QF2103 Project Report

Team 9

April 2024

Contents

1 Abstract

Effective trading strategies are essential for successful investment and risk management in financial markets. With the increasing complexity of global markets and the availability of vast amounts of data, there is a growing need to explore sophisticated techniques for analyzing and predicting market trends.
In this report, we assess two models - ARIMA (Autoregressive Integrated Moving Average) and a Fine-tuned Pre-trained Transformer (Via TimeGPT) - to evaluate their potential for generating profitable trading signals. Our study makes the following key contributions:

1. **Performance Evaluation**: We use backtesting to demonstrate the effectiveness of ARIMA and Pre-trained Transformer Models in generating trading signals against a baseline strategy of buying and holding.
2. **Signal generation strategies**: We investigate the impact of various strategies that incorporate factors like RSI, SMA and MACD while generating the trading signals based on the predicted prices.
3. **Strategies to mitigate volatility**: We explore ways to mitigate the impact of volatility, such as buying/selling only if the predicted change exceeds a threshold.
4. **Practical considerations**: We provide insights into the strengths and limitations to better inform investment decisions.

2 Method 1: ARIMA

2.1 Overview

The AutoRegressive Integrated Moving Average (ARIMA) model is a class of models that forecasts a time series. It is a generalisation and integration of the simpler AutoRegressive Moving Average [Brownlee, 2023]. Looking at each component separately, there are:

- Autoregression (AR): It emphasises the dependent relationship between a data point and the points before or after it.
- Integration (I): In order to obtain a stationary time series without trend or seasonality, differencing is applied. It usually involves subtraction of an observation from its preceding observation.
- Moving Average (MA): By taking a moving average based on lagged observations, the relationship between the observation and the residual error can be mitigated, resulting in more accurate predictions.

2.2 Detailed Explanation

Since the stock prices to be studied does not display seasonality, we have chosen the nonseasonal ARIMA model, classified as an "ARIMA (p, d, q)" model [Nau, 2014], where

- p = the number of autoregressive terms,
- d = the number of nonseasonal differences needed for stationarity, and
- q = the number of lagged forecast errors in the prediction equation.

First, with the definition of d, let y denote the d-th difference of Y, which gives
If $d = 0 : y_t = Y_t$
If $d = 1 : y_t = Y_t - Y_{t-1}$
If $d = 2 : y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) = Y_t - 2Y_{t-1} + Y_{t-2}$
By definitions of p, and q, the general forecasting equation can be written as

$$\hat{y}_t = \mu + \phi_1 y_{t-1} + ... + \phi_p y_{t-p} - \theta_1 e_{t-1} - ... - \theta_q e_{t-q}$$

In this formula,

- μ denotes the constant term, which is the average value around which the time series fluctuates.
- $\phi_1, \phi_2, ..., \phi_p$ are the parameters of the autoregressive (AR) components. They are the coefficients of the lagged variables of the time series y_t up to lag p. Then can be regarded as a measurement of the impact of previous observations on the current observation.
- $y_{t-1}, y_{t-2}, ..., y_{t-p}$ are the lagged values of the time series, up to lag p.
- $\theta_1, \theta_2, ..., \theta_q$ are the coefficients of the lagged errors e_t up to lag q. These terms capture the impact of past forecast errors on the current observation.
- $e_{t-1}, e_{t-2}, ..., e_{t-q}$ are the lagged errors of the model, up to lag q.

2.3 Methodology

2.3.1 Model Fitting on Training Data

We started by using `auto_arima` function on the training data, to find the best (p, d, q) orders for each stock. Auto_arima searches through different combinations of these parameters to find the best fit for the given time series data.

Stock	**Optimal Orders** (p, d, q)
AAPL.O	$(0, 1, 0)$
MSFT.O	$(1, 1, 0)$
INTC.O	$(1, 1, 0)$
AMZN.O	$(0, 1, 0)$
GS.N	$(1, 1, 1)$

Table 1: Optimal ARIMA orders for five stocks

2.3.2 Generating Price Predictions

Then we implemented the `get_pre` function to generate predicted prices from previous days' stock prices. The ARIMA model is first instantiated and fitted with the optimal order obtained, then the function generates a forecast for a single time step ahead. After making each prediction, the observed value is appended to the training data and used for updating the model to generate the next forecast. By updating the training data with the observed value, the model learns from the new data and adapts to any changes in the underlying process.
After fitting the model and making the forecast for the first observation in the test set, this process is repeated for each subsequent observation. For each iteration, the training data is updated with the observed value for the current time step, and a forecast is generated for the next time step. Using these predictions, we generated the signals using the naive strategy that if the predicted price for day X + 1 is greater than the price on day X, we go long, else we go short. These are the results we initially obtained:

	Returns without any tweaks	Returns using benchmark
AAPL.O	1.379305	1.575807
MSFT.O	0.882503	1.710198
INTC.O	2.046265	1.316821
AMZN.O	2.097086	2.078885
GS.N	0.911642	1.277704

Figure 1: Returns without optimization

Based on the current strategy, only two of the five stocks have returns higher than the returns from the benchmark strategy.

2.3.3 Generating Optimized Trading Signals

From the predicted prices, we explored different strategies to generate the trading signals. The strategies we explored are as follows:

1. Relative Strength Index (RSI) trading strategy: RSI is a momentum indicator that evaluates whether the stock is overvalued or undervalued, signalling buying and selling opportunities [Fernando, 2024]. It can be calculated as $\text{RSI} = 100 - \frac{100}{1+\frac{\text{Average Gain}}{\text{Average Loss}}}$. If the RSI value falls below the "low" threshold, it suggests that the stock is oversold, signalling a buying opportunity; if it rises above the "high" threshold, it suggests that the stock is overbought, signalling a selling opportunity.

2. Moving Average Convergence Divergence (MACD) trading strategy: MACD is an oscillator that combines two exponential moving averages (EMA) to indicate the momentum of the price trend. If the MACD line crosses above the signal line of the MACD indicator, there is a buying opportunity, and vice versa.

3. Strategy based on the deviation of current prices from previous true prices: In order to account for the noise and volatility of stock prices across consecutive days, we implemented a strategy where we compared the current predicted price with the previous true/predicted price, and only changed the trading position if the predicted price was above or below a certain small window of margin (computed using a small fraction of the previous price) around the previous price.

We experimented with all these strategies to generate the final trading signals. Additionally, we also tried combinations of these strategies (like using RSI and MACD simultaneously to generate signals), as well as a majority voting approach (If ≥ 2 out of the 3 strategies, suggest long, we go long, else we go short), to come up with the best set of signals for each stock. Aggregating signals from multiple trading strategies helped us in determining a more robust trading decision.

2.4 Parameters Experimented with and Chosen

The following are the parameters for this approach:

1. Order for ARIMA Model (For predicting prices)

2. Window, high and low values (For the RSI trading strategy)

3. Short window, long window and signal window (For the MACD trading strategy)

4. Margin percentage (For the strategy based on the deviation of current prices from previous true prices)

2.4.1 Analysis of Optimization

To further improve the performance of the ARIMA model, we experimented with the various parameters mentioned above. Particularly, to get the ideal order for the ARIMA model, we used the `auto_arima` function on the training data (explained earlier). For the RSI strategy to generate signals, we used a brute force approach to extensively experiment with the various values of the window size (range(5, 30, 2)), and the low (range(25, 40, 2)) and high thresholds (range(60, 65, 1)). By testing a range of values for these parameters, the aim was to identify combinations that yield the most effective trading signals. For the strategy based on the deviation of current prices from previous true prices, we experimented with various values of the window of margin (np.linspace(0.0, 0.01, 100)). By adjusting the window of margin, the strategy aimed to identify significant deviations in prices that could potentially indicate better trading opportunities compared to naively changing positions based on the day to day fluctuations in prices. Additionally, we also enabled these parameters to be different for the different stocks, and tweaked the parameters for each stock separately, in order to account for the diverse nature of the underlying patterns and volatilities of each stock. The results after optimization are as shown in Figure 2.

2.5 Potential Challenges and Risks

ARIMA models, while widely used for time series analysis and forecasting, have limitations that can hinder their effectiveness in capturing various features and patterns present in the data.

1. Due to their linear nature, ARIMA models may struggle to capture nonlinear relationships and interactions within the data and handle complex dynamics such as sudden shocks or regime changes. The linear framework may fail to adequately represent these nonlinear phenomena. Particularly, ARIMA models may perform poorly when the time series contains turning points, as the moving average model tends to generalize values and lose information about these critical points. [Verma, 2022] [Mehra, 2024]

2. ARIMA models may struggle with very short or very long time series. Short-time series may lack sufficient information for accurate pattern capture, while long-time series poses computational challenges. Estimating parameters for ARIMA models, especially for lengthy datasets, can be computationally intensive due to the moving average and integration components of the model. As a result, in some steps, ARIMA models compromise performance in favor of speed. The way auto.arima picks the best model is by fitting several models and calculating its AIC score. The model with the lowest score wins. However, so that the function can find a solution faster, the algorithm skips some steps and approximates the results so that fewer models are fitted.

3. Another significant limitation is ARIMA's inability to handle multivariate time series data, involving more than one variable. ARIMA models lack the capacity to capture interactions and dependencies between variables, such as the impact of external factors like marketing initiatives or competitive actions on the time series.

In summary, while ARIMA models provide a valuable framework for time series analysis, they have notable limitations in capturing complex dynamics, handling multivariate data, and addressing computational constraints. Careful consideration of these limitations and appropriate model adjustments are essential for improving the accuracy and robustness of ARIMA-based forecasts.

2.6 Results of Backtesting

Comparing the trading strategy (implemented as mentioned above) to the benchmark strategy of going long over the entire period, we obtained the following results for the returns:

	Best Low	Best High	Best Window	Best Pct Within	Returns	Returns using benchmark
AAPL.O	25.0	60.0	5.0	0.006364	2.370962	1.575807
MSFT.O	25.0	60.0	5.0	0.001212	2.218228	1.710198
INTC.O	25.0	60.0	5.0	0.000000	2.046265	1.316821
AMZN.O	25.0	60.0	5.0	0.000000	2.097086	2.078885
GS.N	25.0	60.0	5.0	0.000707	1.609137	1.277704

Figure 2: Returns after Optimization

Using this strategy, all five stocks have significantly higher returns than the benchmark strategy.

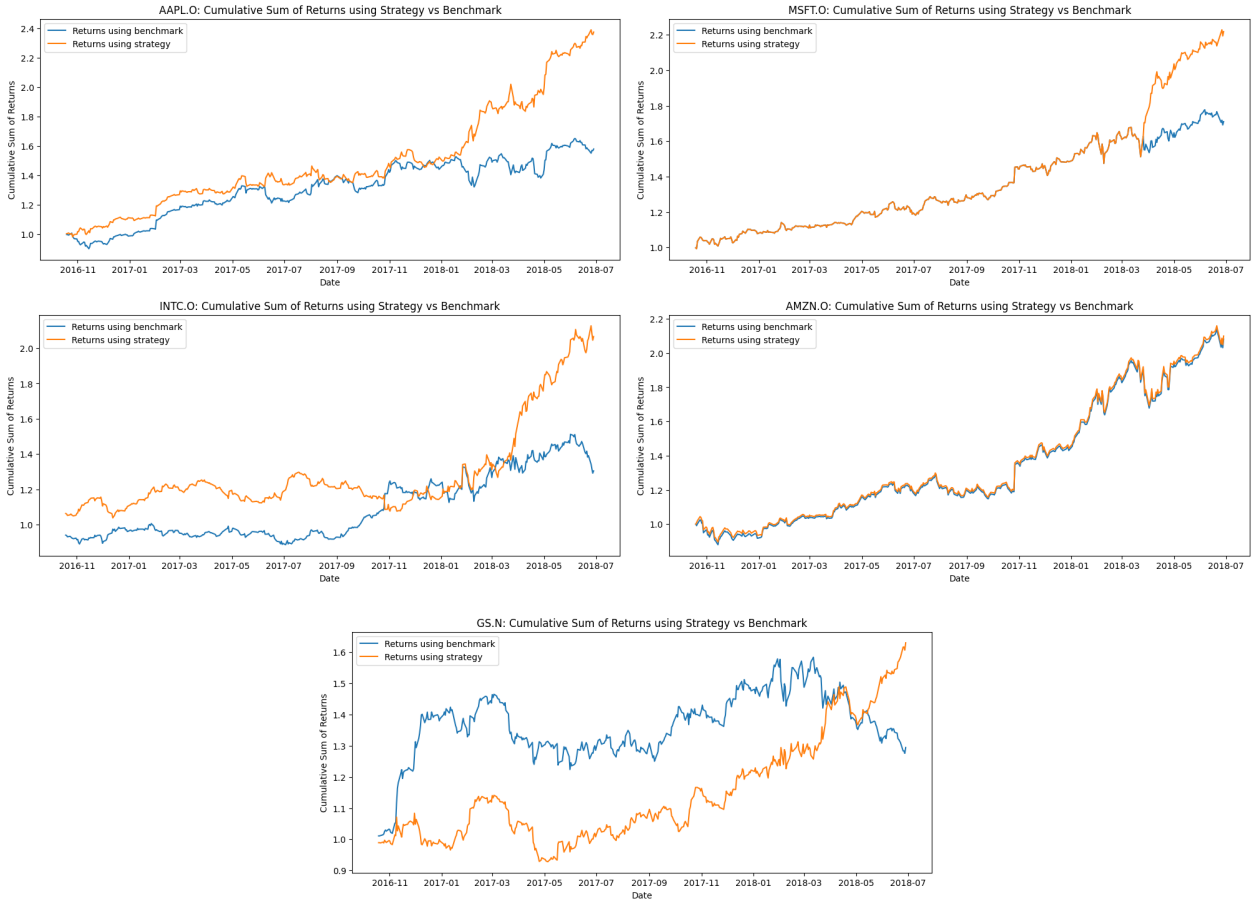

Figure 3: Cumulative sum of Returns using Strategy vs Benchmark

As we can see, for all the 5 stocks, the cumulative returns obtained by the strategy described above at the end of the time period in consideration is higher than the cumulative returns obtained by the benchmark strategy. Additionally, for 4 out of 5 of the stocks, the cumulative returns obtained using the strategy described above is higher than the cumulative returns obtained using the benchmark strategy for the majority of the time interval under consideration. This shows that the strategy has the potential to generalize well to other correlated stocks and other time intervals as well.

2.7 Evaluation and Performance Metrics

2.7.1 Evaluation of Predicted Prices

The selection of appropriate evaluation metrics depends on the specific problem and the expected results. Since regression problems typically use MSEs and MAEs, we will use these metrics to evaluate the prices predicted by the model.

- Mean Squared Error(MSE): MSE quantifies the presence of error in a statistical model by evaluating the mean squared difference between the observed and predicted values. Zero-error models have a zero MSE value that increases proportionally as the model error increases.
- Mean Absolute Error(MAE): MAE calculates the average absolute difference between predicted and actual values. It does not square the error, ensuring that all errors are weighted equally regardless of direction. This feature has proven to be beneficial in measuring the magnitude of errors and ignoring overestimates or underestimates.
- Mean Percentage Error (MPE) and Mean Absolute Percentage Error (MAPE): MAPE and MPE are percentage-based metrics that measure the average percentage difference between the predicted and actual values. MAPE is more suitable for smaller errors, while MPE is more appropriate for larger errors. [Kanaskar, 2023]

Stock	**MSE**	**MAE**	**MPE**	**MAPE**
AAPL.O	3.8000	1.3447	0.0984	0.8626
MSFT.O	1.0230	0.6626	0.1212	0.8317
INTC.O	0.4794	0.4467	0.0565	1.0269
AMZN.O	326.4015	11.8859	0.1610	1.0412
GS.N	10.7298	2.4249	0.0520	1.0272

Table 2: MSE, MAE, MPE, MAPE values for predictions

Except AMZN.O, MSEs of other predicted prices are quite close to 0 which indicates the high accuracy of predicted prices generated by ARIMA models. The reason for the large MSE of AMZN.O is that the prices of this stock are much higher than those of other four stocks, which leads to the larger differences between actual prices and predicted prices. Considering this, the MSE of AMZN.O is still in the acceptable range. A similar situation occurs with MAEs. Across all stocks, the MPE values are relatively low and MAPEs are moderate, suggesting minor deviations from the actual prices on average, indicating reasonably accurate predictions overall.

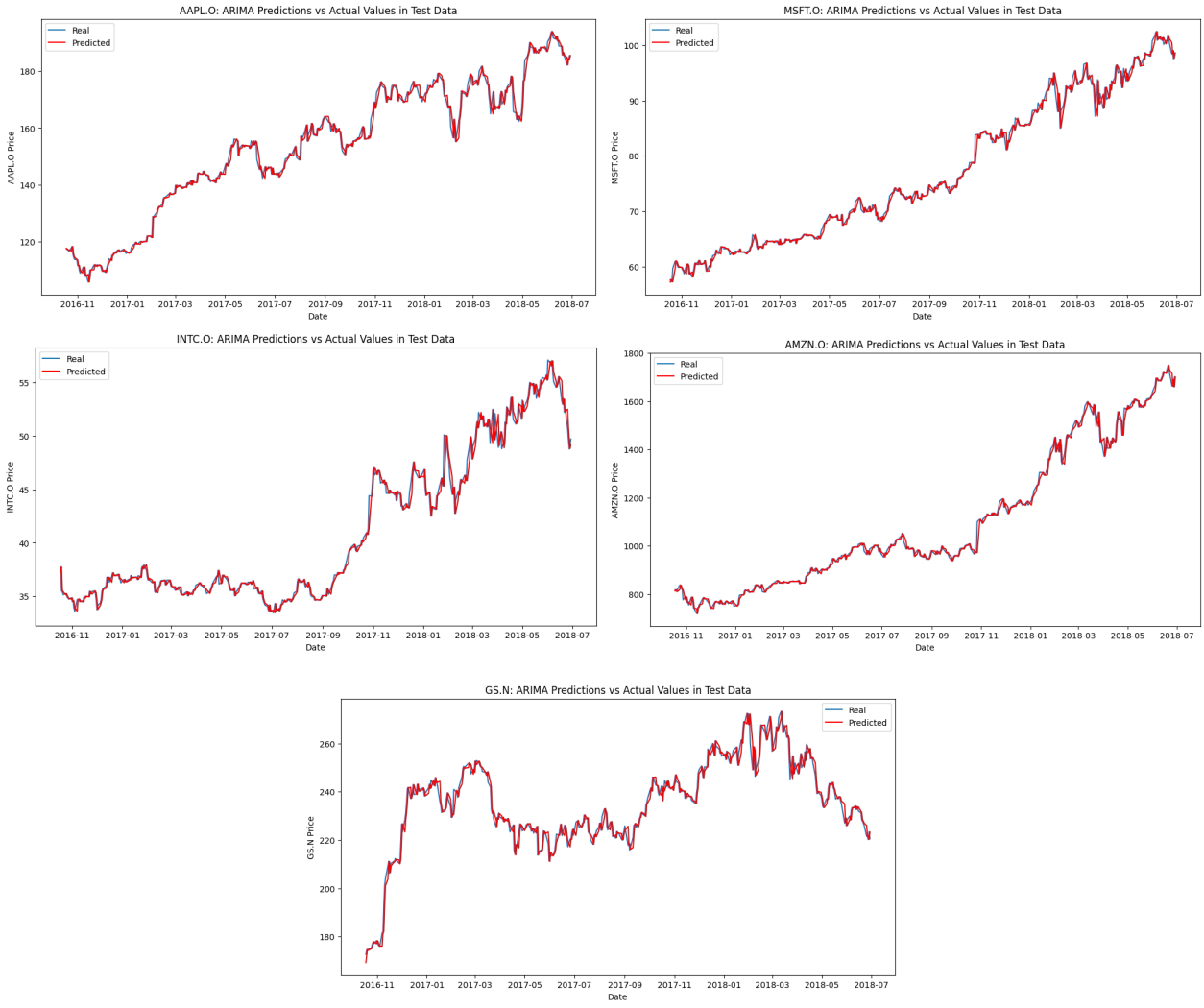

Figure 4: Predicted Prices vs Actual Values in Test Data

As seen in the above figure, the ARIMA model predictions for the stock prices are very similar to the actual values, indicating that the model does a good job at predicting the prices.

2.7.2 Evaluation of Trading Signals

We have compared the following accuracy metrics for the ARIMA trading strategy and benchmark strategy, as shown in Figures 3 and 4 respectively. Here we define going long or 'BUY' position as class 1 or the 'Positive' class, and going short or 'SELL' position as class 0 or the 'Negative' class.

The metrics we have used to evaluate the predictions are:

- Precision: It is a measure of how many of the positive predictions (going LONG) made by the model are correct (true positives).The ARIMA strategy has higher precision values for all five stocks.
- Recall: It measures how often the model correctly identifies true positives from all the actual positive samples (i.e. when going LONG would have been ideal) in the dataset. In this case, the ARIMA strategy gives similar values of recall compared to the benchmark strategy.
- F1 score: It is the harmonic mean of precision and recall. In trading, it gives an overall assessment of the model's ability to identify trading opportunities while minimizing false positives and false negatives.

As shown in the tables, the ARIMA strategy has higher F1 score for all of the five stocks.

- Number of false predictions: It simply counts the total number of predictions made by the model that turned out to be incorrect. In this case, ARIMA has lower or equal number of false predictions than the benchmark for only three stocks. It might be because in general, the stock prices of these large - CAP companies tend to display an upward trend, and hence, the benchmark strategy of holding long position throughout the period would yield a low number of false predictions, while positions from ARIMA have fluctuations resulting in a higher number of false predictions. However, as the expected returns using the signals predicted by the above strategy is much higher than the benchmark strategy, we can conclude that even though the number of false predictions are higher for some of the stocks, the gains from correct predictions outweigh the losses from incorrect ones for all stocks, justifying the efficacy of the model.
- Confusion Matrix: It summarizes the performance of a classification model by stating the number of True Positives, True Negatives, False Positives and False Negatives. As seen in Figure 7, for all 5 stocks, the model does a good job of predicting correct buying opportunities. However, apart from INTC.O and GS.N, for all other stocks, the model does not perform well in identifying correct selling opportunities, and classifies most of the ideally SELL positions as BUY instead.

	Precision	Recall	F1 Score	Number of False Predictions
AAPL.O	0.542136	0.548009	0.537756	193.0
MSFT.O	0.505319	0.533958	0.427934	199.0
INTC.O	0.479709	0.477752	0.475504	223.0
AMZN.O	0.430614	0.564403	0.414737	186.0
GS.N	0.532655	0.533958	0.533299	199.0

Figure 5: Metrics for ARIMA trading strategy

	Precision	Recall	F1 Score	Number of False Predictions
AAPL.O	0.275195	0.524590	0.361008	203.0
MSFT.O	0.285111	0.533958	0.371732	199.0
INTC.O	0.287617	0.536300	0.374429	198.0
AMZN.O	0.323860	0.569087	0.412800	184.0
GS.N	0.255889	0.505855	0.339859	211.0

Figure 6: Metrics for benchmark strategy

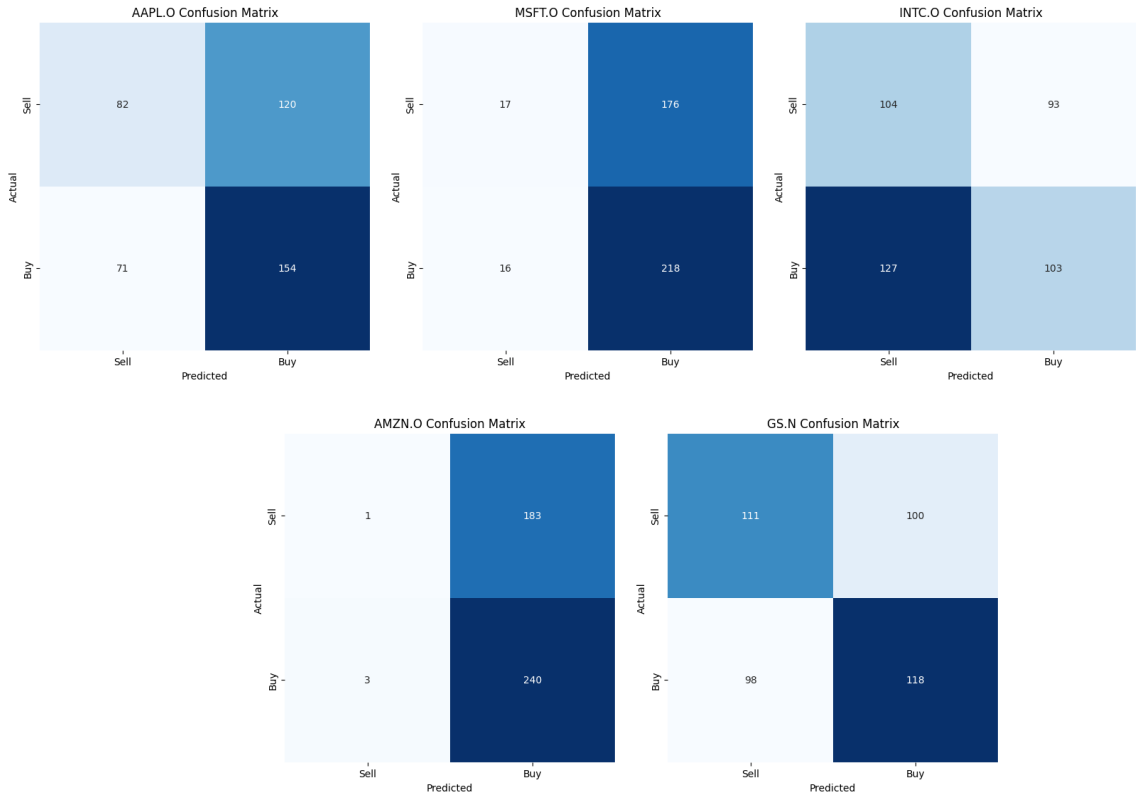

Figure 7: Confusion Matrices

2.8 Scope for Future Improvement

Considering the current limitations we face in ARIMA model, there are some possible areas for future improvement:

1. We can extend the ARIMA model to incorporate exogenous variables or external factors that may influence the time series data. Possible models to be used include the ARIMA with exogenous variables (ARIMAX) model, which integrates additional regressors into the model. With more information considered, the model might yield more accurate predicted prices.

2. Additional features can be incorporated to identify the underlying patterns in the data. For example, Fourier transformation is a mathematical technique based on the principle that all periodic functions can be written as a sum of simple geometric waves with different parameters. Through this method, we can break down the data into frequency components and examine how they affect the overall price patterns. [Kanaskar, 2023]

3 Method 2: Fine-tuning Pre-trained Transformer

3.1 Overview and Detailed Explanation

The Transformer [Vaswani et al., 2017] has achieved remarkable success in various sequence modelling tasks such as NLP [Devlin et al., 2018, Brown et al., 2020], Computer Vision [Dosovitskiy et al., 2020] and more recently, time series modelling [Nie et al., 2022]. It uses an encoder-decoder architecture composed of identical encoder and decoder blocks. Each encoder block consists of a multi-head self-attention module and a position wise feed forward network (FFN). Decoder blocks insert an additional cross-attention module between the self-attention and FFN. Its success is largely attributed to the self-attention mechanism which

allows the model to weigh and prioritize different parts of the input sequence when generating each output element. We provide a brief overview of the attention mechanism below.
Scaled dot product attention is given by:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{D_k}}\right)V,$$

where queries $Q \in R^{N \times D_k}$, keys $K \in R^{M \times D_k}$, values $V \in R^{M \times D_v}$ (M here is the number of keys, N is the number of queries, D_k and D_v are the embedding dimensions of keys and values respectively). Intuitively, the QK^T term is a similarity measure between a query and corresponding keys which indicates how much the transformer should focus on a value.

Multi-head attention performs the attention operation on H different learned projections of the queries and keys, ie

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_n)W^O,$$

where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$, W_i^Q, W_i^K, W_i^V, W^O are learned projection matrices.

In the context of time series forecasting, we want to learn a forecasting model $f_\theta : X \to Y$ mapping from feature space $X = \{y_{[0:t]}\}$ (stock prices for past t days) to target variable $Y = \{y_{[t+1:t+h]}\}$ (stock prices for the next h days), that estimates the conditional distribution $P(y_{[t+1:t+h]}|y_{[0:t]}) = f_\theta(y_{[0:t]})$ [Garza and Mergenthaler-Canseco, 2023].
The model we use, TimeGPT [Garza and Mergenthaler-Canseco, 2023], has been pre-trained on a large collection of time series with over 100 billion data points from a variety of domains such as economics, weather, IoT sensor data, energy, web traffic, sales, transport, and banking etc. We show the results of using the model zero-shot (i.e. without any training), and after fine-tuning on the train data.

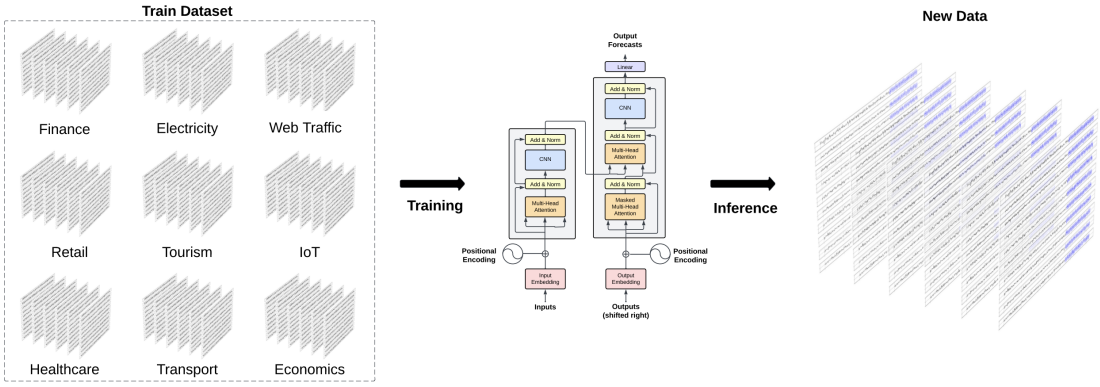

Figure 8: Model Architecture

3.2 Methodology

3.2.1 Single Model for Price Prediction for all Stocks

We observed that prices of 4 out of the 5 stocks (the tech stocks APPL.O, MSFT.O, INTC.O and AMZN.O) are highly correlated and correlation of GS.N with the other stocks is also moderately positive. Hence, we first tried to use the same model to forecast prices of all 5 stocks simultaneously. However, the results we obtained didn't beat the benchmark strategy, and hence we decided to forecast predictions for each stock separately.

3.2.2 Separate Models for Price Prediction for each Stock

Similar to the ARIMA Model, we generate predicted prices from previous day's stock prices. But instead of using the entire training data as context and making predictions for a single day in each step (too slow to

run), we define a context window C of size $|C|$ and a prediction window P of size $|P|$, and in each step, we predict $|P|$ values given C. After this, we shift the C by $|P|$ time steps before making the next prediction. We keeping doing this till we generate predictions for all time steps in the test data. Using these predictions, we generated the signals using the naive strategy that if the predicted price for day X + 1 is greater than the price on day X, we go long, else we go short. These are the results we initially obtained:

	Returns without any tweaks	Returns using benchmark
AAPL.O	3.361030	1.637417
MSFT.O	4.160490	1.711979
INTC.O	5.187056	1.316821
AMZN.O	4.059847	2.030072
GS.N	3.090793	1.367706

Figure 9: TimeGPT Returns without optimization

As we can see, even the zero-shot predictions (i.e. without any further finetuning) outperform the benchmark strategy by a huge margin and even considerably outperforms the ARIMA returns.

3.2.3 Generating Trading Signals

Now, instead of using the naive strategy for generating signals, we use the same strategy that we used to generate signals for the ARIMA model. We firstly get the predicted prices. Then we used a combination of RSI signals, MACD signals and signals generated using a strategy based on the deviation of current prices from previous true prices, to generate the final signals (Refer to 2.3.3 for more details).

3.2.4 Finetuning and other experiments that didn't work

Next, we tried to finetune the model on the training data set using the 'Mean-Squared-Error' and 'Mean-Absolute-Error' as loss functions and T number of fine tune steps. However, we didn't observe any improvement in returns for all stocks apart from INTC.O (where the returns increased from 5.3053 to 5.9962), possibly because of finetuning over a very small dataset. Additionally, we also tried normalizing/standardizing the prices using the `MinMaxScaler` and the `StandardScaler`, as well as using returns, differences in returns, or past ideal positions as input features instead of the prices. However, we obtained inferior performance for all of these techniques.

3.3 Parameters Experimented with and Chosen

The following are the parameters for this approach:

1. Context Window Size $|C|$
2. Parameter Window Size $|P|$
3. Window, high and low values (For the RSI trading strategy)
4. Short window, long window and signal window (For the MACD trading strategy)
5. Margin percentage (For the strategy based on the deviation of current prices from previous true prices)

3.3.1 Analysis of optimization

To further improve the performance of the TimeGPT Model, we experimented with the various parameters mentioned above, as well as the finetuning parameters (finetune steps and loss function, results mentioned in 3.2.4). For $|C|$ and $|P|$, the values that performed best were `300` and `20` respectively. For the RSI strategy to generate signals, we used a brute force approach to extensively experiment with the various

values of the window size (range(5, 30, 2)), and the low (range(25, 40, 2)) and high thresholds (range(60, 65, 1)). By testing a range of values for these parameters, the aim was to identify combinations that yield the most effective trading signals. For the strategy based on the deviation of current prices from previous true prices, we experimented with various values of the window of margin (np.linspace(0.0, 0.01, 100)). By adjusting the window of margin, the strategy aimed to identify significant deviations in prices that could potentially indicate better trading opportunities compared to naively changing positions based on the day to day fluctuations in prices. Additionally, we also enabled these parameters to be different for the different stocks, and tweaked the parameters for each stock separately, in order to account for the diverse nature of the underlying patterns and volatilities of each stock. The results after optimization are as shown in Figure 10.

3.4 Potential Challenges and Risks

1. Inference using TimeGPT is slow since the model has millions of parameters.
2. The data used to train TimeGPT is closed-source, as is the model itself; this means there is limited scope for customization (e.g. using other features).
3. Additionally, the library provides only limited fine-tuning capabilities and there is less transparency and interpretability regarding why the model makes a certain prediction (as is true for any transformer architecture model).

3.5 Results of Backtesting

Comparing the trading strategy (implemented as mentioned above) to the benchmark strategy of going long over the entire period, we obtained the following results for the returns:

	Best Low	Best High	Best Window	Best Pct Within	Returns	Returns using benchmark
AAPL.O	25.0	60.0	5.0	0.000101	3.465853	1.637417
MSFT.O	25.0	60.0	5.0	0.000000	4.160490	1.711979
INTC.O	25.0	60.0	5.0	0.001818	5.996240	1.316821
AMZN.O	25.0	60.0	5.0	0.004949	4.808534	2.030072
GS.N	25.0	60.0	5.0	0.002929	3.972071	1.367706

Figure 10: Returns after Optimization

Using this strategy, we can see that all five stocks have significantly higher returns than not only the benchmark strategy, but also the ARIMA Model strategy.

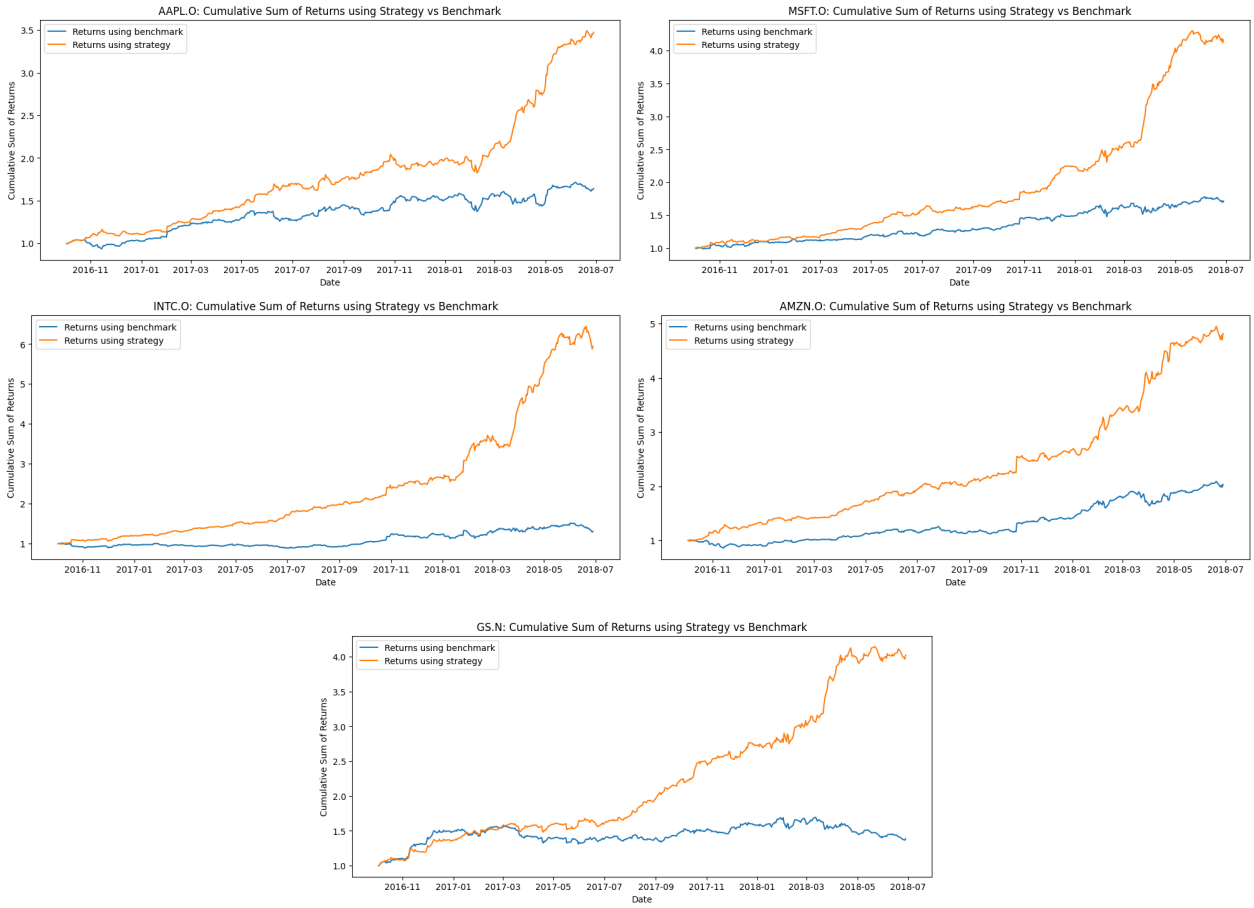

Figure 11: Cumulative sum of Returns using Strategy vs Benchmark

As we can see, for all the 5 stocks, the cumulative returns obtained by the strategy described above at the end of the time period in consideration is considerably higher than the cumulative returns obtained by the benchmark strategy. Additionally, for all of the stocks, the cumulative returns obtained using the strategy described above is higher than the cumulative returns obtained using the benchmark strategy for the majority of the time interval under consideration. This shows that the strategy has the potential to generalize well to other correlated stocks and other time intervals as well.

3.6 Evaluation and Performance Metrics

3.6.1 Evaluation of Predicted Prices

Here, we used the same metrics as for the evaluation of the ARIMA model in Section 2.7.1. The MSEs of predicted prices for MSFT.O and INTC.O are quite close to 0, indicating the high accuracy of the predicted prices generated by TimeGPT model. The reason for the large MSE for AMZN.O, and relatively large MSEs for AAPL.O and GS.N might be because the prices of these stocks are of a larger scale than the other stocks, leading to larger differences between the actual prices and the predicted prices.
As seen in the graphs below, the TimeGPT model predictions for the stock prices are very similar to the actual values, indicating that the model does a good job at predicting the prices. Even though the prices predicted are not as accurate as the prices predicted by the ARIMA model, we get better returns possibly indicating that the TimeGPT model does a better job at predicting the direction of movement of the stock rather than actual prices itself, when compared to the ARIMA model.

Stock	MSE	MAE	MPE	MAPE
AAPL.O	23.0643	3.3591	−0.2327	2.2211
MSFT.O	3.7883	1.3308	0.08386	1.7036
INTC.O	2.2986	1.0250	−0.64084	2.4144
AMZN.O	1756.0163	29.3655	−0.19428	2.5925
GS.N	66.9462	5.5853	−0.88256	2.4725

Table 3: MSE, MAE, MPE, MAPE values for predictions

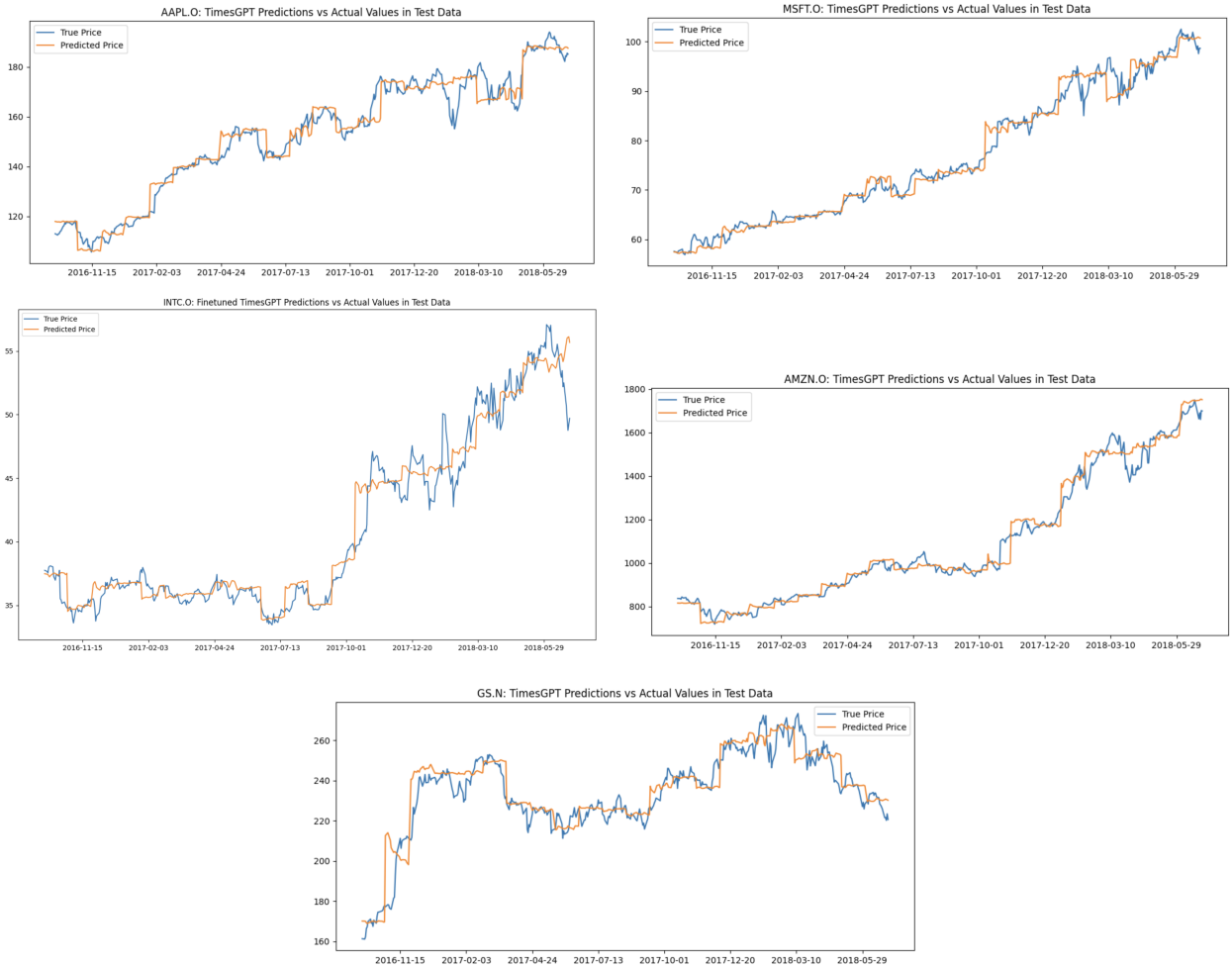

Figure 12: Predicted Prices vs Actual Values in Test Data

3.6.2 Evaluation of Trading Signals

We use the same metrics for evaluation as we did for the ARIMA model. The results are as shown in Figure 14:

- Precision: When compared to the ARIMA predictions, the results from TimeGPT have higher precision rates for all five stocks, showing a higher level of precision in predicting the directions of price movements.
- Recall: It measures how often the model correctly identifies true positives from all the actual positive samples (i.e. when going LONG would have been ideal) in the dataset. In this case, the TimeGPT

strategy gives similar values of recall compared to the benchmark strategy.

- F1 score: It is the harmonic mean of precision and recall. In trading, it gives an overall assessment of the model's ability to identify trading opportunities while minimizing false positives and false negatives. As shown in the tables, the TimeGPT strategy has higher F1 score for all of the five stocks.
- Number of false predictions: TimeGPT has a high number of false predictions (only slightly lower than the ARIMA predictions). This might be because as shown in the plots earlier, TimeGPT predicts the drastic changes in prices more accurately than the smaller changes, and the correct trading decisions made at these points generate returns that outweigh the losses from smaller changes.
- Confusion Matrix: As seen in Figure 15, similar to the ARIMA model, for all 5 stocks, the model does a good job of predicting correct buying opportunities. However, in contrast to the ARIMA model, for 3 out of the 5 stocks (MSFT.O, INTC.O and GS.O), the TimeGPT model performs well in predicting the correct selling opportunities as well, and for the remaining 2 stocks, it's performance is moderate when it comes to predicting correct selling opportunities. Hence, overall, the TimeGPT model is better at predicting both buying and selling opportunities, explaining the high returns obtained.

	Precision	Recall	F1 Score	Number of False Predictions
AAPL.O	0.526832	0.528474	0.527191	207.0
MSFT.O	0.568383	0.560364	0.560364	193.0
INTC.O	0.550737	0.558087	0.553365	194.0
AMZN.O	0.571754	0.571754	0.571754	188.0
GS.N	0.569524	0.569476	0.565840	189.0

Figure 13: Metrics for TimeGPT trading strategy

	Precision	Recall	F1 Score	Number of False Predictions
AAPL.O	0.281697	0.530752	0.368051	206.0
MSFT.O	0.293917	0.542141	0.381181	201.0
INTC.O	0.284121	0.533030	0.370665	205.0
AMZN.O	0.316566	0.562642	0.405168	192.0
GS.N	0.262685	0.512528	0.347346	214.0

Figure 14: Metrics for benchmark strategy

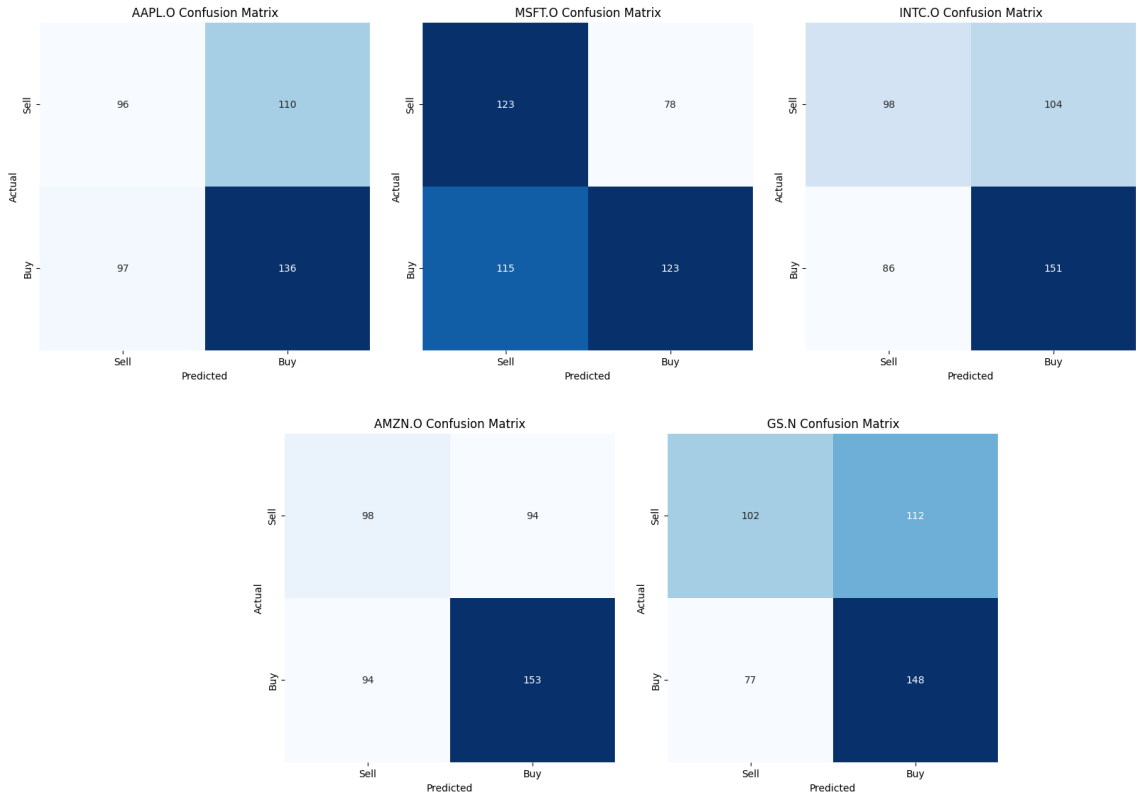

Figure 15: Confusion Matrices

3.7 Scope for Future Improvement

Considering the current limitations we face in TimeGPT model, there are some possible areas for future improvement:

1. If we use an open-source time series forecasting pre-trained transformer model instead, we will have a finer degree of control in the fine-tuning process, potentially generate better results by focusing on more relevant metrics for the types of stocks we are trying to trade.

2. Additionally, by using an open source model, we will be able to incorporate other features like RSI, MACD and SMA, as well as other external features like trading volumes, spreads, etc. within the fine-tuning process itself.

4 Comparison of Two Methods

4.1 Performance

4.1.1 Performance of Predicted Prices

In terms of the accuracy of the predicted prices, measured by metrics like MSEs, the predictions from the ARIMA model has lower error values (Table 2) than those of the TimeGPT model (Table 3). This is because as shown in the predicted prices of the TimeGPT model in Figure 12, it predicts the drastic changes in prices more accurately than the smaller changes, while the ARIMA model is able to better capture the smaller-scale changes, resulting in lower MSE values.
In terms of the accuracy based on f1 score, ARIMA and TimeGPT scored relatively similar except for MSFT.O, INTC.O and AMZN.O. This can be attributed to the more drastic shifts in stock prices within

those stocks, which ARIMA might not be as adequete to predict while TimeGPT might have been trained on those possible stock price shifts within its pre-training data.
In general, both models are fairly accurate in predicting the stock prices, with ARIMA better at capturing small fluctuations in prices and TimeGPT better at predicting the drastic changes in prices.

4.1.2 Performance of Returns from Trading Signals

TimeGPT outperformed the ARIMA model in terms of returns in backtesting. ARIMA consistently outperformed the benchmark (simply holding the stock throughout the time period). As shown in the comparisons in Figure 2, the returns of the ARIMA model after fine-tuning consistently beat those of the benchmark strategy. Compared to the returns from the ARIMA model, the TimeGPT model more than doubled the benchmark returns, as shown in comparisons of returns in Figures 2 and 10, representing a marked improvement over ARIMA.
This might be because the TimeGPT is pre-trained with larger data sets, resulting in a more robust model in predicting the market trend and generating trading signals to maximise the returns.

4.2 Interpretability

ARIMA is a simple linear regression model and its predictions can be easily interpreted. TimeGPT has millions of parameters, so it is very hard to reason about why a prediction was made. This makes debugging harder if the model performs poorly.

4.3 Computation

Large transformer models like TimeGPT require GPUs with sufficient VRAM to load them for inference. Meanwhile ARIMA is much more lightweight in terms of data input and computations due to its simple linear regression model. This also means that ARIMA's inference speed is much faster than TimeGPT for a fixed compute budget.
Overall, we believe that given the hardware capabilities to run such a large model, a transformer based trading strategy is preferred due to its better observed performance during our testing. However, practical constraints make the ARIMA model a more feasible choice due to its less demanding requirements.

5 Team Roles and Responsibilities

5.1 Project Tasks

1. Model exploration
    - Hidden Markov Model
    - LSTM
    - ARIMA
    - Random Forest
    - Pre-trained Transformers - TimeGPT, Moirai
2. Tweaking and Optimizing the 2 chosen models
3. Report writing
4. Presentation slides

5.2 Team Member Involvement

Member	Model(s) trained and explored
Jayanth	Moirai
Jingyi	Hidden Markov Model
Peigeng	Random Forest
Pratham	LSTM, TimeGPT
Menghan	ARIMA

Table 4: Table of models trained by each member

All team members contributed equally to the report, slides and optimizing the model.

References

[Brown et al., 2020] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

[Brownlee, 2023] Brownlee, J. (2023). How to create an arima model for time series forecasting in python. *Machine Learning Mastery*.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Dosovitskiy et al., 2020] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[Fernando, 2024] Fernando, J. (2024). Relative strength index (rsi) indicator explained with formula. *Investopia*.

[Garza and Mergenthaler-Canseco, 2023] Garza, A. and Mergenthaler-Canseco, M. (2023). Timegpt-1. *arXiv preprint arXiv:2310.03589*.

[Kanaskar, 2023] Kanaskar, S. (2023). Predicting stock prices: using arima, fourier transformation and deep learning. *Medium*.

[Mehra, 2024] Mehra, N. (2024). What are the advantages and disadvantages of using arima models for forecasting? *LinkedIn*.

[Nau, 2014] Nau, R. (2014). Notes on nonseasonal arima models. *Fuqua School of Business, Duke University*.

[Nie et al., 2022] Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. (2022). A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[Verma, 2022] Verma, Y. (2022). 5 conditions when the arima model should be avoided. *AIM*.