**Report**

**Stock Closing Price Prediction Using Bayesian Analysis**

**(Group #16)**

***Submitted by***

Sohan Bachuwar (811236035)

Rohith Reddy Poreddy (811254545)

Sathwika Reddy Thipparthi (811255702)

Sai Vamsi Dudigam (811226902)

Venkata Raghava Sanjay Varma Dantuluri (811255424)

**Table of Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No |  | Content | Page No. |
|  |  | Abstract | 3 |
| 1 |  | Introduction | 4 |
| 2 |  | Project Description | 5 |
| 3 |  | Background | 6 |
|  | 3.1 | Software requirements | 7 |
|  | 3.2 | Data Processing Libraries | 8 |
|  | 3.3 | Visualization Libraries | 9 |
|  | 3.4 | Machine Learning Libraries | 10 |
|  | 3.5 | Hardware requirements | 11 |
| 4 |  | Problem Definition | 12 |
|  | 4.1 | Challenges | 13 |
| 5 |  | The Proposed Techniques | 14 |
|  | 5.1 | Approach | 15 |
|  | 5.2 | Dataset | 16-19 |
|  | 5.3 | Implementation | 20-31 |
|  | 5.4 | Comparison between One Step Prediction vs N Step Prediction | 32 |
| 6 |  | Results | 33-34 |
| 7 |  | Conclusion | 35 |

**Abstract**

Stock market prediction is an important part of financial decision-making, having a broad impact on investors and traders. This study focuses on stock closing price prediction using Bayesian analysis, a complex statistical method recognized for its ability to deal with uncertainty. The project's main goal is to use the Bayesian framework to estimate closing prices, which represent a stock's worth at the end of a trading day. In the unpredictable field of finance, Bayesian analysis provides a reliable technique for capturing and evaluating uncertainty in parameter estimates and, hence, forecasts. Within the context of stock market prediction, this study will investigate the fundamentals of Bayesian analysis, such as prior probabilities, likelihood functions, and posterior distributions. In addition, we will investigate data collection and preprocessing approaches, feature selection, and model assessment methodologies that are specific to this area. The project will conclude with the creation and testing of a Bayesian model for forecasting stock closing prices and improving investing decisions by addressing the complex and unpredictable character of financial markets.

**1. Introduction**

This project constitutes a comprehensive exploration into the pivotal realm of stock market prediction, a critical concern for both seasoned financial experts and investors grappling with the nuanced intricacies of financial markets. The ability to accurately anticipate forthcoming stock prices stands as a cornerstone, influencing a diverse array of financial investment decisions. The central focus of this endeavor is to harness the predictive capabilities of Bayesian analysis in estimating closing values—a reflection of a stock's ultimate valuation at the close of a trading day. These closing prices wield considerable sway over investor strategies and decisions.

Given the inherent unpredictability and the substantial financial ramifications associated with stock market predictions, the strategic choice in this project is to employ Bayesian analysis. This method, renowned for its efficacy, is adept at capturing and quantifying the inherent uncertainty linked to parameter estimates. The deliberate integration of Bayesian analysis in this project serves as a robust framework, offering guidance for investment decisions within the dynamic and intricate landscape of stock markets.

The ultimate aspiration of this project extends beyond a mere exploration; it endeavors to deliver valuable insights, sophisticated tools, and innovative methodologies. The overarching aim is to refine the precision and dependability of stock market predictions. Through these efforts, the project seeks to empower investors and financial decision-makers, enabling them to navigate and thrive amidst the multifaceted challenges posed by the ever-evolving and intricate environment of financial markets.

**2. Project Description**

The project embarks on a comprehensive exploration of Amazon's stock price dynamics, spanning the period from January 1, 2014, to December 6, 2019. A meticulous approach is taken in collecting and analyzing daily stock price data, leveraging the "get\_data\_yahoo" function from the Pandas DataReader package. This function facilitates the extraction of stock data from Yahoo Finance, with a specific emphasis on capturing the nuances of daily closing prices.

Recognizing the temporal nature of the dataset, a sophisticated time series analysis is employed to dissect it into three fundamental components: trend, seasonality, and residual. The "trend" component serves as a powerful diagnostic tool, enabling the discernment of long-term trajectories in Amazon's stock values—whether they exhibit upward, downward, or stable trends. Simultaneously, the "seasonality" component unveils recurring patterns influenced by time-related elements, including seasonal variations and cyclic trends.Upon accounting for trends and seasonality, the "residual" component captures unexplained fluctuations, providing insights into the unpredictable dynamics of Amazon's stock values. This multifaceted analysis not only offers a granular understanding of the complex behavior exhibited by Amazon's stocks over time but also serves as a foundation for uncovering underlying patterns and behaviors. Such insights are invaluable for investors and decision-makers seeking a nuanced comprehension of the factors influencing Amazon's stock performance. Ultimately, the project aims to provide actionable intelligence that can inform strategic decision-making in the dynamic landscape of financial markets.

**3. Background**

In the realm of time series data analysis, a critical consideration involves determining the optimal number of past values that contribute significantly to predicting the current value. This necessitates the utilization of partial autocorrelation plots, which unveil the correlation between different time lags and the present value in a time series. Upon careful analysis of these plots, a notable finding emerges: the previous day's value exhibits a substantial correlation with the current day's estimation compared to other historical values. This observation implies that a lag value of one day is deemed appropriate for effectively estimating closing prices.

To validate the suitability of the one-day lag in closing price estimation, a Dickey-Fuller test is employed on the daily percentage change of the closing price. A pivotal aspect of this test lies in examining the p-value it generates; if this p-value falls below the conventional significance threshold of 0.05, it indicates that the percentage change time series is stationary. This stationary behavior implies that the majority of time-varying components in the series are adequately encapsulated within the one-day lag, and the residual component is rendered independent of time.

In essence, this background sets the stage for a refined approach to time series analysis, emphasizing the critical role of partial autocorrelation plots and the Dickey-Fuller test in determining the optimal lag for closing price estimation. The findings suggest that a one-day lag effectively captures the relevant information for predicting closing prices, offering a methodologically sound foundation for subsequent analyses and forecasting in the dynamic landscape of financial markets.

**3.1 Software requirements**

The software requirements for the project are as follows:

1. Operating System: Microsoft Windows 10

2. Integrated Development Environment (IDE): Jupyter Notebook

3. Programming Language: Python

Various Python libraries are used in this entire project.

**3.2 Data Processing Libraries**

**Pandas:** Pandas is an open-source Python library widely used for data manipulation and analysis. It provides powerful data structures like Data Frames and Series, making it easy to work with structured data. With Pandas, users can efficiently handle and clean datasets, perform various operations, and conduct exploratory data analysis. The library supports importing data from various formats, including CSV, Excel, and databases and facilitates data filtering, aggregation, and transformation. Pandas also offers flexible data visualization capabilities, enabling users to create insightful plots and charts. Due to its intuitive syntax and rich functionality, Pandas has become a go-to tool for data scientists, analysts, and researchers to work with tabular data in Python.

**NumPy:** NumPy is a powerful Python library for numerical computing that provides support for large, multi-dimensional arrays and matrices, along with an extensive collection of mathematical functions to operate on these arrays efficiently. It is widely used in data analysis, scientific computing, and machine learning applications. NumPy's core functionality includes array manipulation, element-wise operations, broadcasting, and linear algebra routines. Its fast and memory-efficient operations make it an essential tool for handling and processing large datasets. NumPy serves as the foundation for many other scientific Python libraries, allowing seamless integration with tools like pandas and sci-kit-learn, making it an indispensable asset for any data scientist or researcher.

**3.3 Visualization Libraries**

**Seaborn:** Seaborn is a powerful Python data visualization library built on top of Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics, making it easy to visualize complex datasets concisely. With just a few lines of code, Seaborn can generate visually appealing plots such as scatter plots, bar plots, histograms, box plots, heat maps, and more. It offers a wide range of customization options, allowing users to control colors, styles, and aesthetics. Seaborn's integration with Panda’s data structures makes it particularly convenient for data analysis tasks. Overall, Seaborn is an essential tool for data scientists and analysts to effectively communicate insights from their data.

**Matplotlib:** Matplotlib is a popular Python library for creating high-quality visualizations and plots. With an intuitive and flexible API, it allows users to generate a wide range of charts, graphs, and 2D plots, making it an essential tool for data analysis, scientific research, and data visualization tasks. Matplotlib's functionality includes line plots, scatter plots, bar charts, histograms, pie charts, and more, all customizable to suit specific needs. It provides precise control over plot elements, such as axes, labels, colors, and styles, ensuring visually appealing outputs. Whether for exploring data, communicating insights, or presenting findings, Matplotlib empowers users to create compelling visual representations of their data effortlessly.

* 1. **Machine Learning Libraries**

**Tensorflow:** TensorFlow is an open-source machine learning library developed by the Google Brain team. It provides a comprehensive ecosystem of tools, libraries, and community resources that facilitate the development and deployment of machine learning models. TensorFlow is particularly popular for its flexibility, scalability, and support for deep learning.

At its core, TensorFlow is designed to facilitate the creation and training of machine learning models, especially neural networks, for various tasks such as image recognition, natural language processing, and predictive analytics. The library uses a data flow graph to represent computational processes, allowing developers to define complex models and their relationships.

One of TensorFlow's key strengths is its compatibility with both CPUs and GPUs, enabling efficient computation for training and inference. It also supports distributed computing, allowing developers to scale their models across multiple devices or servers. TensorFlow's versatility extends to its compatibility with various platforms, making it suitable for deployment in diverse environments, including mobile devices and embedded systems.

The library supports a high-level API known as Keras, which simplifies the process of building and training neural networks. This abstraction layer makes it accessible for both beginners and experienced machine learning practitioners. TensorFlow 2.0 and later versions have an eager execution mode, which allows for immediate iteration and debugging, making the development process more intuitive.

TensorFlow boasts a vibrant and supportive community, contributing to an extensive repository of pre-trained models and resources. This accelerates development by providing access to state-of-the-art models that can be fine-tuned for specific tasks.

**Scikit-learn:** Scikit-learn is a popular and user-friendly machine-learning library for Python. With a comprehensive collection of efficient tools, it simplifies the implementation of various machine learning algorithms, including classification, regression, clustering, and more. Scikit-learn is built on NumPy and SciPy and provides a consistent interface for training and evaluating models. It supports data preprocessing, feature selection, and model evaluation. The library also offers capabilities for model selection, hyperparameter tuning, and cross-validation. Widely used in academia and industry, Scikit-learn empowers researchers and developers to build powerful machine-learning solutions with ease, making it a cornerstone of the Python data science ecosystem.

**3.5 Hardware requirements**

The hardware requirements for the project are as follows:

1. Processor: Intel 3rd generation processor

2. RAM: 4GB

3. Hard Disk Space: 100GB

These specifications are recommended for the laptop or PC used in the project.

**4. Problem Definition**

In the pursuit of robust stock market prediction, this project adopts a sophisticated approach by leveraging a Structural Time Series (STS) model—a versatile and probabilistic framework extensively applied in time-series research. The STS model stands out for its intrinsic capability to dissect observable time series data into distinct components, thereby facilitating a more nuanced comprehension of underlying patterns and dynamics.

The components embedded within the STS model are multifaceted and encompass autoregression, which factors in the influence of preceding data points on current observations; moving averages, designed to smooth out short-term fluctuations; local linear trends, capturing nuanced linear trends within the time series; seasonality, discerning recurring patterns linked to time-related factors; and regression, allowing for the incorporation of external variables that may exert influence on the time series.

Through the application of the STS model, the project endeavors to unravel the intricate interplay among these components. By doing so, it seeks to harness their collective insights to generate more precise forecasts of stock prices. This methodological approach takes into consideration a broad spectrum of factors contributing to stock price variations, culminating in a comprehensive and data-driven strategy for stock market prediction.

In essence, the problem definition centers on harnessing the capabilities of the STS model to discern the complex dynamics within time series data. The aim is to utilize this enhanced understanding to significantly improve the accuracy of stock price forecasts, providing a holistic and sophisticated solution to the inherent challenges of stock market prediction.

**4.1 Challenges**

1. Model Complexity and Computational Resources:

Bayesian models can be computationally intensive, particularly as the complexity of the model increases. Complex models may require significant computing resources, and running Bayesian inference might be time-consuming, especially for large datasets.

2. Prior Specification:

The choice of priors in Bayesian analysis is crucial. The selection of informative or non-informative priors can significantly impact the model's results. Determining appropriate priors, especially in the absence of strong prior knowledge, can be challenging.

3. Non-Stationarity of Financial Time Series:

Financial time series often exhibit non-stationary behavior, with statistical properties changing over time. Adapting Bayesian models to capture such non-stationarity and allowing for flexibility in the model structure is a challenge.

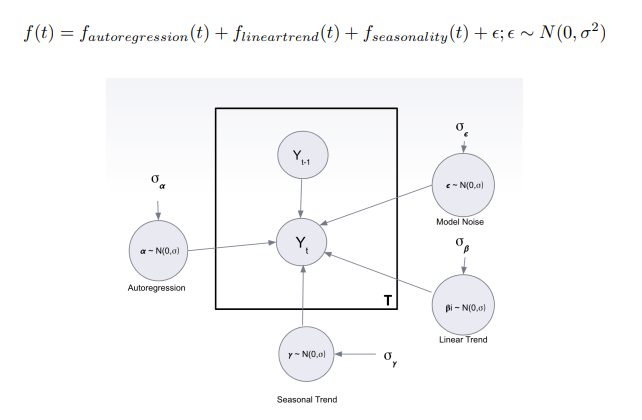
4. Feature Selection and Dimensionality:

Selecting relevant features is crucial for any predictive model. Bayesian models may face challenges in handling a large number of features or in selecting the most informative ones. Careful feature engineering and model selection are necessary.

**5. The Proposed Techniques**

**5.1** **Approach**

Structural time series (STS) are a family of probability models that include many standard time-series modeling ideas, such as autoregression, moving averages, local linear trends, seasonality, and regression. The STS model built in this project expresses an observed time series as sum of following simpler components:



A black text on a white background

Description automatically generated

The individual components are time series governed by a particular structural assumption. One component encodes a day-of-week seasonal effect, another a local linear trend, and the third one is an autoregressive component to model any unexplained residual effects. A simple random walk could have been used, but an autoregressive component was chosen because it maintains bounded variance over time. After deciding the model components, joint distribution of the data and parameters is optimized using variational inference with mean field approximation.

The joint distribution of the parameters is decomposed into product of the distribution of individual components.

q(z) = q(β1) · q(β2) · q(α) · q(γ) · q())

The optimal solution is given as:

log q ∗ j (Z|) = Ei6=j [log p (D, Z)] + constant

The distribution of the individual parameters is assumed to be normal with zero mean. Also the priors used on the parameters are normal and the value of the priors is decided by the TensorFlow probability library. A couple of different priors were tried based on heuristics and prior knowledge of stock market, but the default priors performed best for prediction. A loss function was defined as negative of ELBO which was optimized using stochastic gradient optimization technique, Adam optimizer. After 50 iterations the algorithm converged, and the distribution of the parameters were obtained.

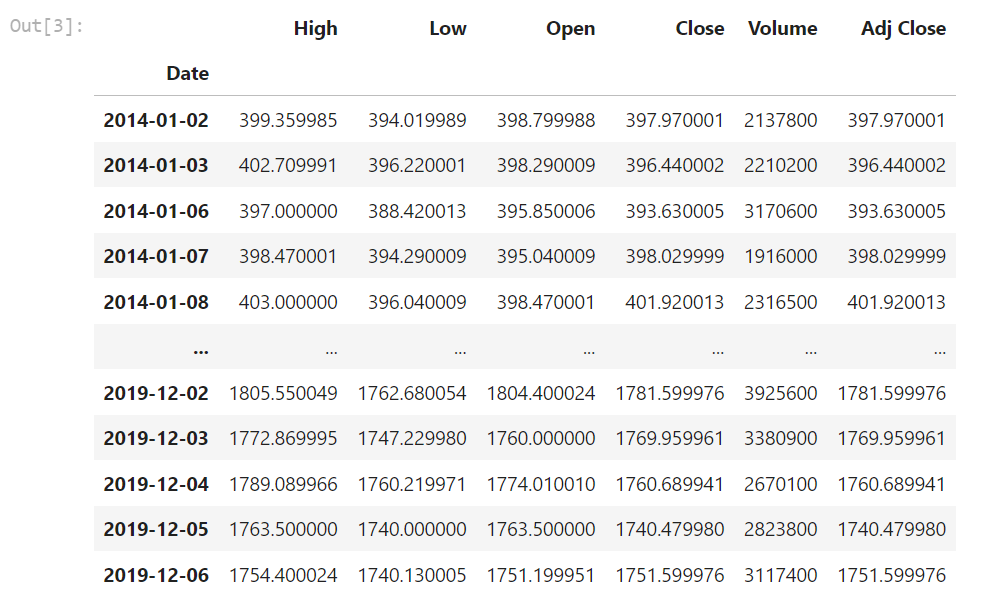
A graph of a person with a graph

Description automatically generated with medium confidence

**5.2 Dataset**

The daily stock price data of Amazon for five years period starting from 1st January 2014 till 6th December 2019 is collected. Data is extracted using get data yahoo method in the pandas data reader package which downloads the stocks data from yahoo finance for a given company and timeframe.

**Sample:**

****

**Description:**

* **Open** is the price of the stock at the beginning of the trading day (it need not be the closing price of the previous trading day).
* **High** is the highest price of the stock on that trading day.
* **Low** is the lowest price of the stock on that trading day.
* **Close** is the price of the stock at closing time. The closing price is the 'raw' price which is just the cash value of the last transacted price before the market closes.
* **Volume** indicates how many stocks were traded.
* **Adj Close** is the updated stock closing price that accurately reflects the stock's value after accounting for any corporate actions. It is the true price of that stock and is often used when examining historical returns or performing a detailed analysis of historical returns.

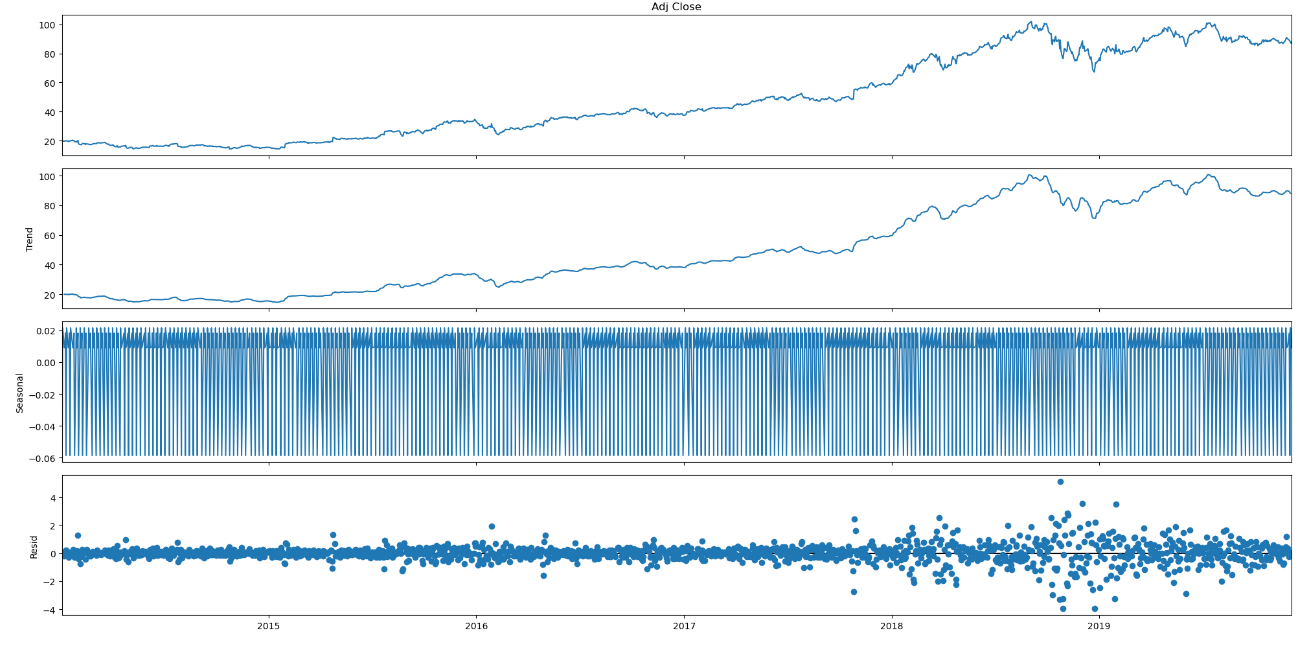
Since the data being considered in this project is the daily closing price of stocks, it can be considered a time series. A time series can be decomposed into these 3 majors components.

• Trend - Trend tries to capture the slope of the series. It mimics the upward and downward slope of the observed values.

A graph showing a line

Description automatically generated

• Seasonality - Seasonality is the constant factor present in the series which is repeated after certain interval of time.



• Residual - Whatever part of the time series is left after removing trend and seasonality is the residual, which is basically random variation.

**5.3 Implementation**

Step 1: Importing all the required packages.

**A screen shot of a computer code

Description automatically generated**

The imported Python libraries, which include NumPy, Pandas, Matplotlib, TensorFlow, and Statsmodels, indicate that the code is most likely being used for data science or machine learning tasks. The libraries like Pandas-datareader and Yfinance are used for reading financial data.

Step 2: Load Dataset

A screenshot of a computer code

Description automatically generated

The Python code given downloads historical stock data from Yahoo Finance using the yfinance module. The code obtains data for a certain stock, such as Amazon (AMZN) or Google (GOOGL), by specifying start and end dates.

Step 3: Exploratory Analysis on Amazon Stock Data.

A screenshot of a computer

Description automatically generatedA graph of a graph

Description automatically generated with medium confidence

In the above candlestick plots, green candlestick indicates a day where the closing price was higher than the open (gain +ve), red candlestick indicates a day where the open was higher than the close (loss -ve).



A graph showing a line

Description automatically generated with medium confidence

Amazon had a relatively consistent growth until 2018, followed by an increase in the closing prices mid 2018, which was then followed by a dip around year end, a slight increase and then another dip. It seems like these stock closing prices might be following mean reversion.

A screenshot of a computer

Description automatically generated

A graph showing a sound wave

Description automatically generated

The relative change in the daily stock prices seems to follow a white noise (stationary distribution). Conducting an augmented dickey-fullter test on this series can confirm if we can assume this is a stationary distribution.

Step 4: Seasonality trends

A screenshot of a graph

Description automatically generated

The above given graph shows the seasonality trend from the year 2014 to 2019.

Step 5: Model

Time series model which includes local linear trend, weekly seasonality and autoregressive models.

A screenshot of a computer program

Description automatically generated

To fit the model to data, we define a surrogate posterior.

A computer screen shot of a computer code

Description automatically generated

We can fit it by minimizing the negative variational evidence lower bound (ELBO).

A close-up of a computer code

Description automatically generated

Forecast for `num\_forecast\_steps` days.

A screenshot of a computer code

Description automatically generated

Forecast the next 20 days using one-step-prediction which will give predictive distribution over observations at each time T, given observations up through time T-1.

A screen shot of a computer code

Description automatically generated

Plot Forecast

A screen shot of a computer

Description automatically generated

Extract inferred value of model Parameters.

A screenshot of a computer program

Description automatically generated

Extract training data for predicting 20 days forecast.

A close-up of a computer code

Description automatically generated

Model Params:

A screenshot of a computer

Description automatically generated

Step 6: N Step Forecast:

A screenshot of a computer code

Description automatically generated

Forecast Estimates and their uncertainty.

A screen shot of a computer code

Description automatically generated

A table of numbers and a few digits

Description automatically generated with medium confidence

Step 7: One Step Forecast

One step Forecast Estimates and their uncertainty.

A screenshot of a computer program

Description automatically generated

A close-up of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

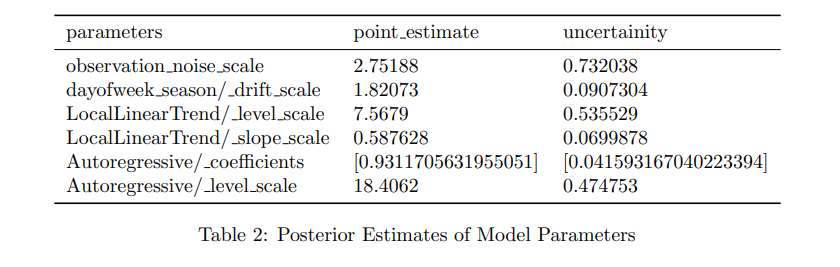
**5.4** **Comparison between One Step Prediction vs N Step Prediction**

A screenshot of a computer

Description automatically generated

**6. Results**

Using mean-field variational approximation we obtain the following posterior distribution for each of the parameters:



1000 samples are generated from the posterior distribution of the parameters and further were used to get the predictive distribution of the closing price of stocks over the next 20 days. This forecasting strategy models the observed time series as a Gaussian state space model and uses filtering to predict the stock price on day T, given the observations until T − 1. Since we want the predictive distribution over 20 days, the posterior distribution for day T − 1 acts as the prior distribution for day. As yesterday’s posterior is considered today’s prior, from Fig. 5 it is observed that moving further in time, the uncertainty is propagated in the forecasts. Hence, while the initial few estimates are relatively accurate with low uncertainty, the estimates obtained towards the end become more and more inaccurate with very high uncertainty. To improve the forecast, one-step-ahead predictive strategy is employed. In this approach, given the samples from the posterior over parameters, a predictive distribution is generated over observations at each day T, given the actual observations up till day T − 1. From Fig. 6, it can be inferred that the one-step-ahead predictive distribution offers better estimates, and all the actual values are within the 95% credible interval.

A graph of a graph of a graph

Description automatically generated with medium confidence

While both the strategies model the series as a gaussian state space model and use filtering, the difference is that in the earlier forecasting strategy stock prices for the next 20 days are forecasted only using the samples and training data (01/02/2014 - 11/07/2019), however in one-step-ahead strategy apart from the samples and training data, additional prices are sent up until the day that has to be forecasted. For instance, to get the forecast for November 15th, stock prices between 11/08/2019 - 11/14/2019 will also be used.

**7. Conclusion**

Using cumulative absolute one-step-ahead prediction error, the prediction accuracy from the two forecast strategies is compared. As expected, the cumulative prediction error from the one-step-ahead prediction is lesser compared to the prediction errors using the N-step forecasting.

A graph with red and blue lines

Description automatically generated

While the obtained estimates for the stock closing prices are relatively accurate, there are a few instances, wherein the price is forecasted to increase but there is a decrease in the stock price. Knowing the relative change of the price is a very important factor in making investment decisions and hence the model forecasts should be improved further.

Possible approaches that can be undertaken are:

• Retraining the model every day to accurately predict the stock prices for tomorrow.

• Using Moving Average component along with the Autoregressive component will additionally include the effect of the previous residual noise on today’s price.

• Considering other features like price-earning (P/E) ratio in forecasting stock closing prices.

**8. References**

1.About stock market prediction retrieved from [https://en.wikipedia.org/wiki/Stock](https://en.wikipedia.org/wiki/Stock%20) market prediction

2. Steven L. Scott, Hal Varian,”Predicting the Present with Bayesian Structural Time Series”, International Journal of Mathematical Modelling and Optimization, 2013.

3. Jinwen Qiu, S. Rao Jammalamadaka, Ning Ning,” Multivariate Bayesian Structural Time Series Model”, Journal of Machine Learning Research, 2018

4. Tensorflow probability documentation retrieved from

<https://www.tensorflow.org/probability>