Lecture 15
Model Selection and Hyperparameter Tuning

Dennis Sun
Stanford University
DATASCI / STATS 112

February 15, 2023

1. Review

2. Model Selection and Hyperparameter Tuning

3. A Word About Assignment 4

1 Review

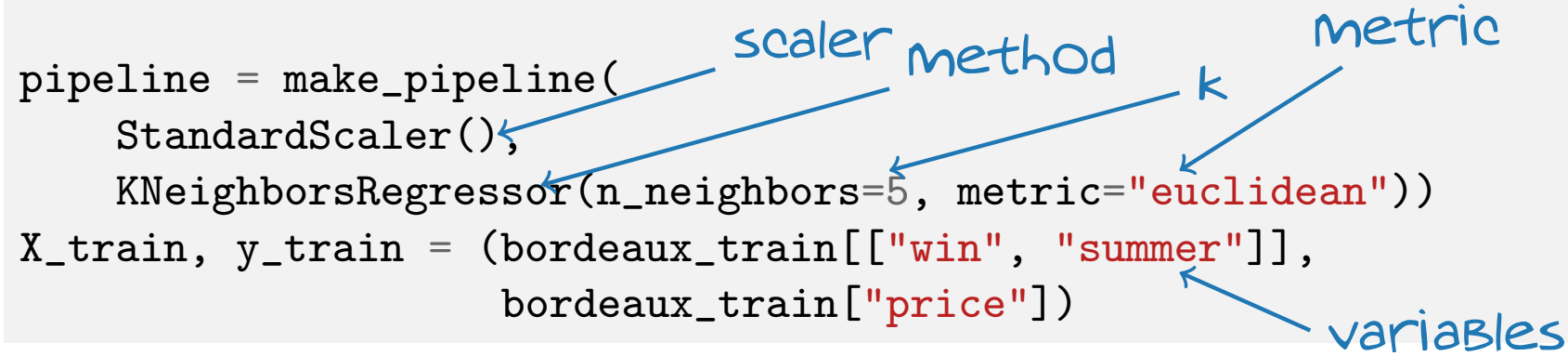
2 Model Selection and Hyperparameter Tuning

3 A Word About Assignment 4

Here's a machine learning model.

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor

pipeline = make_pipeline(
    StandardScaler(),
    KNeighborsRegressor(n_neighbors=5, metric="euclidean"))
X_train, y_train = (bordeaux_train[["win", "summer"]],
                    bordeaux_train["price"])
```



The right way to evaluate machine learning models is *test error*, which is estimated using cross-validation.

```
from sklearn.model_selection import cross_val_score
cross_val_score(
    pipeline,
    X=X_train, y=y_train,
    scoring="neg_mean_squared_error",
    cv=4).mean()
```

```
-375.27166666666665
```

How do we choose between all the options (scaler, k, etc.)?

1 Review

2 Model Selection and Hyperparameter Tuning

3 A Word About Assignment 4

Two Related Problems

Model Selection refers to the choice of:

- which input features to include (e.g., winter rainfall, summer temperature)
- what preprocessing to do (e.g., scaler)
- what machine learning method to use (e.g., k-nearest neighbors)

Hyperparameter Tuning refers to the choice of parameters in the machine learning method.
For k-nearest neighbors, hyperparameters include:

- k
- metric (e.g., Euclidean distance)

The distinction isn't important. For all choices, we use cross-validation and pick the model / hyperparameter with the smallest test error.

Example of Model Selection

Which input features should we include?

- winter rain, summer temp
- winter rain, summer temp, harvest rain
- winter rain, summer temp, harvest rain, Sept. temp

```
pipeline = make_pipeline(
    StandardScaler(),
    KNeighborsRegressor(n_neighbors=5, metric="euclidean"))
for features in [["win", "summer"],
                 ["win", "summer", "har"],
                 ["win", "summer", "har", "sep"]]:
  X_train, y_train = (bordeaux_train[features],
                      bordeaux_train["price"])
  print(features, -cross_val_score(
      pipeline, X_train, y_train,
      scoring="neg_mean_squared_error", cv=4).mean())
```

```
['win', 'summer'] 375.27166666666665
['win', 'summer', 'har'] 363.04047619047617
['win', 'summer', 'har', 'sep'] 402.4507142857142
```

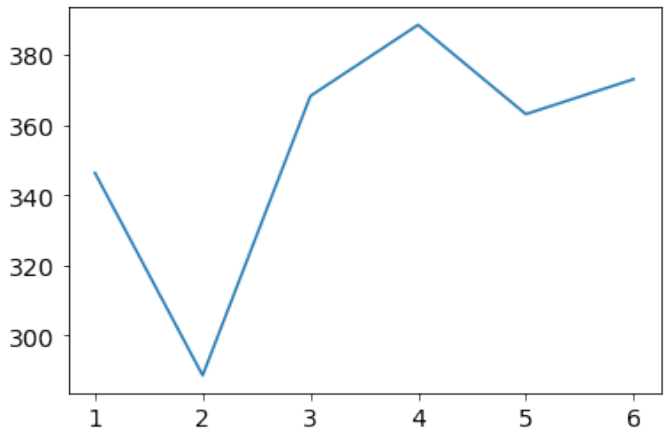
✓

Example of Hyperparameter Tuning

What is the best value of k?

```
X_train, y_train = (bordeaux_train[["win", "summer", "har"]],
                    bordeaux_train["price"])
ks, test_mses = range(1, 7), []
for k in ks:
  pipeline = make_pipeline(
      StandardScaler(),
      KNeighborsRegressor(n_neighbors=k, metric="euclidean"))
  test_mses.append(-cross_val_score(
      pipeline, X_train, y_train,
      scoring="neg_mean_squared_error", cv=4).mean())
```

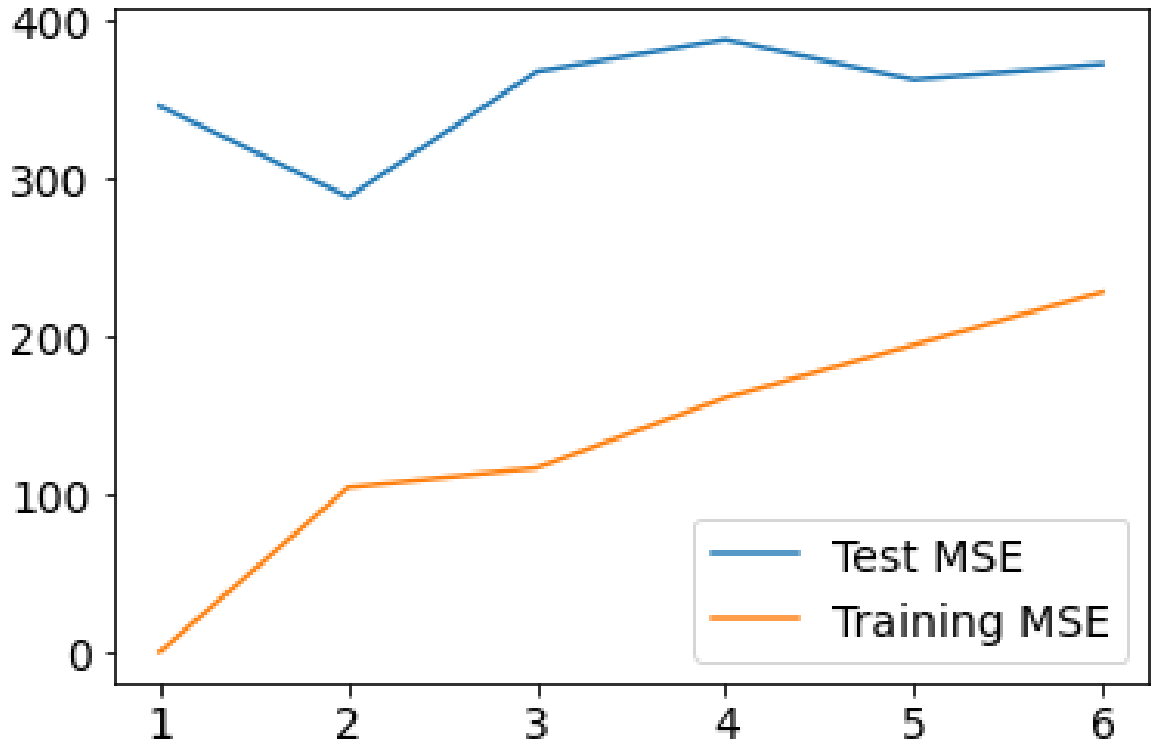
```
pd.Series(test_mses, index=ks).plot.line()
```



The best value of k is 2.

Training vs. Test Error

Here are the training and test MSEs on the same graph.



Notice that training MSE only goes down as we decrease k.

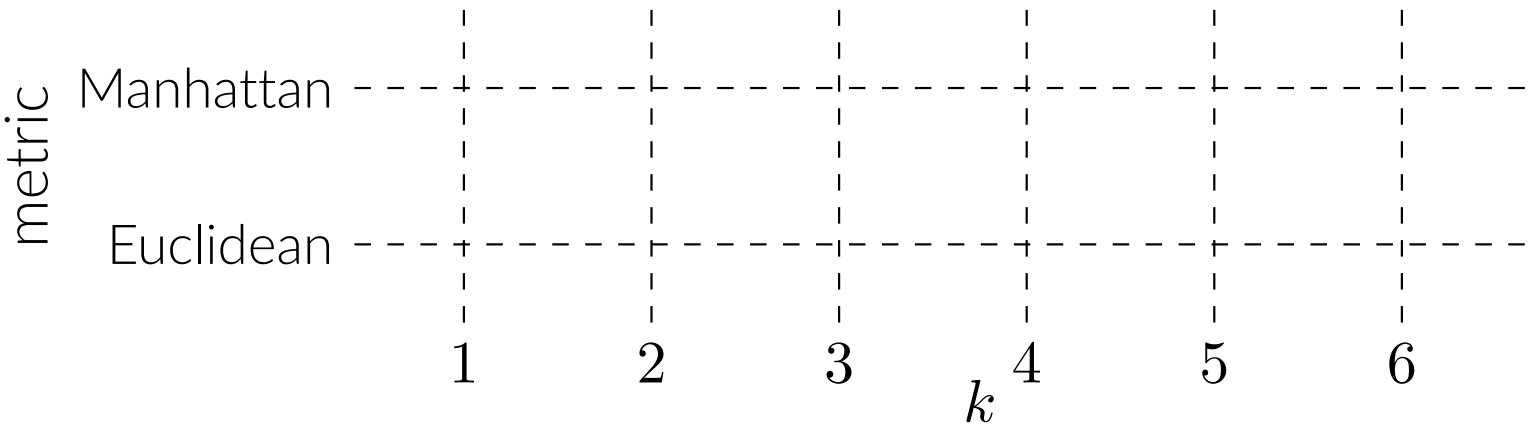
If we optimize for training MSE, then we will pick $k = 1$, but this has worse test MSE.

In other words, the $k = 1$ model is **overfitted** to the training data.

Hyperparameter Tuning Using `GridSearchCV`

Suppose we want to choose k and the metric (Euclidean or Manhattan distance).

We need to try all 12 combinations on the following grid:



Scikit-Learn's **`GridSearchCV`** automates the creation of a grid with all combinations.

Hyperparameter Tuning Using `GridSearchCV`

```
from sklearn.model_selection import GridSearchCV

grid_cv = GridSearchCV(
    pipeline,
    param_grid={
        "kneighborsregressor__n_neighbors": range(1, 7),
        "kneighborsregressor__metric": ["euclidean", "manhattan"],
    },
    scoring="neg_mean_squared_error", cv=4)

grid_cv.fit(X_train, y_train)
grid_cv.best_params_
```

```
{'kneighborsregressor__metric': 'euclidean',
 'kneighborsregressor__n_neighbors': 2}
```

So the model from before was the best.

Where did `"kneighborsregressor"` in `param_grid` come from?

```
pipeline
```

```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('kneighborsregressor',
                 KNeighborsRegressor(metric='euclidean'))])
```

1 Review

2 Model Selection and Hyperparameter Tuning

3 A Word About Assignment 4

Assignment 4

- You have all the tools to complete Questions 1-3 on Assignment 4. It is due *next Monday*.
  (You even have examples from lecture that you can copy!)
- Question 4 is open-ended: try models of your own and submit to the Kaggle competition!
  - You are required to try several models and show your work in the notebook.
  - You must submit at least two of these models to Kaggle to earn full credit.
  - You won't win with k-nearest neighbors, so try out other methods from the Scikit-Learn documentation.
  - The best way to become a machine learning expert is to learn by doing!
- I'm going to up the stakes: in addition to extra credit (and bragging rights), the competition winner(s) win a gift card to Coho.