k-NN Algorithm for Classification

Classification

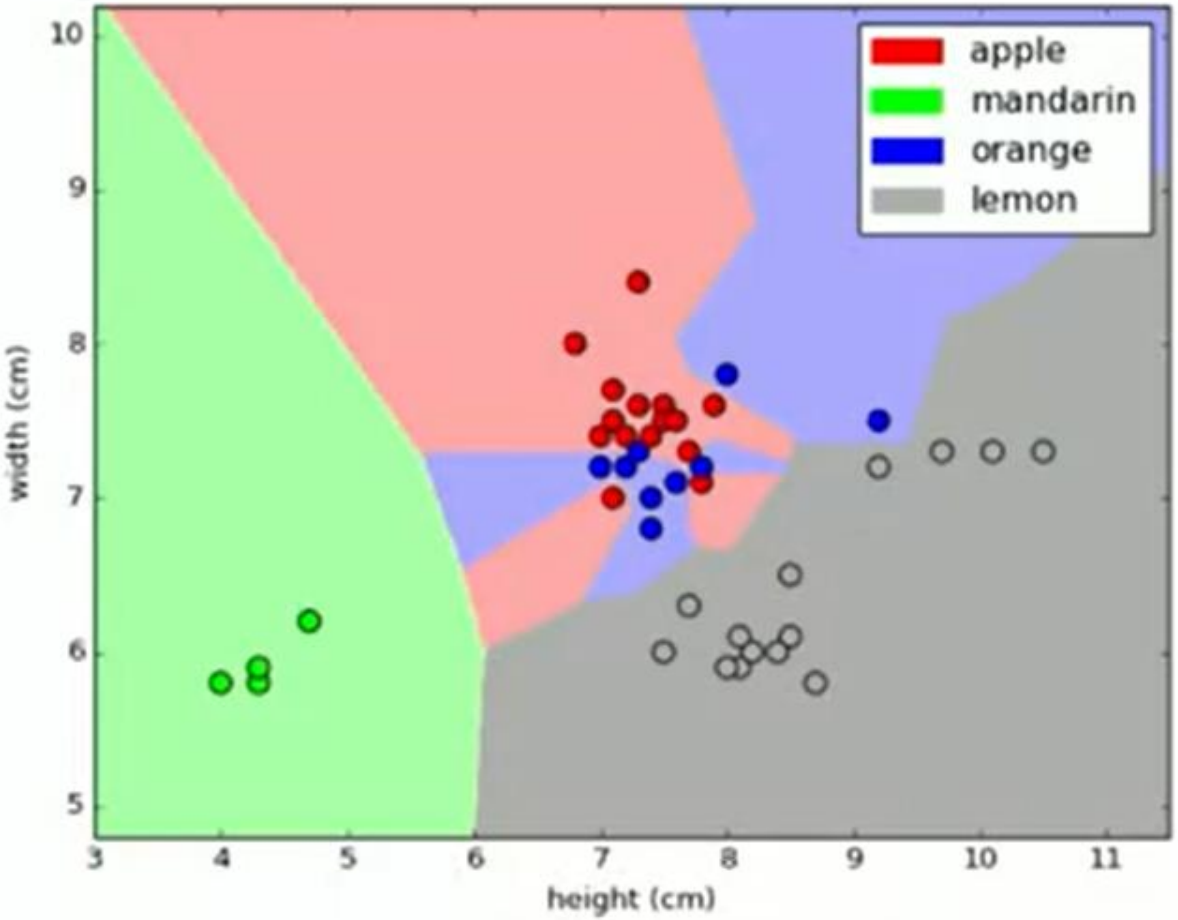
Regression

The k-Nearest Neighbor (k-NN) Classifier Algorithm

Given a training set X_train with labels y_train, and given a new instance x_test to be classified:

1. Find the most similar instances (let's call them X_NN) to x_test that are in X_train.
2. Get the labels y_NN for the instances in X_NN
3. Predict the label for x_test by combining the labels y_NN
   e.g. simple majority vote

A visual explanation of k-NN classifiers



Fruit dataset
Decision boundaries
with k = 1

A nearest neighbor algorithm needs four things specified

1. A distance metric
2. How many 'nearest' neighbors to look at?
3. Optional weighting function on the neighbor points
4. Method for aggregating the classes of neighbor points

A nearest neighbor algorithm needs four things specified

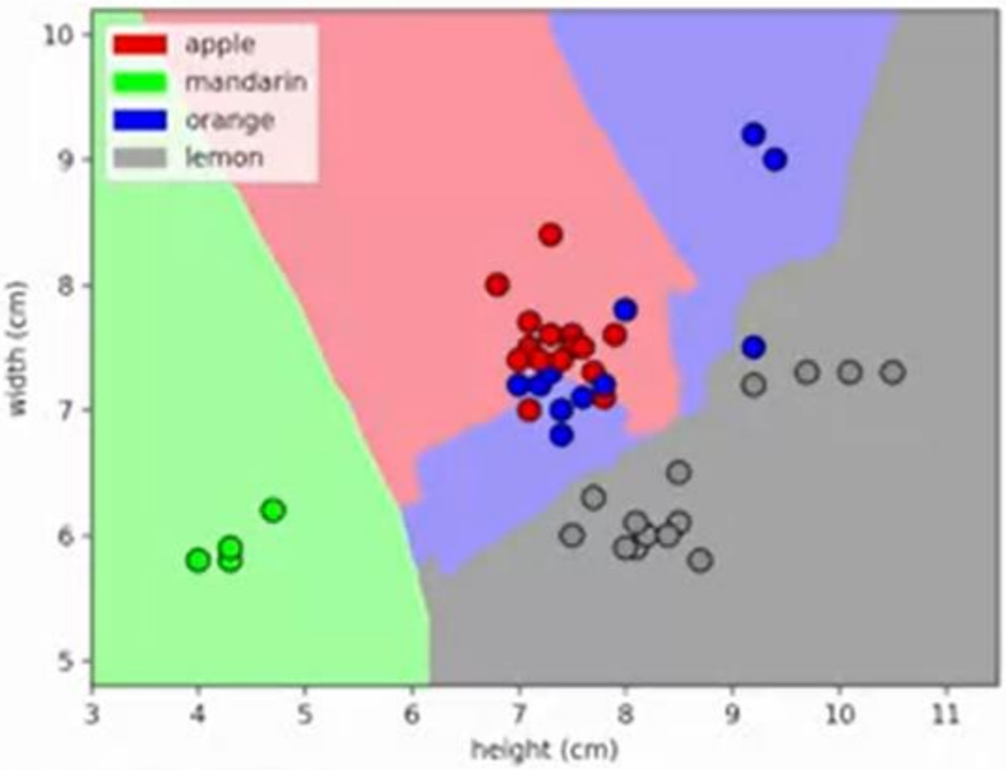
1. A distance metric
   Typically Euclidean (Minkowski with p = 2)
2. How many 'nearest' neighbors to look at?
   e.g. five
3. Optional weighting function on the neighbor points
   Ignored
4. How to aggregate the classes of neighbor points
   Simple majority vote
   (Class with the most representatives among nearest neighbors)

Plot the decision boundaries of the k-NN classifier

In [10]:
```
from adspy_shared_utilities import plot_fruit_knn

plot_fruit_knn(X_train, y_train, 5, 'uniform')
```

Figure 1



In [ ]:

Bias – variance trade-off

- For larger values of K, the areas assigned to different classes are smoother and not as fragmented and more robust to noise in the individual points.
- But possibly with some mistakes, more mistakes in individual points.
- This is an example of what we know as the **bias variance tradeoff**.

- Consider the following example.

K-Nearest Neighbors

Classification

In [*]:

```
from adspy_shared_utilities import plot_two_class_knn

X_train, X_test, y_train, y_test = train_test_split(X_C2, y_C2,
                                                    random_state=0)

plot_two_class_knn(X_train, y_train, 1, 'uniform', X_test, y_test)
plot_two_class_knn(X_train, y_train, 3, 'uniform', X_test, y_test)
plot_two_class_knn(X_train, y_train, 11, 'uniform', X_test, y_test)
```

In [ ]:

```
plot_two_class_knn(X_train, y_train, 11, 'uniform', X_test, y_test)
```

Figure 5

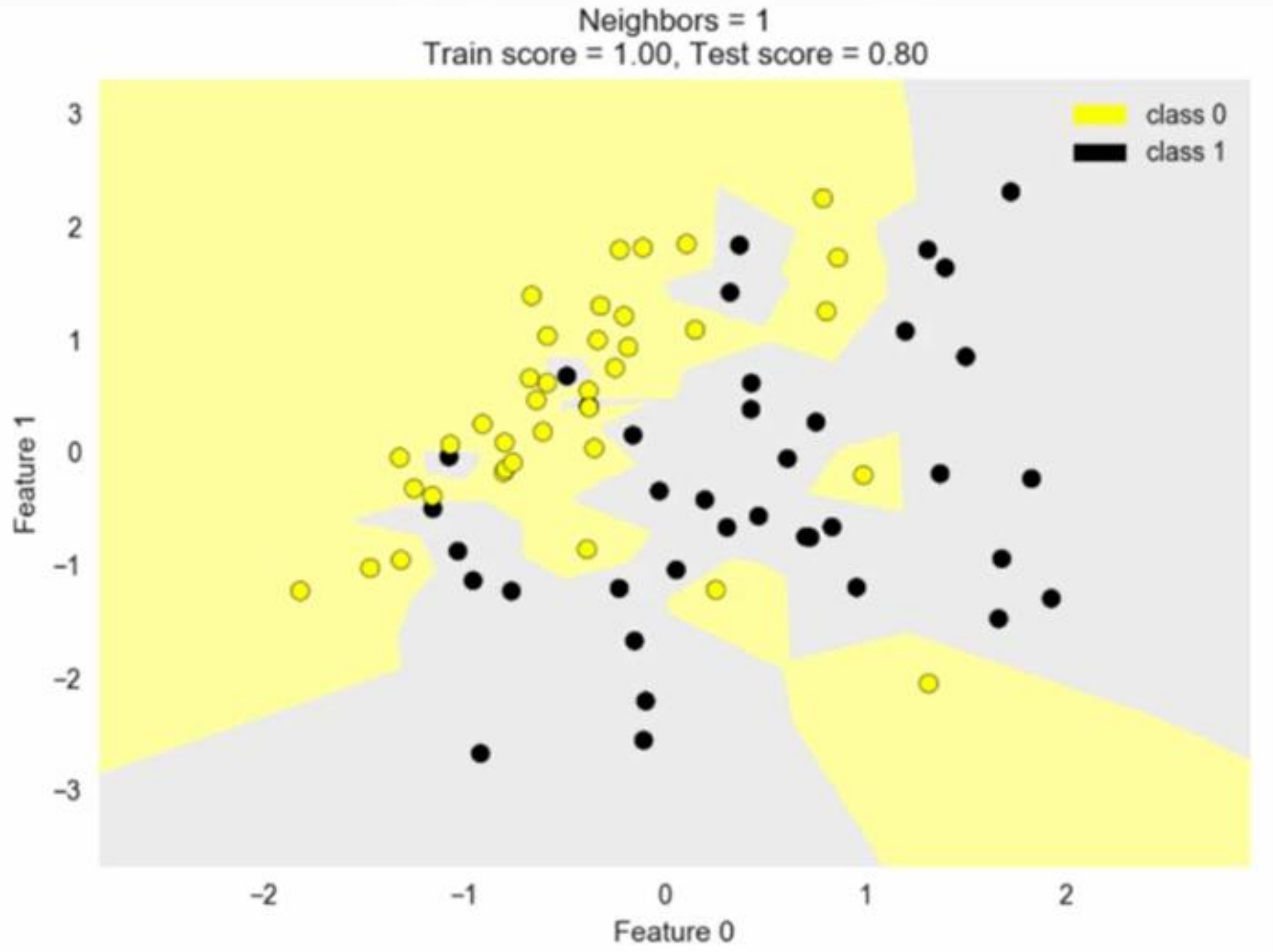


Figure 6

```
plot_two_class_knn(X_train, y_train, 11, 'uniform', X_test, y_test)
```

Figure 5

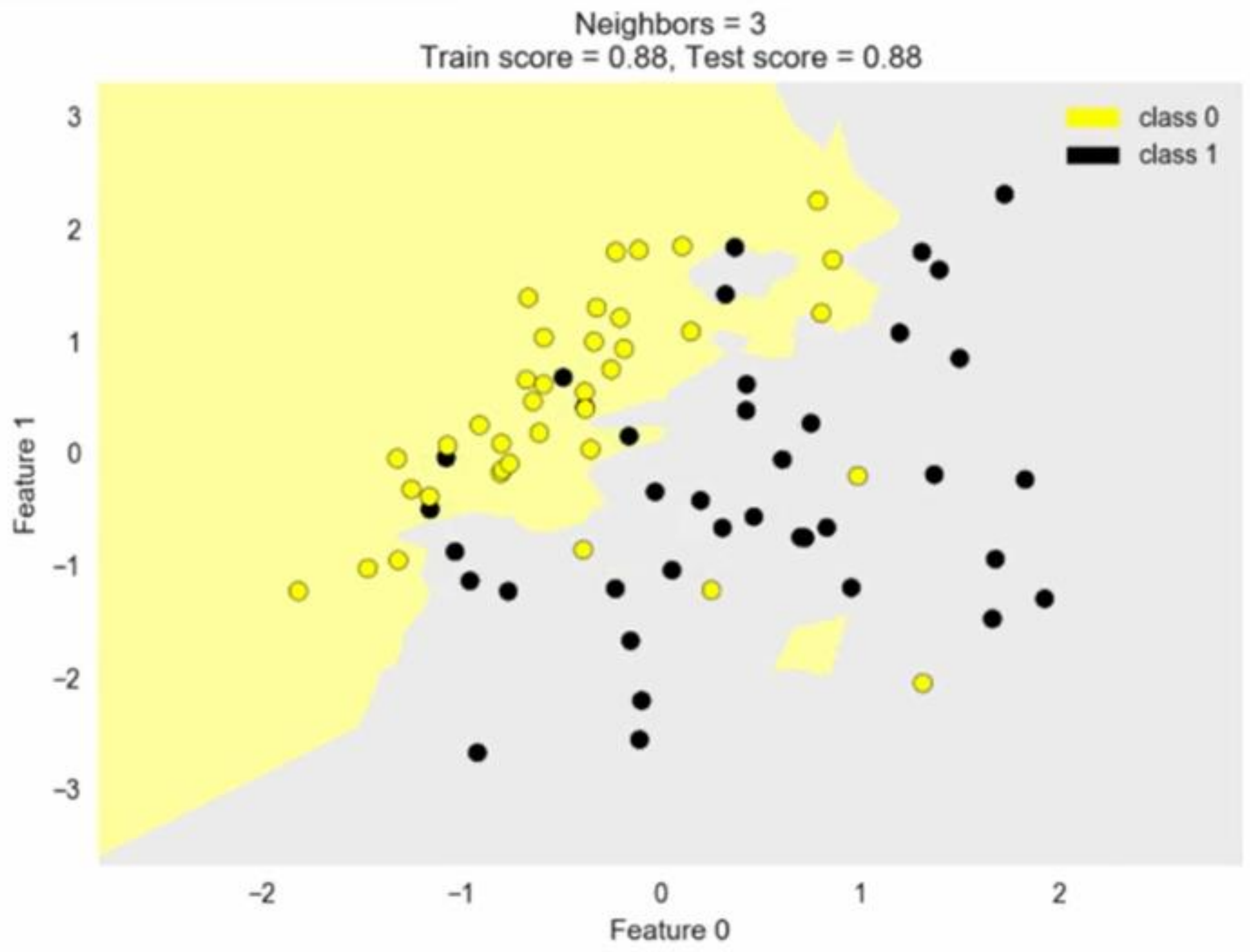
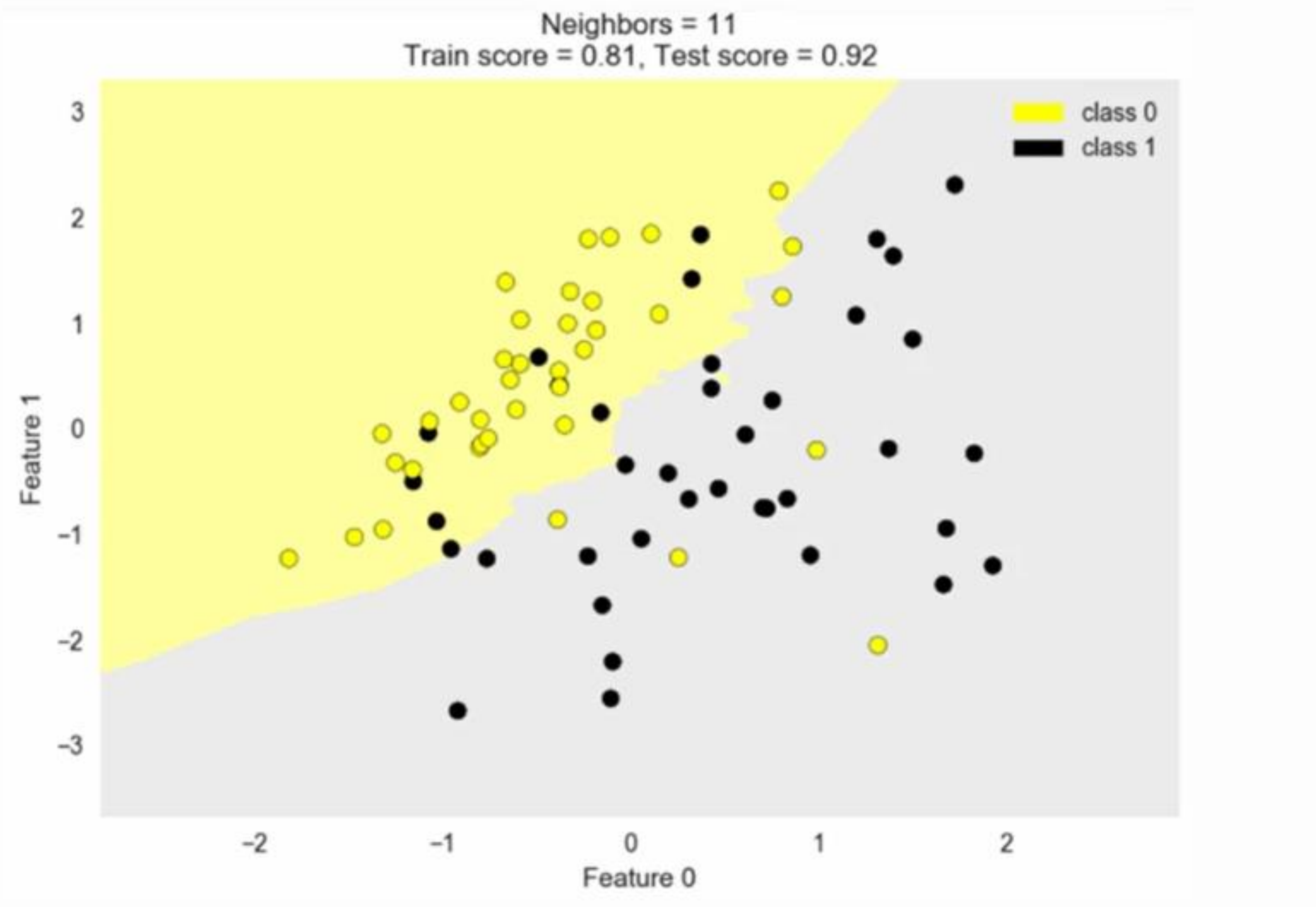


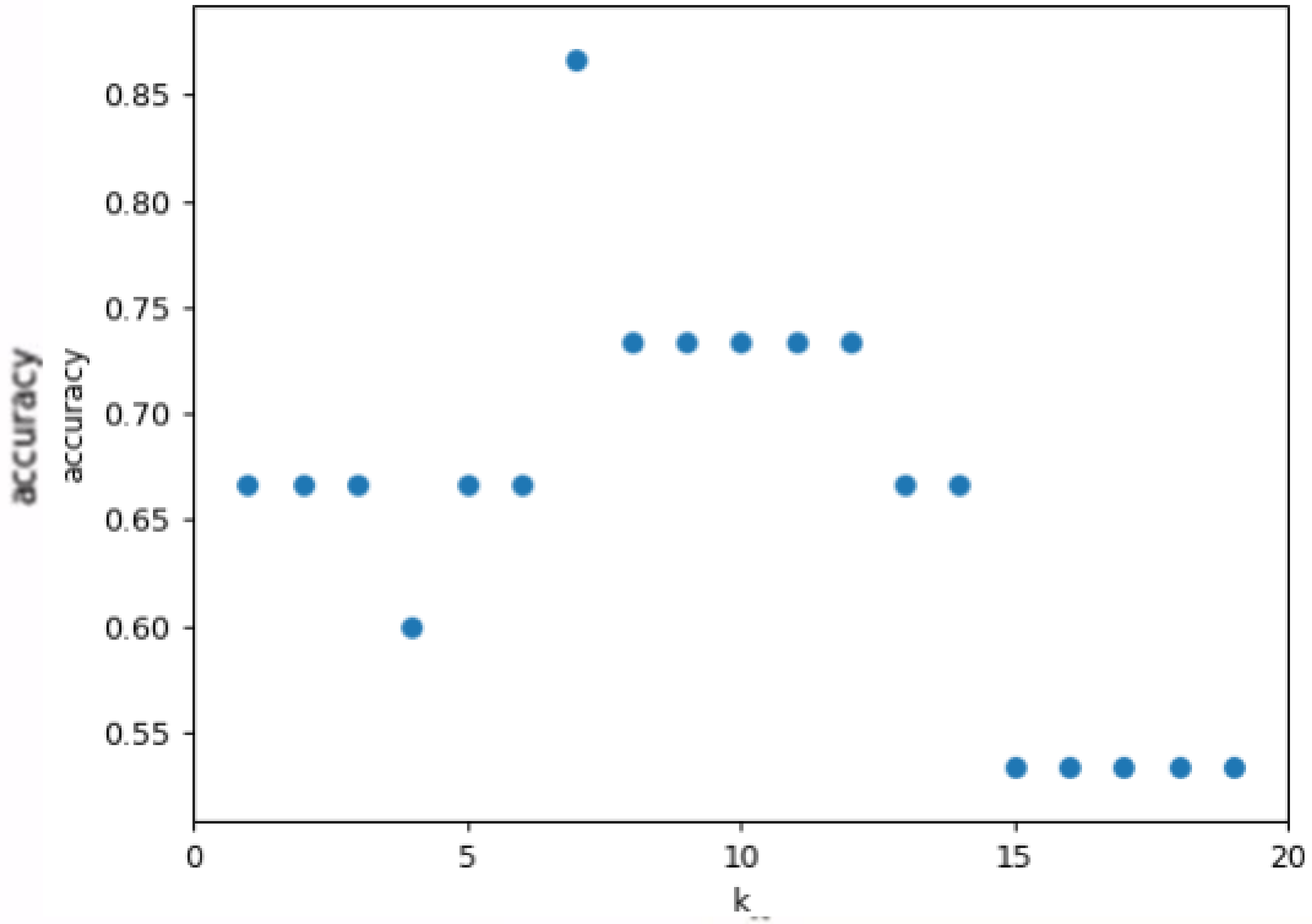
Figure 6

```
plot_two_class_knn(X_train, y_train, 11, 'uniform', X_test, y_test)
```

Figure 5



How sensitive is k-NN classifier accuracy to the choice of 'k' parameter?



Choose only colors as the feature set.

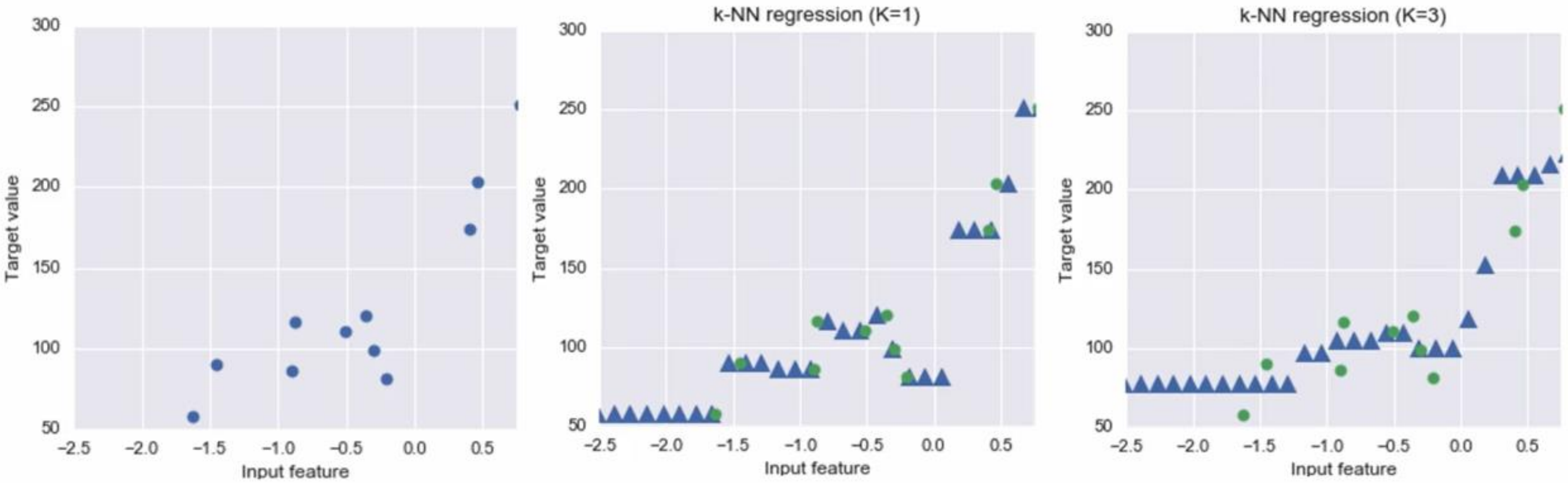
Fruit dataset with 75%/25% train-test split

k-NN Algorithm for Regression

Classification

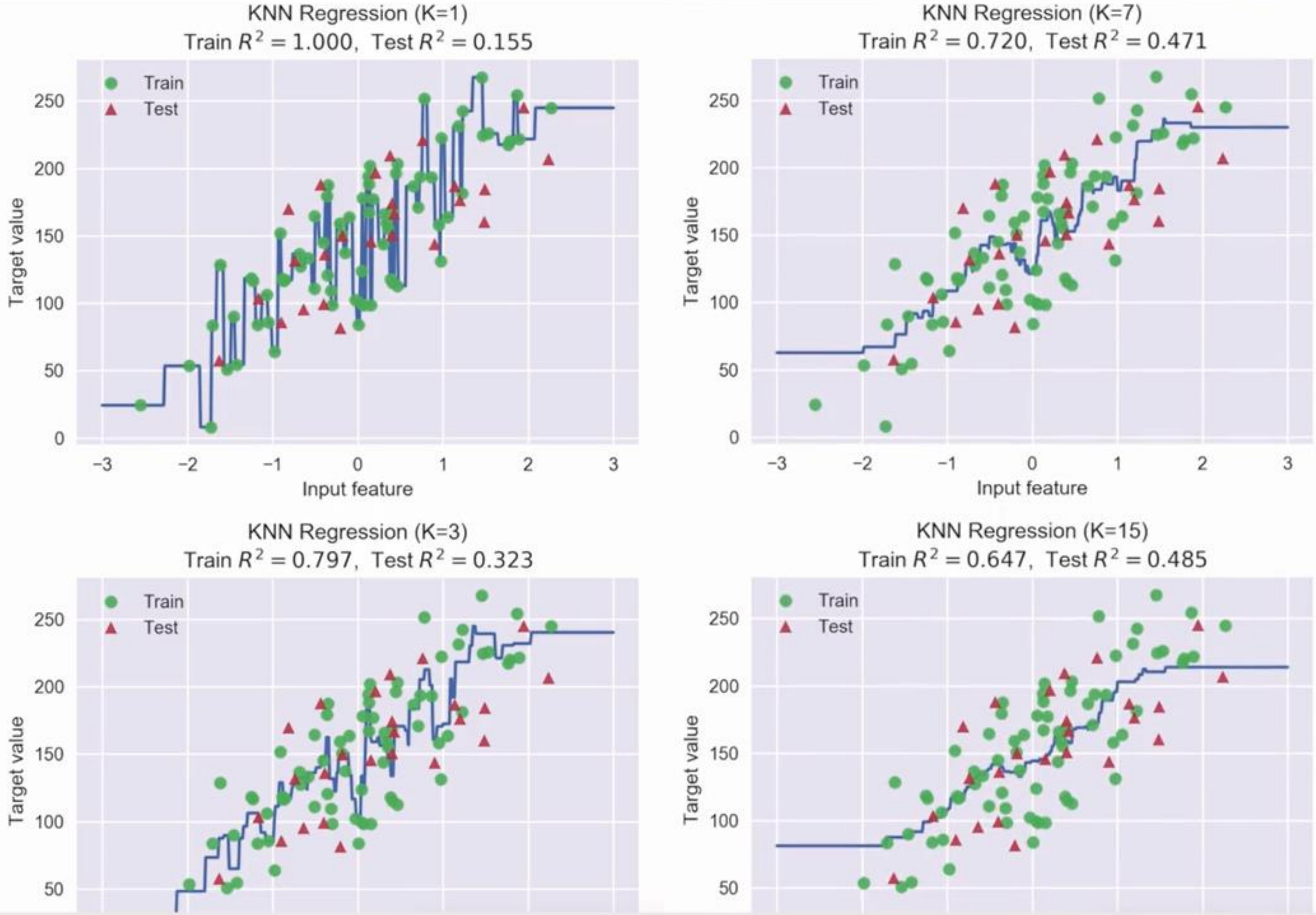
Regression

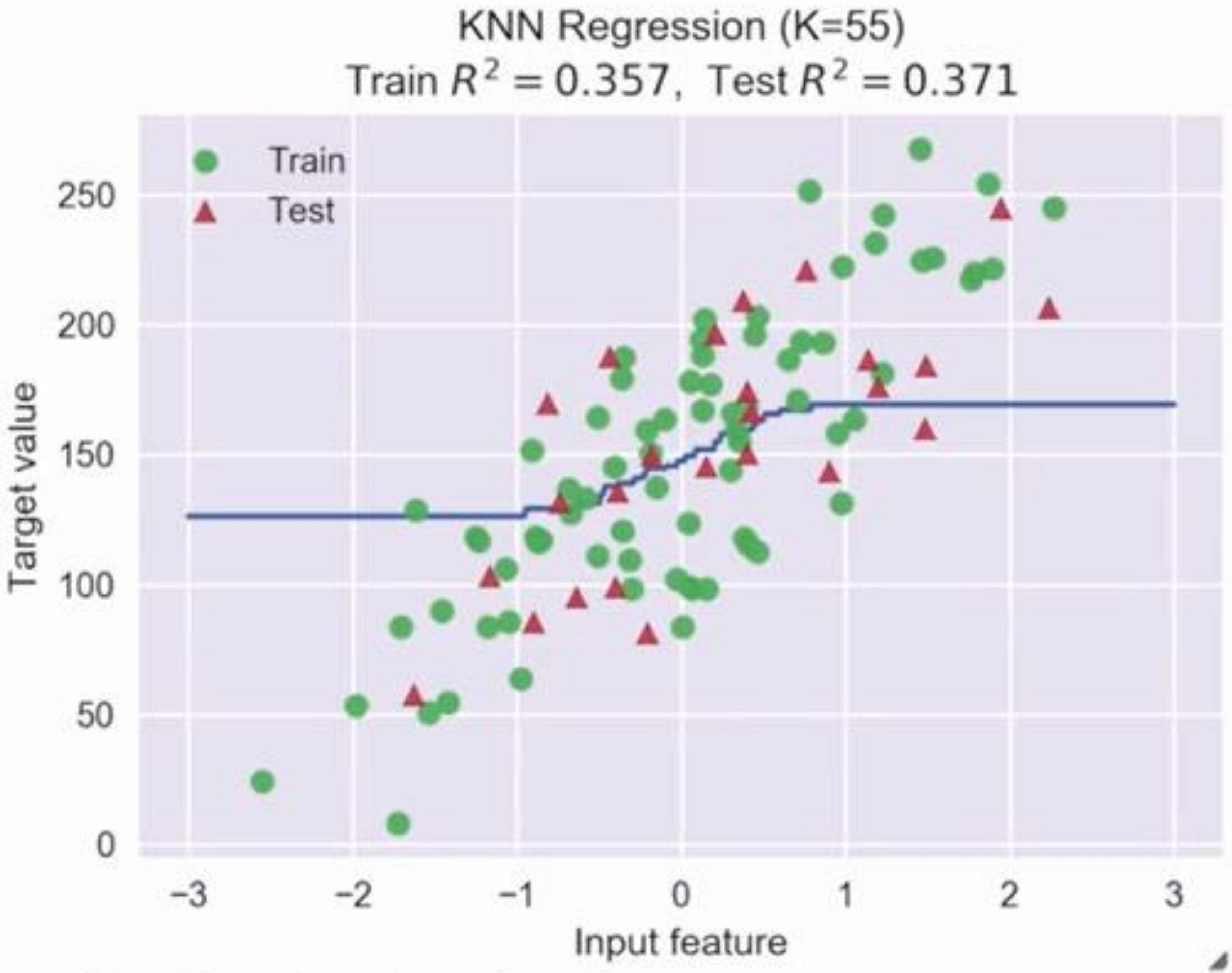
k-Nearest neighbors regression



The R^2 ("r-squared") regression score

- Measures how well a prediction model for regression fits the given data.
- The score is between 0 and 1:
  - *A value of 0 corresponds to a constant model that predicts the mean value of all training target values.*
  - *A value of 1 corresponds to perfect prediction*
- Also known as "coefficient of determination"


KNN Regression (K=1)
Train $R^2 = 1.000$, Test $R^2 = 0.155$
Train
Test
Target value
Input feature
KNN Regression (K=7)
Train $R^2 = 0.720$, Test $R^2 = 0.471$
Train
Test
Target value
Input feature
KNN Regression (K=3)
Train $R^2 = 0.797$, Test $R^2 = 0.323$
Train
Test
Target value
KNN Regression (K=15)
Train $R^2 = 0.647$, Test $R^2 = 0.485$
Train
Test
Target value


KNN Regression (K=55)
Train $R^2 = 0.357$, Test $R^2 = 0.371$
Train
Test
250
200
150
100
50
0
Target value
−3
−2
−1
0
1
2
3
Input feature

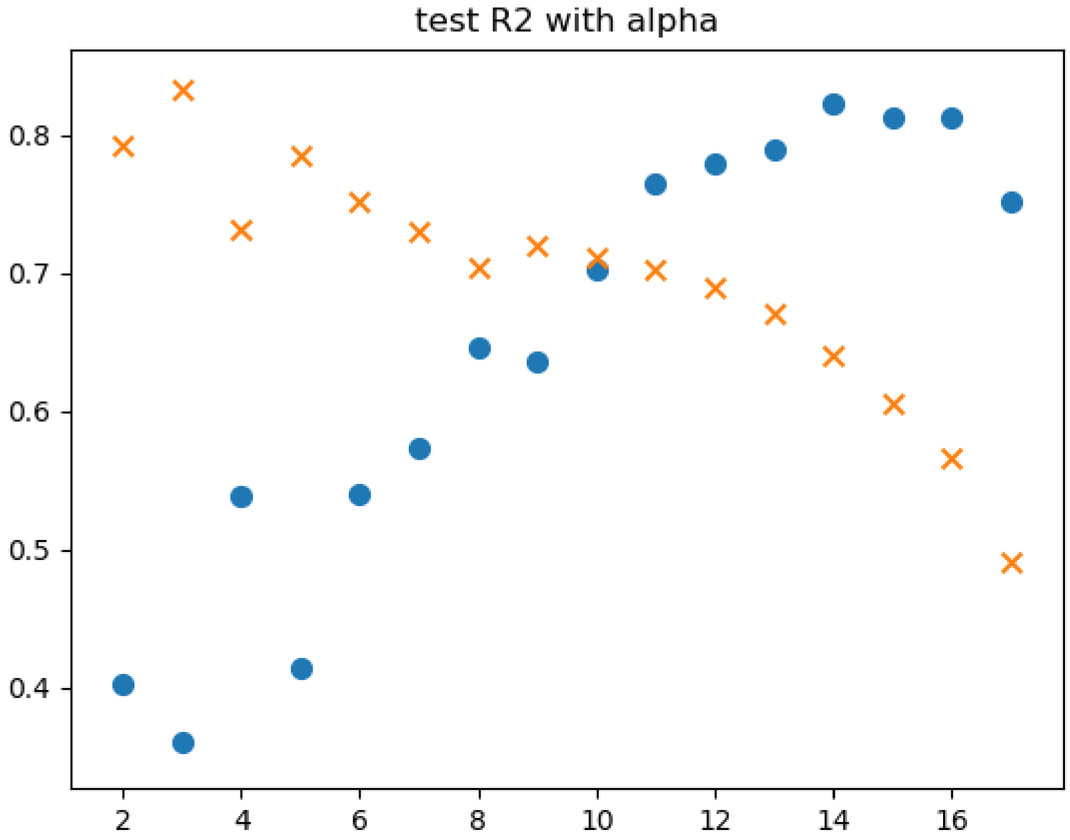
KNeighborsClassifier and KNeighborsRegressor: important parameters

Model complexity

- *n_neighbors* : number of nearest neighbors (k) to consider
  - *Default = 5*

Model fitting

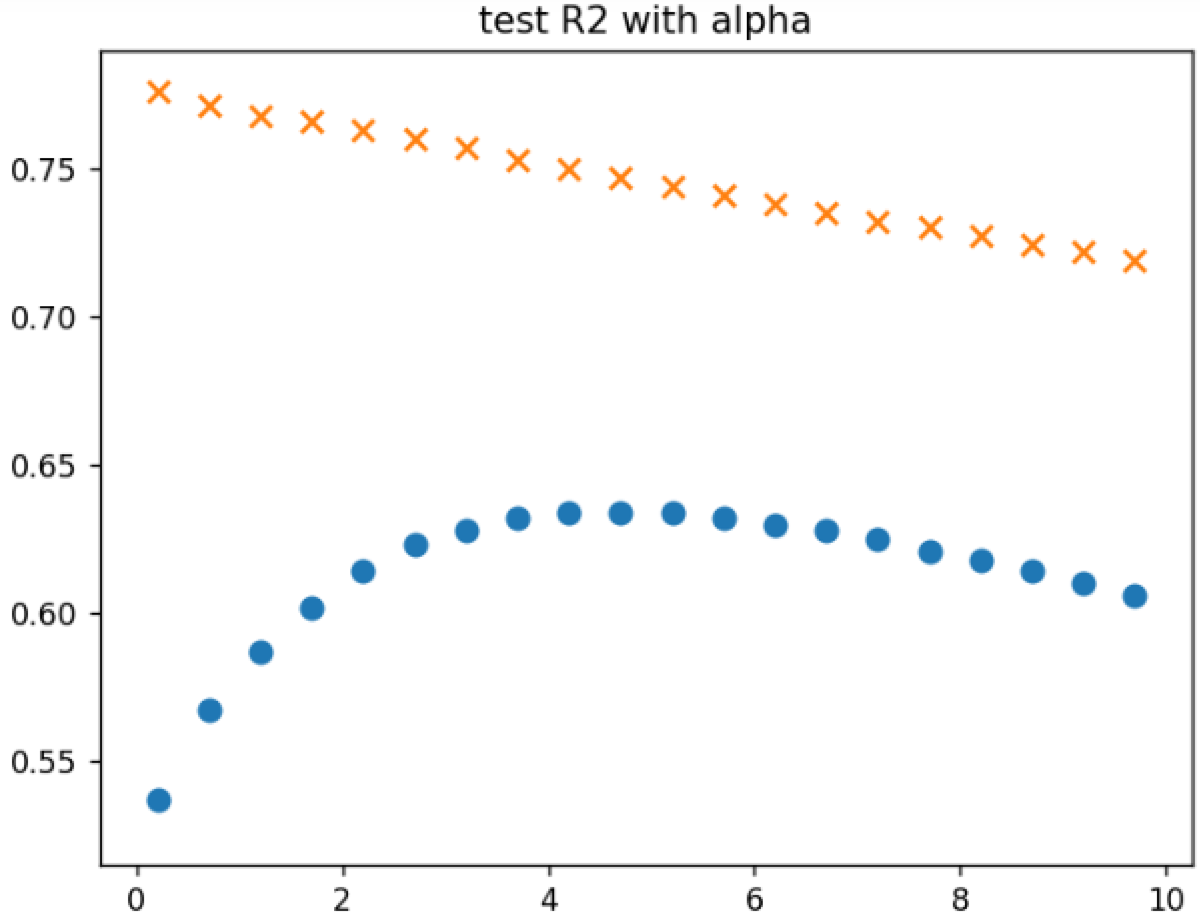
- *metric: distance function between data points*
  - *Default: Minkowski distance with power parameter p = 2 (Euclidean)*



KNN R^2 values for

train (orange) and test (blue)

vs **k** levels



Ridge R^2 values for

train (orange) and test (blue)

vs **alpha** levels