Reinelle Jan Bugnot
G2304329L
Nanyang Technological University, Singapore

May 2, 2024

PROJECT 3

1 Introduction

The data from the daily historical Apple stock (AAPL) price from February 1, 2002 to January 31, 2017, were extracted from the Yahoo finance website. This contains a record of the Apple stock prie every day (3,775 attributes), recorded as Open, Close, Low, High, and Adjusted Close prices in the span of 15 years. The objective of this project is to uncover notable trends in Apple's stock price and devise the most effective model for forecasting.

Data Preprocessing

The Apple stock price data from February 1, 2002, to January 31, 2017, was sourced from the Yahoo Finance website. In Python, the data was imported and the date frequency was adjusted to 'Business days' to align with valid data points, resulting in 137 null values corresponding to weekends and holidays when the market is closed. To handle these null values, a "forward fill" method was employed, substituting null dates with the most recent valid date prior.

Additionally, the dataset comprises five features: Open, Close, Low, High, and Adjusted Close. For forecasting purposes in this project, focus was placed on the Adjusted Close price indicator. This choice is pivotal in technical analysis, where historical price data plays a crucial role in predicting future price movements. Adjusted close price provides a more accurate depiction of a stock's historical performance, facilitating precise calculations of returns on investment and accounting for the impact of significant events.

Lastly, the data originally has 3,775 entries, corresponding to all the valid market days within the 15-year period. Since we are more interested in predicting the overall movement of the AAPL stock price, we can mute the high frequency fluctuations by resampling the data to a monthly period. This also provides us with the added benefit of smoothing the series and simplifying our calculations by reducing the number of data points within the time series from 3,775 to 180. Fig. 1 shows the plot of the Adjusted close price of Apple (AAPL) in the 15-year span of interest for both original data and the resampled version.

Figure 1: Plot of the Original Data (Gray) and the Re-sampled version (Blue)

The 15-year period was split into training and testing, where the first 12 years of data was used for training and the last 3 years (36 months) was used for testing. This step is crucial in the evaluation of our models' performance in the latter part of the report.

2 Exploratory Data Analysis

Seasonal Decomposition

From the plot in Fig. 1, we can visually notice key time series components such as trend and seasonality. We can further analyze this by performing seasonal decomposition using seasonal_decompose() function of statsmodels python library. This method allow us to isolate and study the trend, seasonal, and residual components of our time series data. The trend component captures the underlying pattern in the data, while the seasonal component captures the regular fluctuations due to seasonal factors. The residual component, on the other hand, represents the irregular influences that are not captured by the trend or seasonal components.

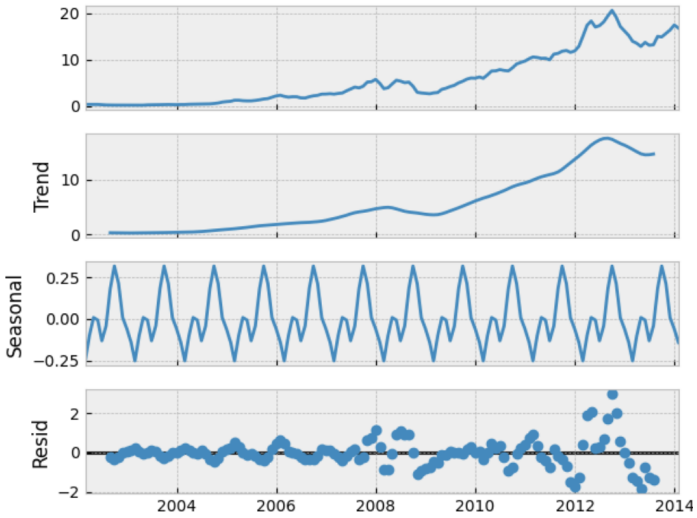

Figure 2: Seasonal Decomposition

Fig. 2 shows the decomposition of our training data into trend, seasonal, and residual components. The trend component shows an increasing trend over the year, which is immediately noticeable from the original data. This tells us that the data is not stationary, and therefore, differencing must be performed later to fit an appropriate ARIMA model. The seasonal component is interesting because the pattern is consistent where a smaller peak is followed by a larger peak. Zooming in further, we can see from Fig. 3 that these peaks happen at regular intervals, where the smaller peak occur around April, while the larger peak occur around October of every year.

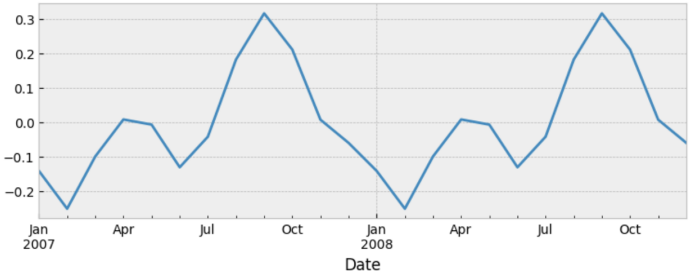

Figure 3: Seasonal Component (zoomed in)

In the context of Apple, this is reasonable because these peaks align with Apple's product release cycle. Typically, Apple releases new products or updates to existing lines twice a year - once in the spring (around April) and once in the fall (around October). The smaller peak in April could be attributed to the anticipation and subsequent release of new or updated products, such as the iPad or MacBook Pro. The larger peak in October is likely due to the release of new iPhone models, which traditionally happens in this month. This

pattern of product releases leads to increased public interest and discussion, which is reflected in our time series data [2].

Finally, through visual inspection on Fig. 2, we can see that there is no varying seasonal component (the seasonal pattern is constant across the 12-year training data period). This is a crucial observation as it allows us to use models that assume a constant seasonal component, such as SARIMA for our forecasting, and possibly without needing to transform the data beforehand using log-transform or Box Cox methods.

Stationarity

One key assumption of using models like ARIMA/SARIMA is that the model is stationary. This means that the properties of the time series do not change over time. In other words, the mean, variance, and autocorrelation structure of the series should remain constant. If a time series is non-stationary, it exhibits trends or seasonality, which can affect the model's ability to make accurate forecasts. Therefore, before applying ARIMA/SARIMA models, we often need to transform the original time series into a stationary one [1]. To do this, we used the Augmented-Dickey Fuller (ADF) test to check for stationarity. Specifically, we look at the output p-value and the test statistic output of the ADF test. The null hypothesis of the ADF test is that the time series is non-stationary. In order to reject this hypothesis, we must fulfill two criteria:

1. p-value < 0.05
2. Test Statistic output value $<$ Critical Values

Table 2 shows the results of ADF tests applied to the data until the stationarity conditions were achieved. Without differencing, we can see that the Test Statistic is greater than the critical values, and the p-value is greater than 0.05. This means we do not have sufficient evidence to reject the null-hypothesis, and therefore, conclude that the original data is non-stationary. After applying differencing once, the Test Statistic value is now less than 2 out of 3 critical values, and the p-value is less than the 0.05 threshold, indicating that the differenced series is stationary.

Diff Order (d)	Test Statistic	C.V. 1%	C.V. 5%	C.V. 10%	p-value
0	0.07946	-3.48168	-2.88404	-2.57877	0.96463
1	-3.46152	-3.48168	-2.88404	-2.57877	0.00903

Table 1: ADF Test Results

Seasonality

Seasonality in a time series is a regular pattern of changes that repeats over S time periods, where S defines the number of time periods until the pattern repeats again. From Fig. 2, we observed that the data contains seasonal components. This means that instead of the regular ARIMA model, we can opt to use its variant called SARIMA, which handles seasonal components as well. From here, we need to perform seasonal differencing in order to remove the seasonal trend present in the data. Doing so will give us an idea of the order of the seasonal differencing parameter for the SARIMA model that we'll fit later.

In order to perform seasonal differencing we subtract the value of the time series at a given point by the value of the time series at a point one season prior. Mathematically, this can be represented as:

$$D_t = Y_t - Y_{t-s} \tag{1-1}$$

where D_t is the differenced series, Y_t is the original series, and S is the length of the season. In this case, $S = 12$, corresponding to 12 months. This operation removes the seasonal effect and results in a series that can be modeled using standard ARIMA. Table 2 shows the results of ADF tests applied to the data sequentially until seasonal stationarity is achieved. Using the same test criteria, we can see that the Test Statistic values as well as the p-values after 1 and 2 orders of differencing does not meet the aforementioned criteria. At differencing order 3, we see that the Test Criteria is now less than the critical values, and the p-value is now less than 0.05, which means we can reject the null hypothesis and conclude that the series has seasonal stationarity.

S. Diff Order (D)	Test Statistic	C.V. 1%	C.V.5%	C.V. 10%	p-value
1	-2.05742	-3.48702	-2.88636	-2.58000	0.261975
2	-2.63511	-3.49360	-2.88921	-2.58153	0.085955
3	-3.27458	-3.48168	-2.88404	-2.57877	0.016053

Table 2: ADF Test Results for Seasonal Differencing

Autocorrelation

The next step is to use the ACF and PACF plots to determine the order of the autoregressive (AR) and moving average (MA) terms in our SARIMA model. The Autocorrelation Function (ACF) plot gives us an idea of the MA order, while the Partial Autocorrelation Function (PACF) plot helps us identify the AR order. We look for a sharp cut-off in the ACF plot after a certain number of lags, which suggests the order for the MA term. Similarly, a sharp cut-off in the PACF plot indicates the order for the AR term. Once we have these orders, we can fit a SARIMA model to our time series.

From the ACF plot in Fig. 4, we can see that there is a significant spike at lag 0, which is expected as a data point is perfectly correlated with itself. There is also a notable spike at lag 1, suggesting a positive autocorrelation at one time interval apart. Additionally, there are small negative spikes at lags 9 and 10, indicating a slight negative autocorrelation at those intervals. This pattern of spikes could be indicative of the underlying data generating process and may suggest that past values have an influence on future values up to one time interval apart, with some seasonality or repeating patterns occurring around lags 9 and 10.

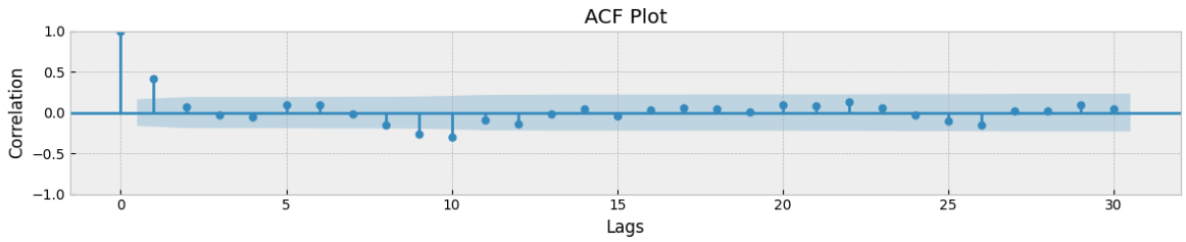

Figure 4: ACF plot

Similarly, from the PACF plot in Fig. 5, we can see that there is a significant spike at lag 0 and lag 1, and a negative spike at lag 12. The spike at lag 1 suggests a strong positive correlation between a value and its immediate predecessor, indicating a potential autoregressive term of order 1. The negative spike at lag 12, which corresponds to our seasonal period of 12 months, suggests a negative autocorrelation at the seasonal lag. Given that we are dealing with Apple stock price data, this could reflect annual patterns in the stock's performance.

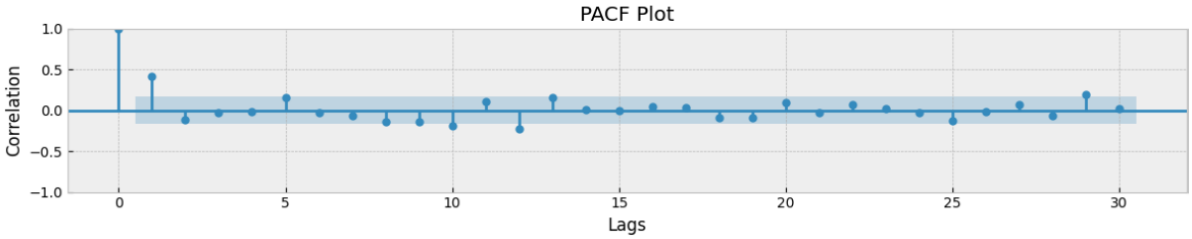

Figure 5: PACF plot

From the ACF and PACF plots, we can choose an AR and MA order of 1; i.e., $ARMA(p, q) = ARMA(1, 1)$, as a starting point for further experiments.

3 Model Fitting and Benchmarks

For this project, we will fit 4 different models on the training series. 3 of those will be SARIMA models whose parameters are identified in 3 different ways: (1) Semi-Automatic, (2) Brute-Force, (3) Fully-Automatic. The 4th method will be implemented using FB Prophet as a representative for more modern approaches to time series forecasting.

Semi-Automatic using Auto ARIMA

Following from the results of the exploratory data analysis performed in the previous section, we identified the following parameter values for the SARIMA(p, d, q)x(P, D, Q) model that we want to fit:

- p = 1 : order of the AR component
- d = 1 : order of the Differencing component
- q = 1 : order of the MA component
- P: unidentified
- D = 3 : order of the Seasonal Differencing component
- Q: unidentified

We can use this as a starting point, SARIMA(**1, 1, 1**)x(P, **3**, Q), and use auto-arima to iteratively search for the optimal P and Q values, based on the *Akaike information criterion* (AIC).

Brute-Force

The second approach is by using a brute-force search to manually test different combinations of order values and selecting the best one based on a set criterion. In this case, for simplicity, I used AIC to select the best model. To allow the search method to test a larger parameter space, I fixed the differencing and seasonal differencing orders to 0 and 3, respectively, and then set p, q, P, and Q parameters as the search variables; i.e., SARIMA(p, **1**, q)x(P, **3**, Q). For computational efficiency, I limited the search space to a range of 0 to 3, which is roughly based from the intuition derived from the ACF and PACF plots in the previous section. Shown below is the code used to execute this brute force search approach.

```
    p = q = range(0, 3)
    pdq = list(itertools.product(p, [1], q))
    seasonal_pdq = [(x[0], 3, x[1], 12) for x in list(itertools.product(p, q))]

    lowest_aic = np.inf
    best_config = []
    best_results = []

    for param in tqdm(pdq):
        for param_seasonal in seasonal_pdq:
            try:
                model = SARIMAX(train,
                                order=param,
                                seasonal_order=param_seasonal,
                                enforce_stationarity=False,
                                enforce_invertibility=False)

                results = model.fit()

                if results.aic < lowest_aic:
```

```
                    best_config = [param, param_seasonal]
                    best_results = [results.aic]

            except:
                continue
```

Fully-Automatic

The third approach is to fully leverage auto-arima by searching on all p, d, q, P, D, and Q parameters. Here, I set the search range as 0 to 3. Auto-arima, in this context, performs a comprehensive search over the specified parameter space, evaluating each combination of parameters to find the best fitting model. It uses the Akaike Information Criterion (AIC) to compare different models and select the one that provides the best balance between model complexity and goodness of fit.

```
    # AUTO ARIMA
    sarima_auto = pm.auto_arima(train,
                                start_p=0, max_p=3,
                                start_d=0, max_d=3,
                                start_q=0, max_q=3,
                                start_P=0, max_P=3,
                                start_D=0, max_D=3,
                                start_Q=0, max_Q=3,
                                m=12,
                                seasonal=True,
                                stepwise=True,
                                suppress_warnings=True,
                                error_action='ignore')

    print(sarima_auto.order)  # (p, d, q)
    print(sarima_auto.seasonal_order)  # (P, D, Q, s)
```

FB Prophet

The final method uses FB Prophet to fit a model for time-series forecasting. FB Prophet is a powerful and flexible open-source library developed by Facebook for forecasting time series data. It is based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data [3]. One of the key advantages of FB Prophet is its ability to handle missing data, outliers, and shifts in the trend . It also provides intuitive parameters that can be easily tuned. In the next step, I will use FB Prophet to fit a model to our data, taking into account the trend and seasonal components we observed in our exploratory analysis. I will then evaluate the performance of this model and compare it with the results obtained from the SARIMA model.

4 Results

Model	AIC
(Full) AUTO: SARIMA(0, 1, 1)x(0, 0, 1, 12)	218.58
Brute-Force: SARIMA(2, 1, 2)x(2, 3, 2, 12)	264.27
(Semi) AUTO: SARIMA(1, 1, 1)x(2, 3, 0, 12)	268.42

Table 3: Best SARIMA Models for each of the 3 methods and their AIC values

Model	RMSE	MAPE	R2
FB Prophet	4.231287	12.784535	-0.593321
Brute-Force: SARIMA(2, 1, 2)x(2, 3, 2, 12)	4.982600	19.093121	-1.209380
(Full) AUTO: SARIMA(0, 1, 1)x(0, 0, 1, 12)	8.959629	32.682766	-6.143951
(Semi) AUTO: SARIMA(1, 1, 1)x(2, 3, 0, 12)	10.739801	36.868868	-9.264808

Table 4: Benchmark Results

Table 4 presents the best SARIMA model configurations for each method described in the previous chapter, and their corresponding AIC values. Here, we can see that the Fully-Auto SARIMA model, SARIMA(0, 1, 1)x(0, 0, 1, 12) achieved the lowest AIC score of 218.58, indicating that is has the best goodness-of-fit among the three SARIMA models in the training data. The next step is to test the performance of the 4 models (3 SARIMA models + FB Prophet model) on the hold-out test set, corresponding to the last 3 years (36 months) of the original stock price data. The metrics used in this project are RMSE, MAPE, and R2. Table 4 shows the benchmark results for all 4 models, where we see that the FB Prophet model was able to outperform all SARIMA models, demonstrating the exceptional fitting capabilities of modern forecasting methods.

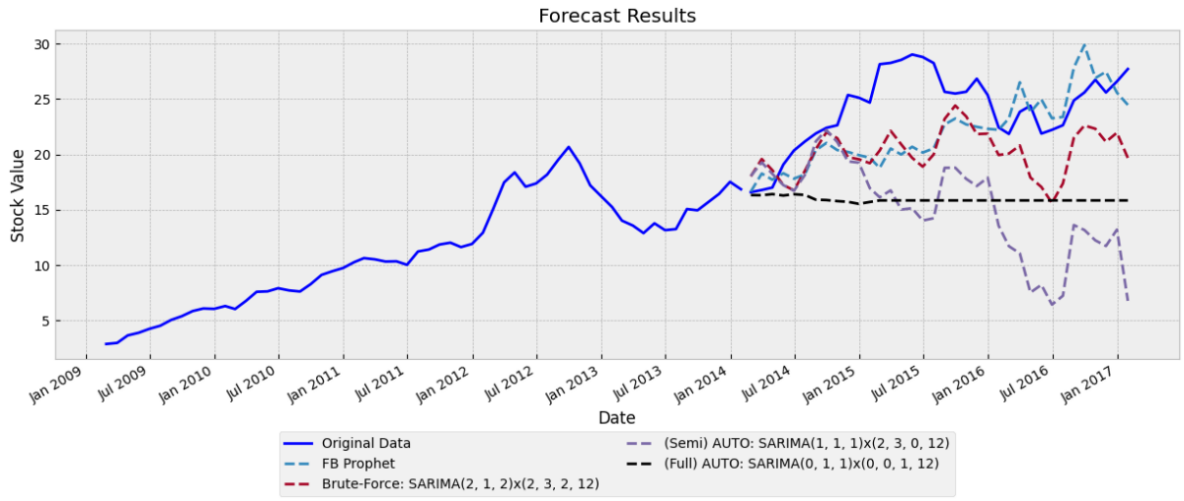

Figure 6: Forecast Results of all 4 models

Among the three SARIMA models fitted on the training data, the model generated from brute-force search achieved the best results on the test set. This finding is intriguing because, as observed in Table 4, the model from the fully-automatic approach exhibited the lowest AIC score. This suggests a potential case of overfitting, wherein the model performed exceptionally well on the training data but performed worse on the testing data. This observation is further supported by the forecast results depicted in Figure 6. The black dashed line, representing the SARIMA model from the fully-automatic approach, notably failed to accurately capture any essential components of the testing series.

Among all the models tested, both FB Prophet and Brute-force SARIMA exhibited outstanding performance, as evidenced by the benchmark results and forecast graphs. Notably, these models effectively captured the seasonal pattern of the original data, as depicted in Figure 7. The blue and red dashed lines was able to adequately represent the small and large peaks observed in the original time series, aligning with the annual sales cycle of Apple. In contrast, the SARIMA model from the semi-automatic approach, represented by the black dashed line, completely failed to capture these peaks, leading to a poor fit and forecast. This highlights the importance of model selection and hyperparameter tuning in time series forecasting. While automatic methods can provide a good starting point, they may not always yield the best results, especially when the data exhibits complex seasonal patterns. Meanwhile, modern forecasting methods like FB Prophet, which are designed to handle such complexities, often outperform traditional methods without much analytical tuning.

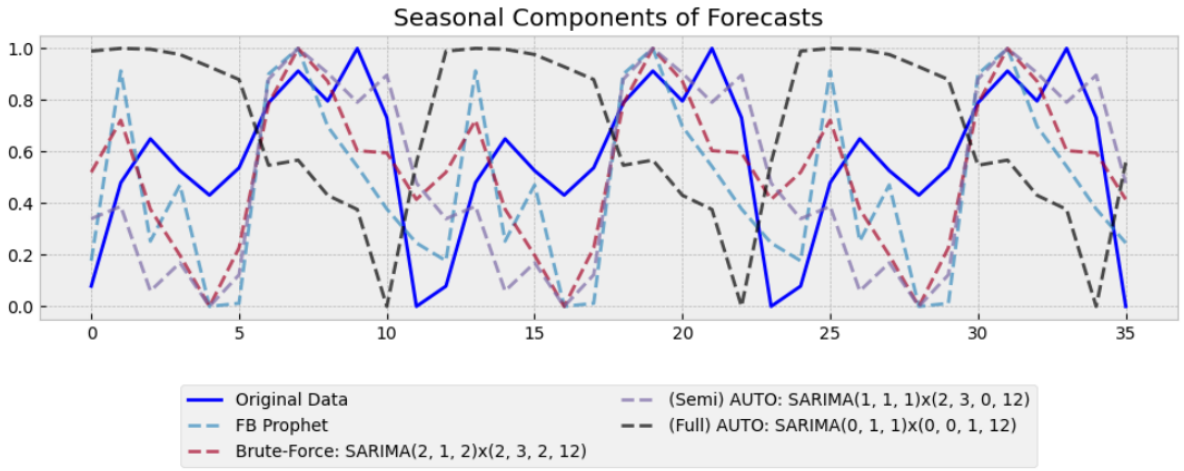

Figure 7: Inspecting the Seasonal Component of all the forecast series (normalized)

5 Conclusion

In conclusion, this project has successfully uncovered notable trends in Apple's stock price; specifically, we were able to clearly observe how the annual sales cycle of Apple influence the company's stock price, and devised effective models for forecasting. The exploration of different methods, including SARIMA and FB Prophet, has provided valuable insights into the strengths and weaknesses of each approach. While the SARIMA models, particularly the one obtained through brute-force search, demonstrated decent performance, the FB Prophet model outperformed them all. This highlights the power of modern forecasting methods, which are designed to handle complex seasonal patterns and trends, making them particularly suitable for stock price prediction.

Furthermore, the project has reinforced the importance of careful model selection and hyperparameter tuning in time series forecasting. The fully-automatic SARIMA model, despite having the lowest AIC score, did not perform as well on the test data, suggesting a potential case of overfitting. On the other hand, the FB Prophet model, which requires less analytical tuning, was able to capture the seasonal pattern of the original data effectively and provided the most accurate forecasts. As we move forward, these insights will guide our efforts to further improve the accuracy of our forecasts and deepen our understanding of the factors driving Apple's stock prices. Surely, more sophisticated models can be utilized to create a more robust, accurate, and consistent model forecasts. Regardless, this project, which utilized most of the fundamental analytical time series concepts learned in class, serves as a stepping stone towards these more advanced models.

References

[1] Brendan Artley. "Time Series Forecasting with ARIMA, SARIMA and SARIMAX". In: (2022).

[2] Karen Haslam. *New Apple Products 2024: Upcoming Apple product releases*. 2024.

[3] Sean J Taylor and Benjamin Letham. "Forecasting at scale". In: *The American Statistician* 72.1 (2018), pp. 37–45. URL: `https://peerj.com/preprints/3190.pdf`.

Submitted by Reinelle Jan Bugnot on May 2, 2024.