Implementing Dataset Enhancements on the THREDDS Data Server

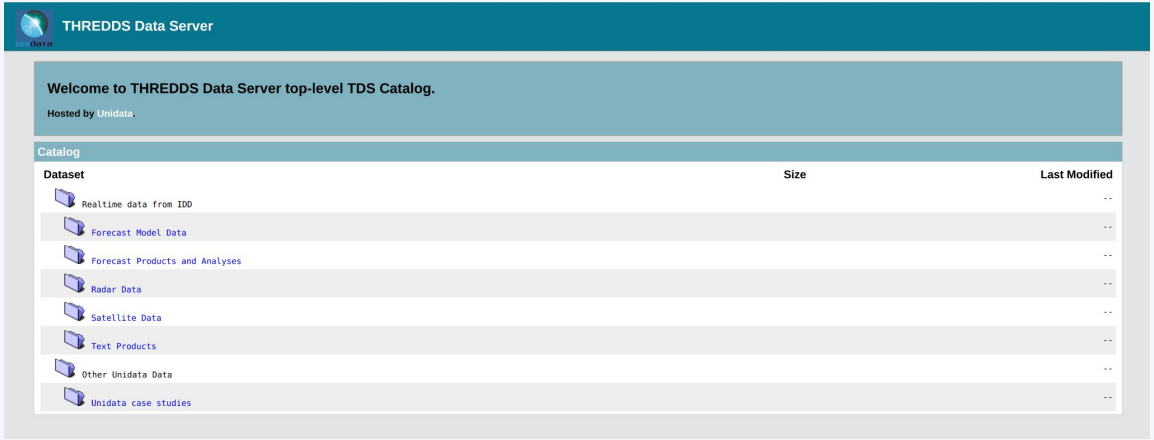
Unidata 2023 Summer Internship

Jessica Souza

Mentor: Tara Drwenski

Background

THREDDS Data Server[1] is a web server that provides catalog, metadata and data access for real-time and archived datasets of environmental data sources at a number of distributed server sites, using a variety of remote data access protocols.



Motivation

There is an expressive use of machine learning methods in earth sciences research.

AI / ML targeted dataset enhancements

Data preprocessing steps in machine learning generally involves cleaning, **rescaling** and splitting the data.

The goal of rescaling is to transform features to be on a similar range.

This improves the performance and training stability of the model.

Standardizer and Normalizer

Motivation

Standardization **(Z- score, Standard Scaler)**

$$s = \frac{z - \mu}{\sigma}$$

z	Data point
μ	Mean value in the variable
σ	Standard deviation value in the variable
s	Standardized data point

Obtain a random variable s with mean 0 and standard deviation 1.

Motivation

Normalization **(Min-Max Scalar)**

$$n = \frac{z - z_{\min}}{z_{\max} - z_{\min}}$$

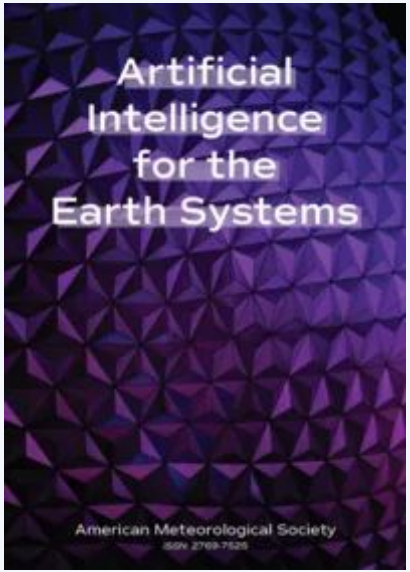
z	Data point
n	Normalized data point

Obtain values between 0 and 1.

Sensitive to outliers.

Motivation

AMS Journal **Artificial Intelligence for the Earth Systems²**



Standardization

Papers

1. Yu et al., 2022
2. White et al., 2022
3. Mamalakis et al., 2022
4. Liu et al 2022
5. Li et al., 2022
6. Miralleset al., 2022
7. Galea et al., 2023
8. Straaten et al., 2023
9. Fulton et al., 2023
10. Connolly et al., 2023

ML

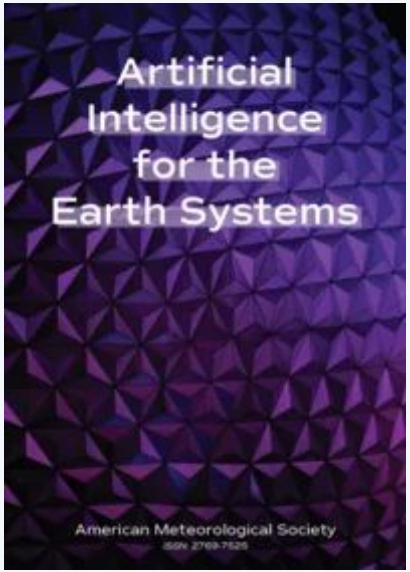
- ➔ Regression model
- ➔ Neural Network
- ➔ Generative Adversarial Network

Dataset

- ➔ GOES16 (ABI channels)
- ➔ Suomi NPP (VIIRS channels)
- ➔ Satellite Altimeter
- ➔ ERA-interim Reanalysis

Motivation

AMS Journal **Artificial Intelligence for the Earth Systems²**



Normalization

Papers		ML		Dataset
1. Cheung et al., 2022	→	Neural Network	→	GEFS
2. Osborne et al., 2022			→	Radar
3. Chen et al., 2023			→	Met Office Forecast

Python Scikit-learn library[3]

`sklearn.preprocessing`: Preprocessing and Normalization

The `sklearn.preprocessing` module includes scaling, centering, normalization, binarization methods.

User guide: See the Preprocessing data section for further details.

`preprocessing.Binarizer`(*[, threshold, copy])	Binarize data (set feature values to 0 or 1) according to a threshold.
`preprocessing.FunctionTransformer`([func, ...])	Constructs a transformer from an arbitrary callable.
`preprocessing.KBinsDiscretizer`([n_bins, ...])	Bin continuous data into intervals.
`preprocessing.KernelCenterer`()	Center an arbitrary kernel matrix K.
`preprocessing.LabelBinarizer`(*[, neg_label, ...])	Binarize labels in a one-vs-all fashion.
`preprocessing.LabelEncoder`()	Encode target labels with value between 0 and n_classes-1.
`preprocessing.MultiLabelBinarizer`(*[, ...])	Transform between iterable of iterables and a multilabel format.
`preprocessing.MaxAbsScaler`(*[, copy])	Scale each feature by its maximum absolute value.
`preprocessing.MinMaxScaler`([feature_range, ...])	Transform features by scaling each feature to a given range.
`preprocessing.Normalizer`([norm, copy])	Normalize samples individually to unit norm.
`preprocessing.OneHotEncoder`(*[, categories, ...])	Encode categorical features as a one-hot numeric array.
`preprocessing.OrdinalEncoder`(*[, ...])	Encode categorical features as an integer array.
`preprocessing.PolynomialFeatures`([degree, ...])	Generate polynomial and interaction features.
`preprocessing.PowerTransformer`([method, ...])	Apply a power transform featurewise to make data more Gaussian-like.
`preprocessing.QuantileTransformer`(*[, ...])	Transform features using quantiles information.
`preprocessing.RobustScaler`(*[, ...])	Scale features using statistics that are robust to outliers.
`preprocessing.SplineTransformer`([n_knots, ...])	Generate univariate B-spline bases for features.
`preprocessing.StandardScaler`(*[, copy, ...])	Standardize features by removing the mean and scaling to unit variance.
`preprocessing.TargetEncoder`([categories, ...])	Target Encoder for regression and classification targets.

`preprocessing.add_dummy_feature`(X[, value])	Augment dataset with an additional dummy feature.
`preprocessing.binarize`(X, *[, threshold, copy])	Boolean thresholding of array-like or scipy.sparse matrix.
`preprocessing.label_binarize`(y, *, classes)	Binarize labels in a one-vs-all fashion.
`preprocessing.maxabs_scale`(X, *[, axis, copy])	Scale each feature to the [-1, 1] range without breaking the sparsity.
`preprocessing.minmax_scale`(X[, ...])	Transform features by scaling each feature to a given range.
`preprocessing.normalize`(X[, norm, axis, ...])	Scale input vectors individually to unit norm (vector length).
`preprocessing.quantile_transform`(X, *[, ...])	Transform features using quantiles information.
`preprocessing.robust_scale`(X, *[, axis, ...])	Standardize a dataset along any axis.
`preprocessing.scale`(X, *[, axis, with_mean, ...])	Standardize a dataset along any axis.
`preprocessing.power_transform`(X[, method, ...])	Parametric, monotonic transformation to make data more Gaussian-like.

Python Scikit-learn library[3]

`sklearn.preprocessing`: Preprocessing and Normalization

The `sklearn.preprocessing` module includes scaling, centering, normalization, binarization methods.

User guide: See the Preprocessing data section for further details.

`preprocessing.Binarizer`(*[, threshold, copy])	Binarize data (set feature values to 0 or 1) according to a threshold.
`preprocessing.FunctionTransformer`([func, ...])	Constructs a transformer from an arbitrary callable.
`preprocessing.KBinsDiscretizer`([n_bins, ...])	Bin continuous data into intervals.
`preprocessing.KernelCenterer`()	Center an arbitrary kernel matrix K.
`preprocessing.LabelBinarizer`(*[, neg_label, ...])	Binarize labels in a one-vs-all fashion.
`preprocessing.LabelEncoder`()	Encode target labels with value between 0 and n_classes-1.
`preprocessing.MultiLabelBinarizer`(*[, ...])	Transform between iterable of iterables and a multilabel format.
`preprocessing.MaxAbsScaler`(*[, copy])	Scale each feature by its maximum absolute value.
`preprocessing.MinMaxScaler`([feature_range, ...])	Transform features by scaling each feature to a given range.
`preprocessing.Normalizer`([norm, copy])	Normalize samples individually to unit norm.
`preprocessing.OneHotEncoder`(*[, categories, ...])	Encode categorical features as a one-hot numeric array.
`preprocessing.OrdinalEncoder`(*[, ...])	Encode categorical features as an integer array.
`preprocessing.PolynomialFeatures`([degree, ...])	Generate polynomial and interaction features.
`preprocessing.PowerTransformer`([method, ...])	Apply a power transform featurewise to make data more Gaussian-like.
`preprocessing.QuantileTransformer`(*[, ...])	Transform features using quantiles information.
`preprocessing.RobustScaler`(*[, ...])	Scale features using statistics that are robust to outliers.
`preprocessing.SplineTransformer`([n_knots, ...])	Generate univariate B-spline bases for features.
`preprocessing.StandardScaler`(*[, copy, ...])	Standardize features by removing the mean and scaling to unit variance.
`preprocessing.TargetEncoder`([categories, ...])	Target Encoder for regression and classification targets.

`preprocessing.add_dummy_feature`(X[, value])	Augment dataset with an additional dummy feature.
`preprocessing.binarize`(X, *[, threshold, copy])	Boolean thresholding of array-like or scipy.sparse matrix.
`preprocessing.label_binarize`(y, *, classes)	Binarize labels in a one-vs-all fashion.
`preprocessing.maxabs_scale`(X, *[, axis, copy])	Scale each feature to the [-1, 1] range without breaking the sparsity.
`preprocessing.minmax_scale`(X[, ...])	Transform features by scaling each feature to a given range.
`preprocessing.normalize`(X[, norm, axis, ...])	Scale input vectors individually to unit norm (vector length).
`preprocessing.quantile_transform`(X, *[, ...])	Transform features using quantiles information.
`preprocessing.robust_scale`(X, *[, axis, ...])	Standardize a dataset along any axis.
`preprocessing.scale`(X, *[, axis, with_mean, ...])	Standardize a dataset along any axis.
`preprocessing.power_transform`(X[, method, ...])	Parametric, monotonic transformation to make data more Gaussian-like.

Code

External library implementations (Apache Commons Mathematics Library [4])

```
import org.apache.commons.math.stat.descriptive.SummaryStatistics;
```

Computes summary statistics for very large data streams.

Data values are not stored in memory.

```
public class Standardizer
```

```
statistics.getMean()
```

```
statistics.getStandardDeviation()
```

```
public class Normalizer
```

```
statistics.getMin()
```

```
statistics.getMax() - statistics.getMin()
```

Code

Integrating with netcdf-java

- CDM.java
  - Create constants / attributes in the Common Data Model class of constants.
- NetcdfDataset.java
  - Include Standardizer / Normalizer in the set of enhancements (ApplyStandardizer, ApplyNormalizer)
- VariableDS.java
  - Check if attribute "standardizer" "normalizer" is present and data is floating point
  - Apply enhancements to the data

Code

Using it in the TDS

The preprocessing can be achieved in the THREDDS catalog.xml through NetCDF Markup Language (NcML)[5].

NcML creates a virtual dataset without changing the original data.

```
<netcdf xmlns="http://www.unidata.ucar.edu/namespaces/netcdf/ncml-2.2" enhance="all">
  <variable name="scaledVar">
    <attribute name="standardize"/>
  </variable>
</netcdf>
```

```
<netcdf xmlns="http://www.unidata.ucar.edu/namespaces/netcdf/ncml-2.2" enhance="all">
  <variable name="scaledVar">
    <attribute name="normalize"/>
  </variable>
</netcdf>
```

Testing

- Test Convert
  - Doubles, Floats
  - **NaNs, Equal values**

Standardizer

- Test Calculate Mean
  - Doubles, Floats
- Test Calculate Standard Deviation
  - Doubles, Floats

Normalizer

- Test Calculate Minimum
  - Doubles, Floats
- Test Calculate Range
  - Doubles, Floats

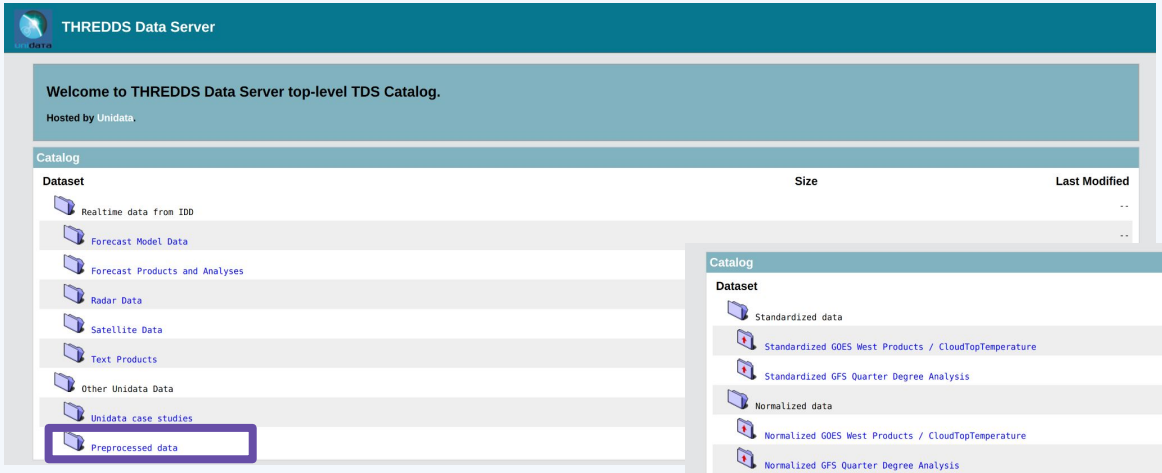
Testing

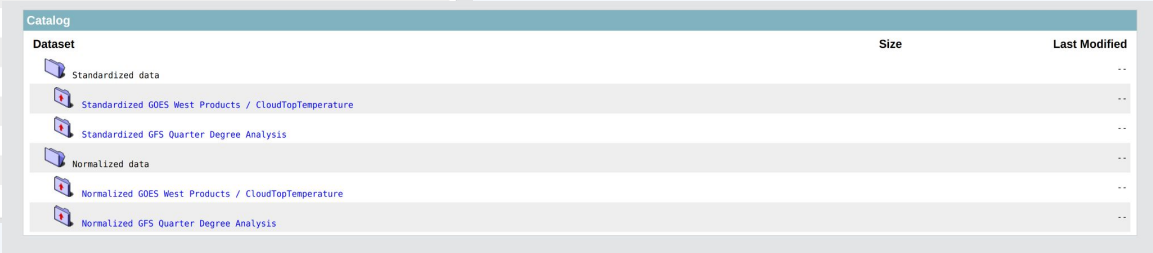
- Read variables from NcML file
- Doubles, Floats
- Ints
- Equal values

```java
@Test
public void testEnhanceStandardizer() throws IOException {
  try (NetcdfFile ncfile = NetcdfDatasets.openDataset( location: dataDir + "testStandardizer.ncml",  enhance: true,  cancelTask: null)) {
    Variable doubleVar = ncfile.findVariable( fullNameEscaped: "doublevar");
    assertThat((Object) doubleVar).isNotNull();
    assertThat(doubleVar.getDataType()).isEqualTo(DataType.DOUBLE);
    assertThat(doubleVar.attributes().hasAttribute( attName: "standardize")).isTrue();
    Array dataDoubles = doubleVar.read();
    assertThat(nearlyEquals(dataDoubles, DATA_DOUBLES)).isTrue();

    Variable sameDoubleVar = ncfile.findVariable( fullNameEscaped: "samedoublevar");
    assertThat((Object) sameDoubleVar).isNotNull();
    assertThat(sameDoubleVar.getDataType()).isEqualTo(DataType.DOUBLE);
    assertThat(sameDoubleVar.attributes().hasAttribute( attName: "standardize")).isTrue();
    Array dataSameDoubles = sameDoubleVar.read();
    assertThat(nearlyEquals(dataSameDoubles, DATA_SAMEDOUBLES)).isTrue(); // The enhancement doesn't apply if all the
                                                                           // values are the equal, so it returns the
                                                                           // same data

    Variable floatVar = ncfile.findVariable( fullNameEscaped: "floatvar");
    assertThat((Object) floatVar).isNotNull();
    assertThat(floatVar.getDataType()).isEqualTo(DataType.FLOAT);
    assertThat(doubleVar.attributes().hasAttribute( attName: "standardize")).isTrue();
    Array dataFloats = doubleVar.read();
    assertThat(nearlyEquals(dataFloats, DATA_FLOATS)).isTrue();

    Variable intVar = ncfile.findVariable( fullNameEscaped: "intvar");
    assertThat((Object) intVar).isNotNull();
    assertThat(intVar.getDataType()).isEqualTo(DataType.INT);
    assertThat(intVar.attributes().hasAttribute( attName: "standardize")).isTrue();
    Array data = intVar.read();
    assertThat(nearlyEquals(data, DATA_INTS)).isTrue(); // The enhancement doesn't apply to ints, so the data should
                                                         // be equal to the input array
  }
}
```

Results

Thredds-test server





Results

Jupyter Notebooks

THREDDS Data Server

Access | **Preview**

Viewers:

Viewer	Type	Description
Godiva3	Browser	
default_viewer.ipynb	Jupyter Notebook	The TDS default viewer attempts to plot any Variable contained in the Dataset.
preprocessed_dataset.ipynb	Jupyter Notebook	Preprocessed Dataset Visualization

Siphon THREDDS Jupyter Notebook - Visualizing Preprocessed Data

Powered By unidata

Dataset: GFS_Global_0p25deg_ana_20230725_1200.grib2

```python
import xarray as xr
from siphon.catalog import TDSCatalog
```

[1] ✓ 1.9s Python

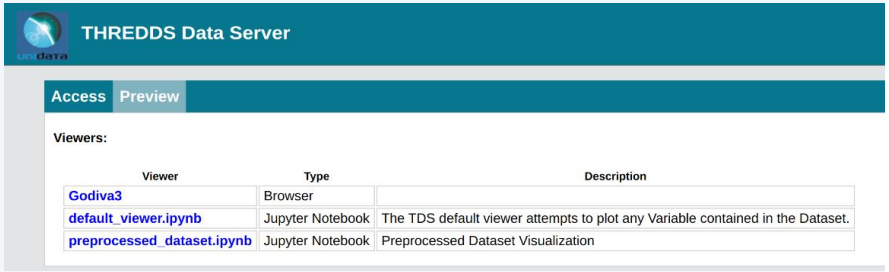
Standardized Data

z: Data point
μ: Mean value in the variable
σ: Standard deviation value in the variable
s: Standardized data point

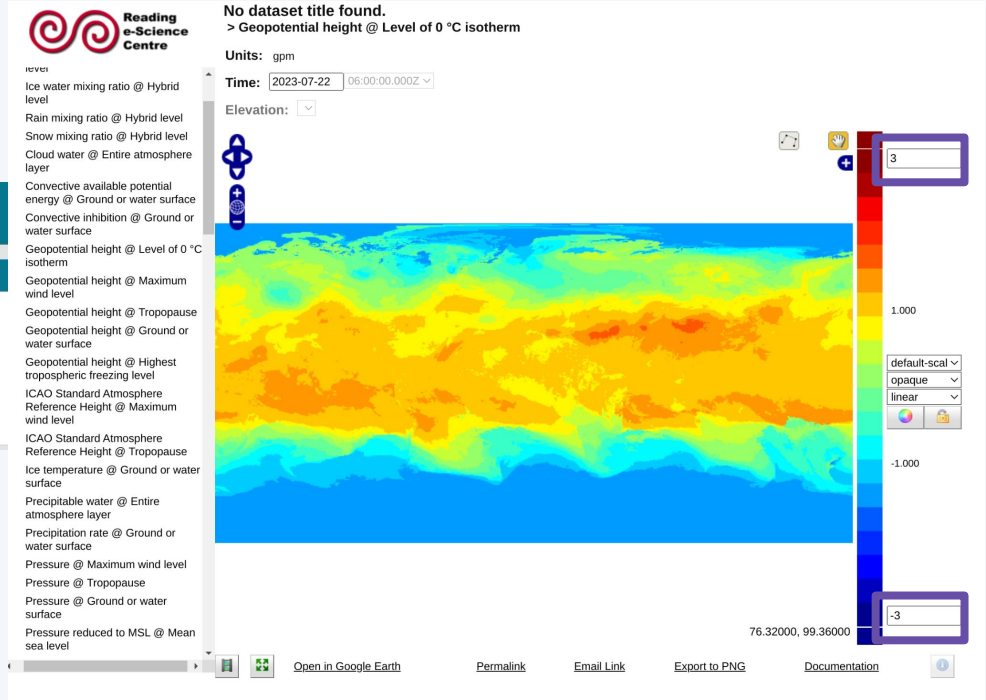
$$s = \frac{z - \mu}{\sigma}$$

Results

Godiva3



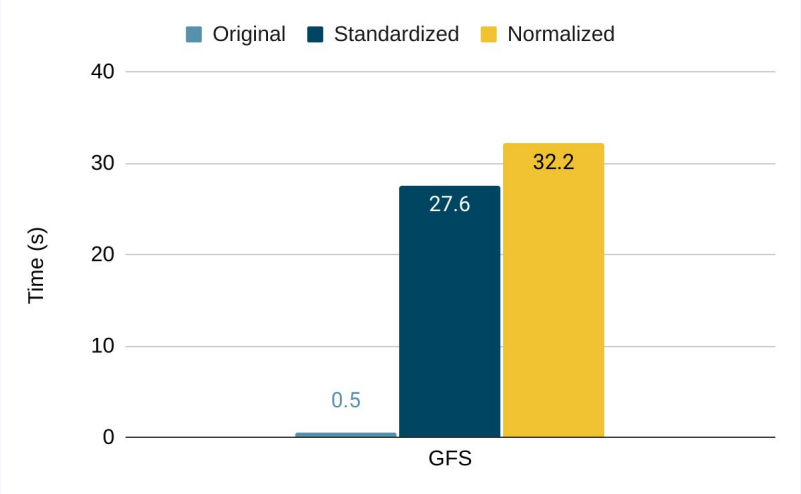
Viewer	Type	Description
Godiva3	Browser	
default_viewer.ipynb	Jupyter Notebook	The TDS default viewer attempts to plot any Variable contained in the Dataset.
preprocessed_dataset.ipynb	Jupyter Notebook	Preprocessed Dataset Visualization



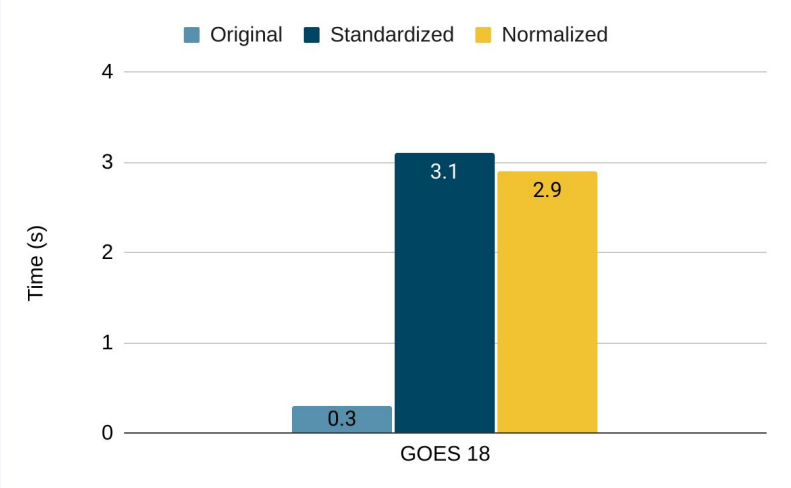
Results

Performance test - Apache HTTP server benchmarking tool[6]
Simple load tests on thredds-test server

GFS 0.25 Degree - Forecast Model Data
+80 variables



GOES 18 Product - Full Conus
1 variable



Final Remarks

Dataset enhancements targeting machine learning applications.

Two most common types of rescaling data as part of preprocessing step.

Available original + standardized & normalized dataset on thredds-test server.

Future work:

- Improvements on performance
  - Caching for example
- Provide more datasets relevant to the users.

References

[1] **THREDDS Data Server**
https://www.unidata.ucar.edu/software/tds/

[2] **Artificial Intelligence for the Earth Systems**
https://www.ametsoc.org/index.cfm/ams/publications/journals/artificial-intelligence-for-the-earth-systems/

[3] **Scikit-learn library**
https://scikit-learn.org/

[4] **Commons Math: The Apache Commons Mathematics Library**
https://commons.apache.org/proper/commons-math/

[5] **NetCDF Markup Language (NcML)**
https://docs.unidata.ucar.edu/netcdf-java/current/userguide/ncml_overview.html

[6] **Apache HTTP server benchmarking tool**
https://httpd.apache.org/docs/2.4/programs/ab.html

Acknowledgements