“Mastering cryptocurrency price movements with Machine Learning”

Problem definition

With more than 18,000 cryptocurrencies available in the market and 300 million users globally, information about these new financial trends is essential. Taking advantage of the movements in the future of the price of the coin you are investing in will make you completely rich or completely poor in a second.

First introduced in 2008 (Nakamoto), Bitcoin has emerged as the main crypto in the market backed by a strong blockchain technology securing peer-to-peer transactions by cryptography. Nowadays the daily movements of BTC, as it is commonly abbreviated, are impressive. According to Statista (2022), by the beginning of this year its daily trade volume was 31.11 billion US dollars with more than 270,000 user transactions. And while the exact number of owners is unknown, it is believed to be between 80 and 106 million (Howarth, 2022). In particular, Bitcoin has been increasingly regarded as an investment asset and because of its highly volatile nature, our data product comes to the aid to give accurate predictions to guide investment decisions.

In this project, we analyze the predictability of this market with enough flexibility to easily adapt our data architecture to more currencies. A ton of studies in the areas of finance, economics and statistics have put different machine learning models to compete to predict the price of Bitcoin, reaching the conclusion that the majority of this tools are good and useful enough to predict price changes in the short and long term. A particular example is that of Jaquart, Dann and Weinhart (2021), that showed that neural networks and gradient boosting classifiers have strong binary predictions but in a future span from 1 to 60 minutes.

With this in mind, if machine learning is good at predicting the price of bitcoin, why not improve it and build a flexible data architecture that could add more coins and more users? In this project we aim to solve this question by predicting with a deep neural network if the price of BTC goes up or goes down by a given probability and a specific number of days ahead set by the user.

System design

The key components of our data product architecture are displayed in Figure 1 to illustrate the interplay between system components. In a nutshell, Google Cloud Platform services guided the process and Python served as the main programming language.

Each of the stages of our data product architecture are described as follows:

- First, we collected the currency price data from the CoinGecko API which allowed us to get new information. We decided to use this service because it was free and reliable. While in the script we did not need a key, to prevent getting locked by the API, we set a limit of 35 calls per minute.

- We stored our data in Buckets inside Google Cloud Storage. We performed an Extract, Transform, Load (ETL) job because we found numerous useless features from the API extraction such as the prices of the currencies in terms of another. After the storage, we executed some illustrative queries in BigQuery to get a glimpse of the data.

- For the development of all the code including writing functions and training models we used Vertex AI notebooks. In order to make the Feature Engineering and a basic Exploratory Data Analysis (EDA), we got the data using BigQuery. We developed both to see the importance of key variables and main characteristics such as its distribution and their usefulness for the problem. For the case of the Feature Engineering process we developed a new Python class that specified “time windows” setting the number of days backwards from the observation, the number of days forward to see the performance, the number of days grouped to get metrics (such as mean and standard deviation for price and market cap), and the minimum percentage of increment to expect in the price for the given window.

- With the selection of our model we constructed another Python class that helped with the process of splitting the data and scaling the features and serves as an input of the training process. For training purposes we constructed and tuned a neural network specifying certain values such as the initial day (January 1, 2021), the number of days back to start the window (60), the number of days forward to make a prediction(7), and the percentage of price increase in the window(0.1) while using only relevant features selected after a variable clustering procedure.

- As a way to ensure the reproducibility of our work, we packaged the model in a Docker container. We used the GCP tools to create the container within the same cloud environment and deposit it in the Container Registry.

- To simplify the job training we created a function that gets the information from the Docker container and uses the JobServiceClient. To save the best model in the cloud we create a function that creates a model through Vertex AI and deploys it into an endpoint from the same service.

- We created a function called prediction.py that requires: a) the number of days back from the selected day, b) the number of days ahead “X” to make the prediction, c) the number of days grouped to get metrics in the observation window, d) the minimum percentage “P” increase to make the prediction, e) a list of selected features for the model, f) the bucket name where the model is stored, g) the endpoint name, and h) the day for the prediction (t0).

- Finally, with all these decisions, we then construct our orchestration in Airflow in order to automate all the (re)training process. Our DAG's make the same process described above with a daily execution at 06:00.

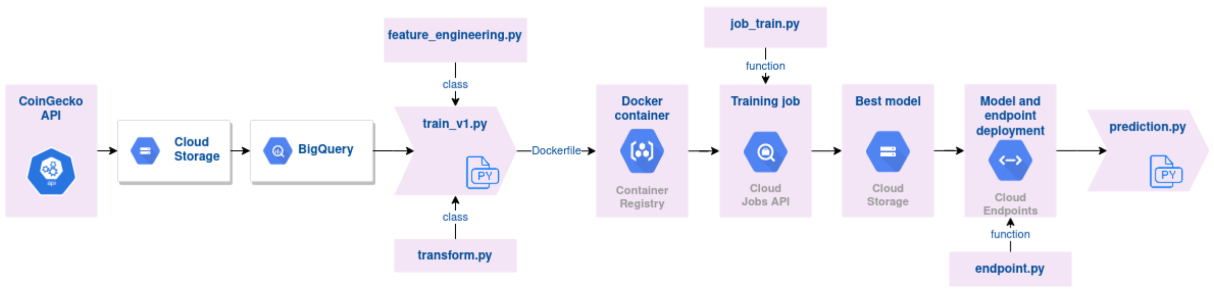


Figure 1.- System Architecture on Google Cloud Platform

Data source, ingestion, and feature engineering

The Data we use for our model comes from the CoinGecko cryptocurrency API that contains live prices, trading volume, exchange volume, trading pairs, historical data, contract address data, crypto categories, crypto derivatives, and images. We choose this API because it is free, reliable, and comprehensive.

From the API documentation, we obtained examples for our API Request Payloads. Because the API is free, our script does not need a key but it has a rate limit of 50 calls/minute. In practice we made a rate limit of 35 calls/minute using Python's time.sleep() function so the API could not block us.

We extract historical data from the API making a request that lists all coins with id, name, and symbol. We performed a transformation intermediate process (that is why we identify our process as an ETL job) because we found numerous useless features from the API extraction such as the prices of the currencies in terms of another. After that, we write the data into Google Cloud Storage. In the first API Request Payload, we get the top10 market cap cryptocurrencies from 01-01-2020 to 28-02-22

For the case of the Feature Engineering process we developed a new Python class that specified "time windows" setting the number of days backwards from the initial day of training (t0), the number of days forward to see the performance, the number of days grouped to get metrics (such as mean and standard deviation for price and market cap), and the minimum percentage of increment to expect in the price for the given window. This feature engineering was required because of the way we posed the problem trying to capture broader trends grouping a specified number of days.

Machine Development

A special case of machine learning algorithms is deep neural networks. Deep neural networks are focused on emulating the learning approach that humans use to gain a certain type of knowledge. These neural networks learn hierarchical structures and levels of representation and abstraction to understand data patterns that come from various types of sources such as images, videos, sound, or text. The main idea behind an artificial neuron is quite simple. Has one or more inputs and one output. Depending on the value of those inputs, the neuron can be activated. Like brain neurons, the method also contains a number of artificial 'neurons' and uses them to identify and store information. This network consists of input, hidden, and output layers.

The neurons take input data and simple operations are performed on those data. The results of these operations are passed to other neurons. Whether the result will be passed, is determined by the activation function. The activation function plays an important role for both feature extraction and classification. While in the biological neural network, the size of dendrites varies in accordance with the importance of inputs. In the network, this is achieved by using weights and biases.

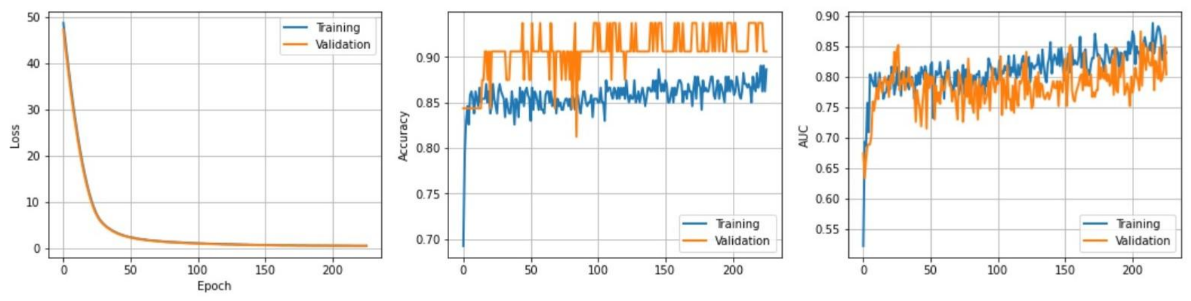
In the implementation of our model, we select basic information about the currency from BigQuery such as current_price, market_cap, and total_volume, then it passes through the feature engineering process previously mentioned (getting information for the specified time window), a feature selection using a variable clustering procedure (VarClus) and a scale and split process. The deep neural network was built considering the specification of certain values such as the initial day (January 1, 2021), the number of days back to start the window (60), the number of days forward to make the prediction (7) and the percentage of price increase in the window (0.1).

We did some iterations of the network changing some parameters on the go. After some tuning, we specified a learning rate of 0.001, a batch size of 16, and ran 300 epochs. The trained network had an initial layer, 5 hidden layers, and an output layer. Each of the layers was decreasing in the number of perceptrons in multiples of 16. We combine leakyReLU and Relu activations for the intermediate ones and determine a sigmoid activation for the output. After the second and third layers, we proposed a dropout of 0.2. As an optimizer, we use "Adam" which is a popular algorithm in the field of deep learning because it achieves good results fast, and as a loss function "Binary Cross Entropy". We choose "Binary Cross Entropy" because it is used for binary classification problems.

Model evaluation

To evaluate the performance of our data product, we used the purely technical metrics of machine learning, as well as tests that we carried out as a team in terms of usefulness. We also reflect whether the product satisfied the objectives set out in the definition of the problem and its limitations.

A summary of the technical model performance is shown in Figure 2. Our model achieves an accuracy of 0.8387. Let's remember that accuracy measures the percentage of cases in which the model has succeeded. Therefore we are in a good position with our specification. The loss function, on the other hand, serves as a prediction error of a deep neural network. In this category, with our Binary Cross Entropy specification we achieved a loss of 0.6023. Finally, we present the result of the AUC measure that measures the performance across all possible classification thresholds. We obtained a result of 0.7827. Overall, this standard machine learning evaluation metrics give evidence that the model is good enough.



- Load best model

```
model = load_model("model_v1.h5")
```

- Testing results

```
test_loss = model.evaluate(x=sp.X_test, y=sp.Y_test, verbose=True)
```

```
1/1 [==============================] - 0s 362ms/step - loss: 0.6023 - accuracy: 0.8387 - auc: 0.7827
```

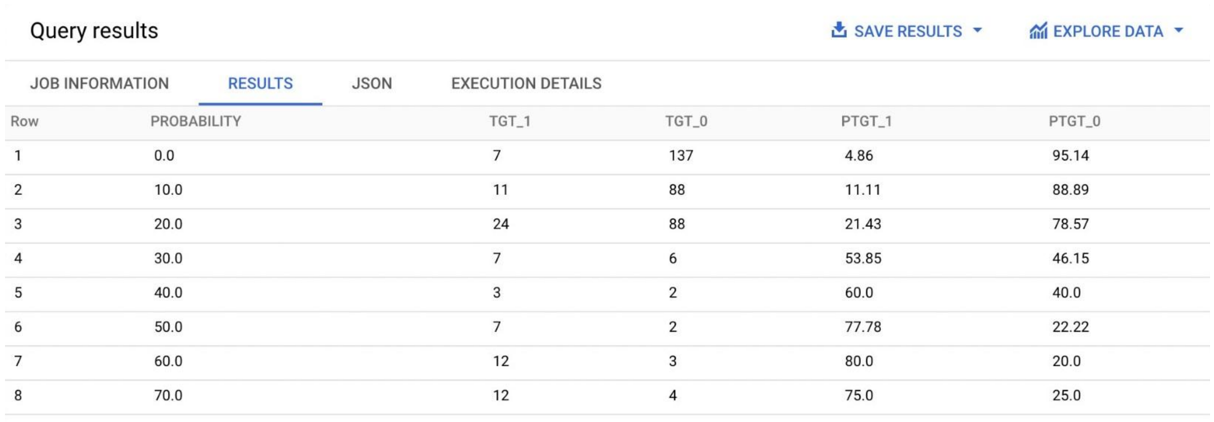
Figure 2.- Graphs about model performance

As a team we also test our data product in terms of its usefulness. We put ourselves in the users' shoes and observed if the predictions made at a certain moment would work for us. Since the first prediction made with the model was made more than a month ago and it was built for the probability of the next seven days, then we could go back to check if the prediction was good. The probability that the price would then rise by 10% for the next 7 days was as low as 15% and fortunately for our purposes (unfortunately for crypto users) the price did not go up. We became confident that the product accomplishes the goal of giving information to the users.

The main limitation we see to our product and model is more related to the current characteristic of the cryptocurrency market. Cryptocurrencies' prices have been decreasing dramatically for the last months with fatal destinies like that of Luna, one of the top 10 most used crypto, which has crashed and its price was virtually $0. If this tendency continues our model will become useless, the probability that the price will go down will be almost always high.

Model Serving

Table 1 shows the results of our prediction model, you can see the number of times that effectively the price goes up compared with the probability that our model has. The probability is incremental (by 0.1). Therefore, in the cases with more probability from our model, we expect seeing more times the price going up from our data


Query results

Row	PROBABILITY	TGT_1	TGT_0	PTGT_1	PTGT_0
1	0.0	7	137	4.86	95.14
2	10.0	11	88	11.11	88.89
3	20.0	24	88	21.43	78.57
4	30.0	7	6	53.85	46.15
5	40.0	3	2	60.0	40.0
6	50.0	7	2	77.78	22.22
7	60.0	12	3	80.0	20.0
8	70.0	12	4	75.0	25.0

Table 1.- Accuracy of our predictions compared with the real movements of the crypto price

With this information, we decided to establish the cut-off point at 0.6 in which 80% of the times we correctly predict that the price goes up. The lift is 4 times superior from the total of the sample.

An additional function would be to create an interface to consult the predictions of the model easily. Currently the process only trains the model and gives the prediction.

We tried as little as possible not to have manual components for automation, but if one component fails we propose to debug the problem and then execute our Airflow instance normally, because our pipeline searches missing information until the current execution day and makes the actualization of all these information in order to do not lose any information day.

Reflection

We believe that we have achieved all of our major objectives with the **“Mastering cryptocurrency price movements with Machine Learning”.** Our project worked as requested in the checkpoints. We decided to implement it only for Bitcoin because it is the most popular crypto and has a major number of potential users. We were able to make accurate predictions with a deep neural network. Thanks to the excellent skills of a team member the final part was not complicated as we expected.

We used the Google Cloud Platform (GCP) to implement the data architecture project and we were able to complete a minimal useful workflow: extract historical data from the API and storage it in a Bucket, perform a basic EDA to visualize trends, execute queries inside BigQuery to get a glimpse of the data, make a basic feature engineering of the variables, test candidate ML models to make the prediction if the price goes up or goes down, tune the hyperparameters of a deep neural network, package the model into a Docker container, and deploy it from an endpoint to make a prediction. Tools from the GCP helped us a lot.

We strongly believe that part of the success of this project was due to the formation of the team. Bringing together profiles with different skills enriched the work dynamics. Assigning the teams by balancing the profiles was an excellent idea. Since the first meeting, we have worked in a fun and friendly way. We appreciate that very much. We worked as a team and each member did their best. Something we liked was the willingness of everyone to help resolve the doubts that arose in the process of implementing this end-to-end data science project.

However, we did not measure all the scopes. We tried to implement the model for different cryptocurrencies, but in the end, we were unable to deploy the data science product for the other cryptos. We did not realize that implementing this model for more cryptos requires more time. We are sure that if we had more time for the project we could replicate it for the other currencies. We also came short in the creation of a web application that would allow users to interact more easily. With more development, the product could even be sold to companies as crypto.com or Bitso.

Today this application is very important. The price of bitcoin and a number of other notable digital assets dropped heavily at the start of December. In early 2022 cryptocurrencies fell again. Bitcoin’s price dropped by around $40,000 per coin towards the end of January and has sunk even lower as the year went on, with the price falling below $30,000 per coin in May.

In the next steps, it would be very interesting to implement the same model to predict the crash of the LUNA cryptocurrency. We already collected the data. We can do a retrospective analysis to predict the crash of LUNA crypto in order to see if our deep neural network could save some investors from bankruptcy.

Broader Impacts

This application is designed for people who want to invest in cryptos and need accurate information to seize their money. We strongly believe that better information leads to better decisions and that this should be attainable for everyone, not only an elite or a small group of people.

Taking advantage of the fact that the crypto market is relatively new and gained a lot of popularity in recent months, many companies and individuals are scamming people. Naive individuals around the world have been close to losing it all by falling in the hands of intermediaries with false promises. It is not realistic to believe that the price of a currency will go up forever and with 100% certainty. The unwanted use of our application would be from deceiving investors that could give people wrong information knowing the truth from our model. To counter that effect, we believe that our product can be used to raise awareness about the characteristics of the market by being freely accessible and with a friendly interface that allows the user to know clearly what he is doing.

Another unintended harm of our product could occur when the prediction of our model does not result as expected and an “honest” user loses money along the way. This could be easily tackled by pitching the product in a way that users understand that our results are not taken from a magical crystal ball but from statistical techniques that try to give the best prediction but could be completely affected by chance.

Contributors

Assignment		**Contributor**
Project Proposal	Discuss with your team and propose a Data Product architecture that involves ingesting data from an API and output a ML prediction (online or offline)	Carlos Miguel Uriel
	Prepare your Project proposal and update your README	Carlos Miguel Uriel
ETL/ELT Script	Create a folder called `src`	Uriel
	Make an API GET Request to the data source identified in your project	Uriel
	Add any sql you are using to transform your data as .sql files in the `src` folder of your Github repo	Uriel Miguel
	Include a README in your `src` folder explaining:	Carlos
ML Model Trained	Create a folder called `model`	Uriel
	Include a clean and well documented ipython script with Exploratory Data Analysis (EDA) + 3 experiments (or iterations on your model) call it: `experiments.ipynb`	Carlos Miguel
	Include a clean python/ipython script that does the following: Train a model and save its output in GCS	Uriel

	EXTRA-POINTS. Package the model using python setup-tools or docker compatible with `Models` from Vertex AI or Docker	Uriel
	EXTRA-POINTS. Train the model using `AI Training` from Vertex AI	Uriel
	Save the model in Vertex AI Models	Uriel
	EXTRA-POINTS. Create an endpoint and create a testing script to test the endpoint	Uriel
	Include a Markdown file in your models folder explaining all aspects of your model	Carlos Uriel
Airflow Instance Setup	Create a GCS Bucket called `airflow`	Carlos
	Create a folder in your GCS bucket called `Dags`	Carlos
	Add a folder in your airflow repo called `Dags`	Carlos
	Create a compute engine instance named `airflow`	Carlos
	Follow the steps above to install and configure airflow on your VM	Carlos
Airflow ETL/ELT DAG	Create a DAG on your airflow VM instance that does the following: Runs your ETL/ELT script for the given execution date	Carlos Uriel
	Create a DAG on your airflow VM instance that does the following: Has at least two tasks. At least one task should be dependent on the successful completion of one or more other tasks	Carlos

	(OPTIONAL) Create an Instance Scheduler that does the following: ● Starts your airflow vm at a specified time ● Turns it off at a specified time	Carlos
	Run your DAG for at least 2 and no more than 5 days	Carlos Uriel
	Provide a README in your dags folder in Github explaining where we can find the data generated by your airflow dag	Carlos
Airflow MLOps DAG	Create a DAG on your airflow VM instance that does the following: ● Generates features for the given execution date	Uriel Miguel
	● Create a DAG on your airflow VM instance that does the following: ● Generates predictions for the given execution date	Uriel
	Create a DAG on your airflow VM instance that does the following: ● Has at least two tasks	Uriel
	Create a DAG on your airflow VM instance that does the following:	Uriel Miguel

	<ul><li>At least one task should be dependent on the successful completion of one or more other tasks</li></ul>	
	Please update the README from the previous checkpoint to include the following:<ul><li>[ ] The name of your MLOPs DAG file</li><li>[ ] The name of your MLOPs DAG</li><li>[ ] How predictions are being generated by your dag</li><li>[ ] Where those predictions are stored</li></ul>	Carlos
Final presentation	Written report	Carlos Miguel
	Slides	Carlos Miguel

References

[1] S. Nakamoto
Bitcoin: a peer-to-peer electronic cash system
Work Pap (2008)
https://bitcoin.org/bitcoin.pdf

[2] Statista
Number of Bitcoin block explorer Blockchain.com wallet users worldwide from November 2011 to April 6, 2022
from https://www.statista.com/statistics/647374/worldwide-blockchain-wallet-users/

[3] How Many People Own Bitcoin?
95 Blockchain Statistics
from https://explodingtopics.com/blog/blockchain-stats

[4] How Many People Own, Hold & Use Bitcoins?
Bitcoin Worldwide. Retrieved May 16, 2022, from
https://www.buybitcoinworldwide.com/how-many-bitcoin-users/

[5] Zheshi Chen, Chunhong Li, Wenjun Sun
Bitcoin price prediction using machine learning: An approach to sample dimension engineering
Journal of Computational and Applied Mathematics

[6] Patrick JaquartDavid DannChristof Weinhardt
Short-term bitcoin market prediction via machine learning
ScienceDirect, The Journal of Finance and Data Science 7 (2021) 45e66