# Documentation for Timeseries Modelling

## Data preparation

The previously cleaned dataset was prepared for machine learning by transforming the data frame from the wide format to the long format. This resulted in a data frame with period/time and countries as indices and government expenditure as a single column. The data was split into training and testing sets to enable evaluation of the machine learning to be used. The first 20 observations (years) were used as training data while the remaining 3 observations (years) were used as test data. The observations were not shuffled before splitting because of the sequential nature of time-series data

## Feature engineering

The data was further prepared for training by creating lagged features and single-value targets for machine learning using the window function of TensorFlow’s data API. Lagged features are simply the values of the target variable at previous time sets. Lagged features are obtained by shifting the target variable by a certain number of timesteps. For instance, if a target variable is measured in 2024, three-year-lagged features will be observations measured in 2021, 2022, and 2023. These features capture the relationship between historical data and present data and leverage that relationship in predicting future data. The dataset was evaluated on one, two, three, and four lagged features; optimal performance was obtained with three lagged features.

## Model Selection and Training

Due to the small size of the dataset, extensive model selection could not be done, so we selected an arbitrary three-layer deep neural network model with 10 neurons in the first and second layers and a single neuron in the output layer. Relu activation functions were used in the first and second layers while the final output layer used a linear function. Each model was trained for 100 epochs using the Adam optimizer at a learning rate of 0.001 and using mean square error as the loss function. The different models trained for each government using their specific country's data were stored in a Python dictionary for efficient retrieval when needed.

## Model evaluation

Each model was evaluated on the naive test data using mean absolute error and mean square error and all models fall within the range of 0.01 - 0.3.

## Predictions

A ‘plot\_forecast’ function was created to enable easy forecasting and graphing of forecasts.