VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF ECONOMICS AND LAW
FACULTY OF FINANCE AND BANKING

FINAL PROJECT

CRYPTO CURRENCIES PREDICTION USING ARIMA

Lecturer: PHAN HUY TÂM

Students:
Nguyễn Tuấn Hưng
K194141723

Tp. Hồ Chí Minh, 06/2022

Table of Contents

1. Overview

In the context of the strong development of the digital age, besides the financial markets that are too familiar to investors such as the stock market, the forex market,... The birth and explosion of Blockchain technology have produced the first coin - Bitcoin and from that milestone until now, the cryptocurrency market is growing strongly with the advent of hundreds of other cryptocurrencies and then gradually gaining a foothold in the market. And as an obvious consequence, the cryptocurrency market is growing in popularity and proving its worth, through the benefits it brings. Cryptocurrencies have a strong advantage over the decentralized financial system - which is supposed to replace traditional centralized finance. The potential that cryptocurrencies bring is really great both now and in the future. Therefore, the author chooses cryptocurrencies as the main financial asset for this topic.

Back to the crypto market for trading purposes, nowadays, the coin is gradually becoming an asset in investors' portfolios. However, accompanied by high returns, the coin is still a risky portfolio because of its high volatility. So, this project aims to build a web application to forecast the trend of coin prices using the Arima model - a fairly famous model in time series forecasting (This project is not aimed at a specific coin, but at all tradable coins on the market).

2. Data processing

The content of this chapter is handled in the file ApiGetData.py

The data was crawled from API of Coinbase with url is https://api.pro.coinbase.com (No key needed like some other sites). Data retrieved by date includes attributes as Open, Close, High, Low and Volume, and price of coin based on USD. By setting params to request as start time, end time and barsize, input is symbol of coin with return json format, we will get the data directly from Coinbase.

```
def getDataApi(sym, timeEnd):
    apiUrl = "https://api.pro.coinbase.com"
    barSize = "86400"

    delta = timedelta(hours=24)
    timeStart = timeEnd - delta * 300

    timeStart = timeStart.isoformat()
    timeEnd = timeEnd.isoformat()

    parameters = {
        "start": timeStart,
        "end": timeEnd,
        "granularity": barSize,
    }

    data = requests.get(f"{apiUrl}/products/{sym}/candles",
                        params=parameters,
                        headers={"content-type": "application/json"})
    return data
```

However, the limitation of this API is that it can only crawl 300 records at a time, so the author must use a While loop to crawl all possible data of a coin.

```
def getAllData(sym):
    timeEnd = datetime.now()
    delta = timedelta(hours=24)
    df_final = pd.DataFrame(columns=["low", "high", "open", "close",
"volume"])

    while True:
        db = getDataApi(sym, timeEnd)
        db = formatData(db)
        if len(db.index) != 0:
            df_final = df_final.append(db)
            timeEnd = timeEnd - delta * 300
        else:
            break

    return df_final
```

Then, format data converts the data type from json to DataFrame, we will get data. (I use def *formatData* in def *getAllData*).

```
def formatData(data):
    df = pd.DataFrame(data.json(),
                      columns=["time", "low", "high", "open", "close",
"volume"])
    df['date'] = pd.to_datetime(df['time'], unit='s')
    df.set_index("date", inplace=True)
    df.drop(["time"], axis=1, inplace=True)
    return df
```

In addition, to diversify the time type and predict in the long term, the data is also formatted from days to a week, two weeks, and a month depending on the input requirements. This means that users will have four data options: day, week, fortnight and month depending on the purpose.

```
def getFinalData(sym, period="DAY"):
    df_origin = getAllData(sym)

    if period == "DAY":
        return df_origin
    else:
        if period == "1WEEK":
            df = df_origin.groupby(pd.Grouper(freq='1W'))
        elif period == "2WEEK":
            df = df_origin.groupby(pd.Grouper(freq='2W'))
        elif period == "MONTH":
            df = df_origin.groupby(pd.Grouper(freq='M'))

        lst_C = []
        for i in df:
            dd = convertData(i)
            lst_C.append(dd)

        final = pd.concat(lst_C)
        final = final[::-1]

        return final
```

The function below processes a tuple into a single record representing that unit of time. As a rule, the close price will be the last price of the period, the high price will be the highest price for the period, the low price will be the low price of the period, the open price will be the first open price, and the volume will be total volume of periods in the data.

```
def convertData(tup):
    index = tup[0]
    dataf = tup[1]
    col = dataf.columns
    high = dataf['high'].max()
    low = dataf['low'].min()
    vol = dataf['volume'].sum()
    close = dataf['close'].iloc[-1]
    open = dataf['open'].iloc[0]
    df = pd.DataFrame([low, high, open, close, vol], columns=[index],
index=col)
    return df.T
```

And for the purpose of the topic based on many coins, and to avoid errors in the data entry step, the project also gets the list of coins from Coinbase, via url https://api.pro.coinbase.com/currencies.

```
def getListCoins():
    url = "https://api.pro.coinbase.com/currencies"
    response = requests.get(url).json()
    newcoins = []
    dct = {}

    for i in range(len(response)):
        if response[i]['details']['type'] == 'crypto':
            s = response[i]['id'] + "-USD"
            newcoins.append(s)
            n = response[i]['name']
            dct[s] = n

    newcoins.sort()
    tup = tuple(newcoins)

    return tup, dct
```

Finally, the data is retrieved quite clean, the required data is quite complete and there is no sign of error, through the above steps, the data is ready to display and feed into the model (This data below is Bitcoin by day).

	low	high	open	close	volume
2021-12-21	46645.05	49339.31	46926.07	48643.92	9155.048092
2021-12-20	45568.00	47548.93	46687.20	46926.07	16039.385104
2021-12-19	46440.11	48351.92	46857.49	46687.19	9518.707128
2021-12-18	45515.72	47368.73	46159.87	46859.46	8170.853032
2021-12-17	45469.32	48000.00	47634.20	46166.50	18556.079000
...	...	...	...	...	...
2015-07-24	276.43	291.52	277.23	289.12	7362.469083
2015-07-23	276.28	279.75	277.96	277.39	5306.919575
2015-07-22	275.01	278.54	277.33	277.89	4687.909383
2015-07-21	276.85	281.27	279.96	277.32	4943.559434
2015-07-20	277.37	280.00	277.98	280.00	782.883420

3. Arima model and App web

3.1 Arima model

The content of this section is handled in the file ArimaModel.py

3.1.1 Introduce

An autoregressive integrated moving average, or ARIMA, is a statistical analysis model that uses time series data to either better understand the data set or to predict future trends. The ARIMA model includes the following input variables (p, d, q):

- Autoregression (AR) (p): refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.
- Integrated (I) (d): represents the differencing of raw observations to allow for the time series to become stationary (i.e., data values are replaced by the difference between the data values and the previous values).
- Moving average (MA) (q): incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

3.1.2 Perform

The input data for Arima models normally requires a large amount of data to ensure that the model is built with sufficient reliability. So the author decided to use the 2-year timeline to issue the warning. (Data less than 2 years old will have a warning).

```
def checkData(self):
    maxday = self.data.index.max()
    minday = self.data.index.min()
    if maxday - minday <= timedelta(days=730):
        warn = "This coin is quite new, the data less than two year, so the
model is not reliable enough"
    else:
        warn = "The length of data is oke"
    return warn
```

The project uses the Close column data as input. However, considering that for coin price data, they tend to be future-specific, which means they are non-stationary. So, to ensure the stationarity of the input data, the project uses the closing price column profit (symbol r_t) through the formula:

$$r_t = \log(\frac{x_t}{x_{t-1}})$$

```
def createDataReturn(self):
    self.dbReturn = pd.DataFrame(np.log(self.data['close'] /
self.data['close'].shift(1)))
    self.dbReturn = self.dbReturn.fillna(self.dbReturn.head().mean())
    return self.dbReturn
```

To ensure that the yield series is stationary, we will use the ADF test to check.

```
def checkStationarity(self):
    result = adfuller(self.dbReturn)
    if result[1] >= 0.05:
        warn = "P-value > 0.05 => Yield series is non-stationary, the model
is not good"
    else:
        warn = "P-value < 0.05 => Yield series is stationary"
```

Because the purpose of building the app is to predict many coins, it is not possible to input the default parameters (p,d,q) because of the different characteristic of each coin. In addition, it can be seen through a simple strategy that the selection of the best model is based on the AIC index. The project will therefore automate this process using auto_arima. For input variables, the author chooses d=0 because the r_t series has guaranteed stationarity. Parameters q and p will be in the range (1,10).

```
model = auto_arima(self.dbReturn, start_p=1, start_q=1,
                   max_p=10, max_q=10, m=1,
                   start_P=0, seasonal=False,
                   d=0, D=0, trace=True,
                   error_action='ignore',
                   suppress_warnings=True,
                   stepwise=False, max_order=10)

self.new_model = SARIMAX(self.dbReturn, order=model.order)
self.result = self.new_model.fit(disp=False)
```

Through the above steps, we have built a specific model for each coin and can be used for prediction.

```
fc = self.result.get_prediction(start=int(self.new_model.nobs),
                                end=self.new_model.nobs + delta - 1,
                                full_reports=True)

prediction = fc.predicted_mean
prediction_ci = fc.conf_int()
```

Then the model will predicts the next series of r_t and returns the actual price. Once I have the return log value, I return the ratio of the difference between the two sessions using the value e^, then use the previous price to calculate the later price, repeat the process and we get the actual price column.

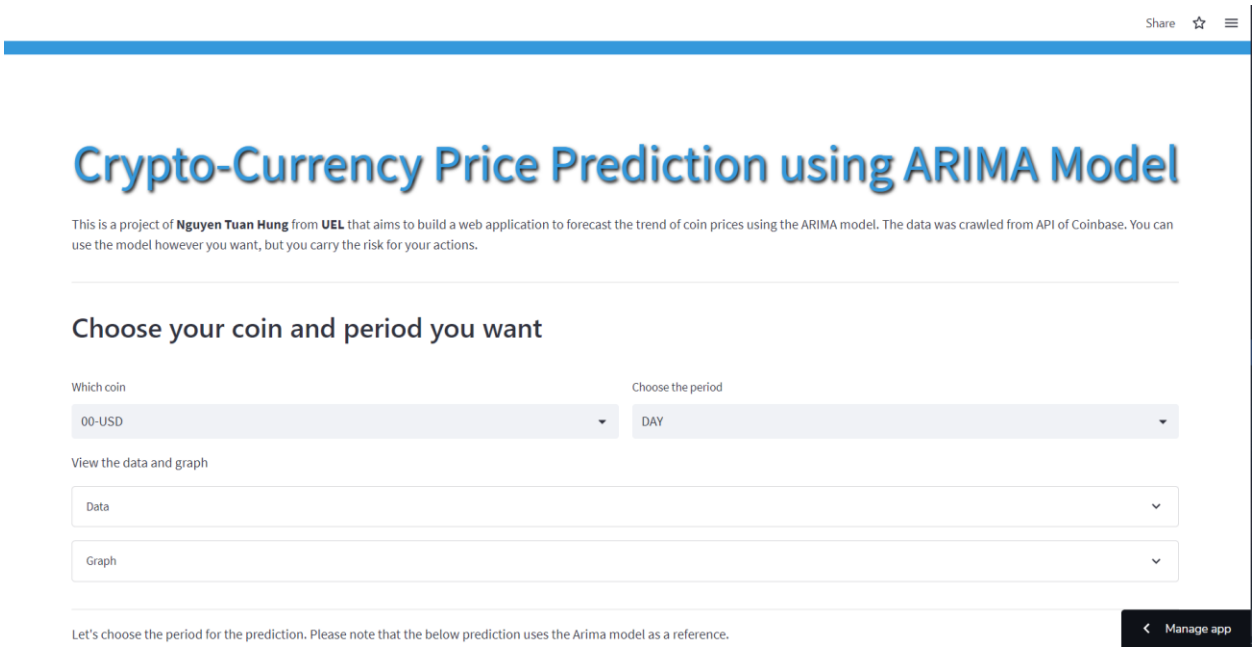
```
def actualPrice(self, lst):
    l_lastprice = list(self.data['close'].iloc[[0]])
    l_exp = list(math.e ** self.dbReturn['close'].iloc[[0]])

    for i in lst:
        a = math.e ** i
        l_exp.append(a)

    for i in l_exp:
        x = l_lastprice[-1] / i
        l_lastprice.append(x)

    l_lastprice.pop()
    return l_lastprice
```

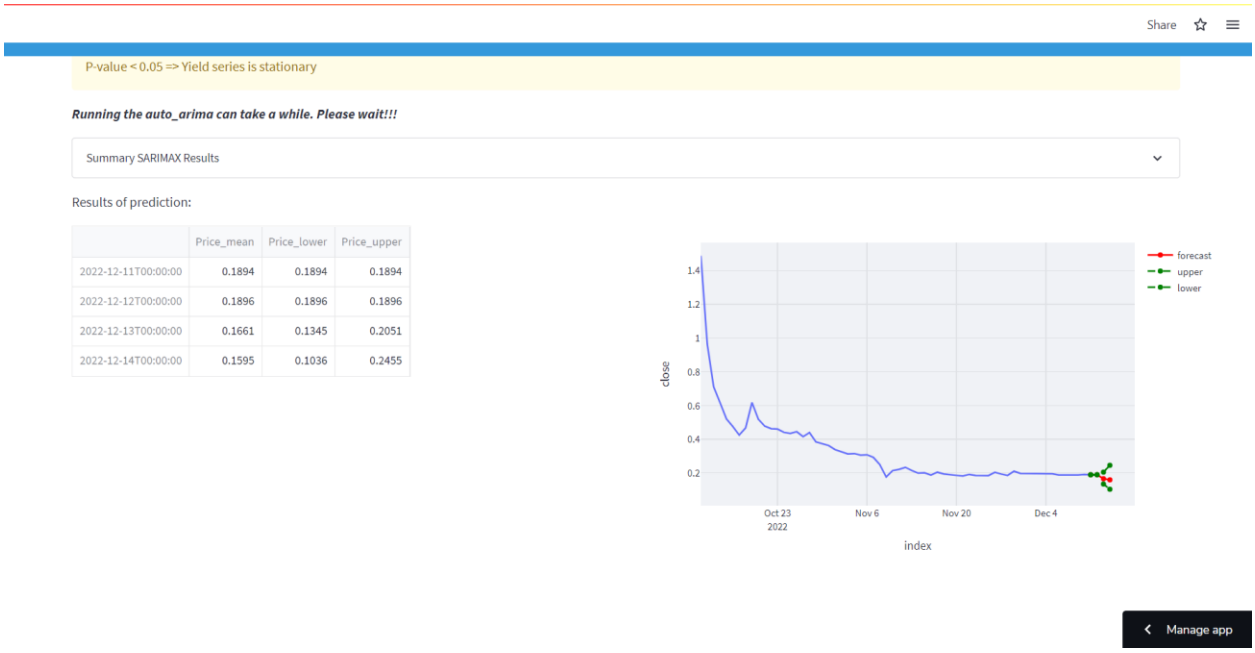
3.2 App Web

The content of this section is handled in the file StreamlitApp.py

After completing the data processing and model building steps, the next step will be to build the app. The project uses Streamlit library in python. Streamlit is an open-source app framework for Machine Learning and Data Science teams.

In this app, you can select your desired coin and period in the sidebar, then view the data and graph. Select the period you want to forecast and press the Predict button, the app will return the model and forecast results for you. You can experience it through the following link: Streamlit





4. Results and Conclusion

Although each coin gives a different set of parameters (p,d,q) resulting in different Arima models. But through general observation, it can be seen that the upper and lower margins of the price fluctuate quite strongly, which can lead to confusion in transactions. For the P-values of the coefficients, with a 95% confidence level, there are occasional coefficients that exceed this level. In addition, the author also found that when predicting the future over a large number of periods, prices will become out of control and unreasonable. Therefore, the author has limited the prediction period to the interval (1, 5). But if investors want to predict in the long term, they can choose data type by week, two weak or month. After all, investors can observe the forecast line (red line) to determine the trend and make decisions.

In summary, the purpose of this project is to provide a tool for investors to refer to, but it is not recommended to completely trust the model's predictions. Surely the project still has a lot of points to improve in terms of modeling, or there may be some bugs in the code (some coin cannot get data) or some other hidden bugs that the author has not discovered yet. After all, the project will try to overcome the weaknesses, try different methods to perfect the model and predict the time series, on the other hand, improve the interface to enhance the investor experience.

Reference:

Khoa học dữ liệu – Khanh’s blog

Quang, B. (2010). Ứng dụng mô hình ARIMA để dự báo VNINDEX.