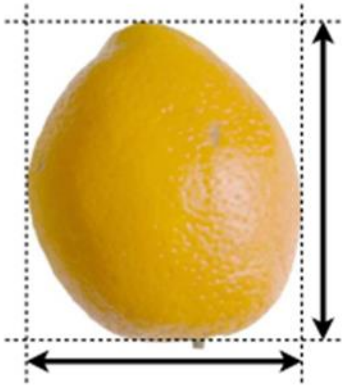
The Fruit Dataset

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/

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

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

```
1  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.

```
1  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.

```
1  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.

```
1  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.

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

X[0], or X[0,:]

X[1], or X[1,:]

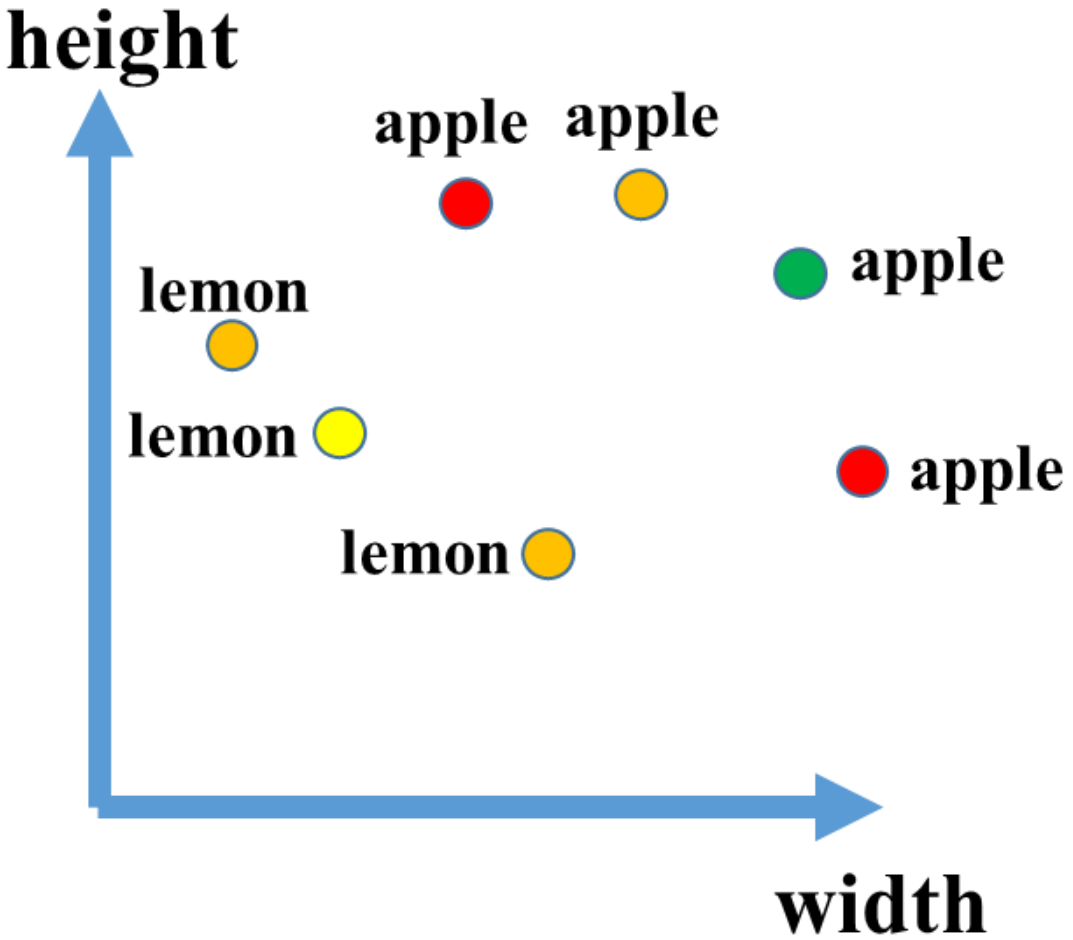
4 classes: {1:apple, 2:mandarin, 3:orange, 4:lemon}

Read KNN_friut_classification.ipynb

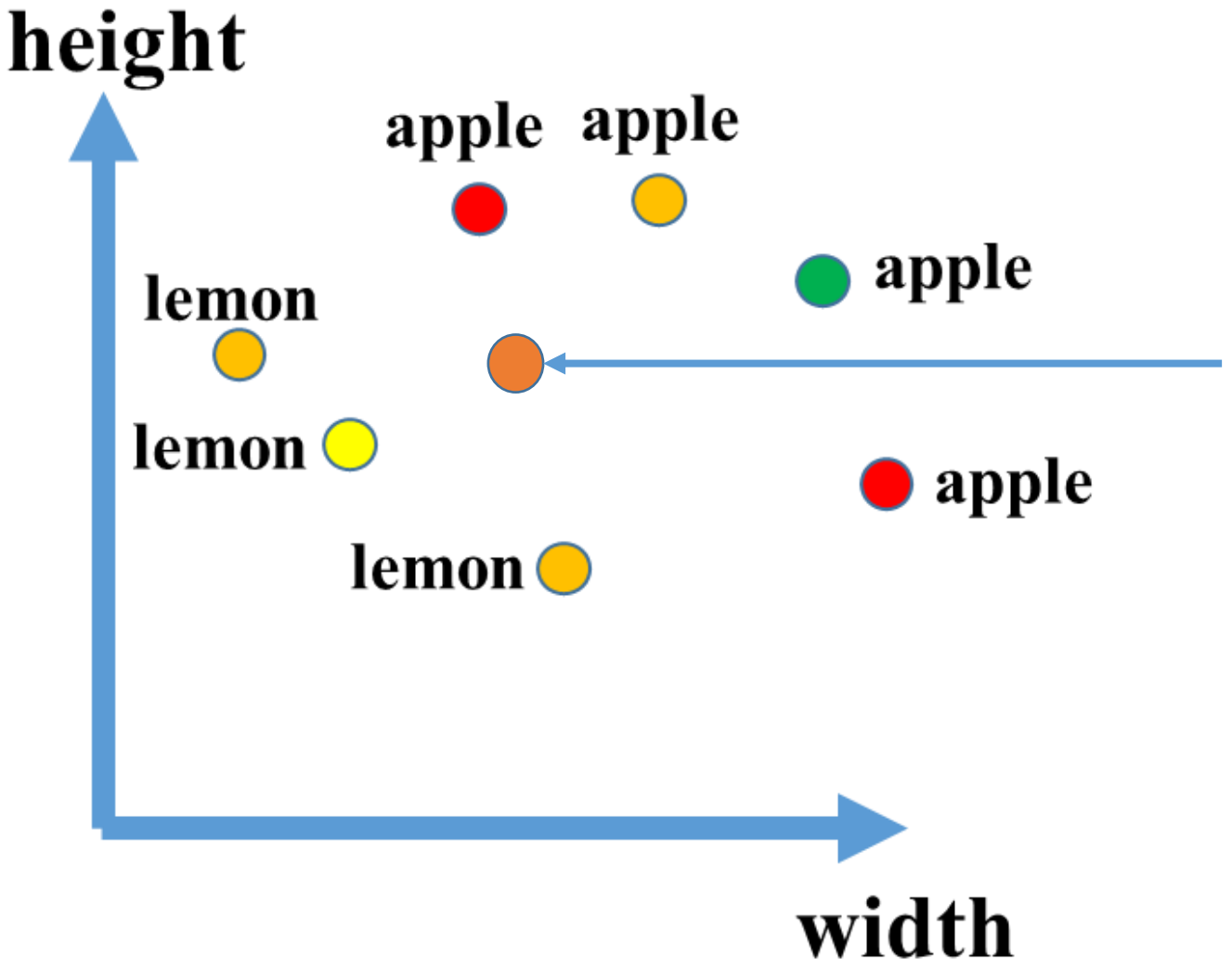
KNN classifier (K-Nearest Neighbor)

- 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

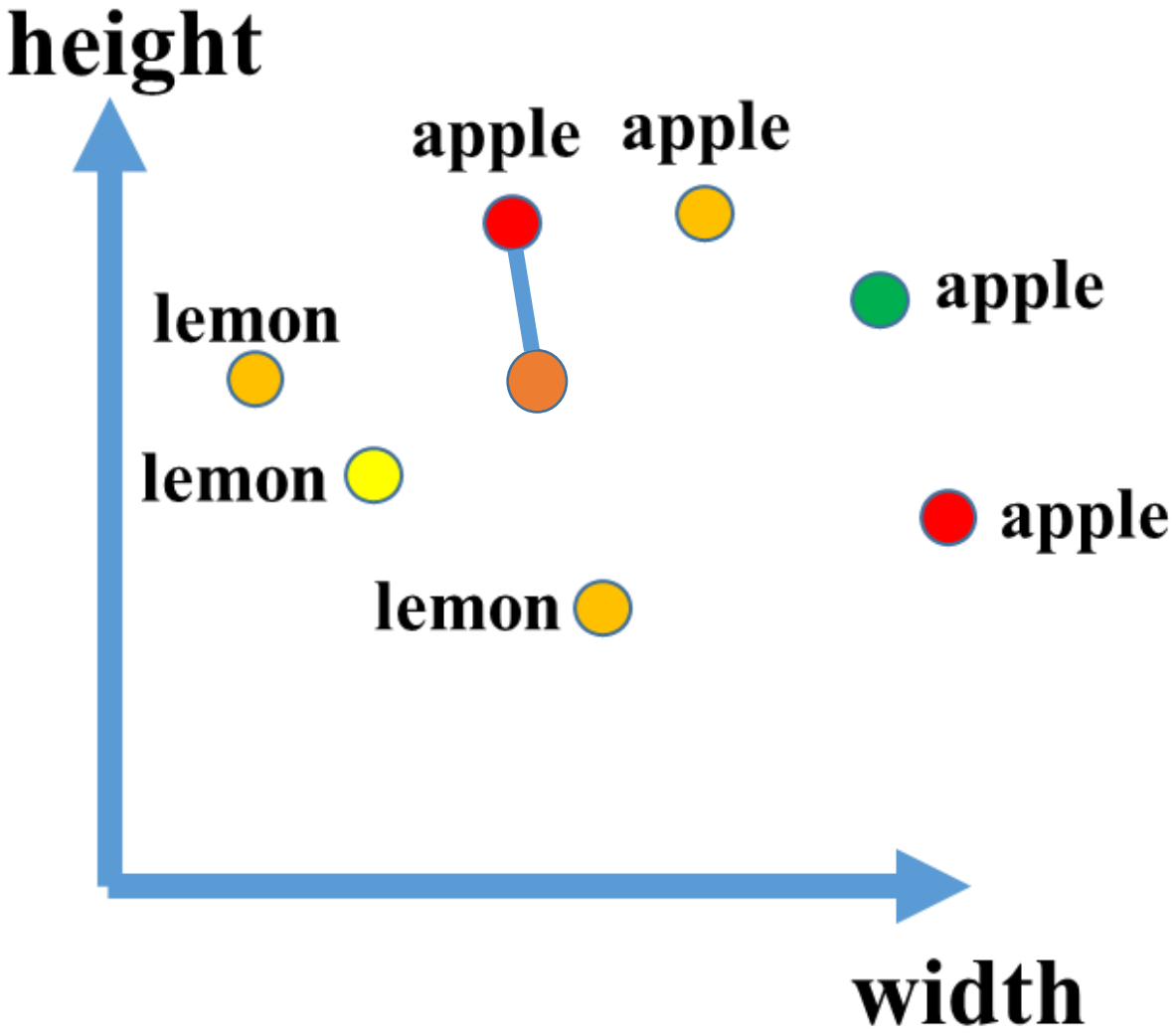


a new sample, class label is unknown

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)



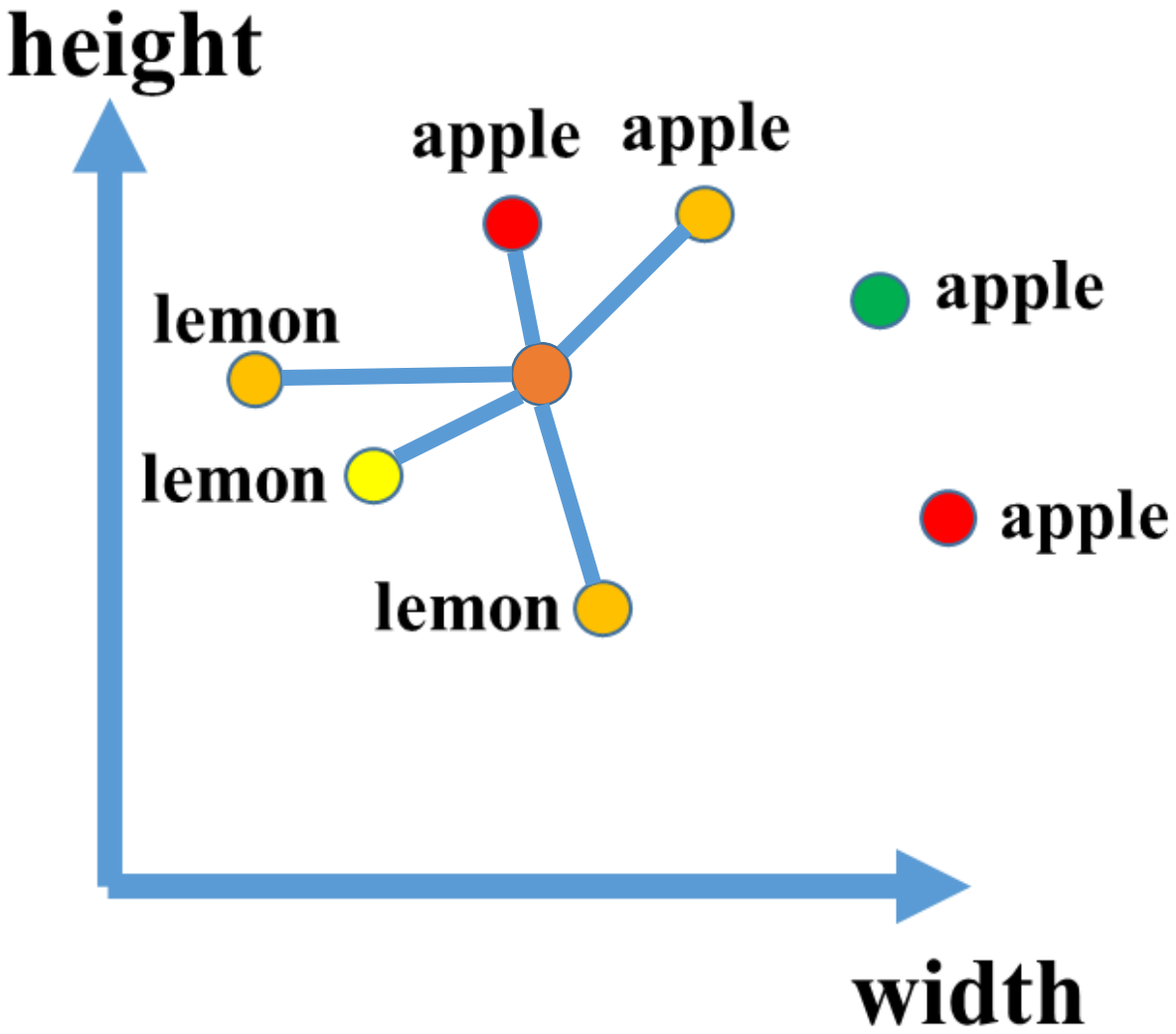
7 samples in training set

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

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

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

K=5

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

```
1  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')
```

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

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

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

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

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

```
0.8723404255319149
```

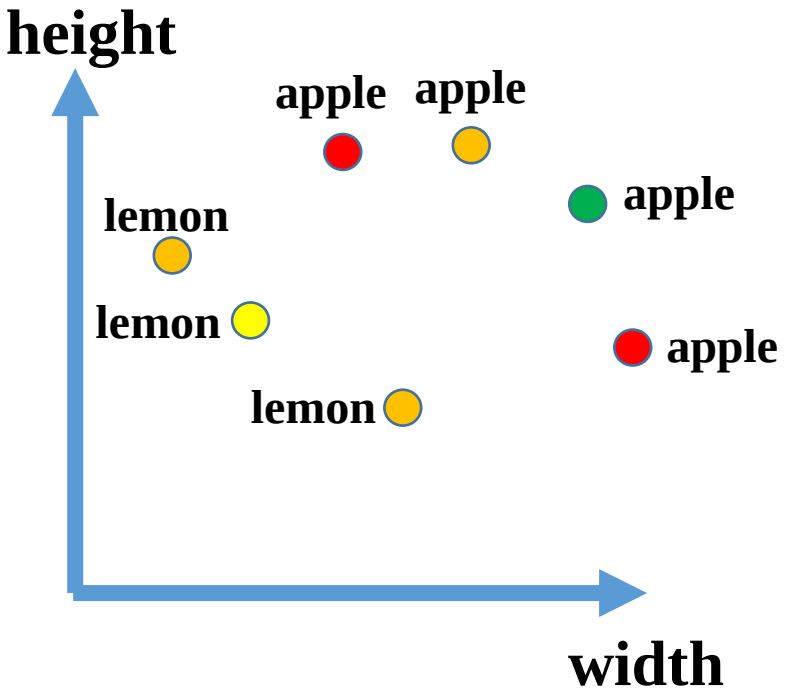
Testing Accuracy: accuracy on testing set (20% of the data)

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

```
0.75
```

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

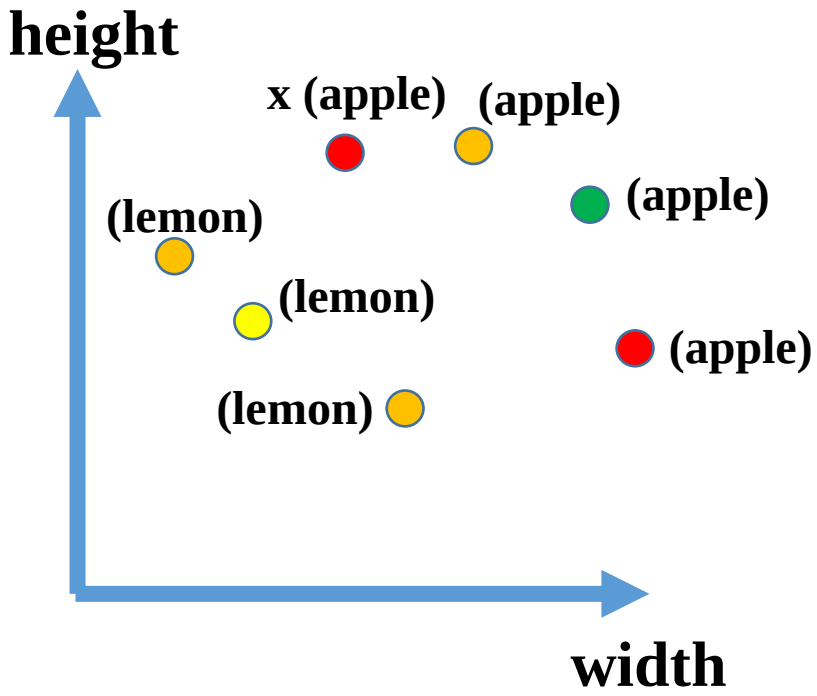
7 samples in the training set



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

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 $\boldsymbol{x_1, x_2, x_3}$ are the (K=3) nearest neighbors of $\boldsymbol{x}$, the target values are $\boldsymbol{y_1, y_2, y_3}$

Then, the predicted target value $\tilde{y}$ of $\boldsymbol{x}$ is $\boldsymbol{(y_1 + y_2 + y_3)/3}$

Question: how do we choose the value of K ?

Cross-Validation or Train-Validation