A Comparison of Inflation Forecasting Models

Group Report for COMP5530

GEORGE BIGNALL, MUHAMMED MURAT KURMAZ, AYESHA RAHMAN, JAMES ZHANGLY, KEVIN RAFFAELLI, SANDRA GURAN, NATALIE LEUNG, and DR TIMON S. GUTLEB*, University of Leeds, United Kingdom

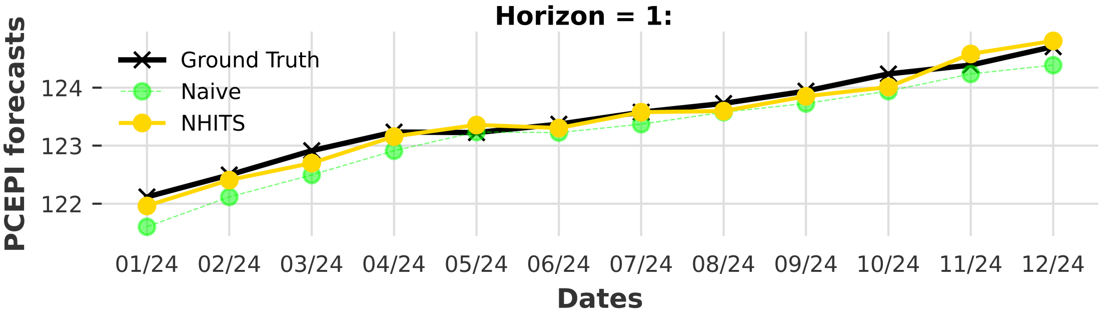


Fig. 1. Our best model's inflation forecasts against the ground truth and naive baseline for a one-month horizon.

Accurate inflation forecasting plays a critical role in guiding economic policy, investment strategies, and financial planning. This study evaluates a broad spectrum of forecasting models, ranging from classical statistical techniques (SARIMAX, ARDL) to machine learning models (Random Forest, XGBoost), and deep learning architectures (LSTM, RNN, TFT, TiDE, N-BEATSx, N-HiTS) to predict U.S. inflation, measured via the Personal Consumption Expenditure Price Index (PCEPI). Exogenous macroeconomic indicators were rigorously selected using Granger causality, cross-correlation, and cointegration testing. All models were trained and tested under a consistent walk-forward forecasting strategy across multiple horizons (1, 3, 6, 12 months). Empirical results demonstrate that N-HiTS consistently achieved the highest performance across all horizons, owing to its hierarchical interpolation, residual refinement, robust exogenous signal integration, and effective overfitting mitigation. Figure 1 shows some of these results. These findings highlight the effectiveness of modular neural forecasting frameworks in capturing non-linear, multi-scale temporal dependencies in macroeconomic time series.

Disclaimer: The code released in this project is purely academic and its output is not financial advice.

CCS Concepts: • **Computing methodologies** → **Machine learning**; **Supervised learning by regression**; *Modeling and simulation*; • **Applied computing** → **Economics**.

Additional Key Words and Phrases: Machine Learning, Supervised learning, Inflation, Inflation forecasting, Time series, Econometrics

1 INTRODUCTION

Inflation is defined as "the general increase in the price of goods and services in a country" [Collins 2020]. Inflation is closely monitored and can indicate the economic health of a country individually and globally. Changes in inflation affect wages, interest rates, and the purchasing power of households, businesses, and governments. High inflation (e.g. >5%) reduces purchasing power and makes long-term investments (and therefore planning) difficult. Low inflation (e.g. <1%) implies weak growth and potential future price reductions, leading people to not invest or hire. Central banks set inflation targets depending on predicted economic conditions; in the UK and US this figure is currently around 2% [Fed 2024] [BoE 2025].

Inflation measure values are calculated on a regular basis, e.g. monthly, and are usually compared between periods, where the most-used period of comparison is annual. One of the most-used inflation measures is the Personal Consumption Expenditure Price Index (PCEPI), which tracks the prices that people, or those buying on their behalf, pay for goods and services. Change in PCE captures inflation, and is the Federal Reserve's preferred measure of this phenomenon [Fed 2025]. Given the world-leading position of the US economy [Groningen Growth and Development Centre 2023], this paper specifically targets US PCEPI measures by the Fed.

Inflation forecasting is the act of predicting inflation. This is important to all parts of economy in planning spending and allocating resources appropriately. For governments, accurate inflation forecasts allow for appropriate policy prioritisation. For businesses, these forecasts can be used to manage investments and risks. For households, they could dictate savings, purchasing, investing, and salary negotiations. Traditionally, statistical and econometric models calibrated to historical data according to economic theory have been used for this task. However, advances in machine learning have provided other methods of inflation forecasting that have yet to be fully explored and compared.

Many models across three major categories are evaluated in this paper, comparing against a naive forecasting baseline, which assumes future values are the same as the most recent observation.

[*] Supervisor to the project

Authors' address: George Bignall, sc20geb@leeds.ac.uk; Muhammed Murat Kurmaz, sc21mmk@leeds.ac.uk; Ayesha Rahman, sc21ar@leeds.ac.uk; James Zhangly, sc20jzl@leeds.ac.uk; Kevin Raffaelli, sc21kgr@leeds.ac.uk; Sandra Guran, sc21samg@leeds.ac.uk; Natalie Leung, sc21nl@leeds.ac.uk; Dr Timon S. Gutleb, T.S.Gutleb@leeds.ac.uk, University of Leeds, Woodhouse, Leeds, West Yorkshire, United Kingdom, LS2 9BW.

The classical statistical models include Autoregressive Distributed Lag (ARDL), Vector Autoregression with eXogenous variables (VAR), the Seasonal AutoRegressive Moving Average model with eXogenous inputs (SARIMAX), Multiple Linear Regression (MLR), and Multivariate Adaptive Regression Splines (MARS). The machine learning models include decision tree-based methods such as Random Forest (RF) and eXtreme Gradient Boosting (XGBoost) and Multivariate Adaptive Regression Splines (MARS). Finally, the deep learning models include Recurrent Neural Network (RNN) methods such as the Elman RNN and variations like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models, Neural Basis Expansion Analysis for Time Series (N-BEATS) and its exogenous extension (N-BEATSx), Neural Hierarchical Interpolation for Time Series (N-HiTS), the Temporal Convolutional Network (TCN), and two new recent models: the Temporal Fusion Transformer (TFT) and Time-Series Dense Encoder (TiDE).

A mixture of exogenous variables, including global indicators and US-specific macroeconomic data, have been compiled to facilitate PCEPI predictions. These variables have been tested for Granger-causality with PCEPI, using Granger's causality test [Granger 1969] to remove variables which could negatively affect the forecasting performance of models. The models using exogenous data are therefore informed both of international trends and domestic factors. Sociological factors, however, such as public sentiment towards current economic performance, social trust, and cultural changes, are not included in this research. This paper compares popular methods to identify the most appropriate solutions for forecasting inflation over different time periods: from months (the smallest timescale at which data are available) to a year. These comparisons demonstrate the validity of current methods, the effectiveness of how they forecast inflation across time scales, and the accuracy of their predictions. See our GitHub repository [Bignall et al. 2025] for implementations and results.

2 BACKGROUND RESEARCH

Inflation forecasting has been a long-standing focus in macroeconomics [Manzan and Zerom 2013], with research evolving from classical statistical models to advanced machine learning and deep learning methods. Each approach introduces trade-offs between interpretability, predictive performance, and computational complexity. In this section, we review a range of models —both conventional and state-of-the-art —used in our project, situating them within the broader research landscape.

2.1 Classical Statistical Models

Traditional econometric models have served as foundational tools in inflation forecasting due to their interpretability and grounding in economic theory. ARIMA (AutoRegressive Integrated Moving Average) [Box et al. 2015] is widely applied for univariate time series forecasting, relying on differencing to achieve stationarity and autoregression to model temporal dynamics. Extensions to ARIMA aim to improve predictive accuracy when seasonality and exogenous variables are introduced, up to and including SARIMAX (Seasonal ARIMA with eXogenous regressors) [Korstanje 2021].

Vector AutoRegression (VAR) models [Eric Zivot 2006] generalise univariate AR models to handle multivariate time series, capturing the joint dynamics between inflation and related economic indicators [Fritzer et al. 2002] among time series. ARDL (AutoRegressive Distributed Lag) (implemented by [Seabold and Perktold 2010]) models provide a framework for combining short- and long-run effects, making them suitable for economic scenarios where variables are a mix of stationary and non-stationary series [Osman et al. 2019].

While these models offer theoretical rigour and transparency, they are constrained by linear assumptions and often struggle with capturing non-linear dynamics, structural breaks, or persistent dependencies in economic data [Bontempi et al. 2013; Ülke et al. 2018].

2.2 Machine Learning Models

To overcome the limitations of linear models, machine learning methods have gained popularity in recent years. These models excel in capturing complex, non-linear patterns without explicit assumptions about data generation [Parmezan et al. 2019]. Random Forest [Ho 1995] and XGBoost [Chen and Guestrin 2016] are ensemble-based methods that aggregate decision trees to improve robustness and predictive power. While Random Forests reduce variance through bagging, XGBoost applies boosting to sequentially correct errors [Mason et al. 2000], often achieving higher performance on structured datasets at the cost of parallelisability.

MARS (Multivariate Adaptive Regression Splines) [Friedman 1991] offers another flexible approach by fitting piecewise linear regressions to optimal segments of the data to produce a continuous model, allowing for the modelling of almost-additive relationships and the interactions between variables [Kao et al. 2013].

Despite their strengths, many machine learning models treat time as a regular input feature and may lack built-in mechanisms to model temporal dependencies effectively [Bontempi et al. 2013].

2.3 Deep Learning Models

Deep learning models, particularly those designed for sequence data, offer powerful tools for modelling the temporal structure of inflation [Lim and Zohren 2021]. Recurrent Neural Networks (RNNs) [Elman 1990], and their variants, GRU (Gated Recurrent Unit) [Cho et al. 2014] and LSTM (Long Short-Term Memory) [Hochreiter and Schmidhuber 1997], have become standard in time series forecasting due to their ability to retain and update temporal information. However, they often suffer from long training times and vanishing gradient issues in the long run [Bengio et al. 1994].

Transformer-based models address some of these issues with self-attention mechanisms that model long-term dependencies more efficiently. General-purpose architectures such as the vanilla Transformer [Bengio et al. 1994] and time-series–specific versions such as the Temporal Fusion Transformer (TFT) [Lim et al. 2020] and Temporal Convolutional Network (TCN) [Bai et al. 2018] have shown strong performance in financial forecasting by incorporating recurrent and self-attention layers and convolutional layers respectively for salience-based feature selection, long-term dependency modelling, and interpretability [Lim et al. 2021].

Recent developments have shown that simpler architectures can outperform transformer-based models. The Time-Series Dense Encoder (TiDE) [Das et al. 2024] utilises multi-layer perceptrons with a encoder-decoder architecture. This allows the model to handle covariates and non-linear dependencies.

While classical machine learning and early deep learning models improved forecasting accuracy, they often lacked interpretability and struggled with integrating exogenous information or capturing multi-resolution trends. This motivated the development of architectures like N-BEATS (Neural Basis Expansion Analysis for Time Series) [Oreshkin et al. 2020], which uses residual blocks to decompose time series into trend and seasonal components. Extensions like N-BEATSx [Olivares et al. 2023] and N-HiTS [Challu et al. 2022a] further improve performance by integrating exogenous variables and applying hierarchical interpolation with multi-rate sampling.

3 METHOD

3.1 Exploratory Data Analysis (EDA)

3.1.1 Data Sourcing and Preprocessing. The data mainly comprises resources from fred.stlouisfed.org, including the target variable PCEPI, but also includes data from 8 other freely available sources. See appendix section C for the full list of citations for the data used. Although PCEPI is monthly data, many other insightful exogenous variables are published at different frequencies, such as daily, monthly or even quarterly. To make use of these, daily and weekly data were aggregated into one monthly value, while quarterly data was linearly interpolated. Other missing values were also interpolated or simply removed if there were too many. Data leakage to the test set was carefully avoided throughout preprocessing, ensuring a fair test when evaluating the models.

To ensure there exists a causal relationship between the exogenous variables and PCEPI, Granger's causality test [Granger 1969] was performed with a significance level of 5%, after the time series were first differenced to achieve stationarity. Exogenous variables that did not have statistically significant Granger-causation were simply removed.

3.1.2 Lag Structure Analysis using ACF and PACF. To better understand the temporal dependencies in the inflation time series (PCEPI), we performed a detailed analysis using the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF). Autocorrelation is the linear dependence of a variable between itself and its past values over different time intervals (lags) [MathWorks 2025]. These results helped us to determine how many lag terms should be used in the autoregressive components of our forecasting models.

We evaluated the persistence of autocorrelation by plotting the ACF [Mathuranathan 2014] of the PCEPI series using 30 lags. We also computed the ACF up to 30 lags to identify the significant seasonal lags where the autocorrelation coefficient $\rho \in [-1, 1]$ exceeds the threshold of 0.5. The potential seasonal lags found are 7, 10 and 15. This plot provided insight into the strength and duration of autocorrelation within the series. Moreover, we applied the Ljung–Box Q-test at lags 7, 10, and 15 to statistically test the presence of autocorrelation [MathWorks 2023]. The choices of these lags was guided by the ACF anaylsis, as they showed significant autocorrelatio with the time series itself. The Ljung-Box p-value was the lowest at lag 15, approximately 7.77×10^{-74} ; although not exactly zero, this still indicated strong autocorrelation within the series at this lag. Insignificantly autocorrelated lags could then be ignored in later model development.

The PACF was further used to identify significant autocorrelation lags without influence from shorter lags. Partial autocorrelation at lag h, denoted $\phi_{h,h}$, represents the correlation between y_t and y_{t-h} after removing any linear dependence on the shorter lags $y_1, y_2, \ldots, y_{t-h+1}$ [MathWorks 2025].

A custom function was implemented to compute PACF values using the Yule-Walker method. The Yule-Walker equations estimate the parameters of an autoregressive model Eshel [2003]. They were chosen for the PACF due to their efficiency and suitability for stationary linear time series (each of our time series were made stationary before Granger causality testing, see above). The function identified the first lag where the PAC value fell within the 95% confidence interval, showing no significance. This cutoff lag was selected as the optimal lag order for future autoregressive modelling. A PACF graph was also generated for visualisation.

Cointegration test Cointegration is when two or more non-stationary series share a stable long-term relationship. We conducted the Johansen cointegration test to assess long-term equilibrium relationships [Gao et al. 2018] between inflation (measured by PCEPI) and the exogenous variables [Gianfreda et al. 2023]. Before testing, we used the Augmented Dickey-Fuller (ADF) [Dickey and and 1979] test to exclude stationary variables to ensure our testing was valid. Only exogenous variables which failed to reject the ADF null hypothesis with a significance level of 5% were retained for pairwise cointegration testing with PCEPI.

A Johansen test was then run for each qualifying pair with a deterministic trend assumption and 1 lag difference, using the trace statistic to determine cointegration. Using a significance level of 5%, statistically significant pairs were considered cointegrated. The resulting list of cointegrated exogenous variables were saved for further use in model development. Models such as Vector Autoregression (VAR) and ARIMAX could benefit from incorporating cointegrated inputs due to their ability to provide short- and long-term parameters [Ali 2015].

3.1.3 Pairwise Variance and Linearity Analysis. We assessed each numerical predictor's association with PCEPI by binning its values into Low/Medium/High tertiles and running one-way ANOVAs ($\alpha = 0.05$) to test for mean differences [Kim 2014]. Nearly all time series (97/137) rejected the null hypothesis (e.g. Household Energy Index: $F = 1467.4, p < 10^{-175}$), showing systematic PCEPI variation across bins. Significant predictors then underwent Tukey's HSD, revealing, for example, that High vs. Low energy prices differ by −31.54 ($p < 0.001$) and High vs. Medium by −16.08 ($p < 0.001$); overall, 125 series exhibited at least one significant contrast [Allen 2017]. Finally, for each significant series we computed cross-correlations with PCEPI over lags −6 to +6, guiding lag selection and autoregressive order in the forecasting models of Section 3.4 [Duan and Stanley 2011].

3.2 Forecasting Strategy

The models were trained to optimise the following forecast horizons in months: 1,3,6,12, with the exception of some statistical models, since changing the forecast horizon for those models would inherently mean changing the nature of the data itself (from monthly to quarterly, for example) and would not be a fair test. Hyperparameter optimisation was used for all models, but the method of optimisation varied for each model as they have different hyperparameter search spaces. It is worth noting that in practice, 12 months is uncommon for forecasting inflation and will most likely be past the model's expiration date. However, as a group, we chose to include a 12-month horizon to observe how forecast errors propagate over time and to evaluate how different models begin to break down in long-term prediction horizons.

3.3 Statistical Models

3.3.1 AutoRegressive Distributed Lag (ARDL). We implemented an ARDL forecasting model using Ridge regression with exogenous variables engineered from macroeconomic data. The target series, PCEPI, was first log-transformed to stabilise variance. The input features were augmented with time-based encodings, momentum, rolling statistics (mean, standard deviation, skew, and kurtosis), and lagged values of both the target and exogenous variables.

To reduce dimensionality and address multicollinearity, we selected the top 30 exogenous variables using cross-correlation with the log-transformed target, ensuring each input had measurable predictive relevance over multiple lags. Principal component analysis was then applied to decorrelate and compress the most informative exogenous signals, making the model more robust to multicollinearity - a known issue in economic data. Ridge regression was chosen for its ability to handle many correlated predictors while applying regularisation, improving generalisation on unseen data.

The model followed the same walk-forward approach as defined by the project model requirements, forecasting up to 12 months ahead. Rather than optimising for a single snapshot, this approach evaluates how the model performs over varying horizons.

3.3.2 Vector Autoregression (VAR). We implemented two Vector Autoregression (VAR) models (VARccf and VARcointegration) to forecast the U.S. inflation target variable (PCEPI) over our four chosen forecast horizons. The difference between VARccf and VARcointegration is a matter of either using cross-correlation or cointegration as a means of feature selection, where the most correlated or cointegrated features were used. Both models share the same architecture and processing pipeline but differ in the selection of exogenous variables. The original VAR model was implemented using the `statsmodels.tsa.api` module in Python [statsmodels contributors 2025]. VARs capture linear interdependencies among several time series by modelling each variable as a linear function of its past value and the past values of the other values [University 2024]. This feature enabled the modelling of the inflation index (PCEPI) with a selected set of macroeconomic variables.

VARccf uses a pre-ranked list of exogenous variables that are cross-correlated with PCEPI, while VARcointegration uses a pre-ranked list of exogenous variables selected based on long-term relationships with inflation using cointegration, improving stability and robustness, especially over longer horizons. Both models use standardised data, a lag order 1 based on the most significant lags of the exogenous variables, and a greedy feature selection process based on root mean squared error (RMSE). Final predictions for both models were generated over 12 months and subsampled according to the forecast horizons.

3.3.3 AutoRegressive Integrated Moving average models. SARIMA, ARIMAX and SARIMAX are all based on the ARIMA model, which consists of three components: the AutoRegressive model, integration, and the Moving Average model [Box et al. 2015]. An autoregressive model (AR) uses p previous values (lags) of the endogenous variable to predict the next value, assuming future values are a function of previous values - a more reliable assumption if the time series is stationary [Box et al. 2015]. A Moving Average model (MA) is similar, instead of regressing q previous errors of the model, assuming the data's noise is stationary. These two models can be combined to form the ARMA model by summing their previous value and error terms, assuming that the time series is stationary. Transformation by integration can be performed to make the series stationary when using ARMA for non-stationary time series. This involves simply differencing the time series with a lag order d to achieve stationarity. Thus, the model name is ARIMA [Box et al. 2015].

Seasonal ARIMA (SARIMA) is similar to ARIMA but accounts for seasonality in the time series by effectively using another ARIMA model to regress the seasonality, using the lags one season prior to the next prediction [Box et al. 2015]. The parameters of this seasonal-regressing ARIMA model are described as P, D, and Q, which each represent the same concepts as p (number of autoregressive terms), d (number of differences needed for stationarity), and q (number of lagged forecast errors), respectively, but lagged by one season.

ARIMAX and SARIMAX are each similar to their ARIMA and SARIMA counterparts but include eXogenous variables to enhance forecasting power [Box et al. 2015]. This is achieved by adding linear regression terms for the exogenous variables to the model.

ARIMA, SARIMA, ARIMAX and SARIMAX were all implemented using `statsmodels` and `pmdarima`. Their optimal hyperparameters: p, q, P, and Q were obtained by performing the step-wise algorithm [Hyndman and Khandakar 2008], essentially a local search of the model's hyperparameter space. The differencing order d and D were obtained by differencing the data at several lags and performing statistical tests such as ADF [Dickey and and 1979] and KPSS [Kwiatkowski et al. 1992] to determine stationarity.

ARIMAX and SARIMAX are very sensitive to the number of exogenous variables, where too many exogenous variables can negatively affect the model. Therefore, a grid search was performed to determine the optimal number of exogenous variables, n^*. This was achieved by first calculating the maximum cross-correlation of each exogenous variable with PCEPI for 24 lags - an upper bound past which cross-correlation values were negligible - which were then sorted in descending order. The grid search was then performed by training and validating the model on a range of n values for differing lists of top-n variables to determine the optimal $n = n^*$ that minimises prediction error on the validation set. The model was then re-trained on the training data using the top n^* exogenous variables by cross-correlation with the target to produce the final model.

3.4 Machine Learning Models

3.4.1 Random Forest. Random Forest (RFX) can capture complex, non-linear response surfaces while remaining robust to multicollinearity and outlier—advantages over single decision trees or linear econometric models [Ho 1995]. Its built-in bagging mechanism reduces variance without requiring stationary assumptions or specific residual distributions, making it well-suited to the inflation's noisy and regime-shifting nature.

Monthly PCEPI forecasting is framed as a regression task on the log-difference of the index, $\Delta \log(\text{PCEPI})$, to stabilise variance and yield an interpretable month-over-month inflation rate—aligning with the assumption of identically distributed residuals in RFX.

The target is regressed against a feature matrix composed of the past twelve values of $\Delta \log(\text{PCEPI})$ (capturing annual seasonality) and the ten strongest exogenous predictors, selected through a three-stage filtering process: ANOVA → Tukey HSD → maximum cross-correlation. These macroeconomic drivers—including GDP momentum, energy prices, money supply, PPI, and supply chain stress, also lagged over twelve months to retain temporal structure.

To preserve strict temporal integrity, any missing values from lagging were first forward-filled using only information available at each time step (i.e. the last observed value from the 1990–2023 training window, avoiding look-ahead bias), then any remaining gaps were median-imputed. All preprocessing and modelling steps were chained into a single scikit-learn pipeline, chosen for its modularity and reproducibility, fold-wise transformations, which culminate in a Random Forest regressor.

Hyperparameter tuning was carried out with Optuna's TPE over 40 trials, each assessed via a five-fold TimeSeriesSplit to ensure strictly past-only training. We explored 200–1000 trees (more trees reduce variance at higher compute cost), max depths of 3–20 or unlimited (shallow depths guard against overfitting, deeper ones capture complexity), 1–15 minimum samples per leaf (larger values smooth noisy splits), and feature subsampling from "sqrt" and "log2" to 30–100% (varying randomness to decorrelate trees). RMSE was minimised, yielding about 327 trees, depth 13, and leaf size 6, which were then retrained on the full 1990–2023 dataset.

During the 2024 walk-forward evaluation, each month's lagged feature vector produced a one-step prediction of $\Delta \log$, which was exponentiated and compounded to derive the level forecast for month $t + 1$. The process was repeated for horizons 1, 3, 6, and 12 months, and all forecasts were serialised for downstream analysis.

3.4.2 Extreme Gradient Boosting (XGBoost). We implemented an XGBoost-based forecasting model using the `XGBRegressor` class from the `xgboost` Python package [Chen and Guestrin 2016]. Following the neural network models, this model was trained on lagged inflation values and engineered exogenous features, including rolling statistical summaries and seasonal encodings. The forecasting target, the next-step inflation value, had input features standardised using z-scores.

Hyperparameter optimisation was performed independently for each forecast horizon using the Optuna framework. The search space included tree depth (3–6), learning rate (0.01–0.3), number of estimators (100–200), and subsampling parameters such as `subsample` and `colsample_bytree`. Each Optuna trial trained a model instance on the training set and evaluated performance using root mean squared error (RMSE) on a held-out validation set. The best-performing configuration was then used to train a final model for each horizon.

XGBoost frames forecasting as a regression problem over fixed-lag tabular inputs. To enhance forecasting accuracy, outputs from trained RNN-based models were also included as additional horizon-specific features. RMSE loss curves per iteration were also saved to visualise convergence during training.

3.4.3 Multivariate Adaptive Regression Splines. MARS is a machine learning model in the sense that the optimal cut-points around which linear functions are hinged are determined automatically through a search of candidate cut-points (knots) that minimises the linear functions' fit quality metric (often MSE) [Friedman 1991]. Otherwise, its simple hinge-based piecewise linear regression nature means, like many statistical approaches, this model is susceptible to exogenous data. After attempting to fit the model on various combinations of cross-correlation ranked exogenous variables, the simplest method of eliminating all exogenous variables and simply relying on an integer ordinal encoding of the dates to regress the target PCEPI variable was found most effective. Linear regression models like this are also invariant to data scaling; none was performed.

The MARS model was, therefore, fitted on the raw ordinally-encoded date series to predict the PCEPI series aligned with these dates. A generalised cross-validation procedure was then undertaken internally to find the most optimal knots to hinge its linear regressions, thereby establishing the best-fitting piecewise linear regressions over the whole training dataset. The final linear regression function of the model was then extrapolated to give predictions for the test data - differing horizons do not alter the values extrapolated.

3.5 Deep Learning Models

Our deep learning methods focus on sequence models that are tailored to capture long-range temporal dependencies and integrate exogenous inputs for enhanced inflation forecasting. We implemented and compared the performance of more traditional Recurrent Neural Networks (RNNs), Gated Recurrent Units (GRUs), and Long Short-Term Memory (LSTM) networks with approaches specialised for time-series forecasting including TFT (Temporal Fusion Transformers), TiDE (Time series Dense Encoders), Temporal Convolutional Networks (TCNs), and the neural basis expansion analysis approaches N-BEATSx and N-HiTS.

3.5.1 Recurrent Neural Network (RNN). We implemented a Recurrent Neural Network (RNN) model in PyTorch [Paszke 2019] as an initial deep learning approach for forecasting inflation from multivariate time series data. The input data included engineered features such as lagged inflation values, rolling statistics (mean and standard deviation), and seasonal encodings generated via sine and cosine transformations. The sequences were constructed using 12-month rolling windows. All input features were standardised using z-scores prior to training.

Hyperparameter tuning was conducted for each horizon using the Optuna framework [Akiba et al. 2019], exploring hidden sizes between 32 and 512 units, 1 to 6 recurrent layers, and learning rates in the range of 1e−5 to 1e−1. Each Optuna trial trained the model for 10 epochs, and early stopping was enabled using trial pruning. After selecting the best-performing configuration in validation error, the final model was retrained for 50 full epochs on the training set.

The RNN was implemented in batch-first mode and optimised using the Adam optimiser with a mean squared error (MSE) loss function. The final hidden state was passed through a fully connected layer to output a single forecast. As a standard RNN, the architecture lacks gating mechanisms, making it more susceptible to vanishing gradients and less effective at modelling long-range dependencies compared to variants such as GRUs and LSTMs. Predictions were inverse-transformed to the original inflation scale using the fitted scaler for evaluation and interpretation.

3.5.2 Gated Recurrent Unit (GRU). We implemented a Gated Recurrent Unit (GRU) model in PyTorch [Paszke 2019] for forecasting inflation, following the same multivariate preprocessing pipeline and training configuration described in Section 3.5.1. Inputs consisted of lag features, rolling statistics, and seasonal time encodings. All features were standardised using z-scores.

The GRU architecture used the same Optuna-based hyperparameter tuning procedure [Akiba et al. 2019] as the RNN model. This included training each trial configuration for 10 epochs with early pruning, followed by retraining the best configuration for 50 epochs, extracting the best weights in terms of validation loss from this process. The search space remained consistent, encompassing hidden units, number of layers, and learning rate.

GRUs introduce reset and update gates that regulate information flow, mitigating vanishing gradients and improving the modelling of longer temporal dependencies. Like the RNN, the GRU model was trained in batch-first mode using the Adam optimiser and an MSE loss function. Forecasts were generated by passing the final hidden state through a dense layer. The resulting predictions were then inverse-transformed to the original inflation values using the fitted scaler, ensuring interpretability and comparability.

3.5.3 Long Short-Term Memory (LSTM). We implemented a deep learning model using the LSTM neural network in PyTorch [Paszke 2019] to process multivariate input sequences over time. The LSTM architecture consists of two stacked layers with a hidden size of 128 units per layer. A dropout rate of 0.3 was applied between layers to prevent overfitting. The model supports bidirectional LSTM layers to capture both forward and backward temporal patterns. The LSTM was set to batch-first mode, allowing the model to efficiently learn from multiple sequences in parallel and speed up the training time.

Following the LSTM layers, the final hidden state from the last time step is passed through layer normalisation and a feed-forward neural network. The feed-forward network includes a linear transformation, ReLU activation, additional dropout for regularisation, and a linear layer that maps the hidden representation to a 12-step output vector. This setup allowed the model to forecast inflation for 12 months. Normalisation and dropout also improved the model's robustness and stabilisation.

Hyperparameter tuning using the Optuna framework [Akiba et al. 2019] was applied across our four forecast horizons to improve performance. The model was trained on 36-step input sequences using a direct multi-step delta forecasting approach. After training, predictions were rescaled to the original inflation values using inverse transformations, giving reliable forecasts across all horizons.

3.5.4 Temporal Fusion Transformer. The Temporal Fusion Transformer (TFT) was initially designed to tackle multiple problems which often arise in time-series forecasting, namely handling different input types (static covariates, exogenous time series, and known future inputs), variable selection and interpretability of the use of inputs [Lim et al. 2020]. There are no static covariates for our inflation forecasting dataset, so those parts of the TFT architecture dedicated to handling them are not used.

TFT introduces the Gated Residual Network (GRN), which is used to select relevant features for forecasting, suppress irrelevant features, and unwanted parts of the architecture [Lim et al. 2020]. The main component of a GRN is the Gated Linear Unit (GLU), which uses the sigmoid activation function to suppress unwanted parts of the architecture that are not required for a given dataset (e.g. in this instance, aspects of the architecture relating to static covariates such as the static enrichment layer). This allows the model to remove noisy inputs, which negatively impact performance [Lim et al. 2020]. TFT also uses an LSTM encoder-decoder to leverage local context by identifying points of significance and recurring patterns to increase model performance and a self-attention transformer to learn long-term dependencies.

The TFT models were developed using the Darts library [Herzen et al. 2022] for Python. Since TFT has variable selection built into its architecture, these models were trained on the whole training dataset (the monthly and quarterly data were also inserted as future covariates) and the values were scaled using a min-max scaler. A hyperparameter search was performed using Optuna with 150 trials for each forecasting horizon on the following search space: input size (number of lags to use) $[1, 36]$, number of LSTM layers $[1, 32]$, number of attention heads $[2, 4]$, hidden size $[1, 32]$, dropout $[0, 0.5]$, learning rate $[1\times10^{-5}, 1\times10^{-1}]$ and loss function (quantile regression for probabilistic output modes or MSE). Once the optimal parameters were found, the models were re-trained for each forecasting horizon using the optimal parameters and early stopping, with patience of 25 and tolerance of 1×10^{-5}.

3.5.5 TiDE. The Darts library's [Herzen et al. 2022] TiDE implementation was used, providing a unified framework for training and evaluating a wide range of time series forecasting models.

The raw input exogenous variables were scaled using a robust scalar (scaling according to the interquartile range), suitable for handling unseen or extreme values in future data. To improve accuracy, these variables were also ranked using their cross-correlation with the target, and the number of highest-ranked exogenous variables used as input was treated as a tunable hyperparameter.

Separate TiDE models were trained and optimised for each forecasting horizon using a rolling window on the test set. The following hyperparameters were optimised: the number of past time steps fed into the model, the number of layers in the encoder and decoder, the number of hidden layers, the dropout rate for regularisation, the learning rate and its scheduling value γ, and whether to apply reversible instance normalisation. This search aimed to minimise validation loss for each horizon, maximising the likelihood of producing a generalisable model.

3.5.6 The Temporal Convolutional Network (TCN). TCNs extend Convolutional Neural Networks (CNNs) to handle relationships through time [Lea et al. 2016]. They capture hierarchical relationships through a range of timescales by layering temporal convolutional (TC), max pooling, and normalisation layers, minimising the vanishing-gradient myopia problems presented by pure RNNs and similar networks and ensuring even receptive field coverage across timescale dependency periods. As has been done in this case, the architecture can be extended to convolve over arbitrary numbers of input features, regardless of the modality.

Similarly to the TFT and TiDE models mentioned above, the Darts implementation of Temporal Convolutional Networks [Herzen et al. 2022] was used here. The TCN models were initially each provided with the raw exogenous and target variables to establish their efficacy, and were later provided with min-max scaled features, slightly improving performance. Further feature engineering (such as adding additional seasonal features or Fourier-transforming features) was attempted, but did not improve the network's performance (it seems to be able to capture these relationships through kernel convolution) and so were not included in the final model.

TCNs function largely like a typical CNN - convolving across time instead of space. As such, its hyperparameter optimisation search space consisted of a variety of kernel sizes, numbers of TC layers, numbers of kernel filters per TC layer, and initial dilation sizes, along with more general considerations like whether weights were normalised, the dropout percentage used for regularisation, and the learning rate used in training. For each horizon, the size of the validation set used in establishing optimal hyperparameters was set to 3 months longer than the input size of the model such that rolling forecasts were all validated fairly an equal number of times.

3.5.7 N-BEATSx and N-HiTS (NeuralForecast). N-BEATSx and N-HiTS (Neural Hierarchical Interpolation for Time Series forecasting) are deep learning forecasting architectures designed for time series prediction. The open-source `NeuralForecast` library's implementations of both the N-BEATSx and N-HiTS models, developed alongside the official N-HiTS architecture publication [Challu et al. 2022a], were used. In addition to model implementations, `NeuralForecast` provides utilities for handling multi-horizon forecasting, exogenous variables, and walk-forward evaluation strategies, aligning closely with the requirements of this project [Challu et al. 2022b, 2024].

N-BEATSx extends the original N-BEATS model, introducing support for exogenous variables and improved basis expansion techniques [Olivares et al. 2023]. The N-BEATS framework predicts future values using a sequence of fully connected neural networks, structured into stacks and blocks, where each block learns components of the time series such as trend, seasonality, or other residual patterns. N-BEATSx enhances this by allowing historical and future exogenous variables to be incorporated into the forecast, improving performance when external factors influence the target series.

N-HiTS, on the other hand, builds on hierarchical interpolation and temporal sampling concepts to further enhance multistep forecasting performance [Challu et al. 2022a] by first producing a low-resolution forecast, then iteratively refining it through higher-frequency reconstructions. The model structure is organised into multiple stacks, each composed of multilayer perceptrons (MLPs) predicting scales/granularities of temporal components through `MaxPool` operations at varying kernel sizes to subsample the input, controlling the frequency band it specialises in for smooth interpolated forecasts allowing the architecture to naturally decompose inflation signals into trend and residual components—aligning well with economic patterns like seasonality, momentum, and shocks.

Both models were trained using augmented exogenous variables with lag-based inflation patterns, time encodings, and seasonal Fourier features along with statistical information like momentum (change in percentage growth), rolling means, standard deviations, skewness, and kurtosis to capture historical dependencies and volatility patterns. These were selected using correlation-based methods and standardised prior to training, ensuring only the most informative variables were retained.

Each model was trained using the rolling window approach used by all the models to simulate realistic forecasting scenarios. The model architecture of N-BEATSx consists of three stacks with four blocks each. When predicting a one-month horizon, the stack types were limited to identity blocks only to satisfy architectural constraints, while for horizons 3, 6, and 12, trend, seasonality, and identity stacks were used. For N-HiTS, three stacks of two blocks each were implemented. Both models used Symmetric Mean Absolute Percentage Error (SMAPE) as the loss function, dropout regularisation, and robust internal scaling mechanisms. Early stopping based on validation loss was used to prevent overfitting. Final predictions were inverse-transformed and evaluated using common forecast metrics, allowing direct comparison across all models.

4 EVALUATION

The evaluation aimed to assess the forecasting performance of each model across multiple forecast horizons (1, 3, 6, and 12 months) over 2024. Standard error metrics were combined with statistical hypothesis testing to determine whether models significantly outperformed a naive baseline. Each horizon was evaluated separately.

4.1 Results

Forecast accuracy was assessed using three standard error metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the coefficient of determination (R^2). These metrics offer a comprehensive view of model precision and comparative performance. Visual summaries are presented in Figures 2, 3, 4, 5.

To complement the error analysis, the Diebold-Mariano (DM) test [Diebold and Mariano 1994] was used to statistically compare each model against the naive forecast. The DM test was chosen over traditional paired t-tests as it accounts for autocorrelation in forecast errors, a common property in inflation time series. The Harvey-Leybourne-Newbold correction [Harvey et al. 1997] was applied to address small sample bias. For each horizon, if multiple models were significantly better than the naive baseline, the model with the lowest MAE was considered best.

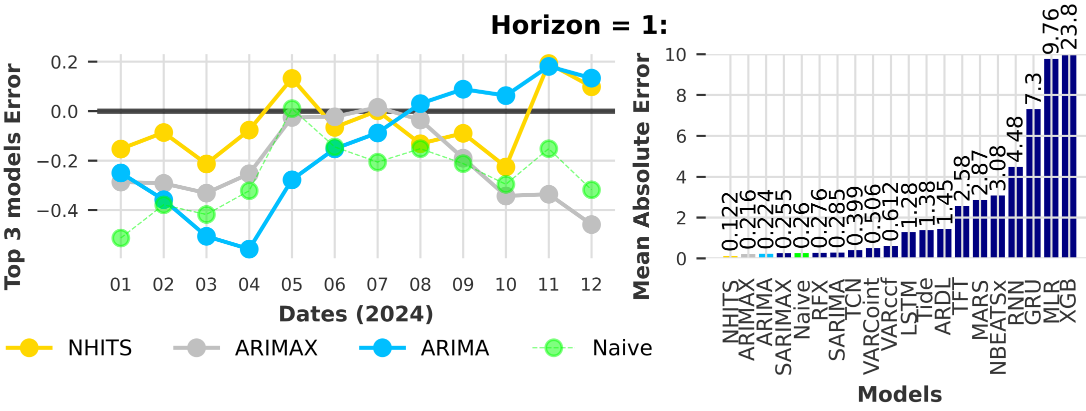


Fig. 2. The top 3 performing models for a forecast horizon of 1 and their associated mean absolute errors.

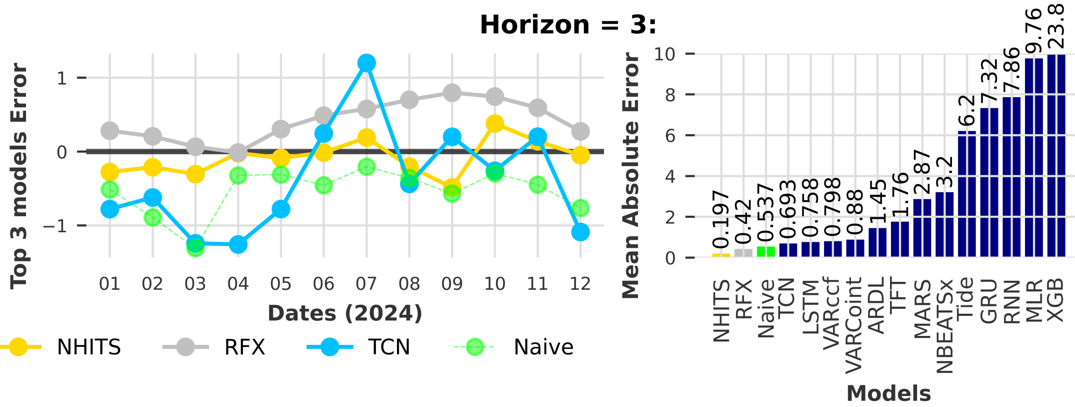


Fig. 3. The top 3 performing models for a forecast horizon of 3 and and their associated mean absolute errors.

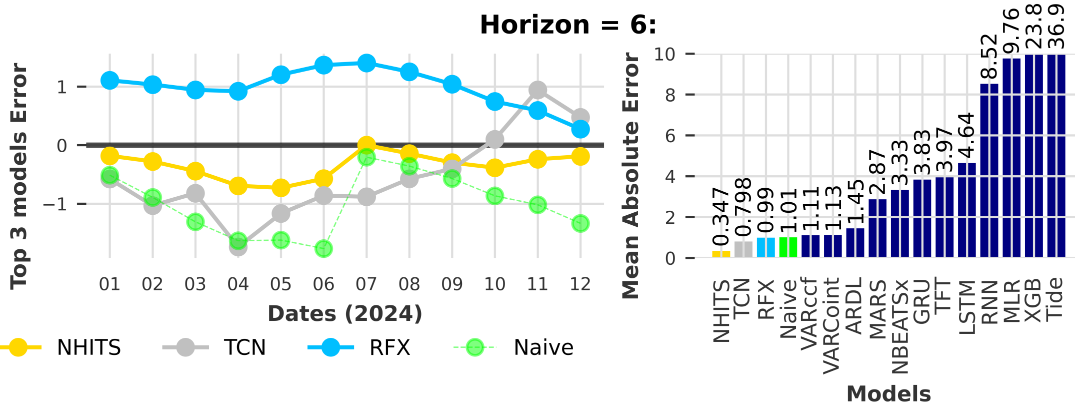


Fig. 4. The top 3 performing models for a forecast horizon of 6 and their associated mean absolute errors.

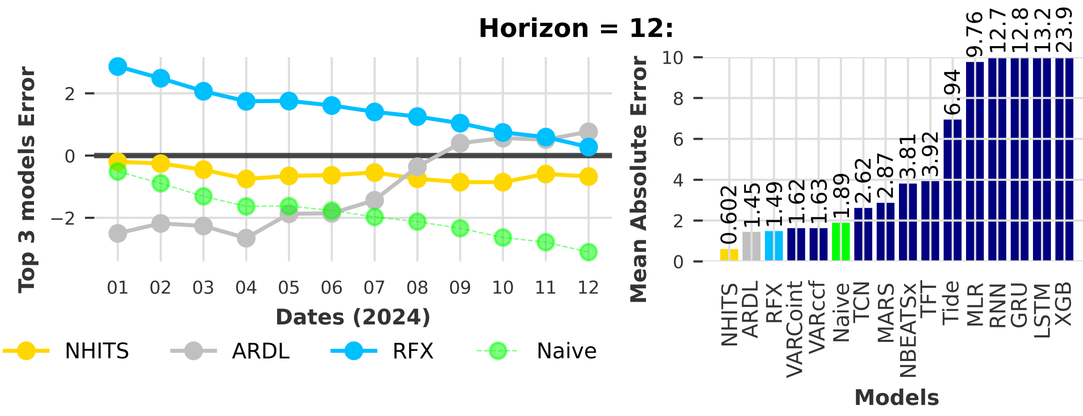


Fig. 5. The top 4 performing models for a forecast horizon of 12 and and their associated mean absolute errors.

N-HiTS was the only model which made a statistically significant improvement over the baseline results across all forecast horizons. At the 12-month horizon, both VARccf and VARcointegration model variant was also significantly better than the naive baseline. However, N-HiTS clearly demonstrated superior predictive accuracy here too (see Figure 5).

5 DISCUSSION

This study evaluated a wide range of models—from classical econometric approaches to modern deep learning architectures—for multi-horizon inflation forecasting. As shown in Figures 2 through 5, N-HiTS consistently outperformed other models across all horizons, as evidenced by both forecast accuracy and statistical significance using the Diebold-Mariano test. Its performance reflects an effective combination of design principles that were either missing or only partially implemented in competing models.

5.1 SHORT TERM FORECASTING (Horizons 1 and 3)

Localised Temporal Attention: In the short term, N-HiTS was the most effective at capturing recent inflation dynamics. It was the only model that maintained both high precision and low variance at 1- and 3-month horizons. Classical models such as ARIMAX and SARIMAX also performed reasonably well at Horizon 1, benefiting from their ability to model autoregressive relationships and include exogenous inputs.

Neural models based on sequential memory, such as LSTM, GRU, and RNN, struggled to remain consistent even in the short term. Although designed to track temporal dependencies, these models often overfit to recent data patterns and lacked the robustness to handle even moderate shifts in inflation behaviour. Meanwhile, TFT and TiDE, which used attention-based mechanisms to capture time-varying influences, failed to translate their design into reliable short-term predictions, likely due to data sensitivity and overparameterisation.

Multi-Resolution Forecasting: N-HiTS's short-term accuracy can be partly attributed to its ability to process patterns at multiple time scales. Its internal pooling mechanisms enable different layers to specialise in short-term movements and longer-term structural changes simultaneously. In contrast, models like TCN—which also rely on temporal hierarchies through dilated convolutions—showed some early promise but ultimately lacked the adaptability and refinement capacity to maintain stable outputs over all horizons.

Models like Random Forest (RFX) and Naive baselines offered a modest benchmark in the short term. RFX handled non-linearity better than linear models but lacked temporal awareness, while the Naive method offered a simple yet surprisingly resilient baseline in some cases. However, neither was able to match N-HiTS's adaptability or predictive accuracy.

Exogenous Signal Sensitivity: Incorporating external variables is critical for inflation forecasting, given its dependence on macroeconomic indicators. N-HiTS integrates exogenous inputs dynamically, adjusting their influence based on both context and forecast horizon. This contrasts with traditional models like ARDL, ARIMAX, and SARIMAX, which require manual specification of lags and treat variable effects as static. N-HiTS's learned integration allowed it to leverage indicators such as interest rates, industrial indices, or commodity prices in a more flexible and horizon-aware manner.

In contrast other deep learning models like N-BEATSx and TFT, though also capable of using exogenous data, struggled to make meaningful use of them. N-BEATSx, for example, treated all input time steps uniformly and lacked mechanisms for differential feature weighting or temporal pooling, contributing to its poor performance even in the short term.

5.2 LONG TERM FORECASTING (6 and 12 Horizons)

Hierarchical Interpolation: As the forecast horizon extended, the performance of most models dropped sharply. N-HiTS, however, continued to produce stable and relatively accurate predictions. This was especially evident at 6 and 12 months, where many other models began to diverge or produce overly simplistic trends. N-HiTS's hierarchical structure, which gradually refines forecasts through multiple layers, allowed it to balance detail with smoothness—avoiding overfitting while preserving essential patterns.

In contrast, recurrent models such as LSTM, GRU, and RNN showed substantial degradation over longer periods, often generating unstable or exaggerated predictions due to cumulative errors.

Similarly, XGBoost, MLR, and MARS—while effective in static regression tasks—were not well suited to dynamic, multi-step forecasting and suffered from a sharp decline in predictive quality at longer horizons.

Smooth Trend Decomposition: N-HiTS's use of stacked residual blocks enabled it to refine forecasts iteratively, addressing error components at each level. This layering process produced smoother long-term trends that remained consistent with underlying inflation dynamics. While N-BEATSx also used residual learning, its lack of pooling or hierarchical resolution meant it could not adapt effectively to longer horizons.

TCN, TFT, and TiDE—while conceptually designed to manage temporal dependencies—also underperformed at longer ranges. Their reliance on fixed receptive fields or complex attention mechanisms made them vulnerable to both data limitations and noise. N-HiTS's trend decomposition appeared to be more resilient to these issues.

Walk-Forward Robustness: A notable strength of N-HiTS was its consistent performance across all horizons under walk-forward evaluation. Unlike many models that needed to be retrained for each forecasting step or horizon, N-HiTS maintained generalisability without specific adjustments. This was especially beneficial in real-world scenarios where frequent retraining is impractical.

Models such as ARDL, RFX, and XGBoost showed severe drops in accuracy when evaluated across extended time spans. Their lack of temporal generalisation and their reliance on fixed training-test splits made them less suitable for time series tasks requiring continual adaptability. In contrast, N-HiTS handled horizon progression naturally, without loss of structural coherence in its forecasts.

Robustness to Overfitting: Overfitting generally became more prominent as the forecast window lengthened. N-HiTS mitigated this risk more effectively through its use of dropout, batch normalisation, and early stopping. Its multi-stage design also contributed to smoother convergence during training. Traditional models like VAR and ARDL were generally more resistant to overfitting, but lacked the flexibility to model complex economic relationships, often resulting in underfitting. N-HiTS struck a better balance, offering both flexibility and control, making it particularly robust in longer-horizon settings.

5.3 Limitations of N-HiTS

Despite its advantages, N-HiTS comes with trade-offs. Its architecture is computationally intensive, requiring significant training time and resources. It also lacks interpretability —unlike ARIMA or ARDL, it does not offer transparent explanations of variable influence, which is a drawback for policy analysis.

The model's performance also depends heavily on careful preprocessing and feature configuration. Poor scaling or window choices can reduce accuracy or lead to unstable training. Additionally, N-HiTS may underperform in low-data environments where its complexity becomes a disadvantage compared to more parsimonious models.

N-HiTS' prediction smoothness is also generally a useful property, but this could affect its performance in scenarios involving sudden economic shocks. However, there are few models that can handle such scenarios.

Despite these challenges, N-HiTS remains a strong candidate for high-accuracy forecasting when resources, data, and expertise are available to support its deployment and tuning.

6 CONCLUSION

This study conducted a comprehensive evaluation of inflation forecasting models ranging from classical econometrics to modern deep learning architectures. Among all models tested, N-HiTS consistently achieved the highest accuracy across multiple horizons, demonstrating robustness to short- and long-term prediction tasks. Its multi-scale interpolation and residual refinement mechanisms aligned well with inflation's structural dynamics, outperforming more rigid or overparameterised alternatives.

Despite this, our findings highlight that model effectiveness depends on specific design choices, data granularity, and feature integration—factors that can drastically alter performance across horizons. Models such as RF, VAR, and MAR underperformed primarily due to sensitivity to overfitting and challenges in generalising across time. Others, like ARDL and XGBoost, struggled with temporal adaptability. These comparative insights offer valuable guidance for selecting forecasting approaches based on scenario-specific requirements.

6.1 Limitations

While our results demonstrate empirical strengths, several real-world limitations constrain generalisability:

Unpredictable Economic Shocks: All models are trained on historical data, making them ill-equipped to anticipate unforeseen disruptions such as pandemics or geopolitical conflicts. These events often introduce structural breaks in inflation dynamics that are difficult to pre-empt algorithmically.

Lack of Social and Political Context: Many inflationary shifts arise from policy decisions, elections, strikes, or public sentiment. Our models do not currently incorporate unstructured data sources such as government announcements, or sentiment indicators, which could offer early signals of change [Baker et al. 2016].

Data Lag Constraints: Official indicators like CPI, GDP, and employment data are published with lags [U.S. Bureau of Economic Analysis 2025; U.S. Bureau of Labor Statistics 2025], which hampers real-time adaptability, particularly during volatile periods such as election years or financial crises.

Reliance on Historical Patterns: Deep learning models, while powerful in pattern recognition, assume continuity in relationships. Economic regime shifts —e.g. rapid monetary tightening —can violate these assumptions, resulting in degraded forecast accuracy.

6.2 Future Work

Several directions can be pursued to improve the forecasting of our models further. Implementing **cross validation** across multiple periods would allow a better estimation of the inter-period accuracy of predictions. Many of the models included here could benefit from further **hyperparameter optimisation**: those prone to overfitting need extensive work to generate acceptable results, including architecture and feature engineering modifications. Although our results show that N-HiTS dominates prediction performance, with further alterations, this may generally differ. Many other models have also not been covered, including ensemble and adaptive models. These may be able to reduce individual model biases and downsides.

Another key element of inflation prediction is social factors, such as the contemporary states of politics, conflict, and social trust both domestically and internationally. Fluctuations in these conditions are often attributed post-hoc as the causes of economic (and therefore inflationary) shocks. Semantic analysis of news articles, government policies, and corporate announcements may allow for more accurate predictions, especially in turbulent economic environments with lesser-understood complex relationships between historical outcomes and future data. These relationships are especially unpredictable in recent times, with the shock of COVID-19 followed by a series of turbulent geopolitical events - reinforcing the potential benefits of such future work.

ACKNOWLEDGMENTS

Thanks to Dr Timon S. Gutleb for his consistent advice, support, and guidance in the management and implementation of this project.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2623–2631. https://doi.org/10.1145/3292500.3330701

Ghulam Ali. 2015. Econometric Approaches for Water Quality Variates: Cointegration, VAR, VECM, and ARIMAX. *Journal of Statistical and Econometric Methods* 4, 1 (2015), 1–38. Online ISSN: 2241-0376.

Mike Allen. 2017. Post Hoc Tests: Tukey Honestly Significant Difference Test. In *The SAGE Encyclopedia of Communication Research Methods*. SAGE Publications, Inc., Thousand Oaks, CA. https://doi.org/10.4135/9781483381411.n452

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. arXiv:1803.01271 [cs.LG] https://arxiv.org/abs/1803.01271

Scott R. Baker, Nicholas Bloom, and Steven J. Davis. 2016. Measuring Economic Policy Uncertainty. *Quarterly Journal of Economics* 131, 4 (2016), 1593–1636. https://doi.org/10.1093/qje/qjw024

Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5, 2 (1994), 157–166.

G. Bignall, M.M. Kurmaz, A. Rahman, J. Zhangly, K. Raffaelli, S. Guran, N. Leung, and T.S. Gutleb. 2025. *COMP5530M-Group-Project-Inflation-Forecasting*. github.com/sc20geb/COMP5530M-Group-Project-Inflation-Forecasting

BoE. Last updated: 16-04-2025. Inflation and the 2% target. https://www.bankofengland.co.uk/monetary-policy/inflation Accessed 23-04-2025.

Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne. 2013. *Machine Learning Strategies for Time Series Forecasting*. Springer Berlin Heidelberg, Berlin, Heidelberg, 62–77.

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza, Max Mergenthaler-Canseco, and Artur Dubrawski. 2022a. N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting. arXiv:2201.12886 [cs.LG]

Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza, Max Mergenthaler-Canseco, and Artur Dubrawski. 2022b. NeuralForecast: Deep learning models for time series forecasting. https://github.com/Nixtla/neuralforecast. [Accessed 28-April-2025].

Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza, Max Mergenthaler-Canseco, and Artur Dubrawski. 2024. NeuralForecast: A Library for Time Series Forecasting with Deep Learning Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 11044–11051. https://ojs.aaai.org/index.php/AAAI/article/view/25854

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, 785–794.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv:1406.1078 [cs.CL]

Collins. 2020. *Collins English Dictionary*. HarperCollins.

Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu. 2024. Long-term Forecasting with TiDE: Time-series Dense Encoder. arXiv:2304.08424 [stat.ML] https://arxiv.org/abs/2304.08424

David A. Dickey and Wayne A. Fuller and. 1979. Distribution of the Estimators for Autoregressive Time Series with a Unit Root. *J. Amer. Statist. Assoc.* 74, 366a (1979), 427–431. https://doi.org/10.1080/01621459.1979.10482531 arXiv:https://doi.org/10.1080/01621459.1979.10482531

Francis Diebold and Roberto Mariano. 1994. Comparing predictive accuracy. *Journal of Business Economic Statistics* (Nov 1994). https://doi.org/10.3386/t0169

WenQi Duan and H.Ẽugene Stanley. 2011. Cross-correlation and the predictability of financial return series. *Physica A: Statistical Mechanics and its Applications* 390, 2 (2011), 290–296. https://doi.org/10.1016/j.physa.2010.09.013

Economic Policy Uncertainty. 2025. *Economic Policy Uncertainty Index*. policyuncertainty.com [Online]. [Accessed: 7 May 2025].

Jeffrey L. Elman. 1990. Finding Structure in Time. *Cognitive Science* 14, 2 (1990), 179–211.

Jiahui Wang Eric Zivot. 2006. *Vector Autoregressive Models for Multivariate Time Series*. Springer New York, New York, NY, 385–429.

Gidon Eshel. 2003. The Yule Walker Equations for the AR Coefficients. http://www-stat.wharton.upenn.edu/~steele/Courses/956/Resource/YWSourceFiles/YW-Eshel.pdf. [Accessed 14-04-2025].

Fed. Last updated: 02-08-2024. Federal Reserve Inflation Figures. https://www.federalreserve.gov/economy-at-a-glance-%inflation-pce.htm Accessed 04-12-2024.

Fed. Last updated: 28-03-2025. Personal Consumption Expenditures. https://fred.stlouisfed.org/series/PCEPI Accessed 17-04-2025.

Federal Reserve Bank of New York. 2025. *Global Supply Chain Pressure Index (GSCPI)*. newyorkfed.org/research/policy/gscpi [Online]. [Accessed: 7 May 2025].

Federal Reserve Bank of St. Louis. 2025. *Inflation Data List*. fredaccount.stlouisfed.org/public/datalist/8270 [Online]. [Accessed: 7 May 2025].

Food and Agriculture Organization of the United Nations. 2025. FAO Food Prices Index (FFPI). fao.org/worldfoodsituation/foodpricesindex [Online]. [Accessed: 7 May 2025].

Jerome H. Friedman. 1991. Multivariate Adaptive Regression Splines. *The Annals of Statistics* 19, 1 (1991), 1 – 67.

Friedrich Fritzer, Gabriel Moser, and Johann Scharler. 2002. Forecasting Austrian HICP and its Components using VAR and ARIMA Models.

Xiangyun Gao, Shupei Huang, Xiaoqi Sun, Xiaoqing Hao, and Feng An. 2018. Modelling cointegration and Granger causality network to detect long-term equilibrium and diffusion paths in the financial system. *Royal Society Open Science* 5, 3 (2018), 172092.

Angelica Gianfreda, Paolo Maranzano, Lucia Parisio, and Matteo Pelagatti. 2023. Testing for integration and cointegration when time series are observed with noise. *Economic Modelling* 125 (May 2023), 106352.

C. W. J. Granger. 1969. Investigating Causal Relations by Econometric Models and Cross-spectral Methods. *Econometrica* 37, 3 (1969), 424–438.

Groningen Growth and Development Centre. 2023. Penn World Table version 10.01.

David Harvey, Stephen Leybourne, and Paul Newbold. 1997. Testing the equality of prediction mean squared errors. *International Journal of Forecasting* 13, 2 (Jun 1997), 281–291.

Julien Herzen, Francesco LÃ¤ssig, Samuele Giuliano Piazzetta, Thomas Neuer, LÃ©o Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, Maxime Dumonal, Jan KoÅ›cisz, Dennis Bader, FrÃ©dÃ©rick Gusset, Mounir Benheddi, Camila Williamson, Michal Kosinski, Matej Petrik, and GaÃ«l Grosch. 2022. Darts: User-Friendly Modern Machine Learning for Time Series. *Journal of Machine Learning Research* 23, 124 (2022), 1–6.

Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, Vol. 1. 278–282 vol.1.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (11 1997), 1735–1780.

Rob J. Hyndman and Yeasmin Khandakar. 2008. Automatic Time Series Forecasting: The forecast Package for R. *Journal of Statistical Software* 27, 3 (2008), 1–22. https://doi.org/10.18637/jss.v027.i03

investing.com. 2025. *investing.com Main Page*. [Online]. [Accessed: 7 May 2025].

Ling-Jing Kao, Chih-Chou Chiu, Chi-Jie Lu, and Chih-Hsiang Chang. 2013. A hybrid approach by integrating wavelet-based feature extraction with MARS and SVR for stock index forecasting. *Decision Support Systems* 54, 3 (2013), 1228–1244.

Hae-Young Kim. 2014. Analysis of variance (ANOVA) comparing means of more than two groups. *Restorative Dentistry & Endodontics* 39, 1 (20 Jan 2014), 74–77. https://doi.org/10.5395/rde.2014.39.1.74

Joos Korstanje. 2021. *The SARIMAX Model*. Apress, Berkeley, CA, 125–131.

Denis Kwiatkowski, Peter C.B. Phillips, Peter Schmidt, and Yongcheol Shin. 1992. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics* 54, 1 (1992), 159–178. https://doi.org/10.1016/0304-4076(92)90104-Y

Colin Lea, Rene Vidal, Austin Reiter, and Gregory D. Hager. 2016. Temporal Convolutional Networks: A Unified Approach to Action Segmentation. arXiv:1608.08242 [cs.CV] https://arxiv.org/abs/1608.08242

Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister. 2020. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. arXiv:1912.09363 [stat.ML]

Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. 2021. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting* 37, 4 (2021), 1748–1764.

Bryan Lim and Stefan Zohren. 2021. Time-series forecasting with deep learning: a survey. *Philos. Trans. A Math. Phys. Eng. Sci.* 379, 2194 (April 2021), 20200209.

Sebastiano Manzan and Dawit Zerom. 2013. Are macroeconomic variables useful for forecasting the distribution of US inflation? *International Journal of Forecasting* 29, 3 (2013), 469–478.

Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. 2000. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems 12 - Proceedings of the 1999 Conference, NIPS 1999 (Advances in Neural Information Processing Systems)*. Neural Information Processing Systems Foundation, 512–518. 13th Annual Neural Information Processing Systems Conference, NIPS 1999 ; Conference date: 29-11-1999 Through 04-12-1999.

Mathuranathan. 2014. Autocorrelation (Correlogram) and Persistence – Time Series Analysis. https://www.gaussianwaves.com/2014/06/autocorrelation-correlogram-and-persistence-time-series-analysis/. [Accessed 14-04-2025].

MathWorks. 2023. Ljung-Box Q-Test - MATLAB & Simulink. https://www.mathworks.com/help/releases/R2021a/econ/ljung-box-q-test.html. [Accessed 14-04-2025].

MathWorks. 2025. Autocorrelation and Partial Autocorrelation - MATLAB & Simulink. https://www.mathworks.com/help/econ/autocorrelation-and-partial-autocorrelation.html. [Accessed 14-04-2025].

Kin G. Olivares, Cristian Challu, Grzegorz Marcjasz, Rafał Weron, and Artur Dubrawski. 2023. Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with NBEATSx. *International Journal of Forecasting* 39, 2 (April 2023), 884–900.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. arXiv:1905.10437 [cs.LG]

Awad Mohamed Osman, Ahmed Osman Ahmed, Mohammed Nour Eltahir, Anas Satti Mohamed, Osman Saad Shidwan, and M Ghada. 2019. Investigating the causes of inflation in Saudi Arabia: an application of autoregressive distributed lag (ARDL) model. *International Journal of Applied Engineering Research* 14, 21 (2019), 3980–3986.

Antonio Rafael Sabino Parmezan, Vinicius M.A. Souza, and Gustavo E.A.P.A. Batista. 2019. Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model. *Information Sciences* 484 (2019), 302–337.

Adam et al. Paszke. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv:1912.01703 [cs.LG] https://arxiv.org/abs/1912.01703

Skipper Seabold and Josef Perktold. 2010. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*.

statsmodels contributors. 2025. statsmodels documentation. statsmodels.org Accessed: 2025-05-08.

The Organisation for Economic Co-operation and Development (OECD). 2025. *Business Confidence Index (BCI)*. oecd.org/en/data/indicators/business-confidence-index-bci.html [Online]. [Accessed: 7 May 2025].

Volkan Ülke, Afsin Sahin, and Abdulhamit Subasi. 2018. A comparison of time series and machine learning models for inflation forecasting: empirical evidence from the USA. *Neural Computing and Applications* 30, 5 (01 Sep 2018), 1519–1527.

Pennsylvania State University. 2024. STAT 510: Applied Time Series Analysis — Lesson 11.2: Vector Autoregressive (VAR) Models. https://online.stat.psu.edu/stat510/lesson/11/11.2. Accessed: 2025-05-03.

U.S. Bureau of Economic Analysis. 2025. National Economic Accounts. https://www.bea.gov/. [Accessed 05-05-2025].

U.S. Bureau of Labor Statistics. 2025. Consumer Price Index and Producer Price Index. https://www.bls.gov/. [Accessed 05-05-2025].

U.S. Energy Information Administration. 2025. Henry Hub Natural Gas Spot Price. eia.gov/dnav/ng/hist/rngwhhdm.htm [Online]. [Accessed: 7 May 2025].

World Bank Group. 2025. *World Bank Public Documents*. thedocs.worldbank.org/en/doc/webdocs [Online]. [Accessed: 7 May 2025].

Yahoo Finance. 2025. *Yahoo Finance Main Page*. uk.finance.yahoo.com [Online]. [Accessed: 7 May 2025].

A PROJECT MANAGEMENT

A.1 Supervisor and Theme

The theme of this project emerged from a shared group interest in artificial intelligence, particularly in machine learning and deep learning. All group members had prior experience in AI-related topics and were eager to explore its application in a real-world domain. Several members had backgrounds in economics and finance and the group collectively identified the importance of inflation forecasting. This intersection of AI and economics provided a meaningful and challenging scope for applying predictive modelling techniques.

The group chose Dr. Timon S. Gutleb as the supervisor because of his experience and interest in machine learning. Dr. Gutleb's research focuses on the development of computational algorithms and tools for scientific domains, including applications in molecular quantum physics, biology, and more recently, machine learning. His expertise in model development and algorithmic thinking was invaluable for a project that aimed to deploy machine learning models in inflation forecasting. His background in model development aligned with the project's aim of developing machine learning models for inflation forecasting. The group met in person with Dr. Gutleb on average every two weeks. These meetings were used to review the project's progress, clarify objectives, and receive valuable feedback.

The team made effective use of the supervisor's expertise by preparing agendas in advance, bringing specific challenges to meetings, and integrating feedback into our model design and evaluation strategies. These sessions provided academic grounding and helped redirect efforts when technical issues arose.

A.2 Group Organisation

The group followed an Agile project management strategy, facilitating steady progress. The project was divided into three phases: implementation, evaluation, and documentation. They were further split into weekly tasks in the project timeline so the project was more manageable. These smaller tasks ensured that all deliverables were met on time.

Although the team shared the same goal, tasks were often completed in smaller groups to improve efficiency and allow deeper focus. For example, model standardisation and data preprocessing were handled collaboratively in pairs or trios. However, tasks such as researching relevant datasets and reviewing academic literature were undertaken by all team members to ensure a strong foundational understanding and shared context for future model development. Each member also independently selected one or more model(s) to develop and optimise. This approach helped diversify contributions while focusing on individuals' strengths and interests.

WhatsApp was used for informal communication, such as raising issues, organising meetings and sharing quick updates. Microsoft Teams was used for more formal virtual meetings and sharing relevant information. Independent internal team meetings were held once or twice per fortnight, depending on the workload. These internal meetings allowed the team to share their progress, review set goals, discuss potential issues, and assign new tasks. The frequency of meetings increased as deadlines approached.

Each meeting had a clear agenda established in advance through informal discussion among the team. Meeting minutes were recorded and stored on GitHub wiki to ensure they were accessible to all the team members. These minutes documented the discussion points, decisions, and tasks assigned to each member. This helped ensure transparency and consistency across the group.

Task distribution was discussed openly to ensure fairness in workload. Individual contributions were tracked and tasks were balanced weekly by reviewing each person's responsibilities. Each task's deadlines were typically aligned with the next meeting and were strictly followed to avoid last-minute delays. When a delay or issue occurred, tasks would be discussed and potentially redistributed with the team's agreement. When team members encountered challenges—such as debugging evaluation pipelines or dealing with model integration—others offered hands-on help via pair programming or group calls. This collaborative support structure helped ensure no member fell behind.

The project's codebase was maintained on GitHub, providing backups, peer-reveiwed version control, workflow automation and collaboration tools for basic project management and wikis. Before the updated code was allowed to be merged, the code had to go through peer review, this meant an approval from at least one other group member was required. This practice ensured code quality, prevented merge conflicts, and allowed team members to work asynchronously.

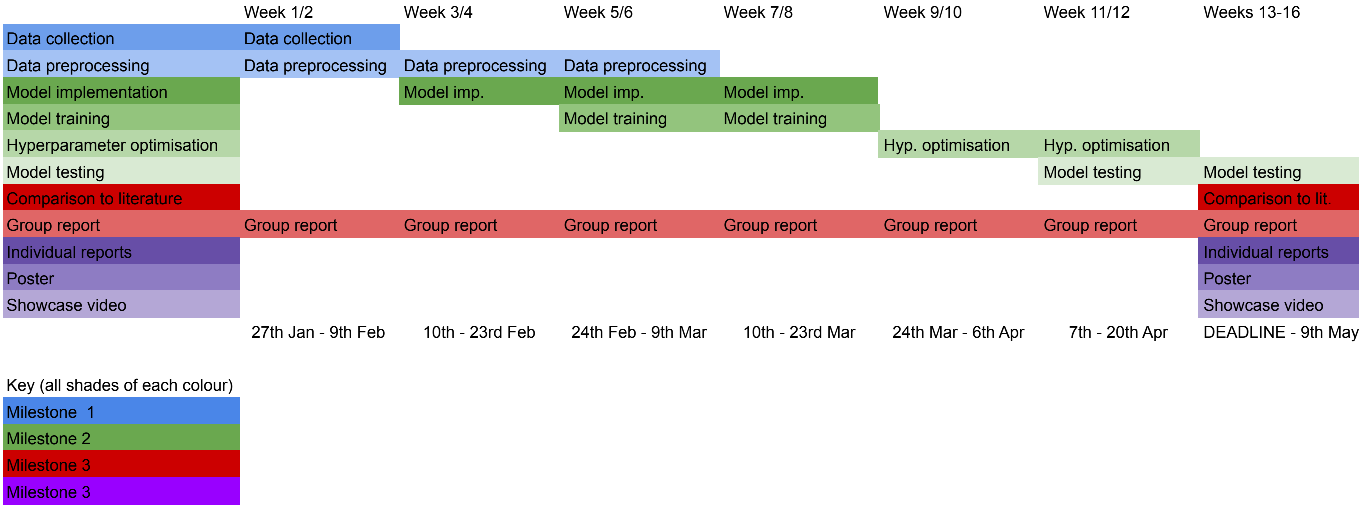
The group organisation enabled effective communication among members and manageable progress tracking. The team maintained a structured task management system and was able to progress steadily while adapting to unforeseen delays or challenges.

A.3 Planning and execution

The initial plan for the project can be seen in the Gantt chart in Figure 6. Our project plan was structured over a 16-week timeline and broken down into several phases to align with project milestones. We planned to complete data collection in the first two weeks, with data preprocessing starting immediately after and continuing through week 6. This schedule allowed us to begin preparing the data as soon as it became available. Model implementation was scheduled to start in week 3, running in parallel with ongoing preprocessing. Model training was planned to start in week 4 after the implementation of the models. We aimed to complete all model implementation and training by week 8, allowing us to begin hyperparameter optimisation from week 9 through week 12. This was to be followed by model testing, scheduled for weeks 11 to 16.

The group report was designed to be a continuous task along with all the coding tasks from weeks 3 to 16, ensuring that our documentation accurately reflected the development process in real time. Comparison to literature, poster, individual reports, and showcase video were all scheduled for the final three weeks after the completion of the codebase. This plan aimed to balance parallel development with clear milestones for effective time management.

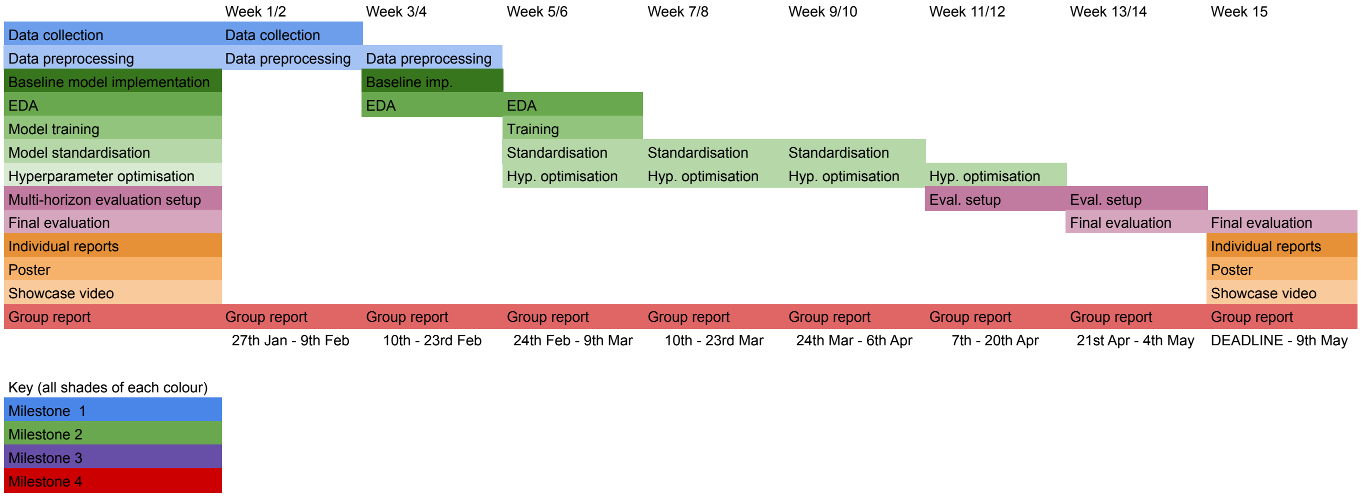
The Gantt chart in Figure 7 shows actual progress. While the project's overall structure remained consistent, some adjustments were made during implementation. Model standardisation was introduced after baseline model implementation when we realised many



	Week 1/2	Week 3/4	Week 5/6	Week 7/8	Week 9/10	Week 11/12	Weeks 13-16
Data collection	Data collection						
Data preprocessing	Data preprocessing	Data preprocessing	Data preprocessing				
Model implementation		Model imp.	Model imp.	Model imp.			
Model training			Model training	Model training			
Hyperparameter optimisation					Hyp. optimisation	Hyp. optimisation	
Model testing						Model testing	Model testing
Comparison to literature							Comparison to lit.
Group report	Group report	Group report	Group report	Group report	Group report	Group report	Group report
Individual reports							Individual reports
Poster							Poster
Showcase video							Showcase video
	27th Jan - 9th Feb	10th - 23rd Feb	24th Feb - 9th Mar	10th - 23rd Mar	24th Mar - 6th Apr	7th - 20th Apr	DEADLINE - 9th May

Key (all shades of each colour)
Milestone 1
Milestone 2
Milestone 3
Milestone 3

Fig. 6. Initial Gantt chart



	Week 1/2	Week 3/4	Week 5/6	Week 7/8	Week 9/10	Week 11/12	Week 13/14	Week 15
Data collection	Data collection							
Data preprocessing	Data preprocessing	Data preprocessing						
Baseline model implementation		Baseline imp.						
EDA		EDA	EDA					
Model training			Training					
Model standardisation			Standardisation	Standardisation	Standardisation			
Hyperparameter optimisation			Hyp. optimisation	Hyp. optimisation	Hyp. optimisation	Hyp. optimisation		
Multi-horizon evaluation setup						Eval. setup	Eval. setup	
Final evaluation							Final evaluation	Final evaluation
Individual reports								Individual reports
Poster								Poster
Showcase video								Showcase video
Group report	Group report	Group report	Group report	Group report	Group report	Group report	Group report	Group report
	27th Jan - 9th Feb	10th - 23rd Feb	24th Feb - 9th Mar	10th - 23rd Mar	24th Mar - 6th Apr	7th - 20th Apr	21st Apr - 4th May	DEADLINE - 9th May

Key (all shades of each colour)
Milestone 1
Milestone 2
Milestone 3
Milestone 4

Fig. 7. Final Gantt chart

of us were implementing similar functions across different models. To ensure consistency and fairness in comparison, we decided to use shared preprocessing, hyperparameter optimisation, and evaluation logic. Hyperparameter optimisation was conducted longer than the initial plan, as it became clear that tuning was important to improve model performance compared to the actual data. Additionally, we introduced multiple forecasting horizons (1, 3, 6, and 12), which required further adjustments to model architecture and output handling. These changes led to a longer model development and evaluation period than the initial plan but had more robust and comparable results.

Despite minor timeline changes, overall progress remained steady and all deliverables were completed on time.

A.4 Risk Mitigation

Experimenting with multiple forecasting models came with inherent risks, including inconsistencies across implementations, integration complexity, and varying model requirements. To address this early, the group agreed on a common preprocessing strategy —applying log transformations to the inflation series, using z-score normalisation for input features, and maintaining a consistent time window across models. This early alignment was key to maintaining comparability throughout the project.

Despite this shared foundation, challenges arose as each model had different architectural needs and performance characteristics. For instance, some models —such as N-BEATS —did not integrate well with exogenous variables or failed to generalise under walk-forward evaluation. To manage this, we adopted a performance-driven model selection process: models that consistently underperformed or lacked compatibility with our standardised pipeline were

Source	Exegenous Variables (Codes)
Federal Reserve	PCEPI, AHETPI, APU000074714, BAA10YM, BAAFFM, CSUSHPISA, EXCAUS, EXINUS, EXKOUS, EXMAUS, EXNOUS, EXSDUS, EXSIUS, EXSLUS, EXTAUS, EXUSAL, EXUSNZ, FEDFUNDS, IPMINE, M2SL, MPRIME, PCUOMFGOMFG, PCUOMINOMIN, PNGASEUUSDM, PPIACO, PSAVERT, TB3MS, TB6MS, TERMCBPER24NS, PSOYBUSDM, A053RC1Q027SBEA, FYGFGDQ188S, GDP, GFDEGDQ188S, FDHBFIN, FDHBPIN, FYGFDPUN, DCOILWTICO, DGS10
Investing.com	OIL (WTI Crude Oil), LCOM5 (Brent Oil), GCM5 (Gold), SIK5 (Silver), HGK5 (Copper), BADI (Baltic Dry Index), IXIC (NASDAQ Composite)
EIA Henry Hub	Henry Hub Natural Gas Spot Price
Yahoo Finance	CBOE Volatility Index (VIX)
FAO	Food Prices Index
OECD	Business Confidence Index (BCI)
New York Fed	Global Supply Chain Pressure Index (GSCPI)
Policy Uncertainty	Economic Policy Uncertainty Index (UK, US)
World Bank	Commodity Prices (Monthly)

Table 1. Summary of data sources used in this project

set aside in favour of more promising architectures like RNN, LSTM, N-BEATSx, N-HiTS, and SARIMAX.

Another layer of complexity involved ensuring fair comparison across models. While implementations began in parallel, we later recognised the need for unified evaluation metrics and output formatting. In response, we refactored shared components into standardised reusable scripts (e.g., `evaluation.ipynb`) and applied consistent MAE and RMSE scoring across all horizons. These efforts reduced code duplication and streamlined final integration.

Overall, our approach to risk mitigation was proactive and iterative. Through early standardisation, regular collaboration, and flexible model triage, we ensured the project remained coherent, efficient, and robust —despite the breadth of models explored.

B GROUP REFLECTION AND SOFT SKILLS ACQUIRED

Collaborating in a large group of seven gave us a lot of challenges and opportunities. Although we had some disagreements during the project, we overcame those issues by effectively and regularly communicating and consistently listening to each other. We developed skills in resolving conflicts and understanding different perspectives, which brought us closer as a team and helped us work smoothly together. These experiences have prepared us for working in a professional environment, where effective teamwork and coordination are important for project success.

In addition to teamwork, we also honed our skills in time management, adaptability, and accountability. Coordinating across multiple technical components —data preprocessing, modelling, evaluation, and report writing —required us to plan ahead, delegate tasks, and meet tight deadlines. We learned to adapt when things didn't go as expected, such as when certain models underperformed or integration took longer than planned. By taking ownership of individual responsibilities while staying aligned with the group's goals, we cultivated a strong sense of trust and mutual support.

C DATASET

Researching and choosing candidate exogenous variables was a group effort, where candidate exogenous variables were largely sourced from fred.stlouisfed.org [Federal Reserve Bank of St. Louis 2025] and 8 other reliable sources as follows: [Economic Policy Uncertainty 2025; Federal Reserve Bank of New York 2025; Food and Agriculture Organization of the United Nations 2025; investing.com 2025; The Organisation for Economic Co-operation and Development (OECD) 2025; U.S. Energy Information Administration 2025; World Bank Group 2025; Yahoo Finance 2025]. See table 1 and the GitHub repository [Bignall et al. 2025] for more details.

Additionally, non-causal exogenous variables were identified and removed using Granger's causality test [Granger 1969] with a standard significance level of 5%, where candidate variables that did not display a statistically significant Granger-causal relationship were removed. See the GitHub repository for more information on specific exogenous variables that Granger-cause PCEPI.