BAB III
METODE PENELITIAN

Agar dalam penyusunan penelitian diperoleh hasil yang baik, diperlukan suatu metode penelitian yang sesuai dengan permasalahan. Pada bagian ini akan dibahas mengenai metode, variabel, data, dan prosedur penelitian yang digunakan dalam penelitian ini.

3.1. Metode Penelitian

Model peramalan yang digunakan dalam penelitian ini adalah model *Facebook Prophet*. *Facebook Prophet* adalah model peramalan untuk data runtun waktu yang terdiri dari fungsi musiman, fungsi tren, fungsi hari libur, dan fungsi regresor tambahan. Selain itu, model *Prophet* juga akan digunakan untuk melakukan dekomposisi data runtun waktu sehingga data runtun waktu yang akan diramalkan dapat dipahami dengan baik.

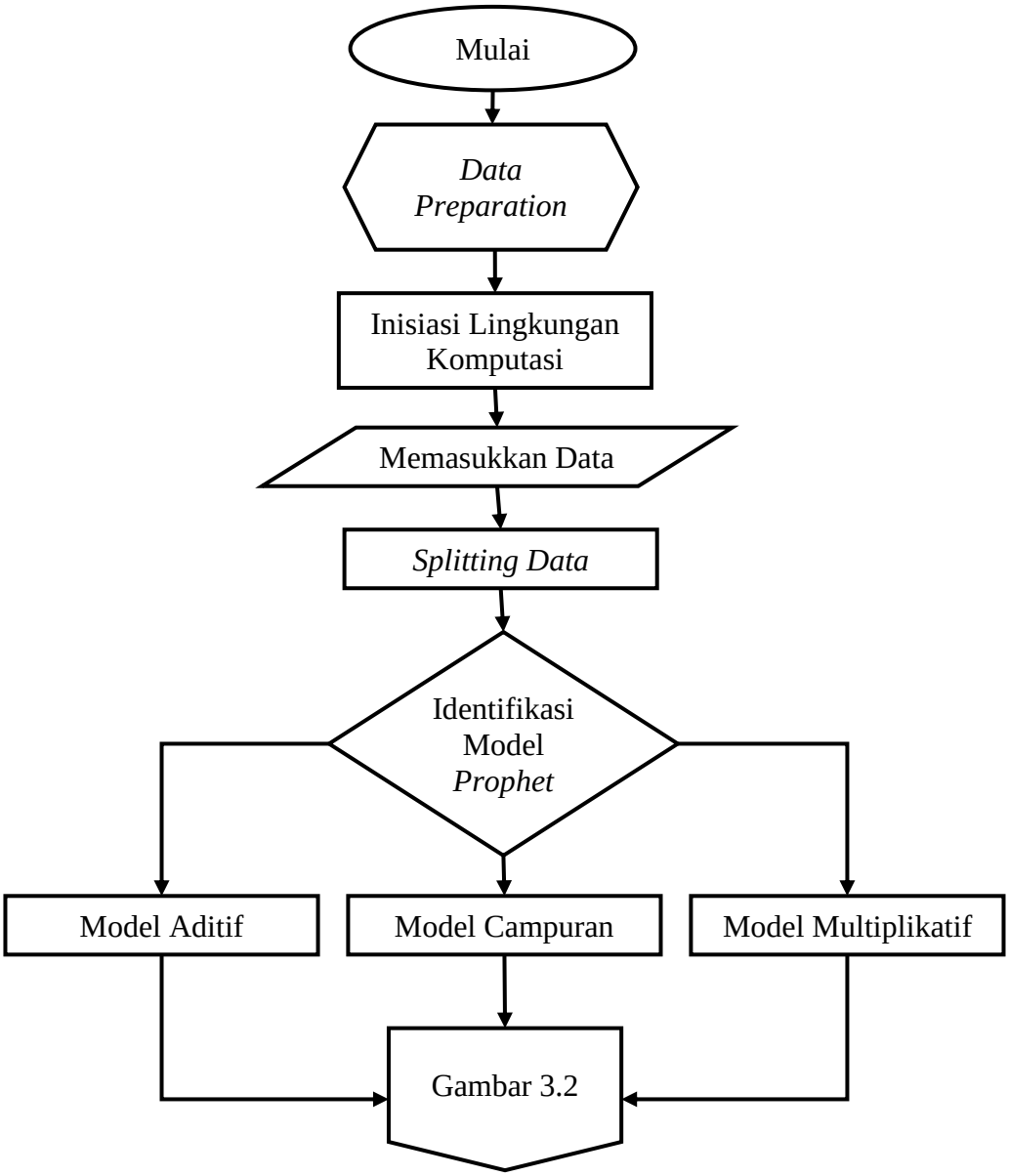
3.2. Jenis dan Sumber Data

Data yang akan digunakan pada penelitian ini adalah data sekunder berupa harga tukar suatu mata uang kripto, yaitu Solana yang diperoleh dari situs bernama CoinMarketCap yang beralamatkan di https://coinmarketcap.com/id/. Data yang tersedia merupakan harga Solana dengan interval 5 menit dari tanggal 27 Februari 2023 sampai dengan 4 April 2023.

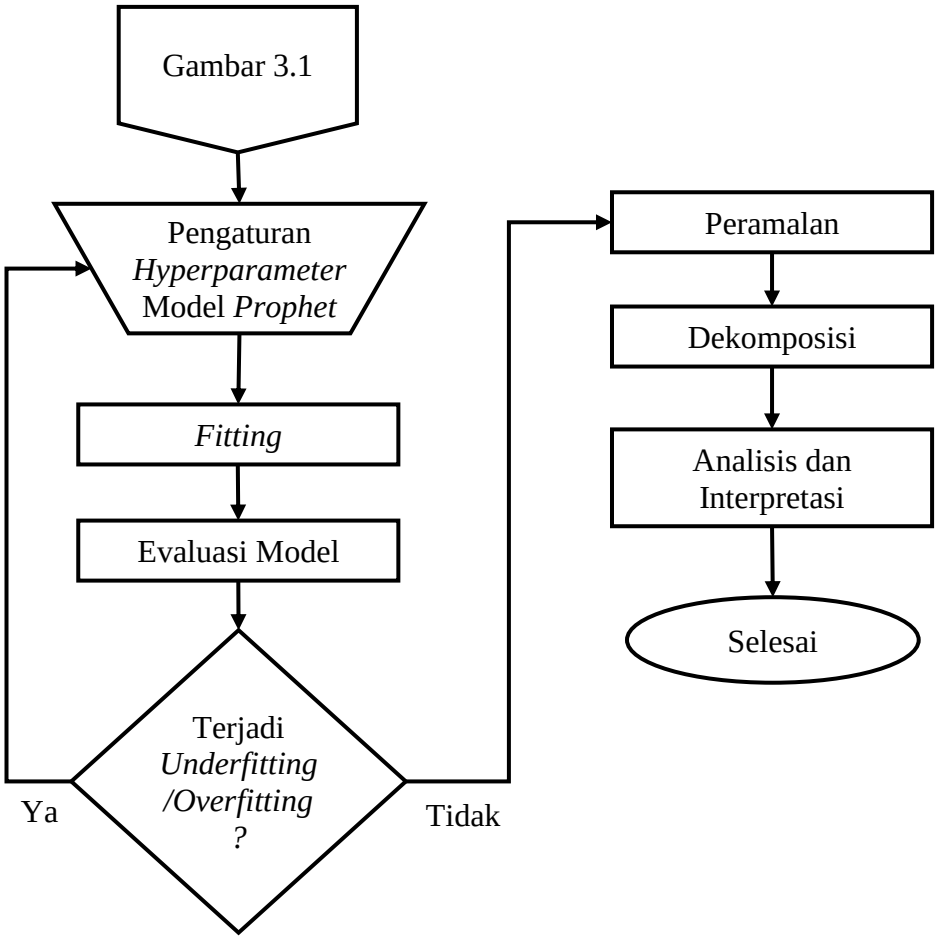
Data yang diperoleh akan dibagi menjadi dua, yaitu data latih dan data uji. Data latih berfungsi untuk membangun model peramalan yang akan digunakan sedangkan data uji digunakan untuk mengukur akurasi dari model peramalan yang akan digunakan. Sebanyak 90% data akan digunakan sebagai data latih dan sebanyak 10% data akan digunakan sebagai data latih.

3.3. Prosedur Penelitian

Dalam melakukan pemodelan untuk model *Prophet*, akan digunakan Microsoft Excel 2019 untuk menyiapkan data dan *library prophet* pada bahasa pemrograman Python 3.11.0 dengan JupyterLab 3.4.4 sebagai lingkungan untuk melakukan komputasi. Berikut adalah rancangan penelitian yang akan dilakukan pada penelitian ini:



Gambar 3. 1 Diagram Alir Penelitian Bagian 1.



Gambar 3. 2 Diagram Alir Penelitian Bagian 2.

3.3.1. *Data Preparation*

Pada tahap ini dilakukan pengambilan data harga mata uang kripto Solana dalam Rupiah dengan interval 5 menit dari situs https://coinmarketcap.com/id/. Selanjutnya data yang diperoleh akan disesuaikan formatnya menjadi format yang diperlukan oleh model *Prophet* dengan menggunakan Microsoft Excel. Format data yang diperlukan oleh model *Prophet* di antaranya sebagai berikut:

1) Nama kolom dari kolom waktu harus diubah menjadi “ds” dan nama kolom dari kolom nilai harus diubah menjadi “y”.
2) Penulisan waktu harus dalam format YYYY-MM-DD atau YYYY-MM-DD HH:MM:SS.

3.3.2. Inisiasi Lingkungan Komputasi

Pada tahap ini dilakukan pengimporan fungsi yang diperlukan pada JupyterLab. *Library* yang perlu diimpor antara lain *pandas* untuk pembuatan dan

pengeditan *dataframe*, *matplotlib.pyplot* untuk membuat grafik, dan *prophet* untuk mengakses model *Prophet*. *Library* dapat diimpor pada JupyterLab dengan mengetikkan kode berikut pada kolom di JupyterLab:

```
import pandas as pd
import matplotlib.pyplot as plt
from prophet import Prophet
```

3.3.3. Memasukkan Data dan *Splitting Data*

Pada tahap ini akan dilakukan pemasukan data ke lingkungan komputasi dengan menggunakan *library* pandas. Untuk memasukkan data ke dalam lingkungan komputasi lalu memasukkannya ke suatu variabel dapat dilakukan dengan mengetikkan kode berikut:

```
data = pd.read_excel(open('Data/Harga_SOL_Skrip.xlsx', 'rb'),
sheet_name='20230227_20230404')
```

Lalu, tahapan selanjutnya adalah memisahkan data menjadi data latih dan data uji. Data dipisahkan dengan perbandingan 90% data latih dan 10% data uji lalu dimasukkan ke dalam variabel data_latih dan data_uji. Untuk memisahkan data tersebut, dapat dilakukan dengan mengetikkan kode berikut:

```
i = int(round(0.9*len(data),0)) #Banyak data latih
data_latih = data[:i]
data_uji = data[i:]
data_uji.reset_index(inplace=True)
del data_uji['index']
```

3.3.4. Identifikasi Model *Prophet*

Pada tahap ini akan dilakukan pembuatan plot dari data latih untuk melihat bentuk dari data yang akan dimodelkan. Plot data dapat dibuat dengan *library* matplotlib.pyplot atau Microsoft Excel. Setelah itu, dilakukan identifikasi model *Prophet* yang mungkin cocok dengan data latih yang akn digunakan untuk membangun model. Untuk menginisiasi model *Prophet* dan memasukkannya ke dalam sebuah variabel dapat dilakukan dengan menggunakan kode berikut:

```
model = Prophet(growth='linear', n_changepoints=25,

yearly_seasonality=False,weekly_seasonality=True,
                daily_seasonality=True,
                seasonality_mode='additive',
                seasonality_prior_scale=10,
                changepoint_prior_scale=0.05)
```

Keterangan:

- Argumen "*growth*" dapat diisi dengan *'flat'*, *'linear'*, atau *'logistic'*. Model *Prophet* secara *default* akan menggunakan fungsi linear untuk memodelkan.
- Argumen "*n_changepoints*" adalah banyak potensi titik ubah yang terdapat pada data untuk mengubah fungsi tren menjadi fungsi bertingkat. Model *Prophet* secara *default* akan menggunakan 25 potensi titik ubah untuk memodelkan.
- Argumen *"yearly_seasonality", "weekly_seasonality", "daily_seasonality"* dapat diisi dengan True, False, atau angka untuk mengatur ordo deret Fourier. Model *Prophet* akan secara otomatis mendeteksi efek musiman mana yang diperlukan dan secara *default* menyetel ordo Fourier manjadi 10 untuk efek musiman tahunan, 3 untuk efek musiman mingguan, dan 4 untuk efek musiman harian.
- Argumen *"seasonality_mode"* dapat diisi dengan *'additive'* atau *'multiplicative'*.
- Argumen "*seasonality_prior_scale"* adalah ukuran kekuatan dari fungsi musiman. Jika nilai ini semakin besar, maka fungsi akan dapat mengaproksimasi fluktuasi yang lebih besar. Model *Prophet* secara *default* menggunakan nilai 10 untuk argumen ini.
- Argumen "*changepoint_prior_scale"* adalah ukuran untuk menentukan fleksibilitas model *Prophet* dalam menentukan titik ubah. Jika nilai ini semakin besar, maka model *Prophet* akan memilih lebih banyak potensi titik ubah. Model *Prophet* secara *default* menggunakan nilai 0,05 untuk argumen ini.

Untuk menginisiasi model campuran atau menambahkan efek musiman selain tahunan, mingguan, ataupun harian, maka dapat menggunakan metode *.add_seasonality()* di akhir baris fungsi *Prophet()*. Untuk menggunakannya dapat dengan mengetikkan kode berikut:

```
model = Prophet().add_seasonality(name='daily', period=1,
    fourier_order=5, mode='additive').add_seasonality(
    name='weekly', period=7, fourier_order=5,
    mode='multiplicative')
```

Keterangan:

- Argumen "*name*" dapat diisi dengan nama dari efek musiman yang ditambahkan.
- Argumen *"period"* dan *"fourier_order"* secara berturut-turut adalah nilai *P* dan *N* pada fungsi musiman.
- Argumen *"mode"* dapat diisi dengan *'additive'* atau *'multiplicative'.*

3.3.5. Pengaturan *Hyperparameter* Model *Prophet*

Pada tahapan ini dilakukan pengaturan pada *hyperparameter* model *Prophet*. Pengaturan ini dilakukan dengan tujuan untuk menghindari terjadinya *underfitting* dan *overfitting*. Pada tahap sebelumnya telah dijelaskan mengenai banyak titik ubah untuk fungsi tren, dan nilai N untuk fungsi tahunan. Pada tahapan ini, untuk iterasi pertama dimaksudkan untuk pemilihan bentuk model *Prophet* yang cocok dengan data, maka banyak titik ubah dan nilai N disamakan untuk setiap model, yaitu 25 dan 5 secara berturut-turut. Setelah dipilih satu model yang paling cocok dengan data, maka banyak titik ubah dan nilai N diubah dan dievaluasi secara berulang untuk menemukan nilai *hyperparameter* yang paling baik untuk model tersebut.

3.3.6. *Fitting*

Tahapan ini merupakan tahapan untuk membangun model *Prophet* yang telah diinisiasi dan diatur *hyperparameter*-nya. Model *Prophet* melakukan tahapan *fitting* dengan secara otomatis memilih titik ubah yang ada di data dan mengestimasi beberapa parameter yang terdapat pada fungsinya, yaitu $k, m, \boldsymbol{\delta}, \boldsymbol{\beta}$, dan σ_{obs}. Estimasi parameter-parameter tersebut dilakukan dengan menggunakan metode MAP (*Maximum a Posteriori*) yang disediakan oleh *library* stan. Untuk melakukan *fitting* dapat dengan menggunakan kode berikut:

```
model.fit(data_latih)
```

3.3.7. Evaluasi Model

Pada tahapan ini model yang telah dibangun akan dievaluasi dengan menggunakan dua buah ukuran, yaitu MAE (*Mean Absolute Error*) dan MAPE (*Mean Absolute Percentage Error*). Setiap model akan dievaluasi pada data latih

dan data uji, sehingga setiap model akan memiliki 4 buah nilai. Untuk menghitung nilai MAE dan MAPE dapat digunakan dua fungsi berikut:

```
def MAE(y, y_hat):
    total=0
    for i in range(len(y)):
        total += abs(y[i]-y_hat[i])
    return total/len(y)

def MAPE(y, y_hat):
    total=0
    for i in range(len(y)):
        total += abs((y[i]-y_hat[i])/y[i])
    return total/len(y)
```

Untuk memperoleh nilai prediksi dari model untuk evaluasi dengan data uji dapat digunakan kode berikut:

```
future = model.make_future_dataframe(periods=len(data_uji),
                                     freq='5T',
                                     include_history=True)
forecast = model.predict(future)
```

sehingga, untuk memperoleh nilai MAE dan MAPE dari model yang telah dibangun dapat digunakan kode berikut:

```
forecast_uji = forecast[i:]
forecast_uji.reset_index(inplace=True)

mae_latih = MAE(data_latih['y'], forecast['yhat'][:i])
mape_latih = MAPE(data_latih['y'], forecast['yhat'][:i])
mae_uji = MAE(data_uji['y'], forecast_uji['yhat'])
mape_uji = MAPE(data_uji['y'], forecast_uji['yhat'])

evaluasi = pd.DataFrame({' ' : ['MAE', 'MAPE'],
                    'Latih' : [mae_latih, mape_latih],
                    'Uji' : [mae_uji, mape_uji]})
print(evaluasi)
```

3.3.8. Identifikasi Gejala *Underfitting* dan *Overfitting*

Pada tahapan ini model yang paling cocok dengan data diidentifikasi apakah mengalami gejala *underfitting*, *overfitting*, atau tidak sama sekali. Bila model masih mengalami gejala *underfitting* dan *overfitting*, maka dilakukan pengaturan kembali *hyperparameter* pada model *Prophet*. Gejala *underfitting* dan *overfitting* dapat diidentifikasi dengan cara membandingkan performa model pada data latih dengan performa model dengan data uji. Berikut adalah ciri-ciri model yang mengalami *underfitting* ataupun *overfitting* (Brownlee, 2016):

- Gejala *underfitting* ditandai dengan performa model pada data latih dan performa model pada data uji yang buruk.
- Gejala *overfitting* ditandai dengan performa model pada data latih yang baik, namun performa model pada data uji yang buruk dan selisih dari keduanya cukup besar.

3.3.9. Peramalan dan Dekomposisi

Peramalan dan dekomposisi adalah tahapan terakhir sebelum dilakukan analisis dan interpretasi hasil peramalan dan dekomposisi. Pada tahapan ini dilakukan peramalan dan dekomposisi dengan menggunakan bentuk model yang paling dengan data dan merupakan model yang *best fit*. Peramalan yang akan dilakukan adalah peramalan dengan jangka waktu pendek, karena model *Prophet* memiliki kualitas peramalan yang baik bila jangka waktu peramalannya tidak terlalu panjang (Xie dkk, 2021). Oleh karena itu, peramalan yang dilakukan adalah untuk 1 jam ke depan atau 12 titik. Untuk melakukan peramalan dan membuat grafiknya, dapat menggunakan kode berikut:

```
future = model.make_future_dataframe(periods=len(data_uji)+12,
                                      freq='5T',
                                      include_history=True)
forecast = model.predict(future)
fig = model.plot(forecast)
```

Untuk melakukan dekomposisi, kode tersebut dapat dilanjutkan dengan kode berikut:

```
fig = model.plot_components(forecast)
```