UNIVERSITY OF HERTFORDSHIRE

School of Physics, Engineering and Computer Science

Final Project Report

Data Science and Analytics Masters Project (7COM1075-0105)

15-05-2023

"STUDYING THE EFFECT OF NEWS HEADLINES ON THE PRICE ACTION OF THE S&P 500 STOCK INDEX."

Name: **Prajwal Karamuttil Augustine**
Student ID: **21025219**
Supervisor: **Ms. Renuga Jayakumar**

ABSTRACT

The stock market serves as a marketplace for investors to buy and sell shares of publicly traded businesses, and it is essential to the economy because it makes it easier for investors to transfer money to businesses (Hirsh, 2021). Sentiment plays an important role in determining the price action of a stock. A positive sentiment towards a stock will result in a positive price action while a negative price action can lead to negative price action. (Chau, Deesomsak, Koutmos, 2016)

In this project, the author aims to create a Sentiment model which can predict the price of an asset from the news headlines fed into it. There are various metrics that current models use to predict stock prices. All these are technical indicators and are short-term price predictors. In the long run, technical indicators are very inefficient. As mentioned above, sentiment is an important factor that needs to be considered to value a stock. In this project, the author compares various sentiment analysis methods to extract numerical scores for news headlines and predict the price of the S&P 500 stock index. The S&P 500 Index consists of 500 leading U.S. publicly traded companies. It is very diverse. It contains companies of different market caps and from different industries. (Kenton, 2022). The author is able to implement an LSTM-based network to predict the price with very high accuracy.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Mr Bente Riegler, the module leader, for providing guidance and support throughout this module.

I would also like to thank my supervisor, Ms Renuga Jayakumar, for their invaluable feedback and advice on my work. Their support has been instrumental in my academic and personal growth, and I am truly grateful for their contributions.

I would like to extend my heartfelt appreciation to the staff at the University of Hertfordshire for their dedication to maintaining the labs and providing me with a comfortable study space throughout the academic year. Their commitment to creating a conducive environment for learning has been instrumental in my academic success.

I would also like to express my gratitude to my friends who have been a constant source of encouragement and support throughout my academic journey. Their enthusiasm for trading sparked my interest and has been instrumental in my growth as a trader.

Finally, I want to thank my parents for their unwavering support both emotional and financially. Their constant encouragement and belief in me have been invaluable, and I would not have achieved this without them.

DECLARATION

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Data Science and Analytics at the University of Hertfordshire (UH).

It is my own work except where indicated in the report.

I did not use human participants in my MSc Project.

I hereby give permission for the report to be made available on the university website provided the source is acknowledged.

Table of Contents

Table of Figures

INTRODUCTION

The stock market is a complex financial system of buyers and sellers who trade equities of various publicly available companies that will determine their value. It is a fascinating invention that continues to play a crucial role in the global economy. They have significantly affected many fields, including business, education, employment, and technology, and as a result, the economy (Shah et al., 2019). The value of a company is influenced by many underlying factors such as fundamentals, technical, and market sentiment (Harper, 2022). Market sentiment is a defining factor for price action. The collective views of other traders impact the decisions made by traders (Checkley et al., 2017). News is one of the major contributors to market sentiment that has a substantial impact on investment decisions and market movements. Positive news like a merger or an acquisition has a higher probability of pushing the prices into an uptrend and negative news like scandals or regulatory investigations results in the latter. Markets depend on fresh information, which is continually being streamed (Schumaker et al., 2012).

Algorithmic trading in the context of financial markets refers to the automation of one or more trading-related processes. These include data analysis, trade signals and trade execution (Nuti et al., 2011). There are three major analytical methods used by most algorithmic trading engines which can be categorised into fundamental analysis, technical analysis, and blending analysis. They also use evolutionary computation (EC) techniques like genetic algorithm-based (GA) and genetic programming-based (GP) techniques. The analytical methods perform better in the uptrend while the evolutionary computation techniques work better in the downtrend (Hu et al., 2015). The EC techniques can be improved by factoring in market sentiment with the help of Natural Language Processing. This is made possible with the availability of Big Data and advancements in sentiment analysis algorithms.

Before the digital era, it was hard to take advantage of the news to successfully trade stocks. But with the dawn of the internet and advanced database warehouses, we can utilize the vast availability of 'Big Data' to gain insights. 'Big Data' refers to Information Assets characterised by High Volume, Velocity and Diversity that need Specific Technology and Analytical Techniques for Transformation into Value (De Mauro et al., 2015). Archives of news headlines can be filtered and used to train models to make accurate and sentiment-oriented price predictions. For many years, stock price forecasting has been a hot issue in finance. Early studies on this subject asserted that stock price fluctuations do not adhere to

any patterns or trends and that the past cannot be utilised to forecast the future. Later research has revealed the opposite, and several methods, including technical analysis, fundamental analysis, and quantitative analysis, have been suggested to forecast stock prices. By examining the sentiment of news stories, social media posts, and other information sources, sentiment analysis has recently become a potent technique for predicting stock values.

Sentiment analysis is a highly studied area in natural language processing with applications in social media, e-commerce, and politics, among other areas. Sentiment analysis is the computational study of opinions and feeling regarding an entity. It is a heavily researched field in text mining (Medhat et al., 2014). Sentiment analysis combined with evolutionary computation techniques can produce powerful models to analyse data. Stock market data and sentiment scores from news headlines can be used to predict stock prices. According to Bordino et al (2012), sentiment analysis is excellent at forecasting stock prices and financial trends based on social media data. Additionally, lexicon-based methods, machine learning, and deep learning techniques have all been proven to be very effective in reading sentiment. (Pang & Lee, 2008). Overall, sentiment analysis has shown to be a valuable method for deciphering public sentiment and forecasting trends across a range of industries.

PROJECT SPECIFICATION

This project aims to predict the price action of the S&P 500 stock index by utilizing the data collected from various sources between 17th December 2017 to 18 July 2020. Price action refers to the movement of an asset over time, as well as the analysis of these movements to identify various patterns and trends or at the very least, the direction. The data will then be used to obtain sentiment scores for the financial news headlines using various researched methods. The scores will then be combined with the previous and current closing stock prices. A Neural network model will be built to predict the current closing stock price. The model will be measured for effectiveness using appropriate evaluation metrics such as mean squared error, root mean squared error, and coefficient of determination (R-squared). To confirm the model's performance on untested data, the dataset will also be divided into training and testing sets. The project will then be clearly documented, outlining the process for data collection, data preprocessing, sentiment analysis, building and implementing neural networks, and evaluation. This project's overall objective is to show how sentiment analysis can be used to predict stock prices to develop a useful tool for stock market and financial industry investors.

PROJECT AIMS AND OBJECTIVES

RESEARCH QUESTION/TITLE

Predicting the price action of the S&P 500 stock index using sentiment analysis.

RESEARCH AIMS

The research aim of this project is to investigate the effectiveness of utilizing sentiment analysis in predicting the price action of the S&P 500 stock index and to build a neural network model that can accurately predict the current closing stock price.

PROJECT OBJECTIVES

1. To build a dataset of news headlines and their respective previous and daily closing prices of the S&P 500 stock index.
2. To generate event tuples of the news headlines using a researched knowledge graph method. The event tuples include source, target, and relation.
3. To get sentiment scores on the event tuples using ‘Sentiment Intensity Analyzer’ and Bidirectional Encoder Representations from Transformers (BERT).
4. To predict the price of the S&P 500 stock index using the sentiment scores and LSTM.

LITERATURE REVIEW

The stock market is a complex system which is influenced by a range of factors which are both internal and global like a country's economics, company performance and global events. As discussed above, stock market news has an immediate impact on stock prices. For example, press releases from the Federal Reserve, and the central bank of America which is responsible for setting the monetary policy have a significant impact on the stock market especially when it comes to interest rates. In December 2015, the FED announced that it was raising the interest rates for the first time in nearly a decade. This had an immediate impact on the stock market, with the S&P 500 falling nearly 1.5% in the days following the announcement (CNBC, 2015). News about the economy, especially data such as GDP, unemployment rates and inflation has had an impact on stock prices. For example, The U.S. Bureau of Labor Statistics announced that the unemployment rate had risen to 14.7% in April 2020. It was the highest level since the Great Depression. But the Dow Jones Industrial Average climbed more than 455 points which didn't make any sense for a beginner investor (CNN, 2020). In July 2021, the Department of Commerce reported a 6.5% growth in the annualized rate in the second quarter of the year. This news had a positive impact on the stock market, with S&P 500 gaining 1% in a single day. (CNBC 2021). News about Central Banks and the economy in general can have a significant impact on the stock market with even small news moving stock prices.

Stock investment is a high-risk activity. Inexperienced investors can lose their investments due to poor investment decisions and a lack of professional knowledge. To improve decision-making and profitability, investors use fundamental and technical analysis. Fundamental analysis involves studying a company's financial condition, management, economics, and products to predict its stock value. Technical analysis uses historical stock price trends to forecast future price trends. Fundamental analysis is more suitable for long-term investments (Chen et al., 2016). Technical Analysis is a method to forecast prices with the help of current price, volume, charts, and other technical indicators like moving averages. The underlying factor behind technical analysis is that stock prices are moved by changing sentiment in investors. These changes can be visually seen in the form of trends and chart patterns. Approximately 90% of traders use technical analysis to trade. It is criticised for being very subjective. Different traders read and interpret charts differently. A

previously unknown trend, like a press release of a scandal, can result in failure. For Example, the Adani Group of stocks which was fundamentally a good investment and was technically on a positive trend had a major crash after the release of the Hindenburg report on January 25th. The combined market capitalization of all 10 Adani stocks has dropped by 60%. The report had accused the Adani Group, a multinational conglomerate based in India, of "dubious" accounting practices, inflating the value of its airport business, and evading taxes through offshore entities. This proves that no matter how strong the technical and fundamental aspect of a stock is all it takes is one piece of news to change the market sentiment.

Machine learning plays an important role in stock price prediction. Initially, classical regression methods were used to predict prices but since stock data is categorized as non-stationary time series data, non-linear machine learning models have been used. Two widely used models were Artificial Neural Networks and Support Vector Machines (Patel et al., 2015). News events can cause sudden changes that technical or fundamental analysis alone cannot foresee. As a result, adding sentiment analysis to regression models can help them become more accurate by better accounting for the effect of news events on stock prices. Sentiment analysis can identify the tone of news stories and headlines, which can then be added as a feature to regression models to increase their ability to predict future events. Even with a highly trained machine learning model, it can still predict incorrectly when market sentiment changes. The trends and patterns the model learns won't match with the real-time trend.

Sentiment analysis scores combined with a predictor are the best approach we can follow to predict stock prices. Sentiment analysis has received a lot of attention recently in the world of finance, especially in the stock market. This is because social media platforms and news articles, which are frequently excellent sources of information, can have a big effect on the stock market. In a study by Tetlock et al. (2008), the researchers found that the sentiment expressed in financial news articles was a significant predictor of stock price. They used a dictionary-based method to analyse the sentiment in financial news articles and discovered that future stock returns were higher the more positive the sentiment expressed in the articles. Positive sentiment, on the other hand, was linked to higher future stock returns. A similar methodology was applied in a different study by Bollen et al. (2011) to examine sentiment in financial news articles and social media platforms. With a prediction

accuracy of 87.6%, they discovered that sentiment analysis of tweets was a reliable indicator of the Dow Jones Industrial Average. The study's authors came to the conclusion that social media sentiment analysis might be a useful tool for forecasting changes in the stock market.

The paper by Das and Chen (2007) describes a methodology for extracting sentiment from online message board postings to build a sentiment index that can be used to predict stock prices. The authors divide messages into three categories: bullish (optimistic), bearish (pessimistic), and neutral using five algorithms. The algorithms combine more conventional statistical techniques with language features like parts of speech tagging. The algorithms are tuned on a training corpus, which is a small subset of pre-classified messages used for training the algorithms, before beginning classification. The pre-classified data set is used by the algorithms to "learn" sentiment classification rules that are then used outside of samples. Before a message is finally classified, a simple majority across the five algorithms is necessary; otherwise, it is discarded. The signal-to-noise ratio for sentiment extraction is improved by this voting method. The authors download messages from the Internet using a Web-scraper programme, which they then feed into the five classification algorithms to classify as buy, sell, or null types. The classification algorithms are supported by three additional databases: a pre-classified training corpus, an electronic English "dictionary," and a "lexicon" of finance terms. The five algorithms access these databases to determine each message's three-way classification. The authors demonstrate that their sentiment index can be used to forecast stock prices and that it has a strong correlation with stock returns. Additionally, sentiment analysis has been used to assess how macroeconomic news has affected the stock market.

In a study by Kamstra et al. (2003), the researchers used a sentiment analysis method to examine how macroeconomic news affected the stock market. They discovered that the stock market was significantly impacted by the sentiment expressed in macroeconomic news releases. Stock prices rose in response to good news while fell in response to bad news. The researchers came to the conclusion that sentiment analysis, as opposed to conventional financial models, could be used to predict stock prices more accurately from macroeconomic news releases.

Sentiment analysis though powerful has some limitations. The issue of subjectivity is one of sentiment analysis's main drawbacks. In order to determine the overall sentiment of a text, sentiment analysis makes the assumption that specific words or phrases have a specific positive or negative connotation. However, based on their viewpoint, experiences, and

biases, what may be favourable to one person may be unfavourable to another. In cases where the sentiment is ambiguous or subtly expressed, this subjectivity can result in sentiment analysis errors (Bollen et al., 2011). Context is a problem that sentiment analysis cannot address. Without taking into account the context of use, sentiment analysis algorithms are made to examine specific words or phrases and ascertain their overall sentiment. For instance, depending on the context, the phrase "not bad" can be interpreted either positively or negatively. It could be positive when used to describe a product, but it could also be negative when used to describe a financial report. Because of this context gap, sentiment analysis results may be inaccurate (Poria et al., 2016).

The availability of data is a problem with sentiment analysis. To make precise predictions, sentiment analysis needs access to a lot of data, especially from news articles and social media posts. Not all this information is helpful for making stock price predictions. For instance, news articles discussing a company's social impact or environmental record may not be as helpful in predicting stock prices as articles focusing on its financial performance. The effectiveness of sentiment analysis can be hindered by the availability of data, which is also impacted by variables like language, location, and time (Ding et al., 2014). The difficulty of predicting the stock market as a complex system is a drawback of sentiment analysis. Economic indicators, political developments, and market trends are just a few of the variables that have an impact on the stock market. These variables can interact in complex and unpredictable ways. Sentiment analysis might reveal how investors feel about a specific business or item, but it might miss more general market trends and forces that affect stock prices. As a result, sentiment analysis may be less successful in the long run at forecasting stock prices (Preis et al., 2013).

There are several ways to mitigate the limitations of sentiment analysis on stock news for price prediction. First off, adding information from sources other than sentiment analysis can increase the precision of price-prediction models. For instance, adding fundamental analysis, technical analysis, and macroeconomic indicators along with sentiment scores can help traders understand the market better and increase the precision of price-prediction models (Ding et al., 2014; Tsai et al., 2016). Including domain-specific knowledge can help sentiment analysis be more accurate. For instance, using lexicons and dictionaries specific to a given field can aid in identifying emotive terms and expressions that are specific to the financial industry (Loughran and McDonald, 2011). In their paper, Li et al (2014) proposed a stock price prediction framework which uses Harvard psychological dictionary and the Loughran–McDonald financial sentiment dictionary to construct a sentiment space. There

are different categories of sentiment dictionaries. Hatzivassiloglou and McKeown is for example a semi-automatic dictionary. The Harvard psychological dictionary and the Loughran–McDonald financial sentiment dictionary are manual directories with 182 and 2 dimensions respectively. They conclude that models which use only positive and negative sentiment don't perform well. The model built based on an instance of dictionary-based sentiment space and price performs better.

The main methods which will be followed in this research are explained below.

Bidirectional Encoder Representations from Transformers (BERT)

BERT is a language representation model introduced by Devlin et al (2018). It is a pre-trained language model that has demonstrated promise in a number of natural language processing (NLP) tasks, such as sentiment analysis, text classification, and question answering. BERT has recently been used in financial analysis to glean insightful information from financial news, social media, and other text-based sources. It follows the steps of pre-training and fine-tuning to tackle a broad set of NLP tasks. BERT scans the entire string of words at once, in contrast to standard NLP models that take a unidirectional approach. Understanding the context of words and sentences is one of BERT's key advantages. As a result of its extensive training in text data, BERT is able to recognise the connections between words and their meanings. Because it can recognise the sentiment and context of financial news articles, social media posts, and other textual sources, BERT is a useful tool for financial analysis.

In their research, Nemes and Kiss (2021) performed sentiment analysis on a dataset of economic headlines using BERT and other techniques. BERT outperformed the Recurrent Neural Network (RNN), which was second in classifying headlines into positive and negative. In their paper, Sousa et al (2019) compared the performance of BERT with three other machine learning models naive Bayes, support vector machines (SVM), and TextCNN. For the naive Bayes and SVM algorithms, the authors used the bag-of-words (bow) and term frequency-inverse document frequency (tfidf) representations, whereas the TextCNN algorithm made use of the fastText average vector of word embeddings. The effectiveness of the algorithms was assessed using a 10-fold cross-validation process. Accuracy, precision, recall, and F1 were the evaluation metrics used. In terms of all evaluation metrics, the authors discovered that BERT performed better than the alternative methods. In Table II, the most impressive results were highlighted in bold. Along with using Random Search with 20 iterations, an RBF kernel, and varying C and gamma on exponential scales of 100

and.1, the authors also modified the SVM's parameters. The results of the paired t-test revealed a significant difference between BERT and TextCNN (with a significance level of 0.05), demonstrating that BERT outperformed TextCNN in the sentiment analysis task. Overall, this comparison shows that BERT outperforms other widely used machine learning algorithms in this task, indicating that it can be a powerful tool for sentiment analysis. The outcomes also emphasise the significance of using appropriate text data representations and refining machine learning algorithm parameters for enhanced performance.

Neural Networks for prediction and feature extraction

Neural Networks have become a popular method for predicting stock prices. In their research, Nemes and Kiss (2021) used a fine-tuned RNN. In a Recurrent Neural Network (RNN), the current states' input is loaded with the output of the previous layers which makes it better at identifying hidden sentiment in the data. Heidein and Parpinelli (2022) trained a Long Short-Term Memory (LSTM) based prediction model that can predict prices up to 35 days in the future. The sentiment analysis was done using Valence Aware Dictionary and Sentiment Reasoner (VADER) which is a rule-based sentiment analysis tool. The sentiment scores along with the historical prices were used to train the LSTM model. The Root-mean-square deviation (RMSE) scores for the model trained with the sentiment scores were significantly better than the model trained with just the historical data. In their paper, Liu et al (2018) proposed a joint learning model which uses a TransE model for representation learning and a Convolutional Neural Network (CNN) for feature extraction. Support Vector Machines (SVM) and LSTM are used to evaluate the strategy. The joint learning method produced a higher accuracy compared to other methods.

In their paper Jangid et al (2018) In place of discrete classes, the authors introduced a modified multichannel convolutional neural network (CNN) for sentiment analysis that can produce sentiment intensity scores in the [-1,1] range. For complex scenarios, like the negation of positive words, rule-based models that use a predefined intensity score of each word to determine the sentiment score of the entire sentence are unsuitable. Deep learning models are therefore better suited for handling situations like these that are complex. The modified CNN model has only one neuron in the output layer with a sigmoid activation function that is trained on a dataset of sentences and their sentiment intensity scores. The input to the model is enhanced word vectors, and the output is the sentiment intensity score in the range of [0,1], which is scaled to [-1,1] before reporting. The choice of word embedding is a hyperparameter, and each channel can have a different word embedding. The

propagation of errors into word vectors is also a hyperparameter that is optimized using Bayesian optimization. The sentiment score is calculated using a modified technique that rescales each word vector according to its relationship with the given target. The model uses enhanced word vectors as input. The aspect model received a weighted average F1 score of 0.69 from the authors, and the sentiment model received an R-squared score of 0.288.

Knowledge Graphs

A method of representation learning between entities and relations in a knowledge base is known as knowledge graph embedding. One of the main applications of knowledge graphs in finance is to identify and analyze the relationships between various financial entities. A knowledge graph, for instance, can be used to show the connections between businesses, stock prices, financial news, and economic indicators. Analysts can gain important insights into market trends and make wise decisions by examining these relationships. The semantic information between the entities and relationships is represented by mapping the entities and relations onto a low-dimensional space. In the method described in the above section, Liu et al (2018) used a TransE model and a CNN to generate knowledge embeddings. Zhong et al (2023) introduced a strategy to process knowledge graphs using semantic matching approaches. This combined with syntax-based and context-based approaches was used to calculate the sentiment. This makes the sentiment more human-like.

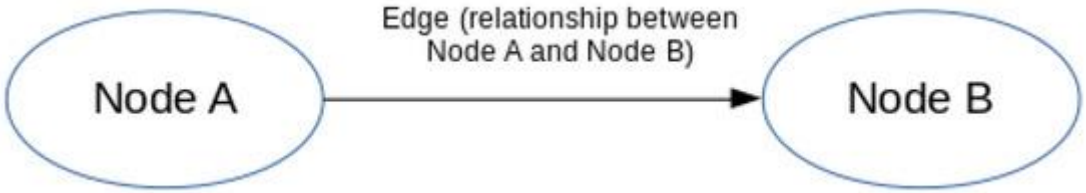

Figure 1: Knowledge graph (Source: Lovera et al., 2021)

In their paper, Guo (2022) proposes a knowledge graph-based sentiment analysis system which has the knowledge graph module, the embedding layer module, and the recurrent convolutional neural network (RCNN). The system was tested with and without the Knowledge Graph embeddings. Knowledge graph embeddings helped boost the performance of the model. Knowledge graphs have proven to be effective in financial analysis in a number of studies. For example, Wang et al. (2018) suggested using knowledge graphs to predict stock prices. In order to depict the connections between

businesses, stock prices, and economic indicators, they created a knowledge graph. They then employed graph-based algorithms to forecast the target company's stock price. Their experimental findings demonstrated that their strategy outperformed a number of industry standards.

Using knowledge graphs the author aims to get the source, target, and relationship from the news headlines. Sentiment analysis targeted at these specific parts of a sentence can improve the accuracy of the sentiment scores.

Consideration of ethical, legal, professional, and social issues

Using sentiment analysis on news headlines can create a complex set of issues. It raises several issues that have to be considered. Legal issues like compliance with SEC regulations, privacy protection, and accountability needs to be addressed. Ethical issues include fairness and non-discrimination, accuracy and transparency, and responsibility for the consequences of using sentiment analysis. Professionally, this can impact the accuracy of prediction and can also create a potential for market manipulation. Socially, it can cause inequality as the influence of sentiment on decision-making can create a divide among traders.

Accuracy and Reliability

This falls under the ethical, legal and professional spectrum. Sentiment analysis can sometimes be very biased which might lead to false predictions. News headlines might not always reflect the reality of the stock market. Market manipulators can seize this opportunity to use fake news as a medium to move stock prices in their favour.

Fairness and non-discrimination

No group or person may be discriminated against through the use of sentiment analysis. All traders and investors must have equal access to news headlines and sentiment analysis findings. There is a risk of insider trading if a certain group got access to news headlines before they were made available to the general public.

Responsibility and accountability

If an Asset management company is using a sentiment-based predictor to invest it must be held accountable for any harm caused to investors or the market. They cannot be selfish with the insights they gain.

Impact on professionals and Institutions

The use of sentiment analysis could decrease the demand for the services of financial analysts and fund managers, who rely on their expertise to make investment decisions. Both job losses and a decline in the standard of investment decision-making could emerge from this.

METHODOLOGY

Choice of method

The approach outlined in the project involves using knowledge graphs, sentiment analysis, and Neural Networks to predict stock prices. A knowledge graph is a graph-based data structure that organises information into a set of entities (nodes) and relationships (edges) between those entities. Financial news nodes can represent things like businesses, people, or financial instruments, while edges can stand in for connections like partnership, investment, or ownership. This method of presenting financial news allows for the capture of intricate dependencies and relationships that may be important for forecasting stock prices. This will help in identifying deeper connections and context in the text. It may stand out compared to other researched methods. The project uses two different sentiment analysis methods to extract sentiment scores from the headlines after creating the knowledge graphs. The first method is based on BERT, a commonly used pre-trained transformer model for tasks involving natural language processing. Because it can capture linguistic context and nuance, which can be crucial for comprehending the sentiment of financial news, BERT is particularly helpful for sentiment analysis. The second method is based on NLTK, a well-known Python natural language processing library. This method extracts sentiment scores from the headlines based on the presence of positive or negative words using a rule-based system. Following that, a long short-term memory (LSTM) based neural network is used to process the stock prices and the sentiment scores that were extracted from the headlines. Recurrent neural networks of the LSTM variety are particularly good at simulating time-series data, such as stock prices. The project aims to capture the impact of financial news on stock prices over time by combining sentiment scores and stock prices in this manner. One of the key advantages of this approach is that it is able to capture complex relationships and dependencies between entities, which may be relevant to predicting stock prices Additionally, the project is able to capture both the presence of positive and negative words as well as the contextual nuances of language by using both BERT and NLTK for sentiment analysis. Finally, the project can take into account the impact of financial news over time by modelling the relationship between sentiment scores and stock prices using LSTM-based neural networks

Tools required

Python 3.7.7 is used for most of this project. It is a very powerful language with a lot of packages useful for sentiment analysis and prediction. Google Colaboratory provides an able platform for executing Python notebooks. Excel is used for cleaning the data frame.

Major Python Packages Required

- re
- pandas
- numpy
- spacy
- sklearn
- keras
- nltk
- torch
- transformers

Minimum System Requirements

x86-64 architecture (AMD 64-bit or INTEL)
16GB RAM
4 CPU cores
4 GB GPU with CUDA cores
200 GB storage
This project uses high RAM and V-RAM.

Data collection and Dataset building

The first step in this project was dataset building. The News headlines datasets were downloaded separately from Kaggle. The data sets were made up of economic news headlines from CNBC, Guardian, and Reuters. The data sets were imported into data frames using the Pandas library in Python. Each data set has a date column. These were converted into a common format using the ‘to_datetime’ function. Once the dates were in a common format, the datasets were concatenated using the pandas ‘concat’ function. Once the news dataset was built, the stock price dataset was imported. The time column of the news dataset was renamed to ‘Date’. The stock price data set and the news dataset were now merged.

First, the Price column in the stock dataset was converted from string to float by removing commas and converting the string to float. A loop iterates over each date in the 'Date' column of the News data frame. For each value of Date, it checks the stock price dataset for the same date. If it exists, it adds the closing price of that date back to the news data frame. If it doesn't exist, it adds the price of the previous day. A similar step is taken to add the closing price of the previous day of the date ending up with two new columns in the news dataset, 'Price' and 'prevclose'.

Cleaning and preprocessing the Dataset

Some of the rows in the Headlines column had unnecessary Line breaks. These line breaks were fixed using the 'TRIM' and 'CLEAN' functions of Excel. The Headlines also had noise in the form of non-alphanumeric characters. The 're' package from Python is used to clean the code. A regular expression with alphanumeric and dollar and pound currency characters is defined. The sub-method of re is used to substitute the noise with a space, the extra white spaces are removed, and the text is converted to lowercase. The function uses regular expressions (regex) to perform text cleaning. Regex is a sequence of characters that define a search pattern. In this case, the pattern '[^a-zA-Z0-9$£]' matches any character that is not an uppercase or lowercase letter, a digit, a dollar sign, or a pound sterling sign. The 're.sub()' function is then used to replace any matches of this pattern with a space. The 'clean_text()' function is applied to the headlines' column of the data frame called. The 'apply()' function is used to apply the 'clean_text()' function to each row of the headlines' column. The cleaned text is then assigned back to the Headlines column of the 'merged_df' data frame. All this is coded into a function which will be called on the headline. Next, the data frame is checked for duplicate headlines. The matched rows are dropped using the 'drop_duplicates' function. Text cleaning is an important step in natural language processing (NLP) and text analysis tasks. It helps to reduce noise in the text data and improve the accuracy of NLP models. The specific cleaning steps used in this code are common and are often used in various NLP tasks such as sentiment analysis, topic modelling, and named entity recognition.

	Headlines	Date	Price	prevclose
0	the guardian view on ryanair s model a union f...	2017-12-17	2690.16	2675.81
1	butchers carve out a niche as uk shoppers opt ...	2017-12-17	2690.16	2675.81
2	this year has been about companies and jobs wi...	2017-12-17	2690.16	2675.81
3	youngest staff to be given uk workplace pensio...	2017-12-17	2690.16	2675.81
4	paul dacre and pay how best to spend the mail ...	2017-12-17	2690.16	2675.81
...	...	...	...	...
53325	world bank calls on creditors to cut poorest n...	2020-07-18	3251.84	3224.73
53326	british airways retires boeing 747 fleet as co...	2020-07-18	3251.84	3224.73
53327	what will changes to england s lockdown rules ...	2020-07-18	3251.84	3224.73
53328	atol protection to be extended to vouchers on ...	2020-07-18	3251.84	3224.73
53329	disney cuts ad spending on facebook amid growi...	2020-07-18	3251.84	3224.73

53330 rows × 4 columns

Figure 2: Final dataset (source: self-generated)

Description of the dataset

The data set has 53330 rows in total and 4 main columns after preprocessing.

- Headlines: Financial news headlines collected from various sources. The data type of the column is an object and will later be converted into a string.
- Date: Date the news headline was released. The data type of the column is datetime64. The date is in the YYYY-MM-DD format.
- Price: Price of the closing price of the S&P 500 stock index for the date. The data type of the column is float. The mean closing price of the index is 2871.
- Prevclose: Price of the closing price of the S&P 500 stock index for the previous day. The data type of the column is float.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53330 entries, 0 to 53329
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Headlines  53330 non-null  object
 1   Date       53330 non-null  datetime64[ns]
 2   Price      53330 non-null  float64
 3   prevclose  53330 non-null  float64
dtypes: datetime64[ns](1), float64(2), object(1)
memory usage: 1.6+ MB
```

Figure 3: Description of the dataset (source: self-generated)

	Price	prevclose
count	53330.000000	53330.000000
mean	2871.109031	2871.170248
std	206.322849	204.872230
min	2237.400000	2237.400000
25%	2730.200000	2730.200000
50%	2856.270000	2856.270000
75%	2995.680000	2993.070000
max	3386.150000	3386.150000

Figure 4: Statistics of 'Price' and 'prevclose' (source: self-generated)

Normalization

'Standard Scaler' is a popular preprocessing technique used in machine learning to standardize the features of a dataset by subtracting the mean and scaling by the standard deviation. Certain machine learning algorithms that are sensitive to the scale of the input features' features can perform better thanks to this transformation, which yields features with zero mean and unit variance.

First, the mean of each feature is computed as:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} (x_i)$$

Then, the standard deviation of each feature is computed as:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Finally, the StandardScaler transformation is applied to each feature as:

$$z = \frac{x - \mu}{\sigma}$$

EXPERIMENT

Building Knowledge graphs on news headlines

The process of creating a knowledge graph is iterative, requiring constant tweaking and improvement. The ability to effectively represent complex information in a way that is useful and meaningful for various applications requires a combination of domain expertise, technical skills, and creativity. Our main goal is to extract two entities which are ‘source’ and ‘target’, and the relationship ‘edge’ between them. Applying sentiment analysis to the source, target, and relation in a knowledge graph can provide valuable insights into the emotions and attitudes expressed in the text data.

Building a knowledge graph involves several steps, including data acquisition, data preprocessing, entity and relationship extraction, and graph construction. In this project, only the first three steps are used. The first two steps were already covered in the data collection and processing steps. It is necessary to extract the entities and relationships from the data. This entails identifying the various entities (such as people, places, and things) as well as the connections among them (such as someone who works at, lives in, or has a product). Natural language processing (NLP) techniques are frequently used in this step to extract and categorise the entities and relationships. The Pandas library is used for creating and modifying the graph data structure, and the ‘spaCy’ library is used for natural language processing to identify the entities and relationships in the text data. The dependencies between words in a sentence are found and parsed.

The script consists of two main functions: ‘get_entities()’ and ‘get_relation()’, as well as an iterative loop that extracts the entities and relations from a dataset of headlines and stores them in a pandas data frame. The entities (i.e., subjects and objects) within the sentence are identified after the sentence has been processed using the ‘spaCy’ library. It accomplishes this by repeatedly going through the sentence's tokens and examining their dependencies on one another. It combines any preceding tokens that are compound modifiers or other modifiers of those entities, specifically those that depend on "subj" or "obj”. The function outputs a list of two strings, the first of which represents the subject entity and the second of which represents the object entity. Similarly, the relationship is identified from ‘get_relation()’. The function gives back a string that symbolises the discovered relationship, or the Knowledge Graph edge. Additionally, it uses the ‘spaCy’ library to process the sentence and determine the relationship (i.e., the edge in the KG) between the sentence's

entities. This is accomplished by using the ‘spaCy’ Matcher class to look for a token pattern that exemplifies the relationship. A root token (the primary verb of the sentence) is followed by an optional preposition, an optional agent, and an optional adjective in the specific pattern used for this function. These two functions are called on each headline in the dataset of headlines by the loop that iterates through them, and the resulting entities and relationships are then stored in a pandas data frame. In the data frame, it specifically adds three columns: "source" for the subject entity, "target" for the object entity, and "edge" for the relationship between the two.

Overall, this script offers an efficient method for automatically identifying entities and relationships in text data and producing a Knowledge Graph to represent those findings. Applications like text classification, information retrieval, and natural language understanding can all benefit from this.

Sentiment analysis using Sentiment Intensity Analyzer

One of the most popular tools for performing sentiment analysis is the Sentiment Intensity Analyzer, which is part of the Natural Language Toolkit (NLTK) library in Python. The ‘Sentiment Intensity Analyzer’ is a rule-based sentiment analysis tool that assesses a text's mood using a word list and lexicon with corresponding sentiment scores. In addition to intensifiers and negations that can change the scores of the words they are associated with, the lexicon includes words that are labelled with scores for positivity, negativity, and neutrality. Tokenizing the text into individual words is required before using the ‘Sentiment Intensity Analyzer’. The NLTK ‘word_tokenize’ function, which separates a sentence into a list of words, can be used to accomplish this. When you have your list of words, you can use the ‘polarity_scores’ function of the ‘Sentiment Intensity Analyzer’ to get a dictionary of sentiment scores. Four keys are included in the dictionary: 'pos', 'neg', 'neu', and 'compound'. The text's score for positivity is found in the 'pos' key, its score for negativity is found in the 'neg' key, its score for neutrality is found in the 'new' key, and its overall sentiment score, which ranges from -1 to 1, is found in the 'compound' key.

The sentiment scores are calculated for a combined entity of ‘source’ and ‘target’ and the relationship, ‘edge’. The 'entity_sentiment_scores' and 'relationship_sentiment_scores' data frames are concatenated to the original data frame, with column names prefixed with 'entity_' and 'relationship_' respectively.

Sentiment analysis using Bidirectional Encoder Representations from Transformers (BERT)

BERT is a potent language model that has demonstrated appreciable advancements over earlier methods for tasks involving natural language processing. Its ability to model the context of a word in a sentence rather than just the individual word is one factor in its success. This is accomplished by using a method known as "masked language modelling," in which a portion of the words in a sentence are arbitrarily changed to a unique token, and the model is trained to predict the original words based on the context of the other words in the sentence. This enables the model to pick up on more subtle meaning nuances and understand the relationships between words in a sentence. BERT's use of a transformer architecture, which enables it to process longer text sequences than earlier models, is another benefit of the system. The model can selectively focus on various elements of the input sequence at various stages of processing thanks to the use of self-attention mechanisms. BERT has also been pre-trained on a sizable corpus of text, enabling it to pick up general language patterns that come in handy for a variety of tasks involving natural language processing. It is simpler and more effective to fine-tune the model for particular tasks, such as sentiment analysis or text classification, thanks to this pre-training.

First, the code loads the BERT tokenizer and pre-trained BERT model for sequence classification from the transformers library. Then, it specifies the device to run the model on, which is set to GPU if available, otherwise CPU. Next, the code extracts the entities and relationships from the data frame and tokenizes them using the BERT tokenizer. The tokenized input is then converted to a tensor and sent to the device. Then, for each entity and relationship, the input tensor is fed through the BERT model, and the output is used to determine the sentiment score. The sentiment score is the index of the highest value in the output tensor, which corresponds to the predicted sentiment label (0 for negative, 1 for positive). Finally, the sentiment scores for each entity and relationship are added back to the data frame as new columns. These columns can be used to further analyze the sentiment of the knowledge graph. Overall, the BERT model is a powerful tool for sentiment analysis as it is pre-trained on a large corpus of text data and has been shown to perform well on various natural language processing tasks. By leveraging the pre-trained model and fine-tuning it on specific data, this code can accurately analyze the sentiment of the entities and relationships in a knowledge graph.

Building an LSTM-based Neural Network to predict the price of the S&P 500 Stock Index

LSTM works by selectively remembering or forgetting information from the previous time steps of a sequence. It consists of a network of repeating units that are connected to each other in a chain-like structure. Two different LSTM models are built with the same architecture. One of the models will take the sentiment scores obtained from the Sentiment Intensity Analyzer. The other model will take the scores obtained from BERT.

Neural Network Architecture

The first LSTM layer has 50 units and is set to return sequences. This means that it will return output for each time step in the input sequence. The LSTM model is defined using the Keras Sequential API. The first layer is an LSTM layer with 50 units and the 'return_sequences' parameter set to True, which means that the layer will output a sequence of values rather than a single value. The input shape is defined based on the number of input features and the number of time steps (which is set to 1). The second layer is another LSTM layer with 50 units. Finally, a Dense layer with a single unit is added, which will output the predicted stock price. The optimizer used in this model is Adam, and the loss function is mean squared error (MSE).

Overall, this architecture is a relatively simple LSTM-based neural network that takes a sequence of input data and produces a single output prediction. The choice of two LSTM layers with 50 units each is a common configuration for this problem.

Train-test split ratio

Initially, 5000 rows are sampled from the data to use as a secondary test set to make sure the model has not overfitted. The data is split 80 to 20 using the 'sklearn.model_selection', 'train_test_split' method. The data is split into 80% training data and 20% testing data with a random state of 42.

Training the model

The model is then trained using the fit method, with early stopping set up using the Early Stopping callback. The training process is monitored for validation loss, and the model will stop training if the validation loss does not improve after 10 epochs. A common issue in machine learning is overfitting, in which a model does well on the training data but fails to generalise to new, untried data. Early stopping is a method for avoiding overfitting when

training a model. Early stopping's fundamental tenet is to halt model training when performance on the validation set stops advancing. This is accomplished by keeping an eye on the validation loss throughout the training phase. When the validation loss stops improving after a specified number of epochs, known as patience, the training process is terminated. In this manner, the model's generalisation performance can be enhanced by avoiding a close fit to the training set of data.

After training, the model is evaluated for its performance using the evaluate method, which calculates the MSE between the predicted and actual stock prices on the testing set. The training and validation loss for each epoch is plotted using the matplotlib library. Mean squared error (MSE) is used to assess how well regression models predict continuous variables. It calculates the average squared difference between the dependent variable's actual and expected values. The model performs better when the MSE value is lower because it shows that the predicted values are more closely aligned with the actual values.

RESULTS

The training process is run for 100 epochs using the "fit" method of the Keras model for both models. The "fit" method takes the training data as input and iteratively updates the weights of the neural network to minimize the mean squared error (MSE) between the predicted output and the actual output. In this case, the MSE is used as the loss function, which is a common metric for regression problems. The training process is monitored using the validation data, which is passed as the "validation_data" parameter to the "fit" method. The validation data is used to evaluate the performance of the model on unseen data and to prevent overfitting. The training process is stopped early using the "EarlyStopping" callback from the Keras library, which monitors the validation loss and stops the training process if the validation loss does not improve for a certain number of epochs. In the case of the SentimentIntensityAnalyzer-LSTM model, the training process is stopped after 32 epochs, as the validation loss does not improve anymore. As for the BERT-LSTM model, it stops at epoch 22. The results of the training process are printed after the training process is completed. The "loss" parameter shows the MSE of the training data, which is gradually decreasing as the model is being trained. The "val_loss" parameter shows the MSE of the validation data, which fluctuates but remains relatively stable throughout the training process.

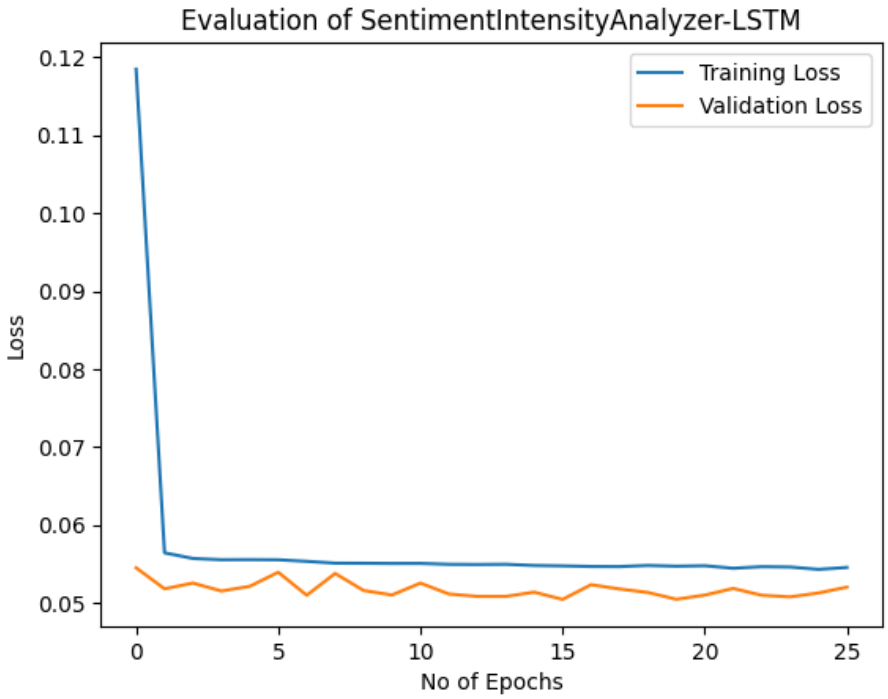

Figure 5: Training and Validation Loss for SentimentIntensityAnalyzer-LSTM model (source: self-generated)

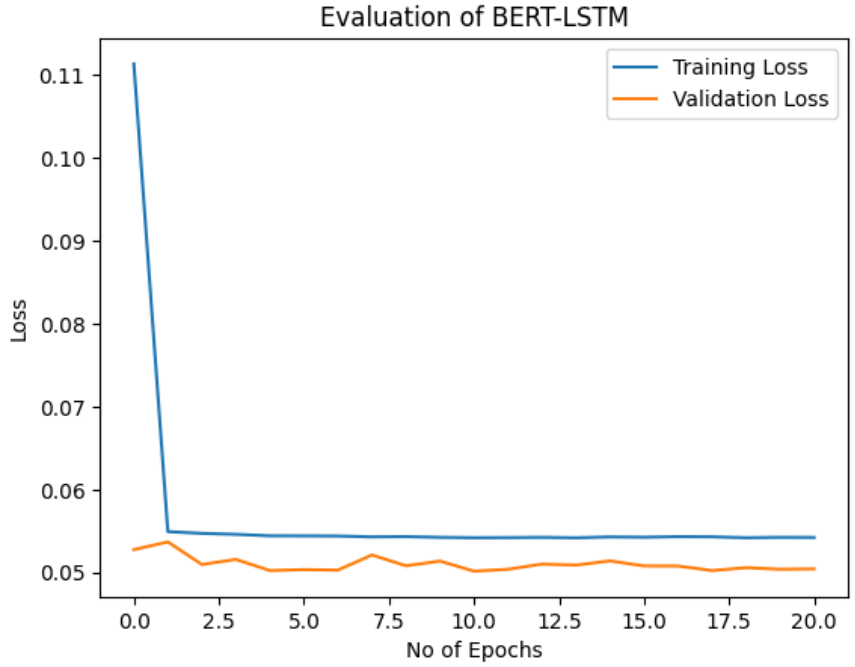

Figure 6: Training and Validation Loss for BERT-LSTM model (source: self-generated)

DISCUSSION

Both the models perform very well on the validation data and the test sample. The MSE of the test data is a metric of the performance of the model on unseen data and is used to evaluate the generalization of the model. In the case of the SentimentIntensityAnalyzer-LSTM model, the MSE of the test data is 0.05246, which is close to the MSE of the validation data, indicating that the model generalizes well to unseen data. The BERT-LSTM model also performs slightly better with an MSE of 0.05049.

Table 1: Accuracy before cross-validation

Model	**MSE**
SentimentIntensityAnalyzer-LSTM	0.05246
BERT-LSTM	0.05049

From the above results, the LSTM Neural Network performs much better than other Neural Network architectures considered in our research.

Using cross-validation

Cross-validation is a statistical technique used to evaluate a predictive model's performance on unseen data. In this study, cross-validation was used to estimate the performance of the models on unseen data, prevent overfitting, and optimize the model. Before using a model on new data, it is common practice in machine learning and data science to assess a model's accuracy, precision, recall, and other metrics. cross-validation is used here to estimate the models' performance on unseen data, to avoid overfitting and also to optimize the model. The number of folds is set to 5 with a random state of 42. K-fold cross-validation is performed using the KFold function from scikit-learn. The model is trained on each fold using the fit function from Keras. Early stopping is used to prevent overfitting, and the training history is stored in a list called history.

Table 2: Accuracy after cross-validation

Model	**Cross-validation MSE**
SentimentIntensityAnalyzer-LSTM	0.01149
BERT-LSTM	0.01145

Both models perform equally well on the unseen data. This shows the model has been fit very accurately and the data has quality. The only difference noticed between the two models is that the BERT-LSTM model reaches higher accuracies in fewer epochs.

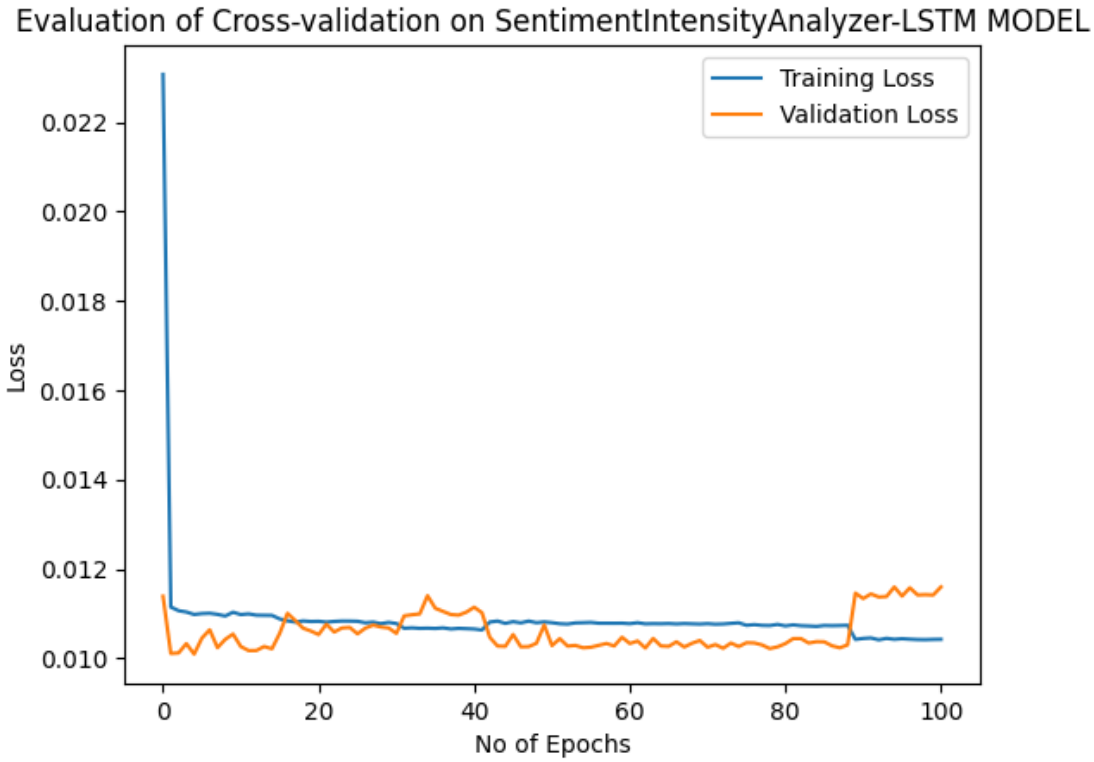

Figure 7: Training and Validation Loss for SentimentIntensityAnalyzer-LSTM model using cross-validation (source: self-generated)

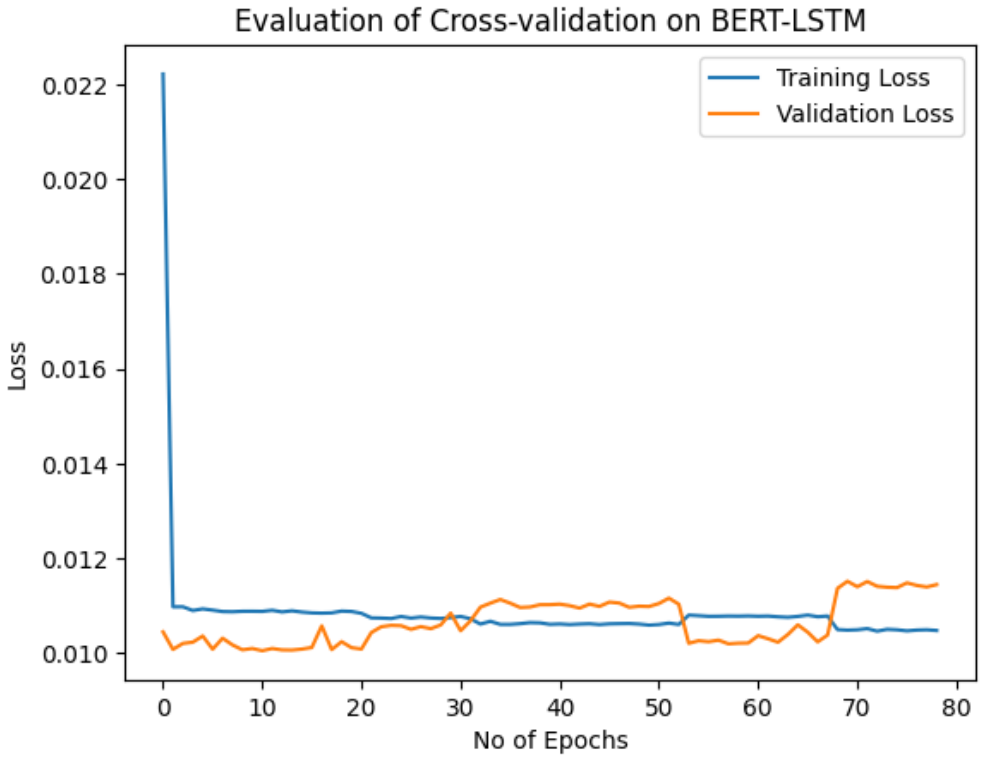

Figure 8: Training and Validation Loss for BERT-LSTM model using cross-validation (source: self-generated).

The predictions of the models were evaluated with the actual data. Visualizations were made initially on the first 150 dates for both models. Then the predictions were consolidated into weeks and a weekly moving average was constructed for 136 weeks which covers the entirety of our dataset.

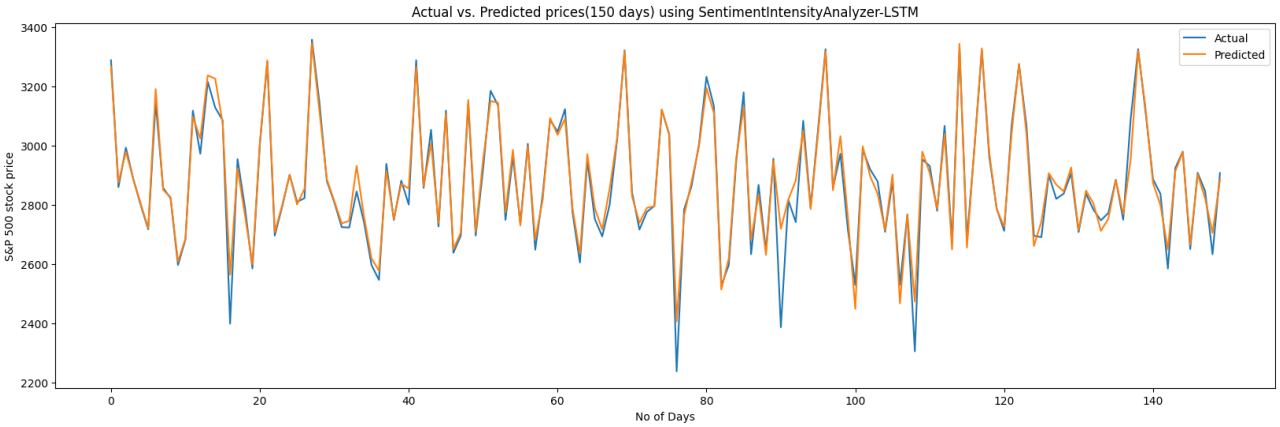

Figure 9: S&P 500 price prediction using SentimentIntensityAnalyzer-LSTM for the first 150 days. (source: self-generated)

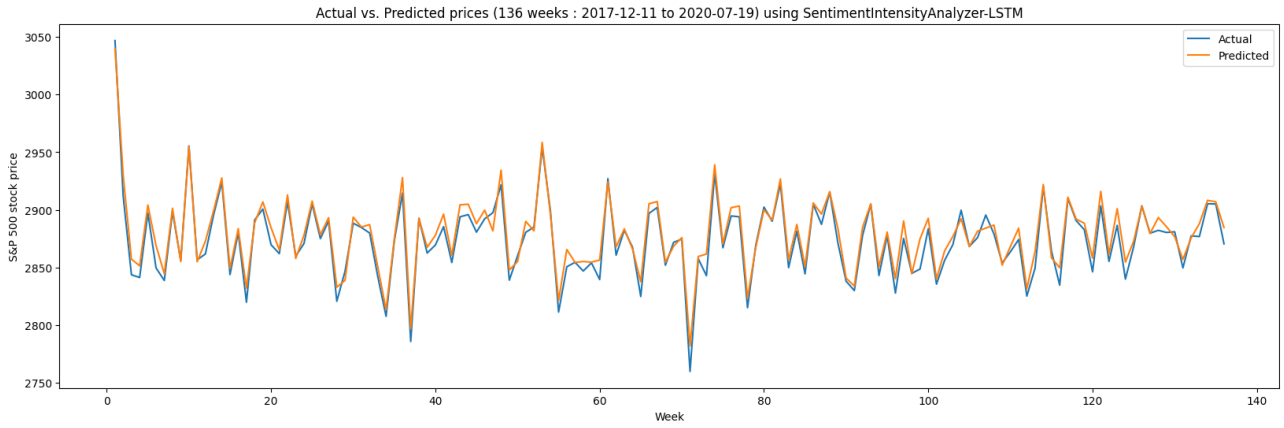

Figure 10: S&P 500 Weekly Average price action predicted using SentimentIntensityAnalyzer-LSTM for 136 weeks. (source: self-generated)

From the visualisations, it is very clear that both models perform very similarly in predicting the price of the S&P 500 stock index.

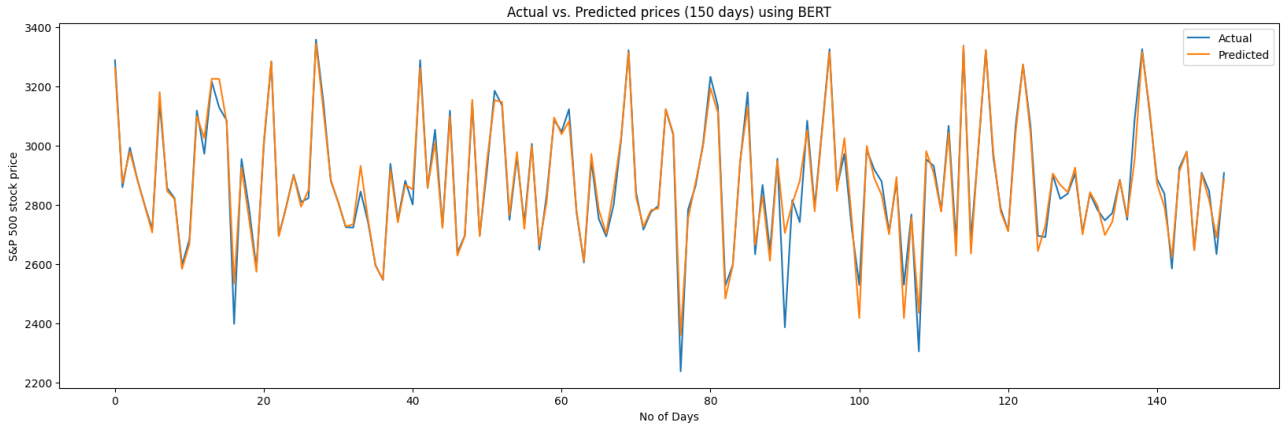

Figure 11: S&P500 price prediction using BERT-LSTM for 150 days (source: self-generated).

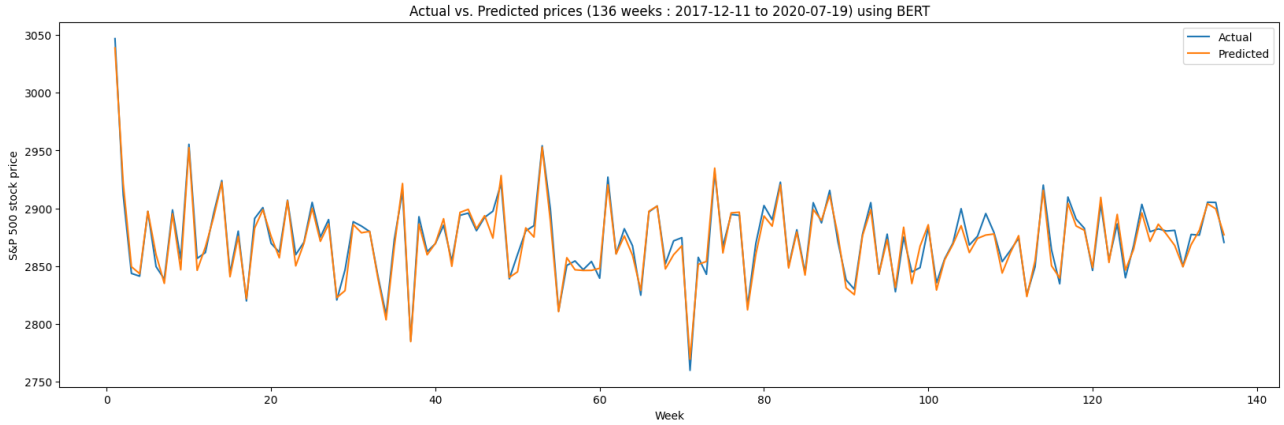

Figure 12: S&P 500 Weekly Average price action predicted using BERT-LSTM for 136 weeks (source: self-generated).

Cross-validation on both models also yields the same results. Both models predict in the same way.

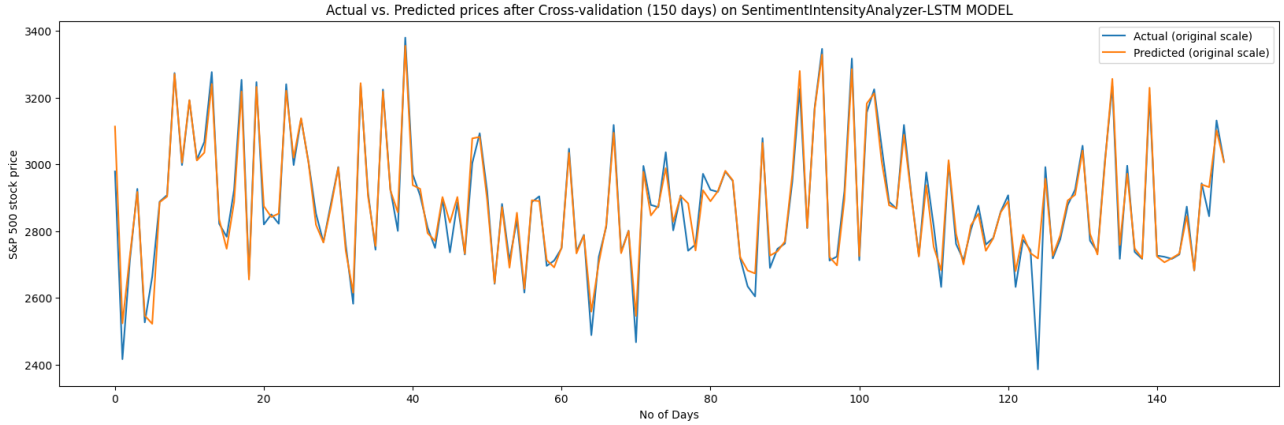

Figure 13: S&P 500 price prediction using SentimentIntensityAnalyzer-LSTM for 150 days after cross-validation (Source: self-generated).

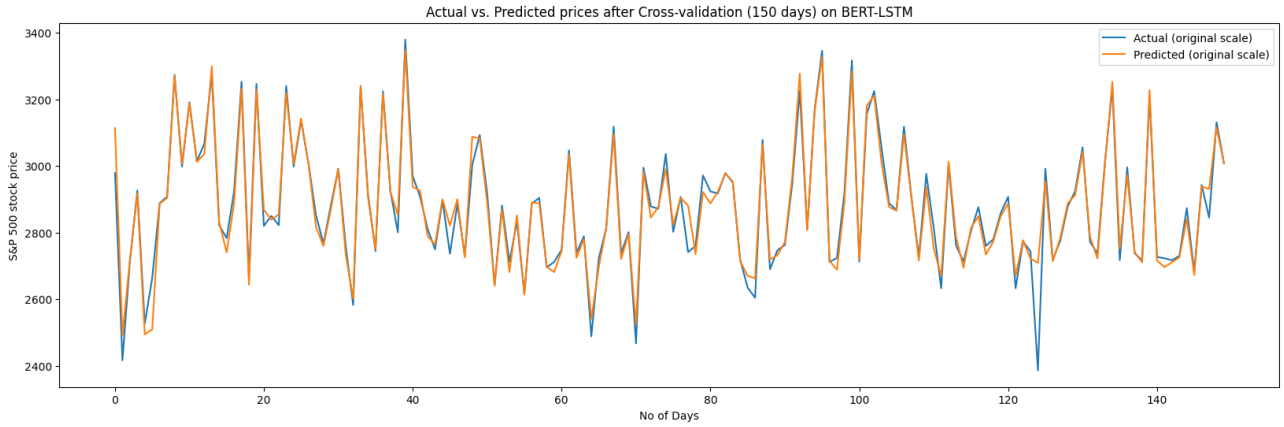

Figure 14: S&P500 price prediction using BERT-LSTM for 150 days after cross-validation (source: self-generated).

One thing noticed from the above graphs is when the stock price has a massive, unexpected dip the model fails to predict the correct closing price.

EVALUATION

Meeting the objectives

The first objective of building a dataset of news headlines and their respective previous and daily closing prices of the S&P 500 stock index was met using various online resources like Kaggle.com and investing.com. These websites helped with the initial step of data collection. Python and Excel were used to preprocess, clean, and merge the datasets. The second objective was to generate event tuples of the news headlines using various researched knowledge graph methods. Even tuples were generated for the majority of the news headlines. These event tuples were used to complete our third objective which was to generate sentiment scores using SentimentIntensityAnalyzer and Bidirectional Encoder Representations from Transformers (BERT). The fourth objective was to predict the price of the S&P 500 stock index using sentiment scores and LSTM. Two highly accurate models were built. Cross-validation techniques were used to make the models perfectly fit.

Personal Reflection

The current chapter reflects my journey through the completion of MSc in Data Science and Analytics Project titled, “Studying the effect of news headlines on the price action of the S&P 500 Stock Index.” The project involved crafted research on existing techniques and methods and newer researched methods. I gained significant insights into how different methods work on different types of data. Initially, I set my goal to build a high-quality data set with verified news headlines. On spending a couple of days looking at various datasets on the internet I realized most of the prebuilt datasets with good quality data were not available for free. Since I could not afford to spend money I had to look at other ways to build a dataset. After weeks of searching, I found a publicly available dataset on Kaggle which combined with the stock prices data from investing.com can be built into a high-quality dataset. Once that obstacle was passed I had to find a system powerful enough to run models with High RAM and V-RAM. That’s when I found Google Colab Pro. I subscribed to the service which provided me with powerful tools and system configurations to train my models with ease. While looking back on my experience through the research process, I appreciate the opportunity to have undertaken such a challenging and rewarding research project on Sentiment Analysis and Neural Networks. I have learned important research, analysis, and critical thinking skills that I will apply to both my academic and professional endeavours. I

think the research was successful in achieving its goals, and the information gathered was pertinent to the project's objectives.

Limitations

Despite the promising results of the study, there are several limitations that must be considered. First, the study only used one type of financial instrument, the S&P 500 stock index, which may not be representative of all financial instruments. Further research could be conducted on other financial instruments to determine if the models perform similarly well. Second, the study only considered a single time frame of the data. It is possible that the models would perform differently in different time periods, such as during market downturns or economic crises. A more comprehensive analysis that considers multiple time periods could provide additional insights into the models' performance. Third, the study only used sentiment analysis tools, like the VADER sentiment analyzer and BERT, to extract sentiment information from news articles. Other sentiment analysis tools may yield different results and affect the models' performance. Fourth, the study only considered a single type of neural network architecture, the LSTM, for both models. There are other neural network architectures that could be explored, such as convolutional neural networks (CNNs) or transformer models like GPT-3, to determine if they perform better than LSTMs in this context. Fifth, the study did not consider the impact of other variables, such as macroeconomic indicators or company-specific news, on stock prices. Incorporating additional variables could improve the models' predictive capabilities and provide a more comprehensive analysis. Lastly, it is important to note that predicting stock prices is a challenging and complex task that is subject to a high degree of uncertainty. While the models in this study performed well on the validation and test data, there is no guarantee that they will perform similarly well in the future. It is important to approach stock price prediction with caution and to consider multiple sources of information when making investment decisions.

Recommendations

- **Obtaining a larger dataset** can help the model tackle real-world changes in market trends. The current model has been trained with only 3 years of data which is not enough. The trends in the stock market have been widespread when it comes to different decades. What works for one timeframe may not work for another.
- **Better System specifications** can improve training. Even though the current setup is good enough to train the models a higher-powered system can cut down training time by a lot.
- **Collecting better-quality news headlines** can improve the process of building knowledge graphs in a better way. We can identify subjects, objects and relationships more accurately.

FUTURE WORK

There are several potential avenues for future work based on the findings of this study. Firstly, one could explore the use of other sentiment analysis models to improve the performance of the LSTM models. The Sentiment Intensity Analyzer used in this study is a lexicon-based approach, which has its limitations. Other approaches, such as machine learning-based sentiment analysis models, may provide better results. Training the models with newer data can keep the model on track with the current market trends. Using additional data like volume and moving averages to train the data can help refine the predictions of the model. Finally, one could explore the use of ensemble methods to combine the predictions of multiple models. This approach has been shown to be effective in other machine-learning tasks and could potentially improve the accuracy and robustness of the stock price prediction model. Overall, there are many opportunities for future work in this area, and further research is needed to develop more accurate and reliable models for predicting stock prices.

CONCLUSION

The performance of two neural network models, SentimentIntensityAnalyzer-LSTM and BERT-LSTM was evaluated in predicting the price of the S&P 500 stock index. Both models were trained using the mean squared error (MSE) as the loss function and monitored with validation data to prevent overfitting. The training process was run for 100 epochs using the Keras "fit" method and was stopped early using the "EarlyStopping" callback from the Keras library. The MSE of the test data was used to evaluate the generalization of the models, and cross-validation was used to estimate their performance on unseen data. The results showed that both models performed well on the validation and test data, with slightly better performance for the BERT-LSTM model. The cross-validation results were consistent with the test data, indicating that the models were accurately fit and the data had good quality. The visualizations of the predictions showed that both models performed similarly in predicting the price of the S&P 500 stock index, with the models failing to predict a massive unexpected dip in the stock price. The performance of the models was evaluated using the MSE, which measures the average squared difference between the dependent variable's actual and expected values. A lower MSE indicates that the predicted values are more closely aligned with the actual values. In this case, the MSE was used to assess how well the models predicted the price of the S&P 500 stock index. The SentimentIntensityAnalyzer-LSTM and BERT-LSTM models were both neural network models that used LSTM layers, but the BERT-LSTM model incorporated the BERT transformer architecture, which has been shown to be effective in natural language processing tasks. The performance of the BERT-LSTM model was slightly better than the SentimentIntensityAnalyzer-LSTM model, which suggests that the BERT architecture contributed to the improved performance. The cross-validation results were consistent with the test data, indicating that the models were accurately fit and the data had good quality. The visualizations of the predictions showed that both models performed similarly in predicting the price of the S&P 500 stock index, with the models failing to predict a massive unexpected dip in the stock price. This suggests that the models were not able to capture all the factors that affect the stock price and may require additional input variables to improve their predictive power. In conclusion, both the SentimentIntensityAnalyzer-LSTM and BERT-LSTM models performed well in predicting the price of the S&P 500 stock index, with a slightly better performance for the BERT-LSTM model. The cross-validation results were consistent with the test data, indicating that the models were accurately fit and the data had good quality. The visualizations of the

predictions showed that both models performed similarly in predicting the price of the S&P 500 stock index, but failed to predict a massive unexpected dip in the stock price. Overall, the models show promise in predicting stock prices but may require additional input variables and further development to improve their predictive power.

REFERENCES

Bollen, J., Mao, H. and Zeng, X. (2011) 'Twitter mood predicts the stock market', Journal of Computational Science, 2(1), pp. 1–8. doi:10.1016/j.jocs.2010.12.007.

Bordino, I. et al. (2012) "Web search queries can predict stock market volumes," PLoS ONE, 7(7). Available at: https://doi.org/10.1371/journal.pone.0040014.

Chau, F., Deesomsak, R. and Koutmos, D. (2016) "Does investor sentiment really matter?," International Review of Financial Analysis, 48, pp. 221–232. Available at: https://doi.org/10.1016/j.irfa.2016.10.003.

Checkley, M.S., Higón, D.A. and Alles, H. (2017) "The hasty wisdom of the mob: How market sentiment predicts stock market behavior," Expert Systems with Applications, 77, pp. 256–263. Available at: https://doi.org/10.1016/j.eswa.2017.01.029.

Chen, Y.-J., Chen, Y.-M. and Lu, C.L. (2016) 'Enhancement of stock market forecasting using an improved fundamental analysis-based approach', Soft Computing, 21(13), pp. 3735–3757. doi:10.1007/s00500-016-2028-y.

CNN (2020) Stock market news today: Dow and S&P 500 updates, CNN. Available at: https://edition.cnn.com/business/live-news/stock-market-news-050820/index.html (Accessed: 03 May 2023).

Cox, J. (2015) Fed raises rates by 25 basis points, first since 2006, CNBC. Available at: https://www.cnbc.com/2015/12/16/fed-raises-rates-for-first-time-since-2006.html (Accessed: 03 May 2023).

Cox, J. (2021) U.S. GDP rose 6.5% last quarter, well below expectations, CNBC. Available at: https://www.cnbc.com/2021/07/29/q2-gdp-rises-at-6point5percent-vs-8point4percent-estimate.html (Accessed: 03 May 2023).

Das, S.R. and Chen, M.Y. (2007) 'Yahoo! for Amazon: Sentiment extraction from small talk on the web', Management Science, 53(9), pp. 1375–1388. doi:10.1287/mnsc.1070.0704.

Devlin, J. et al. (2019) Proceedings of the 2019 Conference of the North [Preprint]. Available at: https://doi.org/10.18653/v1/n19-1423.

De Mauro, A., Greco, M. and Grimaldi, M. (2015) "What is Big Data? A consensual definition and a review of key research topics," AIP Conference Proceedings [Preprint]. Available at: https://doi.org/10.1063/1.4907823.

Ding, X., Liu, B. and Yu, P.S. (2008) ‘A holistic lexicon-based approach to opinion mining’, Proceedings of the international conference on Web search and web data mining - WSDM ’08 [Preprint]. doi:10.1145/1341531.1341561.

Ding, X., Zhang, Y., Liu, T. and Duan, J., 2015, June. Deep learning for event-driven stock prediction. In Twenty-fourth international joint conference on artificial intelligence.

Guo, Y. (2022) “Financial market sentiment prediction technology and application based on Deep Learning Model,” Computational Intelligence and Neuroscience, 2022, pp. 1–10. Available at: https://doi.org/10.1155/2022/1988396.

Harper, D.R. (2022) What drives the stock market?, Investopedia. Investopedia. Available at: https://www.investopedia.com/articles/basics/04/100804.asp (Accessed: March 8, 2023).

Heidein, A. and Parpinelli, R.S. (2022) “Financial News effect analysis on stock price prediction using a stacked LSTM model,” Communication Papers of the 17th Conference on Computer Science and Intelligence Systems [Preprint]. Available at: https://doi.org/10.15439/2022f20.

Hirsh, L. (2021). Understanding the Stock Market. Investopedia. Retrieved from https://www.investopedia.com/terms/s/stockmarket.asp

Hu, Y. et al. (2015) “Application of evolutionary computation for rule discovery in Stock algorithmic trading: A literature review,” Applied Soft Computing, 36, pp. 534–551. Available at: https://doi.org/10.1016/j.asoc.2015.07.008.

Jangid, H. et al. (2018) “Aspect-based financial sentiment analysis using Deep Learning,” Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18 [Preprint]. Available at: https://doi.org/10.1145/3184558.3191827.

Kamstra, M.J., Kramer, L.A. and Levi, M.D. (2003) ‘Winter blues: A sad stock market cycle’, American Economic Review, 93(1), pp. 324–343. doi:10.1257/000282803321455322.

Kenton, W. (2022) S&P 500 index: What it's for and why it's important in investing, Investopedia. Investopedia. Available at: https://www.investopedia.com/terms/s/sp500.asp (Accessed: March 9, 2023).

Lawrence, R., 1997. Using neural networks to forecast stock market prices. University of Manitoba, 333, pp.2006-2013.

Li, X. et al. (2014) “News impact on stock price return via sentiment analysis,” Knowledge-Based Systems, 69, pp. 14–23. Available at: https://doi.org/10.1016/j.knosys.2014.04.022.

Liu, Y. et al. (2018) “Stock price movement prediction from financial news with deep learning and knowledge graph embedding,” Knowledge Management and Acquisition for Intelligent Systems, pp. 102–113. Available at: https://doi.org/10.1007/978-3-319-97289-3_8.

LOUGHRAN, T. and MCDONALD, B. (2011) ‘When is a liability not a liability? textual analysis, dictionaries, and 10-KS’, The Journal of Finance, 66(1), pp. 35–65. doi:10.1111/j.1540-6261.2010.01625.x.

Lovera, F.A., Cardinale, Y.C. and Homsi, M.N. (2021) ‘Sentiment analysis in Twitter based on knowledge graph and deep learning classification’, Electronics, 10(22), p. 2739. doi:10.3390/electronics10222739.

Medhat, W., Hassan, A. and Korashy, H. (2014) “Sentiment analysis algorithms and applications: A survey,” Ain Shams Engineering Journal, 5(4), pp. 1093–1113. Available at: https://doi.org/10.1016/j.asej.2014.04.011.

Nemes, L. and Kiss, A. (2021) “Prediction of stock values changes using sentiment analysis of stock news headlines,” Journal of Information and Telecommunication, 5(3), pp. 375–394. Available at: https://doi.org/10.1080/24751839.2021.1874252.

Nuti, G. et al. (2011) “Algorithmic trading,” Computer, 44(11), pp. 61–69. Available at: https://doi.org/10.1109/mc.2011.31.

Pang, B. and Lee, L. (2008) “Opinion mining and sentiment analysis,” Foundations and Trends® in Information Retrieval, 2(1–2), pp. 1–135. Available at: https://doi.org/10.1561/1500000011.

Patel, J. et al. (2015) ‘Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques’, Expert Systems with Applications, 42(1), pp. 259–268. doi:10.1016/j.eswa.2014.07.040.

Poria, S. et al. (2017) ‘A review of Affective Computing: From unimodal analysis to multimodal fusion’, Information Fusion, 37, pp. 98–125. doi:10.1016/j.inffus.2017.02.003.

Preis, T., Moat, H.S. and Stanley, H.E. (2013) ‘Quantifying trading behavior in financial markets using google trends’, Scientific Reports, 3(1). doi:10.1038/srep01684.

Schumaker, R.P. et al. (2012) “Evaluating sentiment in financial news articles,” Decision Support Systems, 53(3), pp. 458–464. Available at: https://doi.org/10.1016/j.dss.2012.03.001.

Shah, D., Isah, H. and Zulkernine, F. (2019) “Stock market analysis: A review and taxonomy of prediction techniques,” International Journal of Financial Studies, 7(2), p. 26. Available at: https://doi.org/10.3390/ijfs7020026.

Sousa, M.G. et al. (2019) “Bert for stock market sentiment analysis,” 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI) [Preprint]. Available at: https://doi.org/10.1109/ictai.2019.00231.

TETLOCK, P.C., SAAR-TSECHANSKY, M. and MACSKASSY, S. (2008) ‘More than words: Quantifying language to measure firms’ fundamentals’, The Journal of Finance, 63(3), pp. 1437–1467. doi:10.1111/j.1540-6261.2008.01362.x.

Wang, Z. et al. (2018) “Cross-lingual knowledge graph alignment via graph convolutional networks,” Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing [Preprint]. Available at: https://doi.org/10.18653/v1/d18-1032.

Zhong, Q. et al. (2023) “Knowledge graph augmented network towards Multiview Representation Learning for aspect-based sentiment analysis,” IEEE Transactions on Knowledge and Data Engineering, pp. 1–14. Available at: https://doi.org/10.1109/tkde.2023.3250499.

APPENDIX

Dataset Building & cleaning

```
import pandas as pd

dataCNBC = pd.read_csv('/content/drive/MyDrive/stock news/cnbc_headlines.csv')
datagaurd = pd.read_csv('/content/drive/MyDrive/stock news/guardian_headlines.csv')
datareuters = pd.read_csv('/content/drive/MyDrive/stock news/reuters_headlines.csv')

dataCNBC[['Timestamp','Time']] = dataCNBC['Time'].str.split(', ',expand=True)
dataCNBC['Time'] = pd.to_datetime(dataCNBC.Time)
dataCNBC['Time'] = dataCNBC['Time'].dt.strftime('%d/%m/%Y')
dataCNBC = dataCNBC.drop(['Description', 'Timestamp'], axis=1)

datagaurd['Time'] = pd.to_datetime(datagaurd.Time,  errors = 'coerce')
datagaurd['Time'] = datagaurd['Time'].dt.strftime('%d/%m/%Y')
datagaurd = datagaurd.iloc[:,[1,0]]

datareuters['Time'] = pd.to_datetime(datareuters.Time,  errors = 'coerce')
datareuters['Time'] = datareuters['Time'].dt.strftime('%d/%m/%Y')
datareuters = datareuters.drop(['Description'], axis=1)

StockNewsData = pd.concat([dataCNBC, datagaurd, datareuters])
StockNewsData = StockNewsData.reset_index(drop=True)
StockNewsData= StockNewsData.sort_values(by=['Time'])
StockNewsData = StockNewsData.rename(columns={'Time': 'Date'})

spdata =  pd.read_csv('/content/drive/MyDrive/stock news/S&P 500 Historical Data.csv')
spdata['Date'] = pd.to_datetime(spdata['Date'], format='%d/%m/%Y')
spdata['Price'] = spdata['Price'].str.replace(',', '').astype(float)

for date in StockNewsData['Date']:
  checkdate =  date
  if spdata['Date'].isin([date]).any():
     row = spdata.loc[spdata['Date'] == date]
     StockNewsData.loc[StockNewsData['Date'] == date, 'Price'] = row.iloc[0,1]
  else:
     while not spdata['Date'].isin([date]).any():
        date = (pd.to_datetime(date) + pd.Timedelta(days=1,)).strftime("%Y-%m-%d")
     row = spdata.loc[spdata['Date'] == date]
     StockNewsData.loc[StockNewsData['Date'] == checkdate, 'Price'] = row.iloc[0,1]

StockNewsData['prevclose'] = 0.0
```

```
# Iterate over each row in StockNewsData
for i, row in StockNewsData.iterrows():
    date = row['Date']

    # Check if previous day's price is available
    if (spdata['Date'] == date - pd.Timedelta(days=1)).any():
        prev_price = spdata.loc[spdata['Date'] == date - pd.Timedelta(days=1), 'Price'].values[0]
    else:
        prev_date = date - pd.Timedelta(days=1)
        while not spdata['Date'].isin([prev_date]).any():
            prev_date = prev_date - pd.Timedelta(days=1)
        prev_price = spdata.loc[spdata['Date'] == prev_date, 'Price'].values[0]

    # Assign previous day's price to 'prevclose' column
    StockNewsData.at[i, 'prevclose'] = prev_price
StockNewsData.to_csv('/content/drive/MyDrive/cleanedStockNews.csv', index="false")
```

```
import pandas as pd
import re

# read in the dataframe with the column containing the text data
df = pd.read_excel('/content/cleanedStockNews.xlsx')

# define a function to clean the text data
def clean_text(text):
    # replace any non-alphanumeric character except $ and £ with a space
    text = re.sub(r'[^a-zA-Z0-9$£]', ' ', text)
    # remove extra whitespaces and convert to lowercase
    text = re.sub(r'\s+', ' ', text).strip().lower()
    return text

# apply the clean_text function to the column containing the text data
df['Headlines'] = df['Headlines'].apply(clean_text)
```

Building Knowledge graphs

```
import re
import pandas as pd
import bs4
import requests
import spacy
from spacy import displacy
nlp = spacy.load('en_core_web_sm')
from spacy.matcher import Matcher
from spacy.tokens import Span
import networkx as nx
import matplotlib.pyplot as plt
from tqdm import tqdm
nlp = spacy.load('en_core_web_sm')

def get_relations(sent):

  doc = nlp(sent)

  # Matcher class object
  matcher = Matcher(nlp.vocab)

  #define the pattern
  pattern = [{'DEP':'ROOT'},
        {'DEP':'prep','OP':"?"},
        {'DEP':'agent','OP':"?"},
        {'POS':'ADJ','OP':"?"}]

  matcher.add("matching_1", [pattern])

  matches = matcher(doc)
  k = len(matches) - 1

  span = doc[matches[k][1]:matches[k][2]]

  return(span.text)
```

```
import re
import pandas as pd
def get_entities(sent):
    ent1 = ""
    ent2 = ""

    prv_tok_dep = ""
    prv_tok_text = ""

    prefix = ""
    modifier = ""
    for tok in nlp(sent):

        if tok.dep_ != "punct":
            if tok.dep_ == "compound":
                prefix = tok.text
                if prv_tok_dep == "compound":
                    prefix = prv_tok_text + " " + tok.text
            if tok.dep_.endswith("mod") == True:
                modifier = tok.text
                if prv_tok_dep == "compound":
                    modifier = prv_tok_text + " " + tok.text
            if tok.dep_.find("subj") == True:
                ent1 = modifier + " " + prefix + " " + tok.text
                prefix = ""
                modifier = ""
                prv_tok_dep = ""
                prv_tok_text = ""
            if tok.dep_.find("obj") == True:
                ent2 = modifier + " " + prefix + " " + tok.text
            prv_tok_dep = tok.dep_
            prv_tok_text = tok.text


    return [ent1.strip(), ent2.strip()]
```

```
entity_pairs = []

for i in tqdm(df["Headlines"]):
  entity_pairs.append(get_entities(i))

relations = [get_relations(i) for i in tqdm(df['Headlines'])]

# extract subject
source = [i[0] for i in entity_pairs]

# extract object
target = [i[1] for i in entity_pairs]

kg_df = pd.DataFrame({'source':source, 'target':target, 'edge':relations})

merged_df = pd.concat((df,kg_df), axis = 1)
merged_df.to_csv('/content/Knowledgegraphs.csv', index="false")
```

Calculating Sentiment Scores using Sentiment Intensity Analyzer

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
maindf = pd.read_excel('/content/Knowledgegraphs.xlsx')
df = maindf
df = df.dropna()
import nltk
import pandas as pd
from nltk.sentiment import SentimentIntensityAnalyzer
from nltk.tokenize import word_tokenize
nltk.download('punkt')
nltk.download('vader_lexicon')
```

```
# Get entities and relationships from the knowledge graph
df['source'] = df['source'].astype(str)
df['target'] = df['target'].astype(str)
entities = df['source'].str.cat(df['target'], sep=' ')
relationships = df['edge']

# Convert any float values in the entities Series to strings
entities = entities.astype(str)

# Initialize the sentiment analyzer
sid = SentimentIntensityAnalyzer()

# Define a function to extract the sentiment scores
def get_sentiment_scores(text):
    scores = sid.polarity_scores(text)
    return pd.Series({
        'pos_score': scores['pos'],
        'neg_score': scores['neg'],
        'neu_score': scores['neu'],
        'compound_score': scores['compound']
    })

# Analyze sentiment for each entity and relationship
entity_sentiment_scores = entities.apply(get_sentiment_scores)
relationship_sentiment_scores = relationships.apply(get_sentiment_scores)

# Add sentiment scores back to the dataframe
df = pd.concat([df, entity_sentiment_scores.add_prefix('entity_')], axis=1)
df = pd.concat([df, relationship_sentiment_scores.add_prefix('relationship_')], axis=1)
```

```
df = df[['Date', 'entity_pos_score', 'entity_neg_score',     'entity_neu_score',
            'entity_compound_score',          'relationship_pos_score',          'relationship_neg_score',
            'relationship_neu_score',          'relationship_compound_score', 'prevclose', 'Price']]
test_set = df.sample(n=5000, random_state=42)
df = df.drop(test_set.index)
crossdf = df
```

Training SentimentIntensityAnlyser-LSTM model

```
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense, LSTM
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
# Split the data into training and testing sets
X = df.drop(['Date', 'Price'], axis=1)
y = df['Price']
# Create a separate variable for the indices
indices = df.index.values

# Split the data into training and testing sets
X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(X, y, indices, test_size=0.2,
random_state=42)
y_test_actual = y_test

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
y_train = np.array(y_train).reshape(-1, 1)
y_test = np.array(y_test).reshape(-1, 1)
y_train = scaler.fit_transform(y_train)
y_test = scaler.fit_transform(y_test)

# Define the LSTM model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(LSTM(units=50))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model with early stopping
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=10)
history = model.fit(X_train.reshape((X_train.shape[0], X_train.shape[1], 1)),
y_train.reshape((y_train.shape[0], 1)),
                    epochs=100, batch_size=32, validation_data=(X_test.reshape((X_test.shape[0],
X_test.shape[1], 1)), y_test.reshape((y_test.shape[0], 1))), callbacks=[es])

# Evaluate the model
scores = model.evaluate(X_test.reshape((X_test.shape[0], X_test.shape[1], 1)),
y_test.reshape((y_test.shape[0], 1)))
print('MSE: ', scores)
```

```
# Plot accuracy and validation loss for each epoch
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('No of Epochs')
plt.ylabel('Loss')
plt.title('Evaluation of SentimentIntensityAnalyzer-LSTM')
plt.legend()
plt.show()

# Make predictions on the test set
y_pred = model.predict(X_test.reshape((X_test.shape[0], X_test.shape[1], 1)))
```

```
y_test_scaled_back = scaler.inverse_transform(y_test)
y_pred_scaled_back = scaler.inverse_transform(y_pred)

# Print the MSE and plot the actual vs predicted values
print('MSE: ', scores)
plt.clf()
plt.figure(figsize=(20,6))
plt.plot(y_test_scaled_back[:150], label='Actual')
plt.plot(y_pred_scaled_back[:150], label='Predicted')
plt.xlabel('No of Days')
plt.ylabel('S&P 500 stock price')
plt.title('Actual vs. Predicted prices(150 days) using SentimentIntensityAnalyzer-LSTM')
```

```
# Create a DataFrame with the sorted actual and predicted values
dates = df.loc[indices_test].sort_values('Date')['Date']
df_pred = pd.DataFrame({'date': dates, 'actual': y_test_scaled_back.flatten(), 'predicted':
y_pred_scaled_back.flatten()})
# Group the DataFrame by week and calculate the mean values for each week
df_pred['week'] = pd.to_datetime(df_pred['date']).dt.to_period('W').astype(str)
df_pred = df_pred.groupby('week').mean()
# Convert the week index to integers
week_index = range(1, len(df_pred.index)+1)
# Plot the actual and predicted values for each week
plt.clf()
plt.figure(figsize=(20,6))
plt.plot(week_index, df_pred['actual'], label='Actual')
plt.plot(week_index, df_pred['predicted'], label='Predicted')
plt.xlabel('Week')
plt.ylabel('S&P 500 stock price')
plt.title('Actual vs. Predicted prices (136 weeks : 2017-12-11 to 2020-07-19) using
SentimentIntensityAnalyzer-LSTM')
plt.legend()
plt.show()
```

Cross validation on SentimentIntensityAnalyzer-LSTM Model

```
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold
from keras.models import Sequential
from keras.layers import Dense, LSTM
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt

df = crossdf

# Split the data into input and target variables
X = df.drop(['Date', 'Price'], axis=1)
y = df['Price']

# Create a separate variable for the indices
indices = df.index.values

from sklearn.preprocessing import StandardScaler

# Scale the input features
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Scale the target variable
y = np.array(y).reshape(-1, 1)
y_scaler = StandardScaler()
y = y_scaler.fit_transform(y).flatten()

# Define KFold
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Define the LSTM model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X.shape[1], 1)))
model.add(LSTM(units=50))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')
```

```
# Train the model with early stopping and cross-validation
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=10)
history = []
for train_index, test_index in kf.split(X):
   X_train, X_test = X[train_index], X[test_index]
   y_train, y_test = y[train_index], y[test_index]
   model_history = model.fit(X_train.reshape((X_train.shape[0], X_train.shape[1], 1)),
y_train.reshape((y_train.shape[0], 1)),
                  epochs=100, batch_size=32, validation_data=(X_test.reshape((X_test.shape[0],
X_test.shape[1], 1)), y_test.reshape((y_test.shape[0], 1))), callbacks=[es])
   history.append(model_history)

# Evaluate the model
scores = model.evaluate(X_test.reshape((X_test.shape[0], X_test.shape[1], 1)),
y_test.reshape((y_test.shape[0], 1)))
print('MSE: ', scores)

# Plot accuracy and validation loss for each epoch
train_losses = []
val_losses = []
for hist in history:
   train_losses.append(hist.history['loss'])
   val_losses.append(hist.history['val_loss'])

train_losses = np.mean(np.array(train_losses), axis=0)
val_losses = np.mean(np.array(val_losses), axis=0)

plt.plot(train_losses, label='Training Loss')
plt.plot(val_losses, label='Validation Loss')
plt.xlabel('No of Epochs')
plt.ylabel('Loss')
plt.title('Evaluation of Cross-validation on SentimentIntensityAnalyzer-LSTM MODEL')
plt.legend()
plt.show()

# Make predictions on the test set
y_pred = model.predict(X_test.reshape((X_test.shape[0], X_test.shape[1], 1)))
```

Testing on an untouched sample

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from keras.models import load_model

X = test_set.drop(['Date', 'Price'], axis=1)
y = test_set['Price']

# Scale the input features
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Make predictions on the new data
y_pred = model.predict(X.reshape((X.shape[0], X.shape[1], 1)))

# Scale the predicted values back to their original range
y_scaler = StandardScaler()
y = test_set['Price']
y = np.array(y).reshape(-1, 1)
y_scaler.fit(y)
y_pred = y_scaler.inverse_transform(y_pred)

# Print the predicted values
print(y_pred)

# Plot the actual and predicted values for the first 150 dates
plt.figure(figsize=(20,6))
plt.plot(y[:150], label='Actual (original scale)')
plt.plot(y_pred[:150], label='Predicted (original scale)')
plt.xlabel('No of Days')
plt.ylabel('S&P 500 stock price')
plt.title('Actual vs. Predicted prices after Cross-validation (150 days) on SentimentIntensityAnalyzer-
LSTM MODEL')
plt.legend()
plt.show()
```

Calculating sentiment scores using Bidirectional Encoder Representations from Transformers (BERT)

```
df = maindf
df = df.dropna()
!pip install transformers
import pandas as pd
import torch
from transformers import BertTokenizer, BertForSequenceClassification

# Load pre-trained BERT model and tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=2)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model.to(device)

df['source'] = df['source'].astype(str)
df['target'] = df['target'].astype(str)
entities = df['source'].str.cat(df['target'], sep=' ')
relationships = df['edge']

sentiment_scores = []

# Calculate sentiment scores for each headline
for headline in entities:
    # Tokenize the headline and convert to tensor
    encoded_headline = tokenizer.encode_plus(
        headline,
        max_length=128,
        padding='max_length',
        truncation=True,
        return_tensors='pt'
    )
    encoded_headline = {key: tensor.to(device) for key, tensor in encoded_headline.items()}

    # Run the headline through the model and get the prediction
    with torch.no_grad():
        output = model(**encoded_headline)
        scores = output[0].detach().cpu().numpy()[0]
        sentiment_score = scores.argmax()

    sentiment_scores.append(sentiment_score)

# Add the sentiment scores to the dataframe
df['Sentiment Score Entity'] = sentiment_scores
```

Training BERT-LSTM

```
df = df[['Date', 'Sentiment Score Entity', 'Sentiment Score Relation','prevclose', 'Price']]
test_set = df.sample(n=5000, random_state=42)
df = df.drop(test_set.index)
```

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense, LSTM
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt

# Split the data into training and testing sets
X = df.drop(['Date', 'Price'], axis=1)
y = df['Price']

# Create a separate variable for the indices
indices = df.index.values

# Split the data into training and testing sets
X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(X, y, indices, test_size=0.2,
random_state=42)
y_test_actual = y_test

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
y_train = np.array(y_train).reshape(-1, 1)
y_test = np.array(y_test).reshape(-1, 1)
y_train = scaler.fit_transform(y_train)
y_test = scaler.fit_transform(y_test)

# Define the LSTM model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(LSTM(units=50))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')
```

```
# Train the model with early stopping
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=10)
history = model.fit(X_train.reshape((X_train.shape[0], X_train.shape[1], 1)),
y_train.reshape((y_train.shape[0], 1)),
                epochs=100, batch_size=32, validation_data=(X_test.reshape((X_test.shape[0],
X_test.shape[1], 1)), y_test.reshape((y_test.shape[0], 1))), callbacks=[es])

# Evaluate the model
scores = model.evaluate(X_test.reshape((X_test.shape[0], X_test.shape[1], 1)),
y_test.reshape((y_test.shape[0], 1)))
print('MSE: ', scores)

# Plot accuracy and validation loss for each epoch
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('No of Epochs')
plt.ylabel('Loss')
plt.title('Evaluation of BERT-LSTM')
plt.legend()
plt.show()

# Make predictions on the test set
y_pred = model.predict(X_test.reshape((X_test.shape[0], X_test.shape[1], 1)))
```

```
y_test_scaled_back = scaler.inverse_transform(y_test)
y_pred_scaled_back = scaler.inverse_transform(y_pred)

# Print the MSE and plot the actual vs predicted values
print('MSE: ', scores)
plt.clf()
plt.figure(figsize=(20,6))
plt.plot(y_test_scaled_back[:150], label='Actual')
plt.plot(y_pred_scaled_back[:150], label='Predicted')
plt.xlabel('No of Days')
plt.ylabel('S&P 500 stock price')
plt.title('Actual vs. Predicted prices (150 days) using BERT')
plt.legend()
plt.show()
```

```
# Create a DataFrame with the sorted actual and predicted values.
dates = df.loc[indices_test].sort_values('Date')['Date']
df_pred = pd.DataFrame({'date': dates, 'actual': y_test_scaled_back.flatten(), 'predicted':
y_pred_scaled_back.flatten()})

# Group the DataFrame by week and calculate the mean values for each week
df_pred['week'] = pd.to_datetime(df_pred['date']).dt.to_period('W').astype(str)
df_pred = df_pred.groupby('week').mean()

# Convert the week index to integers
week_index = range(1, len(df_pred.index)+1)


# Plot the actual and predicted values for each week
plt.clf()
plt.figure(figsize=(20,6))
plt.plot(week_index, df_pred['actual'], label='Actual')
plt.plot(week_index, df_pred['predicted'], label='Predicted')
plt.xlabel('Week')
plt.ylabel('S&P 500 stock price')
plt.title('Actual vs. Predicted prices (136 weeks : 2017-12-11 to 2020-07-19) using BERT')
plt.legend()
plt.show()
```

Cross-validation on BERT-LSTM model

```
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold
from keras.models import Sequential
from keras.layers import Dense, LSTM
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
df = crossdf

# Split the data into input and target variables
X = df.drop(['Date', 'Price'], axis=1)
y = df['Price']
# Create a separate variable for the indices
indices = df.index.values

from sklearn.preprocessing import StandardScaler
# Scale the input features
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

```
# Define KFold
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Define the LSTM model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X.shape[1], 1)))
model.add(LSTM(units=50))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model with early stopping and cross-validation
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=10)
history = []
for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    model_history = model.fit(X_train.reshape((X_train.shape[0], X_train.shape[1], 1)),
y_train.reshape((y_train.shape[0], 1)),
                    epochs=100, batch_size=32, validation_data=(X_test.reshape((X_test.shape[0],
X_test.shape[1], 1)), y_test.reshape((y_test.shape[0], 1))), callbacks=[es])
    history.append(model_history)

# Evaluate the model
scores = model.evaluate(X_test.reshape((X_test.shape[0], X_test.shape[1], 1)),
y_test.reshape((y_test.shape[0], 1)))
print('MSE: ', scores)

# Plot accuracy and validation loss for each epoch
train_losses = []
val_losses = []
for hist in history:
    train_losses.append(hist.history['loss'])
    val_losses.append(hist.history['val_loss'])

train_losses = np.mean(np.array(train_losses), axis=0)
val_losses = np.mean(np.array(val_losses), axis=0)

plt.plot(train_losses, label='Training Loss')
plt.plot(val_losses, label='Validation Loss')
plt.xlabel('No of Epochs')
plt.ylabel('Loss')
plt.title('Evaluation of Cross-validation on BERT-LSTM')
plt.legend()
plt.show()

# Make predictions on the test set
y_pred = model.predict(X_test.reshape((X_test.shape[0], X_test.shape[1], 1)))
```

Testing on an untouched sample

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from keras.models import load_model

X = test_set.drop(['Date', 'Price'], axis=1)
y = test_set['Price']

# Scale the input features
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Make predictions on the new data
y_pred = model.predict(X.reshape((X.shape[0], X.shape[1], 1)))

# Scale the predicted values back to their original range
y_scaler = StandardScaler()
y = test_set['Price']
y = np.array(y).reshape(-1, 1)
y_scaler.fit(y)
y_pred = y_scaler.inverse_transform(y_pred)

# Print the predicted values
print(y_pred)

# Plot the actual and predicted values for the first 150 dates
plt.figure(figsize=(20,6))
plt.plot(y[:150], label='Actual (original scale)')
plt.plot(y_pred[:150], label='Predicted (original scale)')
plt.xlabel('No of Days')
plt.ylabel('S&P 500 stock price')
plt.title('Actual vs. Predicted prices after Cross-validation (150 days) on BERT-LSTM')
plt.legend()
plt.show()
```