**Mudah Belajar Otodidak Data Science**

**(Praktek Menggunakan Python3)**

**Edisi 2 Tahun 2023**

**Disusun Oleh:**

**Aryajaya Alasmsyah, S.Kom., M.Kom., MTA.**

**Materi Praktikum Data Science**

1. Pengantar Bahasa Python
2. Visualisasi Data Menggunakan Matplotlib dan Seaborn
3. Visualisasi Data Menggunakan Framework Streamlit
4. Teknik-Teknik Praproses Data – Data Tabular, Time Series, dan Spasial
5. Algoritma Klasifikasi Data Bagian 1
6. Algoritma Klastering Data Bagian 2
7. Ujian Tengah Semester (UTS)
8. Algoritma Klastering Data Bagian 1
9. Algoritma Klastering Data Bagian 2
10. Algoritma Regresi Linier
11. Algoritma Neural Network: SBi-LSTM dan SBi-GRU
12. Algoritma Neural Network: SBi-LSTM-XGBoost dan SBi-GRU-XGBoostost
13. Presentasi Projek
14. Ujian Akhir Semester (UAS)

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

|  |
| --- |
| **Tujuan pembelajaran**   * Mahasiswa mampu memahami konsep timeseries. * Mahasiswa mampu memahami konsep Stacked-Bidirectional pada Neural Network. * Mahasiswa mampu menerapkan atau membuat model prediksi timeseries menggunakan metode Stacked-Bidirectional pada Neural Network. |

**Studi kasus: Prediksi bitcoin menggunakan metode SBi-LSTM-RNN dan SBi-GRU-RNN.**

**C01\_visualization.py**

|  |
| --- |
| # Declaration library  import matplotlib.pyplot as plt  from matplotlib.dates import DateFormatter  # ----------------------------------------------------------------------------------------    # function of lineplot  def timeseries\_matplotlib(df, nm\_labels):      # create lineplot    fig, ax = plt.subplots(figsize = (8,4))    for x in range(len(nm\_labels)):      ax.plot(df.iloc[:, 0:1], df.iloc[:, x+1:x+2], label=nm\_labels[x], linewidth=2.5)      # set label-labels    ax.set\_title("", fontsize=12)    ax.set\_xlabel("", fontsize=10)    ax.set\_ylabel("", fontsize=10)    ax.legend(loc="best")    ax.grid(True)      # show lineplot    plt.show()  # ----------------------------------------------------------------------------------------    # func timeseries plot  def lineplot\_matplotlib1(x1, y1, label1, title):      # create lineplot    fig, ax = plt.subplots(figsize = (8,4))    ax.plot(x1, y1, color="tab:blue", label=label1, linewidth=2.5)      # set label-labels    ax.set\_title(title, fontsize=12)    ax.set\_xlabel("", fontsize=10)    ax.set\_ylabel("", fontsize=10)    ax.legend(loc="best")    ax.grid(True)      # show lineplot    plt.show()  # ----------------------------------------------------------------------------------------    # func timeseries plot  def lineplot\_matplotlib2(x1, y1, label1, x2, y2, label2, title):      # create lineplot    fig, ax = plt.subplots(figsize = (8,4))    ax.plot(x1, y1, color="tab:blue", label=label1, linewidth=2.5)    ax.plot(x2, y2, color="tab:red", label=label2, linewidth=2.5)      # set label-labels    ax.set\_title(title, fontsize=12)    ax.set\_xlabel("", fontsize=10)    ax.set\_ylabel("", fontsize=10)    ax.legend(loc="best")    ax.grid(True)      # show lineplot    plt.show()  # ----------------------------------------------------------------------------------------    # func timeseries plot  def lineplot\_matplotlib3(x1, y1, label1, x2, y2, label2, title):      # create lineplot    fig, ax = plt.subplots(figsize = (8,4))    ax.plot(x1, y1, color="tab:blue", label=label1, linewidth=2.5)    ax.plot(x2, y2, color="tab:red", label=label2, linewidth=2.5)      # set label-labels    ax.xaxis.set\_major\_formatter(DateFormatter("%Y"))    ax.set\_title(title, fontsize=12)    ax.set\_xlabel("", fontsize=10)    ax.set\_ylabel("", fontsize=10)    ax.legend(loc="best")    ax.grid(True)      # show lineplot    plt.show()  # ---------------------------------------------------------------------------------------- |

**C02\_model\_predictions.py**

|  |
| --- |
| # lib neural network algorithms  import tensorflow as tf  from keras.models import Sequential  from keras.layers import Bidirectional  from keras.layers import LSTM  from keras.layers import GRU  from keras.layers import Dropout  from keras.layers import Dense    # func model predictions  def get\_models(algorithms, timestep):      # 1. SBi-LSTM-RNN architecture    if algorithms == "SBi-LSTM-RNN":      tf.keras.backend.clear\_session()      model = Sequential()      model.add(Bidirectional(LSTM(units=50, return\_sequences=True, input\_shape=(timestep, 1))))      model.add(Bidirectional(LSTM(units=50, return\_sequences=False)))      model.add(Dropout(0.05))      model.add(Dense(1))      # 1. SBi-GRU-RNN architecture    if algorithms == "SBi-GRU-RNN":      tf.keras.backend.clear\_session()      model = Sequential()      model.add(Bidirectional(GRU(units=50, return\_sequences=True, input\_shape=(timestep, 1))))      model.add(Bidirectional(GRU(units=50, return\_sequences=False)))      model.add(Dropout(0.05))      model.add(Dense(1))      # 2. compile models    model.compile(optimizer="adamax", loss="mean\_squared\_error")      # return values    return model  # ----------------------------------------------------------------------------------------    # func model predictions  def get\_predictions(model, x\_train, y\_train, x\_test, y\_test):      # 3. fitting models    history = model.fit(      x\_train, y\_train,      batch\_size=16, epochs=50, verbose="auto",      validation\_data=(x\_test, y\_test),      use\_multiprocessing=False, shuffle=False    )      # 4. predict models    predictions = model.predict(x\_test, verbose=0)      # return values    return history, predictions  # ---------------------------------------------------------------------------------------- |

**C02\_model\_predictions.py**