Laporan Perbandingan RNN dan LSTM dalam Prediksi Nilai Tukar HKD

Nama : Alif As'ad Ramadhan

NRP : 5054231007

1. Pendahuluan

Dalam Tugas ini, kita membandingkan kinerja model Recurrent Neural Network (RNN) dan Long Short-Term Memory (LSTM) dalam memprediksi nilai tukar HKD. Dengan porsi training dari januari 2001 hingga Desember 2022 dan porsi testing dari januari 2023 hingga Desember 2023. Evaluasi dilakukan dengan Mean Squared Error (MSE) dan visualisasi hasil prediksi.

2. Source Code

2.1 Import Library dan Load Dataset

```
[38] import pandas as pd
     import numpy as np
     import tensorflow as tf
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import SimpleRNN, LSTM, Dense
     from sklearn.preprocessing import MinMaxScaler
     import matplotlib.pyplot as plt

[39] # Enable eager execution
     tf.config.run_functions_eagerly(True)

     # Load dataset
     df = pd.read_excel("Monetary Data.xlsx", sheet_name="Nilai Tukar")
     df.head()
```

2.2 Tampilan Dataset

	Tahun	USD	JPY	DEM	NLG	GBP	FRH	CHF	SGD	MYR	HKD	AUD	CAD
0	2025	0	0	0	0	0	0	0	0	0	0	0	0
1	Januari	16,259.00	10,523.63	0	0	20,245.72	0	17,938.99	12,045.96	3,701.14	2,086.94	10,117.19	11,276.50
2	2024	0	0	0	0	0	0	0	0	0	0	0	0
3	Desember	16,162.00	10,236.25	0	0	20,332.61	0	17,920.95	11,919.34	3,616.48	2,082.02	10,081.88	11,225.18
4	November	15,864.00	10,453.01	0	0	20,067.98	0	17,944.72	11,805.79	3,566.56	2,038.59	10,283.06	11,310.03

2.3 Data Cleaning dan Preprocessing

```
[41] df = df[df["Tahun"] != 0]

     months = ["Januari", "Februari", "Maret", "April", "Mei", "Juni", "Juli", "Agustus",
               "September", "Oktober", "November", "Desember"]

     dates = []
     current_year = None
     for index, row in df.iterrows():
         if str(row["Tahun"]).isdigit():
             current_year = int(row["Tahun"])
         elif row["Tahun"] in months and current_year:
             month_num = months.index(row["Tahun"]) + 1
             dates.append(f"{current_year}-{month_num:02d}")
         else:
             dates.append(None)

[43] df = df[df["Tahun"].isin(months)].reset_index(drop=True)
     df["Date"] = dates
     df.drop(columns=["Tahun"], inplace=True)
     for col in df.columns[:-1]:
```

Penjelasan Data Cleaning dan Preprocessing

1. Menghapus Data yang Tidak Valid:
   - Baris dengan nilai “Tahun” nol dihapus karena dianggap sebagai noise dalam dataset.
2. Konversi “Tahun” dan “Bulan” menjadi format Date:
   - Tahun dikonversi menjadi format numerik.
   - Bulan dikonversi menjadi angka 1-12 dan digabung dengan tahun dalan format YYYY-MM.
3. Penghapusan Kolom “Tahun”:
   - Setelahd ata berhasil dikonversi ke format YYY-MM, kolom “Tahun” tidak diperlukan lagi.
4. Menghilangkan Karakter Tidak Diperlukan:
   - Menghapus koma pada angka untuk memastikan format numerik yang benar.

2.4 Pemisahan Fitur dan Target

```
X = df.drop(columns=["HKD"])
y = df["HKD"]
X["Date"] = pd.to_datetime(X["Date"], format="%Y-%m")
train_mask = (X["Date"] >= "2001-01-01") & (X["Date"] <= "2022-12-01")
test_mask = (X["Date"] >= "2023-01-01") & (X["Date"] <= "2023-12-01")

X_train, X_test = X[train_mask].drop(columns=["Date"]), X[test_mask].drop(columns=["Date"])
y_train, y_test = y[train_mask], y[test_mask]
```

```
[ ] X.head()
```

	Date	USD	JPY	DEM	NLG	GBP	FRH	CHF	SGD	MYR	AUD	CAD
0	2025-01-01	16259.0	10523.63	0.0	0.0	20245.72	0.0	17938.99	12045.96	3701.14	10117.19	11276.50
1	2024-12-01	16162.0	10236.25	0.0	0.0	20332.61	0.0	17920.95	11919.34	3616.48	10081.88	11225.18
2	2024-11-01	15864.0	10453.01	0.0	0.0	20067.98	0.0	17944.72	11805.79	3566.56	10283.06	11310.03
3	2024-10-01	15732.0	10259.89	0.0	0.0	20464.98	0.0	18136.98	11878.14	3587.29	10319.41	11302.94
4	2024-09-01	15138.0	10566.44	0.0	0.0	20237.24	0.0	17899.98	11788.35	3675.19	10417.22	11222.07

2.5 Normalisasi Data

```
scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()
X_train_scaled = scaler_X.fit_transform(X_train)
X_test_scaled = scaler_X.transform(X_test)
y_train_scaled = scaler_y.fit_transform(y_train.values.reshape(-1, 1))
y_test_scaled = scaler_y.transform(y_test.values.reshape(-1, 1))
```

2.6 Transformasi Data Untuk Model

```
[ ] X_train_reshaped = X_train_scaled.reshape(X_train_scaled.shape[0], 1, X_train_scaled.shape[1])
    X_test_reshaped = X_test_scaled.reshape(X_test_scaled.shape[0], 1, X_test_scaled.shape[1])
```

2.7 Konfigurasi Model RNN dan LSTM

```
units = 50
dropout_rate = 0.2
learning_rate = 0.001
```

RNN Model

```
optimizer_rnn = tf.keras.optimizers.Adam(learning_rate=learning_rate)
model_rnn = Sequential([
    SimpleRNN(units, activation="relu", return_sequences=False, input_shape=(1, 11)),
    Dense(1)
])
model_rnn.compile(optimizer=optimizer_rnn, loss="mse")
```

LSTM Model

```
optimizer_lstm = tf.keras.optimizers.Adam(learning_rate=learning_rate)
model_lstm = Sequential([
    LSTM(units, activation="relu", return_sequences=False, input_shape=(1, 11)),
    Dense(1)
])
model_lstm.compile(optimizer=optimizer_lstm, loss="mse")
```

2.8 Training Model

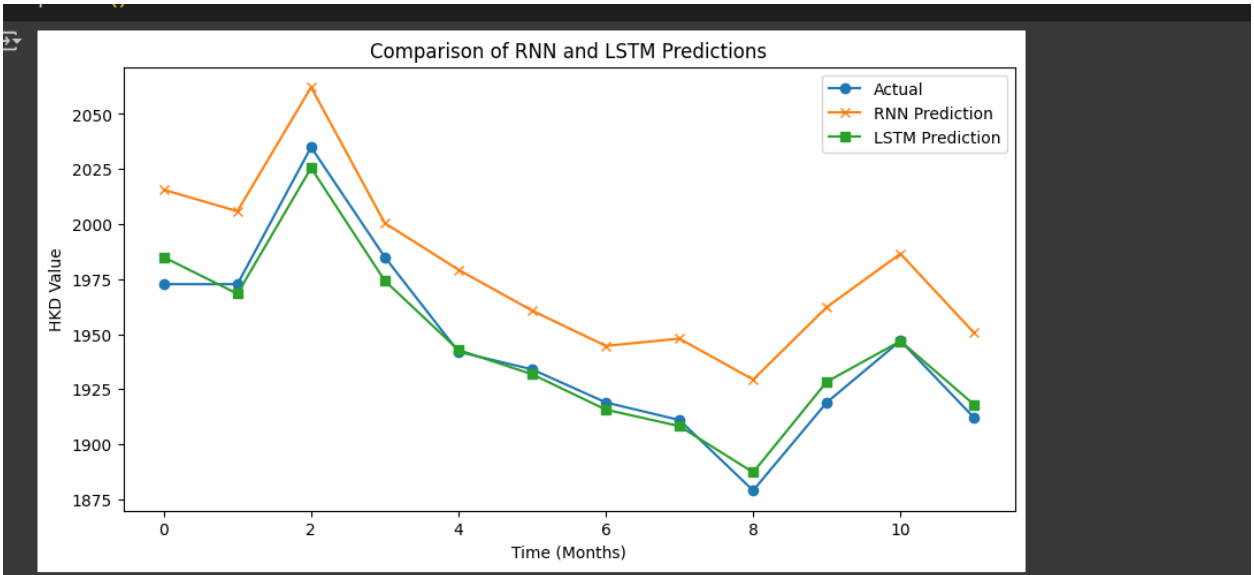
Train Models

```
# Train Models
history_rnn = model_rnn.fit(X_train_reshaped, y_train_scaled, epochs=50, batch_size=16, validation_data=(X_test_reshaped, y_test_scaled), verbose=1)
history_lstm = model_lstm.fit(X_train_reshaped, y_train_scaled, epochs=50, batch_size=16, validation_data=(X_test_reshaped, y_test_scaled), verbose=1)
```

```
Epoch 15/50
17/17 ━━━━━━━━━━━━━━━━━━━━ 1s 49ms/step - loss: 4.7307e-04 - val_loss: 1.7949e-04
Epoch 16/50
17/17 ━━━━━━━━━━━━━━━━━━━━ 1s 49ms/step - loss: 0.0056 - val_loss: 4.5055e-04
Epoch 17/50
17/17 ━━━━━━━━━━━━━━━━━━━━ 2s 76ms/step - loss: 0.0057 - val_loss: 2.5247e-04
Epoch 18/50
17/17 ━━━━━━━━━━━━━━━━━━━━ 2s 48ms/step - loss: 0.0033 - val_loss: 2.2497e-04
Epoch 19/50
17/17 ━━━━━━━━━━━━━━━━━━━━ 1s 49ms/step - loss: 8.5916e-04 - val_loss: 1.4755e-04
```

3. Hasil dan Analisis

Berikut adalah hasil prediksi dan MSE dari masing – masing Model:




```
print(f"MSE RNN: {mse_rnn}")
print(f"MSE LSTM: {mse_lstm}")
```

```
MSE RNN: 1292.1136416000875
MSE LSTM: 49.04725859336663
```

3.1 Hasil Evaluasi

Berdasarkan perhitungan MSE:

- MSE RNN: 1292.11

- MSE LSTM: 49.04

Dari hasul ini, LSTM memiliki nilai MSE yang lebih rendah dibandingkan RNN, menunjukan bahwa LSTM lebih akurat dalam memprediksi nilai tukar HKD.

3.2 Analisis Perbandingan

Berdasarkan grafik yang dihasilkan, prediksi LSTM lebih mendekati nilai aktual dibandingkan RNN. Hal ini menunjukkan bahwa LSTM lebih unggul dalam menangani pola data sekuensial yang kompleks.

4. Kesimpulan

Dari hasil eksperimen, kita menemukan bahwa LSTM memberikan hasil prediksi yang lebih baik dibandingkan RNN dalam kasus prediksi nilai tukar HKD ini. Hal ini menunjukkan bahwa pemanfaatan memori jangka panjang pada LSTM lebih efektif dalam menangkap tren dan pola data dibandingkan RNN. Namun, tuning parameter lebih lanjut dapat dilakukan untuk meningkatkan performa kedua model.