The first step of capturing patterns from data is called **training** or **fitting** the model.

The data used to **fit** the model is called the **training data**.

After the model has been fit, you apply it to new data to **predict** the outcomes.

The point at the bottom of a **decision tree** where we make a prediction is called a **leaf**.

We start by picking variables to focus on using intuition – later courses will cover statistical techniques to allow us to automatically prioritize variables.

We first select the column we want to predict – called the **prediction target** – and assign it to variable y.

The columns inputted into our model (used for predictions) are called **features**.

Sometimes you will use all columns except the target as features – other times you want to use less.

Here we use the **scikit-learn** library to create our models, (written as sklearn).

The steps to building and using a model are:

* **DEFINE** – what type of model will it be?
* **FIT** – capture patterns from provided data.
* **PREDICT** – what it sounds like.
* **EVALUATE** – determine how accurate the model’s predictions are.

from sklearn.tree import DecisionTreeRegressor

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html>

<https://gdcoder.com/decision-tree-regressor-explained-in-depth/>

# Define the model - specify random state to ensure SAME results each run.

model = DecisionTreeRegressor(random\_state=1)

# Fit (or train) the model.

model.fit(X, y)

**MODEL VALIDATION** – you will want to evaluate every model you ever build.

In most applications the relevant measure of model quality is predictive accuracy.

*Will the model’s predictions be close to what actually happens?*

We will start with a metric for summarizing model quality called **Mean Absolute Error** (**MAE**).

The prediction error for each house is: error=actual-predicted.

With the MAE we take the absolute value for each error (convert to a positive number).

from sklearn.metrics import mean\_absolute\_error

You should not use this check with training data – should be performed on data that did not go into creating the model. This way an inaccurate model will not pass the evaluation.

Models’ practical value come from making predictions on new data – we need to measure performance on data that was not used to build the model.

*Exclude data from the model-building process to test the accuracy of it –* **VALIDATION DATA**.

You can split your data with the sklearn function:

from sklearn.model\_selection import train\_test\_split

And store/run this:

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, random\_state=0)

Then you fit the model with the training data and can run MAE on the test data to compare to the answers you know.