

CS 457 Homework 12: Model Validation

for Mon Oct 5 2026

1. Time for some polynomial regression! (We'll talk about the `PolynomialFeatures` function in class on Monday.)

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split

def PolynomialRegression(degree=1):
    return make_pipeline(PolynomialFeatures(degree),
                          LinearRegression())

def make_data(N, err=0.3, rseed=None):
    """generate some data to fit"""
    rng = np.random.RandomState(rseed)
    X = rng.rand(N, 1) * 2.5
    x = X.ravel()
    y_true = x**3 - 3*x**2 + 2*x
    y = y_true + (err * rng.randn(N))
    return X, y

X, y = make_data(40)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)
plt.scatter(X_train.ravel(), y_train, color='black')
axis = plt.axis()
for degree in [2, 3, 6, 10]:
    model = PolynomialRegression(degree).fit(X_train, y_train)
    X_draw_curve = np.linspace(-0.1, 2.6, 500)[: , None]
    y_draw_curve = model.predict(X_draw_curve)
    plt.plot(X_draw_curve.ravel(), y_draw_curve, label='degree={0}'.format(degree))
    print("polynomial of degree", degree)
    print("Training set r^2: {:.2f}".format(model.score(X_train, y_train)))
    print("    Test set r^2: {:.2f}".format(model.score(X_test, y_test)))
plt.ylim(-1.1, 2)
plt.legend()
```

What's going on here? Write up your findings (include the plot).

2. To get serious about model validation, let's use the built-in `validation_curve` function.

```
from sklearn.model_selection import validation_curve
degree = np.arange(0, 16)
train_score, val_score = validation_curve(
    PolynomialRegression(), X, y,
    param_name='polynomialfeatures__degree',
    param_range=degree, cv=7)

plt.plot(degree, np.median(train_score, 1),
         label='training score')
plt.plot(degree, np.median(val_score, 1),
         label='validation score')
plt.legend(loc='best')
```

```
plt.ylim(0, 1)
plt.xlabel('degree')
plt.ylabel('score');
```

What's going on here? Write up your findings (include the plot).

3. Now grab the built-in seaborn `mpg` dataset and see how far you can get building a model that predicts `mpg` from one or two input features. (Hopefully the example code that we've used in the assignments so far can provide some inspiration!) No need to do a validation curve, but do plot the data and your model fit.

```
import seaborn as sns
m = sns.load_dataset("mpg")
X = m[["weight"]]
y = m["mpg"]
```

Write up your findings!

Email your write-up to Jack, CC'ing Wheeler.