21.06.2022, Gliwice

Silesian University of Technology

Faculty of Automatic Control, Electronics and Computer Science

BIAI Project

CRYPTOCURRENCIES PRICE PREDICTION
in Python using neural network technologies

Field of study:	Informatics
Academic year:	2021/2022
Group:	1
Tutor:	Grzegorz Baron

Table Of Contents

1. Introduction

The topic of the project was to create a simple program with GUI or web application that would predict prices of several cryptocurrencies such as bitcoin and Ethereum. Then the program would have the ability to visualize the data, i.e. the actual rates and what the rate should be on a given day according to the program. I was aware that this is a difficult topic, but I hoped to get involved with this subject.

2. Task analysis.

2.1 Neural network models

I have found many sources describing how to predict prices in the stock market so predicting the price of cryptocurrencies should be similar.

I analyzed several models of artificial neural networks, generally I was looking for networks with feedback and therefore recurrent networks such as Recurrent Neural Network.

Ultimately, I focused on LSTM, long short-term memory. In general, it's a network that has 3 gates in it that allow separation of important information (long-term) from less important (short-term) and 'memory cell' that can maintain information in memory for long periods of time.

Advantages of LSTM:

- Separation of long-term and short-term memory components
- Relative insensitivity to gap length
- It works really well on different parameters
- LSTMs deal with vanishing and exploding gradient problem by introducing new gates, such as input and forget gates

Disadvantages of LSTM:

- LSTMs take longer to train
- LSTMs require more memory to train
- LSTMs are easy to overfit
- Dropout is much harder to implement in LSTMs

Another Recurrent Neural Network I will be testing is GRU.

Gated Recurrent Units is a type of Recurrent Neural Network a bit simpler than LSTM. Both of them have similar mechanism for storing important information for later. In GRU we have Update and Reset Gates.

The update gate is in charge of identifying the amount of prior information that needs to be transferred to the next state. This is really helpful because the model can choose to copy all past information and eliminate the risk of a vanishing.

The reset gate is used in the model to decide the amount of past information to discard.

It also uses less hypers parameters than LSTM

In my project I use the following networks:

LSTM:

```
            self.model.add(LSTM(units=50, return_sequences=True,
input_shape=(self.x_train.shape[1], 1)))
            self.model.add(Dropout(0.2))
            self.model.add(LSTM(units=50, return_sequences=True))
            self.model.add(Dropout(0.2))
            self.model.add(LSTM(units=50))
            self.model.add(Dropout(0.2))
            self.model.add(Dense(units=1))
```

GRU:

```
            self.model.add(GRU(units=50, return_sequences=True,
input_shape=(self.x_train.shape[1], 1)))
            self.model.add(Dropout(0.2))
            self.model.add(GRU(units=50, return_sequences=True))
            self.model.add(Dropout(0.2))
            self.model.add(GRU(units=50))
            self.model.add(Dropout(0.2))
            self.model.add(Dense(units=1))
```

LSTM-GRU:

```
            self.model.add(RNN(cell = LSTMCell(50),
return_sequences=True, input_shape=(self.x_train.shape[1], 1)))
            self.model.add(Dropout(0.25))
            self.model.add(GRU(units=50, return_sequences=True))
            self.model.add(Dropout(0.10))
            self.model.add(GRU(units=50))
            self.model.add(Dropout(0.10))
            self.model.add(Dense(units=1))
```

2.2 Datasets

At first I planned to use dataset from kaggle.com but after some time I decided to use online dataset from sites like yahoo finance, stooq, navers. The reason for the change was the ability to access the current exchange rates and also the limitation of the previous database (was not updated since 2018). The online database has values from 2016 and so from the time when cryptocurrencies became popular. My program has the ability to load a dataset from a file.

As for the structure, it looks as follows.

Date	Open	High	Low	Close	Volume
2021-07-12	34836.75	35014.90	34730.15	34996.18	344606907
2021-07-09	34457.51	34893.72	34457.51	34870.16	340542786
2021-07-08	34569.01	34569.01	34145.59	34421.93	374853414

What I took into consideration was the day and also the value of the close.

2.3 Tools

- To create the GUI I used tkinter along with azure theme.
- For data visualization I use matplotlib library.
- Pandas, pands_datareader and numpy were used by me to download and manipulate the data
- Datetime to operate on time(days, dates)
- Tensorflow & Keras for model making and training

3. Internal and external specification

The application is designed in MVC architecture. Tkinter in Vie is responsible for the graphical interface, the program logic is in the model.

Project folder structure:

```
Project/
|-------app.py
|-------model.py
|-------view.py
|-------assets/…
|-------images/…
|-------Dokumentacja/…
```

The images and assets folders contain files for the tkinter theme.

The documentation folder contains this report, presentations, and documentation about the project.

3.1 Functions/ methods

There are callbacks in the file that either change the values of the variables in the tkinter, e.g. labels, or call the selections from the model via the controller.

```
        def get_current_value():
          """ Function gest current value from predicted day variable

          Returns:
              string: How many days of predigion is being set
          """
          return ' Based on last {: .0f}days'.format(prediction_days.get())
```

The above function updates the label

```
         def callback_button2():
          self.controller.set_model_currency(currency_type.get())
          self.controller.set_model_data_source(data_source.get())
          self.controller.set_model_model_ID(model_id.get())
          self.controller.set_model_switch_state(switch_state.get())
          self.controller.set_model_predition_days(prediction_days.get())
          self.controller.set_model_future_days(int(future_day.get()))
          self.controller.set_model_test_start_date(int(test_start_date.get()))
          self.controller.do_train()
```

The above function happens when the train button is pressed. Many variables are passed in and also calls the do_train()

There are 8 functions - setters and 3 main functions Train, Plot, Gain named the same as buttons. You can also distinguish the area between __init__ function and the first function where it defines many variables.

```
    def set_predition_days(self, i):
        """ Basic setter for prediction days

        Args:
            i (int): Value to be set
        """
        self.prediction_days = i
```

The seters are standard – they just set int value to variable, there is one string and one bool variable with correct setters for them.

3.1a train()

This function can be divided into 3 stages:

- checking whether to download the dataset or load it from a file
- preparing the data
- network creation based on selected model

```
        x_test = []
        for x in range(self.prediction_days , len(model_inputs)):
            x_test.append(model_inputs[x - self.prediction_days :x, 0])

        x_test = np.array(x_test)
        x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))

        prediction_prices = self.model.predict(x_test)
        prediction_prices = self.scaler.inverse_transform(prediction_prices)
```

Prediction based on model

```
            print('LSTM')
            self.model.add(LSTM(units=50, return_sequences=True,
input_shape=(self.x_train.shape[1], 1)))
            self.model.add(Dropout(0.2))
            self.model.add(LSTM(units=50, return_sequences=True))
            self.model.add(Dropout(0.2))
            self.model.add(LSTM(units=50))
            self.model.add(Dropout(0.2))
            self.model.add(Dense(units=1))
```

Example model, models are selected based on model_id-{1,2,3) and are being changed in the view radiobutton in frame3. There is a callback that calls function from model via controller.

3.1b plot()

This function can be divided into 3 stages:

- selecting the data range for the chart
- analysis
- drawing a chart

```
        plt.plot(actual_prices, color='cyan', label='Actual Prices')
        plt.plot(prediction_prices, color='indigo', label='Predicted Prices')
        plt.title(f'CryptoCurrency Price Prediction')
        plt.xlabel('Days')
        plt.ylabel('Price')
        plt.legend(loc='upper right')

        plt.show()
```

Making simple chart with matplotlib

3.1c gain()

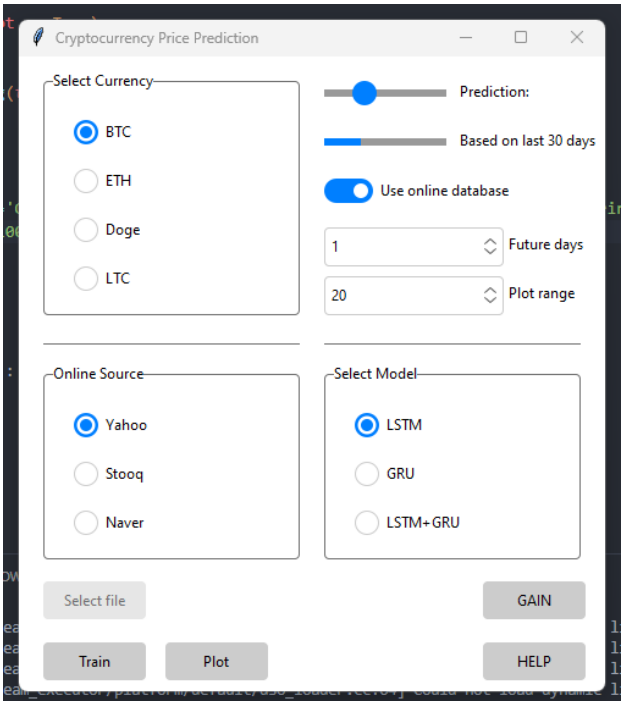
This function takes the initial and final values from a given range and then returns the percentage of the final value in relation to the initial value.

```
    def gain(self):
        """Main fuction for calulating gain

        Returns:
            numpyInt: It returns gain
        """
        start_date_gain = dt.datetime.now() - dt.timedelta (days =
self.prediction_days)
        df = web.DataReader(f'{self.crypto_currency}-{self.against_currency}',
self.data_source, start_date_gain , dt.datetime.now())
        x =df['Close'].iloc[:1].values
        y =df['Close'].iloc[-1:].values
        return (y/x * 100)
```

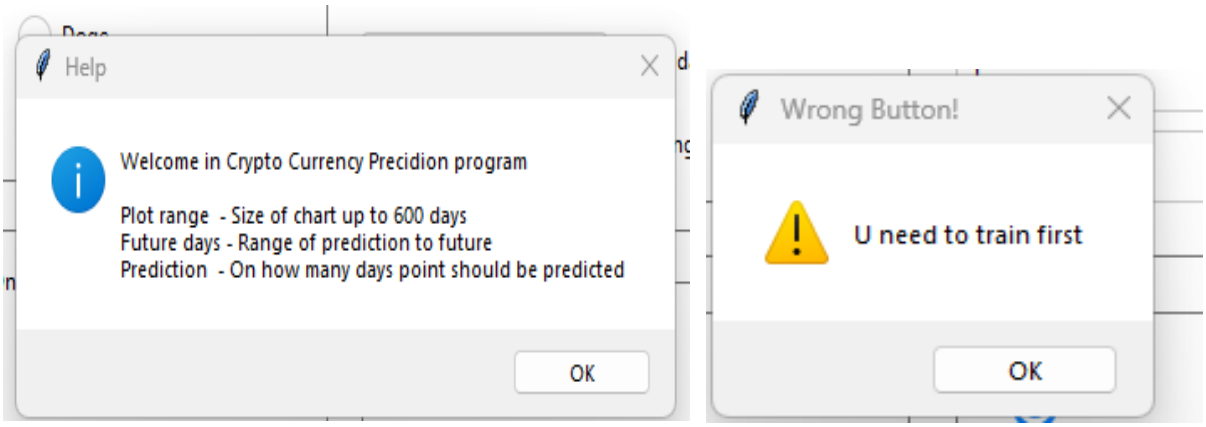
3.2 GUI

The GUI is simple, it was created using the tkinter module. It consists of 3 frames along with radio buttons inside. And some buttons outside the frames.



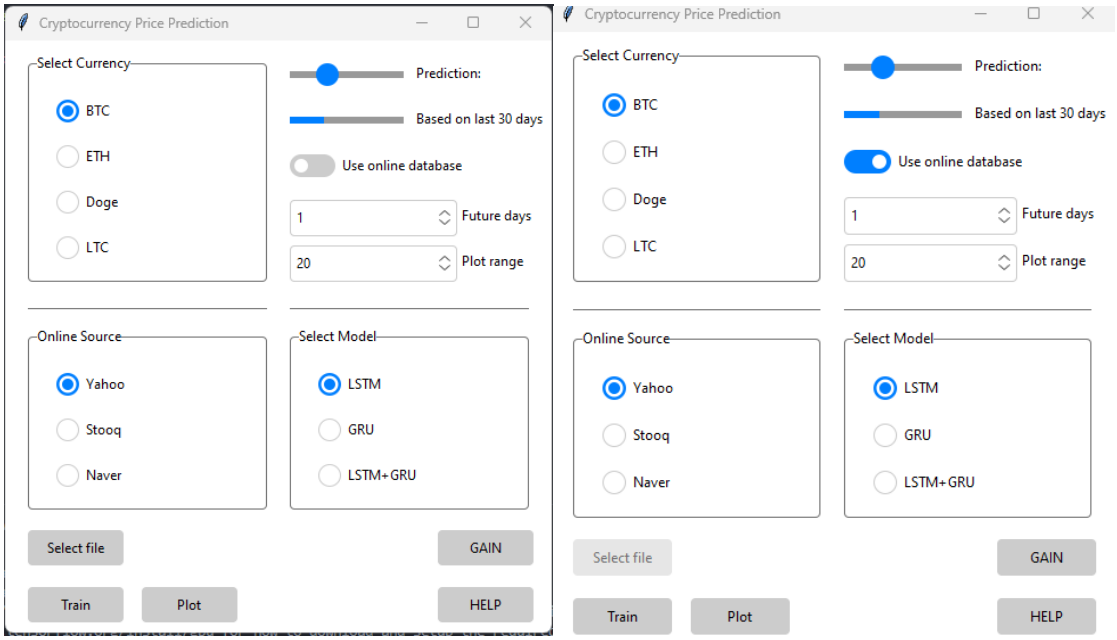
In the upper right corner, the user can select prediction, future day, and range of chart, and decide whether to use the online database.

In addition, the select file button allows you to load a file, the Train, Plot, Gain buttons do exactly what the function of the same name does.



The help button displays help in a new messagebox window with tkinter.

Plot button will not work until first training is done.



Switch for online database disables select file from pc.

4. Experiments

For experiments I was testing using: LSTM, GRU and LSTM-GRU

Most of the charts have range of 200+ days.

Each future point was predicted based on previous 50 days.

4.1 10 Epochs, 2 Future days. From up: GRU vs LSTM vs MIX

Fig1

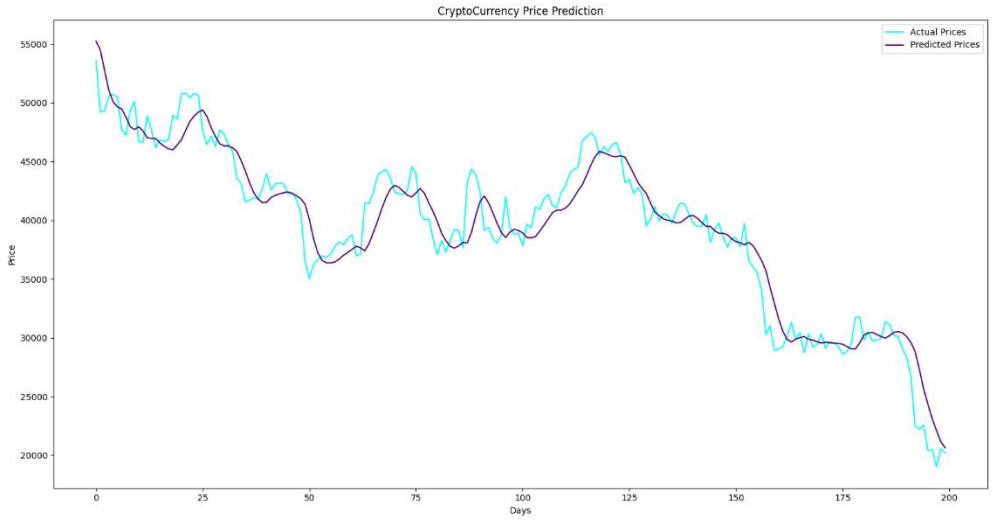


Fig2

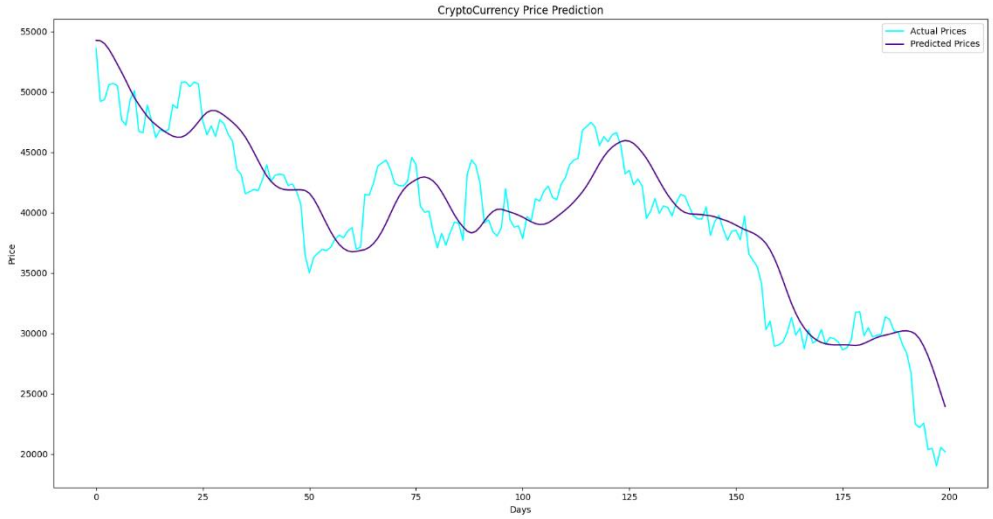
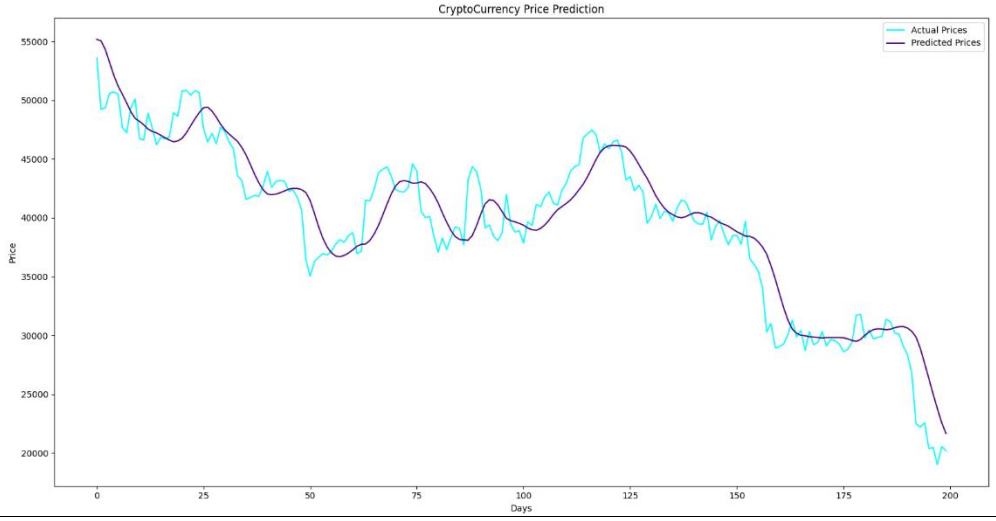


Fig3



We can see that the LSTM model (Fig2) is smoother. But GRU(fig1) more realistic.

4.2 25 Epochs, 1 Future days. From up: GRU vs LSTM vs MIX

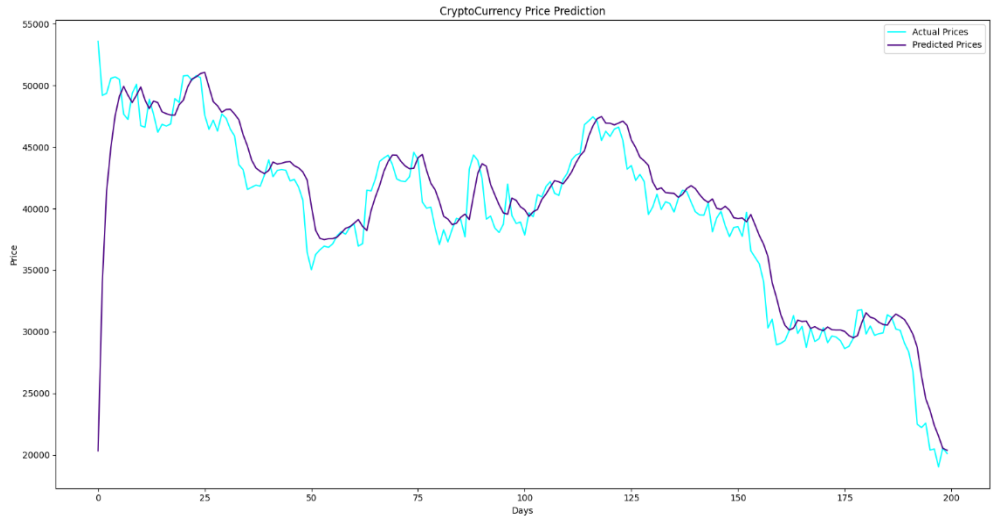


Fig4

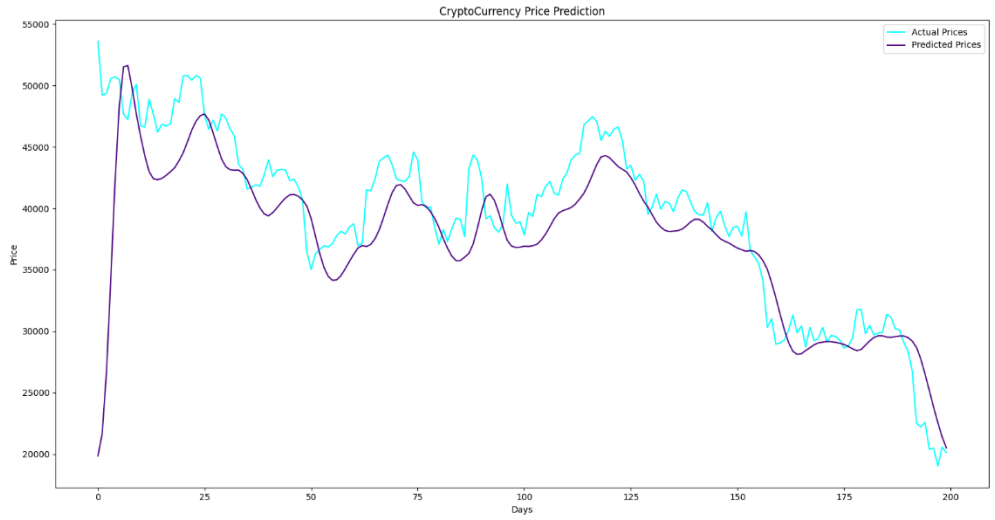


Fig5

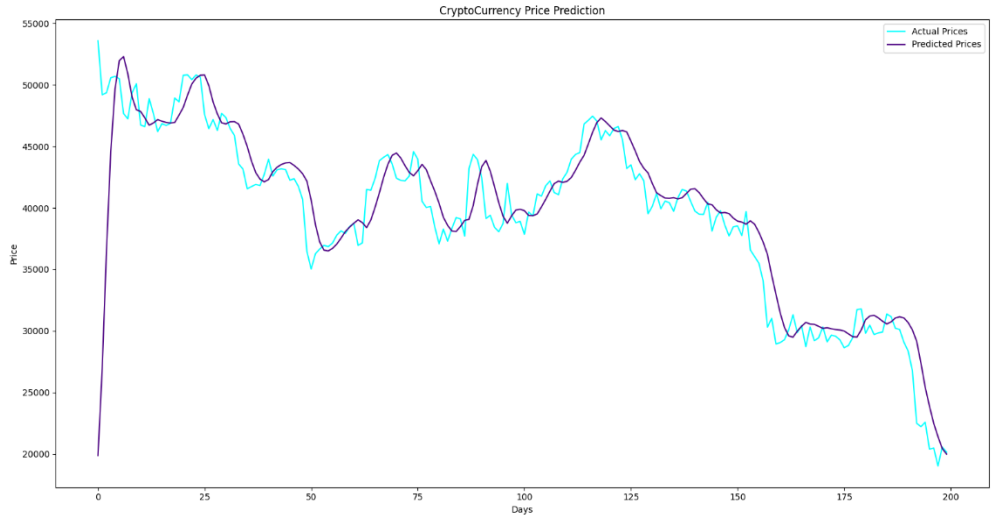


Fig6

Similar to the previous case. LSTM less smoothed out by more epochs. GRU still better.

4.3 25 Epochs, 5 Future days. From up: GRU vs LSTM vs MIX

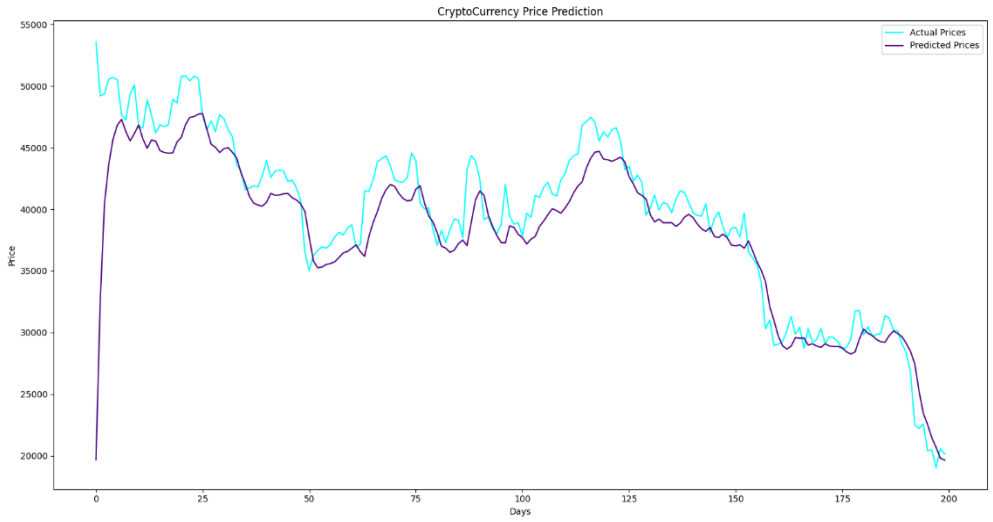


Fig7

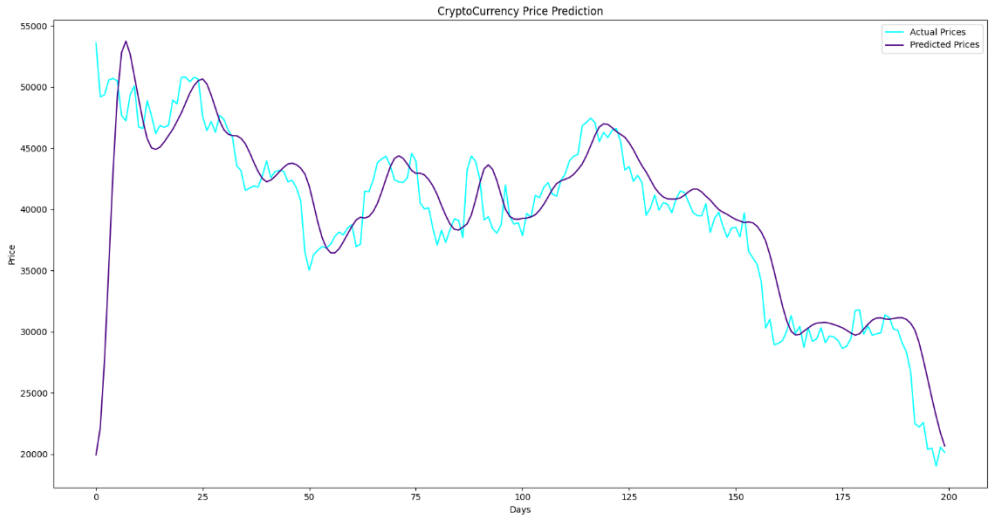


Fig8

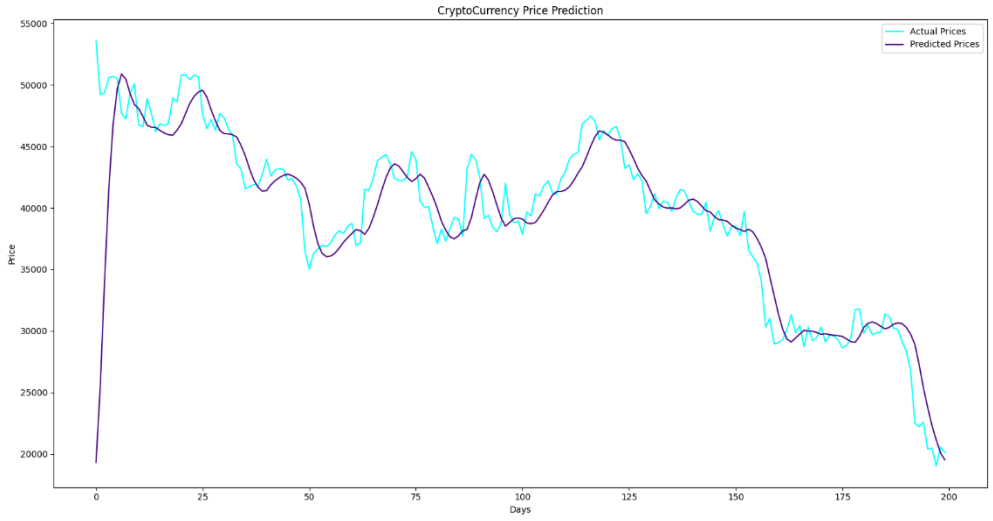


Fig9

Increasing future days will decrease prediction accuracy (especially fig7), and increase smoothing. MIX (fig9) seems to be the best.

4.3 25 Epochs, GRU, 1 vs 5 Future days

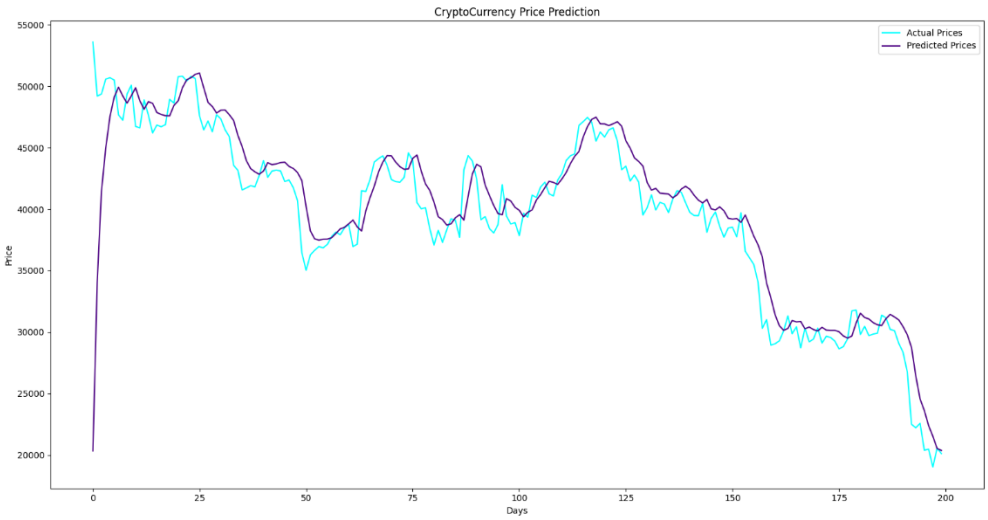


Fig4

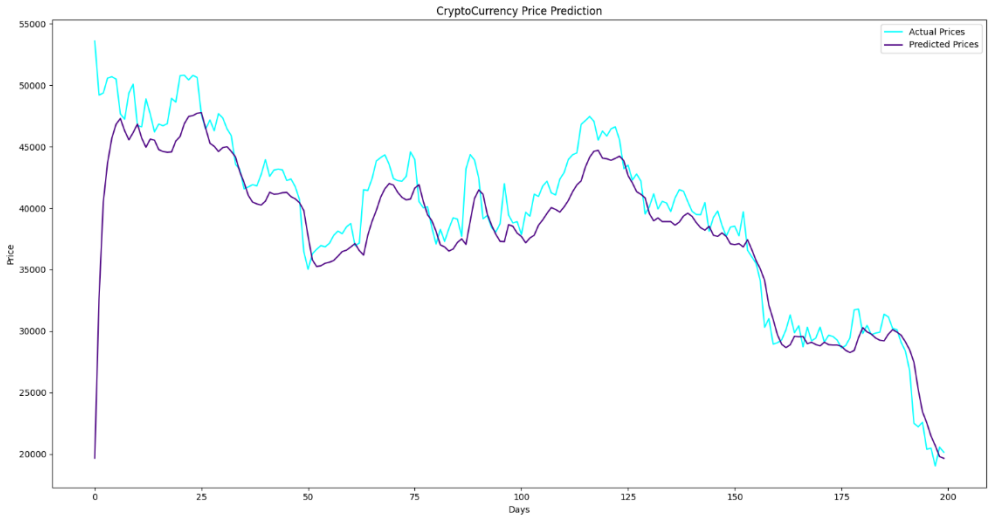


Fig7

Here you can see very well that when increasing the prediction even by a small number of days smooths(fig7) the graph, I will decrease the prediction accuracy, the graph resembles an approximation of a function.

We can also see that the graph is moving kind of downward

4.4 25 Epochs, LSTM, 1 vs 5 Future days

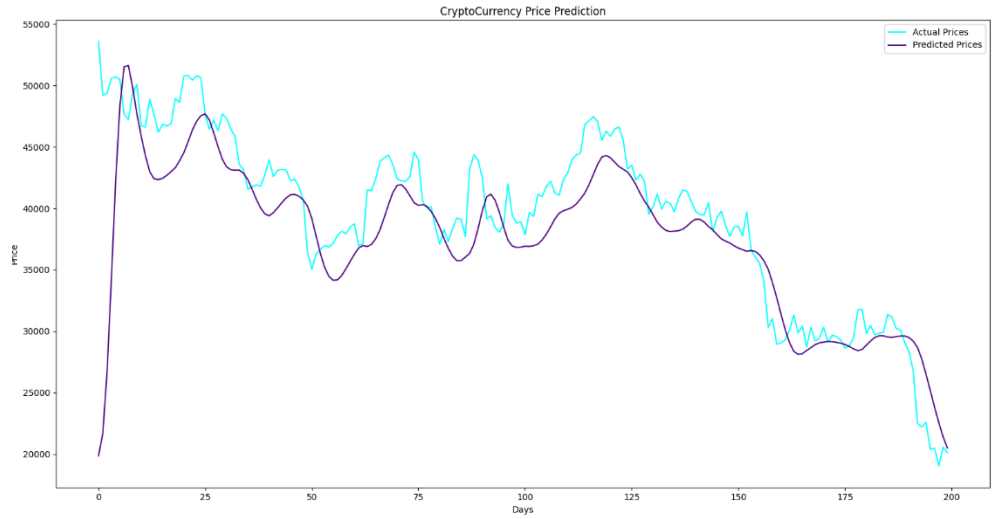


Fig5

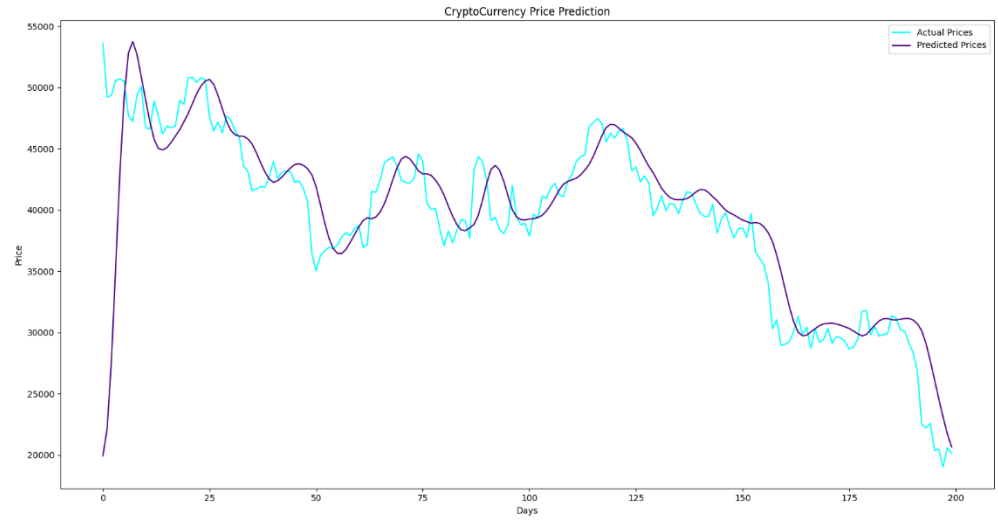


Fig8

Here, on the other hand, you can see that the chart is moving up with an increase in future day.

4.5 GRU, 10 vs 25 vs 100 epochs

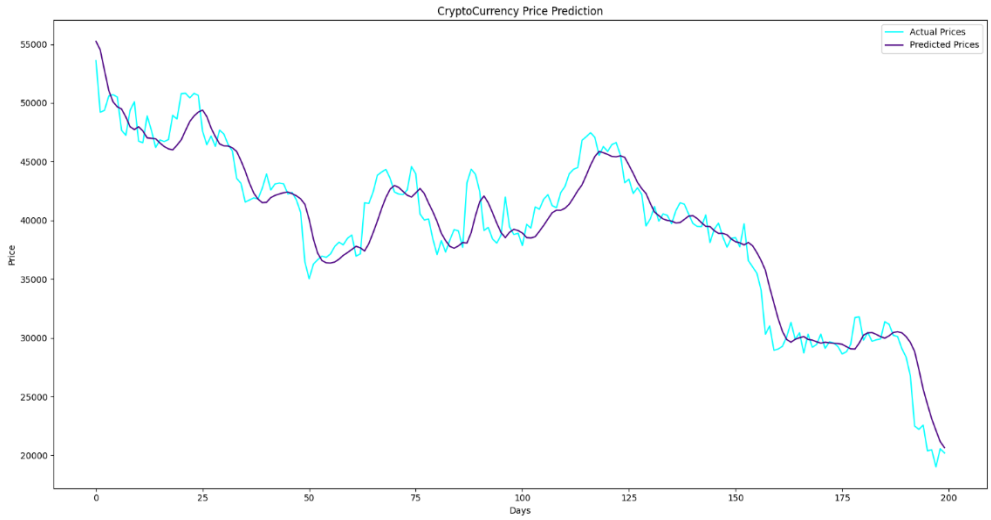


Fig10

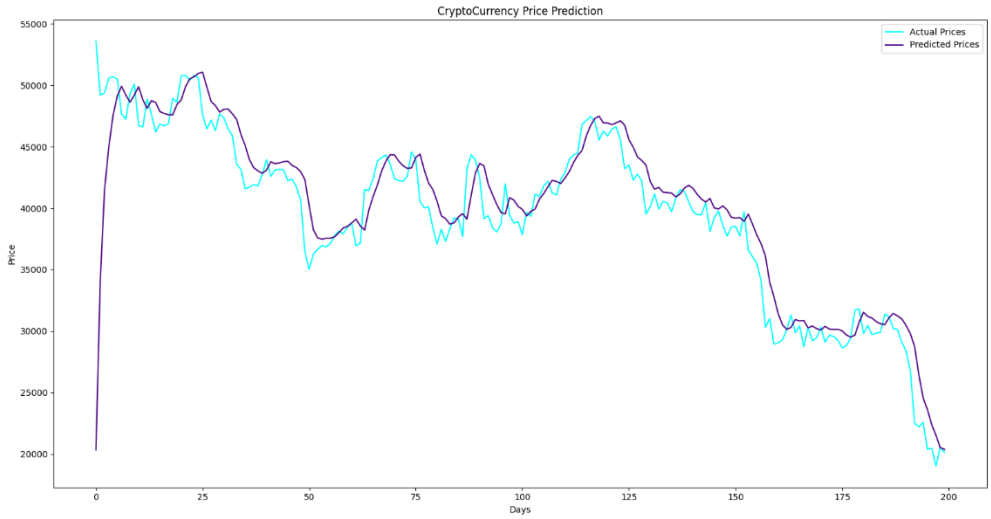


Fig11

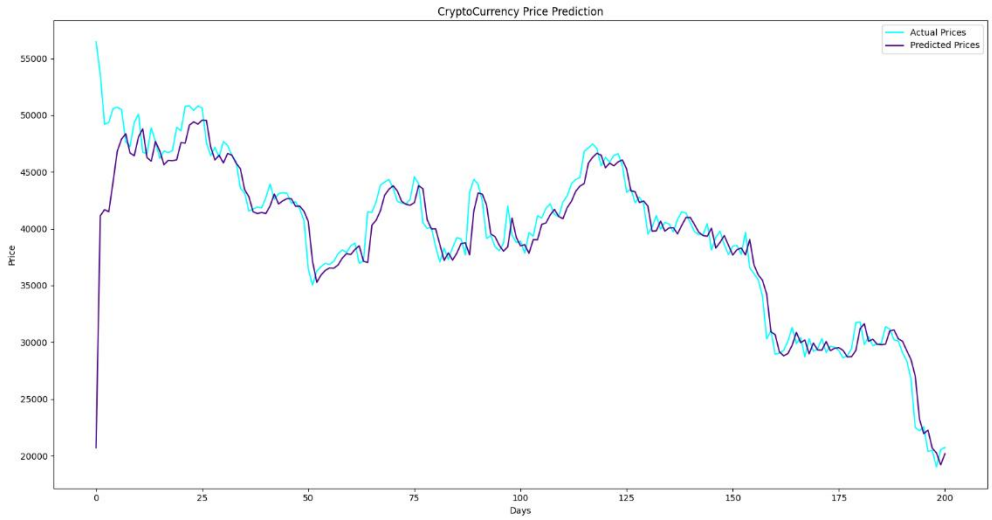


Fig12

The more epochs the more accurate prediction, so closer to actual price.

4.6 LSTM, 10 vs 25 vs 100 epochs

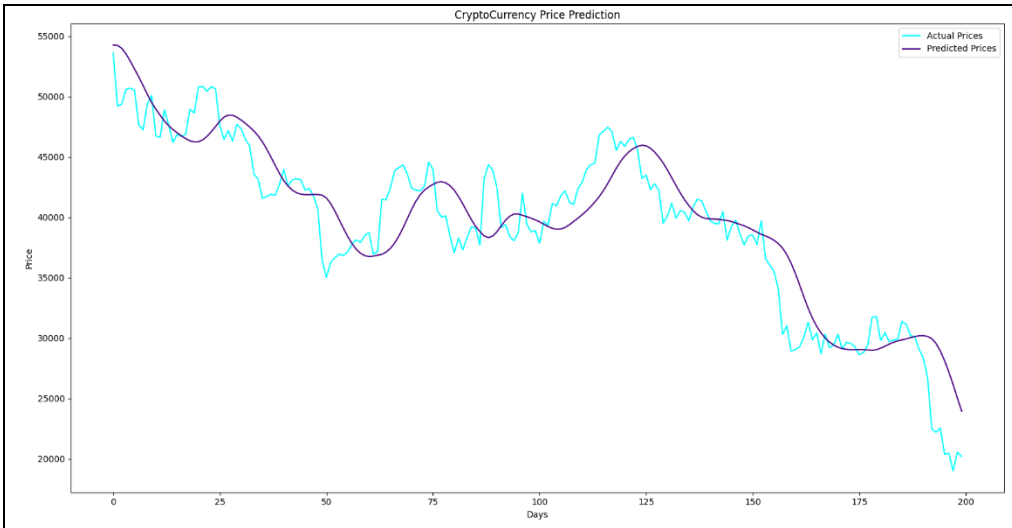


Fig13

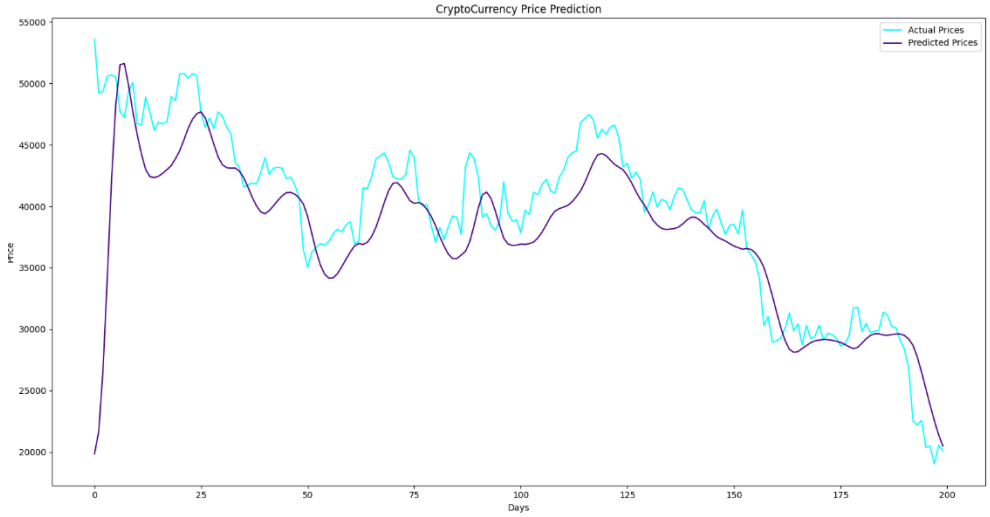


Fig14

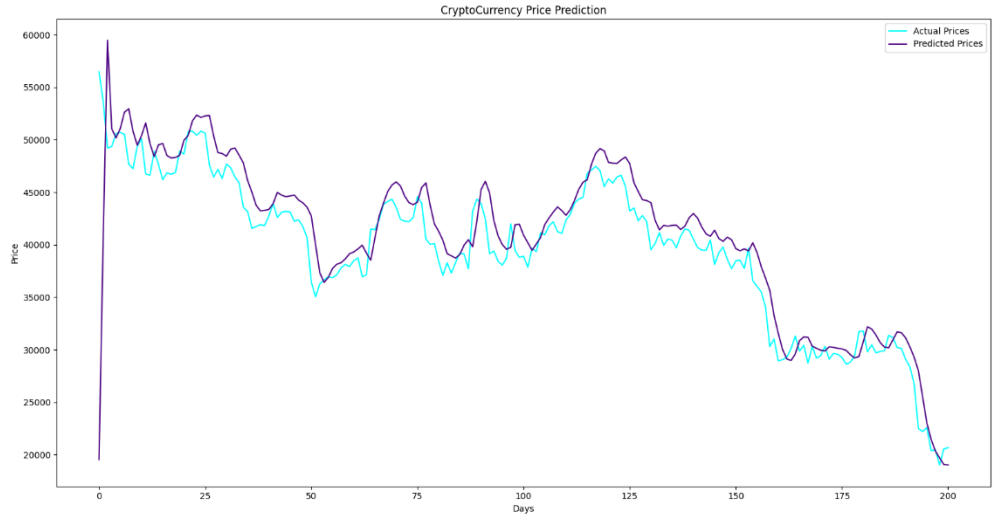


Fig15

We can observe that LSTM with a small number of epochs is worse than GRU, it needs about 100 to start giving results that do not resemble the approximation, being close to the real graph.

4.7 MIX, 10 vs 25 vs 100 epochs (for 2, 5 and 5 future days)

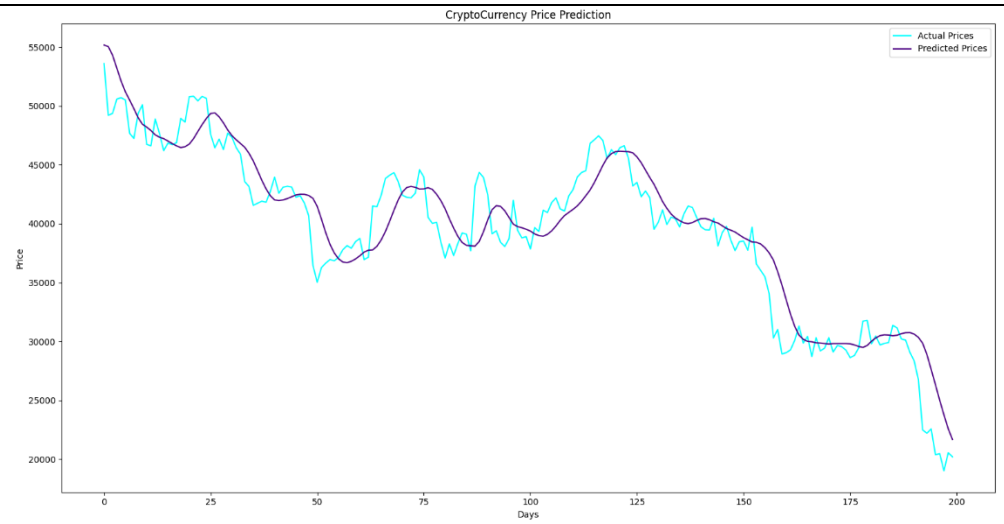


Fig16

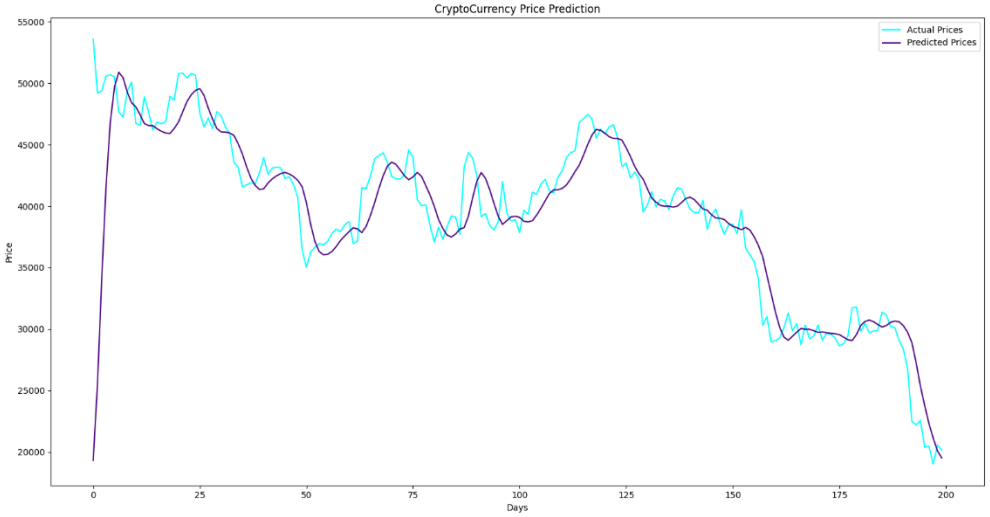


Fig17

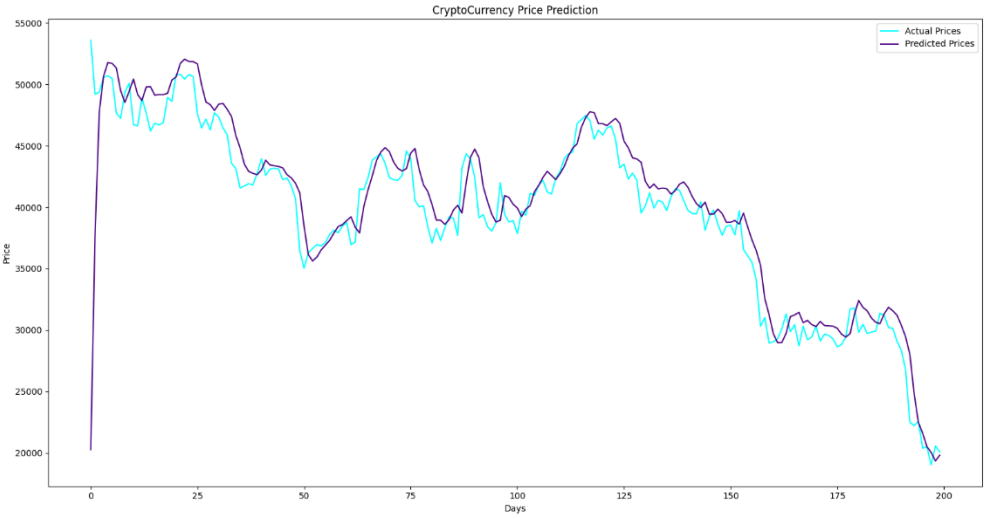


Fig18

We see that the higher the number of epochs, the more aggressive the chart is, i.e. it changes values faster to follow the market. Hence the higher precision.

4.8 GRU, 100 epochs, 5 future_day for: {10,30,50} prediction range.

On how many points (days) each future day will be calculated

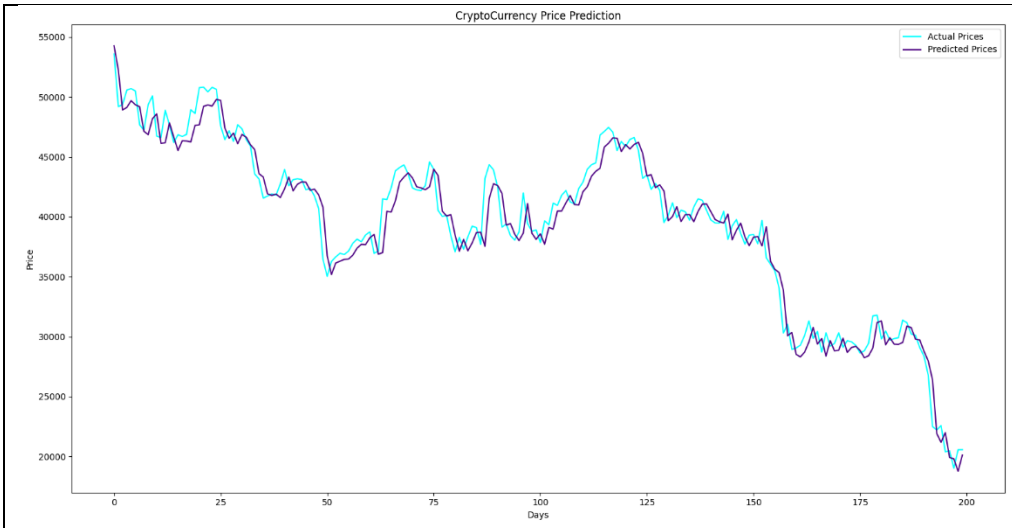


Fig19

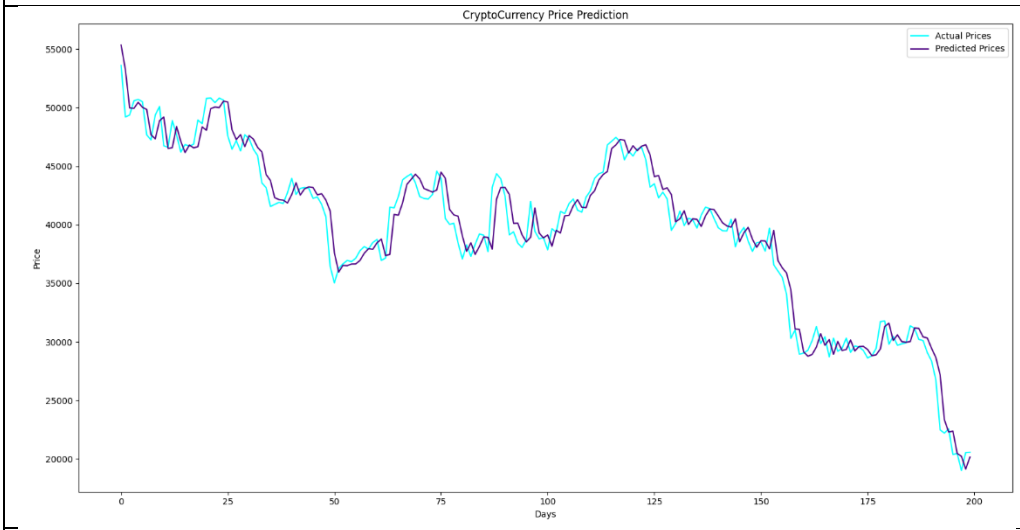


Fig20

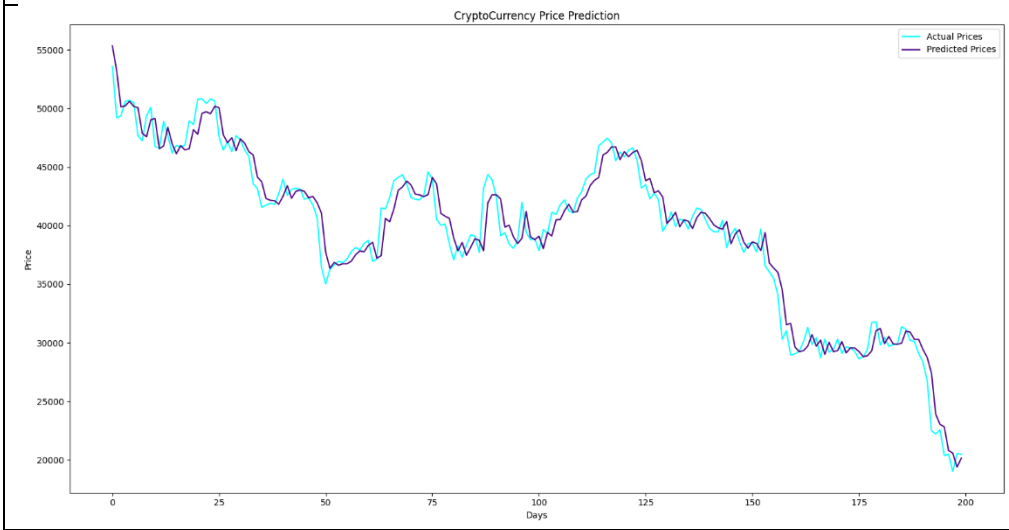
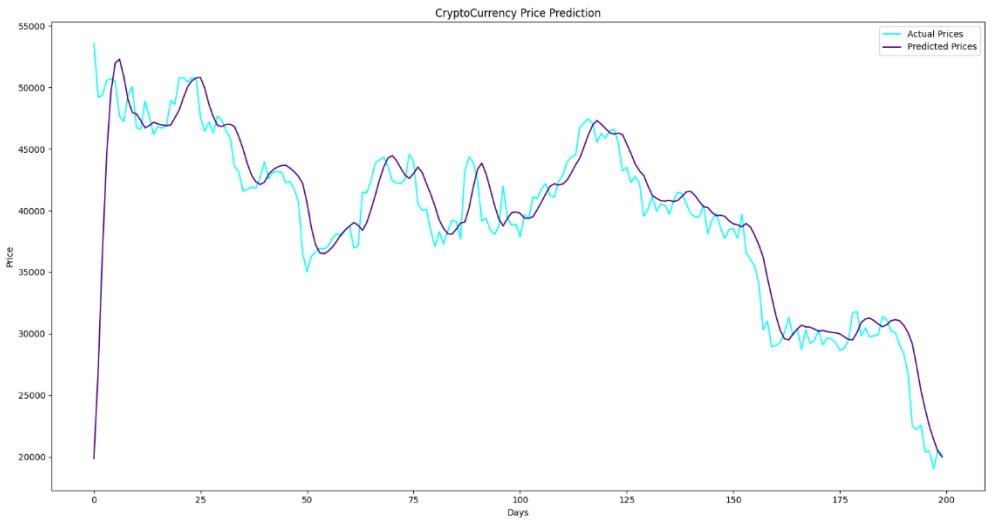
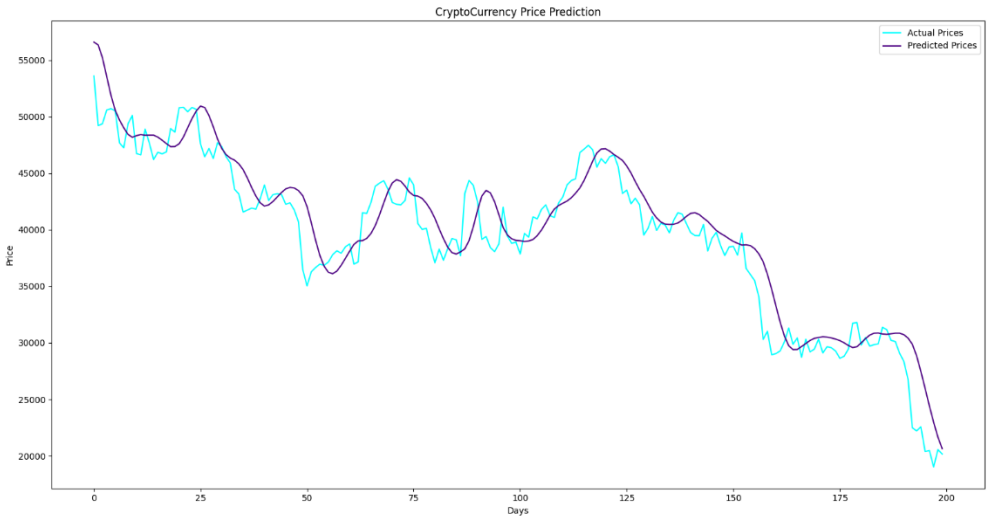


Fig21

We can see that increasing the range of days on the basis of which the future price is calculated has a positive effect on the precision.

4.9 Problem of starting value






```
        array_supp2 = total_dataset[len(total_dataset) - len(test_data)-
self.prediction_days- len(test_data):len(total_dataset) - len(test_data)-
len(test_data)]
        array_supp2 = np.array(array_supp2)

        model_inputs_temp= model_inputs[self.prediction_days:]
        array_supp2 = np.concatenate([array_supp2, model_inputs_temp])

        model_inputs = array_supp2
```

I generally had trouble with some of the charts.

The prediction chart was starting from the minimum or maximum value of the chart on days 1-5. I managed to solve this by adding an support array.

5. Summary

Overall, I found the project to be successful. The cryptocurrency market often has sudden ups and downs which affects sniffing.

I tested various models, they have their advantages and disadvantages, increasing the prediction range affects the decrease in its accuracy, which is logical. Increasing the range of days has a positive effect on the prediction.

Conclusions:

- The larger the future_day the lower the accuracy.
- The larger the number of epochs the higher the accuracy.
- When the number of epochs is small, the LSTM prediction graph performs worst and the GRU prediction graph performs best.
- The future_day parameter makes the GRU prediction graph have smaller values and the LSTM prediction graph have larger values.
- Increasing the epoch makes the charts less smoothed, i.e., they are more accurate, reacting faster to rate changes.
- In the LSTM-GRU version, increasing the future_days makes the prediction graph slightly lower than the actual price, but not as much as in GRU.

6. References

Every Cryptocurrency Daily Market Price

Deep neural networks for cryptocurrencies price prediction

Introduction to Machine Learning with Python: A Guide for Data

Python Deep Learning: Exploring deep learning techniques and neural network architectures

LSTM Vs GRU in Recurrent Neural Network

RNN vs GRU vs LSTM

7. Github