

IMPACT OF SOCIAL MEDIA ON CRYPTOCURRENCY PRICES

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

RAHUL SAI VENKAT MANDAVA (Reg.No - 39110598)
BANDI MOHAN SAI (Reg.No – 39110128)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade “A” by NAAC
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119

APRIL- 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with 'A' grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **RAHUL SAI VENKAT MANDAVA (39110598)** and **BANDI MOHAN SAI (39110128)** who carried out the Project Phase-2 entitled **“IMPACT OF SOCIAL MEDIA ON CRYPTOCURRENCY PRICES”** under my supervision from January 2023 to April 2023.

Internal Guide
Dr.A. Christy MCA, Ph.D.,

Head of the Department
Dr. L. LAKSHMANAN, M.E, Ph.D.



Submitted for Viva voice Examination held on **20.04.2023**

Internal Examiner

ii

External Examiner

DECLARATION

I, **RAHUL SAI VENKAT MANDAVA (39110598)** hereby declare that the Project Phase-2 Report entitled “**IMPACT OF SOCIAL MEDIA ON CRYPTOCURRENCY PRICES**” done by me under the guidance of **Dr.A. Christy, MCA, Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.



DATE:

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D, Dean**, School of Computing, and **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.A. Christy. MCA., Ph. D** for her valuable guidance, suggestions, and constant encouragement paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Cryptocurrencies have become increasingly popular in recent years, with the market capitalization of all cryptocurrencies exceeding \$2 trillion in 2021. As with any investment, investors want to predict cryptocurrency price movements to make informed decisions about buying and selling. One approach to this problem is to use sentiment analysis, a subfield of natural language processing that aims to identify and extract subjective information from text data.

Sentiment analysis has shown promise in predicting cryptocurrency prices by analyzing public sentiment on social media platforms and news articles. By identifying positive and negative sentiment about a particular cryptocurrency, sentiment analysis can provide insights into how people feel about that currency, which can be used to predict its price movements.

However, sentiment analysis is not a foolproof method for predicting cryptocurrency prices. The market is highly volatile, and prices can be influenced by a multitude of factors beyond just public sentiment. Therefore, sentiment analysis should be used in conjunction with other analysis techniques and not relied upon as the sole indicator for making investment decisions.

Despite its limitations, sentiment analysis can provide valuable insights into the market's collective sentiment towards a particular cryptocurrency. As the field of artificial intelligence continues to advance, we may see even more accurate and reliable sentiment analysis tools in the future.

Overall, this project aims to explore the potential of sentiment analysis in predicting cryptocurrency prices and to highlight its strengths and limitations as an investment tool.

LIST OF TABLES

Chapter No	TITLE		Page No.
	ABSTRACT		v
	LIST OF TABLES		vi
	LIST OF FIGURES		viii
	LIST OF ABBREVIATIONS		ix
1	INTRODUCTION		1
	1.1	Existing System	3
	1.2	Scope and Objectives	4
	1.3	Proposed System	4
2	LITERATURE SURVEY		5
	2.1	Inferences from Literature survey	5
	2.2	Open problems in Existing system	8
3	REQUIREMENTS ANALYSIS		9
	3.1	Feasibility Studies/Risk analysis of the project	9
	3.2	Software and Hardware Requirements Specification Document	10
	3.3	System Use case	10
4	DESCRIPTION OF PROPOSED SYSTEM		12
	4.1	Selected Methodology or process model	12
	4.2	Architecture / Overall Design of Proposed System	15
	4.3	Description of Software for Implementation and Testing plan of the Proposed Model/System	17
	4.4	Project Management Plan	19
	4.5	Transition/ Software to Operations Plan	23

5	IMPLEMENTATION DETAILS		24
	5.1	Development and Deployment Setup	24
	5.2	Algorithms	36
	5.3	Testing	37
6	RESULTS AND DISCUSSION		39
7	CONCLUSION		42
	7.1	Conclusion	42
	7.2	Future Work	42
	7.3	Research Issues	45
	7.4	Implementation Issues	46
	REFERENCES		47
	APPENDIX		48
	A. SOURCE CODE		48
	B. SCREENSHOTS		58

LIST OF FIGURES

FIGURE NO	FIGURE NAME	Page No
4.1	System flow chart	17
5.1	Data collection script	26
5.2	Files	26
5.3	Amazon s3	27
5.4	Importing libraries	27
5.5	Importing machine learning libraries	27
5.6	Cleaned dataset	28
5.7	Correlation code	29
5.8	Correlation graph	29
5.9	Prices movement wrt sentiment score	29
5.10	Calculating sentiment scores	31
5.11	Resultant sentiment score data frame	31
5.12	Importing NLTK packages	33
5.13	Cleaning dataset	33
5.14	Cleaning dataset	35
5.15	Splitting dataset for training	35
5.16	Training model	35
5.17	Plotting the models	36
6.1	Model Results	40
6.2	F-1 Score	41
6.3	Confusion matrix	42

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
API	Application programming interface
AWS	Amazon Web Services
CNN	Convolutional neural network
LSTM	Long short-term memory
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
SDK	Software Development Kit

CHAPTER 1

INTRODUCTION

The cryptocurrency market has grown rapidly in recent years, becoming a popular investment opportunity and payment method. Bitcoin, the first cryptocurrency, was created in 2009, and since then, many other cryptocurrencies, such as Ethereum, Litecoin, and Ripple, have been developed. The market has a total capitalization of over \$2 trillion, making it a significant player in the global economy.

One of the primary benefits of cryptocurrency is its decentralization. Unlike traditional currencies, which are controlled by central banks, cryptocurrencies are created and exchanged on a peer-to-peer network. This means that no single entity has control over the market, making it more resistant to the government or institutional influence.

Another advantage of cryptocurrency is its fast and cheap transaction speed. Transactions can be completed instantly and without the need for a third-party intermediary, such as a bank or credit card company. This makes it a more efficient payment method, particularly for international transactions.

Additionally, cryptocurrency offers greater financial privacy and security. The use of cryptography ensures that transactions are secure and transparent, while also allowing for greater anonymity. This is particularly important for individuals who may live in countries with oppressive governments or who want to protect their financial information from hackers.

However, despite its many advantages, the cryptocurrency market also has its downsides. One of the most significant concerns is its volatility. The market can be highly unpredictable and prone to sudden price fluctuations. This can lead to significant losses for investors and traders, particularly those who are inexperienced or do not fully understand the market.

Furthermore, the market is largely unregulated, which makes it vulnerable to scams, fraud,

and market manipulation. Cryptocurrency exchanges and wallets have been hacked in the past, resulting in the loss of millions of dollars' worth of cryptocurrency. Additionally, the lack of regulation means that cryptocurrencies can be used for illegal activities, such as money laundering and terrorism financing.

Another issue with cryptocurrency is its high energy consumption. The process of mining cryptocurrency requires a significant amount of computing power and energy, which has led to concerns about its impact on the environment. The energy consumption of Bitcoin alone is estimated to be equal to that of the entire country of Argentina. This has led to calls for the development of more sustainable and environmentally friendly methods of cryptocurrency mining.

Finally, the lack of widespread adoption and acceptance of cryptocurrency is also a concern. While many businesses and retailers are beginning to accept cryptocurrencies as a payment method, they are still not widely used or accepted. This limits the usefulness of cryptocurrency as a payment method and can also contribute to its volatility.

In conclusion, the cryptocurrency market offers many benefits, including decentralization, fast transaction speed, and greater financial privacy and security. However, it is also prone to volatility, scams, fraud, market manipulation, high energy consumption, and lack of widespread adoption. As with any investment opportunity, it is essential to approach the cryptocurrency market with caution and awareness of its potential risks and downsides. Investors and traders need to be diligent in their research, take steps to protect their assets, and be prepared for the potential risks and volatility of the market. Ultimately, the future of cryptocurrency will depend on the ability of regulators and industry leaders to address these challenges and create a safer and more stable environment for investors and traders.

1.1 Existing System

Sentiment analysis is a natural language processing (NLP) technique that involves using algorithms to analyze large volumes of text data and classify the sentiment or emotion expressed in the text as positive, negative, or neutral. In the context of cryptocurrency price prediction, sentiment analysis can be used to analyze social media posts, news articles, and other text-based sources to gauge the overall sentiment of the market towards a particular cryptocurrency.

The basic idea is that positive sentiment towards a cryptocurrency, expressed through social media posts and other sources, may lead to an increase in demand and therefore a rise in price. Conversely, negative sentiment may lead to a decrease in demand and a drop-in price.

There are several ways in which sentiment analysis can be useful in cryptocurrency price prediction. Firstly, it can provide valuable insights into the overall mood and sentiment of the market towards a particular cryptocurrency. This information can be used to identify trends and patterns that may influence future price movements.

Secondly, sentiment analysis can be used to identify key influencers and opinion leaders in the cryptocurrency community. By tracking their social media activity and sentiment towards a particular cryptocurrency, investors and traders can gain valuable insights into the potential direction of the market.

Finally, sentiment analysis can be used to detect unusual spikes in sentiment or sudden changes in sentiment towards a particular cryptocurrency. These anomalies may be an early warning sign of potential market manipulation or other market-moving events.

In conclusion, sentiment analysis is a powerful tool for cryptocurrency price prediction. By analyzing large volumes of text data, investors and traders can gain valuable insights into the overall sentiment of the market towards a particular cryptocurrency, identify key influencers and opinion leaders, and detect anomalies that may signal potential market-moving events.

1.2 Scope and Objectives

The objective of the project to see the impact of cryptocurrency prices using sentiment analysis of Twitter data is to analyze the relationship between social media sentiment and cryptocurrency price movements. The project aims to use natural language processing techniques to analyze large volumes of tweets related to cryptocurrencies, and to identify patterns and trends in sentiment that may be correlated with changes in cryptocurrency prices.

The goal of this project is to develop a predictive model that can accurately forecast changes in cryptocurrency prices based on sentiment analysis of Twitter data. This model can be used by investors and traders to make more informed decisions about when to buy or sell cryptocurrencies and to mitigate risks associated with price volatility.

Overall, the objective of this project is to improve our understanding of the relationship between social media sentiment and cryptocurrency prices and to develop new tools and techniques for analyzing and predicting price movements in the cryptocurrency market.

1.3 Proposed System

The process of predicting cryptocurrency prices using sentiment analysis of Twitter data involves analyzing large volumes of tweets related to cryptocurrencies and identifying patterns and trends in sentiment that may be correlated with changes in cryptocurrency prices. Natural language processing techniques are used to classify the sentiment of each tweet as positive, negative, or neutral. This sentiment data is then analyzed to identify trends and patterns that may be used to develop predictive models for forecasting changes in cryptocurrency prices. These models can be used by investors and traders to make more informed decisions about when to buy or sell cryptocurrencies based on the sentiment of the market.

CHAPTER 2

LITERATURE SURVEY

2.1. Inferences from Literature survey

ABSTRACT: Sunny King, Scott Nadal

A peer-to-peer crypto-currency design derived from Satoshi Nakamoto's Bitcoin. Proof-of-stake replaces proof-of-work to provide most of the network security. Under this hybrid design proof-of-work mainly provides initial minting and is largely non-essential in the long run. Security level of the network is not dependent on energy consumption in the long term thus providing an energy efficient and more cost-competitive peer-to-peer crypto-currency. Proof-of-stake is based on coin age and generated by each node via a hashing scheme bearing similarity to Bitcoin's but over limited search space. Block chain history and transaction settlement are further protected by a centrally broadcasted checkpoint mechanism.

2.1.1 A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities

ABSTRACT: A. A. Monrat, O. Schelén and K. Andersson, "A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities," in IEEE Access, vol. 7, pp. 117134-117151, 2019, doi: 10.1109/ACCESS.2019.2936094.

Blockchain is the underlying technology of a number of digital cryptocurrencies. Blockchain is a chain of blocks that store information with digital signatures in a decentralized and distributed network. The features of blockchain, including decentralization, immutability, transparency and auditability, make transactions more secure and tamper proof. Apart from cryptocurrency, blockchain technology can be used in financial and social services, risk management, healthcare facilities, and so on. A number of research studies focus on the opportunity that blockchain provides in various application domains. This paper presents a comparative study of the tradeoffs of blockchain and also explains the taxonomy and architecture of blockchain, provides a comparison among different consensus mechanisms and discusses challenges, including scalability, privacy, interoperability, energy consumption and regulatory issues. In addition, this paper also notes the future scope of blockchain technology.

2.1.2 Like It or Not: A Survey of Twitter Sentiment Analysis Methods

ABSTRACT:

Anastasia Giachanou, Fabio Crestani

Sentiment analysis in Twitter is a field that has recently attracted research interest. Twitter is one of the most popular microblog platforms on which users can publish their thoughts and opinions. Sentiment analysis in Twitter tackles the problem of analyzing the tweets in terms of the opinion they express. This survey provides an overview of the topic by investigating and briefly describing the algorithms that have been proposed for sentiment analysis in Twitter. The presented studies are categorized according to the approach they follow. In addition, we discuss fields related to sentiment analysis in Twitter including Twitter opinion retrieval, tracking sentiments over time, irony detection, emotion detection, and tweet sentiment quantification, tasks that have recently attracted increasing attention. Resources that have been used in the Twitter sentiment analysis literature are also briefly presented. The main contributions of this survey include the presentation of the proposed approaches for sentiment analysis in Twitter, their categorization according to the technique they use, and the discussion of recent research trends of the topic and its related fields.

2.1.3 Cryptocurrency Price Prediction Using Tweet Volumes and Sentiment Analysis

ABSTRACT:

Abraham, Jethin; Higdon, Daniel; Nelson, John; and Ibarra, Juan (2018)
"Cryptocurrency Price Prediction Using Tweet Volumes and Sentiment Analysis," SMU
Data Science Review: Vol. 1

In this paper, the author has presented a method for predicting changes in Bitcoin and Ethereum prices utilizing Twitter data and Google Trends data. Bitcoin and Ethereum, the two largest cryptocurrencies in terms of market capitalization represent over \$160 billion dollars in combined value. However, both Bitcoin and Ethereum have experienced significant price swings on both daily and long-term valuations. Twitter is increasingly used as a news source influencing purchase decisions by informing users of the currency and its increasing popularity. As a result, quickly understanding the impact of tweets on price direction can provide a purchasing and selling advantage to a cryptocurrency user or a trader. By analyzing tweets, we found that tweet volume, rather than tweet sentiment (which is invariably overall positive regardless of price direction), is a predictor of price direction. By utilizing a linear model that takes as input tweets and Google Trends data, we were able to accurately predict the direction of price changes. By utilizing this model, a person is able to make better-informed purchase and selling decisions related to Bitcoin and Ethereum.

2.1.4 Bitcoin Spread Prediction Using Social And Web Search Media

Abstract:

In the last decade, Web 2.0 services such as blogs, tweets, forums, chats, email etc. have been widely used as communication media, with very good results. Bitcoin, a decentralized electronic currency system, represents a radical change in financial systems, attracting a large number of users and a lot of media attention. In this work, we investigated if the spread of the Bitcoin's price is related to the volumes of tweets or Web Search media results. We compared trends of price with Google Trends data, volume of tweets and particularly with those that express a positive sentiment. We found significant cross correlation values between Bitcoin price and Google Trends data, arguing our initial idea based on studies about trends in stock and goods market.

2.2 Open problems in Existing system

Predicting Bitcoin price fluctuation with Twitter sentiment analysis

Abstract:

E. Stenqvist, J. Lönnö,

programmatically deriving sentiment has been the topic of many a thesis: it's application in analyzing 140 character sentences, to that of 400-word Hemingway sentences; the methods ranging from naive rule based checks, to deeply layered neural networks. Unsurprisingly, sentiment analysis has been used to gain useful insight across industries, most notably in digital marketing and financial analysis. An advancement seemingly more excitable to the mainstream, Bitcoin, has risen in number of Google searches by three-folds since the beginning of this year alone, not unlike it's exchange rate. The decentralized cryptocurrency, arguably, by design, a pure free market commodity – and as such, public perception bears the weight in Bitcoins monetary valuation. This thesis looks toward these public perceptions, by analyzing 2.27 million Bitcoin-related tweets for sentiment fluctuations that could indicate a price change in the near future. This is done by a naive method of solely attributing rise or fall based on the severity of aggregated Twitter sentiment change over periods ranging between 5 minutes and 4 hours, and then shifting these predictions forward in time 1, 2, 3 or 4 time periods to indicate the corresponding BTC interval time. The prediction model evaluation showed that aggregating tweet sentiments over a 30 min period with 4 shifts forward, and a sentiment change threshold of 2.2%, yielded a 79% accuracy.

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATIONS

3.1 Feasibility Studies/ Risk analysis of the project

Feasibility:

Data availability: Twitter data is freely available and can be easily accessed through the Twitter API.

Technological resources: The project requires a robust computing infrastructure with adequate storage and processing capabilities, along with relevant software tools and libraries.

Expertise: The project requires expertise in machine learning, natural language processing, and data analytics.

Costs: There may be costs associated with acquiring computing infrastructure and data storage solutions, along with hiring skilled personnel.

Regulatory compliance: There may be regulatory compliance requirements related to data privacy, cybersecurity, and intellectual property rights.

Risk:

Data quality: The accuracy and reliability of the Twitter data could affect the accuracy of the predictions.

Model accuracy: The predictive model's accuracy is crucial in making informed decisions, and inaccurate predictions could lead to significant financial losses.

Market volatility: The cryptocurrency market is highly volatile, which could lead to unexpected fluctuations in the predicted prices.

Cybersecurity risks: The system may be vulnerable to cyber-attacks, data breaches, and other security threats.

Mitigation strategies such as regular monitoring of data quality, model validation, risk management frameworks, and robust cybersecurity measures can be put in place to address these risks. Overall, the project's feasibility and risk analysis will help ensure that the project's outcomes are positive and sustainable in the long run.

Overview:

This section provides a general description, including characteristics of the users of this project, the product's hardware, and the functional and data requirements of the product. Section 3 gives the functional requirements, data requirements and constraints and assumptions made while designing the E-Store. It also gives the user viewpoint of product. Section 3 also gives the specific requirements of the product. Section 3 also discusses the external interface requirements and gives detailed description of functional requirements.

3.2 Software and Hardware Requirements Specification Document**Hardware Requirements**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should be what the system does and not how it should be implemented.

Processor - i3/i5/i7

Speed - 1.1 GHz

Ram - 4 GB

Hard Disk - 20 GB

3.2.1 Software Requirements

Operating System - Linux, Ubuntu, Mac, Windows XP, 7, 8, 8.1, 10.

Frontend - HTML, CSS, React JS,

Code editors: VS code, Jupyter

Libraries: Pandas

3.3 System Use Case

The system use case of the project of predicting cryptocurrency prices using sentiment analysis of Twitter data involves several stakeholders and use cases.

Data collection and pre-processing:

The system will collect and pre-process tweets related to cryptocurrency from the

Twitter API. The data will be filtered and cleaned to remove irrelevant information and spam. Pre-processing techniques such as text normalization, tokenization, stop-word removal, and sentiment analysis will be applied to extract relevant features.

Model training and validation:

The pre-processed data will be used to train a machine learning model for cryptocurrency price prediction. The model will be validated using historical price data and corresponding sentiment scores.

Prediction and evaluation:

The trained model will be used to predict cryptocurrency prices for a specific time period based on the sentiment analysis of the tweets collected during that time period. The predicted prices will be compared with actual prices to evaluate the accuracy of the model.

Investment decision-making:

The predicted cryptocurrency prices can be used by investors, traders, and analysts to make informed decisions about cryptocurrency investments. For example, if the sentiment analysis indicates a positive trend, investors may consider buying cryptocurrency, while a negative trend may indicate a sell-off.

Regulatory compliance:

The system will comply with regulatory requirements related to data privacy, cybersecurity, and intellectual property rights.

System maintenance and upgrades:

The system will require regular maintenance and upgrades to ensure that it remains relevant and up-to-date with changes in the cryptocurrency market and technology advancements.

Overall, the system use case of the project of predicting cryptocurrency prices using sentiment analysis of Twitter data will enable stakeholders to make data-driven investment decisions based on the sentiment analysis of social media data.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 PROPOSED METHODOLOGY

A) Data Pre-processing - After extracting twitter data using the Twitter API, pre-processing is the next step to make the data ready for further analysis. The pre-processing of twitter data involves several steps such as data cleaning, data structuring, data transformation and data normalization.

1. Data cleaning: The first step in data pre-processing is to clean the data by removing any irrelevant tweets, duplicate tweets, and irrelevant information such as URLs, mentions, and hashtags. This will help to reduce the amount of noise in the data and make it easier to analyze.

2. Data structuring: The next step is to structure the data in a way that makes it easy to analyze. This could involve converting the tweets into a tabular format, with columns for the tweet text, user, date and time, etc. The text of the tweets can be tokenized to break the tweets into individual words, which can then be used to create word frequency or sentiment analysis.

3. Data transformation: The data should be transformed into a format that is appropriate for the analysis you want to conduct. For example, if you want to look at sentiment analysis, the data may need to be labeled as positive, negative, or neutral.

4. Data normalization: This step is to standardize the data, for example, by converting all text to lowercase or removing stop words.

5. Handling missing data and outliers: It's important to handle missing data, outliers, and invalid data such as URLs, mentions, and hashtags.

Once the data has been preprocessed, it can be analyzed using a variety of techniques such as text mining, natural language processing, sentiment analysis, etc. It's important to note that pre-processing is a crucial step in data analysis, as it ensures that the data is accurate, consistent, and ready for further analysis.

B) Sentiment analysis

We first classify the data by method of sentiment scoring

Sentiment scoring is the process of assigning a numerical value or score to a piece of text, such as a tweet, based on its emotional tone. The score can range from -1 (negative) to 1 (positive) or 0 (neutral). Sentiment scoring for Twitter data on the impact of crypto prices typically involves the following steps:

1. Creating a dictionary of words and their associated sentiments: The first step is to create a lexicon or dictionary of words and their associated sentiments. This lexicon can be created manually or using pre-existing lexicons.
2. Tokenizing tweets: The next step is to tokenize the tweets, which means breaking them down into individual words.
3. Mapping words to sentiments: Each word in the tweet is then mapped to the sentiment in the lexicon. For example, a word like "good" would be mapped to a positive sentiment, while a word like "bad" would be mapped to a negative sentiment.
4. Calculating sentiment scores: The sentiment scores are calculated by summing up the sentiments of all the words in the tweet and normalizing the score to a range of -1 to 1.
5. Handling negations: Some words such as "not" or "never" can change the sentiment of a word, for example "not good" is negative sentiment. Sentiment analysis algorithms consider these words to handle negations and adjust the sentiment score accordingly.
6. Handling Emoji and Emoticons: Emoji and emoticons are widely used in tweets, so sentiment scoring process also consider these elements to classify tweets as positive or negative.
7. Handling sarcasm: Sarcasm is a form of irony, which is difficult for computers to understand, so sentiment scoring process also consider this aspect to classify tweets as positive or negative.
8. Handling subjectivity: Some tweets are subjective and express personal opinions, so sentiment scoring process also consider this aspect to classify tweets as positive or negative.

And we will be using Numerical language processing for sentimental analysis we used Spark NLP as the NLP library for running machine algorithms. Finally, we run this

built pipeline on every algorithm such as Support vector classifier, Logistic Regression, Naive Bayes and Linear regression to finally get accurate scores and choose the best score with most accuracy.

4.2 OVERALL DESIGN OF PROPOSED SYSTEM/ ARCHITECTURE

The proposed system aims to predict Bitcoin prices using sentiment analysis of tweets. The overall design of the system involves several steps.

Firstly, data collection will involve the use of the Twitter API to extract tweets related to Bitcoin from a specific time range. The tweets will be filtered and cleaned to remove duplicates, irrelevant information, and spam.

Secondly, the cleaned tweets will be preprocessed using natural language processing techniques to extract relevant features such as sentiment scores, keywords, and hashtags. This will involve text normalization, tokenization, stop-word removal, stemming or lemmatization, and sentiment analysis.

Thirdly, the preprocessed data will be used to train a machine learning model for Bitcoin price prediction. This will involve the selection of appropriate features, data normalization, and feature engineering. The model will be trained on historical Bitcoin price data and the corresponding sentiment scores of the tweets.

Fourthly, the trained model will be used to predict Bitcoin prices for a specific time period based on the sentiment analysis of the tweets collected during that time period.

Finally, the predicted Bitcoin prices will be compared with the actual Bitcoin prices to evaluate the accuracy of the model. The system will be evaluated using various performance metrics such as accuracy, precision, recall, F1 score, and root mean squared error.

The proposed system has the potential to provide valuable insights into the relationship between social media sentiment and Bitcoin prices. It can be used by investors, traders, and analysts to make informed decisions about Bitcoin investments. The system can also be extended to other cryptocurrencies and social media platforms to provide a comprehensive analysis of the cryptocurrency market.

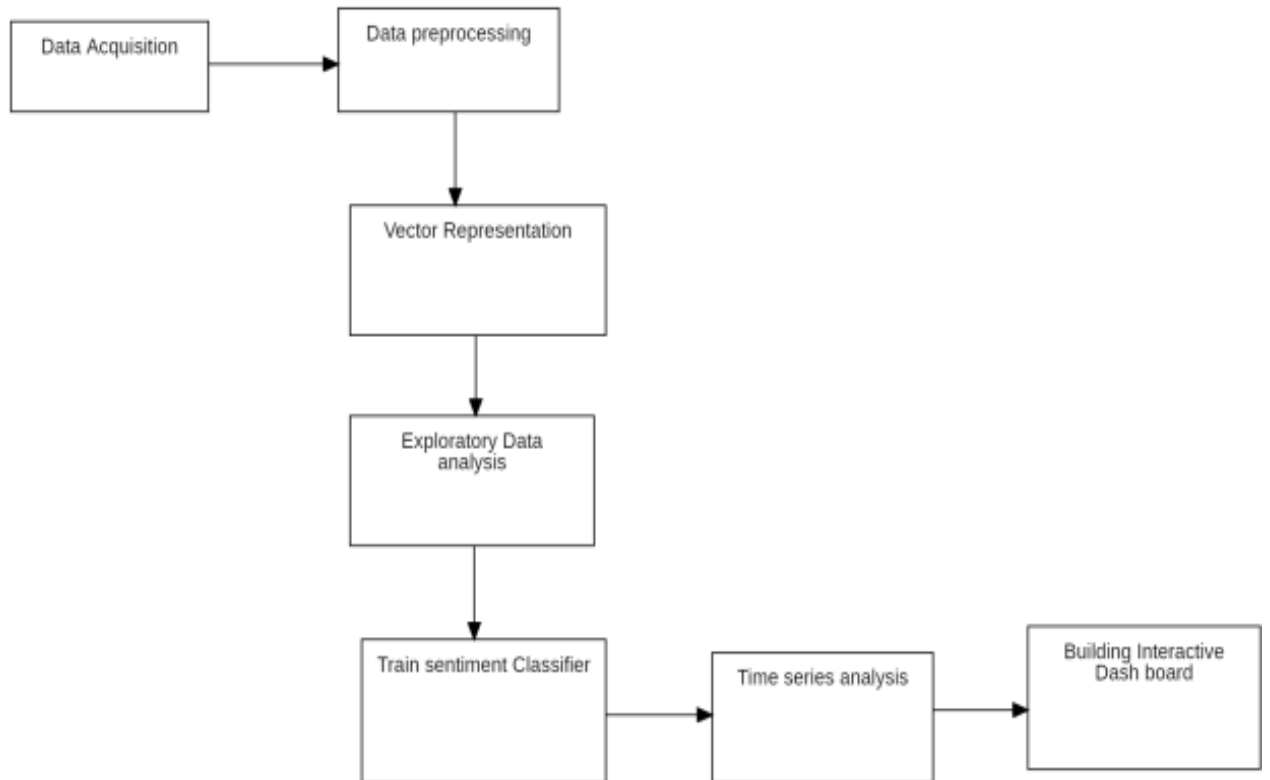


Fig. 4.1 System flow chart

4.3 DESCRIPTION OF SOFTWARE FOR IMPLENENTATION

Jupyter: Jupyter Notebook is an open-source web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It supports over 40 programming languages, including Python, R, and Julia, and can be run on a local machine or on a remote server. Jupyter Notebook is widely used in data science and scientific computing for exploratory data analysis, data visualization, machine learning, and more. Its interactive nature and ease of use make it a popular choice among researchers, data analysts, and educators.

VSCode: Visual Studio Code, commonly known as VS Code, is a free and open-source code editor developed by Microsoft. It provides a streamlined and customizable interface for writing and debugging code across multiple programming languages, including Python, JavaScript, and C++. VS Code offers features such as syntax highlighting, autocompletion, debugging, Git integration, and extensions to enhance productivity and functionality. It runs on multiple platforms, including Windows, macOS, and Linux, and has a large and active community of users and developers contributing to its growth and development.

Pandas: Pandas is a popular open-source data manipulation and analysis library for Python. It provides easy-to-use data structures, such as the Data Frame and Series, for handling and organizing large datasets. Pandas offers a wide range of data wrangling and cleaning functions, including filtering, grouping, and joining data, as well as advanced operations like pivoting and reshaping. It integrates well with other Python libraries, such as NumPy and Matplotlib, for data analysis and visualization tasks. With its rich set of features and extensive documentation, Pandas is widely used in data science and scientific computing.

Vader sentiment: VADER (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool for natural language processing. It is specifically designed to analyze the sentiment of social media text, taking into account nuances such as sarcasm, irony, and emoticons. VADER uses a combination of lexical features and grammatical rules to determine the sentiment of a given text, providing a compound score that represents the overall positivity, negativity, or neutrality of the text. It is widely used in fields such as marketing, customer service, and political analysis.

SKlearn: Scikit-learn(sklearn) is an open-source machine-learning library for Python that provides a wide range of tools for data mining, analysis, and modeling. It includes a comprehensive set of algorithms for supervised and unsupervised learning, as well as tools for feature selection, pre-processing, and evaluation. Sklearn is built on top of other popular scientific computing libraries such as NumPy, SciPy, and Matplotlib, and is widely used in academic and industry settings for a variety of machine learning tasks.

NLTK: The Natural Language Toolkit (NLTK) is a popular open-source library for Python that provides tools for natural language processing (NLP). It includes a wide range of modules and corpora for tasks such as tokenization, part-of-speech tagging, sentiment analysis, and named entity recognition. NLTK also provides tools for building and training models for various NLP tasks. With its rich set of features and extensive documentation, NLTK is widely used in research and industry for a variety of NLP applications.

TensorFlow: TensorFlow is an open-source machine learning library developed by Google that provides tools for building and training various types of neural networks. It includes a wide range of modules and tools for tasks such as image and text classification, object detection, and natural language processing. TensorFlow is known for its ability to handle large-scale data and its ease of use in distributed computing environments. It has become a popular choice for both research and production-level machine learning applications.

4.4 PROJECT MANAGEMENT PLAN

Project initiation

Define project scope, objectives, and deliverables
Identify stakeholders and establish communication channels
Set project timeline and budget

Data collection

Identify relevant Twitter accounts and hashtags for cryptocurrency analysis
Implement data scraping and collection processes
Clean and preprocess the collected data

Sentiment analysis

Perform sentiment analysis on the Twitter data using natural language processing techniques
Train and test machine learning models for sentiment analysis
Evaluate the performance of the sentiment analysis models

Correlation analysis

- 1) Analyze the correlation between sentiment scores and cryptocurrency prices
- 2) Explore other relevant factors that may influence cryptocurrency prices

Model development

- 1) Develop machine learning models for cryptocurrency price prediction using sentiment analysis and other relevant factors
- 2) Train and test the models using historical data
- 3) Evaluate the performance of the models using metrics such as accuracy, precision, and recall

Deployment and maintenance

- 1) Deploy the cryptocurrency price prediction models in a web application or API
- 2) Monitor and update the models with new data and improved algorithms
- 3) Continuously evaluate and optimize the models based on user feedback market trends.

Project Timeline

Week 1:

- Define project scope, objectives, and deliverables

- Identify stakeholders and establish communication channels
- Set project timeline and budget
- Identify relevant Twitter accounts and hashtags for cryptocurrency analysis

Week 2:

- Implement data scraping and collection processes
- Clean and preprocess the collected data

Week 3:

- Perform sentiment analysis on the Twitter data using natural language processing techniques
- Train and test machine learning models for sentiment analysis

Week 4:

- Evaluate the performance of the sentiment analysis models
- Analyze the correlation between sentiment scores and cryptocurrency prices

Week 5:

- Develop machine learning models for cryptocurrency price prediction using sentiment analysis and other relevant factors
- Train and test the models using historical data.

Week 6:

- Evaluate the performance of the models using metrics such as accuracy, precision, and recall
- Deploy the cryptocurrency price prediction models in a web application or API

Week 7:

- Monitor and update the models with new data and improved algorithms
- Continuously evaluate and optimize the models based on user feedback

Risk Management

1. Data risks

Risk: Incomplete or inaccurate data due to errors in data collection or processing

Mitigation: Establish a data quality assurance plan to ensure data is collected and processed accurately. Validate and verify data at each stage of the project.

2. Model risks

Risk: Poor performance or incorrect predictions due to inadequate or flawed modeling techniques

Mitigation: Use established machine learning and natural language processing techniques. Test and validate the models using different datasets and performance metrics.

3. Business risks

Risk: Market changes or external factors that may affect cryptocurrency prices and accuracy of predictions

4. Mitigation: Monitor and adjust the models based on market trends and changes. Continuously evaluate the models and update them with new data and improved algorithms.

5. Security risks

Risk: Data breaches or unauthorized access to sensitive data

Mitigation: Implement strong data security measures, including data encryption, user authentication, and access controls.

6. Communication risks

Risk: Miscommunication or misunderstandings with stakeholders or team members

Mitigation: Establish regular communication channels with stakeholders and team members. Provide clear and concise project updates and progress reports. Ensure that all stakeholders are aware of the project scope and objectives.

7. Resource risks

Risk: Lack of resources or unexpected resource constraints

Mitigation: Develop a resource plan that identifies and allocates necessary resources,

including personnel, equipment, and funding. Continuously monitor resource usage and adjust the plan as needed.

Throughout the project, regular risk assessments should be performed to identify new risks and adjust the risk management plan accordingly. All risks should be documented and communicated to stakeholders and team members.

Communication Plan

1. The project manager will communicate regularly with the development team to ensure that the project is progressing according to the timeline and budget.
2. Regular updates will be provided to stakeholders, including the client and senior management.
3. Communication channels will include email, project management tools, and face-to-face meetings.

Project Deliverables

1. A fully functional e-commerce store with integrated augmented reality
2. User documentation and training materials
3. Quality assurance reports
4. Project closure report

4.5 Transition/ Software to Operations Plan

The transition/software to operations plan for the project of predicting cryptocurrency prices using sentiment analysis of Twitter data involves the following steps:

1)Deployment:

The trained machine learning model and the preprocessing scripts will be deployed to a cloud-based computing infrastructure. The deployment process will involve testing the system's functionality, performance, and scalability.

2)Testing:

The system will undergo rigorous testing to ensure that it meets the requirements and specifications. Testing will include unit testing, integration testing, and system testing.

3)Monitoring:

The system will be continuously monitored for its performance, security, and data quality. Monitoring will involve setting up alert systems and logs to detect and resolve issues quickly.

4)Maintenance:

The system will require regular maintenance, including software upgrades, bug fixes, and data updates. Maintenance will ensure that the system remains reliable, secure, and up-to-date.

5)Documentation:

The project documentation will include technical manuals, user guides, and training materials. The documentation will be regularly updated to reflect changes in the system's functionality and technology.

6)Training:

Users and stakeholders will receive training on how to use the system, interpret the results, and make informed decisions based on the sentiment analysis of social media data.

7)Feedback and Improvements:

The system will be open to feedback from users and stakeholders to identify areas for improvement. The feedback will be used to inform future upgrades and enhancements.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 Development and Deployment Setup

The implementation plan for the impact on Cryptocurrency prices due to Twitter will involve the following steps:

Data collection: Data collection involves creating a Python script in vs code for scraping Twitter for the relevant dataset. For this, we use the Snscape module.

The given code is an implementation of an ETL (Extract, Transform, Load) process for Twitter data related to the hashtag '#Bitcoin'. This process involves collecting, processing, and exporting data from various sources to a data warehouse or database for further analysis.

The code imports the necessary modules such as snscape, pandas, s3fs and datetime, and defines a function called 'run_twitter_etl'. This function initializes a TwitterSearchScraper object that searches for tweets containing the '#Bitcoin' hashtag. The scraper iterates through the results, extracts the desired attributes from each tweet, and stores them in a list.

The function then creates a pandas dataframe called 'tweet_df' that stores the extracted data in columns named 'user_name', 'date', 'likes', 'retweets', and 'content'. Finally, it exports this dataframe to a CSV file named 'Bitcoindaily3.csv' stored in an AWS S3 bucket using s3fs.

The code also sets a limit of 1000 tweets by using a break statement within the for loop. This ensures that only a limited number of tweets are extracted, which can help reduce processing time and storage space.

Overall, the code provides a basic implementation of ETL processes for Twitter data and can be further optimized or customized for specific data requirements.

```

1 import snsrape.modules.twitter as sntwitter
2 import pandas as pd
3 import s3fs
4 from datetime import datetime
5 def run_twitter_etl():
6     scraper = sntwitter.TwitterSearchScraper("#Bitcoin")
7     tweets = []
8     for i,tweet in enumerate(scraper.get_items()):
9         data=[
10             tweet.user.username, # type: ignore
11             tweet.date, # type: ignore
12             tweet.likeCount, # type: ignore
13             tweet.retweetCount, # type: ignore
14             tweet.rawContent # type: ignore
15         ]
16         tweets.append(data)
17         if i>1000:
18             break
19     tweet_df=pd.DataFrame(tweets,columns=['user_name','date','likes','retweets','content'])
20
21
22     tweet_df.to_csv(r's3://rahul-airflow-project/Bitcoindaily3.csv')
23

```

Fig 5.1 Data collection script

Name	Date modified	Type	Size
Bitcoindaily	26-03-2023 19:14	Comma Separated...	218 KB
Bitcoindaily1	26-03-2023 19:18	Comma Separated...	215 KB
Bitcoindaily2	26-03-2023 19:25	Comma Separated...	210 KB
Documents - Shortcut	29-03-2023 22:09	Shortcut	1 KB

Fig 5.2 Files

Storing in S3 and importing the generated dataset: After generating twitter dataset , we store the dataset in Amazon s3 and we import the dataset for Jupyter note book

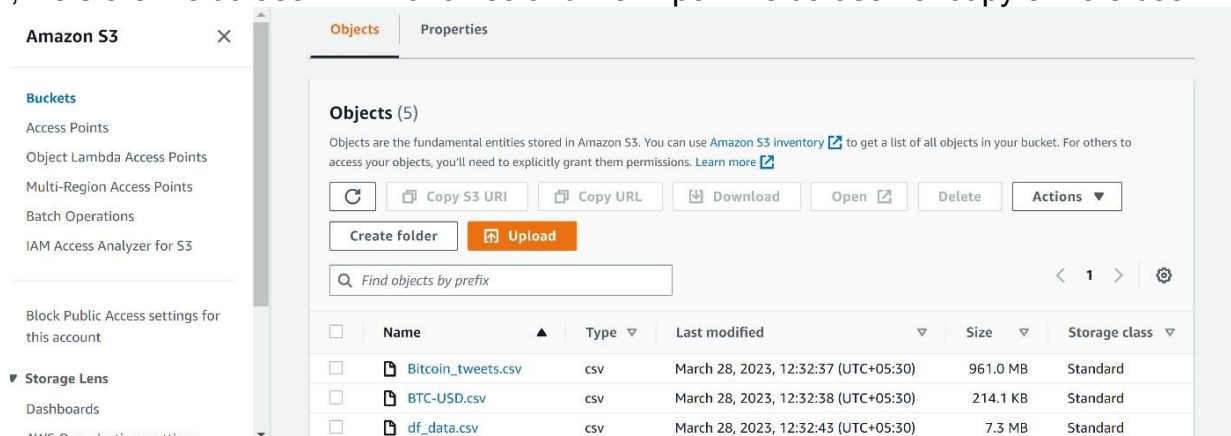


Fig 5.3 Amazon s3

Importing necessary Libraries: The augmented reality feature will be developed using Unity, which will allow users to visualize how products will look in their home or other environments.

```
from time import sleep
import json
import pandas as pd
import io
import re
import numpy as np
from tqdm import tqdm
import pandas as pd
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from tqdm import trange, tqdm_notebook, tqdm

from sklearn import preprocessing
import matplotlib.pyplot as plt
```

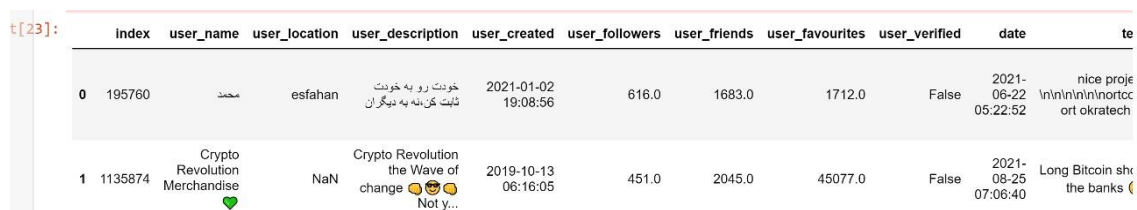
Fig 5.4 Importing libraries

```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import tensorflow.keras.layers as Layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.losses import SparseCategoricalCrossentropy
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Dense, Dropout, Embedding, LSTM, Conv1D, GlobalMaxPooling1D, Bidirectional, SpatialDropout1D
from tensorflow.keras.models import load_model

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

Fig 5.5 Importing machine learning libraries

Data cleaning: we now clean tweets dataset and create a new csv file where we store cleaned dataset. This code reads a data frame `df_raw`, sorts it by the 'date' column, and then randomly samples 1% of the rows using `sample()`. It then resets the index of this subset data frame (`dd`) and iterates through each row of the 'text' column of `dd` using the `tqdm` function to show a progress bar. In each iteration, it cleans the text by removing hashtags, URLs, and usernames using regular expressions (`re`). Finally, it stores the cleaned text back into the 'text' column of `dd`. The last line of code appears to be commented out, but it might be intended to write the cleaned tweets to a file.



	index	user_name	user_location	user_description	user_created	user_followers	user_friends	user_favourites	user_verified	date	text
0	195760	محمد	esfahan	خودت رو به خودت ثابت کن، نه به دیگران	2021-01-02 19:08:56	616.0	1683.0	1712.0	False	2021-06-22 05:22:52	nice proje ort okratech
1	1135874	Crypto Revolution Merchandise ♥	NaN	Crypto Revolution the Wave of change 🌊🌊🌊 Not y...	2019-10-13 06:16:05	451.0	2045.0	45077.0	False	2021-08-25 07:06:40	Long Bitcoin sh the banks 📈

Fig 5.6 Cleaned dataset

Correlation: Importing prices of crypto currency dataset and finding correlation between datasets. This code calculates and plots cross-correlations between two time series datasets: tweets grouped and `crypto_usd_grouped` using three different correlation methods: Pearson, Kendall, and Spearman. It does so by iterating over a range of lags (-20 to 19) and calling the `crosscorr()` function for each lag and correlation method. The resulting correlation values are stored in a list `xcov` for each method. Then, it plots the cross-correlation for each method using Matplotlib's `plt.plot()` function, with lag on the x-axis and correlation on the y-axis. It also sets the title, x-label, and y-label for each plot before displaying it using `plt.show()`.

```
In [28]: xcov = [crosscorr(tweets_grouped, crypto_usd_grouped, lag=i, method="pearson") for i in range(-20,20)]
plt.plot(range(-20,20), xcov)
plt.title("pearson cross-correlation")
plt.xlabel("lag")
plt.ylabel("correlation")
plt.show()

xcov = [crosscorr(tweets_grouped, crypto_usd_grouped, lag=i, method="kendall") for i in range(-20,20)]
plt.plot(range(-20,20), xcov)
plt.title("kendall cross-correlation")
plt.xlabel("lag")
plt.ylabel("correlation")
plt.show()

xcov = [crosscorr(tweets_grouped, crypto_usd_grouped, lag=i, method="spearman") for i in range(-20,20)]
plt.plot(range(-20,20), xcov)
plt.title("spearman cross-correlation")
plt.xlabel("lag")
plt.ylabel("correlation")
plt.show()
```

Fig 5.7 Correlation code

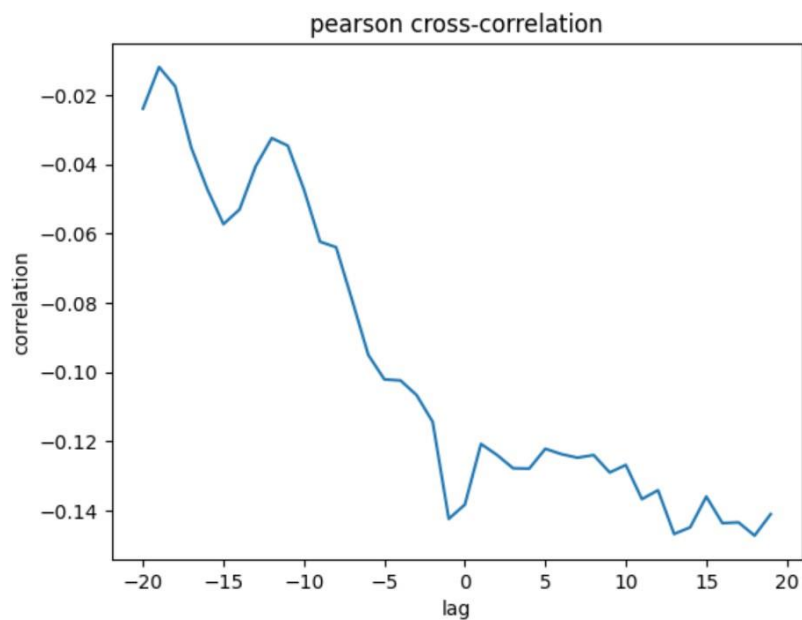


Fig 5.8 Correlation graph

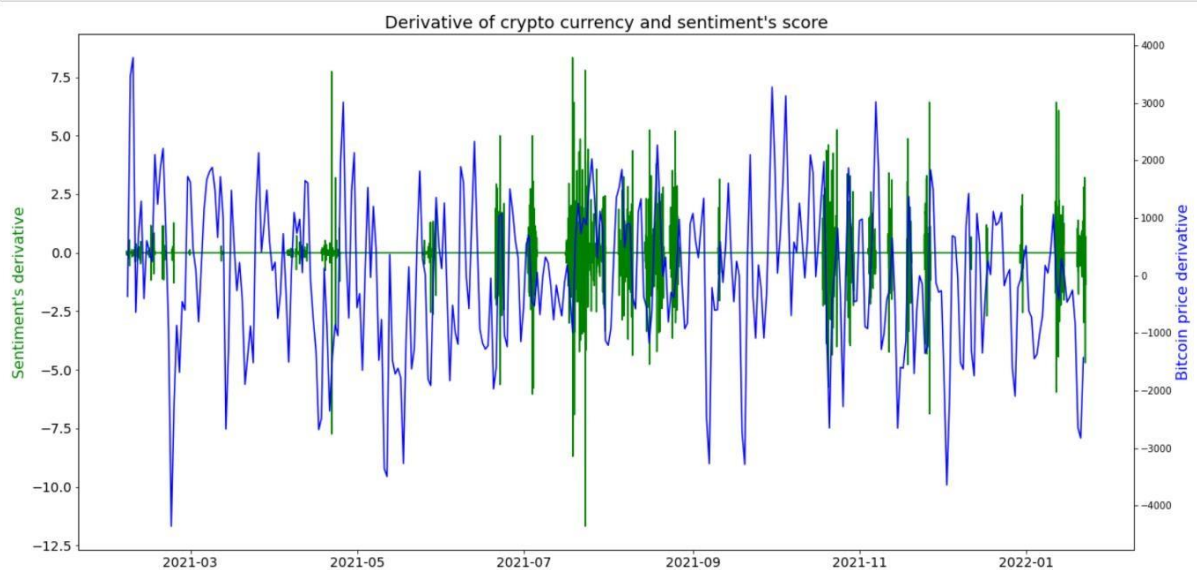


Fig 5.9 Prices movement wrt sentiment score

Calculating Sentiment score:

The first two functions, `getSubjectivity(tweet)` and `getPolarity(tweet)`, are identical to those described in the previous question. They use the `TextBlob` library to analyze the sentiment of a tweet and return the subjectivity and polarity scores as float values between 0-1 and -1-1, respectively.

The `crypto_price_cate(score)` function takes in a sentiment score and categorizes it as either "negative", "neutral", or "positive" based on a threshold of 1. If the score is less than 1, it is considered negative, if it is equal to 1, it is considered neutral, and if it is greater than 1, it is considered positive.

The `observe_period(period)` function takes in a time period as input and calculates the price ratio of the cryptocurrency (represented by the `crypto_usd_grouped` variable) over that period. It then applies the `crypto_price_cate()` function to categorize the sentiment of the price change over that period. The resulting sentiment categories are returned as a pandas series.

The `time_sentiment = observe_period(7)` line calculates the sentiment of the cryptocurrency price change over a period of 7 days and stores it in a pandas series called `time_sentiment`.

The next line, `df['crypto_sentiment'] = df.date_clean.apply(lambda x: time_sentiment[x] if x in time_sentiment else np.nan)`, assigns the sentiment categories from the `time_sentiment` series to each tweet in the dataset based on the date of the tweet. If the date of a tweet is not in the `time_sentiment` series, it is assigned a value of `np.nan`.

Finally, sentiment analysis is performed on each tweet using the `getSubjectivity()` and `getPolarity()` functions, and the resulting subjectivity and polarity scores are added as new columns to the dataframe `df`. The function `head()` is used to display the first few rows of the resulting dataframe.

```
In [35]: def getSubjectivity(tweet):
        return TextBlob(tweet).sentiment.subjectivity

        def getPolarity(tweet):
            return TextBlob(tweet).sentiment.polarity

In [36]: def crypto_price_cate(score):
        if score < 1:
            return 'negative'
        elif score == 1:
            return 'neutral'
        else:
            return 'positive'

        def observe_period(period):
            res = crypto_usd_grouped.shift(period)/crypto_usd_grouped
            res = res.apply(crypto_price_cate)
            return res

        time_sentiment = observe_period(7) # compare price ratio in 7 days. price_7_days_later/ price_now
        df['crypto_sentiment'] = df.date_clean.apply(lambda x: time_sentiment[x] if x in time_sentiment else np.nan)
```

Fig 5.10 Calculating sentiment scores

	tweets	cleaned_tweets	date_clean	crypto_sentiment	subjectivity	polarity
0	nice project \n\n\n\n\n\n\n\nortcoin ort okratech ...	nice project ortcoin ort okratech bitcoin aird...	2021-06-22	positive	1.00	0.600
1	Long Bitcoin short the banks 🤔	Long Bitcoin short bank	2021-08-25	negative	0.35	-0.025
2	Top Trending Cryptocurrency Post - DOGECOIN Se...	Top Trending Cryptocurrency Post DOGECOIN Sell...	2021-07-02	negative	0.40	0.250
3	Can one expect another wave of BTC's decline s...	Can one expect another wave BTC decline soon v...	2021-07-24	negative	0.00	0.000
4	We will see...\n\n\n\n\n\n\n\nbitcoin btc bnb band bake \$btc...	We see bitcoin btc bnb band bake btc dCc xZ DP	2021-05-29	positive	0.00	0.000

Fig 5.11 Resultant sentiment score data frame

Data processing:

This code is used to clean and pre-process text data. It begins by importing the necessary libraries for natural language processing (NLP) - nltk. Then, it downloads the omw (Open Multilingual WordNet) dataset version 1.4.

The function `cleaning(data)` takes the raw text data as input and returns cleaned and pre-processed text data. The pre-processing involves the following steps:

1) Removing URLs: The first step is to remove the URLs from the text. The regular expression `r'http\S+'` is used to match any character string starting with `http` and followed by any non-whitespace characters until the end of the string, which is then replaced with a space.

2) Removing Hashtags: The next step is to remove any hashtags in the text using the regular expression `r'#\w+'`.

3) Removing Mentions and Non-English Characters: The third step is to

remove any mentions in the text using the regular expression `r'@\w+'`. The `re.sub()` function is used to substitute matches of the regular expression with a space. Any non-alphabetic characters are also removed using the regular expression `'[^\w]+'`.

4)Tokenization: The text is then tokenized using the Tweet Tokenizer function from Nltk. The `Tweet Tokenizer()` function is used to tokenize the text data.

5)Removing Punctuation: The next step is to remove any punctuation in the text. The `is alpha()` method is used to check if each token is alphabetic, and only alphabetic tokens are kept.

6)Removing Stopwords: Stopwords, such as "a", "the", and "in", are commonly occurring words that do not add much meaning to the text. The code removes stopwords using a predefined list of stopwords in nltk.

7)Lemmatization: Lemmatization is the process of reducing words to their base form (lemma). The `WordNetLemmatizer` from nltk is used to lemmatize the text data.

8)Joining: Finally, the cleaned tokens are joined together to form a cleaned string of text data.

Overall, this code cleans and pre-processes text data by removing URLs, hashtags, mentions, non-alphabetic characters, punctuation, stop words, and then lemmatizes the remaining tokens to obtain a cleaned string of text data. This process is important in natural language processing tasks such as sentiment analysis and text classification.


```
In [32]: M import nltk
from nltk.stem.wordnet import WordNetLemmatizer

nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('punkt')
stop_words = nltk.corpus.stopwords.words(['english'])

print(stop_words)

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'on', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

Fig 5.12 Importing NLTK packages

```
from nltk.tokenize import TweetTokenizer
from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()

def cleaning(data):
    #remove urls
    tweet_without_url = re.sub(r'http\S+', ' ', data)

    #remove hashtags
    tweet_without_hashtag = re.sub(r'#\w+', ' ', tweet_without_url)

    #3. Remove mentions and characters that not in the English alphabets
    tweet_without_mentions = re.sub(r'@\w+', ' ', tweet_without_hashtag)
    precleaned_tweet = re.sub('[^A-Za-z]+', ' ', tweet_without_mentions)

    #2. Tokenize
    tweet_tokens = TweetTokenizer().tokenize(precleaned_tweet)

    #3. Remove Puncs
    tokens_without_punc = [w for w in tweet_tokens if w.isalpha()]

    #4. Removing Stopwords
    tokens_without_sw = [t for t in tokens_without_punc if t not in stop_words]

    #5. Lemma
    text_cleaned = [lem.lemmatize(t) for t in tokens_without_sw]

    #6. Joining
    return " ".join(text_cleaned)
```

Fig 5.13 Cleaning dataset

Model Building: we clean the dataset and calculate sentiment score and we build convolution neural network The code prepares the data for text classification and then builds a Convolutional Neural Network (CNN) followed by a Long Short-Term Memory (LSTM) layer.

Initially, the input data is cleaned tweets, and the output variable is the sentiment of the tweet. The code uses one-hot encoding to represent the target variable y . The number of classes in the target variable is determined using the `n unique ()` method.

The code then splits the data into training and testing sets using `train_test_split()` function. The training set is used to train the deep learning model, and the testing set is used to evaluate the performance of the model.

The input data `X` is tokenized using the `Tokenizer` class from Keras. Tokenization involves assigning a unique integer to each word in the text corpus. The number of unique words is set to 20000 using the `max_features` parameter. The `texts_to_sequences()` function converts the texts into sequences of integers.

The sequences of integers are then padded to make them equal in length using `sequence.pad_sequences()` function. The length of the sequence is set to 30 using the `max_words` parameter. Padding is necessary as the model requires the inputs to be of equal length.

The model is built using the `Sequential` class from Keras. The model has four layers: an embedding layer, two convolutional layers, an LSTM layer, and a dense output layer. The embedding layer learns the word embeddings from the input data. The convolutional layers extract features from the input data, followed by max-pooling layers to reduce the dimensionality of the features. The LSTM layer captures the temporal dependencies between the input data. The output layer is a dense layer with softmax activation that outputs the predicted probabilities of each sentiment class.

The model is compiled using `categorical_crossentropy` loss function and the Adam optimizer. The model is then trained for 10 epochs with a batch size of 128. The model summary is printed using the `summary()` function. Finally, the accuracy of the model is evaluated using the `metrics=['accuracy']` parameter.

```

nltk.download('omw-1.4')
from nltk.tokenize import TweetTokenizer
from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()

def cleaning(data):
    #remove urls
    tweet_without_url = re.sub(r'http\S+', ' ', data)

    #remove hashtags
    tweet_without_hashtag = re.sub(r'#\w+', ' ', tweet_without_url)

    #3. Remove mentions and characters that not in the English alphabets
    tweet_without_mentions = re.sub(r'\@\w+', ' ', tweet_without_hashtag)
    precleaned_tweet = re.sub(r'[^A-Za-z]+', ' ', tweet_without_mentions)

    #2. Tokenize
    tweet_tokens = TweetTokenizer().tokenize(precleaned_tweet)

    #3. Remove Puncs
    tokens_without_punc = [w for w in tweet_tokens if w.isalpha()]

    #4. Removing Stopwords
    tokens_without_sw = [t for t in tokens_without_punc if t not in stop_words]

    #5. Lemma
    text_cleaned = [lem.lemmatize(t) for t in tokens_without_sw]

    #6. Joining
    return " ".join(text_cleaned)

```

Fig 5.14 Cleaning dataset

```

]: X = df['cleaned_tweets']
   y = pd.get_dummies(df['sentiment']).values
   num_classes = df['sentiment'].nunique()

]: seed = 38 # fix random seed for reproducibility
   np.random.seed(seed)

   X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                       test_size=0.2,
                                                       stratify=y,
                                                       random_state=seed)
   print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

(17936,) (4484,) (17936, 3) (4484, 3)

]: max_features = 20000
   tokenizer = Tokenizer(num_words=max_features)
   tokenizer.fit_on_texts(list(X_train))
   X_train = tokenizer.texts_to_sequences(X_train)
   X_test = tokenizer.texts_to_sequences(X_test)

]: from tensorflow.keras.preprocessing import sequence
   max_words = 30
   X_train = sequence.pad_sequences(X_train, maxlen=max_words)
   X_test = sequence.pad_sequences(X_test, maxlen=max_words)
   print(X_train.shape, X_test.shape)

(17936, 30) (4484, 30)

```

Fig 5.15 Splitting dataset for training

```
In [48]: tf.keras.utils.plot_model(model, show_shapes=True)
```

You must install pydot ('pip install pydot') and install graphviz (see instructions at <https://graphviz.gitlab.io/download/>) for plot_model to work.

```
In [49]: history = model.fit(X_train, y_train, validation_data=(X_test, y_test),
                             epochs=epochs, batch_size=batch_size, verbose=2)
```

```

Epoch 1/10
141/141 - 27s - loss: 0.6510 - accuracy: 0.7300 - val_loss: 0.3959 - val_accuracy: 0.8559 - 27s/epoch - 190ms/step
Epoch 2/10
141/141 - 15s - loss: 0.2855 - accuracy: 0.9004 - val_loss: 0.3498 - val_accuracy: 0.8880 - 15s/epoch - 108ms/step
Epoch 3/10
141/141 - 10s - loss: 0.1266 - accuracy: 0.9588 - val_loss: 0.3316 - val_accuracy: 0.9057 - 10s/epoch - 70ms/step
Epoch 4/10
141/141 - 11s - loss: 0.0498 - accuracy: 0.9848 - val_loss: 0.3137 - val_accuracy: 0.9146 - 11s/epoch - 75ms/step
Epoch 5/10
141/141 - 14s - loss: 0.0255 - accuracy: 0.9936 - val_loss: 0.3703 - val_accuracy: 0.9228 - 14s/epoch - 103ms/step
Epoch 6/10
141/141 - 7s - loss: 0.0162 - accuracy: 0.9961 - val_loss: 0.3652 - val_accuracy: 0.9199 - 7s/epoch - 47ms/step
Epoch 7/10
141/141 - 14s - loss: 0.0130 - accuracy: 0.9969 - val_loss: 0.3982 - val_accuracy: 0.9231 - 14s/epoch - 97ms/step
Epoch 8/10
141/141 - 14s - loss: 0.0094 - accuracy: 0.9977 - val_loss: 0.4070 - val_accuracy: 0.9215 - 14s/epoch - 101ms/step
Epoch 9/10
141/141 - 6s - loss: 0.0071 - accuracy: 0.9982 - val_loss: 0.4206 - val_accuracy: 0.9237 - 6s/epoch - 44ms/step
Epoch 10/10
141/141 - 14s - loss: 0.0068 - accuracy: 0.9985 - val_loss: 0.4469 - val_accuracy: 0.9248 - 14s/epoch - 100ms/step

```

Fig 5.16 Training model

```

1 [50]: In def plot_training_hist(history):
        '''Function to plot history for accuracy and loss'''

        fig, ax = plt.subplots(1,2, figsize=(10,4))
        # first plot
        ax[0].plot(history.history['accuracy'])
        ax[0].plot(history.history['val_accuracy'])
        ax[0].set_title('Model Accuracy')
        ax[0].set_xlabel('epoch')
        ax[0].set_ylabel('accuracy')
        ax[0].legend(['train', 'validation'], loc='best')

        # second plot
        ax[1].plot(history.history['loss'])
        ax[1].plot(history.history['val_loss'])
        ax[1].set_title('Model Loss')
        ax[1].set_xlabel('epoch')
        ax[1].set_ylabel('loss')
        ax[1].legend(['train', 'validation'], loc='best')

        plot_training_hist(history)

```

Fig 5.17 Plotting the models

5.2 Algorithms

The project of predicting cryptocurrency prices using sentiment analysis of Twitter data involves the use of several machine learning algorithms, including Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN), and Vader sentiment analysis.

LSTM:

LSTM is a type of recurrent neural network (RNN) that is well-suited for sequential data such as text. LSTM can capture long-term dependencies in the data by allowing information to be stored in memory cells for an extended period. In the project, LSTM can be used to predict cryptocurrency prices by training the model on the historical cryptocurrency price data and corresponding sentiment scores.

CNN:

CNN is a type of neural network that is commonly used for image processing tasks. However, CNN can also be used for natural language processing tasks such as sentiment analysis. In the project, CNN can be used to extract features from the text data by applying filters to identify important patterns and structures.

Vader sentiment analysis:

Vader (Valence Aware Dictionary and sEntiment Reasoner) is a rule-based sentiment analysis tool that is commonly used for social media analysis. Vader uses a combination of lexical and syntactical analysis to determine the sentiment of the text. In the project, Vader sentiment analysis can be used to extract sentiment scores from the Twitter data related to cryptocurrency.

The overall approach of the project involves combining the sentiment analysis of the Twitter data with the historical cryptocurrency price data to train a machine learning model such as LSTM or CNN. The model will then be used to predict the cryptocurrency prices based on the sentiment analysis of the Twitter data. The Vader sentiment analysis tool can be used to extract sentiment scores from the Twitter data, which can be used as input features for the machine learning model.

5.3 Testing

The testing methodology for the project of predicting cryptocurrency prices using sentiment analysis of Twitter data will involve the following steps:

1) Data Pre-processing Testing:

The data pre-processing step involves cleaning, normalizing, and transforming the data to prepare it for analysis. Testing this step involves verifying that the data is properly cleaned and normalized, and that the data transformation is accurate and consistent.

2) Training Data Testing:

The machine learning model will be trained on historical cryptocurrency prices and corresponding sentiment scores. The training data will be tested to ensure that it is properly labeled, balanced, and representative of the overall population.

3) Model Performance Testing:

The trained machine learning model will be evaluated for its performance in predicting cryptocurrency prices based on the sentiment analysis of Twitter data. Testing this step involves comparing the predicted prices with actual prices to evaluate the accuracy of the model. Metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) will be used to assess the model's performance.

4) Model Generalization Testing:

The model's ability to generalize to new data will be tested by evaluating its performance on a hold-out test set that is not used during training. Testing this step involves verifying that the model performs well on new, unseen data.

5) Robustness Testing:

The system will be tested for its robustness to noise and outliers in the data. This involves adding random noise or outliers to the data to evaluate how the model performs under such conditions.

6) Usability Testing:

The system's usability will be tested by evaluating how easy it is for users to interact with the system, interpret the results, and make informed decisions based on the sentiment analysis of social media data.

6. Results and Discussions

Final Results

The function creates a figure with two subplots, the first one is for plotting the accuracy and the second one is for plotting the loss. The training accuracy and validation accuracy are plotted against the number of epochs on the x-axis and the accuracy on the y-axis. Similarly, the training loss and validation loss are plotted against the number of epochs on the x-axis and the loss on the y-axis.

The function uses `history.history` dictionary, which contains the training and validation accuracy and loss at each epoch. The `plot` method is used to plot the curves for both accuracy and loss for both the training and validation datasets. Finally, the `set_title`, `set_xlabel`, `set_ylabel`, and `legend` methods are used to set the title, x-label, y-label, and legend for the plots.

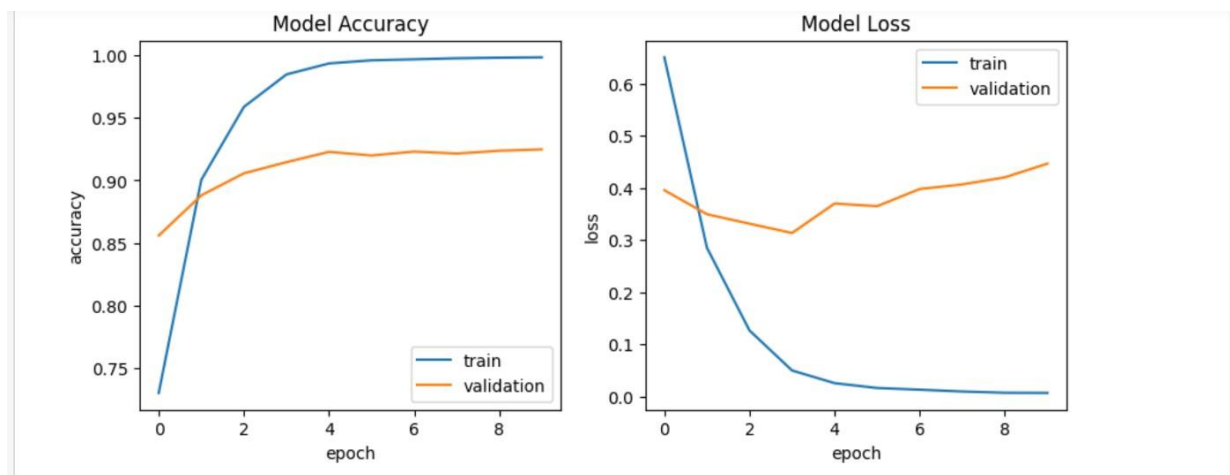


Fig 6.1 Model Results

This code below(Fig-18) predicts the class of the test set using the trained model and prints out the accuracy and classification report. `model.predict(X_test)` returns the predicted probabilities for each class. The `argmax` function is used to return the index of the class with the highest probability for each sample. These predicted class labels are compared with the true class labels from `y_test` using the `accuracy_score` function to

calculate the accuracy of the model. The `classification_report` function is used to generate a report that contains various metrics such as precision, recall, and f1-score for each class, as well as the overall accuracy, weighted average metrics, and support.

```
In [51]: # predict class with test set
y_pred_test = np.argmax(model.predict(X_test), axis=1)
print('Accuracy:\t{:0.1f}%'.format(accuracy_score(np.argmax(y_test,axis=1),y_pred_test)*100))
print(classification_report(np.argmax(y_test,axis=1), y_pred_test))
```

141/141 [=====] - 1s 5ms/step
Accuracy: 92.5%

	precision	recall	f1-score	support
0	0.84	0.72	0.77	477
1	0.94	0.94	0.94	1779
2	0.93	0.95	0.94	2228
accuracy			0.92	4484
macro avg	0.90	0.87	0.89	4484
weighted avg	0.92	0.92	0.92	4484

Fig 6.2 F1 Score

The code below(Fig-19) defines a function called `plot_confusion_matrix` that takes in a trained model, the test set data (`X_test` and `y_test`). The function first defines a list of sentiment classes ('Negative', 'Neutral', 'Positive').

The function then uses the model to make predictions on the test set (`y_pred = model.predict(X_test)`), computes the confusion matrix using the `confusion_matrix` function from `scikit-learn` (`cm = confusion_matrix(np.argmax(y_pred, axis=1), np.argmax(np.array(y_test), axis=1))`), and prints the counts of actual and predicted labels for each sentiment class.

Finally, the function plots the confusion matrix using `seaborn`'s heatmap, with actual labels on the x-axis, predicted labels on the y-axis, and the counts of samples for each class in the corresponding cell. The plot helps visualize the performance of the model in predicting each sentiment class.

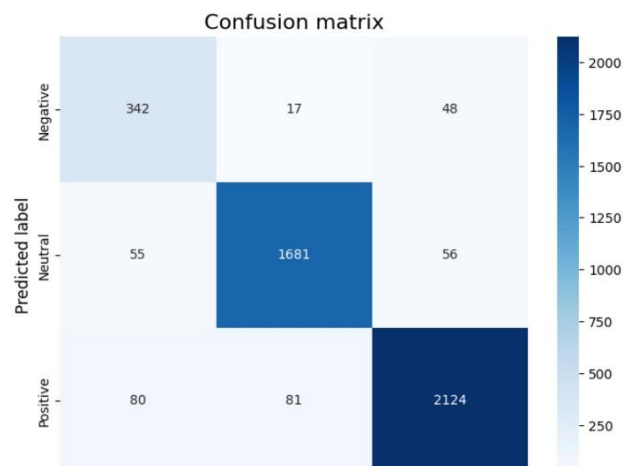


Fig 6.3 Confusion matrix

CHAPTER 7

Conclusion

7.1 conclusion

In conclusion, sentiment analysis has shown promising results in predicting cryptocurrency prices. By analyzing public sentiment on social media platforms and news articles, we can gain insights into how people feel about a particular cryptocurrency and use that information to forecast its price movements.

However, it's important to note that sentiment analysis is not a foolproof method for predicting cryptocurrency prices. The market is highly volatile, and prices can be influenced by a multitude of factors beyond just public sentiment. As such, sentiment analysis should be used in conjunction with other analysis techniques and not relied upon as the sole indicator for making investment decisions.

Overall, sentiment analysis can provide valuable insights into the market's collective sentiment towards a particular cryptocurrency, and as the field of artificial intelligence continues to advance, we may see even more accurate and reliable sentiment analysis tools in the future.

7.2 Future Work

After completing the project of predicting Bitcoin prices using sentiment analysis, the next logical step would be to create a data pipeline that automatically ingests data, performs sentiment analysis using Spark, and displays the results in a dashboard. This pipeline will help in automating the data analysis process, and provide near-real-time insights for better decision-making.

To build this data pipeline, the first step is to choose a streaming platform for data ingestion. Kafka is one of the most popular open-source streaming platforms that can be used for this purpose. Kafka can handle high volume and high velocity data ingestion and has an efficient architecture that supports scalable, fault-tolerant processing of real-time data streams.

Next, a workflow management system such as Airflow can be used to automate the pipeline. Airflow provides a way to define, execute, and monitor complex workflows that are composed of multiple tasks. Using Airflow, we can set up the pipeline to regularly fetch data from the sources, transform and pre-process it, and send it to Spark for sentiment analysis.

In the data pipeline, Spark can be used to perform sentiment analysis on the text data. Spark is a distributed computing framework that provides efficient processing of large datasets. With Spark, we can leverage the power of distributed computing to perform complex data analysis tasks, such as sentiment analysis, on large volumes of data. Spark provides a variety of machine learning libraries, including natural language processing (NLP) libraries, that can be used for sentiment analysis.

Once the data has been processed and analyzed, the results can be visualized in a dashboard. A dashboard can provide real-time insights on the sentiment of the Bitcoin market, which can be useful for making investment decisions. The dashboard can be built using a tool like Tableau, which allows for the creation of interactive visualizations that can be customized to fit the specific needs of the user.

The data pipeline can be broken down into the following steps:

Data Ingestion: Data will be ingested from various sources, including social media platforms, news websites, and other online sources. Kafka can be used to handle high volumes of data ingestion in real-time.

Data Transformation: Once the data has been ingested, it will need to be cleaned and transformed for analysis. This step may involve removing irrelevant data, normalizing data, and performing other data pre-processing tasks.

Sentiment Analysis: The transformed data will be sent to Spark for sentiment analysis. Spark will use machine learning algorithms to analyze

the data and determine the sentiment of the Bitcoin market.

Visualization: The results of the sentiment analysis will be visualized in a dashboard. Tableau can be used to create interactive visualizations that provide real-time insights into the sentiment of the Bitcoin market.

The data pipeline can be automated using Airflow. Airflow provides a way to schedule and execute tasks in a workflow. Tasks can be defined as DAGs (Directed Acyclic Graphs) that specify the dependencies between tasks. For example, a DAG can be defined to fetch data from social media platforms and news websites, preprocess the data, send it to Spark for sentiment analysis, and visualize the results in a dashboard.

In conclusion, building a data pipeline that automatically ingests data, performs sentiment analysis using Spark, and displays the results in a dashboard can provide real-time insights into the sentiment of the Bitcoin market. The pipeline can be built using Kafka for data ingestion, Airflow for workflow management, Spark for sentiment analysis, and Tableau for visualization. This pipeline can be automated to provide near-real-time insights that can be used to make better investment decisions.

7.3 Research Issues

1)Dataset Selection: One of the critical research issues in the project is the selection of the appropriate dataset for training and testing the machine learning model. The dataset needs to be diverse, representative, and unbiased to ensure that the model can generalize well to new data.

2)Feature Selection: Another research issue is the selection of the relevant features from the Twitter data that are most informative for predicting cryptocurrency prices. The choice of features may vary depending on the cryptocurrency and the sentiment analysis tool used.

3)Sentiment Analysis Accuracy: The accuracy of sentiment analysis tools such as Vader may be limited by the complexity of the language used in social media data, which may contain sarcasm, irony, and other forms of figurative language. Research is needed to develop more accurate and robust sentiment analysis tools for social media data.

4)Model Interpretability: The machine learning model used in the project may be opaque and difficult to interpret, making it challenging for users to understand how the model arrives at its predictions. Research is needed to develop more interpretable models that can provide insights into the factors driving cryptocurrency prices.

5)Generalization to New Cryptocurrencies: The project may be limited to a few cryptocurrencies that have sufficient Twitter data available for sentiment analysis. Research is needed to develop methods for generalizing the approach to new cryptocurrencies with limited data.

6)Market Volatility: Cryptocurrency markets are notoriously volatile and subject to sudden changes in price. Research is needed to develop methods for adapting the machine learning model to changing market conditions and incorporating real-time data into the prediction model.

7.4 Implementation Issues

The implementation of the project may face several issues that need to be addressed to ensure the successful deployment of the system. Some of these implementation issues include:

1) Scalability: The system may need to handle a large volume of data in real-time, which may require scalable infrastructure such as cloud computing to handle the processing and storage requirements.

2) Data Quality: The quality of the Twitter data used for sentiment analysis may affect the accuracy of the prediction model. The system needs to ensure that the data is properly cleaned, normalized, and filtered to remove noise and irrelevant information.

3) Model Training: The machine learning model used for prediction may require significant computational resources and time for training. The system needs to ensure that the training process is efficient and that the model is trained on representative and balanced data.

4) Real-time Data Integration: The system may require integrating real-time data sources to provide up-to-date sentiment analysis of Twitter data. The system needs to ensure that the data integration is seamless and the real-time data is processed efficiently.

5) Model Interpretability: The machine learning model used for prediction may be complex and difficult to interpret, making it challenging for users to understand the factors driving cryptocurrency prices. The system needs to ensure that the model is interpretable and provides insights into the factors driving the prediction.

6) User Interface Design: The system may require a user-friendly interface that allows users to interact with the system, view the prediction results, and make informed decisions based on the sentiment analysis of social media data. The system needs to ensure that the user interface is intuitive and easy to use.

REFERENCES: -

1. Nakamoto S., "Bitcoin: A Peer-to-Peer Electronic Cash System", 2008
2. F. Mai, Q. Bai, J. Shan, "From Bitcoin to Big Coin: The Impacts of Social Media

- on Bitcoin Performance”, SSRN Electron. J 1-16, January 2015.
3. Karalevicius, Vytautas, N. Degrande, J. De Weerd. "Using sentiment analysis to predict interday Bitcoin price movements." *The Journal of Risk Finance* (2018).
 4. A. Phillip, JSK Chan, S. Peiris. "A new look at Cryptocurrencies." *Economics Letters* 163 (2018): 6-9.
 5. I. Georgoula, D. Pournarakis, C. Bilanakos, "Using time-series and sentiment analysis to detect the determinants of bitcoin prices." Available at SSRN 2607167 (2015).
 6. D. Garcia, F. Schweitzer. "Social signals and algorithmic trading of Bitcoin." *Royal Society open science* 2.9 (2015): 150288.
 7. V. Pagolu, K. Challa, G. Panda, "Sentiment Analysis of Twitter Data for Predicting Stock Market Movements",
 8. M. Matta, I. Lunesu, M. Marchesi, "Bitcoin Spread Prediction Using Social and Web Search Media".
 9. E. Stenqvist, J. Lönnö, "Predicting Bitcoin price fluctuation with Twitter sentiment analysis" (2017).

APPENDIX

A. SOURCE CODE

1) Data Gathering

```
import snsrape.modules.twitter as sntwitter
```



```

import pandas as pd
import s3fs
from datetime import datetime
def run_twitter_etl():
    scraper = sntwitter.TwitterSearchScraper("#Bitcoin")
    tweets = []
    for i,tweet in enumerate(scraper.get_items()):
        data=[
            tweet.user.username, # type: ignore
            tweet.date, # type: ignore
            tweet.likeCount, # type: ignore
            tweet.retweetCount, # type: ignore
            tweet.rawContent # type: ignore
        ]
        tweets.append(data)
        if i>1000:
            break

tweet_df=pd.DataFrame(tweets,columns=['user_name','date','likes','retwe
ets','content'])

tweet_df.to_csv(r's3://rahul-airflow-project/Bitcoindaily3.csv')

```

2) Data preprocessing:

```

from time import sleep
import json
import pandas as pd
import io
import re

```

```

import numpy as np
from tqdm import tqdm
import pandas as pd
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from tqdm import trange, tqdm_notebook, tqdm

from sklearn import preprocessing
import matplotlib.pyplot as plt
df_raw = df_raw.sort_values(by = 'date')
dd = df_raw.sample(frac=0.01, replace=False, random_state=1)
dd.reset_index(inplace=True)
for i,s in enumerate(tqdm(dd['text'],position=0, leave=True)):
    text = str(dd.loc[i, 'text'])
    text = text.replace("#", "")
    text = re.sub('https?://(?:[-\w.])((?:%[\da-fA-F]{2}))+', "", text,
flags=re.MULTILINE)
    text = re.sub('@\w+ ', "", text, flags=re.MULTILINE)
    dd.loc[i, 'text'] = text
# f = open(tweets_clean_file, 'a+', encoding='utf-8')
dd.to_csv(r"C:\Users\rahul\OneDrive\Desktop\Datasources1\tweets_clean.csv", header=True, encoding='utf-8',index=False)
analyzer = SentimentIntensityAnalyzer()
compound = []
for i,s in enumerate(tqdm(df_clean['text'],position=0, leave=True)):
    # print(i,s)
    vs = analyzer.polarity_scores(str(s))
    compound.append(vs["compound"])
df_clean["compound"] = compound
df_clean.head(2)

scores = []
for i, s in tqdm(df_clean.iterrows(), total=df_clean.shape[0],position=0,
leave=True):
    try:

```

```

        scores.append(s["compound"] * ((int(s["user_followers"]))) *
((int(s["user_favourites"])+1)/int(s["user_followers"]+1))
*((int(s["is_retweet"])+1)))
    except:
        scores.append(np.nan)
df_clean["score"] = scores
df_clean.head(2)

df_price =
pd.read_csv(r"C:\Users\rahul\OneDrive\Desktop\Datasources1\BTC-
USD.csv")
df_price.Date = pd.to_datetime(df_price.Date)
# df_price.Timestamp = pd.to_datetime(df_price.Timestamp,unit='s')
df_price.head(2)

```

3) Correlation and sentiment score:

```

df_clean = df_clean.drop_duplicates()
tweets = df_clean.copy()
tweets['date'] = pd.to_datetime(tweets['date'],utc=True)
tweets.date = tweets.date.dt.tz_localize(None)
tweets.index = tweets['date']

# tweets_grouped = tweets.groupby(pd.TimeGrouper('1h'))['score'].sum()
tweets_grouped = tweets.resample('1h').sum()

crypto_usd = df_price.copy()
crypto_usd['Date'] = pd.to_datetime(crypto_usd['Date'], unit='s')
crypto_usd.index = crypto_usd['Date']
# crypto_usd['Timestamp'] = pd.to_datetime(crypto_usd['Timestamp'],
unit='s')
# crypto_usd.index = crypto_usd['Timestamp']

# crypto_usd_grouped =
crypto_usd.groupby(pd.TimeGrouper('1h'))['Weighted_Price'].mean()

```

```

crypto_usd_grouped = crypto_usd.resample('D')['Close'].mean()

def crosscorr(datax, datay, lag=0, method="pearson"):
    """ Lag-N cross correlation.
    Parameters
    -----
    lag : int, default 0
    datax, datay : pandas.Series objects of equal length
    Returns
    -----
    crosscorr : float
    """
    return datax.corrwith(datay.shift(lag), method=method)['score']
# xcov = [crosscorr(tweets_grouped, crypto_usd_grouped, lag=i, m
#='pearson' ) for i in range(-20,20)]
# tweets_grouped.corrwith(crypto_usd_grouped,method='pearson')

begginig    =    max(tweets_grouped.index.min().replace(tzinfo=None),
crypto_usd_grouped.index.min())
end         =    min(tweets_grouped.index.max().replace(tzinfo=None),
crypto_usd_grouped.index.max())
tweets_grouped = tweets_grouped[begginig:end]
crypto_usd_grouped = crypto_usd_grouped[begginig:end]

fig, ax1 = plt.subplots(figsize=(20,10))
ax1.set_title("Crypto currency evolution compared to twitter sentiment",
fontsize=18)
ax1.tick_params(labelsize=14)
ax2 = ax1.twinx()
ax1.plot_date(tweets_grouped.index, tweets_grouped, 'g-')
ax2.plot_date(crypto_usd_grouped.index, crypto_usd_grouped, 'b-')

ax1.set_ylabel("Sentiment", color='g', fontsize=16)
ax2.set_ylabel("Bitcoin [$]", color='b', fontsize=16)

```

```
plt.show()
```

```
xcov = [crosscorr(tweets_grouped, crypto_usd_grouped, lag=i,
method="pearson") for i in range(-20,20)]
plt.plot(range(-20,20), xcov)
plt.title("pearson cross-correlation")
plt.xlabel("lag")
plt.ylabel("correlation")
plt.show()
```

```
xcov = [crosscorr(tweets_grouped, crypto_usd_grouped, lag=i,
method="kendall") for i in range(-20,20)]
plt.plot(range(-20,20), xcov)
plt.title("kendall cross-correlation")
plt.xlabel("lag")
plt.ylabel("correlation")
plt.show()
```

```
xcov = [crosscorr(tweets_grouped, crypto_usd_grouped, lag=i,
method="spearman") for i in range(-20,20)]
plt.plot(range(-20,20), xcov)
plt.title("spearman cross-correlation")
plt.xlabel("lag")
plt.ylabel("correlation")
plt.show()
min_max_scaler = preprocessing.StandardScaler()
score_scaled =
min_max_scaler.fit_transform(tweets_grouped['score'].values.reshape(-
1,1))
tweets_grouped['normalized_score'] = score_scaled
# crypto_used_grouped_scaled =
min_max_scaler.fit_transform(crypto_usd_grouped.values.reshape(-1,1))
crypto_used_grouped_scaled = crypto_usd_grouped /
max(crypto_usd_grouped.max(), abs(crypto_usd_grouped.min()))
```

```

# crypto_usd_grouped['normalized_price'] =
crypto_used_grouped_scaled

fig, ax1 = plt.subplots(figsize=(20,10))
ax1.set_title("Normalized Crypto currency evolution compared to
normalized twitter sentiment", fontsize=18)
ax1.tick_params(labelsize=14)

ax2 = ax1.twinx()
ax1.plot_date(tweets_grouped.index,
tweets_grouped['normalized_score'], 'g-')
ax2.plot_date(crypto_usd_grouped.index, crypto_used_grouped_scaled,
'b-')

ax1.set_ylabel("Sentiment", color='g', fontsize=16)
ax2.set_ylabel("Bitcoin normalized", color='b', fontsize=16)
plt.show()

#tweets_grouped.T.corr(crypto_usd_grouped, method='pearson')
#tweets_grouped.T.autocorr(crypto_usd_grouped, lag=20)
xcov = [crosscorr(tweets_grouped, crypto_usd_grouped, lag=i) for i in
range(-20,20)]
plt.plot(range(-20,20), xcov)
plt.title("lag's impact on correlation (normalized)")
plt.xlabel("lag")
plt.ylabel("correlation")
plt.show()

```

4) Data cleaning for model building

```

import nltk
nltk.download('omw-1.4')
from nltk.tokenize import TweetTokenizer
from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()

def cleaning(data):

```

```

#remove urls
tweet_without_url = re.sub(r'http\S+', '', data)

#remove hashtags
tweet_without_hashtag = re.sub(r'#\w+', '', tweet_without_url)

#3. Remove mentions and characters that not in the English alphabets
tweet_without_mentions = re.sub(r'@\w+', '', tweet_without_hashtag)
precleaned_tweet = re.sub('[^A-Za-z]+', '', tweet_without_mentions)

#2. Tokenize
tweet_tokens = TweetTokenizer().tokenize(precleaned_tweet)

#3. Remove Puncs
tokens_without_punc = [w for w in tweet_tokens if w.isalpha()]

#4. Removing Stopwords
tokens_without_sw = [t for t in tokens_without_punc if t not in stop_words]

#5. lemma
text_cleaned = [lem.lemmatize(t) for t in tokens_without_sw]

#6. Joining
return " ".join(text_cleaned)

df['cleaned_tweets'] = df['tweets'].apply(cleaning)
df['date'] = df_clean['date']
df['date_clean'] = pd.to_datetime(df['date']).dt.strftime('%Y-%m-%d')
df.drop(columns='date', inplace=True)
df.head()

def getSubjectivity(tweet):
    return TextBlob(tweet).sentiment.subjectivity

def getPolarity(tweet):
    return TextBlob(tweet).sentiment.polarity

def crypto_price_cate(score):
    if score < 1:
        return 'negative'
    elif score == 1:
        return 'neutral'
    else:
        return 'positive'

def observe_period(period):
    res = crypto_usd_grouped.shift(period)/crypto_usd_grouped
    res = res.apply(crypto_price_cate)
    return res

time_sentiment = observe_period(7) # compare price ratio in 7 days.

```

```
price_7_days_later/ price_now
df['crypto_sentiment'] = df.date_clean.apply(lambda x: time_sentiment[x] if x in
time_sentiment else np.nan)
```

```
def getSentiment(score):
    if score < 0:
        return 'negative'
    elif score == 0:
        return 'neutral'
    else:
        return 'positive'
df['sentiment'] = df['polarity'].apply(getSentiment)
df['target'] = df['sentiment'] == df['crypto_sentiment']
df.head()
df.to_csv(r"C:\Users\rahul\OneDrive\Desktop\Datasources1\df_data.csv")
```

5) Model and Results code

```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import tensorflow.keras.layers as Layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.losses import SparseCategoricalCrossentropy
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Dense, Dropout, Embedding, LSTM, Conv1D,
GlobalMaxPooling1D, Bidirectional, SpatialDropout1D
from tensorflow.keras.models import load_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

```
X = df['cleaned_tweets']
y = pd.get_dummies(df['sentiment']).values
num_classes = df['sentiment'].nunique()
```

```
seed = 38 # fix random seed for reproducibility
np.random.seed(seed)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    stratify=y,
                                                    random_state=seed)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
max_features = 20000
tokenizer = Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(list(X_train))
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)
```

```
from tensorflow.keras.preprocessing import sequence
```



```

max_words = 30
X_train = sequence.pad_sequences(X_train, maxlen=max_words)
X_test = sequence.pad_sequences(X_test, maxlen=max_words)
print(X_train.shape,X_test.shape)

import tensorflow.keras.backend as K
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Embedding,Conv1D,MaxPooling1D,LSTM
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

batch_size = 128
epochs = 10

max_features = 20000
embed_dim = 100

np.random.seed(seed)
K.clear_session()
model = Sequential()
model.add(Embedding(max_features, embed_dim, input_length=X_train.shape[1]))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

history = model.fit(X_train, y_train, validation_data=(X_test, y_test),
                    epochs=epochs, batch_size=batch_size, verbose=2)
def plot_training_hist(history):
    """Function to plot history for accuracy and loss"""

    fig, ax = plt.subplots(1,2, figsize=(10,4))
    # first plot
    ax[0].plot(history.history['accuracy'])
    ax[0].plot(history.history['val_accuracy'])
    ax[0].set_title('Model Accuracy')
    ax[0].set_xlabel('epoch')
    ax[0].set_ylabel('accuracy')
    ax[0].legend(['train', 'validation'], loc='best')
    # second plot
    ax[1].plot(history.history['loss'])
    ax[1].plot(history.history['val_loss'])
    ax[1].set_title('Model Loss')
    ax[1].set_xlabel('epoch')
    ax[1].set_ylabel('loss')
    ax[1].legend(['train', 'validation'], loc='best')

plot_training_hist(history)

```

```

# predict class with test set
y_pred_test = np.argmax(model.predict(X_test), axis=1)
print('Accuracy:\t{0.1f}%'.format(accuracy_score(np.argmax(y_test,axis=1),y_pred_test)*100))
print(classification_report(np.argmax(y_test,axis=1), y_pred_test))

from sklearn.metrics import confusion_matrix
import seaborn as sns
def plot_confusion_matrix(model, X_test, y_test):
    """Function to plot confusion matrix for the passed model and the data"""
    sentiment_classes = ['Negative', 'Neutral', 'Positive']
    # use model to do the prediction
    y_pred = model.predict(X_test)
    # compute confusion matrix
    cm = confusion_matrix(np.argmax(y_pred,
axis=1),np.argmax(np.array(y_test),axis=1))





    print(pd.Series(np.argmax(np.array(y_test),axis=1)).value_counts())
    print(pd.Series(np.argmax(y_pred, axis=1)).value_counts())

    # plot confusion matrix
    plt.figure(figsize=(8,6))
    sns.heatmap(cm, cmap=plt.cm.Blues, annot=True, fmt='d',
                xticklabels=sentiment_classes,
                yticklabels=sentiment_classes)
    plt.title('Confusion matrix', fontsize=16)
    plt.xlabel('Actual label', fontsize=12)
    plt.ylabel('Predicted label', fontsize=12)
    plot_confusion_matrix(model, X_test, y_test)

```

B. SCREENSHOTS

```
1 import snsrape.modules.twitter as sntwitter
2 import pandas as pd
3 import s3fs
4 from datetime import datetime
5 def run_twitter_etl():
6     scraper = sntwitter.TwitterSearchScraper("#Bitcoin")
7     tweets = []
8     for i,tweet in enumerate(scraper.get_items()):
9         data=[
10             tweet.user.username, # type: ignore
11             tweet.date, # type: ignore
12             tweet.likeCount, # type: ignore
13             tweet.retweetCount, # type: ignore
14             tweet.rawContent # type: ignore
15         ]
16         tweets.append(data)
17         if i>1000:
18             break
19     tweet_df=pd.DataFrame(tweets,columns=['user_name','date','likes','retweets','content'])
20
21
22     tweet_df.to_csv(r's3://rahul-airflow-project/Bitcoindaily3.csv')
23
```

Name	Date modified	Type	Size
 Bitcoindaily	26-03-2023 19:14	Comma Separated...	218 KB
 Bitcoindaily1	26-03-2023 19:18	Comma Separated...	215 KB
 Bitcoindaily2	26-03-2023 19:25	Comma Separated...	210 KB
 Documents - Shortcut	29-03-2023 22:09	Shortcut	1 KB

Amazon S3

Buckets

Access Points

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

Dashboards








AWS Compliance settings



Objects

Properties


Objects (5)




Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

  Copy S3 URI  Copy URL  Download  Open  Delete 

 Create folder  Upload

< 1 >



<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	 Bitcoin_tweets.csv	csv	March 28, 2023, 12:32:37 (UTC+05:30)	961.0 MB	Standard
<input type="checkbox"/>	 BTC-USD.csv	csv	March 28, 2023, 12:32:38 (UTC+05:30)	214.1 KB	Standard
<input type="checkbox"/>	 df_data.csv	csv	March 28, 2023, 12:32:43 (UTC+05:30)	7.3 MB	Standard

```

from time import sleep
import json
import pandas as pd
import io
import re
import numpy as np
from tqdm import tqdm
import pandas as pd
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from tqdm import trange, tqdm_notebook, tqdm

from sklearn import preprocessing
import matplotlib.pyplot as plt

```

```

import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import tensorflow.keras.layers as Layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.losses import SparseCategoricalCrossentropy
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Dense, Dropout, Embedding, LSTM, Conv1D, GlobalMaxPooling1D, Bidirectional, SpatialDropout1D
from tensorflow.keras.models import load_model

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

```

```
t[23]:
```

	index	user_name	user_location	user_description	user_created	user_followers	user_friends	user_favourites	user_verified	date	te
0	195760	محمد	esfahan	خودت رو به خودت ثابت کنه به دیگران	2021-01-02 19:08:56	616.0	1683.0	1712.0	False	2021-06-22 05:22:52	nice proje ort okratech
1	1135874	Crypto Revolution Merchandise ♥	NaN	Crypto Revolution the Wave of change 🤖🤖🤖 Not y...	2019-10-13 06:16:05	451.0	2045.0	45077.0	False	2021-08-25 07:06:40	Long Bitcoin sh the banks (

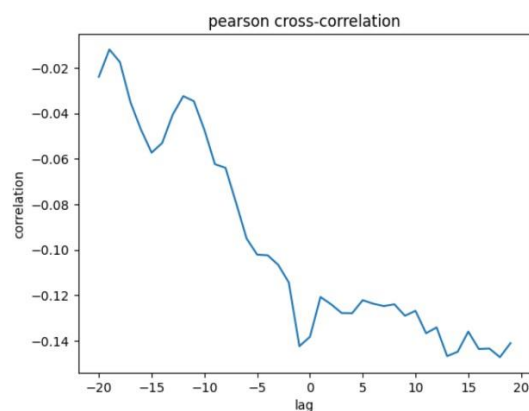
```

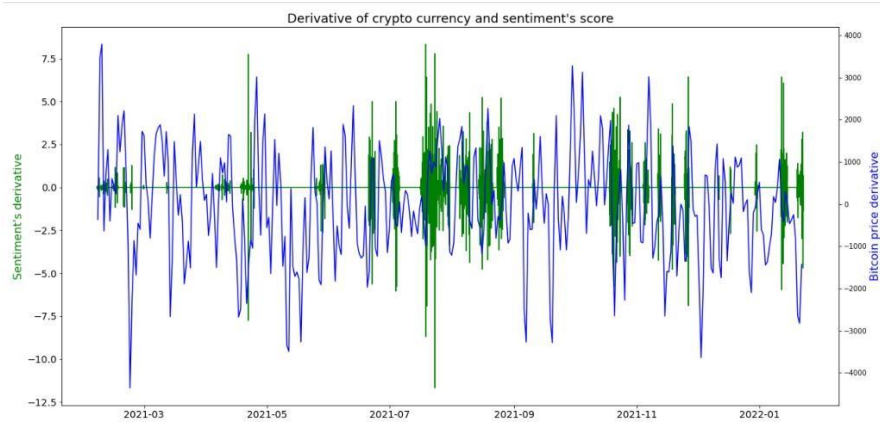
In [28]: xcov = [crosscorr(tweets_grouped, crypto_usd_grouped, lag=i, method="pearson") for i in range(-20,20)]
plt.plot(range(-20,20), xcov)
plt.title("pearson cross-correlation")
plt.xlabel("lag")
plt.ylabel("correlation")
plt.show()

xcov = [crosscorr(tweets_grouped, crypto_usd_grouped, lag=i, method="kendall") for i in range(-20,20)]
plt.plot(range(-20,20), xcov)
plt.title("kendall cross-correlation")
plt.xlabel("lag")
plt.ylabel("correlation")
plt.show()

xcov = [crosscorr(tweets_grouped, crypto_usd_grouped, lag=i, method="spearman") for i in range(-20,20)]
plt.plot(range(-20,20), xcov)
plt.title("spearman cross-correlation")
plt.xlabel("lag")
plt.ylabel("correlation")
plt.show()

```





```
In [35]: def getSubjectivity(tweet):
return TextBlob(tweet).sentiment.subjectivity

def getPolarity(tweet):
return TextBlob(tweet).sentiment.polarity

In [36]: def crypto_price_cate(score):
if score < 1:
return 'negative'
elif score == 1:
return 'neutral'
else:
return 'positive'
def observe_period(period):
res = crypto_usd_grouped.shift(period)/crypto_usd_grouped
res = res.apply(crypto_price_cate)
return res

time_sentiment = observe_period(7) # compare price ratio in 7 days. price_7_days_later/ price_now
df['crypto_sentiment'] = df.date_clean.apply(lambda x: time_sentiment[x] if x in time_sentiment else np.nan)
```

	tweets	cleaned_tweets	date_clean	crypto_sentiment	subjectivity	polarity
0	nice project \n\n\n\n\nortcoin ort okratech ...	nice project ortcoin ort okratech bitcoin aird...	2021-06-22	positive	1.00	0.600
1	Long Bitcoin short the banks 📉	Long Bitcoin short bank	2021-08-25	negative	0.35	-0.025
2	Top Trending Cryptocurrency Post - DOGECOIN Se...	Top Trending Cryptocurrency Post DOGECOIN Sell...	2021-07-02	negative	0.40	0.250
3	Can one expect another wave of BTC's decline s...	Can one expect another wave BTC decline soon v...	2021-07-24	negative	0.00	0.000
4	We will see...\n\n\nbitcoin btc bnb band bake \$btc...	We see bitcoin btc bnb band bake btc dCc xZ dP	2021-05-29	positive	0.00	0.000

```
In [32]: import nltk
from nltk.stem.wordnet import WordNetLemmatizer

nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('punkt')
stop_words = nltk.corpus.stopwords.words(['english'])

print(stop_words)

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'you're', 'you've', 'you'll', 'you'd', 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'she's', 'her', 'hers', 'herself', 'it', 'it's', 'its', 'its', 'elf', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', 'that'll', 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'do', 'ing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'o', 'ut', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'al', 'l', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'don't', 'should', 'should've', 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', 'aren't', 'couldn', 'couldn't', 'didn', 'didn't', 'doesn', 'doesn't', 'hadn', 'hadn't', 'has', 'hasn't', 'haven', 'haven't', 'isn', 'isn't', 'ma', 'mightn', 'mightn't', 'mustn', 'mustn't', 'needn', 'needn't', 'sha', 'n', 'shan't', 'shouldn', 'shouldn't', 'wasn', 'wasn't', 'weren', 'weren't', 'won', 'won't', 'wouldn', 'wouldn't']
```

```

from nltk.tokenize import TweetTokenizer
from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()

def cleaning(data):
    #remove urls
    tweet_without_url = re.sub(r'http\S+', ' ', data)

    #remove hashtags
    tweet_without_hashtag = re.sub(r'#\w+', ' ', tweet_without_url)

    #3. Remove mentions and characters that not in the English alphabets
    tweet_without_mentions = re.sub(r'@\w+', ' ', tweet_without_hashtag)
    precleaned_tweet = re.sub('[^A-Za-z]+', ' ', tweet_without_mentions)

    #2. Tokenize
    tweet_tokens = TweetTokenizer().tokenize(precleaned_tweet)

    #3. Remove Puncs
    tokens_without_punc = [w for w in tweet_tokens if w.isalpha()]

    #4. Removing Stopwords
    tokens_without_sw = [t for t in tokens_without_punc if t not in stop_words]

    #5. Lemma
    text_cleaned = [lem.lemmatize(t) for t in tokens_without_sw]

    #6. Joining
    return " ".join(text_cleaned)

```

```

nltk.download('omw-1.4')
from nltk.tokenize import TweetTokenizer
from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()

def cleaning(data):
    #remove urls
    tweet_without_url = re.sub(r'http\S+', ' ', data)

    #remove hashtags
    tweet_without_hashtag = re.sub(r'#\w+', ' ', tweet_without_url)

    #3. Remove mentions and characters that not in the English alphabets
    tweet_without_mentions = re.sub(r'@\w+', ' ', tweet_without_hashtag)
    precleaned_tweet = re.sub('[^A-Za-z]+', ' ', tweet_without_mentions)

    #2. Tokenize
    tweet_tokens = TweetTokenizer().tokenize(precleaned_tweet)

    #3. Remove Puncs
    tokens_without_punc = [w for w in tweet_tokens if w.isalpha()]

    #4. Removing Stopwords
    tokens_without_sw = [t for t in tokens_without_punc if t not in stop_words]

    #5. Lemma
    text_cleaned = [lem.lemmatize(t) for t in tokens_without_sw]

    #6. Joining
    return " ".join(text_cleaned)

```

```

]: X = df['cleaned_tweets']
y = pd.get_dummies(df['sentiment']).values
num_classes = df['sentiment'].nunique()

```

```

]: seed = 38 # fix random seed for reproducibility
np.random.seed(seed)

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    stratify=y,
                                                    random_state=seed)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

(17936,) (4484,) (17936, 3) (4484, 3)

```

```

]: max_features = 20000
tokenizer = Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(list(X_train))
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)

```

```

]: from tensorflow.keras.preprocessing import sequence
max_words = 30
X_train = sequence.pad_sequences(X_train, maxlen=max_words)
X_test = sequence.pad_sequences(X_test, maxlen=max_words)
print(X_train.shape, X_test.shape)

(17936, 30) (4484, 30)

```

```

In [48]: from keras.utils import plot_model

```

You must install pydot ('pip install pydot') and install graphviz (see instructions at <https://graphviz.gitlab.io/download/>) for plot_model to work.

```

In [49]: history = model.fit(X_train, y_train, validation_data=(X_test, y_test),
                            epochs=epochs, batch_size=batch_size, verbose=2)

Epoch 1/10
141/141 - 27s - loss: 0.6510 - accuracy: 0.7300 - val_loss: 0.3959 - val_accuracy: 0.8559 - 27s/epoch - 190ms/step
Epoch 2/10
141/141 - 15s - loss: 0.2855 - accuracy: 0.9004 - val_loss: 0.3498 - val_accuracy: 0.8880 - 15s/epoch - 108ms/step
Epoch 3/10
141/141 - 10s - loss: 0.1266 - accuracy: 0.9588 - val_loss: 0.3316 - val_accuracy: 0.9057 - 10s/epoch - 70ms/step
Epoch 4/10
141/141 - 11s - loss: 0.0498 - accuracy: 0.9848 - val_loss: 0.3137 - val_accuracy: 0.9146 - 11s/epoch - 75ms/step
Epoch 5/10
141/141 - 14s - loss: 0.0255 - accuracy: 0.9936 - val_loss: 0.3703 - val_accuracy: 0.9228 - 14s/epoch - 103ms/step
Epoch 6/10
141/141 - 7s - loss: 0.0162 - accuracy: 0.9961 - val_loss: 0.3652 - val_accuracy: 0.9199 - 7s/epoch - 47ms/step
Epoch 7/10
141/141 - 14s - loss: 0.0130 - accuracy: 0.9969 - val_loss: 0.3982 - val_accuracy: 0.9231 - 14s/epoch - 97ms/step
Epoch 8/10
141/141 - 14s - loss: 0.0094 - accuracy: 0.9977 - val_loss: 0.4070 - val_accuracy: 0.9215 - 14s/epoch - 101ms/step
Epoch 9/10
141/141 - 6s - loss: 0.0071 - accuracy: 0.9982 - val_loss: 0.4206 - val_accuracy: 0.9237 - 6s/epoch - 44ms/step
Epoch 10/10
141/141 - 14s - loss: 0.0068 - accuracy: 0.9985 - val_loss: 0.4469 - val_accuracy: 0.9248 - 14s/epoch - 100ms/step

```

```

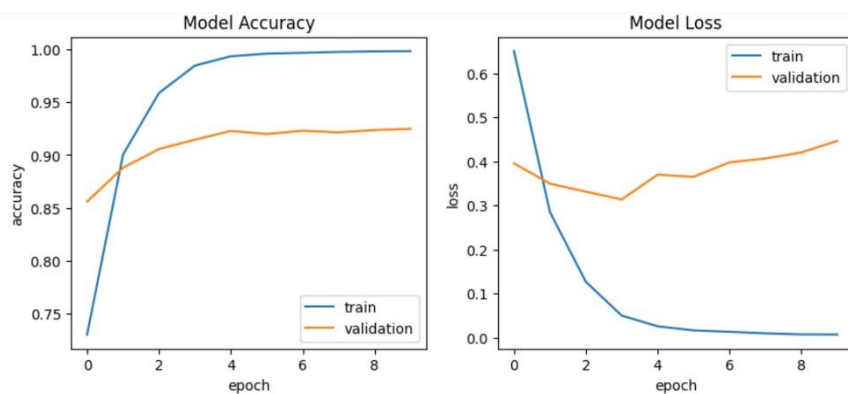
In [50]: def plot_training_hist(history):
          '''Function to plot history for accuracy and loss'''

          fig, ax = plt.subplots(1,2, figsize=(10,4))
          # first plot
          ax[0].plot(history.history['accuracy'])
          ax[0].plot(history.history['val_accuracy'])
          ax[0].set_title('Model Accuracy')
          ax[0].set_xlabel('epoch')
          ax[0].set_ylabel('accuracy')
          ax[0].legend(['train', 'validation'], loc='best')

          # second plot
          ax[1].plot(history.history['loss'])
          ax[1].plot(history.history['val_loss'])
          ax[1].set_title('Model Loss')
          ax[1].set_xlabel('epoch')
          ax[1].set_ylabel('loss')
          ax[1].legend(['train', 'validation'], loc='best')

          plot_training_hist(history)

```



```

In [51]: # predict class with test set
          y_pred_test = np.argmax(model.predict(X_test), axis=1)
          print('Accuracy:\t{:.1f}%'.format(accuracy_score(np.argmax(y_test,axis=1),y_pred_test)*100))
          print(classification_report(np.argmax(y_test,axis=1), y_pred_test))

```

```

141/141 [=====] - 1s 5ms/step
Accuracy: 92.5%

```

	precision	recall	f1-score	support
0	0.84	0.72	0.77	477
1	0.94	0.94	0.94	1779
2	0.93	0.95	0.94	2228
accuracy			0.92	4484
macro avg	0.90	0.87	0.89	4484
weighted avg	0.92	0.92	0.92	4484

