**AICHATBOTBLOCKCHAIN**

**CURRENCYPREDICTIONSYSTEM**

**“ForecastCrypto”**

**Description:**

ForecastCrypto is a sophisticated platform dynamically designed for users, particularly those related to the blockchain community, to engage and communicate with people worldwide and participate in cryptocurrency trading.

This web application is based on Django - a Python framework, infused with Django REST framework, to add comprehensive authentication and authorization mechanisms and make interaction between dierent software components easier by building RESTful APIs. Moreover, a part of this website is built using MachineLearning for successful predictions of the future prices of 20 leading cryptocurrencies. The model for each coin has been carefully trained and tested by feeding it with five-year historical data of each coin that contains attributes such as open price, closing price, high and low prices, adjacent close price, and volume. The data first went through the stages of pre-processing and then a suitable ML model was trained for future predictions.

Just like any other server, like Discord, a user must first authenticate themselves by either logging in to their respective ForecastCrypto accounts or registering themselves if they are new. They can then join a blockchain community out of thousands of communities present on the server, join a room, or they can create their own rooms and start their own blockchain community by leading users to join in successful trading. Here, the user can engage, interact, communicate, trade, and do whatever they wish to do. Furthermore, a separate panel for predictions is present on the main homepage of the server, where the user can select any cryptocurrency, such as Ethereum, of their choice, enter the opening and closing prices, and then predict the future price of the crypto. A chart will also be presented to them that gives users a graphical representation of the prices of that particular cryptocurrency from 2018 - 2023. Based on those prices, a user may wish to find any user worthy of trading and engage in any sort of transaction. Finally, ForecastCrypto also has a feature that allows users to edit their profiles by changing their displays and descriptions, deleting the rooms that they create, and handling messages.

ForecastCrypto stores all the user's data in the database, DBSqlite. The admin can access all the data through Django’s built-in admin panel and can handle users, rooms, and communities. It also stores the displays of the users separately, and its own images statically.

**The 20 leading Cryptocurrencies trained using MachineLearning:**

1. Bitcoin (BTC)
2. Ethereum (ETH)
3. Tether (USDT)
4. Binance Coin (BNB)
5. USD Coin (USDC)
6. Ripple (XRP) USD
7. Cardano (ADA)
8. DogeCoin (DOGE)
9. Solana (SOL)
10. TRON (TRX)
11. Dai (DAI)
12. Polkadot (DOT)
13. Polygon (MATIC)
14. Litecoin (LTC)
15. Wrapped Bitcoin (WBTC)
16. Bitcoin Cash (BCH)
17. SHIBA INU (SHIB)
18. TrueUSD (TUSD)
19. ChainLink (LINK)
20. Stellar (XLM)

**Pre-requisites:**

* VS code is used as an IDE.
* Install Django web framework using pip install Django==5.0
* Install the REST framework for building robust APIs and authentication system in the website using pip install djangorestframework
* Django cors header allows in-browser requests to our blockchain application from other origins. Install it using the command on the

# terminal python -m pip install django-cors-headers

* To work with datasets and preprocessing of raw data in machine learning, we use Pandas. Install it using pip install pandas
* To work with large datasets and for ecient storage and mapping functions, we use Joblib. It is also used as an alternative to pickle files for faster storage and loading of large data sets. Install it using the

# command pip install pandas

* The main tool for implementing machine learning models is the Python sci-kit learn library. Install is using pip install scikit-learn
* The main project name is CryptoProject ● The main app name is base

**Backend:**

* The main step of this blockchain application was to handle the views in the views.py file. The main imports were:



* All the CRUD operations along with predictions and authentication features are managed in the views.
* Decorators are used for applying the login\_required feature on each view.
* The def home(request) view is the main view that handles what features are to be displayed to the user, and also handles .pkl model files that were trained. The models are stored in a dictionary that is passed as a context dictionary for rendering. The context dictionaries handled by this view are:



next main step of this application was to create models in the

models.py file. The four models created are User, Community, Room, and RoomMessage.

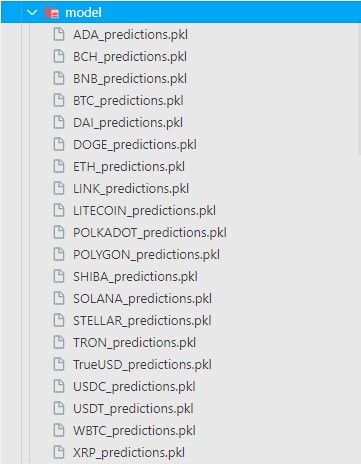
* Along with the built-in User model in Django, we have imported the AbstractUser class which inherits the default User class and adds additional features for the ForecastCrypto Users to be stored in the Database



* In the base, the serializers.py file contains the serializer class for Room that returns JSON data for APIs.
* For handling the REST functionality, we created another file views.py under the api folder in base. The main imports used were:

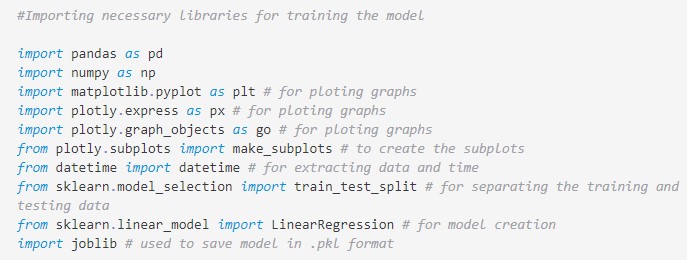


* In the settings.py file, under the installed\_apps, we have added the Django app name, cors header, and rest framework:
* All the models that were trained for crypto predictions are already stored in .pkl files and are stored under the models folder in base



**Machine Learning Models Preprocessing:**

* The necessary imports that were made for training each model are as follows:



large csv files of five-year historical data for each coin were fed for

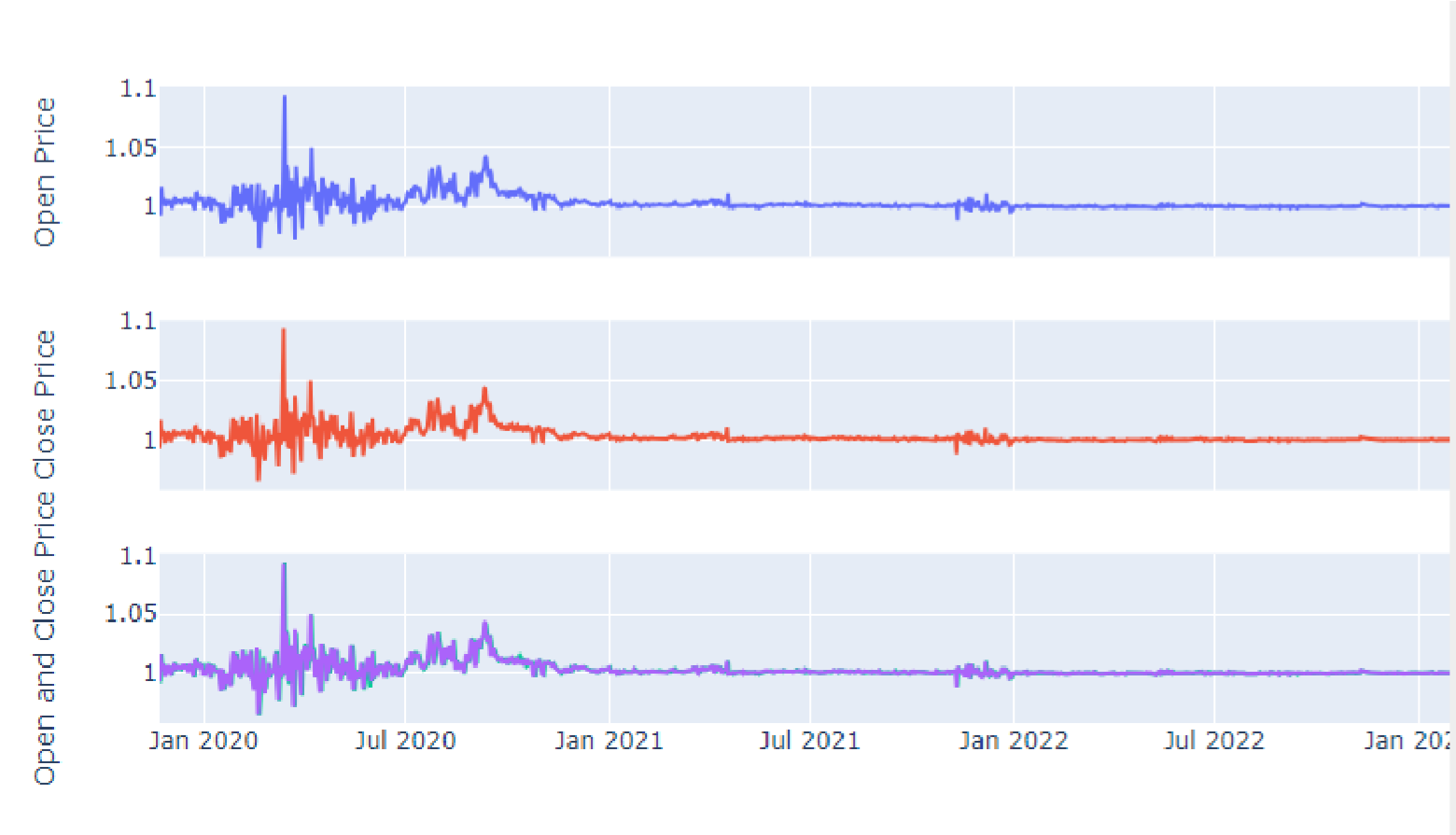
preprocessing as it is raw data, and unnecessary and duplicated columns were dropped.

* Each column for the prices were visualized into plots and subplots using plotly.graph\_objects library.
* Graphs for open prices, high prices, low column values, and close column values were built. For example, DAI coin low price plot was created like this:

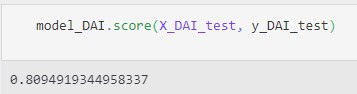


After separate plots were constructed, subplots were visualized for high and low column values, high low and high & low prices values, open and closed prices. For example, subplot for DAI coin looks like:





model was then trained and tested with 0.10% of data. The accuracy was determined for each model. For example, the accuracy of the model for DAI coin is shown below:



* Laslty, user inputs were taken and predictions were performed which showed that the models were correctly predicting the future prices for each coin.

**Frontend:**

* The main.html file contains the main template of this blockchain application.
* The home.html file contains the main structure of this blockchain application. It includes template tagging for defining that which component will be placed on which side.
* The activity\_component.html file contains the user activity panel.
* The form.html file contains the user feature for creating a room or a community.
* The login\_register.html file contains the structuring of complete login and register pages.
* The prediction\_component.html file contains the prediction panel of cryptocurrencies.
* The room.html file contains the room structure that the user gets to see after clicking on each room. It also handles the room messages.
* The topics\_component.html file handles all the communities present on the server.
* The topics.html file contains the page for when the user clicks on the “more” button, the server displays all the communities.
* The update-user.html file contains the structure for editing user profiles where user can also change their displays.

The user\_profile.html file contains the user profile where all the user-created rooms will be shown to other users.

* The style.css file contains all the stylings for the overall website and theme installation.
* All the HTML files contain the Django template engine and template tags, for loops, etc, to add dynamic functionalities and features to the web application. For example, in room.html, we are handling the functionality of sending messages in a room. Only the authenticated users are able to send messages in the room:

