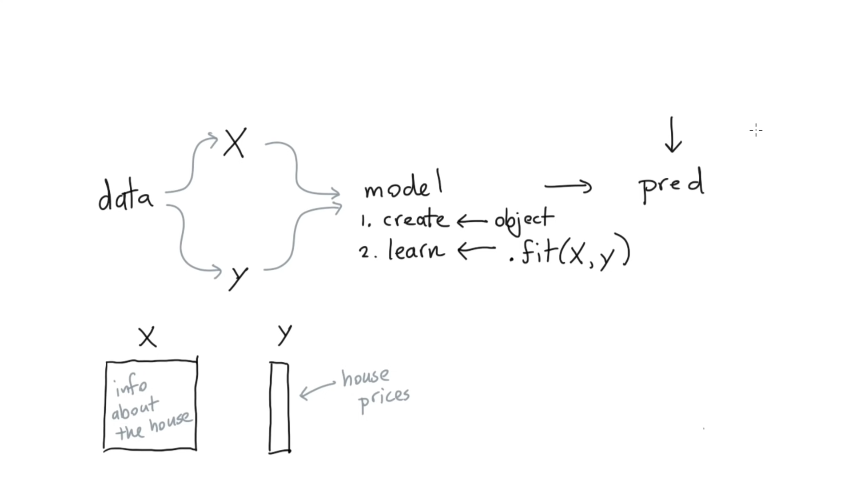
**https://youtu.be/0B5eIE\_1vpU?si=L2\_kBdv5DBMEhraB**

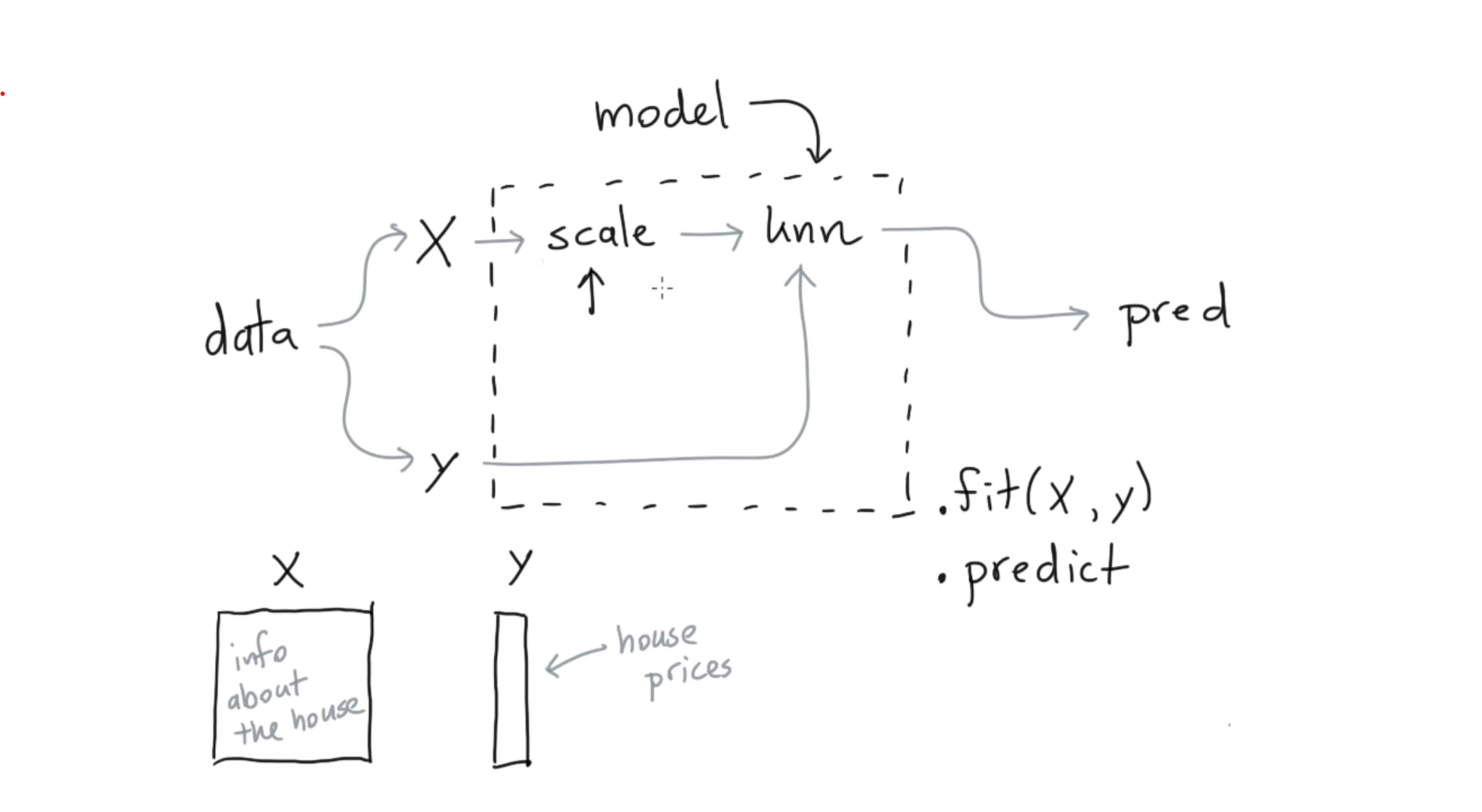
**sklearn.pipeline .Pipeline**

A sequence of data transformers with an optional final predictor.

Pipeline allows you to sequentially apply a list of transformers to preprocess the data and, if desired, conclude the sequence with a final predictor for predictive modeling.



# Redefine ML Model



Pre-prossing step

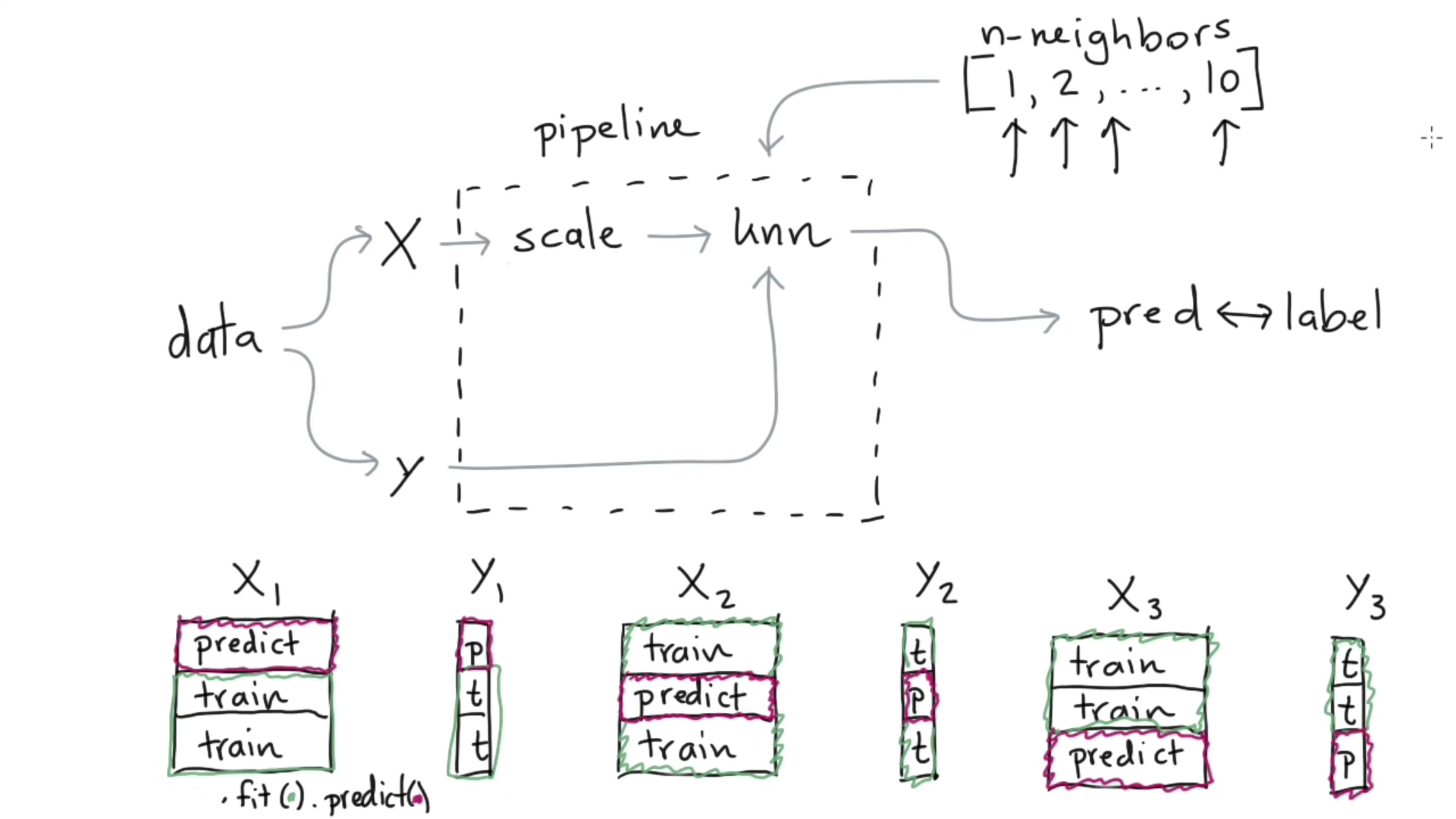
Standard effects the model

a

KNN

Redefine

# 2nd time redefining ML Model



Trick with dataset

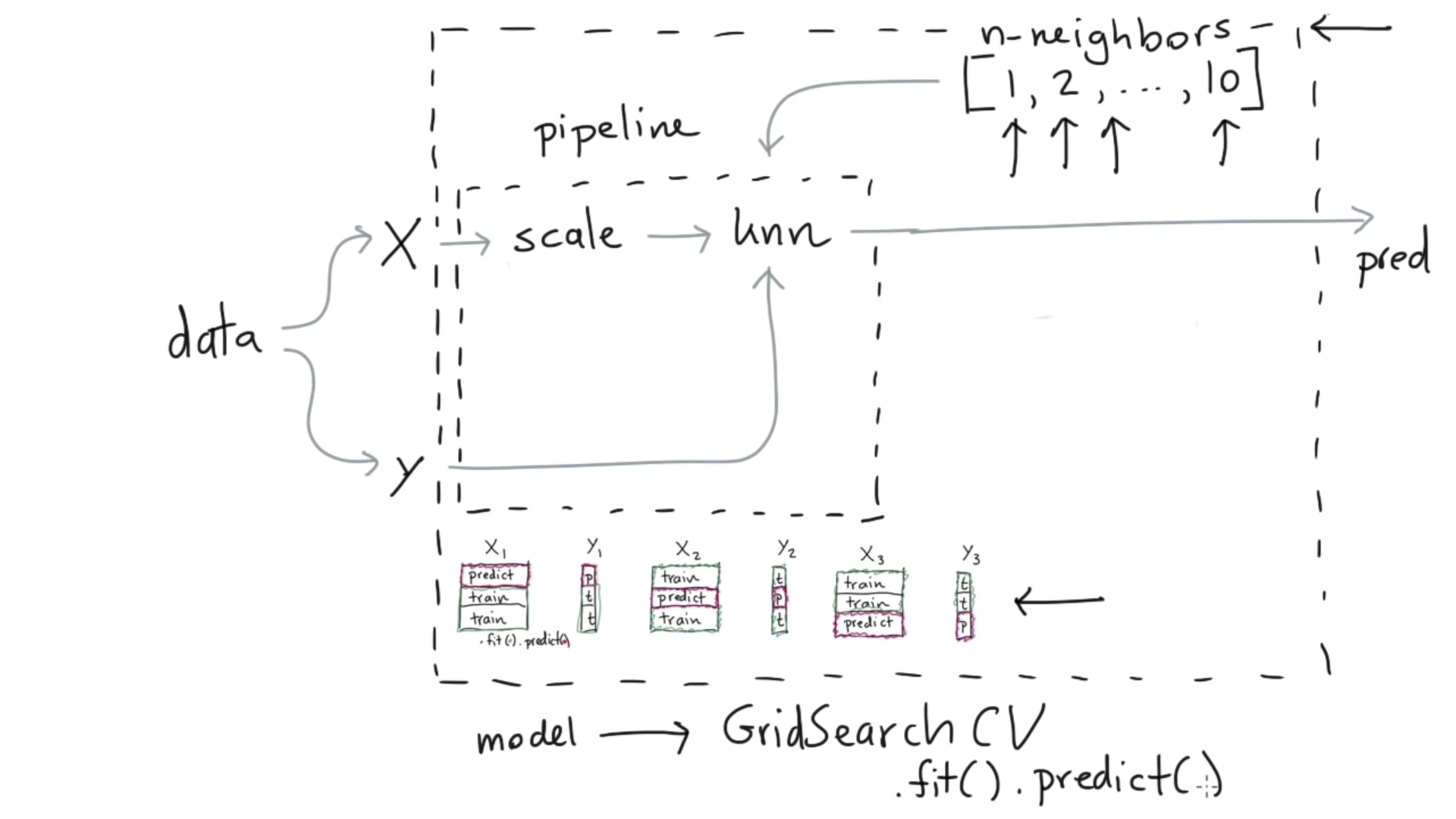
CROSS VALIDATION

PROCEDURE



In KNN neighbors can also define the predictions

# Final PIPELINE



Cross Validation

OBJECT

**import** matplotlib.pylab **as** plt

**from** sklearn.neighbors **import** KNeighborsRegressor

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.datasets **import** load\_boston #Dataset

**from** sklearn.pipeline **import** Pipeline

X, y **=** load\_boston(return\_X\_y**=True**)

pipe **=** Pipeline([

("scale", StandardScaler()),

("model", KNeighborsRegressor())

])

pred **=** pipe**.**fit(X, y)**.**predict(X)

plt**.**scatter(pred, y)

#pipe.get\_params() #check what parameters we can change

# methodology

**from** sklearn.model\_selection **import** GridSearchCV

modl **=** GridSearchCV(estimator**=**pipe,

param\_grid**=**{

'model\_\_n\_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

},

cv**=**3)

modl**.**fit(X, y);

pd**.**DataFrame(modl**.**cv\_results\_) # for every perimeter it tracks the model