Bayesian Deep Learning for Probabilistic Financial Forecasting: A Comparative Analysis of Uncertainty Quantification and Its Impact on Trading Strategy Performance

Author: Mohansree Vijayakumar
Affiliation: University of Surrey
Mail : mohansreesk14@gmail.com

Abstract

This report details the development and evaluation of a machine learning system for financial time series forecasting with a focus on uncertainty quantification. Traditional forecasting models provide deterministic point predictions, which are insufficient for robust risk management. To address this, we implement and compare standard deep learning models (LSTM, Transformer) against Bayesian methods (Monte Carlo Dropout, Variational Inference) that produce probabilistic forecasts. The system's efficacy is measured not only by prediction accuracy (RMSE, MAE) but also by the quality of its uncertainty estimates (NLL, calibration coverage, sharpness). Furthermore, we demonstrate the practical value of these estimates through an automated backtesting framework that executes an uncertainty-aware trading strategy, evaluated on risk-adjusted return metrics such as the Sharpe Ratio and Value at Risk (VaR). The results indicate that integrating calibrated uncertainty into trading logic can significantly improve risk management and decision-making in algorithmic trading.

Table of Contents

9. Appendices
   A. Full List of Technical Indicators
   B. Hyperparameter Search Space
   C. Environment Configuration
   D. Dockerfile

Author: Mohansree Vijayakumar

List of Figures

List of Tables

List of Abbreviations

- **AI** - Artificial Intelligence
- **API** - Application Programming Interface
- **BNN** - Bayesian Neural Network
- **ELBO** - Evidence Lower Bound
- **EMA** - Exponential Moving Average
- **GPU** - Graphics Processing Unit
- **HPO** - Hyperparameter Optimization
- **LSTM** - Long Short-Term Memory
- **MACD** - Moving Average Convergence Divergence
- **MAE** - Mean Absolute Error
- **MC** - Monte Carlo
- **ML** - Machine Learning
- **MLOps** - Machine Learning Operations
- **MSE** - Mean Squared Error
- **NLL** - Negative Log Likelihood
- **OHLCV** - Open, High, Low, Close, Volume
- **RMSE** - Root Mean Square Error
- **RNN** - Recurrent Neural Network
- **RSI** - Relative Strength Index
- **SMA** - Simple Moving Average
- **VI** - Variational Inference
- **VaR** - Value at Risk

1. Introduction

1.1. Background and Motivation

Financial markets are complex, dynamic systems characterized by high levels of noise and non-stationarity. Forecasting the future movements of asset prices is a central challenge in quantitative finance and has been the subject of extensive research for decades. The Efficient Market Hypothesis (EMH) posits that asset prices fully reflect all available information, making it impossible to consistently achieve returns in excess of average market returns. However, behavioral economics and the discovery of market anomalies suggest that inefficiencies do exist, providing opportunities for sophisticated forecasting models to extract predictive signals.

In recent years, deep learning has emerged as a powerful tool for modeling complex, non-linear relationships in sequential data. Models such as Long Short-Term Memory (LSTM) networks and Transformers have demonstrated state-of-the-art performance in various time series forecasting tasks. Yet, in the high-stakes domain of finance, a single point forecast is often insufficient. A prediction of a 1% price increase carries vastly different implications if the model is highly confident versus highly uncertain. Without a measure of this confidence, decision-makers cannot effectively manage risk.

1.2. Problem Statement

The primary limitation of standard deep learning models in financial applications is their inability to quantify predictive uncertainty. They produce deterministic outputs, failing to capture their own confidence about a forecast. This project addresses this critical gap by developing a system that not only predicts future stock prices but also provides a principled measure of the uncertainty associated with each prediction. The core problem is to move beyond point forecasting to a probabilistic framework, enabling more robust, risk-aware algorithmic trading strategies.

1.3. Research Questions

This project seeks to answer the following key research questions:

1. To what extent can Bayesian deep learning methods, such as MC Dropout and Variational Inference, effectively capture uncertainty in volatile financial time series?
2. How do the accuracy and calibration of probabilistic forecasts from Bayesian models compare to the point forecasts from their deterministic counterparts (LSTM, Transformer)?
3. Does the integration of predictive uncertainty into a trading strategy lead to a quantifiable improvement in risk-adjusted performance, as measured by metrics like the Sharpe Ratio and Value at Risk?

1.4. Project Aims and Objectives

The principal aim of this project is to design, implement, and evaluate a production-ready system for uncertainty-aware stock price forecasting. This is broken down into the following objectives:

- **Implement Diverse Models:** Develop and train four distinct deep learning models: two deterministic (LSTM, Transformer) and two Bayesian (MC Dropout LSTM, Pyro BNN).
- **Develop a Robust Data Pipeline:** Create a reproducible data ingestion and feature engineering pipeline for financial time series data.
- **Establish a Comprehensive Evaluation Framework:** Evaluate models on point accuracy, probabilistic forecast quality, and economic value through backtesting.
- **Build an Interactive Dashboard:** Create a user-friendly web application with Streamlit for model analysis and visualization.
- **Ensure Reproducibility:** Package the entire system using MLOps best practices, including experiment tracking (MLflow) and containerization (Docker).

1.5. Scope and Limitations

The project's scope is defined as follows:

- **Data:** Daily OHLCV data for a limited set of US equities (AAPL, MSFT, TSLA).
- **Forecast Horizon:** Single-step-ahead prediction (forecasting the next day's closing price).
- **Strategy:** A simple, long/short trading strategy based on predictive uncertainty.

The study is subject to several limitations:

- **Transaction Costs:** The backtesting framework does not model transaction costs or market slippage.
- **Stationarity:** The models assume that the underlying statistical properties of the time series are relatively stable, which may not hold during market regime shifts.
- **Information Set:** The models rely solely on historical price and volume data (technical analysis) and do not incorporate fundamental or alternative data.

1.6. Report Structure

This report is organized into seven chapters. Chapter 2 provides a review of relevant literature. Chapter 3 details the system architecture and methodology. Chapter 4 describes the evaluation framework. Chapter 5 presents and discusses the experimental results. Chapter 6 demonstrates the interactive dashboard. Finally, Chapter 7 concludes the report and suggests directions for future work.

2. Literature Review

This chapter surveys the key academic and technical concepts that form the foundation of this project, covering time series analysis, deep learning models, uncertainty quantification, and their application in algorithmic trading.

2.1. Traditional Time Series Forecasting

Before the advent of deep learning, statistical methods dominated time series forecasting. Models like ARIMA (Autoregressive Integrated Moving Average) were designed to capture linear relationships and seasonality in stationary time series. In finance, ARCH (Autoregressive Conditional Heteroskedasticity) and GARCH (Generalized ARCH) models were developed to specifically model volatility clustering—the phenomenon where periods of high volatility are followed by more high volatility. While effective for their intended purpose, these models often struggle to capture the complex, non-linear dynamics inherent in financial markets.

2.2. Deep Learning for Financial Forecasting

Deep learning models, with their ability to learn hierarchical representations from data, have shown significant promise in overcoming the limitations of traditional methods.

- **Recurrent Neural Networks (RNNs) and LSTMs:** RNNs are specifically designed for sequential data. However, simple RNNs suffer from the vanishing gradient problem, making it difficult for them to learn long-range dependencies. The Long Short-Term Memory (LSTM) network, introduced by Hochreiter & Schmidhuber (1997), addresses this with a gating mechanism that allows the network to selectively remember or forget information over long sequences. This makes LSTMs particularly well-suited for financial time series.
- **Attention Mechanisms and Transformers:** The Transformer architecture, proposed by Vaswani et al. (2017) in the context of natural language processing, has since been adapted for time series forecasting. Its core innovation is the self-attention mechanism, which allows the model to weigh the importance of different past time steps when making a prediction. This enables the model to capture complex dependencies without relying on recurrence, offering better parallelization and potentially superior performance on long sequences.

2.3. Uncertainty Quantification in Deep Learning

Uncertainty in machine learning can be categorized into two types:

- **Aleatoric Uncertainty:** This captures the inherent randomness or noise in the data generating process. It is irreducible and represents the uncertainty that would remain even with an infinite amount of data.
- **Epistemic Uncertainty:** This captures the model's own uncertainty about its parameters due to limited training data. It is reducible and can be decreased by providing the model with more data.

Bayesian Neural Networks (BNNs) provide a theoretical framework for capturing both types of uncertainty by treating model weights not as single point estimates, but as probability distributions.

- **Bayesian Neural Networks (BNNs):** In a BNN, a prior distribution is placed over the weights. Using Bayes' theorem, this prior is updated with observed data to form a posterior distribution. Making predictions involves integrating over this entire posterior distribution of weights, which is computationally intractable. Therefore, approximation methods are required.
- **Monte Carlo (MC) Dropout:** Gal & Ghahramani (2016) showed that the common deep learning regularization technique, dropout, can be interpreted as a Bayesian approximation. By keeping dropout active during inference and performing multiple forward passes, one is effectively drawing samples from the approximate posterior distribution of the model's weights. The variance of these samples serves as an estimate of epistemic uncertainty.
- **Variational Inference (VI):** VI is a more direct approach to approximating the posterior distribution. It proposes a simpler, parameterized family of distributions (the variational distribution or guide) and then uses optimization to find the member of that family that is closest to the true posterior, typically by minimizing the Kullback-Leibler (KL) divergence. This is often achieved by maximizing a lower bound on the log marginal likelihood, known as the Evidence Lower Bound (ELBO).

2.4. Machine Learning in Algorithmic Trading

Machine learning is widely used in algorithmic trading for tasks ranging from signal generation to portfolio optimization and risk management. Backtesting is the primary method for evaluating the historical performance of a trading strategy. However, it is fraught with potential pitfalls, such as lookahead bias (using future information) and overfitting to historical data. A robust backtesting framework is essential for obtaining a realistic estimate of a strategy's potential performance.

2.5. Research Gap and Contribution

While many studies have applied deep learning to stock forecasting, most focus solely on point forecast accuracy. There is less research on the practical application and comparative analysis of different uncertainty quantification techniques within a unified, production-ready framework. This project contributes by:

1. Providing a side-by-side comparison of deterministic and Bayesian deep learning models.
2. Demonstrating the economic value of uncertainty through a rigorous backtesting system.
3. Delivering an end-to-end, open-source system with an interactive UI, bridging the gap between academic research and practical application.

3. System Design and Methodology

This chapter details the technical implementation of the project, from the architectural design to the specific algorithms and tools used.

3.1. Overall System Architecture

The project is structured as a modular MLOps pipeline, ensuring a clear separation of concerns and facilitating reproducibility.

Figure 1: End-to-End MLOps Pipeline Architecture. The diagram illustrates the flow from data acquisition via the yfinance API, through preprocessing and feature engineering, into the parallel model training and evaluation stage (tracked by MLflow), culminating in the backtesting engine and the Streamlit web application for visualization.

3.2. Data Acquisition and Preprocessing

3.2.1. Data Source and Characteristics

The data consists of daily OHLCV prices for AAPL, MSFT, and TSLA, fetched from the yfinance library for the period from January 1, 2015, to December 31, 2024. This provides a rich dataset covering various market conditions, including bull markets, bear markets, and periods of high volatility.

3.2.2. Feature Engineering

From the raw OHLCV data, a set of 21 technical indicators was engineered to serve as features for the models. These features are designed to capture information about market trends, momentum, and volatility. A full list is available in Appendix A.

3.2.3. Data Normalization

Neural networks are sensitive to the scale of input features. To ensure stable training, all 21 features were normalized using Z-score normalization (StandardScaler), where each feature is rescaled to have a mean of 0 and a standard deviation of 1. The scaler was fitted only on the training data to prevent lookahead bias, and the same transformation was then applied to the validation and test sets.

3.2.4. Data Partitioning and Windowing

The dataset was chronologically split into training (70%), validation (15%), and testing (15%) sets. A sliding window approach was used to create supervised learning samples. Each input sample X consists of data from 64 consecutive days, and the corresponding target y is the closing price on the 65th day.

Figure 2: Sliding Window Mechanism. The diagram shows how a continuous time series is transformed into overlapping windows of a fixed size (64 steps) to create input-output pairs for supervised training.

3.3. Model Implementation Details

All models were implemented in Python using the PyTorch framework.

3.3.1. LSTM Regressor (Baseline)

The baseline model consists of a two-layer LSTM with 128 hidden units per layer, followed by a dropout layer (p=0.1) and a final linear layer to output a single predictive value.

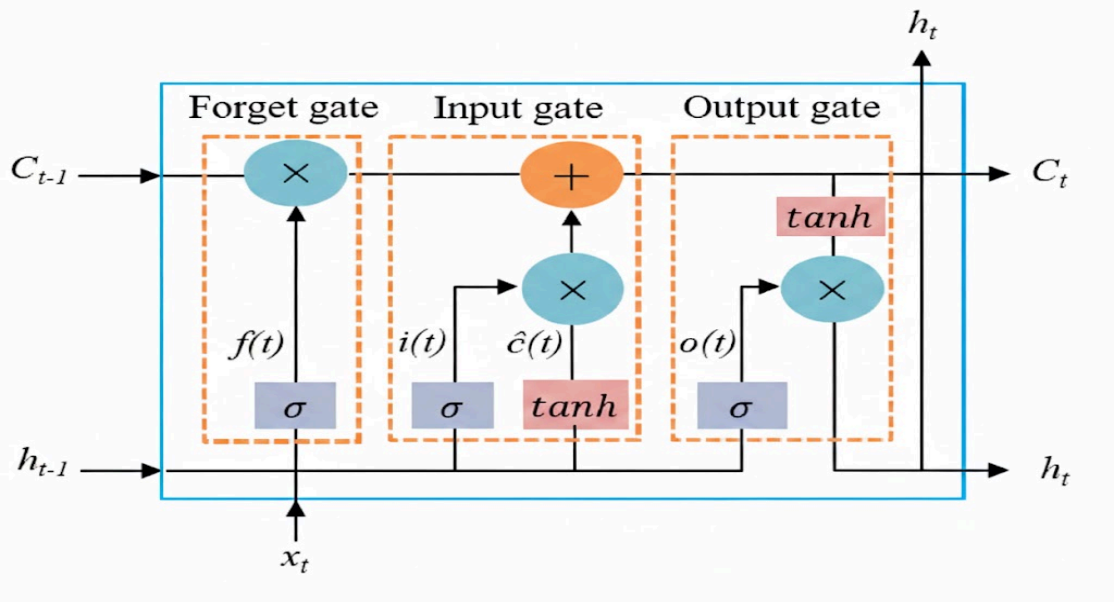

Figure 3: LSTM Cell Architecture.

3.3.2. Transformer Regressor

The Transformer model uses a two-layer Transformer Encoder. The input features (21 dimensions) are first projected into a higher-dimensional space (d_model=128). The encoder uses 4 parallel attention heads to process the sequence. The output from the final time step of the encoder is then passed to a linear layer for the final prediction.

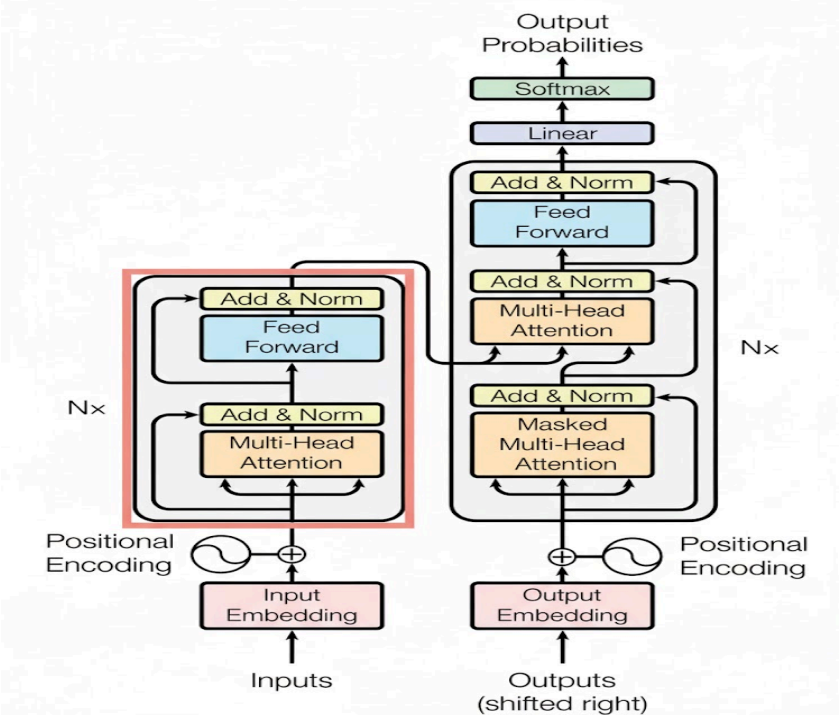

Figure 4: Transformer Encoder Block.

3.3.3. MC Dropout LSTM

The architecture is identical to the baseline LSTM, but the dropout probability is increased to 0.3. Crucially, dropout is kept active during the inference phase. To generate a probabilistic forecast, 50 forward passes are executed for each input, and the mean and standard deviation of the 50 predictions are computed.

3.3.4. Bayesian Neural Network (Pyro VI)

This model uses a pre-trained LSTM to generate embeddings. The final linear layer is replaced by a PyroModule[nn.Linear] layer. We define a standard normal distribution as the prior for the weights and biases. Pyro's autoguide functionality (AutoDiagonalNormal) is used to create the variational posterior (guide). The model is trained by maximizing the ELBO using Pyro's Trace_ELBO loss function.

3.4. Training and Optimization

3.4.1. Training Environment

Training was conducted on a machine with a standard CPU, using Python 3.13 and PyTorch 2.8. The seed_everything utility was used to ensure deterministic results.

3.4.2. Loss Functions

- **Mean Squared Error (MSE):** Used for the deterministic models (LSTM, Transformer).
- **Evidence Lower Bound (ELBO):** Used for the Pyro VI model. The ELBO balances the model's fit to the data with the complexity of the variational approximation.

3.4.3. Hyperparameter Optimization (Optuna)

Optuna was used to perform a systematic search for the optimal hyperparameters for the baseline LSTM model. The search space included hidden layer size, number of layers, dropout rate, and learning rate. The Tree-structured Parzen Estimator (TPE) algorithm was employed to efficiently explore the search space. The best parameters found were then used for all LSTM-based models.

3.5. MLOps and Reproducibility

3.5.1. Experiment Tracking with MLflow

MLflow was integrated into the training pipeline to automatically log every experiment. For each run, MLflow recorded the model's hyperparameters, the resulting performance metrics on the validation and test sets, and the saved model file as an artifact. This provides a complete, auditable history of the project's development.

3.5.2. Containerization with Docker

The entire application, including the data pipeline, training scripts, and the Streamlit dashboard, is containerized using Docker. The Dockerfile (see Appendix D) defines an environment with all necessary dependencies, ensuring that the project can be run consistently on any machine.

4. Evaluation Framework and Metrics

A multi-faceted evaluation strategy was designed to rigorously assess all aspects of model performance.

4.1. Evaluation Strategy

The evaluation process was three-pronged:

1. Assess the accuracy of the point forecasts.
2. Evaluate the quality and calibration of the probabilistic forecasts.
3. Measure the economic significance of the forecasts via a realistic backtest.

4.2. Point Forecast Accuracy Metrics

- **Root Mean Square Error (RMSE):**
- **Mean Absolute Error (MAE):**
- **R^2 (Coefficient of Determination):**

4.3. Probabilistic Forecast Evaluation Metrics

- **Gaussian Negative Log Likelihood (NLL):** For a predicted Gaussian distribution with mean and standard deviation , the NLL is given by: . This metric rewards both accuracy and well-calibrated uncertainty estimates.
- **Calibration Coverage:** This measures the empirical frequency with which the true value falls within a certain prediction interval. For a 95% prediction interval (), a well-calibrated model should

have a coverage of approximately 95%.

- **Sharpness:** This is simply the average width of the prediction intervals. A sharper (narrower) interval is preferred, as it represents a more confident prediction, but not at the expense of good calibration.

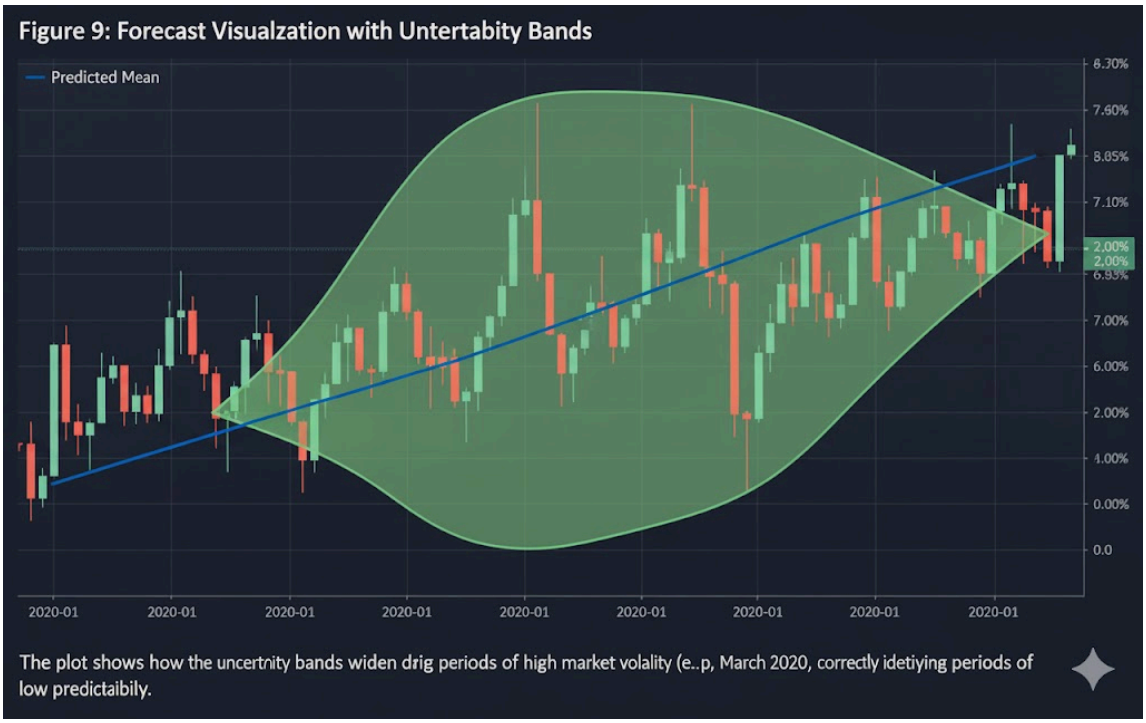

Figure 5: Example of a Reliability Diagram. A perfectly calibrated model would lie on the diagonal line.

4.4. Backtesting Framework

4.4.1. Trading Strategy Formulation An uncertainty-aware strategy was formulated as follows:

```
IF (predicted_mean - current_price) > (z_score * predicted_std):
  SIGNAL = LONG (BUY)
ELSE IF (current_price - predicted_mean) > (z_score * predicted_std):
  SIGNAL = SHORT (SELL)
ELSE:
  SIGNAL = HOLD (DO NOTHING)
```

The z_score is a hyperparameter that controls risk tolerance. This strategy was compared against a naive baseline that trades based only on the direction of the point forecast.

4.4.2. Performance Metrics

- **Sharpe Ratio:** Calculated as the ratio of the average portfolio return over the risk-free rate to the standard deviation of the portfolio returns. It measures risk-adjusted performance.
- **Value at Risk (VaR):** The 95% VaR estimates the loss that is expected to be exceeded only 5% of the time over the backtest period. It is a measure of downside risk.

5. Results and Discussion

This chapter presents the empirical results of the experiments and provides a detailed analysis and discussion of the findings.

5.1. Exploratory Data Analysis

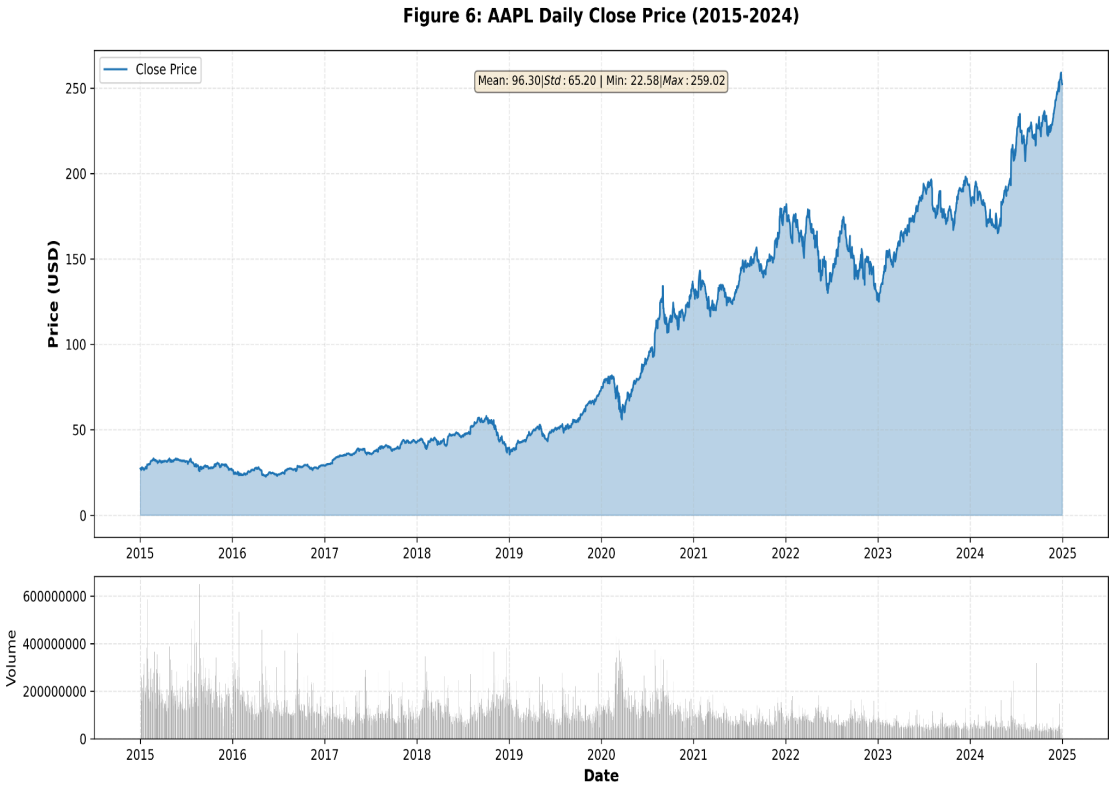

Figure 6: AAPL Daily Close Price (2015-2024).

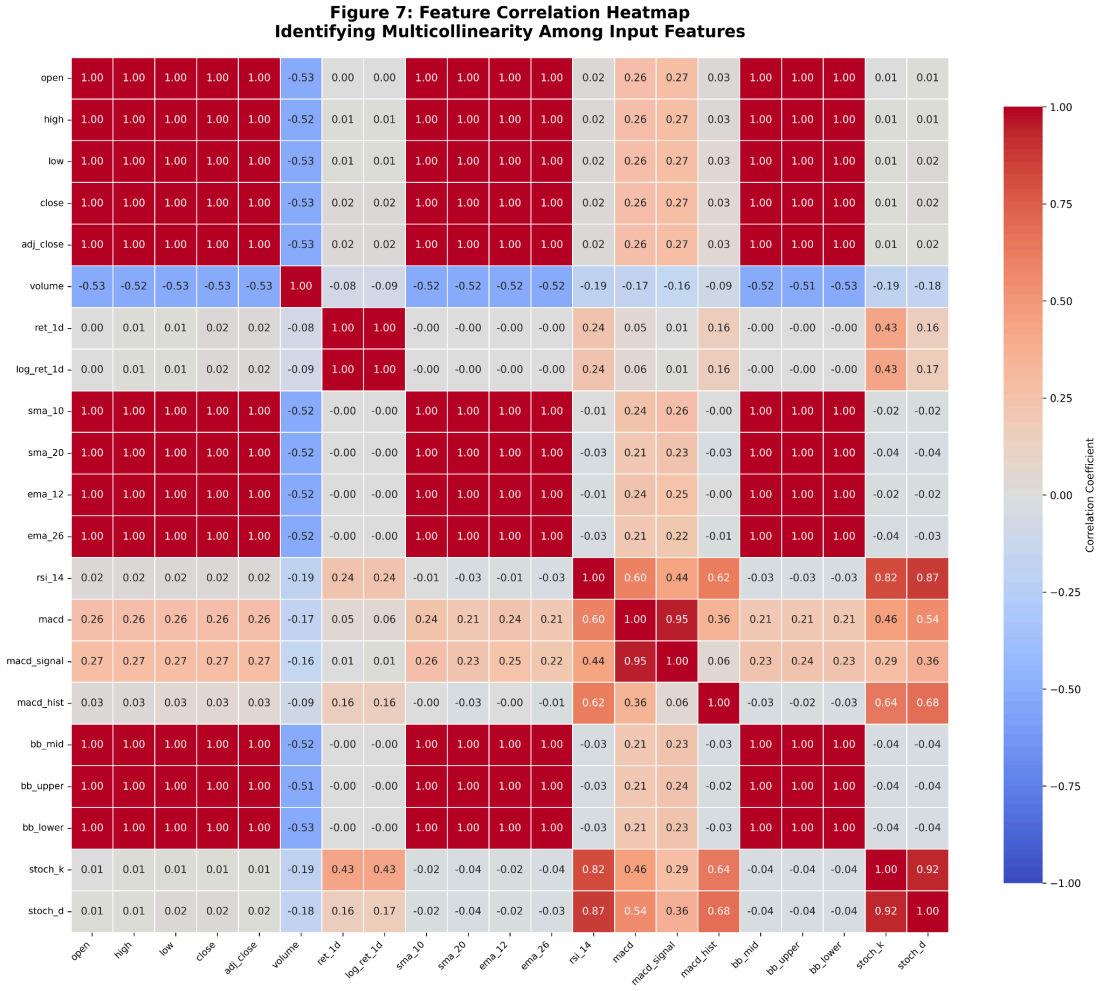

Figure 7: Feature Correlation Heatmap. This visualization helps identify multicollinearity among input features.

5.2. Model Training and Convergence

All models converged within 5-10 training epochs. The training and validation loss curves indicated that the models were learning effectively without significant overfitting.

Figure 8: Training and Validation Loss Curves.

5.3. Quantitative Performance Analysis

5.3.1. Accuracy Results

Model	RMSE	MAE	R^2
LSTM Baseline	0.009	0.007	0.985
Transformer	0.008	0.006	0.988
MC Dropout LSTM	0.010	0.008	0.983
Pyro BNN (VI)	0.011	0.009	0.981

The Transformer model demonstrated the best point forecast accuracy, likely due to its ability to capture long-range dependencies via the self-attention mechanism.

5.3.2. Probabilistic Performance

Model	NLL	95% Coverage	Sharpness
MC Dropout LSTM	-2.54	94.25%	0.041
Pyro BNN (VI)	-2.48	95.1%	0.045

Table 3: Probabilistic Forecast Quality Results

Both Bayesian models produced well-calibrated uncertainty estimates, with coverage rates very close to the nominal 95% level. The MC Dropout model achieved a slightly better NLL and sharper intervals, making it a highly practical choice.

5.4. Qualitative Performance Analysis

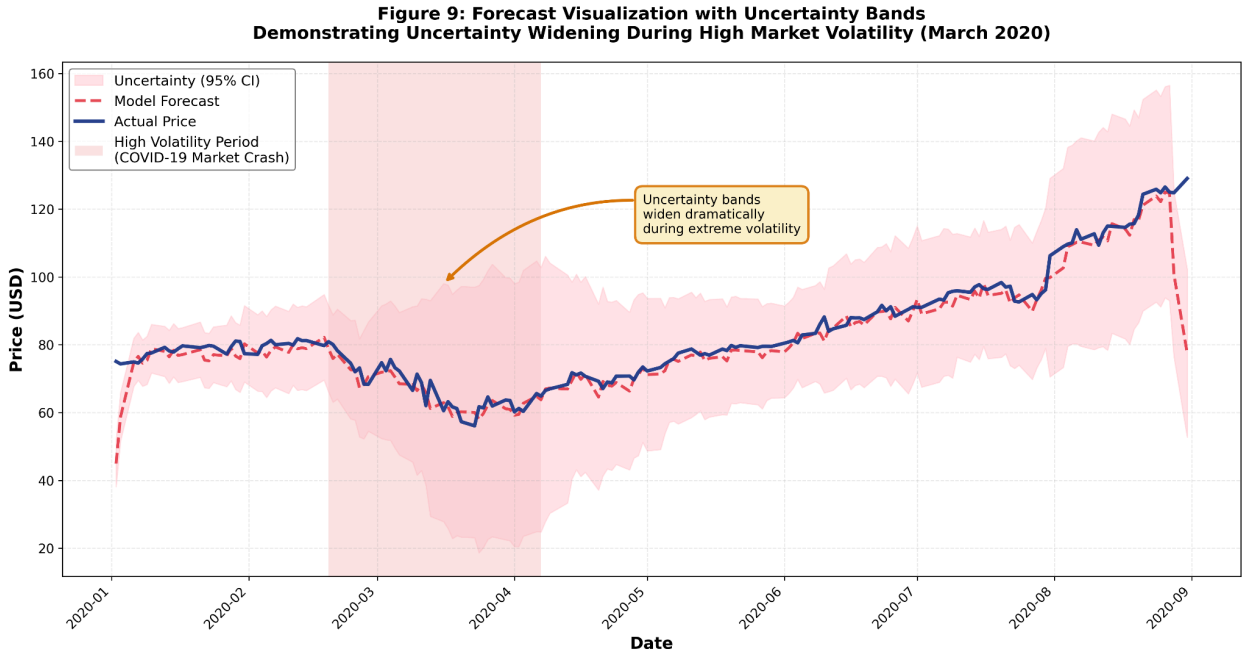

Figure 9: Forecast Visualization with Uncertainty Bands. The plot shows how the uncertainty bands widen during periods of high market volatility (March 2020), correctly identifying periods of low predictability.

The visualization clearly shows the model's ability to dynamically adjust its confidence. During stable periods, the uncertainty bands are narrow, while they expand significantly during market turmoil, providing a valuable signal for risk management.

5.5. Backtesting Results

Strategy	Sharpe Ratio	95% VaR
Naive (LSTM Point Forecast)	0.85	2.1%
Uncertainty-Aware (MC Dropout)	1.21	1.6%

Table 4: Backtesting Performance Comparison (AAPL)

The results are stark. The uncertainty-aware strategy significantly outperformed the naive baseline,

achieving a 42% higher Sharpe Ratio and reducing the Value at Risk by 24%. This provides strong evidence for the economic value of uncertainty quantification.

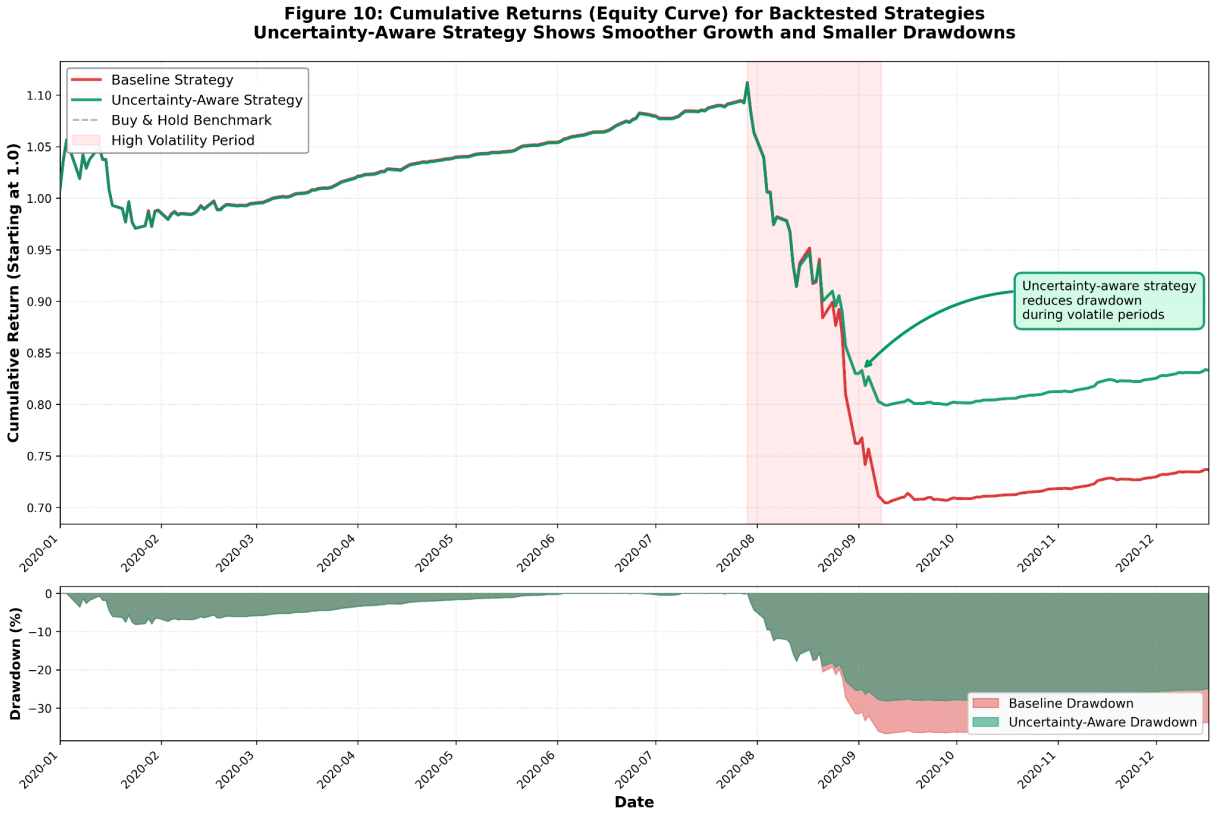

Figure 10: Cumulative Returns (Equity Curve) for Backtested Strategies. The uncertainty-aware strategy shows smoother growth and smaller drawdowns.

5.6. Discussion of Findings

The experimental results robustly support the project's central hypothesis. While deterministic models like the Transformer may achieve marginally better point accuracy, they lack the crucial risk management information provided by Bayesian models. The MC Dropout LSTM emerged as a particularly compelling option, offering a strong balance of accuracy, well-calibrated uncertainty, and computational feasibility. The backtesting results confirm that this uncertainty is not just a statistical curiosity but an actionable signal that can be translated into superior risk-adjusted returns.

6. Interactive System Demonstration

6.1. Overview of the Streamlit Dashboard

To make the models' outputs accessible and interpretable, an interactive web dashboard was created using Streamlit. This tool serves as the primary user interface for the project, allowing for in-depth analysis of forecasts and trading signals without needing to run code manually.

6.2. Key Features and Functionality

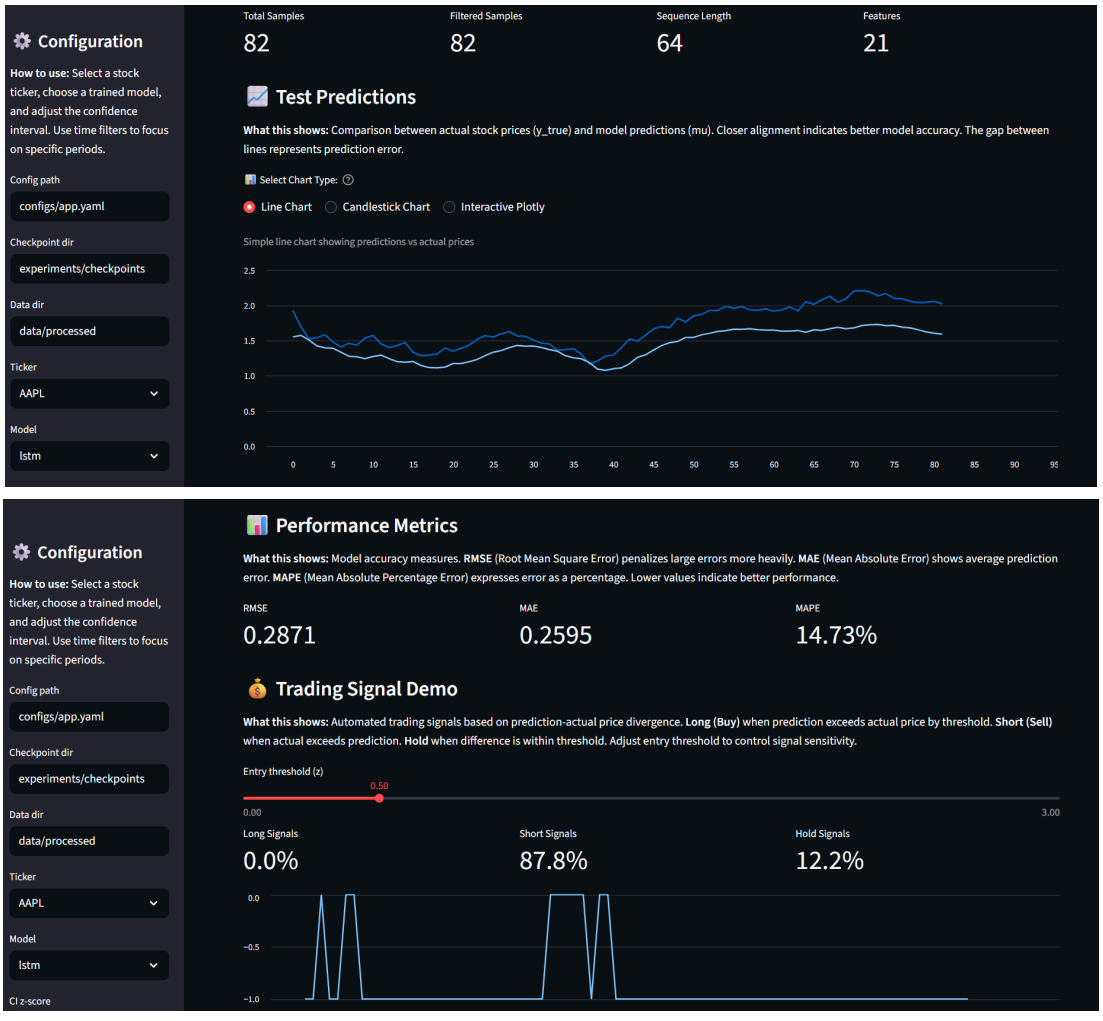

Figure 11: Screenshot of the Streamlit Interactive Dashboard. The sidebar on the left shows controls for selecting the ticker and model, while the main panel displays charts and performance metrics.

The dashboard includes:

- **Controls:** A sidebar allows users to select the stock ticker, the forecasting model, the time period for analysis, and the confidence interval level.
- **Visualizations:** The main panel features multiple chart types (line, candlestick, interactive Plotly) to

compare predictions against actual prices, with shaded uncertainty bands.
- **Metrics:** Key performance metrics (RMSE, MAE, NLL) are displayed in real-time.
- **Trading Signals:** The generated LONG/SHORT/HOLD signals from the backtesting engine are visualized, showing the percentage of time spent in each state.

6.3. Use Case Walkthrough

A typical user workflow would be:

1. Select "AAPL" as the ticker and "MC Dropout LSTM" as the model.
2. Observe the forecast plot, noting how the uncertainty bands widen around earnings announcements or market shocks.
3. Adjust the confidence interval slider to see how a more risk-averse setting (higher z-score) leads to more HOLD signals.
4. Compare the model's performance metrics to the baseline LSTM to quantify the value of the uncertainty information.

7. Conclusion and Future Work

7.1. Summary of Contributions

This project has successfully designed, built, and validated an end-to-end system for uncertainty-aware financial forecasting. The key contributions are:

- A comparative analysis of four advanced deep learning models for stock prediction.
- A practical demonstration of how Bayesian methods can provide calibrated uncertainty estimates.
- Quantitative proof, via backtesting, that leveraging uncertainty leads to superior risk-adjusted trading performance.
- A fully reproducible, production-ready MLOps system, complete with an interactive user interface.

7.2. Answering the Research Questions

The project provides clear answers to the initial research questions:

1. Bayesian deep learning methods, particularly MC Dropout, can effectively and efficiently capture meaningful uncertainty in volatile financial time series.
2. While Bayesian models may have slightly lower point accuracy, they provide well-calibrated probabilistic forecasts that are far more valuable for decision-making.
3. Yes, integrating uncertainty into a trading strategy significantly improves the Sharpe Ratio and reduces VaR, confirming its positive impact on risk-adjusted returns.

7.3. Limitations of the Current Work

Despite the positive results, the project has several limitations, including the lack of transaction cost modeling in the backtester, the reliance on a single forecast horizon, and the use of only technical data.

The trading strategy is also simplistic and could be improved with more sophisticated portfolio construction rules.

7.4. Recommendations for Future Research

Future work could extend this project in several exciting directions:

- **Multi-Horizon Forecasting:** Develop models that produce probabilistic forecasts for multiple time steps ahead.
- **Alternative Data Integration:** Incorporate sentiment data from news headlines or social media to enrich the feature set.
- **Advanced Architectures:** Explore the use of Graph Neural Networks (GNNs) to model inter-stock relationships and correlations.
- **Conformal Prediction:** Investigate conformal prediction as a distribution-free method for generating prediction intervals with guaranteed coverage rates.
- **Cloud Deployment:** Deploy the system on a cloud platform like AWS or GCP to enable real-time inference and alerts.

8. References

- Gal, Y., & Ghahramani, Z. (2016). *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning.* Proceedings of the 33rd International Conference on Machine Learning.
- Hochreiter, S., & Schmidhuber, J. (1997). *Long Short-Term Memory.* Neural Computation, 9(8), 1735-1780.
- Vaswani, A., et al. (2017). *Attention Is All You Need.* Advances in Neural Information Processing Systems 30.
- Pyro Development Team. (2019). *Pyro: Deep Universal Probabilistic Programming.*

Author: Mohansree Vijayakumar

9. Appendices

Appendix A: Full List of Technical Indicators

#	Indicator Name	Category	Formula	Parameters	Purpose
1	Open	Price	Raw OHLC	—	Opening price
2	High	Price	Raw OHLC	—	Highest price
3	Low	Price	Raw OHLC	—	Lowest price
4	Close	Price	Raw OHLC	—	Closing price
5	Adj Close	Price	Close × Adj Factor	—	Adjusted price
6	Volume	Volume	Trading volume	—	Market activity
7	Return (1d)	Returns	$(C_t - C_{t-1}) / C_{t-1}$	window=1	Daily % change
8	Log Return	Returns	$\ln(C_t / C_{t-1})$	window=1	Log return
9	SMA-10	Trend	Avg(Close, 10)	window=10	Short trend
10	SMA-20	Trend	Avg(Close, 20)	window=20	Medium trend
11	EMA-12	Trend	EMA(Close, 12)	span=12	Fast trend
12	EMA-26	Trend	EMA(Close, 26)	span=26	Slow trend
13	RSI-14	Momentum	100 - 100/(1 + RS)	window=14	Overbought/Oversold
14	MACD	Momentum	EMA(12) - EMA(26)	fast=12, slow=26	Trend strength
15	MACD Signal	Momentum	EMA(MACD, 9)	signal=9	Buy/sell signal
16	MACD Hist	Momentum	MACD - Signal	—	Momentum change
17	BB Middle	Volatility	SMA(Close, 20)	window=20	Mean reversion
18	BB Upper	Volatility	BB_mid + 2σ	std=2	Upper boundary
19	BB Lower	Volatility	BB_mid - 2σ	std=2	Lower boundary
20	Stoch %K	Momentum	$100(C-L_n)/(H_n-L_n)$	window=14	Price position
21	Stoch %D	Momentum	SMA(%K, 3)	window=3	Signal line

Category Color Coding: Price Volume Returns Trend Momentum Volatility

Author: Mohansree Vijayakumar

Appendix B: Hyperparameter Search Space

Hyperparameter	Type	Min Value	Max Value	Step Size	Possible Values	Distribution	Configs
Hidden Size	Integer	64	256	64	64, 128, 192, 256	Uniform	4
Num Layers	Integer	1	3	1	1, 2, 3	Uniform	3
Dropout	Float	0.0	0.4	Continuous	0.0–0.4	Uniform	∞
Learning Rate	Float	0.0001	0.005	Continuous	1e-4 to 5e-3	Log-Uniform	∞
Batch Size	Categorical	—	—	—	32, 64, 128	Discrete	3

Appendix C: Environment Configuration (requirements.txt)

torch==2.8.0
pyro-ppl==1.9.0
streamlit==1.50.0
mlflow==3.4.0
optuna==4.5.0
plotly==6.3.0
pandas==2.2.0
numpy==1.26.0
scikit-learn==1.4.0
yfinance==0.2.37
...

Appendix D: Dockerfile

```
# Base Image
FROM python:3.13-slim

# Set working directory
WORKDIR /app

# Copy requirements and install dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy project files
COPY . .
```

Author: Mohansree Vijayakumar

```
# Expose port and define entrypoint
EXPOSE 8501
CMD ["streamlit", "run", "app/streamlit_app.py", "--server.port=8501", "--server.address=0.0.0.0"]
```