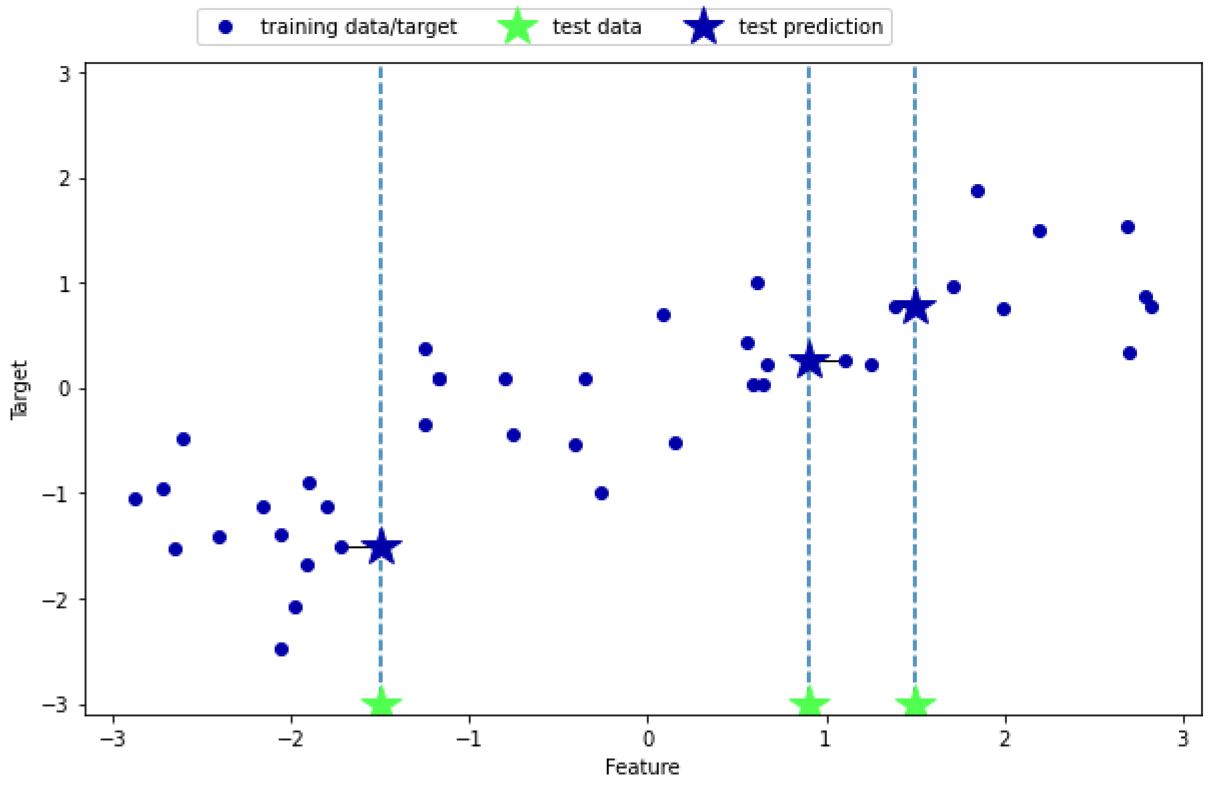
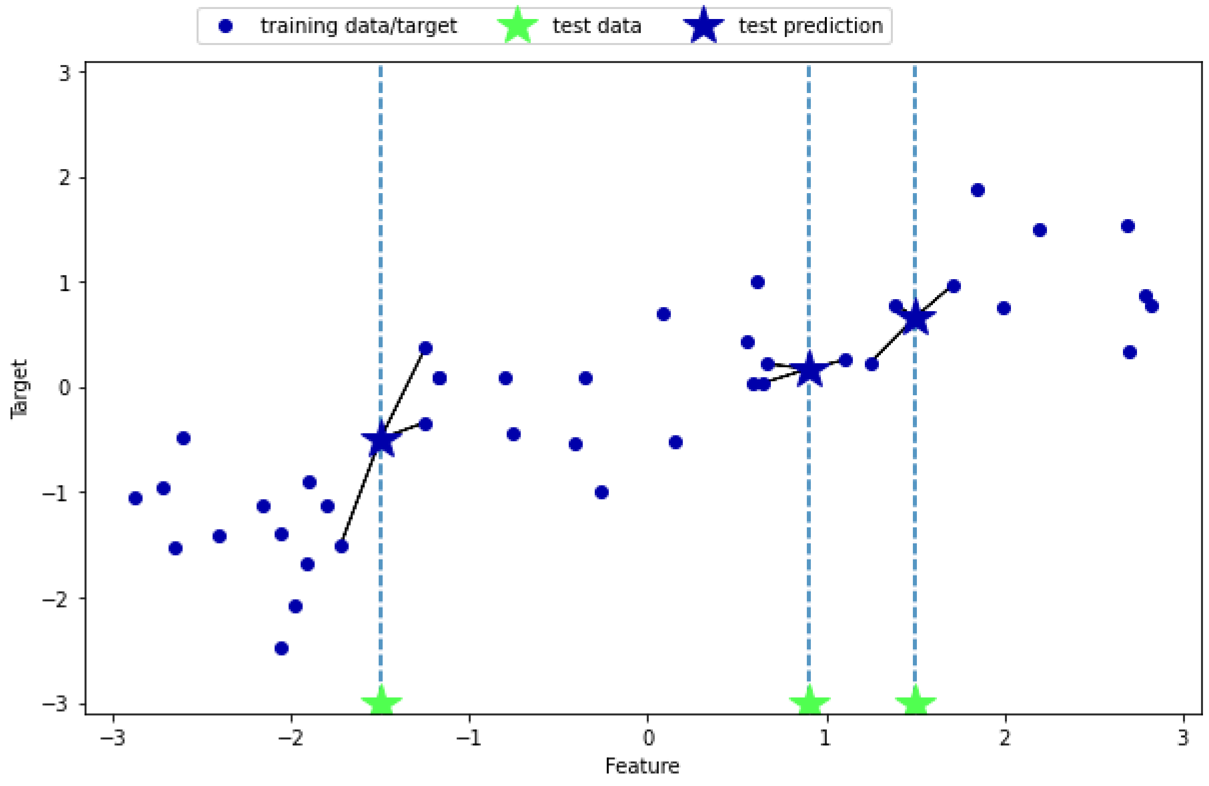
In [1]:
```
import mglearn
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
from sklearn.model_selection import train_test_split
```

In [3]:
```
mglearn.plots.plot_knn_regression(n_neighbors=1)
```



In [4]:
```
mglearn.plots.plot_knn_regression(n_neighbors=3)
```



```
In [5]:
from sklearn.neighbors import KNeighborsRegressor
```

```
In [6]:
X, y = mglearn.datasets.make_wave(n_samples=40)
```

```
In [7]:
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
reg = KNeighborsRegressor(n_neighbors=3)
reg.fit(X_train, y_train)
```

```
Out[7]: KNeighborsRegressor(n_neighbors=3)
```

```
In [8]:
print("Test set predictions:\n{}".format(reg.predict(X_test)))
```

```
Test set predictions:
[-0.05396539  0.35686046  1.13671923 -1.89415682 -1.13881398 -1.63113382
  0.35686046  0.91241374 -0.44680446 -1.13881398]
```

```
In [10]:
print("Test set R^2: {:.2f}".format(reg.score(X_test, y_test)))
```

```
Test set R^2: 0.83
```

Here, the score is 0.83, which indicates a relatively good model fit.

```
In [11]:
fig, axes = plt.subplots(1, 3, figsize=(15, 4))

# create 1,000 data points, evenly spaced between -3 and 3
line = np.linspace(-3, 3, 1000).reshape(-1, 1)

for n_neighbors, ax in zip([1, 3, 9], axes):
 # make predictions using 1, 3, or 9 neighbors
```

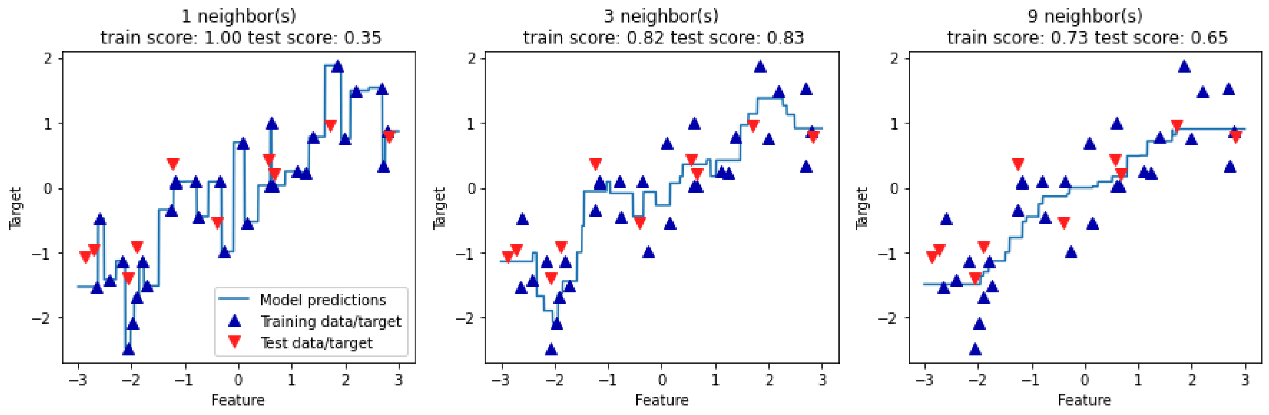
```
    reg = KNeighborsRegressor(n_neighbors=n_neighbors)
    reg.fit(X_train, y_train)

    ax.plot(line, reg.predict(line))
    ax.plot(X_train, y_train, '^', c=mglearn.cm2(0), markersize=8)
    ax.plot(X_test, y_test, 'v', c=mglearn.cm2(1), markersize=8)
    ax.set_title("{} neighbor(s)\n train score: {:.2f} test score: {:.2f}".format(
                 n_neighbors, reg.score(X_train, y_train),
                 reg.score(X_test, y_test)))

    ax.set_xlabel("Feature")
    ax.set_ylabel("Target")
axes[0].legend(["Model predictions", "Training data/target",
                "Test data/target"], loc="best")
```

Out[11]:

```
<matplotlib.legend.Legend at 0x28ee7596d00>
```



Considering more neighbors leads to smoother predictions, but these do not fit the training data as well.

The number of neighbors and how you measure distance between data points. In practice, using a small number of neighbors like three or five often works well, but you should certainly adjust this parameter.

So, while the nearest k-neighbors algorithm is easy to understand, it is not often used in practice, due to prediction being slow and its inability to handle many features.