1. INTRODUCTION

1.1 Overview

The Stock Price Prediction Web App leverages machine learning techniques, specifically the Facebook Prophet and Linear Regression models, to forecast Amazon's stock prices. By analysing historical stock data, the project aims to provide insights into the comparative performance of these models in predicting future stock prices. The dataset includes features such as date, opening and closing prices, enabling a comprehensive exploration of time series trends. The project's objective is to not only develop accurate predictive models but also to create an interactive Streamlit web app for users to visualize and understand the forecasted trends. This initiative holds significance in the financial domain, offering potential benefits to investors, financial analysts, and decision-makers in the stock market. Through this project, the application of machine learning in financial forecasting is showcased, emphasizing its role in enhancing decision-making processes.

1.2 Problem Statement

The challenge in the Stock Price Prediction Web App project lies in developing and comparing machine learning models, specifically the Facebook Prophet and Linear Regression, to accurately forecast Amazon's stock prices. The project seeks to effectively utilize historical stock data, encompassing features such as date, opening and closing prices, to discern patterns and correlations that influence future stock prices. The key problem is to establish robust predictive algorithms capable of handling the dynamic nature of stock market data, providing reliable insights for investors and decision-makers. Successfully addressing this challenge will contribute to improved decision-making in financial forecasting, assisting stakeholders in navigating the complexities of the stock market with confidence and precision.

1.3 Objectives

- Implement and Compare Machine Learning Models

  Develop and implement the Facebook Prophet time series forecasting model and Linear Regression model for stock price prediction and train these models using historical Amazon stock data to capture different aspects of time series patterns and trends.

- Performance Evaluation

  Conduct a comprehensive evaluation of the performance of both the Facebook Prophet and Linear Regression models, utilize key evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R2) to assess the accuracy and reliability of each model.

- Comparative Analysis

  Conduct a comparative analysis of the strengths and weaknesses of the Facebook Prophet and Linear Regression models in the context of stock price prediction. Explore how each model captures and interprets the inherent patterns in the financial data.

1.4 Dataset Description

The dataset used is "amazon.csv," comprising of 6,155 entities spanning various dates.

It includes 7 attributes:

- Date: The dataset consists of 6155 entries representing the date of each stock data record. The datatype is 'object,' suggesting that it may need conversion to a datetime format for time-based analysis.
- Open, High, Low, Close: These numeric columns represent the opening, highest, lowest, and closing prices, respectively. These values are of type 'float64,' indicating decimal numbers.
- Adj Close: This column also contains decimal values and represents the adjusted closing price, which accounts for corporate actions such as dividends and stock splits.
- Volume: The 'Volume' attribute is of type 'int64' and denotes the trading volume, representing the total number of shares traded during a specific period.

The data is free from missing values, and the columns consist of float64 and int64 data types, indicating the presence of both continuous and integer variables.

2. SYSTEM REQUIREMENTS

2.1 Software & Hardware

- Software
  - Python (Streamlit, Prophet, scikit-learn, pandas, plotly, matplotlib)
  - Any Python IDE
- Hardware
  - Intel Core i5 or higher
  - At least 8GB of RAM
  - Storage capacity of at least 20-30 gigabytes (GB)

3. SYSTEM ARCHITECTURE & DESIGN

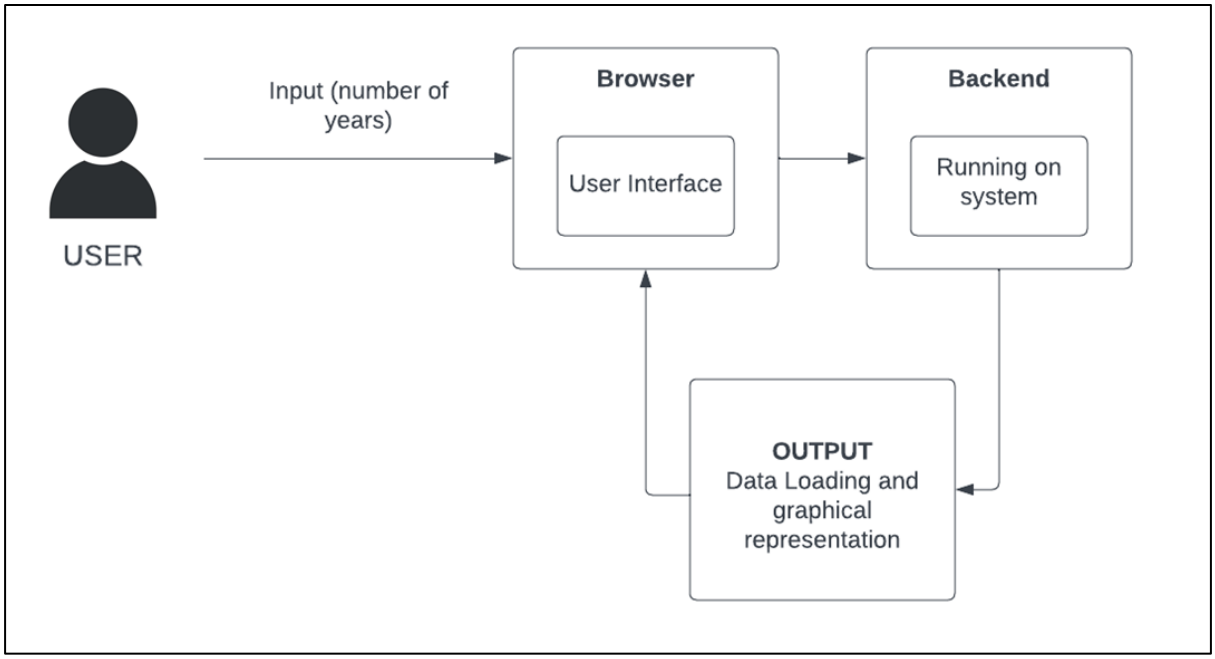

Fig 1 : System Architecture and Design Diagram

- User: Stakeholders engage with the system to forecast Amazon stock prices and analyse market fluctuations, seeking valuable insights for informed decision-making.
- User Interface (Browser): The browser serves as the graphical interface, providing a user-friendly and easily navigable platform for stakeholders to interact with and interpret stock market data.
- Backend System (Local Execution): The backend system, comprising machine learning models for stock price prediction, operates locally on the user's machine. This ensures efficient and secure execution of the predictive algorithms.
- Data Loading and Graphical Representation (Output): The system loads relevant data, facilitating seamless presentation on the frontend via a Streamlit web page. Graphical representations, generated using advanced machine learning Python packages, enhance the visualization of stock market trends for comprehensive analysis.

4. IMPLEMENTATION

4.1 Machine Learning Algorithm Selection

In the machine learning algorithm selection phase, two models were chosen for their distinct characteristics:

- Facebook Prophet

  The Facebook Prophet model was selected due to its suitability for time series forecasting tasks. Prophet is adept at handling daily observations with seasonality, making it well-suited for predicting stock prices that exhibit temporal patterns.

- Linear Regression

  Linear Regression was implemented as a baseline comparison to Prophet. This simple yet widely used algorithm provides a straightforward understanding of the relationship between the input features and the target variable. Its application in this project allows for a comparative analysis of predictive performance.

4.2 Machine Learning Model Building & Evaluation

Following algorithm selection, both the Prophet and Linear Regression models were built and evaluated using the Amazon stock dataset:

4.2.1 Training and fitting the model

```
# Predict forecast with Prophet.
df_train = data[['Date','Close']]
df_train = df_train.rename(columns={"Date": "ds", "Close": "y"})

m = Prophet()
m.fit(df_train)
future = m.make_future_dataframe(periods=period)
forecast = m.predict(future) # Prepare the data for linear regression
df_lr = df_train.copy()
df_lr['ds'] = pd.to_datetime(df_lr['ds'])  # Convert to datetime
df_lr['ds_numeric'] = (df_lr['ds'] - df_lr['ds'].min()).dt.days  # Feature: days since the start

# Prepare future data for linear regression
future_lr_dates = pd.date_range(start=df_lr['ds'].max() + pd.DateOffset(1), periods=period, freq='D')
future_lr_df = pd.DataFrame({'ds': future_lr_dates})
future_lr_values = future_lr_df['ds'].apply(lambda x: (x - df_lr['ds'].min()).days).values.reshape(-1, 1)

# Fit Linear Regression model
model = LinearRegression()
model.fit(df_lr[['ds_numeric']], df_lr['y'])

# Make predictions with Linear Regression for entire data and future data
future_lg = model.predict(df_lr[['ds_numeric']])  # Predict for entire data
predicted_future = model.predict(future_lr_values)  # Predict for future data
```

Fig 4.1: Training and Fitting the Models to the Data

4.2.2 Visualizations

```
# Plot raw data
def plot_raw_data():
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=data['Date'], y=data['Open'], name="stock_open"))
    fig.add_trace(go.Scatter(x=data['Date'], y=data['Close'], name="stock_close"))
    fig.layout.update(title_text='Time Series data with Rangeslider', xaxis_rangeslider_visible=True)
    st.plotly_chart(fig)

plot_raw_data()
```

Fig 4.2: Plotting Raw Data

```
# Show and plot forecast  (Prophet)
st.subheader('Forecast data with Prophet')
st.write(forecast.tail())

st.write(f'Forecast plot for {n_years} year/years')
fig1 = plot_plotly(m, forecast)
fig1.update_traces(line=dict(color='#0000FF'), marker=dict(color='#ADD8E6'))
st.plotly_chart(fig1)
```

Fig 4.3: Plotting Forecasted Data of Prophet Model

```
# Show and plot forecast from Linear Regression
def plot_linear_regression():
    fig = go.Figure()

    # Add scatter plot for actual data
    fig.add_trace(go.Scatter(x=df_lr['ds'], y=df_lr['y'], mode='markers', name='Actual Data'))

    # Add line plot for Linear Regression Line
    fig.add_trace(go.Scatter(x=df_lr['ds'], y=future_lg, mode='lines', name='Linear Regression Line', line=dict(color='red')))

    # Add dashed line plot for Future Predictions
    x_future = pd.date_range(start=df_lr['ds'].max() + pd.DateOffset(1), periods=len(predicted_future), freq='D')
    fig.add_trace(go.Scatter(x=x_future, y=predicted_future, mode='lines', line=dict(color='green', dash='dash'),
                             name='Future Predictions'))

    # Update layout with rangeslider and larger graphical area
    fig.update_layout(xaxis=dict(title='Days since start', rangeslider=dict(visible=True)),
                      yaxis=dict(title='Close Price'),
                      width=1000,  # Set the width of the chart
                      height=600)  # Set the height of the chart

    # Display the figure in Streamlit
    st.plotly_chart(fig)

plot_linear_regression()
```

Fig 4.4: Plotting Forecasted Data of Linear Regression Model

```
st.subheader("Forecast components")
fig2 = m.plot_components(forecast)
st.write(fig2)
```

Fig 4.5: Plotting Forecast Components (Trend, Weekly and Yearly)

4.2.3 Evaluation

```
# Evaluate Prophet model
y_true_prophet = df_train['y'].values
y_pred_prophet = forecast['yhat'].values[:-period]  # Exclude the forecasted future period

mae_prophet = mean_absolute_error(y_true_prophet, y_pred_prophet)
mse_prophet = mean_squared_error(y_true_prophet, y_pred_prophet)
r2_prophet = r2_score(y_true_prophet, y_pred_prophet)

# Evaluate Linear Regression model
y_true_lr = df_lr['y'].values
y_pred_lr = model.predict(df_lr[['ds_numeric']])

mae_lr = mean_absolute_error(y_true_lr, y_pred_lr)
mse_lr = mean_squared_error(y_true_lr, y_pred_lr)
r2_lr = r2_score(y_true_lr, y_pred_lr)

# Create a DataFrame for the metrics
metrics_data = {
    'Metric': ['Mean Absolute Error (MAE)', 'Mean Squared Error (MSE)', 'R-squared (R2)'],
    'Prophet Model': [mae_prophet, mse_prophet, r2_prophet],
    'Linear Regression Model': [mae_lr, mse_lr, r2_lr]
}

metrics_df = pd.DataFrame(metrics_data)

# Display the DataFrame
st.subheader('Regression Evaluation Metrics')
st.write(metrics_df)
```

Fig 4.6: Code Evaluation of Prophet and Linear Regression Models

4.3 Frontend and Backend Development

The collaborative effort in frontend and backend development resulted in the creation of a cohesive Streamlit web application, enhancing user interaction and providing a robust platform for stock price prediction analysis.

5. RESULTS

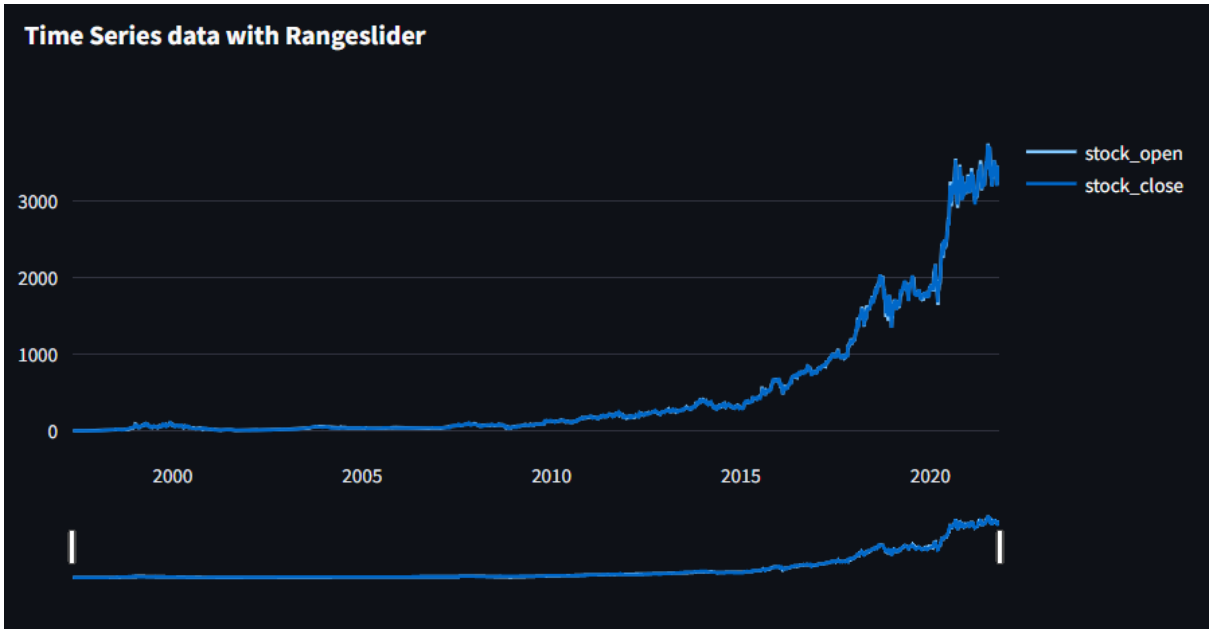

Fig 5.1: Plot of Raw Data

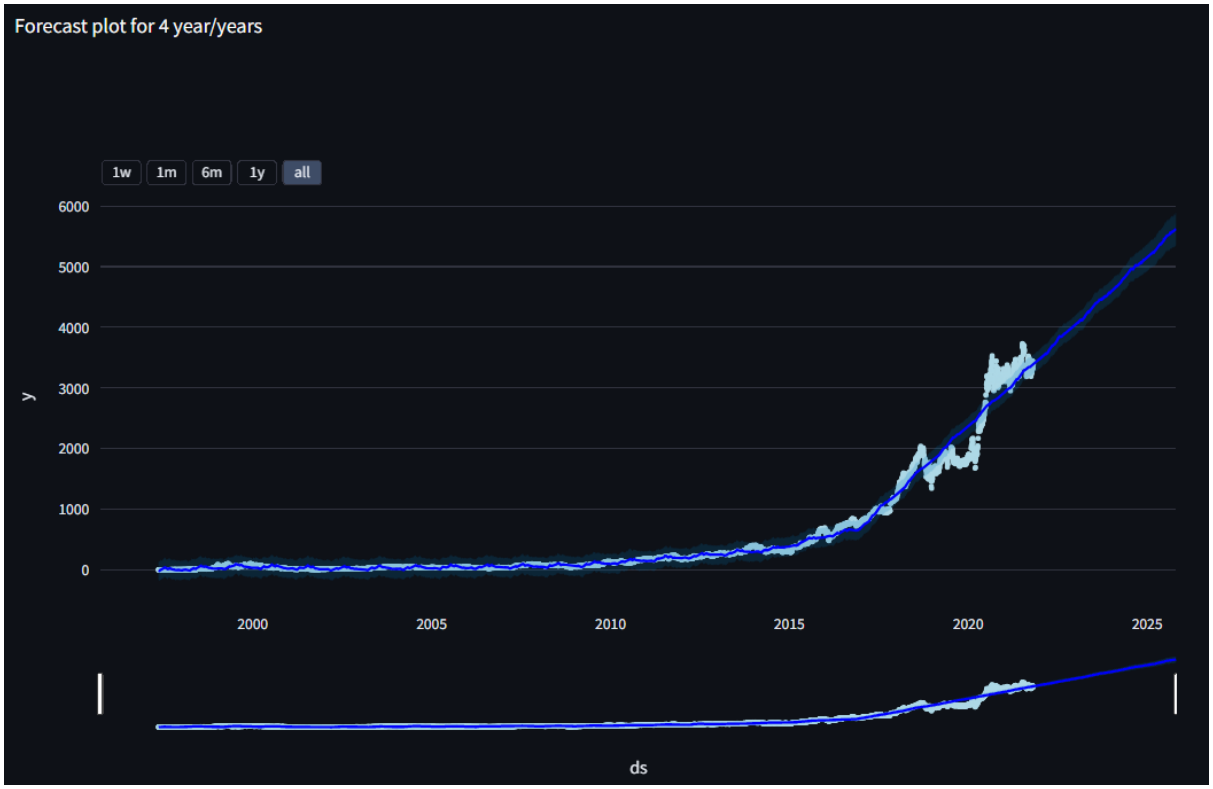

Fig 5.2: Plot of Forecasted Data of Prophet Model

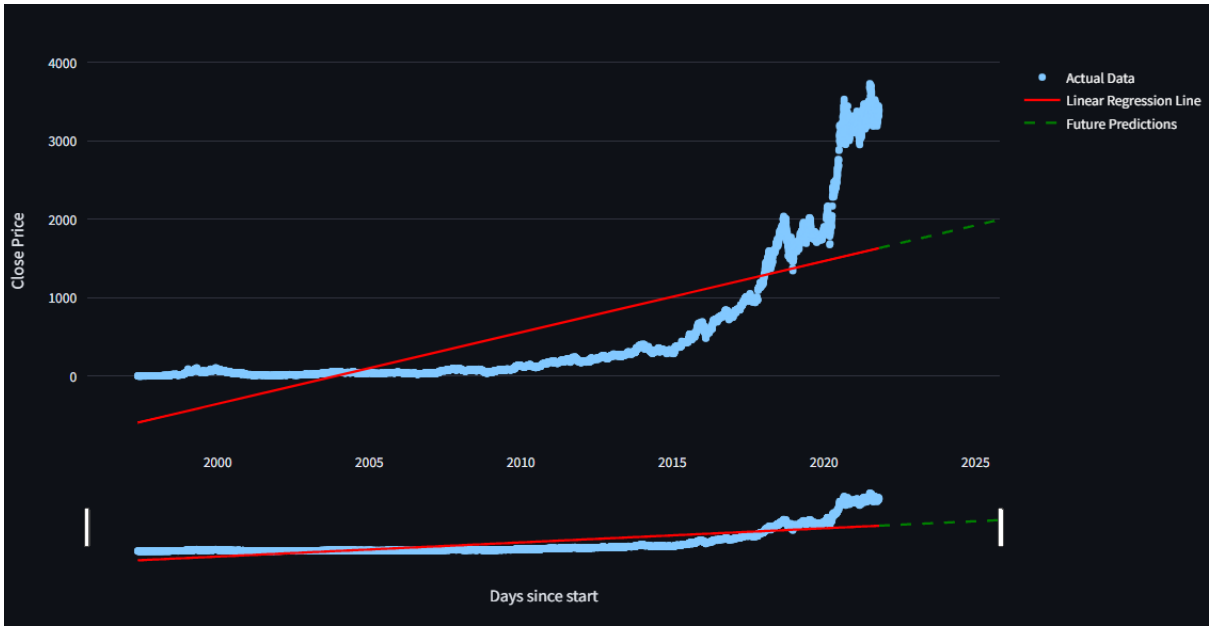

Fig 5.3: Plot of Forecasted data of Linear Regression Model

Regression Evaluation Metrics

	Metric	Prophet Model	Linear Regression Model
0	Mean Absolute Error (MAE)	63.5108	442.2195
1	Mean Squared Error (MSE)	15,522.0988	320,616.1706
2	R-squared (R2)	0.9788	0.5631

Fig 5.4: Regression Evaluation Metrics of Prophet and Linear Regression Models

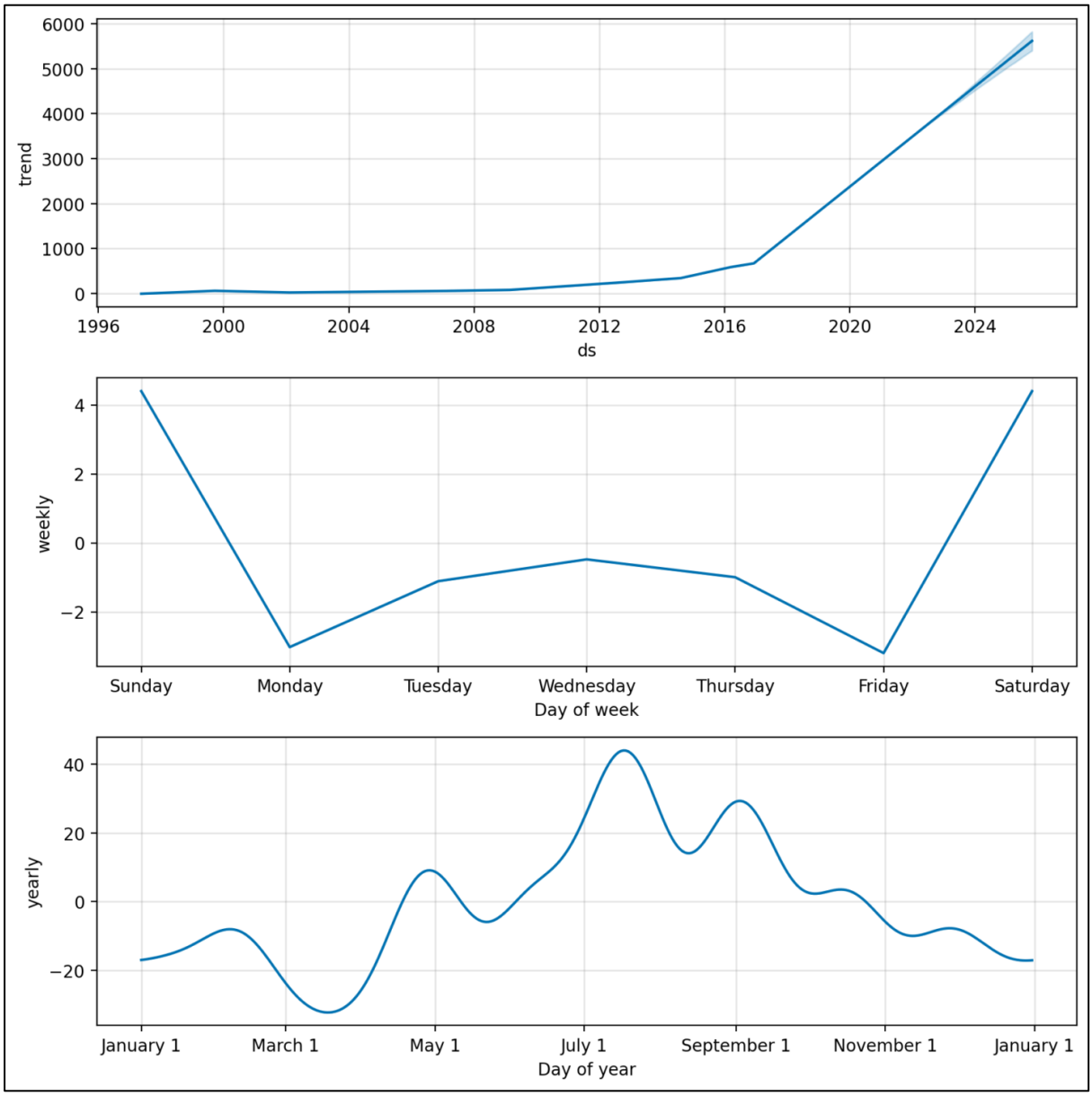

Fig 5.5: Plot of Forecast Components (Trend, Weekly and Yearly)

6. CONCLUSION & FUTURE ENHANCEMENTS

Based on the regression evaluation metrics for the Prophet and Linear Regression models, several conclusions can be drawn:

- Mean Absolute Error (MAE):
  - The Prophet model has a considerably lower MAE (63.51) compared to the Linear Regression model (442.22).
  - A lower MAE indicates that the Prophet model's predictions are, on average, closer to the actual values compared to the Linear Regression model.
- Mean Squared Error (MSE):
  - The MSE for the Prophet model is substantially lower (15522.10) than that of the Linear Regression model (320616.17).
  - The lower MSE for the Prophet model suggests that it exhibits better accuracy in predicting stock prices, with fewer large errors.
- R-squared (R2):
  - The R-squared value for the Prophet model (0.9788) is significantly higher than that of the Linear Regression model (0.5631).
  - A higher R-squared value indicates that a larger proportion of the variance in the stock prices is explained by the Prophet model, suggesting a better fit.

Overall Conclusions:

- The Prophet model consistently outperforms the Linear Regression model across all evaluation metrics.
- The Prophet model demonstrates a high level of accuracy and precision in predicting Amazon stock prices based on the provided dataset.
- The Linear Regression model, while providing predictions, shows limitations in capturing the underlying patterns in the stock price data compared to the more sophisticated Prophet model.

Future Enhancements: To further enhance the Stock Price Prediction Web App, future iterations may focus on refining the predictive models and expanding their capabilities. Fine-tuning algorithms using advanced techniques, such as deep learning or ensemble methods, could elevate the precision of the predictions. Additionally, integrating real-time data feeds and considering emerging market trends would contribute to the adaptability and predictive power of the models. Collaborations with financial experts and leveraging cutting-edge analytics to accommodate shifts in market dynamics could strengthen the model's robustness, offering nuanced insights into stock price fluctuations.