Data Analytics Project:Netflix Stock Price Prediction and Forecasting using Facebook Prophet

Aarushi Singh
Department of Data Science
University of Europe for Applied Sciences
Potsdam, Germany
aarushi.singh@ue-germany.de

Vidyareddy Ponnala
Department of Data Science
University of Europe for Applied Sciences
Potsdam, Germany
Vidya.ponnala@ue-germany.de

Pooja Shalini Suradi
Department of Data Science
University of Europe for Applied Sciences
Potsdam, Germany
pooja.suradi@ue-germany.de

Lauren Rapolu
Department of Business
University of Europe for Applied Sciences
Potsdam, Germany
lauren.rapolu@ue-germany.de

Abstract :

The goal of this research is to forecast Netflix Inc. stock values using the Facebook Prophet time series model. The dataset, collected from Kaggle, contains daily stock data with fields such as Open, High, Low, Close, Volume, and date. For this investigation, the goal variable was the "Close" price. Prophet, developed by Meta, is known for its capacity to deal with complex trends and seasonal patterns, making it an appropriate choice for this purpose. The goal was to develop a forecasting model that was not only accurate but also easy to understand, providing insights that could be useful to investors and analysts. The study continues by evaluating the model's performance, limits, and recommendations for enhancements.

Keywords— Netfilx

I. INTRODUCTION

The stock performance of Netflix (NFLX), one of the most significant streaming services globally, has represented the swift shifts in media consumption and international market dynamics. Since it aids in the development of data-driven investment strategies, stock price prediction is a useful exercise for both investors and data scientists. With non-linear data, Facebook Prophet provides a more adaptable and user-friendly method than previous models. This paper details how the forecast was created and assessed, as well as how previous Netflix stock data was processed and modelled. Making the output visually comprehensible and applicable to actual decision-making was a crucial step in the process.

II. EASE OF USE

Dataset Description

The dataset used comprises historical Netflix stock data obtained from Yahoo Finance, ranging from 23 May 2002 to 30 May 2002. It includes variables such as Open, High, Low, Close, Volume, and Adjusted Close prices. The analysis particularly focuses on the 'Close' price as the target variable for prediction.

Date: This is pretty straightforward – it tells you when the data was recorded (day, month, and year).

* Open: This is the price of the stock at the very beginning of the trading day.

* High: This is the highest price the stock reached during that trading day.

* Low: This is the lowest price the stock dipped to during the trading day.

* Close: This is the price of the stock when the trading day ended. It's a key number because it's often used to calculate gains and losses.

* Volume: This tells you how many shares of the stock were traded during that day. High volume can mean a lot of interest in the stock.

III. Tools Used

*Google Colab: Cloud-based notebook for running code interactively

*Pandas, NumPy: Data manipulation

*Matplotlib: Visualizations

*Facebook Prophet: Core forecasting model

IV. Data Cleaning

We see at the beginning the most important information. With this information we proceed to start the cleaning.

Date	Predicted Price	Lower Bound	Upper Bound
2025-07-08	**752.42**	637.55	874.25
2025-07-09	**751.03**	632.58	875.91
2025-07-10	**749.89**	634.91	863.57
2025-07-11	**748.08**	629.54	867.31
2025-07-12	**749.07**	632.12	870.43

V. Exploratory Data Analysis

*Visualized trends in historical stock prices.

*Generated descriptive statistics (mean, median, standard deviation, etc.).

* Created a bar chart summarizing statistical insights of the dataset.

Step1-

Netflix Dataset Upload

```
from google.colab import files
uploaded = files.upload()
import pandas as pd

df = pd.read_csv('NFLX.csv')  # File name NFLX
df.head()
```

	Date	Close	High	Low	Open	Volume
0	2002-05-23	1.196429	1.242857	1.145714	1.156429	104790000
1	2002-05-24	1.210000	1.225000	1.197143	1.214286	11104800
2	2002-05-28	1.157143	1.232143	1.157143	1.213571	6609400
3	2002-05-29	1.103571	1.164286	1.085714	1.164286	6757800
4	2002-05-30	1.071429	1.107857	1.071429	1.107857	10154200

Step2-

Brief Walk through of the data set

```
print(df.columns)
```

```
Index(['Date', 'Close', 'High', 'Low', 'Open', 'Volume'], dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5817 entries, 0 to 5816
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Date    5817 non-null   object 
 1   Close   5817 non-null   float64
 2   High    5817 non-null   float64
 3   Low     5817 non-null   float64
 4   Open    5817 non-null   float64
 5   Volume  5817 non-null   int64  
dtypes: float64(4), int64(1), object(1)
memory usage: 272.8+ KB
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objs as go
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from statsmodels.tsa.arima.model import ARIMA
from prophet import Prophet  # Optional
import warnings
warnings.filterwarnings("ignore")
```

Step3-

Descriptive statistics

```
import pandas as pd
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv('NFLX.csv')

# Calculate descriptive statistics
summary = df[['Open', 'High', 'Low', 'Close']].describe()
selected_stats = summary.loc[['mean', 'std', 'min', 'max']]
selected_stats.index = ['Mean', 'Standard Deviation', 'Minimum', 'Maximum']

# Transpose for better plotting
plot_data = selected_stats.transpose()

# Set custom colors for visibility
colors = ['#66c2a5', '#fc8d62', '#8da0cb', '#e78ac3']

# Plot the bar chart
ax = plot_data.plot(kind='bar', figsize=(12, 6), edgecolor='black', color=colors)
```

```
# Titles and labels
plt.title('Descriptive Statistics of Netflix Stock Prices', fontsize=16)
plt.ylabel('Price in USD ($)', fontsize=12)
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.legend(title='Statistic', fontsize=10)
plt.tight_layout()

# Add vertical value labels above each bar
for container in ax.containers:
    for bar in container:
        height = bar.get_height()
        ax.text(
            bar.get_x() + bar.get_width() / 2,  # X position at center of bar
            height + 2,                         # Y position slightly above bar
            f'{height:.2f}',                    # Label value
            ha='center',                        # Horizontal alignment
            va='bottom',                        # Vertical alignment
            fontsize=8,
            rotation=360                         # Make text vertical
        )

# Show the plot
plt.show()
```

Step4-

Visualize Historical Stock Prices (Time Series Plot)

```
# Clean column names
df.columns = df.columns.str.strip()

# Convert 'Date' to datetime
df['Date'] = pd.to_datetime(df['Date'])

# Sort by Date
df = df.sort_values('Date')

# Plot the Closing price
import matplotlib.pyplot as plt

plt.figure(figsize=(14, 6))
plt.plot(df['Date'], df['Close'], color='blue')
plt.title('Netflix Closing Stock Price Over Time', fontsize=16)
plt.xlabel('Date')
plt.ylabel('Closing Price ($)')
plt.grid(True)
plt.tight_layout()
plt.show()
```

STEP5-

ONLY USING DATE & CLOSE COLUMNS

```
# Just keep the 'Date' and 'Close' columns
data = df[['Date', 'Close']]

# Set the 'Date' column as index
data.set_index('Date', inplace=True)
```

Step6-

Install,Import and preparing data ready for Prophet

```
!pip install prophet
from prophet import Prophet
# Rename the columns
df_prophet = df[['Date', 'Close']].rename(columns={'Date': 'ds', 'Close': 'y'})
```

```
Requirement already satisfied: prophet in /usr/local/lib/python3.11/dist-packages (1.1.7)
Requirement already satisfied: cmdstanpy>=1.0.4 in /usr/local/lib/python3.11/dist-packages (from prophet) (1.2.5)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.11/dist-packages (from prophet) (2.0.2)
Requirement already satisfied: matplotlib>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from prophet) (3.10.0)
Requirement already satisfied: pandas>=1.0.4 in /usr/local/lib/python3.11/dist-packages (from prophet) (2.2.2)
Requirement already satisfied: holidays<1,>=0.25 in /usr/local/lib/python3.11/dist-packages (from prophet) (0.75)
Requirement already satisfied: tqdm>=4.36.1 in /usr/local/lib/python3.11/dist-packages (from prophet) (4.67.1)
Requirement already satisfied: importlib_resources in /usr/local/lib/python3.11/dist-packages (from prophet) (6.5.2)
Requirement already satisfied: stanio<2.0.0,>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from cmdstanpy>=1.0.4->prophet) (0.5.1)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.11/dist-packages (from holidays<1,>=0.25->prophet) (2.9.0.post0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=2.0.0->prophet) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=2.0.0->prophet) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=2.0.0->prophet) (4.58.4)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=2.0.0->prophet) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=2.0.0->prophet) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=2.0.0->prophet) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=2.0.0->prophet) (3.2.3)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.0.4->prophet) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.0.4->prophet) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil->holidays<1,>=0.25->prophet) (1.17.0)
```

Step7-

Initialize and Train the Model

```
# Create the model
model = Prophet()

# Train the model
model.fit(df_prophet)
```

```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpov9m21x1/1t8usptq.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpov9m21x1/ix1bnusy.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.11/dist-packages/prophet/stan_model/prophet_
17:11:15 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:11:17 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x7a8fe1184e10>
```

```
Top 10 Predicted Netflix Stock Prices (Future Only):
            ds        yhat  yhat_lower  yhat_upper
5817 2025-07-08  752.424089  640.794100  876.519596
5818 2025-07-09  751.029448  636.819799  868.137121
5819 2025-07-10  749.891054  640.524278  860.754562
5820 2025-07-11  748.078573  621.417743  858.188408
5821 2025-07-12  749.074602  631.457268  878.585600
5822 2025-07-13  748.011264  620.810380  864.772409
5823 2025-07-14  747.253136  627.174255  862.988731
5824 2025-07-15  745.019138  625.509286  866.790413
5825 2025-07-16  743.749443  630.074342  866.621685
5826 2025-07-17  742.815661  626.809012  858.014289
```

```python
# Predict 365 days into the future
future = model.make_future_dataframe(periods=365

# Get the forecast
forecast = model.predict(future)
```

Step8 -

Future predictions

```python
import matplotlib.pyplot as plt
from google.colab import files

# Step 1: Filter only future predictions (after last known date)
future_forecast = forecast[forecast['ds'] > df['Date'].max()]

# Step 2: Display top 10 future values
print("Top 10 Predicted Netflix Stock Prices (Future Only):")
print(future_forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].head(10))

# Step 3: Plot future forecast
plt.figure(figsize=(14, 6))

# Actual historical data
plt.plot(df['Date'], df['Close'], label='Historical Price', color='black', linewidth=2)

# Forecasted price
plt.plot(future_forecast['ds'], future_forecast['yhat'], label='Forecasted Price', color='forestgreen', linewidth=2)
```

```python
# Confidence interval (shaded)
plt.fill_between(future_forecast['ds'],
                 future_forecast['yhat_lower'],
                 future_forecast['yhat_upper'],
                 color='palegreen', alpha=0.5, label='Confidence Interval')

# Labels and styling
plt.title('Netflix Stock Price Forecast (Future Only)', fontsize=18)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Price ($)', fontsize=12)
plt.legend(fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()

# Show the chart
plt.show()

# Step 4: Save forecast to CSV
future_forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].to_csv('netflix_forecast.csv', index=False)
```

Download of the forcaste

```python
files.download('netflix_forecast.csv')
```

Forecast Components

```python
fig2 = model.plot_components(forecast)
```

VI. FORECASTING WITH FACEBOOK PROPHET

Prophet models time series using three main elements: trend, seasonality, and holidays/events. In stock forecasting, the trend and weekly seasonality are particularly significant.

Training the Model:

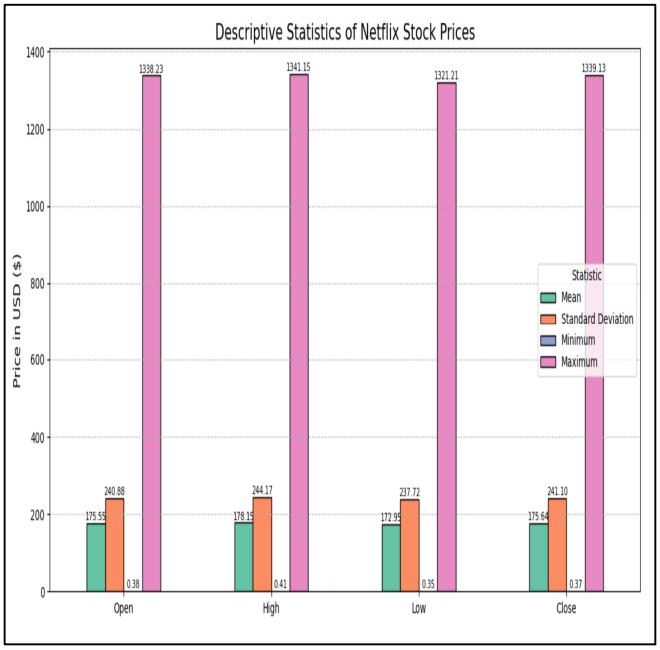
- ✧ Trained Prophet using the cleaned closing price data.
- ✧ The model automatically detected change points, where trend shifts occurred.
- ✧ It also generated upper and lower bounds to reflect uncertainty in the forecast.

Generating the Forecast:

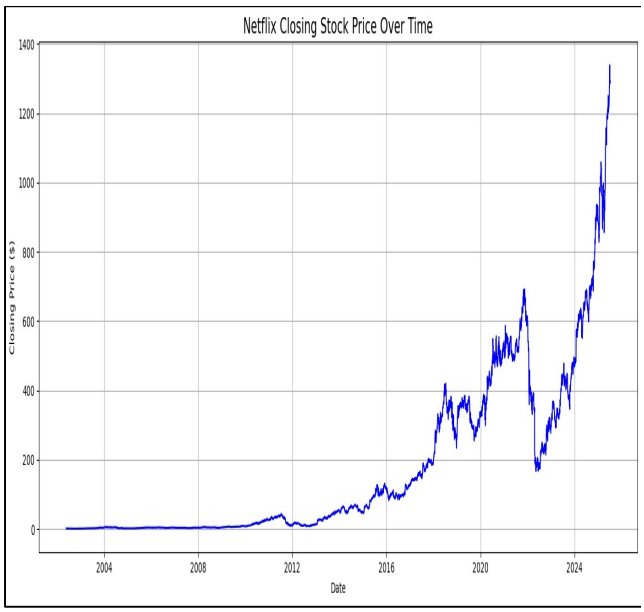
- ✧ A forecast was produced for the next 90 days.
- ✧ The output included:
- ✧ Black line: historical actual
- ✧ Green line: projected values
- ✧ Shaded region: 95% confidence interval
- ✧ Fore Casted data was saved in a CSV format for further exploration.

VII. ANALYZE AND DATA MINING

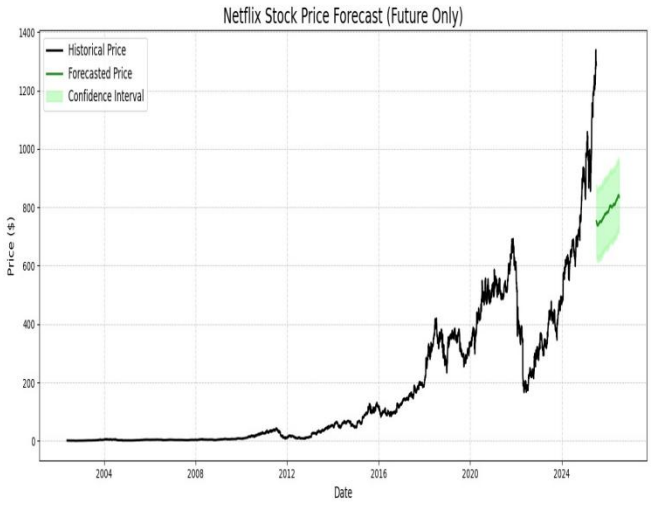
Descriptive statistics



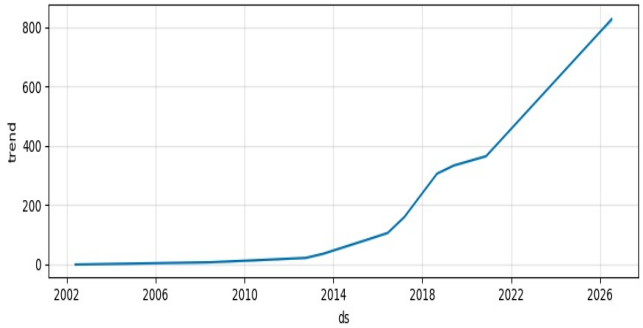
Visualize Historical Stock Prices (Time Series Plot)

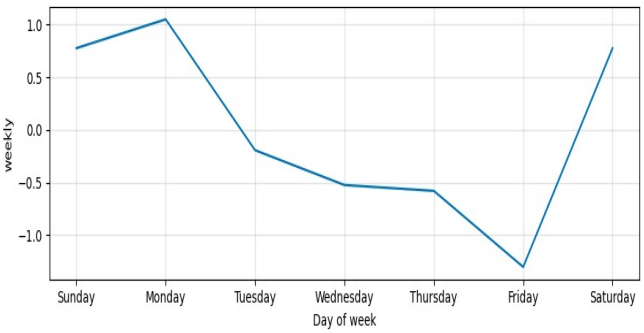


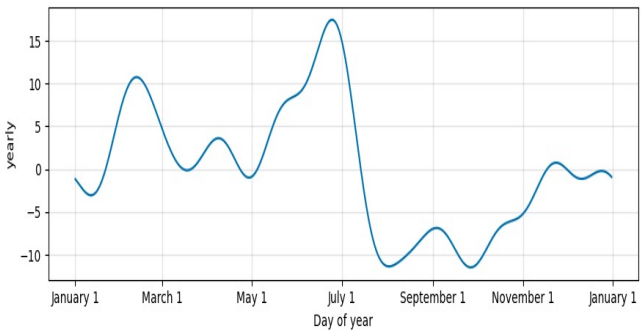
Future predictions



Forecast Components







VIII. Results and Interpretation

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

Insights Gained:

- ✧ Change point Analysis: Prophet clearly identified trend shifts over time.
- ✧ Prediction Ranges: Confidence bands illustrated variability and forecast reliability.
- ✧ Clarity and Accessibility: Visualizations made the forecasts easy to interpret and present.

That said, stock markets are influenced by external factors like earnings reports, global news, or government policies—none of which are accounted for by Prophet. This makes its predictions more general in nature.

IX. Conclusion

Using Facebook Prophet to model and forecast Netflix's stock prices proved to be a reliable and intuitive approach. The model handled the dataset well and produced visually interpretable forecasts with minimal configuration. While Prophet is not suitable for high-frequency trading or day-to-day decisions, it provides a solid foundation for broader financial analysis. The project also demonstrated the importance of clean data and thoughtful preprocessing in achieving meaningful results.

X. Future Improvements

- ✧ Compare Prophet with ARIMA and LSTM models:

  To improve accuracy, future work can include comparing Prophet with ARIMA (a traditional time series model) and LSTM (a deep learning model) to see which performs best on Netflix stock data.

- ✧ Incorporate sentiment analysis:

  Using social media or news sentiment can help capture market emotions. This can be added as a feature to better predict stock price movements.

- ✧ Add macroeconomic indicators:

  Factors like interest rates, inflation, or market indices can be used as external inputs to improve model performance by considering broader economic trends.

XI. References

- Netflix Stock Dataset: https://www.kaggle.com/