Social Media based Cryptocurrency Sentiment Analysis Prediction and Visualization

Ningbo Zhu
Department of Computing Science
University of Alberta
Edmonton, Canada
ningbo@ualberta.ca

Fei Yang
Department of Computing Science
University of Alberta
Edmonton, Canada
fei5@ualberta.ca

Mingzhi Zhu
Department of Computing Science
University of Alberta
Edmonton, Canada
mingzhi4@ualberta.ca

Abstract—**The cryptocurrency market has developed rapidly in recent years and it has the trend of gradually replacing the traditional currency market. Like stock price prediction, the traditional method uses past price to predict future price. Because the past price is independent of the future price, the result always unsatisfactory. In this paper, we focus on using information related to the cryptocurrency market such as social media information to improve the price prediction model. Our method is machine-learning-based and using high-frequency vocabulary gathered from social media, sentiment analysis of social media information, past price, past hash rate as prediction basis. We use seven days as a time range to train our model and try to predict the eighth-day price increasing rate. We also tested our method on LSTM, bi-LSTM and GRU which are common stock price-prediction models. Compare to the traditional stock price prediction, our cryptocurrency prediction method has better accuracy and timeliness.**

I. Introduction

With the price of cryptocurrencies (e.g. Bitcoin) skyrocketed, some machine learning and deep neural network architectures have been proposed for cryptocurrency analysis and prediction tasks. However, many factors, like demand, policy, hash-rate, and more, cloud affect the price of cryptocurrencies. For many reasons, most cryptocurrency prediction algorithms are not accurate, leading to investors' loss. The relationship between supply and demand generally determines the value of goods, and people are an essential factor in the relationship between supply and demand.

The traditional method normally using the past cryptocurrency price to predict the future price. The problem here is, the past price may lead to an increase in future price but there is no direct relationship between past price and future price. Like the stock, future price only depends on how much people want to buy it. In another word, the cryptocurrency price only depends on how society trusts it.

To solve this problem our team come up with a new direction which is using the past cryptocurrency price to predict the future price increasing rate. Due to cryptocurrency is a currency based on society's trust, our team decided to work on finding a relationship between the cryptocurrency increasing rate and social media-based information.

In the end, a website will visualize those pre-processed data as a graphical, user-friendly chart to the user. In addition, the internet user can visit our bitcoin prediction website to make up their decision.

II. Related Work

In this section, we will introduce some the-state-of-arts related to our topic, cryptocurrency price prediction based on social media sentiment analysis. These researches give us a lot of inspiration and help us get a better result.

Mohapatra et al [1] proposed a real-time cryptocurrency price prediction platform based on Twitter sentiments. They used VADER to analyze the Twitter sentiment and used a machine learning model, the XGBoost model, to train the features including the normalized score calculated by the VADER outputs, the previous closing price, the moving averages of the normalized score, and the moving average of the previous closing price. The web platform is built by Spark framework. The overall root mean square error(RMSE) of their model is \$10 between the actual and the predicted price of Bitcoin. In this paper, we also tried VADER to analyze the twitter, which will be talk about in the following section. Many studies have found that Recurrent Neural Network (RNN) can perform well in time series forecasting. Rodolfo Saldanha [2] respectively implemented two different models of RNNs: LSTM [3] and GRU [4], to predict Amazon's future stock prices using past prices and obtained ideal results. His data preprocessing and training model build provide a good inspiration to build the baseline for our project. Saad et al [5] used a multivariate regression model and LSTM network trained using the price of Bitcoin (BTC) and Ethereum (ETH) from January 2017 to January 2019. Select price, mining difficulty, hash rate, and user count as features to regress and obtain the predicted price. Their model achieved a Mean Absolute Error(MAE) of 0.0162 on BTC and 0.0563 on ETH. We can refer to the features they choose to train our model. Jay et al [6] proposed a stochastic MLP network and a Stochastic LSTM network by adding randomness to the activation funtions at runtime. The stochastic module can capture the reaction of the market to improve the predicted result. The MAE for MLP on bitcoin using stochasticity is 0.04042, on ETH is 0.04123, on Litecoin (LTC) is 0.04201. The MAE for LSTM on bitcoin using stochasticity is 0.05671, on ETH is 0.05145, on LTC is 0.04920. Phaladisailoed et al [7] compare the effect of several machine learning methods used to predict the price of bitcoin. They used the scikit-learn library, Theil-Sen and Huber for Regression model, and LSTM and GRU for deep learning models to create models focusing on bitcoin price from January 2012 to January 2018. Select eight features: Close price, open price, highest price, lowest price, Mean price, total bitcoin trade volume, total USD trade volume, and timestamp within a day. The Mean Squared Error(MSE) of GRU and LSTM are 0.00002 and 0.000431 respectively.

III. Relation to My Other Projects

Cryptocurrency price prediction based on social media sentiment analysis is a novel topic that has gradually attracted our attention recently. We don't have other projects related to this project.

IV. Development Work and Novelty

A. Development Environment

In this project, we divided the project into two main parts, front-end and back-end. For all two main parts, we are using independent software or library.

1) **Front-end Development:** ReactJS [8] is chosen for web development for the front-end development. It is a Modern Javascript library for building single-page applications (SPA) with high performance. With this advanced modern development tool, we manage to create and design modern web applications efficiently. With React component feature, it allows team members to re-use the components during the project development stage. Besides, we used Create React App, a tool for creating single-page React applications without configuration. This command and technique also speed up the process of a React App development.

Once the React Apps is created, several libraries are imported and used in this project. Start with Typescript [9], which adds types for javascript for applying Liskov substitution principle in react component. Sass [10], an advanced, stable, mature, and professional language that can be compatible with all CSS versions. With Sass, AntDesign(AntD) [11], and other custom UI libraries are imported into this project. The AntD, enterprise-grade level components, is embedded into the project, and it supports many effects and different types of icons. All UI libraries makes web components to be animated and interactive.

The following crucial component is Recharts [12], a Redefined chart library built with React and D3. This D3 dynamic chart provides many dynamic and interactive charts. In this project, line chat and two-line chat are chosen for visualizing the data.

Lastly, React-Router-Dom [13] and Moment [14] library are also added to the project. React-Router-Dom is a powerful technique to manage navigation panel paths, and it has been utilized in this project as well. Moment is a JavaScript library for parsing, validating, and manipulating the project date and times. That is all the libraries used in the React App within this project.

MongoDB database is the connection of the front-end to the back-end.

2) **Back-end Development:** MongoDB is a Cloud-hosted service on AWS and it is used for storing all data. In order to retrieve data from cloud, a Node server is required. A get data API is build in the Node Server and retrieve the data. Then Node server pass data to the front-end React App to visualize data.

The next part is data collection, a Python scraping library names Twint [15] will be used to collect the Twitter users' data. User data includes all tweets searched by keywords followed by their date. We also have a filter to collect Tweets that have more than one hundred likes. In this part, we also collect all year bitcoin prices and their hash rate from the Coindesk [16] webpage and the Quandl [17] webpage.

Then we will perform a language analysis library called NLTK [18] to preprocessing the raw data for later use. The NLTK library can convert tweets into formalized keywords and remove punctuations. After that, a sentiment analysis module named vaderSentiment [19] has been used to generate a new feature.

We tried several machine-learning-based models and see which one works best. Currently, LSTM, bi-LSTM, and GRU have been tested. All machine learning models are build-in models in the PyTorch library.

B. Deliverables

1) **Data Scraping:** In this machine-learning-based cryptocurrency price prediction project, the first thing we need to do is building our dataset. To build our dataset, we need to get the raw data from the internet and preprocess it until it fits our requirement for model training. The type of cryptocurrency which we have chosen is Bitcoin, it is well known and has the most number of related researches we can compare the result with. Then we download every day of bitcoin price from 01/01/2014 to 12/31/2021 via Coindesk. And we also download the same timeline's bitcoin hash rate from Quandl. Seven years in total, bitcoin price and the bitcoin hash rate is a one-to-one correspondence. The next step is to choose a social media platform. We select Twitter as our source of social media information not only it is the most famous platform but it also has an open API for use. To get our raw data, we used a python scraping tool, Twint. The Twint can use the API provided by Twitter to scraping Twitter user's information, such as Tweet, the Tweet time, user's location etc. Here in this project, we only need Tweet and the Tweet time with keywords filter 'bitcoin' and language filter 'English'. We also uploaded our raw data to MongoDB for later use, more details about MongoDB have been introduced in a later section.

2) **NLP Data Formalize:** After we got all the raw data, the next challenge we are facing is how to use them. Tweet data is full of garbled words, emojis, punctuations and mixed with other languages. These chaotic sentences machines can hardly understand, so we decide to clean them up. We used an NLP-based model NLTK. By using an English dictionary file, input a sentence, the NLTK model can remove the garbled words, punctuations, and other language words. It can also transform the words in the sentences back to their prototype. Emojis are not cleaned, because we consider emojis sometimes represent popular culture, they may be used in future work. The NLTK model can return a cleaned sentence or a cleaned word list of a sentence.

3) **High Frequency Vocabulary Counting and Chosen:** Before we go further, we think about a new feature which is high-frequency vocabularies. The high-frequency vocabularies represent the popular topics in social media, the popularity of some topics may influence the Bitcoin price increasing rate. We used one of the outputs of the NLTP, a cleaned word list to count the frequency of the words. Sort the counts and get the most frequent words up to 100. After we manually select 11 high-frequency vocabularies which may have a relationship with the bitcoin price increasing rate and considering the machine-learning-based model, we make a one-hot vector as a feature for model training. The one-hot vector has a shape of (11,1), each tweet can generate one vector. And each element in the vector represents the appearance of a high-frequency vocabulary. If there is no high-frequency vocabulary in the word list, the vector should like [0,0,0,0,0,0,0,0,0,0,0]. If the second high-frequency vocabulary we have chosen appears in the tweet and the fourth high-frequency vocabulary we have chosen appears in the tweet twice, the one-hot vector should like [0,1,0,2,0,0,0,0,0,0,0]. Due to there is more than one tweet per day, the one-hot vector is summed up by each day. As a feature, the summed up one-hot vector of one day could be like [1023,0,3,579,0,0,0,0,0,1,1].

4) **Sentiment Analysis:** High positive or negative sentiment intensity of the social media environment may also lead to a big cryptocurrency price changing. After we using one of our NLTK outputs, we try to do sentiment analysis to our tweet sentence. We using the other output of our NLTK model and put these cleaned sentences into a sentiment analysis model called vaderSentiment. The vaderSentiment model can output a float number that has a range (-1,1), represents the sentiment intensity of the input sentence. +1 is the strong positive sentiment, -1 is the strong negative sentiment, 0 is neutral sentiment. We use sentiment analysis as a new feature for model training.

5) **Time Range and More Preprocessing:** We also do more data preprocessing and generated more features to train our model. The most important one is changing our target from the bitcoin price to the bitcoin price increasing rate. Due to the past price is independent of the future price, using the future price as the target, the result always unsatisfactory. But the past price may lead to a future price increasing or decreasing. So we convert the target from predicting the price to predicting the price increasing rate. Therefore the problem is transforming to use a machine-learning-based model to find out the relationship between the past price and the future price increasing rate. Then the second point is the time range. It is impossible to put all history data at once into the machine learning model to train and predict the price increasing rate for the next day. We have to come up with a reasonable training strategy. Due to people have different sentiments on each day of a week, divide the data by seven days seems a perfect solution. We combined every seven days of data by shifting one day at once. So we don't lose the number of training data which is already in short supply. As it should be, the same process has been taken on all other features. Then the next preprocessing is normalization. The changing of the bitcoin price in the past seven years is dramatic, from the lowest around 150$ per bitcoin to the highest 63000$ per bitcoin. Without price normalization, the low price input will have low weights after training, but the fact is it should have the same weights as the high price input. Therefore, a price normalization has been applied to the seven days combination of the input data. And hash rate also has the same problem(see Fig.2). The normalization of the hash rate is the same as the price. We noticed that the former part of the dataset (2014-2017)(see Fig.1), which should be the training dataset, is too similar for the first bunch of input so the model cannot learn enough data diversity about price increasing rate. And after 2017, The price of Bitcoin has been affected by international events such as the inancial crisis in 2018 and the COVID-19 in 2020. Those parts of the data was dropped in the validation part and test part. The model cannot validate and test well. Therefore, we shuffle the normalized sliced data, so the well-trained model could have enough robustness.

6) **LSTM, bi-LSTM, GRU:** In this paper, we tested three common stock prediction machine-learning-based models, named LSTM, bi-LSTM, and GRU. And we also test several different feature combinations to predict the increasing rate of bitcoin price. LSTM, bi-LSTM, and GRU are all modified RNNs. RNN is a class of artificial neural networks that can solve tasks such as prediction problems, machine translation, and sentiment classification, by recurrently make use of sequential information. However, RNN has short-term memory problems and cannot handle very long input sequence training. Because with the advancement of the long sequence training process, the vanishing gradient problem will appear. The gradient is used to update the weights of the neural network. If the gradient is too small, the weight of each update will have less impact on the next training, and the model will stop learning or learn very slowly. LSTM is designed to fix this problem, it can save and restore long-term information without using too much memory. Comparing LSTM and GRU [20], LSTM has three gates, while GRU has only two.(See Fig.3) So GRU has fewer parameters and it is easier to converge. Bi-LSTM [21] has one more set of LSTM layers than LSTM model. It has a forward LSTM and a backward LSTM. BiLSTMs effectively increase the amount of information extracted and improve the content available to the network. In the training and validating processing, we use the Root-Mean-Square Error (RMSE, Equation.1) as a loss function and do the weight backpropagation in each epoch. Then we plot the training and validating loss to see whether the model converges well.

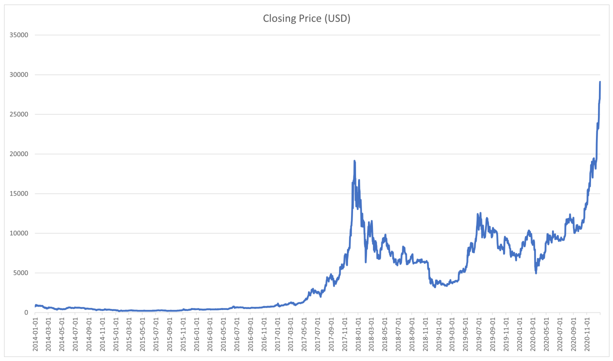

Fig. 1. BTC Price during Janurary 2014 to December 2020 [16]

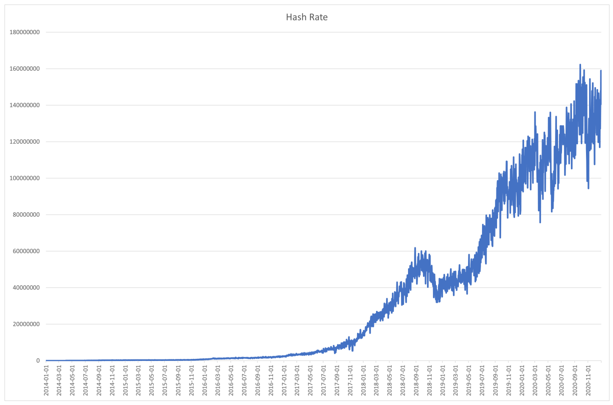

Fig. 2. Hash rate during Janurary 2014 to December 2020 [17]

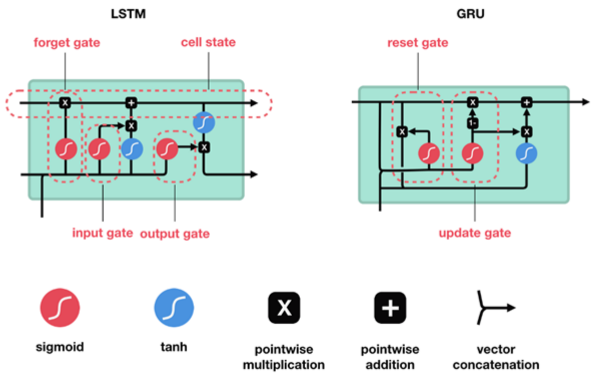

Fig. 3. Model structure of LSTM and GRU [20]

$$RMSE = \sqrt{\sum_{n=1}^{n}(y_i - y_{i_p})^2} \tag{1}$$

*7) **Price Prediction:*** After obtaining all well-trained models, we test all price data from January 2014 to December 2020 to get the price increasing rate during this time range.The first 7 days of the year 2014 have no predict output, they need the last 6 days of data of the year 2013 as an input. Then Calculate the predicted price using Equation.2. The predicted data is convenient for our webapp visualization and then we upload all predicted dataset to MongoDB.

$$\begin{aligned} &PredictedPrice \\ &\qquad = PreciousDayPrice \times \\ &\qquad (1 + PredictedPriceIncreasingRate) \end{aligned} \tag{2}$$

*8) **MongoDB:*** MongoDB [22] plays a key role in the project, and it is the bridge that connect all components. In this project, Twint scraps all data from the social community - twitter and store the data into MongoDB. LSTM, Bi-LSTM, GRU use those data to predict prices. Once predicted price data is generated and it will be re-import back into MongoDB. For the front-end, the NodeJS server fetches processed data and translates data into JASON format. React app visualized the data into a meaningful graph. To work on the MongoDB, proper permission is required to set up for each team member. Although each team member only works on his part, appropriate schema and database rules are configured for use. With the project going forward, the database schema is required to be changed. MongoDB allows multi-user works on the database and changes the database schema simultaneously. The benefit of working on MongoDB is obvious, and any team member can change the collection anytime he/she wants. The drawback is incorrect data type may cause more bugs. With team members working closely, we have an agreement and adequately configure the final database schema to meet all team members' requirements.

*9) **NodeJS Server:*** To retrieve real-time data from MongoDB, a server-side application - NodeJS [23], a JavaScript runtime built on Chrome's V8, is required to be established between the Database and web application. NodeJS is designed to build scalable network applications, which is industry-level back-end server. With NodeJS, network requests can be handled asynchronously, and it won't block any input or output requests. The benefit of NodeJS is innumerable. For example, when no request needs to be fired, NodeJS will turn itself into sleep mode and lowering the server's workload. Besides, NodeJS uses a microservices ecosystem solution that means application components get true separation among each team member's working part. NodeJS also support multi-environment development and allows members work on different environment separately. Most importantly, NodeJS following good generic coding practices, such as Don't repeat yourself(DRY), and separation of concern principles. For those reasons and more, NodeJS is selected as the back-end server to handling requests between MongoDB and React App. Although the NodeJS server looks perfect choice for our project, we encounter many issues during the development stage. The first issue is mongoose configuration; MongoDB does not directly retrieve data from API. For security concern, Mongoose configuration must be set up first using its own method which is mongoose.connect('mongodb://localhost:4000/myapp', user: true);. Also, proper user privilege is required to set up in MongoDB as well. This is a prerequisite to connect MongoDB. The second problem is data type mismatch. Data is storing on the cloud-based Database, and NodeJS has no idea what its going to receive. Therefore, the data type must be fully matched between MongoDB and NodeJS server; otherwise, NodeJS cannot export the data correctly. Once these two issues are resolved, the next one is cross-origin resource sharing (CORS). CORS is an HTTP-header-based mechanism that allows the NodeJS server to send out a request with a header such as domain, scheme, and port. Through header information, the NodeJS's API can send out a request to retrieve data from MongoDB. As A server-side application, NodeJS has its own encoding/decoding method. When NodeJS are sending the data, the data will be encoded into binary format. The reason is Binary data generally use less bandwidth compare with original data. To convert the data to proper format, a library names express is imported to the NodeJS server. It is used for encoding raw data into binary, and then React App will use Axios to convert the data into JSON format. If data contains a status code of 200, its data will be matched to the pre-defined data type. That is how the NodeJS server works.

10) **React App:** Once the NodeJS server is established and corresponded API is created properly, a React App is ready to be built to handing web requests between the end-user and NodeJS server. This task sounds simple but challenging. The first challenge is encoding/decoding, and the React App must decode the buffer data receive from the NodeJS server. To solve this problem, an library names Axois is installed on React. Axois is HTTP client for the React and NodeJS. It enables React App to send asynchronous HTTP requests to REST endpoints and perform CURD operations. The next challenge is to generate a meaningful graphic. The user only interacts with what they can see. To achieve this goal, we imported several libraries and used them in the React App. The React-Router-Dom, Sass, and AntD are applied to create the base design, such as navigation bar, icon, header, footer, etc. Then the Recharts is used to convert the data into proper charts. A sample demo page is shown as the following:

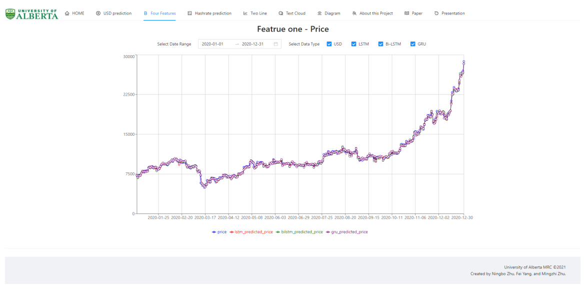

Fig. 4. Price Prediction with Recharts

In Fig.4, the line chart page provides a couple of options. Users can either choose the date range to view the data or select three different models to see the predicted price. With those functions support, users interact with the data in visualized manner. Besides, we provide other tabs for users to see more data. By clicking on the navigation menu, users can navigate to different pages to see other predicted price results. On this website, there are four prediction pages with additional features input. By compare those charts, users can make a wise choice on their investment.

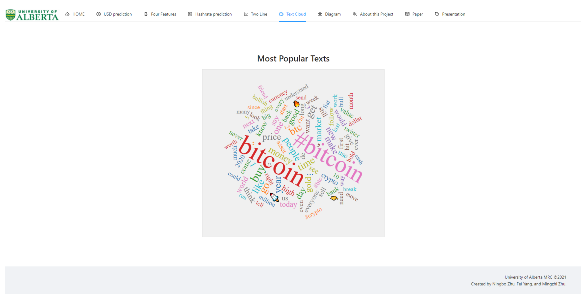

Fig. 5. Words count with Wold-cloud

Since the tweet is used as an input feature in the prediction model, users may be curious what kind of keywords were mentioned and how many times they were counted. Consequently, React-Wordcloud is created for demonstration purposes. In Fig.5, Bitcoin was mentioned more than 24,000 times. Other words like buy, money, currency, dollar also were mentioned many times. From this React-Wordcloud tab, we can tell many people discuss cryptocurrency on Twitter.

C. Contribution by All Team Members

Contributions by each team member are shown in Table.I.

MemberName:	Contribution:
MingzhiZhu	User-Interface Design Web Development Web Libraries utilization and optimization Node Server Development Node Server api Database schema design Data posted process Data Visualization
NingboZhu	Scraping Raw Tweet Data from Twitter Design Data Structure for Model Training NLP Data Preprocessing Sentiment Analyze Data Preprocessing Build Up High-frequency Vocabulary One-hot feature Modify Training Data Structure Database Build Up
FeiYang	Scraping Raw Bitcoin Price Data Scraping Raw Bitcoin Hash Rate Data Machine Learning Baseline Build Up Modify Machine Learning Model Training Data Structure Preprocessing LSTM Model Train and Test bi-LSTM Model Train and Test GRU Model Train and Test

TABLE I
CONTRIBUTION BY ALL TEAM MEMBERS

D. *Novelty*

Compare to other related works, our method firstly uses Bitcoin Hashrate as a model training feature. Bitcoin Hashrate is the speed for a certain CPU to proof a bitcoin trade action, it can also be understood as how difficult to mine bitcoin. When Hashrate is high, it means more trade is happening and more people are mining the bitcoin. It is a signal of people keep buying or selling, which always lead to bitcoin price increasing or decreasing.

Our method is also the first one to use high-frequency vocabulary one-hot vector as a model training feature. High-frequency vocabulary from social media reflects if bitcoin is the hot topic. If more people talk about bitcoin then the social trust of bitcoin is increasing. It will lead to a trend of price increasing in the future.

We also have a fresh new label type for model training which is price increasing rate. Compared to traditional price prediction which is using price as the label, the price increasing rate is more reasonable. It has more relation to the past price and makes the label drops into a reasonable range instead of a large price range.

One more novelty element is prediction visualization. In our well-designed web app, users can visualize the prediction result from different models with different features as input and compare it with the real price. It is easy for users to conclude that the relationship between different features to the price and which model is the best.

E. *Numerical Results*

Table.II and Table.III show the numerical result of the three models we used.

In Table.II, “Sign Correct” represents the amount of result which correctly predicts the increasing rate of price is positive or negative. If the increasing rate is greater than 0, then the price is up, else, the price is down. “Error ¡ 0.05” means that the error between the predicted increasing rate and the groundtruth increasing rate is less than 5%. Therefore, in this paper, if a predicted result has the same sign as the groundtruth and their difference is less than 5%, it is deemed predicted completely correct. If a predicted result only satisfies ”Sign Correct” or ”Error ¡ 0.05”, it is also acceptable. It is obvious that GRU has the best prediction performance when using price as the input feature. It has 72.35% results accuracy satisfied “Sign Correct” and “Error ¡ 0.05”. From the table, we can find out that the result of using hash rate as the only input feature is unsatisfactory, the accuracy of ”Sign CorrectError ¡ 0.05” is only about 45%. So, we can conclude that the relationship between the Hash rate and the increasing rate of price is not very close. Meanwhile, we can also find out that the result of using all those four features at same time for training is unsatisfactory too. We suspect maybe because there are too many feature data, the weight of price is diluted by other features, which leads the model hard to learn useful information.

Table.III shows the evaluation of the train, validation, and test prediction results of three models. The RMSE of the predicted price increasing rate is very small but the RMSE of the predicted price is still large. We can find that the best-result model, GRU, doesn’t have the lowest loss. It might because the error of the wrong estimates is too large, resulting in a high final loss.

F. *Academic Contribution*

There is always a misunderstanding for most of the researcher who tries to predict the bitcoin price, which is using the past price to predict the future price. People need to think more about that, past price is independent of the future price. Past price may lead to an increase in future price but there is no direct relationship between past price and future price. Like the stock, future price only depends on how much people want to buy it. In another word, the bitcoin price only depends on how society trusts it. In this paper we proposed a better way to predict the bitcoin price. We convert the target from predicting the price directly to predicting the price increasing rate.

We also prove that society trust information such as high-frequency-vocabulary, social media sentiment intensity, and hash rate have low relation with the cryptocurrency price increasing rate.

V. Project Status

A. *Front-end Development*

The React App has outplay its role in visualization. web pages are created and fully meet end-user. In additional, it can be used for the final presentation demonstration. In the data visualization session, it is difficult to generate line chart to compare hash-rate and USD. This is caused by huge data difference between two values. To better visualize data into graphic, Ming use two-line chat to visualize data and the result is obvious.

B. *Back-end Development*

1) Tweet Data Scraping: We scraped Tweet datas using Twint and analysis them using NLTK and vaderSentiment. The output should include a Sentiment analysis number representing the sentiment intensity, and a keywords one-hot vector. The keywords one-hot vector includes 11 most frequent keywords. We meet a missing data problem. When bitcoin is not as popular as it is now, there are few tweets mentioned bitcoin. So the tweet dataset is not continuous. Our solution is that for those dates that have no tweet, we add a record including a one-hot with all 0 sized 11 and a natural sentiment intensity, which is 0. This record is totally natural and will not influence the result.

2) Bitcoin Price and Blockchain Hash Rate Collecting and Preprocessing: The bitcoin price is different on the different websites. Due to there are many third-party bitcoin trade platforms, each platform collect the data on the different time which leads to a different trade price every day. We can just choose one of them to do our research. The another challenge we meet in this part is that the result is always one same number and it approximately equals to 0. We first normalized

Features	Processing	LSTM			bi-LSTM			GRU		
		Sign Correct	Error <0.05	Sign Correct & Error <0.05	Sign Correct	Error <0.05	Sign Correct & Error <0.05	Sign Correct	Error <0.05	Sign Correct & Error <0.05
Price	**Train**	0.7830	0.9675	0.7560	0.8197	0.9828	0.8056	0.7793	0.9510	0.7382
	Validation	0.7083	0.8627	0.6127	0.7353	0.8603	0.6446	0.7819	0.9118	0.7132
	Test	0.7725	0.9235	0.7098	0.7706	0.9235	0.7000	**0.7863**	**0.9255**	**0.7235**
Hashrate	**Train**	0.5383	0.8676	0.4746	0.5303	0.8719	0.4641	0.5267	0.8657	0.4629
	Validation	0.4975	0.8284	0.4093	0.4779	0.8284	0.3995	0.4951	0.8284	0.4093
	Test	0.5412	0.8549	0.4588	**0.5510**	**0.8549**	**0.4725**	0.5412	0.8549	0.4588
Price + Hashrate	**Train**	0.7474	0.9761	0.7284	0.7836	0.9859	0.7713	0.7284	0.9626	0.7032
	Validation	0.7034	0.8725	0.6029	0.6838	0.8431	0.5760	0.6961	0.8750	0.6078
	Test	0.7196	0.9039	0.6431	0.7235	0.8941	0.6431	**0.7392**	**0.9098**	**0.6706**
Price+Hashrate+ Tweet Sentiment Analysis+ Tweet Keywords	**Train**	0.5310	0.8688	0.4697	0.5395	0.8700	0.4776	0.5671	0.8749	0.4954
	Validation	0.5049	0.8309	0.4314	0.4632	0.8309	0.3922	0.5294	0.8260	0.4485
	Test	0.5353	0.8549	0.4529	0.5235	0.8608	0.4529	**0.5373**	**0.8510**	**0.4706**

TABLE II
PREDICTION RESULTS OF LSTM, BI-LSTM, AND GRU

Features	**RMSE**	**LSTM**			**bi-LSTM**			**GRU**		
		Price	**Scaled Price**	**Price Increasing Rate**	**Price**	**Scaled Price**	**Price Increasing Rate**	**Price**	**Scaled Price**	**Price Increasing Rate**
Price	Train RMSE	306.92	0.31	0.02	315.91	0.31	0.02	298.14	0.31	0.02
	Validation RMSE	355.56	0.34	0.04	360.56	0.34	0.04	337.44	0.34	0.03
	Test RMSE	479.25	0.33	0.03	472.48	0.33	0.03	469.90	0.33	0.03
Hashrate	Train RMSE	264.77	0.32	0.04	264.52	0.32	0.04	265.69	0.32	0.04
	Validation RMSE	279.75	0.35	0.04	282.78	0.35	0.04	279.28	0.35	0.04
	Test RMSE	354.78	0.34	0.04	354.50	0.34	0.04	354.66	0.34	0.04
Price + Hashrate	Train RMSE	302.99	0.31	0.02	299.11	0.31	0.02	303.33	0.31	0.02
	Validation RMSE	340.97	0.34	0.04	354.94	0.34	0.04	337.89	0.34	0.04
	Test RMSE	472.56	0.34	0.03	509.86	0.33	0.03	482.64	0.34	0.03
Price+ Hashrate+ Tweet Sentiment Analysis+ Tweet Keywords	Train RMSE	264.79	0.32	0.04	264.60	0.32	0.04	267.53	0.32	0.04
	Validation RMSE	282.4	0.35	0.04	281.15	0.35	0.04	290.72	0.35	0.04
	Test RMSE	357.17	0.34	0.04	357.72	0.34	0.04	359.60	0.34	0.04

TABLE III
RMSE OF PREDICTION RESULT OF LSTM, BI-LSTM, GRU.

the whole dataset together, which means the entire dataset share one same scaler. Following, we sliced the data into 7-day groups. The problem here is that the normalization we did before is not enough "normal" for each day's price. The entire dataset sharing the same scaler makes the continued small data in early years lose their weights and changing trend. Our solution is that first slice the data into 7-day groups then normalize them independently.

3) Machine learning Model Training: By solving all the challenges we have met and keep fine-tuning models, we can finally predict a reasonable bitcoin price increasing rate and figure out the relationship between them.

VI. CONCLUDING REMARKS

A. Future-Work

For this project, we managed to predict the future bitcoin price using LSTM, Bi-LSTM, and GRU and create a web application to visualize data into a graphic. However, there still many improvements can be made to the project. From the Table.I, we can see more features input cause the "Sign Correct&Error <0.05" decrease. Currently, expect the price only three features are input into the models. Forgoing forward, we can add more features and different combinations to find out a better result. Also, the current models have used only one kind of social media Twitter to predict the price. Other social media, like Facebook, LinkedIn can be added and applied to our project. There would have a side effect. However, the result is unknown and more experiments are needed. Besides more social media, we can also try our models on different cryptocurrencies, such as ETC, LTC, ADA, and more. This will generate more possibilities and results. Based on the results, we can optimize our models' structure and fine-tune hyper-parameter within the models. As for the other directions, there are changing the structure of the training data and the number of days of testing. Due to the current continuous rise of bitcoin, most of our training data shows an upward trend, which may affect the final prediction results, therefore, balancing the training data may bring different results. Currently, our model is using seven days to predict the price on the eighth day. If we change this structure and add more data, for example, using 30 days to predict the price on the 31st day, this may again lead to different results. However, more experiments are needed to confirm our suspicions, and these will be included in future plans. Apart from the model section, our website also has a lot of areas for improvement. As a website, it deserves to have more simple and clear information as well as an intuitive and easy-to-use interface. Simultaneously, more explanations and details could be added to the site. This allows users to better understand each model's results and conclusions, providing them with more intuitive and reliable information. In addition, a feedback page could be added to allow users to give feedback on the site's shortcomings. This can be used to improve the presentation of the site. All of this needs to be done before the site can be officially put on the web. All of this will be included in our future plans.

B. Conclusion

From the project, we can conclude that due to the past price is independent of the future price, using the future price as the target, the result always unsatisfactory. But convert the target from price to the price increasing rate has unexpected accuracy improvement. After we using the price increasing rate as our target, the sign correct prediction accuracy grows from around 50% to 78.63% and the completely correct prediction accuracy grows from around 40% to 72.35%. Due to the unexpected high accuracy, we add more features immediately and hope to get a better result. The result is a disappointment, new features added dilute the weights of the input price and gives a worse result. Social media information and hash rate may reflect how society trusts bitcoin and have some relationship with bitcoin price but they are not useful for cryptocurrency price increasing rate prediction. Nevertheless, compared to the traditional stock price prediction, our cryptocurrency prediction method has better accuracy and timeliness. Our web app is convenient for users to analyze the relationship between social media and the cryptocurrency market and have a reasonable prediction of the future price.

References

[1] Shubhankar Mohapatra, Nauman Ahmed, and Paulo Alencar. Kryptooracle: A real-time cryptocurrency price prediction platform using twitter sentiments. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 5544–5551. IEEE, 2019.

[2] Rodolfo Saldanha. Stock price prediction with pytorch - lstm and gru to predict amazon's stock prices. https://medium.com/swlh/stock-price-prediction-with-pytorch-37f52ae84632, 2020.

[3] Jürgen Schmidhuber and Sepp Hochreiter. Long short-term memory. *Neural Comput*, 9(8):1735–1780, 1997.

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[5] Muhammad Saad, Jinchun Choi, DaeHun Nyang, Joongheon Kim, and Aziz Mohaisen. Toward characterizing blockchain-based cryptocurrencies for highly accurate predictions. *IEEE Systems Journal*, 14(1):321–332, 2019.

[6] Patel Jay, Vasu Kalariya, Pushpendra Parmar, Sudeep Tanwar, Neeraj Kumar, and Mamoun Alazab. Stochastic neural networks for cryptocurrency price prediction. *IEEE Access*, 8:82804–82818, 2020.

[7] Thearasak Phaladisailoed and Thanisa Numnonda. Machine learning models comparison for bitcoin price prediction. In *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 506–511. IEEE, 2018.

[8] React. React : A javascript library for building user interfaces. https://reactjs.org/, 2021.

[9] TypeScript. Typescript : Typed javascript at any scale. https://www.typescriptlang.org/, 2021.

[10] Sass. Sass : Css with superpowers. https://sass-lang.com/, 2021.

[11] AntD. Antd: Enterprise-level product design system to create an efficient and pleasant working experience. https://ant.design/index-cn, 2021.

[12] Recharts. Recharts: A composable charting library built on react components. https://recharts.org/en-US/, 2021.

[13] REACT ROUTE. React route: Learn once, route anywhere. https://reactrouter.com/, 2021.

[14] momentjs. momentjs: A javascript date library for parsing, validating, manipulating, and formatting dates. https://momentjs.com/, 2021.

[15] Twint. Twint : An advanced twitter scraping osint tool written in python. https://github.com/twintproject/twint, 2021.

[16] CoinDesk. Coindesk: Bitcoin, ethereum, crypto news and price data. https://www.coindesk.com/price/bitcoin, 2021.

[17] Quandl. Quandl: Financial, economic and alternative data. https://www.quandl.com/data/BCHAIN/HRATE-Bitcoin-Hash-Rate, 2021.

[18] nltk. Nltk: Natural language toolkit. https://www.nltk.org/, 2021.

[19] vaderSentiment. vadersentiment : Vader sentiment analysis. https://github.com/cjhutto/vaderSentiment, 2021.

[20] Michael Phi. Illustrated guide to lstm's and gru's: A step by step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21, 2018.

[21] Nils Reimers and Iryna Gurevych. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*, 2017.

[22] MongoDB. Mongodb: Cloud-hosted mongodb service on aws. https://www.mongodb.com/2, 2021.

[23] Nodejs. Nodejs: Node.js® is a javascript runtime built on chrome's v8 javascript engine. https://nodejs.org/en/, 2021.