A DEEP LEARNING APPROACH TO STOCK PREDICTION INTEGRATED BY SENTIMENT ANALYSIS

Boon Si Xian

Asia Pacific University

1 July

Table of Content

Brief Background

Deep Learning (DL)

DL is a branch of machine learning that uses multi-layered neural networks for autonomous learning and decision-making. It excels in stock price prediction by processing large volumes of data and identifying complex patterns.

Sentiment Analysis (SA)

SA evaluates market sentiment using NLP on textual data from sources like social media and news articles. It scores text tokens to gauge sentiment, significantly influencing stock market behavior.

Problem Statement

Challenges with Traditional Methods

- **High volatility:** Influenced by economic, political, and psychological factors.
- Fails to consider public sentiment.

Disadvantages of DL

DL lacks the ability to take into account of public sentiment:

- Financial news can effect the sentiment of public.
- Public sentiment has an important impact on the stock market.

An idea of adding in Sentiment data to DL occurred.

Aim & Objectives

Aim

Develop a DL model to predict stock prices, enhanced by sentiment data.

Objectives

1. Extract public sentiment from textual data Using NLP to enhance prediction accuracy.
2. Develop a DL model combining historical and sentiment data for stock price prediction.
3. Evaluate models using metrics like RMSE and MAE to identify the best model.

Brief Significance

Importance of the Project

- Provides an innovative method potentially superior to traditional models.
- Integrates historical data with sentiment analysis for improved forecasting.
- Offers strategic insights for investors, enhancing decision-making and risk management.
- Expands DL and SA applications beyond stock prediction to areas like political analysis and customer churn studies.
- Contributes educational resources and an open-source model for broader use.

Literature Review

Traditional vs. Modern Approaches

- Traditional methods focus on technical and fundamental analysis.
- Modern deep learning (DL) methods are better.
- Long Short-Term Memory (LSTM) networks can:
  - uncover complex patterns,
  - handle non-linear relationships better.

(Bansal, 2022)

LSTM for Time Series

- LSTM is particularly useful for capturing long-term dependencies in time series data.
- Hence, it is suitable for predicting future stock prices (Velarde et al., 2022).

Literature Review

Role of Sentiment Analysis

- Extracts emotions and opinions from textual data.
- It provides insights into public sentiment that can influence stock prices (Halder, 2022).

Model-Based Integration

Sentiment data adjusts base predictive models, helping to reflect current sentiment and its impact on stock prices (Li, 2022).

Literature Review

Data Cleaning and Transformation

Ensures the quality of data, which is crucial for model performance. Techniques include normalization, standardization, and feature selection (Bansal, 2022).

LSTM and GRU

LSTM handles the temporal dependencies of time series data well. LSTM is known for its ability to remember long-term patterns, while GRU offers similar performance with reduced complexity (Velarde et al., 2022).

Methodology

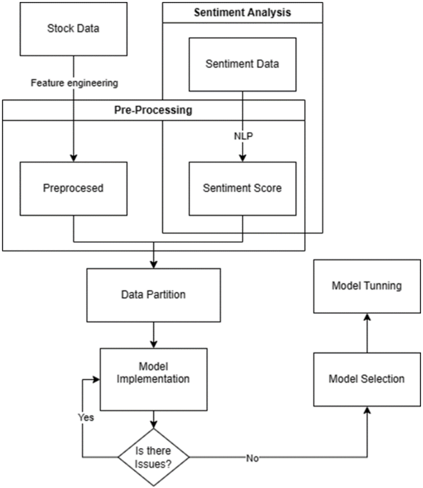

Figure: Framework

Methodology

How will the data be collected?

- Historical and sentiment data will be obtained from Kaggle.
- It provides the High, Low, Open, Close, Volume and Adjusted Close price of the stock.
- The project will be focusing on short-term price fluctuations (1 Year).
- The date and time of the tweet, full text of tweet, the related stock name to the tweet are provided.
- The limitation is less popular stock has less tweets related to it.

Methodology

Data Preprocessing Techniques

- Imputation
- Scaling and Normalization (via MinMaxScaler)
- Get sentiment score via VADER from NLTK
- Feature Transformation: Technical indicators will be included.

How will the data be partitioned?

- First 80% will be training set.
- Last 20% will be the testing set.

Methodology

The LSTM (DL model) will be chosen in this task. It is because LSTM (Long Short-Term Memory) networks are highly effective due to their ability to capture temporal dependencies (Halder, 2022).

Model Architecture

LSTM layer -Dropout layer - LSTM layer - Dropout layer – Dense layer

Justifications:

- We put the dropout layer after the LSTM layer to perform regularization on the model.
- Multiple LSTM layer is used to help in capturing more complex patterns.

Methodology

Evaluation

RMSE and MAE metrics will be chosen, and they are more popular metrics in the papers that were studied (Bansal M., 2022; Kamal, 2021; Darapaneni, 2022).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2},$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|.$$

Where y_i, $\hat{y}_i$, n represent the actual price, predicted price and number of observations.

Model Implementation

Data Exploration and Understanding

- Dataset Overview:
  - Accessed stock tweets CSV from Kaggle.
  - Key columns: Date, Tweet Text, Stock Name.
  - Focused on TSLA, AAPL, and GOOG due to tweet frequency.
- Initial Observations:
  - Dates are in GMT.
  - TSLA has the highest tweet volume.
  - Filtering decision: Select TSLA, AAPL, GOOG for analysis.

Model Implementation

Data Pre-processing

- **Sentiment Data**
  - Sentiment Analysis using VADER:
    - Employed VADER from NLTK for sentiment scores.
    - Normalized text and calculated sentiment scores.
  - Data Aggregation and Merging:
    - Filtered tweets based on market hours.
    - Aggregated sentiment metrics by stock and date.
  - Time Alignment:
    - Shifted sentiment data for time-series analysis.

```
sent_df.head()
```

	Date	Tweet	Stock Name	sentiment_score	Negative	Neutral	Positive
0	2022-09-29 23:41:16+00:00	Mainstream media has done an amazing job at br...	TSLA	0.0772	0.127	0.758	0.115
1	2022-09-29 23:24:43+00:00	Tesla delivery estimates are at around 364k fr...	TSLA	0.0	0.0	1.0	0.0
2	2022-09-29 23:18:08+00:00	3/ Even if I include 63.0M unvested RSUs as of...	TSLA	0.296	0.0	0.951	0.049
3	2022-09-29 22:40:07+00:00	@RealDanODowd @WholeMarsBlog @Tesla Hahaha why...	TSLA	-0.7568	0.273	0.59	0.137
4	2022-09-29 22:27:05+00:00	@RealDanODowd @Tesla Stop trying to kill kids,...	TSLA	-0.875	0.526	0.474	0.0

Figure: head of sent df

```
tweet_ds.head()
```

Python

dex	Stock Name	Date	Mean_sentiment_intraday	Min_sentiment_intraday	Max_sentiment_intraday	Mean_sentiment_before_sod	Min_sentiment_before_sod	Max_sentiment_before_sod
0	AAPL	2021-09-30	0.2739	0.2296	0.3182	0.39536	0.0	0.6892
1	AAPL	2021-10-07	0.1484	0.0	0.368	0.32482	0.0	0.8538
2	AAPL	2021-10-08	0.7842	0.7778	0.7906	0.231475	-0.4215	0.8597
3	AAPL	2021-10-10	0.3612	0.3612	0.3612	-0.01315	-0.5859	0.2023
4	AAPL	2021-10-13	-0.4278	-0.4278	-0.4278	0.516231	0.0	0.893

Figure: Final sentiment data

Model Implementation

Data Pre-processing

- **Stock Data**
  - Historical Stock Data:
    - Retrieved historical prices.
    - Calculated percentage change in Close prices and Volume.
  - Feature Engineering:
    - Created market movement feature based on percentage change.
    - Initial application of Random Forest on the market movement highlighted class imbalance.

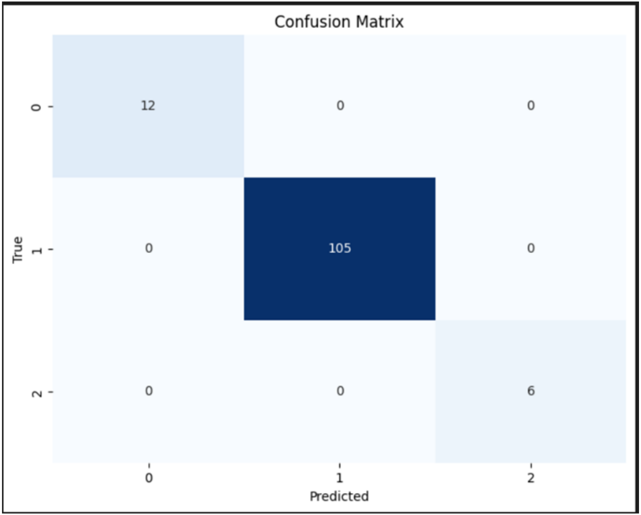

Figure: Confusion Matrix

Model Implementation

Data Visualization and Technical Indicators

- Visualization of Stock Data:
  - Plotted stock prices for AAPL, GOOG, META, TSLA.
- Technical Indicators:
  - Computed 7-day and 20-day moving averages (MA7, MA20).
  - Added indicators like RSI and observed trends.
  - Data scaling to (-1, 1) using MinMaxScaler.

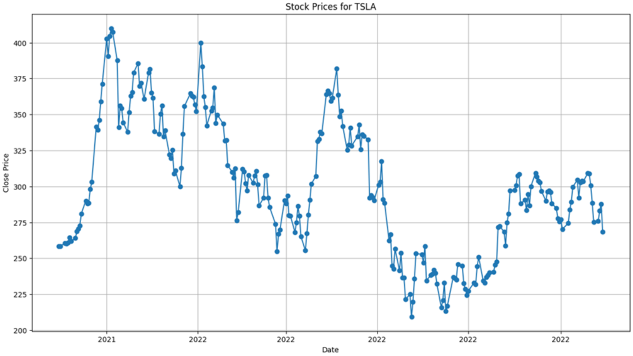

Figure: TSLA.png

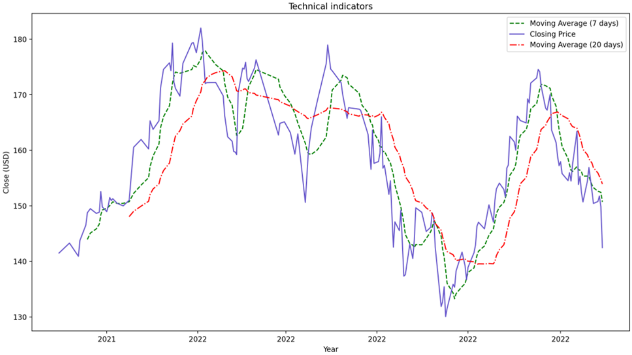

Figure: Visualisation

Model Implementation

Model Initialization

- LSTM Model Architecture:
  - Built an LSTM model with two layers.
  - First layer: 50 neurons, 20
  - Second layer: Final LSTM connected to Dense output layer.
  - Loss function: Mean Squared Error (MSE).
  - Optimizer: Adam.
- Model Justifications:
  - Chose LSTM for handling temporal dependencies.
  - Two-layer setup captures different levels of data abstraction.

We will build a simple LSTM model.

```
def Lstm_model(input_shape):
    model = Sequential()

    #LSTM layer 1
    model.add(LSTM(units=50, return_sequences=True, input_shape=input_shape))
    model.add(Dropout(0.2))

    # LSTM layer 2 (optional, for deeper network)
    model.add(LSTM(units=50))
    model.add(Dropout(0.2))

    # Output layer
    model.add(Dense(units=1, activation='linear'))

    # Compile the model
    model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae'])

    return model
```

Figure: Model Architecture

Baseline Model Results

Model Training Process

- Built with a 50-epoch training process, batch size of 32.

```
model=Lstm_model(input_shape=(X_train.shape[1],1))

history=model.fit(X_train, Y_train, epochs=50, batch_size=32, validation_data=(X_test, Y_test), verbose=1)
```

Figure: Baseline Model with 50 Epochs and Batch Size of 32

Training Results

Training and Validation Results

- Loss and Mean Absolute Error (MAE) for training and validation data.

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - loss: 0.0071 - mae: 0.0666 - val_loss: 0.0063 - val_mae: 0.0640
Epoch 45/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - loss: 0.0068 - mae: 0.0629 - val_loss: 0.0048 - val_mae: 0.0575
Epoch 46/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.0064 - mae: 0.0637 - val_loss: 0.0040 - val_mae: 0.0535
Epoch 47/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - loss: 0.0065 - mae: 0.0643 - val_loss: 0.0051 - val_mae: 0.0592
Epoch 48/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.0094 - mae: 0.0736 - val_loss: 0.0048 - val_mae: 0.0561
Epoch 49/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.0073 - mae: 0.0672 - val_loss: 0.0043 - val_mae: 0.0544
Epoch 50/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.0079 - mae: 0.0719 - val_loss: 0.0055 - val_mae: 0.0611
```

Figure: Training and Validation Results

Inverse Transform Function

Purpose

- Converts predictions back to actual values for meaningful evaluation.

```python
# Create a function to scale it back:
scaler_y = MinMaxScaler(feature_range=(-1, 1))
y_scaled= scaler_y.fit_transform(pd.DataFrame(dataset['Close']))


# Verify
# Use 'scaler_y.inverse_transform'
```

Figure: Inverse Transform Function

Assessment of Impact

- Removed technical indicators to assess their impact.

```
#Train the model using the dataset without the technical indicator and see the performance

dataset_1=dataset.iloc[:, 0:14]
```

Figure: Dataset without Technical Indicators

Retraining without Indicators

Performance Impact

- Retrained model without indicators showed slight performance degradation.

```
model=Lstm_model(input_shape=(X_train.shape[1],1))

history=model.fit(X_train, Y_train, epochs=50, batch_size=32, validation_data=(X_test, Y_test), verbose=1)
```

```
Epoch 45/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - loss: 0.0141 - mae: 0.0943 - val_loss: 0.0108 - val_mae: 0.0832
Epoch 46/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0126 - mae: 0.0872 - val_loss: 0.0096 - val_mae: 0.0751
Epoch 47/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0130 - mae: 0.0896 - val_loss: 0.0090 - val_mae: 0.0747
Epoch 48/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0131 - mae: 0.0856 - val_loss: 0.0090 - val_mae: 0.0740
Epoch 49/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0113 - mae: 0.0864 - val_loss: 0.0115 - val_mae: 0.0846
Epoch 50/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0145 - mae: 0.0949 - val_loss: 0.0092 - val_mae: 0.0793
```

Figure: Model Performance without Technical Indicators

Hyperparameter Tuning Results

Optimization with GridSearchCV

- Used GridSearchCV to optimize hyperparameters (epochs and batch size doubled).
- Optimal parameters: Epochs=100, Batch size=64, Dropout rate=0.4.

```
Best parameters found:  {'batch_size': 64, 'epochs': 100, 'optimizer': 'rmsprop'}
Best RMSE found:  0.2734513979706957
```

Figure: Performance with Optimal Parameters

Create Visual Function

- Created visualization function for consistent results plotting.
- Avoid repeated jargons of code, and more convenient in plotting results.
- Performance metrics are calculated based on the real price (not scaled price).

```python
def create_visual():
    predicted_prices=scaler_y.inverse_transform(model.predict(X_test))
    actual_prices=scaler_y.inverse_transform(pd.DataFrame(Y_test))

    # Plot the results
    import matplotlib.pyplot as plt

    plt.figure(figsize=(14, 5))
    plt.plot(datetime_test, actual_prices, color='red', label='Actual Stock Price')
    plt.plot(datetime_test, predicted_prices, color='blue', label='Predicted Stock Price')
    plt.title('Stock Price Prediction')
    plt.xlabel('Time')
    plt.ylabel('Stock Price')
    plt.legend()
    plt.show()

    print('MAE: ')
    print(mean_absolute_error(predicted_prices, actual_prices))
```

Figure: Create visual function

AAPL Performance

AAPL Results

- Second best performance among stocks with a MAE of $1.576.

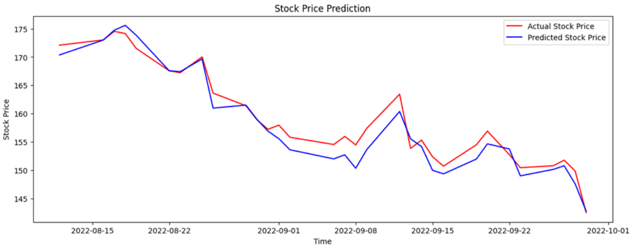

Figure: AAPL Graph

```
MAE:
1.5756609516759073
```

Figure: AAPL MAE

TSLA Results

- Best performance among stocks with a MAE of $1.072.

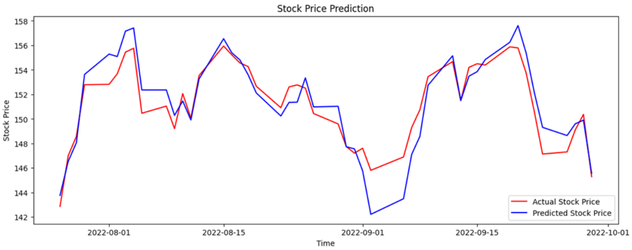

Figure: TSLA: Graph

```
MAE:
1.0718973571029744
```

Figure: TSLA: MAE

GOOG Performance

GOOG Results

- Higher MAE of $2.2302, indicating more variability in data and model performance.

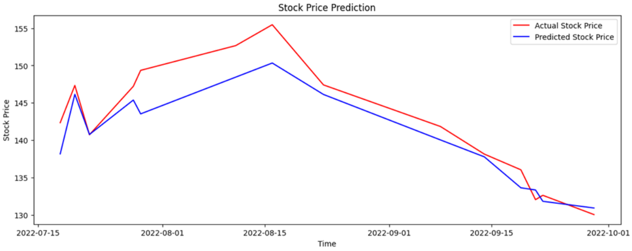

Figure: GOOG Graph


MAE:
2.230234992800015

Figure: GOOG: MAE

Key points of Discussion

- Improved performance due to enriched feature set and better trend recognition.
- Varying amounts of sentiment and historical stock data for each company.
- More sentiment data for popular stocks like Tesla led to better model performance.
- Higher data quality for popular stocks improved model robustness.
- Smaller datasets led to potential overfitting and reduced generalization.

Conclusion & Discussion

Achieved Objectives

- Successfully applied NLP for sentiment analysis.
- Built DL models for stock price prediction.
- Evaluated models using MAE and other metrics.

Conclusion & Discussion

LSTM Model

- Effective for time-series forecasting but it is hard to interpret (Blackbox).

Objective 1: Sentiment Analysis

- Applied VADER for sentiment analysis, linking tweet sentiment with stock price movements.

Objective 2: Model Improvement

- Enhanced model via hyperparameter tuning and incorporation of technical indicators.

Conclusion & Discussion

Objective 3: Model Evaluation

- Created visualizations for model evaluation; MAE used as primary performance metric.
- TSLA: Achieved highest accuracy, demonstrating sensitivity to data quality.

Real-World Application

- Model serves as an academic tool due to stock market volatility.
- Not recommended for actual financial predictions.

Future Recommendations

Enhanced Data Usage

- Include previous days' data to leverage LSTM's long-term memory capabilities.
- For example, we use past 3 days data to predict next day closing price.

Model Complexity

- Consider adding more layers and indicators to capture complex market dynamics.

Expanded Indicators

- Incorporate macroeconomic indicators (e.g. GDP, inflation) for broader understanding of the market.

Project Demonstration

Thank you!