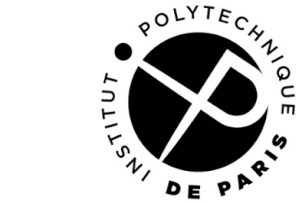

INSTITUT
POLYTECHNIQUE
DE PARIS

Data Stream Processing Project

Real-time Application with Kafka for Stock Market Forecasting

Stream vs Batch Learning

Mohamad EL OSMAN
Mohammed JAWHAR

January 2024

Contents

1 Introduction

In the realm of financial analytics, the imperative for real-time insights and dynamic prediction capabilities has spurred innovative approaches and technologies. Leveraging the robust Kafka architecture and employing a suite of sophisticated streaming models, our research endeavors have successfully culminated in the generation of real-time predictions for stock closing prices. Central to our methodology is the integration of a live dashboard, a dynamic visualization tool meticulously designed to foster seamless comprehension and accessibility to the evolving landscape of financial markets.

This project serves as a comprehensive exploration into how batch and streaming models interact, unveiling their distinct strengths in predicting stock trends. Through thorough comparative analysis, our research highlights the unique advantages each method offers, providing valuable insights for stakeholders dealing with the complexities of financial decision-making.

A vital part of our efforts involved extracting stock data from the yfinance API, a valuable source of real-time market information. Our analysis spans across five major companies, each representing its significance in its respective national financial landscape.The gathered dataset includes the following columns: "open price", "close price", "volume of transactions", "edge close", and "date". The data spans from January 1, 2018, to December 31, 2023.

This diverse dataset offers a complete range of market behaviors, enabling a nuanced understanding of how well the models adapt and perform in different business environments.

2 Stream Modeling

2.1 Introduction

The financial markets persistently exhibit dynamic and intricately patterned behaviors, presenting formidable challenges in their predictability, particularly with regard to stock prices influenced by a myriad of factors. Amidst this complexity, the imperative for precise and timely stock price predictions has driven the integration of streaming models to the forefront of financial analytics.

This segment of the report explores the domain of stream modeling as applied to the prediction of closing prices for five distinct stocks. The employment of streaming models offers the distinctive advantage of real-time analysis, affording expeditious decision-making capabilities within the volatile milieu of financial markets. As part of this exploration, we employ three distinct models— Linear Regression, SARIMAX (Seasonal AutoRegressive Integrated Moving Average with exogenous variables), and SRP (Streaming Random Patches) Regression. Each model brings forth unique strengths and capabilities, collectively contributing to the overarching objective of augmenting predictive accuracy.

Subsequent sections of this report segment will elucidate the Kafka architecture harnessed for stream processing, conduct a comprehensive analysis of the chosen modeling techniques, and culminate in an exhaustive comparison of their respective performances. In an epoch where information is not only synonymous with power but is also time-sensitive, the application of stream modeling to stock price prediction emerges as a promising avenue for investors, financial analysts, and decision-makers within the ever-evolving realm of global finance.

2.2 Kafka Architecture

In the current data-driven landscape, the demand for real-time insights has led to the development of innovative architectures for processing and visualizing stock market data. Our focus is on a system powered by Apache Kafka, designed to handle stock data seamlessly.

This architecture consists of three main components: Data Stream Producers, ModelTrain Consumers, and PredictionPlotter Consumers. Working harmoniously within Kafka, these elements create a smooth pathway for continuous data flow, predictive modeling, and live visualization.

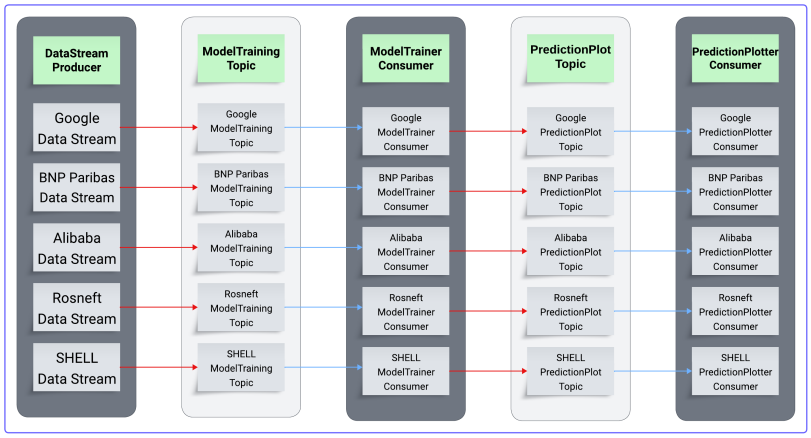

Figure 1: Diagram of Apache Kafka Architecture implemented

Data Stream Producer: The pivotal role of the Data Stream Producer lies at the heart of the architectural design, functioning as the central hub for collecting stock data from diverse sources. Each independent Data Stream Producer directs data into its dedicated ModelTraining Topic, tailored to each specific stock. This meticulous process ensures an uninterrupted and segregated data flow, a prerequisite for subsequent model computations.

ModelTrain Consumer: In the predictive model computation process, the ModelTrain Consumer segment assumes a crucial role. Synchronized with individual ModelTraining Topics, each consumer receives incoming stock data, dynamically updates the model using a selected streaming model from the River Library, executes predictions, and communicates the resulting predictions, actual values, and essential model metrics (e.g., MAE, RMSE, MAPE, CPU time) to the designated PredictionPlot Topic. This segment forms the foundational infrastructure that propels model accuracy and efficacy.

PredictionPlotter Consumer: Concluding the system, the PredictionPlotter Consumer segment orchestrates dynamic visualization. Engaging with the corresponding PredictionPlot Topic, each consumer generates real-time visual representations, including comparative plots of predicted versus actual values and the evolving MAPE metric for the respective stock. These visual insights provide immediate comprehension of model performance, thereby enhancing their practical utility and interpretability.

2.3 Modeling

The chosen stream models for predicting stock closing prices exhibit distinctive characteristics that align with the intricacies of streaming data analytics. Each model brings forth unique attributes, demonstrating adaptability, interpretability, and real-time forecasting capabilities. This essay provides a comprehensive exploration of the selected stream models:

2.3.1 Linear Regression

Linear Regression stands as a foundational statistical technique in the realm of predicting stock closing prices. By establishing a linear relationship between the dependent variable (stock closing price) and various independent variables such as historical prices, trading volumes, and market indicators, this model offers transparency and ease of interpretation.

Its adaptability to streaming data environments is underscored by its computational efficiency, allowing for continuous coefficient updates as new data emerges. The model's simplicity lends itself to real-time forecasting, making it a pragmatic choice for scenarios where quick insights are imperative. However, it is important to acknowledge that the model's linear nature may oversimplify intricate relationships within the data.

2.3.2 Seasonal Non-linear Autoregressive Integrated Moving Average (SARIMAX)

SARIMAX, an extension of the ARIMA model, introduces a level of sophistication by accommodating exogenous variables that might influence stock prices. Its capability to handle seasonality, trends, and external factors makes it a robust choice for dynamic streaming data.

The iterative updating of parameters with new observations enables the model to adjust to evolving market conditions in real-time. Despite its versatility, SARIMAX assumes linearity and stationarity, which may constrain its ability to capture certain nuances in complex market behaviors.

2.3.3 Streaming Random Patches (SRP) Ensemble Regression

The Streaming Random Patches (SRP) Regression model, a variant of symbolic regression, represents a distinctive approach in the prediction of stock closing prices. Unlike conventional models, SRP Regression operates without a predefined functional form, allowing it to dynamically adapt to evolving patterns within the streaming data environment. This adaptability proves particularly

valuable in capturing intricate relationships that might defy the linearity assumed by traditional time series models.

In real-time scenarios, SRP Regression excels at unveiling hidden patterns and providing nuanced insights, making it a powerful tool for dynamic streaming data environments. It is important to note, however, that the non-parametric nature of SRP Regression necessitates careful consideration to prevent potential pitfalls such as overfitting or misinterpretation of complex relationships.

2.4 Model Comparison

As we delve into the comparative analysis of the three streaming models , it becomes imperative to evaluate their predictive prowess in the dynamic realm of stock closing prices. The focus of this assessment revolves around the visual examination of predicted stock values (y-pred) in contrast to actual stock values (y-true) and the concurrent evaluation of Mean Absolute Percentage Error (MAPE) metrics. This detailed comparison aims to unravel the models' adaptability, accuracy, and resilience in navigating the complexities of streaming data environments.

The presented figure illustrates the outcomes and metrics derived from the application of the three implemented streaming models to the Google stock.

Figure 2: Comparison of Predicted vs True Stock Values and MAPE Metric for the three implemented Streaming Models

Notably, it is imperative to highlight that the MAPE curves for all three models exhibit a similar trajectory, indicative of acceptable performance.

As we delve into the MAPE curve, a pivotal component of our assessment, notable patterns emerge. Initial instances depict a rapid MAPE increase, a natural response as models acclimate to the nuances of early sliding windows. Subsequent observations reveal a pronounced decline in MAPE values, indicative of the models' adeptness with accumulated historical data. However, a discernible uptick in MAPE around the year 2020 introduces a compelling narrative, potentially attributed to the unanticipated market shifts during the Covid period. This anomaly prompts a reflection on the models' resilience in adapting to unforeseen events.

Crucially, despite these nuanced challenges, the streaming models consistently showcase a robust capacity to closely align with the original plot. Their intrinsic adaptability and resilience in the face of concept drift underscore their efficacy in navigating evolving data dynamics.

2.5 Live Dashboard

To facilitate a clear comprehension of the outcomes and metrics produced by the employed streaming models, we developed an application featuring a live visual dashboard. Constructed using the Plotly library, this dashboard is designed to dynamically plot the forecasting results and metrics in real-time.

The application draws information from the PredictionPlot Topic, generating live graphs that illustrate the alignment between our predictions and the actual values. Additionally, it provides insights into the temporal evolution of a key metric, the Mean Absolute Percentage Error (MAPE), for each stock.

3 Batch Modeling

3.1 Introduction

In contrast to the stream modeling approach discussed earlier, the batch modeling section adopts a different process, owing to the availability of local access to the data. Unlike the real-time nature of streaming data, batch modeling leverages historical data and doesn't really require the use of Kafka messaging.

The primary objective remains however the same – to forecast the stock prices of five major companies across different countries. This section will provide a detailed account of the data exploration and preprocessing steps, the fundamental principles underlying each of the three models we will be using - LSTM, ARIMA and PROPHET, their training procedures, and the evaluation metrics used to assess their forecasting performance.

By the end of this exploration, we aim to shed light on the strengths and limitations of each model and unveil insights into their predictive performance for stock market forecasting.

Figure 3: Batch learning pipeline for Stock Market Forecasting.

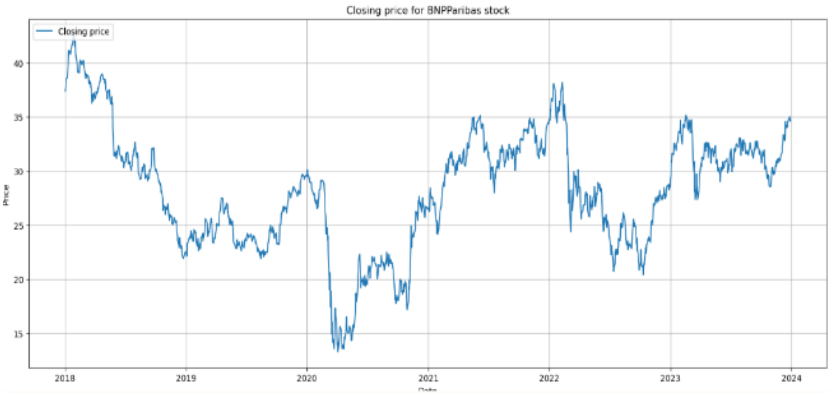

Figure 4: Example of a Stock Market plot.

3.2 Data exploration and preprocessing

In this subsection, we will explore fundamental concepts crucial for time series analysis. We will delve into the concept of stationarity, examine the autocorrelation function (ACF) and partial autocorrelation function (PACF), essential tools for understanding the temporal dependencies within the data, and also investigate for seasonal patterns in the stocks to ensure if our models account for any periodic trends in the stock market data.

3.2.1 Stationarity

To ensure the reliability of time series models, stationarity is a crucial consideration, as it simplifies modeling assumptions and enhances model performance. Particularly, stationary time series exhibit consistent statistical properties over time, including constant mean and variance.

Stationarity Detection

A common method for stationarity detection is the Augmented Dickey-Fuller test (ADF). This statistical test evaluates the null hypothesis H_0 that a unit root is present in the time series, indicating non-stationarity, to the alternative hypothesis H_1 that suggests stationarity.

Mathematically, the ADF test can be expressed as:

$$H_0\textbf{: The time series is non-stationary}$$
$$H_1\textbf{: The time series is stationary}$$

The ADF test compares the observed test statistic to critical values, and a p-value is used to determine the rejection of the null hypothesis. A lower p-value indicates a stronger rejection of non-stationarity. If the test suggests non-stationarity, differencing the time series can be applied to achieve it.

Differencing method

Differencing is a technique involving subtracting the previous value from the current value. This transforms the time series, making it more stable(stationary) and amenable to modeling and capturing underlying patterns.

$$\textbf{Differencing Formula: } \Delta Y_t = Y_t - Y_{t-1}$$

In our analysis, we conducted the ADF test on the time series of the five stocks. The results indicated non-stationarity. Subsequently, we applied differencing and got results similar to this :

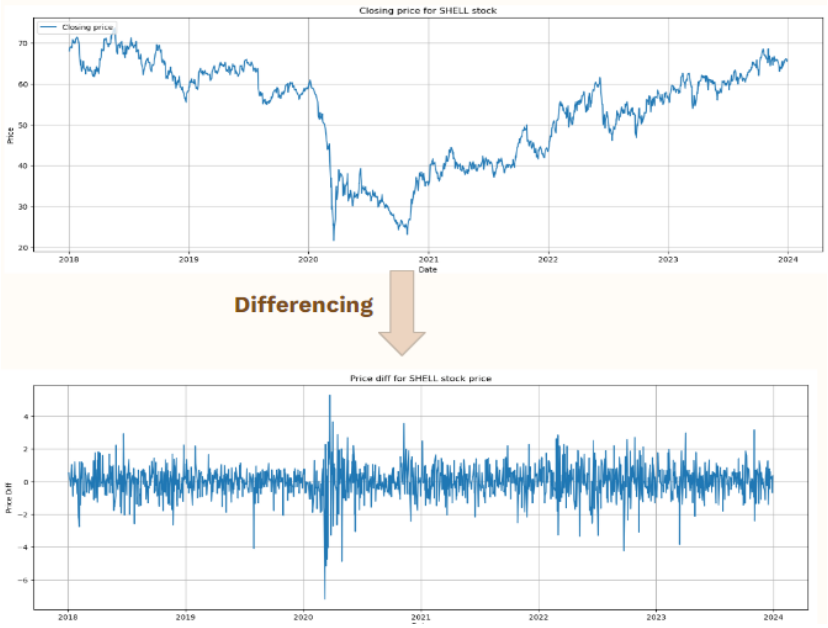

Figure 5: Differencing process on Stock Market Data to make it stationary.

3.2.2 Auto-correlation and Partial Auto-correlation

To further analyze the time series data, we explore auto-correlation (ACF) and partial auto-correlation (PACF). These statistical measures provide insights into the temporal dependencies within the data, aiding in model hyperparameter selection and understanding the underlying patterns.

Auto-correlation measures the linear relationship between a variable and its lagged values. ACF at lag k is denoted as ACF_k, and its mathematical expression is given by:

$$ACF_k = \frac{\text{Cov}(Y_t, Y_{t-k})}{\sqrt{\text{Var}(Y_t) \cdot \text{Var}(Y_{t-k})}}$$

Partial auto-correlation, on the other hand, captures the direct relationship between two variables while controlling for the influence of the intermediate

lagged values. PACF at lag k is denoted as $PACF_k$, and its mathematical expression is:

$$PACF_k = \frac{\text{Cov}(Y_t, Y_{t-k}|Y_{t-1}, ..., Y_{t-k+1})}{\sqrt{\text{Var}(Y_t|Y_{t-1}, ..., Y_{t-k+1}) \cdot \text{Var}(Y_{t-k}|Y_{t-1}, ..., Y_{t-k+1})}}$$

Understanding the shapes of ACF and PACF curves is crucial. For example, a gradual decay in ACF may suggest a non-stationary time series, while a sudden drop in PACF could indicate a stationary series. These insights guide the selection of parameters in time series models.

In particular, the cutoff points - The points where the coefficients go below the significance level- in these two respective curves, correspond to the orders p, and q of the Auto-Regressive and Moving Average models respectively, which are used also for time series forecasting.

In our analysis, we examined the ACF and PACF of the differenced stock market data, providing valuable information for subsequent modeling steps. As we can see in the following plot , we found that p=1 and q=1 for all companies stock data.

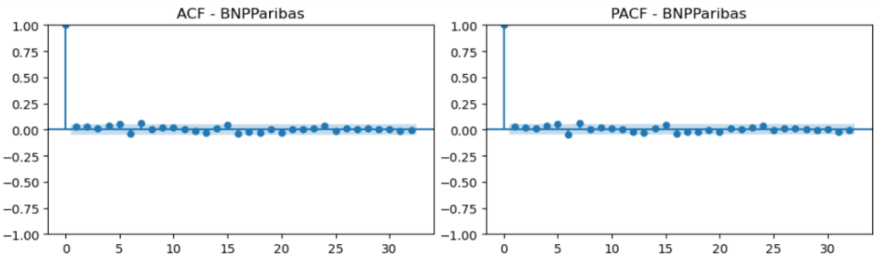

Figure 6: Stock Market Data ACF and PACF plots.

3.2.3 Seasonality

Seasonality refers to recurring patterns or fluctuations in a time series data that repeat at regular intervals. In the context of stock market data, detecting seasonality is important for identifying patterns related to specific times of the year, months, or quarters, and improving the forecasting accuracy. Seasonal trends may be influenced by factors such as holidays, financial reporting cycles, or other external events.

Upon a thorough observation of our stock market data, it appears that there is no apparent seasonality, which is typical for financial data, known for its inherent volatility. Instead, we observe recurrent cyclic peaks, with a tendency

to appear quarterly. However, these peaks lack a well-defined and consistent period.

While the cyclic peaks may not adhere strictly to a regular pattern, considering potential quarterly trends can capture any recurring patterns that might influence the stock market. Our best approach then is to make an assumption of yearly and quarterly seasonality in our analysis.

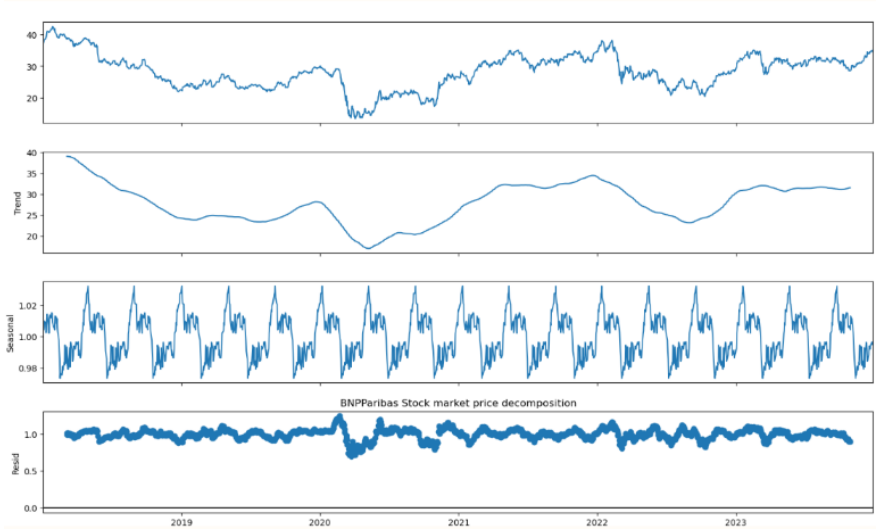

Figure 7: Stock Market Data Seasonal decomposition with period=85.

3.3 Modeling

In this phase of our analysis, we embark on experimenting with three distinct forecasting models: Long Short-Term Memory (LSTM), Autoregressive Integrated Moving Average (ARIMA), and Prophet. Following the model training, we will delve into a thorough comparison of their performance metrics. This evaluation will not only provide insights into the individual capabilities of each model but will also guide us in drawing a comparison between stream, and batch learning for this usecase.

3.3.1 Long Short-Term Memory (LSTM)

The first model we will be experimenting is the Long Short-Term Memory (LSTM), a well-established architecture renowned for its effectiveness in forecasting sequential data, particularly time series. LSTMs are adept at capturing long-term dependencies in the data, making them particularly suitable for mod-

eling the intricate dynamics of stock market trends.

In our training approach, we employed a simple LSTM configuration. Input sequences of length 60 days were used to predict the next day's stock market value. This choice of sequence length allows the model to consider the past two months of data, capturing medium-term patterns.

The prediction process on the test set unfolds in a sequential manner. Starting with the last sequence from the training data, the model predicts the subsequent day's value. This predicted value is then added to the end of the input sequence, and the process iterates, updating the input sequence with each prediction.

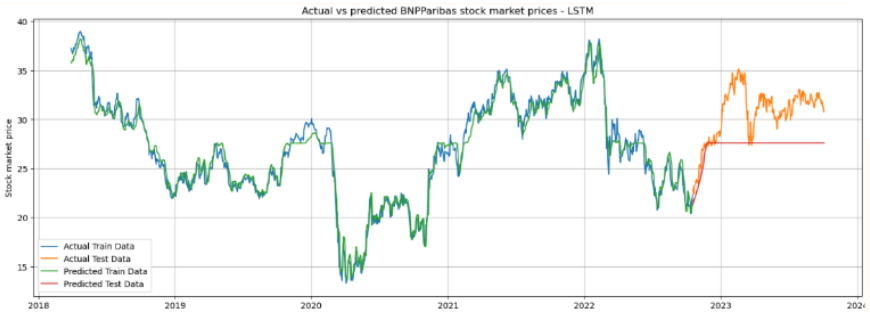

Figure 8: LSTM Stock market Forecasting.

Company	Model	MAE	MAPE	RMSE	Training_time
Alibaba	LSTM	11.07	12.26	15.02	83.28
BNPParibas	LSTM	4.49	13.88	5.07	88.70
Google	LSTM	34.34	26.94	38.62	79.33
ROSNFET	LSTM	114.89	20.14	132.04	53 .11
SHELL	LSTM	10.23	16.33	10.66	82.28

Figure 9: LSTM performance metrics for the five companies Stocks.

Upon assessing this model forecasting capabilities, we observe that :

- The LSTM model requires a substantial training time due to its complex architecture characterized by a high number of neurons and epochs.
- The model manages to follow the underlying trend to a reasonable extent, especially when predicting from sequences with a specific trend(Not a changepoint, going either up or down).

- Forecast accuracy can be low when predictions are initiated from significant change points in the data (pointwith abrupt changes in the curve's trend).

3.3.2 Autoregressive Integrated Moving Average (ARIMA)

The Autoregressive Integrated Moving Average (ARIMA) model is a traditional yet robust and powerful time series forecasting model that combines autoregression (AR), differencing (I), and moving averages (MA). Its formula can be expressed as follows:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q}$$

Here:

- Y_t is the current value of the differenced stock time series,
- $\phi_1, \phi_2, \ldots, \phi_p$ are the autoregressive coefficients,
- $Y_{t-1}, Y_{t-2}, \ldots, Y_{t-p}$ are the lagged values of the time series,
- ε_t is the current forecast error,
- $\theta_1, \theta_2, \ldots, \theta_q$ are the moving average coefficients,
- $\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-q}$ are the lagged independent identically distributed forecast errors terms with zero mean.

In our implementation, we employed ARIMA with orders $p = 1$, $d = 1$, and $q = 1$, which were found to be suitable for our stock market data, and determined during the data exploration and preprocessing phase. This configuration signifies that the current predicted value of the 1-differenced time series depends on the last value, and the last forecasting error.

Although the ARIMA model appears to have inaccurate forecasting consisting in a simple regression, its efficiency lies in its simplicity and adaptability, making it a valuable tool for forecasting in dynamic financial environments.

Notably, in comparison to LSTM, the ARIMA model demonstrates similar MAPE values. However, a key strength lies in its significantly faster training time. This efficiency in training is a notable advantage for models of this class, as it facilitates rapid iterations and optimizations, making ARIMA a pragmatic choice for time-sensitive forecasting tasks in the financial domain.

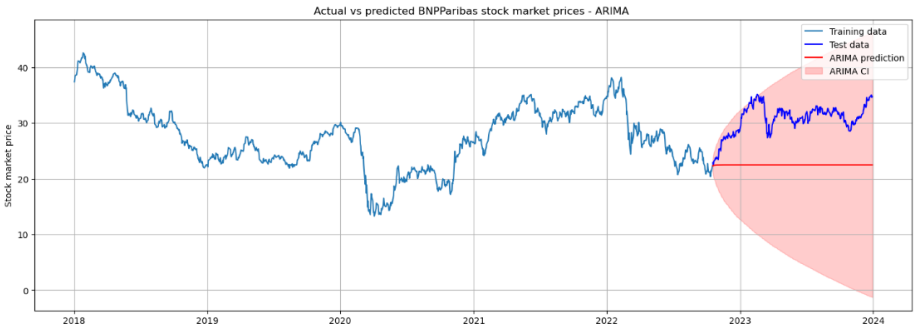

Figure 10: ARIMA Stock market Forecasting.

Company	Model	MAE	MAPE	RMSE	Training_time
Alibaba	ARIMA	12.72	13.61	15.84	0.20
BNPParibas	ARIMA	8.29	26.36	8.68	0.27
Google	ARIMA	18.92	15.19	23.09	0.16
ROSNFET	ARIMA	64.80	0.51	0.48	0.11
SHELL	ARIMA	9.29	0.01	0.98	0.27

Figure 11: ARIMA performance metrics for the five companies Stocks.

3.3.3 PROPHET

The third and final model in our experimental study is Prophet, developed by Meta. Prophet is specifically designed for forecasting time-series data that exhibit patterns such as seasonality and holidays. While we initially assumed that our time series might not have strong seasonal patterns, we chose to leverage Prophet for its unique capabilities, consisting in detecting change points - a challenge encountered with LSTM and ARIMA.

Despite the model exhibiting unpleasant forecasting results and a relatively high Mean Absolute Percentage Error (MAPE) compared to other models, a closer examination reveals noteworthy insights: Prophet adeptly captures the general trend of the test data and effectively identifies change points. However, there is a notable discrepancy in scale, indicating a potential need for hyper-parameter fine-tuning and the incorporation of holidays or special periods to enhance estimation accuracy.

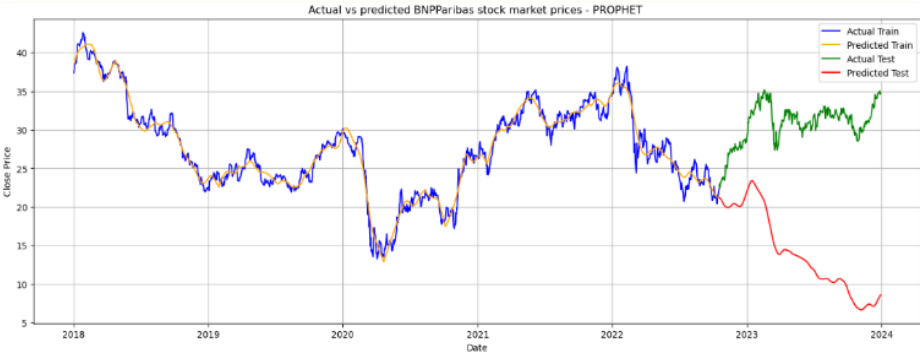

Figure 12: PROPHET Stock Market Forecasting.

Company	Model	MAE	MAPE	RMSE	Training_time
Alibaba	PROPHET	90.73	104.30	93.04	1.48
BNPParibas	PROPHET	19.78	63.58	20.29	1.06
Google	PROPHET	50.91	41.39	58.21	0.96
ROSNEFT	PROPHET	238.39	48.87	263.86	1.02
SHELL	PROPHET	8.83	14.00	11.21	1.23

Figure 13: PROPHET performance metrics for the five companies Stocks.

3.4 Model Comparison

Before concluding, let's draw a comparison between the performance of different models :

3.4.1 Batch Models:

When considering a trade-off between good performance and fast training for stock market data forecasting, ARIMA model emerges as the most favorable choice among the batch models. Although it requires further in depth analysis concerning the regressive orders used and the cyclic patterns(using potentially SARIMA), ARIMA exhibits a balanced approach, incorporating **interpretable and explainable** modeling through its orders (p, d, q) to achieve a satisfactory balance between accuracy and training time.

On the other hand, Prophet's distinctive approach makes it a valuable asset, especially in scenarios where change points and specific temporal patterns need to be identified. However, it requires thorough fine-tuning and refinement to better adapt to our specific use case.

3.4.2 Batch vs. Streaming Models:

Upon comparing batch models with streaming models, a clear trend emerges:

Streaming Models

Advantages

- **Real-Time Adaptability:** Streaming models inherently excel in working with sliding windows, aligning seamlessly with the continuous nature of stock market data presented as time series. This adaptability ensures that the models remain robust in capturing evolving patterns and swiftly adjusting to new information.
- **Concept Drift Detection:** They excel in detecting concept drift, enabling quick adjustments to changing data patterns.
- **Short-Term Performance:** Streaming models generally perform well in the short term, making them effective for applications requiring rapid adjustments.

Challenges

- **Outlier Sensitivity:** Streaming models may be sensitive to outliers, particularly during exceptional periods such as the COVID-19 pandemic.
- **Complexity in Fine-Tuning:** Achieving optimal performance may require fine-tuning hyperparameters and addressing sensitivity issues, adding complexity to model development.

Batch Models

Advantages

- **Historical Data Fitting:** Batch models benefit from fitting on all historical data, providing a comprehensive understanding of long-term trends.
- **Training Speed:** Training **on the entire dataset** seems faster, as we have local access to the entire historical data unlike stream learning where training is lagged by Kafka messaging, .

Challenges

- **Changepoint Detection:** Batch models may struggle to promptly detect changepoints, especially when detailed information about seasonal patterns is lacking.
- **Risk in Long-Term Predictions:** Relying on long-term predictions from batch models poses a risk, as they may struggle to accurately capture evolving market conditions, potentially leading to financial losses.

4 Conclusion and Perspectives

In this project, we explored the intricate task of stock market forecasting, employing both batch and streaming models to analyze the financial performance of major companies in different countries. The combination of stream modeling using Kafka and River models, and batch modeling with LSTM, ARIMA, and Prophet allowed us to gain valuable insights into the strengths and limitations of these approaches.

In summary, the choice between batch and streaming models hinges on the specific requirements of the forecasting task. While batch models offer a good compromise between performance and training speed, streaming models showcase superior adaptability and forecasting accuracy, especially in scenarios where real-time adjustments to changing patterns are paramount.

While this project provides a comprehensive understanding of stock market forecasting using diverse models, there are avenues for future exploration and improvement:

- **Enhanced Fine-Tuning:** Further hyperparameters fine-tuning for some models, especially Prophet, could improve their accuracy and make them more robust to outliers.
- **Incorporating External Factors:** Future work may involve integrating external factors, such as economic indicators or news sentiment analysis, to enhance the models' predictive capabilities.
- **Ensemble Approaches:** Exploring ensemble approaches that combine the strengths of both batch and streaming models could lead to more robust and accurate predictions.

References

[1] *River Library*, https://riverml.xyz/latest/api/overview/,

[2] *Plotly Library*, https://github.com/plotly/plotly.py,

[3] Reinert, G., *Time Series*, https://www.stats.ox.ac.uk/~reinert/time/notesht10short.pdf, 2010.

[4] Sakshi Kulshreshtha and Vijayalakshmi A., *An ARIMA- LSTM Hybrid Model for Stock Market Prediction Using Live Data*, https://pdfs.semanticscholar.org/597f/f5bfd6d85d531cc5e891f1e39fd41c2305ff.pdf, 2020.

[5] Emir Žunić, Kemal Korjenić, Kerim Hodžić, and Dženana onko, *Application of Facebook's PROPHET algorithm for successful sales forecasting based on real-world data*, https://arxiv.org/ftp/arxiv/papers/2005/2005.07575.pdf, 2020.