PRICE PREDICTION: A COMPARISON BETWEEN ARIMA MODEL & LSTM

ISE 537 FINANCIAL ANALYTICS: Final Project

Saisujit Bayanaboyana
UCD ID: 7845906173
bayanabo@usc.edu

ABSTRACT

Our study delves into predicting stock prices in the supply chain and logistics sector by comparing two powerful forecasting methods: Long Short-Term Memory (LSTM) and AutoRegressive Integrated Moving Average (ARIMA). We gather historical stock data from key companies in this industry to train and evaluate both models. By using practical metrics like Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), we aim to see which model best forecasts stock prices. In acknowledging the volatile nature of the stock market, influenced by a multitude of factors such as economic fluctuations, supply and demand dynamics, company health, and prevailing sentiments, predicting stock prices becomes a challenging task. This volatility adds complexity to the prediction process, making it a cumbersome endeavor. We're particularly interested in comparing LSTM, which excels at grasping long-term patterns in data, with the tried-and-tested ARIMA model based on time series analysis. Our goal is a straightforward comparison that sheds light on which approach might be more effective for predicting stock prices.

DATA: We gathered data from Yahoo Finance spanning November 2020 to November 2023, deliberately focusing on the post-pandemic period. We chose companies like CHRW and UPS because they faced ups and downs during the pandemic, reflecting the supply chain industry's challenges. The supply chain industry went through a rollercoaster during the pandemic due to disrupted logistics, changing consumer behavior, and economic uncertainties. After November 2020, there were signs of recovery. The S&P 500's rebound, as reported by The New York Times and Investopedia, signaled a turnaround in the market. Our dataset starts from this post-pandemic recovery period to explore how companies in the supply chain, like CHRW and UPS, navigated these shifts. A little information on these companies:

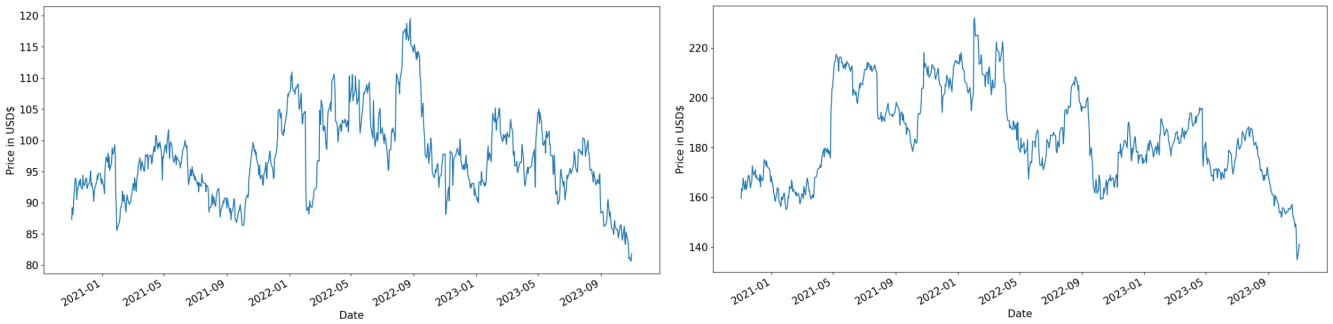
CH Robinson (CHRW):

- Global third-party logistics company offering freight transportation and logistics services since 1905.
- Acts as intermediaries between shippers and carriers, using technology-driven solutions.

UPS (United Parcel Service):

- Multinational package delivery and supply chain management company established in 1907.
- Known for its global logistics network, and innovative supply chain solutions worldwide.

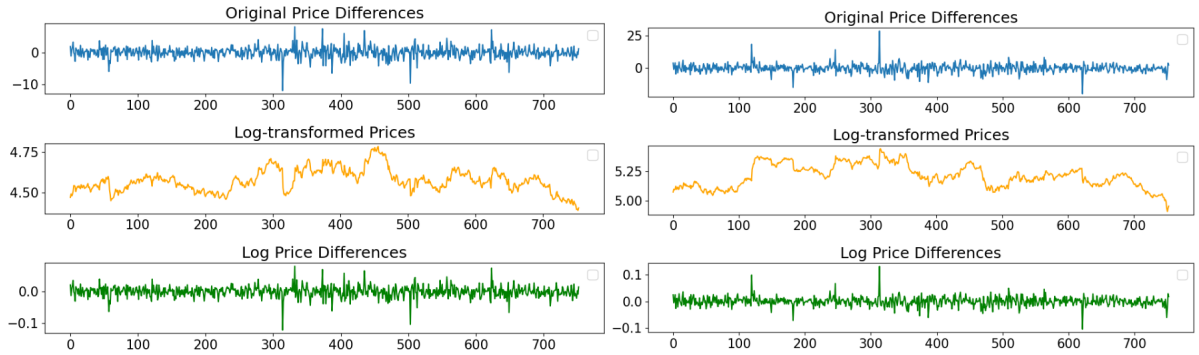
We're interested in understanding how they adapted to the changed economic landscape and capitalized on the market resurgence. This period offers a dynamic window to analyze how these companies performed as the industry bounced back. The dataset utilized in this study encompasses various columns: Open, High, Low, Close, Adj Close, and Volume, all organized in a structured and properly formatted manner. However, for this specific analysis, our primary focus rested solely on the 'Close' prices. The dataset covers a span from November 1st, 2020 to November 1st, 2023, encapsulating a comprehensive three-year period. Within this timeframe, it comprises a total of 754 rows. This selection was deliberate, aiming to concentrate solely on the closing prices of the stocks.



(a) CHRW 'Close' Price History (b) UPS 'Close' Price History

Figure 1: Time series data plot of CHRW & UPS stock

STATIONARITY: A stationary time series is one where statistical properties like mean, variance, and autocorrelation remain constant over time. We observe that both CHRW and UPS 'Close' price history are likely non stationary. To achieve stationarity we have to do transformations to the time series like considering the price difference, taking the log of Prices, or taking the difference in log prices. A stationary time series simplifies forecasting by having consistent statistical properties, enabling reliable predictions over time without changing characteristics.



(a) CHRW Transformed Prices (b) UPS Transformed Prices

Figure 2: Transformed Time series data plot of CHRW & UPS stock

The *Augmented Dickey-Fuller (ADF) test* is a statistical method used to assess whether a time series is stationary or non-stationary. It examines a null hypothesis that the time series possesses a unit root, indicating non-stationarity, against an alternative hypothesis of stationarity. The ADF statistic is compared against critical values at various confidence levels (1%, 5%, and 10%) to determine stationarity. Lower ADF statistics (more negative) and corresponding p-values below a significance level (commonly 0.05) favor the rejection of the null hypothesis, indicating stationarity. For CHRW, the closing prices and log prices indicate the presence of non-stationarity. The price difference and the log price difference yield an ADF statistic of -21.034387 with a p-value of 0.000000 & -21.010212 with a p-value of 0.000000, respectively. Similarly for UPS, the closing prices and log prices indicate the presence of non-stationarity. The price difference and the log price difference yield an ADF statistic of -25.386954 with a p-value of 0.000000 & -19.818510 with a p-value of 0.000000, respectively. The exceptionally low p-value of the closing price difference strongly rejects the null hypothesis, showcasing the highest evidence for stationarity among the given tests, making it the best candidate for a stationary time series based on the ADF test's outcomes.

ARIMA

ARIMA stands for Autoregressive Integrated Moving Average. It's a widely used time series forecasting method that helps model and predict future values based on historical data. ARIMA models are effective for understanding and predicting data points with a strong temporal component, such as stock prices, temperature readings, or sales figures. The three components of an ARIMA model are:

1. Autoregressive (AR) Component: This part models the relationship between an observation and a number of lagged observations (i.e., its own past values). The notation for this is AR(p), where 'p' represents the number of lag observations included in the model. The formula for the AR component is:

$$Y_t = \mu + \sum_{i=1}^{p} \phi_i y_{t-i} + e_t$$

where p is the order of the AR model and ϕ1,ϕ2,⋯,ϕp are p partial autocorrelation parameters for the AR(p) model. AR(p) model has only p partial autocorrelations that are statistically different from zero. The autocorrelation coefficient of the AR(p) model trails off to zero and is restricted between −1 to 1.

2. Integrated (I) Component: This part represents the differencing of raw observations to make the time series stationary. Stationarity implies that statistical properties like mean, variance, and autocorrelation are constant over time. The notation for this is I(d), where 'd' is the degree of differencing.

3. Moving Average (MA) Component: This part models the relationship between the observation and residual errors from a moving average model applied to lagged observations. The notation for this is MA(q), where 'q' represents the number of lagged forecast errors in the prediction equation. The formula for the MA component is:

$$Y_t = \mu + \sum_{i=1}^{q} \theta_t e_{t-i} + e_t$$

Where q is the order of the model and θ1,θ2,⋯, θ q are parameters of the model. The et,et−1,⋯,et−q are the white noise error terms.

Parameters:
We applied the ARIMA(p,d,q) model to analyze the time series data, employing both AIC and BIC criteria to determine the best fit. Considering the balance between AIC and BIC, we opted for ARIMA(3,0,2) for CHRW and ARIMA(1,0,0) for UPS.

```
AIC: 2396.9878159410546, BIC: 2405.791650334509, Order: (0, 0, 0)
AIC: 2398.753636256837, BIC: 2411.9593878470187, Order: (0, 0, 1)
AIC: 2396.538082990531, BIC: 2414.1457517774397, Order: (0, 0, 2)
AIC: 2398.792902143722, BIC: 2411.9986537339037, Order: (1, 0, 0)
AIC: 2399.4842197815597, BIC: 2417.091888568468, Order: (1, 0, 1)
AIC: 2398.423944916007, BIC: 2420.433530899643, Order: (1, 0, 2)
AIC: 2396.5146154018526, BIC: 2414.122284188761, Order: (2, 0, 0)
AIC: 2398.5086755410584, BIC: 2420.5182615246945, Order: (2, 0, 1)
AIC: 2400.3896004368535, BIC: 2426.8011036172165, Order: (2, 0, 2)
AIC: 2398.5095466019307, BIC: 2420.5191325855667, Order: (3, 0, 0)
AIC: 2400.5097260558728, BIC: 2426.921229236236, Order: (3, 0, 1)
AIC: 2397.0921345961638, BIC: 2427.905554973254, Order: (3, 0, 2)
AIC: 2400.4918298515254, BIC: 2426.9033330318885, Order: (4, 0, 0)
AIC: 2402.1703217700556, BIC: 2432.9837421471457, Order: (4, 0, 1)
AIC: 2396.5879874760344, BIC: 2431.803325049852, Order: (4, 0, 2)
AIC: 2401.342017119748, BIC: 2432.155437496838, Order: (5, 0, 0)
AIC: 2403.327783387431, BIC: 2438.5431209612484, Order: (5, 0, 1)
AIC: 2394.7045081870338, BIC: 2434.3217629575784, Order: (5, 0, 2)
Best AIC: 2394.7045081870338, Best BIC: 2405.791650334509
```

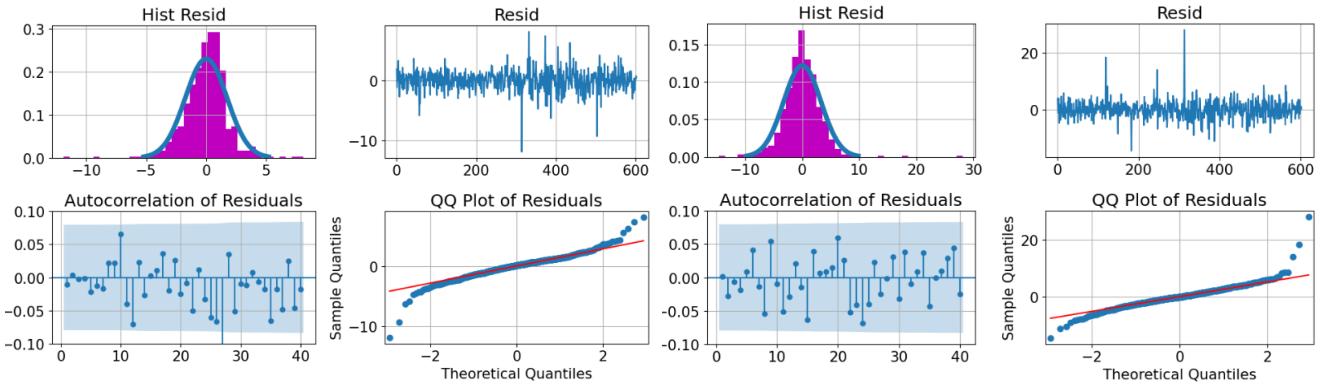
(a) CHRW AIC BIC Params

```
AIC: 3145.9010708564847, BIC: 3154.704905249939, Order: (0, 0, 0)
AIC: 3145.1651329814445, BIC: 3158.370884571626, Order: (0, 0, 1)
AIC: 3146.823341508354, BIC: 3164.4310102952627, Order: (0, 0, 2)
AIC: 3145.3076100581666, BIC: 3158.513361648348, Order: (1, 0, 0)
AIC: 3146.8704720078526, BIC: 3164.478140794761, Order: (1, 0, 1)
AIC: 3148.760176489235, BIC: 3170.769762472871, Order: (1, 0, 2)
AIC: 3146.81200410988, BIC: 3164.4196728967886, Order: (2, 0, 0)
AIC: 3148.7608458576565, BIC: 3170.7704318412925, Order: (2, 0, 1)
AIC: 3150.7600372265856, BIC: 3177.1715404069487, Order: (2, 0, 2)
AIC: 3148.791299075332, BIC: 3170.800885058968, Order: (3, 0, 0)
AIC: 3150.7608106933926, BIC: 3177.1723138737557, Order: (3, 0, 1)
AIC: 3146.290452732462, BIC: 3177.1038731095523, Order: (3, 0, 2)
AIC: 3150.5805587206487, BIC: 3176.992061901011 8, Order: (4, 0, 0)
AIC: 3152.565178423704, BIC: 3183.378598800794, Order: (4, 0, 1)
AIC: 3142.864489031266, BIC: 3178.079826605084, Order: (4, 0, 2)
AIC: 3152.50053000918, BIC: 3183.31395038627, Order: (5, 0, 0)
AIC: 3154.343515372756, BIC: 3189.5588529465736, Order: (5, 0, 1)
AIC: 3146.062465246006, BIC: 3185.6797200165506, Order: (5, 0, 2)
Best AIC: 3142.864489031266, Best BIC: 3154.704905249939
```

(b) UPS AIC BIC Params

Figure 3: AIC & BIC Trade-off for CHRW & UPS stock.

Model Diagnostics:



(a): CHRW (b): UPS

Figure 4: ARIMA Model Diagnostics for CHRW & UPS stock.

In both cases, the histogram shows the distribution of the residuals which are centered around zero, which is a good sign. It is also relatively symmetric, which suggests that there is no strong bias in the model. The autocorrelation plot shows that there is no significant autocorrelation in the residuals. This means that the errors at one time point are not dependent on the errors at other time points. The QQ plot shows that the residuals follow a normal distribution reasonably well. This is a good sign because it means that the models are likely to be robust to outliers.

LSTM

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) architecture designed to handle the issue of vanishing or exploding gradients, which commonly occurs in traditional RNNs. LSTMs are well-suited for sequence modeling tasks such as time series prediction, natural language processing, speech recognition, and more, where context and memory play crucial roles. LSTM Structure:

1. Gates: LSTMs contain various gates that regulate the flow of information through the cell. These gates are:
    - Forget Gate: Determines what information to discard from the cell state.
    - Input (or Update) Gate: Decides what new information to store in the cell state.
    - Output Gate: Chooses what information to output based on the current cell state.

2. Cell State (Ct): The cell state runs linearly through the entire chain of LSTM units, making it the "memory" of the network. The gates control how information is added or removed from this state.

3. Hidden State (Ht): The hidden state is the output of the LSTM unit and carries the information the network decides to pass on to the next unit or to the model's output.

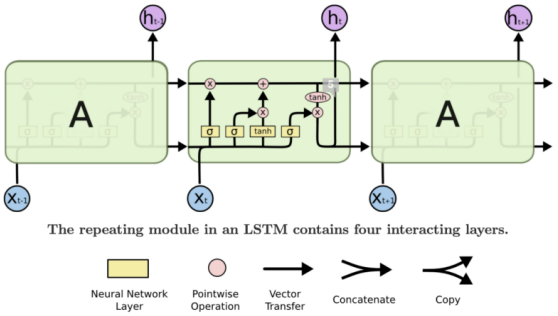
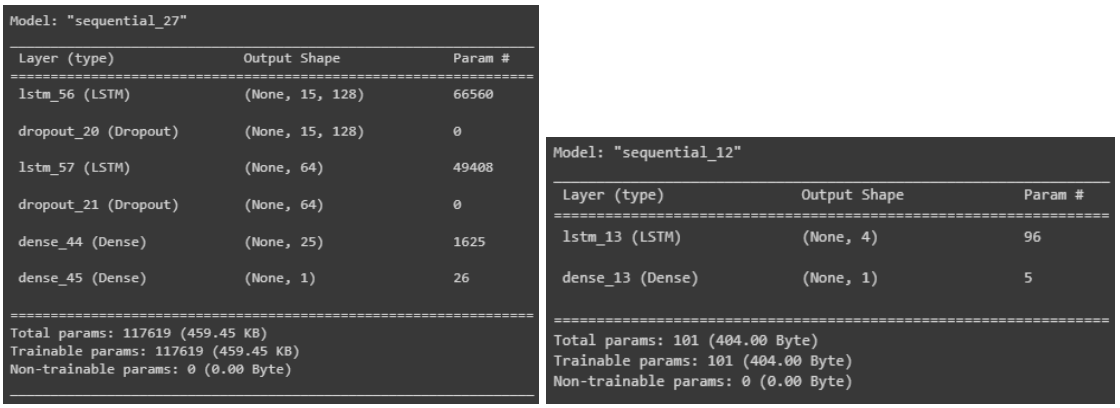


Figure 5: LSTM Architecture.

LSTM networks excel in handling time series data due to their capability to discern significant events with variable time lags between them, addressing the vanishing gradient problem inherent in conventional RNNs. However, their training duration and memory requirements are notably higher. LSTMs are susceptible to overfitting and remarkably sensitive to diverse random weight initializations, posing challenges in achieving optimal performance and necessitating careful tuning to mitigate these issues.

Model Summary:



```
Model: "sequential_27"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_56 (LSTM)              (None, 15, 128)           66560

 dropout_20 (Dropout)        (None, 15, 128)           0

 lstm_57 (LSTM)              (None, 64)                49408

 dropout_21 (Dropout)        (None, 64)                0

 dense_44 (Dense)            (None, 25)                1625

 dense_45 (Dense)            (None, 1)                 26

=================================================================
Total params: 117619 (459.45 KB)
Trainable params: 117619 (459.45 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

(a) CHRW

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_13 (LSTM)              (None, 4)                 96

 dense_13 (Dense)            (None, 1)                 5

=================================================================
Total params: 101 (404.00 Byte)
Trainable params: 101 (404.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

(b) UPS

Figure 6: LSTM Model Summaries for CHRW & UPS.

The first (CHRW) model comprises six distinct layers with a total of 117,619 parameters. It starts with two LSTM layers, the first housing 66,560 parameters and generating sequences of 128 dimensions for input sequences of length 15, while the subsequent LSTM layer contains 49,408 parameters and outputs 64-dimensional vectors. Following each LSTM layer is a dropout layer, serving as a regularization tool. The model continues with two dense layers: one with 25 units, consisting of 1,625 parameters, using a default ReLU function, and the final dense layer with a single neuron requiring 26 parameters, producing the model's output.

The second (UPS) model architecture consists of two layers, encompassing a total of 101 parameters. The initial layer, an LSTM, incorporates 96 parameters and generates an output shape of (None, 4), indicating the production of a 4-dimensional vector for each input sequence. Following this LSTM layer, the dense layer, comprising 5 parameters, serves as the output layer, responsible for computing an output value, given the processed sequence data.

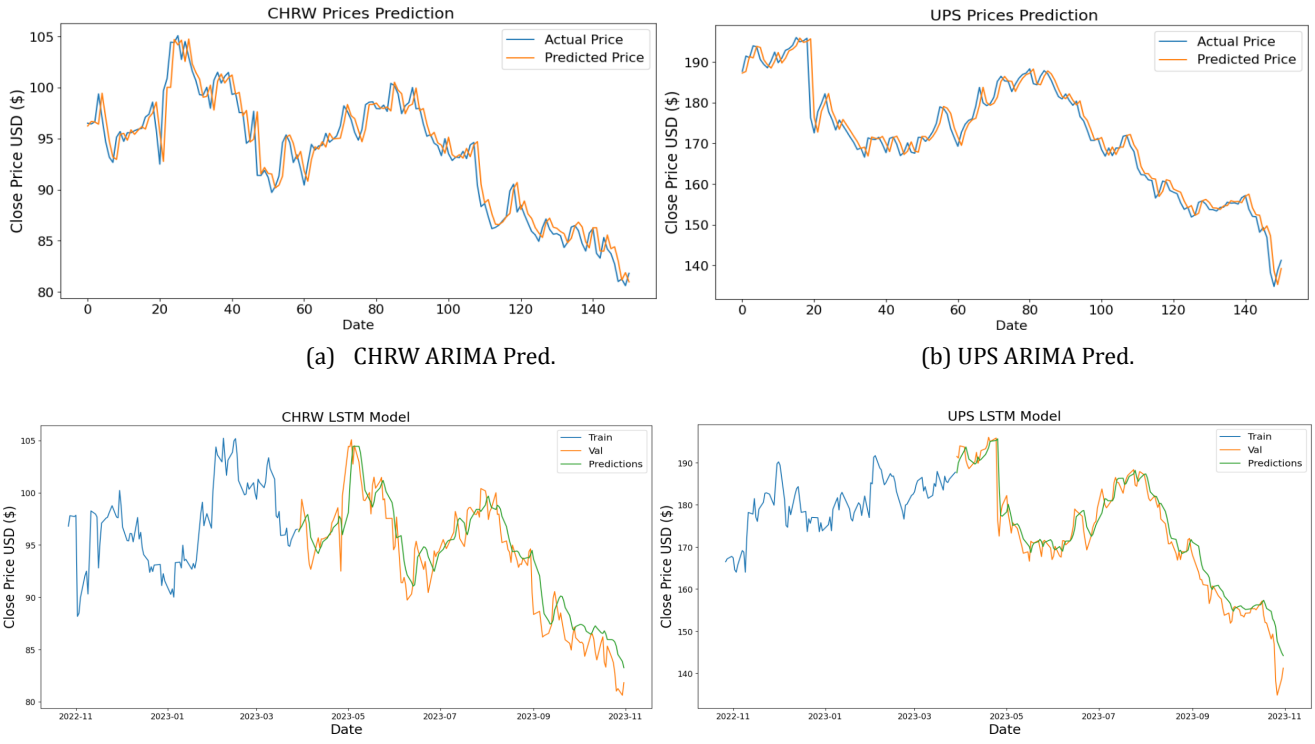
Model 1 and Model 2 present distinct architectures and training configurations. For both the models we had an 80-20 train-test split ratio. Model 1 boasts a more intricate design with two LSTM layers, additional dense layers, and a substantially larger parameter count of 117,619 compared to Model 2's 101 parameters, comprising a single LSTM layer and a dense output layer. During training, Model 1 was fit using a smaller batch size of 128 and 100 epochs, while Model 2 underwent training with a larger batch size of 256 and 1000 epochs. The higher complexity of Model 1 suggests its potential to capture intricate patterns but might be prone to overfitting, while Model 2's simpler architecture might offer better generalization, especially for less complex datasets, albeit at a computationally efficient scale.

PERFORMANCE COMPARISON

RMSE Values Comparison		
STOCK	**ARIMA**	**LSTM**
CHRW	1.5712214512610592	2.0877093707457814
UPS	2.726988428348849	3.5502574609206667

Why RMSE? RMSE, or Root Mean Square Error, is a solid choice for evaluating stock time series models because it straightforwardly measures the average prediction error. In the realm of stocks, where accuracy matters, RMSE reflects how close the predicted values are to the actual stock prices. It gives a clear, easy-to-understand representation of how well the model forecasts future stock movements. Lower RMSE values signify that the model's predictions align closely with the real stock prices, making it a practical and intuitive metric for assessing a model's performance in stock market forecasting.

Across both companies, the ARIMA model generally outperformed the LSTM model in terms of lower RMSE. CHRW had lower RMSE values overall compared to UPS, suggesting potentially better model fit or predictive accuracy for CHRW's data. However, the difference in RMSE between ARIMA and LSTM for both companies is relatively small, indicating closer performance between these model types.



(a) CHRW ARIMA Pred. (b) UPS ARIMA Pred.

(c) CHRW LSTM Pred. (d) UPS LSTM Pred.

Figure 7: Forecast plots for CHRW & UPS. (a & b show only the out-of-sample period)

It's fascinating to observe the remarkable synchronization in the *behavior of both stocks*, not only in their historical data but also in the predicted trends, despite the models being finely tuned to each stock's unique history. This close correlation is probably a result of their significant positions as leading entities within the same industry, influencing their movements in tandem.

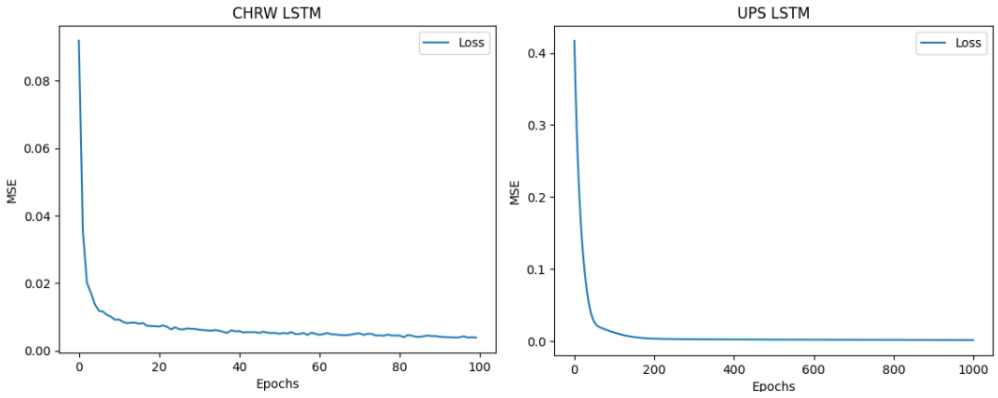


Figure 8: CONVERGENCE of CHRW & UPS models evident by decrease and stabilization of loss function.

ARIMA Advantages:

- Easy to Understand: It's simpler to grasp how ARIMA works compared to neural networks like LSTM.
- Good for Simple Trends: Handles straightforward patterns well in data that doesn't change too wildly.
- Less Data-Hungry: Doesn't always need tons of data to make predictions.

ARIMA Disadvantages:

- Misses Complex Stuff: ARIMA might struggle with picking up on tricky, non-linear patterns in data.
- Needs Data to Behave: Works better if data doesn't jump around much or if it is consistent.
- Not Great for Long-Term Patterns: It's not so hot at catching really long-term trends in the data.

LSTM Advantages:

- Nails Complexity: It is great at catching inconsistent movements and long-term patterns in data.
- Flexible with Changing Data: Can adapt to sudden changes or irregular patterns in the data.
- Learns by Itself: Doesn't always need precise features given to it, it can figure out what's important.

LSTM Disadvantages:

- Needs a Lot of Data: Works better when you've got heaps of data to train it properly.
- Can Overdo It: Might get a little too good at predicting the training data and mess up with new data.
- Not So Clear on Why it Decides: It's like a mystery box - hard to say how it makes some predictions.

Additional improvements can be achieved by employing techniques like:

Sentiment Analysis: Incorporating sentiment analysis involves assessing the emotional tone of news articles, social media posts, or financial reports related to the stock. This analysis helps in capturing market sentiment, which can significantly impact stock prices, thus enhancing the model's understanding of external influences on stock behavior.

Feature Engineering: Feature engineering involves creating additional input variables, such as technical indicators (RSI, MACD, moving averages) and lagged variables (past stock prices/returns), enhancing the model's ability to capture diverse aspects of stock behavior & trends, thereby improving predictive accuracy.

Rolling Window Analysis: Utilizing rolling windows involves training the model on sequential segments of data, enabling adaptation to changing market conditions while preventing overfitting. This approach allows the model to continuously learn from recent data, reflecting the evolving nature of stock markets.

FaceBook PROPHET

Prophet is an open-source, additive model forecasting procedure developed by Facebook for time series data. Its core strengths lie in its ease of use, interpretability, and efficient handling of holidays and seasonality. Here's a detailed breakdown of its methods and functions:

The mathematical equation behind the Prophet model is defined as:

$$y(t) = g(t) + s(t) + h(t) + e(t)$$

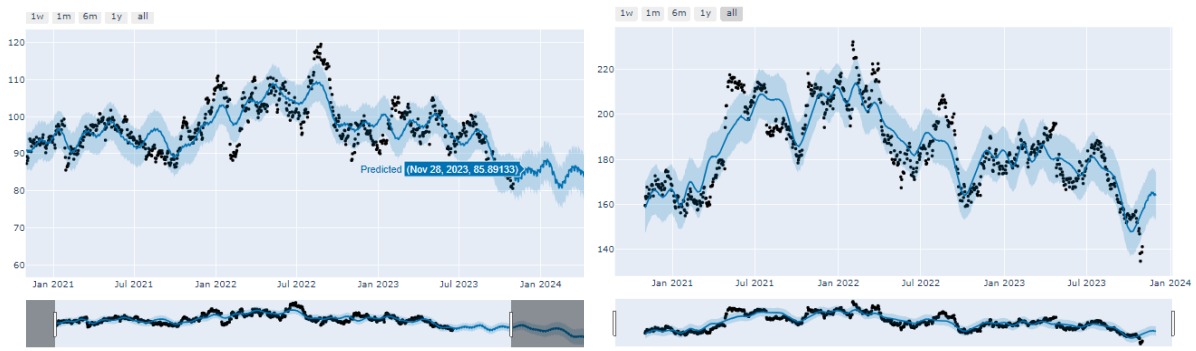
with, g(t) representing the trend. Prophet uses a piecewise linear model for trend forecasting.
s(t) represents periodic changes (weekly, monthly, yearly).
h(t) represents the effects of holidays (recall: Holidays impact businesses).
e(t) is the error term.

Trend: Modeled using a piecewise linear function with changepoints automatically detected from the data. This allows for flexible trend capturing, accommodating changes in direction and slope.
Seasonality: Prophet handles yearly, weekly, and daily seasonality using Fourier series with adaptive learning. This captures seasonal patterns without requiring prior knowledge of specific frequencies.
Holidays: Users can define custom holidays or utilize built-in holiday effects for major events like Black Friday. Prophet then models their impact on the time series.
Growth: An optional component for modeling exponential growth or decay.

Statistical Methods:
a) Bayesian regression: The underlying model is fit using Bayesian regression with a hierarchical prior structure. This facilitates automatic hyperparameter tuning and robust estimation.
b) Stan computational backend: The model fitting is performed in Stan, a probabilistic programming language, ensuring efficient and scalable computation.
c) MAP estimation: Prophet uses maximum a posteriori (MAP) estimation to learn the model parameters. This balances model complexity with data fit, reducing overfitting.

Key Functions:
a) fit(df): Trains the model on the provided time series data (df with ds for timestamps and y for target).
b) make_future_dataframe(periods): Generates a future dataframe for specified prediction periods.
c) predict(future_df): Uses the trained model to make forecasts for the future dates in the provided df.
b) plot(m, future_df): Visualizes the forecast along with trend, seasonality, and holiday components.
d) cross_validation(horizon): Performs k-fold cross-validation to assess model generalizability.

Advanced Features:
- Custom changepoints: Users can specify prior information about potential changepoints in the trend.
- Seasonality growth: Handles increasing or decreasing seasonality trends over time.
- User-defined holidays: Allows incorporating domain knowledge about specific events.
- Parameter tuning: Experienced users can adjust hyperparameters for further model refinement.



(a) CHRW Prophet Pred. (d) UPS Prophet Pred.

Figure 8: Forecast plots for CHRW & UPS using FBProphet.

Blue Line: This represents the predicted trend of data. It generally slopes upwards if your data is increasing or downwards if it's decreasing.

Grey Shaded Area: This is the confidence interval, showing the range within which you can expect the actual value to fall with a certain probability (usually 80%). The wider the shade, the higher the uncertainty in the prediction.

Black Dots: These are actual data points, plotted for reference.

My thoughts on Facebook Prophet:

Facebook Prophet, in comparison to ARIMA and LSTM, offers a simplified approach to time series forecasting, making it more accessible to a wider audience. Prophet's strength lies in its automated handling of seasonality, which is often a cumbersome task in ARIMA. While ARIMA requires manual identification and adjustment of seasonal components, Prophet's ability to automatically detect multiple seasonal patterns and incorporate them into the model streamlines the forecasting process significantly. However, this automated approach might limit its flexibility when dealing with irregular or complex data patterns that ARIMA can potentially handle with more manual intervention and customization.

On the other hand, LSTM, with its ability to capture intricate temporal dependencies and nonlinear relationships, outperforms Prophet in modeling complex data patterns. However, LSTM's strength in capturing nuanced relationships comes at the cost of complexity in architecture design, parameter tuning, and longer training times. Prophet, in contrast, sacrifices some of this complexity for simplicity, providing a more straightforward and interpretable model that allows users, especially those new to time series modeling, to obtain reasonably accurate forecasts without delving into intricate model configurations. Nonetheless, this simplicity may limit its accuracy in scenarios where capturing highly complex temporal dynamics is crucial, posing a trade-off between ease of use and modeling sophistication when compared to LSTM and ARIMA.

CONCLUSION:

In summary, our exploration delved into the Supply Chain and Logistics sector, analyzing non-stationary stock price forecasts for C.H. Robinson Worldwide (CHRW) and United Parcel Service (UPS) from November 1, 2020, to November 1, 2023, employing ARIMA and LSTM models after careful attention to details and multiple parameter selections. We allocated 80% of the data for training the models and reserved 20% for out-of-sample testing. Our analysis indicates that the ARIMA model outperformed LSTM marginally in short-term stock price prediction. Are we satisfied with our models? Yes, but there remains room for enhancement. It's critical to remember that no single model reigns supreme across the board, so a meticulous selection process based on trade-offs is crucial.

SOURCES:

"S&P 500 Tops Pre-Pandemic Level for First Time Since February" - The New York Times (Nov 9, 2020): https://www.nytimes.com/2022/07/29/business/stock-market-july.html

"The S&P 500 Officially Recovers From Its Pandemic Low" - Investopedia (Nov 9, 2020): https://www.investopedia.com/articles/investing/090414/sp-500-index-you-need-know.asp

“Predicting Stock Prices Using Facebook’s Prophet Model” - Medium.com
https://medium.com/analytics-vidhya/predicting-stock-prices-using-facebooks-prophet-model-b1716c733ea6

“Stock market forecasting using Time Series Analysis With ARIMA model” - Analytics Vidhya
https://www.analyticsvidhya.com/blog/2021/07/stock-market-forecasting-using-time-series-analysis-with-arima-model/

https://www.statsmodels.org/dev/generated/statsmodels.tsa.arima.model.ARIMA.html
https://keras.io/
https://www.wikipedia.org/

+ Blackboard materials