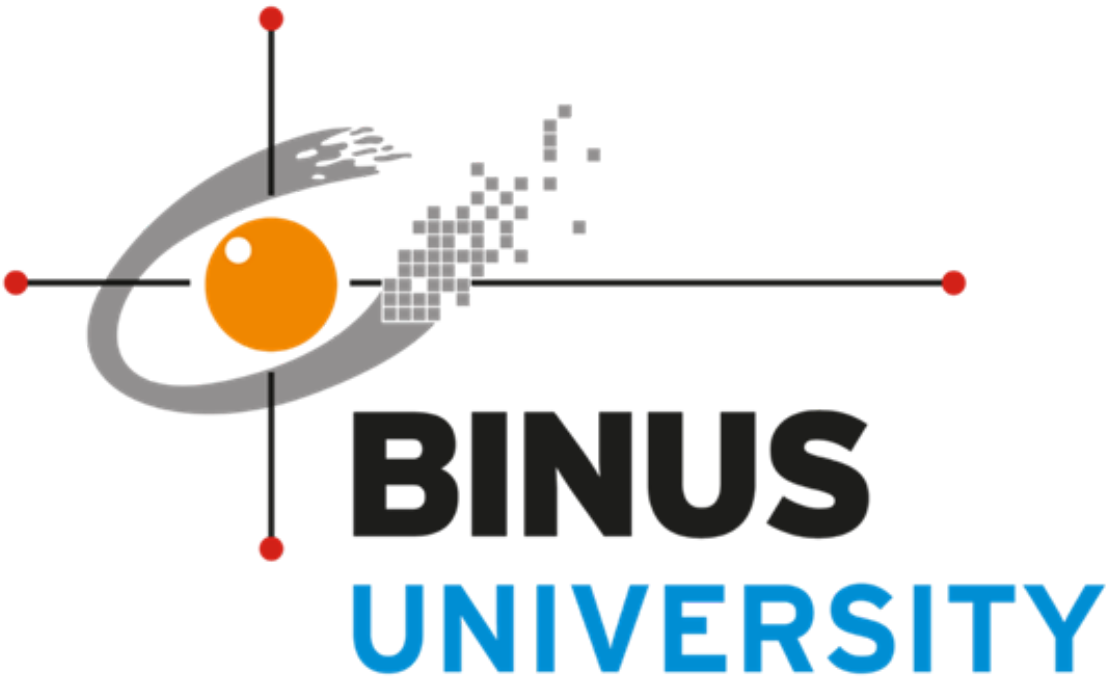
Stock Prediction Python Final Project



Lecturer:
D4017 JUDE JOSEPH LAMUG MARTINEZ, MCS

Arranged by:
2602158626 - Adyatama Mahabarata

14th January 2023
Algorithm and Programming
Computer Science Faculty
Binus International University 2022/2026

Table of Contents

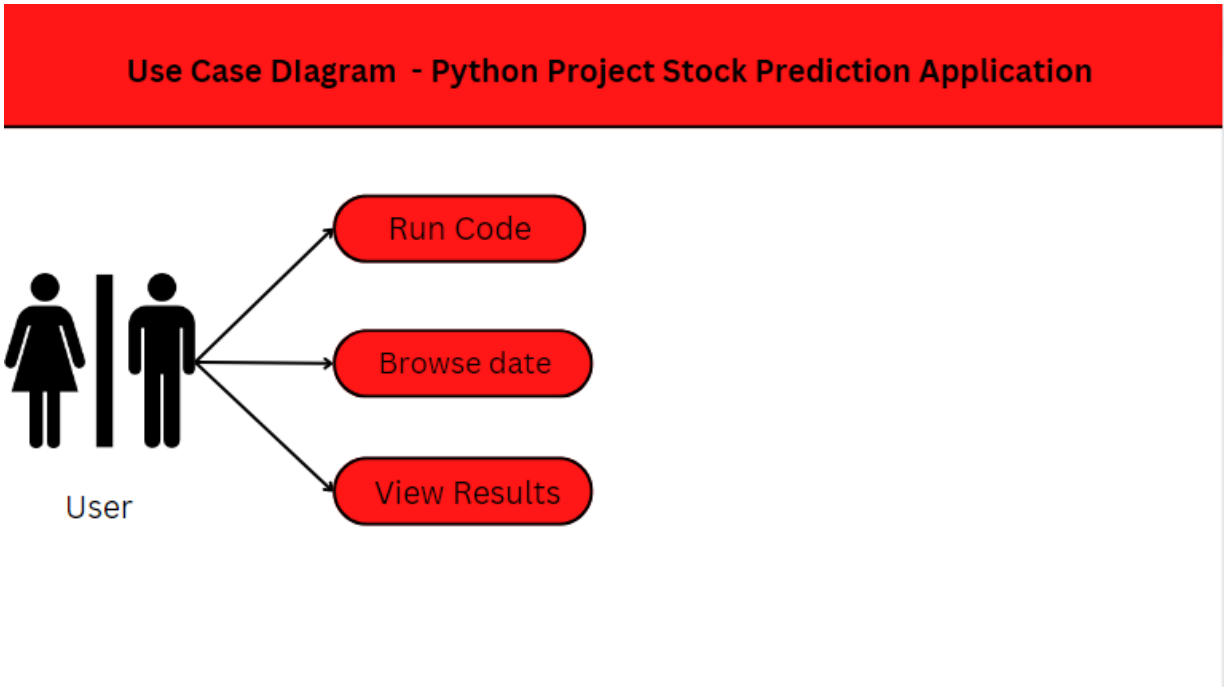
Chapter 1 - Brief Description

The provided code is a Python script that shows how to use several libraries and methods to build a basic stock price forecasting web application. The script builds an interactive web application using the streamlit library that enables the user to choose a stock, specify the number of years for the prediction, and examine the raw data, projected data, and graphs of the data. To get historical stock data, the use of yfinance library comes into play. To train a time series forecasting model, use the prophet library. To visualize the data, use the plotly library. To set the start and end dates for the data, and to obtain the current date, use the datetime module.

The script begins by importing the required libraries, after which it specifies the start and end dates for the data and generates a list of stocks from which the user may select. The user then can select the stock from the list, after that it uses the slider function from streamlit to take input for the number of years for the forecast. The script then employs the cache function of Streamlit to enhance script speed by avoiding repeatedly loading the data during each script run. The script loads the historical data using the yfinance library and shows the raw data using stock.write(). After some time the script will plot the raw data using the plotly library where it uses the prophet library to make the prediction for the future stock prices. Following that the script plots the forecasted data using plotly and displays the forecasted data, forecasted plot and forecasted components to the user using the streamlib.

Chapter 2 – Use-Case Diagram

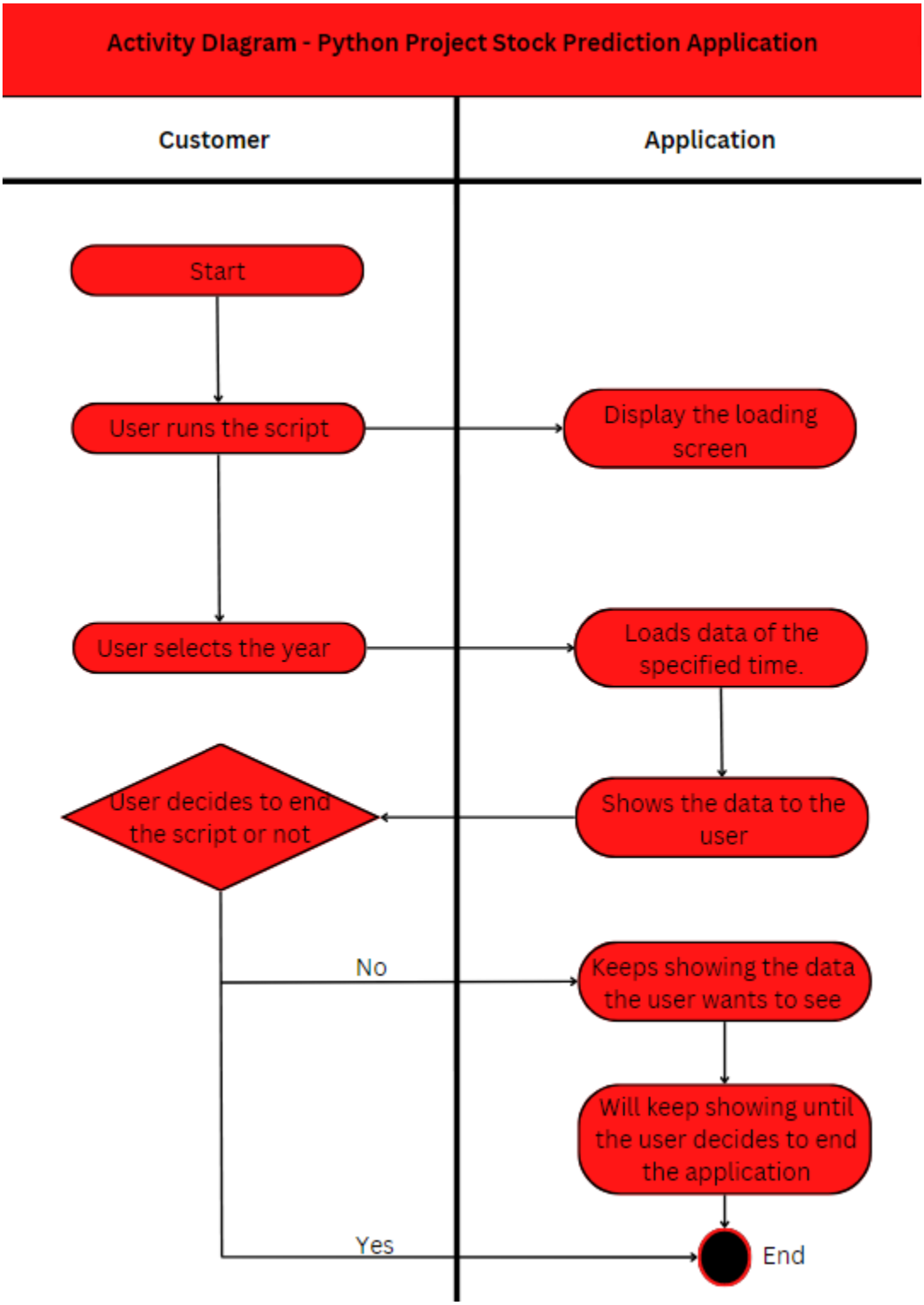
From the code this is the Use Case Diagram:



Use Case Diagram Table

Chapter 3 – Activity Diagram

From the code this is the Activity Diagram:



Activity Diagram Table

Chapter 4 – Class Diagram

From the code this is the Class Diagram:

Class Diagram - Python Project Stock Prediction using Prophet
+ stocks : list + start_date : str + end_date : str + selected_stock : str + n_years : int + period : int
+ __init__(stocks : list, start_date : str, end_date : str) + select_stock(selected_stock : str) + set_period(n_years : int)

Class Diagram Table

Chapter 5 – Modules

From the stock prediction code it makes use of several modules to achieve its task that is the following:

1. **streamlit**: This is a Python library for building web-based applications for machine learning and data science.

2. **datetime**: This is a built-in Python module that provides classes for manipulating dates and times.

3. **yfinance**: This is a third-party Python library for accessing financial data from Yahoo Finance, including stock market data, financial news, and historical data.

4. **prophet**: This is a third-party Python library for time series forecasting. It is built on top of Pystan.

5. **prophet.plot**: This is a submodule of the prophet library that provides functions for creating interactive plots of the forecasted values, uncertainty intervals, and components of the time series model.

6. **plotly**: This is a third-party Python library for creating interactive plots and visualizations.

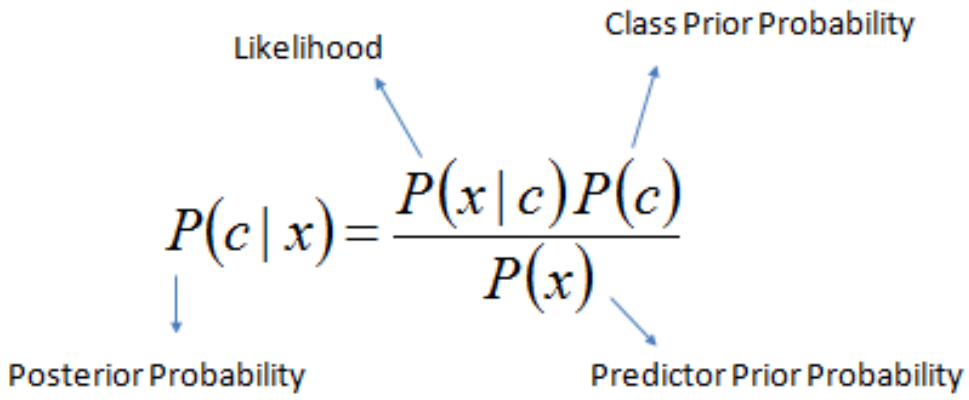
7. **graph_objs**: This is a submodule of the Plotly library that provides functions for creating different types of plots and figures.

Overall, the code shows how several modules may be used to build a straightforward web application for stock price predictions. The prophet library is used to train a time series forecasting model, the plotly library is used to create plots of the data, the streamlit library is used to create an interactive web application, the yfinance library is used to download historical stock data, the datetime module is used to specify the start and end dates for the data, and the current date is obtained using the datetime module. These modules are imported and utilized in the code in a clear and organized manner, and they are a must for every stock price forecasting program.

Chapter 6 – Essential Algorithms

For the Python code, it uses one important technique for it to work, which is the Bayesian technique. The Bayesian theorem/technique is a mechanism of updating an initial probability estimate when new evidence is gathered, and the Bayesian methodology is a statistical approach that is based on it. Using this method, parameters and hypotheses are given probabilities, and the likelihood is updated as new data is gathered. It enables the model to anticipate outcomes more precisely and react to changes in the data's underlying patterns. In contrast to the conventional frequentist approach, this method uses probability distributions to describe the uncertainty in parameters and assumptions. Machine learning, computer vision, natural language processing, and many more domains all employ Bayesian approaches extensively. The Bayesian technique works by using probability distributions to represent uncertainty in parameters and hypotheses. It starts by specifying a prior probability distribution for each parameter or hypothesis, based on prior knowledge or beliefs. Then, as new data is collected, the prior probability distributions are updated using Bayes' theorem to create posterior probability distributions, which are then used to make inferences and predictions. This process can be repeated as new data is collected, allowing the model to adapt to changes in the underlying patterns in the data and make more accurate predictions.

The Bayesian technique formula is based on Bayes' theorem, which is a fundamental concept in Bayesian statistics. The theorem states that the probability of a hypothesis (c) given some data (x) is proportional to the probability of the data given the hypothesis (x|c) multiplied by the prior probability of the hypothesis (c):



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

where:

P(c|X) is the posterior probability, which represents the probability of the hypothesis given the data.
P(x|c) is the likelihood, which represents the probability of the data given the hypothesis.
P(c) is the prior probability, which represents the probability of the hypothesis before the data is collected.
P(x) is the marginal likelihood, which represents the probability of the data.
The posterior probability is used to make inferences and predictions about the hypothesis.

The formula is read as "the probability of the hypothesis given the data is proportional to the probability of the data given the hypothesis multiplied by the prior probability of the hypothesis".

This formula is the foundation of the Bayesian technique and it allows us to update the prior probability as new data is collected and make predictions accordingly.

From the stock prediction code it makes use of several algorithms to achieve its task that is:

1. **Time Series Prediction**: The script uses the Prophet class from the prophet library to fit a time series model to the historical data and make predictions about future stock prices.

2. **Caching**: The script uses the stock.cache decorator to cache the loaded data, to avoid loading the data again and again in each run of the script.

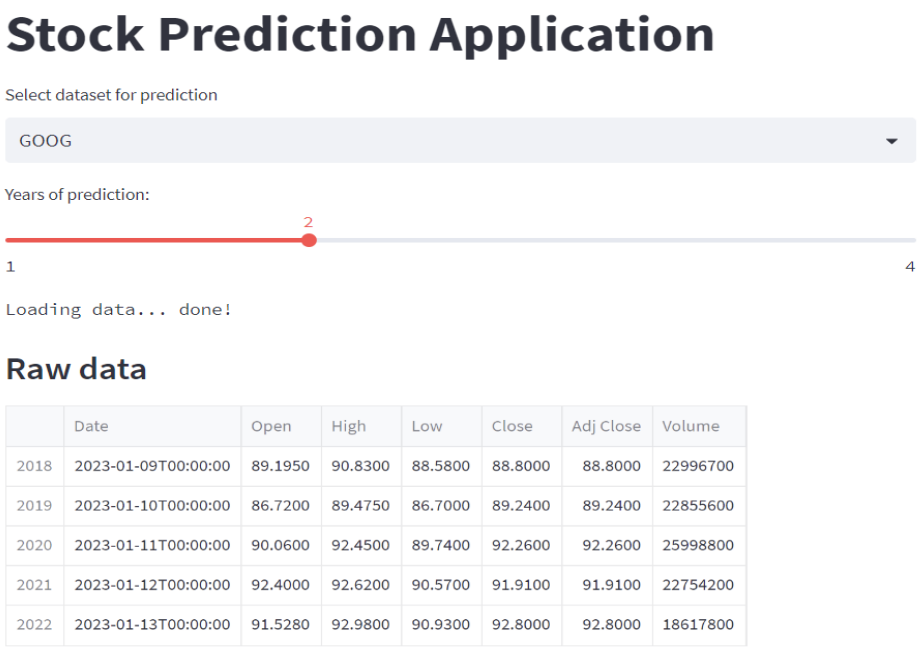
3. **Data visualization**: The script uses the plotly library and streamlib to create interactive plots of the historical and forecasted data, which allows users to easily zoom, pan, and hover over the plots for more details.

4. **Data loading**: The script uses the yfinance library to download the historical data of the selected stock and the selected time range.

5. **Data preparation**: The script prepares the data in a format suitable for the Prophet library by renaming columns and selecting specific columns of the loaded data.

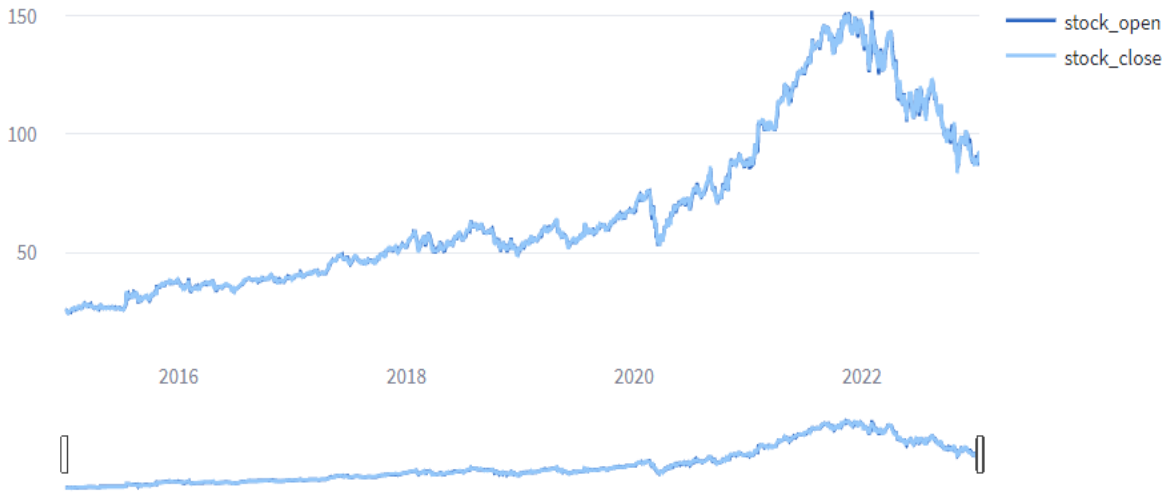
In summary, the code shows how to use multiple approaches to create a basic stock price predicting web application. A time series forecasting model is trained using the prophet library, historical stock data is downloaded using the yfinance library, and data visualization is done using the plotly library. These methods are implemented in the code in a clear and organized manner.

Chapter 7 – Screenshots Of The Application



	Date	Open	High	Low	Close	Adj Close	Volume
2018	2023-01-09T00:00:00	89.1950	90.8300	88.5800	88.8000	88.8000	22996700
2019	2023-01-10T00:00:00	86.7200	89.4750	86.7000	89.2400	89.2400	22855600
2020	2023-01-11T00:00:00	90.0600	92.4500	89.7400	92.2600	92.2600	25998800
2021	2023-01-12T00:00:00	92.4000	92.6200	90.5700	91.9100	91.9100	22754200
2022	2023-01-13T00:00:00	91.5280	92.9800	90.9300	92.8000	92.8000	18617800

Time Series data with Rangeslider

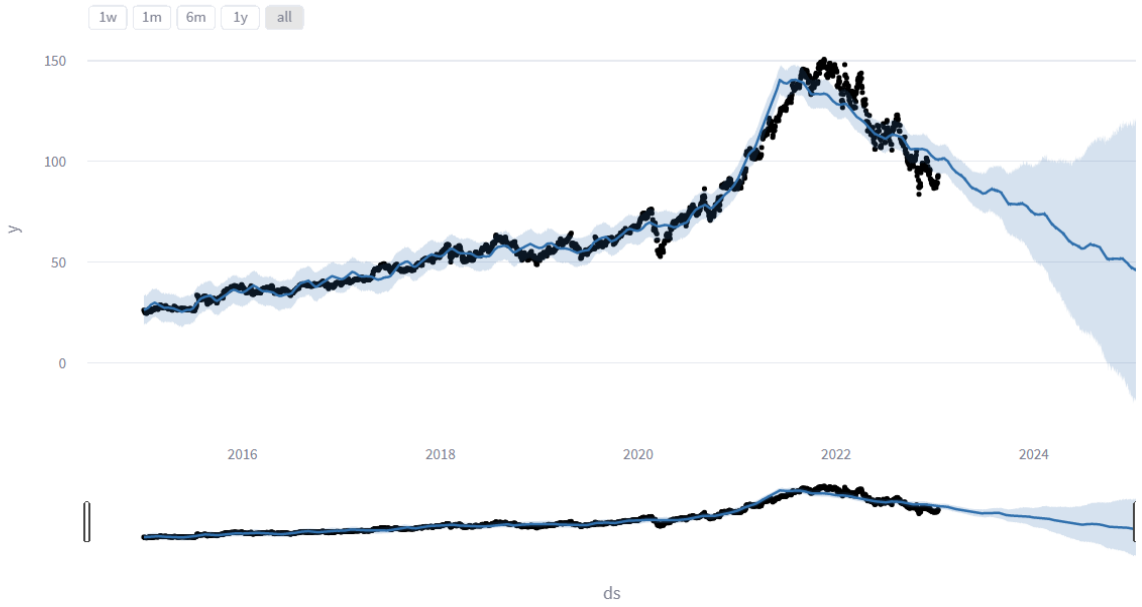


Prediction data

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms
2748	2025-01-08T00:00:00	45.4235	-18.9753	118.9516	-19.9974	117.4298	0.7130
2749	2025-01-09T00:00:00	45.3487	-17.7711	119.3457	-20.1243	117.5173	0.7320
2750	2025-01-10T00:00:00	45.2739	-17.8563	119.3174	-20.2836	117.6490	0.7088
2751	2025-01-11T00:00:00	45.1991	-18.9667	120.1272	-20.4310	117.7808	1.0411
2752	2025-01-12T00:00:00	45.1242	-19.0343	121.6108	-20.6152	117.9125	1.1009

Prediction plot for 2 years

Prediction plot for 2 years



Chapter 8 – Lesson Learned/Reflection

From this code, I can learn the usage of various libraries and functions to create a simple web application for stock price forecasting. It uses the yfinance library to download historical stock data, the prophet library to train a time series forecasting model and make predictions for future stock prices, the plotly library to create plots of the raw data and the forecasted data, and the streamlit library to create a simple web application that allows the user to select a stock, specify the number of years for the forecast, and view the raw data, forecasted data, and plots of the data. The cache feature of streamlit is used to improve the performance of the script by avoiding loading the data again and again in each run of the script. The script also uses the rename() method of pandas dataframe to rename the columns of the dataframe, the make_future_dataframe() method of Prophet to create a new dataframe containing the historical dates and the forecasted dates, and the plot_components() method of Prophet to create a plot of the forecasted data. The code demonstrates the usage of different libraries and functions for a stock price forecasting application. The script is easy to understand and follow, even for someone like me who is not familiar with the codes that make the script to predict the future stock market.