**🎯 Purpose**

This code is performing **polynomial regression** and evaluating models of **different polynomial degrees (1 through 5)** to see which degree best fits the data without overfitting.

**🧠 Key Libraries Used**

* PolynomialFeatures — expands features into polynomial combinations (e.g., ( x, x^2, x^3, ... )).
* LinearRegression — fits a linear model to those polynomial-transformed features.
* cross\_val\_score — performs cross-validation to estimate model performance.
* mean\_squared\_error — measures prediction error.
* score() — gives the ( R^2 ) (coefficient of determination) score.

**🔍 Code Breakdown**

degrees = range(1, 6)

train\_mean\_scores = []

test\_scores = []

error\_train = []

error\_test = []

You’re setting up empty lists to store evaluation metrics for **each polynomial degree**.

**💡 Looping through polynomial degrees**

for d in degrees:

X\_train\_poly = PolynomialFeatures(d).fit\_transform(X\_train)

X\_test\_poly = PolynomialFeatures(d).fit\_transform(X\_test)

model = LinearRegression()

* Transforms X\_train and X\_test into polynomial form.
* For example, if ( d = 3 ), and X = [x], this becomes [1, x, x², x³].

**⚙️ Cross-validation**

train\_scores = cross\_val\_score(model, X\_train\_poly, y\_train)

train\_score = train\_scores.mean()

train\_mean\_scores.append(train\_score)

* Performs k-fold cross-validation (default = 5 folds) on the training data.
* The mean of these scores estimates **training performance** stability.

**🧮 Fit model and compute errors**

model.fit(X\_train\_poly, y\_train)

mse\_train = mean\_squared\_error(model.predict(X\_train\_poly), y\_train)

error\_train.append(mse\_train)

test\_score = model.score(X\_test\_poly, y\_test)

mse\_test = mean\_squared\_error(model.predict(X\_test\_poly), y\_test)

error\_train.append(mse\_test) # ⚠️ likely a bug (should be error\_test.append)

test\_scores.append(test\_score)

* Fits the polynomial regression model on the transformed training data.
* Calculates **Mean Squared Error (MSE)** on both training and test sets.
* model.score() gives ( R^2 ) on the test set.
* ⚠️ **Bug detected:**  
  You append mse\_test to error\_train, not error\_test.  
  ✅ Should be:
* error\_test.append(mse\_test)

**📊 Print results**

print(f"Degree {d}: Train CV Mean R^2 Score = {train\_score:.3f}, Test R^2 Score = {test\_score:.3f}, Train MSE = {mse\_train:.3f}, Test MSE = {mse\_test:.3f}")

This displays the metrics for each degree, helping you see:

* Whether the model overfits (high train R², low test R², large test MSE).
* Which polynomial degree gives the best generalization.

**✅ Fixed and Clean Version**

Here’s a corrected, clear version:

for d in degrees:

X\_train\_poly = PolynomialFeatures(d).fit\_transform(X\_train)

X\_test\_poly = PolynomialFeatures(d).fit\_transform(X\_test)

model = LinearRegression()

train\_scores = cross\_val\_score(model, X\_train\_poly, y\_train)

train\_mean\_scores.append(train\_scores.mean())

model.fit(X\_train\_poly, y\_train)

mse\_train = mean\_squared\_error(y\_train, model.predict(X\_train\_poly))

mse\_test = mean\_squared\_error(y\_test, model.predict(X\_test\_poly))

error\_train.append(mse\_train)

error\_test.append(mse\_test)

test\_score = model.score(X\_test\_poly, y\_test)

test\_scores.append(test\_score)

print(f"Degree {d}: "

f"Train CV Mean R² = {train\_scores.mean():.3f}, "

f"Test R² = {test\_score:.3f}, "

f"Train MSE = {mse\_train:.3f}, "

f"Test MSE = {mse\_test:.3f}")