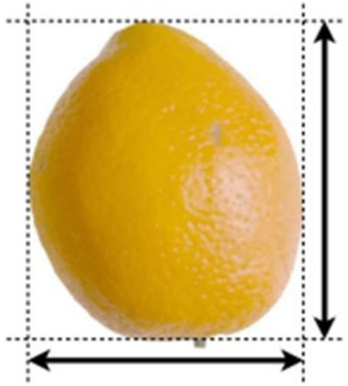
Color category	color_score
Red	0.85 - 1.00
Orange	0.75 - 0.85
Yellow	0.65 - 0.75
Green	0.45 - 0.65

The Fruit Dataset

A bucket of fruits

The fruit dataset was created by Dr. Iain Murray at the University of Edinburgh. He bought a few dozen oranges, lemons and apples, and recorded their features in a table.

4 classes: {1:apple, 2:mandarin, 3:orange, 4:lemon}



The fruit dataset (a table)

Each row contains the information of a fruit sample/instance

fruit label	fruit_name	subtype	mass (g)	width (cm)	height (cm)	color_score
1	apple	granny_smith	192	8.4	7.3	0.55
4	lemon	spanish_belsan	194	7.2	10.3	0.70

In this table: what is input x? what is output y?

http://usapple.org/the-industry/apple-varieties/

In total, there are 59 fruit samples (i.e. 59 rows) in the table

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	mandarin	80	5.8	4.3	0.77
6	2	mandarin	mandarin	80	5.9	4.3	0.81
7	2	mandarin	mandarin	76	5.8	4.0	0.81
8	1	apple	braeburn	178	7.1	7.8	0.92
9	1	apple	braeburn	172	7.4	7.0	0.89
10	1	apple	braeburn	166	6.9	7.3	0.93

4 classes: {1:apple, 2:mandarin, 3:orange, 4:lemon}

Select 3 features: width, height, color_score

```
1  fruits = pd.read_table('fruit_data_with_colors.txt')
```

```
1  fruits
```

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60

```
1  fruits.shape
```

```
(59, 7)
```

```
1  features = fruits.columns[-3:].tolist()
```

```
1  features
```

```
['width', 'height', 'color_score']
```

Split data (59) into a training set (80%, 47) and a testing set (20%, 12)

```
X = fruits[features]
Y = fruits['fruit_label']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2
```

```
X_train.shape
```

```
(47, 3)
```

X_train contains the features of the 47 training samples
Each row of X_train is a feature vector of a training sample.

```
Y_train.shape
```

```
(47,)
```

Y_train contains the class/fruit labels of the 47 training samples
Each element of Y_train is a class label of a training sample.

```
X_test.shape
```

```
(12, 3)
```

X_test contains the features of the 12 testing samples
Each row of X_test is a feature vector of a testing sample.

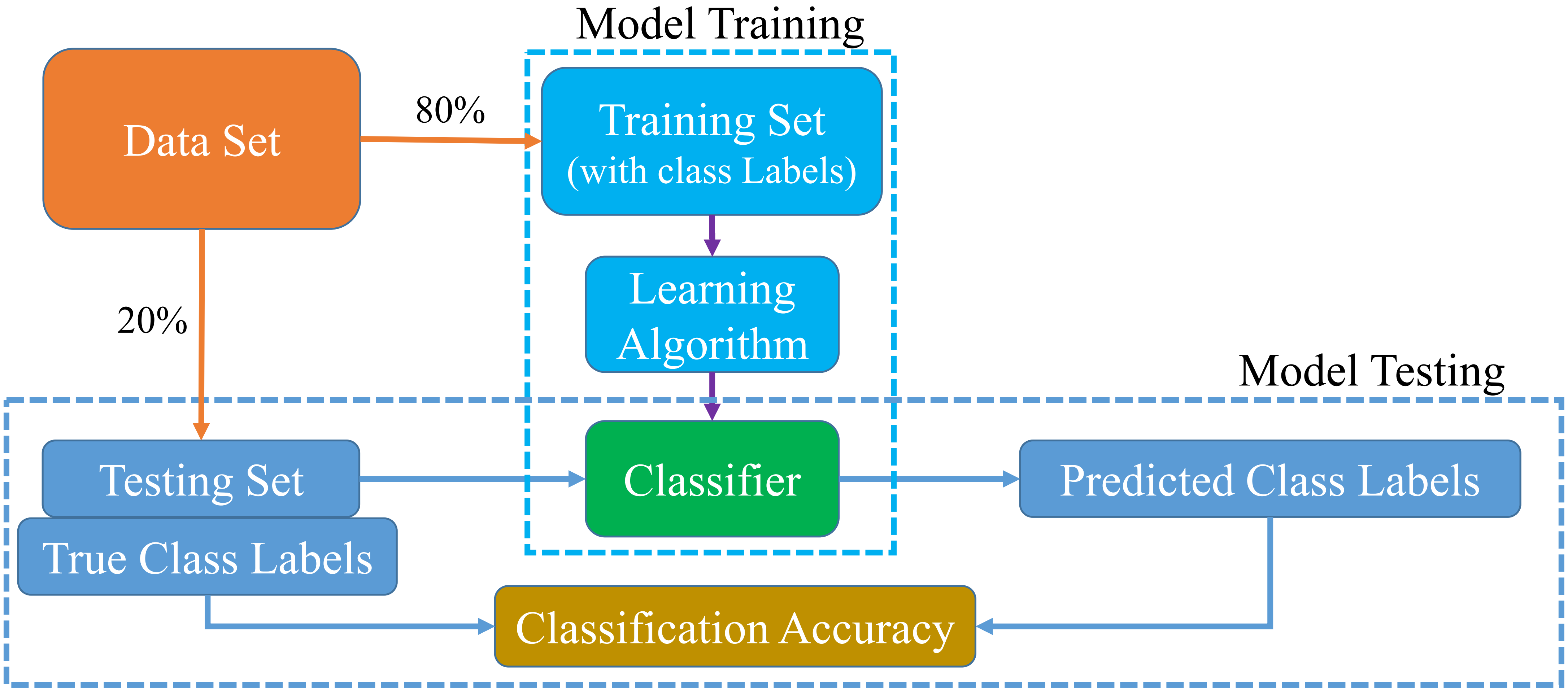
```
Y_test.shape
```

```
(12,)
```

Y_test contains the class/fruit labels of the 12 testing samples
Each element of Y_test is a class label of a testing sample.

The flowchart of a classification study

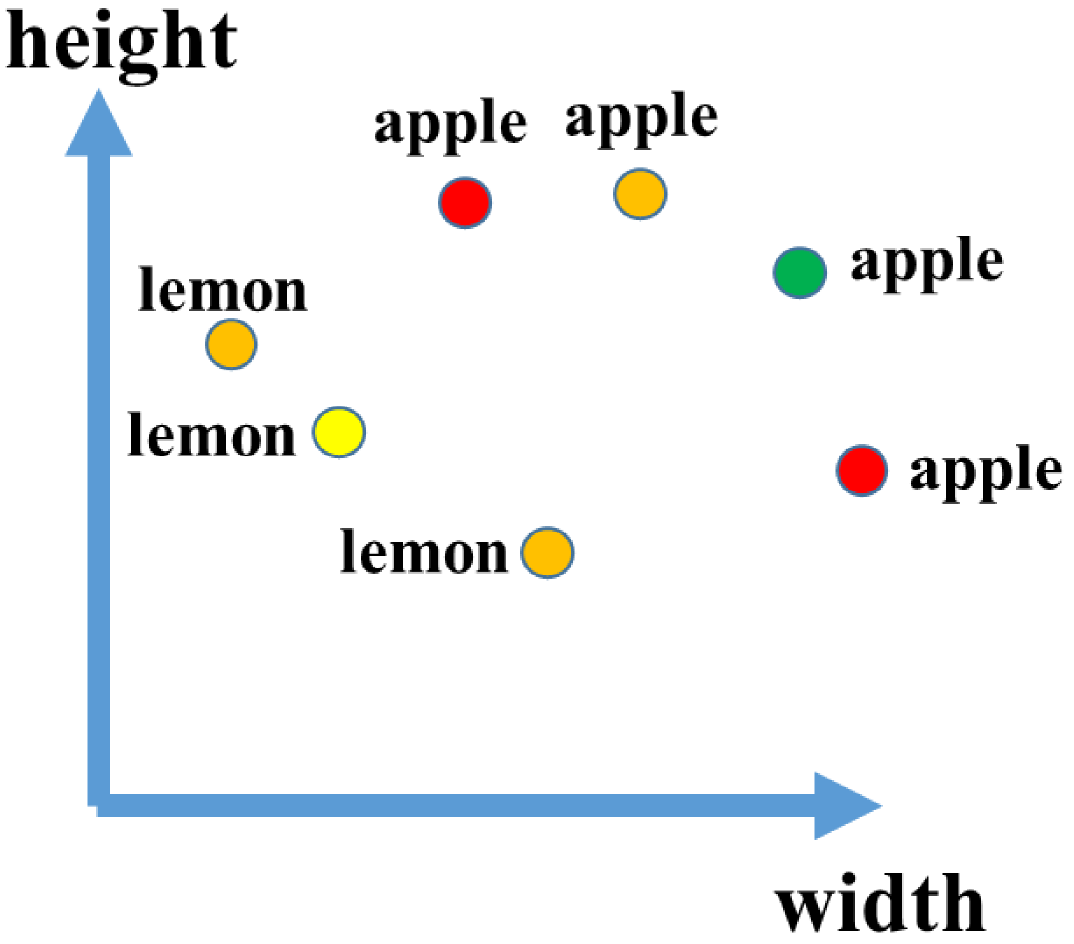
- Classification is a subcategory of supervised learning where the goal is to predict the class labels of new samples.



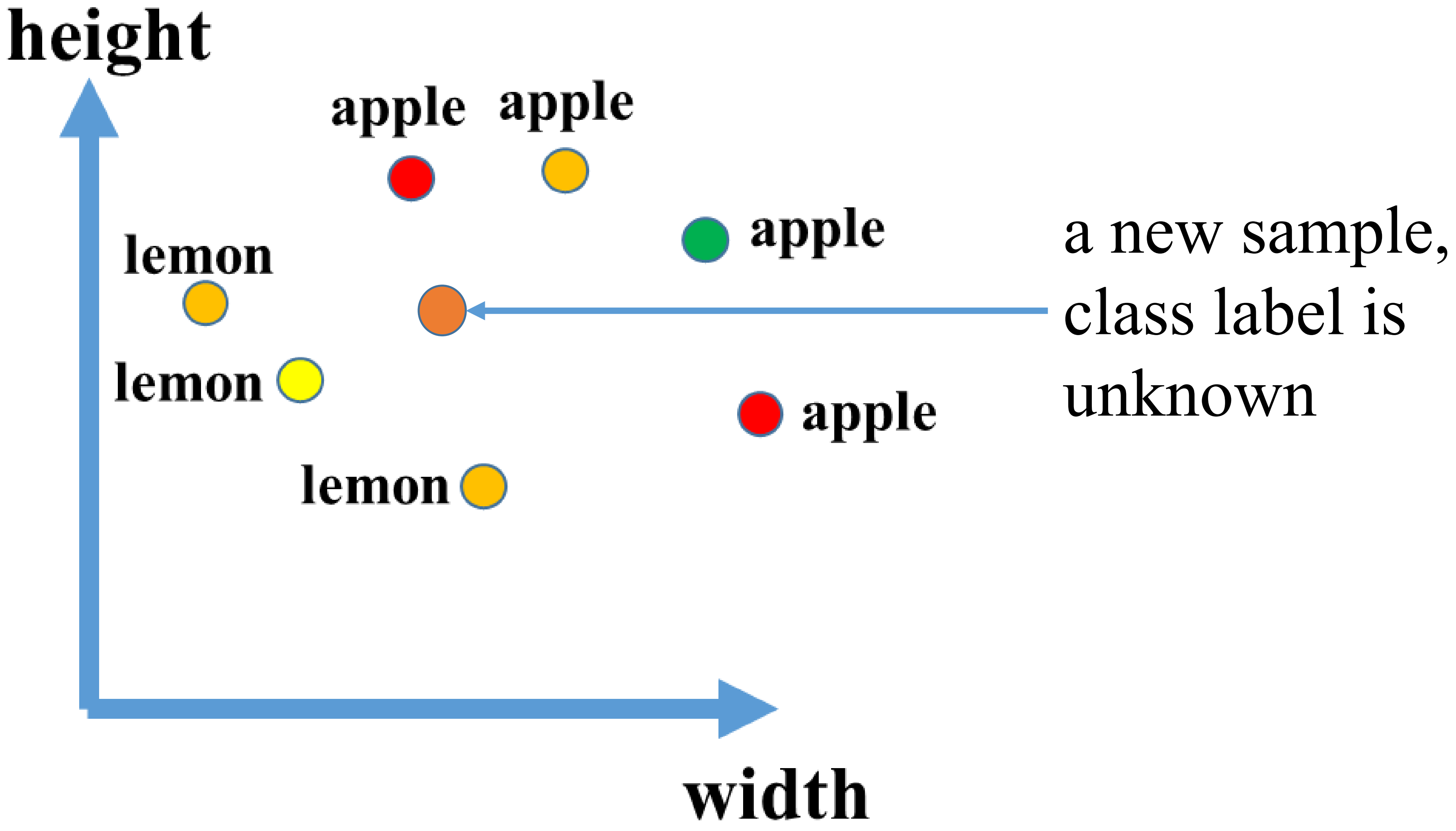
KNN classifier (K-Nearest Neighbor)

- A KNN classifier. The user needs to:

(1) choose the value of K and (2) choose a distance measure



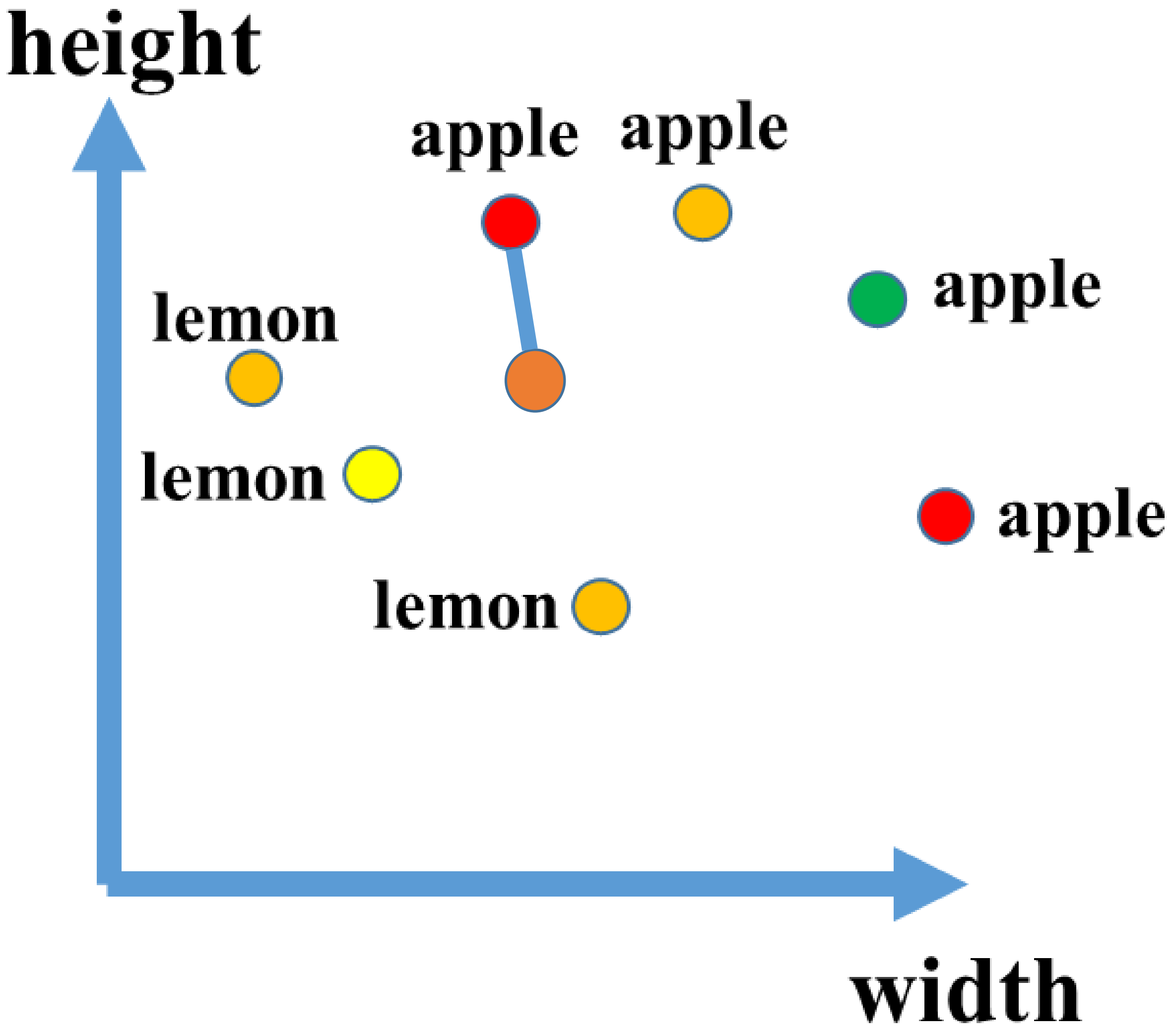
7 samples in the training set



7 samples in training set

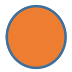
Let's set K=**1** and use L2-based distance measure

Task: Find the nearest neighbor in the training set (by comparing distances)



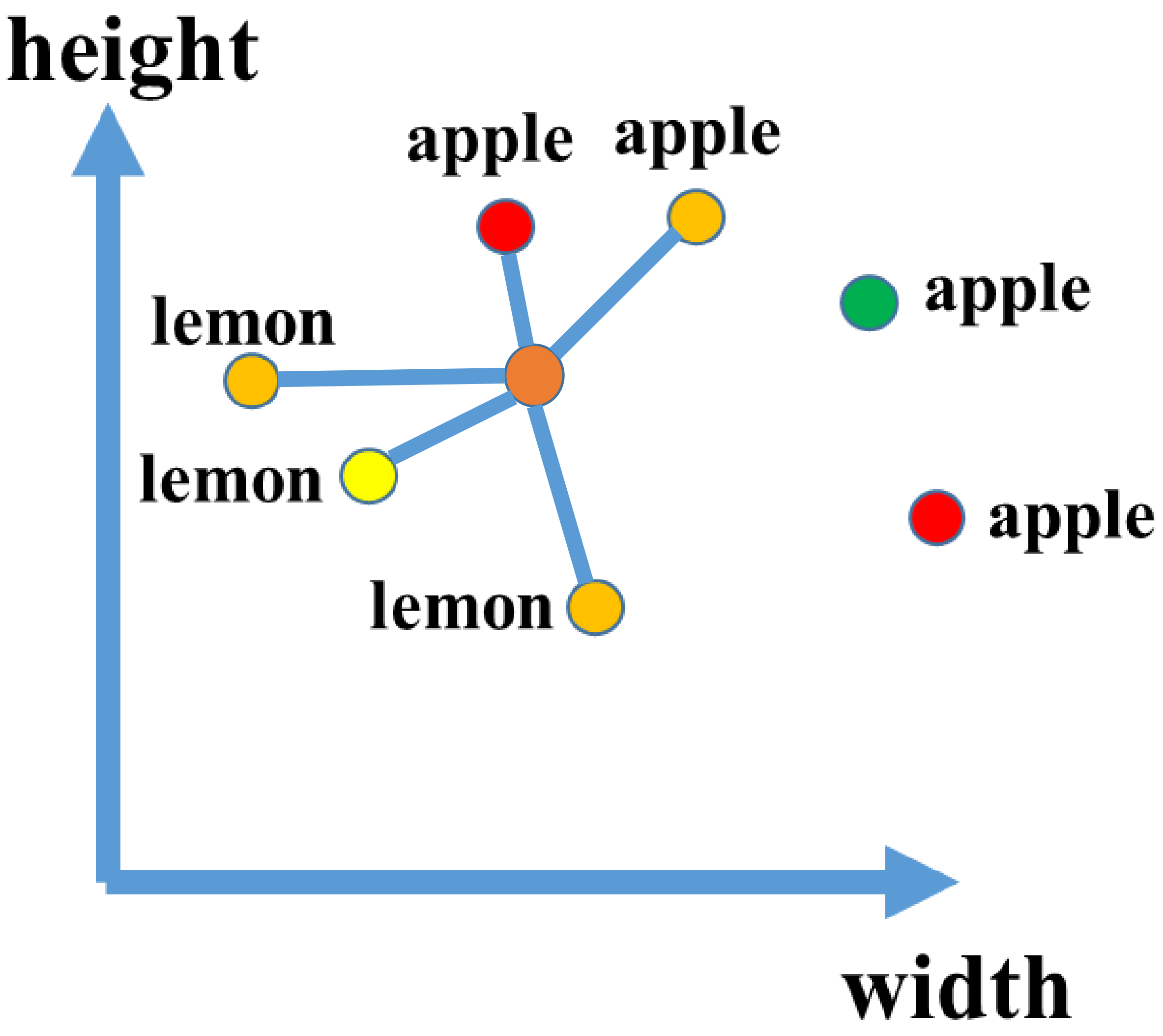
7 samples in training set

the nearest neighbor in the training set is an apple, therefore the KNN classifier will classify the input as an apple

 is classified as an apple because its nearest neighbor is an apple

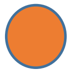
Let's set **K=5** and use L2-based distance measure

Task: Find the **5** nearest neighbor in the training set



7 samples in training set

Among the **5** nearest neighbors in the training set, there are 3 lemons and 2 apples, therefore, based on **majority vote**, the KNN classifier will classify the input as a lemon

 is classified as a lemon because the majority of its *K* nearest neighbors are lemons

Let's build and train a KNN classifier using sk-learn

Build a KNN classifier, name it knn

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

```
# instance of the classifier
knn = KNeighborsClassifier(n_neighbors = 5)
```

K=5

Train the KNN classifier (fit the model to the data)

```
knn.fit(X_train, Y_train)
```

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

L2-based distance

Model training is to let **knn** ***memorize*** all of the training samples (features and labels), and build a tree for K-nearest neighbor search.

Use the trained KNN classifier to classify a sample in testing set

Select a sample in the testing set

```
sample_test = X_test.iloc[0,:]
sample_test
```

```
width          9.60
height         9.20
color_score    0.74
Name: 26, dtype: float64
```

We know the true label of this sample

```
label_true = Y_test.iloc[0]
print('The true label is', label_true, ':', fruit_lable_to_name[label_true])
```

```
The true label is 3 : orange
```

Use knn to Predict the label of this sample

```
label_predicted = knn.predict([sample_test])
print('The label predicted by knn is', label_predicted[0], ':', fruit_lable_to_name[label_predicte
if label_predicted[0] == label_true:
    print('Classification is accurate for this testing sample')
else:a
    print('Classification is wrong for this testing sample')
```

```
The label predicted by knn is 3 : orange
Classification is accurate for this testing sample
```

Use the trained KNN classifier to classify a new, previously unseen sample that is not in the training set nor in the testing set

the Feature Vector of a new sample

```
sample_new = [6.0, 4.0, 0.8]
label_predicted = knn.predict([sample_new])
print('The label predicted by knn is', label_predicted[0], ':', fruit_lable_to_name[label_predicte
```

```
The label predicted by knn is 2 : mandarin
```

Evaluate the Performance of the KNN Classifier (K=5)

- $\text{Classification Accuracy} = \dfrac{\text{the number of correctly classified samples}}{\text{total number of samples}}$

- Training Accuracy: accuracy on training set (80% of the data)

```
knn.score(X_train, Y_train)
```

```
0.8723404255319149
```

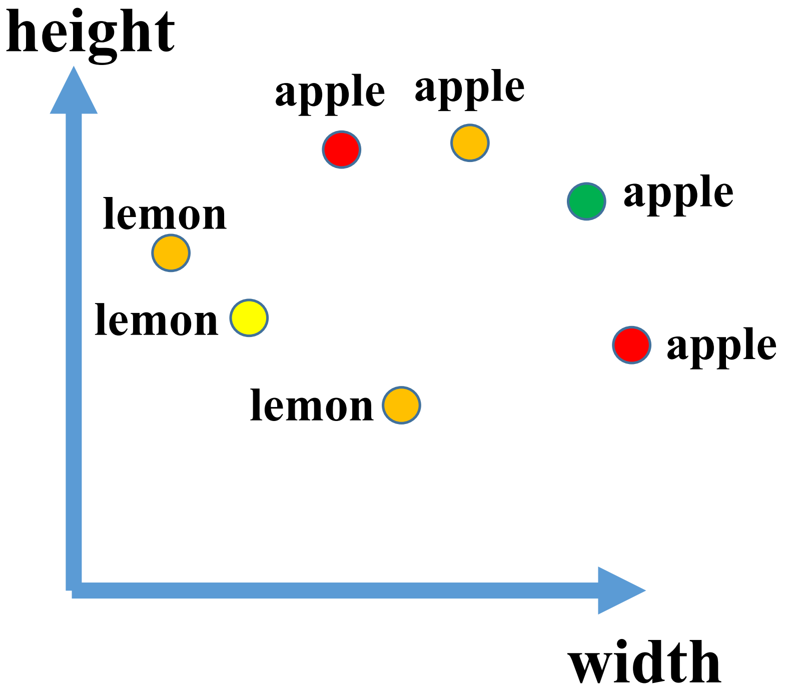
Testing Accuracy: accuracy on testing set (20% of the data)

```
knn.score(X_test, y_test)
```

```
0.75
```

Training Accuracy of KNN classifier is 100% when K=1

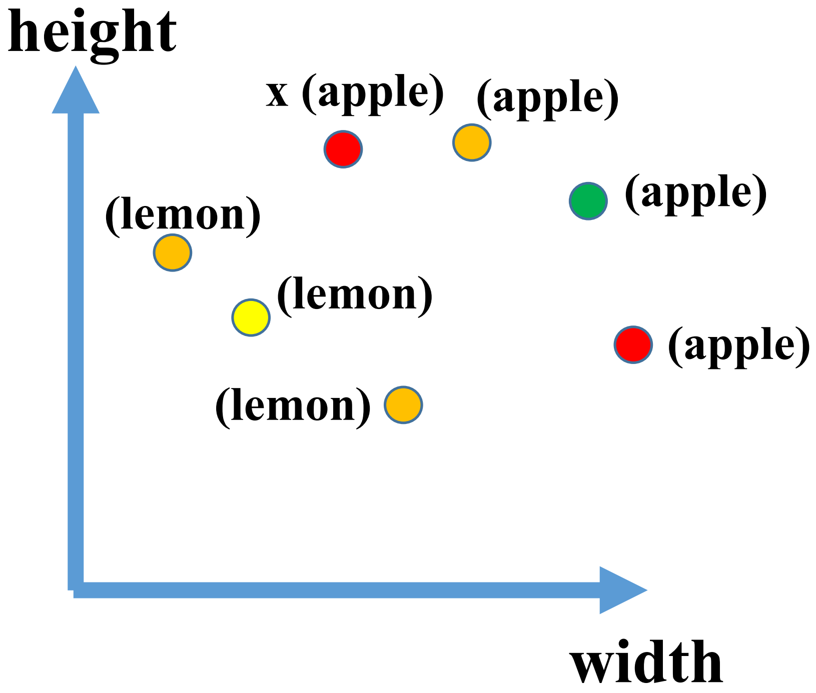
7 samples in the training set



KNN classifier

It memorized all data points in the training set

Use the KNN classifier to predict the label of a sample x that is in the training set

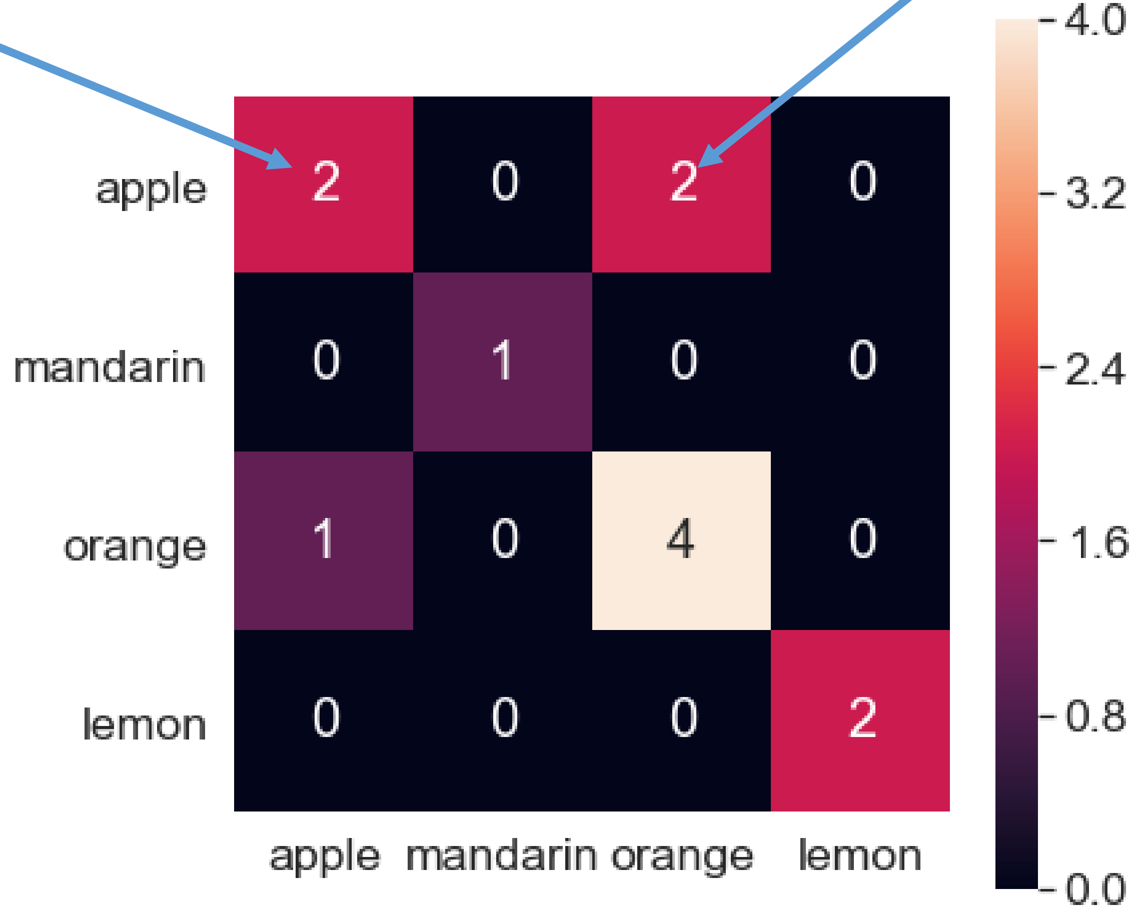


The nearest neighbor of x is itself: x and its label are in KNN's memory

Use confusion matrix to visualize the classification result on the testing set

2 apples are classified as apples

2 apples are classified as oranges



the sum of the numbers in the matrix is the total number of testing samples

The off diagonal numbers show the wrong classifications

The diagonal numbers show the correct classifications

KNN_classification.ipynb

Use hand-engineered features to improve classification accuracy

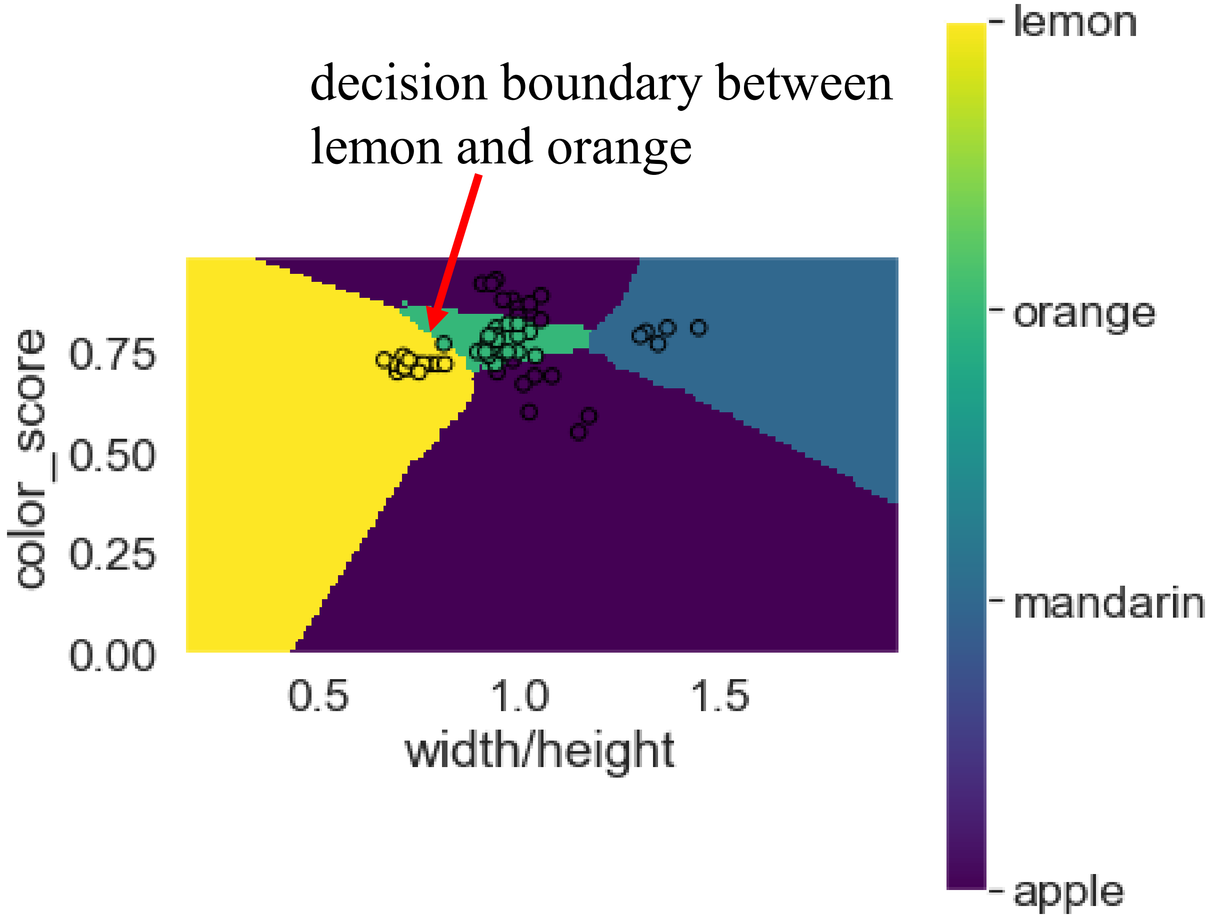
Feature normalization may improve classification accuracy

Plot the **Decision Boundary** to Visualize the Classification Result

A point on the plot represents a sample (a feature vector), which may be in training set or the testing set or unobserved yet.

Roughly speaking, to get the decision boundary plot, we use the KNN classifier to predict the class label of every point on the plot.

In fact, we do not need to check every point: we only need to predict the class labels of the points on a dense grid and interpolate the result.



Question: how do we choose the value of K ?

Question: what is the weakness of KNN classifier ?

What's the difference between clustering and classification ?

Data: $x_1, x_2, x_3, \ldots, x_N$

Input x | Clustering Algorithm | Output y: predicted cluster label

Data: input-output pairs, $(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_N, y_N)$

Input x | Classifier | Output y: predicted class label

KNN can be used for classification and regression

- **For classification**, the output from a KNN classifier is a discrete value (class label), which is done by majority vote
- **For regression**, the output from a KNN regressor is a continues value (target value)
- **For regression**, the average target value of the *K*-nearest neighbors will be the predicted target value of the input $\boldsymbol{x}$

Assume K=3 and training samples $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ are the (K=3) nearest neighbors of $\boldsymbol{x}$, the target values are $\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3$

Then, the predicted target value $\tilde{y}$ of $\boldsymbol{x}$ is $(\boldsymbol{y}_1 + \boldsymbol{y}_2 + \boldsymbol{y}_3)/3$

Boston Housing Dataset

The Housing dataset, which contains information about houses in the different districts of Boston collected by D. Harrison and D.L. Rubinfeld in 1978.

The dataset is a large table that has 506 samples (rows) and 14 columns

Each row contains information/attributes of a region in Boston

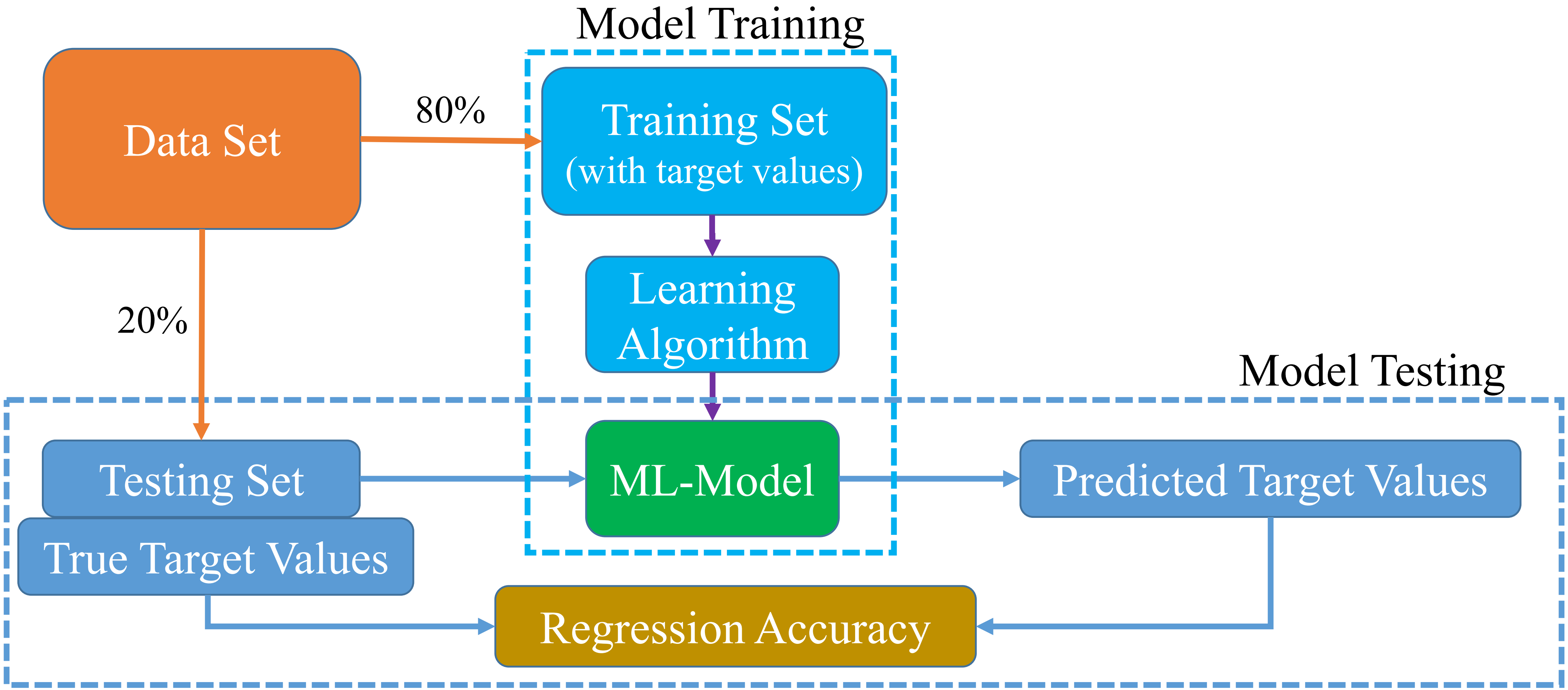
y: target

x: input (13 attributes)

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

MEDV: Median value of owner-occupied homes in $1000s

Regression y=f(x)



KNN_Regression.ipynb