Stock Market Prediction

Using Sentiment Analysis and LSTM

Introduction

- **Background on stock market prediction**

  Importance of sentiment analysis in financial markets:

  - Sentiment analysis plays a crucial role in financial markets by analyzing emotions and opinions from news, social media, and reports, providing real-time insights into investor behavior.
  - Sentiment analysis reduces risks by identifying positive or negative trends early on.

  Brief introduction to LSTM networks:

  - LSTMs utilize memory cells to store information over long periods, overcoming the limitations of traditional RNNs.
  - They are particularly effective for tasks involving time-series data, such as natural language processing and stock market prediction, where understanding context and sequences is vital.

Introduction

- **Background on stock market prediction**

  Objectives and Goals:

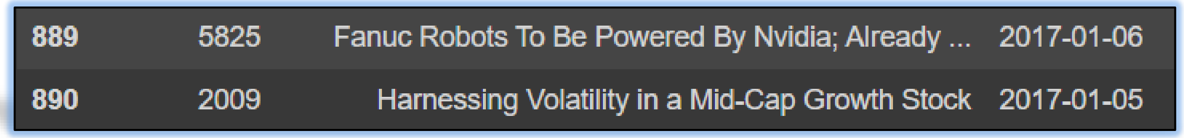
  - Using a transformers model for sentiment analysis of news headlines for each day.
  - To create an LSTM model which takes previous data as input and predicts the closing price of stock for the next day.
  - To create a simple web app where user can enter the ticker of a particular company and the model predicts the stock price for the next 7 days along with a label to buy, sell or hold the particular stock.

Technical Stack and Architecture

- **Python: Core programming language**
  - Flask: Web application framework
  - TensorFlow: LSTM model library
  - Pandas: Data manipulation
  - yfinance: Fetch stock data
  - Transformers: Sentiment Analysis
  - HTML, CSS: Frontend

Data Collection and Preprocessing

- **News Headlines data from Kaggle**
  - We used a Kaggle dataset publicly available at headlines which contains the daily news headlines for 11 years.

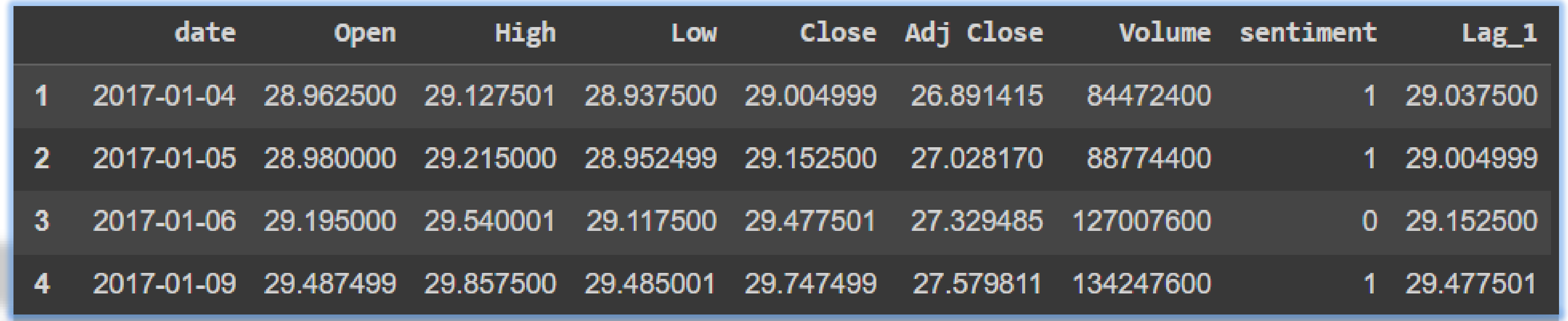


889	5825	Fanuc Robots To Be Powered By Nvidia; Already ...	2017-01-06
890	2009	Harnessing Volatility in a Mid-Cap Growth Stock	2017-01-05

  - We manipulated the data using Pandas into desired format and used the HugginggFace model to predict the sentiment score for each headline and added the sentiment column.

Data Collection and Preprocessing

- **Historical stock data from yfinance**
  - Then we collected the historical price data from the yfinance library, processed and merged it with the sentiment scores calculated earlier.



	date	Open	High	Low	Close	Adj Close	Volume	sentiment	Lag_1
1	2017-01-04	28.962500	29.127501	28.937500	29.004999	26.891415	84472400	1	29.037500
2	2017-01-05	28.980000	29.215000	28.952499	29.152500	27.028170	88774400	1	29.004999
3	2017-01-06	29.195000	29.540001	29.117500	29.477501	27.329485	127007600	0	29.152500
4	2017-01-09	29.487499	29.857500	29.485001	29.747499	27.579811	134247600	1	29.477501

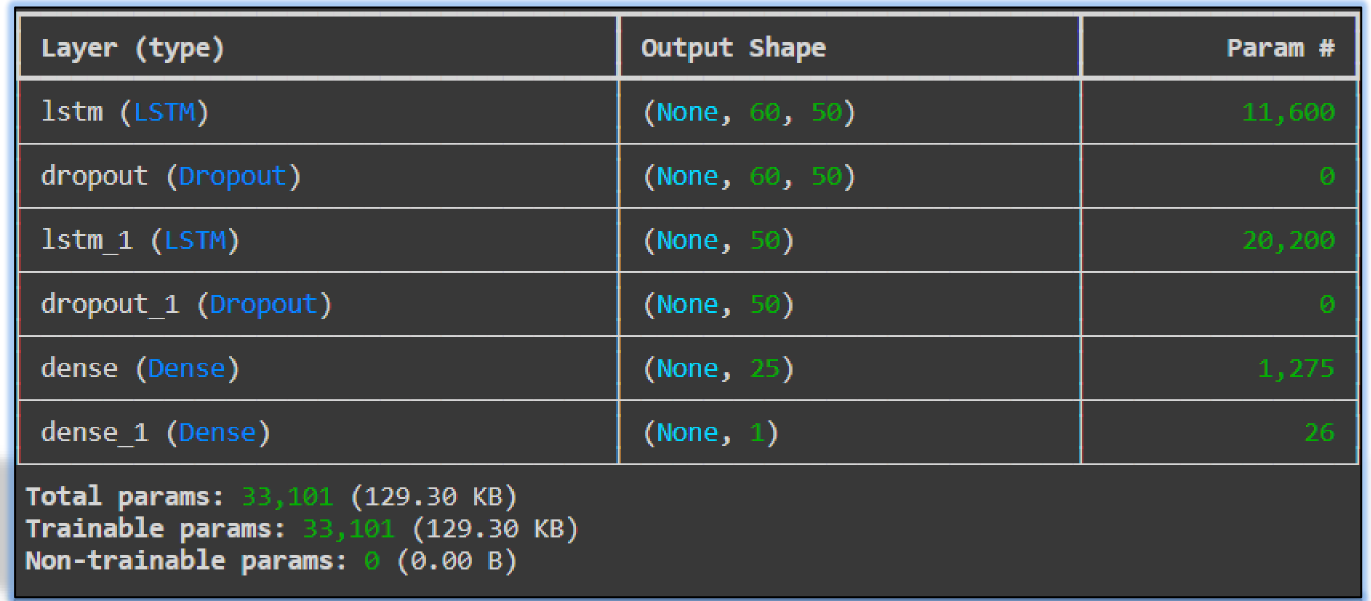
  - The final preprocessed data contained the above columns for 60 previous days, which was given to LSTM model as input, and it predicts the stock price for the next working day.

Model Training and Evaluation

- **Training process:**
  - We used Keras's Train_test split to split the data 80-20.
  - We trained the model of 3 years of data and tested on 20% of it.
  - The Evaluation metrics which we used are:
    - Mean Absolute Error (MAE)
    - Root Mean Squared Error (RMSE)

LSTM Model Architecture

- **Long Short-Term Memory (LSTM) neural network**
  - Model Architecture



Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 60, 50)	11,600
dropout (Dropout)	(None, 60, 50)	0
lstm_1 (LSTM)	(None, 50)	20,200
dropout_1 (Dropout)	(None, 50)	0
dense (Dense)	(None, 25)	1,275
dense_1 (Dense)	(None, 1)	26

```
Total params: 33,101 (129.30 KB)
Trainable params: 33,101 (129.30 KB)
Non-trainable params: 0 (0.00 B)
```

  - Input features:
    - Stock details for previous 60 days (Input_shape: (60, 7))

- The link to our google colab file*

Stock Price Prediction

- **Prediction process:**
  - Input: Last 60 days of data in numpy arrays format.
  - Output: Next 7 days' predicted prices.
  - Decision making:
    - Buy/Hold/Sell based on predicted price movement.

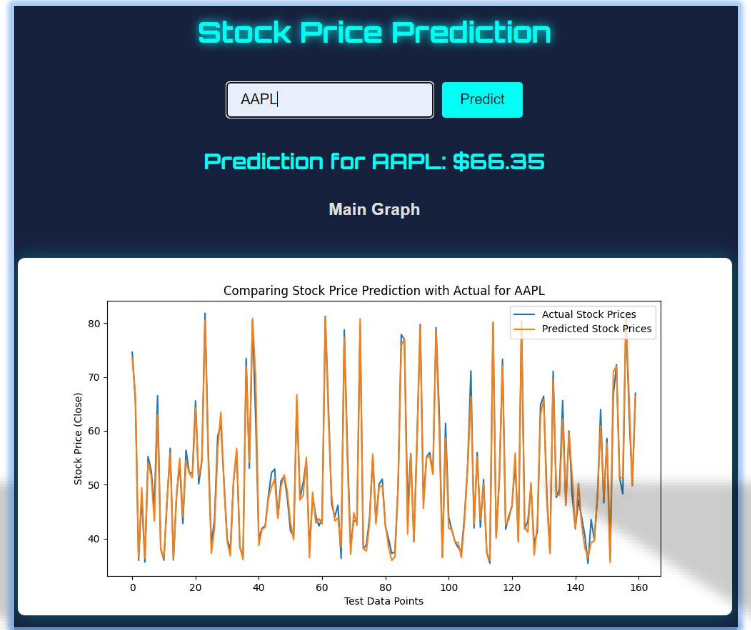
- **Inference:**
    - Our model with sentiment Analysis has given 94.25% accuracy, while the one without sentiment has provided only 87.6% accuracy.
    - This shows how important Sentiment Analysis becomes.

Stock Price Prediction WebApp

- **WebApp:**
  - For the WebApp of this stock price prediction model, we used Flask designed for applications in python.
  - For the FrontEnd part we used HTML, CSS.
  - On our WebApp our user can simply type the ticker for the particular company and submit.
  - Our Webapp provides the Graph for comparing the Stock price prediction for the Test data, along with the prediction graph for next 7 days.
  - And finally our App also displays the decision whether buying/holding/selling the particular stock.

Stock Price Prediction WebApp

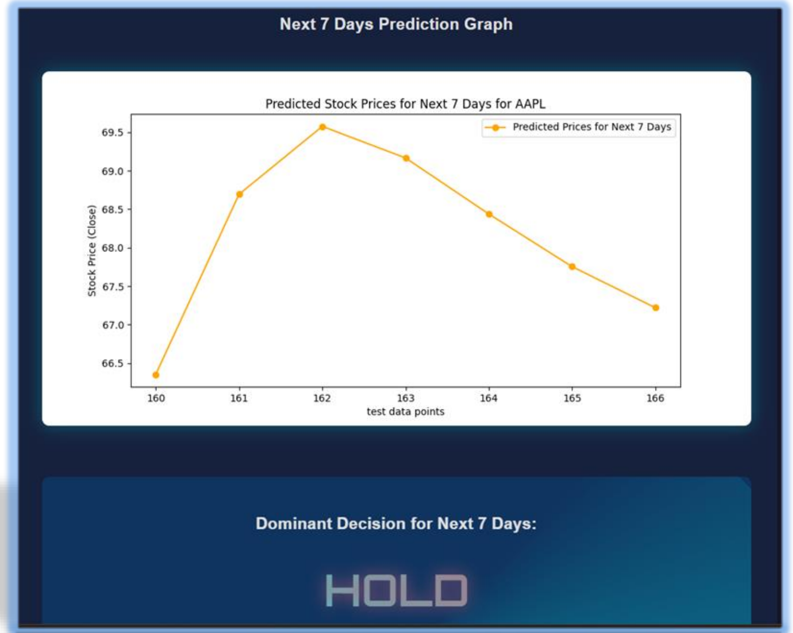
- **WebApp:**
  - Our WebApp Interface:



Stock Price Prediction

Prediction for AAPL: $66.35

Main Graph


Predicted Stock Prices for Next 7 Days for AAPL
Predicted Prices for Next 7 Days
Stock Price (Close)
test data points

Next 7 Days Prediction Graph

Dominant Decision for Next 7 Days:

HOLD

Challenges Faced

- **Challenges**
  - Choosing best model to provide the sentiment scores.
  - Data availability.
  - Tuning of hyperparameters for high accuracy.
  - Challenges while Integrating our model into WebApp.
  - Issues on our localhost while connecting to backend Flask app.

Conclusion and Future Work

- **Summary of project achievements**
  - We are able to implement a simple web application for user to decide the stock they should be preferring using sentiment analysis and LSTM model.

- **Future Work**