Lampiran

```
import os
import pandas as pd
import numpy as np
import math
import datetime as dt

from sklearn.metrics import mean_squared_error,
mean_absolute_error, explained_variance_score, r2_score
from sklearn.metrics import mean_poisson_deviance,
mean_gamma_deviance, accuracy_score
from sklearn.preprocessing import MinMaxScaler

from itertools import cycle
import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots

import seaborn as sns
import matplotlib.pyplot as plt
from colorama import Fore

df = pd.read_csv('/content/dataset.csv')

btc = df.loc[df['crypto_name'] == 'Bitcoin'].copy()
btc

btc = btc.fillna(method = 'ffill')

btc['date'] = pd.to_datetime(btc.date)
btc.head().style.set_properties(subset=['date','close'],**{'ba
ckground-color': 'skyblue'})

fig = plt.figure(figsize = (15,10))
plt.subplot(2, 2, 1)
plt.plot(btc['date'], btc['close'], color="red")
plt.title('Bitcoin Close Price')

last2year_btc = btc[btc['date'] > '01-2020']

fig = plt.figure(figsize = (15,10))
fig.suptitle("Last 1 year close prices of Bitcoin",
fontsize=16)
plt.subplot(4, 1, 1)
plt.plot(last2year_btc['date'], last2year_btc['close'],
color="red")
plt.legend("B")

last2year_btc = btc[btc['date'] > '01-2020']
fig = plt.figure(figsize = (15,15))
```

```
fig.suptitle("Last month comparision of close and open prices
of Bitcoin", fontsize=16)
fig.tight_layout()
plt.subplot(4, 1, 1)
plt.plot(last2year_btc['date'], last2year_btc['close'])
plt.plot(last2year_btc['date'], last2year_btc['open'])
plt.legend(["Close", "Open"])
plt.title("Bitcoin")

plt.show()

fig = plt.figure(figsize = (15,10))

plt.subplot(2, 2, 1)
plt.plot(btc['date'], btc['close'].rolling(50).mean())
plt.plot(btc['date'], btc['close'].rolling(200).mean())
plt.title('Bitcoin Close Price moving average')

fig = plt.figure(figsize = (15,12))

fig.tight_layout()

plt.subplot(4, 1, 1)
sns.histplot(btc['close'],color='darkred', kde=True)
plt.axvline(btc['close'].mean(), color='k',
linestyle='dashed', linewidth=2)
plt.text(50000,400,'Bitcoin Close Price', fontsize=16)

closedf = closedf[closedf['date'] > '2020-01-01']
close_stock = closedf.copy()
print("Total data for prediction: ",closedf.shape[0])

del closedf['date']
scaler=MinMaxScaler(feature_range=(0,1))
closedf=scaler.fit_transform(np.array(closedf).reshape(-1,1))
print(closedf.shape)

training_size=int(len(closedf)*0.70)
test_size=len(closedf)-training_size
train_data,test_data=closedf[0:training_size,:],closedf[traini
ng_size:len(closedf),:1]
print("train_data: ", train_data.shape)
print("test_data: ", test_data.shape)

#Train & test data view
fig, ax = plt.subplots(figsize=(15, 6))
sns.lineplot(x = close_stock['date'][:571], y =
close_stock['close'][:571], color = 'black')
sns.lineplot(x = close_stock['date'][571:], y =
close_stock['close'][571:], color = 'red')
```

```
# Formatting
ax.set_title('Train & Test data', fontsize = 20, loc='center',
fontdict=dict(weight='bold'))
ax.set_xlabel('Date', fontsize = 16,
fontdict=dict(weight='bold'))
ax.set_ylabel('Weekly Sales', fontsize = 16,
fontdict=dict(weight='bold'))
plt.tick_params(axis='y', which='major', labelsize=16)
plt.tick_params(axis='x', which='major', labelsize=16)
plt.legend(loc='upper right' ,labels = ('train', 'test'))

def create_dataset(dataset, time_step=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-time_step-1):
        a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----
99   100
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
    return np.array(dataX), np.array(dataY)

time_step = 15
X_train, y_train = create_dataset(train_data, time_step)
X_test, y_test = create_dataset(test_data, time_step)

print("X_train: ", X_train.shape)
print("y_train: ", y_train.shape)
print("X_test: ", X_test.shape)
print("y_test", y_test.shape)

from xgboost import XGBRegressor
my_model = XGBRegressor(n_estimators=100, gamma=0.01)
my_model.fit(X_train, y_train, verbose=False)

predictions = my_model.predict(X_test)
print("Mean Absolute Error - MAE : " +
str(mean_absolute_error(y_test, predictions)))
print("Root Mean squared Error - RMSE : " +
str(math.sqrt(mean_squared_error(y_test, predictions))))

train_predict=my_model.predict(X_train)
test_predict=my_model.predict(X_test)

train_predict = train_predict.reshape(-1,1)
test_predict = test_predict.reshape(-1,1)

print("Train data prediction:", train_predict.shape)
print("Test data prediction:", test_predict.shape)

train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict)
original_ytrain = scaler.inverse_transform(y_train.reshape(-
```

```
1,1))
original_ytest = scaler.inverse_transform(y_test.reshape(-
1,1))

# shift train predictions for plotting
look_back=time_step
trainPredictPlot = np.empty_like(closedf)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(train_predict)+look_back, :] =
train_predict
print("Train predicted data: ", trainPredictPlot.shape)

# shift test predictions for plotting
testPredictPlot = np.empty_like(closedf)
testPredictPlot[:, :] = np.nan
testPredictPlot[len(train_predict)+(look_back*2)+1:len(closedf
)-1, :] = test_predict
print("Test predicted data: ", testPredictPlot.shape)

names = cycle(['Original close price','Train predicted close
price','Test predicted close price'])

plotdf = pd.DataFrame({'date': close_stock['date'],
                       'original_close': close_stock['close'],
                      'train_predicted_close':
trainPredictPlot.reshape(1,-1)[0].tolist(),
                      'test_predicted_close':
testPredictPlot.reshape(1,-1)[0].tolist()})

fig = px.line(plotdf,x=plotdf['date'],
y=[plotdf['original_close'],plotdf['train_predicted_close'],
                                          plotdf['test_predict
ed_close']],
              labels={'value':'Close price','date': 'Date'})
fig.update_layout(title_text='Comparision between original
close price vs predicted close price',
                  plot_bgcolor='white', font_size=15,
font_color='black',legend_title_text='Close Price')
fig.for_each_trace(lambda t:  t.update(name = next(names)))

fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()
```