Project Report

Course Name: Artificial Intelligence

Course Id: 106266

Course Instructor: Siraj Munir

Project Name: Algorithmic Trading (Ai in finance)

Programming Language: Python

Group Members:

1. Muhammad Daniyal – 9815
2. Muhammad Safdar – 9937

Abstract: The project is about generating trade signals based on the technical analysis using three different indicators Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), exponential moving average (EMA) on top trending crypto-currencies, the project can be used with different stocks as well.

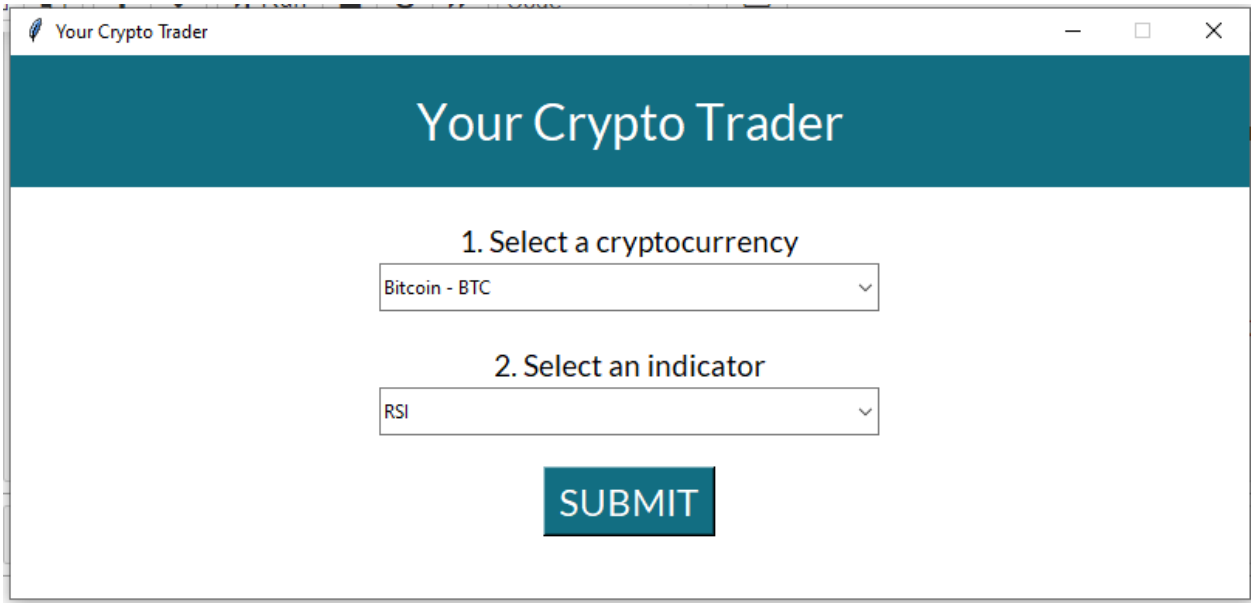
Working Overview: We have provided a small gui which allows the user to choose the crypto-currency and the indicator he is interested in, then we simply visualize the crypto-currency historical data using charts and generate trade singals.

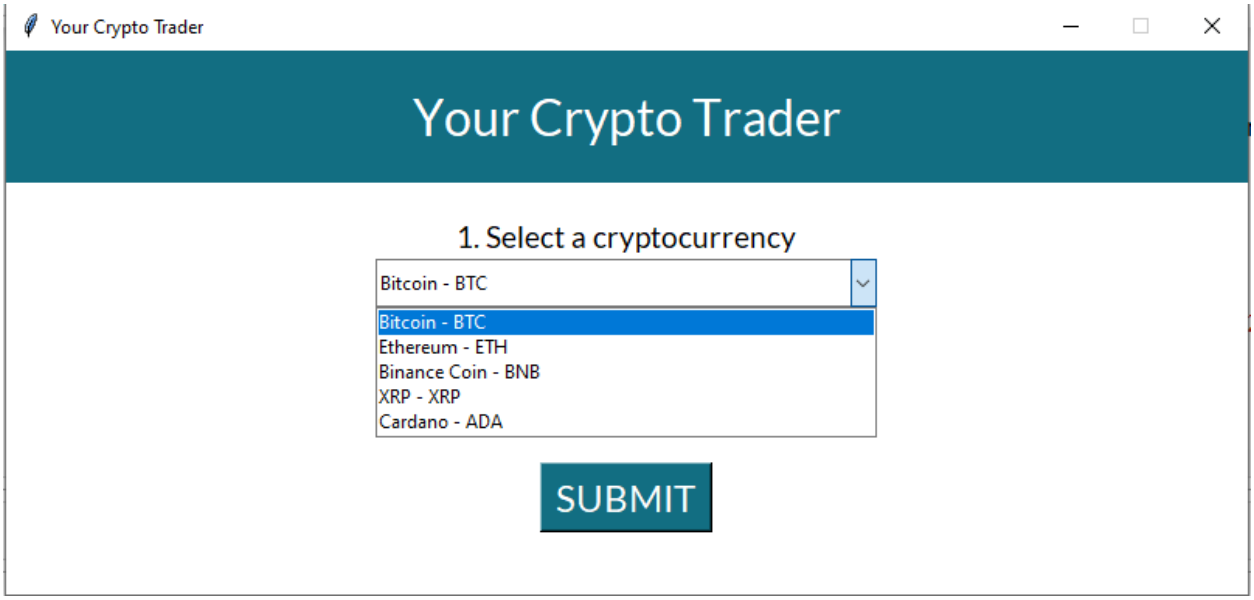
Important Libraries:

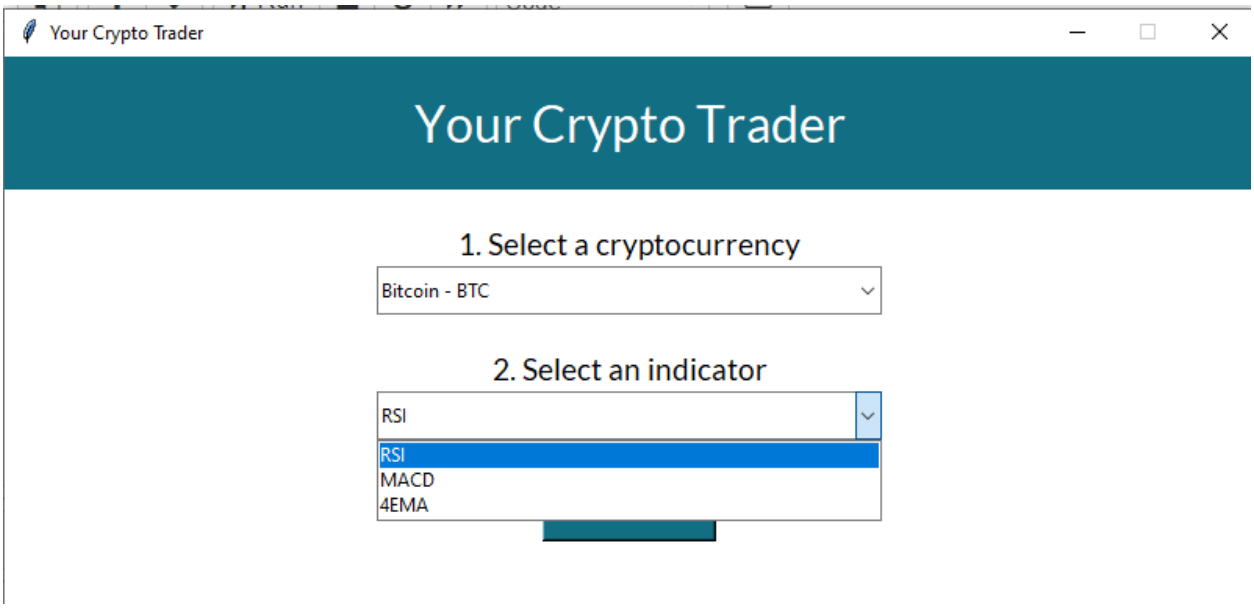
- Cryptocmd (to retrieve historical data of crypto-currency)
- Tkinter (to create GUI)
- Pandas (for data manipulation)
- Plotly (for charting)

GUI:

At the beginning, the user is prompted with a GUI to choose a crypto-currency and an indicator for further proceedings.


Your Crypto Trader
Your Crypto Trader
1. Select a cryptocurrency
Bitcoin - BTC
2. Select an indicator
RSI
SUBMIT


Your Crypto Trader
Your Crypto Trader
1. Select a cryptocurrency
Bitcoin - BTC
Bitcoin - BTC
Ethereum - ETH
Binance Coin - BNB
XRP - XRP
Cardano - ADA
SUBMIT



Getting Historical Data:

Next step is to get and visualize the price over past 1 year for the selected crypto-currency.

```
#Get 1 year historical data for selected crypto currency
def get_price_dataframe(crypto_symbol):
    global master_df

    #Call function to get historical data in dataframe
    scraper = CmcScraper(crypto_symbol, date_one_year_back_string, date_today_string)
    df = scraper.get_dataframe()

    #Set date as index
    df = df.set_index(pd.DatetimeIndex(df['Date'].values))

    #Assign to global variable
    master_df = df
```

RSI:

The relative strength index (RSI) is a momentum indicator used in technical analysis that measures the magnitude of recent price changes to evaluate overbought or oversold conditions in the price of a stock or other asset. The RSI is displayed as an oscillator (a line graph that moves between two extremes) and can have a reading from 0 to 100.

The relative strength index (RSI) is computed with a two-part calculation that starts with the following formula:

$$RSI_{\text{step one}} = 100 - \left[\frac{100}{1+\frac{\text{Average gain}}{\text{Average loss}}}\right]$$

The average gain or loss used in the calculation is the average percentage gain or loss during a look-back period. The formula uses a positive value for the average loss.

The standard is to use 14 periods to calculate the initial RSI value. For example, imagine the market closed higher seven out of the past 14 days with an average gain of 1%. The remaining seven days all closed lower with an average loss of -0.8%. The calculation for the first part of the RSI would look like the following expanded calculation:

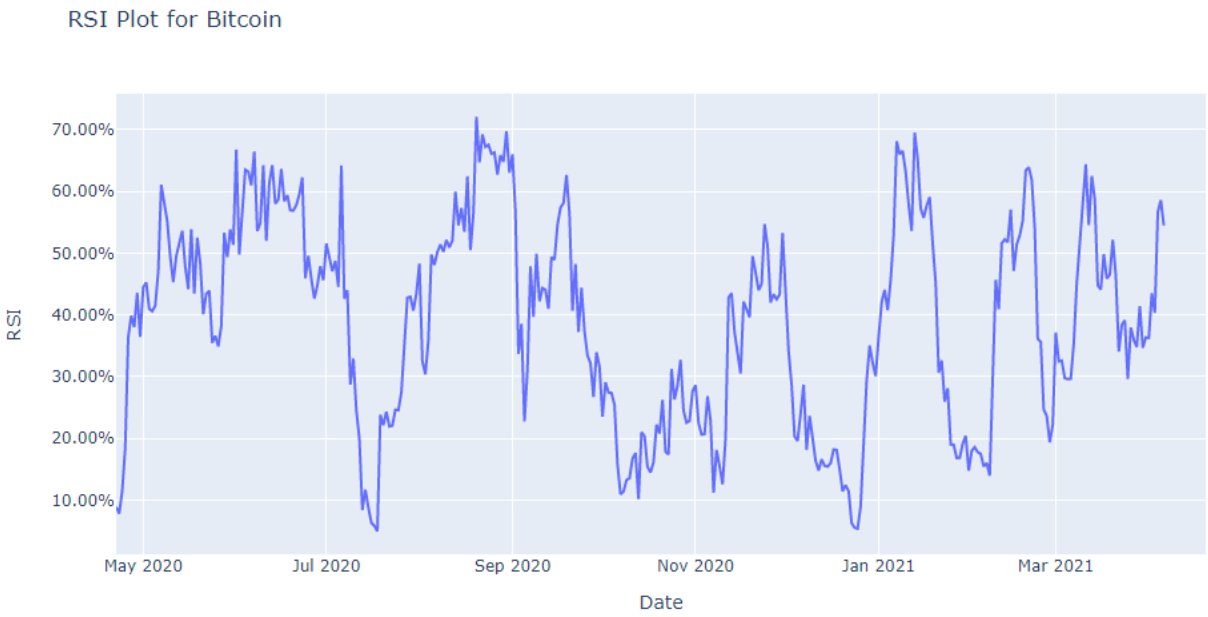
$$55.55 = 100 - \left[\frac{100}{1+\frac{\left(\frac{1\%}{14}\right)}{\left(\frac{-0.8\%}{14}\right)}}\right]$$

Once there are 14 periods of data available, the second part of the RSI formula can be calculated. The second step of the calculation smooths the results.

$$RSI_{\text{step two}} = 100 - \left[\frac{100}{1 + \frac{(\text{Previous Average Gain} \times 13) + \text{Current Gain}}{-((\text{Previous Average Loss} \times 13) + \text{Current Loss})}} \right]$$

```python
#RSI
def calculate_rsi(crypto_name):
    global master_df
    df = master_df

    #Get difference of prices from prev day
    delta = df['Close'].diff(1)

    #Remove NaN
    delta = delta.dropna()

    #Get ups and downs
    up = delta.copy()
    down = delta.copy()

    up[up<0] = 0
    down[down>0] = 0

    #Setting time period of 14 for RSI
    period = 14

    #Average gains and losees
    avg_gains = up.rolling(window=period).mean()
    avg_losses = abs(down.rolling(window=period).mean())

    #Calculate RS
    rs = avg_gains / avg_losses

    #Calculate RSI
    rsi = 100.0 - (100.0 / (1.0 + rs))
```

Then we visualize the RSI% in a line chart over historical data.

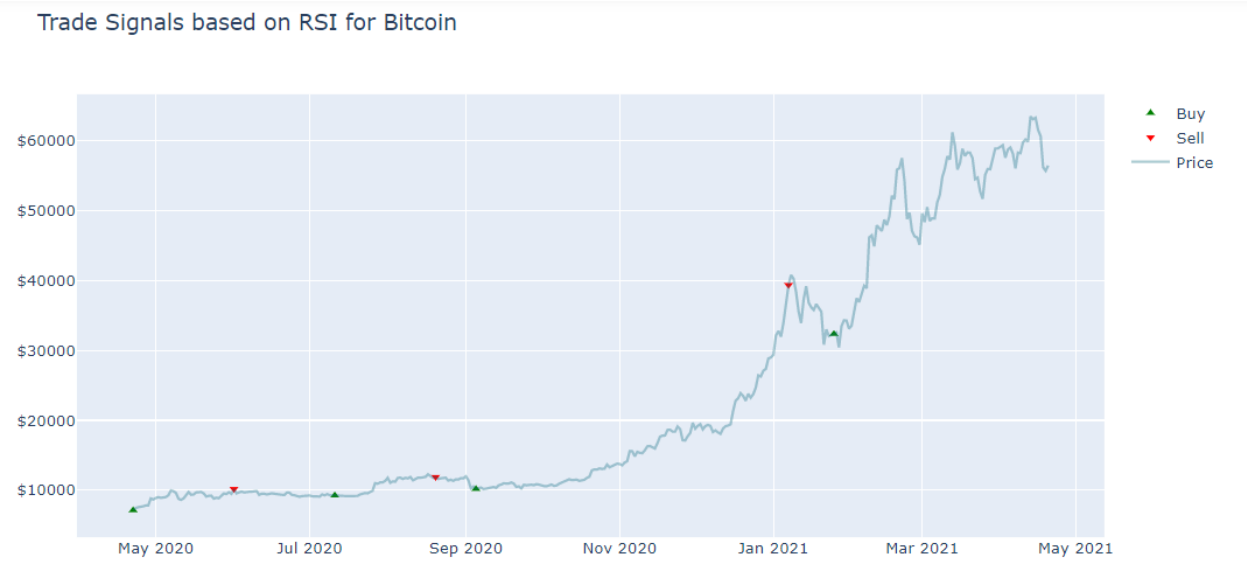


We then, generated trade signals based on the RSI%, if RSI% was less than or equal to 25, it means it's time to buy or take entry in trade, and if the RSI% was greater than or equal to 65, it meant to sell or take exit from the trade.

```
if rsi_df['RSI'][i] <= 25:
    if flag != 1:
        buying_signals.append(rsi_df['Close'][i])
        selling_signals.append(np.nan)
        flag = 1
        date = rsi_df.index[i].strftime('%d-%m-%Y')
        action = 'Buy'
        price = rsi_df['Close'][i]
        profit = profit - price
        investment_data.append([date, action, price, profit])
```

```
elif rsi_df['RSI'][i] >= 65:
    if flag == 1:
        buying_signals.append(np.nan)
        selling_signals.append(rsi_df['Close'][i])
        flag = 0
        date = rsi_df.index[i].strftime('%d-%m-%Y')
        action = 'Sell'
        price = rsi_df['Close'][i]
        profit = profit + price
        investment_data.append([date, action, price, profit])
```

We then, visualized these trade signals in a line chart with prices for references.



We also displayed profit/loss based on the generated trade signals in a nice-looking pandas dataframe

Date	Action	Price ($)	Profit ($)
22-04-2020	Buy	7117.207479	-7117.207479
01-06-2020	Sell	10167.268101	3050.060623
11-07-2020	Buy	9240.346327	-6190.285704
20-08-2020	Sell	11878.371621	5688.085917
05-09-2020	Buy	10169.567221	-4481.481304
07-01-2021	Sell	39371.042353	34889.561049
25-01-2021	Buy	32366.393049	2523.168000

MACD:

Moving average convergence divergence (MACD) is a trend-following momentum indicator that shows the relationship between two moving averages of a security's price. The MACD is calculated by subtracting the 26-period exponential moving average (EMA) from the 12-period EMA.

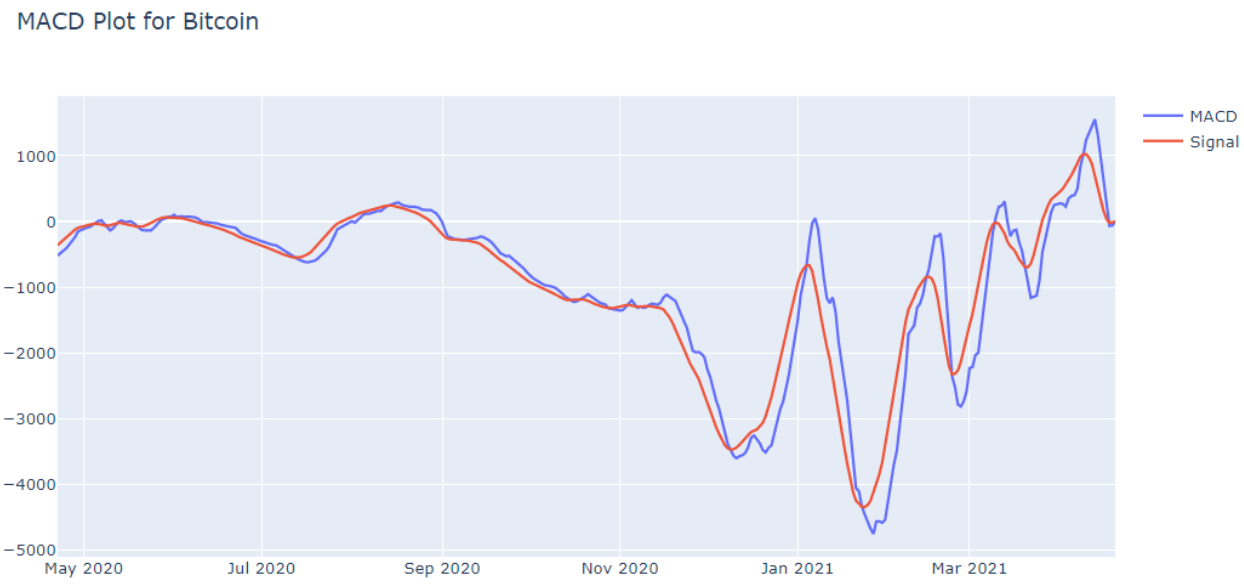
The Formula for MACD Is:

$$\text{MACD} = \text{12-Period EMA} - \text{26-Period EMA}$$

MACD is calculated by subtracting the long-term EMA (26 periods) from the short-term EMA (12 periods). An exponential moving average (EMA) is a type of moving average (MA) that places a greater weight and significance on the most recent data points.

```
#MACD
def calculate_macd(crypto_name):
    global master_df
    df = master_df

    #Calculate short term exponential moving average / ema
    short_ema = df['Close'].ewm(span=12, adjust=False).mean()

    #Calculate short term exponential moving average / ema
    long_ema = df['Close'].ewm(span=26, adjust=False).mean()

    #MACD line
    macd = short_ema - long_ema

    #Signal line
    signal = macd.ewm(span=9, adjust=False).mean()
```

Then we visualize the MACD line and Signal line in a line chart over historical data.



We then, generated trade signals based on the MACD line and Signal line, if signal line is below MACD line, it means we need to buy or take entry in the trade, and if the signal line was above MACD line, it meant to sell or take exit from the trade.

```
if macd_df['MACD'][i] > macd_df['Signal'][i]:
    selling_signals.append(np.nan)
    if flag != 1:
        buying_signals.append(macd_df['Close'][i])
        flag = 1
        date = macd_df.index[i].strftime('%d-%m-%Y')
        action = 'Buy'
        price = macd_df['Close'][i]
        profit = profit - price
        investment_data.append([date, action, price, profit])
```

```
elif macd_df['MACD'][i] < macd_df['Signal'][i]:
    buying_signals.append(np.nan)
    if flag == 1:
        selling_signals.append(macd_df['Close'][i])
        flag = 0
        date = macd_df.index[i].strftime('%d-%m-%Y')
        action = 'Sell'
        price = macd_df['Close'][i]
        profit = profit + price
        investment_data.append([date, action, price, profit])
```

We also displayed profit/loss based on the generated trade signals in a nice-looking pandas dataframe

Date	Action	Price ($)	Profit ($)
05-05-2020	Buy	9003.070178	-9003.070178
09-05-2020	Sell	9593.896734	590.826555
13-05-2020	Buy	9269.987706	-8679.161150
20-05-2020	Sell	9522.981157	843.820006
30-05-2020	Buy	9700.414072	-8856.594065
13-07-2020	Sell	9243.613855	387.019790
14-08-2020	Buy	11768.870619	-11381.850830
16-10-2020	Sell	11322.122686	-59.728143
19-10-2020	Buy	11742.037150	-11801.765294
29-10-2020	Sell	13437.883241	1636.117948
04-11-2020	Buy	14133.707153	-12497.589205
07-11-2020	Sell	14833.754108	2336.164902
08-11-2020	Buy	15479.567038	-13143.402136
09-11-2020	Sell	15332.315597	2188.913461
11-11-2020	Buy	15701.339732	-13512.426271
09-12-2020	Sell	18553.915377	5041.489106
05-01-2021	Buy	33992.429344	-28950.940238
24-01-2021	Sell	32289.378087	3338.437849
15-02-2021	Buy	47945.056832	-44606.618983
23-02-2021	Sell	48824.426869	4217.807886
10-03-2021	Buy	56008.549401	-51790.741516
21-03-2021	Sell	57523.420987	5732.679471
10-04-2021	Buy	59793.235410	-54060.555939
18-04-2021	Sell	56216.185002	2155.629064

4EMA:

An exponential moving average (EMA) is a type of moving average (MA) that places a greater weight and significance on the most recent data points. The exponential moving average is also referred to as the exponentially weighted moving average. An exponentially weighted moving average reacts more significantly to recent price changes than a simple moving average (SMA), which applies an equal weight to all observations in the period.

The Formula for EMA Is

$$EMA_{\text{Today}} = \left(\text{Value}_{\text{Today}} * \left(\frac{\text{Smoothing}}{1+\text{Days}}\right)\right) + EMA_{\text{Yesterday}} * \left(1 - \left(\frac{\text{Smoothing}}{1+\text{Days}}\right)\right)$$

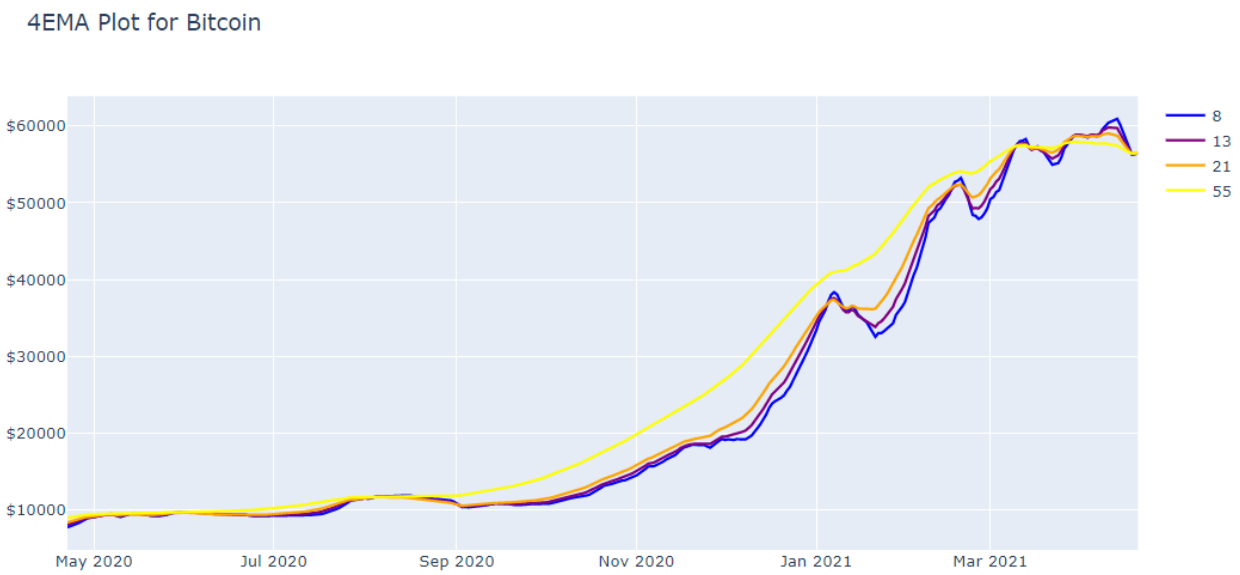
where:

$EMA = \text{Exponential moving average}$

We have used 8, 13, 21, 55 as our four moving averages.

```
#4EMA
def calculate_4ema(crypto_name):
    global master_df
    df = master_df

    #Calculate four moving averages
    length_1 = df['Close'].ewm(span=8, adjust=False).mean()
    length_2 = df['Close'].ewm(span=13, adjust=False).mean()
    length_3 = df['Close'].ewm(span=21, adjust=False).mean()
    length_4 = df['Close'].ewm(span=55, adjust=False).mean()
```

Then we visualize these 4ema lines in a line chart over historical data.



We then, generated trade signals based on these 4 expontential moving averages. 55 works as a support and 8 works as resistence, So, when the resistance line is followed by 13, 21 and support (55), it means we need to buy or enter trade, and when it overlaps each other means, support is followed by 21, 13 and resistance (8) it means to sell or take exit

We also displayed profit/loss based on the generated trade signals in a nice-looking pandas dataframe

Date	Action	Price ($)	Profit ($)
22-04-2020	Buy	7117.207479	-7117.207479
10-03-2021	Sell	56008.549401	48891.341922
20-03-2021	Buy	58313.642609	-9422.300686
08-04-2021	Sell	58323.953580	48901.652894
18-04-2021	Buy	56216.185002	-7314.532109