**Cobblestone Energy**

**Project:** Anomaly Detection in Stock Prices Using Prophet

**Objective**

In this project, we’re using historical stock prices from a chosen company (e.g., AAL for American Airlines) to detect unusual patterns, or anomalies. We’ll use Prophet, a tool developed by Facebook, to analyse past trends and make forecasts. By comparing actual prices to these forecasts, we’ll identify any days where the price was unexpectedly high or low.

**Algorithm used: Prophet**

Prophet is a forecasting tool designed to help us make sense of time series data (data collected over time). It’s particularly useful for data that has repeating patterns, like stock prices.

1. Trend: Prophet identifies a general direction in the data (e.g., prices going up or down).

2. Seasonality: It captures recurring patterns over daily, weekly, and yearly cycles.

3. Confidence Interval: Prophet estimates a range where future values are likely to fall, helping us spot unusual behaviour.

**Step 1: Import Libraries**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from prophet import Prophet

import logging

logging.getLogger('cmdstanpy').setLevel(logging.WARNING)

- pandas and numpy help us load and organize data.

- matplotlib allows us to create visuals for the stock data and detected anomalies.

- Prophet is our main forecasting tool that models the stock prices over time.

- logging is used to reduce unnecessary messages from Prophet.

**Step 2: Load and Preview the Data**

df = pd.read\_csv('all\_stocks\_5yr.csv') # Load the dataset

# Display column names and first few rows to verify structure

print(df.head())

- Load the data: We import stock data over a five-year period from a CSV file.

- Check the structure: By printing column names and a few rows, we get a sense of what’s in the dataset and confirm we have the right columns (like date, stock prices, and stock symbols).

**Step 3: Focus on a Single Stock**

stock\_name = 'AAL' # Specify the stock symbol for analysis

df = df[df['Name'] == stock\_name]

- Filter the data: Since the dataset may contain data for multiple stocks, we narrow it down to a single stock, in this case, American Airlines (AAL).

**Step 4: Prepare Data for Prophet**

df['date'] = pd.to\_datetime(df['date']) # Convert 'date' to datetime format

df.rename(columns={'date': 'ds', 'close': 'y'}, inplace=True) # Rename columns for Prophet

- Convert date format: Prophet requires a specific date format, so we convert the date column to datetime.

- Rename columns: For Prophet to work correctly, we rename the date column to ds (date) and the close column (stock price) to y.

**Step 5: Plot the Historical Stock Prices**

plt.figure(figsize=(10, 5))

plt.plot(df['ds'], df['y'], color='blue')

plt.title(f'Stock Price Data for {stock\_name}')

plt.xlabel('Date')

plt.ylabel('Price')

plt.show()

- Plot the data: We create a simple plot to visualize the stock price history, giving us a sense of how the price has moved over time.

**Step 6: Set Up and Train the Prophet Model**

prophet\_model = Prophet(daily\_seasonality=True, weekly\_seasonality=True, yearly\_seasonality=True)

prophet\_model.fit(df)

- Initialize Prophet: We set up the model to detect daily, weekly, and yearly trends in the stock data, which helps capture any repeating patterns in the prices.

- Train the model: Prophet analyses the historical data, allowing it to learn about the stock’s general trends and seasonal patterns.

**Step 7: Make Forecasts and Detect Anomalies**

# Limit forecast to historical data

future = prophet\_model.make\_future\_dataframe(periods=0, freq='D')

forecast = prophet\_model.predict(future)

- Create a future data frame: We only want to predict values for the dates we already have in our data, so we set periods=0 to prevent Prophet from looking beyond the historical dates.

- Make predictions: Prophet generates a forecast for each date in our dataset, giving us predicted prices and confidence intervals.

**Step 8: Merge Forecasts with Actual Data and Identify Anomalies**

df = df.merge(forecast[['ds', 'yhat', 'yhat\_lower', 'yhat\_upper']], on='ds')

df.rename(columns={'yhat': 'forecast', 'yhat\_lower': 'forecast\_lower', 'yhat\_upper': 'forecast\_upper'}, inplace=True)

- Combine forecasts with actual prices: We merge the forecasted values (Prophet’s output) with the actual stock prices. The forecast (yhat) and its confidence intervals (yhat\_lower and yhat\_upper) are added as new columns.

- Rename for clarity: Renamed yhat to forecast and adjust the other names to make them more intuitive.

df['anomaly'] = 0

df.loc[(df['y'] < df['forecast\_lower']) | (df['y'] > df['forecast\_upper']), 'anomaly'] = 1

- Mark anomalies: For each date, if the actual price is outside the forecasted confidence range, it’s flagged as an anomaly by setting anomaly = 1.

**Step 9: Handle Missing Values for Consistent Plotting**

df['forecast'] = df['forecast'].ffill()

df['forecast\_lower'] = df['forecast\_lower'].ffill()

df['forecast\_upper'] = df['forecast\_upper'].ffill()

- Fill missing values: We use forward-fill to fill any gaps in the forecast columns to ensure a smooth plot without interruptions.

**Step 10: Visualize Stock Prices with Detected Anomalies**

plt.figure(figsize=(14, 7))

plt.plot(df['ds'], df['y'], color='blue', label='Actual Price')

plt.plot(df['ds'], df['forecast'], color='green', linestyle='--', label='Forecasted Price')

plt.fill\_between(df['ds'], df['forecast\_lower'], df['forecast\_upper'], color='gray', alpha=0.3, label='Confidence Interval')

# Highlight anomalies

anomalies = df[df['anomaly'] == 1]

plt.scatter(anomalies['ds'], anomalies['y'], color='red', label='Anomaly', s=50)

plt.xlabel('Date')

plt.ylabel('Price')

plt.title(f'Anomaly Detection in {stock\_name} Stock Prices with Prophet')

plt.legend()

plt.show()

In this project, we use Prophet to forecast stock prices and detect anomalies—days when the actual stock price is unexpectedly high or low. By doing so, we can better understand when stock prices behave unusually.