|  |
| --- |
| Title: Stock Sentiment Analyzer using Macgine Learning |
| Done by : Pritesh Ram Keshri Project Number : 17, Submission Date: 21st July 2025 |
| **Overview:**  This project aims to assist traders and investors by analyzing the **public sentiment** of stock market tickers using recent **news headlines**. The tool scrapes headlines via NewsAPI and classifies them using a transformer-based sentiment analysis model (twitter-roberta-base-sentiment). Based on the analysis, it predicts market mood as **Bullish**, **Bearish**, or **Neutral**. |
| **Libraries used:**   * **transformers, torch –** Sentiment model loading and prediction * **requests –** API communication * **streamlit –** Interactive web UI * **snscrape –** Optional Twitter scraping module * **numpy, pandas –** Data structuring |
| **Dataset Details:**   * **News Headlines Source:** [**NewsAPI.org**](https://newsapi.org/) * **Model: cardiffnlp/twitter-roberta-base-sentiment (Pretrained on Twitter)** * **Stock Input: Any valid stock ticker or company name** * **Output:** * **Sentiment distribution (% Positive, % Neutral, % Negative)** * **Market sentiment classification: Bullish, Bearish, or Neutral** |
| **APIs Integrated:**  No external APIs were used. |
| **Source code 1: File Name : app.py — Main Streamlit Web App**  import streamlit as st  from news\_scraper import get\_news\_headlines  from sentiment\_model import analyze\_sentiment  from utils import predict\_market\_impact  from transformers import AutoTokenizer, AutoModelForSequenceClassification  st.set\_page\_config(page\_title="📊 Stock Sentiment Analyzer", layout="centered")  @st.cache\_resource  def load\_model():  tokenizer = AutoTokenizer.from\_pretrained("cardiffnlp/twitter-roberta-base-sentiment")  model = AutoModelForSequenceClassification.from\_pretrained("cardiffnlp/twitter-roberta-base-sentiment")  return tokenizer, model  tokenizer, model = load\_model()  st.title("📊 Stock Sentiment Analyzer")  st.write("Analyze news sentiment for your favorite stock and predict market mood.")  stock\_input = st.text\_input("Enter stock/company name (e.g., TCS, INFY, Tata Consultancy Services)")  keyword\_map = {  "TCS": "Tata Consultancy Services",  "INFY": "Infosys",  "RELIANCE": "Reliance Industries"  }  api\_key = "" # Replace with your NewsAPI key  if stock\_input:  query = keyword\_map.get(stock\_input.upper(), stock\_input)  st.info(f"🔍 Fetching news for: \*\*{query}\*\*")  headlines = get\_news\_headlines(query, api\_key)  if not headlines:  st.error("❌ No relevant news found.")  else:  st.subheader("📰 Top Headlines")  for i, headline in enumerate(headlines, 1):  st.markdown(f"{i}. {headline}")  st.subheader("🧠 Sentiment Analysis")  sentiment\_scores = analyze\_sentiment(headlines, tokenizer, model)  st.write("### 📊 Sentiment Breakdown")  st.bar\_chart(sentiment\_scores)  impact = predict\_market\_impact(sentiment\_scores)  st.success(f"🔮 \*\*Market Prediction\*\*: {impact}") |
| **Source code 2: news\_scraper.py — NewsAPI Integration**  import requests  def get\_news\_headlines(stock\_name, api\_key):  url = f"https://newsapi.org/v2/everything?q={stock\_name}&apiKey={api\_key}"  res = requests.get(url).json()  return [article['title'] for article in res['articles'][:10]] |
| **Source code 3: sentiment\_model.py — Sentiment Model Loader & Analyzer**  from transformers import AutoTokenizer, AutoModelForSequenceClassification  import torch  def load\_model():  model\_name = "cardiffnlp/twitter-roberta-base-sentiment"  tokenizer = AutoTokenizer.from\_pretrained(model\_name)  model = AutoModelForSequenceClassification.from\_pretrained(model\_name)  return tokenizer, model  def analyze\_sentiment(texts, tokenizer, model):  labels = ['negative', 'neutral', 'positive']  results = {l: 0 for l in labels}  for text in texts:  inputs = tokenizer(text, return\_tensors="pt", truncation=True)  with torch.no\_grad():  outputs = model(\*\*inputs)  scores = torch.nn.functional.softmax(outputs.logits, dim=1)  pred = labels[torch.argmax(scores)]  results[pred] += 1  total = sum(results.values())  return {k: round((v / total) \* 100, 2) for k, v in results.items()} |
|  |
| **Source Code 5: utils.py — Market Impact Prediction**  def predict\_market\_impact(sentiment\_scores):  if sentiment\_scores["positive"] > sentiment\_scores["negative"]:  return "Bullish"  elif sentiment\_scores["negative"] > sentiment\_scores["positive"]:  return "Bearish"  else:  return "Neutral" |
| **Output screenshots:**  **1.** **Initial App Interface-** This is the **landing interface** of the Stock Sentiment Analyzer.    **2.** **Sentiment Output for Stock Query (TCS): Coffee:** Once a stock name is entered (e.g., **TCS**), the app displays. |
| **What you learned:**  Through this project, I gained practical knowledge of:   * Practical integration of NLP transformer models (HuggingFace) * Web-based sentiment dashboard building using Streamlit * API integration and caching * Real-time data processing |
| **What the Skills you gained:**   * Transformer-based Sentiment Analysis * Streamlit UI Development * News API Integration * Sentiment Aggregation & Visualization * API Rate Handling and Error Handling |
| **Real Time Applications:**   * **Investor Sentiment Analysis**: Gauge public mood for specific stocks based on news headlines. * **Trading Signal Generator**: Trigger automated bullish/bearish/neutral signals based on aggregated sentiment. * **News Sentiment Monitoring Tool**: Track how sentiment evolves over time for companies in the stock market. * **Social + News Sentiment Fusion**: Combine Twitter and news headline sentiments for a broader perspective. * **Portfolio Risk Estimation**: Predict sentiment-based volatility for individual or grouped stocks. * **Real-Time Stock Dashboard**: Provide investors with a live dashboard summarizing market emotion. |
| **Further Enhancement Suggestions :**   1. **Twitter Sentiment Integration**: Combine live tweets with news headlines for richer sentiment data. 2. **Sentiment Trend Visualization**: Visualize sentiment movement over past days or weeks. 3. **Stock Price Correlation**: Integrate financial APIs (e.g., Alpha Vantage, Yahoo Finance) to correlate sentiment with stock prices. 4. **Multilingual Support**: Use translation APIs to analyze non-English news content. 5. **Mobile App Deployment**: Convert the app to a mobile interface for traders on-the-go. |