MoDeVa Guide to Data Operations

July 5, 2025

Tutorial Outline

MoDeVa Data Preparation

- Python library for data operations for MoDeVa
- Built around the central `DataSet` class
- Supports end-to-end ML data workflows

Key Features

- Data loading (built-in and external datasets)
- Data summarization
- Preprocessing pipelines
- Dataset registration and management
- MLflow integration for experiment tracking

MoDeVa Data Workflow Overview

Data Loading → Data Summary → Preprocessing → Preparation → Registration

Central Concept

All operations are performed through the `DataSet` class, which maintains data state and provides method chaining capabilities.

Data Loading Overview

Built-in Datasets:

- BikeSharing (Regression)
- CaliforniaHousing (Regression)
- SimuCredit (Classification)
- TaiwanCredit (Classification)

External Data Sources:

- CSV files (`load_csv`)
- Pandas DataFrames (`load_dataframe`)
- Spark DataFrames (`load_spark`)

Usage Pattern

1. Create DataSet instance
2. Load data using appropriate method
3. Data becomes available in `ds.data`

Loading Built-in Datasets

```python
# Create DataSet instance
from modeva import DataSet
ds = DataSet()

# Load built-in dataset
ds.load("SimuCredit")

# Explore the data
ds.data.head(5)
```

Available Built-in Datasets

- `"BikeSharing"` - Bike rental prediction (regression)
- `"CaliforniaHousing"` - Housing prices (regression)
- `"SimuCredit"` - Credit risk simulation (classification)
- `"TaiwanCredit"` - Credit default prediction (classification)

Loading External Data

```python
# Loading from external sources
import pandas as pd
from sklearn.datasets import load_iris
from modeva import DataSet
# Load and prepare external data
iris = load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)

# Create named DataSet and load DataFrame
ds = DataSet(name="IrisData")
ds.load_dataframe(df)
```

Loading Methods

- `load_csv(filepath)` - Direct CSV loading
- `load_dataframe(df)` - From pandas DataFrame
- `load_spark(spark_df)` - From Spark DataFrame

Data Summary Components

Summary Structure

1. **Overall Summary** (`res.table["summary"]`)
   - Sample count, feature types, duplicates
   - Missing and infinite value statistics
2. **Categorical Variables** (`res.table["categorical"]`)
   - Missing values, unique counts
   - Top 2 value frequencies
3. **Numerical Variables** (`res.table["numerical"]`)
   - Descriptive statistics (mean, std, percentiles)
   - Missing/infinite value counts

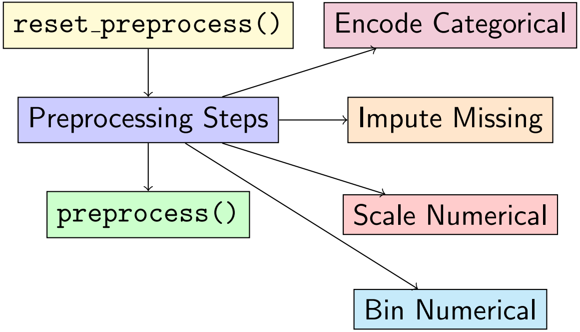
Generating Data Summary

```python
# Generate comprehensive summary
res = ds.summary()
# Access overall dataset summary
overall_summary = res.table["summary"]
print(overall_summary)
# Access categorical variable statistics
categorical_stats = res.table["categorical"]
print(categorical_stats)
# Access numerical variable statistics
numerical_stats = res.table["numerical"]
print(numerical_stats)
```

Key Insight

The summary provides a comprehensive view of data quality issues, distribution characteristics, and variable types before preprocessing.

Preprocessing Pipeline Architecture



Preprocessing Workflow

1. Initialize preprocessing pipeline
2. Define preprocessing steps
3. Execute all steps in sequence

Handling Missing Values

```
# Initialize preprocessing
ds.reset_preprocess()
# Impute numerical features with mean and add indicators
ds.impute_missing(
    features=ds.feature_names_numerical,
    method='mean', add_indicators=True
)
# Impute categorical features with most frequent value
ds.impute_missing(
    features=ds.feature_names_categorical,
    method='most_frequent', add_indicators=True
)
# Handle mixed features with special values
ds.impute_missing(
    features=ds.feature_names_mixed, method='median',
    add_indicators=True, special_values=["SV1", "SV2"]
)
```

Categorical Variable Encoding

```
# One-hot encoding for categorical features
ds.encode_categorical(
    features=("Gender", "Race"),
    method="onehot"
)

# Ordinal encoding (alternative)
ds.encode_categorical(
    features=("Education_Level",),
    method="ordinal"
)
```

Encoding Methods

- **One-hot**: Creates binary columns for each category
- **Ordinal**: Maps categories to ordered integers

Numerical Variable Scaling

```
# Log transformation for skewed features
ds.scale_numerical(
    features=("Mortgage", "Balance"), method="log1p"
)

# Min-max scaling for bounded features
ds.scale_numerical(
    features=("Delinquency",), method="minmax"
)

# Quantile transformation for robust scaling
ds.scale_numerical(
    features=("Inquiry",), method="quantile"
)

# Standardization (alternative)
ds.scale_numerical(
    features=("Income",), method="standardize"
)
```

Numerical Variable Binning

```python
# Uniform binning - equal-width intervals
ds.bin_numerical(
    features=("Utilization",), bins=10, method="uniform"
)
# Quantile binning - equal-frequency intervals
ds.bin_numerical(
    features=("Mortgage", "Balance", "Amount_Past_Due"),
    bins=10, method="quantile"
)
# Execute all preprocessing steps
ds.preprocess()
```

Binning Methods

- **Uniform**: Equal-width intervals
- **Quantile**: Equal-frequency intervals
- **Precompute**: User-defined bin edges

Data Preparation for Modeling

Essential Configuration Steps

1. **Train-Test Splitting**: Define data splits for validation
2. **Target Variable**: Specify the prediction target
3. **Task Type**: Set regression or classification mode
4. **Feature Selection**: Choose active/inactive features
5. **Sample Weighting**: Handle imbalanced datasets

Modeling Readiness

Data preparation ensures the dataset is properly configured for machine learning algorithms with clear targets, appropriate splits, and selected features.

Configuring Dataset for Modeling

```
# Split data into training and testing sets
ds.set_random_split()
# Set target variable for prediction
ds.set_target("Status")
# Set task type (Classification or Regression)
ds.set_task_type("Classification")
# Exclude features from modeling
ds.set_inactive_features(features=('Gender', 'Race'))
# Set sample weights (optional)
ds.set_sample_weight("sample_weight_column")
# Override active features (optional)
ds.set_active_features(features=('Income', 'Age', 'Balance'))
```

Key Configuration Methods

Feature selection can be done through inclusion (`set_active_features`) or exclusion (`set_inactive_features`) approaches.

Dataset Registration and Management

MLflow Integration Benefits

- **Version Control**: Track dataset changes over time
- **Reproducibility**: Ensure consistent data across experiments
- **Collaboration**: Share datasets across team members
- **Metadata Tracking**: Store dataset properties and lineage

Registration Operations:

- Register datasets
- List registered datasets
- Delete datasets

Use Cases:

- Experiment tracking
- Dataset versioning
- Team collaboration

Dataset Registration Operations

```python
# Register the processed dataset
ds.register(name="A0-SimuCredit", override=True)

# List all registered datasets
registered_datasets = ds.list_registered_data()
print(registered_datasets)

# Delete a registered dataset (if needed)
ds.delete_registered_data(name="old_dataset_name")
```

Registration Best Practices

- Use descriptive names with version indicators
- Include preprocessing information in dataset names
- Use `override=True` carefully to avoid data loss
- Regularly clean up unused registered datasets

End-to-End MoDeVa Workflow

```python
from modeva import DataSet
# 1. Data Loading
ds = DataSet()
ds.load("SimuCredit")

# 2. Data Summary
summary_results = ds.summary()
print(summary_results.table["summary"])

# 3. Data Preprocessing
ds.reset_preprocess()
ds.impute_missing(features=ds.feature_names_numerical, method='mean')
ds.encode_categorical(features=ds.feature_names_categorical, method="onehot")
ds.scale_numerical(features=("Income", "Balance"), method="standardize")
ds.preprocess()

# 4. Data Preparation
ds.set_random_split()
ds.set_target("Status")
ds.set_task_type("Classification")
ds.set_inactive_features(features=('ID',))

# 5. Data Registration
ds.register(name="processed_simucredit_v1", override=True)
```

Best Practices and Tips

Data Quality

- Always run `summary()` before preprocessing
- Check for data leakage in feature selection
- Validate preprocessing results

Preprocessing Strategy

- Handle missing values before encoding
- Scale features after encoding categorical variables
- Consider domain knowledge in binning decisions

Experiment Management

- Use consistent naming conventions for registered datasets
- Document preprocessing steps in dataset names
- Maintain preprocessing pipeline documentation

Summary

MoDeVa Capabilities

- **Unified Interface**: Single DataSet class for all operations
- **Comprehensive Processing**: End-to-end data workflow support
- **Built-in Datasets**: Ready-to-use demo datasets
- **Flexible Integration**: Support for various data sources
- **Experiment Tracking**: MLflow integration for reproducibility

Next Steps

- Practice with built-in datasets
- Experiment with different preprocessing combinations
- Integrate with your ML modeling workflows
- Explore advanced MoDeVa features