**Pertemuan 11 Stacked-Bidirectional on Neural Network**

**Tujuan pembelajaran**

* Mahasiswa mampu memahami konsep time series.
* Mahasiswa mampu memahami konsep Neural Network menggunakan metode Long Short-Term Memory (LSTM) dan Gated Recurrent Unit (GRU).
* Mahasiswa mampu memahami konsep Stacked-Bidirectional pada Neural Network.

**Studi kasus: Prediksi harga BTC-USD menggunakan metode SB-GRU-RNN.**

1. **Deklarasi Pustaka**

|  |
| --- |
| # pustaka untuk manipulasi data-frame  import pandas as pd  from pandas import concat  from pandas import read\_csv  from pandas import read\_excel    # pustaka untuk madnipulasi data-array  import numpy as np  from numpy import concatenate  from numpy import array    # pustaka untuk waktu komputasi  import time  from datetime import datetime    # pustaka untuk visualisasi data  import seaborn as sns  from matplotlib import pyplot as plt    # pustaka untuk visualisasi acf dan pacf  import scipy.stats as sc  import statsmodels.api as sm  from statsmodels.graphics.tsaplots import plot\_pacf  from statsmodels.graphics.tsaplots import plot\_acf    # pustaka untuk membuat data latih dan data uji.  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import MinMaxScaler    # pustaka untuk membuat model prediksi LSTM-RNN  import tensorflow as tf  from keras.utils import Sequence  from keras.models import Sequential  from keras.layers import SimpleRNN  from keras.layers import LSTM  from keras.layers import GRU  from keras.layers import Dense  from keras.layers import Dropout  from keras.layers import TimeDistributed  from keras.layers import Bidirectional  from keras.optimizers import Adam, Adamax, RMSprop, SGD    # pustaka untuk  evaluasi model prediksi  from math import sqrt  from sklearn.metrics import mean\_squared\_error  from sklearn.metrics import mean\_absolute\_error  from sklearn.metrics import mean\_absolute\_percentage\_error |

1. **Deklarasi Pustaka**

|  |
| --- |
| # set waktu komputasi  start = time.time()    # set random seed  tf.random.set\_seed(1234)    # membaca dataset via csv file  dataset = read\_csv("dataset/BTC-USD.csv", parse\_dates=['Date'])    # set index tanggal  dataset = dataset.set\_index("Date")    # menampilkan metadata dataset  dataset.info() |
| <class 'pandas.core.frame.DataFrame'>  DatetimeIndex: 2896 entries, 2014-12-09 to 2022-11-12  Data columns (total 6 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 Open 2896 non-null float64  1 High 2896 non-null float64  2 Low 2896 non-null float64  3 Close 2896 non-null float64  4 Adj Close 2896 non-null float64  5 Volume 2896 non-null int64  dtypes: float64(5), int64(1)  memory usage: 158.4 KB |

|  |
| --- |
| # menampilkan dataset BTC-USD  print(np.round(dataset[["Open", "High", "Low", "Close", "Adj Close"]], 2)) |
| Open High Low Close Adj Close  Date  2014-12-09 361.89 363.07 344.95 352.22 352.22  2014-12-10 352.20 352.38 346.36 346.36 346.36  2014-12-11 344.34 361.36 338.76 350.51 350.51  2014-12-12 350.83 352.98 349.29 352.54 352.54  2014-12-13 352.38 352.38 346.59 347.38 347.38  ... ... ... ... ... ...  2022-11-08 20600.67 20664.61 17603.54 18541.27 18541.27  2022-11-09 18543.76 18590.46 15682.69 15880.78 15880.78  2022-11-10 15883.16 18054.31 15834.02 17586.77 17586.77  2022-11-11 17583.25 17650.94 16543.48 17034.29 17034.29  2022-11-12 17066.68 17066.68 16659.35 16752.43 16752.43  [2896 rows x 5 columns] |

1. **Eksplorasi Data Analisis (EDA)**

|  |
| --- |
| # membuat frame  fig,((ax1,ax2),(ax3,ax4)) = plt.subplots(  nrows=2, ncols=2, facecolor="#f0f0f0", figsize=(20, 10)  )    # membuat time series plot  ax1.plot(dataset.index.values, dataset["Open"], color="blue", label="Open Price", linewidth=2)    # membuat label-label  ax1.set\_title("Open Price Bitcoin", fontsize=14)  ax1.set\_xlabel("Date", fontsize=12)  ax1.set\_ylabel("Stock Price", fontsize=12)  ax1.legend(loc='upper left')  ax1.grid(True)  # --------------------------------------------------------------------------------    # membuat time series plot  ax2.plot(dataset.index.values,dataset["Close"], color="green", label="Close Price", linewidth=2)    # membuat label-label  ax2.set\_title("Close Price Bitcoin", fontsize=14)  ax2.set\_xlabel("Date", fontsize=12)  ax2.set\_ylabel("Stock Price", fontsize=12)  ax2.legend(loc='upper left')  ax2.grid(True)  # --------------------------------------------------------------------------------    # membuat time series plot  ax3.plot(dataset.index.values, dataset["High"], color="orange", label="High Price", linewidth=2)    # membuat label-label  ax3.set\_title("High Price Bitcoin", fontsize=14)  ax3.set\_xlabel("Date", fontsize=12)  ax3.set\_ylabel("Stock Price", fontsize=12)  ax3.legend(loc='upper left')  ax3.grid(True)  # --------------------------------------------------------------------------------    # membuat time series plot  ax4.plot(dataset.index.values, dataset["Low"], color="tab:red", label="Low Price", linewidth=2)    # membuat label-label  ax4.set\_title("Low Price Bitcoin", fontsize=14)  ax4.set\_xlabel("Date", fontsize=12)  ax4.set\_ylabel("Stock Price", fontsize=12)  ax4.legend(loc='upper left')  ax4.grid(True)  # --------------------------------------------------------------------------------    # set the spacing between subplots  plt.subplots\_adjust(wspace=0.15, hspace=0.25)    # menampilkan plot  plt.show() |

|  |
| --- |
|  |
| Gambar x. Output program |

1. **Praproses Data**

* **Seleksi Fitur**

|  |
| --- |
| # memilih fitur close price  dataset = dataset.filter(['Close']);    # convert dataframe to series close price  data = dataset.values  np.round(data[:5],5) |
| array([[352.21899],  [346.36499],  [350.50601],  [352.54199],  [347.37601]]) |

|  |
| --- |
| # melihat dimensi data  data.shape |
| (2896, 1) |

* **Normalisasi Data**

|  |
| --- |
| # normalize features  scaler = MinMaxScaler(feature\_range=(0, 1))  scaled\_data = scaler.fit\_transform(np.array(data).reshape(-1,1))  np.round(scaled\_data[:5],5) |
| array([[0.0025838],  [0.0024969],  [0.0025583],  [0.0025885],  [0.0025119]]) |

|  |
| --- |
| # melihat dimensi data  scaled\_data.shape |
| (2896, 1) |

|  |
| --- |
| # membuat frame  fig, ax = plt.subplots(figsize = (10,5))    # membuat time series plot  ax.plot(dataset.index.values, scaled\_data, color="blue", label="CLose Price", linewidth=2)    # membuat label-label  ax.set\_title("Close Price Bitcoin", fontsize=12)  ax.set\_xlabel("Date", fontsize=10)  ax.set\_ylabel("Stock Price", fontsize=10)  ax.legend(loc='upper left')  ax.grid(True)    # menampilkan plot  plt.show() |
|  |
| Gambar x. Output program |

* **Data latih dan Data uji**

|  |
| --- |
| # split data train and test  train\_data, test\_data = train\_test\_split(scaled\_data, train\_size=0.80, test\_size=0.20, shuffle=False)    # data train  np.round(train\_data[:5],5) |
| array([[0.0025838],  [0.0024969],  [0.0025583],  [0.0025885],  [0.0025119]]) |

|  |
| --- |
| # melihat dimensi data  train\_data.shape |
| (2316, 1) |

|  |
| --- |
| # data test  np.round(test\_data[:5],5) |
| array([[0.88613],  [0.9397 ],  [0.93386],  [0.93689],  [0.91105]]) |

|  |
| --- |
| # melihat dimensi data  test\_data.shape |
| (580, 1) |

* **Supervised Learning**

|  |
| --- |
| # function for supervised learning  def create\_dataset(look\_back, dataset):        # declare variable X and Y      dataX = []      dataY = []        # for loop for create supervised learning      for i in range(look\_back, len(dataset)):            # insert value X and Y          dataX.append(dataset[i-look\_back:i, 0])          dataY.append(dataset[i, 0])        # return value X and Y      return np.array(dataX), np.array(dataY) |

|  |
| --- |
| # process supervised learning  look\_back = 60  x\_train, y\_train = create\_dataset(look\_back, train\_data)  x\_test, y\_test = create\_dataset(look\_back, test\_data) |

|  |
| --- |
| # melihat dimensi data  print(x\_train.shape, y\_train.shape) |
| (2256, 60) (2256,) |

|  |
| --- |
| # melihat dimensi data  print(x\_test.shape, y\_test.shape) |
| (520, 60) (520,) |

|  |
| --- |
| # reshape input to be [samples, time steps, features]  x\_train = np.reshape(x\_train, (x\_train.shape[0], x\_train.shape[1], 1))  x\_test = np.reshape(x\_test, (x\_test.shape[0], x\_test.shape[1], 1))  print(x\_train.shape, x\_test.shape) |
| (2256, 60, 1) (520, 60, 1) |

1. **Model prediksi menggunakan metode GRU-RNN**

|  |
| --- |
| # The GRU-RNN architecture  model = tf.keras.Sequential([        # First GRU layer with Dropout regularisation      tf.keras.layers.Bidirectional(          GRU(units=50, return\_sequences=True, input\_shape=(x\_train.shape[1], 1))      ),        # Secound GRU layer with Dropout regularisation      tf.keras.layers.Bidirectional(          GRU(units=50, return\_sequences=False)      ),        # The output layer      tf.keras.layers.Dense(1)  ])    # Compile the model GRU  model.compile(optimizer='adam', loss='mean\_squared\_error')    # fit network  history = model.fit(  x\_train, y\_train, batch\_size=8, epochs=50, verbose="auto",  validation\_data=(x\_test,y\_test), shuffle=False, use\_multiprocessing=True  )  # show summary of model  model.summary() |
| Model: "sequential"  **=================================================================**   Layer (type)                       Output Shape       Param #  **=================================================================**   bidirectional\_0 (Bidirectional)    (8, 60, 100)       20800   bidirectional\_1 (Bidirectional)    (8, 100)           60400   dense (Dense)                      (8, 1)             101  **=================================================================**  Total params: 81301 (317.58 KB)  Trainable params: 81301 (317.58 KB)  Non-trainable params: 0 (0.00 Byte)  **=================================================================** |

|  |
| --- |
| # Proses prediksi harga BTC-USD  predictions = model.predict(x\_test)  print(predictions[:7]) |
| [[0.564491]  [0.574837]  [0.546902]  [0.598751]  [0.612883]] |

|  |
| --- |
| # membuat frame  fig, ax = plt.subplots(figsize = (10,5))    # membuat time series plot  ax.plot(history.history['loss'], color="tab:blue", label="loss func", linewidth=2)    # membuat label-label  ax.set\_xlabel("epoch", fontsize=10)  ax.set\_ylabel("loss function", fontsize=10)  ax.legend(loc='upper right')  ax.grid(True)    # menampilkan plot  plt.show() |
|  |
| Gambar x. Output program |

1. **Evaluasi Model GRU-RNN**

|  |
| --- |
| # evaluasi training dataset  scores = model.evaluate(x\_train, y\_train)  scores |
| 71/71 [==============================] - 9s 28ms/step - loss: 0.0012  0.0012363543501123786 |

|  |
| --- |
| # evaluasi testing dataset  scores = model.evaluate(x\_test, y\_test)  scores |
| 17/17 [==============================] - 1s 29ms/step - loss: 5.8673e-04  0.0005867323488928378 |

|  |
| --- |
| # evaluasi menggunakan mae, rmse, dan mape  mae = mean\_absolute\_error(y\_test, predictions)  print('Test MAE : %.4f' % mae)  rmse = sqrt(mean\_squared\_error(y\_test, predictions))  print('Test RMSE: %.4f' % rmse)    mape = mean\_absolute\_percentage\_error(y\_test, predictions)\*100  print('Test MAPE: %.4f' % mape) |
| Test MAE **:** 0.0168  Test RMSE **:** 0.0239  Test MAPE **:** 3.1457 |

|  |
| --- |
| hasil = np.stack((y\_test.reshape(-1), predictions.reshape(-1)), axis=1)  hasil = pd.DataFrame(hasil, columns = ['data\_aktual','prediksi'])  hasil.head() |
| data\_aktual prediksi  0 0.551373 0.563367  1 0.524931 0.571965  2 0.577541 0.543638  3 0.594170 0.596146  4 0.596957 0.612144 |

|  |
| --- |
| r, p = sc.pearsonr(hasil["data\_aktual"], hasil["prediksi"])  print(  "korelasi data akual dengan hasil prediksi" +" {:.4f} ".format(r)+  "dengan signifikansi" +" {:.4f} ".format(p)  ) |
| korelasi data akual dengan hasil prediksi 0.9932 dengan signifikansi 0.0000 |

|  |
| --- |
| # Set akhir waktu komputasi  end = time.time()    # Proses menghitung waktu komputasi  hours, rem = divmod(end-start, 3600)  minutes, seconds = divmod(rem, 60)    # Hasil waktu komputasi  print("{:0>2}:{:0>2}:{:05.2f}".format(int(hours),int(minutes),seconds)) |
| 00:09:18.24 |

1. **Visualisasi Data**

|  |
| --- |
| # inverse value test predictions  testPredictions = scaler.inverse\_transform(predictions)    # shift test predictions for plotting  testPredictionsPlot = np.empty\_like(scaled\_data)  testPredictionsPlot[:, :] = np.nan  testPredictionsPlot[  (len(dataset) - testPredictions.shape[0]):len(dataset),:] = testPredictions    # membuat frame  fig, ax = plt.subplots(figsize = (15,5))    # membuat time series plot  ax.plot(    dataset.index.values, scaler.inverse\_transform(scaled\_data),    color="tab:blue", label="data aktual", linewidth=2  )  ax.plot(    dataset.index.values, testPredictionsPlot, color="tab:red",    label="data prediksi", linewidth=2  )    # membuat label-label  ax.set\_title("Model Stacked-Bidirectional GRU-RNN", fontsize=14)  ax.set\_xlabel("Years", fontsize=10)  ax.set\_ylabel("Price BTC-USD", fontsize=10)  ax.legend(loc="best")  ax.grid(True)    # menampilkan plot  plt.show() |

|  |
| --- |
|  |
| Gambar x. Output program |

**Selesai, Selamat Mencoba :3**