---

noteId: "0cc3c500dca011f0bd042d3ebcf3abeb"

tags: []

Comparative Analysis and Deployment of Machine Learning Models on Re

Author: AI Portfolio Project

Date: December 18, 2025

Repository: https://github.com/Midnight-W4lker/Ai-porfolio

---

1. Executive Summary

This project presents a comprehensive framework for forecasting financial time series data using a comparative approach between classical statistical methods (ARIMA) and modern machine learning algorithms (XGBoost, Random Forest, SVR, Prophet). The system is engineered as an end-to-end solution, featuring a dynamic data ingestion pipeline, an automated feature engineering module, and a scalable deployment architecture using Streamlit and FastAPI.

Key outcomes include:

- **Robustness:** Successfully handles real-world stock data with noise and missing values.
- **Performance:** Machine learning models (specifically XGBoost) demonstrated superior adaptability to non-linear market trends compared to linear statistical models.
- **Usability:** A user-friendly interactive dashboard allows non-technical stakeholders to perform complex forecasting tasks.

---

2. Problem Statement

Financial market forecasting is notoriously difficult due to the non-stationary, noisy, and chaotic nature of the data. Traditional methods often fail to capture complex, non-linear dependencies. This project aims to:

1. Evaluate the efficacy of different modeling paradigms on the same dataset.
2. Provide a reusable, modular codebase for time series analysis.
3. Deploy the solution as a web service for real-time accessibility.

---

3. Methodology

3.1 Data Acquisition & Preprocessing

- **Source:** Yahoo Finance API (yfinance).
- **Scope:** Daily closing prices, volume, and high/low indicators.
- **Preprocessing:**
- Handling missing values via forward-fill.
- Normalization using MinMax Scaling (critical for SVR and Neural Networks).
- Stationarity checks (ADF Test) implicitly handled by differencing in ARIMA.

3.2 Feature Engineering

To enable supervised learning algorithms to process time series data, we engineered the following features:

- **Lag Features:** t-1, t-2, ..., t-30 to capture autocorrelation.

- **Rolling Statistics:** 7-day and 30-day rolling means and standard deviations to capture trends and volatility.
- **Temporal Features:** Day of week, month, and quarter to capture seasonality.

3.3 Model Selection

We selected a diverse set of models to represent different forecasting philosophies:

Model	Type	Strengths
ARIMA	Statistical	Excellent for short-term linear trends; interpretable parameters (p,d,q).
XGBoost	Gradient Boosting	Handles non-linearities well; robust to outliers; feature importance insights.
Random Forest	Ensemble	Reduces overfitting via bagging; good baseline for ML approaches.
SVR	Kernel Method	Effective in high-dimensional spaces; robust to noise via margin maximization.
Prophet	Additive Model	Designed by Facebook for business time series with strong seasonal effects.

---

4. System Architecture

The project follows a modular microservices-ready architecture:

- **User** interacts with **Streamlit Dashboard**
- **Streamlit** calls **Data Loader** and **Model Trainer**
- **FastAPI** exposes endpoints for external integration

---

5. Results & Analysis

5.1 Performance Metrics

Models were evaluated using:

- **RMSE (Root Mean Squared Error):** Penalizes large errors heavily.
- **MAE (Mean Absolute Error):** Average magnitude of errors.
- **MAPE (Mean Absolute Percentage Error):** Relative error, useful for business context.

5.2 Observations

- **ARIMA:** Performed well on stable, trending stocks but struggled with sudden volatility.
- **XGBoost:** Consistently outperformed others in minimizing RMSE, effectively leveraging the rolling window features.
- **SVR:** Required significant hyperparameter tuning and scaling but showed promise in low-volatility regimes.

---

6. Deployment

The solution is deployed via two interfaces:

1. **Web Application:** A Streamlit-based dashboard for visual exploration, model training, and comparison.
2. **REST API:** A FastAPI backend serving predictions via JSON endpoints, suitable for integration into larger trading systems.

---

7. Future Work

- **Deep Learning:** Integrate LSTM and Transformer-based models (e.g., Temporal Fusion Transformers).
- **Sentiment Analysis:** Incorporate news sentiment data to augment price features.
- **Automated Tuning:** Implement Optuna or GridSearch for automated hyperparameter optimization.

---

8. Conclusion

This project demonstrates that while traditional statistical methods provide a solid baseline, modern machine learning techniqueswhen combined with robust feature engineeringoffer superior performance for financial time series forecasting. The deployed application serves as a practical tool for democratizing access to these advanced analytical capabilities.