International Journal of Basic and Applied Sciences, 14 (2) (2025) 504-519

International Journal of Basic and Applied Sciences

Website: www.sciencepubco.com/index.php/IJBAS
https://doi.org/10.14419/y2zvbk18
Research paper



Enhancing Cryptocurrency Market Predictability Using Artificial Intelligence: A Platform for Predicting Prices and Trends of Bitcoin, Ethereum, and Cardano

Y. Suganya [1] *, Ch. Sudha Sree [2], S. Kayalvili [3], D. Joel Jebadurai [4], M. Sowmya Vani [5], S. Dhivya [6], Sivakrishna K [7], R. Vijaya Kumar Reddy [8]

[1] Department of Computer Science and Engineering, School of Computing, SRM Institute of Science and Technology, Tiruchirapalli, Tamil Nadu 621105, India
[2] Department of Computer Science and Engineering (Data Science), RVR&JC College of Engineering, Guntur, Andhra Pradesh 522019, India.
[3] Department of Artificial Intelligence, Kongu Engineering College, Perundurai, Tamil Nadu 638060, India
[4] Department of Management Studies, St. Joseph's College of Engineering, Chennai, Tamil Nadu 600119, India
[5] School of Computing, Mohan Babu University, Tirupati, Andhra Pradesh 517102, India
[6] Department of Research and Innovation, Saveetha School of Engineering, SIMATS, Chennai, Tamil Nadu 602105, India
[7] Department of Computer Science and Engineering (AI-ML), MLR Institute of Technology, Hyderabad, Telangana 500043, India
[8] Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh 522502, India
**Corresponding author E-mail: suganyay@srmist.edu.in*

Received: May 10, 2025, Accepted: June 18, 2025, Published: June 30, 2025

Abstract

Despite its rapid expansion in the last ten years, the cryptocurrency market remains highly volatile and unpredictable, creating difficulties for both investors and market participants. Using artificial intelligence (AI) and machine learning techniques, this project seeks to build a predictive model for cryptocurrency prices, which will focus on Bitcoin, Ethereum, and Cardano to solve market challenges. The project aims to develop an end-to-end solution covering all stages of data management, which begins with setting up a real-time data ETL pipeline to collect information from the Binance and Yahoo Finance APIs. The system places data into a structured database that holds historical price information as well as technical indicators. Historical data serves as the basis for training and testing machine learning models to forecast price trends, which enhances cryptocurrency trading and investment decisions.

Keywords: *Cryptocurrency; Market Predictability; Artificial Intelligence; Bitcoin; Ethereum; Cardano.*

1. Introduction

Money has been a cornerstone of human civilization, evolving from barter systems and commodity money to coinage, representative money, and finally to fiat currency, especially after the abandonment of the gold standard in 1971. In today's digital era, money continues its transformation with the rise of cryptocurrencies—digital assets underpinned by blockchain technology. Since the release of Bitcoin's white paper in 2008 by the pseudonymous Satoshi Nakamoto, cryptocurrencies like Bitcoin, Ethereum, and Cardano have revolutionized financial systems by removing centralized intermediaries and enabling decentralized, peer-to-peer transactions (Betancourt & Chen, 2021, Aamir & Zaidi, 2019, Weng et al., 2020, Marcot & Penman, 2019). As this market grows, so does its complexity and volatility, creating challenges for investors and market analysts.

Despite its rapid expansion over the last decade, the cryptocurrency market remains highly volatile and unpredictable, presenting significant challenges for investors and market participants. In response to these challenges, this project leverages artificial intelligence (AI) and machine learning (ML) to build a predictive model focused on three key cryptocurrencies: Bitcoin, Ethereum, and Cardano (Corbet et al., 2020, Vidal-Tomás, 2021, Surden, 2021, Nguyen, et al., 2019).

The objective is to create an end-to-end solution encompassing all stages of data handling, starting from a real-time ETL (Extract, Transform, Load) pipeline that collects market data through the Binance and Yahoo Finance APIs. The data is then stored in a structured database enriched with historical prices and technical indicators. This dataset serves as the training and testing ground for ML models aimed

Copyright © Y. Suganya et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

at forecasting cryptocurrency price movements. Ultimately, this predictive framework (Chakraborty et al., 2021; Sattarov et al., 2020) is designed to support more informed trading strategies and investment decisions in an otherwise uncertain market.

2. Literature review

Lee et al. (2018) presented an innovative approach to simulate synthetic Bitcoin transactions using inverse reinforcement learning (IRL) and agent-based modeling. Their study provided insights into trader behavior in crypto markets and illustrated how agents learn optimal strategies from historical transaction patterns. The use of IRL allowed the researchers to model reward functions indirectly from observed market data, which enriched the simulation and enhanced the realism of price movement predictions.

Schnaubelt (2022) employed deep reinforcement learning (DRL) to optimize the placement of cryptocurrency limit orders, demonstrating how autonomous agents can adapt to varying market liquidity and price volatility. This work is significant for algorithmic trading, as it shows that DRL agents can learn and improve trading performance over time by interacting with a simulated limit order book environment.

Lucarelli and Borrotti (2020) developed a Deep Q-Learning (DQL) based framework for portfolio management in cryptocurrency markets. Their model was designed to dynamically adjust asset allocations based on market conditions, maximizing returns while managing risks. The reinforcement learning approach enabled the system to learn effective trading policies without relying on static rules, marking a shift toward adaptive financial decision-making systems.

Koker and Koutmos (2020) explored machine learning (ML) applications in cryptocurrency trading, comparing multiple supervised models, including random forests, support vector machines, and neural networks for return prediction and strategy formulation. Their findings showed that ML models could outperform traditional benchmark strategies, especially in highly volatile markets like Bitcoin and Ethereum, where nonlinear patterns are prevalent.

Makarov and Schoar (2020) took a more structural view of the cryptocurrency market, examining arbitrage opportunities and inefficiencies across exchanges. Their research underscored the fragmented nature of crypto markets and the resulting price discrepancies that machine learning systems could potentially exploit. Although the study wasn't directly ML-based, it offered critical context for why data-driven trading strategies could have an edge.

Meng et al. (2021) proposed an enhanced deep reinforcement learning algorithm called Off-Policy Trust Region Policy Optimization (OPTRPO). This method ensures monotonic policy improvement, overcoming instability issues common in standard DRL models. Though not specific to cryptocurrencies, the algorithm is highly relevant for building robust trading agents that avoid erratic policy updates, improving learning efficiency and convergence.

Lahmiri and Bekiros (2020) investigated chaotic behaviors in intraday Bitcoin trading and proposed intelligent forecasting systems using machine learning techniques, such as Long Short-Term Memory (LSTM) networks and hybrid models. Their results highlighted the unpredictable and nonlinear nature of Bitcoin prices and the effectiveness of hybrid neural architectures in managing this complexity.

Sun et al. (2020) introduced a novel forecasting model based on Light Gradient Boosting Machine (LightGBM) to predict cryptocurrency price trends. Their model combined technical indicators and historical pricing to generate accurate forecasts. Compared to traditional ML methods, LightGBM offered faster training times and superior performance on large datasets, making it well-suited for high-frequency trading environments.

3. Design and specifications

3.1. Specifications

The project needs to be divided into four main sections to be able to create a functional application.

The first section is the ETL to get the data on the prices of the cryptocurrencies and commodities from some API and store it in our databases.

The functional requirements for this section are:

- Create an ELT that makes HTTP requests to the APIs.
- Must make requests to API endpoints with private keys.
- Extract historical data since 2021 of the assets.
- Extract data at a periodic time to have the latest prices of the assets updated.

The second component entails the establishment of a database that will serve as a repository for asset price information obtained from the ETL (Extract, Transform, Load) process. The objective is to successfully store historical data about various asset prices. To fulfill this purpose, a Postgres database has been chosen as the designated storage system.

The functional requirements for this section are:

- Persist information about the historic price of the assets.
- Create a PostgreSQL database to store the information.
- Create an ERD model.

The third component comprises the backend infrastructure, encompassing an API that facilitates access to the data stored in the databases. Additionally, this API incorporates all the machine learning models, enabling the retrieval of future time predictions by invoking specific functions within the API.

The functional requirements for this section are:

- The backend should have an API REST that allows it to interact with the model.
- The backend will hold all the business logic of the project.
- The backend will oversee the prediction model.

The final component entails the development of the front end, which will establish a connection with the backend. This interface will serve as the medium through which the predictions generated by the models will be presented to the end users.

The functional requirements for this section are:

- Have an interface that allows users to select a currency and receive an answer to the future price of that asset, setting some parameters.

3.2. Restrictions

- Since we are going to deploy the models in a virtual machine, it will have its constraints of consumption and resources given by the university. The virtual machine we are going to use has a of RAM 32 GBs and of memory 50 GBs.
- The minimum interval of the cryptocurrency price given by the API of Binance is in minutes.
- We have other data sources with different granularities and historical data limits.
- The virtual network has no permissions for HTTP requests made to the Binance API.

4. Implementation application

The following aspects are explained in this section.

4.1. Project arquitectiure design

In this section, we will explain the following:
- The design of the full structure of the project.
- The selected technologies, programming languages, and frameworks.

As illustrated in Figure 1, our setup involves the utilization of two virtual machines, each possessing specific specifications. The first virtual machine will host the databases, backend, and frontend components, all of which will be deployed within Docker containers.
We end up having two virtual machines with the specifications shown. We will have the databases, backend, and frontend, and all three parts will be created on Docker containers.

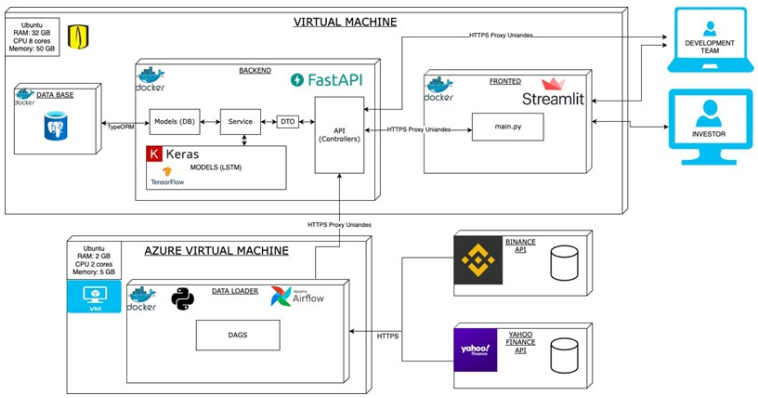


Fig. 1: Application Architecture.

Due to limitations imposed either by the Technical University, Chennai, or the Virtual machine, faced difficulties in making HTTP requests to the Binance API. As a workaround, we opted to deploy the data loader on an Azure virtual machine, which incorporates an Airflow implementation. This setup involves the utilization of various DAGS (Direct Acyclic Graphs, components implemented by Airflow to perform an ETL, that extract cryptocurrency information from the Binance API and Yahoo Finance API. Subsequently, this data is transmitted to the backend for storage and processing.
The main programming language used for all the parts of the project is Python, and we will use a lot of libraries and frameworks such as FastAPI, Streamlit, Airflow, Keras, and TensorFlow.

4.1.1. Data loader

The data loader component will be responsible for orchestrating the ETL process, enabling the retrieval of currency prices from the Binance API and commodities, as well as bond prices from the Yahoo Finance API. To achieve this, we will leverage the capabilities of Apache Airflow to create recurrent tasks that periodically extract the required information. Each DAG within Airflow will contain Python code that interacts with the Backend microservice, facilitating the storage of the obtained data.

4.1.2. Data models

The prediction models employed in this project will undergo training and deployment utilizing the Keras library, which is built upon the Tensorflow framework. Specifically, the chosen machine learning models will be LSTM (Long Short Term Memory) models, enabling predictions to be made on time series data. LSTM models are well-suited for capturing dependencies and patterns in sequential data, making them particularly suitable for our purposes.

4.1.3. Backend

The backend infrastructure will be responsible for hosting the prediction model, storing newly acquired price information from the data loader, and facilitating access to the system via an API. To achieve this, the backend will be deployed utilizing the FAST API framework.

4.1.4. Database

The database will store all the data that is necessary for the project it will follow a structure defined in the ERD model Figure 4 so that we can store historic information on the different cryptocurrencies. The database that we will use is a PostgreSQL database.

4.1.5. Frontend

The front end will be deployed using the Streamlit library in Python and will be deployed in a web browser so that users can interact with the model.

4.2. Data gathering for the algorithm's input and building ETL

To acquire the desired information, as depicted in Figure 1, we employed two distinct APIs. Firstly, the Binance API was utilized to retrieve cryptocurrency prices. Secondly, the Yahoo Finance API was employed to obtain prices of commodities, ETFs, and bonds. These two APIs were utilized in tandem to collect the comprehensive set of data required for our project.
Concerning the Binance API, a single API call enabled us to retrieve 920 price data points. We had unrestricted access to API calls, granting us the ability to retrieve data for all cryptocurrencies listed on Binance across various timeframes, including 1 minute, 1 hour, and 1 day. However, for this project, we made the decision to solely focus on obtaining information for major cryptocurrencies such as Bitcoin, Ethereum, Ada, and Binance Coin, encompassing the timeframes.
The utilization of the Yahoo Finance API granted us access to unlimited historical data; however, the available timeframe was restricted to one day. Consequently, we were able to retrieve and incorporate additional assets from this API, as depicted in Figure 2, to augment our dataset.

```
dict_assets_extra = {
    "gold_price": "GC=F",
    "silver_price": "SI=F",
    "natural_gas_price": "NG=F",
    "cotton_price": "CT=F",
    "coffee_price": "KC=F",
    "sugar_price": "SB=F",
    "cocoa_price": "CC=F",
    "rice_price": "ZR=F",
    "corn_price": "ZC=F",
    "wheat_price": "KE=F",
    "soybean_price": "ZS=F",
    "oats_price": "ZO=F",
    "spy500_price": "ES=F",
    "dow_jones_price": "YM=F",
    "nasdaq_price": "NQ=F",
    "russell_2000_price": "RTY=F",
    "us_10_year_treasury_price": "ZN=F",
    "us_5_year_treasury_price": "ZF=F",
    "us_2_year_treasury_price": "ZT=F",
    "usbond_price": "ZB=F"
}
```

Fig. 2: Extra Assets.

Subsequently, we commenced the development of code responsible for extracting the data, which was subsequently uploaded to our designated databases. For this data extraction process, we leveraged the functionalities provided by Apache Airflow. As illustrated in Figure 3.3, each currency and timeframe combination corresponds to a distinct DAG, which orchestrates the ETL procedures. The ETL process involves retrieving data from the relevant sources and transmitting it via a post-HTTP request to our API for insertion into our dedicated Postgres database. Each DAG performs the ETL process for both currency prices and additional asset prices.

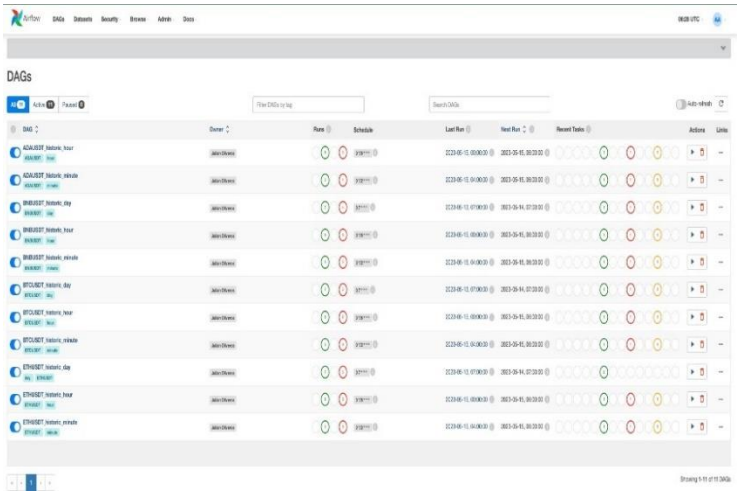


Fig. 3: Airflow Dags.

4.3. Application backend

To be able to create the backend, we use a framework named FastAPI, which allows us to create an MVC architecture design pattern

- Model: will have the entities of the databases we For this, we use Sqlalchemy
- View or Services: will have all the business logic and will use TypeORM to perform DDL (Data Definition Language) operations such as Create, Drop, Alter, and DML (Data Manipulation Language) operations such as Select, Insert, Update, Delete, and Merge.
- DTO: will be used to transfer data and will have almost the same structure as the entities, so that the information can be instantiated when it's received from an HTTP request or sent.
- Controller: will have all the operations and paths of the application API.
- The ML models will have all the machine learning models that were created so that they can be used.

Figure 4 will have the ERD that we will construct for each of these classes. we will have a model, which includes all the attributes and relations to store the data in the database.

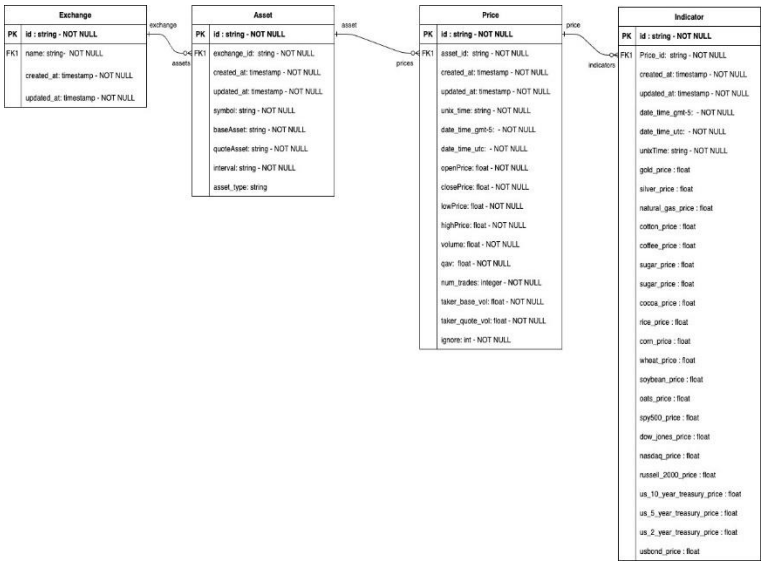


Fig. 4: ERD for the Databases.

All the paths are under the following path <ip>:5000. And to be able to check all the documentation paths, we used Swagger, which can be found under the path <ip>:5000/docs#/. Figures 5 and 6 show how the documentation is shown using Swagger.
The path to perform the CRUD of exchange is in the path <ip>:5000/exchanges/ and will have all the methods shown in Figure 5.
Since the Assets have a many-to-one Relation with the Exchange, the path for the Assets will always depend on the exchange, so the path would be
<ip>:5000/exchanges/{exchange_id}/asset/ and by this path we will have all the CRUD operations.
For the Prices entity, it will depend on the asset, so the path would be
<ip>:5000/assets/{asset_id}/prices/. The CRUD operations are on this path, and we also add 3 more methods:

- assets/{asset_id}/last_price: It will return the last price that is in the app for that Asset. The last price is determined by the last timestamp registered.
- assets/{asset_id}/indicators_unix: This method also receives a Query parameter that corresponds to the unix_time of the price attributes. It will return the price entity related to that unix_time.
- assets/{asset_id}/indicators_unix_between: This method has two Query parameters the unix_time_start and the unix_time_end and will return all the Price entities that have their unix_time between the parameters passed.

The Path for the Indicators method is <ip>:5000/prices/{price_id}/indicators/. A will have the same CRUD and 3 extra methods that the price controller has.
And last, we have the predict method that has the following path
<ip>:5000/predict/{asset_id}/future_time/{future_time}/ this method will receive the asset that is going to be performed the prediction and the future time in the format (1h, 3h, 1d, 3d, 8d) and is going to be the time in the future that we want to predict theses are default values set by the models that we have. The model will return the close_price of the unix_time in the future that it's going to be predicted.

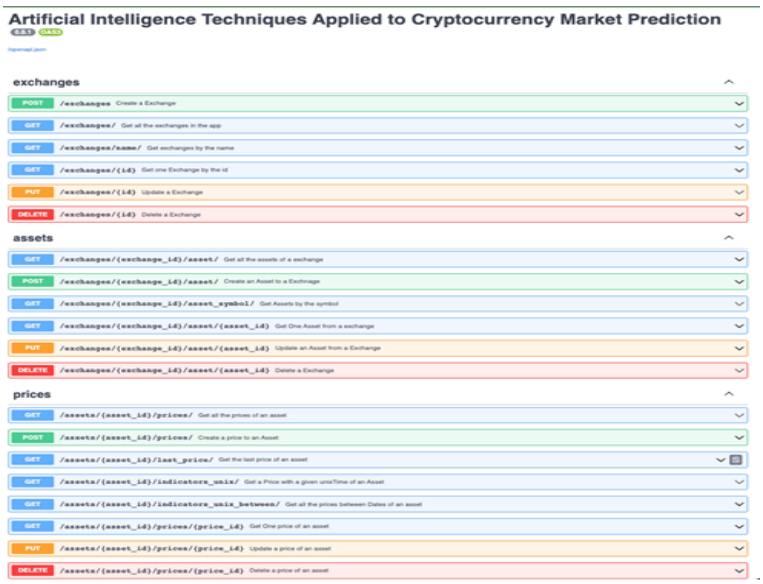


Fig. 5: API Documentation Methods 1.

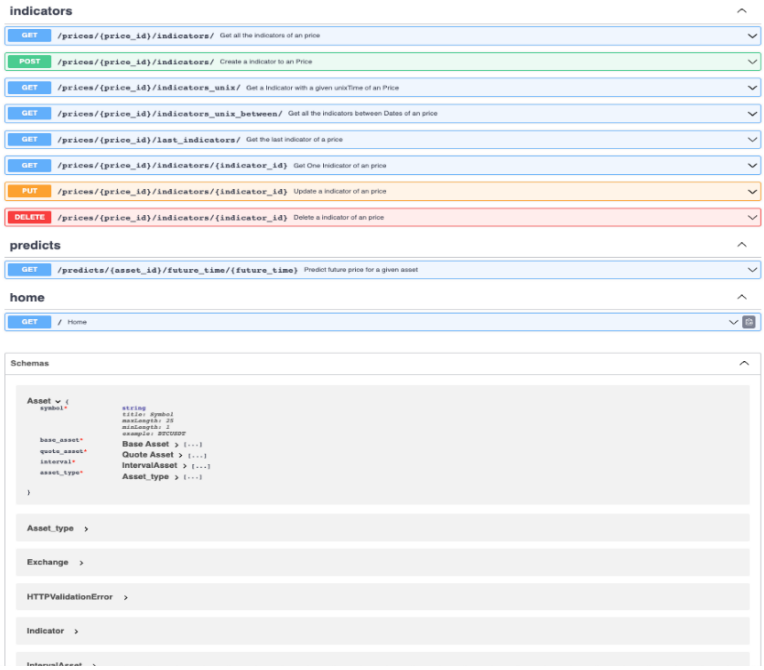


Fig. 6: API Documentation Methods 2.

5. API development for model consumption

Once the training part of the model is done, all the models (The models have a .h5 extension) will be uploaded into the backend so that they can be consumed. The swagger documentation of this method is shown in Figure 7.
The predicted API can be accessed via a GET request directed to the designated URL: <ip>:5000/predicts/{asset_id}/future_time{future_time}. This URL accepts parameters specified by the user, including the asset ID for which the prediction is desired and the future_time parameter denoting the specific time in the future that the user seeks to ascertain. It should be noted that the models are specifically trained to predict certain time intervals, resulting in pre-defined default values such as 1 hour (1h), 3 hours (3h), 6 hours (6h), 12 hours (12h), 1 day (1d), 3 days (3d), and 7 days (7d). Also, it has a query parameter named unix_time_end that represents which timestamp the user wants to make the prediction.

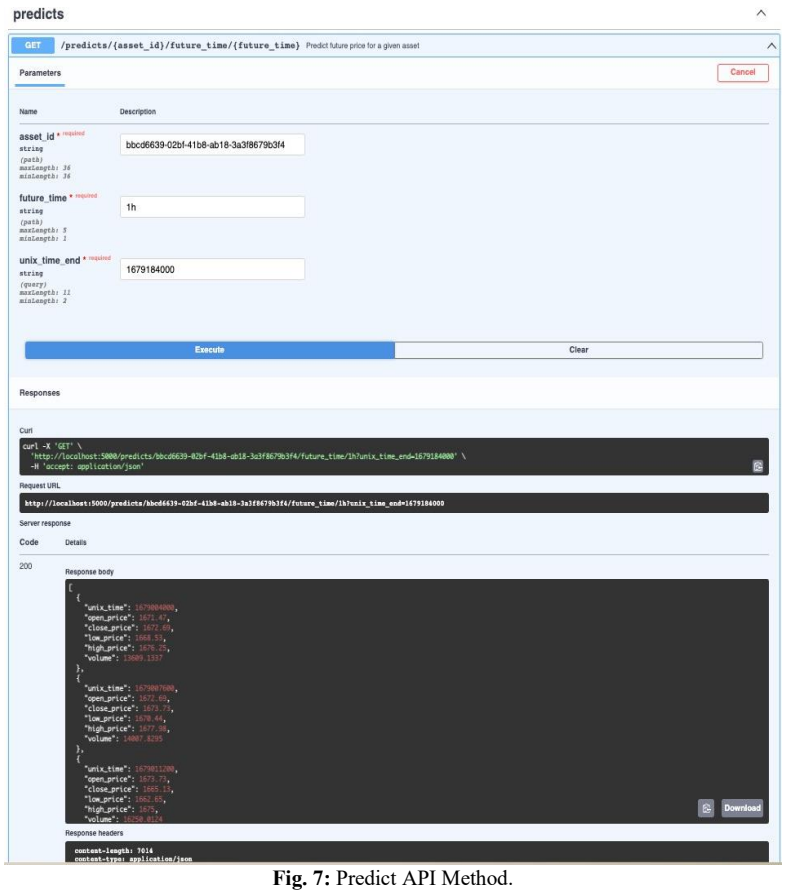


Fig. 7: Predict API Method.

5.1. Build a web app

In this section, we will explain the following:

- How do we create the Front-End of the app so that a person can interact with the ML models?

The front-end is how our final user is going to interact with the app, so that it can predict the prices of a cryptocurrency for different time frames.

The front end was created using the programming language Python and an open-source library called Streamlit, which allows it to show information throw a web browser. The project will be deployed on port 8081. The Frontend will communicate throw the backend API to be able to show the information and run the models.

As shown in Figure 8, the user can choose an exchange, cryptocurrency, a timeframe to predict, and the time in the future that they want the prediction. Also, the user can select a start date and time since it wants to make the prediction, it has to be in the past, and can choose a timezone. Figure 9 shows the first outcome when the investor clicks on the predict button, which will display the value and a chart with the last 30 timestamps and the future timestamps that are predicted. Finally, Figure 10 will display the information in a candlestick chart. The candles that have a blue border mean that these candles are the ones that are predicted, and the other ones are the historical data.



Fig. 8: Front-End.

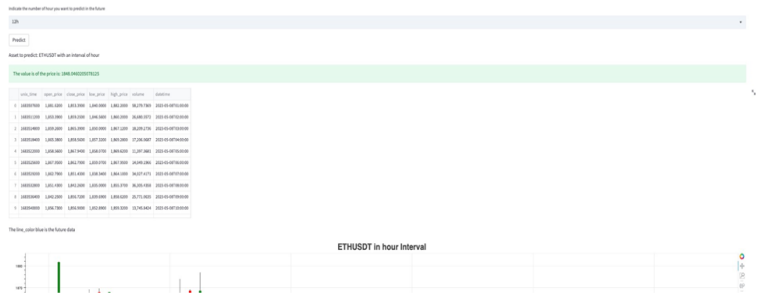


Fig 9: Front-End Output Predictions.



Fig. 10: Front-End Charts.

5.2. Testing and deployment

As shown in Figure 1, the architecture of the app will consist of two virtual machines, one of which was provided by the Technical University, Chennai, and it will have 32 GB of RAM, 8 Cores, and 50 GB of memory. This virtual machine will host the backend, database, and frontend. Each of these components will be hosted in a Docker container, which creates an independent and isolated environment for each component. To be able to deploy all these components, we will use the Docker Compose function, which allows us to create multiple containers that can be connected to the same network. We chose Docker since it's the best option to deploy code, since you only need to have Docker installed and run the command docker-compose up

--build, and no matter the machine, there will be no trouble running the application. Figure 11 shows all the logs of the containers after running the command mentioned before, and Figure 12 shows the configurations of the running containers.
The IP of this virtual machine is 172.24.100.128, and to be able to access it must be configured in the browser.



Fig. 11: Docker Containers Running Virtual Machine.

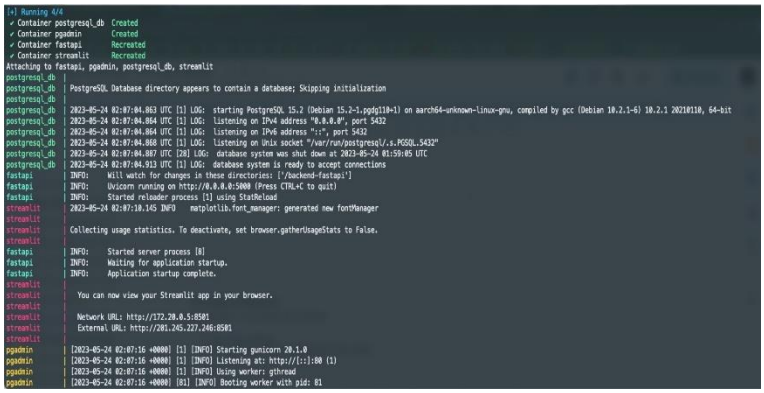


Fig. 12: Docker Containers Logs.

The second Virtual Machine was provisioned by Azure; it has 2 GB of RAM, 1 Core, and 5 GB of memory. This virtual machine will host Apache Airflow so that it can perform the ETL process to get the prices of the cryptocurrencies and commodities. Figure 13 shows all 7 Docker containers needed to run Apache Airflow. Figure 14 shows the configuration on the Azure portal of the virtual machine.

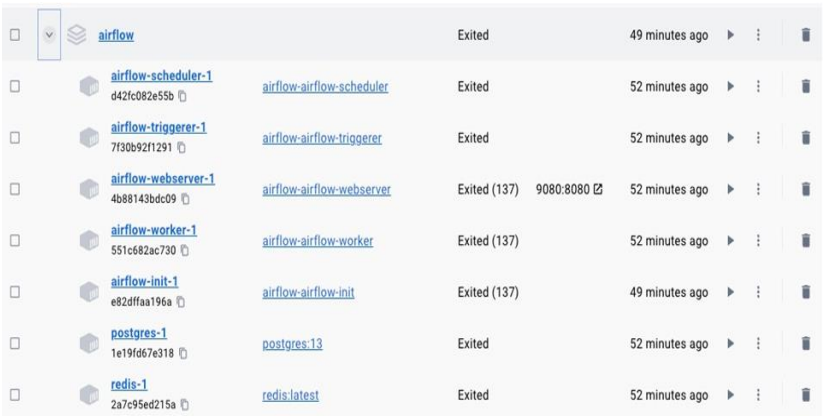


Fig. 13: Docker Airflow Containers.

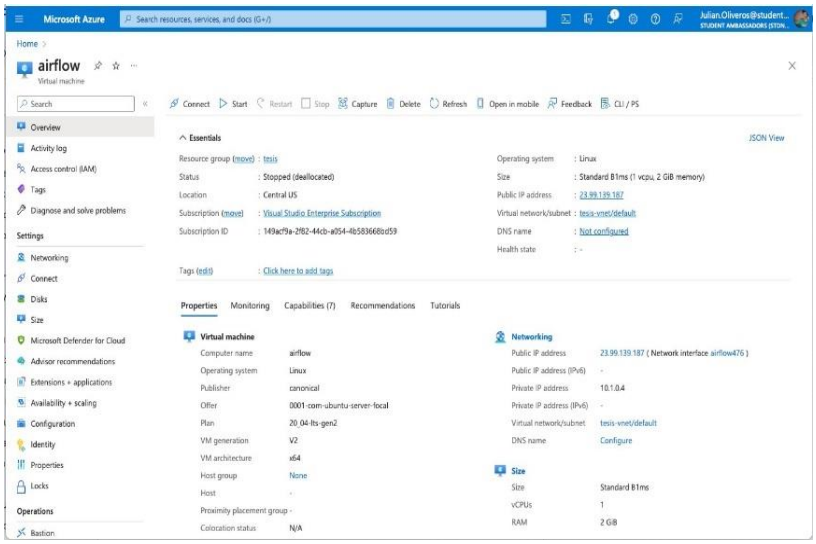


Fig. 14: Airflow Virtual Machine Configuration.

6. Machine learning models

6.1. Dataset

As mentioned before, we are going to use historical market data to predict market movements. For this, we have built a standard dataset for each of the three currencies within the scope of the project. Also, for all the coins, we are observing them compared to the United States Dollar.
The dataset is composed of three main parts:

- Market data
- Calculated Technical Indicators
- External Market Prices

We are looking at historical data since January 1st, 2021, and data will be updated daily with the platform ETL. We have also built the same dataset but with two 2 granularities, Hourly and Minutely. The minute is 60 times larger in terms of storage, as it will have 60 data points for every data point in the hourly dataset. The objective is to determine if more granular data could lead to a more precise understanding of the market. Also, it is important to highlight that these markets are never closed, so we have available 24/7 market information.

6.2. Data preprocessing and normalization

As data is coming directly from the market, which is open 24/7, we have 100% complete data, and it doesn't need any other preprocessing than normalization. The only columns that need special processing are the external market prices. As we are getting other prices from assets that are in the commodities market, they have some restrictions, and that is that they close on weekends, and our source doesn't provide smaller granularity than day by day. For this, we will just take the price of the day and complete weekends and holidays, which are days that the market is closed, with the previous closing price.
The goal of this section is to define the best normalization technique by comparing models only considering information for the year 2022 and comparing their results. As neural networks require values ranging from 0 to 1, we need to normalize our data using a specific technique to transform our real market data into the desired format. There are many possible ways of normalizing data, so we will look at which normalization technique fits better to our data better so that models have a better performance.
For this, we will have an iterative approach in which we will first explore using a couple of Normalization techniques and using three different Neural Network models to define the best-performing model for the task. The second step will be iterating by training and validating the model with data transformed over seven different normalization techniques to determine the best one for our problem.

6.2.1. First iteration

We did a first exploration running three types of neural networks with three different normalization techniques.
Models:

- Recurrent Neural Network (RNN)
- LSTM (Long Short Term Memory Neural Network)
- BiLSTM (Bidirectional LSTM) Normalization Techniques:
- MinMax
- Robust
- Standard Results

Table 1: First Iteration Results MAE index = 100 * Train MAE * Test MAE

	Normalization	Model	Train MAE	Test MAE	MAE Index
5	Minmax	BiLSTM	0.005842	0.055216	0.032059
4	Minmax	LSTM	0.007962	0.045961	0.045939
7	Robust		0.011326	0.056219	0.063078
3	Minmax	RNN	0.015294	0.080912	0.123526
6	Robust		0.016049	0.178159	0.284112
2	Standard	BiLSTM	0.018001	0.320354	0.576814
0	Standard	RNN	0.023459	0.264264	0.6202198

With this first exploration, interrupted due to long running time (6 hours) we got three conclusions

- RNN had the worst results, especially in the testing data.
- Standard Normalization across all three models had the worst performance, so we will not include it in the second iteration
- Bidirectional LSTMs and LSTMs show better performance than RNN, although Bidirectional LSTMs trade off performance with training time, as they are the most intensive and therefore slowest model to train. Because of this, the selected model for the next iteration is the LSTM.

6.2.2. Second iteration

This iteration will also take the 2022 dataset and build an LSTM model for six different normalization techniques, which are trained over the MSE metric to understand which can be the best normalization technique.

Table 2: Second Iteration Results- LSTM

	Scaler	MSE	Val MSE	MAE	Val MAE	MSE Index	MAE Index
5	Normalize	0.000004	0.000004	0.001982	0.001924	0.0	0.0036
1	Minmax	0.000234	0.004136	0.009321	0.058892	0.000098	0.054911
3	Quantile	0.000431	0.024431	0.012534	0.135624	0.0011161	0.170171
2	Robust	0.001634	0.065487	0.0246	0.209576	0.01072	0.521837
4	Power	0.001388	0.114015	0.024292	0.30624	0.015945	0.746894

According to the results, the best normalization candidates are:

- Normalizer
- MinMax

We will explore models with these two techniques, which were by far the best in terms of performance. The results of the Normalize Transformer were biased in the transformation as 90% of the price data got reduced to between values 0.33 and 0.36, which happened because of the distribution of the price data, which does not follow a Gaussian distribution, something that is needed for this type of normalization. Therefore, for the rest of the project, we are going to normalize our datasets with the MinMax transformer to have a first starting point that can give us room for comparability across models and coins.

6.3. Regression model exploration

The models described in this section are designed to predict the price of a given currency at a specified future time.
As mentioned before, we are going to use our final dataset, composed mainly of historical cryptocurrency market data, calculated technical indicators, and prices of other non-crypto assets in the market.

6.3.1. Iteration framework

- Dataset and Cryptocurrency Selection

We have two types of datasets, whose main difference is the time interval or granularity between data points. We have minute-by-minute and hour-by-hour datasets, meaning that for each minute or hour, we have a data point with all the characteristics of the dataset. Both datasets have pros and cons. In this process, we will have to define what can be best used to have better models that lead to better investment decisions. On the other hand, the three selected currencies have different market characteristics and are also influenced by other external factors, so we might find differences in model performance over different cryptocurrencies.

- Prediction Delta Selection

This is the key part of the project, defining what time horizon we are looking to predict. The cryptocurrency market has two huge advantages over more traditional investment markets. The first one is that the market never closes, so you can trade 24 hours and 7 days a week. And the second one is real-time trading, meaning that the investor shouldn't wait for the investment to be executed, enabling even intra-minute trading.
What this means is that we should define at what time horizon we have better-predicting performance, so that in a real scenario, the model is more useful for the investor.

- Model Exploration

To enhance the predictive performance, we will conduct a model exploration phase by experimenting with various neural network architectures and models. Firstly, we will establish a baseline scenario by iterating with simple LSTMs, considering the best dataset and most predictable cryptocurrencies determined earlier, along with the optimized model parameters. This will serve as a control scenario for comparison. Subsequently, we will proceed to explore alternative models to further improve the prediction capabilities.

- Possible next steps based on results

The last step will be to increase the scope of the project based on the results obtained. As previously, we were looking to predict the price at a given timestamp; we can now change the target of the models or look for other characteristics.

6.3.2. First modeling iteration

We need to compare the predictive performance when changing the time delta of the target price. Therefore, we will perform an iterative process with all three coins and all three datasets to define the best time delta for predicting the price. The hypothesis is that as the time delta is longer, meaning the prediction is further away in the future, the performance will be worse. In this iteration, we compare the following time deltas:
Time deltas to consider:

- 1 Hour
- 3 Hours
- 6 Hours
- 12 Hours
- 24 Hours
- 3 Days
- 7 Days

In this iteration process, having two types of datasets, three cryptocurrencies, and 7 different target deltas, we will use only a LSTM of one layer and 32 units as a base model. This will result in 42 models, of which we will compare the results.

Table 3: Performance Summary

Currency (Dataset)	1Hour	3 Hours	6 Hours	12 Hours	1 Day	3 Days	7 Days
Bitcoin (Minute)	0.0003	0.0016	0.0013	0.0009	0.0013	0.0031	0.0008
Ethereum (Minute)	0.0007	0.0011	0.0012	0.0008	0.007	0.0012	0.0005
Bitcoin (Minute)	0.0009	0.0017	0.0036	0.0012	0.0044	0.0053	0.0006
Ethereum (Hour)	0.0016	0.0016	0.0018	0..52	0.0043	0.0021	0.0029
Cardano (Hour)	0.0013	0.0010	0.0004	0.0022	0.0013	0.0014	0.0043
	0.0009	0.0008	0.0022	0.0011	0.0005	0.0004	0.0006

In general, more data means better predictive power, as we can see from the results above, although there are some exceptions, for almost every instance. In the minute-by-minute models, we see an overall better performance, although results from the hour dataset are relatively similar. When considering the computational costs of training the models with the minute-by-minute data, they have a strong disadvantage as they are much more computationally intensive and slow from a time-efficiency perspective. So, on a cost-benefit analysis, it

results in better to use the hourly dataset and optimizing the models. The Cardano (Hour) model exhibits superior performance compared to other models across
different time targets. However, a closer analysis reveals that the validation metrics surpass those of the training data from an early epoch. This raises concerns as it suggests potential overfitting on the validation data or a scenario where the validation sample was easier than the training data due to market conditions. In the best-case scenario, it could imply that the model is exceptionally accurate, but such performance is not typically expected during initial attempts. Consequently, we will disregard this model to avoid introducing complexities and potential issues associated with it. The best-performing coin is Ethereum, which is a big coin and not as volatile as the other two. This behavior can also be a second-order effect of following Bitcoin prices, so the overall behavior ends up being much more stable.

6.3.3. Second modeling iteration

Our next step is optimizing some of these models to try to get a more accurate performance and then draw conclusions based on that. To simplify and speed up the optimization process, we are going to stay with the Hour-by-Hour Datasets and only stay with the largest cryptocurrencies in the crypto market: Bitcoin and Ethereum.
Also, we are going to reduce our prediction targets to only 6 hours of prediction. The rationale behind this is: we have good results on these models, and it is a decent time to make intra-day trading feasible, which is more complicated in shorter time frames. This will leave us in a better position to optimize our models and get better prediction results.
The approach to this analysis will be:

- Coins: Bitcoin and Ethereum
- Prediction Time Target: 6 Hours
- Models to test:
- LSTM
- Bidirectional LSTM
- GRUs

For each model, we are going to try different architectures

Table 4: Second Iteration Results

Model	Specifications	Bitcoin		Ethereum	
		MSE	MAE	MSE	MAE
LSTM	1 Layer- 16 Units	0.0005	0.0172	0.0004	0.0171
	1 Layer- 32 Units	0.008	0.0230	0.0002	0.0116
	1 Layer- 64 Units	0.0019	0.0353	0.0016	0.0316
	2 Layer- 16 Units	0.0015	0.0643	0.0085	0.0805
	2 Layer- 32 and 16 Units	0.0060	0.0736	0.0030	0.0457
BiLSTM	1 Layer- 32 Units	0.0004	0.0173	0.0011	0.0286
	1 Layer- 64 Units	0.0015	0.0296	0.0005	0.0174
	2 Layer- 16 Units	0.0020	0.0360	0.00035	0.0566
Gated Recurrent Unit	1 Layer- 32 Units	0.0031	0.0459	0.0025	0.0378
	1 Layer- 64 Units	0.0009	0.0241	0.0025	0.0426
	2 Layer- 16 Units	0.0035	0.0496	0.0019	0.0337

After iterating with these three models and different architectures, we obtained the results illustrated in the previous figure. In general, for almost every model iterated the results are better for Ethereum by getting lower MSE and MAE. This follows the trend found in the previous iteration. Only Bidirectional LSTMs have better results for the prediction of Bitcoin. This exposes an opportunity to explore more of this type of model, which can be better for a different coin. There is also an interesting pattern for both types of LSTMs; the simpler model seems to have a better performance than more complex models. As we iterate to find better models, we find better prediction results, but are still far from making exact predictions. This follows a common concept of the market called the Random Walk, which says that financial commodities follow a random and unpredictable pattern.

6.3.4. Comparing xgboost with neural networks

To have a more integral exploration of Machine Learning techniques, we are going to compare XgBoost models with Neural Networks. These XgBoost, Extreme Gradient Boosting models have proven to be very effective and are very popular in both academic and industrial contexts as they are flexible, efficient, and versatile models.
To compare both types of models, we will iterate using the first approach as in the first iteration of neural networks, but only for Bitcoin and Ethereum, using only the hourly dataset. Additionally, these models were optimized using cross-validation to determine the optimal parameters.

Table 5: MSE Results

Prediction Delta	Bitcoin		Ethereum	
	MSE	MAE	MSE	MAE
1 Hour	0.0011	0.0225	0.0010	0.0208
3 Hours	0.0007	0.0191	0.0008	0.0199
6 Hours	0.0013	0.0265	0.0013	0.0262
12 Hours	0.0008	0.0207	0.0008	0.0203
24 Hours	0..35	0.0432	0.0029	0.0382
3 Days	0.0025	0.03777	0.0024	0.0374

Like the Neural Networks, we see the same pattern, as the prediction target gets further away in the future, the performance will decrease. Also, Ethereum is a more predictable coin than Bitcoin in general, for almost every prediction target, they have better results than Bitcoin, but the difference in these models is much smaller than in Neural Networks. When compared to Neural Networks, these models have two key advantages: Much faster to train, and they use much less computational cost. Easier to understand than Neural Networks, which are commonly referred to as Black Boxes in non-academic environments, and therefore, XgBoost can be tuned in a desired way. In general,

we get similar results in neural networks, especially in longer prediction delta models; still, the MSEs and MAEs of Neural Networks are much better for the best results. In the best neural network, we got an MSE for a 6-hour prediction of 0.0005 for Bitcoin, while the XgBoost achieved 0.0013, more than two times the result of the Neural Network.

7. Discussion

From a general standpoint, Neural Networks are a very good approach to exploring Cryptocurrency Markets. They have good metrics for predicting the price of currency, and as shown in many cases, they exhibit optimal behavior in the training stage, indicating that we can still optimize our models to achieve better results. The following are the main conclusions of the project.

7.1. Neural networks are resource-intensive machine learning models

For this project, our main restriction or limitation was the computer we got available for the development of the project, a 32 GB RAM machine. Although that is relatively a lot of computing power, we noted it might still not be enough when working with neural networks, as these models require a lot of computer power during the training stage. Therefore, it is necessary to try to optimize the data and the models as much as possible when working with limited computer resources.

In this project, we had two datasets, one with much more information but also 60 times heavier than the other. When training the models with the heavy datasets, we got training rounds of more than 8 hours running, with each epoch of the model taking 25 to 30 minutes. This adds time to the project and a dependency to wait the time for the model to be ready.

The other thing that added training time to the models but also better results was the sequence length, which is the number of data points with all of the characteristics in the dataset that the model will require to predict a point in time. So the greater the sequence length, the more data it will need, and also in training and also it will get better results. So this is an important trade-off to optimize when dealing with these types of models, which deal with time series. Also, it implies a limitation on building models, as less powerful computers would be less capable of supporting larger sequence lengths. In our case, when we did models with sequence lengths greater than 420, considering our 102 characteristic datasets, we encountered out-of-memory errors in the execution.

Finally, architectures also add complexity to training; this can be seen as we add units and layers to Neural Networks or use more complex models as a Bidirectional LSTM, when compared with a simple LSTM.

In the project, we iterated over a difference and built 72 documented models, which got an aggregated training time of 187 hours under our hardware settings. The following table illustrates the relationship of training time with the mentioned factors that affect training time.

Table 6: Comparison of Training Time and Accuracy of Different Models. Assumptions: Sequence Length: 198, Batch Size: 64, Prediction Delta: 6 Hours, Cryptocurrency: Bitcoin

Model	Layers	Units	Training Datapoints	MSE	Training Time (minutes)
LSTM	1	16	680K (Minute Dataset)	0.0004	341
	1	32	680K (Minute Dataset)	0.0005	374
	1	64	680K (Minute Dataset)	0.0011	415
	1	16	115K (Hour Dataset)	0005	52
	1	32	115K (Hour Dataset)	0.0008	62
	1	64	115K (Hour Dataset)	0.0019	65
	2	16-16	115K (Hour Dataset)	0.0015	82
	2	32-16	115K (Hour Dataset)	0.0060	87
BiLSTM	1	32	115K (Hour Dataset)	0.0004	76
	1	64	115K (Hour Dataset)	0.0015	86
XgBoost	2	16-16	115K (Hour Dataset)	0.0020	90
	2	32-16	115K (Hour Dataset)	0.0013	3

In conclusion, when dealing with these types of models, it is primordial to consider and estimate hardware settings and costs that could support intensive computing requirements. Also, it is very helpful to have optimized data and models from an early standpoint and a training schedule so that the machines aren't busy training models during working hours. Also, it is very important to consider other types of models that could be faster, even though they could sacrifice predicting performance like XgBoost.

7.2. The most trivial part of working with neural networks is defining an optimal architecture

In many Machine Learning models, the modeler optimizes parameters based on a desired change in the behavior of the model as they have an expected impact. Different from these models, Neural Networks which for this precise reason are sometimes referred to as black boxes, have an unexpected behavior with changes in their main parameters, especially on architecture. This results in a much more exhaustive optimization process to find the best model for a problem.

In the case of this project, we noticed that when exploring different architectures, and can be evidenced in the training evolution of the different models. The example exposed here will be models predicting Bitcoin price in a time horizon of 6 hours. As shown in Figure 15, there is no clear relationship between the changes in architecture, but the one that changed the most was when adding a new layer. The uppermost images and the bottom left show an expected training curve, with ones adjusting much faster, and the bottom left has a big gap between training and validation results. These first three are models with only one layer, different from the other one, which is an LSTM of two 32-unit layers. This last one was overfitted since the first epoch and remained with the default weights on the units.

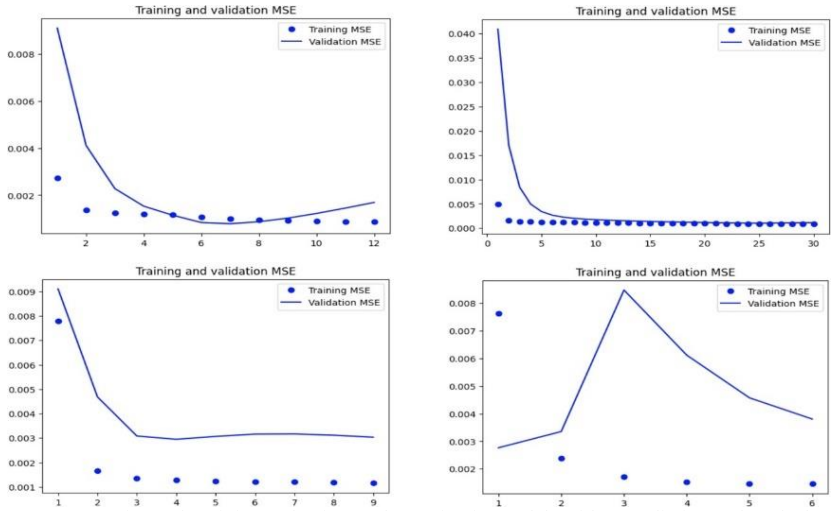


Fig. 15: Comparison of Training Behavior on three Bitcoin Models with a Prediction Delta of 6 hours.

Top Left: LSTM - 16 - 32 Units Top Right: LSTM - 32 - 32 Units Bottom Left: GRU - 32 Units Bottom Right: GRU - Two Layers, 32 Units Each

7.3. Recurrent neural networks are working to predict cryptocurrency markets

Recurrent neural network models, including Long Short-Term Memory (LSTM), bidirectional LSTM, and Gated Recurrent Units (GRU), have demonstrated their effectiveness in predicting cryptocurrency markets. First, cryptocurrencies are known for their complex and highly volatile nature, and as shown in the results, neural networks excel at capturing intricate patterns and relationships within vast amounts of data. Also, they had better results than the XgBoost models. Throughout the project, these three models show great performance with MSEs as low as 0.0003 and MAEs as low as 0.015. On our best one-day prediction delta model predicting Ethereum, we had an average prediction error equivalent to 11 dollars. When compared to the 26 dollars of average daily change in price, this is a very good error, as the model can be overestimating the changes, but still has an accurate prediction.
MSE is a good overall metric, but in the investment industry, the most important thing is predicting the trend of the price of an asset because this is what makes a successful investor.
When looking at the predictions on the validation and testing sets, we could conclude that Neural Networks did a very good job on this prediction task. This can be concluded as the models gave optimistic predictions, meaning they were higher than the real price when the market was changing to an upward trend, and predicted lower than the real price when the market was about to have a downturn. This is an opposed behavior to what happened with the XgBoost prices, which tend to exaggerate current trends instead of anticipating the coming trends.
The following figures show for LSTMs the prediction behavior compared to real prices.

7.4. Ethereum LSTM with 6 hours prediction delta

- The validation set exposes the trend prediction.
- On the test set, it is mostly an optimistic prediction, but also the trend of this period was mainly upward. As shown in Figure 16

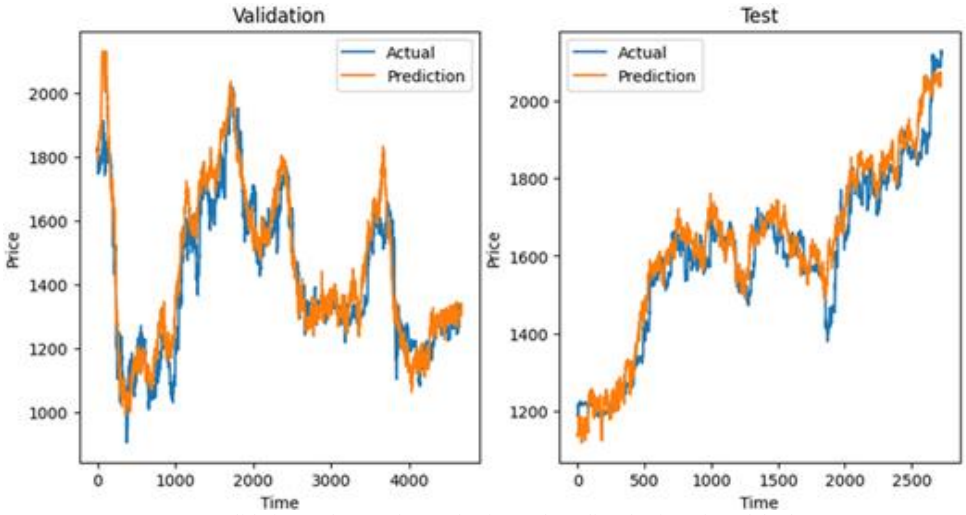


Fig. 16: Validation and Test of Actual Price and Predicted Price of Model 6 hours.

7.5. Ethereum LSTM with 1 hour prediction delta

- Like the previous conclusions, although the prediction was more exaggerated. As shown in Figure 17

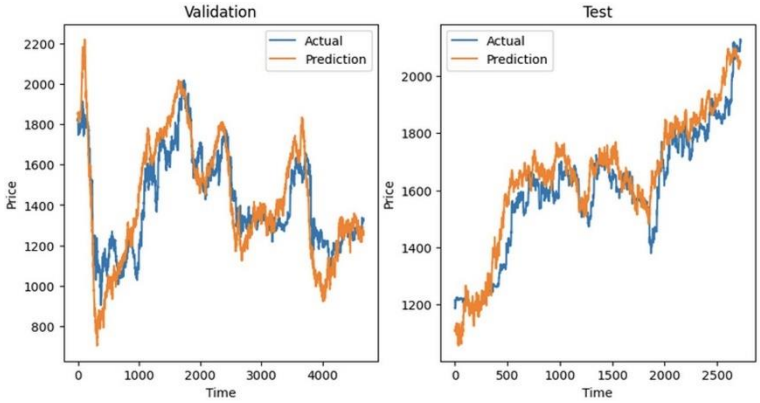


Fig. 17: Validation and Test of Actual Price and Predicted Price of Model 1 hour.

7.6. Bitcoin LSTM with 6-hour prediction delta

- Bitcoin shows in Figure 18 a more volatile behavior, but still, the model accurately predicts the trend in both testing and validation sets.
- Interesting behavior during the first 250 time points of the test set, where the market was at the lowest point of a period. The model quickly started predicting the following increase in price.

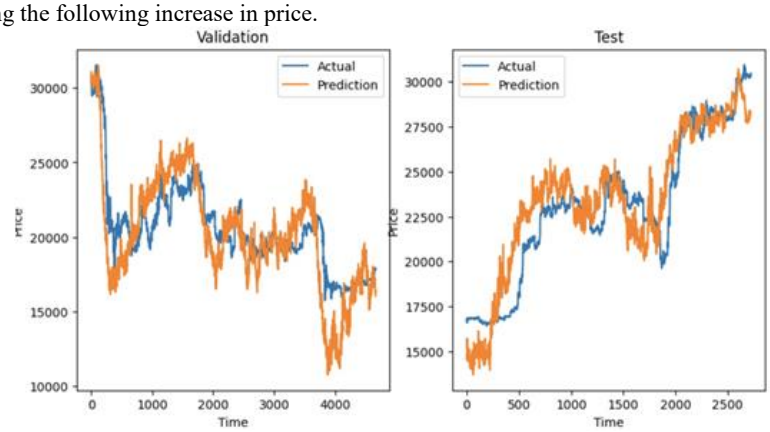


Fig. 18: Validation and Test of Actual Price and Predicted Price of Model 6 hours

7.7. Other modeling conclusions

This section's objective is to recapitulate modeling conclusions that can be found in the modeling section. The following is the list of conclusions.

- Normalizing techniques have a strong impact on model performance. For this market scenario, the best technique was Minimum-Maximum normalizing.
- More data means better predictive performance, although it requires much more computing power. This was evidenced when using the Minutely dataset, the results were better, but they lasted up to 8 times more.
- More complex models don't mean better performance. For the three architectures, Long Short-Term Memory (LSTM), bidirectional LSTM, and Gated Recurrent Units (GRU), there was a pattern that as more complexity was added through units and layers, the performance would worsen.
- Architecture has a big impact on prediction. If we were to order them, bidirectional LSTM was the best, followed by LSTM, and the worst performing was the GRU. This can also be evidenced in training behavior, where GRU has, in most cases, a big gap between training and validation metrics, indicating that the model is not able to generalize as deeply as other models.
- These models are highly dependent on the input data and, therefore the problem that is intended to solve. This makes it important to have data quality checks and data monitoring when the models are deployed, as they are highly sensitive if some variables change.

This can be evidenced when looking at the graphs in Figure 19 that compare predicted versus actual prices for the Cardano coin, as shown in the following figure. The model has a completely different behavior from the data for this coin.

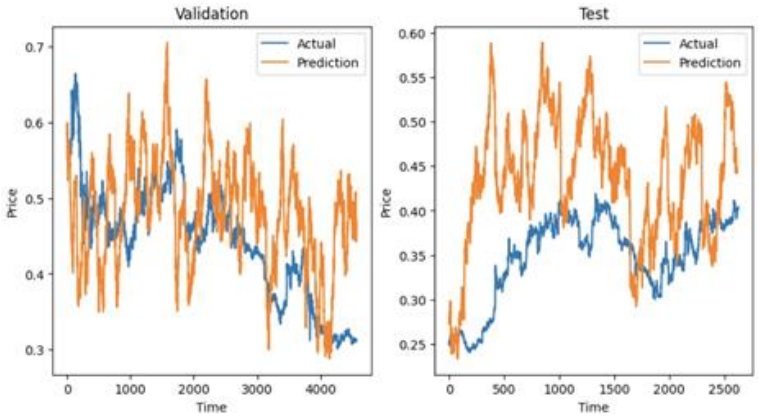


Fig. 19: Validation and Test of Actual Price and Predicted Price of Cardano Model.

7.8. Regression is not the optimal approach to predicting market prices

Regression analysis is a statistical technique that is widely used in many fields because of its ability to establish relationships between variables and make predictions. Some examples where regression is useful are in the field of economics, where regression can be used to analyze the relationship between factors such as income, education, and employment rates. Also in healthcare, regression analysis helps understand how variables like age, diet, and exercise influence health outcomes. However, regression analysis may not be suitable for predicting the price of assets, such as stocks or cryptocurrencies, due to the inherent complexity and volatility of financial markets. Financial assets follow a concept described in financial literature as Random Walk Behavior, which indicates that in the short term, assets follow a random behavior in price movement. This is because asset prices are influenced by a multitude of factors, including economic indicators, investor sentiment, geopolitical events, market trends, and other external factors that might be impossible to quantify. The relationship between these factors and asset prices is nonlinear and subject to sudden shifts, making it challenging for regression models to capture the intricacies accurately.

Instead of focusing on predicting the exact price of future investments, investors are more concerned with understanding the direction in which asset prices are likely to move. The ability to anticipate market trends and make accurate decisions based on the expected price direction becomes crucial. This is where other approaches for these models could be better, especially a classification one.

8. Conclusions

Neural Networks have shown great potential in predicting cryptocurrency prices due to their ability to capture complex temporal patterns in highly volatile data. Despite their effectiveness, they are computationally intensive and require significant hardware resources, making model optimization and data preprocessing critical. Throughout this project, Recurrent Neural Network architectures such as LSTM, Bidirectional LSTM, and GRU delivered promising results, with Bidirectional LSTM outperforming the rest in most scenarios. However, increasing model complexity did not always translate into better performance and often led to overfitting or excessive training time. Importantly, the models were better at capturing price trends rather than exact values, suggesting that classification approaches focusing on trend direction might be more practical than regression. The success of these models was also highly dependent on the quality and scale of data, as well as the choice of normalization techniques. Furthermore, platform tools like FastAPI, Streamlit, Airflow, and Docker enabled effective deployment and visualization. Future work should explore classification-based modeling, model monitoring in production, and the creation of real-time prediction systems for improved decision-making in cryptocurrency trading.

References

[1] Betancourt, C., & Chen, W. H. (2021). Deep reinforcement learning for portfolio management of markets with a dynamic number of assets. *Expert Systems with Applications, 164*, 114002. https://doi.org/10.1016/j.eswa.2020.114002.

[2] Aamir, M., & Zaidi, S. M. A. (2019). DDoS attack detection with feature engineering and machine learning: The framework and performance evaluation. *International Journal of Information Security, 18*, 761–785. https://doi.org/10.1007/s10207-019-00434-1.

[3] Weng, L., Sun, X., Xia, M., Liu, J., & Xu, Y. (2020). Portfolio trading system of digital currencies: A deep reinforcement learning with multidimensional attention gating mechanism. *Neurocomputing, 402*, 171–182. https://doi.org/10.1016/j.neucom.2020.04.004.

[4] Marcot, B. G., & Penman, T. D. (2019). Advances in Bayesian network modelling: Integration of modelling technologies. *Environmental Modelling & Software, 111*, 386–393. https://doi.org/10.1016/j.envsoft.2018.09.016.

[5] Corbet, S., Hou, Y. G., Hu, Y., Larkin, C., & Oxley, L. (2020). Any port in a storm: Cryptocurrency safe havens during the COVID-19 pandemic. *Economics Letters, 194*, 109377. https://doi.org/10.1016/j.econlet.2020.109377.

[6] Vidal-Tomás, D. (2021). Transitions in the cryptocurrency market during the COVID-19 pandemic: A network analysis. *Finance Research Letters, 43*, 101981. https://doi.org/10.1016/j.frl.2021.101981.

[7] Surden, H. (2021). Machine learning and law: An overview. In *Research Handbook on Big Data Law* (pp. xx–xx). Cheltenham, UK: Edward Elgar Publishing. https://doi.org/10.4337/9781788972826.00014.

[8] Nguyen, T. V. H., Nguyen, B. T., Nguyen, T. C., & Nguyen, Q. Q. (2019). Bitcoin return: Impacts from the introduction of new altcoins. *Research in International Business and Finance, 48*, 420–425. https://doi.org/10.1016/j.ribaf.2019.02.001.

[9] Chakraborty, G., Chandrashekhar, G., & Balasubramanian, G. (2021). Measurement of extreme market risk: Insights from a comprehensive literature review. *Cogent Economics & Finance, 9*, 1920150. https://doi.org/10.1080/23322039.2021.1920150.

[10] Sattarov, O., Muminov, A., Lee, C. W., Kang, H. K., Oh, R., Ahn, J., Oh, H. J., & Jeon, H. S. (2020). Recommending cryptocurrency trading points with deep reinforcement learning approach. *Applied Sciences, 10*, 1506. https://doi.org/10.3390/app10041506.

[11] Lee, K., Ulkuatam, S., Beling, P., & Scherer, W. (2018). Generating synthetic Bitcoin transactions and predicting market price movement via inverse reinforcement learning and agent-based modeling. *Journal of Artificial Societies and Social Simulation, 21*, 5. https://doi.org/10.18564/jasss.3733.

[12] Schnaubelt, M. (2022). Deep reinforcement learning for the optimal placement of cryptocurrency limit orders. *European Journal of Operational Research, 296*, 993–1006. https://doi.org/10.1016/j.ejor.2021.04.050.

[13] Lucarelli, G., & Borrotti, M. (2020). A deep Q-learning portfolio management framework for the cryptocurrency market. *Neural Computing and Applications, 32*, 17229–17244. https://doi.org/10.1007/s00521-020-05359-8.
[14] Koker, T. E., & Koutmos, D. (2020). Cryptocurrency trading using machine learning. *Journal of Risk and Financial Management, 13*, 178. https://doi.org/10.3390/jrfm13080178.
[15] Makarov, I., & Schoar, A. (2020). Trading and arbitrage in cryptocurrency markets. *Journal of Financial Economics, 135*, 293–319. https://doi.org/10.1016/j.jfineco.2019.07.001.
[16] Meng, W., Zheng, Q., Shi, Y., & Pan, G. (2021). An off-policy trust region policy optimization method with monotonic improvement guarantee for deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems, 33*, 2223–2235. https://doi.org/10.1109/TNNLS.2020.3044196.
[17] Lahmiri, S., & Bekiros, S. (2020). Intelligent forecasting with machine learning trading systems in chaotic intraday Bitcoin market. *Chaos, Solitons & Fractals, 133*, 109641. https://doi.org/10.1016/j.chaos.2020.109641.
[18] Sun, X., Liu, M., & Sima, Z. (2020). A novel cryptocurrency price trend forecasting model based on LightGBM. *Finance Research Letters, 32*, 101084. https://doi.org/10.1016/j.frl.2018.12.032.