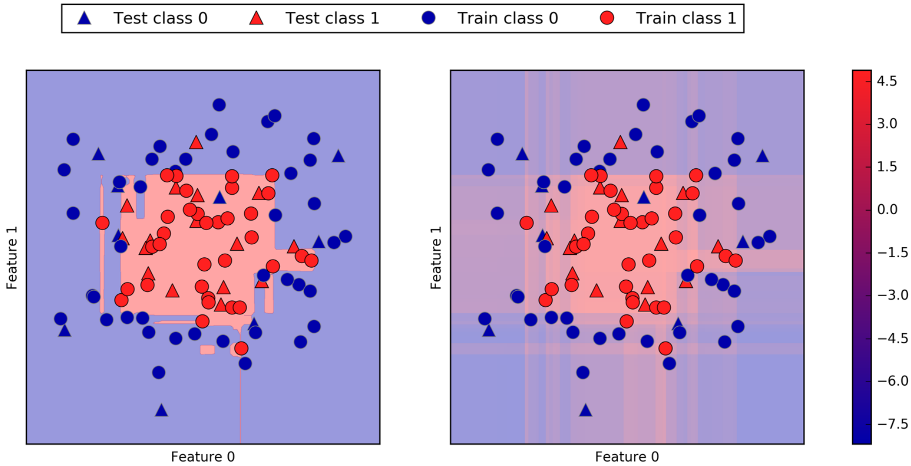

Figure 2-55. Decision boundary (left) and decision function (right) for a gradient boosting model on a two-dimensional toy dataset

Encoding not only the predicted outcome but also how certain the classifier is provides additional information. However, in this visualization, it is hard to make out the boundary between the two classes.

Predicting Probabilities

The output of `predict_proba` is a probability for each class, and is often more easily understood than the output of `decision_function`. It is always of shape `(n_samples, 2)` for binary classification:

In[112]:

```
print("Shape of probabilities: {}".format(gbrt.predict_proba(X_test).shape))
```

Out[112]:

```
Shape of probabilities: (25, 2)
```

The first entry in each row is the estimated probability of the first class, and the second entry is the estimated probability of the second class. Because it is a probability, the output of `predict_proba` is always between 0 and 1, and the sum of the entries for both classes is always 1:

In[113]:

```
# show the first few entries of predict_proba
print("Predicted probabilities:\n{}".format(
    gbrt.predict_proba(X_test[:6])))
```

Out[113]:

```
Predicted probabilities:
[[ 0.016  0.984]
 [ 0.843  0.157]
 [ 0.981  0.019]
 [ 0.974  0.026]
 [ 0.014  0.986]
 [ 0.025  0.975]]
```

Because the probabilities for the two classes sum to 1, exactly one of the classes will be above 50% certainty. That class is the one that is predicted.[13]

You can see in the previous output that the classifier is relatively certain for most points. How well the uncertainty actually reflects uncertainty in the data depends on the model and the parameters. A model that is more overfitted tends to make more certain predictions, even if they might be wrong. A model with less complexity usually has more uncertainty in its predictions. A model is called *calibrated* if the reported uncertainty actually matches how correct it is—in a calibrated model, a prediction made with 70% certainty would be correct 70% of the time.

In the following example (Figure 2-56) we again show the decision boundary on the dataset, next to the class probabilities for the class 1:

In[114]:

```
fig, axes = plt.subplots(1, 2, figsize=(13, 5))

mglearn.tools.plot_2d_separator(
    gbrt, X, ax=axes[0], alpha=.4, fill=True, cm=mglearn.cm2)
scores_image = mglearn.tools.plot_2d_scores(
    gbrt, X, ax=axes[1], alpha=.5, cm=mglearn.ReBl, function='predict_proba')

for ax in axes:
    # plot training and test points
    mglearn.discrete_scatter(X_test[:, 0], X_test[:, 1], y_test,
                             markers='^', ax=ax)
    mglearn.discrete_scatter(X_train[:, 0], X_train[:, 1], y_train,
                             markers='o', ax=ax)
    ax.set_xlabel("Feature 0")
    ax.set_ylabel("Feature 1")
cbar = plt.colorbar(scores_image, ax=axes.tolist())
axes[0].legend(["Test class 0", "Test class 1", "Train class 0",
                "Train class 1"], ncol=4, loc=(.1, 1.1))
```

13 Because the probabilities are floating-point numbers, it is unlikely that they will both be exactly 0.500. However, if that happens, the prediction is made at random.

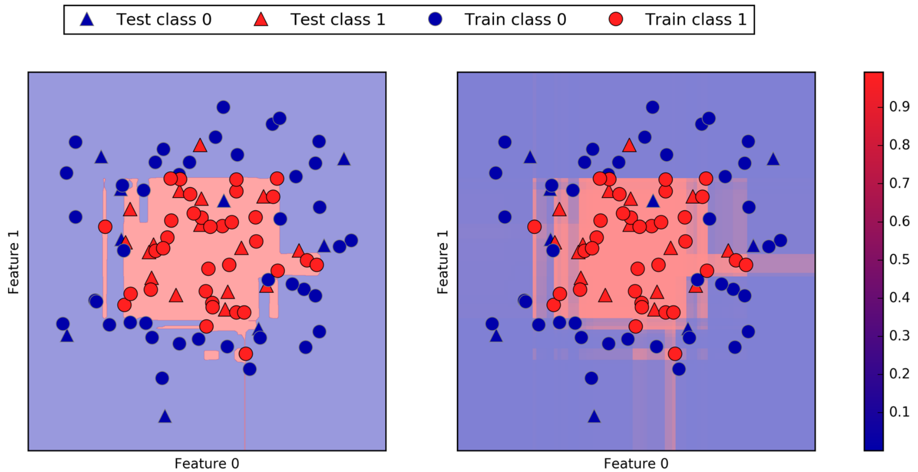

Figure 2-56. Decision boundary (left) and predicted probabilities for the gradient boosting model shown in Figure 2-55

The boundaries in this plot are much more well-defined, and the small areas of uncertainty are clearly visible.

The `scikit-learn` website has a great comparison of many models and what their uncertainty estimates look like. We've reproduced this in Figure 2-57, and we encourage you to go though the example there.

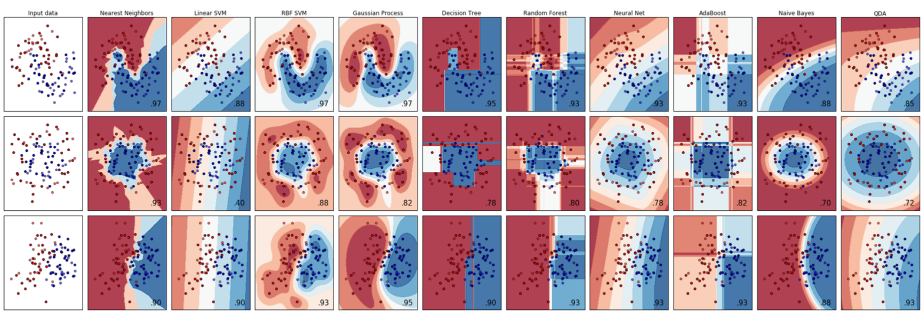

Figure 2-57. Comparison of several classifiers in scikit-learn on synthetic datasets (image courtesy http://scikit-learn.org)

Uncertainty in Multiclass Classification

So far, we've only talked about uncertainty estimates in binary classification. But the `decision_function` and `predict_proba` methods also work in the multiclass setting. Let's apply them on the Iris dataset, which is a three-class classification dataset: