Evaluating The Performance Of Cryptocurrency Trading Signal Providers On Social Media Platforms

MSc Research Project
Data Analytics

Mohib Ur Rehman
Student ID: x23256541

School of Computing
National College of Ireland

Supervisor: Furqan Rustom

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Mohib Ur Rehman
Student ID:	x23256541
Programme:	Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Furqan Rustom
Submission Due Date:	12/12/2024
Project Title:	Evaluating The Performance Of Cryptocurrency Trading Signal Providers On Social Media Platforms
Word Count:	3187
Page Count:	25

Signature:	
Date:	14th January 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Evaluating The Performance Of Cryptocurrency Trading Signal Providers On Social Media Platforms

Mohib Ur Rehman
x23256541

Abstract

The growing volume of trading signals shared across platforms like Telegram presents both an opportunity and a challenge for cryptocurrency traders. This paper addresses automated classification and evaluation of these trading signals. In this paper, we introduce a NLP based machine learning methodology to extract meaningful entities from raw text like coin names, trading pairs, entry points, and target prices relevant to trading. Developing a backtesting framework to test the efficacy of these signals on historical performance, measuring profit/loss, Sharpe ratio and drawdown. To further improve signal analysis, we apply clustering techniques, including KMeans and Gaussian Mixture Models (GMM), to group similar signals and assess their success rates. Through our results we show that it will be possible to use NLP and machine learning as a starting point to automate trading strategy evaluation, leading to insights that will change the way cryptocurrency traders make decisions and manage risk. We propose a robust framework for optimizing trading signal evaluation, that can be applied to other asset classes in financial markets, such as stocks, commodities, forex, and even bonds.

1 Introduction

Cryptocurrency trading is on rise due to the popularity of digital assets and the expansion of online trading platforms. With the advantages of decentralisation, transparency and security, ownership control, resilience to economic crisis, and global reach cryptocurrency has started to challenge the traditional currency system Subramanian et al. (2024). Social media and the popularity of crypto influencers provide trading insight to their followers which is one of the main factors of popularity of cryptocurrency trade. With so many different crypto currencies in existence, it is easy to see why many traders now make use of crypto trading signals, — recommendations on when to buy, sell or hold an asset based on market conditions.

Trading signal providers generate trading signals through studying different ways of analysis such as technical, fundamental, and sentiment analysis. These trading signals allow traders to recognize potential opportunities, set up the entry and exit strategies, defining their risk and directions of the trade N. and D. (2022). Trading signals are a valuable tool, but keeping in mind that they aren't foolproof. These signal providers are humans and can make mistakes, they can misinterpret data or miss the important information that impacts the market. Some signal providers intentionally misrepresent information or manipulate signals for their own gain. And it can end up costing traders who only use trading signals to make their investment decisions. While some signal providers may use sophisticated algorithms and analysis techniques and claim to have a high success rate.

Like any other investment platforms there is so much to choose from — the sheer amount of available trading signals is overwhelming— and not all of them are trustworthy. Up to 80% of these signals are estimated to be fraudulent or misleading as traders become exposed to scams and large losses Matsuzaki et al. (2023). It is one of the biggest concern: the risk of having a cryptocurrency's price artificially inflated through 'pump and dump' schemes where individuals or groups artificially inflate the price of a cryptocurrency, luring novice investors and then selling off their holdings while they are making in profits leaving people in losses at the end of the day. A major risk is misinformation and unreliable trading signals, as the unpredictable nature of the cryptocurrency market means traders can't really filter out what is authentic, and what is a fraudulent or bogus trading signal. When this lack of trustworthiness occurs, it can negatively affect the decision to invest – especially if someone is new to the market. This behaviour aggravates market volatility, while also leading to price movements deviation from the actual value of the assets, and therefore disrupting market efficiency. There is a problem with information asymmetry between signal providers and the rest of the public which gives an unfair advantage to those who are privileged with the information.In addition, trading signal providers often promote impulsive behaviour, high reward and high risk behaviour, including fear of missing out (FOMO) and herd mentality where people want to trade but don't have proper research. Because of these risks, there is a need for a framework that can validate the performance of these signal providers. A framework that evaluates a provider's track record with accuracy, responsiveness to market changes and transparency about their technical expertise or models that they are using.This study provides a multi-stage pipeline to efficiently evaluate the performance of the post trading signals made by providers using historical data.

2 Related Work

2.1 Extraction of key information using Machine Learning Models

Machine learning models combined with text vectorization techniques are of the earliest techniques used to extract meaningful information from raw data. Reddy et al. (2024) propose an automated pipeline based on kafka to process real-time twitter posts for a machine learning analysis of social discourse. The processing of the twitter posts is conducted using pySpark which allows to perform analytics using K-Means clustering, Jaccard similarity, and aggregation. To measure the resemblance of the posts with social justice, jaccard similarity uses a list of words to provide a numerical measure of the tweets alignment. The research conducted by Wagh and Punde (2018) highlights the challenges caused in event identification due to the short length of twitter posts and informal language. Their research proposes a Semi-supervised approach consisting of components like tweet crawling, filtering, pre-processing, and event message identification. A training phase is incorporated in the system for training the model on labelled data including manually labelled tweets followed by a SVM classifier. The results of the study show that the Semi-supervised approach outperforms other machine learning techniques such as logistic regression and naive bayes models in predicting the class of unseen data. Another paper presented by Bharati and M.Upadhyaya. (2024) utilises NLP and SVM classification for interpreting sarcastic messages on twitter. The authors first highlight the issues related with the identification of sarcastic posts i.e juxtaposing positive sentiment words within a negative context and the noise in the messages.. To approach this issue, the authors implemented various NLP techniques for feature extraction, focusing on contextual features that reflect sarcasm. As sarcasm posts reflect high dimensional spaces and imbalanced datasets, the authors provide extensive explanation of SVM model in adapting to such scenarios for producing robust results. Christy and Meeragandhi (2020) utilises a systematic feature selection procedure to identify the metadata for enhancing clustering of features by filtering out extra and harmful data. This canopy technique aims to focus on the data that is beneficial meanwhile, mitigating the overhead of processing caused by irrelevant data. For the clustering stage, the KMeans algorithm is used due to its simple implementation and reliability on large datasets.

2.2 Extraction of key information using Generalised Regular Expressions

Thinking about different ways to extract useful information from unstructured data using regular expressions. The use of regular expressions for information extraction is a popular technique for information mining. Mande et al. (2018) highlight in their study that Regex is a fast technique for text extraction compared to machine learning models which require contextual information and large resources. Research by Li et al. (2008) introduces a method called ReLIE, which improves regex learning by starting with basic predefined patterns and refining them through a series of transformations. This method is particularly efficient and effective when some initial patterns are available to guide the process. Another research by Rosenfeld et al. (2017) identifies an automated technique that creates regex patterns directly from the data itself, without any starting patterns. This approach is very adaptable and can be applied across various types of text, making it useful in situations where no initial patterns are available. This research will help us to use any of these techniques to extract key information. This will help simplify the diverse writing styles of the signals into a standardised format.

The research conducted by Carloni et al. (2023) introduces an efficient way to use regex. They use principles of Domain Specific Architectures (DSAs) to produced tailored architectures for regular expressions therefore allowing the regex systems to run on low powered IoT devices. Employing this technique on this study can be beneficial as a tailored system could produce effective results. Another research by Haghighat and Li (2018) focuses on the utilisation of automata based techniques such as DFA and NFA to improve the text extraction capability of regex. The authors introduce the concept of HES which utilizes automata techniques to match thousands of patterns in a reasonable time. Experiments from the research show that the E-HES method significantly improves the text matching procedure without showing spatial or temporal limitations.

Generation of an efficient regex expression is one of the key challenges, a perfect balance between processing time and pattern structure is required to produce an efficient regex pattern. Zhang et al. (2023) introduce InfeRE in their research which is a paradigm for step-wise generation of regex expressions. This approach delivers significant improvements over conventional autoregressive models that often fail to capture the real order of text processing behind regexes.

2.3 Extraction of key information using large language models

Recent research by Dagdelen et al. (2024) tells the important role of prompt engineering, specifically for unstructured information extraction tasks such as cryptocurrency signals. This research tells that detailed prompts solve the problem of accuracy relatively well but struggle to deal with a wide range of text formats as the contexts and extraction guidelines become unclear and too specific. Likewise, research by K and R (2019) suggest importance of schemas to define what and how to extract information, but can be inflexible when handling new data structures. Entity boundaries are often unclear in unstructured real world data, a challenge facing LLMs like Llama 2. The problem becomes compounded in cryptocurrency signals where the language is informal, there are abbreviations, and formatting is inconsistent, making it a difficult task to extract key details accuracy from. This is where pre-processing steps such as cleaning text and expanding abbreviations can help and designing more specific prompts for different formats is a must.

Research by Yang et al. (2023) experiments with the BERT model for processing legal documents. The researchers of this study introduce a text segmentation approach to overcome the limitations of 512-token input limit. This text segmentation is not only useful for lengthy segments but can also be used on smaller input sizes such as for processing calls in this study. The results of the study show that by analyzing the highest-scoring text segment can improve the precision of BERT-based classification, reducing manual review effort by 14% at a 95% recall level. Another study conducted by Bhavya and M.Nidd. (2023) utilises text segmentation to develop the SegLLM for efficient phone call segmentation and topic extraction. The results of the study showed superior performance in segmentation accuracy and topic assignment compared to existing methods.

The study conducted by Fariha et al. (2024) proposes a hybrid approach for automated analysis and detection of anomaly from unstructured log files. To cater the issue of unstructured data, the study proposes the use of LLM to construct regular expressions for parsing the log data and then applies LLM-based parsing and embedding to transform the raw log data into a format suitable for the deep learning model. This hybrid approach, therefore provides flexibility in handling unstructured data without the need for manual feature engineering or parsing rules.

Thus, in summary, LLMs can be adapted for specific extraction tasks in a tailored manner by sensible prompt design and finetuning if strong data preparation, domain knowledge, and the thought of going further via a synergy with rule-based systems to raise accuracy.

3 Methodology

The system follows a multi stage pipeline to create an architecture capable of categorizing post trading signals efficiently. A Telegram scraper is used to extract raw messages from trading groups first. These messages often contain a mixture of valid trading signals and invalid trading signals (e.g, some general discussions on any crypto coin) . A solution to this is accomplished by the use of a Random Forest Classifier as a data filtration process. The classifier identifies and filters out valid trading signals (those that actually produce actionable predictions) using labeled training data. Natural language processing (NLP) techniques are then applied to standardize the format of different writing styles within the dataset. This preprocessing step ensures that all the trading signals gets cleaned, tokenized and formatted to a single standardized format to be analysed. The preprocessed signals are then provided for backtesting. In this step the trading signal is evaluated at that point of time when the trading signal was shared on telegram and historical data is gathered from BinanceAPI, one for each token listed in the signals. Backtesting module produces performance metrics like number of targets achieved, stop loss events, and risk reward ratios. These features allow us to assess how well the trading signals performed in real market conditions. After feature extraction, trading signals are grouped into Good, Average and Bad clusters using K-means, Gaussian Mixture Model (GMM) and Spectral clustering. The final stage of the architecture aggregates the categorized signals to derive the success rate of each trading signal provider. This multi stage pipeline flow ensures that signals are first validated, then standardized, analyzed, and finally categorized providing a comprehensive performance evaluation. By adopting this structured approach only high quality valid signals are used and the application of back testing with clustering provides actionable insights from trading signals. The proposed methodology overview is shown in Figure 1.

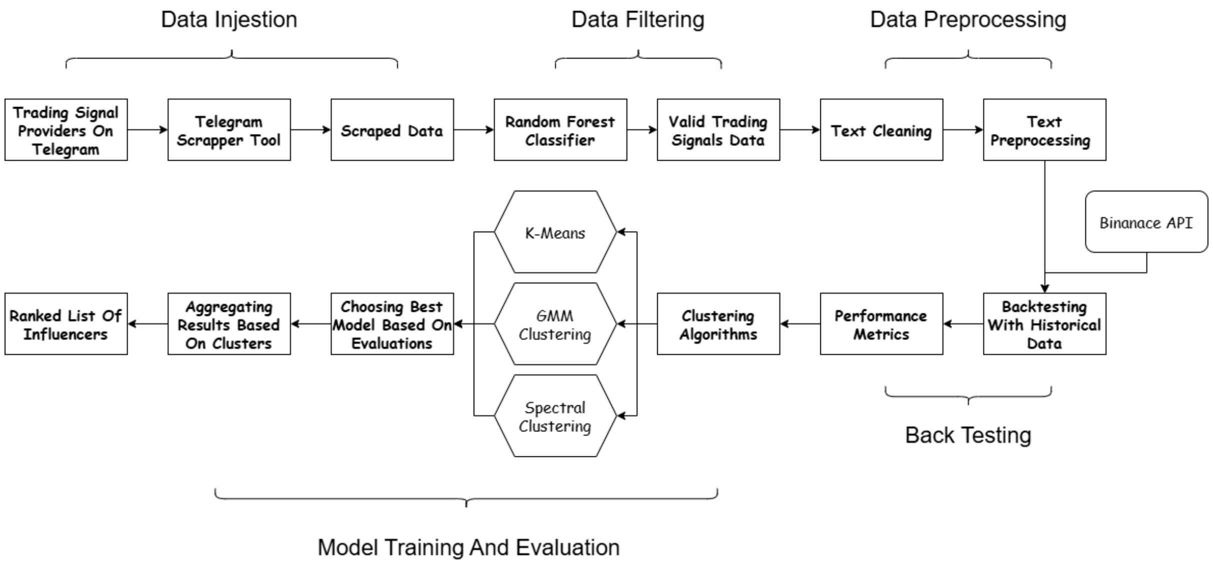


Figure 1: Methodology Overview

The Design is divided in 5 phases such that this whole process stream lines the trading signal evaluation. The First part of this design is Data Collection, The data collection process involves collecting all the text messages from different trading signal providers who give their signal on Telegram. This phase stores all their text messages in raw format. The next phase invloves Data filteration, this phase is intended to gather all the data stored and pass it to a filteration process, where machine learning classification model is trained to classify all

3.1 Data Collection

Gathering Telegram data by a scraper tool starts with identifying and selecting popular Telegram trading signal providers. These groups can be public or private, the providers are selected focusing on those who are known for providing good, real time trading insights. Obtaining group links or access tokens where possible also means easier access to private groups. Using scraper tool a set number of messages (3,000 or more) messages are scraped after every 2 minutes, respecting rate limits. The data is saved in a format that can be forwarded to next modules for further processing.

3.2 Data Filtration

This section discusses a systematic keyword-based approach to label our text messages that distinguishes trading signals and non-trading signals (general conversational messages). The classification process is designed to comprehensively evaluate the content of text messages by searching for specific, hierarchically organized trading-related terms. The keyword search process is classified into four steps. Keywords like Buy, Sell, Long or Short are used in the first step to identify fundamental market action indicators and sign of potential trading intent. If the method detects these market action terms, the method moves to the second step of searching for the entry-related keywords such as "Entry," "Entries," "enter," or "buy" or "buying" to determine when a trade was started. The third step identifies keywords that imply a profit objective (target, targets, take-profit, tp, tp1) in order to confirm that take-profit targets are mentioned in the messages . The fourth tier looks at the occurrence of stop-loss related terms including 'stop', 'stoploss' or 'sl' or 'stop loss' that indicate price when the trade should be terminated. Figure 2 depicts, If all four categories of terms are present, it is assigned as a trade signal, if not, the message is labelled as a non trade signal. This rigorous and structured methodology guarantees precise separation of trading signals from general text messages.

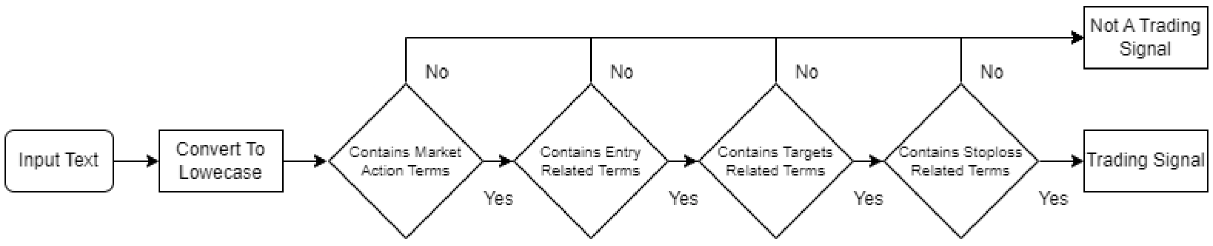

Not A Trading Signal
No
No
No
No
Input Text
Convert To Lowecase
Contains Market Action Terms
Contains Entry Related Terms
Contains Targets Related Terms
Contains Stoploss Related Terms
Trading Signal
Yes
Yes
Yes
Yes

Figure 2: Labeling Text Messages

After labeling the text messages, the next step involves creating a machine learning pipeline that can use this labelled data for trading signal classification. Initial step of pipeline involves converting text into Term Frequency-Inverse Document Frequency (TF-IDF) vectorization to give more weights to words that are important like ("Buy", "Targets") should be given more weightage than filler words like ("is", "the"). Added custom features to enhance the classification by checking length of the messages, if the text contains any numerical value that represents price, lookup for trading pairs (e.g., "BTC/USDT") and identifying presence of important keywords like entry price, target prices, stop-loss, or leverage. This custom feature extraction allows the model to comprehend more easily as these are the only features that are needed for further processing. Figure 3, below represents overall architecture for trading signal classifier.

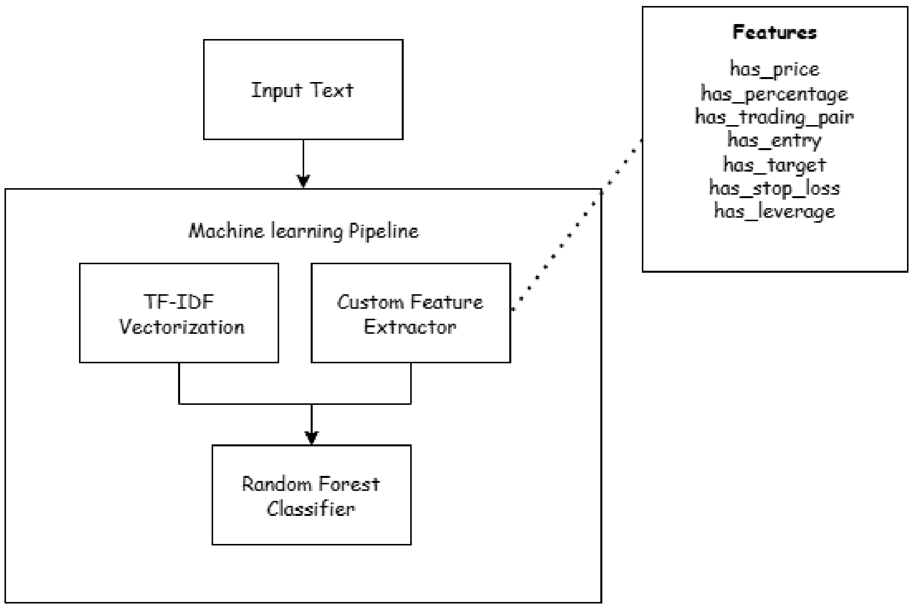
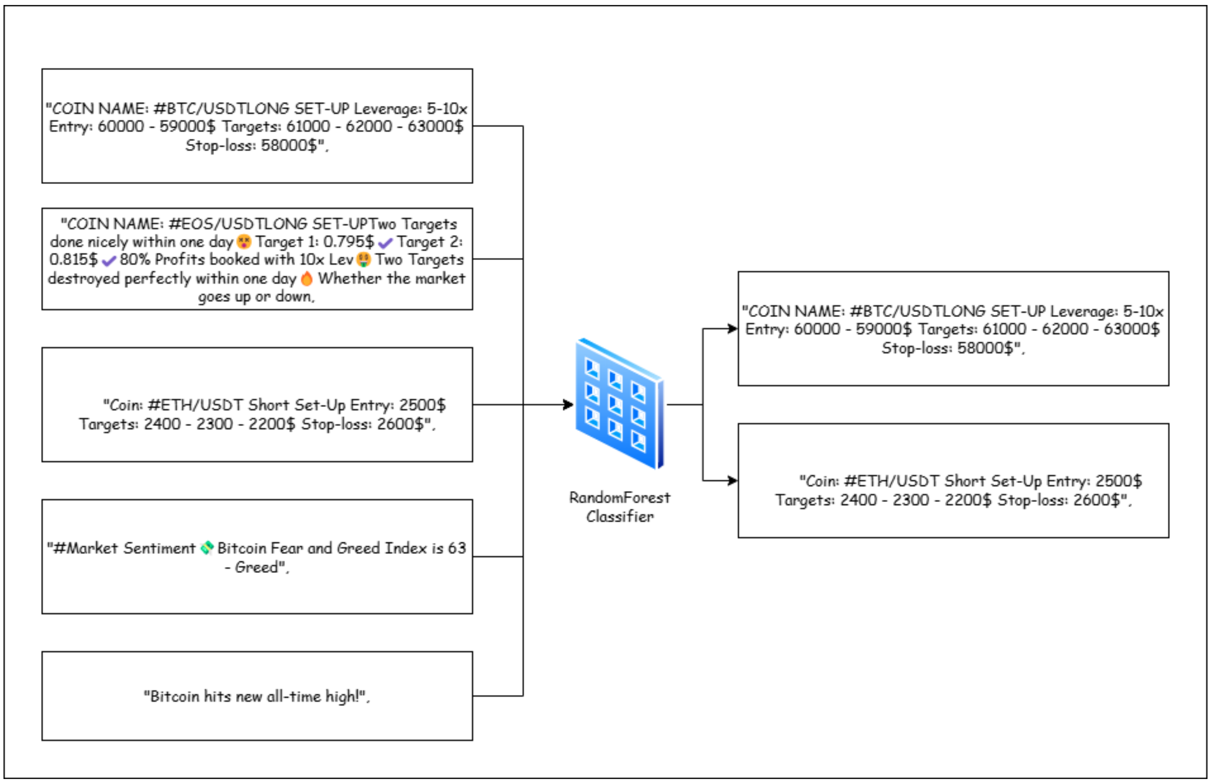


Figure 3: Trading Classifier Overview

The classification of trading signals is performed using a Random Forest Classifier because of its robust classification by aggregating predictions from multiple decision trees. Once classified, the trading signals are filtered out from the dataset and sent forward for the next phase of data filtration. Figure 4 provides sample results generated from this classifier.


"COIN NAME: #BTC/USDTLONG SET-UP Leverage: 5-10x Entry: 60000 - 59000$ Targets: 61000 - 62000 - 63000$ Stop-loss: 58000$",

"COIN NAME: #EOS/USDTLONG SET-UPTwo Targets done nicely within one day 😮 Target 1: 0.795$ ✔ Target 2: 0.815$ ✔ 80% Profits booked with 10x Lev 🤑 Two Targets destroyed perfectly within one day 🔥 Whether the market goes up or down,

"Coin: #ETH/USDT Short Set-Up Entry: 2500$ Targets: 2400 - 2300 - 2200$ Stop-loss: 2600$",

"#Market Sentiment 💸 Bitcoin Fear and Greed Index is 63 - Greed",

"Bitcoin hits new all-time high!",

"COIN NAME: #BTC/USDTLONG SET-UP Leverage: 5-10x Entry: 60000 - 59000$ Targets: 61000 - 62000 - 63000$ Stop-loss: 58000$",

"Coin: #ETH/USDT Short Set-Up Entry: 2500$ Targets: 2400 - 2300 - 2200$ Stop-loss: 2600$",

Figure 4: Trading Signal Classifier Results

3.3 Data Preprocessing

The second step is to take the filtered trading signal data, and clean and format it to make it consistent across all trading signal. The aim of this step is to reduce unstructured, inconsistent message forms into a more readable while retaining the important trading information. Here's an in-depth explanation of how this process is carried out:

3.3.1 Handling Camelcase Words

Camelcased words are most often used in trading signal messages, to elaborate more multiple words are tied together without spaces like "EntryTargets" or "TakeProfit". In this step, we do something to make the CamelCase words more clear and we split them by inserting the space between them. For example: "EntryTargets" is changed to "Entry Targets". "TakeProfit" becomes "Take Profit". And this makes the message more human readable, and it is easier to identify individual parts in the signal.

3.3.2 Formatting Numbers and Targets

Trading signals usually contains many numerical values in between the text , for example, entry prices, take-profit targets, stop loss. These numbers are often concatenated with words or some other numbers in the raw text, making it hard to follow the sequence. Therefore, in order to solve this problem we format the numbers to facilitate spacing between the number and their appropriate terms or symbols. It is useful to establish clear differences among different values. For example, if the text contains: "Target1 1240Target2 1280 Target3 1290" It will be reformatted to: "Target 1 1240 Target 2 1280 Target 3 1290" Additionally, when take-profit targets are mentioned, regular expressions (regex) are used to extract these numerical values from the text. The extracted values are then reordered into a numbered list to enhance clarity and consistency. For example: "2.3 3.5 4.0" will be transformed into "1) 2.3 2) 3.5 3) 4.0". This makes it easier for users to identify each target in a sequential manner.

3.3.3 Final Cleanup

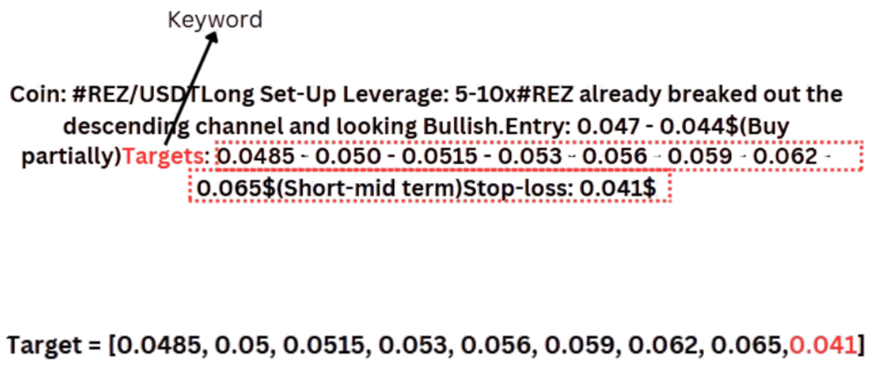
Once all the above steps are done and the text is properly formatted, a final cleanup process is performed. This ensures the signal text is free of any extraneous spaces or formatting errors, such as:

1. Inconsistent punctuation usage
2. Irregular spacing around symbols (such as "$", "%", or "/")
3. Improper word separation

We trim the unnecessary spaces from the trading signal text, also fixing the formatting issues to keep the trading signal text compact, neat and easy for extraction. Symbols, numbers, and key trading terms are carefully preserved to retain their significance.

3.3.4 Extracting Key Entites

Regular expressions are used to extract entities like coin name, entry price, targets, and stop-loss. The coin name is detected by locating the trading pair, like "USDT", which always followed by coin name (e.g., in "BTC/USDT", "BTC" is the coin name, and "USDT" is the trading pair). For extracting values like entry price, targets, and stop-loss, a targeted search is performed using specific keywords such as "entry", "target", or "stop-loss". Once a keyword is identified in the text, the regex captures the numbers that follow, continuing until a predefined stopping condition(known as 'number_count') is reached. With this approach the extraction task is performed robustly. Figure 3.3.4, below provides example illustration, where red-outline represents search window.



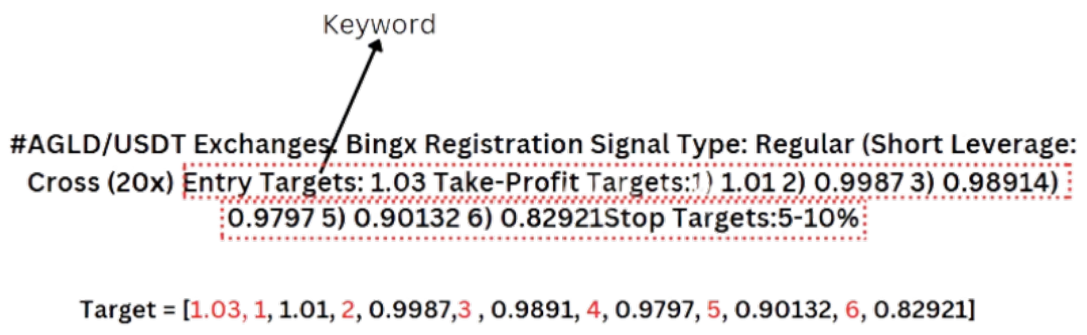
Coin: #REZ/USDTLong Set-Up Leverage: 5-10x#REZ already breaked out the descending channel and looking Bullish.Entry: 0.047 - 0.044$(Buy partially)Targets: 0.0485 - 0.050 - 0.0515 - 0.053 - 0.056 - 0.059 - 0.062 - 0.065$(Short-mid term)Stop-loss: 0.041$

Target = [0.0485, 0.05, 0.0515, 0.053, 0.056, 0.059, 0.062, 0.065,0.041]

Figure 5: Targeted Search Result

3.3.5 Validating Results

The targeted search approach has several limitations, such as including extra information that are not even part of the intended targets. For example, stop loss values can sometimes becomes part of targets as shown in figure 3.3.4 and 5. Furthermore, in many cases, targets are paired with indexes (1, 2, 3), making accurate extraction more difficult. Therefore, domain knowledge has to be used to prevent the numbers that belong to one (stop loss or entry price) shouldn't belong to another entity. Targets are always higher than stop loss and entry points; therefore, any value that's higher should be discarded. In addition, targets should always be in increasing order. As a solution to the indexing issue, the system should look for sequential pattern such as 1, 2, 3 that are supposed to be the indexes and not the target values themselves. And hence, should be removed. Standard deviation of the list should be implemented such that any number that is an outlier should be removed. In figure 6, 1.03 is removed because it belongs to Entry price Entity, and the numbers 1,2,3,4,5,6 are removed because they are out of standard deviation as well as follow sequential pattern.



#AGLD/USDT Exchanges: Bingx Registration Signal Type: Regular (Short Leverage: Cross (20x) Entry Targets: 1.03 Take-Profit Targets:1) 1.01 2) 0.9987 3) 0.98914) 0.9797 5) 0.90132 6) 0.82921Stop Targets:5-10%

Target = [1.03, 1, 1.01, 2, 0.9987,3 , 0.9891, 4, 0.9797, 5, 0.90132, 6, 0.82921]

Figure 6: Validating Results Example

3.4 Backtesting

After the trading signals have been standardized to a common format, they are passed to a backtesting module and performance features are derived using historical data from Binance. From the point of time when the trading signal was initiated, historical price data for the next 20 days is gathered for that particular coin. This historical data is then used to compute a variety of features that provide insights into the signal's performance:

1. Percentage_Hit : Total number of targets successfully reached divided by total targets mentioned in the signal.

2. Stoploss_Hit : Boolean feature indicating whether stop loss level was hit or not during the trade duration.

3. Stoploss_Duration : How much time the trade was active before stop loss was hit, if there was one.

4. Max_Profit : The highest peak generated from price movements within the time frame.

5. Max_Loss : The lowest trough seen in the trade within the timeframe.

6. Max_Drawdown : It represents the risk of substantial losses from the largest peak to trough decline in the price of the coin during the trade period.

7. Trade_Duration : The time that the trade was active, starting from the timestamp that signal was initiated to close, whether that was from hitting all the targets, achieving the stop loss, or by completing 20 days of time evaluation with no exit.

3.5 Model Training And Selection

Model selection is carefully analysed and tested throughout all the clustering algorithms namely KMeans, Spectral Clustering and Gaussian Mixture Model,DBScan, and Hierarchical Clustering. But the choice of method is based on:

K-means Clustering: For its simplicity and the fact that it can handle large datasets quickly, this model was chosen. In many cases Kmeans works well, especially when the clusters are spherical and well separated. It is computationally efficient for large datasets as it quickly partitions the dataset into a fixed number of clusters.

Spectral Clustering: The data was not particularly complex, so spectral clustering was applied. Based on a similarity measure, this algorithm can partition a dataset graph structure and capture complex patterns in the data. It is however more appropriate than most for data that can be portrayed as a graph and may be less practical for very large sized data sets.

Gaussian Mixture Model (GMM): The dataset fit this model by assuming the points are drawn from a mixture of a number of Gaussian distributions. GMM's ability to capture clusters of varying shapes and sizes is done by having a Gaussian for every cluster. K-means is fast, very fast in fact, but there is no reason that clusters should be elliptical and GMM is more flexible, especially when the clusters are elliptical, but GMM can be more computationally intensive.

The features calculated using the backtesting module are passed to each clustering algorithm, allowing them to categorize each trading signal into one of three categories: Good, Average, or Bad. Once categorized, the results are aggregated to assess the overall performance of all trading signal providers. This approach enables a comprehensive evaluation of the quality of trading signals, providing insights into which providers provide most profitable trading insights.

4 Design Specification

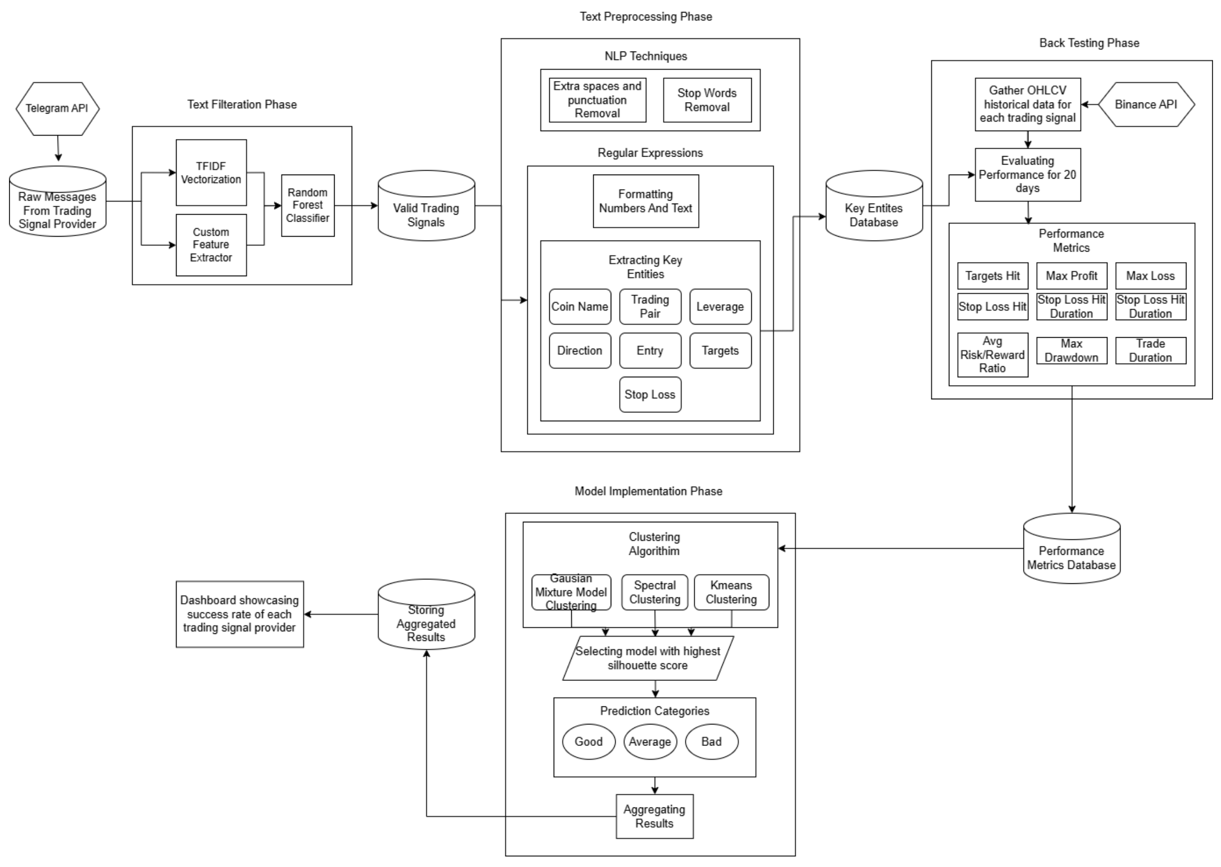


Figure 7: Design Specification Diagram

The proposed system shown in figure 7 follows a modular and multi-stage pipeline designed to efficiently categorize and analyze post-trading signals. The architecture begins with a Telegram scraping tool, implemented using library SNScrape, to extract raw messages from trading groups. These messages often contain both valid trading signals and irrelevant data. To address this, a Random Forest Classifier is employed as a data filtration process to identify actionable trading signals. The classifier is trained on labeled datasets and differentiates between valid signals and non-valid signals (e.g, Discussions). Following this, natural language processing (NLP) techniques, including tokenization and normalization, are applied along with Regular Expressions to standardize diverse writing styles into a unified format, ensuring that named entities like coin name, targets, entry points can be easily extracted from raw messages. The named entities are stored and are then subjected to backtesting module, where historical market data is retrieved from BinanceAPI to evaluate the performance of trad- ing signals at the time they were shared. Performance metrics such as the number of targets achieved, stop loss achieved, and trade durations are calculated to provide quantitative insights into the signals' effectiveness. Once the backtesting is complete, the signals are categorized into performance-based clusters (Good, Average, and Bad) using unsupervised learning algorithms such as K-Means, Gaussian Mixture Models (GMM), and Spectral Clustering. The final stage of the architecture aggregates these categorized signals to determine the success rates of individual trading signal providers, offering a comprehensive evaluation of their reliability. By implementing this structured approach, the system bridges raw data extraction, signal validation, and performance analysis, providing a fast and effective solution for evaluating trading signals.

5 Implementation

5.1 Data Injestion

For the data collection process, we utilized the Python library SNScrape to gather text messages from Telegram. The focus of our data collection was on five prominent trading signal providers who actively share trading signals. Their Telegram handles are:

1. Wallstreet Queen Official
2. Crypto Signals Orge
3. Binance Killers VIP Channel
4. Crypto Club Pump Signal
5. WolfxCrypto VIP

From each group, we scraped 3,000 recent text messages, resulting in a comprehensive dataset of 15,000 messages. The column representation of this dataset is shown in Table 1

5.2 Data Filteration

5.2.1 Creating Labeled Data

The text is first converted into lowercase. A nested conditional block containing filtration process is implemented. Each stage checks for a specific set of keywords shown in table 2,

Column	**Non-Null Count**	**Dtype**
Channel Name	15000 non-null	string
Date	15000 non-null	string
Time	15000 non-null	string
Messages	15000 non-null	string

Table 1: DataFrame Summary

such as entry-related, target-related, stop-loss-related, and leverage-related keywords. If a set of keywords is found in the text, the function moves on to the next stage; otherwise, it exits early and returns False.

By utilizing set membership checks `in` function efficiently determine whether any of the relevant keywords are present in the text. If all the keywords are found the function returns True.

Keyword	Related Keywords
Entry	'entry','enter','entries','buy','buying'
Target	'target','targets','take-profit','tp','tp1'
Stop-Loss	'stop','stoploss','sl','stop-loss'
Leverage	'leverage','lev','cross','x'

Table 2: Related Keywords Lookup Table

5.2.2 Custom Feature Extractor

After the data is labeled. For each text message feature engineering is done. The new features are generated such that model will be trained on these features.Table 3 describes each feature.

Feature	Data Type	Description
keyword_count	Integer	The total number of keywords identified in the text
has_price	Boolean	Indicates whether a price is mentioned in the text
has_trading_pair	Boolean	Indicates the presence of a trading pair in the text
has_entry	Boolean	Indicates whether any entry-related keywords are mentioned.
has_target	Boolean	Indicates whether any target-related keywords are mentioned
has_stop_loss	Boolean	Indicates the presence of stop-loss related keywords in the text
has_leverage	Boolean	Indicates if leverage-related keywords are mentioned in the text

Table 3: Custom Feature Description

5.2.3 Pipeline Implementation

The pipeline is constructed using the sklearn.pipeline module and comprises several components designed for text classification. Everything is first fed into a TF-IDF vectorizer with max_features = 1000 and common english stop words are ignored and unigrams and bigrams is considered with (n_gram_range= (1, 2)). Custom feature extractor followed by TF-IDF vectorization is implemented to strengthen the feature space and fed into pipeline. Finally, the pipeline concludes with a RandomForestClassifier, from the sklearn.ensemble module, set on 200 estimators and class weights set balanced to counteract the possible existence of class imbalance.

5.2.4 Model Implementation

The pipeline is initiated using K-Fold cross-validation from sklearn.model_selection, with 5 folds selected for evaluation. The dataset is split into training and testing subsets using train_test_split, allocating 80% of the data for training and 20% for testing. The model is then trained by applying pipeline.fit on the training data, X_train. To generate predictions, pipeline.predict is utilized on the test data, X_test. Predictions labeled as 1 are identified as valid trading signal text messages and are subsequently forwarded to the next phase of implementation.

5.3 Data Preprocessing

After data filtration, all the valid trading signals are preprocessed and the entities are extracted using series of operations:

5.3.1 Formatting Text

The messages that contain Unicode characters, extra spaces, acronyms and emojis are removed using regular expressions with the `re` library. The `substitute` function is used to replace all Unicode characters with an empty string. CamelCase words are separated such that when an uppercase character is followed by a lowercase character, a space is inserted between them.

5.3.2 Extracting Entities

The utility functions find_numbers_before_keyword and find_numbers_after_keyword are implemented to extract numbers that appear before or after a specified keyword in a given text. Both functions take three parameters: text, which is the input string to search within; keyword, the term to locate; and max_number, which limits the number of extracted values. The process begins with the `re.search` function to identify the position of the keyword in the text. For find_numbers_after_keyword, once the keyword is found, the text is truncated from the end of the keyword onward using `text[match.end():]`. Conversely, for find_numbers_before_keyword, the text is sliced up to the start of the keyword using `text[:match.start()]`. In both cases, after the text is appropriately sliced, `re.finditer` is used to extract all numbers from the truncated text until the limit set by max_number is reached. The extracted numbers are then returned as a list. This utility function will extract target prices, stop loss prices, entry prices.For coin name, it was noted that, every single message has the same format, the coin name is always followed directly by the trading pair ”USDT”. `re.findall` method is used to find first occurrence of the word ”USDT”, using that coin name is extracted.

5.3.3 Validating Results

The utility function used to extract target prices, stop-loss prices, and entry prices can sometimes include extra information, such as index numbers (e.g., 1, 2, 3) that are mentioned with the targets. To help mitigate this issue, the clean_target function is employed. It calculates the standard deviation of the list of extracted numbers and removes any values that significantly deviate from the rest, as these are not valid prices. This ensures that only correct values are stored in the list.

To handle the issue of extracting index numbers, the is_likely_index function analyzes all the numbers in the list and checks for a sequential pattern, such as 1, 2, 3, which typically indicates indexing rather than price levels. If such a sequence is observed, the function removes these numbers from the list.

Additionally, the extracted target prices are validated based on common cryptocurrency trading rules. For example, target prices should always be in ascending order. The validation ensures that the list is sorted in increasing order. Furthermore, the stop-loss and entry prices are checked against the first target price; both the stop-loss and entry price should always be less than the first target, following standard trading principles.

5.4 Backtesting Using Historical Data

5.4.1 Data Gathering

The first step of backtesting involves retrieving the OHLCV (Open, High, Low, Close, Volume) historical price data using ‘ccxt‘ Python library. 4 hour candles are used as a timeframe. For each trading signal shared in Telegram groups, data is gathered starting from the timestamp when the trading signal was shared on telegram and extending over the period of next **20 days**, covering 120 4-hour candles. Retrieved data contains the OHLCV values, and stored in the pandas DataFrame for evaluating performance metrics.

5.4.2 Removing Already Achieved Targets

During implementation, it was noted that some of the targets listed in the signals had already been achieved before the user could react to them. An improvement to address this is that the system will see if the market price at the signal’s timestamp has already exceeded any targets. If a target has already been reached, it is discarded for a given signal from the list of achievable targets. It is done by comparing each target with the signal price (the price when the signal was initiated). The targets are removed for long signals if the signal price is higher than, or equal to any of the targets, and for short signals the targets that are lower than, or equal to, are removed. This filtering ensures that only valid, unachieved targets are considered for evaluation.

5.4.3 Calculating Performance Metrics

After revising the targets, the following performance metrics are calculated for each signal based on the OHLCV historical data:

1. Percentage_Hit : The system tracks for each signal whether the price of that token exceeds each of the remaining targets, using a counter variable for each target achieved it computes the proportion of target hits among the total number of targets. If all the targets are achieved the evaluation stops.

2. Stoploss_Hit : The system monitors whether the the price of that token gets below the stop-loss at any point of time within 20 days timeframe. When the market price

reaches the stop-loss level before touching all the targets, stop loss is triggered and the evaluation stops.

3. Max_Profit : High value of each candlestick is compared with the maximum value seen for all historical OHCV data. After the identification of the highest price for a given cryptocurrency, the percentage change is calculated using :

$$\text{Max\_Profit \% Change} = \frac{\text{Max\_Price} - \text{Entry\_Price}}{\text{Entry\_Price}} \times 100$$

4. Max_Loss : Low value of each candlestick is compared with the minimum value seen for all historical OHCV data. After the identification of the lowest price for a given cryptocurrency, the percentage change is calculated using

$$\text{Max\_Loss \% Change} = \frac{\text{Entry\_Price} - \text{Min\_Price}}{\text{Min\_Price}} \times 100$$

5. Trade_Duration : The system continuously monitors exit conditions during each evaluation iteration. An exit is triggered when either all targets are met or a stop-loss is achieved. The time duration is calculated by subtracting the timestamp of the trading signal's initiation from the timestamp of the triggered event. If no exit conditions are met within a 20-day evaluation period, the trade duration is capped at 480 hours (the total number of hours in 20 days).

5.5 Model Training And Evaluation

5.5.1 Feature Weighting and Selection

The model begins by selecting key performance metrics calculated in the backtesting module. These features include:

Features	Description
Percentage_Hit	The percentage of targets hit.
Stoploss_Hit	Whether the stop loss was hit or not.
Max_Profit	The maximum profit achieved during the trade
Max_Loss	The maximum loss sustained during the trade
Targets_Distribution	Price difference distribution between each take-profit targets
Moving_Average	Trend direction and strength
Trade_Duration	The total active duration of the trade

Table 4: Feature Description

Each of these features has varying importance in the context of evaluating trading signals. Therefore, a set of predefined weights is assigned to each feature to reflect its significance in the clustering process. The table 5 below defines all the weights assigned.

Feature	Weight
Percentage_Hit	2
Stop_loss_Hit	1.5
Max_profit	1.5
Max_loss	1.5
Targets_Distribution	1.0
Trade_Duration	1.0

Table 5: Feature Weights Assignment

Percentage_Hit has the highest weight (2), while other metrics such as Max_profit and Trade_Duration are assigned lower weights (1.0). These weights are used to influence the clustering algorithms in capturing the most relevant patterns for evaluating trading signal performance.

5.5.2 Data Normalization And Weight Application

After the feature selection, the data is normalized using MinMaxScaler. This ensures that each feature has been scaled to a standard range (between 0 and 1) so that features with larger magnitudes (e.g, Max_profit) do not dominate the clustering process.

Once normalized, the assigned weights are applied to each feature. To accomplish this, we multiply the values of each feature with their respective weights associated, effectively controlling the influence of the features on the results of the clustering. Thus it can give more weight to more important features, like Percentage_Hit over other less impacting features, Max_Loss.

5.5.3 Model Implementation

We used three clustering techniques to group trading signals: KMeans, Spectral Clustering, and Gaussian Mixture Model (GMM), table 6 shows tuning range for all the models and the implementation is as follows:

- KMeans : From `sklearn.cluster`, we set `n_clusters=3`, `random_state=42`, and `n_init='auto'`. The model was trained using the `fit_predict` function, and the results were stored in a column called ‘k_means_cluster‘.

- Spectral Clustering : Also from ‘sklearn.cluster‘, we used `n_clusters=3`, `random_state=42`, `affinity='nearest_neighbors'`, and `gamma=0.02`. The clustering results, obtained via `fit_predict`, were stored in the ‘spectral_cluster‘ column.

- Gaussian Mixture Model (GMM): Using `sklearn.mixture`, we set `n_components=3`, `random_state=42`, and `covariance_type='spherical'`. The output was saved in a column named ‘gmm_cluster‘.

To evaluate the performance of these clustering methods, we calculated the `Silhouette score` for each one using the `silhouette_score` function from `sklearn.metrics`.

The challenge we faced was that the clustering results were just numerical labels (e.g., 0, 1, 2), and we can't assume that 0 is 'bad' signal. To address this, we calculated a 'composite score' by grouping the signals by their cluster label and aggregating the average 'percentage_hit' for each cluster.

We chose 'percentage_hit' because it indicates how many targets were hit by a trading signal. A high 'percentage_hit' suggests a strong signal, regardless of other metrics. Finally, we mapped the clusters to meaningful labels: the cluster with the highest composite score was labeled as "Good," the lowest as "Bad," and the middle as "Average."

Model	**Hyperparameters**	**Tuning Range**

K Nearest Neighbour	n_clusters = 3
	random_state = 42
	n_init = 'auto'
		n_clusters = [1,10]
		n_init = 'auto' or [1,inf]

Spectral Clustering	n_clusters = 3
	random_state = 42
	affinity = 'rbf'
		n_clusters = [1,10]
		affinity = ['nearest_neighbors', 'rbf', 'precomputed']

Gaussian Mixture Model	n_components = 3
	random_state = 42
	covariance_type = 'spherical'
		n_components = [1,10]
		covariance_type = ['full', 'tied', 'diag', 'spherical']

Table 6: Clustering Models Hyperparameter Configuration

6 Evaluation

6.1 Case Study 1: Clustering Analysis Using Principle Component Analysis (PCA)

This study examines how three different clustering algorithms namely Gaussian Mixture Model (GMM), Spectral Clustering, and K-Means classify the trading signals from various providers based on their performance. Figure 8, below helps understand how each algorithm categorizes the signals and how well they separate on horizontal axis by means of different performance levels. To make the analysis-plot, Principal Component Analysis (PCA) was applied to reduce the complexity of the high-dimensional data. This transformation allowed the performance metrics to be visualized in a two-dimensional space, with the first two principal components (PCA1 and PCA2) explaining 61.85% and 15.06% of the variance, respectively, offering a clear representation of the data's spread.

In each of the three clustering models, the trading signals were grouped into three categories based on their success rates: good-performing signals were represented by green clusters, average signals by yellow clusters, and poor-performing signals by red clusters. The clustering performance was evaluated using silhouette scores, which measure how well the signals were grouped.

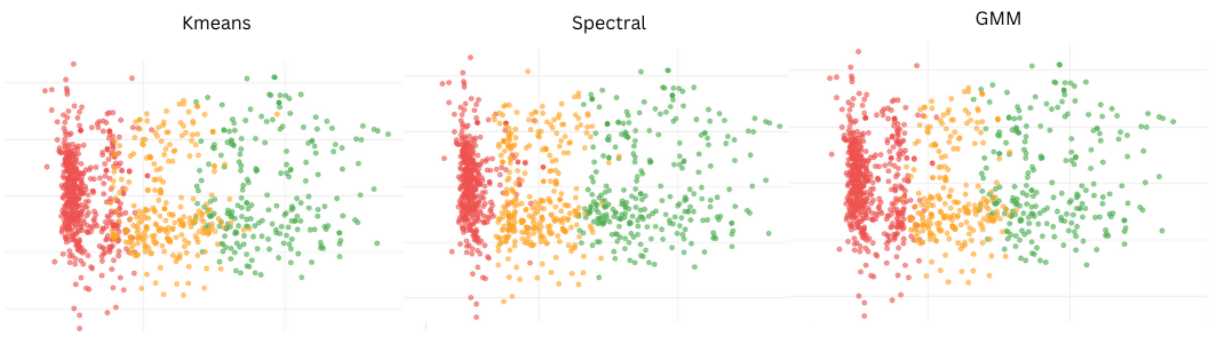


Figure 8: Clustering Scatterplots

K-Means achieved a silhouette score of 0.589, indicating decent horizontal separation between clusters. While the algorithm clearly separated the three groups, it struggled with distinguishing between "Good" and "Average" signals, with some "Average" signals being misclassified as "Good."

The GMM model performed the best, achieving the highest silhouette score of 0.61. This model showed the clearest horizontal separation between clusters, especially the "Bad" cluster. However, it did occasionally misclassify some "Average" signals as "Good," although this was less frequent than in K-Means. GMM's strong performance is reflected in the very little overlap between clusters and its ability to define clear boundaries.

Spectral Clustering, on the other hand, performed the weakest, with the lowest silhouette score of 0.482. This model struggled to separate the "Average" and "Bad" signals, often misclassifying "Bad" signals as "Average." Despite correctly identifying the very poor signals as "Bad," its significant overlap between clusters contributed to its lower score.

Overall, GMM emerged as the most reliable clustering method due to its strong separation of signals and minimal misclassification. Therefore, GMM's results will be used in further analysis.

6.2 Case Study 2: Clustering Results Analysis by Trading Signal Provider

In this case study, we study how the GMM cluster categorizes each trading signal provider based on their performance category. As shown in Figure 9, the signals from CryptoSignals_orge are overwhelmingly classified as "Bad," with 83.3% of the signals falling into poor category. In contrast, the counts for "Good" and "Average" quality signals are very low. BinanceKillersVip stands out among all the trading signal providers, with 32.2% of its signals classified as "Good" and 39.1% classified as "Average." This result cumulates to a total of 71.3% being either "Good" or "Average," which is considered a strong performance. The other providers, namely wallstreetofficial and cryptoclubpumpsignal, show more mixed results, with the majority of their signals being classified as "Average."

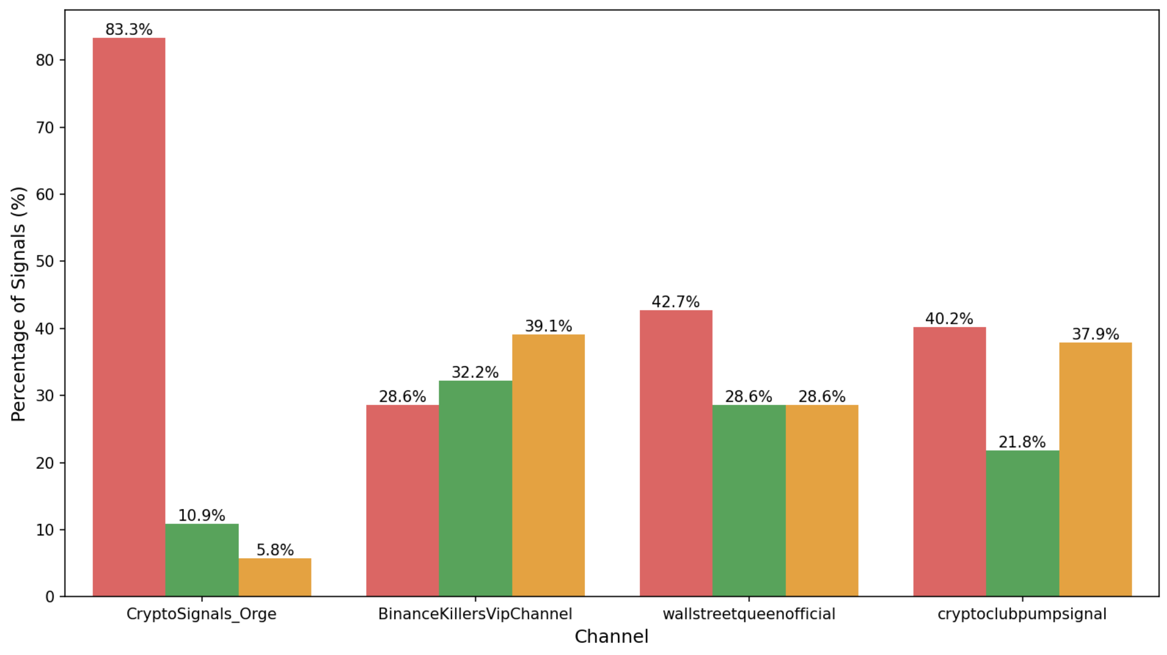


Figure 9: GMM Countplot For Each Provider

6.3 Case Study 3: Clustering Results Statistics Analysis

In this case study the performance of each signal provider using Gausian mixture Model (GMM) clustering is evaluated via several performance metrics.

Table 7 provides an overview of the distribution of signals across three categories: Good, Average, and Bad. This table shows that Bad signals (571) significantly outnumber Good signals (223), indicating that most of the signals are not reliable.

Signal Type	Total Count
Average	273
Bad	571
Good	223

Table 7: Total Counts by Signal Type

Table 8 extends this analysis to show specific breakdown of each signal types by providers, respectively. For instance, CryptoSignals_Orge is shown to have a particularly high number of Bad signals (374) underperforming in relation with other signal providers like BinanceKillersVipChannel, whose signal distribution is much balanced.

Metric	BinanceKillersVipChannel	CryptoSignals_Orge	cryptoclubpumpsignal	wallstreetqueenofficial
Average	119	26	33	65
Bad	87	374	35	97
Good	98	49	19	65

Table 8: Signal Counts by Channel

Furthermore, Table 9 delves into the success rate of each provider and finds that BinanceKillersVipChannel proves to be having the highest percentage of successful signals in all categories. In contrast, CryptoSignals_Orge's performance is much lower, further emphasizing its unreliability.

Signal Type	Channel	Count	Mean	Std Dev	Min	Max
Average	BinanceKillersVipChannel	119	29.62	8.45	19.31	44.44
	CryptoSignals_Orge	26	27.69	7.57	16.67	32.94
	cryptoclubpumpsignal	33	27.78	7.98	13.21	33.38
	wallstreetqueenofficial	65	29.56	7.73	14.29	42.12
Bad	BinanceKillersVipChannel	87	5.41	5.17	0.00	14.29
	CryptoSignals_Orge	374	0.53	2.94	0.00	18.27
	cryptoclubpumpsignal	35	5.71	8.03	0.00	14.31
	wallstreetqueenofficial	97	7.83	6.31	0.00	16.67
Good	BinanceKillersVipChannel	98	58.58	11.94	37.50	100.00
	CryptoSignals_Orge	49	68.00	13.68	51.12	88.75
	cryptoclubpumpsignal	19	69.30	12.75	53.00	100.00
	wallstreetqueenofficial	65	63.69	14.60	37.50	100.00

Table 9: Percentage Hit Statistics by Signal Type and Channel

Lastly, Table 10 consolidates all the performance metrics into a comprehensive overview, where we can equitably compare success rate,profit, loss, and signal duration by provider. From this table we can clearly see that BinanceKillersVipChannel is the top performing provider while CryptoSignals_Orge is not capable of returning a good risk adjusted returns and are always close to low success rates.

Metric	CryptoSignals_Orge	BinanceKillersVipChannel	wallstreetqueenofficial	cryptoclubpumpsignal
Total Signals	449.00	304.00	227.00	87.00
Success Rate (%)	19.38	78.51	73.70	63.56
Avg Profit	5.34	14.65	11.94	12.86
Avg Loss	0.32	-9.68	-0.33	2.44
Avg Duration (hrs)	51.80	280.41	371.90	194.74

Table 10: Channel Performance Overview

6.4 Discussion

6.4.1 Overview of experimental design

The goal of these experiments was to evaluate the performance and compare the results of different trading signal providers using KMeans, Gaussian mixture models (GMM) and Spectral clustering. The success rate of each provider were tested by categorizing the input signals as Good, Average, or Bad signals, and through an analysis of the following performance metrics: success rates, maximum profit, maximum loss, and average durations of the signals. This work provided an overall assessment of the performances using logical reasoning and the statistics produced with the help of historical data.

Experiments were designed so that clustering models can cluster the signals by their characteristics. This choice in design permitted identification of patterns both in each provider's signal behavior and in the differences in that behavior over time. Several of these experiments were met with success, but were heavily dependent on the quality of the data supplied and the underlying assumptions made in the clustering models, which introduces certain limitations that need to be critically discussed.

6.4.2 Evaluation of the Results

KMeans and GMM Clustering : One of the most notable findings was the consistency between GMM and Kmeans clustering methods particularly with the performance of CryptoSignals_Orge being have an entirely different fundamental architechture it still primarily labelled them as Bad. This alligns with the hypothesis that CryptoSignals_Orge are unreliable trading signal providers. From the results we can deduce that these models are good at identifying underperforming providers as both clustering methods provided almost same classification. This supports the literature done by, Chen et al. (2019) who evaluate the quality of trading signals in algorithmic trading using models similar to those used here and observed that trading signals with lower accuracy are more often found in 'poor category'. Though, the cluster boundaries used by both models have not been examined in detail to better understand what classifying factors contributions in segregation.
Performance Metrics Analysis : The performance metrics, including success rate, average profit, maximum profit, maximum loss, provided valuable insights into the profitability and risk associated with each provider's signals. For instance, BinanceKillersVipChannel demonstrated a strong performance across all metrics, with high success rates and favorable profit statistics in the Good signal category. This is consistent with prior studies that suggest that providers with more balanced success rates tend to offer more reliable signals. On the other hand, CryptoSignals_Orge showed dismal results, with low success rates and minimal profit, confirming its status as a less reliable provider. However, a major limitation of the performance metrics is the reliance on historical data, which may not always accurately reflect future performance. The use of historical success rates and profit margins, while informative, assumes that past performance is indicative of future results—a common pitfall in trading signal evaluations. It would be beneficial to extend the analysis to include backtesting in a dynamic market environment, where external factors such as market shifts or unexpected events (e.g., economic news) could influence signal success.
Analysis of Signal Duration : The average duration of signals was another area of interest. While some providers, such as CryptoSignals_Orge, exhibited much shorter signal durations, others like BinanceKillersVipChannel provided signals with much longer durations. This may indicate differing trading strategies, with longer-duration signals possibly suggesting a more strategic, long-term approach to trading. However, it would be essential to analyze the win rate relative to duration—whether short-duration signals are consistently more volatile or more likely to result in loss. Future experiments could look into clustering signals based on duration and performance to discern whether the length of time a signal remains active is a predictor of success.

6.4.3 Critique of Experiment Design and Suggestions for Improvement

While the design of the experiments provided useful insights, there are several areas where improvements could be made to enhance the results:

Data Quality and Granularity: The clustering models and performance metrics were based on historical trading signals. However, the data used in these experiments lacked additional context that might have been important, such as market conditions, asset volatility, or trader behavior. A more granular dataset, incorporating real-time data and market features, could improve the predictive power of the models.
Feature Selection: The experiments relied on basic signal features (e.g., profit, loss, duration). Future experiments could incorporate more sophisticated features such as market momentum, asset correlations, and even sentiment analysis from social media or news sources. This would provide a more complete picture of the signal's potential value.
Data Extraction: A more robust approach for extracting signal details (e.g., coin name, targets, entry prices) should involve the use of large language models. Trained on massive datasets, these models would automatically and accurately extract relevant trading information from any given text, improving the reliability of the data processing.
BTC Influence on ALT coins: Another factor to consider is the price movement of Bitcoin (BTC), as many altcoins tend to follow the same trend as BTC. It's important to note for whether a price change in an altcoin was driven by technical analysis or simply a result of BTC's price fluctuation.
Clustering Methodology: While KMeans, GMM, and Spectral Clustering were all effective in certain respects, further refinement of these models could be beneficial. For instance, using Hierarchical Clustering or DBSCAN could reveal more granular differences between providers, especially in the presence of noise or outliers.
Signal Categorization: The classification into Good, Average, and Bad signals was somewhat broad. A finer categorization could improve the accuracy of the results. For instance, instead of categorization, a regression result would be more insightful.

7 Conclusion and Future Work

This research successfully achieved its primary objective of evaluating the performance of trading signals through a comprehensive multi-step architecture. The findings highlight the strengths and limitations of the techniques employed. One significant insight was the use of regular expressions for named entity extraction. While these were robust and flexible, they struggled to capture patterns across diverse and inconsistent text formats, especially from new signal providers. Despite refinement efforts, the variability in textual data resulted in a high probability of missing entities presented in non-standard formats. For datasets with such variation, the study concludes that pretrained large language models (LLMs) are a more effective solution for data extraction, although requiring substantial computational resources and fine-tuning. Using a custom knowledge base and providing a large amount of training data, can detect named entities like coin names, targets, stop loss in a more reliable way. In comparison to the clustering algorithms, revealing varied classification. We deduced that simpler algorithms such as KMeans were less effective for higher dimensional data, as they depend on linear separations and only accomplish the ability to understand only single dimensional relationships. On the other hand, the performance of the Gaussian Mixture Model (GMM) and Spectral Clustering, had more pronounced boundaries between the clusters and fewer misclassifications. These results indicate that GMM followed by Spectral clustering are preferable for datasets with complicated feature distributions. To complement future work incorporating more advanced clustering techniques such as Hierarchical Clustering and DBSCAN could result in more refined signal groupings. Finally, this research quantified the performance of trading signal providers. Among those analyzed, BinanceKillersVIP emerged as the most profitable provider, while CryptoSignal_Orge consistently underperformed. The hypothesis was validated through multiple clustering algorithms, all of which produced consistent results. This study provides a strong foundation for future enhancements in trading signal evaluation, offering actionable insights and establishing a starting point for developing better solutions in this domain.

References

Bharati, R. and M.Upadhyaya. (2024). Svm-based sarcasm detection system: Nlp using heuristic approach, *2024 8th International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, Pune, India, pp. 1–6.

Bhavya, B. and M.Nidd. (2023). Exploring large language models for low-resource it information extraction, *2023 IEEE International Conference on Data Mining Workshops (ICDMW)*, Shanghai, China, pp. 1203–1212.

Carloni, F., Panseri, L., Conficconi, D., Sironi, M. and Santambrogio, M. D. (2023). Enabling efficient regular expression matching at the edge through domain-specific architectures, *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, St. Petersburg, FL, USA, pp. 71–74.

Christy, A. and Meeragandhi, G. (2020). Feature selection and clustering of documents using random feature set generation technique, pp. 67–79.

Dagdelen, J., Dunn, A., Lee, S., Walker, N., Rosen, A. S., Ceder, G., Persson, K. A. and Jain, A. (2024). Structured information extraction from scientific text with large language models, *Nature Communications* **15**(1): 1418.
URL: *https://doi.org/10.1038/s41467-024-45563-x*

Fariha, A., Gharavian, V., Makrehchi, M., Rahnamayan, S., Alwidian, S. and Azim, A. (2024). Log anomaly detection by leveraging llm-based parsing and embedding with attention mechanism, *2024 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, Kingston, ON, Canada, pp. 859–863.

Haghighat, M. H. and Li, J. (2018). Toward fast regex pattern matching using simple patterns, *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, Singapore, pp. 662–670.

K, A. and R, A. (2019). Limitations of information extraction methods and techniques for heterogeneous unstructured big data, *International Journal of Engineering Business Management* **11**.

Li, Y., Krishnamurthy, R., Raghavan, S., Vaithyanathan, S. and Jagadish, H. (2008). Regular expression learning for information extraction., pp. 21–30.

Mande, R., Yelavarti, K. C. and JayaLakshmi, G. (2018). Regular expression rule-based algorithm for multiple documents key information extraction, *2018 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India, pp. 262–265.

Matsuzaki, N., Fukumitsu, M. and Kita, Y. (2023). Emergency button: Evacuation of crypto asset when key loss, *2023 Eleventh International Symposium on Computing and Networking Workshops (CANDARW)*, Matsue, Japan, pp. 246–252.

N., R. and D., K. P. (2022). Crypto-currency price prediction using machine learning, *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)* pp. 1455–1458.

Reddy, P. N., S., S. A., Alapati, R. and D., R. (2024). Real-time tweets analysis using machine learning and bigdata, *2024 IEEE North Karnataka Subsection Flagship International Conference (NKCon)*, Bagalkote, India, pp. 1–6.

Rosenfeld, A., Ade-Ibijola, A. and Ewert, S. (2017). Regex parser ii: Teaching regular expression fundamentals via educational gaming.

Subramanian, P., M.J.Williams and Cheriyan., A. (2024). Should crypto integrate micro-finance option?, *2024 International Conference on Trends in Quantum Computing and Emerging Business Technologies*, Pune, India, pp. 1–7.

Wagh, R. and Punde, P. (2018). Survey on sentiment analysis using twitter dataset, pp. 208–211.

Yang, J., Wei, F., Huber-Fliflet, N., Dabrowski, A., Mao, Q. and Qin, H. (2023). An empirical analysis of text segmentation for bert classification in extended documents, *2023 IEEE International Conference on Big Data (BigData)*, Sorrento, Italy, pp. 2793–2797.

Zhang, S., Gu, X., Chen, Y. and Shen, B. (2023). Infere: Step-by-step regex generation via chain of inference, *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Luxembourg, Luxembourg, pp. 1505–1515.