Journal

December 2, 2024:

On this day, our team Illyiaz, Kshitij, Shravani, and I, Hitesh discussed a lot about the selection of the project topic and the division of tasks among us. We all came together to discuss what we need to do for the project and what results we want to get. Since our project is going to be immensely dataset- and machine-learning-algorithm-focused, we thought it mandatory to brainstorm and come up with a pertinent topic for our work.

December 5, 2024:

Our next team meeting was about deciding on the project topic. After our last meeting, we all understood what the project was about, and each team member thought of some possible topics. Suggestions included ideas like sentiment analysis and predicting the weather using AI/ML techniques. In the end, we all agreed to work on **"Sentiment Analysis"** using data from Twitter.

December 10, 2024:

We decided to change the topic because we wanted to do something different.
Our teammate Illiyaz had worked on a project in Data Analytics using sentiment analysis with Twitter data. Illiyaz knew algorithms like Logistic Regression and Classifiers but wanted to learn about other algorithms for time series forecasting. After talking as a team, we chose to work on Stock Price Analysis and Prediction using regression models.

December 15, 2024:

We collected the dataset and did the initial analysis.
Our team decided to look at different websites and datasets. Then, we chose good datasets for the project. Kshitij pointed out a built-in library called y-finance, which gives data for different stock prices. The idea sounded good, so we started to figure out how to use y-finance in our project and concentrated on the coding part to make it work well.

December 20, 2024:

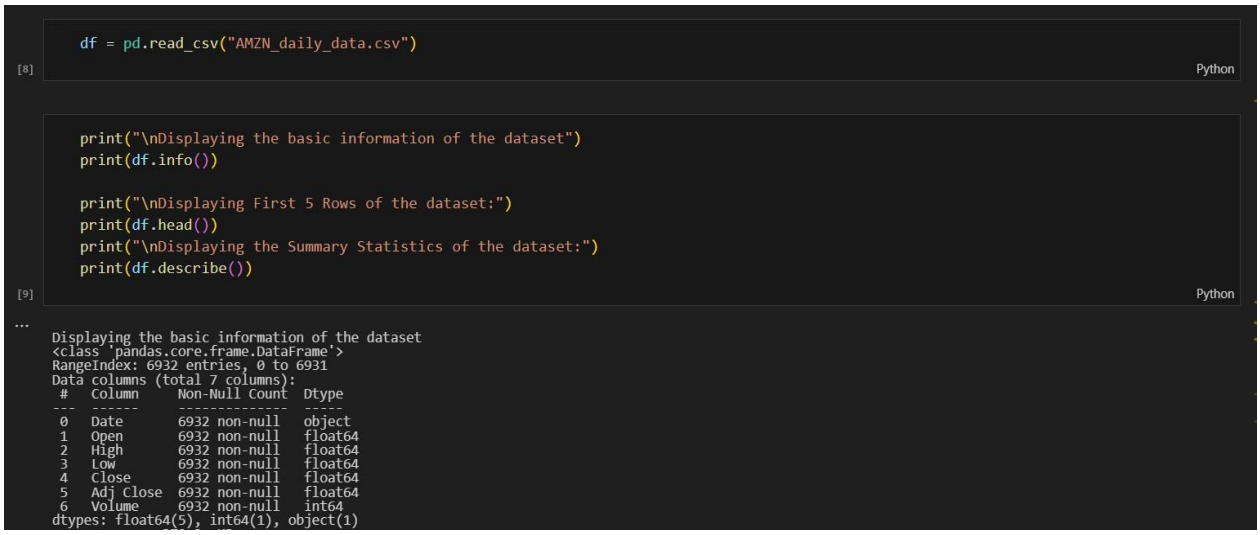
Coding Guide
Kshitij was busy connecting the y-finance module to get the stock data, while Illiyaz was thinking about how he could use machine learning algorithms on the dataset to get the best results. First, Linear Regression seemed a good choice since it is very intuitive, easy to implement, and its results are easy to interpret. Even though Linear Regression can sometimes fit too closely to the training data, this issue can be handled with dimensionality reduction techniques.
However, Illiyaz understood that Linear Regression works best when the data has straight-line relationships. Since stock prices and their values are not usually in a straight line changing unpredictably every day it was decided to look at other regression models that are better for this kind of data.

December 22, 2024:

Changing to proper datasets (CSV files) instead of using the y-finance module.
I mentioned that the y-finance module has some limits and may not fully meet all the needs of this project. Since we will need to collect, prepare data, and constantly check and compare trends, it would be better to build on a good dataset. A
dataset of stock prices containing opening, closing, and high prices for each
day would suit better.

Shravani found a Kaggle dataset that's just right for our project. Here is the link to the dataset: (https://www.kaggle.com/datasets/zongaobian/netflix-stock-data-and-key-affiliated-companies). Netflix Stock Data and Key Affiliated Companies. The dataset contains stock prices for several connected companies. We decided to focus on the task of predicting Netflix stock prices and related stocks like Amazon, Nvidia, and Sony in order to study how they depend on each other.

December 23, 2024:
And each team member chose a dataset that they would work on and started working on it with the coding tasks. I picked the Amazon dataset. First, I imported all the necessary libraries and tools, then loaded the dataset into VS Code. After that, I started analyzing it
by looking at its size and finding the main features.



```
[8]
df = pd.read_csv("AMZN_daily_data.csv")
```
Python

```
[9]
print("\nDisplaying the basic information of the dataset")
print(df.info())

print("\nDisplaying First 5 Rows of the dataset:")
print(df.head())
print("\nDisplaying the Summary Statistics of the dataset:")
print(df.describe())
```
Python

```
...
Displaying the basic information of the dataset
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6932 entries, 0 to 6931
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       6932 non-null   object
 1   Open       6932 non-null   float64
 2   High       6932 non-null   float64
 3   Low        6932 non-null   float64
 4   Close      6932 non-null   float64
 5   Adj Close  6932 non-null   float64
 6   Volume     6932 non-null   int64
dtypes: float64(5), int64(1), object(1)
```

It is important to know the main features and types of data in the dataset, so I chose to visualize it. But first, I had to look for any missing values. Missing values
can cause problems and lead to mistakes when using machine
learning methods or making graphs.

```
print("\nChecking Missing Values in the dataset:")
print(df.isnull().sum())
```

[10] Python

```
Checking Missing Values in the dataset:
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

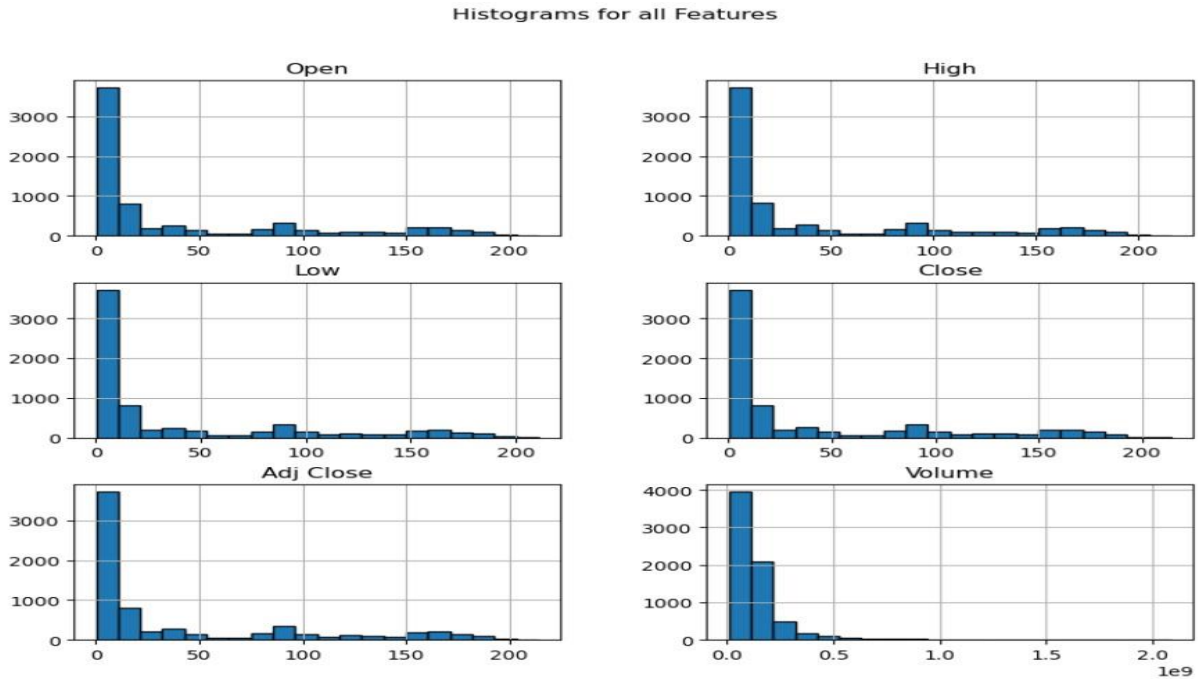
I realized that this data set was already clean, with no null values. So there was no necessity to drop any rows or columns.

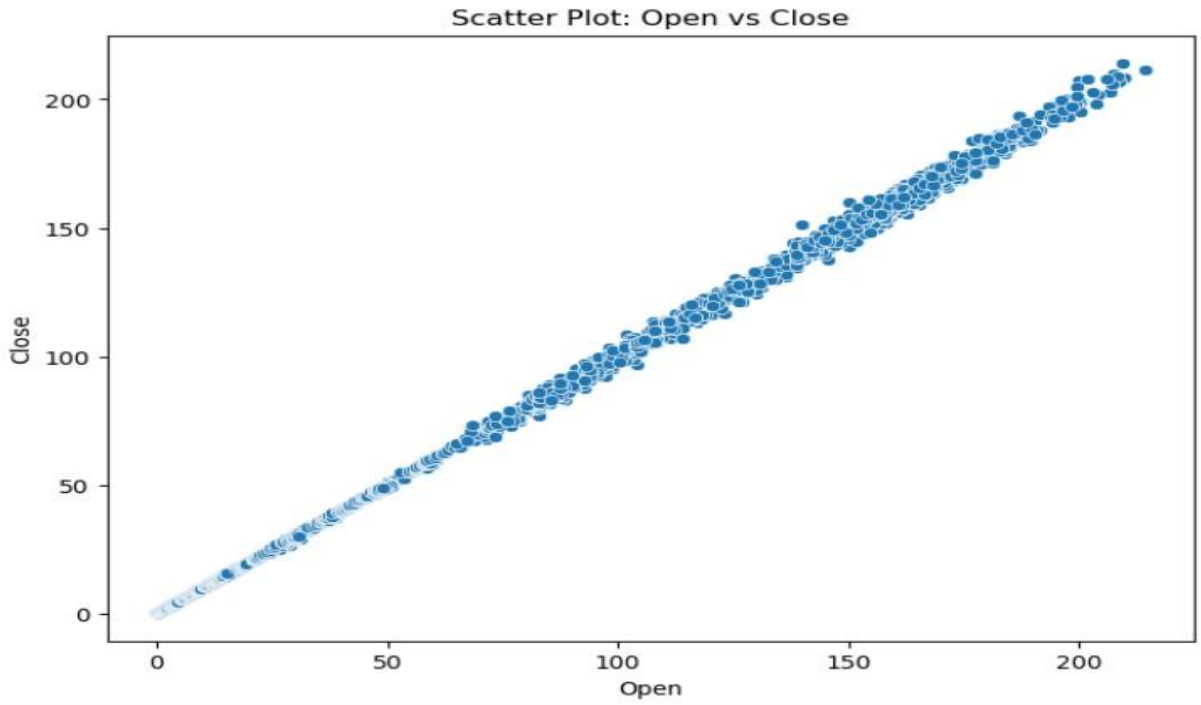
December 24, 2024:

Data Preprocessing and Exploratory Data Analysis (EDA)

After checking that there were no missing values, I thought about removing outliers to make the data easier to use. But then I learned that in stock price prediction, it's not a good idea to remove outliers because every data point matters. We need daily data to correctly analyze and predict future stock prices. So, I decided not to remove any outliers.

Next, I started by visualizing the data. First, I made histograms for all the features so that I could understand their distribution. Then, I plotted the scatter plots between opening and closing stock values to show me the upper and lower limits of the stock prices.





Next, I created pair plots to understand the relationships of all the features against each other. Among the most important ones were the (Open, Close) and (High, Low) pairs for a clear vision of stock prices over the years.

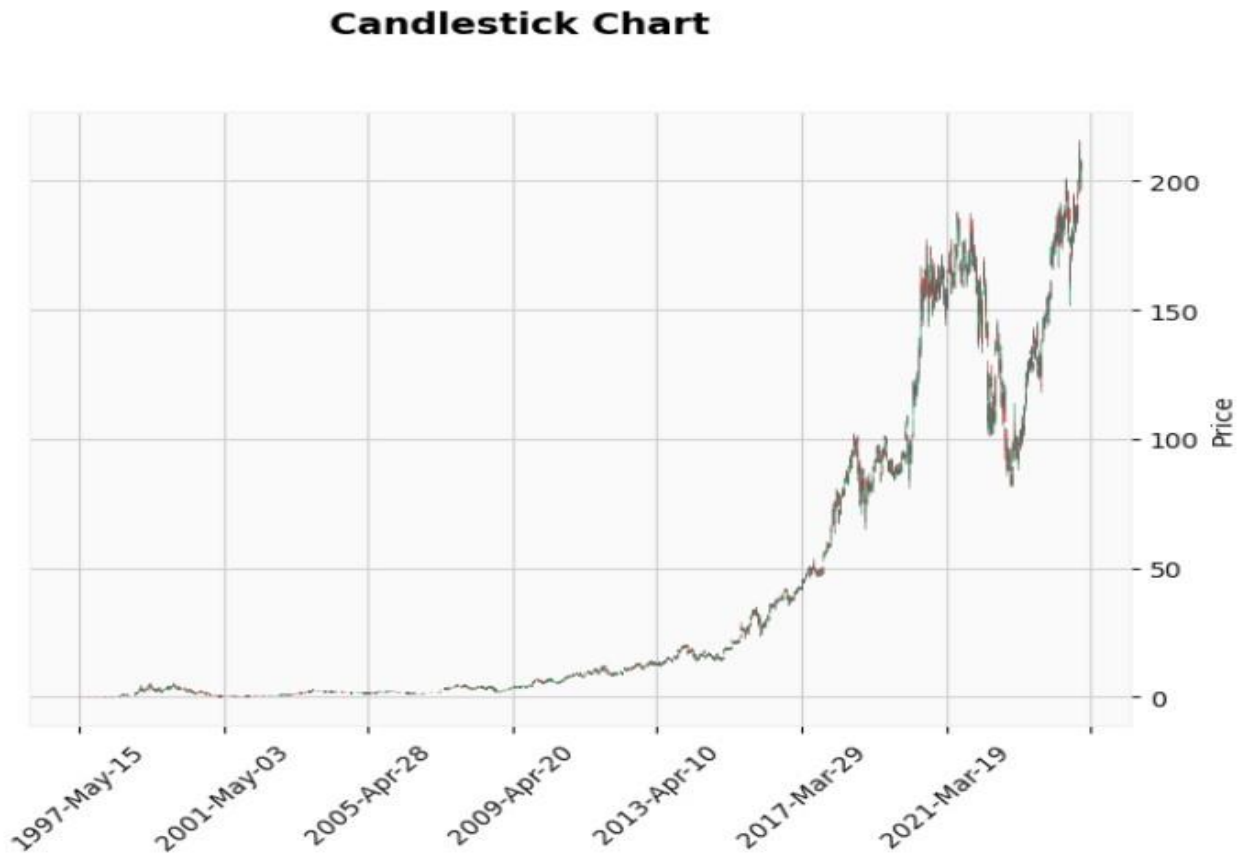
I also made a line chart to show the “Closing Price” over the years. This made it easier to see how much the stock price has gone up over time and to spot trends in how it performed.

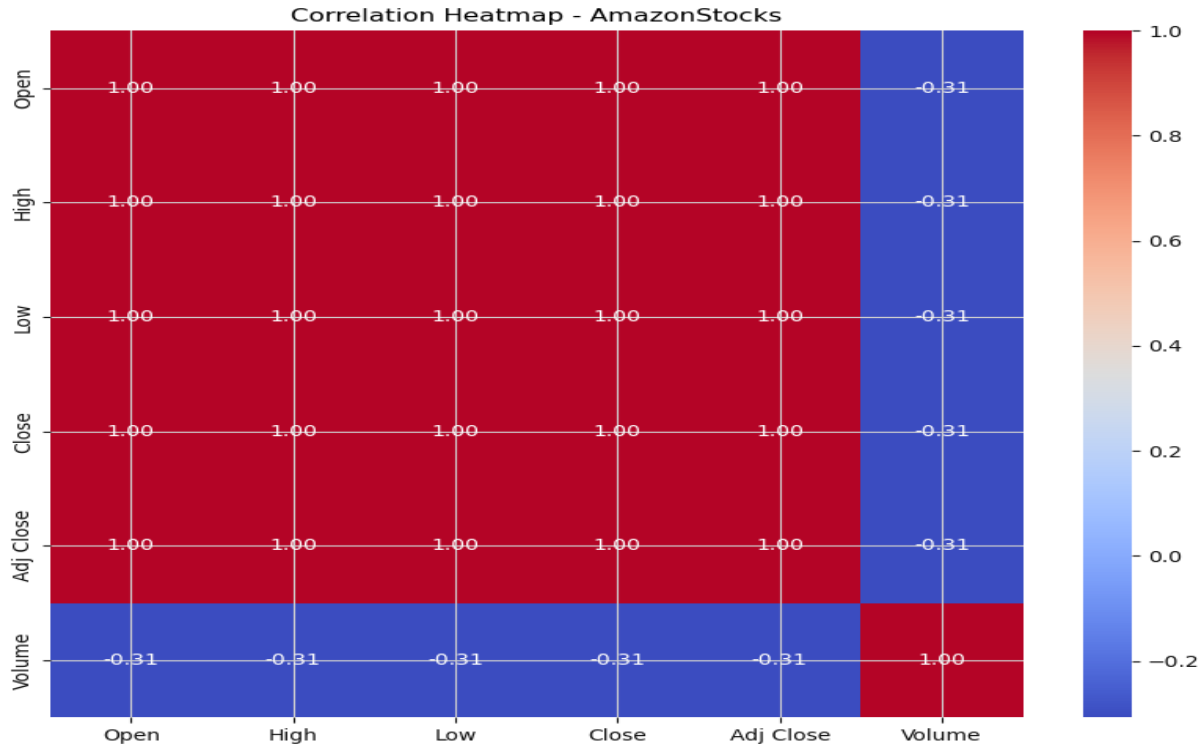
December 26, 2024:

Candlestick Graph Plotting and Heatmap

I really like trading stocks, and one of the most useful ways to see stock data is through candlestick graphs. To incorporate that into my analysis, I searched how to create candlestick graphs for datasets. With help from my team members and some web searches, I discovered the "mplfinance" module, which has made creating candlestick graphs simple.

I studied the syntax, applied it to our dataset, and successfully generated the candlestick graph that gave a detailed visual representation of the stock's performance.





December 28, 2024:

Using Machine Learning Algorithms

This was the hardest and most important part of our project. Our team had to choose the best machine learning algorithms for predicting stocks. To make this work better, we decided that each member would focus on a different ML algorithm, look at the results and graphs, and then pick the best one for our project.

- Shravani picked the Linear Regression model.
- I (Hitesh) chose to use the RandomForestClassifier.
- Kshitij used the LSTM model.
- Illiyaz selected the Prophet model.

For my part, I used the RandomForestClassifier. I began by preparing the data to make sure the algorithm would work well. I first used MinMaxScaler to scale the data because it helps make the model more accurate and efficient. After scaling, I divided the dataset into training and testing sets. Since stock prediction needs the model to learn from more data, I chose an 80%-20% split, using 80% of the data for training and 20% for testing. Now that the data was ready, it was time to train the RandomForestClassifier and examine its results in predicting stocks.

December 29th, 2024:

Many Errors in Code (RandomForestClassifier)
While working with the RandomForestClassifier, I encountered many challenges and errors in using the model. Initially, I was pretty confident in using this algorithm, but as it went

on, I realized that I did not know how to prepare the data correctly for this classifier.

The first error occurred at the preprocessing stage. RandomForestClassifier requires categorical data to be properly encoded, and I had not considered this for some of the features. After encountering errors, I looked up encoding methods like one-hot encoding and label encoding and applied them to the necessary columns in the dataset.

The next challenge was tuning the parameters. The default settings of the RandomForestClassifier did not give good results. To make it better, I tried hyperparameter tuning using GridSearchCV, where I changed settings like the number of estimators, maximum depth, and minimum samples split. This took a lot of time but improved how well the model worked by a large margin.

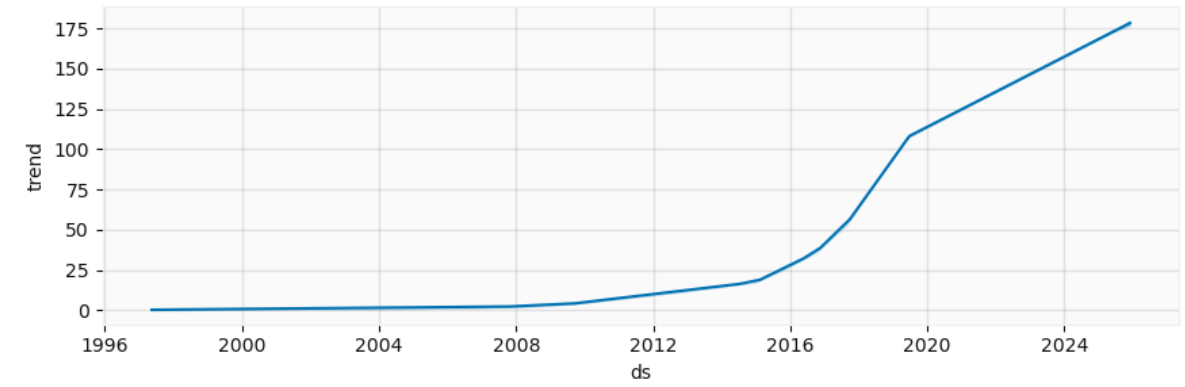
Also, I had problems with feature importance visualization because I didn't know how to get and show the feature importances from the RandomForestClassifier at first. After looking at documentation and online tutorials, I fixed this and managed to create a graph that shows how important each feature is in making predictions.

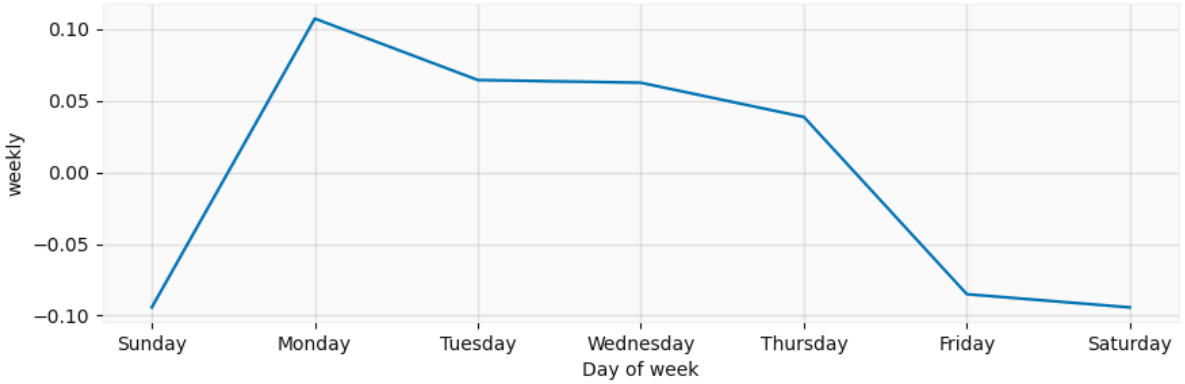
Finally, I split the data into two parts: a training set and a test set. I used an 80%-20% split, which would let the model learn from more data. After training the RandomForestClassifier on the prepared data, I checked how well it was doing using accuracy, precision, and recall.

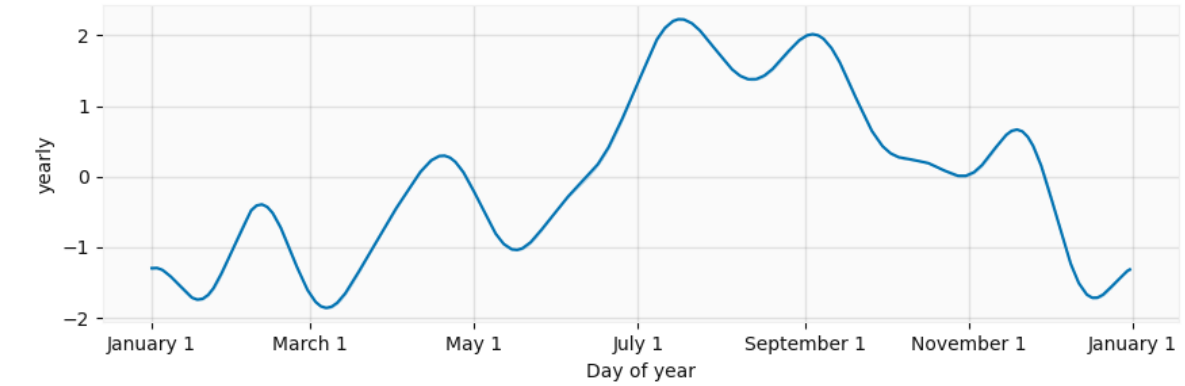
Some charts, like the feature importance graph and a comparison of real vs. predicted stock prices, were created to look at the model's results. Even with the difficulties, the model did a good job and gave useful information for predicting stocks.

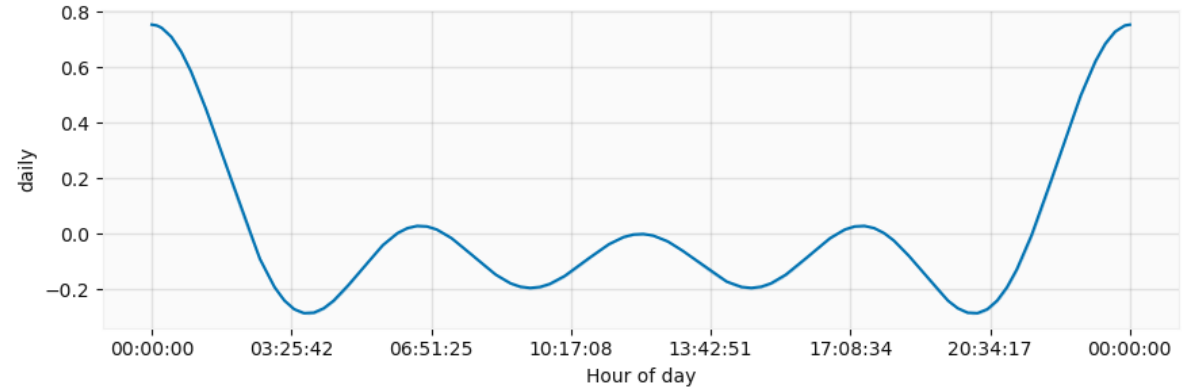
```
# Convert the 'Date' column to datetime format
df['Date'] = pd.to_datetime(df['Date'])

# Prepare data for Prophet (rename columns to 'ds' and 'y')
prophet_df = df[['Date', 'Close']].rename(columns={'Date': 'ds', 'Close': 'y'})

# Initialize the Prophet model
model = Prophet(daily_seasonality=True)

# Fit the model to the data
model.fit(prophet_df)

# Make a dataframe for future predictions
# Let's predict the next 365 days
future = model.make_future_dataframe(periods=365)

# Generate predictions
forecast = model.predict(future)
```


Amazon Stock Price Forecast
Close Price
200
150
100
50
0
1996
2000
2004
2008
2012
2016
2020
2024
Date


trend
175
150
125
100
75
50
25
0
1996
2000
2004
2008
2012
2016
2020
2024
ds


weekly
0.10
0.05
0.00
−0.05
−0.10
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week


yearly
2
1
0
−1
−2
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year


daily
0.8
0.6
0.4
0.2
0.0
−0.2
00:00:00
03:25:42
06:51:25
10:17:08
13:42:51
17:08:34
20:34:17
00:00:00
Hour of day

January 2, 2025:

Working on PPT and Report**

By this time, everybody finished the code and obtained results, but we only selected the best models based on their accuracy and how well they predicted. Since we knew which models were the best at this point, we worked to finalize our report and presentation.

Everybody contributed to the ideas and shared the research papers used for the project. Shravani and Me (Hitesh) helped organize the team and initiated the report writing process. We started collecting information from all the team members and began writing parts like the Introduction, Abstract, and Literature Review. Kshitij and I worked on the Methodology and Results sections, sharing all the important details for the report. **January 3, 2025: PPT**

For the presentation, Kshitij and Illiyaz teamed up to make the slides. We emphasized the main points and key parts of the project in order to make sure that the presentation was clear and useful.

January 4, 2025:

Finishing the Report and PPT

Once the report and PPT were completed, the entire team got together to review everything. We thoroughly checked for grammar errors, formatting issues, and alignment. Upon completion of the necessary changes, we cleaned up the report to ensure that it was visually appealing and professional looking.

January 5, 2025:

Making the Code Consistent Across Datasets**

We completed the report and the PowerPoint presentation, so we aligned the code. We decided to follow the same methods for each dataset, so the project would be clear and consistent. In this way, it would be easier for a person reviewing the code to understand and compare all datasets and results.

We also hosted all the important files on GitHub and added the link in the report. Me and Kshitij helped make instructions and README files to help people run the code more easily.

January 6, 2025:

Video Presentation

The team came together for a video presentation on Teams. I showed my screen, and we each talked about the code and ideas in our project. We also showed how the code worked by running it live during the call.