Problem with 'in sample' scores

02 July 2024 12:38

The measure we just computed can be called an "in-sample" score. We used a single
both building the model and evaluating it. Here's why this is bad.

Imagine that, in the large real estate market, door color is unrelated to home price.

However, in the sample of data you used to build the model, all homes with green d
expensive. The model's job is to find patterns that predict home prices, so it will see
always predict high prices for homes with green doors.

Since this pattern was derived from the training data, the model will appear accurat

But if this pattern doesn't hold when the model sees new data, the model would be
used in practice.

Since models' practical value come from making predictions on new data, we measu
that wasn't used to build the model. The most straightforward way to do this is to e
the model-building process, and then use those to test the model's accuracy on data
This data is called **validation data**.

The scikit-learn library has a function `train_test_split` to break up the da
some of that data as training data to fit the model, and we'll use the other data
calculate `mean_absolute_error`.

Here is the code:

linkcode

```
from sklearn.model_selection import train_test_split
# split data into training and validation data, for both features and target
# The split is based on a random number generator. Supplying a numeric value t
```

e "sample" of houses for

doors were very
e this pattern, and it will

ce in the training data.

very inaccurate when

ure performance on data
xclude some data from
a it hasn't seen before.

ata into two pieces. We'll use
as validation data to

In [3]:

```
# The split is based on a random number generator. Supplying a numeric value t
# the random_state argument guarantees we get the same split every time we
# run this script.
train_X, val_X, train_y, val_y = train_test_split(X, y, random_state = 0)
# Define model
melbourne_model = DecisionTreeRegressor()
# Fit model
melbourne_model.fit(train_X, train_y)
# get predicted prices on validation data
val_predictions = melbourne_model.predict(val_X)
print(mean_absolute_error(val_y, val_predictions))
>
```