

Package: nixtlar (via r-universe)

September 2, 2024

Title A Software Development Kit for 'Nixtla's 'TimeGPT'

Version 0.5.4

Description A Software Development Kit for working with 'Nixtla's 'TimeGPT', a foundation model for time series forecasting. 'API' is an acronym for 'application programming interface'; this package allows users to interact with 'TimeGPT' via the 'API'. You can set and validate 'API' keys and generate forecasts via 'API' calls. It is compatible with 'tsibble' and base R. For more details visit <<https://docs.nixtla.io/>>.

License Apache License (>= 2.0)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Depends R (>= 2.10)

LazyData true

Imports dplyr, future, future.apply, ggplot2, htr2, lubridate, rlang, tidyr, tidyselect, tsibble

Suggests httptest2, knitr, rmarkdown, testthat (>= 3.0.0), usethis

Config/testthat/edition 3

URL <https://nixtla.github.io/nixtlar/>, <https://docs.nixtla.io/>

VignetteBuilder knitr

Repository <https://nixtla.r-universe.dev>

RemoteUrl <https://github.com/nixtla/nixtlar>

RemoteRef HEAD

RemoteSha e4834b5c5272ab9a5283f3c0a4cfce9e88930a41

Contents

.nixtla_client_historic_seq	2
electricity	3

electricity_exo_vars	4
electricity_future_exo_vars	5
infer_frequency	6
nixtla_client_cross_validation	6
nixtla_client_detect_anomalies	8
nixtla_client_forecast	9
nixtla_client_historic	10
nixtla_client_plot	12
nixtla_set_api_key	13
nixtla_validate_api_key	14

Index 15

.nixtla_client_historic_seq
Sequential version of 'nixtla_client_historic' This is a private function of 'nixtla'

Description

Sequential version of 'nixtla_client_historic' This is a private function of 'nixtla'

Usage

```
.nixtla_client_historic_seq(
  df,
  freq = NULL,
  id_col = NULL,
  time_col = "ds",
  target_col = "y",
  level = NULL,
  quantiles = NULL,
  finetune_steps = 0,
  finetune_loss = "default",
  clean_ex_first = TRUE
)
```

Arguments

df	A tsibble or a data frame with time series data.
freq	Frequency of the data.
id_col	Column that identifies each series.
time_col	Column that identifies each timestep.
target_col	Column that contains the target variable.
level	The confidence levels (0-100) for the prediction intervals.
quantiles	Quantiles to forecast. Should be between 0 and 1.

`finetune_steps` Number of steps used to finetune 'TimeGPT' in the new data.

`finetune_loss` Loss function to use for finetuning. Options are: "default", "mae", "mse", "rmse", "mape", and "smape".

`clean_ex_first` Clean exogenous signal before making the forecasts using 'TimeGPT'.

Value

'TimeGPT's forecast for the in-sample period.

Examples

```
## Not run:
nixtlar::nixtla_set_api_key("YOUR_API_KEY")
df <- nixtlar::electricity
fcst <- nixtlar::nixtla_client_historic(df, id_col="unique_id", level=c(80,95))

## End(Not run)
```

electricity	<i>Electricity dataset</i>
-------------	----------------------------

Description

Contains prices of different electricity markets.

Usage

```
electricity
```

Format

`electricity`:

A data frame with 8400 rows and 3 columns:

- unique_id** Unique identifiers of the electricity markets.
- ds** Date in format YYYY:MM:DD hh:mm:ss.
- y** Price for the given market and date.

Source

<https://raw.githubusercontent.com/Nixtla/transfer-learning-time-series/main/datasets/electricity-short.csv>

electricity_exo_vars *Electricity dataset with exogenous variables*

Description

Contains prices of different electricity markets with exogenous variables.

Usage

```
electricity_exo_vars
```

Format

electricity_exo_vars:

A data frame with 8400 rows and 12 columns:

unique_id Unique identifiers of the electricity markets.

ds Date in format YYYY:MM:DD hh:mm:ss.

y Price for the given market and date.

Exogenous1 An external factor influencing prices. For all markets, some form of day-ahead load forecast.

Exogenous2 An external factor influencing prices. For "BE" and "FR" markets, the day-ahead generation forecast. For "NP", the day-ahead wind generation forecast. For "PJM", the day-ahead load forecast in a specific zone. For "DE", the aggregated day-ahead wind and solar generation forecasts.

day_0 Binary variable indicating weekday.

day_1 Binary variable indicating weekday.

day_2 Binary variable indicating weekday.

day_3 Binary variable indicating weekday.

day_4 Binary variable indicating weekday.

day_5 Binary variable indicating weekday.

day_6 Binary variable indicating weekday.

Source

<https://raw.githubusercontent.com/Nixtla/transfer-learning-time-series/main/datasets/electricity-short.csv>

electricity_future_exo_vars

Future values for the electricity dataset with exogenous variables

Description

Contains the future values of the exogenous variables of the electricity dataset (24 steps-ahead). To be used with `electricity_exo_vars`.

Usage

```
electricity_future_exo_vars
```

Format

`electricity_future_exo_vars`:

A data frame with 120 rows and 11 columns:

unique_id Unique identifiers of the electricity markets.

ds Date in format YYYY:MM:DD hh:mm:ss.

Exogenous1 An external factor influencing prices. For all markets, some form of day-ahead load forecast.

Exogenous2 An external factor influencing prices. For "BE" and "FR" markets, the day-ahead generation forecast. For "NP", the day-ahead wind generation forecast. For "PJM", the day-ahead load forecast in a specific zone. For "DE", the aggregated day-ahead wind and solar generation forecasts.

day_0 Binary variable indicating weekday.

day_1 Binary variable indicating weekday.

day_2 Binary variable indicating weekday.

day_3 Binary variable indicating weekday.

day_4 Binary variable indicating weekday.

day_5 Binary variable indicating weekday.

day_6 Binary variable indicating weekday.

Source

<https://raw.githubusercontent.com/Nixtla/transfer-learning-time-series/main/datasets/electricity-short-future-ex-vars.csv>

infer_frequency *Infer frequency of a data frame.*

Description

Infer frequency of a data frame.

Usage

```
infer_frequency(df)
```

Arguments

df A data frame with time series data.

Value

The inferred frequency.

Examples

```
df <- nixtla::electricity
infer_frequency(df)
```

nixtla_client_cross_validation
Perform cross validation with 'TimeGPT'.

Description

Perform cross validation with 'TimeGPT'.

Usage

```
nixtla_client_cross_validation(  
  df,  
  h = 8,  
  freq = NULL,  
  id_col = NULL,  
  time_col = "ds",  
  target_col = "y",  
  X_df = NULL,  
  level = NULL,  
  quantiles = NULL,  
  n_windows = 1,
```

```

    step_size = NULL,
    finetune_steps = 0,
    finetune_loss = "default",
    clean_ex_first = TRUE,
    model = "timegpt-1",
    num_partitions = NULL
  )

```

Arguments

<code>df</code>	A tibble or a data frame with time series data.
<code>h</code>	Forecast horizon.
<code>freq</code>	Frequency of the data.
<code>id_col</code>	Column that identifies each series.
<code>time_col</code>	Column that identifies each timestep.
<code>target_col</code>	Column that contains the target variable.
<code>X_df</code>	A tibble or a data frame with future exogenous variables.
<code>level</code>	The confidence levels (0-100) for the prediction intervals.
<code>quantiles</code>	Quantiles to forecast. Should be between 0 and 1.
<code>n_windows</code>	Number of windows to evaluate.
<code>step_size</code>	Step size between each cross validation window. If NULL, it will equal the forecast horizon (h).
<code>finetune_steps</code>	Number of steps used to finetune 'TimeGPT' in the new data.
<code>finetune_loss</code>	Loss function to use for finetuning. Options are: "default", "mae", "mse", "rmse", "mape", and "smape".
<code>clean_ex_first</code>	Clean exogenous signal before making the forecasts using 'TimeGPT'.
<code>model</code>	Model to use, either "timegpt-1" or "timegpt-1-long-horizon". Use "timegpt-1-long-horizon" if you want to forecast more than one seasonal period given the frequency of the data.
<code>num_partitions</code>	A positive integer, "auto", or NULL specifying the number of partitions. When set to "auto", the number of partitions is equal to the number of available cores. When NULL, it defaults to a single partition.

Value

A tibble or a data frame with 'TimeGPT's cross validation result.

Examples

```

## Not run:
nixtlar::nixtla_set_api_key("YOUR_API_KEY")
df <- nixtlar::electricity
fcst <- nixtlar::nixtla_client_cross_validation(df, h = 8, id_col = "unique_id", n_windows = 5)

## End(Not run)

```

`nixtla_client_detect_anomalies`*Detect anomalies with 'TimeGPT'*

Description

Detect anomalies with 'TimeGPT'

Usage

```
nixtla_client_detect_anomalies(  
  df,  
  freq = NULL,  
  id_col = NULL,  
  time_col = "ds",  
  target_col = "y",  
  level = c(99),  
  clean_ex_first = TRUE,  
  model = "timegpt-1",  
  num_partitions = NULL  
)
```

Arguments

<code>df</code>	A tsibble or a data frame with time series data.
<code>freq</code>	Frequency of the data.
<code>id_col</code>	Column that identifies each series.
<code>time_col</code>	Column that identifies each timestep.
<code>target_col</code>	Column that contains the target variable.
<code>level</code>	The confidence level (0-100) for the prediction interval used in anomaly detection. Default is 99.
<code>clean_ex_first</code>	Clean exogenous signal before making the forecasts using 'TimeGPT'.
<code>model</code>	Model to use, either "timegpt-1" or "timegpt-1-long-horizon". Use "timegpt-1-long-horizon" if you want to forecast more than one seasonal period given the frequency of the data.
<code>num_partitions</code>	A positive integer, "auto", or NULL specifying the number of partitions. When set to "auto", the number of partitions is equal to the number of available cores. When NULL, it defaults to a single partition.

Value

A tsibble or a data frame with the anomalies detected in the historical period.

Examples

```
## Not run:
nixtlar::nixtla_set_api_key("YOUR_API_KEY")
df <- nixtlar::electricity
fcst <- nixtlar::nixtla_client_anomaly_detection(df, id_col="unique_id")

## End(Not run)
```

nixtla_client_forecast

Generate 'TimeGPT' forecast

Description

Generate 'TimeGPT' forecast

Usage

```
nixtla_client_forecast(
  df,
  h = 8,
  freq = NULL,
  id_col = NULL,
  time_col = "ds",
  target_col = "y",
  X_df = NULL,
  level = NULL,
  quantiles = NULL,
  finetune_steps = 0,
  finetune_loss = "default",
  clean_ex_first = TRUE,
  add_history = FALSE,
  model = "timegpt-1",
  num_partitions = NULL
)
```

Arguments

df	A tsibble or a data frame with time series data.
h	Forecast horizon.
freq	Frequency of the data.
id_col	Column that identifies each series.
time_col	Column that identifies each timestep.
target_col	Column that contains the target variable.

X_df	A tibble or a data frame with future exogenous variables.
level	The confidence levels (0-100) for the prediction intervals.
quantiles	Quantiles to forecast. Should be between 0 and 1.
finetune_steps	Number of steps used to finetune 'TimeGPT' in the new data.
finetune_loss	Loss function to use for finetuning. Options are: "default", "mae", "mse", "rmse", "mape", and "smape".
clean_ex_first	Clean exogenous signal before making the forecasts using 'TimeGPT'.
add_history	Return fitted values of the model.
model	Model to use, either "timegpt-1" or "timegpt-1-long-horizon". Use "timegpt-1-long-horizon" if you want to forecast more than one seasonal period given the frequency of the data.
num_partitions	A positive integer, "auto", or NULL specifying the number of partitions. When set to "auto", the number of partitions is equal to the number of available cores. When NULL, it defaults to a single partition.

Value

'TimeGPT's forecast.

Examples

```
## Not run:
nixtlar::nixtla_set_api_key("YOUR_API_KEY")
df <- nixtlar::electricity
fcst <- nixtlar::nixtla_client_forecast(df, h=8, id_col="unique_id", level=c(80,95))

## End(Not run)
```

nixtla_client_historic

Generate 'TimeGPT' forecast for the in-sample period (historical period).

Description

Generate 'TimeGPT' forecast for the in-sample period (historical period).

Usage

```
nixtla_client_historic(
  df,
  freq = NULL,
  id_col = NULL,
  time_col = "ds",
```

```

    target_col = "y",
    level = NULL,
    quantiles = NULL,
    finetune_steps = 0,
    finetune_loss = "default",
    clean_ex_first = TRUE,
    model = "timegpt-1",
    num_partitions = NULL
  )

```

Arguments

<code>df</code>	A tibble or a data frame with time series data.
<code>freq</code>	Frequency of the data.
<code>id_col</code>	Column that identifies each series.
<code>time_col</code>	Column that identifies each timestep.
<code>target_col</code>	Column that contains the target variable.
<code>level</code>	The confidence levels (0-100) for the prediction intervals.
<code>quantiles</code>	Quantiles to forecast. Should be between 0 and 1.
<code>finetune_steps</code>	Number of steps used to finetune 'TimeGPT' in the new data.
<code>finetune_loss</code>	Loss function to use for finetuning. Options are: "default", "mae", "mse", "rmse", "mape", and "smape".
<code>clean_ex_first</code>	Clean exogenous signal before making the forecasts using 'TimeGPT'.
<code>model</code>	Model to use, either "timegpt-1" or "timegpt-1-long-horizon". Use "timegpt-1-long-horizon" if you want to forecast more than one seasonal period given the frequency of the data.
<code>num_partitions</code>	A positive integer, "auto", or NULL specifying the number of partitions. When set to "auto", the number of partitions is equal to the number of available cores. When NULL, it defaults to a single partition.

Value

'TimeGPT's forecast for the in-sample period.

Examples

```

## Not run:
nixtlar::nixtla_set_api_key("YOUR_API_KEY")
df <- nixtlar::electricity
fcst <- nixtlar::nixtla_client_historic(df, id_col="unique_id", level=c(80,95))

## End(Not run)

```

nixtla_client_plot *Plot the output of the following nixtla_client functions: forecast, historic, anomaly_detection, and cross_validation.*

Description

Plot the output of the following nixtla_client functions: forecast, historic, anomaly_detection, and cross_validation.

Usage

```
nixtla_client_plot(  
  df,  
  fcst = NULL,  
  h = NULL,  
  id_col = NULL,  
  time_col = "ds",  
  target_col = "y",  
  unique_ids = NULL,  
  max_insample_length = NULL,  
  plot_anomalies = FALSE  
)
```

Arguments

df	A tibble or a data frame with time series data (insample values).
fcst	A tibble or a data frame with the 'TimeGPT' point forecast and the prediction intervals (if available).
h	Forecast horizon.
id_col	Column that identifies each series.
time_col	Column that identifies each timestep.
target_col	Column that contains the target variable.
unique_ids	Time series to plot. If NULL (default), selection will be random.
max_insample_length	Max number of insample observations to be plotted.
plot_anomalies	Whether or not to plot anomalies.

Value

Plot with historical data and 'TimeGPT's output (if available).

Examples

```
## Not run:
nixtlar::nixtla_set_api_key("YOUR_API_KEY")
df <- nixtlar::electricity
fcst <- nixtlar::nixtla_client_forecast(df, h=8, id_col="unique_id", level=c(80,95))
nixtlar::timegpt_plot(df, fcst, h=8, id_col="unique_id")

## End(Not run)
```

nixtla_set_api_key *Set 'API' key in global environment*

Description

Set 'API' key in global environment

Usage

```
nixtla_set_api_key(api_key)
```

Arguments

api_key The user's 'API' key. Get yours here: <https://dashboard.nixtla.io/>

Value

A message indicating the 'API' key has been set in the global environment.

Examples

```
## Not run:
nixtlar::nixtla_set_api_key("YOUR_API_KEY")

## End(Not run)
```

```
nixtla_validate_api_key
    Validate 'API' key
```

Description

Validate 'API' key

Usage

```
nixtla_validate_api_key()
```

Value

A status code and a message indicating whether the 'API' key is valid.

Examples

```
## Not run:
nixtlar::nixtla_set_api_key("YOUR_API_KEY")
nixtlar::nixtla_validate_api_key

## End(Not run)
```

Index

* datasets

- electricity, [3](#)
- electricity_exo_vars, [4](#)
- electricity_future_exo_vars, [5](#)

* private

- .nixtla_client_historic_seq, [2](#)
- .nixtla_client_historic_seq, [2](#)

- electricity, [3](#)
- electricity_exo_vars, [4](#)
- electricity_future_exo_vars, [5](#)

- infer_frequency, [6](#)

- nixtla_client_cross_validation, [6](#)
- nixtla_client_detect_anomalies, [8](#)
- nixtla_client_forecast, [9](#)
- nixtla_client_historic, [10](#)
- nixtla_client_plot, [12](#)
- nixtla_set_api_key, [13](#)
- nixtla_validate_api_key, [14](#)