Predicting Investors' Fear and Greed Sentiment Using Bitcoin News

Bita Tarfiee- bitta693
Course Code: 732A81

Abstract

News influences investors as they make investment decisions. This study examines how news affects investor sentiment by predicting investors' fear and greed using Bitcoin news. We compared three models: Logistic regression, RNN, and LSTM, evaluating them based on precision, recall, F1-score, and accuracy. The LSTM model achieved the highest accuracy of 0.88, followed by Logistic Regression with 0.87 and RNN with 0.85. The LSTM model also provided the best balance between precision and recall, making it the most reliable for sentiment prediction in Bitcoin.

1 Introduction

Bitcoin is a decentralized cryptocurrency. Its transactions are recorded on a public distributed ledger known as a blockchain. Public sentiment and global news significantly influence Bitcoin's value. The market experiences substantial fluctuations driven by worldwide events and existing sentiments (Nair et al., 2022). Social media has become a primary channel for generating and disseminating public opinion. On the other hand, news articles are essential in influencing investors' judgment in the crypto market (Hai-Yuan, 2016). News often expresses investors' sentiments about the market, impacting their investment decisions. When news about Bitcoin comes out, its prices typically respond accordingly. For instance, Bitcoin prices tend to increase when the news is positive, while negative news generally leads to declining prices.

This study aims to predict investors' fear and greed by analyzing Bitcoin news, which can help forecast future market price movements and changes. Our analysis can be used for price prediction, similar to studies that have demonstrated the ability to predict cryptocurrency prices, particularly Bitcoin, based on sentiment analysis (Nair et al., 2022; Gaies et al., 2023).

To achieve this, we used deep learning models, including Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks. Our dataset combines news data and Fear and Greed Index values, enabling a robust analysis of sentiment.

2 Theory

We focus on opinion mining for the classification task. In this context, the main objective of classification is sentiment analysis, which has applications in areas such as blog analysis, web texts, and general elections (Patel and Tiwari, 2019). Our specific focus is on predicting investors' sentiment using Bitcoin news. To achieve this, we trained different models using various machine learning approaches, including logistic regression and deep learning classification techniques, RNN and LSTM.

2.1 Logistics regression

Logistics regression is widely used in machine learning applications. This model takes a vector of variables, calculates coefficients or weights for each input variable, and then predicts the class of a given input as a word vector. The detailed mathematics of logistic regression is shown below:

$$logit(S) = b_0 + b_1 M_1 + b_2 M_2 + b_3 M_3 + \ldots + b_k M_k \quad (1)$$

Where S is the probability of the presence of the feature of interest, $M_1, M_2, \ldots, M_k$ are the predictor values, and $b_0, b_1, \ldots, b_k$ are the coefficients of the model.

Assumptions of logistic regressions are: A linear relationship between the dependent and independent variables do not exist; the dependent variable must be dichotomous, the independent variable must be linearly related, neither normally distributed nor of equal variance within a group, and the groups must be mutually exclusive (Prabhat and Khullar, 2017).

2.2 Recurrent Neural Network (RNN)

RNNs are a deep learning approach for sentiment analysis. They generate output based on prior computations by utilizing sequential data. RNNs are highly effective for sentiment analysis because they use memory cells that capture information from long sequences (Patel and Tiwari, 2019).

Traditional neural networks, which rely on independent inputs, are not suitable for specific tasks in Natural Language Processing, such as word prediction in a sentence. In contrast, RNNs can handle these tasks because they generate output based on previous computations using sequential information. This allows them to effectively capture the temporal dependencies in text data.

The basic formula for RNNs is shown in the following equation:

$$a_t = f(h_{t-1}, x_t) \tag{2}$$

Where a_t represents the output from the previous node, the activation function f is a tanh function, and x_t denotes the input sequences $(x_0, x_1, x_2, \ldots, x_t)$. RNNs represent a deep tree structure that can grasp the semantics of texts. However, this process is time-consuming (Patel and Tiwari, 2019).

Theoretically, the RNN language model could cover the time order structure of the whole text and deal with the long-term dependency problem. In practice, however, RNNs cannot learn this knowledge effectively. When the interval between the relevant information in the text and the current location to be predicted becomes large, RNNs tend to learn recent words better than earlier ones, leading to the vanishing gradient problem. To address this issue, RNNs use LSTM networks, which have proven effective in various applications and show great promise in language modeling (Li and Qian, 2016).

2.3 Long Short-Term Memory (LSTM)

LSTM networks are designed to address the issue of long-term dependencies in sequential data; in practice, retaining long-term information is their default behavior. Currently, LSTM networks are the most widely used type. They replace the RNN node in the hidden layer with an LSTM cell specifically designed to store text history information. LSTM uses three gates—input, forget, and output gates—to control the use and updating of text history information. These memory cells and gates enable LSTM to read, save, and update long-term historical details effectively. An LSTM network is the same as standard RNN, except that the summation units in the hidden layer are replaced by memory blocks.

The cell state in an LSTM is vital. LSTM cells manage to add or remove information via 'gates,' enabling selective information processing. First, the forget gates decide which information in the cell state should be discarded. Next, the input gates determine which new information should be stored in the cell state. The third step involves updating the old cell state by using data from both the input and forget gates to compute the new cell state value. Finally, the output gates establish the output value based on the current cell state (Li and Qian, 2016).

The LSTM calculation process is discussed as follows: Calculate the values of the forget gate and input gate, update the state of the LSTM cell, calculate the value of output gates, and update the output of the whole cell. The detailed formula is shown below:

Input Gates:

$$a_t^i = \sum_{i=1}^{I} w_{il} x_i^t + \sum_{h=1}^{H} w_{hl} b_h^{t-1} + \sum_{c=1}^{C} w_{cl} s_c^{t-1} \tag{3}$$

$$b_l^t = f(a_l^t) \tag{4}$$

Forget Gates:

$$a_\phi^t = \sum_{i=1}^{I} w_{i\phi} x_i^t + \sum_{h=1}^{H} w_{h\phi} b_h^{t-1} + \sum_{c=1}^{C} w_{c\phi} S_c^{t-1} \tag{5}$$

$$b_\phi^t = f(a_\phi^t) \tag{6}$$

Cells:

$$a_c^t = \sum_{i=1}^{I} w_{ic} x_i^t + \sum_{h=1}^{H} w_{hc} b_h^{t-1} \tag{7}$$

$$s_c^t = b_\phi^t s_c^{t-1} + b_l^t g(a_c^t) \tag{8}$$

Output Gates:

$$a_l^t = \sum_{i=1}^{I} w_{lw} x_i^t + \sum_{h=1}^{H} w_{hw} b_h^{t-1} + \sum_{c=1}^{C} w_{cw} s_c^t \tag{9}$$

$$b_w^t = f(a_w^t) \tag{10}$$

Cell Outputs:

$$b_c^t = b_w^t h(s_c^t) \tag{11}$$

Where $g(z)$ is the sigmoid function, and $h(z)$ is the tanh function.

3 Data

In this study, we created our dataset by using two datasets from Kaggle, "Bitcoin & Fear and Greed" (Bhatti, 2023) and "Bitcoin Price Prediction" (Azeem, 2024). The first dataset includes daily closing prices of Bitcoin, daily volumes of Bitcoin, and the Fear and Greed Index values for the overall crypto market. The second dataset was collected from a popular Twitter page (BTCTN, 2024), BTCTN, which is the most well-known Bitcoin news page on Twitter. It contains news related to each specific date.

The Fear and Greed Index is designed to analyze emotions and sentiments from various sources and condense them into a single number that represents the sentiment in the Bitcoin and larger cryptocurrency markets. The index ranges from 0 to 100, where a value of 0 signifies "Extreme Fear" and a value of 100 signifies "Extreme Greed" (Alternative.me, 2024).

- **Extreme Fear**: This could indicate that investors are overly worried.
- **Extreme Greed**: This indicates that investors are becoming too greedy.

Figure 1: The Fear and Greed Index provides a single number representing investors' sentiment, ranging from 0 (Extreme Fear) to 100 (Extreme Greed) (Alternative.me, 2024).

To construct our dataset, we merged the two datasets. We used the fear and greed index and label from the first dataset and the news data from the second dataset, aligning them by their corresponding dates. The final dataset comprised 2360 rows. The Bitcoin & Fear and Greed dataset's value classification column contains five classes including fear, extreme fear, greed, extreme greed, and neutral. For simplicity, we reduced the number of classes by mapping the data into two categories, greed (which includes greed and extreme greed) and fear (which provides for fear and extreme fear), and eliminated the neutral class. Figure 1 represented the Fear and Greed Index in alternative.

4 Method

4.1 Data-preprocessing

This section describes the steps taken to preprocess the data for our sentiment analysis model.

Mapping Sentiment Classes

The "Bitcoin Fear and Greed" dataset contains sentiment classes classified into five categories, fear, extreme fear, greed, neutral, and extreme greed. For simplicity, we reduced the number of classes by mapping them into two categories, Greed (1): Includes both greed and extreme greed. Fear (0): Includes both fear and extreme fear. We also dropped the neutral class.

Data Cleaning

We performed some preprocessing steps as follows:

Fill missing values: Convert text data to a string and fill in missing values.

Remove duplicates: Drop duplicate entries to avoid redundancy.

Text Normalization

Stop words removal: Removed common stop words.

Lemmatization: Reduced words to their base form using the WordNet lemmatizer.

Expand contractions: Mapped contractions to their expanded forms for consistency.

Remove HTML: Remove any HTML tags using BeautifulSoup.

Remove emojis and punctuation: Removed emojis and punctuation.

Convert to lowercase: Converted all text to lowercase for uniformity.

Remove URLs: Remove any URLs from the text.

Finally, our dataset contains 2360 rows. Figure 2 shows the distribution of fear and greed labels in the dataset.

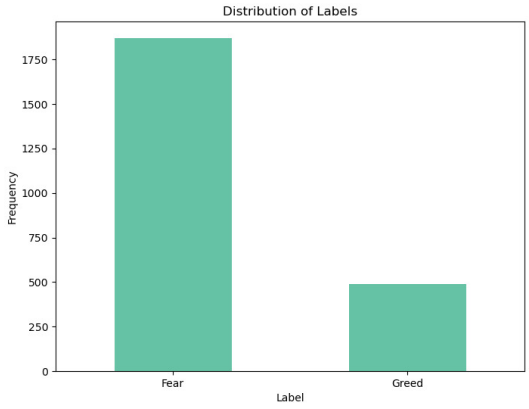

Figure 2: Distribution of fear and greed labels in the dataset.

Splitting data

After pre-processing the data, we split the dataset into training with 80% and testing sets 20%.

Addressing Class Imbalance

We computed class weights on our train dataset to address the class imbalance. These weights will be used in our models to give more importance to the minority class and help the model learn effectively from imbalanced data.

4.2 Models

4.2.1 Logistic Regression

We created a pipeline with a count vectorizer for text feature extraction and a logistic regression classifier. The count vectorizer transforms the text data into a matrix of token counts, which is then used as input for the logistic regression model. The trained model was used to make predictions on the test data. The performance of the model was evaluated using accuracy and classification report metrics.

4.2.2 Recurrent Neural Network (RNN)

This section describes the method used for training and evaluating an RNN model.

Text Vectorization

Before training the RNN and LSTM model, the text data was vectorized:

- **TF-IDF Vectorization**: Converts the text data into TF-IDF feature vectors.
- **Count Vectorization**: Converts the text data into count-based feature vectors.
- **Tokenization**: Converts text data into sequences of tokens, which are then padded to ensure uniform sequence length.

RNN Architecture

Series of RNN layers to process the sequential text.

- **Embedding Layer**: Converts input tokens into dense vectors of fixed size.
- **Batch Normalization**: Normalizes the output of the previous layer.
- **Dropout Layers**: Randomly sets input units to 0 with a frequency of 0.3 and 0.5 during training to prevent overfitting.
- **Convolutional Layer**: Applies convolution operation to extract local features.
- **Max Pooling Layer**: Reduces the dimensionality of the feature maps.
- **Bidirectional RNN Layer**: Processes the input sequence in both forward and backward directions.

Table 1 shows the detailed layer of our RNN model.

Layer (type)
input_layer_3 (InputLayer)
embedding_3 (Embedding)
batch_normalization_3 (BatchNormalization)
dropout_9 (Dropout)
conv1d_3 (Conv1D)
dropout_10 (Dropout)
max_pooling1d_3 (MaxPooling1D)
bidirectional_3 (Bidirectional)
simple_rnn_1 (SimpleRNN)
dropout_11 (Dropout)
dense_3 (Dense)

Table 1: RNN Model Layers

We trained our RNN model with the following parameters, 120 epochs, a learning rate of 0.000007, an embedding dimension of 64, and binary cross-entropy as the loss function.

4.2.3 Long Short-Term Memory (LSTM) Model

This section describes the method used for training and evaluating an LSTM model. The text data was vectorized the same as described in the RNN model.

Model Architecture

Series of LSTM layers to process the sequential text.

- **Embedding Layer**: Converts input tokens into dense vectors of fixed size.
- **Batch Normalization**: Normalizes the output of the previous layer.
- **Dropout Layers**: Randomly sets input units to 0 with a frequency of 0.3 and 0.5 during training to prevent overfitting.
- **Convolutional Layer**: Applies convolution operation to extract local features.
- **Max Pooling Layer**: Reduces the dimensionality of the feature maps.
- **Bidirectional LSTM Layer**: Processes the input sequence in both forward and backward directions.
- **LSTM Layer**: Processes the sequence data.
- **Dense Layer**: Fully connected layer with a sigmoid activation function to produce the output.

Layer (type)
input_layer_1 (InputLayer)
embedding_1 (Embedding)
batch_normalization_1 (BatchNormalization)
dropout_3 (Dropout)
conv1d_1 (Conv1D)
dropout_4 (Dropout)
max_pooling1d_1 (MaxPooling1D)
bidirectional_1 (Bidirectional)
lstm_3 (LSTM)
dropout_5 (Dropout)
dense_1 (Dense)

Table 2: Summary of the LSTM model layers

Table 2 shows the detailed layer of our LSTM model. We trained our LSTM model with the following parameters, 140 epochs, a learning rate of 0.000007, an embedding dimension of 64, and binary cross-entropy as the loss function.

We split the training set for validation purposes to evaluate the performance of both the RNN and LSTM models. The data was split into 80% for training and 20% for validation.

5 Result

Logistic Regression

Table 3 shows the classification report for the logistic regression model. Figure 3 represents the Confusion matrix for the logistic regression model.

Class	Precision	Recall	F1-Score
0 (Fear)	0.90	0.94	0.92
1 (Greed)	0.75	0.63	0.68
Accuracy		0.87	

Table 3: Classification Report for Logistic Regression Model

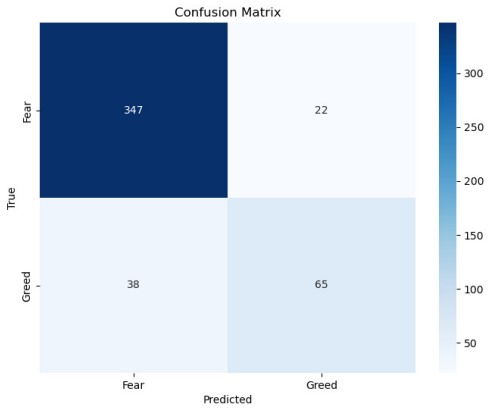

Figure 3: Confusion matrix for logistic regression model showing the performance in predicting 'Fear' and 'Greed' classes.

RNN Model

Table 4 shows the classification report for the RNN model. Figure 4 represents the Confusion matrix for the RNN model.

Class	Precision	Recall	F1-Score
0 (Fear)	0.95	0.86	0.90
1 (Greed)	0.62	0.84	0.72
Accuracy		0.85	

Table 4: Classification Report for RNN Model

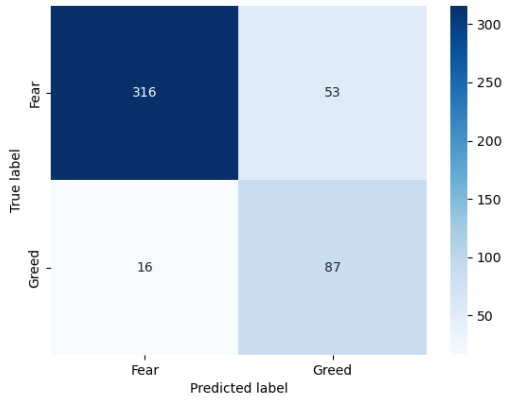

Figure 4: Confusion matrix for RNN model showing the performance in predicting 'Fear' and 'Greed' classes.

LSTM Model

Table 5 shows the classification report for LSTM model.

Class	Precision	Recall	F1-Score
0 (Fear)	0.93	0.91	0.92
1 (Greed)	0.70	0.77	0.73
Accuracy	0.88		

Table 5: Classification Report for LSTM Model

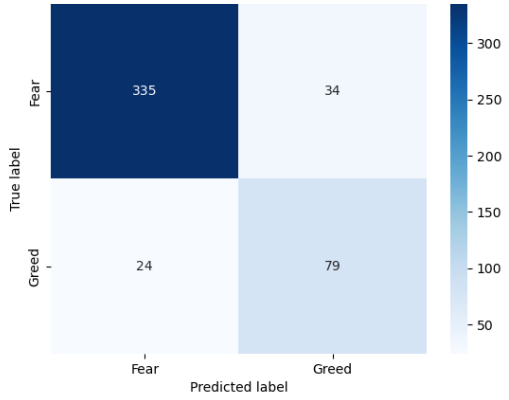

Figure 5: Confusion matrix for LSTM model showing the performance in predicting 'Fear' and 'Greed' classes

6 Discussion

In this study, we used news data to compare three different models—Logistic Regression, RNN, and LSTM- to predict investors' fear and greed Sentiment in Bitcoin. Each model was evaluated based on precision, recall, F1-score, and overall accuracy. The classification reports for each model are summarized in tables 3, 4, 5.

The LSTM model achieved the highest accuracy of 0.88, followed by the logistic regression model with 0.87 and the RNN model with 0.85. This indicates that all three models perform well, with the LSTM model slightly performing better than the others in terms of overall accuracy.

For the Fear class (0), all three models show high precision and recall, indicating that they correctly identify fear in the news. The RNN model achieved the highest precision of 0.95 but a slightly lower recall of 0.86, while the LSTM model showed a balanced precision of 0.93 and a recall of 0.91. There is more variability in the Greed class (1). The Logistic Regression model has the lowest recall performance, 0.63, indicating it misses more Greed instances compared to the other models. The LSTM model provides the best balance, with a precision of 0.70 and a recall of 0.77.

7 Limitation

Our analysis used approximately 2,600 data points, which may need to be improved, particularly for deep learning methods. Increasing the dataset size could lead to better results. Additionally, hyperparameter tuning was conducted manually; employing techniques such as grid search could enhance the performance of the deep learning models.

8 Conclusion

Overall, while all three models have their strengths, the LSTM model provides the best balance in terms of precision, recall, and F1 score across both classes, even if the greed distribution in our dataset is low. This makes it a more reliable choice for predicting investor sentiment in Bitcoin using news data.

References

Alternative.me. 2024. Fear and greed index. https://alternative.me/crypto/fear-and-greed-index/. [Accessed: June 8, 2024].

Muhammed Abdul Azeem. 2024. Bitcoin price prediction. https://www.kaggle.com/datasets/muhammedabdulazeem/bitcoin-price-prediction/data. [Accessed: June 8, 2024].

Adil Bhatti. 2023. Bitcoin amp; fear and greed.

BTCTN. 2024. Btctn twitter page. https://x.com/BTCTN. [Accessed: June 8, 2024].

Brahim Gaies, Mohamed Sahbi Nakhli, Jean-Michel Sahut, and Denis Schweizer. 2023. Interactions between investors' fear and greed sentiment and bitcoin prices. *The North American Journal of Economics and Finance*, 67:101924.

YIN Hai-Yuan. 2016. A study on effect of media reports on investor sentiment: evidence from china's stock market. *Journal of Xiamen University (Arts & Social Sciences)*, 2:11.

Dan Li and Jiang Qian. 2016. Text sentiment analysis based on long short-term memory. In *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, pages 471–475. IEEE.

Karthik Nair, Arham Pawle, Aryan Trisal, and Sunantha Krishnan. 2022. Bitcoin price prediction using sentimental analysis-a comparative study of neural network model for price prediction. In *2022 2nd Asian Conference on Innovation in Technology (ASIANCON)*, pages 1–4. IEEE.

Alpna Patel and Arvind Kumar Tiwari. 2019. Sentiment analysis by using recurrent neural network. In *Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE)*.

Anjuman Prabhat and Vikas Khullar. 2017. Sentiment classification on big data using naïve bayes and logistic regression. In *2017 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–5. IEEE.