# GizaAutoML User Manual

## 1. Installation

To use GizaAutoML, install the package using the following command:

```
pip install git+https://github.com/giza-data-team/GizaAutoML.git@0.1
```

## 2. Package Overview

GizaAutoML is designed for univariate time series forecasting. The package performs the following steps:

- **Data Preprocessing**: Handles missing value imputation.
- **Feature Extraction**: Extracts features such as time, trend, seasonality, and lags.
- **Algorithm Selection**: Recommends the best 3 algorithms based on the series meta features and the meta features saved in the engine knowledge base.
- **SMAC Optimization**: Applies SMAC optimization to the recommended algorithms, returning the best-performing model with minimal cost.

## 3. Usage

### Initialization

```
from GizaAutoML.auto_series_forecaster import AutoSeriesForecaster

# Instantiate the AutoSeriesForecaster class
auto_ml = AutoSeriesForecaster(raw_dataframe, optimization_metric="MAE",
    time_budget=10, save_results=True, random_seed=1, target_col="Target")
```

### Fit

```
# Fit the model
pipeline = auto_ml.fit()
```

### Transform

```
# Transform new data using the trained pipeline
transformed_data = auto_ml.transform(new_data)
```

## 4. Parameters

- **raw_dataframe**: The input raw time series data.
- **optimization_metric**: The metric used for optimizing algorithms during the process (default is "MAE").
- **time_budget**: The maximum time allowed for optimization in minutes (default is 10 minutes).
- **save_results**: Boolean flag to update the knowledge base with new meta features and best algorithm results on the dataset under investigation (default is True).
- **random_seed**: Seed for reproducibility (default is 1).
- **target_col**: The name of the column containing the time series.
- **processed_dataframe**: Preprocessed data if already available (no preprocessing will be applied if provided).
- **dataset_name** and **dataset_instance**: Auto-generated in case saving results is set to True.

## 5. Output

The `fit` method returns the trained machine learning pipeline, which can be used for making predictions on new data.

## 6. Examples

```
# Example usage
from GizaAutoML.auto_series_forecaster import AutoSeriesForecaster

# Load your time series data into a DataFrame called 'raw_data'
auto_ml = AutoSeriesForecaster(raw_dataframe=raw_data, optimization_metric="MAE
    ", time_budget=10, save_results=True, random_seed=1, target_col="Target")

# Fit the model
pipeline = auto_ml.fit()

# Transform new data
new_data_transformed = auto_ml.transform(new_data)
```

### 0.1 Timestamp Column Format

One crucial aspect for successful time series forecasting with GizaAutoML is the proper formatting of the timestamp column in your dataset. Follow these guidelines to ensure accurate results:

- **Format Requirement:** The timestamp column should be in the datetime format.
- **Example Conversion:** If your timestamp is not in datetime format, you can use the following code to convert it:

```
    import pandas as pd

    # Assuming 'Timestamp' is the name of your timestamp column
    df['Timestamp'] = pd.to_datetime(df['Timestamp'])

```

- **Optimal Performance:** Proper formatting of the timestamp is crucial for optimal performance. GizaAutoML leverages temporal information, and accurate timestamp representation enhances forecasting accuracy.

## 7. Troubleshooting

If you encounter any issues or have questions, refer to the documentation or contact our support team.

## 8. Acknowledgments

GizaAutoML is a powerful tool for automating time series forecasting. We appreciate your feedback and contributions to make this package even more robust.

Happy forecasting with GizaAutoML!