Awesome! Let’s go step-by-step through each **linear regression metric**, with:

1. 📐 **Formula**
2. 🔍 **Explanation of the formula**
3. 💻 **Python (scikit-learn) syntax** for implementation

**✅ 1. Mean Absolute Error (MAE)**

**📐 Formula:**

MAE=1n∑i=1n∣yi−y^i∣\text{MAE} = \frac{1}{n} \sum\_{i=1}^{n} \left| y\_i - \hat{y}\_i \right|

**🔍 Explanation:**

* yiy\_i: actual value
* y^i\hat{y}\_i: predicted value
* nn: number of samples
* You calculate the **absolute difference** between actual and predicted values for each point, and take the average.

💡 **Gives average error in original units** (e.g., dollars if predicting price).

**💻 ML Code:**

from sklearn.metrics import mean\_absolute\_error

mae = mean\_absolute\_error(y\_true, y\_pred)

**✅ 2. Mean Squared Error (MSE)**

**📐 Formula:**

MSE=1n∑i=1n(yi−y^i)2\text{MSE} = \frac{1}{n} \sum\_{i=1}^{n} \left( y\_i - \hat{y}\_i \right)^2

**🔍 Explanation:**

* Same as MAE, but you **square the error** before averaging.
* **Squaring** penalizes larger errors more heavily than small ones.

💡 **Useful when large errors are more serious than small ones**.

**💻 ML Code:**

from sklearn.metrics import mean\_squared\_error

mse = mean\_squared\_error(y\_true, y\_pred)

**✅ 3. Root Mean Squared Error (RMSE)**

**📐 Formula:**

RMSE=1n∑i=1n(yi−y^i)2\text{RMSE} = \sqrt{ \frac{1}{n} \sum\_{i=1}^{n} (y\_i - \hat{y}\_i)^2 }

**🔍 Explanation:**

* RMSE is just the **square root of MSE**.
* Keeps units same as target (like MAE) but **penalizes large errors**.

**💻 ML Code:**

import numpy as np

rmse = np.sqrt(mean\_squared\_error(y\_true, y\_pred))

**✅ 4. R-squared (R² Score)**

**📐 Formula:**

R2=1−∑(yi−y^i)2∑(yi−yˉ)2R^2 = 1 - \frac{\sum (y\_i - \hat{y}\_i)^2}{\sum (y\_i - \bar{y})^2}

**🔍 Explanation:**

* Numerator: total error made by model
* Denominator: total variance in actual data
* yˉ\bar{y}: mean of actual values
* R² = 1 → perfect prediction; R² = 0 → model does no better than mean
* Measures **how much of the variance in y is explained** by the model.

**💻 ML Code:**

from sklearn.metrics import r2\_score

r2 = r2\_score(y\_true, y\_pred)

**✅ 5. Adjusted R-squared**

**📐 Formula:**

Adjusted R2=1−((1−R2)(n−1)n−p−1)\text{Adjusted } R^2 = 1 - \left( \frac{(1 - R^2)(n - 1)}{n - p - 1} \right)

**🔍 Explanation:**

* nn: number of samples
* pp: number of features
* Penalizes model for adding **too many features**.
* If a new feature doesn't improve the model, Adjusted R² will **decrease**.

**💻 ML Code:**

Scikit-learn doesn’t provide it directly, so calculate manually:

n = len(y\_true)

p = X\_test.shape[1] # number of features

adj\_r2 = 1 - ((1 - r2\_score(y\_true, y\_pred)) \* (n - 1)) / (n - p - 1)

**✅ 6. Mean Absolute Percentage Error (MAPE)**

**📐 Formula:**

MAPE=100%n∑i=1n∣yi−y^iyi∣\text{MAPE} = \frac{100\%}{n} \sum\_{i=1}^{n} \left| \frac{y\_i - \hat{y}\_i}{y\_i} \right|

**🔍 Explanation:**

* Shows error **as a percentage** of the true value.
* Good for **business/finance**, where relative error matters more than raw error.
* Not defined if any yi=0y\_i = 0.

**💻 ML Code:**

from sklearn.metrics import mean\_absolute\_percentage\_error

mape = mean\_absolute\_percentage\_error(y\_true, y\_pred)

**✅ 7. Residuals & Residual Analysis**

**📐 Formula:**

Residual=yi−y^i\text{Residual} = y\_i - \hat{y}\_i

**🔍 Explanation:**

* Just the raw error for each prediction.
* Use these to plot residual graphs and check for **patterns**, **bias**, or **non-linearity**.

**💻 ML Code:**

residuals = y\_true - y\_pred

**🧪 Example ML Flow (Quick View)**

from sklearn.linear\_model import LinearRegression

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

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

# Prepare your data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

# Train model

model = LinearRegression()

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

# Predict

y\_pred = model.predict(X\_test)

# Evaluate

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

Would you like a full working example with data and visualizations next?