README

Project Overview

- This project explores sentiment analysis and market trend analysis, combining Telegram data scraping with historical stock market data from Kaggle. The objective is to uncover relationships between social media sentiment and stock market behaviour using techniques like text preprocessing, feature engineering, and statistical analysis.

Steps Followed:-

1. Web Scraping (Telegram Data)

- Data Collection
- Data was scraped from Telegram channels related to the Indian stock market using the Telethon library.
    1. Prerequisites
    2. Visit Telegram App Development Tools.
- Obtain your API ID and API Hash by creating an application.
- Setup and Code
- python

Code:-

```
from telethon import TelegramClient

from telethon.tl.functions.messages import
GetHistoryRequest

import pandas as pd
```

```
import re
import asyncio

# Replace with your credentials
api_id = 'your_api_id'
api_hash = 'your_api_hash'
phone_number = 'your_phone_number'

# Telegram client
client = TelegramClient('session_name', api_id, api_hash)

# Preprocessing function
def preprocess_message(message):
message = re.sub(r"http\S+", "", message)  # Remove URLs
message = re.sub(r"[^a-zA-Z\s]", "", message)  # Remove special characters
message = message.lower().strip()  # Normalize text
return message

# Function to scrape data
async def fetch_telegram_channel_data(channel_username, total_messages=10000):
await client.connect()
if not await client.is_user_authorized():
```

```
await client.send_code_request(phone_number)
await client.sign_in(phone_number, input("Enter the code: "))

channel = await client.get_entity(channel_username)
all_messages = []
offset_id = 0
while len(all_messages) < total_messages:
history = await client(GetHistoryRequest(
peer=channel,
limit=100,
offset_id=offset_id
))
messages = history.messages
if not messages:
break
all_messages.extend(messages)
offset_id = messages[-1].id

processed_messages = [
{"date": msg.date, "message":
preprocess_message(msg.message)}
for msg in all_messages if msg.message
]
df = pd.DataFrame(processed_messages)
```

```
df.to_csv(f"{channel_username}_stock_messages.csv", index=False)

await client.disconnect()


# Scrape multiple channels

channels = ['indian_share_stock_market_group', 'indian_stock_share_market']

for channel in channels:

await fetch_telegram_channel_data(channel)
```

Output

- Processed CSV files for each Telegram channel, containing cleaned and preprocessed messages.

---

2. Historical Stock Data (Nifty)

- **Data Source**
    1. Historical stock price data was downloaded from Kaggle, including all NSE stocks till 2024.
- **Code**

```
import pandas as pd

import numpy as np


# Load and preprocess the dataset
```

```
file_path = "path_to_csv"

data = pd.read_csv(file_path)

data['Date'] = pd.to_datetime(data['Date'], errors='coerce')

filtered_data = data[(data['Date'] >= '2021-01-01') &
(data['Date'] <= '2024-12-31')]


# Forward fill missing data

filtered_data = filtered_data.fillna(method='ffill')

filtered_data['Market_Index'] = filtered_data.iloc[:,
1:].mean(axis=1)


# Calculate hourly trends

def simulate_hourly_market_trends(data):
    hourly_data = []
    timestamps = []
    directions = []
    for i in range(len(data)):
        row = data.iloc[i]
        for hour in range(24):
            if i > 0:
                prev_row = data.iloc[i - 1]
                daily_change = (row['Market_Index'] -
prev_row['Market_Index']) / 24
                hourly_market_index = prev_row['Market_Index'] +
daily_change * hour
```

```
            else:
                hourly_market_index = row['Market_Index']
            direction = "Increased" if hourly_data and
hourly_market_index > hourly_data[-1] else "Decreased"
            hourly_data.append(hourly_market_index)
            timestamps.append(row['Date'] +
pd.Timedelta(hours=hour))
            directions.append(direction)
    return pd.DataFrame({'Timestamp': timestamps,
'Hourly_Market_Index': hourly_data, 'Direction': directions})


hourly_trends = simulate_hourly_market_trends(filtered_data)
hourly_trends.to_csv("hourly_market_trends_with_directions.c
sv", index=False)
```

- **Output**

  Generated hourly_market_trends_with_directions.csv, containing:

  - Timestamp: Exact time of hourly data points.
  - Hourly_Market_Index: Market index value for each hour.
  - Direction: Indicates whether the market increased or decreased.

3.Data Preprocessing

1. Text Cleaning Pipeline

- Removed links, special characters, stopwords, and HTML tags.
- Expanded abbreviations (e.g., "LOL" → "laughing out loud").
- Converted emojis to text using emoji.demojize.
- Standardized text to lowercase.
- Corrected spelling with TextBlob.
- Focused on meaningful words by removing stopwords.

2. Combining Processed Data

Preprocessed chunks of data (10k, 15k, 6k rows) were merged for sentiment analysis.

4.Sentiment Analysis

1. Using Vader Sentiment Analyzer:-Custom stock market-related terms were added to Vader's lexicon:

- Positive: "bullish," "breakout," "gains."
- Negative: "bearish," "crash," "loss."
- Neutral: "valuation," "support," "resistance."

2. Sentiment Labelling:-Messages were labeled as Positive, Negative, or Neutral based on the compound sentiment score.

3. Results:-Sentiment-labeled datasets were exported for further analysis.

5.Model Development

1. Text Feature Extraction

- Used CountVectorizer to represent text data as numerical features.
- Applied English stopword removal.

2. Train-Test Split:-Split data into 80% training and 20% testing sets.

3. Random Forest Classifier

- Trained a Random Forest model on vectorized text features.
- Achieved initial accuracy, followed by 5-fold cross-validation.

4.Heuristic Features:-We aimed to incorporate heuristic features into the model but faced computational constraints due to which we could not implement. Some of the features planned:

- Message-Based:
  - Length of the text (text_length) and word count (word_count).
  - Financial keyword presence (keyword_count).
- Time-Based:
  - Extracted hour of day, day of week, is_weekend, pre-market, and post-market flags from the timestamp.
- Sentiment-Based:
  - Polarity score calculated using TextBlob.

Files Generated

1. Telegram Data: Cleaned CSV files for each channel.
2. Market Trends: hourly_market_trends_with_directions.csv.

3. 3 csv files after text preprocessing
4. 1 csv file after sentiment labelling and concatenation

Usage Instructions

1. Install dependencies:

Bash:-

pip install telethon pandas numpy

2. Set up Telegram API credentials.
3. Run the scraper script to collect Telegram data.
4. Use the Kaggle dataset and preprocessing script to calculate hourly trends.