HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

**School of Information and Communications Technology**



**Statistical applications to economics, modelling of economics and financial data**

**Subject: Applied Statistics and Experimental Design**

|  |  |
| --- | --- |
| **Supervisor:** | **Nguyen Linh Giang** |
| **Members:** | **Chu Trung Anh – 20225564**  **Vu Duc Thang – 20225553**  **Dao Minh Quang – 20225552**  **Nguyen Sy Quan - 20225585** |
| **School:** | School of Information and Communications Technology |
| **HANOI, 6/2024** | |

# INTRODUCTION

## Background

The stock market is a dynamic and complex system, heavily influenced by various economic, political, and social factors. Understanding stock price movements and predicting future trends is crucial for investors, financial analysts, and policymakers. The S&P 500 index, which includes 500 of the largest publicly traded companies in the U.S., serves as a key indicator of the overall health of the U.S. stock market and economy.

## Problem Formulation

The objective of this report is to analyze the stock price movements of Apple Inc. (AAPL) using historical data. By leveraging statistical models and machine learning techniques, we aim to identify patterns, understand the factors driving stock prices, and develop predictive models to forecast future stock prices. This analysis can provide valuable insights for investors and help in making informed investment decisions.

## Aims

1. To explore and preprocess the historical stock price data of Apple Inc.
2. To check the stationarity of the time series data and transform it if necessary.
3. To decompose the time series to understand its underlying components.
4. To build and evaluate predictive models for forecasting future stock prices.
5. To interpret the results and provide actionable insights for investors.

# DATASET

## Dataset Description

**Description**

We are utilizing the [S&P 500 stock data](https://www.kaggle.com/datasets/camnugent/sandp500) dataset from Kaggle for our analysis. This dataset, last updated in 2018, is a comprehensive collection of historical stock prices for all companies currently listed on the S&P 500 index. The data spans a period of 5 years, providing a rich source of information for our study.

The dataset is well-structured and can be divided to analyze each individual company separately. This is facilitated by the **individual\_stocks\_5yr** folder, which contains data files for individual stocks. Each file is labelled by the respective company's stock ticker name, making it easy to locate and analyze data for a specific company.

Each file in the dataset contains the following columns:

* **Date:** This column records the date of the trading day in the format: yy-mm-dd. It allows us to track the stock's performance over time.
* **Open:** This column records the price of the stock at market open. This data is from the NYSE, so all prices are in USD. It provides a starting point for the day's trading.
* **High:** This column records the highest price the stock reached during the trading day. It gives us an idea of the stock's potential for the day.
* **Low:** This column records the lowest price the stock reached during the trading day. It provides insight into the stock's risk for the day.
* **Close :** Closing price of the stock at the end of the trading day. Tt is a commonly used reference point for investors to assess the performance of a particular stock over time.
* **Volume:** This column records the number of shares traded during the trading day. It gives us an idea of the stock's liquidity and popularity.
* **Name:** This column records the ticker name of the stock. It allows us to identify the stock.

To provide a more focused context for this project, our team has chosen to analyze a specific company - Apple, one of the world's largest technology companies. Its stock is listed on the index as AAPL. The dataset we're using for this analysis is named **AAPL\_data.csv**. Following is an overview visualization of the dataset:

A table with numbers and text

Description automatically generated

A noteworthy characteristic of this dataset is the data column. Notably, the date **08-02-2013** falls on a Friday, and the subsequent date in the dataset, **11-02-2013**, is a Monday. This suggests that the data is recorded only on weekdays.

**Key Features**

Some statistic about this dataset:

A screenshot of a computer code

Description automatically generated

A black and white text

Description automatically generated with medium confidence

So there is no null value in this dataset

A table with numbers and letters

Description automatically generated

## Preprocessing

### Stationary check

In oder to use AR, MA, ARMA, Auto-ARIMA models, we first had to make sure that out time seri is stationary (theory about stationary more detail in Part 3 Theory).

First let have a look at our time seri:

A line graph with numbers and text

Description automatically generated

A graph of a stock market

Description automatically generated

Chart 1 Candlestick chart of stocks

As seen from the graph, the time series exhibits trends and possibly seasonal patterns, which indicate non-stationarity.

To confirm this, we can perform statistical tests such as the Augmented Dickey-Fuller (ADF) test.

### Transform to Stationary

To transform the time series to a stationary one, we can use techniques such as differencing, logging, or detrending.

#### Differencing

Differencing is a common method used to remove trends and stabilize the mean of a time series. The first difference of the series can be calculated as:

Where is the original value and is differenced value.

#### Logging

Taking the natural logarithm of the series can help stabilize the variance:

### Decomposition of the Series

Decomposition involves breaking down the time series into its fundamental components: trend, seasonality, and residuals. This helps in understanding the underlying patterns and in building better predictive models.

Time series are a combination of (mainly) three components: Trend, Seasonality and Residuals/Remainder. Let's break each of these down.

**Trend:** This is the overall motion of the series. It may be consistently increasing overtime, decreasing overtime or a combination of both.

**Seasonality:** Any regular seasonal pattern in the series. For example, ice cream sales are regularly higher in summer than winter.

**Residual/Remainder:** This is the bit that is left over after we take into account the trend and seasonality. It can also be thought of as just statistical noise.

# MODELS

## Theory

### Autoregressive (AR) Model

The AR model specifies that the output variable depends linearly on its own previous values:

where is the value at time t, c is a constant, are the coefficients, and ​ is the white noise error term.

### Moving Average (MA) Model

The MA model specifies that the output variable depends linearly on the current and various past error terms:

where are the coefficients.

### ARMA Model

The ARMA model combines AR and MA models:

### ARIMA Model

The ARIMA model (AutoRegressive Integrated Moving Average) is a generalization of the ARMA model that includes differencing to make the time series stationary:

where is the differenced series. The parameters are:

* **p**: number of lag observations in the model
* **d**: number of times that the raw observations are differenced
* **q**: size of the moving average window

### Prophet

Prophet is a forecasting tool developed by Facebook that is particularly effective for time series with daily observations that display patterns on different time scales. It decomposes the time series into trend, seasonality, and holiday effects:

where:

* models the trend,
* models the seasonality,
* models the holidays,
* ​ is the error term.

## Implementation

### Hyperparameter

Using PACF and ACF plots, we determine the appropriate values for p and q parameters for the AR and MA components.

A graph with blue dots

Description automatically generated

**Auto-correlation interpretation** - Slow decline in auto-correlation indicates time-series not stationary, we can prove the stationarity of time-series by Dicky-fuller test

A graph with blue dots

Description automatically generated

**Partical auto-correlation interpretation** - For both stocks PACF suggest that time-series dependence can be captured only 1 lag owing to its significance than other laged time-series

One lags with significant greater than 0 should be added to the model. PACF helps to decide PACF in AR model while ACF is for MA model. Between AR and MA, we should choose simpler model (model using fewer lags). Unless the complex model provides significant better prediction. To measure that 'significant' we use Log-likelihood ratio test (for models with different lags) or AIC, BIC (for models with same lags). If our model fir well there should be no trend we fail to account for, the residual for the model should resemble white noise (NO patterns we have missed when overtraining)

### Stationarity test

**Dicky-Fuller Test:**

Hypothesis to prove dicky-fuller tests

H0 - Beta = 1 (the time-series is non-stationary)

HA - Beta < 1 (the time-series is stationary)

A white background with black text

Description automatically generated

The stocks time-series is not stationary as p-values are much greater than 0.05, hence we cannot reject the null-hypothesis.

**Finding degree of differencing**

The degree of differencing is 1 for APPLE by ADF test

## Train

### Auto-ARIMA

In Auto ARIMA, the model itself will generate the optimal p, d, and q values which would be suitable for the data set to provide better forecasting.

**Auto-Regressive (p)** -> Number of autoregressive terms.

**Integrated (d)** -> Number of nonseasonal differences needed for stationarity.

**Moving Average (q)** -> Number of lagged forecast errors in the prediction equation.

In the Auto ARIMA model, note that small p,d,q values represent non-seasonal components, and capital P, D, Q represent seasonal components. It works similarly like hyper tuning techniques to find the optimal value of p, d, and q with different combinations and the final values would be determined with the lower AIC, BIC parameters taking into consideration

Here, we are trying with the p, d, q values ranging from 0 to 3 to get better optimal values from the model.

A computer screen shot of a code

Description automatically generatedA screenshot of a computer program

Description automatically generated

Below is the summary of the model.

A screenshot of a computer

Description automatically generated

**Model diagnostics interpretation:**

**A screenshot of a graph

Description automatically generated**

**1) Standardized residual:** It is an error term of price forecating and actual price of stocks

**2) Histogram plus estimated density:** Histogram reresents normal distribution of errors, KDE plots and N(0,1) is notation of indicate mean is ZERO and variance of the distribution is ONE.

**3) Normal Q-Q:** Normal Q-Q plot implies normality of distribution as sample quantities mostly inline with theoretical quanitites. any deviation in such alignment would indicate distribution is skewed, or in layman terms error is either positive or negative side.

**4) Correlogram:** It simply indicates partial auto-correlation of time-series and shows which laged time-series is significant in forecasting actual time-series

## Prophet

Prophet is a procedure for forecasting time series data 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. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

**Out-of-Sample Forecast**

That is, we can make a forecast on data used as input to train the model. Ideally, the model has seen the data before and would make a perfect prediction.

Nevertheless, this is not the case as the model tries to generalize across all cases in the data.

This is called an out-of-sample forecast. We can achieve this in the same way as an in-sample forecast and simply specify a different forecast period.

A forecast is made by calling the **predict()** function and passing a DataFrame that contains one column named ‘**ds**‘ and rows with date-times for all the intervals to be predicted.



This DataFrame can then be provided to the **predict()** function to calculate a forecast.

The result of the **predict()** function is a DataFrame that contains many columns. Perhaps the most important columns are the forecast date time (‘**ds**‘), the forecasted value (‘**yhat**‘), and the lower and upper bounds on the predicted value (‘**yhat\_lower**‘ and ‘**yhat\_upper**‘) that provide uncertainty of the forecast.

For example, we can print the first few predictions

A number with black text

Description automatically generated with medium confidence

**Forecasted results**

**A graph with lines and numbers

Description automatically generated**

We can see the training data are represented as black dots and the forecast is a blue line with upper and lower bounds in a blue shaded area.

**A graph of a line

Description automatically generated with medium confidence**

## Envaluation

### ARIMA Model Results

After fitting the ARIMA model and generating forecasts, we compare the forecasted values with the actual test data and calculate the evaluation metrics.



**CONCLUSION**