Package: transformerForecasting (via r-universe)

March 8, 2025

Type Package

Title Transformer Deep Learning Model for Time Series Forecasting

Version 0.1.0

Maintainer G H Harish Nayak <harishnayak626@gmail.com>

Description Time series forecasting faces challenges due to the non-stationarity, nonlinearity, and chaotic nature of the data. Traditional deep learning models like Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) process data sequentially but are inefficient for long sequences. To overcome the limitations of these models, we proposed a transformer-based deep learning architecture utilizing an attention mechanism for parallel processing, enhancing prediction accuracy and efficiency. This paper presents user-friendly code for the implementation of the proposed transformer-based deep learning architecture utilizing an attention mechanism for parallel processing. References: Nayak et al. (2024) <doi:10.1007/s40808-023-01944-7> and Nayak et al. (2024) <doi:10.1016/j.simpa.2024.100716>.

Imports ggplot2, keras, tensorflow, magrittr, reticulate (>= 1.20)

Suggests dplyr, knitr, lubridate, readr, rmarkdown, utils

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Author G H Harish Nayak [aut, cre], Md Wasi Alam [ths], B Samuel Naik [ctb], G Avinash [ctb], Kabilan S [ctb], Varshini B S [ctb], Mrinmoy Ray [ths], Rajeev Ranjan Kumar [ths]

Depends R (>= 4.0.0)

LazyData true

VignetteBuilder knitr

NeedsCompilation no

Date/Publication 2025-03-07 11:10:06 UTC

Config/pak/sysreqs libpng-dev python3

Repository https://harish11999.r-universe.dev

RemoteUrl https://github.com/cran/transformerForecasting

RemoteRef HEAD

RemoteSha 1b61751eb1131e422dd8766ec78466f283a295fb

Contents

`install_r_dependencies` *Install Package Dependencies*

Description

Installs Python dependencies (TensorFlow, Keras, Pandas) in a Conda environment for the transformerForecasting package.

Usage

```
install_r_dependencies(env_name = "r-reticulate")
```

Arguments

`env_name`	Character string specifying the Conda environment name (default: "r-reticulate").

Value

Invisibly returns NULL after attempting to install dependencies.

Examples

```
## Not run:
  install_r_dependencies()

## End(Not run)
```

S_P_500_Close *S&P 500's closing price data*

Description

The S&P 500's closing price as data provides an excellent real-world application. The S&P 500 index, a key benchmark for U.S. stock market performance, is vital for portfolio management, risk assessment, and market analysis. The original data frame contains 1340 rows and 2 variables. Here, we have taken only 200 rows and 2 variables for demonstration.

Usage

```
S_P_500_Close
```

Format

A data frame with 200 rows and 2 variables:

Date Formatted date

Price Numerical price values

TRANSFORMER *Transformer Model for Time Series Forecasting*

Description

Transformer model for time series forecasting

Usage

```
TRANSFORMER(
  df,
  study_variable,
  sequence_size = 10,
  head_size = 512,
  num_heads = 4,
  ff_dim = 4,
  num_transformer_blocks = 4,
  mlp_units = c(128),
  mlp_dropout = 0.4,
  dropout = 0.25,
  epochs = 300,
  batch_size = 64,
  patience = 10
)
```

Arguments

df	Input file
study_variable	The Study_Variable represents the primary variable of interest in the dataset, (Ex:Closing price)
sequence_size	Sequence size
head_size	Attention head size
num_heads	Number of attention heads
ff_dim	Size of feed-forward network
num_transformer_blocks	Number of transformer blocks
mlp_units	Units for MLP layers
mlp_dropout	Dropout rate for MLP
dropout	Dropout rate for transformer
epochs	Number of epochs
batch_size	Batch size
patience	Early stopping patience

Details

This function creates and trains a Transformer-based model for time series forecasting using the Keras library. It allows customization of key architectural parameters such as sequence size, attention head size, number of attention heads, feed-forward network dimensions, number of Transformer blocks, and MLP (multi-layer perceptron) configurations including units and dropout rates.

Before running this function, we advise the users to install Python in your system and create the virtual conda environment. Installation of the modules such as 'tensorflow', 'keras' and 'pandas' are necessary for this package. If the user does not know about these steps, they can use the `install_r_dependencies()` function which is available in this package.

The function begins by generating training sequences from the input data (`df`) based on the specified `sequence_size`. Sliding windows of input sequences are created as `x`, while the subsequent values in the series are used as targets (`y`).

The model architecture includes an input layer, followed by one or more Transformer encoder blocks, a global average pooling layer for feature aggregation, and MLP layers for further processing. The final output layer is designed for the forecasting task.

The model is compiled using the Adam optimizer and the mean squared error (MSE) loss function. Training is performed with the specified number of `epochs`, `batch_size`, and early stopping configured through the `patience` parameter. During training, 20% of the data is used for validation, and the best model weights are restored when validation performance stops improving.

The package requires a dataset with two columns: Date (formatted as dates) and the Close price (numerical). After loading the data and formatting it appropriately, the TRANSFORMER function trains a Transformer-based model to predict future closing prices. It outputs essential performance metrics like RMSE, MAPE, and sMAPE, along with visualizations such as training loss trends and an actual vs. predicted plot. These features make it an invaluable tool for understanding and forecasting stock market trends effectively..

Value

A list containing the following results:

- PREDICTIONS: The predicted values generated by the model.
- RMSE: Root Mean Squared Error, measuring the average magnitude of the prediction error.
- MAPE: Mean Absolute Percentage Error, representing the prediction accuracy as a percentage.
- MAE: Mean Absolute Error, showing the average absolute difference between actual and predicted values.
- MSE: Mean Squared Error, quantifying the average squared difference between actual and predicted values.
- sMAPE: Symmetric Mean Absolute Percentage Error, a variant of MAPE considering both over- and under-predictions.
- RRMSE: Relative Root Mean Squared Error, RMSE scaled by the mean of the actual values.
- Quantile_Loss: The quantile loss metric for probabilistic forecasting.
- Loss_plot: A ggplot object showing the loss curve over iterations or epochs.
- Actual_vs_Predicted: A ggplot object visualizing the comparison between actual and predicted values.

Examples

```
# Load sample data
data(S_P_500_Close)
df <- S_P_500_Close

# Run TRANSFORMER (will use mock results if Python is unavailable)
result <- TRANSFORMER(df = df,
  study_variable = "Price",
  sequence_size = 10,
  head_size = 128,
  num_heads = 8,
  ff_dim = 256,
  num_transformer_blocks = 4,
  mlp_units = c(128),
  mlp_dropout = 0.3,
  dropout = 0.2,
  epochs = 2,
  batch_size = 32,
  patience = 15
)

# Display results
result$PREDICTIONS
result$RMSE
result$MAE
result$MAPE
result$sMAPE
result$Quantile_Loss
# Plots are NULL if Python is unavailable
```

```
if (!is.null(result$Loss_plot)) result$Loss_plot
if (!is.null(result$Actual_vs_Predicted)) result$Actual_vs_Predicted
```

Index