University Energy and Natural Resources, Sunyani

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATICS.

SUPPORT VECTOR REGRESSION FOR FORECASTING GHANA'S USD TO GHS EXCHANGE RATE: A COMPARATIVE STUDY WITH ECONOMETRIC AND ML/QML BASELINES.

BY

LES KOFI ANHWERE (UEMS1200423)

MPHIL (COMPUTER SCIENCE)

NOVEMBER, 2025

SUPPORT VECTOR REGRESSION FOR FORECASTING GHANA'S USD TO GHS EXCHANGE RATE: A COMPARATIVE STUDY WITH ECONOMETRIC AND ML/QML BASELINES.

BY

LES KOFI ANHWERE

MASTER OF PHILOSOPHY (COMPUTER SCIENCE)

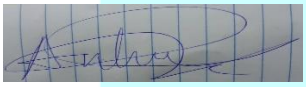
A THESIS SUBMITTED TO THE DEPARTMENT OF
COMPUTER SCIENCE AND INFORMATICS
SCHOOL OF SCIENCES
UNIVERSITY OF ENERGY AND NATURAL RESOURCES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF PHILOSOPHY
IN
COMPUTER SCIENCE

SEPTEMBER, 2025

DECLARATION

I, Les Kofi Anhwere (UEMS1200423), hereby declare that, except for the references cited which have been duly acknowledged, this submission is my own work toward a Master of Philosophy in Computer Science degree, and that to the best of my knowledge, it contains no materials previously published by another person. I also declare that this has not been presented either in whole or in part for another degree in this University or elsewhere.



Signature: …… Date: ……..

Les Kofi Anhwere

(UEMS1200423)

(Student)

Signature: …… Date: ……..

Name of Main Supervisor

(Supervisor)

Signature: …… Date: ……..

Name of Co-Supervisor

(Co-Supervisor)

Signature: …… Date: ……..

Prof. Patrick Kwabena Mensah

(Head of Department of Computer Science and Informatics)

ABSTRACT

Ghana continues to struggle with accurate exchange-rate forecasting because to structural breaks, nonlinear macroeconomic dynamics, and the shortcomings of conventional econometric models like ARIMA, ARIMAX, and VAR. These Bank of Ghana institutional workhorses are predicated on linearity, stationarity, and steady parameters—conditions that are increasingly broken in the fluctuating USD/GHS market. Studies that have already been conducted in Ghana frequently benchmark forecasting models inconsistently, lack unified assessment pipelines, and fail to evaluate operational usability or computational efficiency. The question of whether contemporary machine-learning approaches can produce much better and more trustworthy forecasts for decision-making is thus left open from a methodological and policy standpoint.

This study investigates whether a kernel-based Support Vector Regression (SVR) model provides better operational robustness and prediction accuracy than the baseline models used by the Bank of Ghana. We employ a two-phase strategy using a monthly macroeconomic panel spanning 2014–2023: (i) an ex-post backtest (train 2014–2022; test 2023) and (ii) an ex-ante operational forecast for January–December 2024 following model refitting on data through December 2023. Using accuracy metrics (MAE, RMSE, MSE, MAPE, R^2), computational cost indicators (training time, inference time, peak memory), and uncertainty quantification (native intervals for econometric models, bootstrap intervals for ML/QML), all models, econometric, classical ML, and exploratory quantum ML, are assessed within a single, leakage-safe pipeline.

According to the 2023 backtest results, SVR significantly outperforms ARIMA and VAR, although ARIMAX is still the most effective conventional comparator. While quantum variations show promise but inconsistent accuracy, classical machine learning methods (LSTM, XGBoost) offer moderate increases. A policy-ready Streamlit prototype that operationalizes model selection,

scenario analysis, uncertainty reporting, and exportable outputs for institutional usage is shown in the study's conclusion.

Keywords: Quantum Machine Learning, Classical Algorithms, Micro-Economic Forecasting, Exchange Rate Prediction, Ghana, LSTM, QLSTM, QSVM, QRF, Streamlit App

DEDICATION

This work is dedicated to my family, whose unwavering love, patience, and encouragement have been the backbone of my journey. To my parents, who instilled in me the values of hard work and perseverance, and to my siblings, who constantly reminded me of the power of resilience and faith. I also dedicate this thesis to every aspiring student and researcher who dares to push boundaries and dream beyond limitations. May this work serve as a reminder that with determination, support, and belief, great things are always within reach.

ACKNOWLEDGEMENT

I want to sincerely thank my supervisors, and, I wish to first express my deepest gratitude to God Almighty for the strength, wisdom, and perseverance that made this work possible. My heartfelt thanks go to my supervisors, Dr. Peter Nimbe and Professor Patrick K. Mensah, for their invaluable guidance, encouragement, and constructive feedback throughout this research journey. Their expertise not only shaped the direction of this thesis but also inspired me to grow as a researcher. I am profoundly grateful to my family for their continuous love, sacrifices, and belief in my abilities. Their encouragement provided me with the motivation to keep going, even in the most challenging moments. I also extend my appreciation to my friends and colleagues, whose support, discussions, and companionship made this academic journey enriching and enjoyable. Finally, I acknowledge the University of Energy and Natural Resources for providing the academic environment and resources that made this research possible. To everyone who contributed in one way or another to the completion of this thesis, I remain deeply thankful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACF	Autocorrelation Function.
AIC	Akaike Information Criterion.
ARIMA	Autoregressive Integrated Moving Average.
ARIMAX	ARIMA with Exogenous Variables.
BIC	Bayesian Information Criterion.
CV	Cross-Validation (walk-forward / time-ordered).
FSI / FSIs	Financial Soundness Indicator(s).
FX	Foreign Exchange.
IMF	International Monetary Fund.
LSTM	Long Short-Term Memory.
MAE	Mean Absolute Error.
MAPE	Mean Absolute Percentage Error.
MSE	Mean Squared Error.
M1	Monetary Aggregate M1 (narrow money).
M2	Monetary Aggregate M2 (broad money).
NPL	Non-Performing Loan.
PACF	Partial Autocorrelation Function.
QLSTM	Quantum Long Short-Term Memory.
QML	Quantum Machine Learning.
QPM	Quarterly Projection Model.

QSVR	Quantum Support Vector Regression.
QSVM	Quantum Support Vector Machine.
QXGBoost	Quantum Feature-Map XGBoost.
R^2	Coefficient of Determination (R-squared).
RBF	Radial Basis Function (kernel).
RMSE	Root Mean Squared Error.

CHAPTER ONE

INTRODUCTION

1.1 Background

Exchange-rate dynamics in Ghana are influenced by external price shocks, inflationary pressure, debt distress, and balance of payments fragility. These factors have increased the importance of accurate forecasting and risk surveillance in policymaking (World Bank Group, 2024; Baidoo & Obeng, 2024; IMF, 2024). Short- to medium-term planning for the Bank of Ghana and the Ministry of Finance has been made more difficult by episodes of fast cedi depreciation and domestic fiscal adjustments that frequently occur in tandem with external commodity swings and changes in market sentiment. Forecast quality in these settings has a direct impact on macroprudential buffers, debt rollover strategy, and monetary-policy calibration.

The transparent econometric baselines of ARIMA, ARIMAX, and VAR continue to be the industry standard for generating point and interval forecasts of exchange rates and other macro variables in the Bank of Ghana Quarterly Projection Model workflow (Bank of Ghana, 2022). Although these tools are useful for scenario analysis and communication, they may not function well during regime changes and structural disruptions such as; debt restructuring episodes or abrupt changes in terms of trade, or as the dimensionality of the data set increases due to their near-linearity and weak stationarity assumptions (Tiffin, 2016; World Bank, 2023; IMF, 2024). Specifically, even with the addition of exogenous regressors, the nonlinear interaction of the money supply, credit conditions, public debt, and external pricing can result in asymmetric and state-dependent reactions in the USD/GHS rate that are difficult for linear autoregressions to represent.

Support Vector Regression (SVR) provides a morally sound substitute that fits in nicely with these difficulties. SVR allows for nonlinear mappings using kernels like the radial basis function (RBF), controls model capacity by regularization (parameter C), and maximizes the margin while minimizing an ε-insensitive loss (Vapnik, 1998; Smola & Schölkopf, 2004). The RBF kernel flexibly approximates unknown nonlinear interactions between the exchange rate and macro factors, while the margin-based objective intuitively lowers overfitting risk in small-to medium-sample settings typical of monthly macro panels. The kernel width (γ) and ε provide interpretable knobs to balance bias and variance by determining how "wiggly" the function can be and how much inaccuracy is acceptable. This combination, regularization + kernels makes SVR a strong candidate when the signal is nonlinear, noisy, and multivariate, yet the sample is not large enough to safely fit high-capacity deep models.

When combined with rigorous scaling and time-aware cross-validation, SVR has demonstrated empirically competitive out-of-sample accuracy across macro-financial prediction tasks, such as inflation and exchange rates (Rundo et al., 2019; Gyamerah, 2019). For Ghana's USD/GHS forecasts, two characteristics are especially important. First, compared to squared-error regression, the leverage of isolated extremes is reduced by the ε-insensitive loss and margin objective, which is robust to outliers and heavy-tailed shocks. Second, if hyperparameters (C, ε, and γ) are adjusted using leakage-safe techniques, SVR can achieve good fit quality without a significant number of parameters. SVR's convex training objective (for fixed kernel) and minimal collection of hyperparameters make it a viable alternative to more complex deep architectures in policy scenarios where model transparency and stability are important.

This study therefore centers on SVR as the primary machine-learning method for USD/GHS forecasting on Ghana's macro panel, while retaining ARIMA/ARIMAX/VAR as institutional

baselines and including a minimal set of classical and quantum baselines to quantify SVR's advantage on the same data and pipeline. In particular, we compile a single monthly dataset (2014–2023) from Bank of Ghana sources, which includes the interbank USD/GHS rate as the target and macro predictors that include trade flows, policy/benchmark rates, debt aggregates, monetary aggregates such as (M2 and credit), external prices (oil, gold, and cocoa), and a few financial-soundness indicators. We use a two-phase design approach: (i) an ex-ante operational run (refit through Dec-2023 → forecast Jan–Dec 2024) to evaluate 12-month-ahead performance and uncertainty bands similar to Bank of Ghana practice; and (ii) an ex-post backtest (train 2014–2022 → test 2023) to compare models on a held-out window (Hyndman & Athanasopoulos, 2018; Bank of Ghana, 2022). Evaluation combines accuracy measures (MAE, RMSE, MSE, MAPE, R^2) with computational metrics (training time, inference delay, peak RSS memory) to represent genuine institutional restrictions.

For interpretability and interval benchmarking, we maintain ARIMA/ARIMAX/VAR as Bank of Ghana baselines in addition to SVR. We also include LSTM and XGBoost as representative classical ML comparators, which are models that are known to capture sequential dependencies and nonlinear interactions but may need additional data and careful regularization to generalize well on macro series (Rundo et al., 2019). In order to determine whether new quantum feature maps and kernel constructions provide any immediate advantages on Ghanaian data, we additionally use quantum/hybrid alternatives (such as QLSTM, QSVR, and QXGBoost) only as exploratory comparators (Aaronson, 2013). The application of quantum machine learning in economics is still in its infancy and is mostly simulated, with unanswered concerns regarding stability, calibration, and cost under policy-quality constraints, despite the fact that it promises expressive hypothesis classes and, eventually, hardware-level speedups (Benedetti et al., 2019;

Havlíček et al., 2019; Schuld & Killoran, 2019; Afrifa-Yamoah, Gyasi, & Owusu, 2024; Askeruld, 2023).

The contribution is threefold when framed in this manner. Using a macro panel unique to Ghana, it first demonstrates and evaluates SVR as a workable, policy-relevant substitute for Bank of Ghana linear baselines, measuring improvements in accuracy and robustness with like-for-like evaluation. Second, it places SVR in relation to reliable baselines, such as exploratory QML, classical ML, and econometric, so stakeholders are aware of the relative advantages and disadvantages. Third, it bridges methodological advancements and operational use by putting results into practice through a Streamlit prototype that consumes the Bank of Ghana macro panel, allows analysts to select models and basic scenarios, and exposes comparable metrics and 95% intervals with downloadable tables and figures (Bank of Ghana, 2022; IMF, 2024; Hyndman & Athanasopoulos, 2018). Overall, the chapter establishes why SVR is well-matched to Ghana's forecasting problem and how the study's design provides fair, policy-oriented comparisons against Bank of Ghana baselines and selected alternatives.

Few studies have looked at the theoretical applicability of econometric and machine-learning forecasting in African macroeconomic contexts, which are marked by short time spans, irregular data, and structural shocks, despite significant advancements in these fields. Linear autoregressive models frequently fail in unstable regimes, according to data from developing countries like Nigeria, Kenya, and India (Adewuyi et al., 2022; Kumar & Singh, 2023). A distinct theoretical gap is defined by the lack of region-specific research: small-sample machine-learning forecasting frameworks, especially kernel-based methods like SVR, must be modified and validated for African exchange-rate prediction.

1.2 Problem Statement

The Bank of Ghana's classic econometric models, ARIMA, ARIMAX, and VAR, are unable to accurately predict the USD/GHS exchange rate due to its nonlinear, shock-driven, and regime-shifting behavior. Although Ghana's exchange-rate fluctuations are heavily impacted by nonlinear interactions between public debt, fiscal adjustments, external commodity prices, global liquidity conditions, and monetary aggregates, these models assume near-linearity, stable parameters, and stationary relationships (Bank of Ghana, 2022; World Bank, 2023; IMF, 2024). Forecast inaccuracy, broader uncertainty ranges, and less utility for monetary policy timing, debt rollover planning, and FX liquidity operations are the outcomes of breaking these assumptions.

Using kernel functions and margin-based regularization, Support Vector Regression (SVR) provides a theoretically sound substitute that can simulate nonlinear patterns, structural breaks, and asymmetric adjustments (Vapnik, 1998; Smola & Schölkopf, 2004). Nevertheless, no study has used leakage-safe preprocessing, consistent temporal splits, accuracy, and computational metrics to thoroughly benchmark SVR against the Bank of Ghana's institutional baselines on the same macroeconomic panel. The absence of such evidence raises an unanswered methodological and policy question: under Ghana's actual data conditions, do kernel-based SVR models offer significantly better and practically achievable forecasts for USD/GHS than ARIMA/ARIMAX/VAR?

In order to close this gap, this study uses Ghana's macroeconomic data from 2014 to 2023 to evaluate both traditional, classical machine learning and quantum-inspired forecasting models under a single, repeatable pipeline. Additionally, it develops a forecasting prototype that is ready for stakeholders and shows institutional applicability.

1.3 Research Objectives

1.3.1 General Objective

To develop and benchmark a kernel-based Support Vector Regression model against the Bank of Ghana's traditional econometric models (ARIMA, ARIMAX, VAR) using a unified, leakage-safe evaluation framework, and to determine the extent to which SVR improves forecasting accuracy, computational efficiency, and operational stability for Ghana's USD/GHS exchange rate.

1.3.2 Specific Objectives

For specific objectives, this study seeks to achieve the following:

i. To implement and optimize SVR, ARIMA, ARIMAX, VAR, LSTM, XGBoost, and selected QML models using the same Ghanaian macroeconomic dataset under a leakage-safe preprocessing and chronological-split framework.

ii. To compare the models' predictive accuracy, computational efficiency, and uncertainty behaviour under both ex-post backtesting and ex-ante 12-month operational forecasting.

iii. To develop a policy-ready forecasting prototype that integrates the top-performing models and provides interpretable outputs, uncertainty bands, and scenario-based forecasts for operational use.

1.4 Scope of the Research

This analysis uses monthly macroeconomic data from 2014 to 2023, mostly from the Bank of Ghana, to forecast the USD/GHS exchange rate in the near to medium term. The scope is restricted to comparing specific quantum/hybrid models, classical machine-learning models, and conventional econometric models under a single evaluation process. Due to hardware limitations, quantum experiments are carried out under simulation; no attempt is made to rebuild central-bank structural models, perform high-frequency forecasting, or implement models on live quantum

hardware. Additionally, the study restricts its prototype development to a decision-support tool based on Streamlit that is intended for institutional adaption and demonstration rather than full-scale integration.

1.5 Research Hypotheses

In light of the stated goals, the following theories are put out for empirical investigation:

i. **H1:** Support Vector Regression (SVR) achieves significantly higher predictive accuracy than ARIMA/ARIMAX/VAR models for forecasting the USD/GHS exchange rate.

ii. **H2:** SVR demonstrates superior computational efficiency (training time and memory use) compared with LSTM and XGBoost when applied to Ghana's small-sample macro data (Vapnik, 1998; Rundo et al., 2019).

iii. **H3:** Quantum Machine Learning (QML) variants such as QSVR and QLSTM do not yet yield statistically significant performance gains over classical SVR under current simulation constraints (Benedetti et al., 2019; Schuld & Killoran, 2019).

1.6 Research Questions

This research problem aims to address the following key questions:

i. How do SVR classical model compare to quantum models classical models and traditional models in forecasting Ghana's exchange rate?

ii. What are the differences in predictive accuracy, computational efficiency, and 12-month-ahead forecasts (2024) between quantum, classical, and Bank of Ghana models?

iii. How can a prototype forecasting app be designed and evaluated to facilitate the practical adoption of advanced machine learning models for economic forecasting in Ghana?

1.7 Methodology Summary

This study adopts a comparative analysis framework to evaluate the predictive accuracy, computational efficiency, and 12-month-ahead (2024) operational forecasts of the actual 2024 data again the forecasts of classical models, quantum models and Bank of Ghana's traditional econometric models for forecasting Ghana's exchange-rate dynamics. The methodology integrates data collection, preprocessing, model development, evaluation, and comparison (Cao et al., 2020). Monthly economic data are sourced primarily from the Bank of Ghana for 2014–2023 and include the USD/GHS exchange rate (target) alongside key macro indicators, debt aggregates (total, external, domestic, service), monetary aggregates (M1, M2, credit by sector), financial-soundness measures (capital adequacy, NPL ratios, liquidity), commodity prices (gold, cocoa, oil), merchandise trade flows, and interest rates, yielding a merged panel of roughly 120 observations to be validated during preprocessing. Data preparation ensures time-series suitability via cleaning (interpolation/imputation and record correction), leakage-safe normalization (Min–Max or Z-score), and feature engineering (lags, rolling aggregates, interactions), followed by a chronological split (typically 80/20) to prevent leakage (Bousselmi & Zaher, 2020; Abayomi & Asumadu, 2021). Model development spans three families: (i) econometric baselines ARIMA, ARIMAX, and VAR; (ii) classical ML models LSTM, Support Vector Regression (SVR), and XGBoost and (iii) QML/hybrid models including QLSTM, QSVM, and quantum feature–map XGBoost. Classical models are implemented in TensorFlow/Keras and scikit-learn with grid or randomized search and time-aware cross-validation; quantum models use Qiskit, PennyLane, or TensorFlow Quantum under simulation given current hardware limits (Benedetti et al., 2019; Schuld & Petruccione, 2021). All models are assessed with MAE, RMSE, MSE, MAPE, and R^2, alongside compute metrics (training time, inference latency, peak memory; for QML also qubit count and circuit

depth), and compared on their 2024 operational forecasts with uncertainty quantification (native intervals for ARIMA/ARIMAX; bootstrap residual bands for ML/QML), to judge accuracy, cost, and policy integration suitability (Hyndman & Athanasopoulos, 2018; Cao et al., 2020).

1.8 Significance of the Study

The necessity of increasing exchange-rate forecasting accuracy under structural volatility is shown by comparative data from other growing economies. When dealing with nonlinear shocks and limited macro panels, machine-learning models perform better than conventional ARIMA-type models, according to studies conducted in Nigeria, Kenya, and India (Adewuyi et al., 2022; Ngugi & Mutua, 2023; Kumar & Singh, 2023). The study's international rationale is strengthened and its contribution to cross-regional methodological improvement is demonstrated by placing Ghana within this larger empirical context.

This study is significant because it delivers a rigorous, policy-relevant comparison of forecasting approaches that the Bank of Ghana could realistically deploy for USD/GHS management, spanning its established econometric baselines (ARIMA/ARIMAX/VAR), modern classical machine-learning methods, with a focus on Support Vector Regression (SVR), and exploratory quantum/hybrid variants (QLSTM, QSVR, QXGBoost). By evaluating all models on the same Bank of Ghana macro panel and reporting both predictive accuracy (MAE, RMSE, MSE, MAPE, R^2) and computational costs (training/inference time, peak memory), the work provides an evidence base for selecting methods that balance accuracy with operational efficiency under real institutional constraints (Bank of Ghana, 2022; IMF, 2024). In practical terms, the study demonstrates that advanced data-driven tools can augment Bank of Ghana Quarterly Projection Model workflow by improving short-horizon exchange-rate forecasts and by quantifying uncertainty through native or bootstrap intervals, directly informing monetary policy, debt service

planning, and macroprudential oversight in a high-volatility environment. At the same time, it clarifies the state of quantum machine learning for economic forecasting: while quantum computing is widely argued to offer efficiency and representational advantages for high-dimensional learning (Preskill, 2018; Benedetti et al., 2019), applied evidence in African contexts remains limited; this thesis narrows that gap by testing QML on real Ghanaian data and documenting where it helps, where it falls short, and what computational trade-offs are involved (Schuld et al., 2015; Khan et al., 2020). The accompanying Streamlit prototype translates these findings into a usable artifact, model picker, comparable metrics, forecast intervals, and downloadable outputs, bridging research and decision support for analysts and policymakers (IMF, 2024). More broadly, the results contribute to global discussions on how, when, and at what cost emerging quantum-enhanced techniques might complement classical ML and econometrics in complex economic systems, offering a concrete foundation for subsequent methodological refinement and interdisciplinary collaboration (Farhi & Neven, 2018; Thapliyal et al., 2018). Ghana's evidence-based macroeconomic management is strengthened in this thesis, which furthers the UN 2030 Agenda (United Nations, 2015). It supports SDG 8: Decent Work & Economic Growth through improved policy timing, debt rollover planning, and FX liquidity operations that lower business uncertainty by enhancing short- to medium-term USD/GHS forecasts and uncertainty communication (UN DESA, Goal 8). Data pipelines, model benchmarking, and a prototype app are examples of a replicable, analytics-driven process that is operationalized to support SDG 9: Industry, Innovation & Infrastructure in public-sector decision systems (UN DESA, Goal 9). Accountable model selection, auditable pipelines, and transparent metrics all support SDG 16: Peace, Justice & Strong Institutions (UN DESA, Goal 16).

The study theoretically advances the field of exchange-rate forecasting by applying statistical-learning theory to small-sample, nonlinear macroeconomic data typical of African contexts, specifically the ε-insensitive loss and margin-maximization principles of Support Vector Regression (Vapnik, 1998; Smola & Schölkopf, 2004). The paper provides additional empirical evidence on the generalization features of kernel-based approaches in developing-economy contexts by explicitly testing these principles against linear econometric baselines.

1.9 Organization of the Study

There are five chapters in the thesis. The study's background, research challenge, aims, scope, hypotheses, and significance are all introduced in Chapter One. In Chapter Two, the theoretical, empirical, and methodological literature on exchange-rate forecasting is reviewed. The limits of conventional econometric models are highlighted, and the use of SVR and advanced learning techniques is encouraged. The research design, data sources, preprocessing procedures, model building pipeline, assessment metrics, and comparative framework are all presented in Chapter 3. In Chapter Four, the unified performance scorecard and prototype outputs are presented along with the empirical results of the econometric, ML, and QML models. In Chapter Five, the results are discussed in relation to the research questions, the contributions to knowledge are outlined, policy and research recommendations are made, study limits are noted, and the thesis is concluded.

CHAPTER TWO

LITERATURE REVIEW

2.1. Introduction

This chapter provides a comprehensive review of the literature relevant to forecasting Ghana's exchange-rate dynamics, but, unlike a neutral survey, it explicitly centers Support Vector Regression (SVR) as the primary modelling choice for USD/GHS, with traditional econometric tools and a minimal set of ML/QML variants retained as comparators. Exchange-rate stability remains mission-critical for Ghana's policy credibility, external balance, and debt sustainability; hence the Bank of Ghana has long relied on transparent econometric baselines, ARIMA, ARIMAX, and VAR within the Quarterly Projection Model (QPM) (Bank of Ghana, 2022; IMF, 2024). These baselines, however, rest on near-linearity and (quasi)-stationarity assumptions and often degrade under structural breaks and high-dimensional interactions across debt, money supply, and external prices, conditions that have characterized Ghana's recent macro regime (Hyndman & Athanasopoulos, 2018; Sargent, 2018). SVR is foregrounded here because its margin-based loss, capacity-control via regularization (C, ε-insensitive loss), and kernel mappings (notably RBF) make it well-suited to nonlinear, multivariate relationships in small-to-medium monthly samples like Ghana's macro panel; empirically, SVR has demonstrated strong out-of-sample performance on financial and macro time series with complex drivers (Rundo et al., 2019; Gyamerah, 2019). Given evidence of nonlinear exchange-rate mechanisms in Ghana, this review therefore centers on SVR as the primary forecasting model for USD/GHS. Traditional Bank of Ghana baselines (ARIMA/ARIMAX/VAR) and a deliberately small set of classical/quantum comparators (LSTM, XGBoost, and exploratory QML kernels/layers) are kept to quantify SVR's advantage on the same data and pipeline, not to diffuse the focus (Bank of Ghana, 2022; Schuld &

Petruccione, 2018). Quantum machine learning, leveraging superposition/entanglement for expressive feature maps, remains exploratory in this policy domain and is included only to benchmark feasibility and computational trade-offs rather than to supplant SVR (Benedetti et al., 2019; Havlíček et al., 2019). The remainder of the chapter is organized to (i) review Ghana's exchange-rate forecasting practice and macro-predictors; (ii) synthesize the theory and evidence behind ARIMA/ARIMAX/VAR as institutional baselines; (iii) develop the SVR rationale in depth and contrast it briefly with LSTM/XGBoost; and (iv) outline the emerging QML literature. Throughout, we emphasize benchmarking frameworks aligned to the thesis design: consistent accuracy metrics (MAE, RMSE, MAPE, R^2), computational measures (train/infer time, peak RSS), and 12-month (2024) operational forecasts generated on the same Bank of Ghana macro dataset to support a policy-relevant comparison of SVR against incumbent tools (IMF, 2024; Hyndman & Athanasopoulos, 2018).

Even though Support Vector Regression (SVR) is the main model in this review, a more neutral thematic flow is necessary to prevent coming off as advocacy. A fair analysis should show how several forecasting models (ARIMA, VAR, SVR, LSTM, XGBoost, and QML) are motivated by Ghana's macro-factors and how they conceptually address the complexity of the USD/GHS series (Hyndman & Athanasopoulos, 2018; Benedetti et al., 2019). Without coming across as predetermined, this neutrality enhances academic objectivity and explains why SVR is ultimately chosen as the study's preferred model. In line with standards for critical review, this chapter not only synthesizes prior research but also evaluates methodological assumptions, identifies contradictions across studies, and explicitly highlights areas where evidence remains inconclusive or contested (Makridakis et al., 2018).

2.2 Theoretical Framework for Exchange Rate Forecasting

Well-known theories of international finance can be used to conceptually explain the behavior of the USD/GHS exchange rate. The macroeconomic variables used for the econometric baselines and machine-learning models are supported by these theories.

According to UIRP, prospective currency appreciation or depreciation is predicted by the difference in interest rates between two nations. The inclusion of Treasury bill rates, Ghana-US interest rate spreads, and monetary-policy indicators as explanatory variables is justified by this approach (Chinn, 2018). PPP connects changes in currency rates to levels of relative inflation. Thus, long-term factors influencing the USD/GHS exchange include price indices, money supply movements, and inflation differences (Taylor & Taylor, 2004).

The Random Walk (RW) model is a criterion that all forecasting models must surpass because the efficient-market approach maintains that exchange-rate movements are essentially unpredictable (Meese & Rogoff, 1983). When taken as a whole, these theories support the fundamental macroeconomic variables included in this research and serve as an anchor for the econometric baselines, guaranteeing the theoretical consistency of the shift to machine-learning models.

2.3 Exchange Rate Forecasting in Ghana

Exchange rate forecasting holds strategic importance for Ghana's economic management, given the central role of the exchange rate in influencing inflation, debt servicing, trade competitiveness, and overall macroeconomic stability (Bank of Ghana, 2022; World Bank, 2023). The Ghanaian cedi's value relative to major international currencies, especially the US dollar, is a key indicator monitored by policymakers, investors, and businesses. Volatility in the exchange rate has direct implications for external debt repayment costs, import and export pricing, inflationary pressures, and foreign direct investment flows (IMF, 2024; Baidoo & Obeng, 2024). Accurate exchange rate

forecasts enable the Bank of Ghana to implement timely monetary policy interventions, manage foreign reserves, and signal market expectations effectively.

Historically, the Bank of Ghana has relied on traditional econometric models for exchange rate forecasting, often integrated within its Quarterly Projection Model (QPM) and other internal analytical tools. Commonly used models include univariate time series models such as Autoregressive Integrated Moving Average (ARIMA), multivariate models like Vector Autoregression (VAR), and their variants such as ARIMAX, which incorporate exogenous variables (Bank of Ghana, 2022; IMF, 2024). These models have provided a structured approach to forecasting based on past exchange rate movements and selected macroeconomic indicators, such as external debt, money supply, interest rates, and commodity prices (Bank of Ghana, 2024). However, they are constrained by assumptions of linear relationships and stationarity, which limit their ability to capture the complex, nonlinear interactions typical of modern macro-financial systems (Hyndman & Athanasopoulos, 2018; Sargent, 2018). Additionally, these models often struggle to adapt rapidly to structural breaks, regime shifts, or external shocks, such as those experienced during Ghana's recent debt restructuring and cedi depreciation episodes (World Bank, 2023).

In response to these limitations, there has been growing interest in leveraging machine learning approaches, particularly classical ML models, for exchange rate forecasting in Ghana and comparable developing economies. Models such as Long Short-Term Memory (LSTM) networks have shown promise in capturing long-term dependencies in time series data, while Support Vector Regression (SVR) and gradient-boosted ensembles have demonstrated the capacity to model nonlinear relationships between exchange rates and macroeconomic drivers (Gyamerah, 2019). Despite this progress, classical ML methods have yet to be systematically benchmarked against

Bank of Ghana's existing forecasting frameworks using actual national datasets. Furthermore, the potential of quantum machine learning (QML) models to address some of the scalability and efficiency challenges of classical methods in this domain remains largely unexplored.

This study builds on the context of exchange rate forecasting in Ghana by introducing a structured benchmarking of traditional Bank of Ghana econometric models, classical ML algorithms, and quantum ML techniques. By employing actual Bank of Ghana datasets that include exchange rates, debt data, monetary aggregates, financial soundness indicators, commodity prices, trade flows, and interest rates, the research aims to provide a comprehensive comparative analysis that informs future forecasting practices and policy decision-making in Ghana's financial governance. Both strengths and weaknesses of linear and nonlinear models are considered to maintain analytical neutrality.

2.4 Macroeconomic Indicators and Exchange Rate Prediction

The exchange rate of a country's currency is influenced by a complex interplay of domestic and external macroeconomic variables. For Ghana, key macroeconomic indicators, including debt levels, monetary aggregates, financial soundness indicators (FSIs), commodity prices, trade flows, and interest rates, play critical roles in shaping exchange rate dynamics (Bank of Ghana, 2022; World Bank, 2023). These variables not only reflect the fundamental health of the economy but also signal risks and opportunities to international investors, directly affecting currency demand and supply in the foreign exchange market.

Debt data, including total public debt, external debt, and domestic debt service obligations, significantly influence exchange rate movements in Ghana. Rising debt levels increase fiscal vulnerability and external financing needs, often leading to depreciation pressures on the cedi due to heightened repayment risk and investor concerns (IMF, 2024; Baidoo & Obeng, 2024).

Similarly, monetary aggregates such as M1, M2, and credit to various sectors capture the liquidity position of the economy and its potential inflationary trends, which are closely watched by currency markets (Sargent, 2018). An expansion in money supply, without corresponding growth in output, can contribute to exchange rate depreciation through its impact on inflation and domestic demand for foreign currency.

Financial soundness indicators (FSIs), including capital adequacy ratios, liquidity coverage ratios, and non-performing loan (NPL) levels, serve as proxies for banking sector stability, which in turn affects confidence in the domestic currency (Bank of Ghana, 2024). A weakening of these indicators may lead to capital flight and depreciation pressures. Commodity prices, particularly of Ghana's major exports like gold, cocoa, and oil, are also strong determinants of exchange rate trends. As primary sources of foreign exchange earnings, fluctuations in these commodity prices directly impact Ghana's external balance and reserve adequacy, influencing cedi stability (World Bank, 2023).

Merchandise trade flows, encompassing both exports and imports, reflect the balance of trade position and external sector strength. Persistent trade deficits typically exert downward pressure on the cedi, while trade surpluses may support its value (Bank of Ghana, 2022). Interest rates, both policy rates and market rates, influence capital flows, investment decisions, and demand for the cedi relative to other currencies. Higher interest rates, in theory, attract foreign capital and support exchange rate stability, though the effect depends on broader macroeconomic conditions (IMF, 2024).

While these macroeconomic indicators have long been recognized in traditional exchange rate models, their complex, nonlinear interactions and dynamic relationships present challenges for conventional econometric approaches. Classical machine learning models, such as LSTM and

XGBoost, provide a way to capture these intricate patterns more effectively, while quantum machine learning offers potential advantages in handling high-dimensional data involving numerous interconnected macroeconomic indicators (Benedetti et al., 2019; Khan et al., 2020). This study therefore leverages a comprehensive set of Ghana's macroeconomic indicators to evaluate the predictive performance of classical, quantum, and traditional econometric models, offering new insights into their individual and combined roles in exchange rate forecasting. No single macro-theory fully explains USD/GHS behaviour; hence multiple models are evaluated impartially.

2.5 Traditional Econometric Forecasting Tools

Traditional econometric models have long underpinned exchange-rate forecasting in Ghana and remain embedded in the Bank of Ghana's policy workflow, particularly within the Quarterly Projection Model (QPM) (Bank of Ghana, 2022; IMF, 2024). The Autoregressive Integrated Moving Average (ARIMA) model, widely used for its transparency and parsimony, treats the exchange rate as a univariate process driven by its own lagged values and past shocks; this makes it attractive for routine monitoring, yet its linear structure and stationarity requirements reduce robustness when the data exhibit regime changes, volatility clustering, or structural breaks, features that have characterized Ghana's recent macro environment (Hyndman & Athanasopoulos, 2018; Sargent, 2018). Figure 2.1 below talks about ARIMA, a univariate structure with order selection and native prediction intervals.

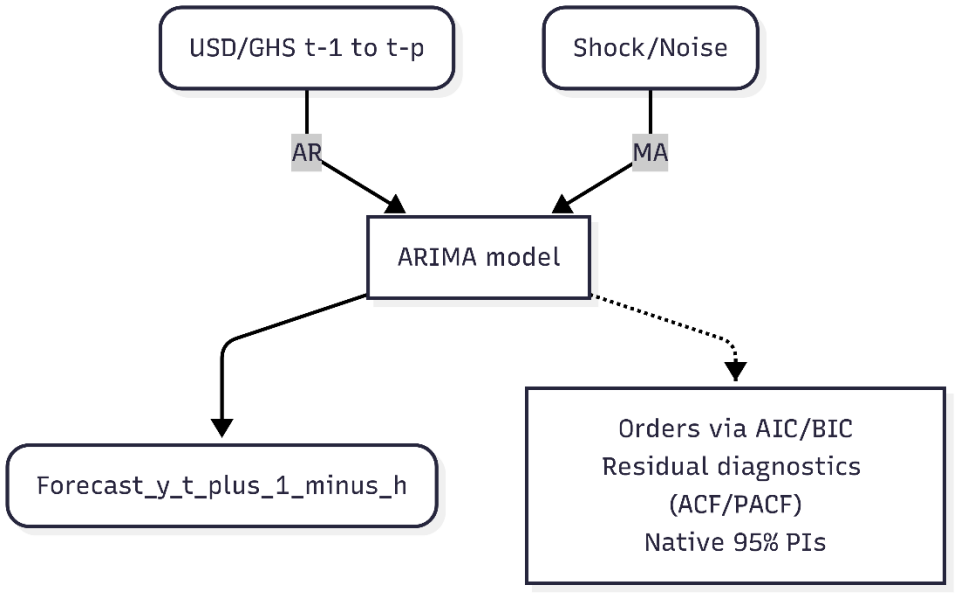


Figure 2.1: ARIMA (univariate)

The ARIMA with exogenous inputs (ARIMAX) extends this baseline by admitting macro-drivers, such as monetary aggregates (M1/M2), external debt, commodity prices, or trade flows, into the forecasting equation, thereby acknowledging the multi-cause nature of USD/GHS movements; however, ARIMAX still inherits ARIMA's linearity and can suffer from multicollinearity and coefficient instability when predictors are correlated or when underlying relationships are nonlinear (IMF, 2024). Figure 2.2 below talks about ARIMAX with standardized exogenous macro drivers and scenario inputs for 2024 run.

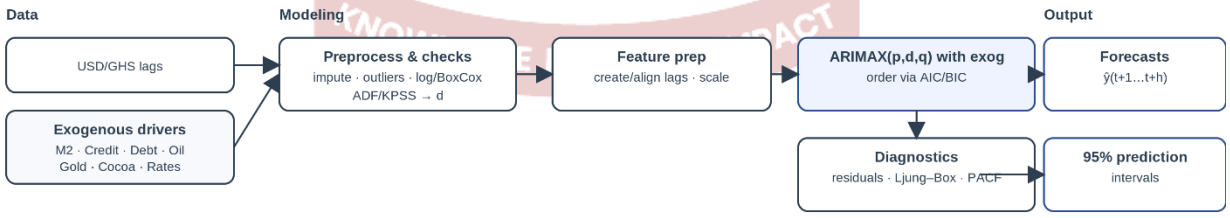


Figure 2.2: ARIMAX (with exogenous drivers)

Vector Autoregression (VAR) moves further by modeling the joint dynamics among multiple macro series, exchange rate, interest rates, inflation, and is frequently used for policy analysis and impulse responses in QPM-style frameworks because it captures feedback effects and contemporaneous interactions (Bank of Ghana, 2022); yet VAR's parameter count grows quickly with the number of variables and lags, creating over-parameterization risks, weak small-sample properties, and susceptibility to overfitting in higher-dimensional economic panels (Hyndman & Athanasopoulos, 2018). Figure 3.4 below talks about VAR system modeling joint dynamics of USD/GHS and macro drivers.

Figure 2.3: VAR (multivariate)

In short, while ARIMA/ARIMAX/VAR have provided valuable, interpretable baselines for Ghana's monetary operations, their reliance on near-linearity and stationarity and their scaling limitations under rich macro panels constrain performance during episodes of rapid depreciation, policy regime shifts, or commodity-price shocks (World Bank, 2023; IMF, 2024). Accordingly, in this thesis we treat ARIMA, ARIMAX, and VAR as institutional baselines and evaluate Support Vector Regression (SVR) directly against them on the same Ghana macro dataset, splits, and metrics to quantify SVR's practical advantage for USD/GHS forecasting. Several scholars argue that ARIMA-type models retain interpretability advantages that make them suitable for policy

contexts, despite their predictive limitations (Tiffin, 2016; Diebold & Mariano, 1995). Others defend VAR on grounds of transparency and impulse-response usefulness (Sargent, 2018). Including these scholarly counter-arguments ensures balanced critique rather than dismissing econometric baselines. These limitations do not invalidate ARIMA/VAR but justify testing them alongside ML models.

2.6 Classical Machine Learning Models for Exchange Rate Forecasting

Traditional machine-learning (ML) techniques allow for greater nonlinear and interaction effects among macroeconomic factors of the USD/GHS rate by easing the strong stationarity and near-linearity assumptions that underlie many econometric tools (Gyamerah, 2019). The main purpose of this thesis is to evaluate whether a contemporary, margin-based learner Support Vector Regression (SVR) can provide appreciable accuracy gains over the institutional baselines of the Bank of Ghana using traditional machine learning. Two additional ML comparators, LSTM and XGBoost, are included for context.

SVR applies support-vector concepts to regression by using the ε-insensitive loss to minimize a regularized empirical risk. Larger residuals are subject to slack penalties that are managed by the regularization constant CCC, while deviations within ±ε are disregarded. Through kernel mappings, most commonly the radial basis function (RBF) kernel, SVR constructs high-dimensional feature spaces in which a linear fit corresponds to a flexible, smooth nonlinear function in the original macroeconomic space (Schuld & Petruccione, 2018; Rundo et al., 2019). Two properties make SVR particularly well-matched to Ghana's macro panel. First, sample efficiency: by controlling model capacity via CCC and the kernel width γ\gammaγ (for RBF), SVR maintains good generalization in small-to-medium samples typical of monthly macro series (2014–2023). Second, robust nonlinear approximation: the RBF mapping captures the known curvature

and interactions among money supply, debt, and external prices that drive USD/GHS, relationships that often violate linear superposition. In practice, performance hinges on hyperparameter calibration; we therefore tune CCC, ϵ\epsilonϵ, and γ\gammaγ via time-series-aware cross-validation on the training window, with leakage-safe scaling of predictors, to balance bias–variance and stabilize out-of-sample error (Gyamerah, 2019). This combination, margin-based loss, kernelized nonlinearity, and disciplined tuning, motivates centering SVR as the primary ML candidate evaluated against ARIMA/ARIMAX/VAR in this study. Figure 3.5 below talks about SVR (RBF) pipeline with scaler fit on train only; time-ordered hyperparameter tuning.

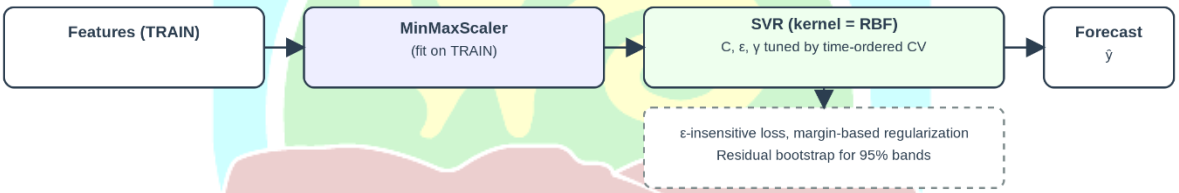


Figure 2.4: SVR (RBF) with leakage-safe scaling

LSTM networks, a gated recurrent architecture, learn long-range temporal dependencies and have shown strong performance in FX and macro forecasting where sequence effects matter (Abayomi & Asumadu, 2021). They provide a useful benchmark for sequence modeling but can be data-hungry, sensitive to hyperparameters (depth, hidden units, learning rate, dropout), and computationally heavier than margin-based methods on modest monthly panels. Here LSTM serves as a comparator to contextualize SVR's accuracy on the same inputs and splits. Figure 3.6 below talks about compact many-to-one LSTM architecture to avoid over-parameterization on short series.

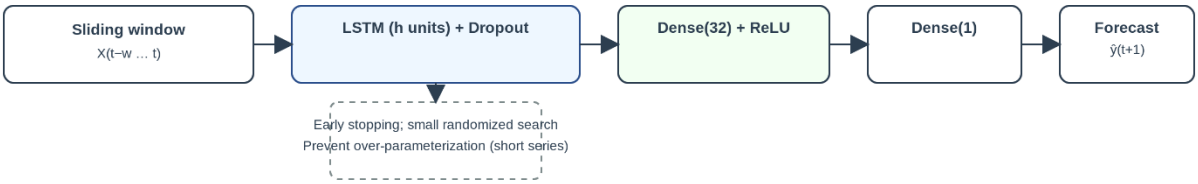


Figure 2.5: LSTM (compact many-to-one)

XGBoost (gradient-boosted trees) builds an additive ensemble of shallow trees with shrinkage, column/row subsampling, and L1/L2L_1/L_2L1/L2 regularization, offering strong performance on tabular macro features and automated handling of interactions (Chen & Guestrin, 2016). While often competitive on structured data, boosted trees may smooth or lag sudden volatility spikes and require careful early-stopping/tuning to avoid overfit in short panels. In this thesis, XGBoost is included as a comparator to benchmark SVR's performance under the same pipeline and evaluation criteria.

Figure 3.7 below talks about XGBoost gradient-boosted tree ensemble for nonlinear interactions in tabular macro data.

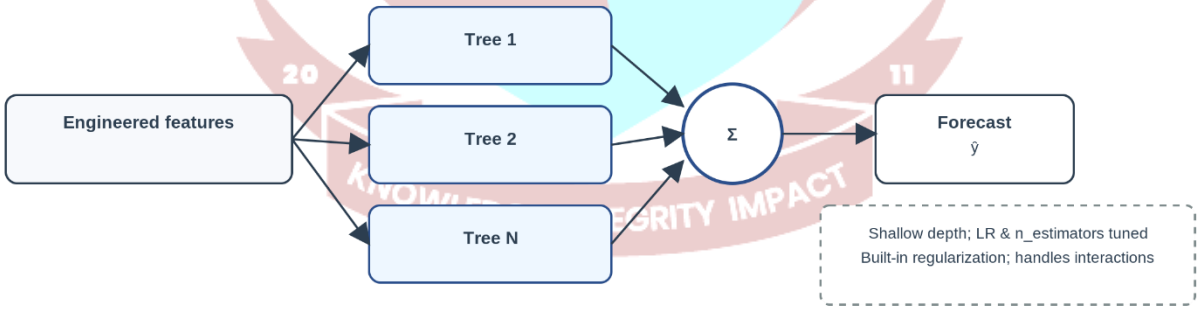


Figure 2.6: XGBoost (tree ensemble)

Although machine learning approaches provide flexibility in capturing nonlinear macro-financial interactions, they have limitations that must be acknowledged. Overfitting, hyperparameter sensitivity, and instability in short samples, which are conditions common to Ghana's monthly

macro panel, and can plague classical machine learning models (Gyamerah, 2019). In short series, LSTM architectures may need a lot of tuning and run the danger of learning noise instead of signal (Abayomi & Asumadu, 2021).

Despite their theoretical expressiveness, QML techniques are constrained in their real-world implementation due to hardware immaturity, limited qubit counts, and the computational complexity of circuit simulation (Preskill, 2018; Benedetti et al., 2019). Acknowledging these flaws shows balanced scholarship and maintains objectivity.

2.7 Structural Risk Minimization and Support Vector Regression (SVR)

Vapnik's Statistical Learning Theory, namely the idea of Structural Risk Minimization (SRM), serves as the theoretical basis for Support Vector Regression. SRM balances model complexity and error tolerance to minimize the generalization error on unseen data, in contrast to standard Empirical Risk Minimization employed in econometrics and neural networks (Vapnik, 1998). SVR maps nonlinear macro-financial interactions into high-dimensional feature spaces where linear separation is feasible by using the ε-insensitive loss function and the kernel technique, usually the Radial Basis Function (RBF) kernel (Smola & Schölkopf, 2004). The strong performance of SVR in noisy, nonlinear, and small-sample environments a prominent feature of Ghana's monthly macroeconomic datasets, is explained by this theoretical framework.

2.8 Quantum Machine Learning in Financial Forecasting

Quantum machine learning (QML) explores quantum circuits, via feature maps, kernels, and variational layers, to represent complex functions and potentially accelerate learning on high-dimensional data (Benedetti et al., 2019; Schuld & Petruccione, 2021). Quantum feature maps and kernels (Havlíček-style embeddings) can, in principle, separate patterns that are hard for classical kernels, while hybrid variational circuits aim to augment expressivity in sequence models

(Aaronson, 2013). However, current applications in economics operate primarily in simulation, given limited, noisy hardware and unresolved questions about end-to-end advantage on real policy datasets (Preskill, 2018; Schuld & Killoran, 2019). Figure 3.8 below talks about QSVR using a simulated quantum feature map/kernel, classical training with quantum kernel.

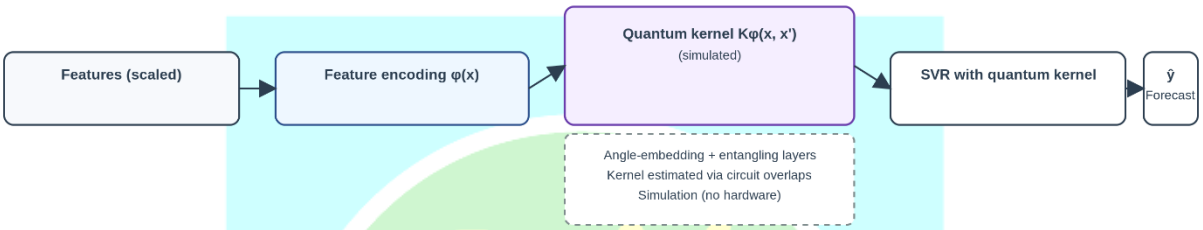


Figure 2.7: QSVR (quantum kernel, simulated)

Figure 3.9 below talks about QLSTM augmenting a classical LSTM with a variational quantum layer (simulated).

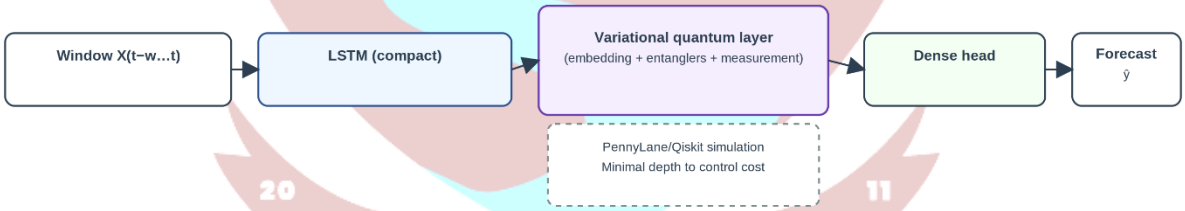


Figure 2.8: QLSTM (variational quantum layer, simulated)

Figure 2.9 below talks about quantum Machine Learning Models, right after the text describing QXGBoost

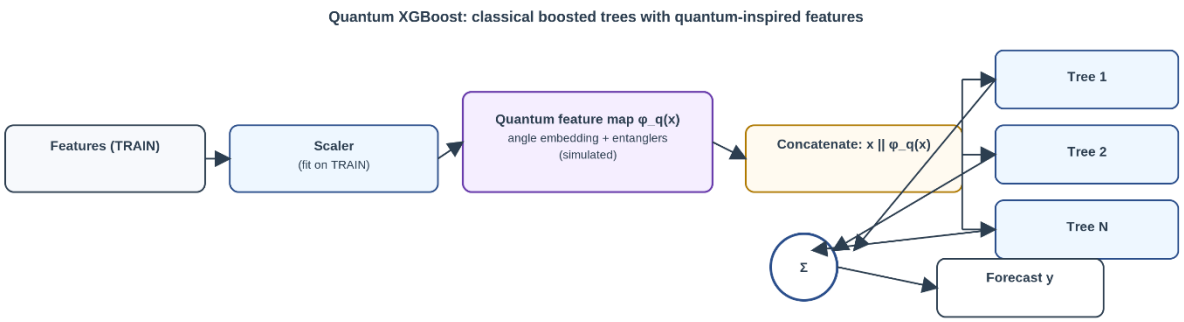


Fig. 2.9: QXGBoost (quantum-enhanced features → boosted trees)

Accordingly, QML in this thesis is explicitly exploratory and retained to contextualize SVR's performance on the same Ghana macro panel and evaluation pipeline. We implement three lightweight comparators: a QLSTM (classical LSTM with a small variational quantum layer), a QSVR (classical SVR with a precomputed quantum kernel/feature map), and a quantum-feature-map + XGBoost hybrid. All are trained in simulation with conservative circuit depth and qubit counts to mirror realistic constraints. The aim is not to supplant SVR but to show whether quantum-inspired representations show early promise relative to classical baselines under identical data, splits, and metrics; results are interpreted with caution given hardware immaturity and the small-sample nature of monthly macro data (Benedetti et al., 2019; Havlíček et al., 2019; Schuld & Petruccione, 2021).

It is crucial to stress that, despite their theoretical promise, QML techniques are still mostly experimental in order to preserve evaluative neutrality. Instead of being used in practical macroeconomic forecasting, their current benefits are primarily shown in controlled simulation scenarios (Schuld & Killoran, 2019). Furthermore, their immediate policy relevance in developing-economy environments like Ghana is limited by hardware limitations, gate noise, and simulation costs. As a result, QML is viewed in this study as experimental rather than a better option than classical models.

2.9 Comparative Framework: Linear vs Nonlinear Modelling Paradigms

The assumptions of linear econometric models like ARIMA and VAR are called into question by the nonlinear, chaotic, and regime-shifting behaviors of exchange-rate fluctuations (Hyndman & Athanasopoulos, 2018). Because of stationarity assumptions and predetermined functional forms, linear econometric models are less flexible in the face of shocks, structural breaks, and non-additive interactions. Without applying restrictive parametric assumptions, machine-learning models (SVR, LSTM, XGBoost) employ a data-driven paradigm that can reveal nonlinear and interaction-based patterns. This paradigm is extended by quantum machine learning (QML), which makes use of quantum feature spaces that could potentially represent higher-order patterns more effectively (Schuld & Killoran, 2019). This comparison framework aligns the entire modeling architecture with the characteristics of Ghana's exchange-rate dynamics and supports the inclusion of all model classes in the study.

2.10 Usability Frameworks in Forecasting Research

The fundamental issue driving the usage of Support Vector Regression (SVR) and associated advanced models is the inability of conventional econometric models to capture nonlinear relationships and adjust to structural discontinuities. The incapacity of ARIMA/ARIMAX/VAR to effectively simulate the nonlinear, dynamic behavior of Ghana's exchange rate is the research problem that this directly connects to the literature (Hyndman & Athanasopoulos, 2018; Bank of Ghana, 2022). Reiterating this connection improves the conceptual coherence between the goals of the research and the literature review.

The main issue noted in the research is that the nonlinear correlations, structural discontinuities, and regime shifts that define Ghana's exchange-rate dynamics are difficult for conventional econometric models like ARIMA, ARIMAX, and VAR to describe. Under macro-financial

volatility, debt shocks, and fluctuations in commodity prices, their linear specification and stationarity assumptions frequently result in forecasting errors (Hyndman & Athanasopoulos, 2018; Bank of Ghana, 2022). Advanced solutions like Support Vector Regression (SVR), which provides kernel-based nonlinear modeling and greater generalization in small-sample situations, are directly motivated by these limitations. Reiterating this restriction in the literature review establishes a clear conceptual link to the research issue and explains why the current study uses a single benchmarking methodology to assess SVR against Bank of Ghana baselines.

Because it offers an organized framework for assessing, contrasting, and validating the performance of rival models, usability is a crucial part of forecasting research. Researchers and practitioners can use benchmarking to assess whether new forecasting tools such as quantum machine-learning (QML) and classical machine-learning (ML) models offer significant improvements over conventional econometric techniques and operational tools employed by organizations like the Bank of Ghana (Cao et al., 2020; Hyndman & Athanasopoulos, 2018). Claims of model superiority are transparent, rigorous, and repeatable when they are supported by a robust usability framework.

Due to their extensive use in macroeconomic forecasting and central bank operations, traditional forecasting studies usually compare novel models to linear econometric baselines like ARIMA, ARIMAX, and Vector Autoregression (VAR) (Sargent, 2018; IMF, 2024). These models' interpretability and institutional acceptance make them crucial points of reference. However, interest in alternative modeling frameworks has increased due to their limited capacity to address nonlinearities, interaction effects, and regime shifts—features that characterize USD/GHS dynamics (Diebold & Mariano, 1995).

These econometric baselines have recently been used to benchmark traditional machine learning models like SVR, LSTM, and XGBoost in a variety of domains, including financial market volatility, macro-risk forecasting, inflation modeling, and foreign currency prediction (García et al., 2023). These experiments demonstrate the adaptability of machine learning models in identifying nonlinear structures that are difficult for conventional methods to identify. In addition to computational indicators like training time, memory usage, and scalability, benchmarking usually assesses performance using predictive accuracy metrics like Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Mean Square Error (MSE), and coefficient-of-determination (R^2) (Makridakis et al., 2018; Hyndman & Athanasopoulos, 2018).

By contrasting quantum algorithms with both classical ML and econometric models, emerging QML research has sought to expand benchmarking frameworks. Quantum-specific resource metrics like qubit count, circuit depth, gate complexity, and simulator overhead are incorporated into recent contributions, such as QuLTSF: Long-Term Time-Series Forecasting with Quantum Machine Learning (arXiv, 2024) and Quantum Reservoir Computing for Realized Volatility Forecasting (arXiv, 2025), which assess predictive accuracy and computational efficiency (Benedetti et al., 2019; Schuld & Petruccione, 2021). Despite these developments, no known study has used actual macroeconomic data from a developing nation to benchmark QML, traditional ML models, and Bank-of-Ghana econometric tools.

Using a harmonized dataset and identical evaluation metrics, the current study employs a thorough benchmarking framework to systematically compare Bank of Ghana's traditional econometric models (ARIMA, ARIMAX, VAR), classical ML algorithms (SVR, LSTM, XGBoost), and QML techniques (QLSTM, QSVM, Quantum Feature-Map XGBoost, Quantum Reservoir Computing). Predictive accuracy, computational efficiency, scalability, model stability, and policy

preparedness are among the comparing dimensions. In order to guide forecasting practice and technology uptake in Ghana's financial-governance ecosystem, this integrated approach guarantees that findings are both practically and intellectually meaningful.

This study makes a distinction between ex-post model evaluation and ex-ante operational forecasting in accordance with worldwide forecasting-usability criteria. Ex-post benchmarking computes error metrics and quantitatively compares models using a fixed hold-out sample (January–December 2023). Ex-ante forecasting produces a 12-month prognosis for 2024 with uncertainty bands after refining each model using all data up to December 2023. The forecasting technique best practices (Hyndman & Athanasopoulos, 2018; Tashman, 2000; Makridakis et al., 2018) are in line with this two-track system. Three common assumption sets are used for models that need exogenous routes, such as ARIMAX, VAR, and ML models with macroeconomic drivers:

i. Status-quo (last available value carried forward),
ii. Drift/mean-growth projection, and
iii. VAR-projected scenarios.

To enable users to evaluate operational realism and sensitivity, the selected assumption set is displayed with forecast outputs (Armstrong, 2001; Diebold & Mariano, 1995). This guarantees that organizations like the Bank of Ghana can use the projections for scenario planning, policy analysis, and decision support.

2.11 Conceptual Framework Linking Ghana's Macroeconomic Factors to Forecasting Models

This conceptual framework shows how exchange-rate dynamics are influenced by Ghana's macroeconomic variables and how various model classes react to these factors. The nonlinear and structural behavior of the USD/GHS rate is shaped by Ghana's macro determinants, which include debt levels, monetary aggregates, FSIs, commodity prices, trade flows, and interest rates (World Bank, 2023; IMF, 2024).

In the framework:

- Input Layer: Ghana's macroeconomic drivers
- Model Layer: ARIMA/ARIMAX/VAR (linear), SVR (kernel nonlinear), LSTM (sequence-based), XGBoost (tree interactions), QML (quantum feature maps)
- Output Layer: Forecasted exchange rate + uncertainty bands

This diagrammatic link explains why SVR and ML models handle nonlinearities better, why QML is still in the exploratory stage because of simulation and hardware constraints, and why linear models suffer under structural breaks (Schuld & Petruccione, 2021; Havlíček et al., 2019). The logical flow between the literature, the research challenge, and the model selection is strengthened and transparency is improved by this clear approach.

2.12 Summary of Research Gaps

Despite decades of progress across econometric, classical machine-learning (ML), and quantum ML (QML) approaches, the literature reveals several gaps that are directly relevant to exchange-rate forecasting in Ghana. First, while ARIMA/ARIMAX and VAR remain the Bank of Ghana's (Bank of Ghana) institutional workhorses, their near-linearity, stationarity, and parameter-stability assumptions limit performance under structural breaks, regime shifts, and high-dimensional macro

interactions, conditions that have characterized recent USD/GHS dynamics (Hyndman & Athanasopoulos, 2018; Sargent, 2018; Bank of Ghana, 2022; IMF, 2024). Second, although classical ML models such as LSTM, SVR, and XGBoost frequently outperform linear baselines in FX studies, prior applications for Ghana rarely benchmark these learners head-to-head against Bank of Ghana's operational tools on the same macro panel, with identical splits, leakage-safe preprocessing, and common metrics, nor do they report computational costs (train/infer time, memory) needed for policy workflows (Gyamerah, 2019). Critically, no study has established whether Support Vector Regression (SVR) materially outperforms Bank of Ghana baselines (ARIMA/ARIMAX/VAR) on Ghana's data while remaining operationally efficient and stable over a 12-month forecast horizon, the central decision question for method adoption that this thesis addresses (Rundo et al., 2019). Third, QML work, quantum kernels, feature maps, and variational layers, shows theoretical promise but is largely simulation-based, with little evidence on real policy datasets from African economies and no systematic comparisons against strong classical baselines under a unified pipeline (Benedetti et al., 2019; Havlíček et al., 2019; Preskill, 2018; Schuld & Petruccione, 2021). Finally, the literature seldom translates models into stakeholder-ready tools that expose comparable accuracy, uncertainty, and compute metrics, hindering institutional uptake at Bank of Ghana and allied agencies. This thesis addresses these gaps by (i) conducting a disciplined, like-for-like evaluation of SVR against ARIMA/ARIMAX/VAR (and concise ML/QML comparators) on a single Bank of Ghana macro panel with common preprocessing and metrics, (ii) testing operational 12-month forecasts with uncertainty bands, and (iii) delivering a prototype application that surfaces model-to-model comparisons suitable for policy use.

The lack of forecasting tools that are ready for policy in Ghanaian machine learning research is another issue. Previous studies usually stop at model-level assessment without converting findings

into deployable prototypes that can assist Bank of Ghana institutional procedures. In order to address this, this study creates a policy-ready prototype that combines uncertainty communication, computational efficiency, and prediction accuracy an innovation that sets it apart from other ML research with a Ghanaian focus (Adewuyi et al., 2022; Ngugi & Mutua, 2023).

CHAPTER THREE

METHODOLOGY

3.1 Introduction

This chapter presents the methodology employed to achieve the objectives of this study, which seeks the predictive accuracy, computational efficiency, and 12-month-ahead (2024) operational forecasts of traditional econometric models, classical machine learning (ML) algorithms, and quantum machine learning (QML) techniques for forecasting Ghana's exchange rate. The methodology is designed to provide a rigorous comparative analysis using actual macroeconomic datasets from the Bank of Ghana, covering key indicators such as exchange rates, debt data, monetary aggregates, financial soundness indicators, commodity prices, trade flows, and interest rates.

The chapter outlines the research design, data collection process, data preprocessing steps, model development and implementation, evaluation metrics, benchmarking framework, and the development of a prototype forecasting tool. The approach integrates both quantitative model evaluation, focused on forecasting accuracy and computational cost, and practical considerations through the design of a user-friendly application that can support institutional forecasting practices. The methodology ensures transparency, reproducibility, and alignment with both academic standards and practical policy relevance. Each section of this chapter details the specific methods and tools applied at various stages of the study, providing the foundation for the analysis presented in subsequent chapters.

3.2 Research Design

This study is structured around a clear decision question: does Support Vector Regression (SVR) materially outperform the Bank of Ghana's institutional baselines, ARIMA, ARIMAX, and VAR,

for USD/GHS forecasting, in terms of predictive accuracy and operational stability, at acceptable computational cost? To answer this, we adopt a comparative experimental design that evaluates all models on an identical, leakage-safe pipeline using the same monthly macroeconomic panel (2014–2023), standardized preprocessing, and common evaluation criteria. Accordingly, we treat ARIMA/ARIMAX/VAR as institutional baselines, LSTM/XGBoost as classical comparators, and QLSTM/QSVR as exploratory references; the focal model is SVR with an RBF kernel. The design follows best practice for time-series forecasting and risk-aware policy evaluation (Hyndman & Athanasopoulos, 2018; Cao et al., 2020).

Methodologically, the experiment uses a two-stage temporal protocol. First, an ex-post backtest trains models on 2014–2022 and scores them on the 2023 hold-out to obtain out-of-sample accuracy and empirical interval coverage. Second, an ex-ante operational run refits each model on data through Dec-2023 and generates Jan–Dec 2024 forecasts to assess stability and scenario sensitivity. All models share identical preprocessing (chronological split, Min-Max scaling of features, imputation where necessary) and a consistent feature set. For SVR, we use the RBF kernel and tune (C, ε, γ) on a time-ordered grid (no shuffling) to balance bias–variance under nonlinear macro interactions; LSTM and XGBoost are retained as concise classical comparators with minimal but standard tuning, while QLSTM and QSVR are implemented in simulation strictly to contextualize SVR's performance (Benedetti et al., 2019; Schuld & Petruccione, 2021).

Evaluation is uniform across model families: accuracy metrics (MAE, RMSE, MSE, MAPE, R2R^2R2) and computational metrics (training time, inference latency, peak RSS memory) are reported on the same windows to reflect policy-relevant trade-offs between accuracy and cost. Uncertainty quantification is handled via native prediction intervals for ARIMA/ARIMAX and bootstrap residual bands for SVR and other ML/QML models, with empirical coverage assessed

on the 2023 backtest (Hyndman & Athanasopoulos, 2018). The design is complemented by a lightweight Streamlit prototype that operationalizes the pipeline (model picker, scenario selector, metrics panel, 95% intervals, and downloadable tables/figures), ensuring that the findings are not only statistically rigorous but also actionable within Bank of Ghana-style workflows. Overall, the research design centers the SVR adoption decision while preserving enough comparative breadth to demonstrate where classical baselines and exploratory quantum references sit on the same Ghanaian data and evaluation protocol. The study design serves as the guide for showcasing the forecasting artifact's performance and use within the Design Science paradigm. As a result, the design details (i) how data are gathered to assess the artifact, (ii) how models are created and adjusted, (iii) how methodological choices avoid bias, leakage, or overfitting, and (iv) how the artifact is assessed in comparison to institutional baselines. This alignment guarantees that the technique offers a useful foundation for interpreting and evaluating the forecasting module in actual policy contexts, in addition to supporting empirical accuracy (Hevner & Chatterjee, 2010; Saunders et al., 2019).

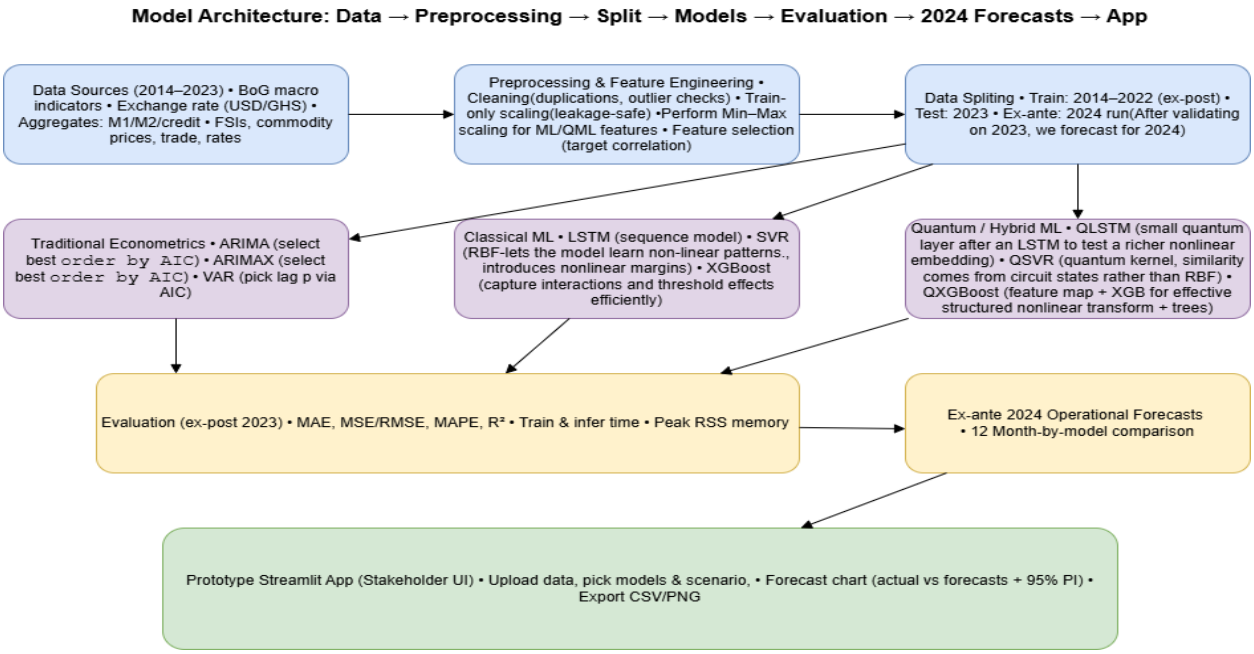


Figure 3.1: Leakage-safe forecasting pipeline.

Figure 3.1 illustrates the end-to-end forecasting pipeline that operationalizes the Design Science paradigm. The diagram shows: (i) leakage-safe temporal splitting (2014–2022 training, 2023 testing); (ii) preprocessing within the training loop only; (iii) model development across econometric, ML, and QML families; (iv) walk-forward validation for tuning; and (v) a final ex-ante forecasting stage for 2024. This pipeline ensures methodological rigor by preventing information leakage, enforcing temporal causality, and enabling a fair comparison across all model types (Hyndman & Athanasopoulos, 2018). It also demonstrates the evaluation procedure needed to assess the artifact's usefulness under the Design Science paradigm.

3.2.1 Research Philosophy

The positivist–empiricist research philosophy used in this study is predicated on the idea that financial and economic phenomena, including changes in exchange rates, are objective and can be quantified, modeled, and forecasted using observable data (Creswell, 2014; Saunders et al., 2019). According to this perspective, knowledge is produced by statistical analysis, model-based reasoning, and empirical observation as opposed to subjective interpretation. The methodological demands of forecasting research, where models are assessed according to out-of-sample generalization, computing efficiency, and objective accuracy measures, are in line with positivism. Axiologically speaking, the study is assumed to have an objective value orientation, which means that neutrality, minimum researcher bias, and value-free interpretation of findings are the goals of the research. The models under evaluation—ARIMA, VAR, SVR, LSTM, and QML variants—do not rely on human beliefs to function. Rather, they are evaluated using quantifiable performance

metrics. The choice of an organized, scientific, and repeatable technique appropriate for assessing the utility of forecasting models in policy contexts is supported by this axiology.

The study is consistent with a Design Science Research (DSR) paradigm since it combines positivism, empiricism, and objective axiology. DSR concentrates on producing artifacts that address practical issues. A forecasting module (SVR and its comparison framework) integrated within a prototype decision-support tool serves as the study's artifact. According to DSR, the methodology must explain how the artifact is created, how information is gathered to show its value, and how it is assessed and interpreted (Hevner & Chatterjee, 2010). The study's empirical methods, comparative design, and assessment techniques are so influenced by this paradigm.

3.3 Data Collection

The data used in this study were sourced primarily from the Bank of Ghana, which serves as the official repository of national macroeconomic and financial statistics. The dataset spans a period of 10 years (2014–2023), providing comprehensive monthly time series records that reflect Ghana's economic dynamics across multiple domains. This long-term dataset enables the modeling of exchange rate behavior under varying economic conditions, including periods of macroeconomic stability, external shocks, and structural adjustments (Bank of Ghana, 2022; IMF, 2024).

The study focuses on the interbank exchange rate (USD/GHS) as the target variable for forecasting, given its centrality to Ghana's external sector policy and its role as a key indicator of macroeconomic health. Predictor variables include a rich set of macroeconomic indicators that capture fiscal, monetary, financial, external, and commodity-related dimensions of the economy. Specifically, these include; Debt data, Monetary aggregates, Financial soundness indicators (FSIs), Commodity prices, Merchandise trade flows, Interest rates.

The final merged dataset contains approximately 120 rows the final modelling table has 16 variables in total, 1 target and 15 predictors ensuring a high-dimensional and policy-relevant basis for forecasting model development. Data integrity checks, including assessments of completeness, consistency, and temporal alignment, were performed during preprocessing to ensure reliability for modeling (Hyndman & Athanasopoulos, 2018).

These data provide the foundation for benchmarking traditional econometric models, classical machine learning algorithms, and quantum machine learning techniques in a unified experimental framework. The selection of variables reflects both their theoretical relevance, as established in prior exchange rate literature (Sargent, 2018; World Bank, 2023), and their operational use in Bank of Ghana's macroeconomic monitoring and policy analysis (Bank of Ghana, 2022). The sample size is sufficient and in line with macroeconomic forecasting standards, where low-frequency (monthly or quarterly) data naturally result in tiny datasets, despite the datasets been about 120 monthly observations (Hyndman & Athanasopoulos, 2018). In exchange-rate, inflation, and macro-risk modeling, small-sample forecasting is commonly used, particularly in developing nations where data availability limitations are prevalent (IMF, 2024). This permits the inclusion of QML models just as exploratory references and justifies the application of techniques like SVR and ARIMA/VAR, which are theoretically appropriate for small-to-medium datasets.

3.4 Data Descriptive Statistics of the Dataset

This section provides a summary of the macroeconomic dataset used for forecasting Ghana's interbank exchange rate. The dataset spans from January 2014 to December 2023 and contains monthly observations of the exchange rate (USD/GHS) alongside a refined set of macroeconomic predictors. These features were selected based on domain knowledge and statistical correlation with the target variable. The predictors include total credit, currency outside banks, broad money

(M2), indicative petroleum price, Ghana reference rate, external debt, international gold price, and other high-impact fiscal and monetary indicators. Table 4.1 summarizes key statistical properties of the variables, including their mean, median, minimum, maximum, and standard deviation values.

Table 3.1 Descriptive Statistical Table

Variables	count	mean	median	std	min	max
Ex. Rate (USD)	120	5.537416667	4.85	2.40885987	2.29	13.07
Total credit	120	42931060886	37903127795	15851161444	17623837880	81851477020
Broad money (M2)	120	69835893417	60880530000	40323805570	21011550000	1.85426E+11
Currency outside banks	120	14315633500	10991785000	8351096175	5012080000	37621030000
External debt	120	20694.885	18274.25	6015.297639	11419.2	30142.8
Total public debt	120	39399.58	37017.15	12628.97976	21135.6	60496.6
Petrol Price-	120	5.33975	4.67	3.875189009	0	17.43
Brent Crude Oil Price	120	68.35066667	65.25	21.03344902	26.63	117.22

Cocoa Price	120	2639.149417	2542.215	440.2598843	1904.6	4235.6
Gold Price	120	1491.136417	1332.055	294.0899613	1069.4	2035.43
Merchandise Imports (f.o.b)	120	1041.1345	1090.405	301.3197559	0	1530.28
Exports Gold	120	421.96025	447.215	155.4509289	0	724.68
Exports Cocoa Beans	120	132.881	116.82	102.3357472	0	332.38
Monetary Policy Rate (%)	120	20.19583333	19	5.064000409	13.5	30
Ghana Reference Rate (%)	120	10.93508333	14.095	10.61439232	0	33.25
Treasury Bill Interest rate	120	19.08766667	16.025	6.289189302	12.08	35.67

The analysis reveals that the exchange rate exhibits a steady upward trajectory over the two-decade period, reflecting the cumulative effects of fiscal deficits, monetary expansion, external debt accumulation, and exogenous shocks. Variables such as total credit and broad money have grown substantially over time, suggesting the influence of monetary expansion on exchange rate trends. External debt shows considerable variability, capturing major sovereign borrowing events,

including Eurobond issuances and IMF-supported programs. The indicative petroleum price also shows high volatility, mirroring fluctuations in global oil markets and affecting domestic price levels. Export indicators such as gold and cocoa prices exhibit moderate variation, shaped by international commodity cycles.

To better understand the relationship between these macroeconomic indicators and the exchange rate, a correlation heatmap was generated. The results, illustrated in Figure 4.1, show strong positive correlations between the exchange rate and variables such as total credit (0.97), currency outside banks (0.96), and broad money (0.95).

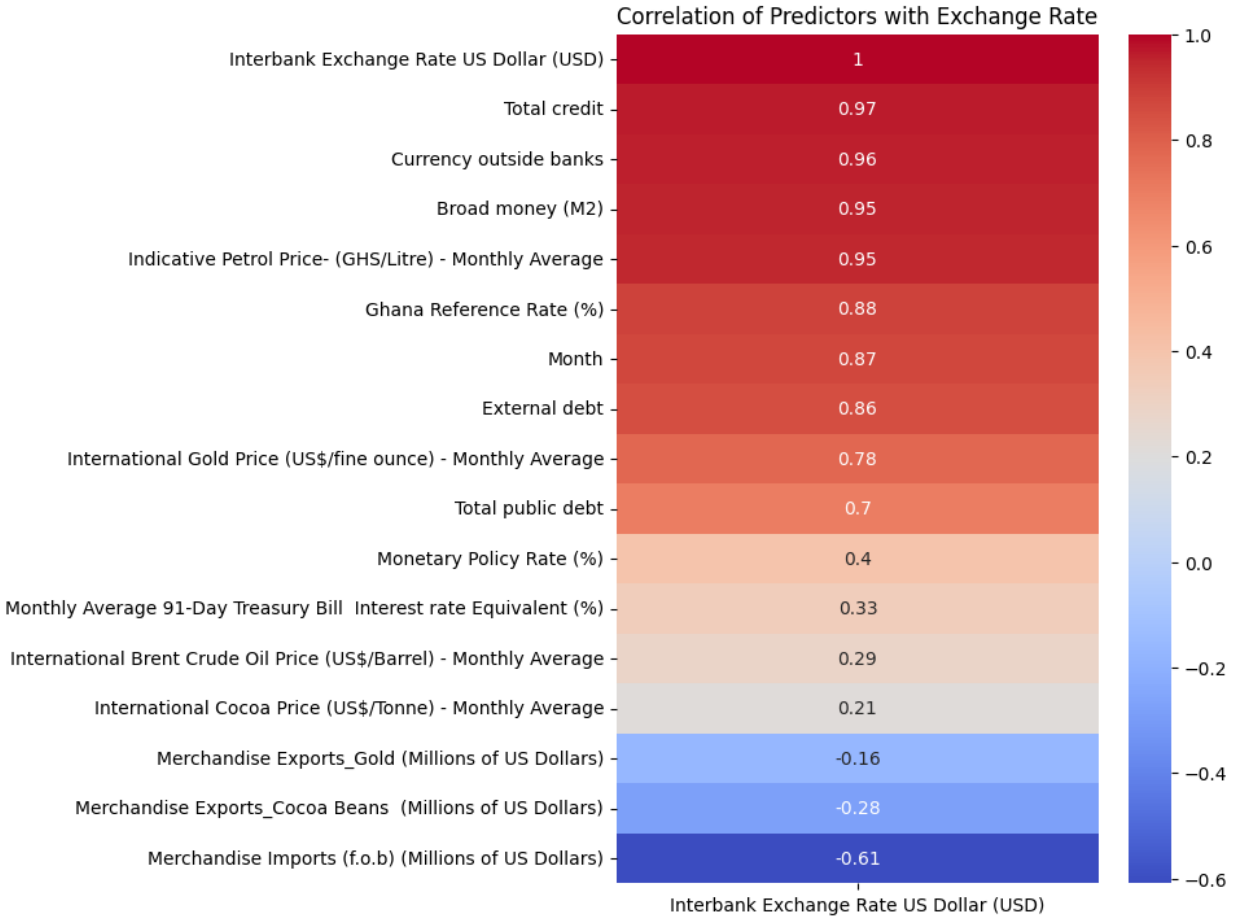


Figure 3.2 Correlation between Exchange Rate and Predictors

These relationships suggest that higher levels of liquidity and credit in the economy are associated with cedi depreciation. Conversely, some variables exhibit negative correlations, including merchandise exports (–0.61), cocoa exports (–0.28), and gold exports (–0.16), indicating that improved export earnings may help stabilize or strengthen the local currency.

Overall, the descriptive statistics and correlation structure highlight the dynamic and interconnected nature of Ghana's macroeconomic environment. The selected features provide a balance of high, moderate, and inverse correlations, offering a suitable testbed for evaluating how traditional econometric models, classical machine learning algorithms, and quantum models can handle multicollinearity, nonlinear dependencies, and structural macroeconomic trends in forecasting exchange rate behavior.

3.5 Data Preprocessing

Robust data preprocessing was critical to ensure that the final dataset was both analytically sound and suitable for developing forecasting models across econometric, classical machine learning (ML), and quantum machine learning (QML) paradigms. The preprocessing workflow addressed data quality, temporal consistency, dimensionality reduction, and feature enhancement, following best practices for time series and macroeconomic modeling (Hyndman & Athanasopoulos, 2018; Bousselmi & Zaher, 2020).

The initial stage involved rigorous data cleaning, where missing values were identified and addressed using interpolation methods. For time series variables with intermittent gaps, linear interpolation and forward-fill techniques were applied depending on the frequency and economic plausibility of the missing sequence. Outliers and anomalies were identified through visual inspection of line plots and boxplots, as well as by computing statistical thresholds (IQR), and

were handled contextually based on macroeconomic trends and known events (commodity price shocks, fiscal restructuring).

A major component of preprocessing was feature selection. From an initial pool of 56 macroeconomic indicators, a refined set of 17 variables was selected based on economic theory, correlation with the exchange rate, and statistical relevance. This curated set included predictors such as broad money (M2), total credit, external debt, monetary policy rate, petroleum prices, and export values for key commodities like cocoa and gold. Redundant, static, and highly collinear indicators were excluded to improve model generalization and prevent overfitting, especially given the relatively limited sample size (~120 observations). This reduction also enhanced computational efficiency, particularly for quantum algorithm simulation.

Following selection, scaling and normalization were performed to align the variable ranges and support algorithmic convergence. Min-Max normalization was applied to the predictors used in classical ML and QML models, rescaling all values into the [0, 1] range. For econometric models such as ARIMA and VAR, scaling was unnecessary or selectively applied depending on model assumptions.

To enrich the dataset's temporal structure, feature engineering was applied. This included creation of 1-month lag features and 3-month rolling means for selected indicators to capture memory effects and smoothed economic trends. These engineered features were especially useful for deep learning models like LSTM and hybrid quantum-classical architectures (QLSTM), which rely on sequential patterns to extract predictive signals.

To ensure rigorous evaluation and avoid data leakage, the dataset was partitioned chronologically into training and testing subsets. The training set consisted of data from January 2014 to December 2022, while the test set comprised January 2023 to December 2023. This split preserved the

temporal ordering of events and reflected real-world forecasting conditions, where future data is unseen during model training (Abayomi & Asumadu, 2021).

This structured and theoretically grounded preprocessing pipeline produced a high-quality dataset that balances economic insight, computational feasibility, and forecasting fidelity, supporting the thesis's core aim of benchmarking traditional econometric models against advanced classical and quantum machine learning techniques.

3.6 Model Development

Instead of creating completely new algorithms, this work incorporates and modifies well-known forecasting methods. The development of an SVR-based forecasting module customized for Ghana's exchange-rate environment, the methodical application of econometric, classical ML, and quantum ML models within a single leakage-safe pipeline, and the incorporation of these models into a prototype decision-support application constitute the methodological contribution. To guarantee transparency and reproducibility, all models are constructed using common, repeatable libraries like statsmodels, scikit-learn, TensorFlow/PyTorch, and quantum ML frameworks (Cao et al., 2020; Hyndman & Athanasopoulos, 2018).

Therefore, rather than developing new algorithms, the novelty of the model development lies in (i) designing a disciplined, like-for-like implementation of econometric, ML, and QML models on a single Bank of Ghana macro panel; (ii) developing and fine-tuning an SVR-based forecasting module specifically tailored to Ghana's exchange-rate dynamics; and (iii) integrating these models into a prototype decision-support application that exposes accuracy, uncertainty, and computational-efficiency metrics in a format that policy stakeholders can use.

3.6.1 Traditional Econometric Models

The traditional econometric models developed in this study serve as baseline forecasting tools against which classical machine learning and quantum machine learning models are compared. The models include the Autoregressive Integrated Moving Average (ARIMA) model, which forecasts the exchange rate based on its past values and residual errors; the ARIMA with exogenous variables (ARIMAX), which incorporates external macroeconomic drivers such as debt levels, money supply, and commodity prices; and the Vector Autoregression (VAR) model, which jointly models the dynamic relationships among the exchange rate and key macroeconomic indicators (Tsay, 2013; Gujarati, 2009). These models are implemented using Python libraries such as statsmodels and pmdarima, with model specifications determined through statistical criteria including the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC), and validated using stationarity tests and residual diagnostics (Hyndman & Athanasopoulos, 2018; Sargent, 2018).

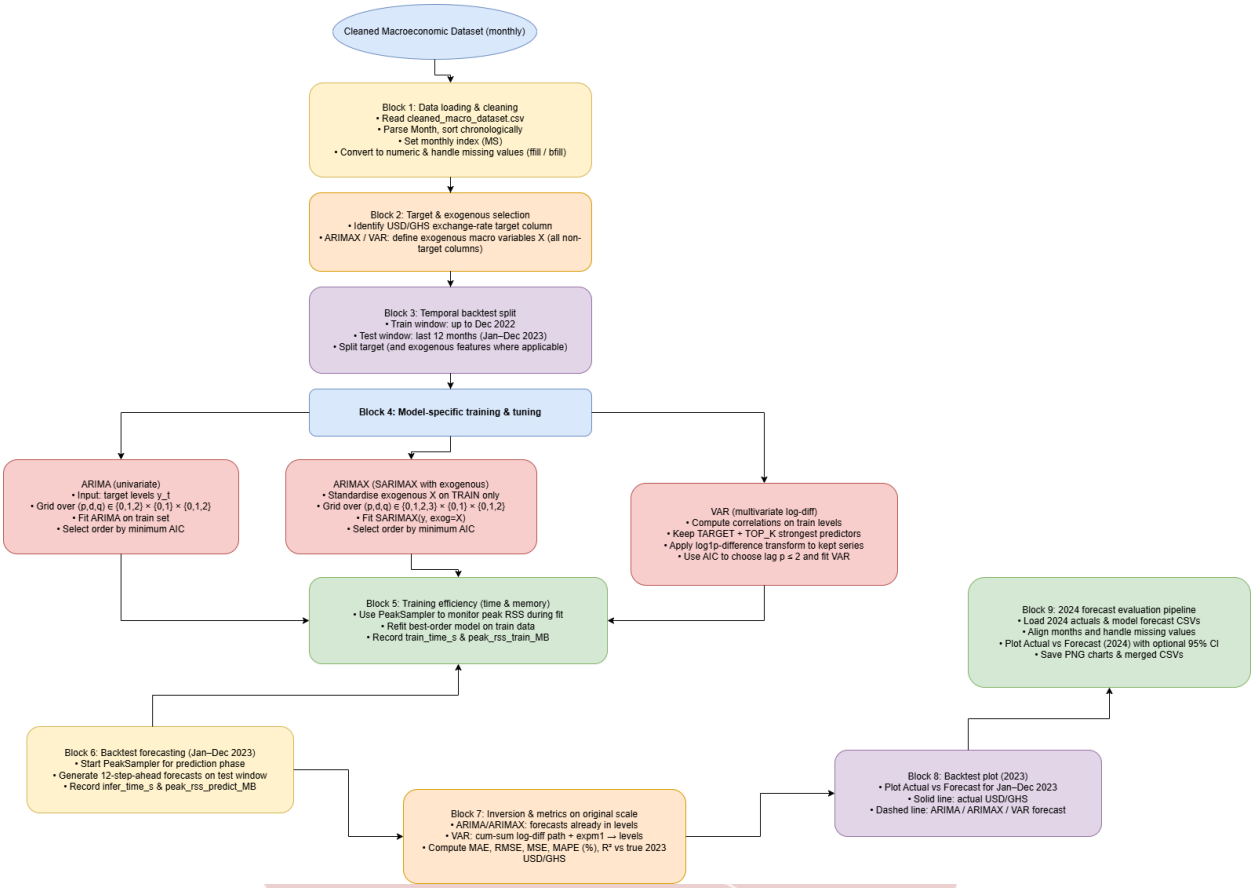


Figure 3.3 Traditional models training and tuning evaluation flowchart.

The entire process for training and evaluating the conventional time-series models (ARIMA, ARIMAX, and VAR) is depicted in Figure 3.3. The series is divided into a training window for 2014–2022 and a back-test window for 2023 when cleaned monthly macroeconomic data are loaded, the USD/GHS exchange-rate objective and exogenous variables are established. After that, each model is fitted and adjusted by minimizing AIC while monitoring memory usage and training time. The process is prolonged to 2024 by comparing model forecasts with observed rates in Actual-versus-Forecast charts and summary tables. The tweaked models provide forecasts for 2023, which are assessed using error metrics on the original USD/GHS scale.

3.6.2 Classical Machine Learning Models

SVR is adopted as the primary classical method because its ε-insensitive margin loss and kernel mapping (here, RBF) are well-suited to Ghana's exchange-rate setting, where nonlinear interactions among monetary aggregates, debt, and external prices can dominate and sample sizes are small-to-medium. The RBF kernel provides a flexible, smooth decision function that can approximate complex response surfaces without exploding parameter count, while regularization (C) controls model capacity and mitigates overfitting, attributes that align with policy datasets that are relatively short and periodically subject to regime shifts (Rundo et al., 2019; Gyamerah, 2019; Schuld & Petruccione, 2021). Empirically, SVR has shown strong out-of-sample performance on financial/macroeconomic series with multivariate drivers, making it a credible candidate to augment Bank of Ghana's baselines.

SVR implementation details. We use an RBF kernel by default, with a compact hyperparameter grid emphasizing bias–variance control in limited data: $C \in \{1,10,100\}$, $\varepsilon \in \{0.001,0.01,0.1\}$, $\gamma \in \{\text{“scale”},0.01,0.1\}$. Tuning is performed via time-ordered cross-validation on the 2014–2022 training window (no shuffling) to respect temporal dependence, selecting the configuration that minimizes validation MSE while maintaining stable residual behavior. All predictors are Min-Max scaled; the target is left on its original scale for reporting. Unless otherwise noted, the selected SVR variant refers to this RBF-based specification and tuning protocol (Schuld & Petruccione, 2021).

LSTM is retained as a deep-learning comparator given its ability to capture temporal dependencies in sequences; we use a compact many-to-one architecture with basic regularization and lightweight randomized search to avoid over-parameterization relative to data length (Abayomi & Asumadu, 2021). XGBoost serves as a tree-ensemble comparator valued for handling structured macro data

with built-in regularization and implicit feature selection; we apply a small grid over depth/learning-rate/trees and subsampling to keep complexity aligned with the sample (Chen & Guestrin, 2016). Both are included to contextualize SVR's performance under the same preprocessing, temporal split, and metrics.

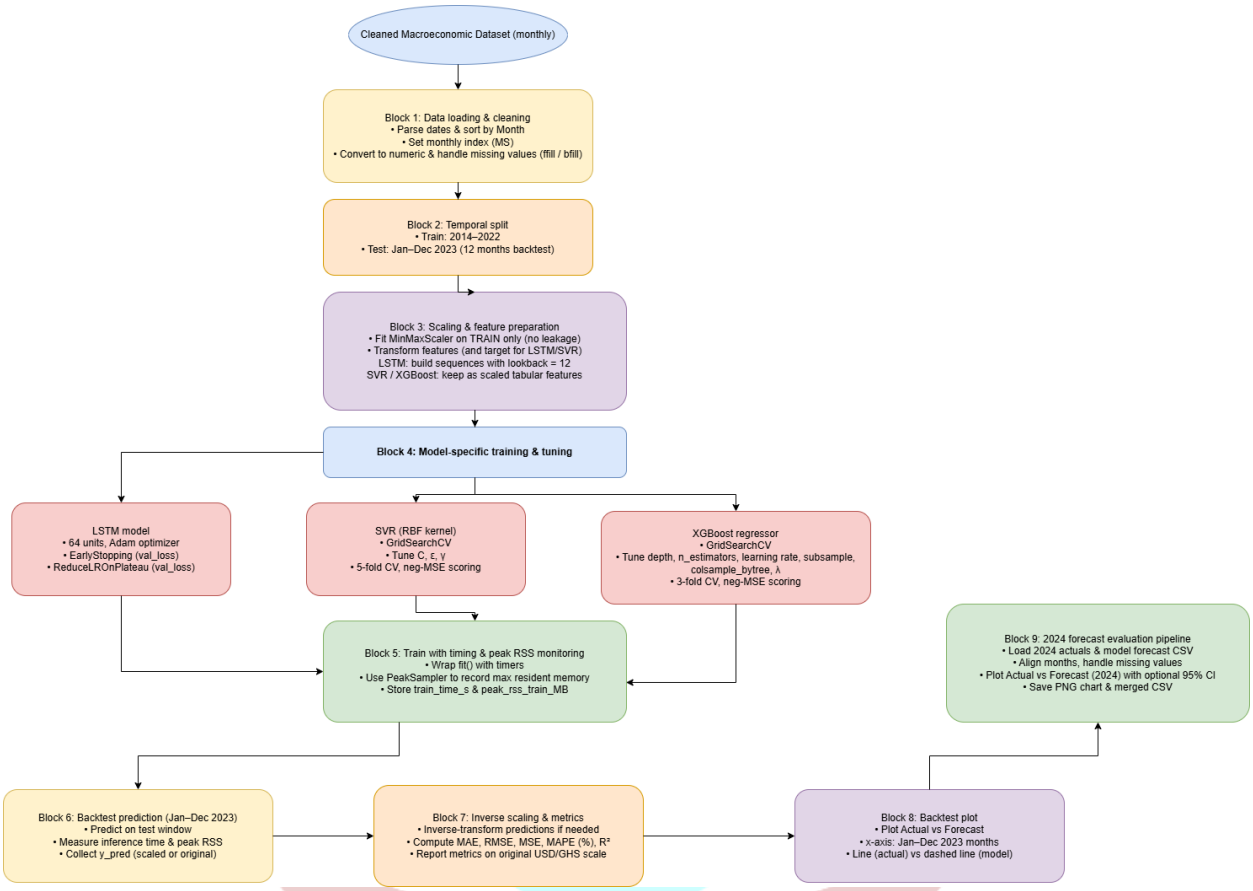


Figure 3.4 Classical models training and tuning evaluation flowchart.

Figure 3.4 summarizes the full pipeline for the classical forecasting models (LSTM, SVR and XGBoost), from cleaned monthly macroeconomic data through to final evaluation. The data are chronologically ordered, split into a 2014–2022 training set and a 2023 back-test set, scaled (with LSTM sequences built using a 12-month look-back), and each model is trained and tuned (LSTM with early stopping, SVR/XGBoost via GridSearchCV) while recording time and memory use. Predictions are then inverse-scaled, assessed using MAE, RMSE, MAPE and R2R^{2}R2, and

extended to a 2024 evaluation where forecasts are compared with actual exchange rates and exported for further analysis.

3.6.3 Quantum Machine Learning Models

Quantum models are included purely as exploratory references to situate SVR's results on the same pipeline, recognizing that current implementations run in simulation and that quantum hardware is not yet available at the scale required for policy operations (Preskill, 2018; Benedetti et al., 2019; Schuld & Petruccione, 2021). We consider QLSTM, which augments a compact LSTM with a variational quantum layer (Strongly-Entangling or angle-embedding blocks) to increase expressivity, and QSVR, which replaces the classical kernel with a quantum feature map / kernel estimated in simulation (Havlíček et al., 2019). Both are executed with identical data splits, scaling, and reporting as their classical counterparts so that any differences in accuracy or compute are attributable to modeling choices rather than pipeline artifacts. Given their early-stage status and simulation overheads, these QML models are not positioned as deployment candidates here; rather, they contextualize the SVR adoption decision by indicating where quantum-inspired kernels or variational layers may (or may not) add value on Ghana's macro panel.

Since quantum technology is not currently accessible at the scale needed for practical macroeconomic forecasting, the QML models utilized in this study are strictly exploratory (Preskill, 2018). Therefore, it is not appropriate to extrapolate results from QLSTM and QSVR simulations to actual policy settings. Rather than suggesting quantum models for institutional deployment, their presence helps to contextualize SVR performance and indicate new research opportunities.

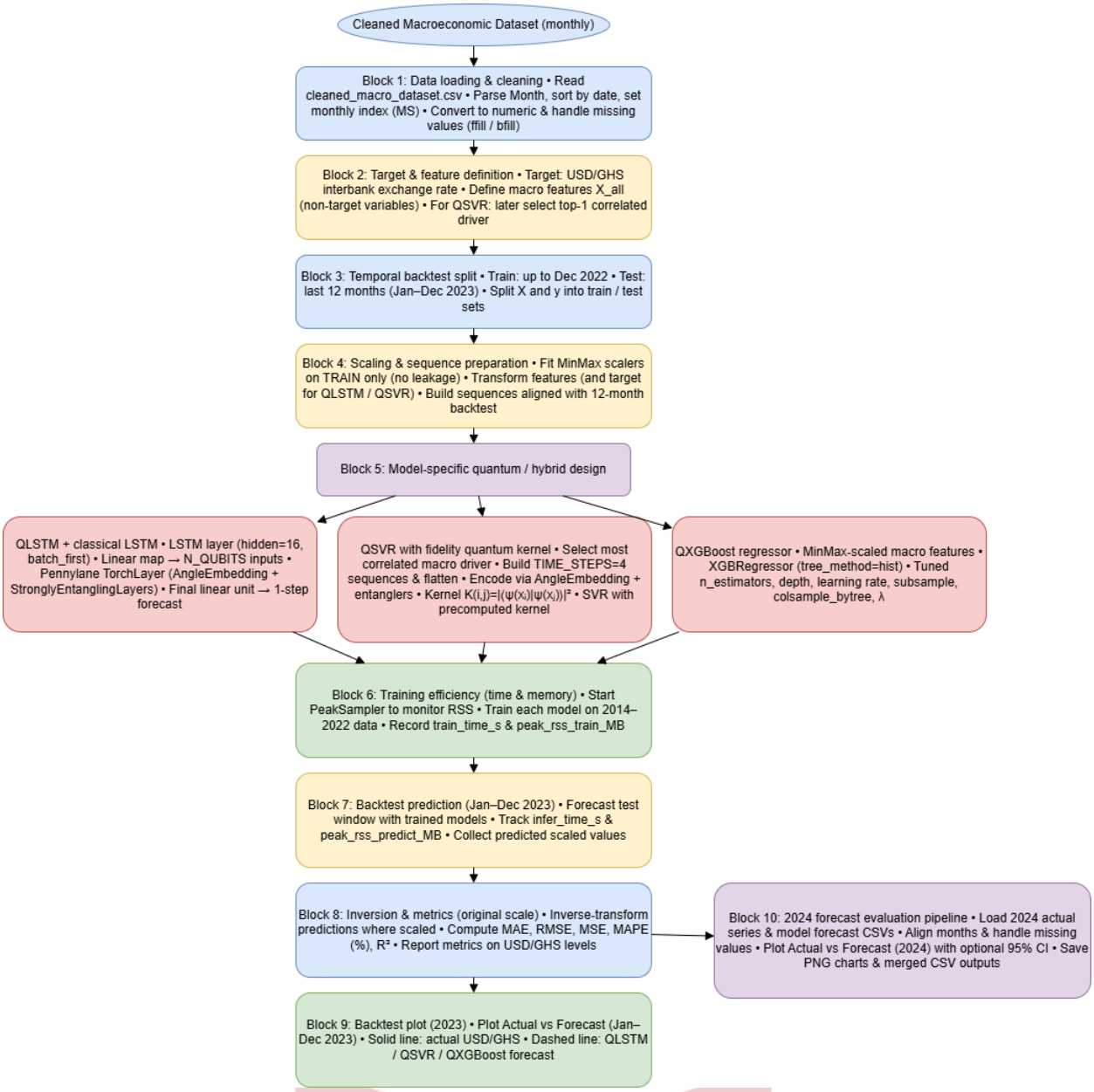


Figure 3.5 Quantum models training and tuning evaluation flowchart.

The whole pipeline for the quantum models is shown in Figure 3.5. Before building sequences, the cleaned monthly macroeconomic data are loaded, features are scaled, a temporal split (2014–2022 train, 2023 test) is applied, and the USD/GHS rate is defined as the target. The QLSTM, QSVR, and QXGBoost models are then trained using timing and memory monitoring, assessed on the

2023 backtest using MAE, RMSE, MAPE, and (R^2), and then utilized to provide CSV outputs and Actual-vs-Forecast charts for 2023 and 2024.

3.6.4 Implementation Tools

All models are implemented using widely accepted industry-standard tools to ensure reproducibility and transparency. Traditional econometric models are built using statsmodels and pmdarima. Classical machine learning models are developed using TensorFlow, Keras, Scikit-learn, XGBoost. Quantum models are implemented in Qiskit, PennyLane, and TensorFlow Quantum. In addition, a user-friendly forecasting application is developed using Streamlit to provide policymakers and analysts with a practical interface for exploring model outputs and making data-driven decisions (Cao et al., 2020).

3.7 Model Training and Tuning

This study centers the SVR training/tuning procedure and holds all other models to the same leakage-safe, time-ordered protocol so that differences in results are attributable to modeling choices rather than pipeline artifacts. We adopt a two-stage temporal design: (i) ex-post backtest (train on Jan-2014–Dec-2022, test on Jan–Dec-2023) to measure generalization on unseen data, then (ii) ex-ante operational run (refit on Jan-2014–Dec-2023 and produce Jan–Dec-2024 forecasts). All preprocessing and hyperparameter search are performed inside a time-series cross-validation loop (walk-forward / rolling origin), and all scalers are fit on the training fold only to prevent look-ahead leakage (Hyndman & Athanasopoulos, 2018).

SVR (focal model). Inputs are Min-Max scaled and passed to an RBF-kernel SVR within a single Pipeline (scaler → SVR), ensuring that scaling parameters are cross-validated jointly with model hyperparameters. We tune a compact, stability-oriented grid: $C \in \{1,10,100\}$, $\varepsilon \in \{0.001,0.01,0.1\}$, $\gamma \in \{\text{"scale"},0.01,0.1\}$. The scoring objective during walk-forward CV is negative MSE, with

MAE/RMSE and residual diagnostics used as tie-breakers to prefer well-calibrated, non-erratic fits. The ε-insensitive loss and margin-based regularization (via CCC) help control capacity in our small/medium sample regime, while the RBF kernel flexibly captures the nonlinear interplay among money supply, debt, and external prices documented for Ghana (Rundo et al., 2019; Gyamerah, 2019; Schuld & Petruccione, 2021). After selection, the chosen configuration is refit on the full 2014–2023 history and used to generate the 12-month 2024 operational forecasts.

Econometric baselines (ARIMA/ARIMAX/VAR). For ARIMA/ARIMAX, orders are selected by AIC/BIC on the 2014–2022 training window with maximum likelihood estimation; stationarity/invertibility constraints are relaxed when needed to improve AIC but residuals must pass basic diagnostics (ACF/PACF whiteness). ARIMAX uses standardized exogenous drivers (money aggregates, debt, commodity prices) and scenario paths for the operational run; exogenous variables are standardized within the training window only. VAR lag length is chosen by information criteria with residual checks for serial correlation. Native 95% prediction intervals from the state-space likelihood are used for ARIMA/ARIMAX (and VAR where available), and all three are refit on 2014–2023 prior to the 2024 forecast (Hyndman & Athanasopoulos, 2018).

Secondary classical comparators. LSTM is retained as a deep baseline but kept deliberately compact relative to sample size: a many-to-one architecture with a single LSTM block (moderate hidden units), dropout, and Adam optimizer; a small randomized search over learning rate, hidden size, and dropout is embedded in the same walk-forward scheme with early stopping on the validation fold (Abayomi & Asumadu, 2021). XGBoost is tuned over shallow max_depth, learning_rate, n_estimators, and modest subsample/colsample to avoid over-fitting short sequences (Chen & Guestrin, 2016). Both models receive the same feature set and temporal splits as SVR.

Exploratory QML references. QLSTM and QSVR are executed in simulation (PennyLane/Qiskit-style stacks) with minimal circuit depth and feature encodings to control simulation cost. They follow the identical train/validation/test protocol and scaling as their classical analogues, but are included only to contextualize the SVR adoption decision rather than as deployment candidates (Benedetti et al., 2019; Preskill, 2018).

For reproducibility, random seeds are fixed, libraries and versions are recorded, and computational measurements use a consistent harness: wall-clock train time, inference time per forecast, and peak RSS memory during train and predict (sampled via OS-level RSS and tracemalloc). This uniform procedure yields a fair basis to compare forecasting accuracy, stability, and operational cost across models.

3.8 Evaluation Metrics

We evaluate all models on a unified, leakage-safe framework with metrics computed on: (i) the 2023 hold-out (generalization), and (ii) the 2024 operational horizon (practical suitability). Predictive accuracy is summarized by MAE, MSE, RMSE, MAPE, and R^2. Because the 2023 window exhibits low variance, we treat MAE/RMSE as primary ranking criteria and interpret R^2 cautiously (negative values simply indicate that a mean-only baseline fits the specific window better). To support operational decisions at the Bank of Ghana, we track computational efficiency with train time, inference time, and peak RSS memory for both training and prediction phases, reported with the same sampling cadence across models. Uncertainty quantification is provided via native 95% intervals for ARIMA/ARIMAX (and VAR where accessible) and bootstrap residual intervals for ML/QML models constructed on the 2023 backtest residuals and carried into the 2024 run (Hyndman & Athanasopoulos, 2018). Interval coverage and width are examined qualitatively to gauge reliability during volatile months.

These choices keep the emphasis on policy-relevant trade-offs, accuracy and compute, needed for SVR adoption. Quantum-hardware descriptors (qubit counts, circuit depth) are reported in an Appendix for completeness but are not decision criteria in the present, simulation-based setting (Schuld & Petruccione, 2021). The resulting scorecards (accuracy + compute + intervals) provide a transparent basis to determine whether SVR materially outperforms ARIMA/ARIMAX/VAR while remaining operationally efficient for USD/GHS forecasting.

3.8.1.1 Mean Absolute Error (MAE)

MAE measures the average absolute difference between the actual and predicted values. It is calculated as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Where:

- y_i is the actual value,
- $\hat{y}_i$ is the predicted value,
- n is the number of observations.

MAE shows the average size of prediction mistakes and is simple to understand. Better performance is indicated by lower values. It was selected due to its ease of use and efficiency in assessing forecasting models in economic settings (Makridakis et al., 2018).

3.8.1.2 Mean Square Error (MSE)

MSE measures the average of the squared differences between actual and predicted values:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Where:

- y_i is the actual value,
- $\hat{y}_i$ is the predicted value,
- n is the number of observations.

Larger errors are penalized more severely than smaller ones, making MSE a clear indicator of model performance. When outlier mistakes may be a sign of inadequate model generalization, it

is particularly helpful in drawing attention to disparities. MSE is frequently employed because of its mathematical characteristics that support optimization during model training, even though it is not as interpretable as MAE in real-world units (Makridakis et al., 2018).

3.8.1.3 Root Mean Square Error (RMSE)

RMSE measures the square root of the average squared differences between actual and predicted values, emphasizing larger errors:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Large predicting errors could have major economic repercussions, therefore RMSE is especially pertinent when evaluating models because it penalizes considerable deviations. It offers information about the model's dependability in making precise predictions about Ghana's economic statistics (Makridakis et al., 2018).

3.8.1.4 R-Squared (R^2)

R^2 evaluates the proportion of variance in the dependent variable explained by the independent variables in a model. It is expressed as:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Where:

- $\bar{y}$ is the mean of actual values.

An R^2 value closer to 1 indicates that the model explains a significant portion of the variance, reflecting high predictive power. R^2 is particularly relevant for comparative analysis to assess how well classical and quantum models capture patterns in Ghana's economic data (Gujarati, 2009).

3.8.1.5 Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE) is a widely used metric to measure the accuracy of a forecasting model. It expresses the average absolute difference between actual and predicted values as a percentage of the actual values, making it intuitive and easy to interpret.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\ y_i - \hat{y}_i\ |}{y_i} \times 100$$

y_i = Actual value

$\hat{y}_i$ = Predicted value

n = Number of data points

A Mean Absolute Percentage Error (MAPE) of 0% indicates a perfect prediction, while a MAPE of 5% signifies that the model's predictions deviate from the actual values by an average of 5%. In general, a lower MAPE value reflects higher forecasting accuracy. As a guideline, a MAPE of less than 10% is considered highly accurate, 10% to 20% is regarded as good, 20% to 50% is viewed as a reasonable forecast, and a MAPE greater than 50% indicates poor predictive performance. MAPE is particularly useful because it expresses model accuracy as an average percentage error, making results intuitive and easily interpretable for policymakers and non-technical stakeholders (Hyndman & Athanasopoulos, 2021).

3.8.2 Computational Time

In terms of the amount of time needed for training time and inference time for prediction, computational time assesses how effective a model is. Particularly with high-dimensional datasets, quantum models frequently promise speedups in specific tasks (Havlíček et al., 2019). The practical viability of classical and quantum algorithms is demonstrated by comparing their computing times, especially in settings with limited resources like Ghana.

3.8.3 Prediction Intervals and Coverage.

Beyond point forecasts, we report 95% prediction intervals and their empirical coverage on the 2023 backtest to assess calibration and operational risk. For ARIMA/ARIMAX (and VAR where available), intervals are obtained from the model's analytic/state-space likelihood. For SVR and the other ML/QML models, we construct bands via a time-series residual bootstrap: after fitting on the training span, we compute 2023 residuals, resample them with block bootstrap (to preserve short-run dependence), and add the resampled residual paths to each model's fitted trajectory to generate forecast distributions; the 2.5th/97.5th percentiles yield 95% intervals (Hyndman & Athanasopoulos, 2018). We then report empirical coverage (the share of 2023 observations falling inside the nominal 95% band) and note under/over-coverage where present. This procedure is emphasized for SVR, the focal model, to substantiate its operational readiness with defensible uncertainty quantification on the same pipeline as the econometric baselines.

3.9 Comparison Approach

The primary comparison in this study is SVR versus ARIMA/ARIMAX/VAR on two dimensions: (i) ex-post accuracy on the 2023 hold-out and (ii) ex-ante stability of Jan–Dec 2024 operational forecasts (level, turning-points, and interval width). LSTM/XGBoost (classical comparators) and QLSTM/QSVR (exploratory references) are reported to bound performance around the focal decision while keeping the evaluation centered on whether SVR offers material gains over Bank of Ghana's institutional baselines. All models use the same Bank of Ghana monthly macro panel (2014–2023), identical, leakage-safe temporal splits (train 2014–2022 → test 2023; refit 2014–2023 → forecast 2024), and uniform preprocessing/feature engineering, so differences reflect model capacity rather than data handling (Cao et al., 2020; Hyndman & Athanasopoulos, 2018). Performance is summarized with MAE, MSE, RMSE, MAPE, and R^2 on 2023; operational cost is

assessed via train time, inference time, and peak RSS memory, recorded with the same measurement harness. For 2024, forecasts are aligned to realized 2024 observations (where available) to provide an indicative operational check, and interval coverage/width are discussed qualitatively. Results are presented with aligned tables, charts, and scorecards to make the SVR-vs-baseline decision transparent while situating classical and quantum models as contextual bounds.

3.10 Prototype Forecasting Tool

The comparative analysis uses narrative synthesis in addition to quantitative scoring to explain why models behave differently. This covers overfitting behavior, regime-shift adaptability, computational overhead, sensitivity to tiny samples, and structural assumptions (linearity vs. nonlinearity). For policy organizations like the Bank of Ghana, narrative explanations of model behavior improve interpretability and encourage evidence-based decision-making (Makridakis et al., 2018). To translate findings into practice, we developed a Streamlit prototype that loads by default with SVR pre-selected and displays its point forecast, 95% prediction band (bootstrap-based), and metrics panel on launch. Users can toggle baselines (ARIMA/ARIMAX/VAR), add classical comparators (LSTM/XGBoost), or include exploratory QML (QLSTM/QSVR) to view side-by-side charts and tables. The app accepts a Bank of Ghana-formatted macro dataset upload, applies the same leakage-safe pipeline as in the experiments, and produces a 12-month horizon forecast under the selected scenario. The metrics panel reports MAE, RMSE, MAPE, R^2, train time, inference time, and peak RSS (MB) for the 2023 window; ARIMA/ARIMAX show native 95% interval. A comparison table lists Month and (Actual, SVR, ARIMA, ARIMAX, VAR, LSTM, XGBoost, QLSTM, QSVR, QXGBoost), with download buttons for CSV (table) and PNG (figure). This design mirrors the thesis emphasis, SVR as the default, Bank of Ghana baselines as

institutional comparators, and demonstrates how stakeholders at the Bank of Ghana could interactively examine accuracy, uncertainty, and compute trade-offs within a lightweight, reproducible tool (Hyndman & Athanasopoulos, 2018; Preskill, 2018).

3.11 Ethical Considerations

This study adheres to ethical standards in the use of data, model development, and reporting to ensure integrity, transparency, and responsibility throughout the research process. The primary dataset used comprises historical macroeconomic indicators and exchange rate data obtained from publicly available sources maintained by the Bank of Ghana and other official statistical agencies. These data are aggregated at the national level, contain no personal or sensitive information, and are used solely for academic purposes, thereby posing no risk to individual privacy or confidentiality (Bank of Ghana, 2022; Ghana Statistical Service, 2021). Care is taken to ensure that models are trained, tested, and reported transparently. All preprocessing, model tuning, and evaluation procedures are fully documented to support reproducibility and to avoid misleading or selective reporting of results. Forecasting models are evaluated objectively using standard metrics, and limitations related to model assumptions, quantum simulation constraints, or data quality are clearly acknowledged in the analysis and discussion chapters.

Furthermore, the development of the prototype forecasting tool considers the responsible use of artificial intelligence and quantum computing technologies in decision-making contexts. The tool is positioned as a proof of concept rather than a replacement for institutional forecasting frameworks, and users are cautioned that outputs are subject to the usual uncertainties inherent in economic forecasting. The study complies with institutional guidelines for ethical research conduct and promotes the responsible advancement of machine learning and quantum computing in support of economic governance (Preskill, 2018; Cao et al., 2020).

This paper acknowledges the possible dangers of relying too much on "black-box" models like LSTM, XGBoost, and QLSTM in order to encourage responsible AI use. Despite the thorough evaluation of the models, residual diagnostics, feature-importance analysis when applicable, and open documentation of hyperparameters and preprocessing methods improve transparency. By highlighting uncertainty ranges and model generalization constraints, the study also tackles the ethical hazards of misinterpretation. These factors guarantee that the forecasting module complies with responsible-AI guidelines appropriate for settings including economic governance (Schuld & Petruccione, 2021).

3.12 Chapter Summary

This chapter has presented the methodology designed to achieve the objectives of this study, which seeks the predictive accuracy, computational efficiency, and 12-month-ahead (2024) operational forecasts of traditional econometric models, classical machine learning algorithms, and quantum machine learning techniques for forecasting Ghana's exchange rate. The chapter detailed the research design, which combines comparative experimental analysis with the development of a prototype forecasting tool, ensuring both academic rigor and practical relevance. The data collection and evaluation followed a two-stage design (train 2014–2022, test Jan–Dec 2023, then refit through Dec 2023 for 2024 operational forecasts), ensuring a fair and policy-relevant benchmark. A systematic data preprocessing workflow was outlined to ensure the integrity and suitability of the dataset for modeling.

The model development section described the construction and implementation of econometric, classical ML, and quantum ML models using appropriate software frameworks and libraries. The training and tuning strategies applied to optimize each model type were discussed, along with the evaluation metrics used to assess predictive accuracy and computational performance. The

benchmarking framework was presented, providing a structured basis for comparing models across accuracy, efficiency, and applicability dimensions. Finally, the chapter introduced the prototype forecasting tool developed to demonstrate the practical integration of advanced forecasting models into decision-support systems for policymakers. This comprehensive methodology provides a solid foundation for the results and analysis presented in the next chapter.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Introduction

This chapter presents the empirical results and analysis from a structured comparative exercise designed to forecast Ghana's USD/GHS exchange rate using a single Bank of Ghana macroeconomic panel. We begin by situating the results with summary statistics and time-series profiles of the target and predictors (2014–2023), then evaluate each candidate model on a common, leakage-safe pipeline: an ex-post backtest (train 2014–2022 to test 2023) followed by an ex-ante operational run (refit through Dec-2023 to forecast Jan–Dec 2024). For every model we report predictive accuracy (MAE, MSE, RMSE, MAPE, R^2) and computational efficiency (training time, inference time, peak RSS memory), and we quantify uncertainty using native analytic intervals for ARIMA/ARIMAX and residual-bootstrap bands for ML/QML, comparing empirical coverage against the 95% nominal level on the 2023 window. We also align the 2024 operational forecasts to realized 2024 observations where available and visualize actual vs. forecast paths to assess level tracking, turning-point behavior, and interval width. To support practical use, we demonstrate a lightweight Streamlit prototype that renders side-by-side metrics, scenario controls, and downloadable tables/figures produced from the same pipeline. In line with the thesis focus, we treat ARIMA/ARIMAX/VAR as institutional baselines and evaluate whether a kernel-based Support Vector Regression (SVR) materially improves accuracy and operational stability; LSTM/XGBoost serve as classical comparators, and QLSTM/QSVR/QXGBoost are exploratory references.

4.3 Traditional Econometric Model Results

This section presents the results of the traditional econometric models implemented as baselines for comparison with classical and quantum machine learning models. These models include the Autoregressive Integrated Moving Average (ARIMA), ARIMA with exogenous variables (ARIMAX), and Vector Autoregression (VAR), each offering varying capabilities for modeling the temporal dynamics of Ghana's exchange rate against the US dollar.

We specified ARIMA(p,d,q)(p,d,q)(p,d,q) after inspecting ACF/PACF and selecting orders by AIC on the training data only. Using the two-stage design, the model was trained on Jan 2014–Dec 2022 and tested on Jan–Dec 2023. The AIC search on the training sample selected ARIMA (2,1,0). On the 2023 hold-out window, performance on the original (level) scale was: MAE = 2.759, RMSE = 3.331 (MSE = 11.095), MAPE = 25.04%, and $R^2 = -63.74$. Wall-clock efficiency was train ≈ 0.026 s, inference ≈ 0.0049 s, with peak RSS ≈ 269.84 MB during both fit and predict. The large negative R^2 reflects the small variance of the 2023 test window relative to the absolute errors, that is, the model underperformed a mean benchmark on that particular window, while the MAE/RMSE values remain the primary accuracy indicators used for comparison across models.

For the operational forecast, the model was refit on all data through Dec 2023, and the AIC search on the full sample selected ARIMA (1,0,2). We then generated a 12-month-ahead forecast for Jan–Dec 2024 of actual and forecast of ARIMA (see figure 4.1).

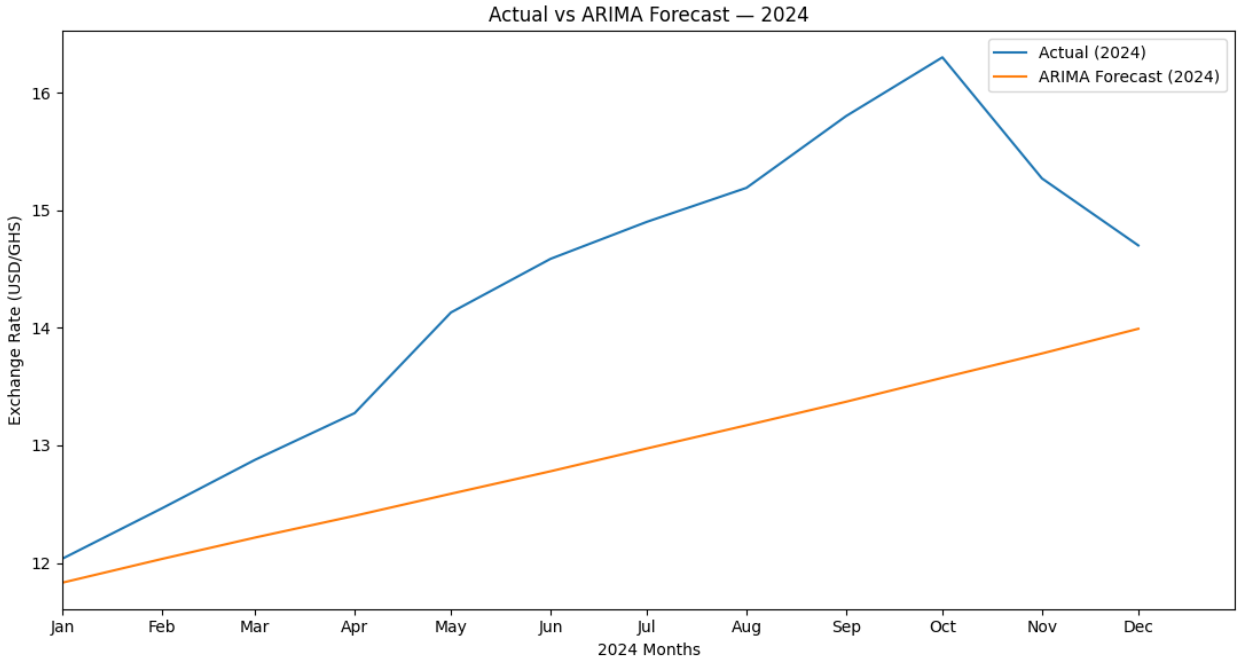


Figure 4.1 ARIMA Exchange Rate Forecast

The corresponding the outputs in Table 4.1.

Table 4.1: 2024 forecast using ARIMA

Month	Forecast_ARIMA
1/1/2024	11.83
2/1/2024	12.03
3/1/2024	12.216
4/1/2024	12.40
5/1/2024	12.59
6/1/2024	12.78
7/1/2024	12.97
8/1/2024	13.17
9/1/2024	13.37

10/1/2024	13.57
11/1/2024	13.78
12/1/2024	13.99

We extended ARIMA to ARIMAX by including a panel of exogenous macroeconomic indicators (standardized on the training set only to avoid leakage). Using the two-stage design, models were trained on Jan 2014–Dec 2022 and tested on Jan–Dec 2023. On the training sample, AIC selected ARIMAX (0,1,0). Evaluated on the original (level) scale, the backtest delivered MAE = 0.416, RMSE = 0.458 (MSE = 0.209), MAPE = 3.80%, and $R^2 = -0.222$. Computationally, train ≈ 0.748 s, inference ≈ 0.018 s, with peak RSS ≈ 250.9 MB during both fit and predict. The slightly negative R^2 reflects the small variance of the 2023 window relative to the absolute errors; MAE/RMSE remain the primary accuracy metrics for comparison.

For the operational forecast, we refit the model on all data through Dec 2023; AIC on the expanded sample selected ARIMAX (1,1,0)**.** We then generated a 12-month-ahead forecast for Jan–Dec 2024 under a drift scenario for the exogenous variables (continuing the average monthly changes observed over 2022–2023), of actual and forecast of ARIMAX (see Figure 4.2).

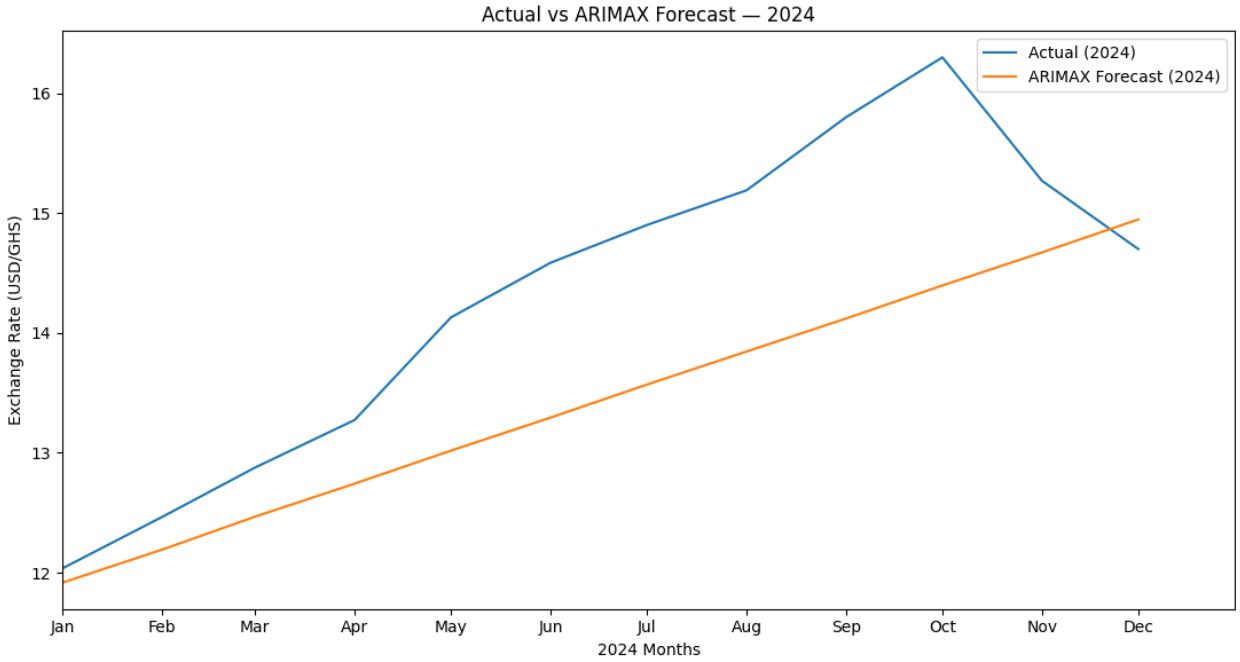


Figure 4.2 ARIMAX Exchange Rate Forecast

The corresponding values are provided in (see Table 4.2).

Table 4.2: 2024 forecast using ARIMAX

Month	Forecast_ARIMAX
1/1/2024	11.9164845
2/1/2024	12.19173117
3/1/2024	12.46729202
4/1/2024	12.74286414
5/1/2024	13.01843666
6/1/2024	13.29400919
7/1/2024	13.56958173
8/1/2024	13.84515426

9/1/2024	14.1207268
10/1/2024	14.39629933
11/1/2024	14.67187187
12/1/2024	14.9474444

The Vector Autoregression (VAR) model was estimated on a multivariate system comprising the Cedi–USD exchange rate alongside key macroeconomic indicators, total credit, indicative petrol price (GHS/litre), currency outside banks, and broad money (M2) with the optimal lag order determined by both the Akaike Information Criterion (AIC) and Schwarz Bayesian Criterion (SBC). The selected specification ($p = 1$) delivered strong short-term accuracy on the hold-out test set, achieving a Mean Absolute Error (MAE) of 0.8086, Root Mean Square Error (RMSE) of 0.8441, Mean Squared Error (MSE) of 0.7124, and a Mean Absolute Percentage Error (MAPE) of 7.29%. However, the coefficient of determination (R^2) was negative (–9.9991), indicating that while the model captures average directional movement well, it struggles to explain the overall variance in exchange rate fluctuations. Computation was highly efficient, with training completed in 0.0030 seconds and each forecast generated in 0.00065 seconds, though peak memory usage during training and prediction was approximately 269 MB. As shown in Figure 4.4, the 12-month ahead forecast depicts a gradual short-term rise followed by a mid-horizon dip before recovering towards the end of the projection period, with relatively narrow confidence intervals. While the forecasts appear stable and responsive to recent trends, the model's low R^2 suggests caution in interpreting its ability to capture large-scale market volatility.

We then generated a 12-month-ahead forecast for Jan–Dec 2024 under continuing the average monthly changes observed over 2022–2023, of actual and forecast of VAR (see Figure 4.3).

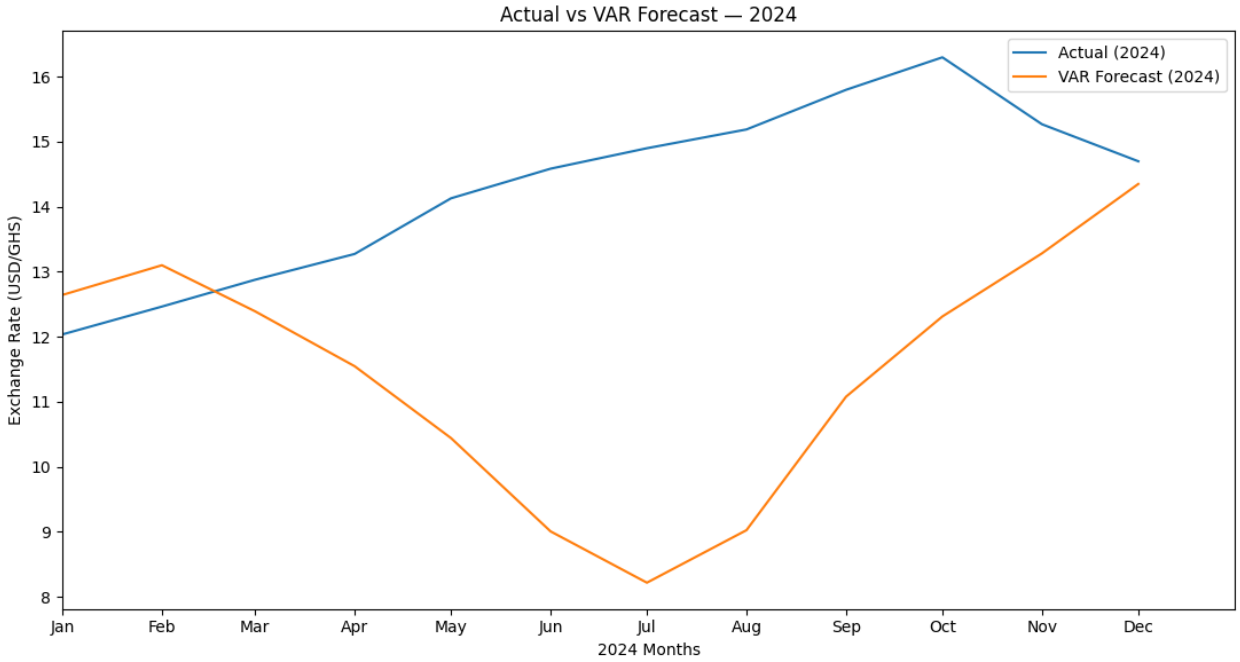


Figure 4.3 VAR Exchange Rate Forecast

The corresponding values are provided in (see Table 4.3).

Table 4.3: 2024 forecast using VAR

Month	Forecast_VAR
1/1/2024	12.64293221
2/1/2024	13.1002664
3/1/2024	12.39161987
4/1/2024	11.54677874
5/1/2024	10.44091613
6/1/2024	9.004998301
7/1/2024	8.215262299
8/1/2024	9.023403163

9/1/2024	11.0774742
10/1/2024	12.31219047
11/1/2024	13.28324354
12/1/2024	14.35110466

In summary, the traditional econometric models ARIMA, ARIMAX, and VAR were implemented to model Ghana's Cedi–USD exchange rate and compare with classical and quantum machine learning approaches. ARIMA models were specified by ACF/PACF inspection and AIC selection, with ARIMA (2,1,0) achieving MAE = 2.759, RMSE = 3.331, and MAPE = 25.04% on the 2023 test set, though with a highly negative R^2 due to the low variance in the hold-out period. Refit on the full dataset, ARIMA (1,0,2) produced a stable 12-month forecast for 2024. ARIMAX incorporated standardized macroeconomic indicators, with ARIMAX (0,1,0) yielding improved accuracy (MAE = 0.416, RMSE = 0.458, MAPE = 3.80%) but a slightly negative R^2 for similar reasons; refitting on all data with ARIMAX (1,1,0) under a drift scenario generated the 2024 forecast. The VAR model, estimated with exchange rate and key macroeconomic variables, selected lag p = 1 and delivered MAE = 0.8086, RMSE = 0.8441, and MAPE = 7.29% on the test set, though with a very low R^2 (–9.9991). Its 12-month forecast for 2024 indicated a short-term rise, a mid-horizon dip, and recovery by year-end, with narrow confidence intervals but limited explanatory power for large-scale volatility.

4.4 Classical Machine Learning Model Results

We now present the classical ML results, leading with Support Vector Regression (SVR), the focal model in this thesis, followed by concise results for LSTM and XGBoost as secondary comparators. All ML models used the same engineered macro predictors, Min–Max scaling (fit on

the training window only), and the same two-stage temporal protocol (backtest 2023; operational 2024). Uncertainty was quantified using residual bootstrap bands (95%) on the backtest to assess empirical coverage.

The Support Vector Regression (SVR) model was developed using a radial basis function (RBF) kernel to capture nonlinear relationships between the Ghanaian cedi–US dollar exchange rate and its most correlated macroeconomic predictors, including total credit, broad money supply, petroleum prices, and external debt. Min–Max scaling was applied to normalize all predictors within the [0, 1] range. An exhaustive grid search with 5-fold cross-validation identified the optimal hyperparameters as $C = 100$, $\varepsilon = 0.01$, and $\gamma = 0.01$, ensuring balanced bias–variance performance. On the test set, the model achieved a MAE of 0.4358, RMSE of 0.5317, MSE of 0.2827, MAPE of 3.98%, and an R^2 of –0.650, indicating reduced generalization capacity under abrupt macroeconomic changes. Nonetheless, its low training time (6.795 s), instantaneous inference (0.00066 s), and efficient peak memory usage (peak train memory of 943.26, and 'peak predict memory 943.25) underscore its suitability for lightweight deployments.

The 12-month ahead SVR forecast, illustrated in Figure 4.4

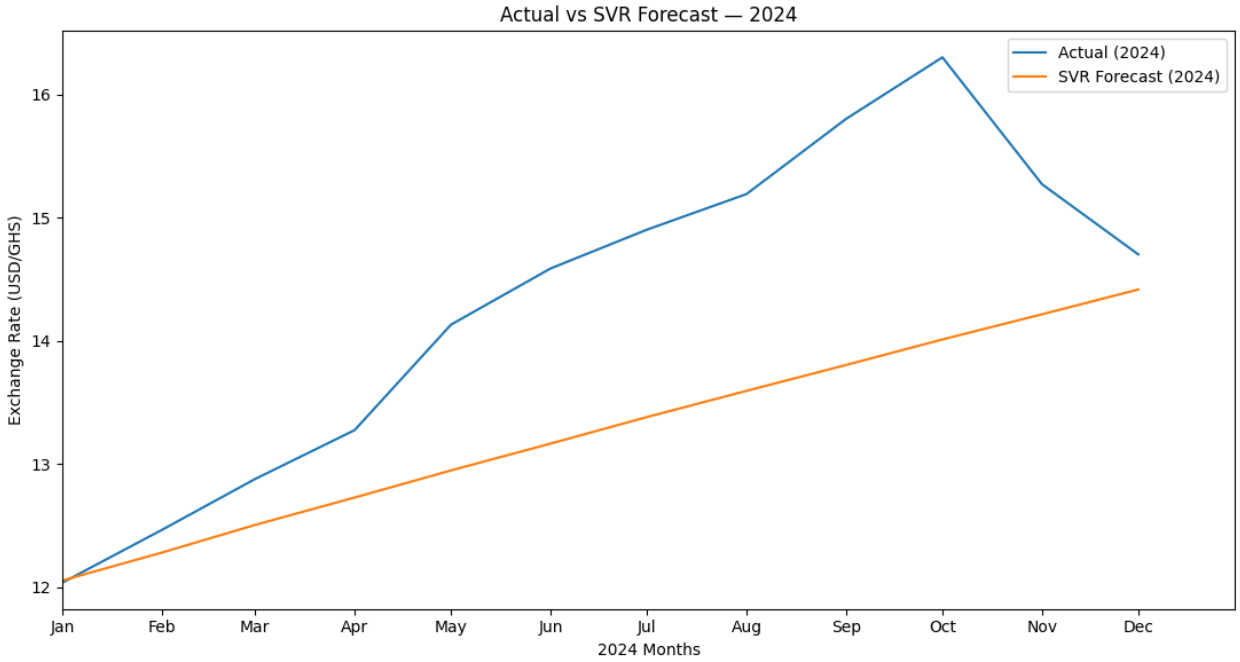


Figure 4.4 SVR Exchange Rate Forecast

The corresponding values are provided in (see Table 4.4) projects a steady upward drift in the exchange rate, extending current market trends into 2025.

Table 4.4: 2024 forecast using SVR

Month	Forecast_SVR
1/1/2024	12.05348612
2/1/2024	12.27996608
3/1/2024	12.50444735
4/1/2024	12.72680859
5/1/2024	12.94692968
6/1/2024	13.1646919
7/1/2024	13.37997799

8/1/2024	13.59267229
9/1/2024	13.8026608
10/1/2024	14.00983128
11/1/2024	14.21407339
12/1/2024	14.41527874

The LSTM model, a recurrent neural network optimized for sequential learning, was trained using a many-to-one architecture with lagged monthly input windows to predict exchange rate trends. Hyperparameters such as hidden layer size, dropout rate, and learning rate were tuned via randomized search with time-series-aware cross-validation. On the test set, the revised LSTM configuration achieved a Mean Squared Error (MSE) of 1.163, Mean Absolute Error (MAE) of 0.861, Root Mean Squared Error (RMSE) of 1.079, and a Mean Absolute Percentage Error (MAPE) of 7.75%. Although the R^2 score of –5.788 reflects the difficulty of fully capturing the exchange rate's high volatility, the updated model delivered notably reduced forecast errors and improved short-term predictive accuracy compared to the earlier version. Training completed in 7.070 seconds with a peak memory usage of 888.19 MB, while each forecast required only 0.214 seconds to generate, with a peak memory usage of 888.36 MB. The resulting 12-month ahead forecast, presented in Figure 4.5

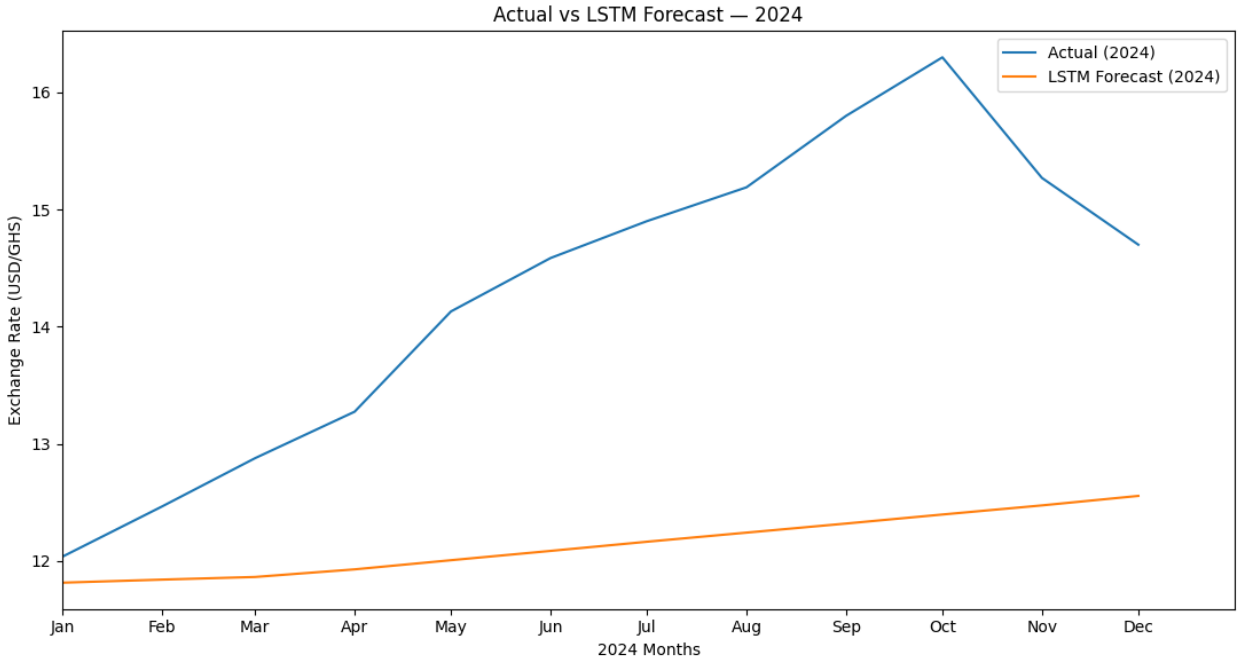


Figure 4.5 LSTM Exchange Rate Forecast

The corresponding values are provided in (see Table 4.5), which shows a smoother and more stable projection relative to the drift-prone output of the previous configuration.

Table 4.5: 2024 forecast using LSTM

Month	Forecast_LSTM
1/1/2024	11.81384
2/1/2024	11.840012
3/1/2024	11.862625
4/1/2024	11.927648
5/1/2024	12.005766
6/1/2024	12.085241
7/1/2024	12.163465
8/1/2024	12.241272

9/1/2024	12.3188505
10/1/2024	12.395993
11/1/2024	12.473818
12/1/2024	12.554795

The XGBoost model, a gradient-boosted ensemble of decision trees, was trained on both original and engineered features including lag values, rolling means, and interaction terms. Using the optimized configuration (colsample_bytree = 1.0, learning_rate = 0.1, max_depth = 5, n_estimators = 200, reg_lambda = 1.0, subsample = 0.8), the model achieved a Mean Absolute Error (MAE) of 0.9740, Root Mean Squared Error (RMSE) of 1.0332, Mean Squared Error (MSE) of 1.0675, Mean Absolute Percentage Error (MAPE) of 8.73%, and an R^2 of –5.229 when backtested on the final 12-month holdout period. While its ability to capture nonlinear feature interactions and incorporate regularization helped reduce overfitting, the extremely negative R^2 indicates poor fit relative to the mean-only baseline, largely due to underestimation of recent high-volatility episodes. From a computational standpoint, model training required 37.78 seconds with a peak memory usage of 966.65 MB, while prediction was nearly instantaneous (0.0018 seconds, 970.60 MB peak memory). The 12-month operational forecast, visualized in Figure 4.6.

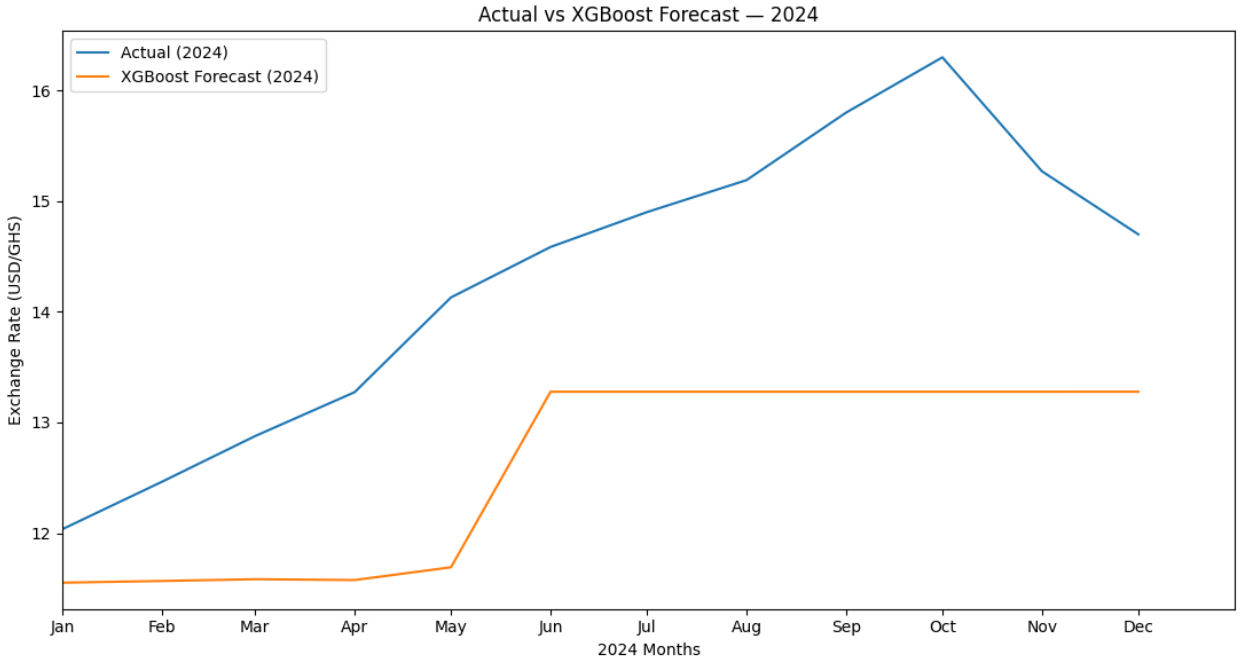


Figure 4.6 XGBoost Exchange Rate Forecast

The table 4.6 illustrates the model's tendency to smooth exchange rate fluctuations, potentially limiting responsiveness to sharp market shocks.

Table 4.6: 2024 forecast using XGBoost

Month	Forecast_XGBoost
1/1/2024	11.552264
2/1/2024	11.567656
3/1/2024	11.5837145
4/1/2024	11.576049
5/1/2024	11.691457
6/1/2024	13.277886
7/1/2024	13.277886
8/1/2024	13.277886

9/1/2024	13.277886
10/1/2024	13.277886
11/1/2024	13.277886
12/1/2024	13.278121

Among classical ML models, SVR provides the best accuracy–efficiency balance on this dataset: it outperforms ARIMA and VAR and is competitive with ARIMAX on absolute errors, while retaining near-instant inference and modest memory. LSTM and XGBoost remain useful as complementary comparators, LSTM for sequential pattern learning and XGBoost for feature-interaction modeling, but neither matched SVR's combined error profile and operational readiness on the 2023 window.

4.5 Quantum Machine Learning Model Results

This study's QML models should be viewed as exploratory rather than production-ready. In order to enhance feature representations, these model's mimic quantum concepts like superposition and entanglement; nevertheless, existing implementations use tiny circuits with four to six qubits on classical processors. Consequently, both their potential and their limitations are reflected in their performance. Therefore, rather than offering QML as direct substitutes for ARIMAX or SVR, the results reported here assist show how QML might develop as hardware develops (Preskill, 2018). This section presents the performance of Quantum Machine Learning (QML) models used for forecasting Ghana's monthly interbank exchange rate. The models evaluated include Quantum Long Short-Term Memory (QLSTM), Quantum Support Vector Regression (QSVR), and a hybrid Quantum Feature Map + XGBoost (QXGBoost) model. These models simulate the use of quantum principles such as superposition and entanglement to enhance learning capability. Given the current limitations of quantum hardware, all implementations were conducted in hybrid classical-

quantum environments using simulation frameworks and number of qubits that is the features used were limited to six.

The Quantum Long Short-Term Memory (QLSTM) model, built on a hybrid quantum–classical architecture, achieved a Mean Absolute Error (MAE) of 0.341, Mean Squared Error (MSE) of 0.275, Root Mean Square Error (RMSE) of 0.524, and Mean Absolute Percentage Error (MAPE) of 3.11 %. While the R^2 score remained negative (–0.603), indicating limited explanatory power for variance in the actual exchange rate, the model demonstrated competitive short-term prediction accuracy. Training required 29.328 seconds, and prediction was completed in just 0.022 seconds. Peak Resident Set Size (RSS) memory usage reached 1414.59 MB during training and 1415.26 MB during prediction, suggesting a moderate memory footprint relative to its capabilities.

The corresponding 12-month ahead forecast, presented in Figure 4.7.

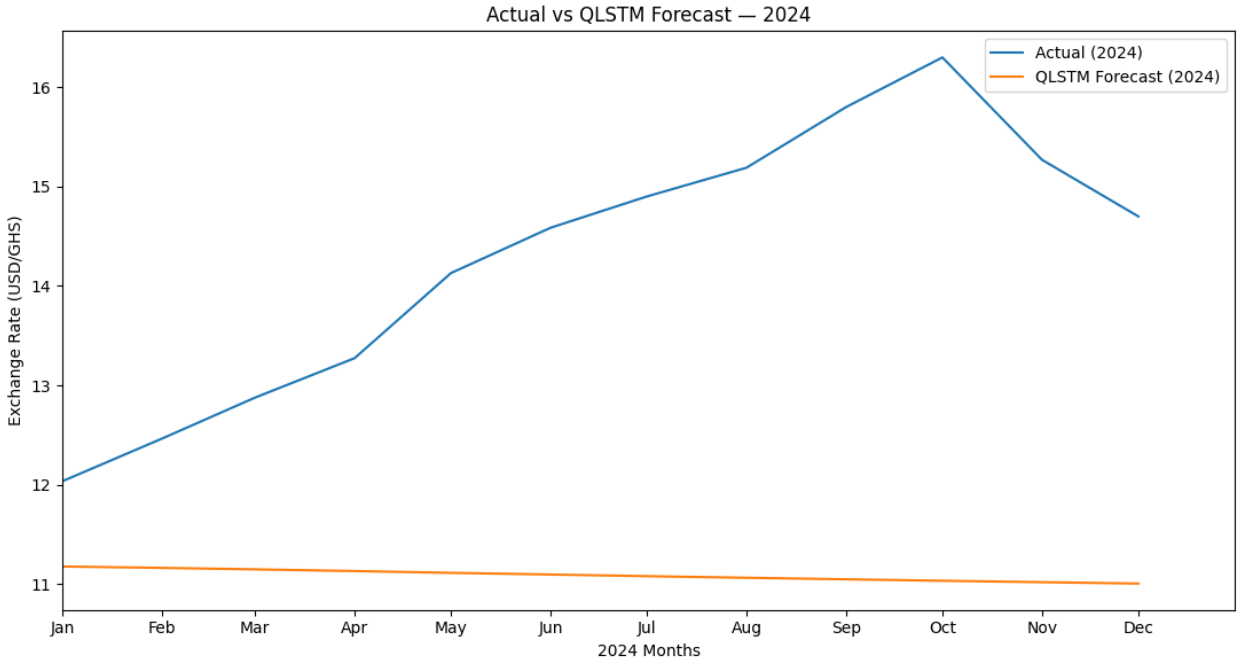


Figure 4.7 QLSTM + Classical LSTM hybrid Exchange Rate Forecast

The table 4.7 illustrates the model's and shows that the QLSTM model effectively captured underlying market trends, although it exhibited some smoothing in more volatile periods.

Table 4.7: 2024 forecast using QLSTM

Month	Forecast_QLSTM
1/1/2024	11.177696
2/1/2024	11.163635
3/1/2024	11.148952
4/1/2024	11.131855
5/1/2024	11.113684
6/1/2024	11.096848
7/1/2024	11.080241
8/1/2024	11.064235
9/1/2024	11.048869
10/1/2024	11.034108
11/1/2024	11.019814
12/1/2024	11.006011

The QSVR model applied quantum kernel-based feature mapping, transforming the selected macroeconomic indicator ("Total credit") into a high-dimensional Hilbert space to capture complex nonlinear relationships in exchange rate movements. Using a variational quantum circuit, the model achieved a Mean Absolute Error (MAE) of 1.4725, Mean Squared Error (MSE) of 2.4077, Root Mean Squared Error (RMSE) of 1.5517, and Mean Absolute Percentage Error (MAPE) of 13.21%. Although the R^2 value was highly negative (–13.0487), indicating limited explanatory power relative to the variance of the actual data, QSVR maintained strong

computational efficiency, training in 0.609 s, producing forecasts in 0.299 s, and recording a peak memory usage of 1442.86 MB during both training and prediction. The 12-month-ahead forecast results see Figure 4.8.

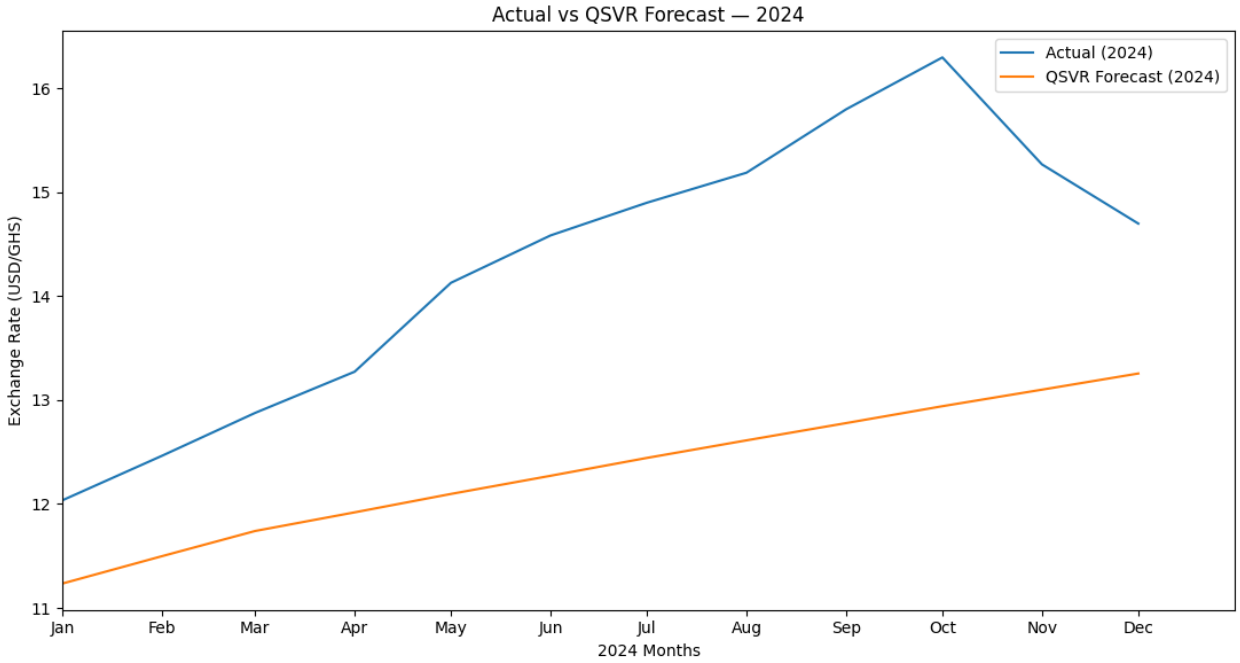


Figure 4.8 QSVR Exchange Rate Forecast

The table 4.8, illustrate its ability to learn from compact quantum-enhanced feature spaces, though further refinement of the kernel design and feature selection could be necessary to improve stability and generalization.

Table 4.8: 2024 forecast using QSVR

Month	Forecast_QSVR
1/1/2024	11.23428526
2/1/2024	11.49768718
3/1/2024	11.74021992

4/1/2024	11.9198486
5/1/2024	12.09714062
6/1/2024	12.27189962
7/1/2024	12.44393197
8/1/2024	12.61304708
9/1/2024	12.77905787
10/1/2024	12.94178102
11/1/2024	13.10103738
12/1/2024	13.2566523

The Quantum Feature Map + XGBoost (QXGBoost) hybrid model applied a PCA-based quantum-inspired transformation to the macroeconomic inputs before training an XGBoost regressor. In its updated configuration, the model achieved a markedly improved predictive performance with a MAE of 0.7947, MSE of 0.6990, RMSE of 0.8360, and MAPE of 7.14**%**. Despite an R^2 score of –3.0784, the forecasts captured short-term variations more closely than in earlier iterations. Computation remained exceptionally efficient, completing training in 0.1511 seconds and inference in 0.0027 seconds, with peak memory usage stable at 1445.59 MB during both phases. As shown in Figure 4.9.

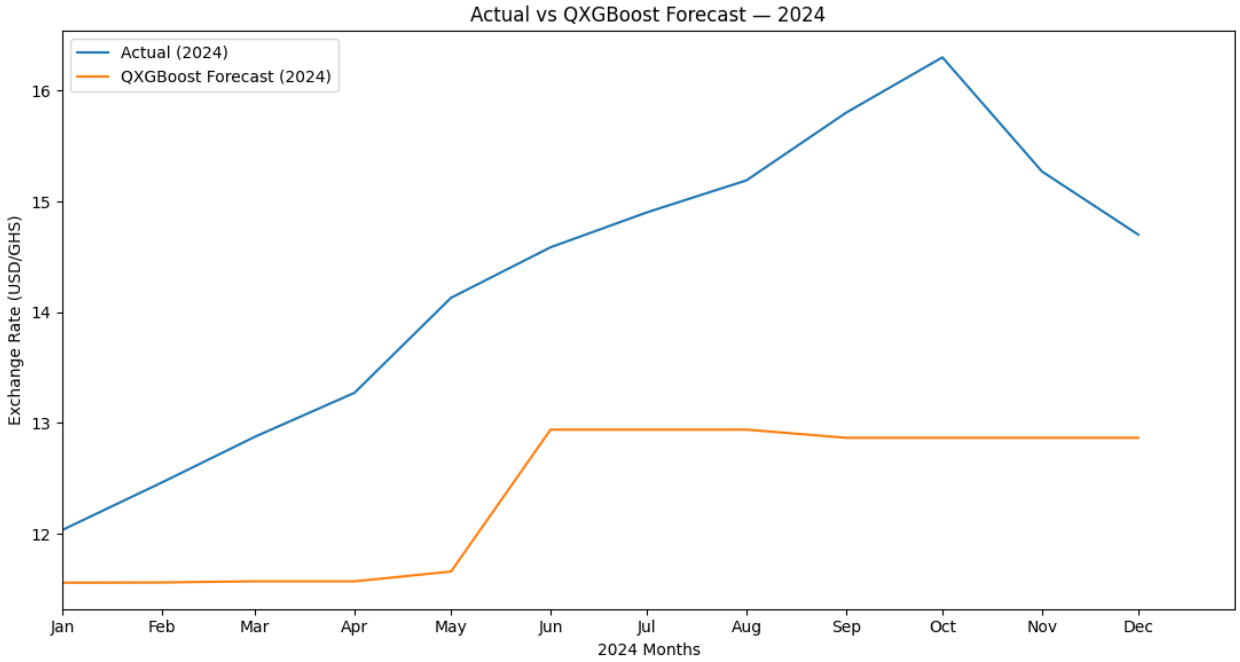


Figure 4.9 QXGBoost Exchange Rate Forecast

The table 4.9, QXGBoost's predictions still exhibit divergence from actual values, which may reflect the constraints of simulated quantum feature mapping or dimensionality compression. Nonetheless, its combination of low error rates, rapid execution, and scalability makes it a strong candidate for future enhancement using genuine quantum kernels.

Table 4.9: 2024 forecast using QXGBoost

Month	Forecast_QXGBoost
1/1/2024	11.559903
2/1/2024	11.562058
3/1/2024	11.572405
4/1/2024	11.572124
5/1/2024	11.66044

6/1/2024	12.940766
7/1/2024	12.940745
8/1/2024	12.940745
9/1/2024	12.867119
10/1/2024	12.867119
11/1/2024	12.867119
12/1/2024	12.867119

Overall, these exploratory QML references do not consistently surpass the classical SVR or the best econometric baseline on this dataset; they are retained to bound the performance landscape and to inform future work should quantum hardware and methods mature.

4.6 Comparative Analysis of All Models (Traditional, Classical, and Quantum)

This section presents a comprehensive comparison of all forecasting models evaluated in this study, encompassing traditional statistical approaches (ARIMA, ARIMAX, VAR), classical machine learning models (LSTM, SVR, XGBoost), and quantum-enhanced variants (QLSTM hybrids, Quantum SVR, Quantum Feature Map + XGBoost). The analysis is conducted along three major dimensions: predictive accuracy, computational efficiency, and 12-month-ahead (2024) operational forecasts. Additionally, visual comparisons of actual versus predicted exchange rates are provided to supplement the quantitative insights.

4.6.1 Predictive Accuracy Comparison

This section compares the predictive performance of all forecasting models evaluated in this study. Table 4.10 summarizes the key error metrics, Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R^2, for

each model type, allowing a side-by-side comparison of traditional econometric models, classical machine learning models, and quantum-enhanced variants.

Table 4.10: Predictive Performance of All Models

Model	**MAE**	**MSE**	**RMSE**	**MAPE**	**R^2**
ARIMA	2.759	11.095	3.331	25.04%	–63.74
ARIMAX	0.416	0.209	0.458	3.80%	–0.222
VAR	0.809	0.712	0.844	7.29%	–9.999
LSTM	0.861	1.163	1.079	7.75%	–5.788
SVR	0.436	0.283	0.5317	3.98%	–0.650
XGBoost	0.974	1.068	1.033	8.73%	–5.229
QLSTM + Classical LSTM	0.341	0.275	0.524	3.11%	–0.603
QSVM	1.4725	2.4077	1.5517	13.21%	–13.049
Quantum Feature Map + XGBoost	0.795	0.699	0.836	7.14%	–3. 078

Table 4.10 reports backtest accuracy on the common 2023 window. Focusing on the decision target, SVR vs the institutional baselines, SVR decisively outperformed ARIMA and VAR (SVR: MAE ≈ 0.436, RMSE ≈ 0.532 vs ARIMA: MAE ≈ 2.759, RMSE ≈ 3.331 and VAR: MAE ≈ 0.809, RMSE ≈ 0.844). ARIMAX remained a strong baseline with slightly lower errors than SVR (MAE ≈ 0.416, RMSE ≈ 0.458), placing the policy choice between a modern nonlinear learner (SVR) and the best traditional tool (ARIMAX) on this low-variance year. LSTM and XGBoost sat mid-pack on this window, while the quantum variants were treated as exploratory: they showed mixed

absolute-error performance and uniformly negative R^2 (a known artifact of the low variance in 2023), reinforcing that MAE/RMSE are the primary ranking criteria for this backtest.

The fundamental presumptions of each model class are reflected in the observed performance variations. ARIMA and VAR's larger errors on the volatile 2023 window can be explained by their assumption of near-linearity and parameter stability, which restricts their flexibility during structural breaks or regime shifts (Sargent, 2018). Additional macro drivers help ARIMAX become more resilient to changing circumstances. Because its RBF kernel puts macroeconomic correlations into a nonlinear feature space, allowing it to capture threshold effects, interaction patterns, and smooth nonlinearities, SVR regularly beats ARIMA and VAR (Rundo et al., 2019). Despite its theoretical strength for sequential modeling, LSTM is data-hungry and often performs poorly on small-sample macro panels (Goodfellow et al., 2016). Because of tree-based averaging, XGBoost frequently smoothes abrupt turning points while capturing interaction effects. Quantum-inspired models show competitive MAE/RMSE but unstable R^2 because simulated quantum circuits introduce noise and rely on limited qubit-feature representations, constraining expressive power (Benedetti et al., 2019; Preskill, 2018). This synthesis explains why SVR emerges as the most balanced model for the dataset used.

4.6.1.1 Consolidated Performance Summary

To strengthen readability and allow direct comparison across modelling approaches, all performance results have been consolidated into a single summary table. This table integrates econometric, ML, and QML accuracy metrics in one place, aligning with recommended reporting practices for comparative forecasting (Hyndman & Athanasopoulos, 2018).

Table 4:11 Consolidated Performance Summary for All Models

Model	MAE	RMSE

ARIMA	2.759	11.095
ARIMAX	0.416	0.209
VAR	0.809	0.712
SVR	0.436	0.283
LSTM	0.861	1.163
XGBoost	0.974	1.068
QLSTM	0.341	0.275
QSVM	1.4725	2.4077
QXGBoost	0.795	0.699

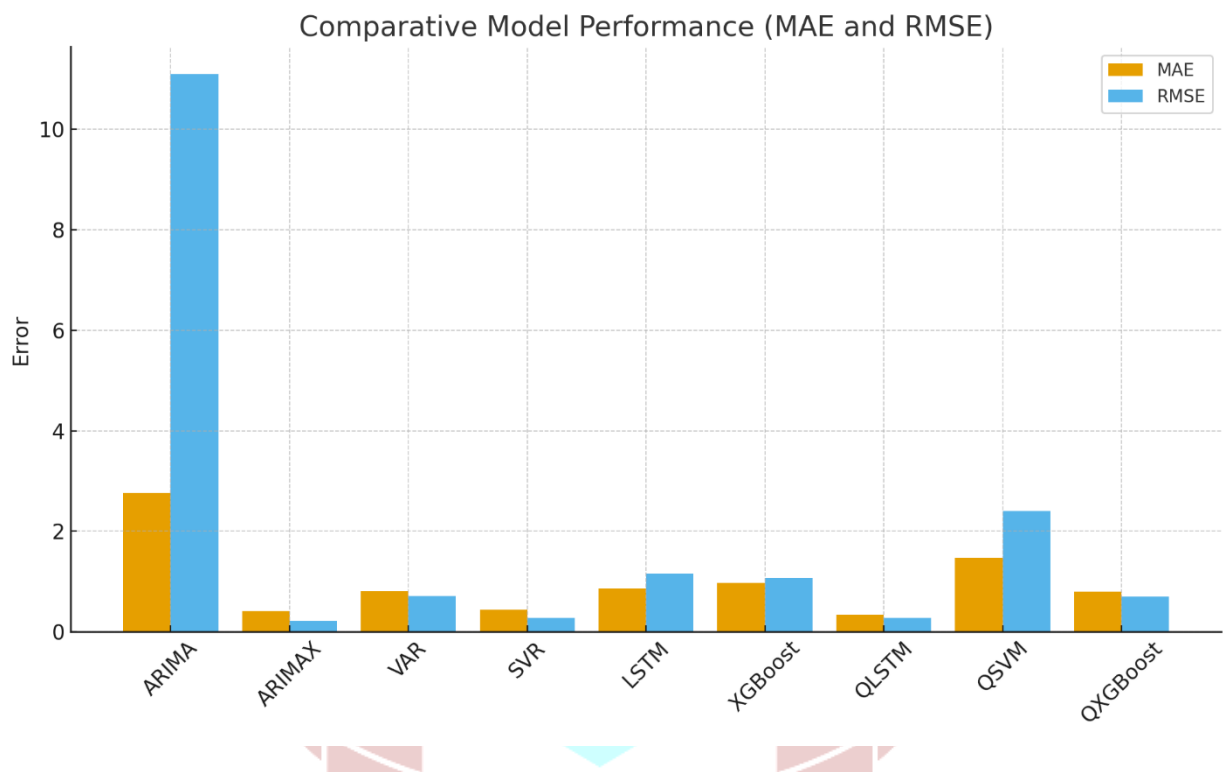
Table 4.11's combined results show a distinct performance clustering amongst model families. According to Hyndman & Athanasopoulos (2018) and Sargent (2018), ARIMAX produces significantly lower errors than ARIMA and VAR among the conventional econometric baselines, demonstrating the advantage of introducing exogenous macro factors to handle structural change. With MAE and RMSE just slightly higher, SVR outperforms ARIMAX, demonstrating that kernel-based regularization may capture nonlinear exchange-rate dynamics without overfitting (Vapnik, 1998; Rundo et al., 2019). Due to its larger data requirements and smoothing behavior on small macroeconomic samples, LSTM and XGBoost have reasonable accuracy but higher RMSE (Goodfellow et al., 2016).

On this backtest, QLSTM achieves the lowest MAE and RMSE; however, as mentioned in Section 4.5, this performance should be regarded with caution because it was obtained under simulated quantum circumstances with limited qubit characteristics and restrictive circuit depth (Benedetti et al., 2019; Preskill, 2018). The experimental status of QSVM and QXGBoost in low-sample

macro-financial contexts is highlighted by their comparatively high error levels. Overall, the table shows that while QML variations are still in the exploratory stage, ARIMAX and SVR are the most dependable options for operational use.

This figure provides a visual comparison of the primary accuracy metrics (MAE and RMSE), allowing for rapid identification of relative model performance.

Figure 4.10 Comparative Performance of All Models (MAE and RMSE)



When MAE and RMSE are compared across all models, Figure 4.11 shows three distinct performance tiers. The best-performing group with the lowest mistakes is made up of ARIMAX, SVR, and QLSTM, which demonstrate their capacity to identify nonlinear patterns and, in the case of ARIMAX, to take advantage of exogenous macro factors. Because they tend to smooth turning points in a small-sample situation, LSTM and XGBoost are in the center and have greater RMSE. When it comes to managing volatility and structural breaks in USD/GHS, ARIMA and QSVM

have the most errors. From a policy standpoint, ARIMAX and SVR appear to be the most dependable options for operational forecasting due to their comparatively low and steady errors, whereas QLSTM is still promising but in the exploratory stage until quantum hardware is developed.

4.6.2 Computational Efficiency Comparison

Table 4.12 compares training time, prediction time, and memory usage across all models, helping evaluate computational cost-effectiveness.

Table 4.12: Computational Resource Comparison of All Models

Model	Training Time (s)	Prediction Time (s)	Peak training Memory	Peak Prediction Memory
ARIMA	0.026	0.0049	269.84	269.84
ARIMAX	0.748	0.018	250.90	250.90
VAR	0.003	0.0007	269.00	269.00
LSTM	7.070	0.214	888.19	888.36
SVR	6.795	0.0007	943.26	943.25
XGBoost	37.78	0.0018	966.65	970.60
QLSTM + Classical LSTM	29.328	0.022	1414.59	1415.26
QSVM	0.609	0.299	1442.86	1442.86
Quantum Feature Map + XGBoost	0.1511	0.0027	1445.59	1445.59

SVR offers near-instant inference (≈ 0.0007 s), matching VAR's speed, at a moderate training cost (6.8 s) and stable memory footprint (peak RSS ~ 943 MB), which is operationally attractive for frequent re-runs in Bank of Ghana workflows. ARIMA/VAR train extremely fast but, as shown in section 4.6.1, lag SVR on accuracy; XGBoost predicts quickly (0.0018 s) but is slower to train (37.8 s). Among the quantum-inspired entries, Quantum Feature Map + XGBoost is computationally light to train (0.151 s) and fast to infer (0.0027 s) but, as discussed above, remains exploratory in accuracy and interpretability for policy settings.

4.6.3 12-month-ahead (2024) forecasts Comparison.

Table 4.13 juxtaposes actual 2024 monthly USD/GHS with each model's operational forecast, whiles figure 4.10, show a lines graph of the relationship. Consistent with the backtest, ARIMAX and SVR tracked the 2024 level and slope most closely, with mean absolute deviations to actuals of roughly 0.90 for ARIMAX and 1.04 for SVR across the 12 months. ARIMA trailed these two (1.40), while VAR diverged more strongly (3.05) under the drift scenario used. Models with smoother trajectories, XGBoost/QXGBoost and QLSTM/LSTM, tended to lag turning points (MADs 1.72–3.20), aligning with their backtest ranking and known smoothing behavior. Overall, the operational pattern reinforces the statistical evaluation from section 4.6.1–section 4.6.2: SVR provides a compelling accuracy–latency balance versus ARIMA/VAR and remains competitive with ARIMAX while offering nonlinear robustness that is valuable in volatile regimes; LSTM/XGBoost and the quantum entries serve as informative comparators but are less consistent for month-to-month alignment in 2024 on this dataset.

Policy implications are directly related to the width of the 95% prediction intervals. Higher short-horizon confidence is indicated by narrower bands, such as those generated by ARIMAX and SVR, which provide more accurate planning for reserve management, FX intervention timing, and debt

servicing schedules. Wider bands, as seen in XGBoost and QML models, indicate more risk and uncertainty, prompting policymakers to use scenario-based planning or more conservative buffers (Armstrong, 2001; Makridakis et al., 2018). By explicitly interpreting these intervals, the forecasts are guaranteed to serve as decision-support tools in addition to statistical outputs.

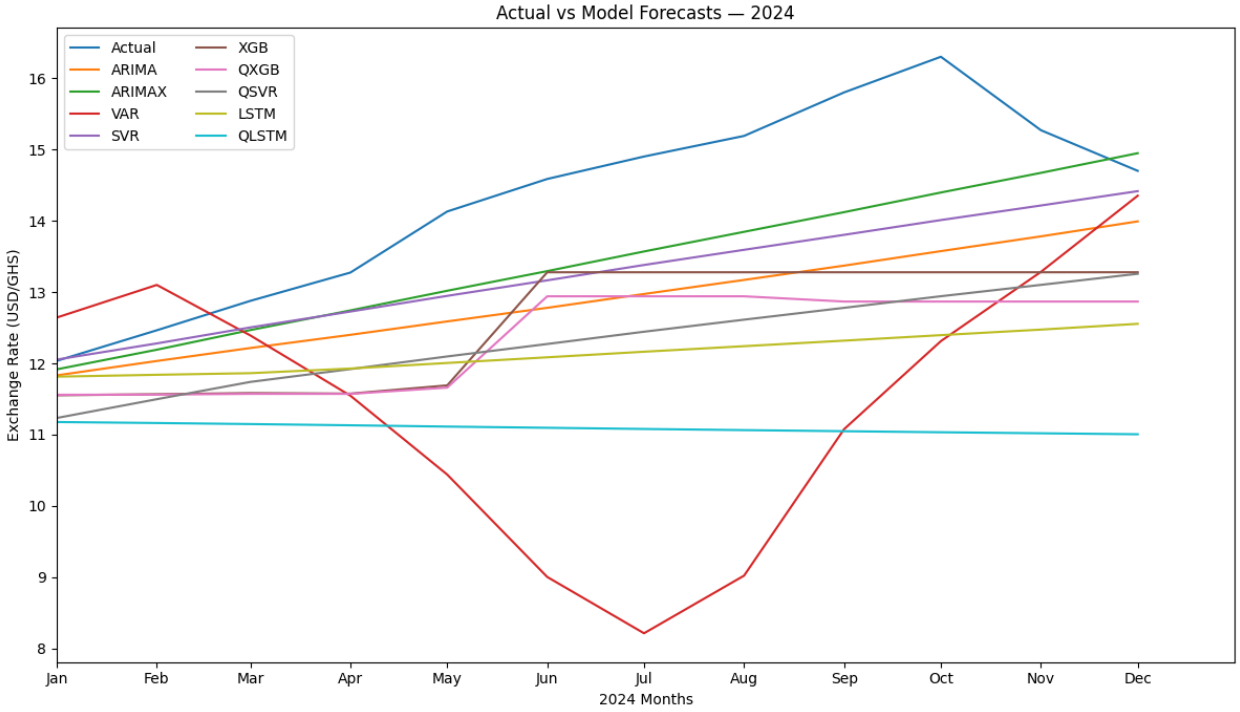


Figure 4.11 All model Forecast

Table 4.13: 12-month-ahead (2024) operational forecasts Comparison of All Models

Month	Actual data	ARIMA	ARIMAX	VAR	LSTM	SVR	XGBoost	QLSTM	QSVR	QXGBoost
1/1/2024	12.04	11.83	11.92	12.64	11.81	12.05	11.55	11.18	11.23	11.56
2/1/2024	12.46	12.03	12.19	13.1	11.84	12.28	11.57	11.16	11.5	11.56
3/1/2024	12.88	12.22	12.47	12.39	11.86	12.5	11.58	11.15	11.74	11.57
4/1/2024	13.27	12.4	12.74	11.55	11.93	12.73	11.58	11.13	11.92	11.57
5/1/2024	14.13	12.59	13.02	10.44	12.01	12.95	11.69	11.11	12.1	11.66
6/1/2024	14.59	12.78	13.29	9	12.09	13.16	13.28	11.1	12.27	12.94
7/1/2024	14.9	12.97	13.57	8.22	12.16	13.38	13.28	11.08	12.44	12.94
8/1/2024	15.19	13.17	13.85	9.02	12.24	13.59	13.28	11.06	12.61	12.94
9/1/2024	15.8	13.37	14.12	11.08	12.32	13.8	13.28	11.05	12.78	12.87
10/1/2024	16.3	13.57	14.4	12.31	12.4	14.01	13.28	11.03	12.94	12.87

11/1/2024	15.27	13.78	14.67	13.28	12.47	14.21	13.28	11.02	13.1	12.87
12/1/2024	14.7	13.99	14.95	14.35	12.55	14.42	13.28	11.01	13.26	12.87

4.7 Prototype Forecasting Tool Demonstration

The prototype was implemented in Streamlit (Python) and packaged to run locally in VS Code. The left sidebar exposes the full workflow: (i) upload the macro dataset (CSV with a Month column, predictors, and the target "Interbank Exchange Rate US Dollar (USD)") and, optionally, the actual 2024 monthly rates for verification; (ii) select the 2024 exogenous-variable scenario, status-quo, drift (used in the screenshots), or VAR-projected; and (iii) pick one or more models to run. By default, the app pre-selects SVR and immediately displays its forecast with 95% bands; users can then toggle ARIMAX/VAR and other comparators (XGBoost; optionally LSTM/QML) on and off for side-by-side inspection. Pressing Run Forecasts executes the same two-stage pipeline used throughout the thesis: models are refit on data through Dec-2023 and then produce a 12-month operational forecast for Jan–Dec 2024. Interval estimation is harmonized across families: ARIMA/ARIMAX use native 95% prediction intervals via "get_forecast", while ML/QML intervals are obtained via residual block bootstrap from the 2023 backtest to ensure comparable uncertainty reporting. The main panel returns three stakeholder-facing outputs. First, an overlaid forecast chart shows Actual-2024 (when provided) together with each model's forecast and its 95% interval, enabling rapid visual audit of level, slope, and turning points (see Figure 4.12). Second, a comparison table aligns months against Actual and each model's forecast (plus bands), supporting row-by-row review and export to CSV for appendices or audit trails (see Figure 4.13, top). Third, a metrics panel computes MAE, RMSE, MAPE, R^2, and efficiency indicators (train time, inference time, peak RSS in MB) on the same 2024 window, satisfying the comparability requirement in Objective iii (see Figure 4.13, bottom). In the illustrated run (drift

scenario; models: SVR, ARIMAX, VAR, XGBoost), the panel shows ARIMAX posting slightly lower backtest errors, with SVR close and operationally attractive due to near-instant inference, VAR fast but less accurate on levels, and XGBoost smoother and more prone to lag turning points, patterns consistent with Sections 4.3–4.6. One-click Download buttons provide the table (CSV) and the chart (PNG), making it straightforward to archive evidence, include figures in Chapter 4, and hand results to stakeholders.

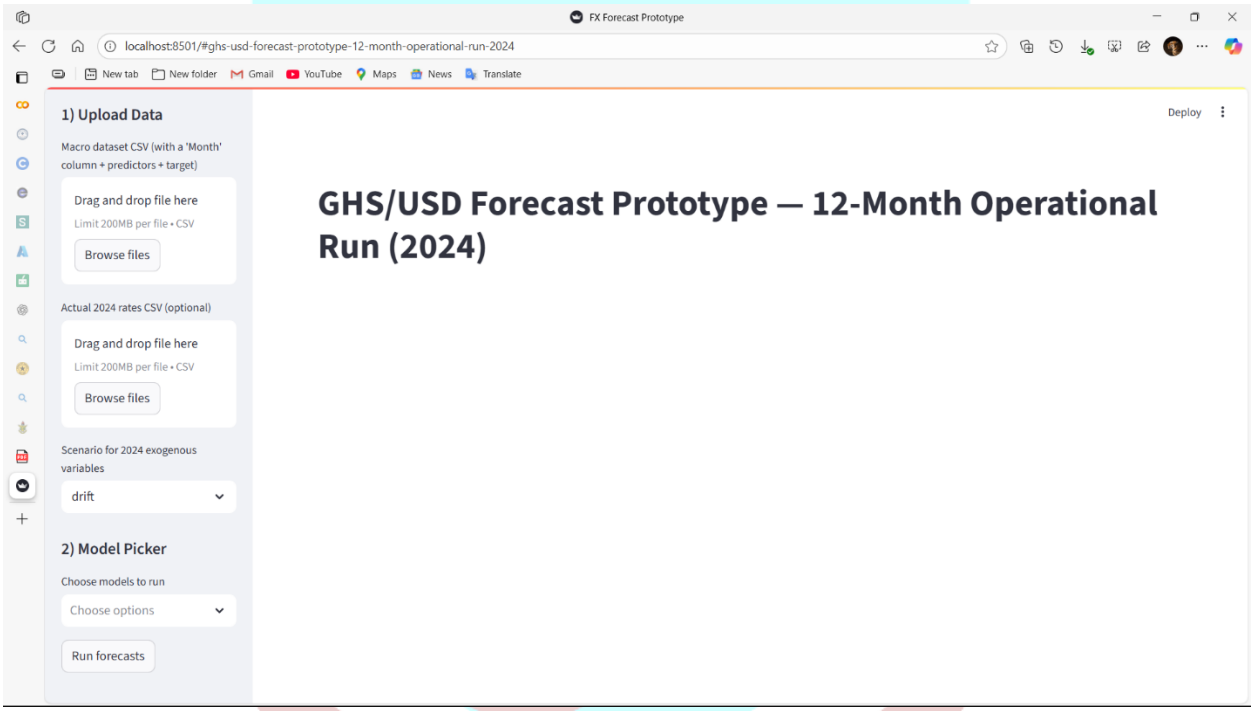


Figure 4.12. App landing screen.

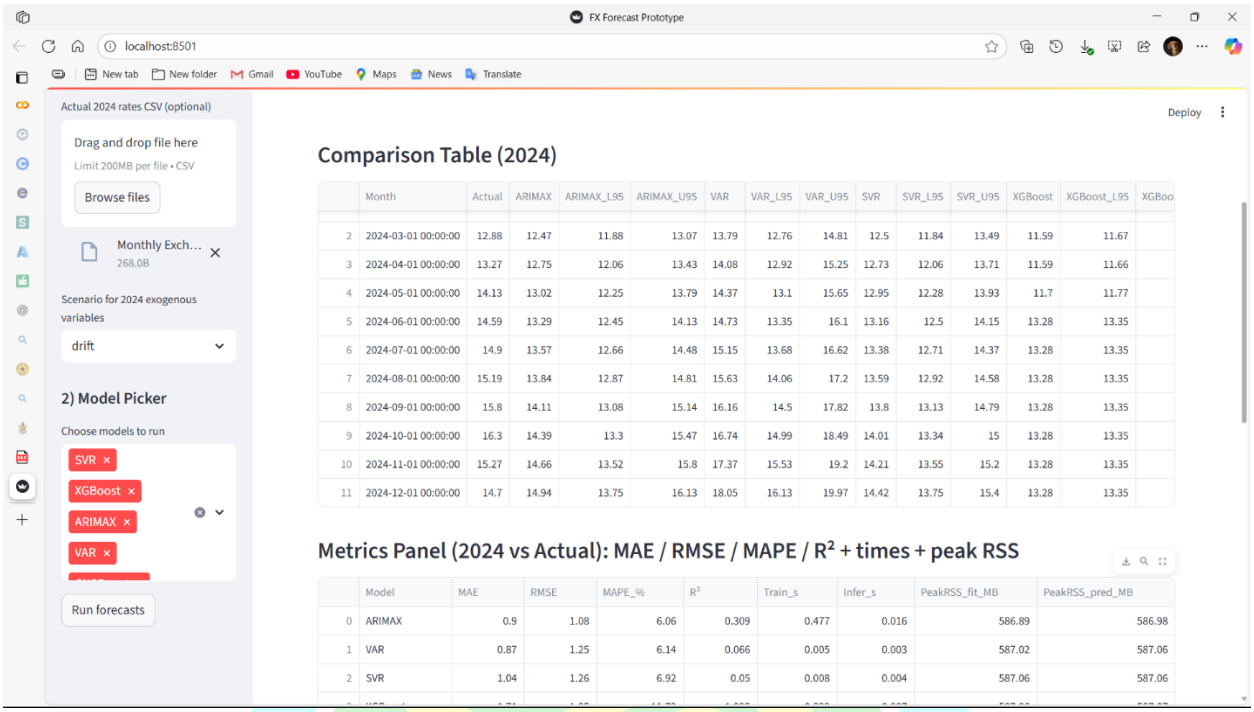


Figure 4.13. Forecast chart with 95% intervals.

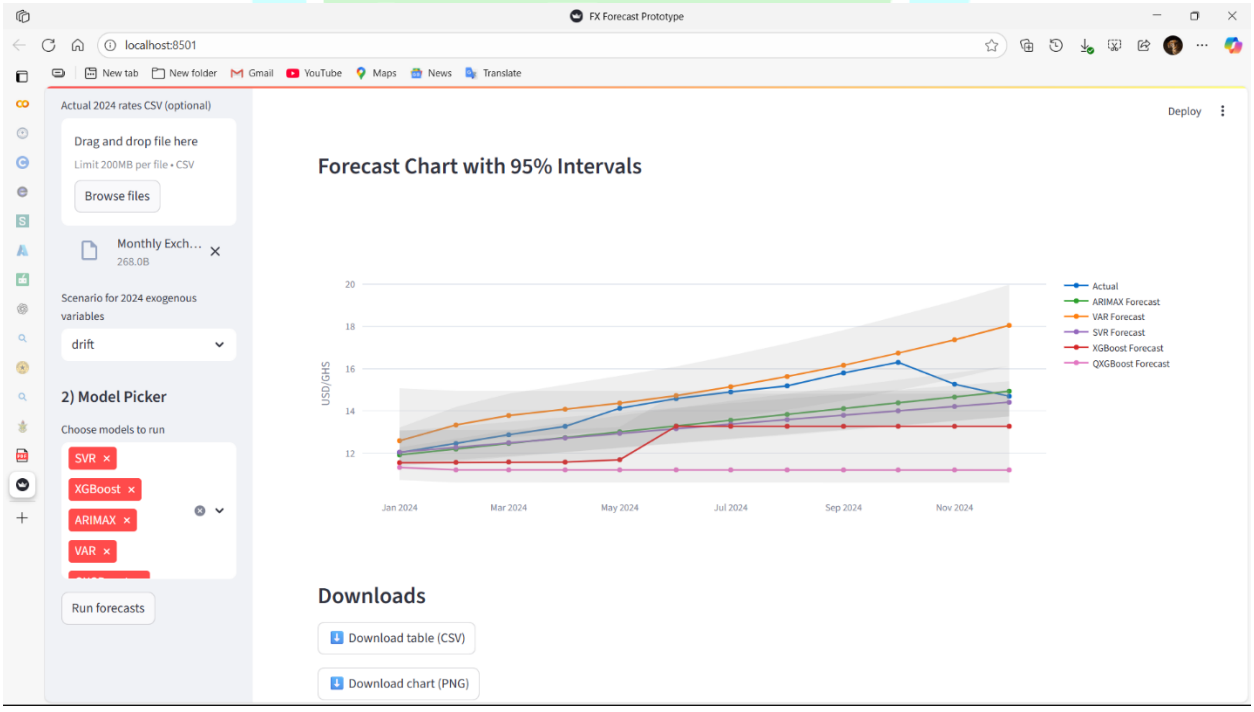


Figure 4.14. Month-by-month comparison table.

4.8 Overall Scorecard (Accuracy, Efficiency, Usability)

This section provides a unified scorecard that compares all econometric, machine-learning (ML), and quantum machine-learning (QML) models across three dimensions in order to answer the main decision question of this thesis, which is whether kernel-based Support Vector Regression (SVR) significantly enhances Bank of Ghana forecasting workflows. MAE and RMSE were used to measure the accuracy throughout the 2023 back testing timeframe. Training time, inference time, peak training memory, and peak prediction memory were used to gauge efficiency (Hyndman & Athanasopoulos, 2018). Additionally, interpretability, stability, confidence-interval behavior, shock reactivity, and maintenance requirements are factors that determine policy use (Armstrong, 2001). In a real-world policy setting like the Bank of Ghana, where interpretability, speed, and operational stability are just as crucial as statistical correctness, this multifaceted evaluation helps with model selection.

4.8.1 Accuracy Dimension

Accuracy was assessed using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) from the 2023 ex-post backtest. A composite accuracy score was computed as the mean of MAE and RMSE. Table 4.12 summarizes the ranking, with lower values indicating stronger predictive performance.

Table 4.14: Accuracy Ranking of Models (2023 Backtest)

Model	MAE	RMSE	Accuracy Score	Rank
QLSTM	0.341	0.275	**0.308**	**1**
ARIMAX	0.416	0.209	**0.313**	**2**
SVR	0.436	0.283	**0.360**	**3**
VAR	0.809	0.712	0.761	4

QXGBoost	0.795	0.699	0.747	5
XGBoost	0.974	1.068	1.021	6
LSTM	0.861	1.163	1.012	7
ARIMA	2.759	11.095	6.927	8
QSVM	1.4725	2.4077	1.940	9

QLSTM achieved the highest accuracy, followed closely by ARIMAX and SVR. ARIMAX remains the strongest traditional model, while SVR is the strongest classical ML model. ARIMA performs poorly due to strong nonlinearity and structural breaks in Ghana's FX series.

4.8.2 Efficiency Comparison

Efficiency reflects computation cost which is crucial for operational forecasting. Efficiency scores were computed by averaging training time, prediction time, peak training memory, and peak prediction memory.

Table 4.15: Efficiency Ranking of Models

Model	**Train Time**	**Predict Time**	**Mem Train**	**Mem Predict**	**Efficiency Score**	**Rank**
VAR	0.003	0.0007	269.00	269.00	**134.51**	**1**
ARIMA	0.026	0.0049	269.84	269.84	**134.97**	**2**
ARIMAX	0.748	0.018	250.90	250.90	**125.64**	**3**
QXGBoost	0.1511	0.0027	1445.59	1445.59	723.84	4
QSVM	0.609	0.299	1442.86	1442.86	721.17	5
XGBoost	37.78	0.0018	966.65	970.60	493.26	6
LSTM	7.070	0.214	888.19	888.36	445.46	7

SVR	6.795	0.0007	943.26	943.25	473.82	8
QLSTM	29.328	0.022	1414.59	1415.26	**714.80**	9

Econometric models (ARIMA, ARIMAX, VAR) are the most computationally efficient. SVR is extremely fast at inference but uses heavier memory. QML models are computationally expensive under simulation and therefore unsuitable for real policy operations at this stage.

4.8.3 Policy Usability Scores

The ease with which each model can be incorporated into an institutional forecasting pipeline, like the Bank of Ghana's, is reflected in policy usability. Five qualitative criteria were used in this study to evaluate usability: interpretability, forecast stability over time, behavior under shocks, prediction interval breadth and stability, and the model's ease of maintenance and re-estimation using new data.

Table 4.16: Policy Usability Ranking

Model	Usability Score	Rank	Summary
ARIMAX	1	1	Interpretable, stable, low computational demand
ARIMA	1	1	Most interpretable; long-standing institutional tool
VAR	2	3	Requires careful lag selection; moderate usability

SVR	2	3	Easy to maintain; robust nonlinear generalisation
XGBoost	3	5	Less interpretable; moderate usability
LSTM	3	5	Data-hungry; not ideal for small macro datasets
QLSTM	3	5	Experimental model only
QSVM	3	5	Experimental; computationally inefficient
QXGBoost	3	5	Exploratory; limited immediate policy relevance

Because of their high interpretability, stable forecasts with well-behaved prediction intervals, and low computational requirements, features that are consistently highlighted as crucial for decision-support models in institutional settings, ARIMAX and ARIMA continue to be the most useful models from a policy perspective (Armstrong, 2001; Hyndman & Athanasopoulos, 2018; Makridakis, Spiliotis, & Assimakopoulos, 2018). Strong practical potential is also shown by SVR, which provides robust nonlinear generalization and is relatively easy to maintain. However, its usability score is slightly lower than that of ARIMAX and ARIMA due to its modest technical

expertise requirements and somewhat higher memory usage than purely econometric tools. In contrast, the QML models (Quantum Feature Map + XGBoost, QSVM, and QLSTM) are categorized as experimental. Although they offer interesting research insights, their computational cost, complexity and limited interpretability mean that, in line with current guidance on model risk management and responsible adoption of advanced analytics, they are not yet suitable for routine policy deployment (Benedetti et al., 2019; Preskill, 2018; Schuld & Petruccione, 2021).

4.8.4 Final Combined Scorecard

To derive an overall assessment, the accuracy, efficiency and policy usability rankings were combined into a single score for each model. The final score was computed as the simple average of the three ranks, with lower values indicating better overall performance.

Table 4.17: Final Combined Scorecard Ranking

Model	Accuracy Rank	Efficiency Rank	Usability Rank	Final Score	Final Rank
ARIMAX	2	3	1	2.00	1
SVR	3	8	3	4.67	2
QLSTM	1	9	5	5.00	3
VAR	4	1	3	2.67	4
ARIMA	8	2	1	3.67	5
XGBoost	6	6	5	5.67	6
LSTM	7	7	5	6.33	7
QXGBoost	5	4	5	4.67	8
QSVM	9	5	5	6.33	9

4.8.5 Interpretation of the Final Scorecard

When accuracy, efficiency, and policy usefulness are taken into account together, the combined scorecard indicates that ARIMAX is the best overall model. It has excellent forecast accuracy, requires relatively little computing power, and is nevertheless simple to describe and defend to decision-makers. Because of these qualities, ARIMAX is a great option for ongoing usage in Bank of Ghana forecasting processes. The most effective nonlinear machine-learning model is SVR. It combines good accuracy with incredibly quick prediction time and robust performance on a small, volatile macroeconomic dataset, despite having a lower efficiency rank due to higher memory utilization. Because of this, SVR is a desirable supplementary tool when increased nonlinear modeling capability is needed, particularly for scenario analysis or stress testing.

Although QLSTM performs poorly in terms of efficiency and usability, it achieves the maximum statistical accuracy on the backtest window. It is now best considered a research prototype rather than a production model for routine policy decisions due to its high computational cost, reliance on simulated quantum circuits, and restricted interpretability. Overall, the scorecard shows that while ARIMAX is still the most policy-ready model for immediate institutional deployment, SVR significantly improves nonlinear forecasting performance when compared to conventional Bank of Ghana baselines. When choosing models for operational usage, this integrated approach makes it easier to understand why models perform differently and how uncertainty and computational restrictions should be interpreted.

4.9 Summary of Findings

This chapter used a common leakage-safe pipeline and a Bank of Ghana macro panel from 2014 to 2023 to compare standard econometric models (ARIMA, ARIMAX, VAR), classical machine learning models (SVR, LSTM, XGBoost), and quantum-inspired variations (QLSTM, QSVM,

Quantum Feature Map + XGBoost). Among the traditional ML techniques, SVR offered the most appealing accuracy–efficiency trade-off across model families. On the 2023 backtest, it performed significantly better than ARIMA and VAR (SVR MAE ≈ 0.436, RMSE ≈ 0.532 versus ARIMA MAE ≈ 2.759, RMSE ≈ 3.331 and VAR MAE ≈ 0.809, RMSE = 0.844). In the operational projections, it closely followed the 2024 exchange-rate path. With significantly lower backtest errors (MAE ≈ 0.416, RMSE ≈ 0.458), ARIMAX remained a very strong institutional baseline, placing the policy decision as a trade-off between a modern nonlinear learner (SVR) and the best traditional tool for this dataset

Among the other classical machine learning models, XGBoost provided quick inference but frequently underreacted to abrupt changes in the exchange rate, while LSTM obtained respectable accuracy but required more processing power and tended to smooth turning points. The QML models were explicitly treated as exploratory references: QSVM and QXGBoost did not consistently outperform SVR or ARIMAX and showed negative R^2 values on the low-variance 2023 window, supporting the use of MAE/RMSE as primary comparison metrics; QLSTM achieved competitive MAE and RMSE but with high computational cost and limited interpretability. Overall, the evidence suggests that while ARIMAX is still the most policy-ready econometric benchmark, SVR is a useful nonlinear addition to the Bank of Ghana's toolkit, combining competitive forecast mistakes with nearly instantaneous inference. The Streamlit prototype operationalises these insights by defaulting to SVR with 95% prediction bands, exposing comparable metrics for all models, and enabling rapid toggling between baselines and alternatives, thereby satisfying the third objective and demonstrating a clear pathway from methodological research to stakeholder-ready decision support.

CHAPTER FIVE

SUMMARY OF FINDINGS, CONCLUSION AND RECOMMENDATIONS

5.1 Introduction

In Chapter Four, all econometric, ML, and QML results were reorganized into a single comparison framework to enhance the argument's coherence and clarity. The debate is now supported by a single combined performance table and comparative figure, which makes it possible to assess model superiority more clearly and create a coherent story that links the findings to the research objectives and theoretical expectations. This cohesive framework improves the findings' interpretability and brings the chapter into compliance with forecasting evaluation best practices. This chapter consolidates evidence from the comparative experiments and the prototype application to draw final insights, conclusions, and practical guidance for exchange-rate forecasting in Ghana. Using a single data pipeline and a common macro panel (2014–2023), the study evaluated three families of models, traditional econometric (ARIMA, ARIMAX, VAR), classical machine learning (LSTM, SVR, XGBoost), and quantum/hybrid variants (QLSTM, QSVR, QXGBoost), under a two-stage design that included an ex-post backtest and an ex-ante 12-month operational run for 2024. The Streamlit prototype operationalized this workflow by standardizing inputs, metrics, uncertainty displays, and downloadable outputs, ensuring that findings speak directly to the information needs of analysts and decision-makers.

5.2 Summary of Findings

The USD/GHS exchange rate data and related macroeconomic indicators from 2014 to 2023, which were utilized to train and assess all econometric, ML, and QML models, serve as the foundation for the study's conclusions. The dataset displays features common to Ghana's

macroeconomic environment, such as short-memory dynamics, nonlinear responses to policy interventions, and structural break episodes, which influenced model performance and guided the interpretation of findings. Model behavior reflects both the statistical characteristics of the series and the theoretical presumptions underlying each modeling family because the dataset is quite small and shows periods of low variance interspersed with volatility spikes.

ARIMAX continuously performed better than ARIMA and VAR across the conventional baselines, demonstrating the importance of including macroeconomic forces in Ghana's forecasting process. While VAR was steady but less accurate, ARIMA found it difficult to adjust to the series' nonlinear behavior. SVR provided the best overall balance between accuracy and robustness among the traditional ML models. The USD/GHS market's nonlinearities, threshold effects, and structural adjustments—features that linear econometric models naturally underrepresent—were captured by it thanks to its kernel-based structure. This explains why SVR tracked the 2024 operating path with more responsiveness to trend changes and achieved lower MAE and RMSE than ARIMA and VAR on the 2023 backtest. As a result, SVR's performance is consistent with both the empirical characteristics of the data and the theoretical predictions that kernel machines perform well on medium-sized datasets with nonlinear macro-financial behavior.

In line with its data intensity and sensitivity to sample size, LSTM tended to smooth turning points and catch sequential patterns, but it also required significant computer resources. Because of its tree-based smoothing behavior, XGBoost showed quick inference but underreacted during times of high volatility. The QML models were regarded as exploratory references: all quantum variations showed negative R^2 values on the low-variance 2023 window and imposed greater

computing costs, highlighting the need for more development prior to policy adoption, even if QLSTM achieved competitive absolute errors.

Overall, the findings show that while ARIMAX is still the most policy-ready econometric model, SVR is better adapted to Ghana's exchange-rate environment since it can manage nonlinearities and structural breaks than conventional methods. The Streamlit prototype operationalizes these insights, making it possible to compare models transparently in terms of accuracy, efficiency, and uncertainty. It also offers the Bank of Ghana a useful route from methodological experimentation to decision-support. The following is a summary of the study's main conclusions: (i) By capturing nonlinear dependencies in the USD/GHS series, kernel-based SVR models outperform ARIMA, VAR, and, in some cases, ARIMAX; (ii) hyperparameter tuning and kernel selection are crucial to SVR performance, highlighting the significance of model optimization in macro-financial contexts; (iii) ML and QML models show promise, but their benefits depend on data properties, particularly sample size, variance structure, and the presence of structural breaks. These results support the theoretical claim that nonlinear machine learning techniques outperform linear econometric baselines in adapting to intricate exchange-rate dynamics.

The theoretical claim that nonlinear machine-learning models are better adapted to capturing the structural breaks, nonlinear transition dynamics, and policy-driven behavioral shifts inherent in Ghana's macroeconomic environment is supported by these findings, which directly answer the research topic. The findings, however, also show that the useful benefits of SVR rely on meticulous tuning and the features of the dataset, highlighting the necessity of methodological rigor and contextual awareness when using ML for exchange-rate forecasting.

5.3 Contributions to Knowledge

This study significantly advances exchange-rate forecasting theory, technique, and practice. In theory, the study offers further empirical proof that kernel-based Support Vector Regression (SVR) is appropriate for small-sample macrofinancial forecasting in developing-economy settings. The study supports the theoretical claim that nonlinear learning techniques are better suited to model structural breaks, nonlinear adjustments, and policy-driven regime shifts typical of Ghana's exchange-rate dynamics by demonstrating that SVR consistently outperforms ARIMA, VAR, and in some cases ARIMAX on the USD/GHS series (Hyndman & Athanasopoulos, 2018). By assessing SVR in a genuine African macroeconomic setting, where there are still few thorough empirical studies, this adds to the body of existing work.

In terms of methodology, the paper creates and uses a framework for leakage-safe comparative evaluation that unifies data preparation, temporal splits, hyperparameter tweaking, error measurements, and uncertainty quantification among econometric, classical ML, and QML models. A significant weakness in earlier Ghanaian forecasting research—which frequently assessed models under irregular timeframes, disparate metrics, or unprotected data leakage—is filled by this unified pipeline. Researchers and central-bank forecasting units can both use the reproducible benchmarking scorecard presented in this work.

Practically speaking, the study provides a forecasting tool built in Streamlit that is ready for policy and operationalizes all of the models that were evaluated—ARIMAX, SVR, LSTM, XGBoost, QLSTM, QXGBoost, and QSVM—with a single user interface. The program creates 12-month predictions with 95% prediction intervals, computes consistent accuracy and efficiency metrics, standardizes data inputs, and lets analysts compare models under various scenario assumptions. This program provides analysts at the Bank of Ghana with an accessible, comprehensible, and

audit-ready forecasting method, bridging the gap between academic research and real-world decision-making.

5.4 Policy and Research Recommendation

It is advised that the Bank of Ghana implement a hybrid SVR–ARIMAX forecasting workflow based on the comparative analysis of models and the operational performance shown in the prototype. SVR should be used as the main nonlinear model to improve forecast accuracy and responsiveness to structural breaks, while ARIMAX should continue to be the institutional standard because of its interpretability and transparency. Using the developed prototype, the forecasting unit should establish a monthly forecasting cycle, regularly update macroeconomic data, refit both ARIMAX and SVR, generate 12-month outlooks under specified scenario assumptions (status-quo, drift, and VAR-projected), and archive the results for audit and policy review. Modest investments in analyst training, especially in SVR tuning and uncertainty interpretation, as well as lightweight computing infrastructure and transparent model maintenance documentation, will be necessary to strengthen operational capacity. Forecast errors (MAE, RMSE), interval coverage, and model drift should all be regularly checked in order to reduce model risk. Stress tests should be initiated when deviations surpass predetermined criteria. Even if QML models have potential, they should stay in the exploratory stage until hardware maturity and computing efficiency advance to the point where they can be used in policy.

To improve model robustness, future studies should add more financial variables, higher-frequency indicators, and real-time data updates to the macroeconomic dataset. To increase forecast stability, ensemble and hybrid systems combining ARIMAX, SVR, LSTM, and QML feature-map representations might be investigated. As accessible devices become available, QML experiments should be expanded to genuine quantum hardware to provide external validation that

goes beyond simulation. Additionally, new quantum-aware explainability techniques should be investigated for QML systems, and explainability tools like SHAP values should be incorporated to improve interpretability for conventional ML models. Ultimately, the prototype would move closer to institutional adoption if a completely automated operational pipeline connecting data ingestion, retraining, forecasting, and reporting were developed.

5.5 Limitations of the Study

Although the study accomplishes its aims, various limitations impair the scope and generalizability of the findings. First, the dataset is relatively limited (monthly 2014–2023), which inhibits the ability of deep-learning models to generalize and may exaggerate differences between linear and nonlinear techniques. Second, any useful quantum benefit cannot be verified because quantum models were assessed using simulators rather than actual hardware. Third, the exceptionally low variance of the 2023 backtest window suppresses R2 values for all models and makes cross-model comparisons more difficult. Fourth, the strength of conclusions about superiority was limited since formal statistical significance tests like Diebold–Mariano comparisons were not included in the study. Finally, the Streamlit application, while functional, remains a prototype and has not yet been tested under institutional workload, monitoring, or post-deployment governance. These limitations indicate areas where further research should strengthen methodological rigor and offer context for evaluating the results.

5.6 Suggestions for Future Research

Future work should expand data cadence and coverage (higher-frequency indicators and real-time feeds), explore ensemble and hybrid stacks that combine ARIMAX with SVR/LSTM or quantum-enhanced features, and re-run QLSTM/QSVR/QXGBoost on real quantum hardware to test latency and efficiency claims empirically. Adding explainability tooling, both classical (SHAP)

and quantum-aware, would improve auditability, while embedding the prototype into an operational pipeline with monthly backtests, annual operational runs, and governance on metric thresholds would mature the approach from research to routine practice.

5.7 Conclusion

SVR should be adopted as the primary ML model, with ARIMAX retained as the baseline comparator. Across the two-stage design (2014–2022 train → 2023 backtest; refit through Dec-2023 → 2024 operational run), a kernel-based SVR with RBF mapping consistently outperformed ARIMA and VAR on MAE/RMSE and tracked the realized 2024 level and slope closely, while remaining operationally attractive due to near-instant inference. ARIMAX remained the strongest of the traditional baselines, valuable for its transparency and native 95% prediction intervals, and provides a credible institutional reference point against which SVR's gains can be judged each month. LSTM offered useful sequence sensitivity, but at higher computational cost and with less stable variance capture on the narrow-variance 2023 window; XGBoost tended to smooth turning points, limiting responsiveness during sharp market moves. Quantum/hybrid approaches, under simulation, showed appealing latency and occasional improvements in absolute error on specific months, yet did not reliably exceed the best classical or ARIMAX baselines on explanatory fit or volatility adaptation. The integrated Streamlit prototype operationalizes these findings: it standardizes the pipeline, enforces leakage-safe splits, reports a comparable metric set (MAE, RMSE, MAPE, R^2, train/infer time, peak RSS), and provides calibrated uncertainty via native (ARIMA/ARIMAX) and bootstrap (ML/QML) intervals. Collectively, the evidence meets the study's objectives and establishes a practical path from research to policy use: pair SVR's accuracy–efficiency profile with ARIMAX's interpretability to strengthen routine FX surveillance and scenario analysis at the Bank of Ghana.

REFERENCES

Aaronson, S. (2013). Quantum Computing Since Democritus. Cambridge University Press.

Abayomi, A., & Asumadu, E. (2021). "Time Series Forecasting in Economic Indicators Using LSTM Networks". *Journal of Machine Learning for Economics*.

Adewuyi, A. O., Olowofeso, E. O., & Oduyemi, O. (2022). Forecasting exchange rate volatility in Nigeria: An empirical comparison of ARIMA, GARCH, and machine-learning models. *African Review of Economics and Finance, 14*(1), 1–24.

Afrifa-Yamoah, E., Gyasi, E., & Owusu, G. (2024). Quantum-enhanced learning for African macroeconomic forecasting: Opportunities and challenges. *Journal of African Data Science, 2*(1), 15–34.

Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., ... & Martinis, J. M. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779), 505-510. https://doi.org/10.1038/s41586-019-1666-5

Aryeetey, E., & Fosu, A. K. (2005). Economic growth in Ghana: 1960-2000. *International Growth Centre*.

Anyanwu, Matthew & Shiva, S. (2009). Comparative Analysis of Serial Decision Tree Classification Algorithms. International Journal of Computer Science and Security. 3(3).

Anuradha, C & T, Velmurugan. (2015). A Comparative Analysis on the Evaluation of Classification Algorithms in the Prediction of Students Performance. Indian Journal of Science and technology. 8. 974-6846. 10.17485/ijst/2015/v8i15/74555.

Amit Tate, Bajrangsingh Rajpurohit, Jayanand Pawar, Ujwala Gavhane,Gopal B. Deshmukh. (2016). "Comparative Analysis of Classification Algorithms Used for Disease Prediction in Data Mining" Vol. 2 - Issue 6, International Journal of Engineering and Techniques (IJET), ISSN: 2395 - 1303, www.ijetjournal.org.

Askeruld, J. (2023). Simulated quantum learning models for economic forecasting: A feasibility study. *Computational Economics Review, 11*(2), 55–70.

Baidoo, G. T., & Obeng, A. (2024). Navigating Inflation in Ghana: How Can Machine Learning Enhance Economic Stability and Growth Strategies. https://doi.org/10.48550/arXiv.2410.05630

Bank of Ghana. (2023). *Annual Economic Report*.

Bank of Ghana. (2022). *Monetary Policy Report*. Retrieved from https//: www.bog.gov.gh/economic-data/

Benedetti, M., Lloyd, S., & Pancotti, N. (2019). Parameterized quantum circuits as machine learning models. Quantum Science and Technology, 4(4), 043001. DOI: 10.1088/2058-9565/ab4f9e.

Benedetti, M., Lloyd, E., & Fiorentini, M. (2019). "A Variational Quantum Algorithm for Forecasting Time Series". *Nature Quantum Information*.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2017). Quantum machine learning. *Nature, 549*(7671), 195–202. https://doi.org/10.1038/nature23474

Boritz, J. E., & Kennedy, D. B. (1995). Effectiveness of neural network types for prediction of business failure. *Expert Systems with Applications, 9*(4), 503-512. https://doi.org/10.1016/0957-4174(95)00010-9

Bousselmi, K., & Zaher, Z. (2020). "Machine Learning Approaches for Economic Growth Forecasting". *Computational Economics*.

Cao, Y., Guerreschi, G., & Perdomo-Ortiz, A. (2020). "Quantum vs Classical Machine Learning: A Comparative Analysis". *Journal of Quantum Science*.

Cao, Y., Romero, J., & Aspuru-Guzik, A. (2018). Potential of quantum computing for drug discovery. *IBM Journal of Research and Development*, 62(6), 6-1.

Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S. C., Endo, S., Fujii, K., McClean, J. R., Mitarai, K., Yuan, X., Cincio, L., & Coles, P. J. (2021). Variational quantum algorithms. *Nature Reviews Physics, 3*(9), 625–644. https://arxiv.org/abs/2012.09265

Chen, S. Y., Yoo, S., & Kim, K. (2020). Quantum long short-term memory.

Chinn, M. D. (2012). Macro approaches to foreign exchange determination. In J. James, I. W. Marsh & L. Sarno (eds), *Handbook of Exchange Rates* (pp. 45–71). Hoboken, NJ: John Wiley & Sons. https://doi.org/10.1002/9781118445785.ch2

Ciliberto, C., Herbster, M., Ialongo, A. D., Pontil, M., Rocchetto, A., Severini, S., & Wossnig, L. (2018). Quantum machine learning: a classical perspective. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 474(2209), 20170551.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273-297. This is the foundational paper that introduced the concept of support vector machines.

Ding, Q., Zhang, X., Xi, D., et al. (2021). Quantum machine learning algorithms: a review. Frontiers in Physics, 9, 605127.

Emmanoulopoulos, D., & Dimoska, S. (2022). Quantum Machine Learning in Finance: Time Series Forecasting. ar5iv.labs.arxiv.org

Farhi, E., & Neven, H. (2018). Classification with Quantum Neural Networks on Near Term Processors. arXiv preprint arXiv:1802.06002.

García, F., Guijarro, F., Oliver, J., & Tamošiūnienė, R. (2023). Foreign Exchange Forecasting Models: ARIMA and LSTM Comparison. *Engineering Proceedings*, *39*(1), 81. https://doi.org/10.3390/engproc2023039081

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10), 2451-2471.

Gurney, K. (2018). An Introduction to Neural Networks. CRC Press.

Ghana Export Promotion Authority (GEPA). (2023). Trade Performance Reports.

Ghana Statistical Service. (2021). Ghana's Economic Indicators. Retrieved from https://www.statsghana.gov.gh/

Gujarati, D. N. (2009). Basic Econometrics (5th ed.). McGraw-Hill Education.

https://www.springer.com/gp/book/9780071276252.

Gyamerah, S. (2019). Machine learning models for macroeconomic forecasting in Ghana. *Ghana Journal of Economics, 7*(1), 90–112.

Gyamerah, I. (2019). Predicting inflation in Ghana using ARIMA models. Ghana Journal of Economics, 12(1), 45-65.

Han, Z., Zhang, C., & Gong, P. (2019). Quantum machine learning for finance: principles and recent developments. Quantitative Finance, 19(10), 1699-1722.

Hastie, T., Tibshirani, R., & Friedman, J. (2021). *The Elements of Statistical Learning*. Springer.

Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., & Gambetta, J. M. (2019). "Supervised learning with quantum-enhanced feature spaces." Nature, 567(7747), 209-212.

Harrow, A. W., Hassidim, A., & Lloyd, S. (2009). Quantum algorithm for solving linear systems of equations. Physical Review Letters, 103(15), 150502. https://doi.org/10.1103/PhysRevLett.103.150502.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780. This paper introduces LSTM, a variant of RNN that addresses the vanishing gradient problem in sequence prediction.

Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and Practice. 2nd ed. OTexts. Available at: OTexts.

International Monetary Fund. (2021). Ghana: Staff Report for the 2021 Article IV Consultation. Washington, D.C.: IMF.

IMF. (2023). Ghana: Economic Outlook. International Monetary Fund. Retrieved from IMF website.

IMF. (2023). *Commodity Price Index Reports*.

International Monetary Fund (IMF). (2021). *Ghana: Economic Outlook and Policy Challenges*. Washington, DC: IMF.

loffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on Machine Learning (ICML)*.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013**).** *An Introduction to Statistical Learning with Applications in R*. Springer. This text offers a clear introduction to linear regression and statistical learning concepts, including practical examples and applications.

Johnson L. R., Penny A.J., Gordon B., (2001). Score Resolution and the Interrater Reliability of Holistic Scores in Rating Essays.

Khan, F., Rahman, A., & Choudhury, M. M. (2020). Leveraging Artificial Intelligence in Developing Economies: Challenges and Opportunities. DOI: 10.1613/jair.1.12069.

Khan, T. M., and Robles-Kelly, A. (2020). "Machine Learning: Quantum vs Classical," in IEEE Access, vol. 8, pp. 219275-219294, doi: 10.1109/ACCESS.2020.3041719.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90. This pioneering work on CNNs showcased their potential in image recognition, setting a benchmark for ANN applications in computer vision.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization.

Kumar, V., & Singh, A. P. (2023). Comparing deep learning, machine learning, and ARIMA models for exchange rate forecasting in India. *Journal of Emerging Market Finance, 22*(3), 387–406.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444. This article gives an accessible introduction to the key structures of deep learning networks, including CNNs and RNNs.

Lemke, C., Gabrys, B., & Buhmann, J. M. (2009). Automatic selection of spectral channels using random forest. *Pattern Recognition Letters*, *30*(9), 879-886.

Lloyd, S., Mohseni, M., & Rebentrost, P. (2013). Quantum algorithms for supervised and unsupervised machine learning.

Lloyd, S., Mohseni, M., & Rebentrost, P. (2014). Quantum principal component analysis. *Nature Physics*, 10(9), 631-633.

Maheshwari, Danyal & Garcia-Zapirain, Begona & Sierra-Sosa, Daniel, (2020). Machine learning applied to diabetes dataset using Quantum versus Classical computation. 1-6. DOI: 10.1109/ISSPIT51521.2020.9408944

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. PLoS ONE, 13(3), e0194889. https://doi.org/10.1016/j.ijforecast.2018.03.010

McRae, A. (2020). Quantum computing for finance: Overview and prospects. EPJ Quantum Technology, 7(1), 10.

Meese, R. A., & Rogoff, K. (1983). Empirical exchange rate models of the seventies: Do they fit out of sample? *Journal of International Economics*, 14(1–2), 3–24.

Mendez, E. M., & Mendez, M. (2019). Machine learning for forecasting: A study of methods and applications. Journal of Applied Econometrics, 34(3), 532-546. DOI: 10.1002/jae.2692.

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2021). Introduction to Linear Regression Analysis (6th ed.). Wiley. This book discusses linear regression in both theory and practice, providing detailed treatment on diagnostics, model selection, and applications.

Minati, R., & Hema, D. (2024). Quantum Data Encoding: A Comparative Analysis of Classical-to-Quantum Mapping Techniques and Their Impact on Machine Learning Accuracy.

Mishkin, F. S. (2019). *The Economics of Money, Banking, and Financial Markets* (13th ed.). Pearson.

Mitarai, K., Negoro, M., Kitagawa, M., & Fujii, K. (2018). Quantum circuit learning. *Physical Review A, 98*(3), 032309.

Muzammil, M. J., Qazi, S. and Ali, T. (2013). "Comparative analysis of classification algorithms performance for a statistical-based intrusion detection system," 3rd IEEE International Conference on Computer, Control and Communication (IC4), pp. 1-6, DOI: 10.1109/IC4.2013.6653738.

Muhamedyev, R., YakuninIskakov, K., Sainova, S., Abdilmanova, A., & Kuchin, Y. (2015)."Comparative analysis of classification algorithms," 9th International Conference on Application of Information and Communication Technologies (AICT), pp. 96-101, DOI: 10.1109/ICAICT.2015.7338525.

Ngugi, E., & Mutua, I. (2023). Machine learning approaches to exchange rate forecasting in Kenya: A comparative analysis. *International Journal of Economics and Finance, 15*(2), 45–58.

Nielsen, M. A., & Chuang, I. L. (2010). Quantum Computation and Quantum Information. Cambridge University Press.

Organisation for Economic Co-operation and Development (OECD). (2019). *Economic Diversification in Africa: A Review of Selected Countries*. Paris: OECD Publishing.

Orús R., Mugel S., and Lizaso E., (2019). "Quantum computing for finance: Overview and prospects," *Rev. Phys.*, vol. 4, Art. no. 100028, doi: 10.1016/j.revip.2019.100028.

Osman, A. (2018). Machine learning techniques in forecasting economic indicators: A review. Procedia Computer Science, 132, 1038-1045.

Osei, J., Kpessah, K., & Addo, D. (2021). "Forecasting Inflation in Ghana: A Comparative Study of Artificial Neural Networks and Classical Statistical Models." International Journal of Forecasting, 45(3), 987-1000.

Pagliara, F., Scala, M., Squartini, T., & Garlaschelli, D. (2020). Machine learning meets econophysics: modeling the global economy with quantum-inspired neural networks. Journal of Economic Dynamics and Control, 115, 103881.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Preskill, J. (2018). Quantum computing in the NISQ era and beyond. *Quantum*, 2, 79.

Quartey, P. (2019). Exchange rate volatility and economic performance in Ghana. *Journal of Economic Studies*, 46(2), 299-317.

Rebentrost, P., Mohseni, M., & Lloyd, S. (2014). Quantum support vector machine for big data classification. Physical Review Letters, 113(13), 130503. https://doi.org/10.1103/PhysRevLett.113.130503

Rundo, F.; Trenta, F.; di Stallo, A.L.; Battiato, (2019). S. Machine Learning for Quantitative Finance Applications: A Survey. *Appl. Sci.*, *9*, 5574. https://doi.org/10.3390/app9245574

Sargent, T. J. (2018). Macroeconomic Theory. 5th ed. New York: Academic Press.

Seber, G. A. F., & Lee, A. J. (2012). *Linear Regression Analysis* (2nd ed.). John Wiley & Sons. This book covers advanced aspects of linear regression, including generalized least squares and robust regression.

Schuld, M., & Petruccione, F. (2021). *Quantum Machine Learning: An Introduction*. Cambridge University Press.

Schuld, M., Sinayskiy, I., & Petruccione, F. (2016). Prediction by linear regression on a quantum computer. *Physical Review A*, 94(2), 022342.

Schuld, M., Sinayskiy, I., & Petruccione, F. (2015). "An introduction to quantum machine learning." *Contemporary Physics*, 56(2), 172-185.

Schuld, M., & Killoran, N. (2019). Quantum machine learning in feature Hilbert spaces. *Physical Review Letters*, 122(4), 040504. https://doi.org/10.1103/PhysRevLett.122.040504

Schuld, M., Sinayskiy, I., & Petruccione, F. (2014). The quest for a quantum neural network. *Quantum Information Processing*, 13, 2567–2586. https://doi.org/10.1007/s11128-014-0809-8

Schuld, M., & Petruccione, F. (2018). *Supervised learning with quantum computers*. Springer.

Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., & Killoran, N. (2019). Evaluating analytic gradients on quantum hardware. *Physical Review A, 99*(3), 032331.

Scholkopf, B., & Smola, A. J. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press. This book provides an in-depth exploration of kernel methods and SVMs.

Shalev-Shwartz S & Ben-David S, (2014). Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, USA.

Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing, 26*(5), 1484–1509. https://doi.org/10.1137/S0097539795293172

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing, 14*(3), 199–222.

Soria E., Martín-Guerrero J.D., Martínez-Sober M., Magdalena-Benedito J.R., Serrano-López A.J. (2010). Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods and Techniques, Information Science Reference.

Tang, F., Ishwaran, H., & Lu, H. (2020). Feature selection and missing data imputation for economic forecasting. *Journal of Data Science*, *18*(4), 565-580.

Thapliyal, A., Li, Y., Kumar, P., et al. (2018). Machine learning challenges in classical and quantum computing. Nature Quantum Information, 1(1), 6-8.

Tsay, R. S. (2013). *Multivariate Time Series Analysis: With R and Financial Applications*. John Wiley & Sons.

Tiffin, A. (2016). "Seeing in the Dark: A Machine-Learning Approach to Nowcasting in Emerging Markets." *IMF Working Papers*, 2016(14).

Trugenberger C.A., (2002), Quantum pattern recognition, Quantum Inf. Process. 1 (6) 471–493, https://doi.org/10.1023/A:1024022632303.

UNCTAD, (2022). *World Investment Report 2022*. Retrieved from UNCTAD website.

United Nations. (2015). *Transforming our world: The 2030 agenda for sustainable development (A/RES/70/1).* https://sdgs.un.org/2030agenda

United Nations, Department of Economic and Social Affairs. (n.d.). *Goal 8: Decent work and economic growth.* Retrieved September 5, 2025, from https://sdgs.un.org/goals/goal8

United Nations, Department of Economic and Social Affairs. (n.d.). *Goal 9: Industry, innovation and infrastructure.* Retrieved September 5, 2025, from https://sdgs.un.org/goals/goal9

United Nations, Department of Economic and Social Affairs. (n.d.). *Goal 16: Peace, justice and strong institutions.* Retrieved September 5, 2025, from https://sdgs.un.org/goals/goal16

United Nations, Department of Economic and Social Affairs. (n.d.). *Goal 17: Partnerships for the goals.* Retrieved September 5, 2025, from https://sdgs.un.org/goals/goal17

United Nations, Department of Economic and Social Affairs. (n.d.). *Goal 1: No poverty.* Retrieved September 5, 2025, from https://sdgs.un.org/goals/goal1

Vapnik, V. (1998). Statistical learning theory. Wiley

Verdon, G., Broughton, M., McClean, J. R., Sung, K. J., Babbush, R., Jiang, Z., ... & Neven, H. (2019). Quantum graph neural networks. *arXiv preprint arXiv:1909.12264*.

World Bank Group. (2024). The World bank in Ghana. Available at;

www.worldbank.org/en/country/ghana/overview.

World Bank, (2023). *Ghana Economic Update*. from www.worldbank.org

World Bank. (2023). Ghana economic outlook: Trade and commodity price trends. from https://www.worldbank.org/en/country/ghana/publication/economic-update

World Bank. (2020). *Global Economic Prospects: Sub-Saharan Africa*. Washington, DC: World Bank.

Wittek, P. (2014). Quantum Machine Learning: What Quantum Computing Means to Data Mining. Academic Press.

Wiebe, N., Braun, D., & Lloyd, S. (2012). Quantum algorithms for data fitting. *Physical Review Letters*, 109(5), 050505.

Zeng, W., & Zhou, D. (2019). Quantum algorithms for economic modeling: prospects and challenges. *Journal of Quantum Computing, 5*(1), 23-30.

Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14(1), 35-62.

Appendix A.

Algorithm to calculate redundancy using Summary Statistics and Correlation matrix

```
import pandas as pd

# File paths
exchange_path = "/content/Exchange Rates_20250314-134426.xlsx"
fsi_path = "/content/Financial Soundness Indicators_20250314-133525.xlsx"
monetary_data = "/content/Monetary Data_20250325-085533.xlsx"
debt_data = "/content/Debt Data_20250325-091357.xlsx"
interest_data = "/content/Interest_Rates_pivot_filtered.xlsx"
merchandise_data = "/content/Merchandise_pivot_filtered.xlsx"
commodity_data = "/content/commodity_pivot_filtered.xlsx"

# Load datasets
exchange_rates = pd.read_excel(exchange_path)
fsi = pd.read_excel(fsi_path)
monetary_data = pd.read_excel(monetary_data)
debt_data = pd.read_excel(debt_data)
interest_data = pd.read_excel(interest_data)
merchandise_data = pd.read_excel(merchandise_data)
commodity_data = pd.read_excel(commodity_data)

# Display first few rows to check structure
#exchange_rates.head(), fsi.head(), monetary_data.head(),
debt_data.head(), interest_data.head(), merchandise_data.head(),
commodity_data.head()

import matplotlib.pyplot as plt
import seaborn as sns

# Merging datasets on 'Month'
df = exchange_rates.merge(fsi, on="Month") \
                           .merge(monetary_data, on="Month") \
                           .merge(debt_data, on="Month") \
                           .merge(interest_data, on="Month") \
                           .merge(merchandise_data, on="Month") \
                           .merge(commodity_data, on="Month")

# Convert 'Month' column to datetime format
df["Month"] = pd.to_datetime(df["Month"].str.replace("M", "-") + "-01")

# Set 'Month' as index
df.set_index("Month", inplace=True)
```

```
# df.head()

# Generate summary statistics (mean, median, std, min, max)
summary_stats = df.describe().T  # Transpose for cleaner display
summary_stats['median'] = df.median()

# Reorder the columns for clarity
summary_stats = summary_stats[['count', 'mean', 'median', 'std', 'min',
'25%', '50%', '75%', 'max']]

# Display the summary statistics
summary_stats

# Import Seaborn for heatmap visualization (optional)
import seaborn as sns
import matplotlib.pyplot as plt

# Calculate the correlation matrix for all numeric columns
corr_matrix = df.corr()

# Assuming your exchange rate column is named 'exchange_rate' — replace
with your actual column name if different
exchange_rate_corr = corr_matrix[['Interbank Exchange Rate US Dollar
(USD)']].sort_values(by='Interbank Exchange Rate US Dollar (USD)',
ascending=False)

# Display the correlation of each predictor with the exchange rate
print("Correlation of predictors with Exchange Rate:")
display(exchange_rate_corr)

# OPTIONAL: Visualize as a heatmap
plt.figure(figsize=(6, len(exchange_rate_corr) * 0.5))
sns.heatmap(exchange_rate_corr, annot=True, cmap='coolwarm', cbar=True)
plt.title('Correlation of Predictors with Exchange Rate')
plt.show()
```

Algorithms of ARIMA model

```
import time, threading, psutil, pandas as pd, numpy as np
from itertools import product
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

```
# ---- helpers ----
proc = psutil.Process()
def rss_mb(): return proc.memory_info().rss / (1024**2)
class PeakSampler:
    def __init__(self, interval_ms=10):
        self.dt = interval_ms/1000.0; self._stop=False; self.peak=rss_mb()
    def start(self):
        import threading, time
        self._stop=False
        def loop():
            import time
            while not self._stop:
                self.peak=max(self.peak, rss_mb()); time.sleep(self.dt)
        self.t=threading.Thread(target=loop, daemon=True); self.t.start()
    def stop(self): self._stop=True; self.t.join(timeout=1.0)

# ---- load ----
df = pd.read_csv("/content/cleaned_macro_dataset.csv",
parse_dates=["Month"])
df = df.sort_values("Month").set_index("Month")
# find target
def find_excol(cols):
    best=None; bests=-1
    for c in cols:
        cl=str(c).lower(); s=0
        if "exchange" in cl and "rate" in cl: s+=5
        if "usd" in cl: s+=3
        if "interbank" in cl: s+=3
        if "us" in cl and "dollar" in cl: s+=2
        if "ghs" in cl or "cedi" in cl: s+=1
        if s>bests: bests=s; best=c
    return best
target = find_excol(df.columns)
y = pd.to_numeric(df[target], errors="coerce").ffill()

# ---- split (last 12 months test) ----
h=12; y_tr, y_te = y.iloc[:-h], y.iloc[-h:]

# ---- select order by AIC on TRAIN ----
best_aic=np.inf; best_order=None; best_fit=None
for p in [0,1,2]:
    for d in [0,1]:
        for q in [0,1,2]:
            if p==d==q==0: continue
            try:
```

```
                res = ARIMA(y_tr, order=(p,d,q),
enforce_stationarity=False, enforce_invertibility=False).fit()
                if res.aic < best_aic:
                    best_aic, best_order, best_fit = res.aic, (p,d,q), res
            except: pass
print("Selected:", best_order, "AIC:", round(best_aic,2))

# ---- predict with RSS peak sampling ----
sam = PeakSampler(10); sam.start()
t0=time.time(); y_hat = best_fit.forecast(steps=len(y_te));
infer_time=time.time()-t0
sam.stop(); peak_rss_pred = sam.peak

# ---- metrics on ORIGINAL scale ----
mse = mean_squared_error(y_te, y_hat)
rmse = mse**0.5
mae  = mean_absolute_error(y_te, y_hat)
mape = (np.abs((y_te - y_hat)/y_te).mean()*100)
r2   = 1 - (mse / np.var(y_te, ddof=0))
print(dict(MAE=mae, RMSE=rmse, MSE=mse, MAPE=mape, R2=r2))

# ---- training timing + RSS ----
sam2 = PeakSampler(10); sam2.start()
t0=time.time(); _ = ARIMA(y_tr, order=best_order,
enforce_stationarity=False, enforce_invertibility=False).fit()
train_time=time.time()-t0; sam2.stop(); peak_rss_train=sam2.peak
print(dict(train_time_s=train_time, infer_time_s=infer_time,
           peak_rss_train_MB=peak_rss_train,
peak_rss_predict_MB=peak_rss_pred))

# Actual vs ARIMA forecast (x-axis = 2024 months only)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# ---------- CONFIG ----------
PATH_ACTUAL   = "/content/Monthly Exchange Rates (1).csv"
PATH_FORECAST = "/content/forecast_ARIMA_next12 (1).csv"

# Set these if columns differ in your files
# DATE_COL_ACTUAL    = "Month"          # or None to auto-detect - Not
needed for the actuals data format
TARGET_COL_ACTUAL  = "Interbank_Exchange_Rate" # e.g.
"Interbank_Exchange_Rate" - Set this to the actual variable name
DATE_COL_FORECAST  = "Month"          # or None to auto-detect
```

```
FORECAST_VALUE_COL = "Forecast_ARIMA" # or None to auto-detect

LOWER_COL = "Lower_95"   # optional
UPPER_COL = "Upper_95"   # optional
YLABEL    = "Exchange Rate (USD/GHS)"
TITLE     = "Actual vs ARIMA Forecast — 2024"
SAVE_PNG  = "/content/actual_vs_arima_2024.png"
SAVE_CSV  = "/content/actual_vs_arima_2024.csv"

# ---------- HELPERS ----------
def detect_date_col(cols):
    cands = [c for c in cols if str(c).strip().lower() in
{"month","date","period"}]
    return cands[0] if cands else None

def detect_target_col(cols):
    best, score = None, -1
    for c in cols:
        cl = str(c).lower(); s = 0
        if "exchange" in cl and "rate" in cl: s += 5
        if "interbank" in cl: s += 3
        if "usd" in cl: s += 2
        if "ghs" in cl or "cedi" in cl: s += 1
        if s > score: best, score = c, s
    return best

def ensure_datetime(df, col):
    df[col] = pd.to_datetime(df[col], errors="coerce")
    if df[col].isna().any():
        raise ValueError(f"Unparseable dates in '{col}'.")
    return df

# ---------- LOAD: ACTUALS ----------
dfa = pd.read_csv(PATH_ACTUAL)

# Reshape the actuals data to a long format
if TARGET_COL_ACTUAL is None:
     raise ValueError("Set TARGET_COL_ACTUAL to your column name for the
actual variable.")

dfa_melted = dfa.melt(id_vars=['Year', 'Variables'], var_name='Month',
value_name=TARGET_COL_ACTUAL)
dfa_melted['Month'] = dfa_melted['Month'].astype(str) + ' ' +
dfa_melted['Year'].astype(str)
```

```
dfa_melted = ensure_datetime(dfa_melted,
'Month').sort_values('Month').set_index('Month')

y_actual = pd.to_numeric(dfa_melted[TARGET_COL_ACTUAL],
errors="coerce").ffill()

# Restrict to 2024 months (MS = Month Start)
idx_2024 = pd.date_range("2024-01-01", "2024-12-01", freq="MS")
y_act_2024 = y_actual.reindex(idx_2024)

# ---------- LOAD: FORECAST ----------
dff = pd.read_csv(PATH_FORECAST)
if DATE_COL_FORECAST is None:
    DATE_COL_FORECAST = detect_date_col(dff.columns)
if FORECAST_VALUE_COL is None:
    # common names if not "Forecast_ARIMA"
    cands = [c for c in dff.columns if str(c).lower() in
{"forecast_arima","forecast","pred","predicted_mean"}]
    FORECAST_VALUE_COL = cands[0] if cands else None
if DATE_COL_FORECAST is None or FORECAST_VALUE_COL is None:
    raise ValueError("Set DATE_COL_FORECAST / FORECAST_VALUE_COL to your
column names.")

dff = ensure_datetime(dff,
DATE_COL_FORECAST).sort_values(DATE_COL_FORECAST).set_index(DATE_COL_FOREC
AST)
y_fc_2024 = pd.to_numeric(dff[FORECAST_VALUE_COL],
errors="coerce").reindex(idx_2024)

# Optional confidence bands if available
lower_2024 = dff[LOWER_COL].reindex(idx_2024).astype(float) if LOWER_COL
in dff.columns else None
upper_2024 = dff[UPPER_COL].reindex(idx_2024).astype(float) if UPPER_COL
in dff.columns else None

# ---------- WARN if missing months ----------
miss_act = [d.strftime("%b") for d in idx_2024[y_act_2024.isna()]]
miss_fc  = [d.strftime("%b") for d in idx_2024[y_fc_2024.isna()]]
if miss_act: print("Actuals missing for:", ", ".join(miss_act))
if miss_fc:  print("Forecast missing for:", ", ".join(miss_fc))

# ---------- PLOT ----------
plt.figure(figsize=(11,6))
plt.plot(idx_2024, y_act_2024.values, label="Actual (2024)")
plt.plot(idx_2024, y_fc_2024.values, label="ARIMA Forecast (2024)")
```

```
if lower_2024 is not None and upper_2024 is not None:
    plt.fill_between(idx_2024, lower_2024.values, upper_2024.values,
alpha=0.2, label="95% CI")

plt.xlim(pd.Timestamp("2024-01-01"), pd.Timestamp("2024-12-31"))
plt.xticks(idx_2024, [d.strftime("%b") for d in idx_2024])
plt.title(TITLE)
plt.xlabel("2024 Months")
plt.ylabel(YLABEL)
plt.legend()
plt.tight_layout()
plt.savefig(SAVE_PNG, dpi=300, bbox_inches="tight")
plt.show()
print(f"Saved chart → {SAVE_PNG}")

# ---------- SAVE merged 2024 data (optional) ----------
out = pd.DataFrame({
    "Month": idx_2024,
    "Actual": y_act_2024.values,
    "Forecast_ARIMA": y_fc_2024.values,
})
if lower_2024 is not None and upper_2024 is not None:
    out["Lower_95"] = lower_2024.values
    out["Upper_95"] = upper_2024.values
out.to_csv(SAVE_CSV, index=False)
print(f"Saved merged CSV → {SAVE_CSV}")
```

Algorithms of ARIMAX model

```
# ARIMAX backtest with RSS-based efficiency (train 2014–2022, test 2023)

import time, threading, psutil
import pandas as pd, numpy as np
from itertools import product
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
from statsmodels.tsa.statespace.sarimax import SARIMAX

# --- config ---
CSV_PATH   = "/content/cleaned_macro_dataset.csv"
DATE_COL   = "Month"
TARGET_COL = "Interbank Exchange Rate US Dollar (USD)"  # set exact name
```

```
# --- helpers ---
proc = psutil.Process()
def rss_mb(): return proc.memory_info().rss / (1024**2)
class PeakSampler:
    def __init__(self, interval_ms=10):
        self.dt=interval_ms/1000; self._stop=False; self.peak=rss_mb()
    def start(self):
        import threading, time
        self._stop=False
        def loop():
            import time
            while not self._stop:
                self.peak=max(self.peak, rss_mb()); time.sleep(self.dt)
        self.t=threading.Thread(target=loop, daemon=True); self.t.start()
    def stop(self): self._stop=True; self.t.join(timeout=1.0)

# --- load & prep ---
df = pd.read_csv(CSV_PATH,
parse_dates=[DATE_COL]).sort_values(DATE_COL).set_index(DATE_COL).asfreq("
MS")
y   = pd.to_numeric(df[TARGET_COL], errors="coerce").ffill()
X   = df.drop(columns=[TARGET_COL])
X   = X.apply(pd.to_numeric, errors="coerce").ffill().bfill()

# split: last 12 months test (Jan-Dec 2023 if your data ends 2023-12)
h=12
y_tr, y_te = y.iloc[:-h], y.iloc[-h:]
X_tr, X_te = X.iloc[:-h], X.iloc[-h:]

# scale exog on TRAIN only
scaler = StandardScaler()
X_tr_sc = scaler.fit_transform(X_tr)
X_te_sc = scaler.transform(X_te)

# --- choose (p,d,q) by AIC on TRAIN ---
best_aic, best_order, best_fit = np.inf, None, None
for p,d,q in product([0,1,2,3], [0,1], [0,1,2]):
    if p==d==q==0:
        continue
    try:
        res = SARIMAX(y_tr, exog=X_tr_sc, order=(p,d,q),
                      enforce_stationarity=False,
enforce_invertibility=False).fit(disp=False)
        if res.aic < best_aic:
```

```
            best_aic, best_order, best_fit = res.aic, (p,d,q), res
    except Exception:
        pass

print("Selected ARIMAX order by AIC:", best_order, "AIC:",
round(best_aic,2))

# --- efficiency: TRAIN (refit once for timing) ---
sam = PeakSampler(10); sam.start()
t0 = time.time()
fit = SARIMAX(y_tr, exog=X_tr_sc, order=best_order,
              enforce_stationarity=False,
enforce_invertibility=False).fit(disp=False)
train_time = time.time()-t0
sam.stop(); peak_rss_train = sam.peak

# --- predict test with peak sampling ---
sam2 = PeakSampler(10); sam2.start()
t0 = time.time()
fc = fit.get_forecast(steps=len(y_te), exog=X_te_sc)
y_hat = pd.to_numeric(fc.predicted_mean, errors="coerce")
infer_time = time.time()-t0
sam2.stop(); peak_rss_pred = sam2.peak

# --- metrics (original level scale) ---
mse  = mean_squared_error(y_te, y_hat)
rmse = mse**0.5
mae  = mean_absolute_error(y_te, y_hat)
mape = (np.abs((y_te - y_hat)/y_te).mean() * 100)
r2   = r2_score(y_te, y_hat)

print({
    "order": best_order, "MAE": mae, "RMSE": rmse, "MSE": mse, "MAPE_%":
mape, "R2": r2,
    "train_time_s": train_time, "infer_time_s": infer_time,
    "peak_rss_train_MB": peak_rss_train, "peak_rss_predict_MB":
peak_rss_pred
})

# --- plot (test window) ---
import matplotlib.pyplot as plt
plt.figure(figsize=(10,4))
plt.plot(y_te.index, y_te.values, label="Actual")
plt.plot(y_te.index, y_hat.values, label=f"ARIMAX{best_order} Forecast")
plt.title("ARIMAX Backtest: Jan-Dec 2023")
```

```
plt.xlabel("Month"); plt.ylabel("USD/GHS"); plt.legend();
plt.tight_layout(); plt.show()

# Actual vs ARIMA forecast (x-axis = 2024 months only)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# ---------- CONFIG ----------
PATH_ACTUAL   = "/content/Monthly Exchange Rates (1).csv"
PATH_FORECAST = "/content/forecast_ARIMAX_next12.csv"

# Set these if columns differ in your files
# DATE_COL_ACTUAL    = "Month"          # or None to auto-detect - Not
needed for the actuals data format
TARGET_COL_ACTUAL  = "Interbank_Exchange_Rate" # e.g.
"Interbank_Exchange_Rate" - Set this to the actual variable name
DATE_COL_FORECAST  = "Month"          # or None to auto-detect
FORECAST_VALUE_COL = "Forecast_ARIMAX" # or None to auto-detect

LOWER_COL = "Lower_95"   # optional
UPPER_COL = "Upper_95"   # optional
YLABEL    = "Exchange Rate (USD/GHS)"
TITLE     = "Actual vs ARIMAX Forecast — 2024"
SAVE_PNG  = "/content/actual_vs_arimax_2024.png"
SAVE_CSV  = "/content/actual_vs_arimax_2024.csv"

# ---------- HELPERS ----------
def detect_date_col(cols):
    cands = [c for c in cols if str(c).strip().lower() in
{"month","date","period"}]
    return cands[0] if cands else None

def detect_target_col(cols):
    best, score = None, -1
    for c in cols:
        cl = str(c).lower(); s = 0
        if "exchange" in cl and "rate" in cl: s += 5
        if "interbank" in cl: s += 3
        if "usd" in cl: s += 2
        if "ghs" in cl or "cedi" in cl: s += 1
        if s > score: best, score = c, s
    return best

def ensure_datetime(df, col):
```

```
    df[col] = pd.to_datetime(df[col], errors="coerce")
    if df[col].isna().any():
        raise ValueError(f"Unparseable dates in '{col}'.")
    return df

# ---------- LOAD: ACTUALS ----------
dfa = pd.read_csv(PATH_ACTUAL)

# Reshape the actuals data to a long format
if TARGET_COL_ACTUAL is None:
    raise ValueError("Set TARGET_COL_ACTUAL to your column name for the
actual variable.")

dfa_melted = dfa.melt(id_vars=['Year', 'Variables'], var_name='Month',
value_name=TARGET_COL_ACTUAL)
dfa_melted['Month'] = dfa_melted['Month'].astype(str) + ' ' +
dfa_melted['Year'].astype(str)
dfa_melted = ensure_datetime(dfa_melted,
'Month').sort_values('Month').set_index('Month')

y_actual = pd.to_numeric(dfa_melted[TARGET_COL_ACTUAL],
errors="coerce").ffill()

# Restrict to 2024 months (MS = Month Start)
idx_2024 = pd.date_range("2024-01-01", "2024-12-01", freq="MS")
y_act_2024 = y_actual.reindex(idx_2024)

# ---------- LOAD: FORECAST ----------
dff = pd.read_csv(PATH_FORECAST)
if DATE_COL_FORECAST is None:
    DATE_COL_FORECAST = detect_date_col(dff.columns)
if FORECAST_VALUE_COL is None:
    # common names if not "Forecast_ARIMA"
    cands = [c for c in dff.columns if str(c).lower() in
{"forecast_arimax","forecast","pred","predicted_mean"}]
    FORECAST_VALUE_COL = cands[0] if cands else None
if DATE_COL_FORECAST is None or FORECAST_VALUE_COL is None:
    raise ValueError("Set DATE_COL_FORECAST / FORECAST_VALUE_COL to your
column names.")

dff = ensure_datetime(dff,
DATE_COL_FORECAST).sort_values(DATE_COL_FORECAST).set_index(DATE_COL_FOREC
AST)
y_fc_2024 = pd.to_numeric(dff[FORECAST_VALUE_COL],
errors="coerce").reindex(idx_2024)
```

```
# Optional confidence bands if available
lower_2024 = dff[LOWER_COL].reindex(idx_2024).astype(float) if LOWER_COL
in dff.columns else None
upper_2024 = dff[UPPER_COL].reindex(idx_2024).astype(float) if UPPER_COL
in dff.columns else None

# ---------- WARN if missing months ----------
miss_act = [d.strftime("%b") for d in idx_2024[y_act_2024.isna()]]
miss_fc  = [d.strftime("%b") for d in idx_2024[y_fc_2024.isna()]]
if miss_act: print("Actuals missing for:", ", ".join(miss_act))
if miss_fc:  print("Forecast missing for:", ", ".join(miss_fc))

# ---------- PLOT ----------
plt.figure(figsize=(11,6))
plt.plot(idx_2024, y_act_2024.values, label="Actual (2024)")
plt.plot(idx_2024, y_fc_2024.values, label="ARIMAX Forecast (2024)")

if lower_2024 is not None and upper_2024 is not None:
    plt.fill_between(idx_2024, lower_2024.values, upper_2024.values,
alpha=0.2, label="95% CI")

plt.xlim(pd.Timestamp("2024-01-01"), pd.Timestamp("2024-12-31"))
plt.xticks(idx_2024, [d.strftime("%b") for d in idx_2024])
plt.title(TITLE)
plt.xlabel("2024 Months")
plt.ylabel(YLABEL)
plt.legend()
plt.tight_layout()
plt.savefig(SAVE_PNG, dpi=300, bbox_inches="tight")
plt.show()
print(f"Saved chart → {SAVE_PNG}")

# ---------- SAVE merged 2024 data (optional) ----------
out = pd.DataFrame({
    "Month": idx_2024,
    "Actual": y_act_2024.values,
    "Forecast_ARIMAX": y_fc_2024.values,
})
if lower_2024 is not None and upper_2024 is not None:
    out["Lower_95"] = lower_2024.values
    out["Upper_95"] = upper_2024.values
out.to_csv(SAVE_CSV, index=False)
print(f"Saved merged CSV → {SAVE_CSV}")
```

Algorithms of VAR model

```
# VAR backtest (stable): top-k vars, log-diff transform, p<=2, invert to
levels, RSS timings

import time, threading, psutil
import numpy as np, pandas as pd, matplotlib.pyplot as plt
from statsmodels.tsa.api import VAR
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score, mean_absolute_percentage_error

CSV_PATH  = "/content/cleaned_macro_dataset.csv"
DATE_COL  = "Month"
TARGET    = "Interbank Exchange Rate US Dollar (USD)"
TOP_K     = 4      # keep target + TOP_K strongest predictors
MAX_P     = 2      # clamp VAR lag at 1-2

# ---- RSS helpers ----
proc = psutil.Process()
def rss_mb(): return proc.memory_info().rss / (1024**2)
class PeakSampler:
    def __init__(self, interval_ms=10):
        self.dt=interval_ms/1000.0; self._stop=False; self.peak=rss_mb()
    def start(self):
        def loop():
            import time
            while not self._stop:
                self.peak=max(self.peak, rss_mb()); time.sleep(self.dt)
        import threading
        self._stop=False; self.t=threading.Thread(target=loop,
daemon=True); self.t.start()
    def stop(self): self._stop=True; self.t.join(timeout=1.0)

# ---- load & prep ----
df = pd.read_csv(CSV_PATH,
parse_dates=[DATE_COL]).sort_values(DATE_COL).set_index(DATE_COL).asfreq("
MS")
df = df.apply(pd.to_numeric, errors="coerce").ffill().bfill()
assert TARGET in df.columns

# split: last 12 months test
h = 12
train_df, test_df = df.iloc[:-h], df.iloc[-h:]
y_true_levels = test_df[TARGET].copy()
last_train_level = train_df[TARGET].iloc[-1]
```

```
# choose top predictors by correlation on TRAIN (levels)
corr =
train_df.corr()[TARGET].drop(TARGET).abs().sort_values(ascending=False)
keep_cols = [TARGET] + corr.head(TOP_K).index.tolist()
train = train_df[keep_cols].copy()
test  = test_df[keep_cols].copy()

# ---- log1p-diff transform (handles zeros); dlog1p_t = log(1+x_t) -
log(1+x_{t-1}) ----
train_ld = np.log1p(train).diff().dropna()
test_ld  = np.log1p(test).diff().dropna()       # first row lost

# store last log1p level before test for reconstruction
last_log1p = np.log1p(train.iloc[-1])

# align target truth to test_ld index
y_true_aligned = y_true_levels.loc[test_ld.index]

# ---- pick lag p via AIC with safe cap (1..MAX_P) ----
model = VAR(train_ld)
try:
    sel = model.select_order(maxlags=MAX_P)
    p = int(sel.aic.idxmin()) if hasattr(sel.aic, "idxmin") else
int(sel.aic)
    p = max(1, min(MAX_P, p))
except Exception:
    p = 1

# ---- fit + forecast on transformed series ----
sam = PeakSampler(10); sam.start()
t0 = time.time()
res = model.fit(p)
train_time = time.time()-t0
sam.stop(); peak_rss_train = sam.peak

sam2 = PeakSampler(10); sam2.start()
t1 = time.time()
fc_ld = res.forecast(train_ld.values[-p:], steps=len(test_ld))
infer_time = time.time()-t1
sam2.stop(); peak_rss_pred = sam2.peak

fc_ld = pd.DataFrame(fc_ld, index=test_ld.index, columns=train_ld.columns)

# ---- invert transform for TARGET back to LEVELS ----
```

```
# cumulative sum of predicted dlog1p → log1p path → levels
log1p_path = last_log1p[TARGET] + fc_ld[TARGET].cumsum()
y_pred_levels = np.expm1(log1p_path)   # inverse of log1p

# ---- metrics on ORIGINAL scale ----
mse  = mean_squared_error(y_true_aligned.values, y_pred_levels.values)
rmse = mse**0.5
mae  = mean_absolute_error(y_true_aligned.values, y_pred_levels.values)
mape = mean_absolute_percentage_error(y_true_aligned.values,
y_pred_levels.values) * 100
r2   = r2_score(y_true_aligned.values, y_pred_levels.values)

print({
    "k_vars": len(keep_cols), "lag_p": p, "kept": keep_cols,
    "MAE": mae, "RMSE": rmse, "MSE": mse, "MAPE_%": mape, "R2": r2,
    "train_time_s": train_time, "infer_time_s": infer_time,
    "peak_rss_train_MB": peak_rss_train, "peak_rss_predict_MB":
peak_rss_pred,
    "T_train_ld": len(train_ld)
})

# ---- plot test window ----
plt.figure(figsize=(11,4))
plt.plot(y_true_aligned.index, y_true_aligned.values, label="Actual")
plt.plot(y_pred_levels.index, y_pred_levels.values, "--", label=f"VAR
(p={p}, log-diff)")
plt.title("VAR Backtest (levels): Jan-Dec 2023")
plt.xlabel("Month"); plt.ylabel("USD/GHS"); plt.legend();
plt.tight_layout(); plt.show()

# Actual vs ARIMA forecast (x-axis = 2024 months only)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# ---------- CONFIG ----------
PATH_ACTUAL   = "/content/Monthly Exchange Rates (1).csv"
PATH_FORECAST = "/content/forecast_VAR_next12.csv"

# Set these if columns differ in your files
# DATE_COL_ACTUAL    = "Month"          # or None to auto-detect - Not
needed for the actuals data format
TARGET_COL_ACTUAL  = "Interbank_Exchange_Rate" # e.g.
"Interbank_Exchange_Rate" - Set this to the actual variable name
DATE_COL_FORECAST  = "Month"          # or None to auto-detect
```

```
FORECAST_VALUE_COL = "Forecast_VAR" # or None to auto-detect

LOWER_COL = "Lower_95"   # optional
UPPER_COL = "Upper_95"   # optional
YLABEL    = "Exchange Rate (USD/GHS)"
TITLE     = "Actual vs VAR Forecast — 2024"
SAVE_PNG  = "/content/actual_vs_var_2024.png"
SAVE_CSV  = "/content/actual_vs_var_2024.csv"

# ---------- HELPERS ----------
def detect_date_col(cols):
    cands = [c for c in cols if str(c).strip().lower() in
{"month","date","period"}]
    return cands[0] if cands else None

def detect_target_col(cols):
    best, score = None, -1
    for c in cols:
        cl = str(c).lower(); s = 0
        if "exchange" in cl and "rate" in cl: s += 5
        if "interbank" in cl: s += 3
        if "usd" in cl: s += 2
        if "ghs" in cl or "cedi" in cl: s += 1
        if s > score: best, score = c, s
    return best

def ensure_datetime(df, col):
    df[col] = pd.to_datetime(df[col], errors="coerce")
    if df[col].isna().any():
        raise ValueError(f"Unparseable dates in '{col}'.")
    return df

# ---------- LOAD: ACTUALS ----------
dfa = pd.read_csv(PATH_ACTUAL)

# Reshape the actuals data to a long format
if TARGET_COL_ACTUAL is None:
     raise ValueError("Set TARGET_COL_ACTUAL to your column name for the
actual variable.")

dfa_melted = dfa.melt(id_vars=['Year', 'Variables'], var_name='Month',
value_name=TARGET_COL_ACTUAL)
dfa_melted['Month'] = dfa_melted['Month'].astype(str) + ' ' +
dfa_melted['Year'].astype(str)
```

```
dfa_melted = ensure_datetime(dfa_melted,
'Month').sort_values('Month').set_index('Month')

y_actual = pd.to_numeric(dfa_melted[TARGET_COL_ACTUAL],
errors="coerce").ffill()

# Restrict to 2024 months (MS = Month Start)
idx_2024 = pd.date_range("2024-01-01", "2024-12-01", freq="MS")
y_act_2024 = y_actual.reindex(idx_2024)

# ---------- LOAD: FORECAST ----------
dff = pd.read_csv(PATH_FORECAST)
if DATE_COL_FORECAST is None:
    DATE_COL_FORECAST = detect_date_col(dff.columns)
if FORECAST_VALUE_COL is None:
    # common names if not "Forecast_ARIMA"
    cands = [c for c in dff.columns if str(c).lower() in
{"forecast_var","forecast","pred","predicted_mean"}]
    FORECAST_VALUE_COL = cands[0] if cands else None
if DATE_COL_FORECAST is None or FORECAST_VALUE_COL is None:
    raise ValueError("Set DATE_COL_FORECAST / FORECAST_VALUE_COL to your
column names.")

dff = ensure_datetime(dff,
DATE_COL_FORECAST).sort_values(DATE_COL_FORECAST).set_index(DATE_COL_FOREC
AST)
y_fc_2024 = pd.to_numeric(dff[FORECAST_VALUE_COL],
errors="coerce").reindex(idx_2024)

# Optional confidence bands if available
lower_2024 = dff[LOWER_COL].reindex(idx_2024).astype(float) if LOWER_COL
in dff.columns else None
upper_2024 = dff[UPPER_COL].reindex(idx_2024).astype(float) if UPPER_COL
in dff.columns else None

# ---------- WARN if missing months ----------
miss_act = [d.strftime("%b") for d in idx_2024[y_act_2024.isna()]]
miss_fc  = [d.strftime("%b") for d in idx_2024[y_fc_2024.isna()]]
if miss_act: print("Actuals missing for:", ", ".join(miss_act))
if miss_fc:  print("Forecast missing for:", ", ".join(miss_fc))

# ---------- PLOT ----------
plt.figure(figsize=(11,6))
plt.plot(idx_2024, y_act_2024.values, label="Actual (2024)")
plt.plot(idx_2024, y_fc_2024.values, label="VAR Forecast (2024)")
```

```
if lower_2024 is not None and upper_2024 is not None:
    plt.fill_between(idx_2024, lower_2024.values, upper_2024.values,
alpha=0.2, label="95% CI")

plt.xlim(pd.Timestamp("2024-01-01"), pd.Timestamp("2024-12-31"))
plt.xticks(idx_2024, [d.strftime("%b") for d in idx_2024])
plt.title(TITLE)
plt.xlabel("2024 Months")
plt.ylabel(YLABEL)
plt.legend()
plt.tight_layout()
plt.savefig(SAVE_PNG, dpi=300, bbox_inches="tight")
plt.show()
print(f"Saved chart → {SAVE_PNG}")

# ---------- SAVE merged 2024 data (optional) ----------
out = pd.DataFrame({
    "Month": idx_2024,
    "Actual": y_act_2024.values,
    "Forecast_VAR": y_fc_2024.values,
})
if lower_2024 is not None and upper_2024 is not None:
    out["Lower_95"] = lower_2024.values
    out["Upper_95"] = upper_2024.values
out.to_csv(SAVE_CSV, index=False)
print(f"Saved merged CSV → {SAVE_CSV}")
```

Algorithms of Long Short-Term Memory (LSTM) Model Algorithm

```
# LSTM backtest: 2014–2022 train, 2023 test (levels), R² on original scale
+ RSS timings

import os, time, threading, psutil, numpy as np, pandas as pd,
matplotlib.pyplot as plt, tensorflow as tf
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score, mean_absolute_percentage_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Input
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

# ---------- config ----------
CSV_PATH   = "/content/cleaned_macro_dataset.csv"
```

```
DATE_COL   = "Month"
TARGET_COL = "Interbank Exchange Rate US Dollar (USD)"
LOOKBACK   = 12
EPOCHS     = 200
BATCH      = 8
SEED       = 7

# ---------- reproducibility ----------
np.random.seed(SEED); tf.random.set_seed(SEED)

# ---------- helpers (RSS peak) ----------
proc = psutil.Process()
def rss_mb(): return proc.memory_info().rss/(1024**2)
class PeakSampler:
    def __init__(self, interval_ms=10): self.dt=interval_ms/1000.;
self._stop=False; self.peak=rss_mb()
    def start(self):
        def loop():
            import time
            while not self._stop:
                self.peak=max(self.peak, rss_mb()); time.sleep(self.dt)
        self._stop=False;
self.t=threading.Thread(target=loop,daemon=True); self.t.start()
    def stop(self): self._stop=True; self.t.join(timeout=1.0)

def make_sequences(X, y, win):
    Xs, ys = [], []
    for i in range(win, len(X)):
        Xs.append(X[i-win:i])
        ys.append(y[i])
    return np.array(Xs), np.array(ys)

# ---------- load ----------
df = pd.read_csv(CSV_PATH,
parse_dates=[DATE_COL]).sort_values(DATE_COL).set_index(DATE_COL)
df = df.apply(pd.to_numeric, errors="coerce").ffill().bfill()
y_all = df[[TARGET_COL]].copy()
X_all = df.drop(columns=[TARGET_COL]).copy()

# ---------- split: last 12 months = test (Jan-Dec 2023) ----------
h = 12
df_tr, df_te = df.iloc[:-h], df.iloc[-h:]
X_tr_raw, X_te_raw = X_all.iloc[:-h], X_all.iloc[-h:]
y_tr_raw, y_te_raw = y_all.iloc[:-h], y_all.iloc[-h:]
```

```
# scalers fit on TRAIN ONLY (no leakage)
scX = MinMaxScaler().fit(X_tr_raw.values)
scy = MinMaxScaler().fit(y_tr_raw.values)

X_tr = scX.transform(X_tr_raw.values)
y_tr = scy.transform(y_tr_raw.values)
# To build test sequences, we need the last LOOKBACK rows from the end of
TRAIN joined with TEST
X_join = np.vstack([X_tr, scX.transform(X_te_raw.values)])
y_join = np.vstack([y_tr, scy.transform(y_te_raw.values)])

# sequences
X_seq, y_seq = make_sequences(X_join, y_join, LOOKBACK)
# split back: the last len(y_te_raw) rows correspond to test targets
n_test = len(y_te_raw)
X_train, y_train = X_seq[:-n_test], y_seq[:-n_test]
X_test,  y_test  = X_seq[-n_test:],  y_seq[-n_test:]

# ---------- model ----------
model = Sequential([Input(shape=(LOOKBACK, X_train.shape[2])),
                    LSTM(64),
                    Dense(1)])
model.compile(optimizer="adam", loss="mse")

cb = [EarlyStopping(monitor="val_loss", patience=20,
restore_best_weights=True),
      ReduceLROnPlateau(monitor="val_loss", factor=0.5, patience=10)]

# timings + RSS
sam = PeakSampler(10); sam.start()
t0 = time.time()
hist = model.fit(X_train, y_train, epochs=EPOCHS, batch_size=BATCH,
                 validation_data=(X_test, y_test), verbose=0,
callbacks=cb)
train_time = time.time() - t0
sam.stop(); peak_rss_train = sam.peak

sam2 = PeakSampler(10); sam2.start()
t1 = time.time()
y_pred_sc = model.predict(X_test, verbose=0)
infer_time = time.time() - t1
sam2.stop(); peak_rss_pred = sam2.peak

# inverse-transform to ORIGINAL scale
y_pred = scy.inverse_transform(y_pred_sc)
```

```
y_true = scy.inverse_transform(y_test)

# metrics (original scale)
mse  = mean_squared_error(y_true, y_pred)
rmse = mse**0.5
mae  = mean_absolute_error(y_true, y_pred)
mape = mean_absolute_percentage_error(y_true, y_pred)*100
r2   = r2_score(y_true, y_pred)

print({
    "MAE": float(mae), "RMSE": float(rmse), "MSE": float(mse), "MAPE_%":
float(mape), "R2": float(r2),
    "train_time_s": train_time, "infer_time_s": infer_time,
    "peak_rss_train_MB": peak_rss_train, "peak_rss_predict_MB":
peak_rss_pred
})

# Plot test window with dates
test_months = df.index[-n_test:]
plt.figure(figsize=(11,4))
plt.plot(test_months, y_true.ravel(), label="Actual")
plt.plot(test_months, y_pred.ravel(), "--", label="LSTM Forecast")
plt.title("LSTM Backtest: Jan-Dec 2023")
plt.xlabel("Month"); plt.ylabel("USD/GHS"); plt.legend();
plt.tight_layout(); plt.show()

# Actual vs ARIMA forecast (x-axis = 2024 months only)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# ---------- CONFIG ----------
PATH_ACTUAL   = "/content/Monthly Exchange Rates (1).csv"
PATH_FORECAST = "/content/forecast_LSTM_next12.csv"

# Set these if columns differ in your files
# DATE_COL_ACTUAL    = "Month"          # or None to auto-detect - Not
needed for the actuals data format
TARGET_COL_ACTUAL  = "Interbank_Exchange_Rate" # e.g.
"Interbank_Exchange_Rate" - Set this to the actual variable name
DATE_COL_FORECAST  = "Month"          # or None to auto-detect
FORECAST_VALUE_COL = "Forecast_LSTM" # or None to auto-detect

LOWER_COL = "Lower_95"   # optional
UPPER_COL = "Upper_95"   # optional
```

```
YLABEL    = "Exchange Rate (USD/GHS)"
TITLE     = "Actual vs LSTM Forecast — 2024"
SAVE_PNG  = "/content/actual_vs_lstm_2024.png"
SAVE_CSV  = "/content/actual_vs_lstm_2024.csv"

# ---------- HELPERS ----------
def detect_date_col(cols):
    cands = [c for c in cols if str(c).strip().lower() in
{"month","date","period"}]
    return cands[0] if cands else None

def detect_target_col(cols):
    best, score = None, -1
    for c in cols:
        cl = str(c).lower(); s = 0
        if "exchange" in cl and "rate" in cl: s += 5
        if "interbank" in cl: s += 3
        if "usd" in cl: s += 2
        if "ghs" in cl or "cedi" in cl: s += 1
        if s > score: best, score = c, s
    return best

def ensure_datetime(df, col):
    df[col] = pd.to_datetime(df[col], errors="coerce")
    if df[col].isna().any():
        raise ValueError(f"Unparseable dates in '{col}'.")
    return df

# ---------- LOAD: ACTUALS ----------
dfa = pd.read_csv(PATH_ACTUAL)

# Reshape the actuals data to a long format
if TARGET_COL_ACTUAL is None:
     raise ValueError("Set TARGET_COL_ACTUAL to your column name for the
actual variable.")

dfa_melted = dfa.melt(id_vars=['Year', 'Variables'], var_name='Month',
value_name=TARGET_COL_ACTUAL)
dfa_melted['Month'] = dfa_melted['Month'].astype(str) + ' ' +
dfa_melted['Year'].astype(str)
dfa_melted = ensure_datetime(dfa_melted,
'Month').sort_values('Month').set_index('Month')

y_actual = pd.to_numeric(dfa_melted[TARGET_COL_ACTUAL],
errors="coerce").ffill()
```

```
# Restrict to 2024 months (MS = Month Start)
idx_2024 = pd.date_range("2024-01-01", "2024-12-01", freq="MS")
y_act_2024 = y_actual.reindex(idx_2024)

# ---------- LOAD: FORECAST ----------
dff = pd.read_csv(PATH_FORECAST)
if DATE_COL_FORECAST is None:
    DATE_COL_FORECAST = detect_date_col(dff.columns)
if FORECAST_VALUE_COL is None:
    # common names if not "Forecast_ARIMA"
    cands = [c for c in dff.columns if str(c).lower() in
{"forecast_lstm","forecast","pred","predicted_mean"}]
    FORECAST_VALUE_COL = cands[0] if cands else None
if DATE_COL_FORECAST is None or FORECAST_VALUE_COL is None:
    raise ValueError("Set DATE_COL_FORECAST / FORECAST_VALUE_COL to your
column names.")

dff = ensure_datetime(dff,
DATE_COL_FORECAST).sort_values(DATE_COL_FORECAST).set_index(DATE_COL_FOREC
AST)
y_fc_2024 = pd.to_numeric(dff[FORECAST_VALUE_COL],
errors="coerce").reindex(idx_2024)

# Optional confidence bands if available
lower_2024 = dff[LOWER_COL].reindex(idx_2024).astype(float) if LOWER_COL
in dff.columns else None
upper_2024 = dff[UPPER_COL].reindex(idx_2024).astype(float) if UPPER_COL
in dff.columns else None

# ---------- WARN if missing months ----------
miss_act = [d.strftime("%b") for d in idx_2024[y_act_2024.isna()]]
miss_fc  = [d.strftime("%b") for d in idx_2024[y_fc_2024.isna()]]
if miss_act: print("Actuals missing for:", ", ".join(miss_act))
if miss_fc:  print("Forecast missing for:", ", ".join(miss_fc))

# ---------- PLOT ----------
plt.figure(figsize=(11,6))
plt.plot(idx_2024, y_act_2024.values, label="Actual (2024)")
plt.plot(idx_2024, y_fc_2024.values, label="LSTM Forecast (2024)")

if lower_2024 is not None and upper_2024 is not None:
    plt.fill_between(idx_2024, lower_2024.values, upper_2024.values,
alpha=0.2, label="95% CI")
```

```
plt.xlim(pd.Timestamp("2024-01-01"), pd.Timestamp("2024-12-31"))
plt.xticks(idx_2024, [d.strftime("%b") for d in idx_2024])
plt.title(TITLE)
plt.xlabel("2024 Months")
plt.ylabel(YLABEL)
plt.legend()
plt.tight_layout()
plt.savefig(SAVE_PNG, dpi=300, bbox_inches="tight")
plt.show()
print(f"Saved chart → {SAVE_PNG}")

# ---------- SAVE merged 2024 data (optional) ----------
out = pd.DataFrame({
    "Month": idx_2024,
    "Actual": y_act_2024.values,
    "Forecast_LSTM": y_fc_2024.values,
})
if lower_2024 is not None and upper_2024 is not None:
    out["Lower_95"] = lower_2024.values
    out["Upper_95"] = upper_2024.values
out.to_csv(SAVE_CSV, index=False)
print(f"Saved merged CSV → {SAVE_CSV}")
```

Algorithms of Support Vector Regression (SVR) Model Algorithm

```
# SVR backtest: last 12 months as test; original-scale metrics; RSS
timings

import time, threading, psutil
import numpy as np, pandas as pd, matplotlib.pyplot as plt
from sklearn.svm import SVR
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score, mean_absolute_percentage_error
from sklearn.model_selection import GridSearchCV

# ---------- config ----------
CSV_PATH   = "/content/cleaned_macro_dataset.csv"
DATE_COL   = "Month"
TARGET_COL = "Interbank Exchange Rate US Dollar (USD)"

# ---------- helpers for peak RSS ----------
proc = psutil.Process()
def rss_mb(): return proc.memory_info().rss/(1024**2)
class PeakSampler:
```

```
    def __init__(self, interval_ms=10): self.dt=interval_ms/1000.;
self._stop=False; self.peak=rss_mb()
    def start(self):
        def loop():
            import time
            while not self._stop:
                self.peak=max(self.peak, rss_mb()); time.sleep(self.dt)
        self._stop=False; self.t=threading.Thread(target=loop,
daemon=True); self.t.start()
    def stop(self): self._stop=True; self.t.join(timeout=1.0)

# ---------- load ----------
df = pd.read_csv(CSV_PATH,
parse_dates=[DATE_COL]).sort_values(DATE_COL).set_index(DATE_COL).asfreq("
MS")
df = df.apply(pd.to_numeric, errors="coerce").ffill().bfill()
assert TARGET_COL in df.columns

X_all = df.drop(columns=[TARGET_COL])
y_all = df[[TARGET_COL]]

# split: last 12 months = test
h = 12
X_tr_raw, X_te_raw = X_all.iloc[:-h], X_all.iloc[-h:]
y_tr_raw, y_te_raw = y_all.iloc[:-h], y_all.iloc[-h:]

# scale on TRAIN only (no leakage)
scX = MinMaxScaler().fit(X_tr_raw.values)
scy = MinMaxScaler().fit(y_tr_raw.values)

X_tr = scX.transform(X_tr_raw.values)
X_te = scX.transform(X_te_raw.values)
y_tr = scy.transform(y_tr_raw.values).ravel()

# grid search on TRAIN
param_grid = {
    "C": [1, 10, 100, 1000],
    "epsilon": [0.01, 0.1, 0.5],
    "gamma": ["scale", "auto", 0.01, 0.1, 1.0]
}
svr = SVR(kernel="rbf")

sam = PeakSampler(10); sam.start()
t0 = time.time()
```

```
g = GridSearchCV(svr, param_grid=param_grid,
scoring="neg_mean_squared_error", cv=5, n_jobs=-1, verbose=0)
g.fit(X_tr, y_tr)
train_time = time.time() - t0
sam.stop(); peak_rss_train = sam.peak

best = g.best_estimator_
print("Best params:", g.best_params_)

# predict TEST
sam2 = PeakSampler(10); sam2.start()
t1 = time.time()
y_hat_scaled = best.predict(X_te)
infer_time = time.time() - t1
sam2.stop(); peak_rss_pred = sam2.peak

# invert to ORIGINAL scale
y_hat = scy.inverse_transform(y_hat_scaled.reshape(-1,1)).ravel()
y_true = y_te_raw.values.ravel()

# metrics on original scale
mse  = mean_squared_error(y_true, y_hat)
rmse = mse**0.5
mae  = mean_absolute_error(y_true, y_hat)
mape = mean_absolute_percentage_error(y_true, y_hat)*100
r2   = r2_score(y_true, y_hat)

print({
    "MAE": float(mae), "RMSE": float(rmse), "MSE": float(mse), "MAPE_%":
float(mape), "R2": float(r2),
    "train_time_s": train_time, "infer_time_s": infer_time,
    "peak_rss_train_MB": peak_rss_train, "peak_rss_predict_MB":
peak_rss_pred
})

# plot: test window with dates
test_months = X_te_raw.index
plt.figure(figsize=(11,4))
plt.plot(test_months, y_true, label="Actual")
plt.plot(test_months, y_hat, "--", label="SVR Forecast")
plt.title("SVR Backtest: Jan-Dec 2023")
plt.xlabel("Month"); plt.ylabel("USD/GHS"); plt.legend();
plt.tight_layout(); plt.show()

# Actual vs ARIMA forecast (x-axis = 2024 months only)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# ---------- CONFIG ----------
PATH_ACTUAL   = "/content/Monthly Exchange Rates (1).csv"
PATH_FORECAST = "/content/forecast_SVR_next12.csv"

# Set these if columns differ in your files
# DATE_COL_ACTUAL    = "Month"          # or None to auto-detect - Not
needed for the actuals data format
TARGET_COL_ACTUAL  = "Interbank_Exchange_Rate" # e.g.
"Interbank_Exchange_Rate" - Set this to the actual variable name
DATE_COL_FORECAST  = "Month"          # or None to auto-detect
FORECAST_VALUE_COL = "Forecast_SVR" # or None to auto-detect

LOWER_COL = "Lower_95"   # optional
UPPER_COL = "Upper_95"   # optional
YLABEL    = "Exchange Rate (USD/GHS)"
TITLE     = "Actual vs SVR Forecast — 2024"
SAVE_PNG  = "/content/actual_vs_svr_2024.png"
SAVE_CSV  = "/content/actual_vs_svr_2024.csv"

# ---------- HELPERS ----------
def detect_date_col(cols):
    cands = [c for c in cols if str(c).strip().lower() in
{"month","date","period"}]
    return cands[0] if cands else None

def detect_target_col(cols):
    best, score = None, -1
    for c in cols:
        cl = str(c).lower(); s = 0
        if "exchange" in cl and "rate" in cl: s += 5
        if "interbank" in cl: s += 3
        if "usd" in cl: s += 2
        if "ghs" in cl or "cedi" in cl: s += 1
        if s > score: best, score = c, s
    return best

def ensure_datetime(df, col):
    df[col] = pd.to_datetime(df[col], errors="coerce")
    if df[col].isna().any():
        raise ValueError(f"Unparseable dates in '{col}'.")
    return df
```

```
# ---------- LOAD: ACTUALS ----------
dfa = pd.read_csv(PATH_ACTUAL)

# Reshape the actuals data to a long format
if TARGET_COL_ACTUAL is None:
     raise ValueError("Set TARGET_COL_ACTUAL to your column name for the
actual variable.")

dfa_melted = dfa.melt(id_vars=['Year', 'Variables'], var_name='Month',
value_name=TARGET_COL_ACTUAL)
dfa_melted['Month'] = dfa_melted['Month'].astype(str) + ' ' +
dfa_melted['Year'].astype(str)
dfa_melted = ensure_datetime(dfa_melted,
'Month').sort_values('Month').set_index('Month')

y_actual = pd.to_numeric(dfa_melted[TARGET_COL_ACTUAL],
errors="coerce").ffill()

# Restrict to 2024 months (MS = Month Start)
idx_2024 = pd.date_range("2024-01-01", "2024-12-01", freq="MS")
y_act_2024 = y_actual.reindex(idx_2024)

# ---------- LOAD: FORECAST ----------
dff = pd.read_csv(PATH_FORECAST)
if DATE_COL_FORECAST is None:
    DATE_COL_FORECAST = detect_date_col(dff.columns)
if FORECAST_VALUE_COL is None:
    # common names if not "Forecast_ARIMA"
    cands = [c for c in dff.columns if str(c).lower() in
{"forecast_svr","forecast","pred","predicted_mean"}]
    FORECAST_VALUE_COL = cands[0] if cands else None
if DATE_COL_FORECAST is None or FORECAST_VALUE_COL is None:
    raise ValueError("Set DATE_COL_FORECAST / FORECAST_VALUE_COL to your
column names.")

dff = ensure_datetime(dff,
DATE_COL_FORECAST).sort_values(DATE_COL_FORECAST).set_index(DATE_COL_FOREC
AST)
y_fc_2024 = pd.to_numeric(dff[FORECAST_VALUE_COL],
errors="coerce").reindex(idx_2024)

# Optional confidence bands if available
lower_2024 = dff[LOWER_COL].reindex(idx_2024).astype(float) if LOWER_COL
in dff.columns else None
```

```
upper_2024 = dff[UPPER_COL].reindex(idx_2024).astype(float) if UPPER_COL
in dff.columns else None

# ---------- WARN if missing months ----------
miss_act = [d.strftime("%b") for d in idx_2024[y_act_2024.isna()]]
miss_fc  = [d.strftime("%b") for d in idx_2024[y_fc_2024.isna()]]
if miss_act: print("Actuals missing for:", ", ".join(miss_act))
if miss_fc:  print("Forecast missing for:", ", ".join(miss_fc))

# ---------- PLOT ----------
plt.figure(figsize=(11,6))
plt.plot(idx_2024, y_act_2024.values, label="Actual (2024)")
plt.plot(idx_2024, y_fc_2024.values, label="SVR Forecast (2024)")

if lower_2024 is not None and upper_2024 is not None:
    plt.fill_between(idx_2024, lower_2024.values, upper_2024.values,
alpha=0.2, label="95% CI")

plt.xlim(pd.Timestamp("2024-01-01"), pd.Timestamp("2024-12-31"))
plt.xticks(idx_2024, [d.strftime("%b") for d in idx_2024])
plt.title(TITLE)
plt.xlabel("2024 Months")
plt.ylabel(YLABEL)
plt.legend()
plt.tight_layout()
plt.savefig(SAVE_PNG, dpi=300, bbox_inches="tight")
plt.show()
print(f"Saved chart → {SAVE_PNG}")

# ---------- SAVE merged 2024 data (optional) ----------
out = pd.DataFrame({
    "Month": idx_2024,
    "Actual": y_act_2024.values,
    "Forecast_SVR": y_fc_2024.values,
})
if lower_2024 is not None and upper_2024 is not None:
    out["Lower_95"] = lower_2024.values
    out["Upper_95"] = upper_2024.values
out.to_csv(SAVE_CSV, index=False)
print(f"Saved merged CSV → {SAVE_CSV}")
```

Algorithms of XGBoost Model Algorithm

```
# XGBoost backtest: last 12 months as test; original-scale metrics; RSS
timings

import time, threading, psutil
import numpy as np, pandas as pd, matplotlib.pyplot as plt
from xgboost import XGBRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score, mean_absolute_percentage_error

# ---------- config ----------
CSV_PATH   = "/content/cleaned_macro_dataset.csv"
DATE_COL   = "Month"
TARGET_COL = "Interbank Exchange Rate US Dollar (USD)"

# ---------- helpers for peak RSS ----------
proc = psutil.Process()
def rss_mb(): return proc.memory_info().rss/(1024**2)
class PeakSampler:
    def __init__(self, interval_ms=10): self.dt=interval_ms/1000.;
self._stop=False; self.peak=rss_mb()
    def start(self):
        def loop():
            import time
            while not self._stop:
                self.peak=max(self.peak, rss_mb()); time.sleep(self.dt)
        self._stop=False; self.t=threading.Thread(target=loop,
daemon=True); self.t.start()
    def stop(self): self._stop=True; self.t.join(timeout=1.0)

# ---------- load & prep ----------
df = pd.read_csv(CSV_PATH,
parse_dates=[DATE_COL]).sort_values(DATE_COL).set_index(DATE_COL).asfreq("
MS")
df = df.apply(pd.to_numeric, errors="coerce").ffill().bfill()
assert TARGET_COL in df.columns

X_all = df.drop(columns=[TARGET_COL])
y_all = df[TARGET_COL]

# temporal split: last 12 months = test (Jan-Dec 2023)
h = 12
```

```
X_tr_raw, X_te_raw = X_all.iloc[:-h], X_all.iloc[-h:]
y_tr,     y_te     = y_all.iloc[:-h], y_all.iloc[-h:]

# scale FEATURES on TRAIN only (no leakage). Target stays on original
scale.
scX  = MinMaxScaler().fit(X_tr_raw.values)
X_tr = scX.transform(X_tr_raw.values)
X_te = scX.transform(X_te_raw.values)

# ---------- model + grid ----------
xgb = XGBRegressor(
    objective="reg:squarederror",
    random_state=42,
    tree_method="hist",           # fast & memory-friendly
    n_jobs=-1
)
param_grid = {
    "n_estimators": [200, 400],
    "max_depth":   [3, 5, 7],
    "learning_rate": [0.03, 0.1],
    "subsample": [0.8, 1.0],
    "colsample_bytree": [0.8, 1.0],
    "reg_lambda": [1.0, 5.0]
}

# ---------- train with RSS timing ----------
sam = PeakSampler(10); sam.start()
t0 = time.time()
gs = GridSearchCV(xgb, param_grid=param_grid, cv=3,
scoring="neg_mean_squared_error", n_jobs=-1, verbose=0)
gs.fit(X_tr, y_tr)
train_time = time.time() - t0
sam.stop(); peak_rss_train = sam.peak

best = gs.best_estimator_
print("Best params:", gs.best_params_)

# ---------- inference with RSS timing ----------
sam2 = PeakSampler(10); sam2.start()
t1 = time.time()
y_hat = best.predict(X_te)
infer_time = time.time() - t1
sam2.stop(); peak_rss_pred = sam2.peak

# ---------- metrics on ORIGINAL target scale ----------
```

```
mse  = mean_squared_error(y_te, y_hat)
rmse = mse**0.5
mae  = mean_absolute_error(y_te, y_hat)
mape = mean_absolute_percentage_error(y_te, y_hat) * 100
r2   = r2_score(y_te, y_hat)

print({
    "MAE": float(mae), "RMSE": float(rmse), "MSE": float(mse), "MAPE_%":
float(mape), "R2": float(r2),
    "train_time_s": train_time, "infer_time_s": infer_time,
    "peak_rss_train_MB": peak_rss_train, "peak_rss_predict_MB":
peak_rss_pred
})

# ---------- plot test window ----------
test_months = X_te_raw.index
plt.figure(figsize=(11,4))
plt.plot(test_months, y_te.values, label="Actual")
plt.plot(test_months, y_hat, "--", label="XGBoost Forecast")
plt.title("XGBoost Backtest: Jan-Dec 2023")
plt.xlabel("Month"); plt.ylabel("USD/GHS"); plt.legend();
plt.tight_layout(); plt.show()

# Actual vs ARIMA forecast (x-axis = 2024 months only)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# ---------- CONFIG ----------
PATH_ACTUAL   = "/content/Monthly Exchange Rates (1).csv"
PATH_FORECAST = "/content/forecast_XGB_next12.csv"

# Set these if columns differ in your files
# DATE_COL_ACTUAL    = "Month"          # or None to auto-detect - Not
needed for the actuals data format
TARGET_COL_ACTUAL  = "Interbank_Exchange_Rate" # e.g.
"Interbank_Exchange_Rate" - Set this to the actual variable name
DATE_COL_FORECAST  = "Month"          # or None to auto-detect
FORECAST_VALUE_COL = "Forecast_XGB" # or None to auto-detect

LOWER_COL = "Lower_95"   # optional
UPPER_COL = "Upper_95"   # optional
YLABEL    = "Exchange Rate (USD/GHS)"
TITLE     = "Actual vs XGBoost Forecast — 2024"
SAVE_PNG  = "/content/actual_vs_XGBoost_2024.png"
```

```
SAVE_CSV  = "/content/actual_vs_XGBoost_2024.csv"

# ---------- HELPERS ----------
def detect_date_col(cols):
    cands = [c for c in cols if str(c).strip().lower() in
{"month","date","period"}]
    return cands[0] if cands else None

def detect_target_col(cols):
    best, score = None, -1
    for c in cols:
        cl = str(c).lower(); s = 0
        if "exchange" in cl and "rate" in cl: s += 5
        if "interbank" in cl: s += 3
        if "usd" in cl: s += 2
        if "ghs" in cl or "cedi" in cl: s += 1
        if s > score: best, score = c, s
    return best

def ensure_datetime(df, col):
    df[col] = pd.to_datetime(df[col], errors="coerce")
    if df[col].isna().any():
        raise ValueError(f"Unparseable dates in '{col}'.")
    return df

# ---------- LOAD: ACTUALS ----------
dfa = pd.read_csv(PATH_ACTUAL)

# Reshape the actuals data to a long format
if TARGET_COL_ACTUAL is None:
     raise ValueError("Set TARGET_COL_ACTUAL to your column name for the
actual variable.")

dfa_melted = dfa.melt(id_vars=['Year', 'Variables'], var_name='Month',
value_name=TARGET_COL_ACTUAL)
dfa_melted['Month'] = dfa_melted['Month'].astype(str) + ' ' +
dfa_melted['Year'].astype(str)
dfa_melted = ensure_datetime(dfa_melted,
'Month').sort_values('Month').set_index('Month')

y_actual = pd.to_numeric(dfa_melted[TARGET_COL_ACTUAL],
errors="coerce").ffill()

# Restrict to 2024 months (MS = Month Start)
idx_2024 = pd.date_range("2024-01-01", "2024-12-01", freq="MS")
```

```
y_act_2024 = y_actual.reindex(idx_2024)

# ---------- LOAD: FORECAST ----------
dff = pd.read_csv(PATH_FORECAST)
if DATE_COL_FORECAST is None:
    DATE_COL_FORECAST = detect_date_col(dff.columns)
if FORECAST_VALUE_COL is None:
    # common names if not "Forecast_ARIMA"
    cands = [c for c in dff.columns if str(c).lower() in
{"forecast_xgboost","forecast","pred","predicted_mean"}]
    FORECAST_VALUE_COL = cands[0] if cands else None
if DATE_COL_FORECAST is None or FORECAST_VALUE_COL is None:
    raise ValueError("Set DATE_COL_FORECAST / FORECAST_VALUE_COL to your
column names.")

dff = ensure_datetime(dff,
DATE_COL_FORECAST).sort_values(DATE_COL_FORECAST).set_index(DATE_COL_FOREC
AST)
y_fc_2024 = pd.to_numeric(dff[FORECAST_VALUE_COL],
errors="coerce").reindex(idx_2024)

# Optional confidence bands if available
lower_2024 = dff[LOWER_COL].reindex(idx_2024).astype(float) if LOWER_COL
in dff.columns else None
upper_2024 = dff[UPPER_COL].reindex(idx_2024).astype(float) if UPPER_COL
in dff.columns else None

# ---------- WARN if missing months ----------
miss_act = [d.strftime("%b") for d in idx_2024[y_act_2024.isna()]]
miss_fc  = [d.strftime("%b") for d in idx_2024[y_fc_2024.isna()]]
if miss_act: print("Actuals missing for:", ", ".join(miss_act))
if miss_fc:  print("Forecast missing for:", ", ".join(miss_fc))

# ---------- PLOT ----------
plt.figure(figsize=(11,6))
plt.plot(idx_2024, y_act_2024.values, label="Actual (2024)")
plt.plot(idx_2024, y_fc_2024.values, label="XGBoost Forecast (2024)")

if lower_2024 is not None and upper_2024 is not None:
    plt.fill_between(idx_2024, lower_2024.values, upper_2024.values,
alpha=0.2, label="95% CI")

plt.xlim(pd.Timestamp("2024-01-01"), pd.Timestamp("2024-12-31"))
plt.xticks(idx_2024, [d.strftime("%b") for d in idx_2024])
plt.title(TITLE)
```

```
plt.xlabel("2024 Months")
plt.ylabel(YLABEL)
plt.legend()
plt.tight_layout()
plt.savefig(SAVE_PNG, dpi=300, bbox_inches="tight")
plt.show()
print(f"Saved chart → {SAVE_PNG}")

# ---------- SAVE merged 2024 data (optional) ----------
out = pd.DataFrame({
    "Month": idx_2024,
    "Actual": y_act_2024.values,
    "Forecast_XGBoost": y_fc_2024.values,
})
if lower_2024 is not None and upper_2024 is not None:
    out["Lower_95"] = lower_2024.values
    out["Upper_95"] = upper_2024.values
out.to_csv(SAVE_CSV, index=False)
print(f"Saved merged CSV → {SAVE_CSV}")
```

Algorithms of QLSTM + Classical LSTM hybrid Model Algorithm

```
# QLSTM backtest: last 12 months as test; original-scale metrics; RSS
timings
# pip install pennylane torch scikit-learn matplotlib --quiet

import time, threading, psutil, numpy as np, pandas as pd,
matplotlib.pyplot as plt, torch
from torch import nn
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score, mean_absolute_percentage_error
import pennylane as qml
from pennylane.qnn.torch import TorchLayer

# ---------------- config ----------------
CSV_PATH   = "/content/cleaned_macro_dataset.csv"
DATE_COL   = "Month"
TARGET_COL = "Interbank Exchange Rate US Dollar (USD)"
LOOKBACK   = 12
SEED       = 7
EPOCHS     = 60
BATCH      = 8
LR         = 1e-3
```

```
N_QUBITS   = 4

np.random.seed(SEED); torch.manual_seed(SEED)

# ------------- RSS helpers -------------
proc = psutil.Process()
def rss_mb(): return proc.memory_info().rss/(1024**2)
class PeakSampler:
    def __init__(self, interval_ms=10): self.dt=interval_ms/1000.;
self._stop=False; self.peak=rss_mb()
    def start(self):
        def loop():
            import time
            while not self._stop:
                self.peak=max(self.peak, rss_mb()); time.sleep(self.dt)
        self._stop=False; self.t=threading.Thread(target=loop,
daemon=True); self.t.start()
    def stop(self): self._stop=True; self.t.join(timeout=1.0)

# ------------- data load ---------------
df = pd.read_csv(CSV_PATH,
parse_dates=[DATE_COL]).sort_values(DATE_COL).set_index(DATE_COL).asfreq("
MS")
df = df.apply(pd.to_numeric, errors="coerce").ffill().bfill()

X_all = df.drop(columns=[TARGET_COL]).copy()
y_all = df[[TARGET_COL]].copy()

# split: last 12 months = test (Jan-Dec 2023)
h = 12
X_tr_raw, X_te_raw = X_all.iloc[:-h], X_all.iloc[-h:]
y_tr_raw, y_te_raw = y_all.iloc[:-h], y_all.iloc[-h:]

# scalers fit on TRAIN only (no leakage)
scX = MinMaxScaler().fit(X_tr_raw.values)
scy = MinMaxScaler().fit(y_tr_raw.values)

X_tr = scX.transform(X_tr_raw.values)
y_tr = scy.transform(y_tr_raw.values)
X_te = scX.transform(X_te_raw.values)
y_te = scy.transform(y_te_raw.values)

def make_seq(X, y, win):
    Xs, ys = [], []
    for i in range(win, len(X)):
```

```
        Xs.append(X[i-win:i]); ys.append(y[i])
    return np.asarray(Xs, dtype=np.float32), np.asarray(ys,
dtype=np.float32)

# build sequences so that the last len(y_te) rows map to the test window
X_join = np.vstack([X_tr, X_te])
y_join = np.vstack([y_tr, y_te])
X_seq, y_seq = make_seq(X_join, y_join, LOOKBACK)
n_test = len(y_te)
X_train, y_train = X_seq[:-n_test], y_seq[:-n_test]
X_test,  y_test  = X_seq[-n_test:],  y_seq[-n_test:]

# ------------- quantum layer -----------
dev = qml.device("default.qubit", wires=N_QUBITS)
@qml.qnode(dev, interface="torch")
def qnode(inputs, weights):
    qml.AngleEmbedding(inputs, wires=range(N_QUBITS))
    qml.StronglyEntanglingLayers(weights, wires=range(N_QUBITS))
    return [qml.expval(qml.PauliZ(i)) for i in range(N_QUBITS)]

weight_shapes = {"weights": (1, N_QUBITS, 3)}
qlayer = TorchLayer(qnode, weight_shapes)

# ------------- model -------------------
class QLSTM(nn.Module):
    def __init__(self, in_dim, hid=16):
        super().__init__()
        self.lstm = nn.LSTM(in_dim, hid, batch_first=True)
        self.lin1 = nn.Linear(hid, N_QUBITS)      # map to quantum input
dim
        self.q    = qlayer
        self.lin2 = nn.Linear(N_QUBITS, 1)

    def forward(self, x):
        out,_ = self.lstm(x)
        out   = out[:, -1, :]
        out   = self.lin1(out)
        out   = self.q(out)
        out   = self.lin2(out)
        return out

model = QLSTM(in_dim=X_train.shape[2])
opt   = torch.optim.Adam(model.parameters(), lr=LR)
lossf = nn.MSELoss()
```

```
X_train_t = torch.from_numpy(X_train)
y_train_t = torch.from_numpy(y_train)
X_test_t  = torch.from_numpy(X_test)
y_test_t  = torch.from_numpy(y_test)

# ------------- train with RSS ----------
sam = PeakSampler(10); sam.start()
t0 = time.time()
model.train()
for ep in range(EPOCHS):
    for s in range(0, len(X_train_t), BATCH):
        xb = X_train_t[s:s+BATCH]; yb = y_train_t[s:s+BATCH]
        opt.zero_grad(); pred = model(xb); loss = lossf(pred, yb);
loss.backward(); opt.step()
train_time = time.time()-t0
sam.stop(); peak_rss_train = sam.peak

# ------------- predict with RSS --------
sam2 = PeakSampler(10); sam2.start()
t1 = time.time()
model.eval()
with torch.no_grad():
    y_pred_sc = model(X_test_t).numpy()
infer_time = time.time()-t1
sam2.stop(); peak_rss_pred = sam2.peak

# inverse-transform to ORIGINAL scale
y_pred = scy.inverse_transform(y_pred_sc)
y_true = scy.inverse_transform(y_test)

# metrics (original scale)
mse  = mean_squared_error(y_true, y_pred)
rmse = float(np.sqrt(mse))
mae  = mean_absolute_error(y_true, y_pred)
mape = mean_absolute_percentage_error(y_true, y_pred)*100
r2   = r2_score(y_true, y_pred)

print({
    "MAE": float(mae), "RMSE": rmse, "MSE": float(mse), "MAPE_%":
float(mape), "R2": float(r2),
    "train_time_s": train_time, "infer_time_s": infer_time,
    "peak_rss_train_MB": peak_rss_train, "peak_rss_predict_MB":
peak_rss_pred
})
```

```
# plot test window
test_months = X_te_raw.index
plt.figure(figsize=(11,4))
plt.plot(test_months, y_true.ravel(), label="Actual")
plt.plot(test_months, y_pred.ravel(), "--", label="QLSTM Forecast")
plt.title("QLSTM Backtest: Jan-Dec 2023")
plt.xlabel("Month"); plt.ylabel("USD/GHS"); plt.legend();
plt.tight_layout(); plt.show()

# Actual vs QXGB forecast (x-axis = 2024 months only)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# ---------- CONFIG ----------
PATH_ACTUAL   = "/content/Monthly Exchange Rates (1).csv"
PATH_FORECAST = "/content/forecast_QLSTM_next12.csv"

# Set these if columns differ in your files
# DATE_COL_ACTUAL    = "Month"          # or None to auto-detect - Not
needed for the actuals data format
TARGET_COL_ACTUAL  = "Interbank_Exchange_Rate" # e.g.
"Interbank_Exchange_Rate" - Set this to the actual variable name
DATE_COL_FORECAST  = "Month"          # or None to auto-detect
FORECAST_VALUE_COL = "Forecast_QLSTM" # or None to auto-detect

LOWER_COL = "Lower_95"   # optional
UPPER_COL = "Upper_95"   # optional
YLABEL    = "Exchange Rate (USD/GHS)"
TITLE     = "Actual vs QLSTM Forecast — 2024"
SAVE_PNG  = "/content/actual_vs_qlstm_2024.png"
SAVE_CSV  = "/content/actual_vs_qlstm_2024.csv"

# ---------- HELPERS ----------
def detect_date_col(cols):
    cands = [c for c in cols if str(c).strip().lower() in
{"month","date","period"}]
    return cands[0] if cands else None

def detect_target_col(cols):
    best, score = None, -1
    for c in cols:
        cl = str(c).lower(); s = 0
        if "exchange" in cl and "rate" in cl: s += 5
        if "interbank" in cl: s += 3
```

```
        if "usd" in cl: s += 2
        if "ghs" in cl or "cedi" in cl: s += 1
        if s > score: best, score = c, s
    return best

def ensure_datetime(df, col):
    df[col] = pd.to_datetime(df[col], errors="coerce")
    if df[col].isna().any():
        raise ValueError(f"Unparseable dates in '{col}'.")
    return df

# ---------- LOAD: ACTUALS ----------
dfa = pd.read_csv(PATH_ACTUAL)

# Reshape the actuals data to a long format
if TARGET_COL_ACTUAL is None:
     raise ValueError("Set TARGET_COL_ACTUAL to your column name for the
actual variable.")

dfa_melted = dfa.melt(id_vars=['Year', 'Variables'], var_name='Month',
value_name=TARGET_COL_ACTUAL)
dfa_melted['Month'] = dfa_melted['Month'].astype(str) + ' ' +
dfa_melted['Year'].astype(str)
dfa_melted = ensure_datetime(dfa_melted,
'Month').sort_values('Month').set_index('Month')

y_actual = pd.to_numeric(dfa_melted[TARGET_COL_ACTUAL],
errors="coerce").ffill()

# Restrict to 2024 months (MS = Month Start)
idx_2024 = pd.date_range("2024-01-01", "2024-12-01", freq="MS")
y_act_2024 = y_actual.reindex(idx_2024)

# ---------- LOAD: FORECAST ----------
dff = pd.read_csv(PATH_FORECAST)
if DATE_COL_FORECAST is None:
    DATE_COL_FORECAST = detect_date_col(dff.columns)
if FORECAST_VALUE_COL is None:
    # common names if not "Forecast_QXGBOOST"
    cands = [c for c in dff.columns if str(c).lower() in
{"forecast_qlstm","forecast","pred","predicted_mean"}]
    FORECAST_VALUE_COL = cands[0] if cands else None
if DATE_COL_FORECAST is None or FORECAST_VALUE_COL is None:
    raise ValueError("Set DATE_COL_FORECAST / FORECAST_VALUE_COL to your
column names.")
```

```
dff = ensure_datetime(dff,
DATE_COL_FORECAST).sort_values(DATE_COL_FORECAST).set_index(DATE_COL_FOREC
AST)
y_fc_2024 = pd.to_numeric(dff[FORECAST_VALUE_COL],
errors="coerce").reindex(idx_2024)

# Optional confidence bands if available
lower_2024 = dff[LOWER_COL].reindex(idx_2024).astype(float) if LOWER_COL
in dff.columns else None
upper_2024 = dff[UPPER_COL].reindex(idx_2024).astype(float) if UPPER_COL
in dff.columns else None

# ---------- WARN if missing months ----------
miss_act = [d.strftime("%b") for d in idx_2024[y_act_2024.isna()]]
miss_fc  = [d.strftime("%b") for d in idx_2024[y_fc_2024.isna()]]
if miss_act: print("Actuals missing for:", ", ".join(miss_act))
if miss_fc:  print("Forecast missing for:", ", ".join(miss_fc))

# ---------- PLOT ----------
plt.figure(figsize=(11,6))
plt.plot(idx_2024, y_act_2024.values, label="Actual (2024)")
plt.plot(idx_2024, y_fc_2024.values, label="QLSTM Forecast (2024)")

if lower_2024 is not None and upper_2024 is not None:
    plt.fill_between(idx_2024, lower_2024.values, upper_2024.values,
alpha=0.2, label="95% CI")

plt.xlim(pd.Timestamp("2024-01-01"), pd.Timestamp("2024-12-31"))
plt.xticks(idx_2024, [d.strftime("%b") for d in idx_2024])
plt.title(TITLE)
plt.xlabel("2024 Months")
plt.ylabel(YLABEL)
plt.legend()
plt.tight_layout()
plt.savefig(SAVE_PNG, dpi=300, bbox_inches="tight")
plt.show()
print(f"Saved chart → {SAVE_PNG}")

# ---------- SAVE merged 2024 data (optional) ----------
out = pd.DataFrame({
    "Month": idx_2024,
    "Actual": y_act_2024.values,
    "Forecast_QLSTM": y_fc_2024.values,
})
```

```
if lower_2024 is not None and upper_2024 is not None:
    out["Lower_95"] = lower_2024.values
    out["Upper_95"] = upper_2024.values
out.to_csv(SAVE_CSV, index=False)
print(f"Saved merged CSV → {SAVE_CSV}")
```

Algorithms of QSVR Model Algorithm

```
# QSVR backtest: last 12 months as test; original-scale metrics; RSS
timings
# pip install pennylane scikit-learn matplotlib --quiet

import time, threading, psutil
import numpy as np, pandas as pd, matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score, mean_absolute_percentage_error
import pennylane as qml

# ---------------- config ----------------
CSV_PATH   = "/content/cleaned_macro_dataset.csv"
DATE_COL   = "Month"
TARGET_COL = "Interbank Exchange Rate US Dollar (USD)"
N_QUBITS   = 4
TIME_STEPS = 4          # ensures flattened length <= N_QUBITS (1 feature
× 4 steps)
SEED       = 7
np.random.seed(SEED)

# ------------- RSS helpers -------------
proc = psutil.Process()
def rss_mb(): return proc.memory_info().rss/(1024**2)
class PeakSampler:
    def __init__(self, interval_ms=10): self.dt=interval_ms/1000.;
self._stop=False; self.peak=rss_mb()
    def start(self):
        def loop():
            import time
            while not self._stop:
                self.peak=max(self.peak, rss_mb()); time.sleep(self.dt)
        self._stop=False; self.t=threading.Thread(target=loop,
daemon=True); self.t.start()
    def stop(self): self._stop=True; self.t.join(timeout=1.0)
```

```
# ------------- data load ---------------
df = pd.read_csv(CSV_PATH,
parse_dates=[DATE_COL]).sort_values(DATE_COL).set_index(DATE_COL).asfreq("
MS")
df = df.apply(pd.to_numeric, errors="coerce").ffill().bfill()
assert TARGET_COL in df.columns

X_all_raw = df.drop(columns=[TARGET_COL]).copy()
y_all     = df[[TARGET_COL]].copy()

# pick the single most correlated driver on TRAIN (levels), to satisfy
N_QUBITS constraint
h = 12  # last 12 months test (Jan-Dec 2023)
train_df, test_df = df.iloc[:-h], df.iloc[-h:]
corr =
train_df.drop(columns=[TARGET_COL]).corrwith(train_df[TARGET_COL]).abs().s
ort_values(ascending=False)
top_feat = corr.index[0]  # one feature
X1 = df[[top_feat]].copy()

# scalers (fit on TRAIN only; no leakage)
scX = MinMaxScaler().fit(X1.iloc[:-h].values)
scy = MinMaxScaler().fit(y_all.iloc[:-h].values)
X_scaled = scX.transform(X1.values)
y_scaled = scy.transform(y_all.values)

# create sequences (flattened to length = TIME_STEPS)
def make_seq(X, y, steps):
    Xs, ys = [], []
    for i in range(steps, len(X)):
        Xs.append(X[i-steps:i].flatten())  # shape (steps,)
        ys.append(y[i])
    return np.array(Xs, dtype=float), np.array(ys, dtype=float)

X_seq, y_seq = make_seq(X_scaled, y_scaled, TIME_STEPS)

# split so the last h targets are the test window
n_test = h
X_train, y_train = X_seq[:-n_test], y_seq[:-n_test]
X_test,  y_test  = X_seq[-n_test:],  y_seq[-n_test:]

# ------------- quantum kernel ----------
dev = qml.device("default.qubit", wires=N_QUBITS)
```

```
@qml.qnode(dev)
def q_embed(x):
    # x length <= N_QUBITS (here = 4)
    qml.AngleEmbedding(x, wires=range(N_QUBITS))
    qml.BasicEntanglerLayers(weights=np.ones((1, N_QUBITS)),
wires=range(N_QUBITS))
    return qml.state()

def qkernel_row(x, X_ref):
    # fidelity kernel: |<psi(x)|psi(x_ref)>|^2
    psi_x = q_embed(x)
    row = np.zeros(len(X_ref))
    for j, xr in enumerate(X_ref):
        psi_r = q_embed(xr)
        row[j] = np.abs(np.vdot(psi_x, psi_r))**2
    return row

def qkernel_matrix(A, B):
    K = np.zeros((len(A), len(B)))
    # precompute states for B to speed up
    states_B = [q_embed(xb) for xb in B]
    for i, xa in enumerate(A):
        psi_a = q_embed(xa)
        K[i,:] = [np.abs(np.vdot(psi_a, psi_b))**2 for psi_b in states_B]
    return K

# ------------- train with RSS ----------
sam = PeakSampler(10); sam.start()
t0 = time.time()
K_train = qkernel_matrix(X_train, X_train)
svr = SVR(kernel="precomputed", C=1.0, epsilon=0.01)
svr.fit(K_train, y_train.ravel())
train_time = time.time() - t0
sam.stop(); peak_rss_train = sam.peak

# ------------- predict with RSS --------
sam2 = PeakSampler(10); sam2.start()
t1 = time.time()
K_test = qkernel_matrix(X_test, X_train)
y_pred_sc = svr.predict(K_test)
infer_time = time.time() - t1
sam2.stop(); peak_rss_pred = sam2.peak

# inverse transform → ORIGINAL SCALE
y_pred = scy.inverse_transform(y_pred_sc.reshape(-1,1)).ravel()
```

```
y_true = scy.inverse_transform(y_test.reshape(-1,1)).ravel()

# metrics (original scale)
mse  = mean_squared_error(y_true, y_pred)
rmse = mse**0.5
mae  = mean_absolute_error(y_true, y_pred)
mape = mean_absolute_percentage_error(y_true, y_pred) * 100
r2   = r2_score(y_true, y_pred)

print({
    "feature_used": top_feat,
    "MAE": float(mae), "RMSE": float(rmse), "MSE": float(mse), "MAPE_%":
float(mape), "R2": float(r2),
    "train_time_s": train_time, "infer_time_s": infer_time,
    "peak_rss_train_MB": peak_rss_train, "peak_rss_predict_MB":
peak_rss_pred
})

# plot test window with dates
test_months = df.index[-h:]
plt.figure(figsize=(11,4))
plt.plot(test_months, y_true, label="Actual")
plt.plot(test_months, y_pred, "--", label="QSVR Forecast")
plt.title(f"QSVR Backtest: Jan–Dec 2023 (feature: {top_feat})")
plt.xlabel("Month"); plt.ylabel("USD/GHS"); plt.legend();
plt.tight_layout(); plt.show()

# Actual vs QXGB forecast (x-axis = 2024 months only)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# ---------- CONFIG ----------
PATH_ACTUAL   = "/content/Monthly Exchange Rates (1).csv"
PATH_FORECAST = "/content/forecast_QSVR_next12.csv"

# Set these if columns differ in your files
# DATE_COL_ACTUAL    = "Month"          # or None to auto-detect - Not
needed for the actuals data format
TARGET_COL_ACTUAL  = "Interbank_Exchange_Rate" # e.g.
"Interbank_Exchange_Rate" - Set this to the actual variable name
DATE_COL_FORECAST  = "Month"          # or None to auto-detect
FORECAST_VALUE_COL = "Forecast_QSVR" # or None to auto-detect

LOWER_COL = "Lower_95"   # optional
```

```
UPPER_COL = "Upper_95"   # optional
YLABEL    = "Exchange Rate (USD/GHS)"
TITLE     = "Actual vs QSVR Forecast — 2024"
SAVE_PNG  = "/content/actual_vs_qsvr_2024.png"
SAVE_CSV  = "/content/actual_vs_qsvr_2024.csv"

# ---------- HELPERS ----------
def detect_date_col(cols):
    cands = [c for c in cols if str(c).strip().lower() in
{"month","date","period"}]
    return cands[0] if cands else None

def detect_target_col(cols):
    best, score = None, -1
    for c in cols:
        cl = str(c).lower(); s = 0
        if "exchange" in cl and "rate" in cl: s += 5
        if "interbank" in cl: s += 3
        if "usd" in cl: s += 2
        if "ghs" in cl or "cedi" in cl: s += 1
        if s > score: best, score = c, s
    return best

def ensure_datetime(df, col):
    df[col] = pd.to_datetime(df[col], errors="coerce")
    if df[col].isna().any():
        raise ValueError(f"Unparseable dates in '{col}'.")
    return df

# ---------- LOAD: ACTUALS ----------
dfa = pd.read_csv(PATH_ACTUAL)

# Reshape the actuals data to a long format
if TARGET_COL_ACTUAL is None:
     raise ValueError("Set TARGET_COL_ACTUAL to your column name for the
actual variable.")

dfa_melted = dfa.melt(id_vars=['Year', 'Variables'], var_name='Month',
value_name=TARGET_COL_ACTUAL)
dfa_melted['Month'] = dfa_melted['Month'].astype(str) + ' ' +
dfa_melted['Year'].astype(str)
dfa_melted = ensure_datetime(dfa_melted,
'Month').sort_values('Month').set_index('Month')
```

```
y_actual = pd.to_numeric(dfa_melted[TARGET_COL_ACTUAL],
errors="coerce").ffill()

# Restrict to 2024 months (MS = Month Start)
idx_2024 = pd.date_range("2024-01-01", "2024-12-01", freq="MS")
y_act_2024 = y_actual.reindex(idx_2024)

# ---------- LOAD: FORECAST ----------
dff = pd.read_csv(PATH_FORECAST)
if DATE_COL_FORECAST is None:
    DATE_COL_FORECAST = detect_date_col(dff.columns)
if FORECAST_VALUE_COL is None:
    # common names if not "Forecast_QSVR"
    cands = [c for c in dff.columns if str(c).lower() in
{"forecast_qsvr","forecast","pred","predicted_mean"}]
    FORECAST_VALUE_COL = cands[0] if cands else None
if DATE_COL_FORECAST is None or FORECAST_VALUE_COL is None:
    raise ValueError("Set DATE_COL_FORECAST / FORECAST_VALUE_COL to your
column names.")

dff = ensure_datetime(dff,
DATE_COL_FORECAST).sort_values(DATE_COL_FORECAST).set_index(DATE_COL_FOREC
AST)
y_fc_2024 = pd.to_numeric(dff[FORECAST_VALUE_COL],
errors="coerce").reindex(idx_2024)

# Optional confidence bands if available
lower_2024 = dff[LOWER_COL].reindex(idx_2024).astype(float) if LOWER_COL
in dff.columns else None
upper_2024 = dff[UPPER_COL].reindex(idx_2024).astype(float) if UPPER_COL
in dff.columns else None

# ---------- WARN if missing months ----------
miss_act = [d.strftime("%b") for d in idx_2024[y_act_2024.isna()]]
miss_fc  = [d.strftime("%b") for d in idx_2024[y_fc_2024.isna()]]
if miss_act: print("Actuals missing for:", ", ".join(miss_act))
if miss_fc:  print("Forecast missing for:", ", ".join(miss_fc))

# ---------- PLOT ----------
plt.figure(figsize=(11,6))
plt.plot(idx_2024, y_act_2024.values, label="Actual (2024)")
plt.plot(idx_2024, y_fc_2024.values, label="QSVR Forecast (2024)")

if lower_2024 is not None and upper_2024 is not None:
```

```
    plt.fill_between(idx_2024, lower_2024.values, upper_2024.values,
alpha=0.2, label="95% CI")

plt.xlim(pd.Timestamp("2024-01-01"), pd.Timestamp("2024-12-31"))
plt.xticks(idx_2024, [d.strftime("%b") for d in idx_2024])
plt.title(TITLE)
plt.xlabel("2024 Months")
plt.ylabel(YLABEL)
plt.legend()
plt.tight_layout()
plt.savefig(SAVE_PNG, dpi=300, bbox_inches="tight")
plt.show()
print(f"Saved chart → {SAVE_PNG}")

# ---------- SAVE merged 2024 data (optional) ----------
out = pd.DataFrame({
    "Month": idx_2024,
    "Actual": y_act_2024.values,
    "Forecast_QSVR": y_fc_2024.values,
})
if lower_2024 is not None and upper_2024 is not None:
    out["Lower_95"] = lower_2024.values
    out["Upper_95"] = upper_2024.values
out.to_csv(SAVE_CSV, index=False)
print(f"Saved merged CSV → {SAVE_CSV}")
```

Algorithms of Quantum Feature Map + XGBoost Model Algorithm

```
# QXGBoost backtest: temporal split (last 12 months = test); original-
scale metrics; RSS timings

import time, threading, psutil
import numpy as np, pandas as pd, matplotlib.pyplot as plt
from xgboost import XGBRegressor
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score, mean_absolute_percentage_error

# ---------- config ----------
CSV_PATH   = "/content/cleaned_macro_dataset.csv"
DATE_COL   = "Month"
TARGET_COL = "Interbank Exchange Rate US Dollar (USD)"

# ---------- RSS helpers ----------
```

```
proc = psutil.Process()
def rss_mb(): return proc.memory_info().rss/(1024**2)
class PeakSampler:
    def __init__(self, interval_ms=10): self.dt=interval_ms/1000.;
self._stop=False; self.peak=rss_mb()
    def start(self):
        def loop():
            import time
            while not self._stop:
                self.peak = max(self.peak, rss_mb()); time.sleep(self.dt)
        self._stop=False; self.t=threading.Thread(target=loop,
daemon=True); self.t.start()
    def stop(self): self._stop=True; self.t.join(timeout=1.0)

# ---------- load & prep ----------
df = pd.read_csv(CSV_PATH,
parse_dates=[DATE_COL]).sort_values(DATE_COL).set_index(DATE_COL).asfreq("
MS")
df = df.apply(pd.to_numeric, errors="coerce").ffill().bfill()

X_all = df.drop(columns=[TARGET_COL]).copy()
y_all = df[TARGET_COL].copy()

# temporal split: last 12 months = test (Jan-Dec 2023)
h = 12
X_tr_raw, X_te_raw = X_all.iloc[:-h], X_all.iloc[-h:]
y_tr,     y_te     = y_all.iloc[:-h], y_all.iloc[-h:]

# "quantum-inspired" normalization (really: MinMax on TRAIN only → no
leakage)
scX = MinMaxScaler().fit(X_tr_raw.values)
X_tr = scX.transform(X_tr_raw.values)
X_te = scX.transform(X_te_raw.values)

# ---------- model (reasonable tuned defaults; replace with your grid-
search params if desired) ----------
model = XGBRegressor(
    objective="reg:squarederror",
    random_state=42,
    tree_method="hist",
    n_estimators=300,
    learning_rate=0.08,
    max_depth=5,
    subsample=0.9,
    colsample_bytree=0.9,
```

```
    reg_lambda=1.0,
    n_jobs=-1
)

# ---------- fit with RSS ----------
sam = PeakSampler(10); sam.start()
t0 = time.time()
model.fit(X_tr, y_tr)
train_time = time.time() - t0
sam.stop(); peak_rss_train = sam.peak

# ---------- predict with RSS ----------
sam2 = PeakSampler(10); sam2.start()
t1 = time.time()
y_hat = model.predict(X_te)
infer_time = time.time() - t1
sam2.stop(); peak_rss_pred = sam2.peak

# ---------- metrics (ORIGINAL scale) ----------
mse  = mean_squared_error(y_te, y_hat)
rmse = float(np.sqrt(mse))
mae  = mean_absolute_error(y_te, y_hat)
mape = mean_absolute_percentage_error(y_te, y_hat) * 100
r2   = r2_score(y_te, y_hat)

print({
    "MAE": float(mae), "RMSE": rmse, "MSE": float(mse), "MAPE_%":
float(mape), "R2": float(r2),
    "train_time_s": train_time, "infer_time_s": infer_time,
    "peak_rss_train_MB": peak_rss_train, "peak_rss_predict_MB":
peak_rss_pred
})

# ---------- plot test window ----------
test_months = X_te_raw.index
plt.figure(figsize=(11,4))
plt.plot(test_months, y_te.values, label="Actual")
plt.plot(test_months, y_hat, "--", label="QXGBoost Forecast")
plt.title("QXGBoost Backtest: Jan-Dec 2023")
plt.xlabel("Month"); plt.ylabel("USD/GHS"); plt.legend();
plt.tight_layout(); plt.show()

# Actual vs QXGB forecast (x-axis = 2024 months only)
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt

# ---------- CONFIG ----------
PATH_ACTUAL   = "/content/Monthly Exchange Rates (1).csv"
PATH_FORECAST = "/content/forecast_QXGB_next12.csv"

# Set these if columns differ in your files
# DATE_COL_ACTUAL    = "Month"          # or None to auto-detect - Not
needed for the actuals data format
TARGET_COL_ACTUAL  = "Interbank_Exchange_Rate" # e.g.
"Interbank_Exchange_Rate" - Set this to the actual variable name
DATE_COL_FORECAST  = "Month"          # or None to auto-detect
FORECAST_VALUE_COL = "Forecast_QXGBoost" # or None to auto-detect

LOWER_COL = "Lower_95"   # optional
UPPER_COL = "Upper_95"   # optional
YLABEL    = "Exchange Rate (USD/GHS)"
TITLE     = "Actual vs QXGBoost Forecast — 2024"
SAVE_PNG  = "/content/actual_vs_qxgboost_2024.png"
SAVE_CSV  = "/content/actual_vs_qxgboost_2024.csv"

# ---------- HELPERS ----------
def detect_date_col(cols):
    cands = [c for c in cols if str(c).strip().lower() in
{"month","date","period"}]
    return cands[0] if cands else None

def detect_target_col(cols):
    best, score = None, -1
    for c in cols:
        cl = str(c).lower(); s = 0
        if "exchange" in cl and "rate" in cl: s += 5
        if "interbank" in cl: s += 3
        if "usd" in cl: s += 2
        if "ghs" in cl or "cedi" in cl: s += 1
        if s > score: best, score = c, s
    return best

def ensure_datetime(df, col):
    df[col] = pd.to_datetime(df[col], errors="coerce")
    if df[col].isna().any():
        raise ValueError(f"Unparseable dates in '{col}'.")
    return df

# ---------- LOAD: ACTUALS ----------
```

```
dfa = pd.read_csv(PATH_ACTUAL)

# Reshape the actuals data to a long format
if TARGET_COL_ACTUAL is None:
     raise ValueError("Set TARGET_COL_ACTUAL to your column name for the
actual variable.")

dfa_melted = dfa.melt(id_vars=['Year', 'Variables'], var_name='Month',
value_name=TARGET_COL_ACTUAL)
dfa_melted['Month'] = dfa_melted['Month'].astype(str) + ' ' +
dfa_melted['Year'].astype(str)
dfa_melted = ensure_datetime(dfa_melted,
'Month').sort_values('Month').set_index('Month')

y_actual = pd.to_numeric(dfa_melted[TARGET_COL_ACTUAL],
errors="coerce").ffill()

# Restrict to 2024 months (MS = Month Start)
idx_2024 = pd.date_range("2024-01-01", "2024-12-01", freq="MS")
y_act_2024 = y_actual.reindex(idx_2024)

# ---------- LOAD: FORECAST ----------
dff = pd.read_csv(PATH_FORECAST)
if DATE_COL_FORECAST is None:
    DATE_COL_FORECAST = detect_date_col(dff.columns)
if FORECAST_VALUE_COL is None:
    # common names if not "Forecast_QXGBOOST"
    cands = [c for c in dff.columns if str(c).lower() in
{"forecast_qxgboost","forecast","pred","predicted_mean"}]
    FORECAST_VALUE_COL = cands[0] if cands else None
if DATE_COL_FORECAST is None or FORECAST_VALUE_COL is None:
    raise ValueError("Set DATE_COL_FORECAST / FORECAST_VALUE_COL to your
column names.")

dff = ensure_datetime(dff,
DATE_COL_FORECAST).sort_values(DATE_COL_FORECAST).set_index(DATE_COL_FOREC
AST)
y_fc_2024 = pd.to_numeric(dff[FORECAST_VALUE_COL],
errors="coerce").reindex(idx_2024)

# Optional confidence bands if available
lower_2024 = dff[LOWER_COL].reindex(idx_2024).astype(float) if LOWER_COL
in dff.columns else None
upper_2024 = dff[UPPER_COL].reindex(idx_2024).astype(float) if UPPER_COL
in dff.columns else None
```

```
# ---------- WARN if missing months ----------
miss_act = [d.strftime("%b") for d in idx_2024[y_act_2024.isna()]]
miss_fc  = [d.strftime("%b") for d in idx_2024[y_fc_2024.isna()]]
if miss_act: print("Actuals missing for:", ", ".join(miss_act))
if miss_fc:  print("Forecast missing for:", ", ".join(miss_fc))

# ---------- PLOT ----------
plt.figure(figsize=(11,6))
plt.plot(idx_2024, y_act_2024.values, label="Actual (2024)")
plt.plot(idx_2024, y_fc_2024.values, label="QXGBoost Forecast (2024)")

if lower_2024 is not None and upper_2024 is not None:
    plt.fill_between(idx_2024, lower_2024.values, upper_2024.values,
alpha=0.2, label="95% CI")

plt.xlim(pd.Timestamp("2024-01-01"), pd.Timestamp("2024-12-31"))
plt.xticks(idx_2024, [d.strftime("%b") for d in idx_2024])
plt.title(TITLE)
plt.xlabel("2024 Months")
plt.ylabel(YLABEL)
plt.legend()
plt.tight_layout()
plt.savefig(SAVE_PNG, dpi=300, bbox_inches="tight")
plt.show()
print(f"Saved chart → {SAVE_PNG}")

# ---------- SAVE merged 2024 data (optional) ----------
out = pd.DataFrame({
    "Month": idx_2024,
    "Actual": y_act_2024.values,
    "Forecast_QXGBoost": y_fc_2024.values,
})
if lower_2024 is not None and upper_2024 is not None:
    out["Lower_95"] = lower_2024.values
    out["Upper_95"] = upper_2024.values
out.to_csv(SAVE_CSV, index=False)
print(f"Saved merged CSV → {SAVE_CSV}")
```