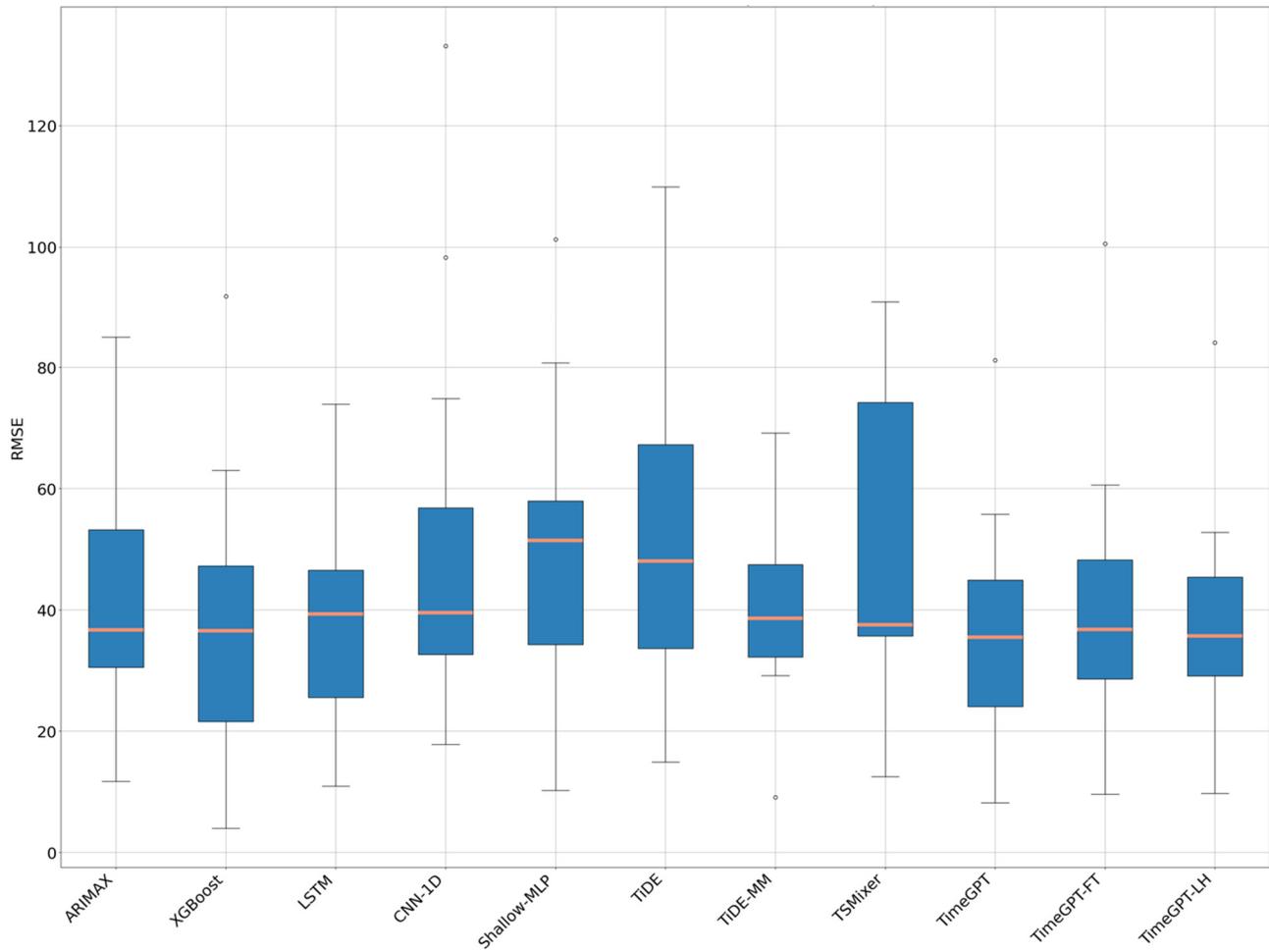
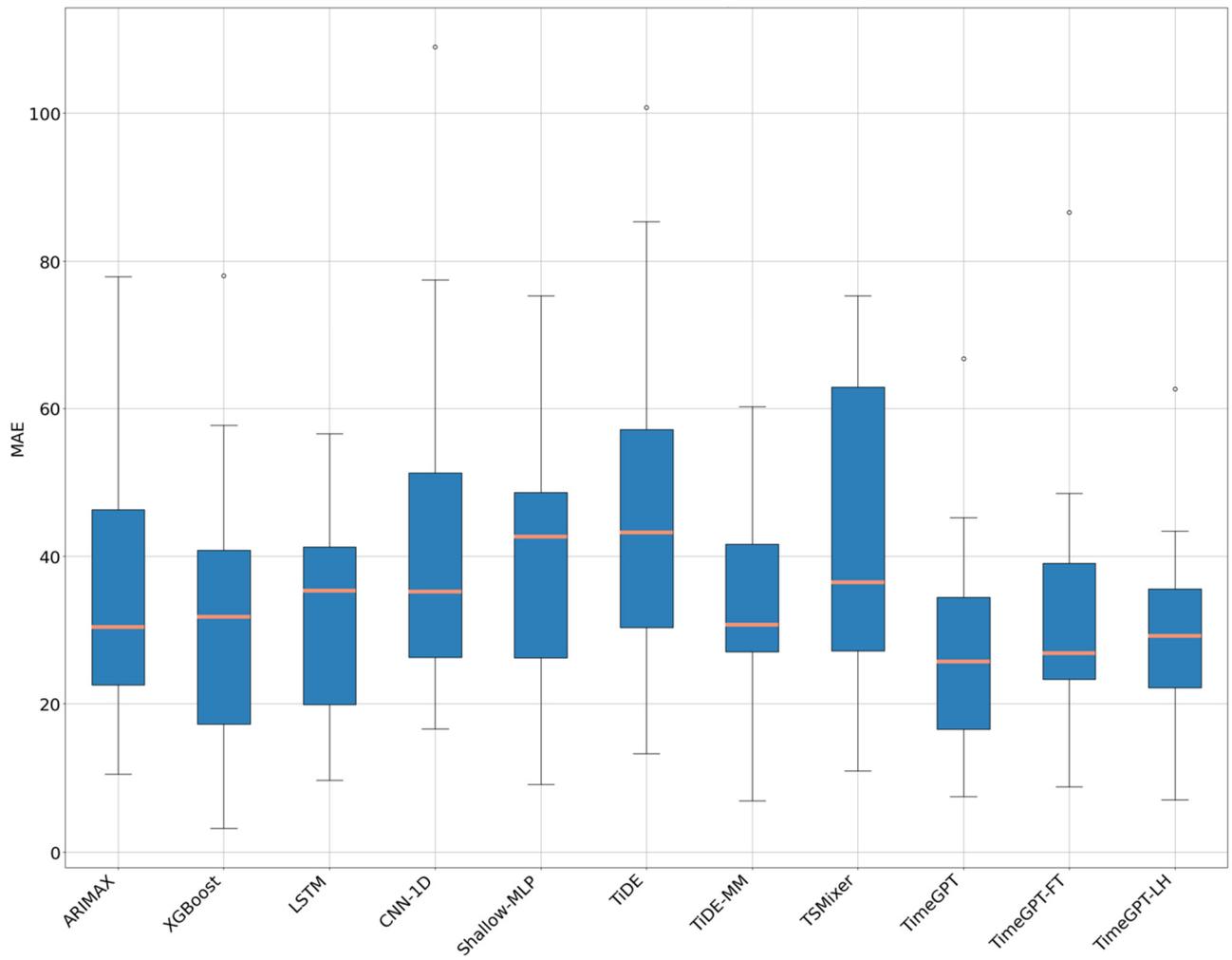


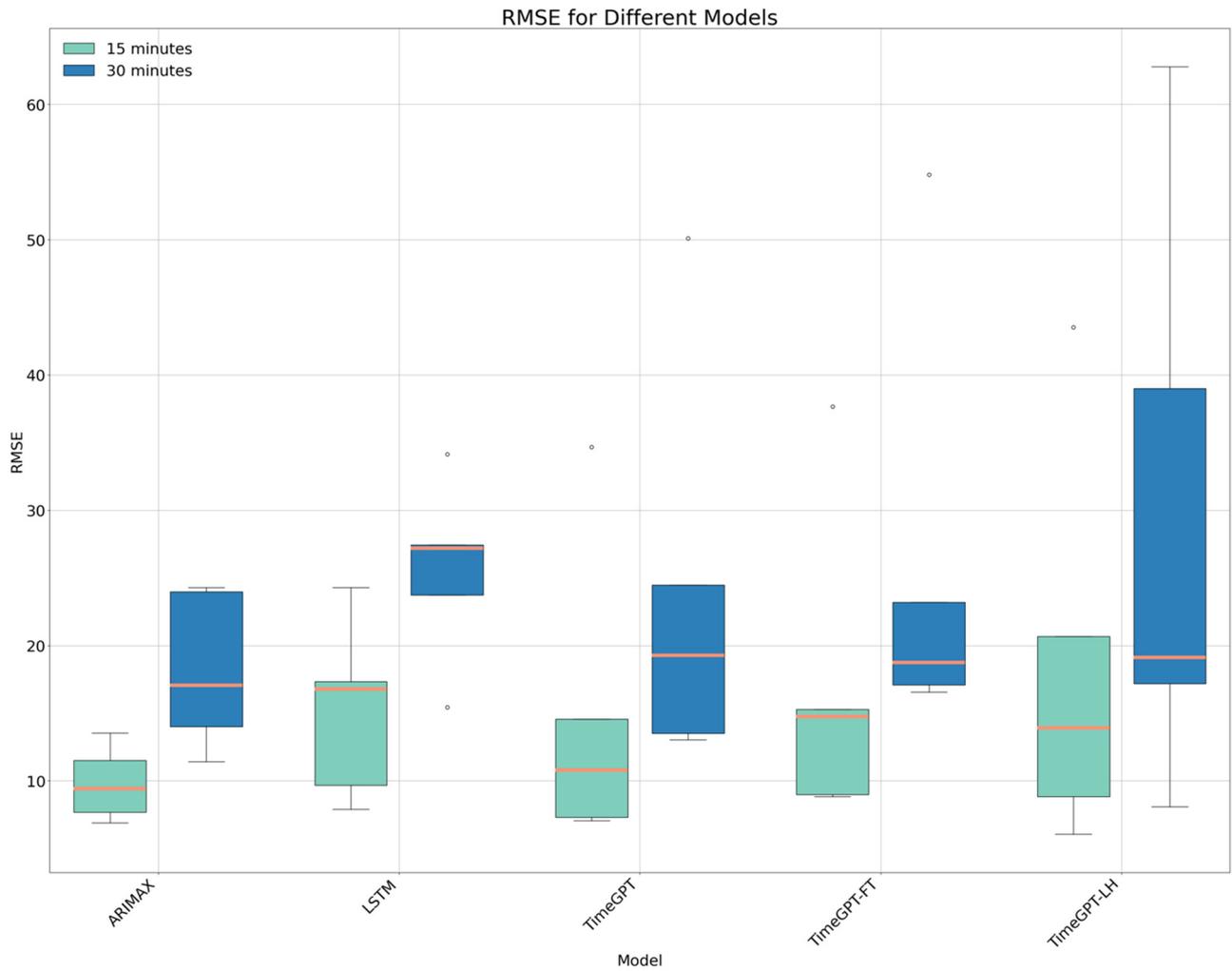
# Supplementary Materials



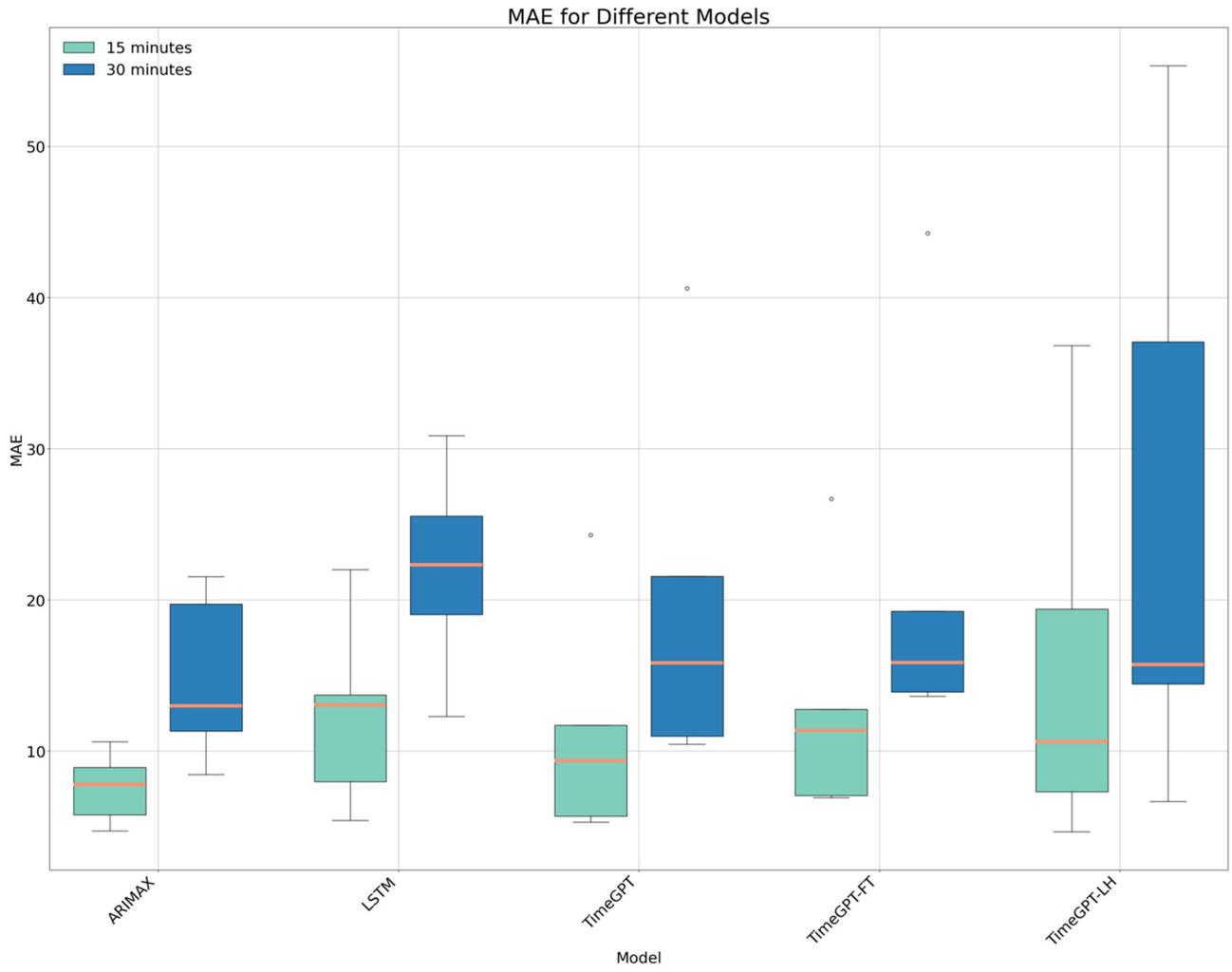
**Figure S1:** Boxplot of Root Mean Square Error (RMSE) values across different models for 60-minute forecasting horizons. The horizontal orange line within each box represents the median RMSE (ARIMAX: Autoregressive Integrated Moving average with Exogenous variables; XGBoost: Extreme Gradient Boosting; LSTM: Long Short-Term Memory; CNN-1D: 1-Dimension Convolutional Neural Network; Shallow-MLP: Shallow-Multilayer Perceptron; TiDE: Time-series Dense Encoder; TiDE-MM: Time-series Dense Encoder–Multi-patients Model; TSMixer: Time-Series Mixer; TimeGPT; TimeGPT-FT: TimeGPT Fine-Tuned; TimeGPT-LH: TimeGPT Long-Horizon).



**Figure S2:** Boxplot of Mean Absolute Error (MAE) values across different models for 60-minute forecasting horizons. The horizontal orange line within each box represents the median MAE.



**Figure S3:** Boxplot of Root Mean Square Error (RMSE) values for different models at 15-minute and 30-minute forecasting horizons on the OhioT1DM 2018 dataset. The horizontal orange line within each box represents the median RMSE.



**Figure S4:** Boxplot of Mean Absolute Error (MAE) values for different models at 15-minute and 30-minute forecasting horizons on the OhioT1DM 2018 dataset. The horizontal orange line within each box represents the median MAE.

**Table S1.** Clarke Error Grid average values of different models for the 60-minute forecasting horizon.

<b>Model</b>	<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>
ARIMAX	48,66	46,66	0,66	3,78	0,22
XGBoost	56	38,67	2,44	3,11	0
LSTM	52,44	43,55	1,11	2,88	0
CNN-1D	44,66	46,44	5,33	3,55	0
Shallow-MLP	48,66	44,44	3,11	3,55	0,22
TiDE	37,77	51,77	2,88	6,22	1,33
TiDE-MM	48,22	48,44	0	3,33	0
TSMixer	48,66	44,44	3,11	3,55	0,22
TimeGPT	64	31,11	2,22	2,66	0
TimeGPT-FT	64	31,11	2,22	2,66	0
TimeGPT-LH	60,88	33,33	3,33	2,44	0

## TimeGPT-1

TimeGPT represents a significant advancement in time series forecasting through its innovative use of transformer-based architecture. Unlike traditional time series models, which often rely on manual feature engineering and domain-specific adjustments, TimeGPT leverages deep learning principles, specifically transformers, to handle diverse and complex time series data more effectively.

At its core, TimeGPT builds on the transformer model, initially developed for natural language processing. The key component of this architecture is the self-attention mechanism, which allows the model to dynamically assess the importance of different time points in the input sequence. This capability enables TimeGPT to capture long-term dependencies and relationships within the data, which are crucial for accurate forecasting. The self-attention mechanism computes a weighted sum of input features, with their weights determined by the relevance of each feature to others, enabling the model to focus on the most important aspects of the time series.

TimeGPT adopts an encoder–decoder structure, typical of transformer models. The encoder processes the input time series data, creating encoded representations that capture essential information from the sequence. These representations are then used by the decoder to generate forecasts. Each layer in the encoder and decoder includes a self-attention mechanism followed by a feed-forward neural network, enhanced by residual connections and layer normalization for improved training stability and performance.

Time series data are inherently sequential, and understanding the order of data points is critical. To address this, TimeGPT incorporates positional encoding to maintain the sequential information of the input data. These positional encodings help the model differentiate between positions in the sequence, allowing it to capture temporal relationships and the order of events in the time series.

To ensure efficient training and convergence, TimeGPT employs layer normalization and residual connections within its layers. Layer normalization stabilizes the learning process by normalizing the inputs to each layer, while residual connections facilitate the gradient flow through the network, preventing vanishing gradient issues and enabling the training of deeper networks.

TimeGPT is designed to handle various characteristics of time series data, such as trends, seasonality, and noise. Its transformer-based architecture, with self-attention mechanisms, allows it to adapt to different frequencies and patterns in the data, making it suitable for a wide range of applications, including finance, healthcare, weather forecasting, and IoT data analysis.

One of the key advantages of TimeGPT is that it is pre-trained, meaning it can be used immediately, without the need for extensive further training, which is often resource-intensive. Additionally, TimeGPT is scalable and can be fine-tuned on other datasets, making it adaptable to specific applications or domains.

TimeGPT operates by taking a window of historical time series data as input and producing a forecast for future time points. The process begins with feeding the historical data, along with any additional exogenous variables, into the encoder. Positional encodings are added to retain the temporal order of the data. The encoder processes these data through multiple layers of self-attention and feed-forward networks, generating encoded representations that capture underlying patterns and relationships in the data. The decoder then uses these encoded representations, along with information about the positions of the future time points, to generate the forecast. The final output is passed through a linear layer to produce the forecasted values, representing the model's predictions for future time points.