A. Data Understanding

Membaca dataset yang sudah diupload ke google drive

```
#menghubungkan ke google
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Dapat dilihat dataset memiliki 2923 baris dan 7 kolom yang dimana atribut tersebut adalah Date, Open, High, Low, Close, Adj Close, dan Volume. Data yang digunakan diambil melalui Yahoo finance, dataset bitcoin yang diambil dari tahun 2014 sampai October 2022.

```
import pandas as pd
import numpy as np

df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/dataset/BTC-USD-Oct.csv')
df = df.sort_values('Date').reset_index(drop=True)
df.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2014-10-12	362.605988	379.433014	356.144012	378.549011	378.549011	17552800
1	2014-10-13	377.920990	397.226013	368.897003	390.414001	390.414001	35221400
2	2014-10-14	391.691986	411.697998	391.324005	400.869995	400.869995	38491500
3	2014-10-15	400.954987	402.226990	388.765991	394.773010	394.773010	25267100
4	2014-10-16	394.518005	398.807007	373.070007	382.556000	382.556000	26990000

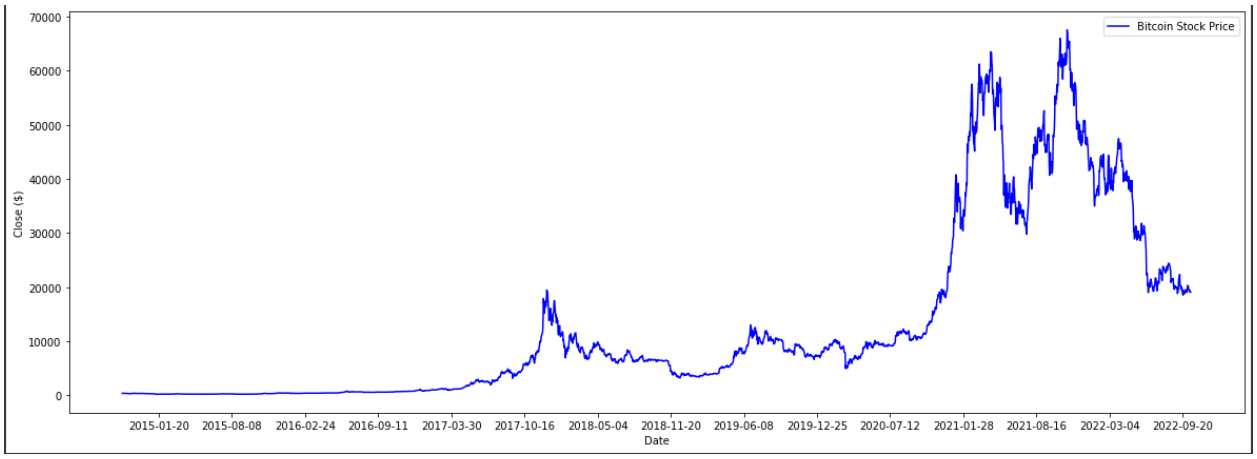
```
[ ] df.shape
```

```
(2923, 7)
```

B. Visualisasi Data

Visualisasi data adalah proses menggunakan elemen visual seperti diagram, grafik, atau peta untuk merepresentasikan data. Visualisasi data menerjemahkan yang kompleks, bervolume tinggi, atau numerik menjadi representasi visual yang lebih mudah diproses. Berikut merupakan contoh code untuk visualisasi dataset harga bitcoin dengan menggunakan atribut ‘Close’.

```
import matplotlib.pyplot as plt
from datetime import datetime

df['Close'] = df['Close'].astype(float)

plt.figure(figsize=(20,7))
plt.plot(df['Date'].values, df['Close'].values, label = 'Bitcoin Stock Price', color = 'blue')
plt.xticks(np.arange(100,df.shape[0],200))
plt.xlabel('Date')
plt.ylabel('Close ($)')
plt.legend()
plt.show()
```



C. Data Preprocessing

Data preprocessing adalah proses yang mengubah data mentah ke dalam bentuk yang lebih mudah dipahami. Proses ini penting dilakukan karena data mentah sering kali tidak memiliki format yang teratur.

Membagi data menjadi data latih dan data test, langkah selanjutnya yaitu melakukan normalisasi data yang ada. Normalisasi yang digunakan dalam proyek ini adalah normalisasi Min-Max. Itu Metode normalisasi data berikutnya adalah Min-Max. Cara kerjanya, masing-masing nilai dalam sebuah fitur dikurangi dengan nilai minimum fitur, kemudian dibagi dengan range nilai atau nilai maksimum dikurangi nilai minimum fitur. Min-Maks normalisasi akan menghasilkan nilai baru dengan rentang hasil normalisasi dari 0 sampai 1.

Data Preprocessing

```
[ ]  #Mebagi data test dan data latih
     num_shape = 2000

     train = df.iloc[:num_shape, 1:2].values
     test = df.iloc[num_shape:, 1:2].values
```

```
[ ]  #Scaling feature
     from sklearn.preprocessing import MinMaxScaler

     sc = MinMaxScaler(feature_range = (0, 1))
     train_scaled = sc.fit_transform(train)
```

```
[ ]  X_train = []

     #Price on next day
     y_train = []

     window = 60

     for i in range(window, num_shape):
         X_train_ = np.reshape(train_scaled[i-window:i, 0], (window, 1))
         X_train.append(X_train_)
         y_train.append(train_scaled[i, 0])
     X_train = np.stack(X_train)
     y_train = np.stack(y_train)
```

D. Membuat Model

Setelah melakukan preprocessing data, langkah selanjutnya yaitu melakukan pemodelan. Berikut merupakan langkah untuk membuat model LSTM

Membuat Model

```
[ ]  from keras.models import Sequential
     from keras.layers import Dense, LSTM, Dropout, GRU
     from keras.layers import *
```

```
# Initializing the Recurrent Neural Network
model = Sequential()
#Adding the first LSTM layer with a sigmoid activation function and some Dropout regularization
#Units - dimensionality of the output space

model.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))
model.add(Dropout(0.2))

model.add(LSTM(units = 50, return_sequences = True))
model.add(Dropout(0.2))

model.add(LSTM(units = 50, return_sequences = True))
model.add(Dropout(0.2))

model.add(LSTM(units = 50))
model.add(Dropout(0.2))

# Adding the output layer
model.add(Dense(units = 1))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 50)            10400

 dropout (Dropout)           (None, 60, 50)            0
```

Setelah membuat sebuah model, kini saatnya melatih model yang telah diciptakan. Dalam proses ini, harus memperhatikan seberapa baik model melakukan generalisasi. Hal inilah yang biasa disebut overfitting dan underfitting, di mana keseimbangan dari optimalisasi dan generalisasi tidak seimbang.

```
[ ]  #melakukan training data
     from sklearn.metrics import mean_squared_error, mean_absolute_error
     from sklearn.model_selection import train_test_split

     model.compile(optimizer = 'adam', loss = 'mean_squared_error')
     model.fit(X_train, y_train, epochs = 100, batch_size = 128);
```

```
Epoch 1/100
16/16 [==============================] - 13s 238ms/step - loss: 0.0219
Epoch 2/100
16/16 [==============================] - 4s 235ms/step - loss: 0.0062
Epoch 3/100
16/16 [==============================] - 4s 231ms/step - loss: 0.0042
Epoch 4/100
16/16 [==============================] - 4s 234ms/step - loss: 0.0036
Epoch 5/100
16/16 [==============================] - 4s 234ms/step - loss: 0.0034
Epoch 6/100
16/16 [==============================] - 5s 316ms/step - loss: 0.0032
Epoch 7/100
16/16 [==============================] - 4s 229ms/step - loss: 0.0030
Epoch 8/100
16/16 [==============================] - 6s 360ms/step - loss: 0.0026
Epoch 9/100
16/16 [==============================] - 7s 429ms/step - loss: 0.0028
Epoch 10/100
16/16 [==============================] - 5s 339ms/step - loss: 0.0026
Epoch 11/100
16/16 [==============================] - 4s 230ms/step - loss: 0.0027
Epoch 12/100
```

Membuat model menggunakan algoritma GRU

```
# The GRU architecture
modelGRU = Sequential()

modelGRU.add(GRU(units=50, return_sequences=True, input_shape=(X_train.shape[1],1)))
modelGRU.add(Dropout(0.2))

modelGRU.add(GRU(units=50, return_sequences=True, input_shape=(X_train.shape[1],1)))
modelGRU.add(Dropout(0.2))

modelGRU.add(GRU(units=50, return_sequences=True, input_shape=(X_train.shape[1],1)))
modelGRU.add(Dropout(0.2))

modelGRU.add(GRU(units=50))
modelGRU.add(Dropout(0.2))

modelGRU.add(Dense(units=1))
modelGRU.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru_4 (GRU)                 (None, 60, 50)            7950

 dropout_4 (Dropout)         (None, 60, 50)            0

 gru_5 (GRU)                 (None, 60, 50)            15300

 dropout_5 (Dropout)         (None, 60, 50)            0

 gru_6 (GRU)                 (None, 60, 50)            15300

 dropout_6 (Dropout)         (None, 60, 50)            0

 gru_7 (GRU)                 (None, 50)                15300

 dropout_7 (Dropout)         (None, 50)                0

 dense_1 (Dense)             (None, 1)                 51
```

Melatih model GRU

```
[ ] modelGRU.compile(optimizer='adam', loss='mean_squared_error')
    modelGRU.fit(X_train, y_train, epochs=100, batch_size=128)

    Epoch 1/100
    16/16 [==============================] - 8s 124ms/step - loss: 0.0213
    Epoch 2/100
    16/16 [==============================] - 2s 123ms/step - loss: 0.0031
    Epoch 3/100
    16/16 [==============================] - 2s 121ms/step - loss: 0.0022
    Epoch 4/100
    16/16 [==============================] - 2s 122ms/step - loss: 0.0022
    Epoch 5/100
    16/16 [==============================] - 2s 124ms/step - loss: 0.0018
    Epoch 6/100
    16/16 [==============================] - 2s 122ms/step - loss: 0.0018
    Epoch 7/100
    16/16 [==============================] - 2s 121ms/step - loss: 0.0018
    Epoch 8/100
    16/16 [==============================] - 2s 120ms/step - loss: 0.0018
    Epoch 9/100
    16/16 [==============================] - 2s 121ms/step - loss: 0.0020
    Epoch 10/100
    16/16 [==============================] - 2s 122ms/step - loss: 0.0016
    Epoch 11/100
    16/16 [==============================] - 2s 122ms/step - loss: 0.0017
    Epoch 12/100
    16/16 [==============================] - 2s 121ms/step - loss: 0.0017
    Epoch 13/100
    16/16 [==============================] - 2s 121ms/step - loss: 0.0015
    Epoch 14/100
    16/16 [==============================] - 2s 122ms/step - loss: 0.0015
    Epoch 15/100
    16/16 [==============================] - 3s 191ms/step - loss: 0.0015
    Epoch 16/100
```

Menyimpan hasil testing ke sebuah variable

```
[ ] predict = modelGRU.predict(X_test)
    predict = sc.inverse_transform(predict)

    29/29 [==============================] - 1s 16ms/step
```

E. Evaluasi

Mean Squared Error (MSE) adalah rata-rata kesalahan kuadrat antara data aktual dan data prediksi. *Root Mean Squared Error* (RMSE) adalah akar kuadrat dari MSE, *Mean Absolute Error* (MAE) adalah rata-rata dari nilai absolut error.

Semakin kecil nilai MSE, RMSE, dan MAE, semakin baik pula performansi suatu model.

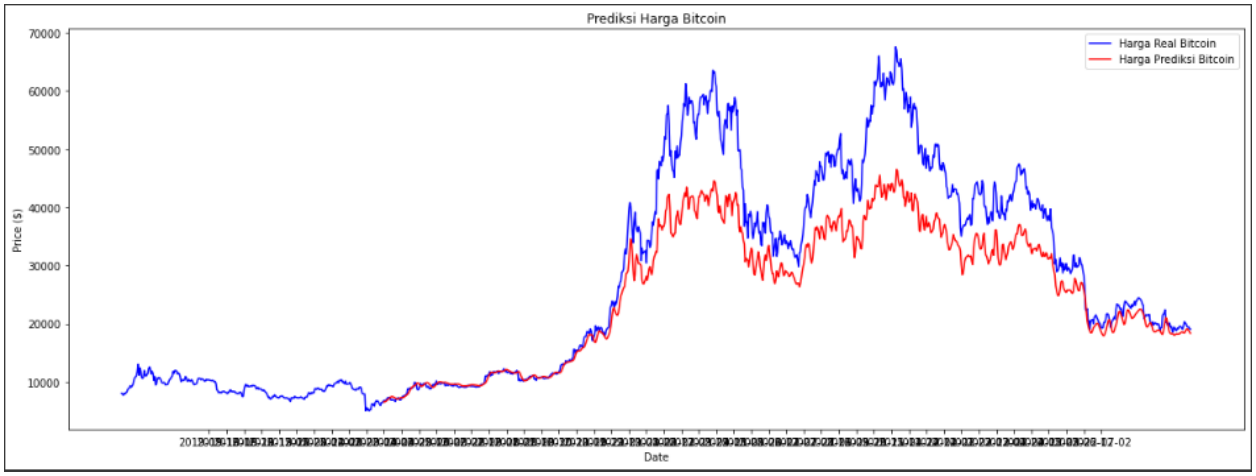
```
[ ] #mengevaluasi model
    diff = predict - test

    print("MSE:", np.mean(diff**2))
    print("MAE:", np.mean(abs(diff)))
    print("RMSE:", np.sqrt(np.mean(diff**2)))

    MSE: 69907506.65134256
    MAE: 6035.06067498517
    RMSE: 8361.070903379696
```

Evaluasi model GRU

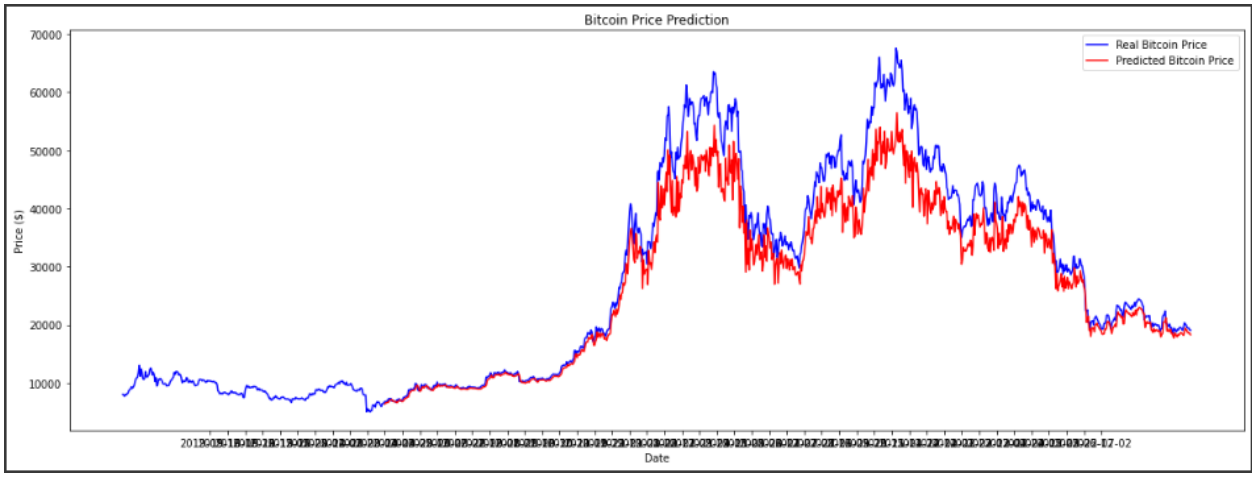
```
[ ]  diff = predict - test

     print("MSE:", np.mean(diff**2))
     print("MAE:", np.mean(abs(diff)))
     print("RMSE:", np.sqrt(np.mean(diff**2)))

     MSE: 19690920.54847911
     MAE: 3144.7506322631025
     RMSE: 4437.445272730595
```

F. Hasil Prediksi

Hasil prediksi model LSTM



Hasil prediksi menggunakan model GRU



Prediksi 10 hari ke depan menggunakan model GRU

Karena model GRU menurut penulis lebih baik dibandingkan dengan model LSTM, penulis ingin memprediksi harga bitcoin 10 hari kedepan. Berikut code yang digunakan

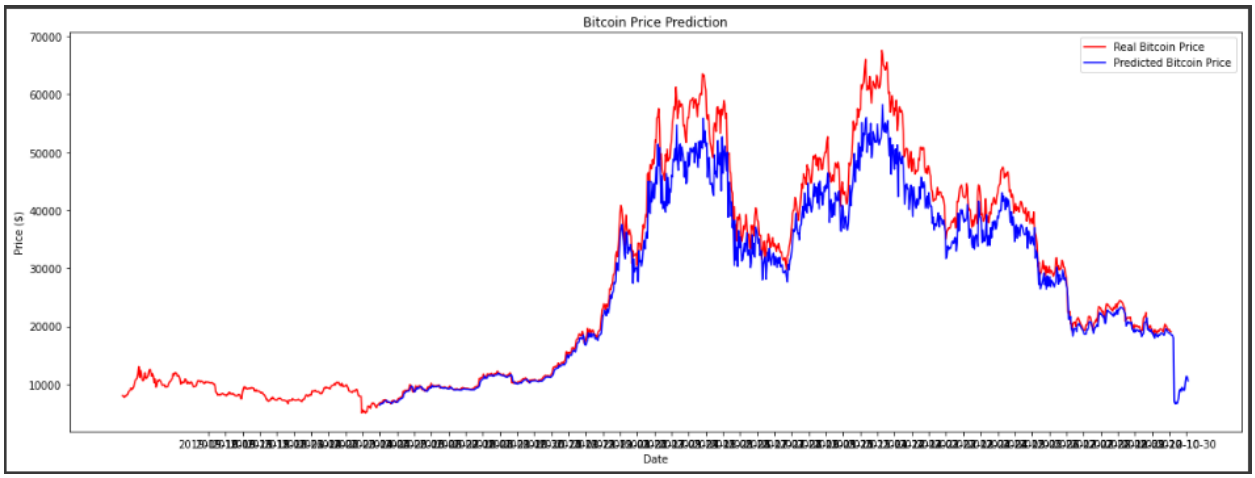
```
pred_ = predict[-1].copy()
prediction_full = []
window = 60
df_copy = df.iloc[:, 1:2][1:].values

for j in range(20):
    df_ = np.vstack((df_copy, pred_))
    train_ = df_[:num_shape]
    test_ = df_[num_shape:]

    df_volume_ = np.vstack((train_, test_))

    inputs_ = df_volume_[df_volume_.shape[0] - test_.shape[0] - window:]
    inputs_ = inputs_.reshape(-1,1)
    inputs_ = sc.transform(inputs_)

    X_test_2 = []

    for k in range(window, num_2):
        X_test_3 = np.reshape(inputs_[k-window:k, 0], (window, 1))
        X_test_2.append(X_test_3)

    X_test_ = np.stack(X_test_2)
    predict_ = modelGRU.predict(X_test_)
    pred_ = sc.inverse_transform(predict_)
    prediction_full.append(pred_[-1][0])
    df_copy = df_[j:]
```

```
29/29 [==============================] - 1s 29ms/step
29/29 [==============================] - 0s 15ms/step
29/29 [==============================] - 0s 16ms/step
29/29 [==============================] - 0s 16ms/step
29/29 [==============================] - 0s 16ms/step
29/29 [==============================] - 0s 16ms/step
29/29 [==============================] - 0s 16ms/step
29/29 [==============================] - 0s 16ms/step
```

```
[ ]  prediction_full_new = np.vstack((predict, np.array(prediction_full).reshape(-1,1)))

     df_date = df[['Date']]

     for h in range(20):
         kk = pd.to_datetime(df_date['Date'].iloc[-1]) + pd.DateOffset(days=1)
         kk = pd.DataFrame([kk.strftime("%Y-%m-%d")], columns=['Date'])
         df_date = df_date.append(kk)
     df_date = df_date.reset_index(drop=True)
```

Hasil prediksi 10 hari ke depan

Pada grafik di bawah ditunjukan bahwa harga bitcoin pada 10 hari kedepan diprediksi akan mengalami penurunan yang sangat signifikan. Kemungkinan ada kesalahan pada prediksi algoritma dalam meprediksi data. Sehingga penulis perlu mengkaji ulang untuk membuat algoritma yang lebih baik kedepannya.



G. Kesimpulan

Berdasarkan evaluasi yang didapatkan model dari algoritma GRU lebih kecil nilai MSE, RMSE, dan MAE dibandingkan model LSTM sehingga algoritma GRU lebih baik daripada algoritma LSTM.