R2 Squared Error

The R-squared (R^2) error, also known as the coefficient of determination, is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model.

In simpler terms, R^2 indicates how well the data fit the regression model.

Interpretation of R-squared

R^2 = 1: The model explains all the variability of the response data around its mean. This indicates a perfect fit.

R^2 = 0: The model does not explain any of the variability of the response data around its mean. This means the model fails to capture any of the patterns in the data.

0 < R^2 < 1: The model explains a portion of the variability, where higher values indicate a better fit.

Calculation of R-squared

The R-squared value is calculated as:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Where:

- SS_{res} (Residual Sum of Squares): The sum of the squared differences between the observed values and the predicted values.
- SS_{tot} (Total Sum of Squares): The sum of the squared differences between the observed values and the mean of the observed values.

Residual Sum of Squares (SS_res)

$$SS_{res} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Where:

- y_i is the actual value.
- $\hat{y}_i$ is the predicted value by the model.

Total Sum of Squares (SS_tot)

$$SS_{tot} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

Where:

- y_i is the actual value.
- $\bar{y}$ is the mean of the actual values.

Example Calculation

Let's assume we have a dataset with actual values y and predicted values y'.

We'll calculate the R-squared value for this dataset.

In [6]:
```
import numpy as np

# Actual values
y = np.array([3, -0.5, 2, 7])

# Predicted values
y_pred = np.array([2.5, 0.0, 2, 8])

# Calculate the mean of actual values
y_mean = np.mean(y)

# Calculate SS_res and SS_tot
ss_res = np.sum((y - y_pred) ** 2)
ss_tot = np.sum((y - y_mean) ** 2)

# Calculate R-squared
r_squared = 1 - (ss_res / ss_tot)
print(f'R-squared: {r_squared}')
```

```
R-squared: 0.9486081370449679
```

Using Scikit-Learn to Calculate R-squared

In practice, you can use scikit-learn to easily calculate the R-squared value for a regression model.

In [5]:
```
from sklearn.metrics import r2_score

# Calculate R-squared using scikit-learn
r_squared = r2_score(y, y_pred)
print(f'R-squared (scikit-learn): {r_squared}')
```

```
R-squared (scikit-learn): 1.0
```

Example with Polynomial Regression

Here's how you would calculate and interpret R-squared in the context of polynomial regression:

In [4]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score


# Sample data
x = [1,2,3,4,5]
y = [1, 4, 9, 16, 25]

x=pd.DataFrame(x)
y=pd.Series(y)

# Transform to polynomial features
poly = PolynomialFeatures(degree=2)
x_poly = poly.fit_transform(x)

# Fit the model
model = LinearRegression()
model.fit(x_poly, y)

# Predict
y_pred = model.predict(x_poly)

# Calculate R-squared
r_squared = r2_score(y, y_pred)

print(f'R-squared: {r_squared}')
```

```
R-squared: 1.0
```

In this example, we would expect a high R-squared value because the data perfectly follows a quadratic pattern, which our polynomial regression model of degree 2 should capture very well.

In [3]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Sample data
```

```
x = [1,2,3,4,5]
y = [1, 8, 27, 64, 125]

x=pd.DataFrame(x)
y=pd.Series(y)

# Transform to polynomial features
poly = PolynomialFeatures(degree=2)
x_poly = poly.fit_transform(x)

# Fit the model
model = LinearRegression()
model.fit(x_poly, y)

# Predict
y_pred = model.predict(x_poly)

# Calculate R-squared
r_squared = r2_score(y, y_pred)
print(f'R-squared Error: {r_squared}')
```

```
R-squared Error: 0.998614051973051
```

In [ ]: