Tugas Data Science

Nama : Mochammad Qussay Alhindi Achmadi  
NIM : 1103213087

1. Perhitungan error pada regresi
2. SMAPE - Symmetric Mean Absolute Percentage Error

which is an accuracy measure commonly used in forecasting and time series analysis.

Given the actual values y and the predicted values y\_hat, the SMAPE is calculated as the average of the absolute percentage errors between the two, where each error is weighted by the sum of the absolute values of the actual and predicted values.

The resulting score ranges between 0 and 1, where a score of 0 indicates a perfect match between the actual and predicted values, and a score of 1 indicates no match at all. A smaller value of SMAPE is better, and it is often multiplied by 100% to obtain the percentage error. Best possible score is 0.0, smaller value is better. Range = [0, 1].

**from** **numpy** **import** array

**from** **permetrics.regression** **import** RegressionMetric

*## For 1-D array*

y\_true = array([3, -0.5, 2, 7])

y\_pred = array([2.5, 0.0, 2, 8])

evaluator = RegressionMetric(y\_true, y\_pred)

print(evaluator.symmetric\_mean\_absolute\_percentage\_error())

*## For > 1-D array*

y\_true = array([[0.5, 1], [-1, 1], [7, -6]])

y\_pred = array([[0, 2], [-1, 2], [8, -5]])

evaluator = RegressionMetric(y\_true, y\_pred)

print(evaluator.SMAPE(multi\_output="raw\_values"))

1. NSE - Nash-Sutcliffe Efficiency

is calculated as the ratio of the mean squared error between the observed and simulated streamflow to the variance of the observed streamflow. The NSE ranges between -inf and 1, with a value of 1 indicating perfect agreement between the observed and simulated streamflow

**from** **numpy** **import** array

**from** **permetrics.regression** **import** RegressionMetric

*## For 1-D array*

y\_true = array([3, -0.5, 2, 7])

y\_pred = array([2.5, 0.0, 2, 8])

evaluator = RegressionMetric(y\_true, y\_pred)

print(evaluator.nash\_sutcliffe\_efficiency())

*## For > 1-D array*

y\_true = array([[0.5, 1], [-1, 1], [7, -6]])

y\_pred = array([[0, 2], [-1, 2], [8, -5]])

evaluator = RegressionMetric(y\_true, y\_pred)

print(evaluator.NSE(multi\_output="raw\_values"))

1. MASE - Mean Absolute Scaled Error

* Best possible score is 0.0, smaller value is better. Range = [0, +inf)
* m = 1 for non-seasonal data, m > 1 for seasonal data

**from** **numpy** **import** array

**from** **permetrics.regression** **import** RegressionMetric

*## For 1-D array*

y\_true = array([3, -0.5, 2, 7])

y\_pred = array([2.5, 0.0, 2, 8])

evaluator = RegressionMetric(y\_true, y\_pred)

print(evaluator.mean\_absolute\_scaled\_error())

*## For > 1-D array*

y\_true = array([[0.5, 1], [-1, 1], [7, -6]])

y\_pred = array([[0, 2], [-1, 2], [8, -5]])

evaluator = RegressionMetric(y\_true, y\_pred)

print(evaluator.MASE(multi\_output="raw\_values"))

1. MSLE - Mean Squared Logarithmic Error

Where means the natural logarithm of x. This metric is best to use when targets having exponential growth, such as population counts, average sales of a commodity over a span of years etc. Note that this metric penalizes an under-predicted estimate greater than an over-predicted estimate.

The Mean Squared Logarithmic Error (MSLE) [5] is a statistical measure used to evaluate the accuracy of a forecasting model, particularly when the data has a wide range of values. It measures the average of the squared differences between the logarithms of the predicted and actual values.

The logarithmic transformation used in the MSLE reduces the impact of large differences between the actual and predicted values and provides a better measure of the relative errors between the two values. The MSLE is always a positive value, with a smaller MSLE indicating better forecast accuracy.

The MSLE is commonly used in applications such as demand forecasting, stock price prediction, and sales forecasting, where the data has a wide range of

values and the relative errors are more important than the absolute errors. + It is important to note that the MSLE is not suitable for data with negative values or zero values, as the logarithm function is not defined for these values. + Best possible score is 0.0, smaller value is better. Range = [0, +inf)

**from** **numpy** **import** array

**from** **permetrics.regression** **import** RegressionMetric

*## For 1-D array*

y\_true = array([3, -0.5, 2, 7])

y\_pred = array([2.5, 0.0, 2, 8])

evaluator = RegressionMetric(y\_true, y\_pred)

print(evaluator.mean\_squared\_log\_error())

*## For > 1-D array*

y\_true = array([[0.5, 1], [-1, 1], [7, -6], [1, 2]])

y\_pred = array([[0, 2], [-1, 2], [8, -5], [1.1, 1.9]])

evaluator = RegressionMetric(y\_true, y\_pred)

print(evaluator.MSLE(multi\_output="raw\_values"))

1. SE - Squared Error

* Best possible score is 0.0, smaller value is better. Range = [0, +inf)
* Note: Computes the squared error between two numbers, or for element between a pair of list, tuple or numpy arrays.
* The Squared Error (SE) is a metric used to evaluate the accuracy of a regression model by measuring the average of the squared differences between the

**from** **numpy** **import** array

**from** **permetrics.regression** **import** RegressionMetric

*## For 1-D array*

y\_true = array([3, -0.5, 2, 7])

y\_pred = array([2.5, 0.0, 2, 8])

evaluator = RegressionMetric(y\_true, y\_pred)

print(evaluator.single\_squared\_error())

*## For > 1-D array*

y\_true = array([[0.5, 1], [-1, 1], [7, -6]])

y\_pred = array([[0, 2], [-1, 2], [8, -5]])

evaluator = RegressionMetric(y\_true, y\_pred)

print(evaluator.SE())