SYNOPSIS

Cryptocurrencies are becoming increasingly relevant in the financial world and can be considered as an emerging market. The low barrier of entry and high data availability of the cryptocurrency market makes it an excellent subject of study, from which it is possible to derive insights into the behaviour of markets through the application of sentiment analysis and machine learning techniques for the challenging task of Cryptocurrency market prediction.

Cryptocurrency market prediction has always been regarded as a challenging task that has attracted attention from both academia and investors.The complexity of the task can be attributed to the multiple factors and uncertainties that interact in the markets including economic and political conditions, as well as human behaviour. Being able to consistently predict the market price movements is quite difficult, but not impossible. According to academic research, movements in the market prices are not random, but behave in a highly non-linear and dynamic way. Previous studies have also shown that it is not necessary to be able to foretell the exact value of the future price in order to make profit in financial predictions. In reality, predicting the market direction as compared to its value can result in higher profits.

This project proposes a Cryptocurrency price prediction model which is a combinational feature of technical analysis and sentiment analysis (SA). The features of sentiment analysis is based on information in the tweets where opinions are highly unstructured, heterogeneous and are either positive or negative, or neutral in some cases. The features of technical analysis based on historical cryptocurrency prices are used to train and develop Time-series models for cryptocurrency price prediction using deep-learning techniques to forecast future momentum. Experimental results show that the use of sentiment analysis and technical analysis achieves higher performance than that without sentiment analysis in predicting Cryptocurrency price.

CONTENTS

1. INTRODUCTION

News sources, especially social networks such as twitter are increasingly used as 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 analysing 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 utilising a linear model that takes as input tweets.

Technical analysis is based on historical developed regularities in the cryptocurrency exchange with an assumption that the same result will repeat in the future. That is identifying trends in the data to identify the moment of prices in the market.

Sentiment analysis of twitter/reddit data which is helpful to analyse the information in the tweets where opinions are highly unstructured, heterogeneous and are either positive or negative, or neutral in some cases.

There are multiple factors that make the cryptocurrency prices unstable that change rapidly over time and also makes its prediction difficult.Thus historical cryptocurrency prices are used to train and develop Time-series models for cryptocurrency price prediction using deep-learning techniques to forecast future prices.

1.1 PROBLEM STATEMENT

The proposed work aims at building a stock price prediction system using combinational features from sentimental analysis of stock news using twitter data and technical analysis of trading information.

1.2 OBJECTIVE

The main objective is developing a stock price prediction model based on technical analysis of trading information and sentiment analysis using twitter data .

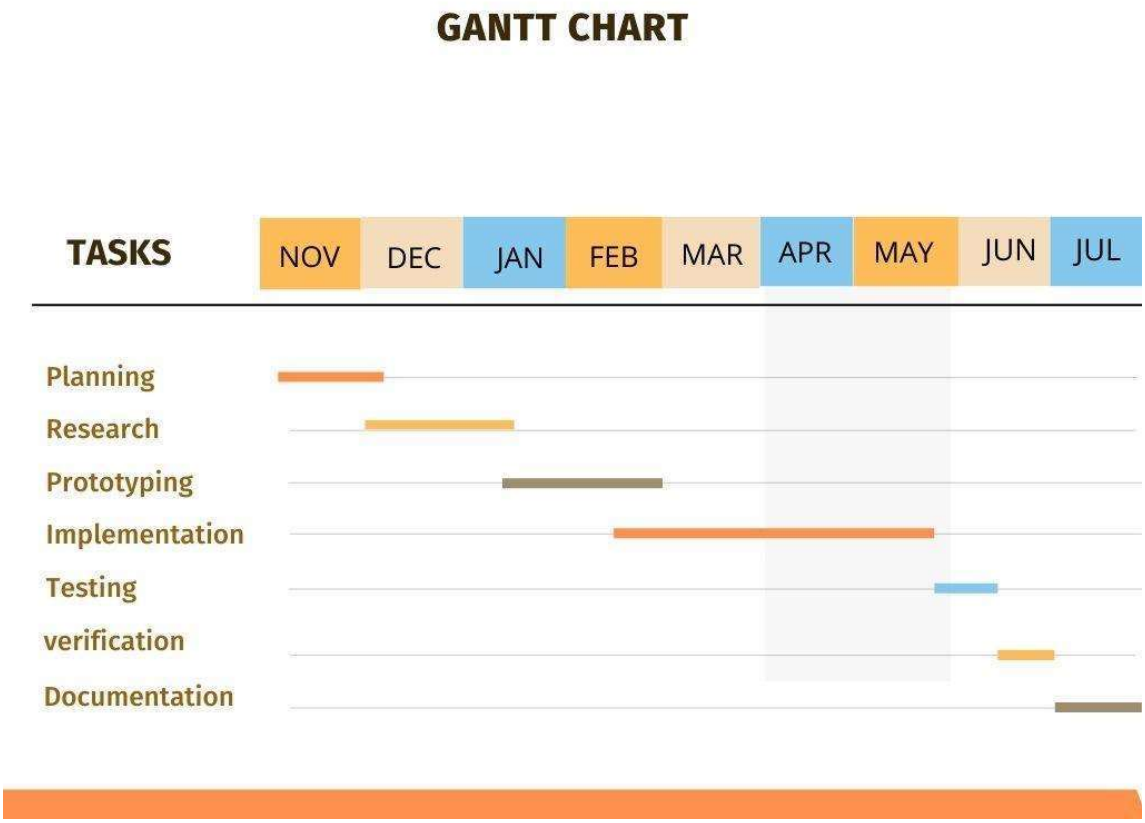
Achieving this entails:

- To scrape data from twitter and perform sentiment analysis on the scrapped tweets to get the public opinion about a certain topic.
- To preprocess the text and classify it as positive, negative or neutral tweets.
- To model Cryptocurrency prices correctly, so as an investor you can reasonably decide when to buy assets and when to sell them to make a profit.
- Providing a good machine learning model that can look at the history of a sequence of data and correctly predict what the future elements of the sequence are going to be.

1.3 MOTIVATION

The price of the cryptocurrencies has been a subject of curiosity for researchers across the globe.The prices of cryptocurrencies are volatile and are dependent on various factors such as transaction cost, mining difficulty, market trends, popularity, price of alternate coins, sentiments, and some legal factors. The aforementioned factors make the cryptocurrency prices unstable that change rapidly over time and also makes its prediction difficult. Hence, forecasting has been a very challenging and crucial task for the researchers.

1.4 GANTT CHART



2. LITERATURE SURVEY

Research Paper	Methods/Algorithm	Conclusion	Future Scope
[1] A Deep Learning-based Cryptocurrency Price Prediction Scheme for Financial Institutions.	The data is collected from the daily cryptocurrency prediction tables and then preprocess it. Ie:normalised the data using Min-Max normalisation to convert the values in the range of 0 to 1. Then, the data is split into two sets named as training and testing datasets. The proposed hybrid model is trained using training data. For forecasting, the last w (input window length) is fed into the model(GRU+LSTM), which predicts the next day's price. This predicted price is again fed into the model to forecast the next day's price consequently. This process is repeated a number of times equal to the prediction window length. The test data set is used to assess the performance after the prices have been forecasted.	• This paper presents a comprehensive study on the various existing schemes to predict the cryptocurrency prices. • A new deep learning, i.e., LSTM and GRU-based hybrid model is proposed to predict the prices of Litecoin and Monero cryptocurrencies accurately within the stipulated window sizes, i.e., 1,3, and 7. • Performance evaluation of the proposed hybrid model has been done using the evaluation matrices such as MSE, RMSE, MAE, and MAPE for Litecoin and Monero by comparing it with the traditional LSTM based approaches.	Introduce a more complex model along with the incorporation of sentiment data, which can improve the prediction results for cryptocurrencies
[2]Predictive Analysis of Crypto-currency Price Using Deep Learning.	The paper proposes a framework for predicting cryptocurrency price using deep learning techniques by considering the nonlinear nature of cryptocurrency price. The proposed approach consists of four units of LSTM input layers for	These model Predicts crypto-currency price by considering various factors such as market cap, volume, circulating supply, and maximum supply based on deep learning techniques such as the RNN and the LSTM.	Future research should extend the proposed approach by considering additional parameters such as the political environment, human relations, and regulations, which vary across countries.

	modelling and a sigmoid activation function for controlling the flow of information and memorising all patterns formed in cryptocurrency data. Keras with tensorflow is used as the backend library to make the model more accurate.The Keras sequential model consists of two layers named LSTM and dense layer	The performance of the model was checked using the most commonly used metrics like no. of epochs,Amount of losses,Correlation coefficient and accuracy on BTC and Litecoin dataset. The proposed model achieved 87% accuracy for 200M data of BTC. The model generated a maximum of 10% loss on 12 epochs, which is good for a prediction model.	
[3]Stock Price Prediction Using Combinational Features From Sentimental Analysis of Stock News and Technical Analysis of Trading Information.	1.Pre-process the text data and detect seed word set Extract sentiment features by PMI method .Also calculate sentiment intensity for each stock news SI = p_strength n_strength If SI >0 => +ve sentiment else negative sentiment 2.Calculate technical indices based on price and vol of training info to generate feature sets a)SA+TA b)TA+seed word of SA c)only TA and plot charts for them 3.SVR(support vector regression) model was used to learn prediction model On the kernel function selection, RBF functions were used to get better performance in SVR model 4.Daily future stock price was predicted by calculating avg sentiment intensity of stock news of each day	The proposed model can design various feature sets for stock price prediction. The use of sentimental analysis improves the prediction performance. The PMI method used can capture effective sentiment features from stock news.	Analyse the relationship between seed word and expansion word ie:sentence pattern,sequence and word distribution.

	and combining the technical indices to stock price prediction model		
[4]Combining Technical Analysis with Sentimental Analysis For Stock Price Prediction	There are 4 components in proposed models:1.Raw Data Preprocessing Component 2. Sentiment AnalysisComponent 3.Feature Extraction Component 4.Prediction & Evaluation Component They use SentiWordNet to do sentiment analysis. SentiWordNet can analyze a sentiment score (Positive, Negative, Objective) for each word. They use a preceding one week (7-days) data as the input features to predict stock prices of a day. Our prediction is one day ahead, so that the output is the change rate of stock prices from day T-1 to day T. First, the RDP component downloads data of time series, news, and comments; second, the SA component analyse the overall sentiment for every comments and news; third, the FE component extracts features from the three different kinds of data sources; then, the P&E component predicts the stock price based on a MKL regression framework and evaluates the prediction results based on some magnitude	In this paper, we studied a stock prediction model which includes Raw Data Preprocessing (RDP) component, Sentiment Analysis (SA) component, Feature Extraction (FE) component, Price Prediction and Evaluation (P&E) component. In all the MKL training periods, weights of technical analysis are relatively smaller than numerical dynamics and sentiment analysis, which shows that features extracted from sufficient news and comments may contribute more than features from historical prices or volumes for stock price prediction	They only have researched on news and comments of Engadget as the social networks source. Changing of the social network (e.g.Yahoo Finance) to investigate the news and comments of different web sources can be the future scope of this research. They have not considered the contextual dynamics (i.e., word order) of news and comments. Applying more deep text mining for sentiment analysis of news and comments with this research is another research direction

[5]Short-Term Stock Price Forecasting Based on Similar Historical Patterns Extraction .	Similar historical features extraction method for short-term stock price prediction using a modified dynamic time warping approach. A Suitable range of the length of historical subsequences is defined.	The Euclidean distance based and traditional dynamic time warping based similar measure Both approaches fail to detect the most similar part. This is because the length of each subsequence is fixed.So using the modified DTW the length of each sub-sequence is set to be in a suitable range rather than a fixed value. These provide the best estimation of the distance between the testing sequence and the historical sub-sequences.It turned out that the short-term stock price prediction results produced by the proposed approach are promising.	Historical features extraction method for short-term stock price prediction using a modified dynamic time warping approach was used. It turned out that the short-term stock price prediction results produced by the proposed approach are promising.
[6]Sentiment Analysi of Twitter Data : A Survey of Techniques	Mathematically we can represent an opinion as a quintuple (o, f, so, h, t),where t = time when the opinion is expressed. (o)Object:An entity which can be a person, event, product ,organisation, or topic (f)Feature:An attribute (or a part) of the object with respect to which evaluation is made. (so)Opinion orientation or polarity:The orientation of an opinion on a feature f represents whether the opinion is positive, negative or neutral . Opinion holder: The holder of an opinion is the person or organisation or an entity that expresses the	Research results show that machine learning methods, such as SVM and naive Bayes have the highest accuracy and can be regarded as the baseline learning methods. We can conclude that with cleaner data, more accurate results can be obtained.	Identifying subjective parts of text: Subjective parts represent sentiment-bearing content. The same word can be treated as subjective in one case, or an objective in some other. This makes it difficult to identify the subjective portions of text.

	opinion . ML Classification: 1.Naive Bayes 2.Maximum Entropy 3.SVM Levels of Sentiment Analysis: 1.Word level 2.Sentence level 3.Document level 4.Feature Based		
[7]Decision-Making Simulator For Buying and Selling Stock Market Shares Based on Twitter Indicators and Technical Analysis	Document-level and supervised methods were used. Each tweet is considered a document because it is atomic in its semantic context, enclosing in it all necessary elements and objectives for the identification of the global sentiment. The technical analysis strategies were EMA and MACD Crowd Analysis: (I)Twitter - Tw - the system only uses tweets; (II) Tw + EMA; (III) Tw + MACD; Technical Analysis: (I) EMA; (II) MACD.	This study showed that it is possible to obtain market dynamics information on the Twitter social network and this information could be used to compose stock buying and selling strategies.	It is important to note that the results obtained in the tables do not take into account the costs of financial operations. A next step in the development of architecture is to consider them. Another important issue is the effect on the results observed because of the relationship between the threshold setting and the volume of tweets collected., Regarding the profit objective, a careful study should be carried out to elaborate a model that defines a possible causal relationship between it, the algorithms of the architecture and the profit achieved.
[8]Forecasting with Twitter Data.	The first step deals with collecting Twitter data, cleaning and preprocessing in order to create a sentiment index. Once obtained the sentiment index, we apply statistical tests to determine the relation between the time series corresponding to the sentiment index and the target time series Then,	In this work we set out to assess whether the addition of public information extracted from Twitter is of any use for obtaining better time series predictions. A large number of studies have been published on the usage of data derived from social networks for improving predictions,	For the time series, we could include regression models in order to predict the actual values of the time series. Another improvement would be to consider ensembles of forecasting models as opposed to single models, given the recent popularity and

	proceed to apply forecasting models to predict the target time series both using the sentiment index and No using it, obtaining a table of results which tells us, for each experimental scenario given by a particular assignment of values to parameters,the observed accuracy.. Finally, use our summary trees to explore.	mostly on the subject of stock market and sales. implementation of a decision tree allowed us to orderly summarise the observations and Group the experiments into different sets of parameters that lead to similar results. Using the summary tree as a guide we extracted those instances of model and target that make best use of Twitter data	success of this approach in many practical tasks.

3. REQUIREMENT SPECIFICATION

3.1 TECHNOLOGIES USED

Programming languages: Python,

Frontend: ReactJs

Backend: Django , Machine Learning

3.1.1 PROGRAMMING LANGUAGES

PYTHON - LIBRARIES

NumPy

NumPy is a library adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. NumPy brings the computational power of languages like C and Fortran to Python.

Pandas

Pandas is a Python library for data analysis. A powerful and flexible quantitative analysis tool, pandas has grown into one of the most popular Python libraries written for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series

Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. Seaborn

NLTK

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language.

It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

RE - Regular Expression

This module provides regular expression matching operations. A regular expression specifies a set of strings that matches it; the functions in this module let you check if a particular string matches a given regular expression.

TEXTBLOB

TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

TWEEPY

An easy-to-use Python library for accessing the Twitter API.

Tweepy is open-sourced, hosted on GitHub and enables Python to communicate with Twitter platform and use its API using python. Tweepy includes a set of classes and methods that represent Twitter’s models and API endpoints, and it transparently handles various

implementation details, such as: Data encoding and decoding,HTTP requests ,Results pagination,OAuth authentication,Rate limits,Streams.

Note that Twitter levies a rate limit on the number of requests made to the Twitter API. To be precise, 900 requests/15 minutes are allowed; Twitter feeds anything above that is an error.

DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. The Django REST Framework (DRF) is a package built on top of Django to create web APIs. One of the most remarkable features of Django is its Object Relational Mapper (ORM) which facilitates interaction with the database in a Pythonic way. However, we can not send Python objects over a network, and hence need a mechanism to translate Django models in other formats like JSON, XML, and vice-versa. This sometimes challenging process, also called serialisation, is made super easy with the Django REST Framework.

PILLOW(PIL)

Python Imaging Library is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats.

Pillow is the friendly PIL fork by Alex Clark and Contributors. PIL is the Python Imaging Library, is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats.

DATETIME

The datetime module supplies classes for manipulating dates and times.The focus of the implementation is on efficient attribute extraction for output formatting and manipulation.

SKLEARN

Sklearn(also known as scikit learn) is a library for Python. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient

3.1.2 FRONTEND TECHNOLOGY

ReactJs

React is an open source, JavaScript library for developing user interface (UI) in web applications.

Currently, ReactJS is one of the most popular JavaScript front-end libraries which has a strong foundation and a large community.

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application.

It can be used to develop small applications as well as big, complex applications. ReactJS provides a minimal and solid feature set to kick-start a web application. React community compliments React library by providing a large set of ready-made components to develop web applications in a record time.

Material UI

Material UI is an open-source **React** component library that implements Google's Material Design.

It includes a comprehensive collection of prebuilt components that are ready for use in production right out of the box.

Material UI is beautiful by design and features a suite of customization options that make it easy to implement your own custom design system on top of our components.

3.1.3 BACKEND TECHNOLOGY

Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. It takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.The Django REST Framework (DRF) is a package built on top of Django to create web APIs. One of the most remarkable features of Django is its Object Relational Mapper (ORM) which facilitates interaction with the database in a Pythonic way . However, we can not send Python objects over a network, and hence need a mechanism to translate Django models in other formats like JSON, XML, and vice-versa. This sometimes challenging process, also called serialization, is made super easy with the Django REST Framework.

Machine Learning

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so.

3.2 MODEL USED

LONG-SHORT TERM MEMORY (LSTM)

It is a type of Recurrent neural network with extended features (mathematical functions and cell state in particular fashion) to overcome the problem of vanishing and exploding gradients. Due to the additional feature the LSTM network gets the advantage of memory. In RNN only the previous prediction was given as input with the new input therefore it was only able to remember short sequences. But due to the availability of memory in the LSTM network it stores and remembers the long sequences also. LSTMs are a special kind of RNN, which is also capable of long-term dependencies. These also have a chain-like structure instead of a single network layer. The core idea behind LSTMs:

The key of LSTM is the cell state, which runs down the entire chain, with some interactions. It's ability to add or remove info to cell state is regulated by structures like gates.

1.forget gate: takes the input from previously hidden layers and outputs 0 or 1. 0 means forget and 1 means remember.

$Ft = \sigma(Wf . [ht-1, xt]+ bf)$

2.Input Gate: decides what new information to update in cell state. It has two parts:

a. A sigmoid function which decides values to be updated.

$It = \sigma (wi . [ht-1, xt] + bi)$

b.tanh function which creates a vector of new candidate values.

$Ct = tanh (Wc . [ht-1, xt] + bc)$

3.Update gate: Update the old cell state, ct-1 information to new cell state ct. This is the new candidate values, scaled by to update each state value. $Ct = ft * ct-1 + it * ct$

4. Next output, it will be based on filtered version cell state. First sigmoid layer decides which parts of the cell state are going to be the output. Then put the cell state through tanh squish the values between -1 to +1 and multiply it by the output of the sigmoid function, so you can get only those parts you want.

The above diagram shows the repeating module of LSTM. Here xt is the actual input at time t, ht-1 is previous prediction (i.e. prediction at time t-1) and ct-1 is the cell state(the key idea behind LSTM) output of the previous repeating module. ht-1 and xtare concatenated and provided as input to the activation functions in the diagram. is the sigmoid function which gives output between 0 and 1, tanh is hyperbolic tan function which gives output between -1 and 1. Pointwise operation like vector addition takes place in the pink coloured circles. All the operations are performed and again the prediction and cell state output is produced at time t.

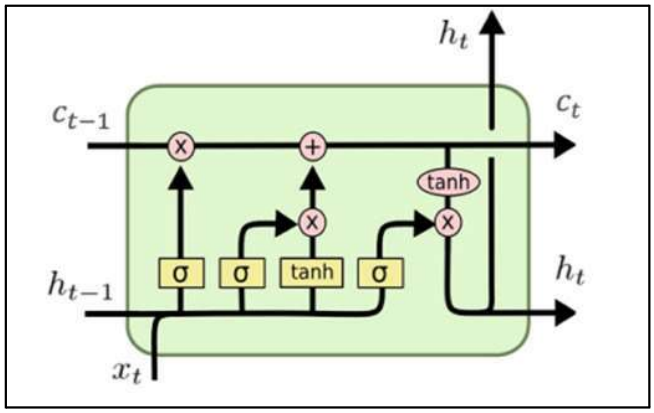


Fig 3.1 : LSTM

4. DESIGN

4.1 USE CASE DIAGRAM

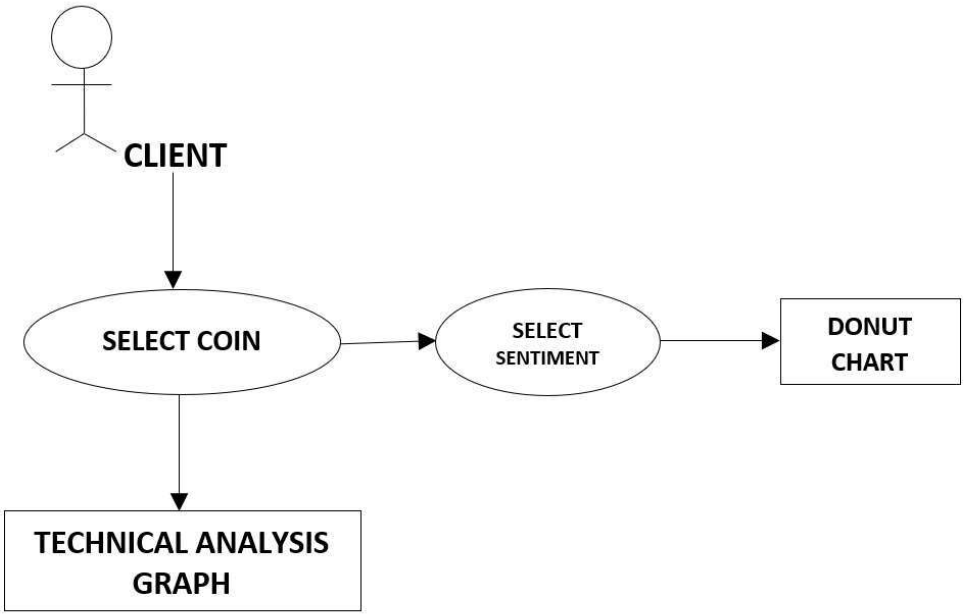


Fig 4.1 : Use case diagram

- The client can choose a coin from the list of coins available at the home page.
- The client will then be redirected to the single coin page where the technical analysis graph is displayed.
- There is a button on the page for sentiment,on clicking of which the sentiment analysis results are presented in the form of a donut chart.

4.2 SYSTEM ARCHITECTURE

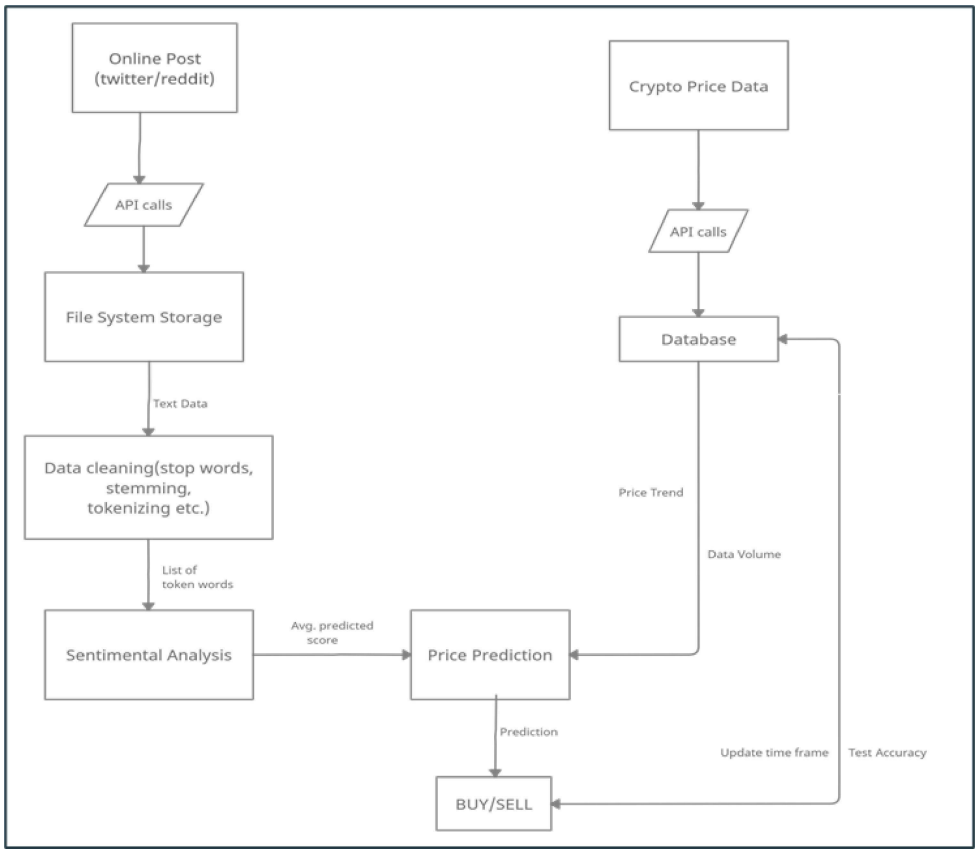


Fig 4.1 System Architecture

4.3 SENTIMENT BASED PREDICTION FLOW

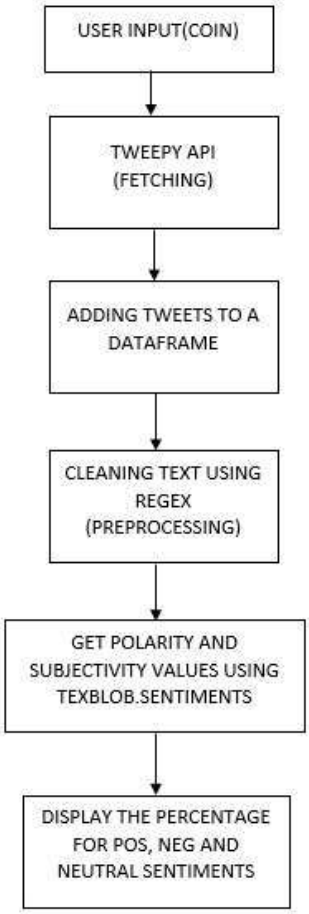
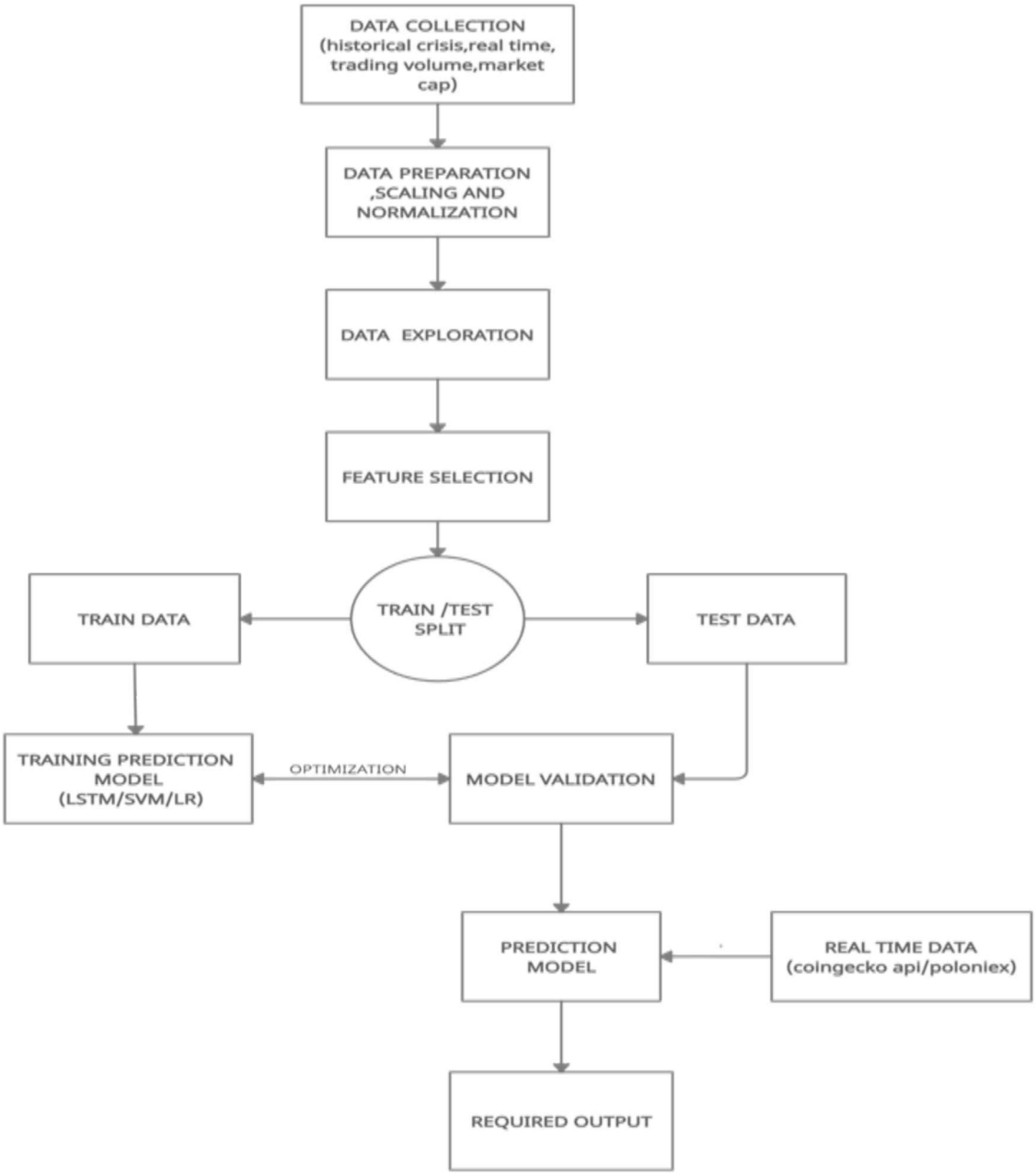


Fig 4.2 sentiment based prediction flow

4.4 TECHNICAL FEATURE BASED PREDICTION FLOW



5. IMPLEMENTATION

5.1 SENTIMENT ANALYSIS

Sentiment Analysis can help us decipher the mood and emotions of the general public and gather insightful information regarding the context. Sentiment Analysis is a process of analysing data and classifying it based on the need of the research.

These sentiments can be used for a better understanding of various events and impact caused by it.

5.1.1 SENTIMENT ANALYSIS IN PRICE PREDICTION

One of the important issues for forecasting market trends is to know the sentiment of stock news, whether it's a good or bad trend, when the financial stock prices go through the up/down cycle. The sentimental analysis (SA) can be applied to make trading decisions where some of the potentially important information affects investors to do investment.

The sentimental analysis is a different way to mining stock information compared to using the trading information to predict future stock trends. In addition, many researchers are using textual information to improve prediction performance . However, these approaches have a problem that textual data is highly complex information representation, whether using dictionary or manual lexicon by analyser may miss many of distinctive features.

In the machine learning/deep learning space, natural language processing (NLP) is one aspect used to solve a lot of problems, and sentiment analysis is a branch of NLP. In simple terms, sentiment analysis is the act of using NLP to help us understand the opinion of the public in a particular context in natural language.

Popular applications of sentiment analysis

- Social media sentiment analysis

- Movie reviews analysis
- News sentiment analysis

The evolution in NLP has been remarkable because we now have tools and packages that make solving problems with NLP very easy. One of those tools is Textblob.

Our main objective of sentiment analysis is to scrape data from twitter using tweepy and perform sentiment analysis on the scrapped tweets to get the public opinion about a certain topic.

To scrape data from twitter we need a Twitter developer account. Further, we require NLTK and Tweepy to be installed.

5.1.2 INSTALLING TWEEPY AND OTHER LIBRARIES

Tweepy is an open source Python package that gives you a very convenient way to access the Twitter API with Python. Tweepy includes a set of classes and methods that represent Twitter's models and API endpoints, and it transparently handles various implementation details, such as:

- Data encoding and decoding
- HTTP requests
- Results pagination
- OAuth authentication
- Rate limits
- Streams

If you weren't using Tweepy, then you would have to deal with low-level details having to do with HTTP requests, data serialisation, authentication, and rate limits. This could be time consuming and prone to error. Instead, thanks to Tweepy, you can focus on the functionality

you want to build. Almost all the functionality provided by Twitter API can be used through Tweepy.

```
from textblob import TextBlob
from wordcloud import WordCloud
import tweepy
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import re
plt.style.use('fivethirtyeight')
```

Fig 5.1 : installing tweepy and other libraries

5.1.3 TWITTER AUTHENTICATION

After importing the required modules, we need to connect to Twitter API which we accomplish by creating an authentication object.

Tweepy tries to make OAuth as painless as possible for you. To begin the process we need to register our client application with Twitter. Create a new application and once you are done you should have your consumer token and secret.

Step 1 — Install pip and Tweepy

Step 2 — Create Your Twitter Application

Step 3 — Modify Your Application's Permission Level and Generate Your Access Tokens

Step 4 — Create a Python application to authenticate with Twitter

```
def connect():
  # Replace the xxxxx with your twitter api keys
  consumer_key = 'xxxxx'
  consumer_secret = 'xxxxx'
  access_token = 'xxxxx'
  access_token_secret = 'xxxxx'

  try:
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    api = tweepy.API(auth)
    return api
  except:
    print("Error")
    exit(1)
```

Fig 5.2 : twitter authentication

5.1.4 PREPROCESSING

```python
def percentage(part,whole):
 return 100 * float(part)/float(whole)

keyword = input('Please enter keyword or hashtag to search: ')
noOfTweet = int(input ('Please enter how many tweets to analyze: '))

tweets = tweepy.Cursor(api.search, q=keyword, lang='en', tweet_mode='extended').items(noOfTweet)
positive = 0
negative = 0
neutral = 0
polarity = 0
tweet_list = []
neutral_list = []
negative_list = []
positive_list = []

i = 1


for tweet in tweets:
  tweet_list.append(tweet.full_text)
  # print(str(i) + ')' + tweet.full_text)
  i = i+1
```

Fig 5.3 : Fetch tweets

What we primarily do here is remove punctuations, all the mentions (@ username), stop words (words like ‘the’, ‘and’ etc.). These parameters are of very little significance as they do not give any indication of sentiments and removing these saves us memory, computational power and sometimes also increases the accuracy of the model.

We are using regular expressions here to do the preprocessing.

Creating a dataframe with column called tweets

```python
df = pd.DataFrame(tweet_list , columns = ['Tweets'])

df.head()
```

	Tweets
0	RT @BitcoinMagazine: #Bitcoin is freedom money...
1	RT @gemhostofficial: 🔥 HYPE IDO WHITELIST $SST...
2	RT @allgiveawaysda1: 🎁 #Gift & #Giveaway 🎁...
3	RT @WatcherGuru: We're giving away $5,000 #Bit...
4	#Podcast on the #TerraLuna & #UST crash.\n...

Fig 5.4 : Display Tweets

We then fetch the tweets of a particular coin the user wants to check using Twitter API and then using pandas create a Dataframe (Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns).

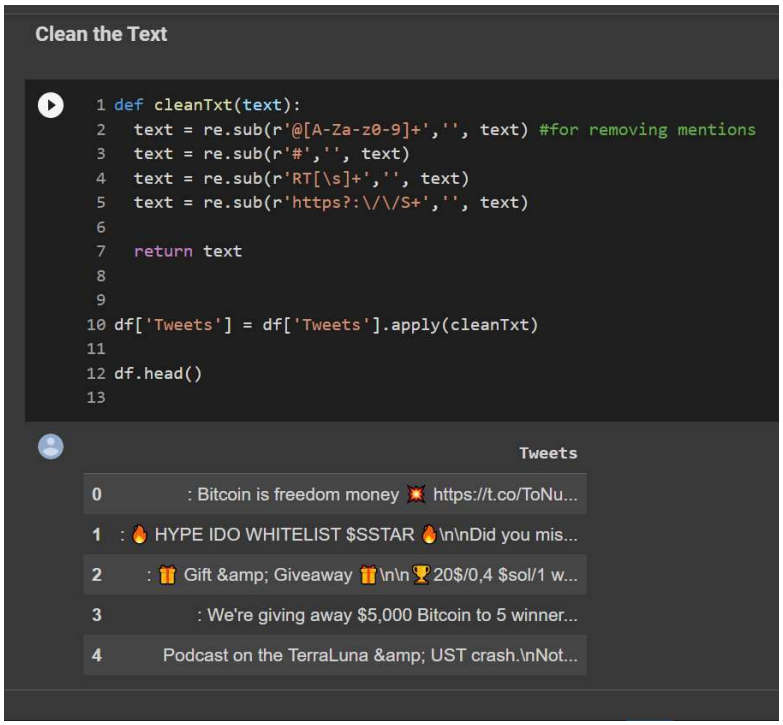


Fig 5.5 : cleaning the text

We clean the tweets using Regex:

- remove all the mentions
- remove All the hashtags
- remove RT(retweet)
- remove any links if present

5.1.5 SENTIMENT ANALYSIS USING TEXTBLOB

TextBlob is a python library for Natural Language Processing (NLP). TextBlob actively uses Natural Language ToolKit (NLTK) to achieve its tasks. NLTK is a library which gives easy access to a lot of lexical resources and allows users to work with categorization, classification

and many other tasks. TextBlob is a simple library which supports complex analysis and operations on textual data.

```
[ ]
def getSubjectivity(text):
  return TextBlob(text).sentiment.subjectivity


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


df['Subjectivity'] = df['Tweets'].apply(getSubjectivity)
df['Polarity'] = df['Tweets'].apply(getPolarity)


df.head()
```

	Tweets	Subjectivity	Polarity
0	: Bitcoin is freedom money 💥 https://t.co/ToNu...	0.000000	0.000
1	: 🔥 HYPE IDO WHITELIST $SSTAR 🔥\n\nDid you mis...	0.000000	0.000
2	: 🎁 Gift & Giveaway 🎁\n\n🏆20$/0,4 $sol/1 w...	0.000000	0.000
3	: We're giving away $5,000 Bitcoin to 5 winner...	0.000000	0.000
4	Podcast on the TerraLuna & UST crash.\nNot...	0.383333	0.325

Fig 5.6 : get polarity and subjectivity

We perform sentiment analysis by using the TextBlob object.

We first convert the text into a TextBlob object and then use .sentiment method to see whether the given statement is positive, negative or neutral.

Using the .sentiment method, we get the polarity and subjectivity of each statement.

Polarity is an integer ranging from [-1, 1] where -1 denotes that the statement is negative and 1 denotes that the given statement has a positive tone.

Subjectivity tells us whether the given statement is a personal opinion or if it is based on facts.It basically quantifies the amount of personal opinion and factual information contained in the text. The higher subjectivity means that the text contains personal opinion rather than factual information.

```
[ ]  1 def getAnalysis(score):
     2   if score < 0:
     3     return 'Negative'
     4   elif score == 0:
     5     return 'Neutral'
     6   else:
     7     return 'Positive'
     8
     9 df['Analysis'] = df['Polarity'].apply(getAnalysis)
    10
    11
    12 df.tail()
```

	Tweets	Subjectivity	Polarity	Analysis
495	Altoogle Crypto Mood\n(algo generated index)\n...	0.000000	0.000000	Neutral
496	Interest rates are rising though, no more pri...	0.500000	-0.250000	Negative
497	: $KCS I think Kucoin is setting up to be the ...	0.362963	0.114815	Positive
498	: BOOOOM💥💥💥\n\nKILLER'S ON https://t.co/LQ78kA...	0.500000	-0.312500	Negative
499	__: Good night everyone! Back to the grind fr...	0.475000	0.306250	Positive

Fig 5.7 : analyse the score

```
j =0
sortedDF = df.sort_values(by=['Polarity'])
for i in range(0, sortedDF.shape[0]):
  if(sortedDF['Analysis'][i] == 'Positive'):
    j = j + 1

k =0
sortedDF = df.sort_values(by=['Polarity'])
for i in range(0, sortedDF.shape[0]):
  if(sortedDF['Analysis'][i] == 'Negative'):
    k = k + 1


print('Total Positive Tweets are: ' + str(j))
print('Total Negavtive Tweets are: ' + str(k))


avgSub = df['Subjectivity'].mean()

avgSub
```

```
Total Positive Tweets are: 159
Total Negavtive Tweets are: 93
1.0
```

Fig 5.8 : printing positive and negative tweets

Calculating Percentage

```
ptweets = df[df.Analysis == 'Positive']
ptweets = ptweets['Tweets']

round((ptweets.shape[0] / df.shape[0])*100, 1)
```

```
60.0
```

```
plt.title('Sentimental Analysis')
plt.xlabel('Sentiment')
plt.ylabel('Counts')
df['Analysis'].value_counts().plot(kind='bar')
plt.show
```

```
<function matplotlib.pyplot.show>
```

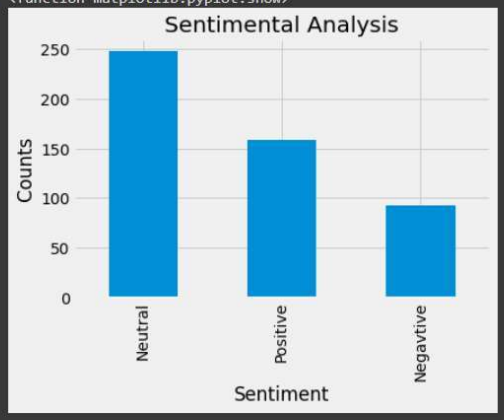


Fig 5.9 : calculating the percentage

5.2 TECHNICAL ANALYSIS

Stock information has multiple categories, i.e. stock closed price, trading volume, stock news, stock message, and expert analysis. Many studies have been done to predict the stock price by using statistics and machine learning techniques using historical stock price and trading volume . In general, technical analysis (TA) is based on historical developed regularities in the stock exchange with an assumption that the same result will repeat in the future .
There are many influential indicators and trading rules based on them. Technical indicators might provide advice to traders on whether a trend will continue or whether a stock is oversold or overbought. Investment managers calculate different indicators from available data and plot them as charts. Observations of price, direction, and volume on the charts assist managers in making decisions on their investment portfolios.

CoingeckoAPI : CoinGecko has independently sourced crypto data such as live prices, trading volume, exchange volumes, trading pairs, historical data, contract address data, crypto categories, crypto derivatives, images, and more. The Free API has a rate limit of 50 calls/minute.

TIME SERIES : Time series analysis is a specific way of analysing a sequence of data points collected over an interval of time. In time series analysis, analysts record data points at consistent intervals over a set period of time rather than just recording the data points intermittently or randomly. However, this type of analysis is not merely the act of collecting data over time.

5.2.1 DATA EXPLORATION

Importing Required Libraries

First, all the identified libraries are imported. Any missing libraries are installed using PIP.

```
from pycoingecko import CoinGeckoAPI
import pandas as pd
from datetime import datetime
import matplotlib.pyplot as plt
import plotly.express as px
import numpy as np
import datetime
import time
```

```
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler,LabelEncoder
from tensorflow.keras import layers
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
```

Fig 5.10 : Importing libraries

Function for API call to get Data of Specified coin for last 365 days (mean data of each day)

```
def initial_call_data(coin_name):

  #api call
  cg = CoinGeckoAPI()
  try:
    print(cg.ping())
    coin_data = cg.get_coin_market_chart_by_id(f'{coin_name}' , vs_currency = 'USD' , days = '356')
    return coin_data

  except :
    print('something went wrong')

```

Fig 5.11 :API call function

Step 1 — Initialize CoingeckoAPI

Step 2 — ping the api to ensure connection has been established and the api is working.

If ping does not work got to step 4

Step 3 — use the cg.get_coin_market_by_id to get 365 days of data.

Step 4 — print error and go to step 1

Function for cleaning data received from API call

```
def preprocess_data_to_store(coin_data):

  #create dataframe
  coin_prices_temp_df = pd.DataFrame(coin_data['prices'])
  coin_prices_temp_df.rename(columns={0:'Time', 1 : 'prices'}, inplace =True)
  coin_market_temp_df = pd.DataFrame(coin_data['market_caps'])
  coin_market_temp_df.rename(columns={0:'Time', 1 : 'market_caps'}, inplace =True)
  coin_totalvolumes_temp_df = pd.DataFrame(coin_data['total_volumes'])
  coin_totalvolumes_temp_df.rename(columns={0:'Time', 1 : 'total_volumes'}, inplace =True

  #merge
  coin_data_df = pd.merge(coin_market_temp_df ,coin_totalvolumes_temp_df , on = 'Time'  )
  coin_data_df = pd.merge(coin_data_df ,coin_prices_temp_df , on = 'Time'  )
  print(coin_data_df.size)
  #time unix to human
  coin_data_df['Time'] = coin_data_df['Time'].floordiv(1000)
  for i in range(357):
    temp_date = datetime.fromtimestamp(coin_data_df['Time'][i])
    coin_data_df['Time'][i] = temp_date.strftime('%d/%m/%Y')

  return coin_data_df
```

Fig 5.12 : cleaning data function

Preprocessing functions need to be performed on the dataset to make it usable such as:

- Renaming
- Concatenating,
- Merging
- formatting of data.

Function Plotting Data using PlotlyExpress (which gives a more Interactive graph)

```
def coin_graph_plot(coin_data_df):
  fig = px.line(coin_data_df, x="Time", y="prices", title='1Y Coin Price')
  fig.update_layout(
      xaxis = dict(tickmode = 'linear',
          tick0 = 5,dtick = 20,linecolor = 'black'),

      yaxis = dict(
          linecolor = 'black'),
  )
  fig.show()
```

Fig 5.13 : Plotting Data using PlotlyExpress

Defining the graphs and attributes of the graph as a function for ease of use.

Main Function to test functions created

```
def main():
  coin_name = input(" enter required coin name")
  coin_data = initial_call_data(coin_name)
  coin_data_df = preprocess_data_to_store(coin_data)
  print(coin_data_df)
  coin_graph_plot(coin_data_df)
  #adf_test(coin_data_df['prices'])

if __name__ == "__main__":
    main()
```

Fig 5.14 : Main Function

OUTPUT

```
 enter required coin nameethereum
{'gecko_says': '(V3) To the Moon!'}
1428
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:19: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:670: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

          Time   market_caps  total_volumes       prices
0    27/01/2021  1.554466e+11   4.036749e+10  1355.233724
1    28/01/2021  1.417741e+11   3.875241e+10  1253.141672
2    29/01/2021  1.528774e+11   3.266889e+10  1328.773619
3    30/01/2021  1.583541e+11   4.945450e+10  1380.284259
4    31/01/2021  1.574107e+11   2.958191e+10  1372.427229
..          ...           ...            ...          ...
352  14/01/2022  3.880399e+11   1.399977e+10  3256.758866
353  15/01/2022  3.946646e+11   1.231629e+10  3311.986061
354  16/01/2022  3.968992e+11   8.659658e+09  3331.246655
355  17/01/2022  4.002322e+11   9.002834e+09  3356.295444
356  17/01/2022  3.895859e+11   9.968681e+09  3263.553663

[357 rows x 4 columns]
```

Fig 5.15 : output of cryptocurrency details

Fig 5.16 : coin price graph

5.2.2 DATA COLLECTION

Using the previously identified preprocessing techniques to fetch relevant data

```
def get_start_dates(end):
  end = datetime.datetime.fromtimestamp(end)
  start = end - datetime.timedelta(days=50)
  start=start.timestamp()
  return start
```

```
[ ] end = int(datetime.datetime.now().timestamp())
    start = get_start_dates(end)
    merge_df = pd.DataFrame()
    for _ in range(14):
      update_data=cg.get_coin_market_chart_range_by_id('bitcoin' , vs_currency = 'USD' , from_timestamp=start, to_timestamp=end)
      update_df = pd.DataFrame(update_data['prices'])
      update_df.rename(columns={0:'Time', 1 : 'prices'}, inplace =True)
      update_df['Time'] = update_df['Time'].floordiv(1000)
      frames = [update_df, merge_df]
      merge_df = pd.concat(frames)
      end=start
      start = get_start_dates(start)
```

```
[ ] merge_df=merge_df.reset_index()
    merge_df= merge_df.drop('index', 1)
```

Fig 5.17 : Fetch data

```
[ ]  merge_df
```

	Time	prices
0	1589900944	9680.679052
1	1589904278	9669.623987
2	1589907869	9682.944667
3	1589911511	9661.065134
4	1589915082	9695.462600
...	...	...
16807	1650362622	40645.731265
16808	1650366201	40771.475941
16809	1650369783	40774.029229
16810	1650373344	40997.694520
16811	1650376998	41381.634401

16812 rows × 2 columns

Fig 5.18 : data fetched

```
final_data = merge_df
```

```
final_data

0         9680.679052
1         9669.623987
2         9682.944667
3         9661.065134
4         9695.462600
              ...
16807    40645.731265
16808    40771.475941
16809    40774.029229
16810    40997.694520
16811    41381.634401
Name: prices, Length: 16812, dtype: float64
```

Fig 5.19 : final data output

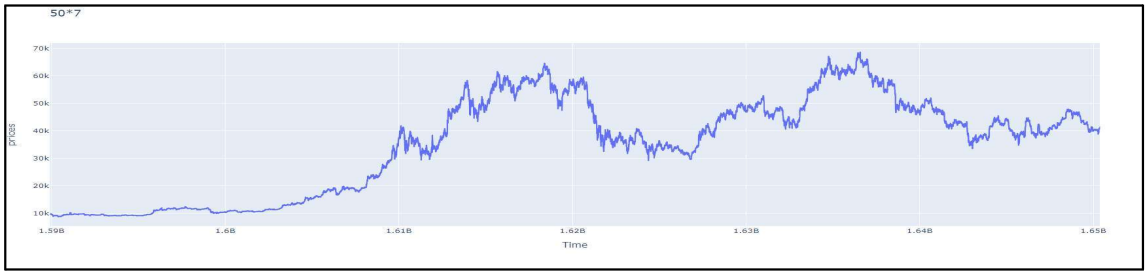


Fig 5.20 : final data graph

5.2.3 DATA PREPARATION

Reshaping and Train_Test split

```
[ ] hist_scaled = hist_scaled.reshape((len(hist_scaled), length, 1))
    print(hist_scaled.shape)

    (14796, 2016, 1)

    X_train = hist_scaled[:split_value,:,:]
    X_test = hist_scaled[split_value:,:,:]

    y_train = target_scaled[:split_value,:]
    y_test = target_scaled[split_value:,:]
```

Fig 5.21 : Train_Test split

5.2.4 MODEL TRAINING

```
model = tf.keras.Sequential()

model.add(layers.LSTM(units=32, return_sequences=True,
                  input_shape=(2016,1), dropout=0.2))

model.add(layers.LSTM(units=32, return_sequences=True,
                  dropout=0.2))

model.add(layers.LSTM(units=32, dropout=0.2))

model.add(layers.Dense(units=1))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_3 (LSTM)               (None, 2016, 32)          4352

 lstm_4 (LSTM)               (None, 2016, 32)          8320

 lstm_5 (LSTM)               (None, 32)                8320

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 21,025
Trainable params: 21,025
Non-trainable params: 0
_________________________________________________________________
```

Fig 5.22 : model training

5.2.5 MODEL OUTPUT

```
[ ] pred_transformed = sc.inverse_transform(pred2)
    y_test_transformed = sc.inverse_transform(y_test)
```

```
plt.figure(figsize=(12,8))
plt.plot(y_test_transformed[val:], color='blue', label='Real')
plt.plot(pred_transformed[val:], color='red', label='Prediction')
plt.title('Bitcoin Price Prediction')
plt.legend()
plt.show()
```

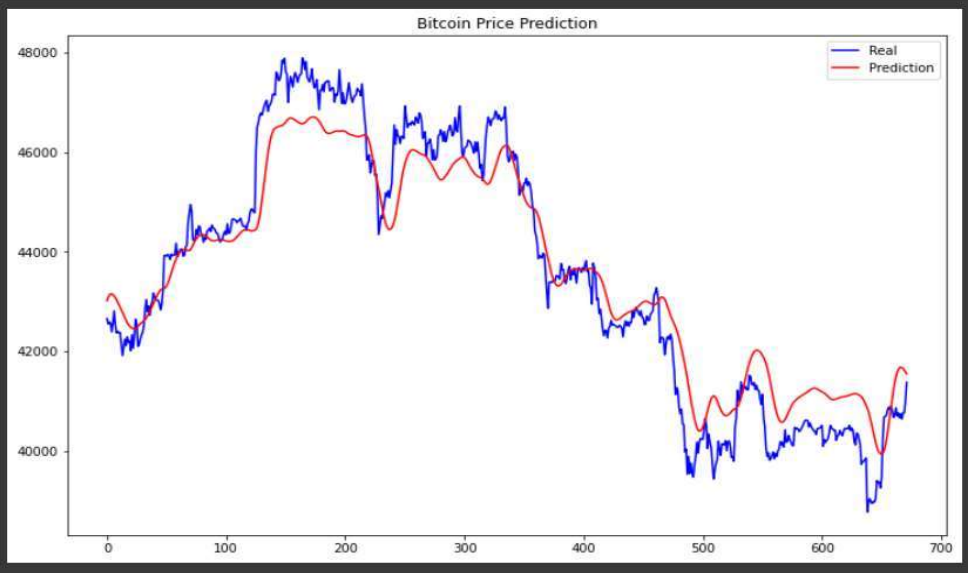
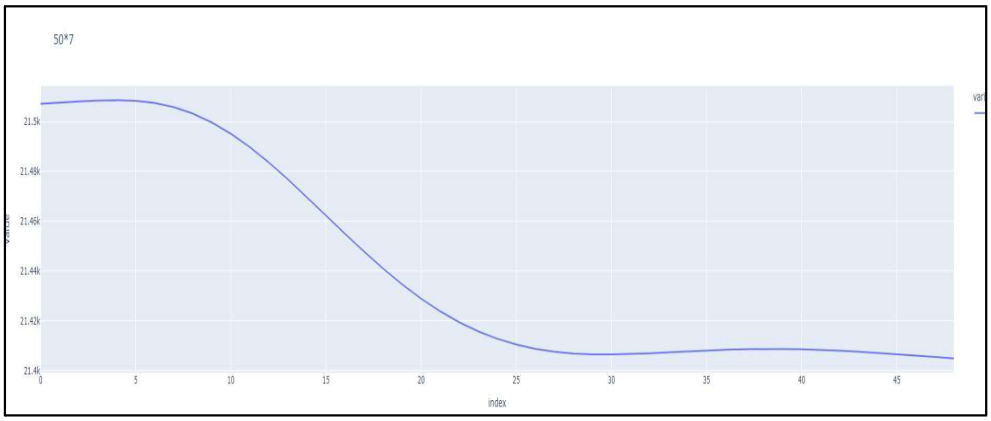


Fig 5.23 : price prediction output



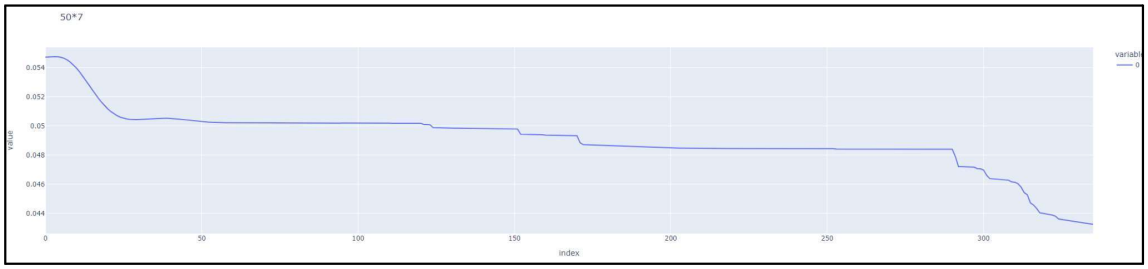


Fig 5.24 : price momentum

5.3 FRONTEND AND BACKEND

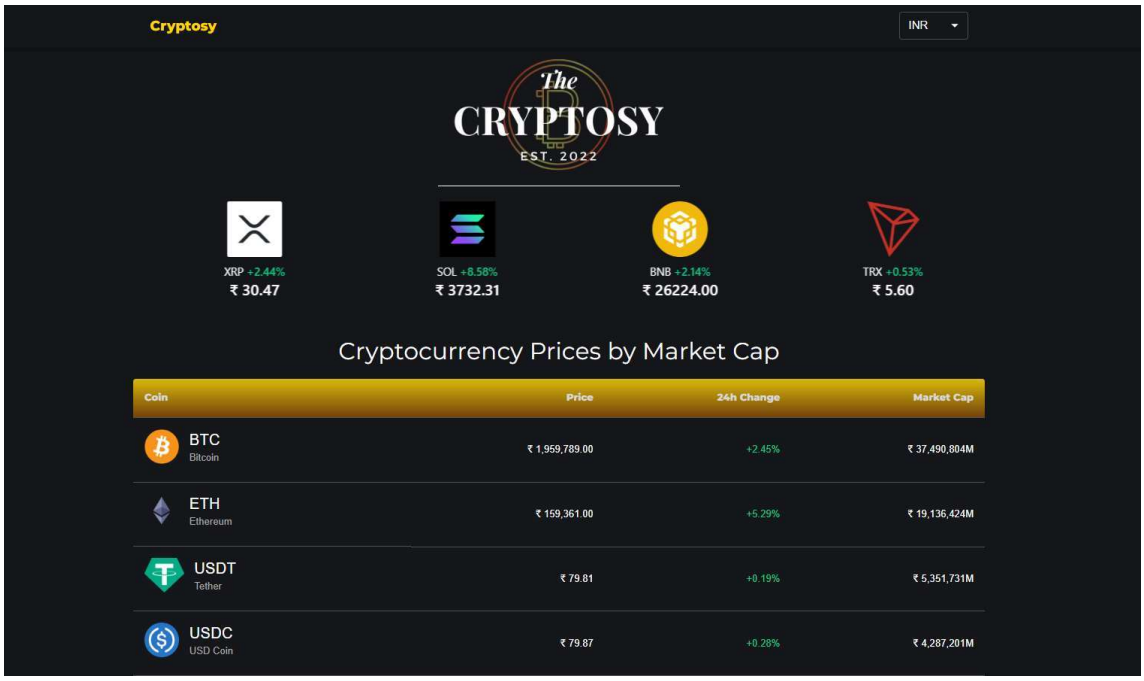


Fig 5.25 : Homepage

5.3.1 WORKFLOW

The Backend is used for REST APIs .It contains two REST APIs

One for Technical analysis and other for Sentiment Analysis.

The flow is as follows :-

In the Urls.py file for a particular url path we define a function in views.py

The Views.py will run the function and will return an html page with the

corresponding data.

The Homepage contains various Cryptocurrencies coin List .

Say you clicked any one of the coins, let's say Bitcoin the url becomes

‘http://localhost:3000/coins/bitcoin’ .

The page will show a brief information of bitcoin , a technical graph of momentum (y-axis) vs time(x-axis)(in hrs) and a button Labelled as Sentiments .

Fig 5.26 : coin page

Now as the page renders the Technical Analysis Graph loads . This Graph calls the REST API of technical analysis defined in views.py which is in the backend.

Code snippet for REST API of technical analysis is shown below.

```python
@api_view(["POST"])
def TechnicalAnalysis(coinid):
    try:
        import json
        import csv
        data= json.loads(coinid.body)
        temp_coin=data['h'] #naming the key as h while accepting the json data-POST method
        coin_name=str(temp_coin).lower()
        #schedule.every(2).day.at('00:00').do(func_api.run(coin_name))
        coinlist=['bitcoin','ethereum','ripple']
        if coin_name not in coinlist:
            coin_name='bitcoin'
        func_api.run(coin_name)

        f= open(f'{coin_name}.csv',"r")
        reader=csv.reader(f)
        people={}

        for row in reader:
            people[row[0]]={'magnitude':row[1]}

        return JsonResponse(people)
    except ValueError as e:
        return Response(e.args[0], status.HTTP_400_BAD_REQUEST)
```

Fig 5.27 : code snippet for REST api

5.3.2 FUNCTIONING OF REST API

It takes in the coin name and does the prediction using the trained machine learning model . In this case the ‘bitcoin’ model is loaded and predictions are done .The values are stored in a bitcoin.csv file . This file is read and sent as an API response in the JSON RESPONSE form. The response consists of 50 momentum and time values each.The Graph uses these values to print .

Next is On clicking the sentiments button a Donut Chart is displayed . Donut chart uses REST API data .The REST API for Sentiment Analysis is called when the button is clicked . The API is https://cryptosyapi.herokuapp.com/SentimentAnalyzer/ .

For the Sentiment Analysis we use live tweets to find the current sentiments for a particular coin in this case bitcoin . The tweets are obtained from twitter using tweepy library. And necessary Techniques are used to obtain the sentiment scores. The response is the value of sentiments i.e Positive , Neutral , Negative and Subjectivity values. In our case the Donut chart shows bitcoin live sentiment (Fig : 5.28)

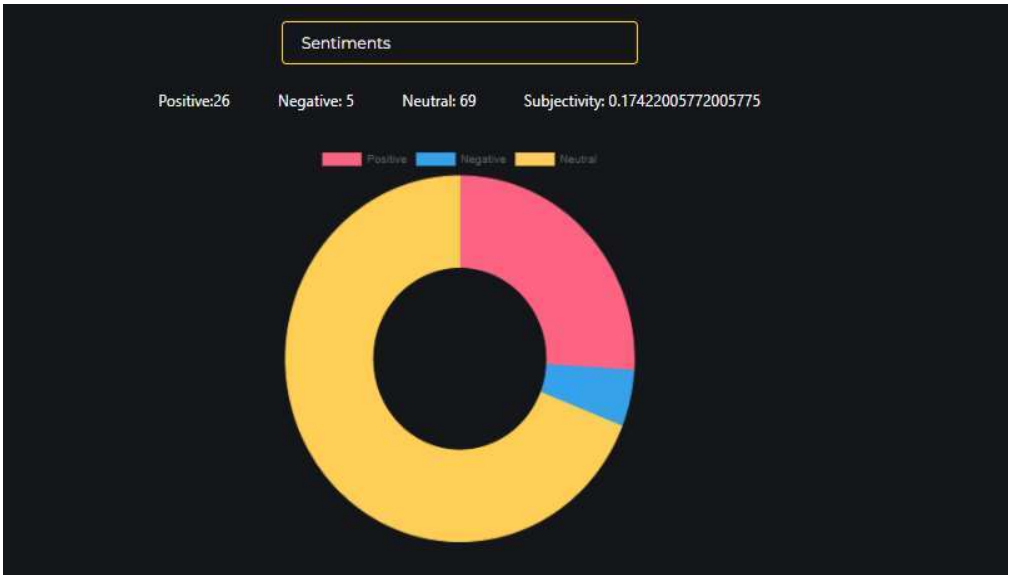


Fig 5.28 : Donut chart for bitcoin sentiments

6. RESULTS

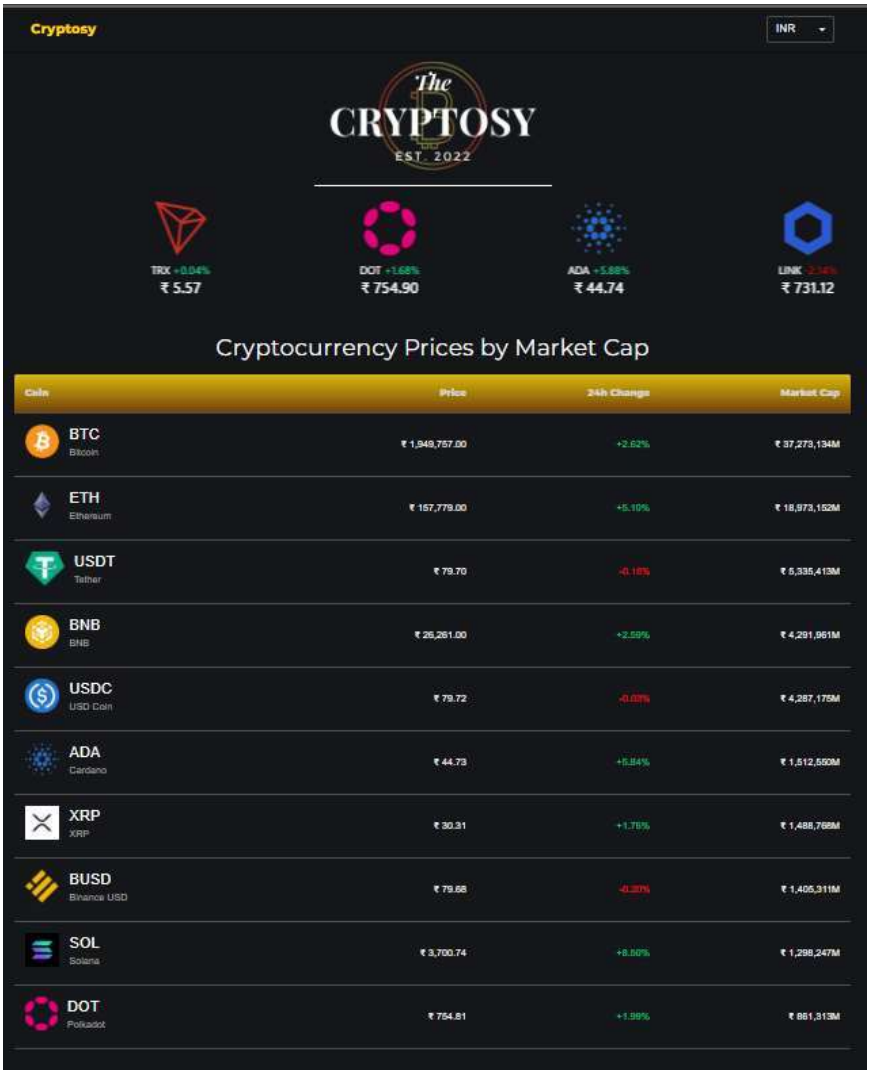


Fig 6.1: Homepage

This is the homepage that displays current prices of top ten cryptocurrencies.

In the top right corner ,there is an option to select the currency(INR/UDC) as shown in fig 6.2. This uses context api that provides the select state to the whole of the app.

On selecting INR, all the values will be displayed in Indian Rupees whereas on selecting UDC it will be displayed in US Dollar.

Just above the coin table, the trending coins carousel is displayed.This is achieved by using coinGecko/trendingCoins api.

Fig 6.2 : currency selection

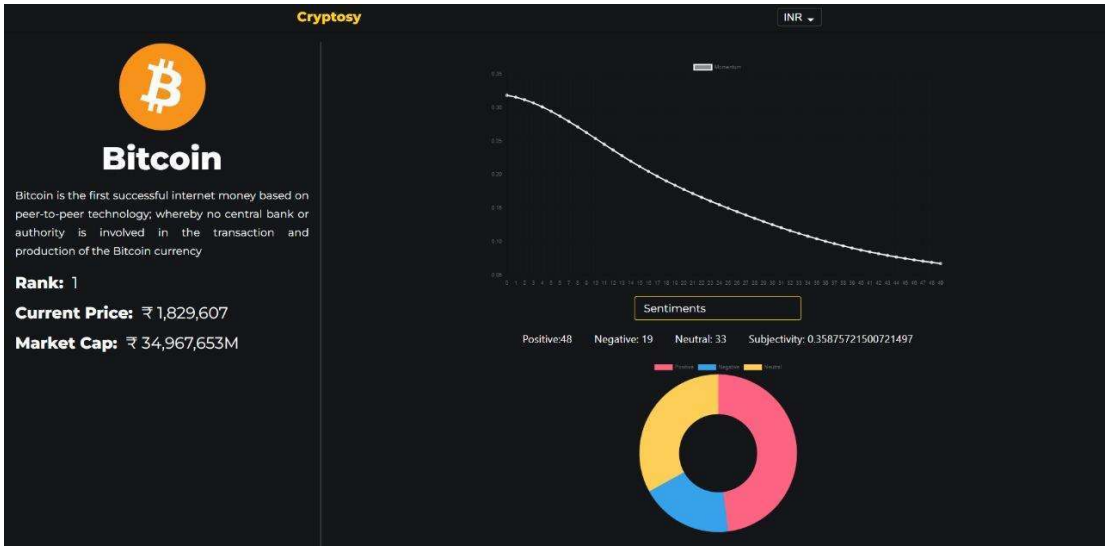


Fig 6.3 : output of technical and sentiment analysis

Once clicked on a particular coin from the coin table, considering Bitcoin for example ,the above web page is displayed (Fig 6.3). On the left , a small coin description, its rank , its current price and the market cap value is presented whereas on the right momentum graph is visible. The sentiment result is displayed on clicking the sentiment button in the form of a donut chart .

It also displays the numeric values for positive , negative and neutral sentiments and the subjectivity of that particular coin.

7. TEST CASES

7.1 BITCOIN

Predicted **Actual**

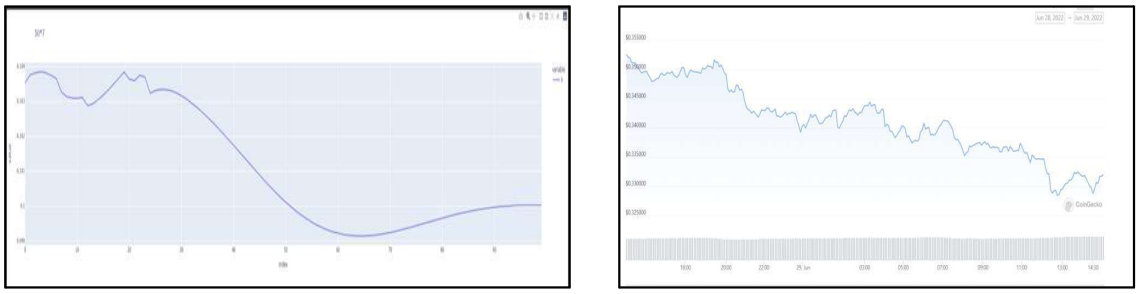


Fig 7.1 : Bitcoin 28 june 22 -30 june 22 (3 days)

Predicted **Actual**

Fig 7.2 : Bitcoin 13 july 22 (24hrs)

Anomaly was detected by Sentiment Analysis:

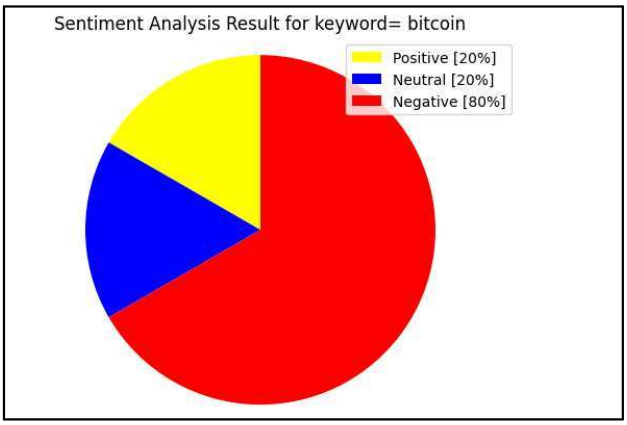


Fig 7.3 : sentiment analysis result for bitcoin(13 jul 22)

Predicted **Actual**

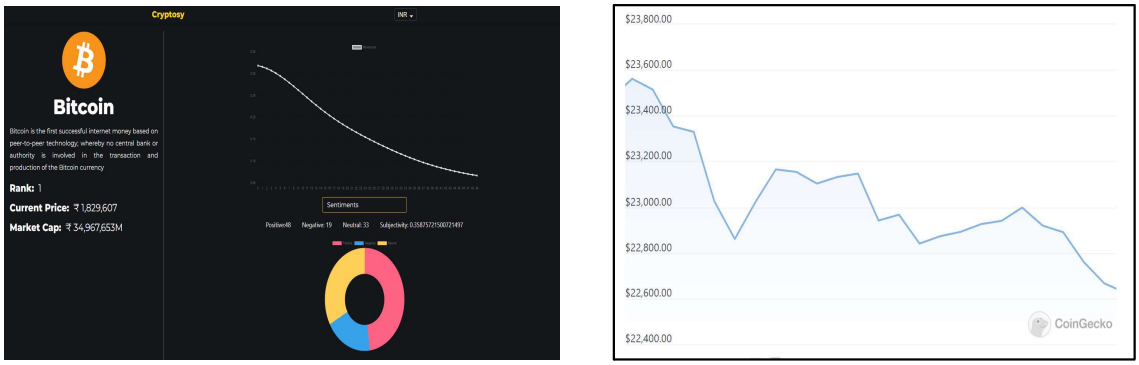


Fig 7.4 : Bitcoin 4 aug 22(24hrs)

7.2 RIPPLE

Predicted **Actual**

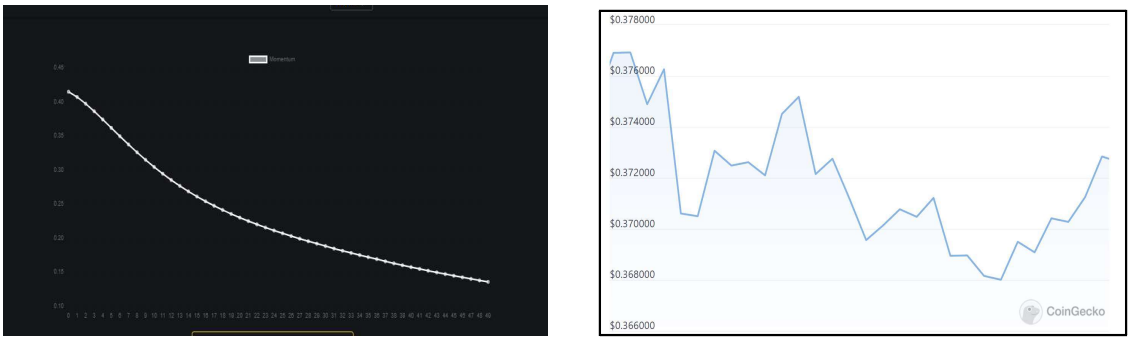


Fig 7.5 : Ripple 4 aug 22(36hrs)

7.3 ETHERIUM

Predicted **Actual**

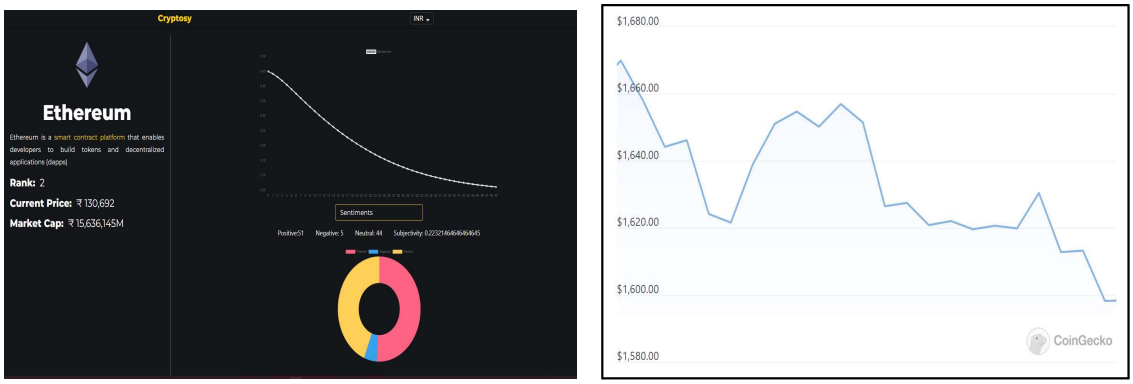


Fig 7.6 : Etherium 4 aug 22(24hrs)

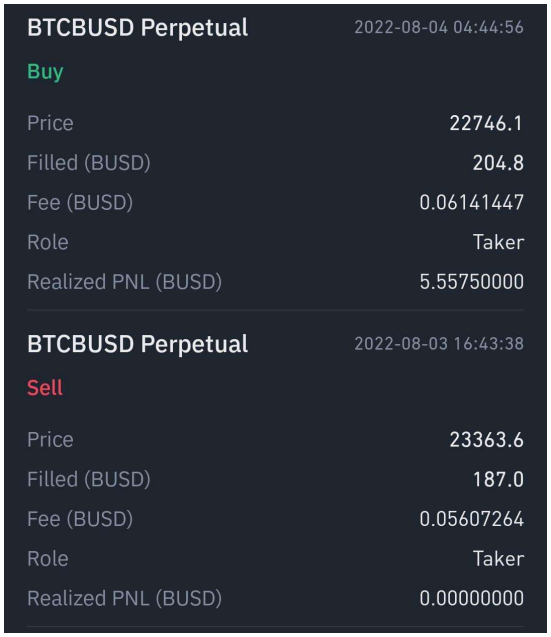


Fig 7.7 : Trade history 1

There was a resistance at 23500 and our model also predicted downward momentum, so we opened a short and closed it at 22700.

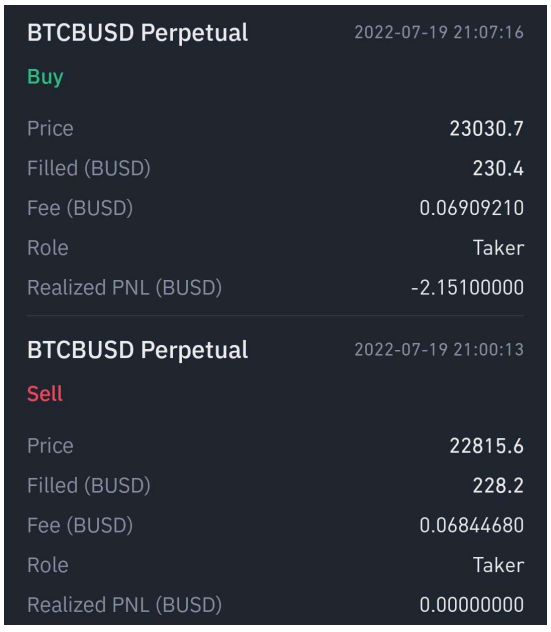


Fig 7.8 : Trade history 2

Later that day (same date as Fig 7.8) we opened another short as there was sudden increase and price action was also in our favour but the sentiments changed suddenly and trade went against us.

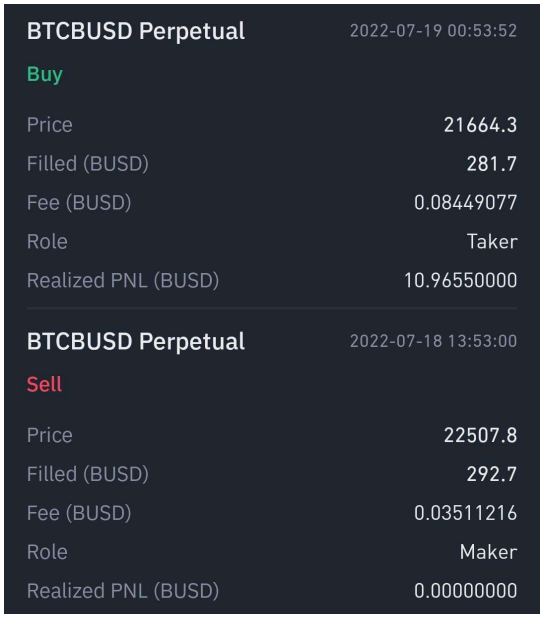


Fig 7.9 : Trade history 3

We opened a short at 22500 as per our model, price-action and sentiments and it worked perfectly and we closed our trade at 21600.

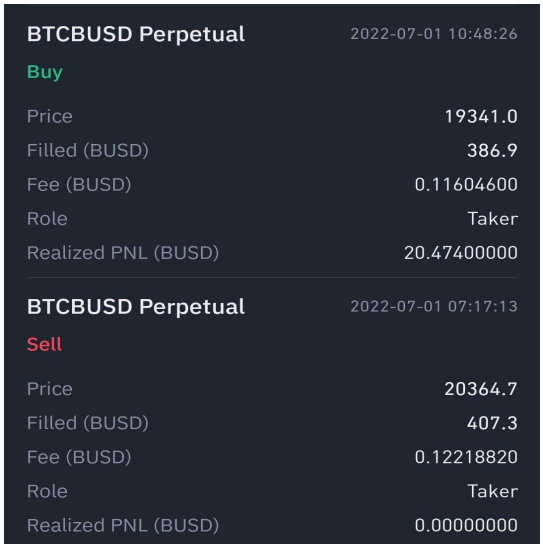


Fig 7.10 : Trade history 4

We opened a short at 20300 as per the sentiments and price action and there was a drop in price next day and trade went in our favour and we closed it at 19400.

CONCLUSION

The decentralisation of cryptocurrencies has sharply weakened central control. Further, wide fluctuations in cryptocurrency price indicate an urgent need for methods to accurately forecast cryptocurrency price. This paper proposes a novel method to predict cryptocurrency price by considering various factors such as market cap, volume, circulating supply, and maximum supply based on deep learning techniques such as the LSTM.

We found various pre-existing solutions and observed that none of them used Technical (Time-series prediction) and Sentimental (Twitter based sentiment) based combinational analysis. Through various research papers and articles we found the LSTM Model as a viable option in time-series prediction since it allows to model both long-term and short-term data and we used Textblob for Sentiment Analysis for which 2500 LiveTweets were analysed.

The results verify the applicability of the proposed approach for the accurate prediction of cryptocurrency price.

FUTURE SCOPE

Future research should extend the proposed approach by considering additional parameters such as the political environment, human relations, and regulations, which vary across countries.

Many of the improvements can go along the line of improving the sentiment classifier and index.

We could also improve the datasets used for training the sentiment classifier

A restriction of our current methodology is to limit the predictions to the direction of the time series, we could include regression models in order to predict the actual values of the time series.

Another improvement would be to consider ensembles of forecasting models as opposed to single models, given the recent popularity and success of this approach in many practical tasks

BIBLIOGRAPHY

[1] Mohil Maheshkumar Patel a , Sudeep Tanwar a , Rajesh Gupta a , Neeraj Kumar *:A Deep Learning-based Cryptocurrency Price Prediction Scheme for Financial Institutions (Journal of Information Security and Applications vol. 55,(August 2020)

[2] Yecheng Yao1* , Jungho Yi2 , Shengjun Zhai3 , Yuwen Lin4 , Taekseung Kim5 Guihongxuan Zhang6 , Leonard Yoonjae Lee7, “Predictive Analysis of Crypto-currency Price Using Deep Learning.”, Science Publishing Corporation Publisher of International Academic Journals,(258-264 pg, 2018)

[3] Jheng-Long Wu 1 , Chen-Chi Su 1 , Liang-Chih Yu 1 and Pei-Chann Chang:Stock Price Prediction using Combinational Features from Sentimental Analysis of Stock News and Technical Analysis of Trading Information(IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, DASC 2011, 12-14 December 2011, Sydney, Australia)

[4] Shangkun DENG, Takashi MITSUBUCHI, Kei SHIODA, Tatsuro SHIMADA, Akito SAKURAI: Combining Technical Analysis with Sentiment Analysis for Stock Price Prediction (2011 Ninth IEEE International Conference on Dependable, Autonomic and Secure Computing)

[5] Xinxin Yao; Hua-Liang Wei , "Short-term stock price forecasting based on similar historical patterns extraction" Published in: 2017 23rd International Conference on Automation and Computing (ICAC) .

[6] Vishal A. Kharde, S.S. Sonawane: Sentiment Analysis of Twitter Data: A Survey of Techniques
(International Journal of Computer Applications (0975 – 8887) Volume 139 – No.11, April 2016)

[7] Deborah S. A. Fernandes; Marcio G. C. Fernandes; Geovany A. Borges; Fabrizzio A.A.M.N. Soares :Decision-Making Simulator for Buying and Selling Stock Market Shares Based on Twitter Indicators and Technical Analysis (Published in: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC))

[8] MARTA ARIAS, ARGIMIRO ARRATIA, RAMON XURIGUERA : Forecasting with Twitter Data(Article in ACM Transactions on Intelligent Systems and Technology · December 2013)